HITACHI
Inspire the Next

COBOL2002　操作ガイド

操作書

4010-1J-801-10

COBOL2002

前書き

■ 対象製品

P-2636-2354 COBOL2002 Net Developer 05-10（適用 OS：Windows 10(x64)，Windows 11，Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2436-5354 COBOL2002 Net Server Runtime 05-10（適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2636-3354 COBOL2002 Net Client Runtime 05-10（適用 OS：Windows 10(x64)，Windows 11）

P-2436-6354 COBOL2002 Net Server Suite 05-10（適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2636-4354 COBOL2002 Net Client Suite 05-10（適用 OS：Windows 10(x64)，Windows 11）

P-2936-2354 COBOL2002 Net Developer(64) 05-10（適用 OS：Windows 10(x64)，Windows 11，Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2936-5354 COBOL2002 Net Server Runtime(64) 05-10（適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2936-6354 COBOL2002 Net Server Suite(64) 05-10（適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2636-7354 COBOL2002 Developer Professional 05-10（適用 OS：Windows 10(x64)，Windows 11，Windows Server 2019，Windows Server 2022，Windows Server 2025）

P-2936-7354 COBOL2002 Developer Professional(64) 05-10（適用 OS：Windows 10(x64)，Windows 11，Windows Server 2019，Windows Server 2022，Windows Server 2025）

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

COBOL2002 は，COBOL2002 規格（ISO/IEC 1989:2002）の主な機能に対応しています。
COBOL85 互換機能（コンパイラオプション）指定時，COBOL2002 は，COBOL85 規格（ISO85，ANSI85，JIS88，JIS92）の上位水準に準拠します。

■ 輸出時の注意

本製品を輸出される場合には、外国為替および外国貿易法の規制ならびに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

記載の会社名，製品名などは，それぞれの会社の商標もしくは登録商標です。

■ 発行

2026 年 2 月　4010-1J-801-10

■ 著作権

All Rights Reserved. Copyright (C) 2025, 2026, Hitachi, Ltd.

変更内容

変更内容（4010-1J-801-10） COBOL2002 Net Developer 05-10，COBOL2002 Net Server Runtime 05-10，COBOL2002 Net Client Runtime 05-10，COBOL2002 Net Server Suite 05-10，COBOL2002 Net Client Suite 05-10，COBOL2002 Net Developer(64) 05-10，COBOL2002 Net Server Runtime(64) 05-10，COBOL2002 Net Server Suite(64) 05-10，COBOL2002 Developer Professional 05-10，COBOL2002 Developer Professional(64) 05-10

追加・変更内容	変更箇所
次の製品の適用 OS に「Windows Server 2025」を追加した。 • P-2636-2354 COBOL2002 Net Developer • P-2436-5354 COBOL2002 Net Server Runtime • P-2436-6354 COBOL2002 Net Server Suite • P-2936-2354 COBOL2002 Net Developer(64) • P-2936-5354 COBOL2002 Net Server Runtime(64) • P-2936-6354 COBOL2002 Net Server Suite(64) • P-2636-7354 COBOL2002 Developer Professional • P-2936-7354 COBOL2002 Developer Professional(64)	－
「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムで初期化漏れチェック機能を使用したとき，EXIT PROGRAM 文を GOBACK 文とみなすか CONTINUE 文とみなすかを切り替えることができるようにした。 これに伴い，環境変数 CBLUNINITDATA_EXITPROGRAM を追加した。	2.4.2，2.4.3

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，次に示すプログラムプロダクトの機能と操作方法について説明したものです。

- P-2636-2354 COBOL2002 Net Developer
- P-2436-5354 COBOL2002 Net Server Runtime
- P-2636-3354 COBOL2002 Net Client Runtime
- P-2436-6354 COBOL2002 Net Server Suite
- P-2636-4354 COBOL2002 Net Client Suite
- P-2936-2354 COBOL2002 Net Developer(64)
- P-2936-5354 COBOL2002 Net Server Runtime(64)
- P-2936-6354 COBOL2002 Net Server Suite(64)
- P-2636-7354 COBOL2002 Developer Professional
- P-2936-7354 COBOL2002 Developer Professional(64)

■ 対象読者

このマニュアルは，開発マネージャ，テストデバッガなどの開発ツールの使用方法を知りたい方を対象としています。また，COBOL の基本的な言語仕様と Windows の操作方法について理解していることを前提としています。

Windows(x64) COBOL2002 をお使いになる場合は，必要に応じて「第 4 編」の「18.　64bit アプリケーションの開発」をお読みください。

■ このマニュアルで使用する記号

このマニュアルで使用する記号を次に示します。

記号	意味
[　] キー	文字キーや F（ファンクション）キーを意味する。
[　] ＋ [　] キー	＋の前のキーを押したまま，あとのキーを押すことを意味する。
{　}	この記号で囲まれている複数の項目のうちから一つを選択することを意味する。項目が縦に複数行にわたって記述されている場合は，そのうちの 1 行分を選択する。
{　} ＋	この記号で囲まれている複数の項目のうちから一つを選択することを意味する。同じ要素の繰り返し指定はできないが，異なる要素であれば，複数指定できる。例えば，{A ｜ B ｜ C} ＋と表記されている場合は，A・B・C をすべて選択してもよい。項目が縦に複数行にわたって記述されている場合は，そのうちの 1 行分を選択する。

記号	意味
[ ]	この記号で囲まれている項目は，省略してもよいことを意味する。 複数の項目が縦または横に並べて記述されている場合には，すべてを省略するか，記号 { } と同じく，どれか一つを選択することを意味する。
…	記述が省略されていることを意味する。 この記号の直前に示された項目を繰り返して複数個指定できる。
下線	括弧で囲まれた複数の項目のうち 1 項目に対して使用され，括弧内のすべてを省略したときにシステムが仮定する標準値を意味する。
|	横に並べられた複数の項目に対して項目間の区切りを示し，「または」を意味する。
□	オペランドの名称の短縮形を示す。この記号で囲まれている文字だけを指定すれば短縮形となる。
[ ]	ウィンドウのメニューバーから選択するメニュー，コマンド，またはボタンを意味する。

■ キーの表記

このマニュアルでのジェネリックキー表記と，ご利用のキーボードのキー表記は異なる場合があります。キーの対応については，Windows のマニュアルを参照して確認してください。

■ ［COBOL2002］下のメニューと，ウィンドウのタイトルバーの表記

Windows(x64) COBOL2002 の場合，Windows のプログラム一覧にある［COBOL2002］下のメニューと，ウィンドウのタイトルバーには，末尾に「64bit」が付きます。このマニュアルでは，Windows(x86) COBOL2002 のメニューとタイトルバーの表記を使用します。Windows(x64) COBOL2002 をご使用になる場合は，表記を読み替えてください。

Windows(x86) COBOL2002 での表記	Windows(x64) COBOL2002 での表記
COBOL2002	COBOL2002 64bit
COBOL エディタ for COBOL2002	COBOL エディタ for COBOL2002 64bit
COBOL コマンドプロンプト for COBOL2002	COBOL コマンドプロンプト for COBOL2002 64bit
ODBC レコード定義生成 for COBOL2002	ODBC レコード定義生成 for COBOL2002 64bit
カバレージ for COBOL2002	カバレージ for COBOL2002 64bit
テストデバッガ for COBOL2002	テストデバッガ for COBOL2002 64bit
ファイル・レコード定義 for COBOL2002	ファイル・レコード定義 for COBOL2002 64bit
画面定義 for COBOL2002	画面定義 for COBOL2002 64bit
開発マネージャ for COBOL2002	開発マネージャ for COBOL2002 64bit
実行支援 for COBOL2002	実行支援 for COBOL2002 64bit

■ コンパイラオプション，および環境変数について

このマニュアルには，Windows(x64) COBOL2002 では使用できないコンパイラオプション，および環境変数についても記載しています。

使用できないコンパイラオプションについては「18.1.2　使用できないコンパイラオプション」を，使用できない環境変数については「18.1.3　使用できない環境変数」をそれぞれ参照してください。

■ プログラム例について

このマニュアルのプログラム例は，断り書きがない場合は Windows(x86) COBOL2002 用です。プログラム例を Windows(x64) COBOL2002 で使用するには，プログラムの記述に変更が必要な場合がありますのでご注意ください。

目次

前書き　2
変更内容　4
はじめに　5

第 1 編　翻訳・実行

1　COBOL2002 による開発環境　25
1.1　概要　26
1.1.1　開発資源のプロジェクトでの管理　26
1.1.2　プロジェクトマスタによるプロジェクトの管理　26
1.1.3　開発マネージャによる開発管理　27
1.1.4　開発マネージャで使用するファイル　27
1.1.5　開発マネージャで使用するファイル名，およびフォルダ名の規則　29
1.2　開発マネージャの起動と終了　30
1.2.1　開発マネージャの起動方法　30
1.2.2　開発マネージャの終了方法　30
1.3　開発マネージャのウィンドウ　31
1.3.1　ウィンドウの構成　31
1.3.2　メニューバー　31
1.3.3　ツールバー　36
1.3.4　ツリービューウィンドウ　38
1.3.5　メッセージウィンドウ　42
1.3.6　開発マネージャのウィンドウの注意事項　43
1.4　プロジェクトマスタの定義　44
1.4.1　プロジェクトマスタの新規作成　44
1.4.2　プロジェクトマスタを開く　45
1.4.3　プロジェクトマスタを閉じる　46
1.4.4　プロジェクトマスタの保存　47
1.4.5　プロジェクトマスタの定義内容の確認　47
1.4.6　プロジェクトマスタの複写　48
1.5　プロジェクトの定義　50
1.5.1　プロジェクトの作成　50
1.5.2　プロジェクト間の関係設定　56
1.5.3　作業中プロジェクトの設定　58
1.5.4　プロジェクトの定義内容の確認　59

1.5.5 プロジェクトの削除 63
1.5.6 プロジェクトの設定 64
1.5.7 プロジェクト情報の出力と取り込み 66
1.5.8 プロジェクト名の変更 68
1.6 ファイルの登録と管理 70
1.6.1 ファイルの新規作成 70
1.6.2 ファイルの登録と除外 70
1.6.3 ファイルを開く 77
1.6.4 ファイルの情報の確認 79
1.6.5 ファイルの設定 83
1.6.6 フォルダ指定によるファイルの追加 84
1.7 開発マネージャのカスタマイズ 86
1.7.1 動作オプションの設定 86
1.7.2 ユーザインタフェースの設定 87
1.7.3 テキストエディタの設定 91
1.7.4 ファイルの種類の登録 95
1.8 各種支援ツールでの環境設定 107
1.9 プロジェクトマスタ情報の CSV ファイル入出力 108
1.9.1 CSV ファイルの使用方法 109
1.9.2 CSV ファイル編集時の規則 110
1.9.3 プロジェクトマスタ情報の CSV ファイルの形式 112
1.9.4 CSV ファイルのセクションの形式 113
1.9.5 CSV ファイルに指定できるプロジェクトの種類 128
1.9.6 CSV ファイルに指定できるファイルの種別 128
1.9.7 CSV ファイル入出力時のエラーチェック 130
1.10 構成管理ツールとの連携（COBOL2002 Developer Professional でだけ有効） 133
1.10.1 構成管理ツールの仕様 134
1.10.2 構成管理ツールの操作 134
1.10.3 構成管理ツールの設定ダイアログボックス 137
1.11 外部ツールとの連携（COBOL2002 Developer Professional でだけ有効） 141
1.11.1 連携できる外部ツールの仕様 141
1.11.2 外部ツールの操作 141
1.11.3 外部ツールの設定ダイアログボックス 143

2 コンパイル 148

2.1 概要 149
2.1.1 ビルドとは 149
2.1.2 クイックビルドとは 149
2.1.3 リビルドとは 149

2.1.4 ファイル単位のコンパイルとは 150
2.2 コンパイラオプションの設定 151
2.2.1 デフォルトオプションの設定 151
2.2.2 プロジェクト単位，またはファイル単位のオプションの設定 151
2.2.3 オプションの設定方法 152
2.2.4 注意事項 154
2.3 コンパイラオプション 155
2.3.1 構文規則 155
2.3.2 一般規則 156
2.3.3 コンパイラオプションの優先順位 156
2.3.4 コンパイラオプションの一覧 163
2.3.5 最終生成物の種類（プロジェクトの種類）の設定 174
2.3.6 他製品との連携の設定 177
2.3.7 実行の設定 180
2.3.8 プログラムの最適化の設定 183
2.3.9 デバッグの設定 185
2.3.10 リンクの設定 195
2.3.11 規格の設定 202
2.3.12 他システムとの移行の設定 207
2.3.13 リスト出力の設定 234
2.3.14 その他の設定 236
2.3.15 プロジェクトの種類の設定時の注意事項 254
2.4 環境変数の設定 255
2.4.1 環境変数の設定方法 255
2.4.2 コンパイラ環境変数の一覧 259
2.4.3 コンパイラ環境変数の詳細 260
2.5 リンカオプションの設定 274
2.5.1 リンカオプションの設定方法 274
2.6 コンパイルの方法 276
2.6.1 ビルド 276
2.6.2 クイックビルド 278
2.6.3 リビルド 279
2.6.4 ファイル単位のコンパイル 279
2.6.5 すべてのプロジェクトに対する操作 280

3 実行 281

3.1 概要 282
3.1.1 実行方法の種類 282
3.1.2 実行支援で使用するファイル 282

3.2 実行の開始と終了 283
3.2.1 実行の開始 283
3.2.2 実行の終了 284
3.3 実行支援ウィンドウでの操作 285
3.3.1 実行支援ウィンドウの構成 285
3.3.2 実行環境の設定 286
3.3.3 画面環境の設定 296
3.3.4 印刷書式の設定 300
3.3.5 実行方法 306
3.3.6 実行環境ファイルの操作 307
3.3.7 実行支援のエラーメッセージ表示について 308

第 2 編 開発ツール

4 COBOL エディタ 309
4.1 概要 310
4.1.1 COBOL 編集に適した機能 310
4.1.2 その他の機能 312
4.2 COBOL エディタの起動と終了 313
4.2.1 COBOL エディタの起動 313
4.2.2 COBOL エディタのプロセス 313
4.2.3 COBOL エディタの終了 314
4.3 COBOL エディタウィンドウ 315
4.3.1 ウィンドウの構成 315
4.3.2 メニューバー 315
4.3.3 ツールバー 319
4.3.4 ポップアップメニュー 321
4.3.5 ルーラー 322
4.3.6 ステータスバー 324
4.3.7 カーソルの形状 324
4.3.8 メニューバーのアイコンとファイル種別 325
4.4 ファイルの操作 326
4.4.1 ファイルの作成 326
4.4.2 ファイルのオープン 326
4.4.3 ファイルの保存 328
4.4.4 ファイルのクローズ 328
4.4.5 ファイル情報の参照 328
4.5 テンプレート機能 331
4.5.1 プログラムテンプレート 331

4.5.2 構文テンプレート 333
4.6 文字列の編集 337
4.6.1 基本操作 337
4.6.2 編集操作 339
4.7 文字列の検索 343
4.7.1 検索 343
4.7.2 置換 345
4.7.3 ブックマークの設定 346
4.7.4 ジャンプ 346
4.7.5 一連番号の検索 347
4.7.6 不当な一連番号の検索 347
4.7.7 80 バイトを超える行の検索 348
4.7.8 タグジャンプ 348
4.8 キーボードマクロ 350
4.8.1 キーボードマクロの登録 350
4.8.2 キーボードマクロの実行 350
4.8.3 キーボードマクロに登録できるキー操作 350
4.9 構文の入力を支援する機能 352
4.9.1 構文チェック 352
4.9.2 構文チェックの環境設定 352
4.9.3 キーワード補完 354
4.10 オプション 357
4.10.1 編集動作の設定 357
4.10.2 表示の設定 358
4.10.3 色の設定 361
4.10.4 画面フォントの設定 364
4.10.5 ファイルの種類の設定 365
4.10.6 保存の設定 366
4.10.7 予約語の設定 367
4.10.8 COBOL ソースの設定 372
4.11 カスタマイズ情報の外部ファイル保存機能 376
4.11.1 保存できるカスタマイズ情報 376
4.11.2 カスタマイズ情報のエクスポート方法 376
4.11.3 カスタマイズ情報のインポート方法 377
4.12 印刷 378
4.12.1 ページレイアウトの設定 378
4.12.2 印刷プレビュー 379
4.12.3 印刷 380

5 ODBC レコード定義生成 381
5.1 概要 382
5.1.1 データソースとは 382
5.1.2 前提条件 382
5.1.3 レコード定義生成の流れ 382
5.2 レコード定義生成の規則 384
5.2.1 データソース中の表から求められる列の情報 384
5.2.2 COBOL2002 のレコード定義を生成するときの規則 385
5.3 レコード定義生成の操作 387
5.3.1 操作方法 387
5.4 レコード定義の生成例 391

6 画面定義 393
6.1 概要 394
6.1.1 画面定義の流れ 394
6.1.2 使用するファイル 394
6.1.3 予約語の扱い 395
6.2 画面定義の開始と終了 396
6.2.1 画面定義の開始 396
6.2.2 画面定義の終了 396
6.3 画面定義ウィンドウ 397
6.3.1 画面定義ウィンドウの構成 397
6.3.2 フィールド構成ウィンドウ 399
6.3.3 フォームエディタウィンドウ 400
6.3.4 属性エディタウィンドウ 401
6.3.5 画面定義の手順例 407
6.4 画面定義の操作 410
6.4.1 基本的な編集操作 410
6.4.2 集団項目の編集 411
6.4.3 フィールドの編集 412
6.4.4 表の編集 415
6.4.5 表要素の編集 419
6.4.6 罫線の編集 421
6.4.7 属性の設定 422
6.5 COBOL ソースファイルの生成 426
6.6 画面定義時のオプション機能 427
6.6.1 環境設定 427
6.6.2 COBOL ソースファイル生成の許可の設定 428
6.6.3 ファイル情報の表示 428

6.7 定義ファイルの操作 429
6.7.1 定義ファイルの作成 429
6.7.2 定義ファイルのオープン 430
6.7.3 定義ファイルの保存 430

7 ファイル／レコード定義 431
7.1 概要 432
7.1.1 ファイル／レコード定義の流れ 432
7.1.2 使用するファイル 432
7.1.3 予約語の扱い 433
7.2 ファイル／レコード定義の開始と終了 434
7.2.1 ファイル／レコード定義の開始 434
7.2.2 ファイル／レコード定義の終了 434
7.3 ファイル／レコード定義ウィンドウ 436
7.3.1 ファイル／レコード定義ウィンドウの構成 436
7.3.2 ファイル仕様定義ウィンドウ 438
7.3.3 レコード仕様定義ウィンドウ 441
7.4 定義ファイルの操作 446
7.4.1 定義ファイルの作成 446
7.4.2 定義ファイルのオープン 447
7.4.3 定義ファイルの保存 447
7.5 データ項目の編集 448
7.5.1 データ項目の選択 448
7.5.2 データ項目の挿入 448
7.5.3 データ項目の属性の編集 448
7.5.4 文字列の複写 449
7.5.5 データ項目の移動 449
7.5.6 データ項目の複写 449
7.5.7 データ項目の削除 450
7.5.8 データ項目の検索 450
7.5.9 データ項目の置換 450
7.5.10 データ項目の位置付け 451
7.5.11 取消 452
7.6 ISAM 情報 453
7.6.1 ISAM 情報の設定 453
7.6.2 合成キーの登録 455
7.6.3 合成キー情報の設定 457
7.6.4 ISAM ファイルの生成 457
7.7 COBOL ソースファイルの生成 458

7.8 ファイル／レコード定義の印刷 459
7.8.1 印刷ページの設定 459
7.8.2 印刷プレビュー 460
7.8.3 印刷 460
7.9 ファイル／レコード定義時のオプション機能 465
7.9.1 環境設定 465
7.9.2 COBOL ソース生成の許可の設定 465
7.9.3 ファイル情報の表示 465
7.9.4 CSV ファイルへの出力 466

第 3 編 テストデバッガ，カバレージ，TD コマンド生成機能，および Unicode 機能

8 テストデバッガ，カバレージ，TD コマンド生成機能，および Unicode 機能の概要 468
8.1 テストデバッガとは 469
8.1.1 デバッグ機能 469
8.1.2 テスト機能 470
8.1.3 GUI モードとバッチモード 470
8.1.4 テストデバッガの機能一覧 471
8.2 カバレージとは 473
8.2.1 カバレージ情報 473
8.2.2 カウント情報 473
8.2.3 GUI モードとバッチモード 473
8.2.4 カバレージの機能一覧 474
8.3 TD コマンド生成機能とは 475
8.3.1 TD コマンド生成機能を使用するときに指定するコンパイラオプション 475
8.3.2 TD コマンド生成機能の対象となるプログラム 475
8.4 Unicode 機能 476
8.4.1 機能の概略 476
8.4.2 機能の詳細 478
8.4.3 その他の注意事項 484

9 テストデバッガの概要と規則 486
9.1 テストデバッガの前提条件 487
9.1.1 テストデバッガの入出力構成と使用するファイル 487
9.1.2 プログラムのコンパイル 490
9.1.3 テストしたいプログラムの実行 491
9.1.4 テストプログラムの構成 492
9.1.5 文字コード 492
9.1.6 その他の注意事項 492

9.2 テストデバッグの概要 494
9.2.1 プログラムの中断 494
9.2.2 プログラムの実行 497
9.2.3 プログラムの状態の表示 501
9.2.4 プログラムの実行の追跡 502
9.2.5 データの操作 503
9.2.6 領域の操作 505
9.2.7 プログラムの単体テスト 506
9.2.8 バッチによるテスト 519
9.2.9 テストデバッグのための設定 523
9.2.10 GUI の操作のための機能 526
9.3 データ操作の規則 527
9.3.1 データの比較・代入規則 527
9.3.2 データ名・ファイル名が参照できる範囲 533
9.3.3 プログラムの実行環境の文字コード 534
9.3.4 ファクトリオブジェクトとインスタンスオブジェクト 534
9.4 その他の機能 539
9.4.1 マルチスレッド対応 COBOL プログラムのテストデバッグ 539
9.4.2 OLE2 オートメーションのテストデバッグ 540
9.4.3 プログラムからの連動実行 542
9.4.4 再帰によるソース要素の実行 542
9.4.5 共通例外を発生させるプログラムのテストデバッグ 543
9.4.6 数字項目のけた拡張機能（Windows(x64) COBOL2002 で有効） 543
9.4.7 英数字定数の定数長拡張機能 546
9.5 ソース要素・中断点・通過点の表示形式 547
9.5.1 ソース要素の表示形式 547
9.5.2 入口・出口・文 547
9.5.3 スレッド ID 548

10 GUI モードによるテストデバッグ 549
10.1 GUI モードによるテストデバッグの概要 550
10.1.1 テストデバッグの手順 550
10.2 起動と終了 553
10.2.1 テストデバッガの起動 553
10.2.2 テストデバッガの終了 556
10.3 テストデバッグで使用するウィンドウ 558
10.3.1 テストデバッグウィンドウ 558
10.3.2 ソーステキストウィンドウ 563
10.3.3 テストデバッグ一覧ウィンドウ 566

10.3.4 呼び出し一覧ウィンドウ 567
10.3.5 常時データ表示ウィンドウ 568
10.3.6 モニタウィンドウ 571
10.3.7 TD コマンド入力ウィンドウ 572
10.3.8 標準配置 573
10.3.9 テストデバッグで使用するウィンドウのカスタマイズ 573
10.4 テストデバッグ対象とする実行可能ファイルと DLL ファイルの変更 578
10.5 テストデバッグ機能の初期化 579
10.6 TD コマンド格納ファイルの入力 580
10.7 ソーステキストの表示 581
10.7.1 テストデバッグ一覧ウィンドウからの表示 581
10.7.2 呼び出し一覧ウィンドウからの表示 582
10.7.3 ソーステキストウィンドウからの表示 582
10.7.4 登録集原文の表示／非表示 582
10.7.5 中断時のソーステキストの表示 583
10.7.6 指定行の表示 584
10.7.7 すべてのソーステキストウィンドウを閉じる 584
10.8 中断点の設定と解除 585
10.8.1 中断点の設定 585
10.8.2 中断点の解除 586
10.8.3 中断点一覧の表示 587
10.8.4 中断点の全解除 589
10.9 データ監視条件の設定と解除 590
10.9.1 データ監視条件の設定 590
10.9.2 データ監視条件一覧の表示と解除 593
10.10 プログラムの実行制御 595
10.10.1 プログラムの実行開始 595
10.10.2 連続実行 596
10.10.3 ステップイン 596
10.10.4 ステップオーバー 596
10.10.5 ジャンプ 597
10.10.6 ジャンプ実行 597
10.10.7 実行の中止 597
10.10.8 実行の割り込み中断 598
10.11 プログラムの実行追跡 599
10.11.1 アニメーション 599
10.11.2 アニメーション抑止の設定 599
10.11.3 アニメーション抑止の解除 600
10.11.4 バックトレース 600

10.12 実行時要素の切り替え 602
10.13 データ値の表示と代入 603
10.13.1 データ値の表示 603
10.13.2 データ値の代入 605
10.14 現在位置の表示 606
10.15 ソースファイルの編集 607
10.16 文字列の検索 608
10.16.1 ウィンドウ内の文字列検索 608
10.17 プログラムからの連動実行 609
10.17.1 操作手順 609
10.17.2 DLL ファイルだけの連動実行について 613
10.17.3 連動実行のエラーについて 615
10.18 テストデバッグのオプション設定 617
10.18.1 実行環境 617
10.18.2 ソーステキスト 618
10.18.3 モニタ 619
10.18.4 指定規則 620
10.18.5 データ表示 621

11 バッチモードによるテストデバッグ 622
11.1 バッチモードによるテストデバッグの概要 623
11.2 バッチモードでのテストの方法 624
11.2.1 テストの手順 624
11.2.2 環境変数の指定 624
11.3 cbltd2k コマンドの指定形式 625

12 TD コマンド 627
12.1 TD コマンドの指定方法 628
12.1.1 TD コマンドの形式 628
12.1.2 TD コマンドの行 628
12.1.3 空白と括弧を含む文字列の指定方法 628
12.1.4 英大文字と英小文字，英数字文字と拡張文字を区別しての文字列の指定方法 629
12.1.5 注釈 629
12.1.6 等価規則 629
12.2 オペランドの指定方法 631
12.2.1 ソース要素の名称 631
12.2.2 行番号（＝一連番号） 631
12.2.3 文番号 631
12.2.4 入口名 632

12.2.5 手続き名 632
12.2.6 データ名 632
12.2.7 定数 633
12.2.8 既定義オブジェクト名 635
12.2.9 オペランドの補助語 635
12.2.10 TD 利用者定義語 635
12.2.11 翻訳単位指定 636
12.2.12 ソース要素指定 637
12.2.13 文番号指定 638
12.2.14 手続き名指定 638
12.2.15 入口，出口指定 638
12.2.16 データ名指定 639
12.2.17 ファイル名指定 639
12.2.18 TD コマンド群 639
12.2.19 カウンタ変数 641
12.2.20 記号名 642
12.2.21 繰り返し指定 643
12.2.22 入出力文選択指定 643
12.2.23 比較条件式 643
12.2.24 OLE プロパティ指定 644
12.3 TD コマンドの一覧 645
12.3.1 プログラムの実行の制御と追跡 645
12.3.2 データの操作 645
12.3.3 単体テスト 646
12.3.4 バッチによるテスト 647
12.3.5 テストデバッグの環境設定 647
12.4 TD コマンドの詳細 648
12.4.1 ALLOCATE AREA／FREE AREA（領域の確保と解放） 648
12.4.2 ASSIGN ADDRESS（アドレスの取得） 649
12.4.3 ASSIGN CASECODE（ケースコードの設定） 650
12.4.4 ASSIGN DATA（データの値代入） 651
12.4.5 CASE（テストケースの指定） 651
12.4.6 DISPLAY BREAK（中断点の表示） 652
12.4.7 DISPLAY COMMENT（注釈の表示） 653
12.4.8 DISPLAY DATA（データの値表示） 653
12.4.9 DISPLAY FLOW（フロー情報の表示） 658
12.4.10 DISPLAY OBJECT／DISPLAY FACTORY（オブジェクトのデータ値の表示） 660
12.4.11 DISPLAY POINT（現在位置の表示） 666
12.4.12 GETPROPERTY（プロパティの値表示） 667

12.4.13 GO（実行の開始／再開） 669
12.4.14 GO END・GO INVALID・GO EOP・GO ERROR（入出力条件シミュレーション） 670
12.4.15 IF（データの値比較） 671
12.4.16 QUIT（テストデバッグの終了） 672
12.4.17 REPEAT（繰り返し指定） 672
12.4.18 SELECT ACTION（入出力文選択指定） 673
12.4.19 SET BREAK／RESET BREAK（中断点の設定と解除） 673
12.4.20 SET FLOW／RESET FLOW（フロー情報の蓄積開始と終了） 677
12.4.21 SET LOG／RESET LOG（モニタウィンドウ表示内容のファイル出力の設定と解除） 678
12.4.22 SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除） 679
12.4.23 SET QUALIFICATION／RESET QUALIFICATION（翻訳単位・ソース要素の指定と解除） 680
12.4.24 SET TRACE／RESET TRACE（トレース表示の開始と中止） 680
12.4.25 SET WATCH／RESET WATCH（データ監視条件の設定と解除） 682
12.4.26 STEP IN（ステップイン実行の開始／再開） 684
12.4.27 STEP OVER（ステップオーバー実行の開始／再開） 684
12.4.28 STEP TO（ステップツー実行の再開） 685
12.4.29 STOP（プログラムの強制終了） 686
12.4.30 SIMULATE DC（DC シミュレーションの設定） 686
12.4.31 SIMULATE FILE（ファイルシミュレーションの設定） 687
12.4.32 SIMULATE MAIN（主プログラムシミュレーションの設定） 688
12.4.33 SIMULATE SUB（副プログラムシミュレーションの設定） 688
12.4.34 TEST（テストケースの選択） 689
12.4.35 #INCLUDE（TD コマンド格納ファイルの取り込み） 690
12.4.36 #OPTION（オプションの変更） 691
12.4.37 レベル番号（記号名構造指定） 692

13 カバレージ機能の概要 693
13.1 概要 694
13.1.1 カバレージ機能の入出力構成と使用するファイル 694
13.1.2 プログラムのコンパイル 699
13.1.3 テストしたいプログラムの実行 699
13.1.4 テストプログラムの構成 700
13.1.5 カバレージ情報の表示と操作 700
13.1.6 カウント情報の表示 700
13.1.7 その他の注意事項 701
13.2 カバレージ情報の表示と操作 702
13.2.1 カバレージ情報 702
13.2.2 カバレージ情報表示の手順 706
13.2.3 プログラムのコンパイル 707

13.2.4 カバレージ情報の蓄積 708
13.2.5 カバレージ情報の表示（テキスト形式） 708
13.2.6 カバレージ情報の表示（CSV 形式） 717
13.2.7 カバレージ情報の操作 723
13.3 カウント情報の表示 725
13.3.1 カウント情報 725
13.3.2 カウント情報表示の手順 725
13.3.3 カウント情報の表示 725

14 GUI モードによるカバレージ 728
14.1 概要 729
14.1.1 カバレージの手順 729
14.1.2 環境変数の指定 729
14.2 開始と終了 731
14.2.1 開発マネージャからの開始 731
14.2.2 ［COBOL2002］下のメニューからの開始 731
14.2.3 終了 731
14.3 カバレージのウィンドウ 732
14.3.1 ウィンドウ 732
14.3.2 メニュー 732
14.3.3 カバレージ情報 733
14.3.4 カバレージ情報のマージ 736
14.3.5 カウント情報 737

15 バッチモードによるカバレージ 740
15.1 概要 741
15.2 カバレージ情報の蓄積 742
15.2.1 コマンドによる方法 742
15.2.2 プログラムからの連動実行による方法 744
15.3 カバレージ情報の表示（テキスト形式） 747
15.3.1 カバレージ情報の表示の方法（テキスト形式） 747
15.4 カバレージ情報の表示（CSV 形式） 750
15.4.1 カバレージ情報の表示の方法（CSV 形式） 750
15.5 カバレージ情報の操作 754
15.5.1 コマンドによる方法 754
15.6 カウント情報の表示 757
15.6.1 コマンドによる方法 757
15.6.2 プログラムからの連動実行による方法 759

16 TD コマンド生成機能 762
16.1 概要 763
16.1.1 機能の概要 763
16.1.2 TD コマンド格納ファイルの構成と出力先 763
16.2 中断点情報とシミュレーション情報 765
16.2.1 中断点情報 765
16.2.2 シミュレーション情報 768
16.2.3 データ設定値 771
16.3 TD コマンドの生成例 774

第 4 編 障害調査支援ツール

17 COBOL2002 情報抽出ツール 781
17.1 概要 782
17.1.1 COBOL2002 情報抽出ツールとは 782
17.1.2 COBOL2002 情報抽出ツールの入出力構成と使用するファイル 782
17.2 COBOL2002 情報抽出ツールの使用方法 785
17.2.1 COBOL2002 情報抽出ツールの使用手順 785
17.2.2 cbl2kinfoextra コマンドの指定形式 785
17.3 情報抽出ツール出力結果リストの出力形式 787
17.3.1 プロセスダンプからの抽出情報 788
17.3.2 実行可能ファイル，DLL ファイルからの抽出情報 797
17.3.3 オブジェクトファイル，標準ライブラリからの抽出情報 799
17.3.4 順編成ファイル（可変長），相対編成ファイルからの抽出情報 800
17.4 COBOL2002 情報抽出ツール使用時の注意事項 803
17.5 COBOL2002 情報抽出ツールのメッセージ 804
17.5.1 メッセージレベル 804
17.5.2 メッセージレベルが「ERR」のメッセージの一覧 804
17.5.3 メッセージレベルが「INF」のメッセージの一覧 805

第 5 編 64bit アプリケーションの開発

18 64bit アプリケーションの開発 807
18.1 Windows(x64) COBOL2002 について 808
18.1.1 使用できない機能 808
18.1.2 使用できないコンパイラオプション 809
18.1.3 使用できない環境変数 810
18.1.4 実行可能ファイルと DLL ファイル 811
18.1.5 オブジェクトファイルとライブラリファイル 811
18.1.6 入出力ファイル 811

18.1.7 プログラム情報ファイル 812
18.1.8 リポジトリファイル 812
18.2 Windows(x64) COBOL2002 開発マネージャの機能 813
18.2.1 表示の相違 813
18.2.2 プロジェクトマスタファイルの相違 813
18.2.3 操作できるファイルの扱いの相違 813
18.2.4 ファイルメニューの相違 814
18.2.5 作成できるプログラムの相違 814
18.2.6 プロジェクト設定の相違 815
18.2.7 ユーザ資産の移行 816
18.3 Windows(x64) COBOL2002 の実行支援の機能 821
18.3.1 実行支援で使用できない機能 821

付録 822
付録 A Windows OS 固有の注意事項 823
付録 A.1 管理者権限についての注意事項 823
付録 A.2 ファイル格納先についての注意事項 828
付録 A.3 文字コードについての注意事項 828
付録 A.4 実行可能ファイルの実行権限についての注意事項 829
付録 A.5 実行時の注意事項 829
付録 A.6 開発マネージャの使用上の注意事項 830
付録 B メニュー項目一覧 833
付録 B.1 開発マネージャウィンドウのメニュー項目一覧 833
付録 B.2 実行支援ウィンドウのメニュー項目一覧 837
付録 B.3 テストデバッグウィンドウのメニュー項目一覧 838
付録 B.4 カバレージウィンドウのメニュー項目一覧 841
付録 B.5 画面定義ウィンドウのメニュー項目一覧 841
付録 B.6 ファイル／レコード定義ウィンドウのメニュー項目一覧 842
付録 B.7 COBOL エディタウィンドウのメニュー項目一覧 844
付録 C 制限値，限界値 848
付録 C.1 コンパイラの制限値，限界値 848
付録 C.2 開発マネージャの制限値，限界値 855
付録 C.3 実行支援の制限値，限界値 856
付録 C.4 画面定義の制限値，限界値 856
付録 C.5 ファイル／レコード定義の制限値，限界値 857
付録 C.6 テストデバッガの制限値，限界値 857
付録 D テストデバッガの例題 860
付録 D.1 例題プログラム 860
付録 D.2 TD コマンドとその結果 864

付録 D.3　カバレージ情報の操作　869
付録 E　カバレージ情報の表示例　871
付録 E.1　カバレージ情報の表示例（テキスト形式）　871
付録 E.2　カバレージ情報の表示例（CSV 形式）　875
付録 E.3　カウント情報の表示例　877
付録 F　このマニュアルの参考情報　878
付録 F.1　関連マニュアル　878
付録 F.2　このマニュアルでの表記　879
付録 F.3　KB（キロバイト）などの単位表記について　881
付録 G　用語解説　882

索引　893

1 COBOL2002 による開発環境

この章では，COBOL2002 を使ってプログラムを開発するときの基本的な操作について説明します。

1.1　概要

COBOL2002 を使ってプログラムを開発するときの環境について説明します。

COBOL2002 の開発環境には，単体のプログラムプロダクトである COBOL2002 Net Developer と，COBOL2002 Net Developer に COBOL2002 Professional Tool Kit とを同梱したプログラムプロダクトである COBOL2002 Developer Professional があります。COBOL2002 Developer Professional でだけ使える機能を次に示します。

COBOL2002 Developer Professional でだけ使える機能

構成管理ツールとの連携

外部ツールとの連携

1.1.1　開発資源のプロジェクトでの管理

プログラムの開発作業を効率的に進めるために，COBOL2002 ではソースファイルや画面定義ファイルなどの複数の資源をプロジェクトとして管理します。プロジェクトとは，実行可能ファイルやダイナミックリンクライブラリ（DLL），標準ライブラリの作成に必要な資源を一括して管理するための概念です。

プロジェクトを定義すると複数のソースファイルのコンパイルや一つの実行可能ファイルの作成などを自動的にできます。また，プロジェクト内のファイル間の依存関係が管理できます。このため，必要なファイルだけを対象とした効率の良いコンパイル・リンケージができます。

開発資源をプロジェクトとして管理するために次に示すような定義をします。

- プロジェクトを構成するソースファイル
- プロジェクトを格納するフォルダ
- 主プログラムのファイル名称
- コンパイル時などに使用するツールのオプションや環境情報
- 実行可能ファイルに登録するアイコン
- コンパイル・リンク時に必要となるプログラム間の依存関係

これらの情報はプロジェクトごとに管理されます。

1.1.2　プロジェクトマスタによるプロジェクトの管理

プロジェクトマスタとは，複数のプロジェクトを一括して管理するための概念です。開発マネージャを使うと，実行可能ファイルや DLL，標準ライブラリごとに必要な複数のプロジェクトを，プロジェクトマスタで一元的に管理できます。

プロジェクトマスタの定義情報，およびプロジェクトマスタに含まれるプロジェクトの定義情報は，すべてプロジェクトマスタファイル（.hmf）に格納されます。

1.1.3　開発マネージャによる開発管理

(1)　プロジェクトの管理

プロジェクトは，開発マネージャで管理します。開発マネージャが管理する情報はウィンドウ上で確認できます。

(2)　各種支援ツールの起動

COBOL2002 の開発環境では，実行支援，テストデバッガなどのツールの起動を開発マネージャが支援しています。ツールの起動操作は，開発マネージャのウィンドウからできます。

起動できる支援ツールを次に示します。

- コンパイラ
- COBOL エディタ
- テストデバッガ
- カバレージ
- 実行支援
- ODBC レコード定義生成
- 画面定義
- ファイル／レコード定義

1.1.4　開発マネージャで使用するファイル

開発マネージャで使用するファイルの一覧を表 1-1 に示します。

一覧に記載されていない種類のファイルでも，一般ファイルとして開発マネージャに登録できます。また，ファイルの種類や，ファイルに関連づけるツールを開発マネージャに登録できます。開発マネージャにファイルの種類を登録する方法については，「1.7.4　ファイルの種類の登録」を参照してください。

なお，一般ファイルは登録されるだけで，ビルドの対象になりません。一般ファイルとして登録されたファイルは，ファイルの拡張子が OS 上で登録されている場合には，ファイルが存在していれば編集メニューで開くことができます。

開発マネージャに登録できるファイルは，一般ファイルを除き拡張子の指定が必須です。ただし，次に示す場合は，拡張子のない登録集原文が登録できます。

- COBOL ソースファイルのコンパイル後に，COBOL ソース中で使用している拡張子のない登録集原文が依存ファイルへ自動反映される場合
- 拡張子のない登録集原文を含む COBOL85 Version 5.0 以前のプロジェクトファイル(.pmi)を COBOL2002 へ移行する場合

表 1-1　開発マネージャで使用するファイルの一覧

拡張子	ファイル種別	内容
.bkf	バックアップファイル	プロジェクトマスタファイルのバックアップ情報を格納するファイル。
.cbl ほか※1	COBOL ソースファイル	COBOL 原始プログラムを格納するファイル。
.cbw	stdcall 呼び出し指示ファイル	DLL ファイル中の STDCALL 属性のプログラムを呼び出すとき，呼び出すプログラムの名称を登録するためのファイル。
.cxc※2	カタログファイル	公開識別子とファイルの対応づけを定義するファイル。 拡張子は.cxc でなければならない。
.cxd※2	XML データ定義ファイル	XML ドキュメントの要素と COBOL のデータ構造とのマッピングを記述したファイル。
.dll	ダイナミックライブラリ	DLL（ダイナミックリンクライブラリ）を格納するファイル。
.dtd※2 または .xml※2	XML 文書型定義ファイル	XML ドキュメントのマークづけ要素とその構造を定義したファイル。
.exe	実行可能ファイル	コンパイル，リンケージをして実行可能になったプログラムを格納するファイル。
.flf	ファイル定義ファイル	ファイル定義の情報を格納するファイル。
.hmf	プロジェクトマスタファイル	プロジェクトマスタに関する情報を格納するファイル。
.hmp	プロジェクト情報ファイル	プロジェクトマスタから出力したプロジェクトの情報を格納するファイル。
.ico	アイコンファイル	生成される実行可能ファイルのアイコンを格納するファイル。
.lib	ライブラリファイル	ライブラリを格納するファイル。ライブラリには，オブジェクトプログラムのライブラリである標準ライブラリと，DLL の関数情報を保持するインポートライブラリがある。
.obj	オブジェクトファイル	コンパイルの結果であるオブジェクトプログラムを格納するファィル。
.rep	リポジトリファイル	翻訳単位（クラス定義，インタフェース定義，および関数定義）の定義情報を格納するファイル。
.pkg	パッケージ情報ファイル	起動条件など，ツールの情報を格納するファイル。
.rdf	レコード定義ファイル	レコード定義の情報を格納するファイル。
.wdf	画面定義ファイル	画面定義の情報を格納するファイル。
.utsproj※3	テストプロジェクトファイル	COBOL2002 Professional Tool Kit の機能で使用する，単体テスト支援のテストプロジェクトファイル。

注※1

目的に応じて次の拡張子を使用します。

- 固定形式正書法で書かれた原始プログラムをコンパイルする場合
  .cbl，.cob，.ocb，または環境変数 CBLFIX で指定した拡張子
- 自由形式正書法で書かれた原始プログラムをコンパイルする場合
  .cbf，.ocf，または環境変数 CBLFREE で指定した拡張子

注※2

XML 連携機能で使用できます。

注※3

使用方法については，マニュアル「COBOL2002 Professional Tool Kit 単体テスト支援ガイド」を参照してください。

1.1.5 開発マネージャで使用するファイル名，およびフォルダ名の規則

開発マネージャでは，次の規則に従ったファイル名，およびフォルダ名を指定してください。

ファイル名，およびフォルダ名の規則

- ファイル名およびフォルダ名の長さは，半角 255 文字以内で指定してください。また，パスを含めたファイル名の長さは，絶対パスで 259 文字以内になるようにしてください。
- 次の文字は，ファイル名およびフォルダ名に使用できません。
  ¥ / : * ? " < > | @
- ピリオド，または空白文字で終わるファイル名，およびフォルダ名は指定できません。
- ピリオド，または空白文字で始まるフォルダ名は指定できません。
- 半角かたかなが含まれる UNC 形式(先頭が¥¥)のパスを設定した場合はエラーになりません。
- 開発マネージャではドライブの直後に¥が存在しないパス(例，c:work)を設定した場合はエラーになります。この設定をした場合，実際のパスは OS 依存になり，ユーザが意識していないパスを使用してしまう場合があります。

1.2　開発マネージャの起動と終了

開発資源をプロジェクトとして管理したり，各種支援ツールを利用したりするために開発マネージャを起動します。

1.2.1　開発マネージャの起動方法

開発マネージャの起動方法を説明します。

- Windows のプログラム一覧にある［COBOL2002］下のメニューから［開発マネージャ for COBOL2002］を選ぶ。
- エクスプローラからプロジェクトマスタファイル（.hmf）をダブルクリックする。
  開発マネージャが起動されてウィンドウが表示されます。

ウィンドウには，プロジェクトマスタファイル名，プロジェクトマスタを構成するプロジェクト名，およびプロジェクトを構成するファイル名が表示されます。プロジェクトを構成するファイルは，ソースファイル，依存ファイルなど，ファイルの種類に応じて分類されたツリー構造で表示されます。

開発マネージャは同時に二つ以上起動できます。ただし，一つのプロジェクトマスタファイルは，複数の開発マネージャから同時に開けません。

補足事項

開発マネージャの起動と同時に，前回編集したプロジェクトマスタファイルを自動的に開くように設定できます。詳細は，「1.7.1　動作オプションの設定」を参照してください。

1.2.2　開発マネージャの終了方法

開発マネージャの終了方法を説明します。

- 開発マネージャのウィンドウの［プロジェクトマスタ］メニューから［開発マネージャの終了］を選ぶ。
  開発マネージャが終了してウィンドウが閉じます。

注意事項

開発マネージャを終了する場合，プロジェクトマスタが開いているときは，まずプロジェクトマスタが閉じられます。プロジェクトマスタに設定した情報を保存していない場合には，プロジェクトマスタファイルを保存するかどうかを問い合わせるダイアログボックスが表示されます。

1.3　開発マネージャのウィンドウ

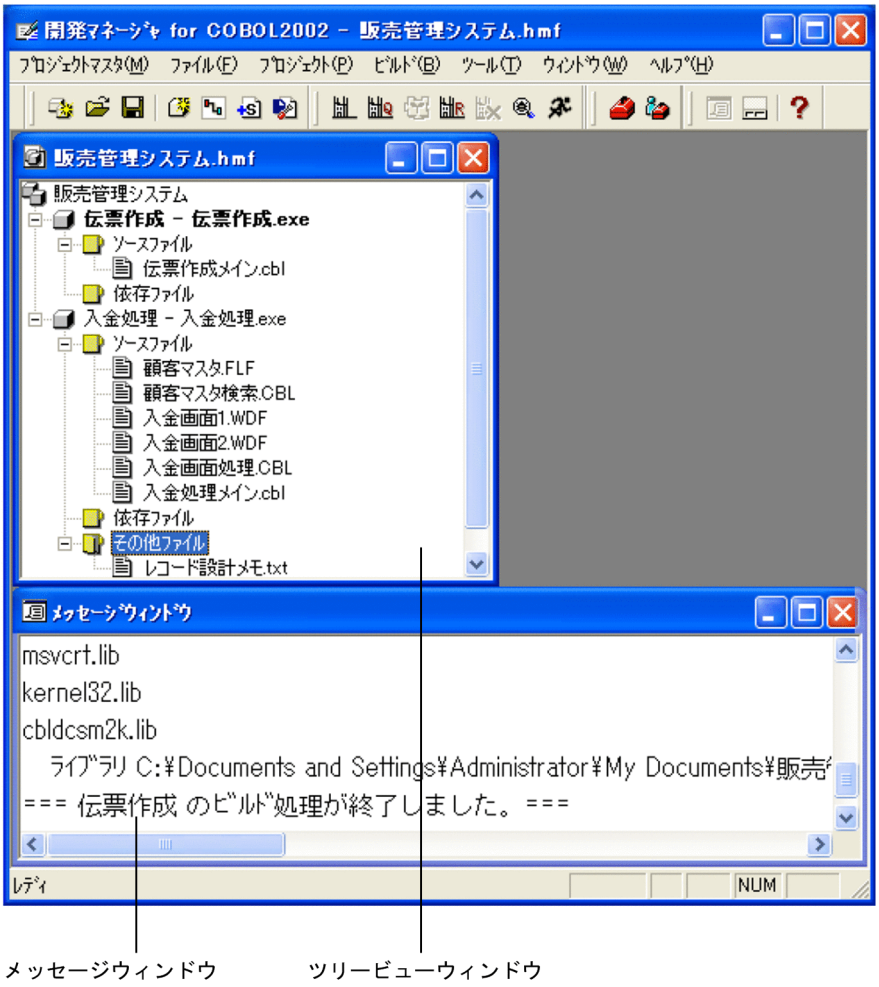
開発マネージャのウィンドウは，プロジェクトの定義やコンパイルの作業時に使用したり，実行支援やテストデバッガなどのツールを起動したりするときに使用します。

ここでは，開発マネージャのウィンドウ構成とメニューの一覧について説明します。

1.3.1　ウィンドウの構成

開発マネージャのウィンドウを次に示します。

図 1-1　開発マネージャのウィンドウ



1.3.2　メニューバー

開発マネージャのウィンドウのメニュー一覧を次に示します。

表 1-2　開発マネージャのウィンドウのメニュー一覧

メニューバー	メニュー項目	機能
プロジェクトマスタ(M)	新規作成(N)	プロジェクトマスタを新規に作成する。
	開く(O)	編集やコンパイルの対象となるプロジェクトマスタを開く。
	閉じる(C)	編集中のプロジェクトマスタを閉じる。
	保存(S)	開発マネージャで編集中のプロジェクトマスタ情報を，プロジェクトマスタファイルに保存する。
	デフォルトオプションの設定(D)※1	プロジェクトを作成したときに仮定されるオプションを設定する。
	プロパティ(P)	プロジェクトマスタの情報を表示する。
	最新のファイル（最新のプロジェクトマスタファイル名が４個まで表示される）	指定したプロジェクトマスタを開く。
	開発マネージャの終了(X)	開発マネージャを終了し，ウィンドウを閉じる。
ファイル(F)	新規作成(N)	プロジェクトに登録するファイルを新規に作成する。
	開く(O)	ツリービューウィンドウで選択しているファイルを，関連づけられたツールで開く。
	削除(D)	ツリービューウィンドウで選択しているファイルを削除する。
	チェックアウト(T)※2	構成管理ツールと連携して，ファイルをチェックアウトする。
	チェックイン(I)※2	構成管理ツールと連携して，ファイルをチェックインする。
	最新を取得(G)※2	構成管理ツールと連携して，最新のファイルを取得する。
	チェックアウト状態の変更(H)※2	構成管理ツールと連携して，チェックアウトされていない場合は，チェックアウト状態にする。また，チェックアウトされている場合は，チェックアウトされていない状態にする。
	ファイルの設定(S)	コンパイラオプション，およびコンパイラ環境変数を，ファイルごとに設定する。
	出力ファイルの編集(E)	ツリービューウィンドウで選択しているファイルの出力ファイルを編集する。
	出力ファイルの変更(U)	定義ファイルをコンパイルしたときに出力先となるファイルを指定する。
	外部ツールを操作するメニュー	外部ツールとの連携で登録済みのメニューを表示する。 COBOL2002 Professional Tool Kit がインストールされていない，または［外部ツールの設定］ダイアログ

メニューバー	メニュー項目	機能
		ボックスにメニュー名が指定されていない場合は，このメニューは表示されない。
	プロパティ(P)	ツリービューウィンドウで選択しているファイルの情報を表示する。
プロジェクト(P)※3	プロジェクトの作成(A)	プロジェクトを新規に作成し，編集中のプロジェクトマスタに追加する。
	プロジェクト名の変更(N)	プロジェクトの名称を変更する。
	プロジェクトのエクスポート(X)	プロジェクトの情報をファイルに出力する。
	プロジェクトのインポート(I)	プロジェクトの情報をファイルから取り込む。
	作業中プロジェクトの設定(W)※4	作業中プロジェクトを切り替える。
	プロジェクトの削除(E)※4	プロジェクトを削除する。
	プロジェクトの関係設定(R)※4	プロジェクトの親子関係を定義する。
	ソースファイルの追加(F)	[ファイルの追加(C)]： プロジェクトにファイルを追加する。 [フォルダ指定によるファイルの追加(D)]： フォルダを指定して，プロジェクトにファイルを追加する。
	プロジェクトの設定(S)	コンパイラオプション，およびコンパイラ環境変数を，プロジェクトごとに設定する。
	依存ファイルの設定(D)	ソースファイルに対する依存ファイルを，依存ファイルの仮登録から選ぶ。
	依存ファイルの仮登録(T)	依存ファイルをプロジェクトマスタに登録する。
	カバレージ(O)	カバレージを起動する。
	プロパティ(P)	プロジェクトの情報を表示する。
ビルド(B)※3	ビルド(B)	作業中プロジェクト，およびその子プロジェクトに属するソースファイルをコンパイル，リンケージする。また，定義ファイルからはソースを生成する。このとき，コンパイルの不要なプログラムや生成の不要な定義ファイルはビルドの対象としない。 オブジェクト指向機能または利用者定義関数を使用したプログラムのプロジェクトの場合，ファイルの依存関係を再構築する。
	クイックビルド(Q)	ビルドと同じ処理をする。ただし，オブジェクト指向機能または利用者定義関数を使用したプロジェクトの場合，ファイルの依存関係を再構築しない。
	コンパイル(C)	ツリービューウィンドウでソースファイルを選択している場合，そのソースファイルをコンパイルする。

メニューバー	メニュー項目	機能
		ツリービューウィンドウで定義ファイルを選択している場合，その定義ファイルからソースを生成する。
	リビルド(R)	ビルドと同様にプロジェクトに登録したファイルをすべてコンパイル，リンケージする。 ビルドと異なり，プログラムやコンパイラオプションなどに変更がなくても，無条件にすべてのファイルをコンパイル，リンケージし，最終生成ファイルを出力する。
	すべてのプロジェクトに対する操作(P)※4	[ビルド(B)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，ビルドを実行する。 [クイックビルド(Q)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，クイックビルドを実行する。 [リビルド(R)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，リビルドを実行する。 [依存関係更新(A)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，依存関係の更新を実行する。
	ビルドの中止(S)	ビルド，クイックビルド，またはリビルドを中止する。
	依存関係更新(A)	作業中プロジェクト，およびその子プロジェクトに登録されているすべてのソースファイルについて，依存関係を更新する。
	デバッガ(D)	テストデバッガを起動する。
	実行(E)	実行可能ファイルを，実行環境を設定しないでそのまま実行する。
	実行支援(L)	実行支援を起動する。
ツール(T)※3	ODBC 定義(D)	ODBC レコード定義生成を起動する。
	開発マネージャの動作オプション(O)	開発マネージャの動作，およびツールバーの表示を設定する。
	開発マネージャのユーザカスタマイズ(C)	[ユーザインタフェース(U)]： キー操作，およびメッセージウィンドウの文字フォントや色を設定する。 [エディタ(E)]： テキストファイルを編集するエディタを設定する。 [新しいファイルの種類の登録(F)] ※1： 新しいファイルの種類を登録する。 [構成管理ツールの設定(C)] ※1 ※2： 構成管理ツールの操作方法を登録する。

メニューバー	メニュー項目	機能
		[外部ツールの設定(X)] ※1 ※2： 外部ツールを登録する。
	プロジェクトマスタ情報の入出力(P)	[CSV 入力(C)]： CSV ファイルを入力する。 [CSV 出力(S)]： CSV ファイルを出力する。
	COBOL ソース解析(A)※2	COBOL2002 Developer Professional の COBOL ソース解析の機能で使用するメニュー。※5 [ソース解析プロジェクトの使用開始(T)]： ソース解析プロジェクト名を設定し，ソース解析プロジェクトを開始する。 ソース解析プロジェクトが使用中の場合，メニュー名の後ろに，使用中のソース解析プロジェクト名を表示する。 [ソース解析プロジェクトの使用終了(F)]： 使用中のソース解析プロジェクトを終了する。 [ソース解析プロジェクトの削除(D)]： 使用中のソース解析プロジェクトを削除する。 [解析の実行(E)]： COBOL ソースファイルの解析を実行する。 [解析の中止(P)]： 実行中の COBOL ソースファイルの解析を中止する。 [設定(S)]： ［解析生成オプション設定］画面を表示する。 [解析結果(HTML)の表示(H)]： HTML 形式の解析結果のプロジェクトサマリ画面（index.html）を表示する。 [解析結果(CSV)の表示(C)]： CSV 形式の解析結果を格納したフォルダを表示する。
	データ影響波及分析(I)※2	COBOL2002 Developer Professional のデータ影響波及分析の機能で使用するメニュー。※5 [設定(S)]： ［設定］ダイアログボックスを表示する。 [データ影響波及分析用 DB 作成(D)]： 開発マネージャで開いているプロジェクトのデータ影響波及分析用データベースを作成する。 [すべてのプロジェクトの DB 作成(A)]： プロジェクトマスタにあるすべてのプロジェクトのデータ影響波及分析用データベースを作成する。

メニューバー	メニュー項目	機能
	単体テスト支援(T)※2	COBOL2002 Developer Professional の単体テスト支援の機能で使用するメニュー。※5 [単体テスト支援(N)]： 選択した COBOL ソースファイルの単体テスト支援を実行する。
ウィンドウ(W)	新しいウィンドウを開く(N)	ツリービューウィンドウを新しいウィンドウに表示する。
	重ねて表示(C)	ツリービューウィンドウ，およびメッセージウィンドウを重ねて表示する。
	並べて表示(T)	ツリービューウィンドウ，およびメッセージウィンドウを並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	メッセージウィンドウ(M)	メッセージウィンドウを表示する。
	ステータスバー(S)	ステータスバーの表示／非表示を切り替える。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※1

プロジェクトマスタを閉じている状態のときだけ表示されるコマンドです。

注※2

COBOL2002 Developer Professional でだけ表示されるメニュー項目です。

注※3

開発マネージャに登録されているツールがほかにある場合は，メニュー項目が追加されます。

注※4

編集中のプロジェクトマスタに複数のプロジェクトが登録されている場合に使用できます。

注※5

使用方法については，次に示すマニュアルを参照してください。

「COBOL2002 Professional Tool Kit COBOL ソース解析ガイド」

「COBOL2002 Professional Tool Kit データ影響波及分析ガイド」

「COBOL2002 Professional Tool Kit 単体テスト支援ガイド」

1.3.3 ツールバー

ツールバーのボタンを選ぶとメニューと同じ操作ができます。

ツールバーには，プロジェクト管理用，ビルド処理用，ツール用，およびウィンドウ用の 4 種類があります。これらは，通常は一つのツールバーとして表示されていますが，それぞれをマウスで選んで移動できます。

また，これらのツールバーは，［ツール］メニューの［開発マネージャの動作オプション］で表示される動作オプションの設定ダイアログボックスで表示するかどうかを指定できます。詳細は，「1.7.1　動作オプションの設定」を参照してください。

(1)　プロジェクト管理用ツールバー

プロジェクト管理用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. ［プロジェクトマスタ］メニューの［新規作成］
2. ［プロジェクトマスタ］メニューの［開く］
3. ［プロジェクトマスタ］メニューの［保存］
4. ［プロジェクト］メニューの［プロジェクトの作成］
5. ［プロジェクト］メニューの［プロジェクトの関係設定］
6. ［プロジェクト］メニューの［ソースファイルの追加］
7. ［プロジェクト］メニューの［フォルダ指定によるファイルの追加］
8. ［プロジェクト］メニューの［依存ファイルの設定］

(2)　ビルド処理用ツールバー

ビルド処理用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. ［ビルド］メニューの［ビルド］
2. ［ビルド］メニューの［クイックビルド］
3. ［ビルド］メニューの［コンパイル］
4. ［ビルド］メニューの［リビルド］
5. ［ビルド］メニューの［ビルドの中止］
6. ［ビルド］メニューの［デバッガ］
7. ［ビルド］メニューの［実行］

(3)　ツール用ツールバー

ツール用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [ツール] メニューの [開発マネージャの動作オプション]
2. [ツール] メニューの [開発マネージャのユーザカスタマイズ] の [ユーザインタフェース]

(4)　ウィンドウ用ツールバー

ウィンドウ用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。

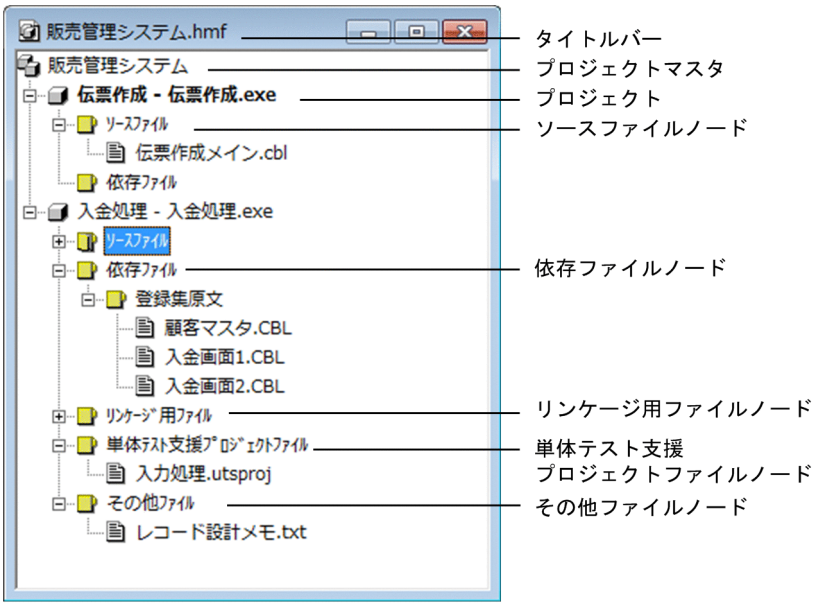


1. [ウィンドウ] メニューの [メッセージウィンドウ]
2. [ウィンドウ] メニューの [ステータスバー]
3. [ヘルプ] メニューの [バージョン情報]

1.3.4　ツリービューウィンドウ

ツリービューウィンドウには，プロジェクトマスタに登録されているプロジェクト，ソースファイル，依存ファイルなどの情報が表示されます。ツリービューウィンドウを次に示します。

図 1-2　ツリービューウィンドウ



タイトルバー

プロジェクトマスタファイル名が表示されます。また，ツリービューウィンドウを複数開いている場合，「：(コロン)」に続いてウィンドウの番号が表示されます。

プロジェクトマスタ

プロジェクトマスタを表します。

プロジェクト

プロジェクトマスタに登録されているプロジェクトを表します。名称が太字で表示されているプロジェクトは，作業中プロジェクトを表します。

ソースファイルノード

プロジェクトに登録されているソースファイルのグループを表します。

プロジェクトに登録したソースファイルは，このノードの下位に表示されます。

ソースファイルとして登録されるファイルの種類を，次に示します。

- COBOL ソースファイル
- ファイル定義ファイル
- レコード定義ファイル
- 画面定義ファイル
- HTML ファイル
- XML データ定義ファイル

ソースファイルの状態は，ツリービューのアイコンでわかります。詳細は「(2)　ツリービューウィンドウのアイコン」を参照してください。

依存ファイルノード

プロジェクトに登録されている依存ファイルのグループを表します。

プロジェクトに登録した依存ファイルは，このノードの下位に表示されます。

依存ファイルとして登録されるファイルの種類を，次に示します。

- COBOL 登録集原文
- stdcall 呼び出し指示ファイル（プロジェクトに関連づける場合）
- stdcall 呼び出し指示ファイル（ファイルごとに関連づける場合）
- リポジトリファイル
- XML 文書型定義ファイル

リンケージ用ファイルノード

プロジェクトに登録されているリンケージ用ファイルのグループを表します。

プロジェクトに登録したリンケージ用ファイルは，このノードの下位に表示されます。

リンケージ用ファイルとして登録されるファイルの種類を，次に示します。

- アイコンファイル
- オブジェクトファイル
- ライブラリファイル

単体テスト支援プロジェクトファイルノード

COBOL2002 Developer Professional の単体テスト支援を使用した場合に作成されます。

単体テスト支援のテストプロジェクトファイルのグループを表します。

テストプロジェクトファイルは，このノードの下位に表示されます。

単体テスト支援については，マニュアル「COBOL2002 Professional Tool Kit 単体テスト支援ガイド」を参照してください。

その他ファイルノード

プロジェクトに登録されている一般のファイルのグループを表します。

ソースファイル，依存ファイル，およびリンケージ用ファイル以外のファイルをプロジェクトに登録した場合，このノードの下位に表示されます。

注意事項

プロジェクト名，およびファイル名は，259 バイトまでツリービューウィンドウに表示されます。

(1) コメントの設定

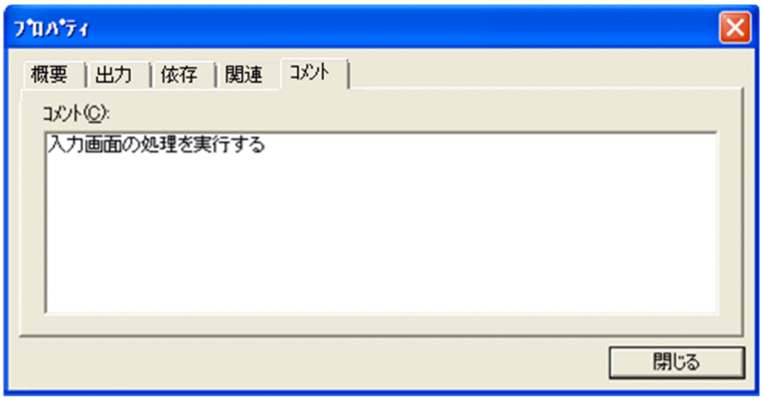
開発マネージャのツリービューウィンドウでは，登録されているプロジェクト，およびファイルにコメントを設定できます。ファイルにコメントを設定する手順を次に示します。プロジェクトにコメントを設定する手順については，「1.5.4 プロジェクトの定義内容の確認」を参照してください。

1. 開発マネージャのウィンドウのツリービューウィンドウで，コメントを登録したいファイルを選ぶ。
2. 開発マネージャのウィンドウの［ファイル］メニューから［プロパティ］を選ぶ。

   プロパティダイアログボックスが表示されます。

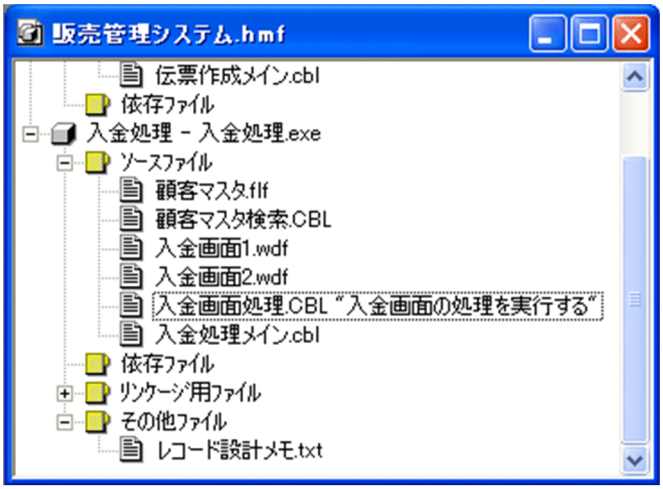
3. プロパティダイアログボックスの［コメント］タブを選ぶ。

コメントを入力するテキストボックスが表示されます。



4. コメントを入力して［閉じる］ボタンを選ぶ。

ファイルにコメントが設定されます。動作オプションの設定ダイアログボックスで，「ツリービューウィンドウにコメントを表示する」が設定されている場合だけ，ツリービューウィンドウに，次のようにコメントが表示されます。



コメントは，128 バイトまでツリービューウィンドウに表示されます。また，128 バイト未満で改行を入力した場合は，改行までの文字が表示されます。

(2)　ツリービューウィンドウのアイコン

次の条件で，ツリービューのアイコンが切り替わります。

- ソースファイルにファイル単位のオプションが設定されている。
- ソースファイルがチェックアウトされている。
- ソースファイルにファイル単位のオプションが設定されていて，かつチェックアウトされている。

ソースファイルの状態とツリービューのアイコンの対応を次に示します。

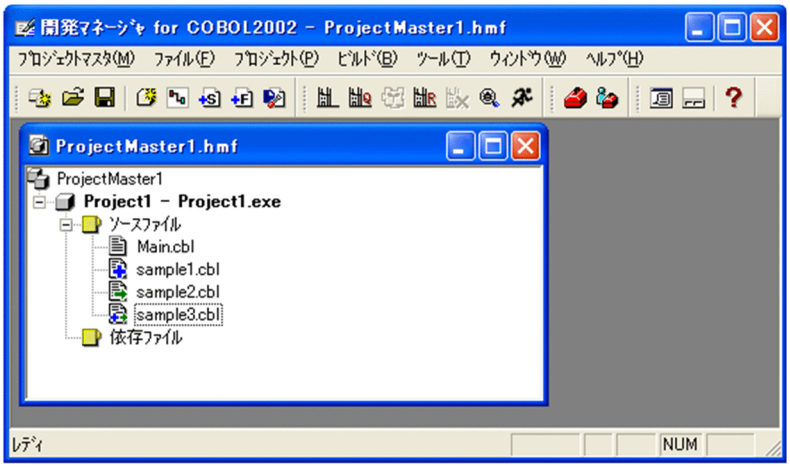
　　通常（上記の条件に該当しない）

 　ファイル単位のオプション設定あり

 　チェックアウト時

 　ファイル単位のオプション設定ありで，かつチェックアウト時

オプションが設定してあるソースファイルを含むプロジェクトのメイン画面を次に示します。



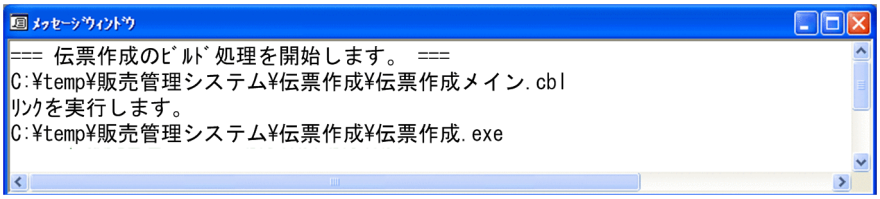
(3)　コメントの表示

ツリービューウィンドウのコメントは，表示／非表示を切り替えられます。

コメントの表示方法は，動作オプションの設定ダイアログボックスで設定します。詳細は，「1.7.1　動作オプションの設定」を参照してください。

1.3.5　メッセージウィンドウ

図 1-3　メッセージウィンドウ



メッセージウィンドウには，ビルド処理の状況が表示されます。

テキストエディタにエラージャンプ時の起動オプションが設定されている場合，メッセージウィンドウに表示されているエラーメッセージをダブルクリックすると，エラーの発生したファイルをエディタで参照できます。詳細は，「1.7.3　テキストエディタの設定」および「2.6.1　ビルド」の「(5)　コンパイルエラーの修正」を参照してください。

メッセージウィンドウの文字フォントや配色は，ユーザが使いやすいように変更できます。詳細は，「1.7.2 ユーザインタフェースの設定」の「(2)　メッセージウィンドウの表示の設定」を参照してください。

1.3.6　開発マネージャのウィンドウの注意事項

プロジェクトマスタ，プロジェクト，およびファイルのプロパティダイアログボックスで表示される日付と時刻の形式は，Windows のシステムの設定に従います。詳細は，Windows のマニュアルを参照してください。

1.4　プロジェクトマスタの定義

プロジェクトマスタは，複数のプロジェクトをまとめて管理するための概念です。

ここでは，プロジェクトマスタの操作方法，およびプロジェクトマスタに定義した情報の確認方法について説明します。

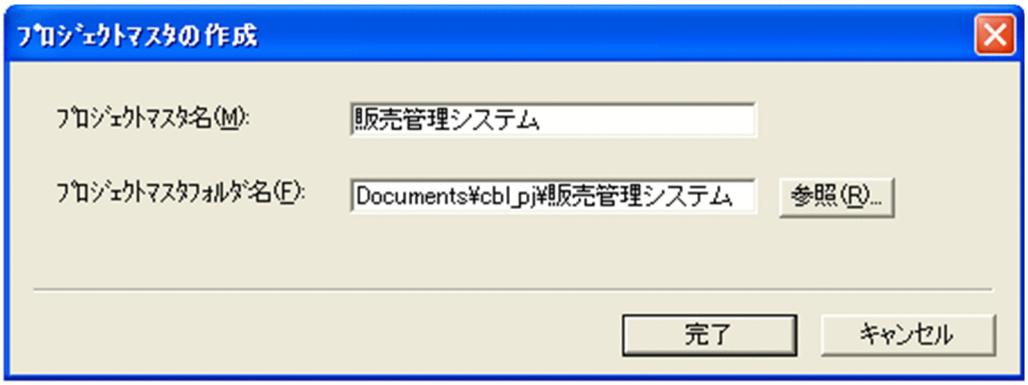
1.4.1　プロジェクトマスタの新規作成

プロジェクトマスタが未作成の場合や，必要なプロジェクトマスタがない場合には，新規にプロジェクトマスタを作成する必要があります。作成したプロジェクトマスタには，プロジェクトや，プログラムの作成に必要なファイルを登録します。ここでは，プロジェクトマスタを新規に作成する操作方法を説明します。

プロジェクトマスタを作成する場合は，プロジェクトマスタの作成ダイアログボックスで定義します。このダイアログボックスでは，プロジェクトマスタの名称，およびプロジェクトマスタで使用するフォルダを指定します。

プロジェクトマスタの作成ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのウィンドウの［プロジェクトマスタ］メニューから［新規作成］を選ぶ。
   次に示すプロジェクトマスタの作成ダイアログボックスが表示されます。



2. 「プロジェクトマスタ名」，および「プロジェクトマスタフォルダ名」を指定して［完了］ボタンを選ぶ。
   プロジェクトマスタが作成されます。

プロジェクトマスタを作成すると，プロジェクトマスタフォルダにプロジェクトマスタファイル（プロジェクトマスタ名.hmf）が作成されます。プロジェクトマスタファイルは，プロジェクトマスタ，およびプロジェクトマスタに含まれるすべてのプロジェクトの情報を管理するファイルです。

プロジェクトマスタを新規に作成すると，プロジェクトの作成ダイアログボックスが表示され，プロジェクトを作成する手順に入ります。引き続きプロジェクトを作成してください。プロジェクトの作成手順については，「1.5.1　プロジェクトの作成」の手順 2.以降を参照してください。

(1) プロジェクトマスタ名

プロジェクトマスタ名を指定します。

プロジェクトマスタ名には，Windows のファイル名の規則に従った名称を指定してください。ただし，「@」を含む名称は指定できません。

(2) プロジェクトマスタフォルダ名

プロジェクトマスタファイルを作成するフォルダを指定します。パス名を直接入力するか，［参照］ボタンからフォルダを選択してください。

なお，指定したフォルダがない場合は，新しいフォルダを作成してよいか確認するダイアログボックスが表示されます。

(3) プロジェクトマスタフォルダ名の初期値

プロジェクトマスタフォルダ名の初期値は，次のようになります。

- 1 回もプロジェクトマスタを作成していない場合
  カレントフォルダ
- 2 回目以降の作成の場合
  直前にプロジェクトマスタを作成したフォルダの一つ上のフォルダ

プロジェクトマスタ名に文字列を入力すると，プロジェクトマスタフォルダ名に"¥+プロジェクトマスタ名文字列"が自動的に反映されます。

また，プロジェクトマスタフォルダ名に指定した文字列が絶対パスでない場合，［完了］ボタンを選んだタイミングで「直前にプロジェクトマスタを作成したフォルダの一つ上のフォルダ」からの相対パスとしてプロジェクトマスタフォルダが仮定されます。

1.4.2 プロジェクトマスタを開く

(1) メニューから開く

1. 開発マネージャのウィンドウの［プロジェクトマスタ］メニューから［開く］を選ぶ。
   プロジェクトマスタを開くためのダイアログボックスが表示されます。

2. プロジェクトマスタファイル名を指定して［開く］ボタンを選ぶ。
   指定したプロジェクトマスタが開いて，開発マネージャのウィンドウに表示されます。
   このとき，ファイルの種類を切り替えることによって，プロジェクトマスタファイル（.hmf）でないファイルを選択できます。プロジェクトマスタファイルでないファイルを選択した場合は，プロジェク

トマスタファイルに変換され保存されます。以降は変換後のプロジェクトマスタファイルを使用してください。

COBOL プロジェクトファイル（.pmi），およびプロジェクトマスタファイル（.hmf）を COBOL2002 のプロジェクトマスタファイルに移行できます。これは，COBOL85 から COBOL2002 へ移行するための移行機能です。

(2) ファイル一覧から開く

1. 開発マネージャのウィンドウの［プロジェクトマスタ］メニューに表示されているファイル一覧からプロジェクトマスタファイル名を選ぶ。

   指定したプロジェクトマスタが開かれ，開発マネージャのウィンドウに表示されます。

   ファイル一覧には，最近編集したプロジェクトマスタファイルが 4 個まで表示されます。

(3) ドラッグ＆ドロップで開く

1. エクスプローラから開発マネージャのウィンドウへプロジェクトマスタファイルをドラッグ＆ドロップする。

   指定したプロジェクトが開かれ，開発マネージャのウィンドウに表示されます。

注意事項

- 複数のプロジェクトマスタファイルをドラッグ＆ドロップすることはできません。
- プロジェクトマスタファイルとその他のファイル（ソースファイルなど）をまとめてドラッグ＆ドロップすることはできません。

(4) プロジェクトマスタファイルのバックアップファイル

プロジェクトマスタを開くと，現在のプロジェクトマスタファイルの内容がバックアップファイル（プロジェクトマスタ名.bkf）として保存されます。

バックアップファイルは，プロジェクトマスタフォルダに作成されます。プロジェクトマスタフォルダに作成できない場合は，Windows のテンポラリフォルダに作成されます。どちらのフォルダにも作成できなかった場合は，エラーメッセージがメッセージウィンドウに表示され，プロジェクトマスタは開かれません。

1.4.3 プロジェクトマスタを閉じる

1. 開発マネージャのウィンドウの［プロジェクトマスタ］メニューから［閉じる］を選ぶ。

   編集中のプロジェクトマスタが閉じます。

   プロジェクトマスタに設定した情報を保存していない場合には，プロジェクトマスタファイルを保存するかを確認するダイアログボックスが表示されます。

また，開発マネージャから COBOL エディタを起動している場合，開発マネージャを終了すると COBOL エディタで編集中のファイルも閉じられます。この場合，ファイルの変更が保存されていないときは，ファイルを保存するかどうかを確認するダイアログボックスが表示されます。

注意事項

- ビルド，クイックビルド，リビルド，または依存関係更新処理を実行中の場合，およびテストデバッガ，カバレージを起動中の場合，プロジェクトマスタを閉じることはできません。
- プロジェクトマスタを閉じる場合，COBOL エディタで編集していたファイルがすべて閉じられます。このとき，保存していないファイルがあると，保存するかどうかを問い合わせるダイアログボックスが表示されます。

1.4.4　プロジェクトマスタの保存

1. **開発マネージャのウィンドウの［プロジェクトマスタ］メニューから［保存］を選ぶ。**

   現在開いているプロジェクトマスタの情報がプロジェクトマスタファイルに保存されます。

   プロジェクトマスタの保存時にも，プロジェクトマスタファイルのバックアップファイルが作成されます。プロジェクトマスタの保存時にバックアップファイルが作成できなかった場合は，エラーメッセージがメッセージウィンドウに表示されますが，プロジェクトマスタの保存は実行されます。

   バックアップファイルの詳細については，「1.4.2　プロジェクトマスタを開く」の「(4)　プロジェクトマスタファイルのバックアップファイル」を参照してください。

1.4.5　プロジェクトマスタの定義内容の確認

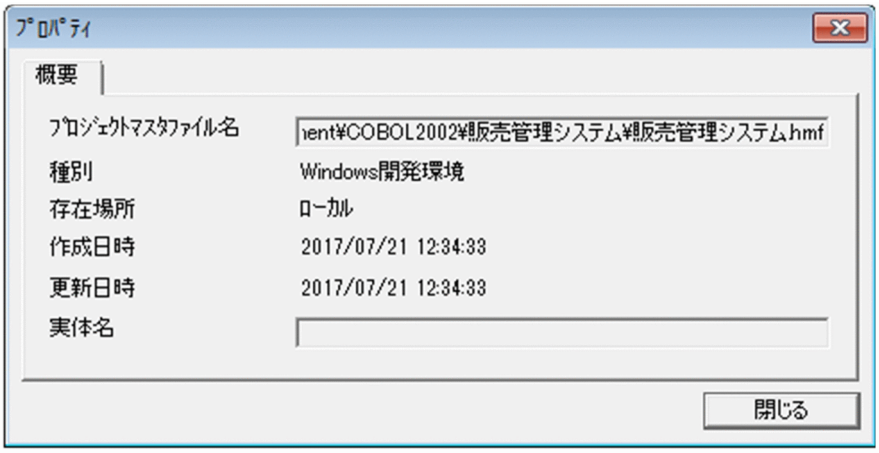
プロジェクトマスタに定義したプロジェクトマスタファイル名，種別，更新日時などの情報は，プロジェクトマスタのプロパティダイアログボックスで確認できます。プロジェクトマスタのプロパティダイアログボックスの表示手順を次に示します。

1. **開発マネージャのウィンドウの［プロジェクトマスタ］メニューから［プロパティ］を選ぶ。**

   プロジェクトマスタのプロパティダイアログボックスが表示されます。

次に，このダイアログボックスで確認できる内容について説明します。

(1) 概要



プロジェクトマスタファイル名

プロジェクトマスタファイルの名称が，絶対パス名で表示されます。

種別

プロジェクトマスタの利用する開発環境が表示されます。

存在場所

プロジェクトマスタファイルのある場所が表示されます。

- ローカル

  ローカル（Windows）上にファイルがあります。

作成日時

プロジェクトマスタファイルを作成した日時が表示されます。

更新日時

プロジェクトマスタファイルを最後に更新した日時が表示されます。

実体名

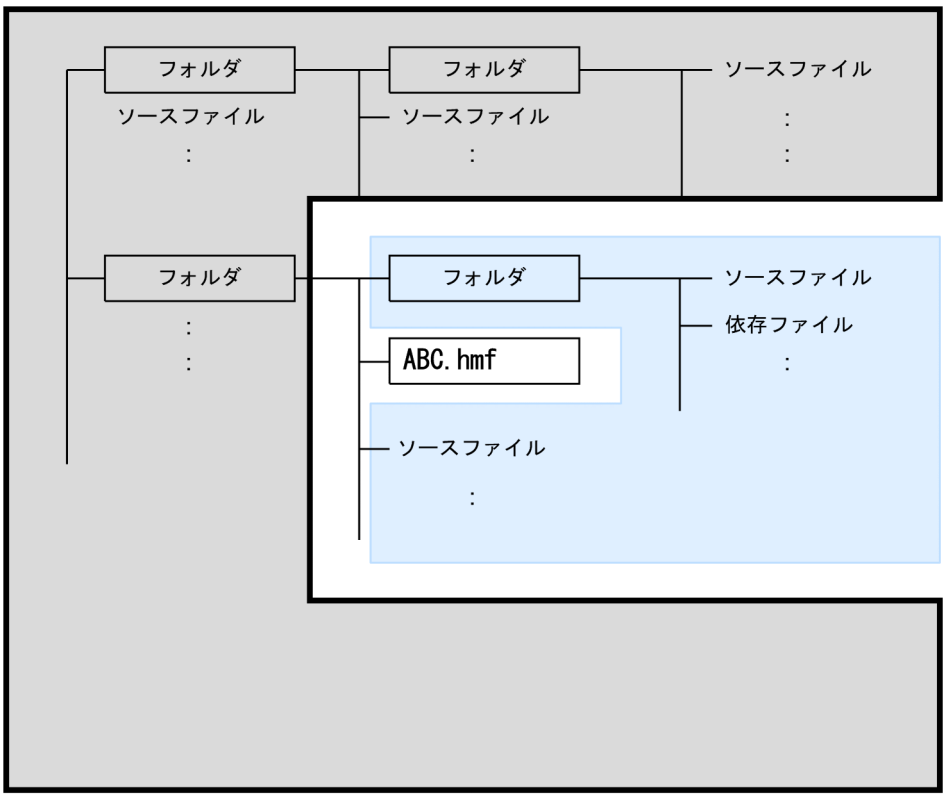
何も表示されません。

1.4.6 プロジェクトマスタの複写

プロジェクトマスタファイル内では，プロジェクトマスタフォルダ（プロジェクトマスタファイルがあるフォルダ）より下位にあるファイルやフォルダは，プロジェクトマスタフォルダからの相対位置で管理されます。そのため，プロジェクトマスタに登録したファイルやフォルダが，すべてプロジェクトマスタフォルダの下位にある場合は，エクスプローラなどでプロジェクトマスタフォルダごと別の場所に複写できます。

しかし，プロジェクトマスタフォルダより上位にあるファイルやフォルダは絶対パスで管理されるため，複写する際注意しなければならない事項があります。

次のようにファイルやフォルダが構成されている場合を例に，プロジェクトマスタの複写について説明します。



（凡例）

：1.

：2.

1 の範囲のファイルがプロジェクトマスタファイル（ABC.hmf）に登録されている場合

プロジェクトマスタファイル（ABC.hmf）があるフォルダを下位構成ごと複写すれば，複写したプロジェクトマスタ内のプロジェクトをビルドまたはリビルドできます。

2 の範囲のファイルがプロジェクトマスタファイル（ABC.hmf）に登録されている場合

プロジェクトマスタファイル（ABC.hmf）を複写し，ビルドまたはリビルドすると，構成ファイルを検索できないためエラーになる場合があります。

この場合，プロジェクトマスタに登録されているファイルのうち，2 の範囲のファイルをいったんプロジェクトマスタから削除して，再度プロジェクトマスタに登録する必要があります。

1.5　プロジェクトの定義

プロジェクトは，一つの最終生成ファイル（実行可能ファイル，DLL，および標準ライブラリ）を作成するために必要なファイルの集合を表す概念です。

プロジェクトは，プロジェクトマスタの下位に定義します。一つのプロジェクトマスタの下位には，複数のプロジェクトを定義できます。

ここでは，プロジェクトの操作方法について説明します。

1.5.1　プロジェクトの作成

新しいプロジェクトを作成し，プロジェクトマスタに追加する方法について説明します。

プロジェクトを作成する場合には，プロジェクトの作成ダイアログボックス，およびプロジェクト作成ウィザードで定義します。プロジェクトの作成手順を次に示します。

1. **開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトの作成］を選ぶ。**
   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。

2. **プロジェクト名，プロジェクトフォルダ，作業フォルダを指定して［OK］ボタンを選ぶ。**
   プロジェクトの作成ダイアログボックスが閉じ，プロジェクト作成ウィザードが表示されます。

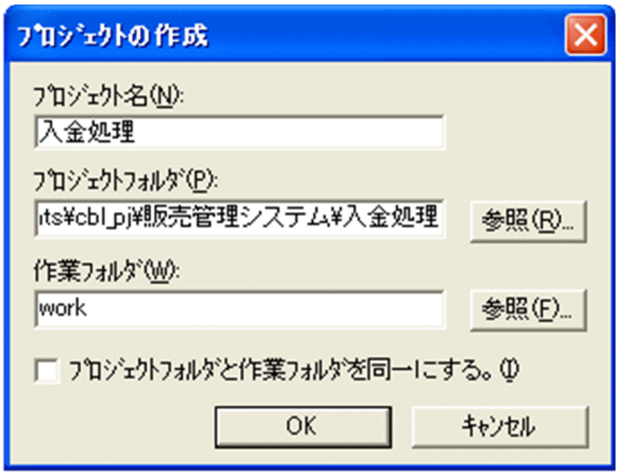
3. **ウィザードに従って，プロジェクトを定義する。**
   次の順序でプロジェクトを定義するダイアログボックスが表示されます。必要な項目を指定してください。
   - 最終生成物の種類
   - プロジェクトの種類
   - 出力ファイル名
   - メインファイル名

   すべての定義が完了すると，プロジェクトが新規に作成され，プロジェクトマスタに登録されます。

次に，プロジェクトの作成で使用するダイアログボックスについて説明します。

(1)　プロジェクトの作成ダイアログボックス

プロジェクトの作成ダイアログボックスでは，プロジェクトの名称，およびプロジェクトで使用するフォルダについて定義します。



プロジェクト名

作成するプロジェクトの名称を指定します。

プロジェクト名には，Windows のファイル名の規則に従った名称を指定してください。ただし，「@」を含む名称は指定できません。

また，一つのプロジェクトマスタ中には，同名のプロジェクトを複数作成できません。プロジェクトの名称は，プロジェクトマスタ内で一意になるように定義してください。

プロジェクトフォルダ

プロジェクトで使用する資源を格納するためのフォルダです。資源とは，COBOL プログラムの作成に必要なソースファイル，定義ファイルなどのことです。

プロジェクトフォルダを相対パス名で指定した場合には，プロジェクトマスタフォルダからの相対指定となります。

なお，プロジェクトフォルダに存在しないフォルダを指定した場合，新規にフォルダを作成するかを確認するダイアログボックスが表示されます。このダイアログボックスで［はい］ボタンを選ぶと，指定した名称でプロジェクトフォルダが作成されます。

作業フォルダ

プロジェクトで使用する中間ファイルを出力するフォルダを，プロジェクトフォルダからの相対パス名で指定します。中間ファイルとは，コンパイラやその他のツールが出力する，.obj（オブジェクトファイル），.cbp（プログラム情報ファイル），.tdi（TD コマンド格納ファイル）などの情報ファイルのことです。

開発マネージャでプロジェクトに対してコンパイラやツールを起動する場合，カレントフォルダがそのプロジェクトの作業フォルダに変更されてから起動します。

なお，作業フォルダに存在しないフォルダを指定した場合，新規にフォルダを作成するかを確認するダイアログボックスが表示されます。このダイアログボックスで［はい］ボタンを選ぶと，指定した名称で作業フォルダが作成されます。

作業フォルダは，プロジェクトの作成後にも変更できます。変更方法については，「1.5.6　プロジェクトの設定」を参照してください。

プロジェクトフォルダと作業フォルダを同一にする

このチェックボックスをオンにすると，作業フォルダの欄に入力できなくなり，作業フォルダにはプロジェクトフォルダが仮定されます。

また，このチェックボックスの状態は，次回のプロジェクト作成時に引き継がれます。

(2)　プロジェクト作成ウィザード

プロジェクト作成ウィザードは，次に示す四つのダイアログボックスから構成されています。

(a)　最終生成物の種類を指定するダイアログボックス

(b)　プロジェクトの種類を指定するダイアログボックス

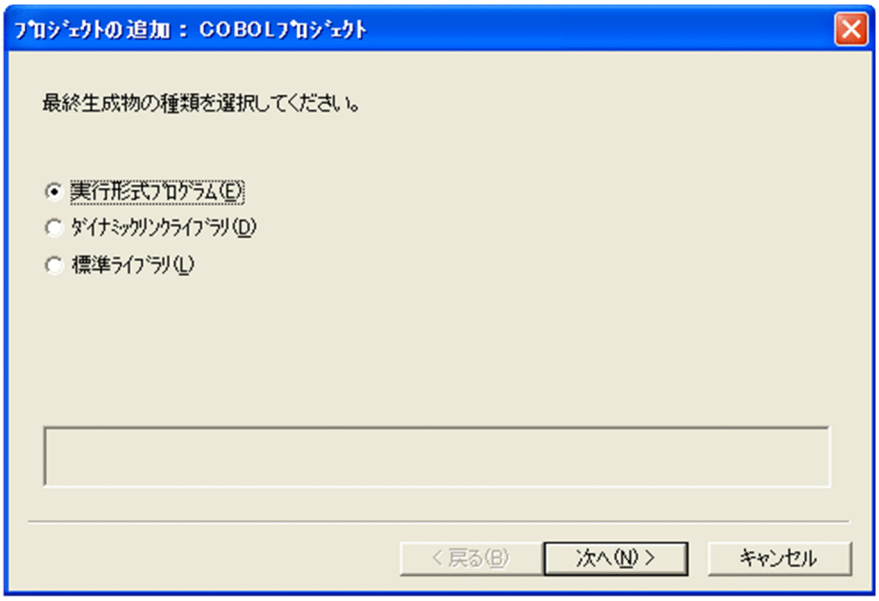
(c)　出力ファイル名を指定するダイアログボックス

(d)　メインファイル名を指定するダイアログボックス

次に，各ダイアログボックスについて説明します。

(a)　最終生成物の種類を指定するダイアログボックス

このダイアログボックスでは，プロジェクトから生成されるプログラムの種類を指定します。実行形式プログラム，ダイナミックリンクライブラリ，標準ライブラリのどれかを選びます。



実行形式プログラム

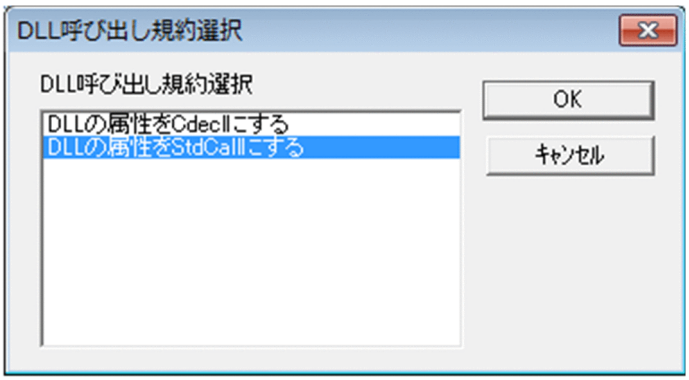
実行可能ファイル（.exe）を作成したい場合，指定します。

ダイナミックリンクライブラリ

ダイナミックリンクライブラリ（.dll）を作成したい場合，指定します。

Windows(x86) COBOL2002 の場合，プロジェクト作成時に，デフォルトオプションに-Dll,Stdcallか-Dll,Cdecl を選択するダイアログボックスが表示されます。-Dll,Stdcall か-Dll,Cdecl のどちらかを選択してください。DLL についてはマニュアル「COBOL2002　ユーザーズガイド」のダイナミックリンク機能についての説明を参照してください。

ダイナミックリンクライブラリの属性の選択



標準ライブラリ

標準ライブラリ（.lib）を作成したい場合，指定します。標準ライブラリは，プロジェクトに共通のサブルーチンを一つのライブラリにまとめてリンクするために使用します。

注意事項

次のコンパイラオプションを指定した場合，必要なライブラリは，開発マネージャが自動で指定します。

コンパイラオプション	必要なライブラリ
-XMAP,LinePrint※	x3klib32.lib
-CompatiM7※	jcm7lib.lib
-OpenTP1	libmcf.lib

注※

Windows(x86) COBOL2002 で有効です。

ただし，ユーザの操作によって，すでにライブラリがプロジェクトに指定されている場合は指定されません。

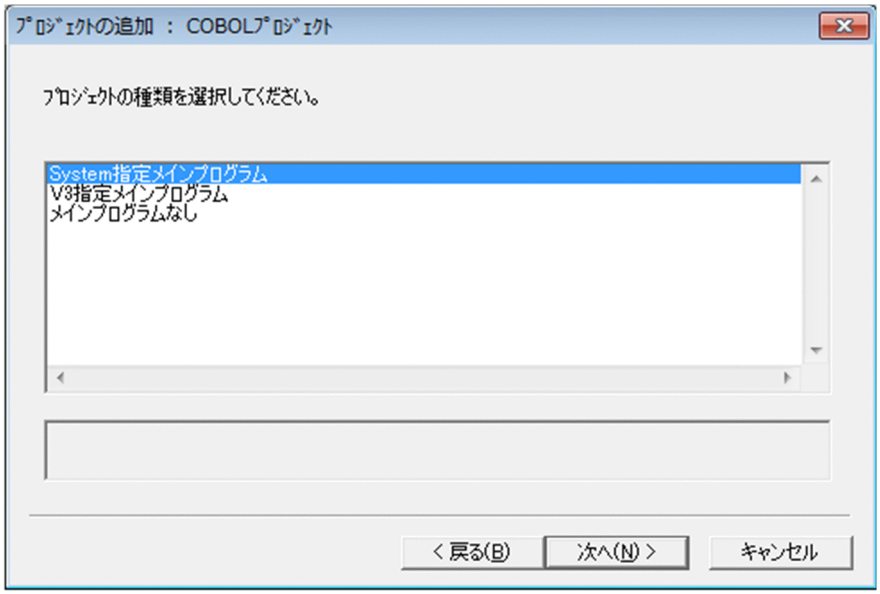
また，［ユーザ設定］タブに上記の表で示したコンパイラオプションを指定した場合は，ユーザの操作によってプロジェクトのライブラリ指定で上記の表で示した必要なライブラリを指定する必要があります。

Windows(x64) COBOL2002 の場合，プロジェクト作成ウィザードに相違があります。相違については，「18.2.5　作成できるプログラムの相違」を参照してください。

(b)　プロジェクトの種類を指定するダイアログボックス

このダイアログボックスでは，プロジェクトの種類を指定します。選択できるプロジェクトの種類は，(a)で指定した最終生成物の種類によって異なります。標準ライブラリを指定した場合は，このダイアログボックスは表示されません。

実行形式プログラムで選択できるプロジェクトの種類



System 指定メインプログラム

この項目を指定すると，メインプログラムが制御プログラムから受け取る引数をシステム固有の argc，argv の形式に合わせます。

V3 指定メインプログラム

この項目を指定すると，メインプログラムが制御プログラムから受け取る引数をメインフレーム（VOS3）の形式に合わせます。

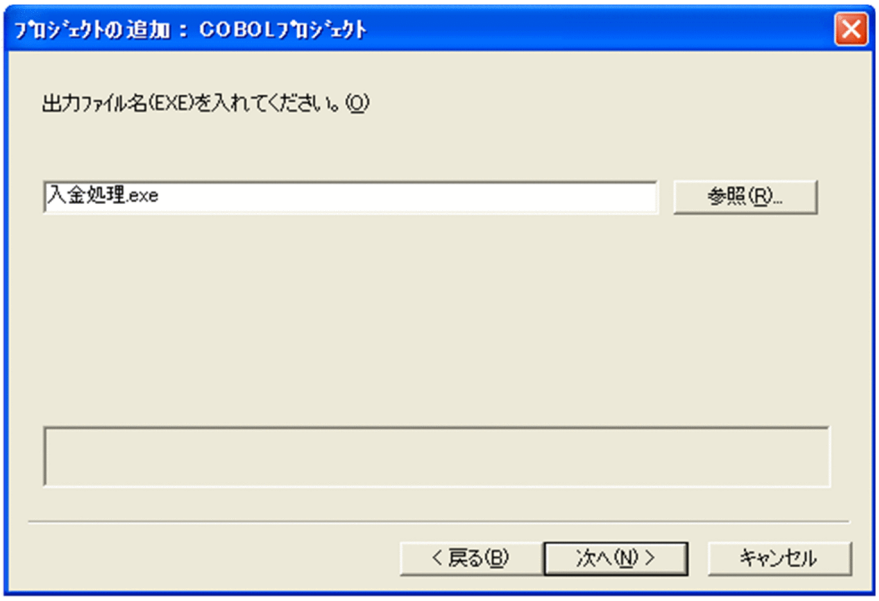
メインプログラムなし

この項目を指定すると，プロジェクトを構成する COBOL ソース中にはメインプログラムが含まれなくなります。メインプログラムのオブジェクトファイルまたはライブラリをプロジェクトに追加する必要があります。

Windows(x64) COBOL2002 の場合，プロジェクト作成ウィザードに相違があります。相違については，「18.2.5　作成できるプログラムの相違」を参照してください。

(c)　出力ファイル名を指定するダイアログボックス

このダイアログボックスでは，プロジェクトから生成されるファイルの名称を指定します。



出力ファイル名

プロジェクトから生成されるファイルの名称を指定します。

ファイル名は，絶対パス名，またはプロジェクトの作業フォルダからの相対パス名で指定します。

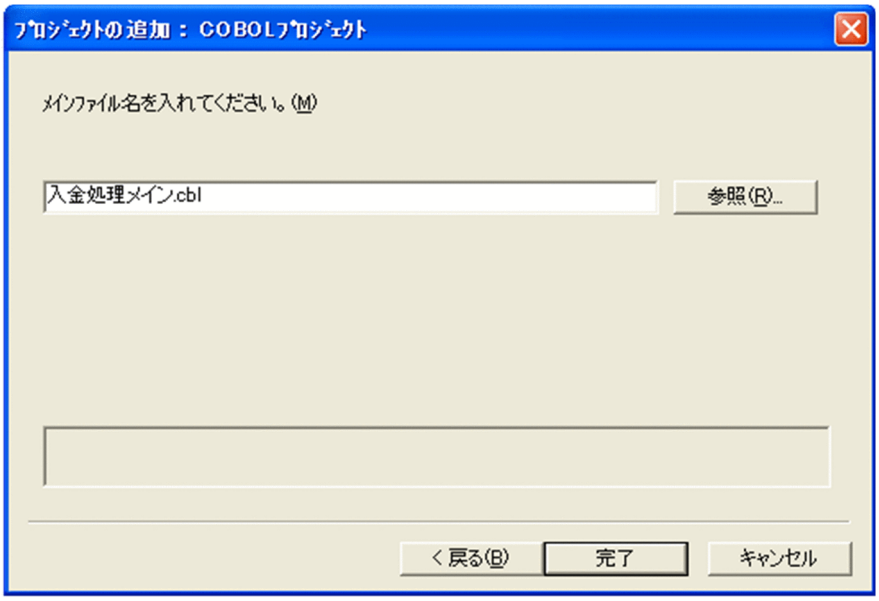
拡張子を指定しなかった場合，選択した最終生成物の種類によって「.exe」，「.dll」，または「.lib」が仮定されます。

出力ファイル名に何も指定しなかった場合，「noname.exe」，「noname.dll」，または「noname.lib」という名称が仮定されます。

出力ファイル名は，プロジェクトの作成後にも変更できます。変更方法については，「1.5.6　プロジェクトの設定」を参照してください。

(d)　メインファイル名を指定するダイアログボックス

このダイアログボックスでは，プロジェクトのメインとなるソースファイルを指定します。標準ライブラリを指定した場合は，このダイアログボックスは表示されません。



メインファイル名

プロジェクトのメインとなるソースファイルの名称を指定します。

ファイル名は，絶対パス名，またはプロジェクトフォルダからの相対パス名で指定します。

拡張子を指定しなかった場合，ソースファイルの種類を問い合わせるダイアログボックスが表示されます。

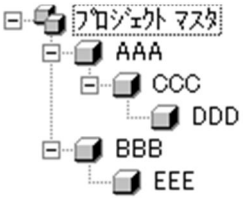
メインファイル名に何も指定しなかった場合，プロジェクトに最初に追加したソースファイルがメインファイルに仮定されます。プロジェクトにファイルを登録する手順については，「1.6.2　ファイルの登録と除外」を参照してください。

1.5.2　プロジェクト間の関係設定

プロジェクトマスタに複数のプロジェクトを登録した場合，プロジェクト間に階層的な関係を設定できます。

プロジェクト間の関係の例を，次に示します。

図 1-4　プロジェクト間の関係の例



あるプロジェクトの上位にあるプロジェクトを，そのプロジェクトの親プロジェクトといいます。反対に，あるプロジェクトの下位にあるプロジェクトを，そのプロジェクトの子プロジェクトといいます。「図 1-4　プロジェクト間の関係の例」の場合，プロジェクト AAA の子プロジェクトは，プロジェクト CCC およびプロジェクト DDD となります。また，プロジェクト EEE の親プロジェクトは，プロジェクト BBB となります。

また，親プロジェクトを持たないプロジェクト（プロジェクトマスタの直下に位置するプロジェクト）のことを，トップレベルプロジェクトといいます。「図 1-4　プロジェクト間の関係の例」の場合，プロジェクト AAA，およびプロジェクト BBB がトップレベルプロジェクトとなります。

トップレベルプロジェクトのビルドを実行すると，その下位のプロジェクトのビルドが必要かどうかが自動的にチェックされ，必要なプロジェクトは自動的にビルドされます。これによって，関連するプロジェクトをまとめてビルドできます。プロジェクトの関係とビルドの実行順序については，「2.6.1　ビルド」の「(2)　ビルドの実行順序」を参照してください。

プロジェクトの関係を設定するには，プロジェクトの関係設定ダイアログボックスを使用します。プロジェクトの関係設定ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトの関係設定］を選ぶ。
   プロジェクトの関係設定ダイアログボックスが表示されます。



(1)　プロジェクト間の関係を追加する

1. プロジェクトの関係ダイアログボックスのプロジェクトの関係ツリービューで，親プロジェクトとなるプロジェクトを選び，［登録］ボタンを選ぶ。
   子プロジェクトの登録ダイアログボックスが表示されます。このダイアログボックスには，選択したプロジェクトの子プロジェクトとして登録できるプロジェクトの一覧が表示されます。

2. 子プロジェクトに登録したいプロジェクトを選び，［OK］ボタンを選ぶ。
   プロジェクト間の親子関係が追加されます。

(2) プロジェクト間の関係を削除する

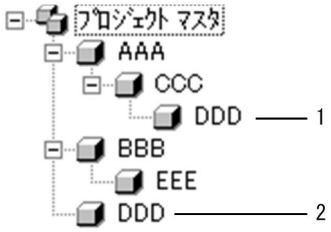
1. プロジェクトの関係ダイアログボックスのプロジェクトの関係ツリービューで，削除したいプロジェクトを選び，［削除］ボタンを選ぶ。

   プロジェクト間の関係が削除されます。

(3) プロジェクト間の関係設定での規則

- 一つの親プロジェクトに，複数の子プロジェクトを設定できます。また，一つの子プロジェクトに，複数の親プロジェクトを設定できます。
- プロジェクトの直接的な親に当たるプロジェクトは，子プロジェクトに設定できません。

  （例）



  1.のプロジェクト DDD には，直接的な親プロジェクトに当たるプロジェクト CCC，およびプロジェクト AAA を子プロジェクトに設定できません。

  2.のプロジェクト DDD には，プロジェクト CCC，およびプロジェクト AAA を子プロジェクトに設定できます。

- 子プロジェクトを持つプロジェクトを削除した場合，子プロジェクトの位置が 1 階層上に移動します。
- プロジェクトを削除した結果，同じレベルに同名のプロジェクトが複数ある場合は，プロジェクト間の関係を削除できません。
- プロジェクトマスタに登録されているプロジェクトは，プロジェクトの関係ツリービュー上に最低一つある必要があります。
- プロジェクトの関係ツリービュー上に一つしかないプロジェクトを削除した場合，そのプロジェクトはトップレベルプロジェクトに移動します。プロジェクトの関係ツリービュー上に一つしかないプロジェクトは，トップレベルプロジェクトから削除できません。

1.5.3 作業中プロジェクトの設定

作業中プロジェクトとは，プロジェクトマスタに登録されているプロジェクトの中で，プログラムの生成および実行操作の対象とするプロジェクトのことです。作業中プロジェクトに指定されたプロジェクトは，開発マネージャのウィンドウの［ビルド］メニューから選択できる次のコマンド，および［プロジェクト］メニューから選択できるカバレージの対象となります。

- ビルド

- リビルド
- クイックビルド
- 依存関係更新
- デバッガ
- 実行
- 実行支援

上記のコマンドを実行した場合，作業中プロジェクト，およびその下位のプロジェクトが処理の対象となります。ビルド処理の詳細は，「2.6.1　ビルド」を参照してください。

作業中プロジェクトに設定されているプロジェクトは，開発マネージャのツリービューウィンドウで，プロジェクト名が太字で表示されます。

作業中プロジェクトは，プロジェクトマスタに登録されているプロジェクトのうち，トップレベルプロジェクトの一つを選択して設定します。作業中プロジェクトの設定方法を次に示します。

1. **開発マネージャのウィンドウの［プロジェクト］メニューから［作業中プロジェクトの設定］を選ぶ。**
   作業中プロジェクト設定ダイアログボックスが表示されます。

2. **作業中プロジェクトに設定したいプロジェクトを指定し，［OK］ボタンを選ぶ。**
   選択したプロジェクトが作業中プロジェクトに設定されます。

（補足事項）
開発マネージャのメニューバーから実行する［ビルド］［実行］などの特定プロジェクトを対象とするコマンドは，作業中プロジェクトに対して実行されます。作業中プロジェクト以外のプロジェクトを対象にしたい場合は，ツリービューウィンドウで対象にしたいプロジェクトを選択し，マウスの右ボタンを押して表示されるメニューから「選択中のプロジェクトのビルド」を選択してください。

1.5.4　プロジェクトの定義内容の確認

プロジェクトに定義したプロジェクトフォルダ，入力ファイル名，出力ファイル名などの情報は，プロジェクトのプロパティダイアログボックスで確認できます。プロジェクトのプロパティダイアログボックスでは，プロジェクトにコメントを設定できます。プロジェクトのプロパティダイアログボックスの表示手順を次に示します。

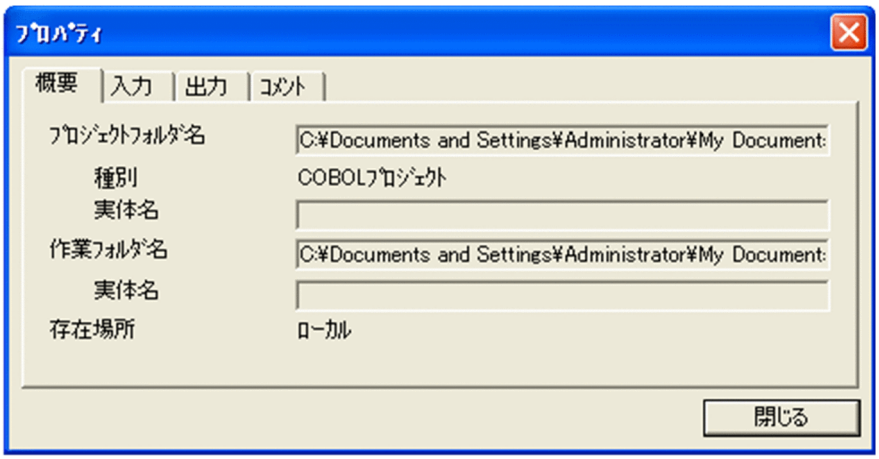
1. **開発マネージャのツリービューウィンドウで，定義内容を確認したいプロジェクトを選ぶ。**

2. **［プロジェクト］メニューから［プロパティ］を選ぶ。**
   プロジェクトのプロパティダイアログボックスが表示され，ツリービューウィンドウで選んだプロジェクトの定義内容が確認できます。

プロパティダイアログボックスを表示した状態でも，ツリービューウィンドウを操作できます。また，プロパティダイアログボックスを表示した状態でツリービューウィンドウからほかのプロジェクトやソースファイルを選ぶと，プロパティダイアログボックスの表示内容は選んだアイテムのものに切り替わります。

次に，プロジェクトのプロパティダイアログボックスで確認できる内容について，各タブを選ぶと表示されるページごとに説明します。

(1) 概要



プロジェクトフォルダ名

プロジェクトフォルダの絶対パス名が表示されます。

種別

プロジェクトの種別が表示されます。

- COBOL プロジェクト

  Windows にあるファイルだけでプロジェクトが構成されている開発環境です。

実体名

何も表示されません。

作業フォルダ名

プロジェクトの作業フォルダの絶対パス名が表示されます。

実体名

何も表示されません。

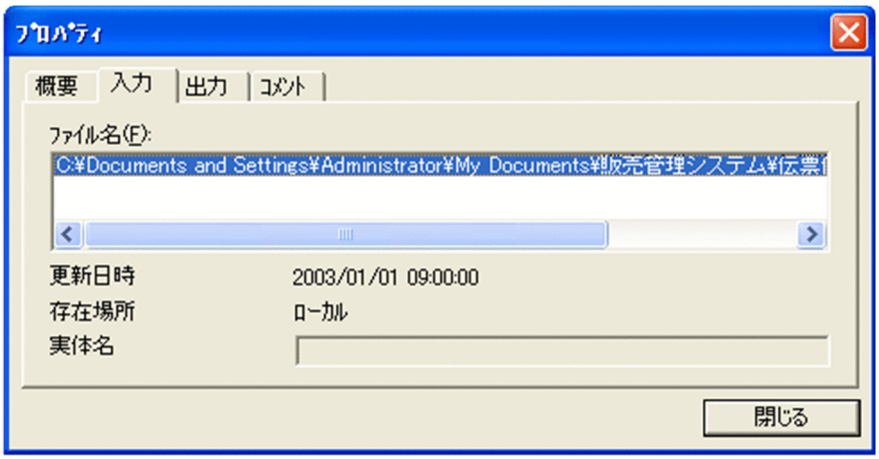
存在場所

プロジェクトのある場所が表示されます。

- ローカル

  ローカル（Windows）上にプロジェクトがあります。

(2)　入力



ファイル名

選択中のプロジェクトの入力ファイル（オブジェクトファイル，ライブラリファイルなど）が表示されます。

ファイルを起動するツールが関連づけられている場合，ファイル名をダブルクリックしてファイルを開くことができます。ファイルの種類とツールの関連づけについては，「1.6.3　ファイルを開く」を参照してください。

更新日時

選択中のプロジェクトの入力ファイルを最後に更新した日時が表示されます。

存在場所

選択中のプロジェクトの入力ファイルのある場所が表示されます。

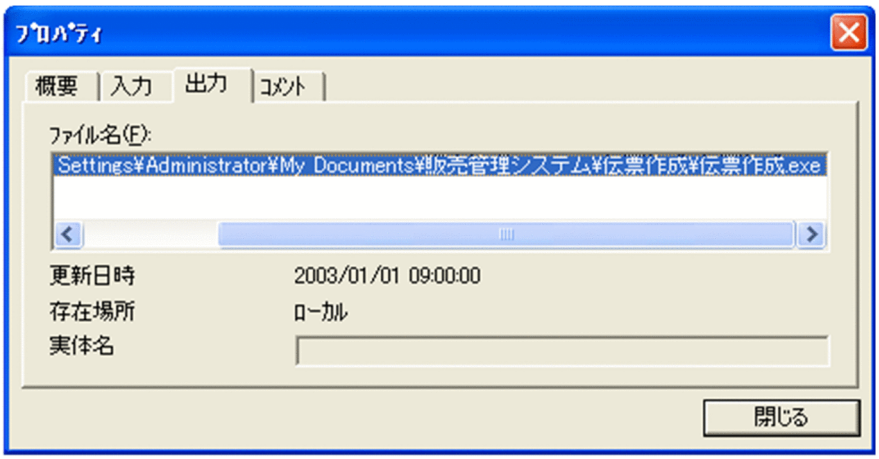
- ローカル

  ローカル（Windows）上にファイルがあります。

実体名

何も表示されません。

(3)　出力



ファイル名

選択中のプロジェクトの出力ファイル（最終生成物）が表示されます。

ファイルを起動するツールが関連づけられている場合，ファイル名をダブルクリックしてファイルを開くことができます。ファイルの種類とツールの関連づけについては，「1.6.3　ファイルを開く」を参照してください。

更新日時

選択中のプロジェクトの出力ファイルを最後に更新した日時が表示されます。

存在場所

選択中のプロジェクトの出力ファイルのある場所が表示されます。

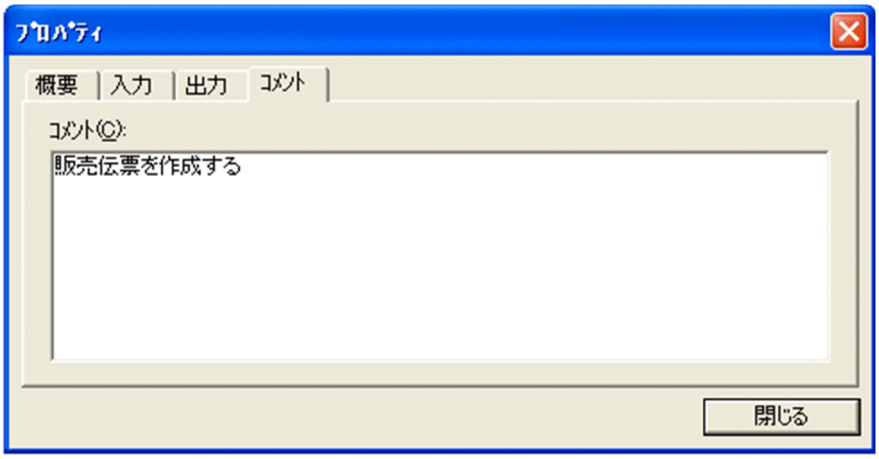
- ローカル

  ローカル（Windows）上にファイルがあります。

実体名

何も表示されません。

(4)　コメント



コメント

プロジェクトに対する説明，注釈などのコメントを設定します。［コメント］タブからほかのタブにフォーカスを移動させた場合，または［閉じる］ボタンを選んだ場合，入力したコメントが設定されます。コメントは，動作オプション設定ダイアログボックスで，「ツリービューウィンドウに，コメントを表示する」が設定されている場合だけ，ツリービューウィンドウに表示されます。動作オプションの設定方法については，「1.7.1　動作オプションの設定」を参照してください。

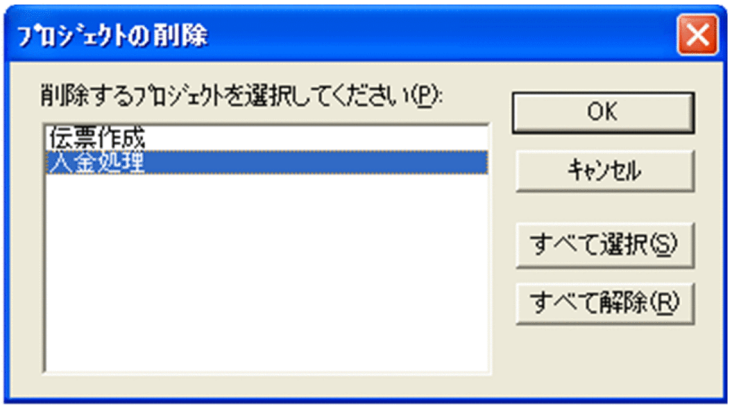
コメントは，128 バイトまでツリービューウィンドウに表示されます。また，128 バイト未満で改行を入力した場合は，改行までの文字が表示されます。

1.5.5　プロジェクトの削除

プロジェクトマスタからプロジェクトを削除する方法を，次に示します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトの削除］を選ぶ。

   プロジェクトの削除ダイアログボックスが表示されます。



2. 削除するプロジェクトを選び，［OK］ボタンを選ぶ。

   選択したプロジェクトが削除されます。

注意事項

- プロジェクトマスタにプロジェクトが一つしか登録されていない場合，プロジェクトは削除できません。プロジェクトマスタには，最低一つのプロジェクトが登録されている必要があります。
- トップレベルプロジェクトが一つしかない場合は，トップレベルプロジェクトを削除できません。
- 作業中プロジェクトを削除した場合，残ったトップレベルプロジェクトの中から作業中プロジェクトが仮定されます。必要な場合は，作業中プロジェクトを再設定してください。

  プロジェクトを削除すると，プロジェクト間の関係も変更されます。プロジェクト間の関係の設定規則については，「1.5.2　プロジェクト間の関係設定」の「(3)　プロジェクト間の関係設定での規則」を参照してください。なお，この規則に反するようなプロジェクトの削除はできません。例えば，プロジェクトを削除することによって，同じ階層に同名のプロジェクトが複数ある場合は，プロジェクトを削除できません。

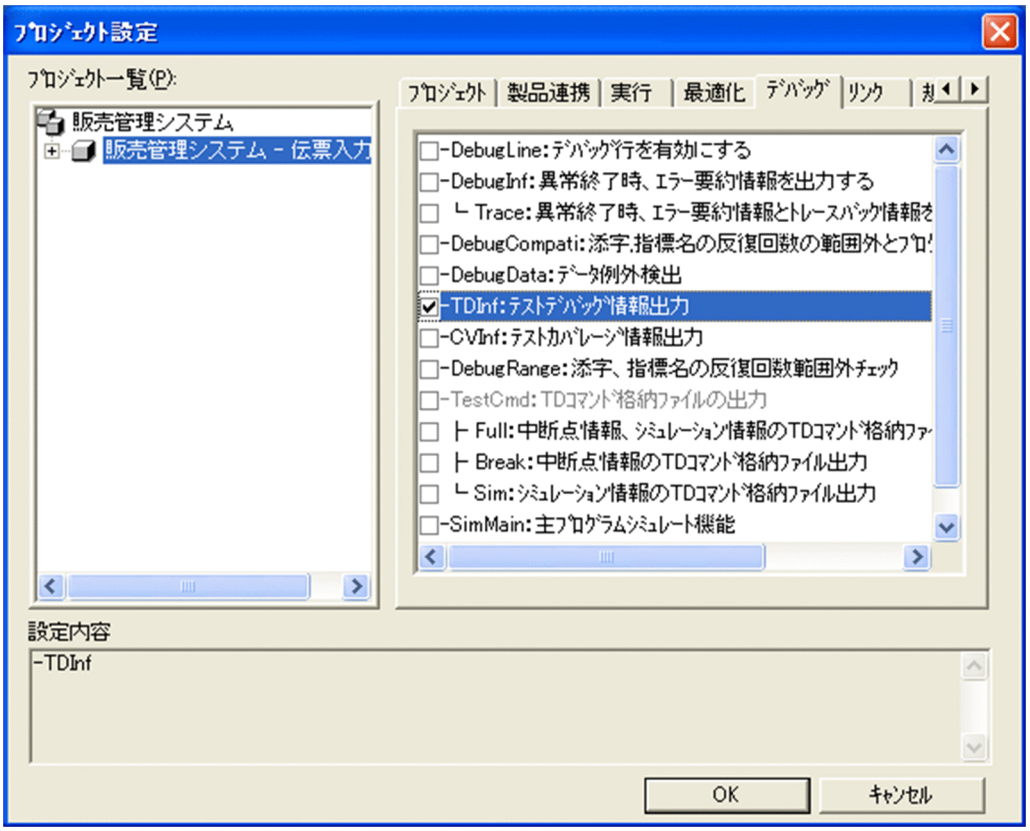
  このような場合は，プロジェクト間の関係を削除してから，プロジェクトを削除してください。

- プロジェクトを削除しても，プロジェクトフォルダやオブジェクトファイルなどの，プロジェクトが開発に使用したファイルやフォルダは削除されません。

1.5.6　プロジェクトの設定

開発マネージャでは，プロジェクトごとにコンパイラオプション，環境変数などを設定できます。また，プロジェクトの作成時に指定した作業フォルダや出力ファイル名を変更できます。

プロジェクトの設定には，プロジェクト設定ダイアログボックスを使用します。プロジェクト設定ダイアログボックスの表示方法を次に示します。



1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトの設定］を選ぶ。

   プロジェクト設定ダイアログボックスが表示されます。

2. プロジェクト設定ダイアログボックスのプロジェクト一覧ツリービューで，動作条件を変更したいプロジェクトを選ぶ。

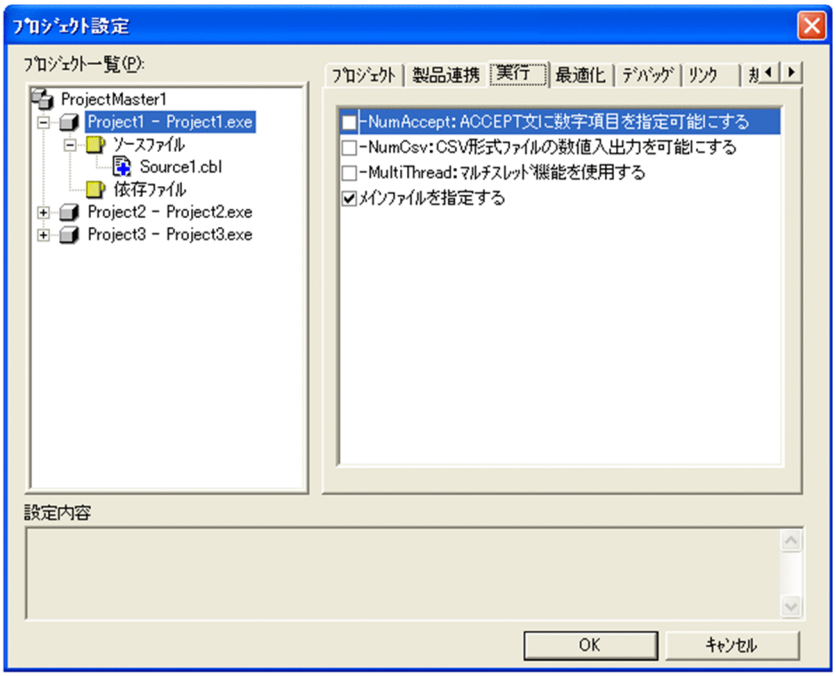
   プロジェクト設定ダイアログボックスの右側に，現在設定されているプロジェクトの設定が表示されます。必要に応じて，設定を変更してください。

3. ［OK］ボタンを選ぶ。

   ダイアログボックスで設定した内容がプロジェクトに反映されます。

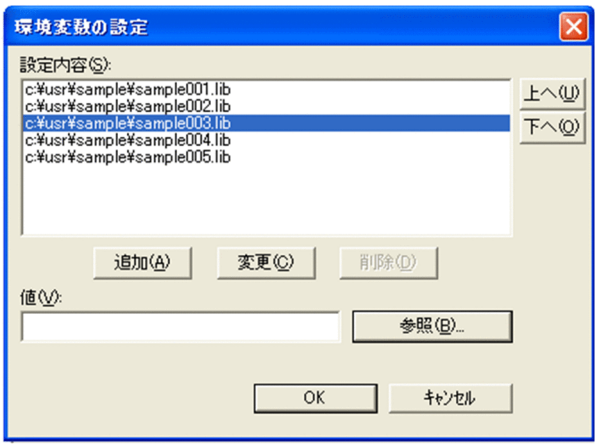
ファイル単位のオプション指定によるファイルアイコンの切り替わり

オプションはファイルごとに指定することもできます。ファイルごとに指定すると，ファイルに対応するアイコンの表示が変わります。オプションを指定したあとで［OK］ボタンを選ぶと，ソースファイルに対応するファイルアイコンの表示が変わります。



オプションの引数および環境変数の値を複数指定する場合

複数の値を設定できるオプションの引数および環境変数には，プロジェクト設定ダイアログボックスで複数の値を指定できます。コンパイル時に，コンパイラに渡す値の順序を変更するときは，［上へ］［下へ］ボタンで変更します。



(1)　コンパイラオプションの設定

コンパイラオプションの設定方法については，「2.2　コンパイラオプションの設定」を参照してください。

また，コンパイラオプションの説明については「2.3　コンパイラオプション」を参照してください。

(2)　コンパイラ環境変数の設定

コンパイラ環境変数の設定方法，および説明については，「2.4　環境変数の設定」を参照してください。

(3)　リンカオプションの設定

リンカオプションの設定方法，および説明については，「2.5　リンカオプションの設定」を参照してください。

(4)　プロジェクトの作業フォルダの設定

プロジェクトの作業フォルダを設定する手順を次に示します。

1. プロジェクト設定ダイアログボックスで，［リンク］タブを選ぶ。

2. 作業フォルダの欄に作業フォルダのパス名を入力する。
   絶対パス，またはプロジェクトフォルダからの相対パスで指定します。
   また，［参照］ボタンで表示されるダイアログボックスからフォルダを選んでも，設定できます。

(5)　出力ファイル名の設定

プロジェクトの出力ファイル（実行可能ファイル，DLL，または標準ライブラリ）の名称を設定する手順を次に示します。

1. プロジェクト設定ダイアログボックスで，［リンク］タブを選ぶ。

2. 最終生成物名称の欄に出力ファイル名を入力する。
   絶対パス名，またはプロジェクトの作業フォルダからの相対パス名で指定します。ファイル名だけを指定した場合，出力ファイルはプロジェクトの作業フォルダに作成されます。
   また，［参照］ボタンで表示されるダイアログボックスからフォルダを選んでも，設定できます。

1.5.7　プロジェクト情報の出力と取り込み

開発マネージャでは，プロジェクトマスタ中に定義されているプロジェクトの情報を，プロジェクト情報ファイル（.hmp）に出力できます。また，プロジェクト情報ファイルからプロジェクトマスタに，プロジェクトの情報を取り込めます。この機能を使うと，プロジェクトマスタ間でプロジェクトを複写して利用できます。

(1)　プロジェクト情報の出力（エクスポート）

プロジェクトマスタからプロジェクト情報を出力する手順を次に示します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトのエクスポート］を選ぶ。

プロジェクトのエクスポート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には，現在編集中のプロジェクトマスタに含まれるプロジェクトの一覧が表示されます。



2. 情報を出力するプロジェクトを選んで，［OK］ボタンを選ぶ。

   プロジェクトの情報が「プロジェクト名.hmp」という名称のプロジェクト情報ファイルに出力されます。

規則

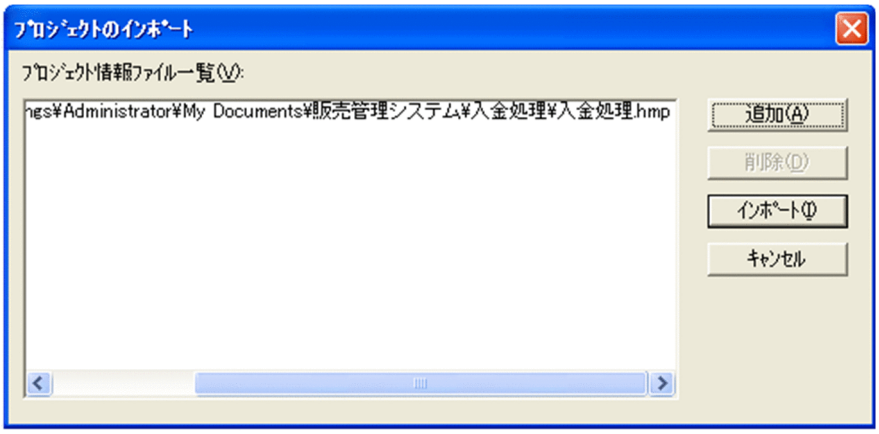
- プロジェクト情報ファイルは，プロジェクトフォルダに作成されます。
- 出力しようとしたプロジェクト情報ファイルがすでにプロジェクトフォルダにある場合，上書きしてもよいか確認するメッセージが表示されます。

(2)　プロジェクト情報の取り込み（インポート）

プロジェクトマスタにプロジェクト情報を取り込む手順を次に示します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトのインポート］を選ぶ。

   プロジェクトのインポートダイアログボックスが表示されます。



2. 次のどちらかの方法で，「プロジェクト情報ファイル一覧」にファイル名を追加する。

   - ［追加］ボタンを選ぶと表示されるダイアログボックスで，取り込むプロジェクト情報ファイルを選ぶ。

- エクスプローラからプロジェクトのインポートダイアログボックスへ，プロジェクト情報ファイル（.hmp）をドラッグ＆ドロップする。

追加したプロジェクト情報ファイル名が「プロジェクト情報ファイル一覧」に表示されます。

なお，「プロジェクト情報ファイル一覧」に表示されているプロジェクト情報ファイル名を指定して［削除］ボタンを選ぶと，ファイルを一覧から除外できます。

3. 2.の操作を繰り返して，取り込みたいプロジェクト情報ファイルをすべて「プロジェクト情報ファイル一覧」に登録する。

4. ［インポート］ボタンを選ぶ。

プロジェクトの情報が，現在編集中のプロジェクトマスタに取り込まれます。

規則

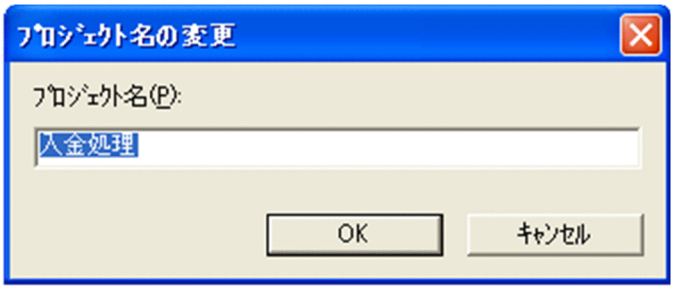
- 取り込もうとするプロジェクトの情報と同名のプロジェクトが現在編集中のプロジェクトマスタにある場合，プロジェクトの情報は取り込めません。
- プロジェクトの情報を取り込んだ場合，プロジェクト情報ファイルがあるフォルダがそのプロジェクトのプロジェクトフォルダになります。

1.5.8　プロジェクト名の変更

開発マネージャでは，プロジェクトの名称を変更できます。プロジェクトの名称を変更する手順を次に示します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクト名の変更］を選ぶ。

プロジェクト名の変更ダイアログボックスが表示されます。



2. 「プロジェクト名」に変更後のプロジェクト名を設定して，［OK］ボタンを選ぶ。

作業中プロジェクトのプロジェクト名が変更されます。

プロジェクト間に設定された関係は，プロジェクトの名称を変更したあとも有効です。

選んだプロジェクトの名称を変更する方法

1. 開発マネージャのツリービューウィンドウで，名前を変更したいトップレベルプロジェクトを右クリックし，表示されるメニューから［プロジェクト名の変更］を選ぶ。

プロジェクト名の変更ダイアログボックスが表示されます。「プロジェクト名」に変更後のプロジェクト名を設定して，［OK］ボタンを選ぶと，プロジェクト名が変更されます。

1.6　ファイルの登録と管理

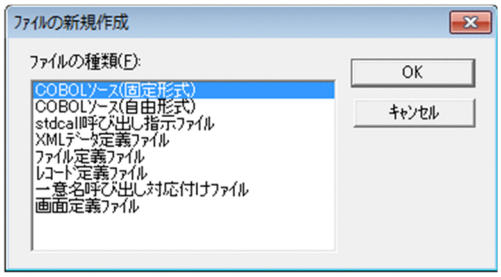
ここでは，開発マネージャのプロジェクトにファイルを登録する方法，および登録したファイルを操作する方法について説明します。

1.6.1　ファイルの新規作成

COBOL プログラムの開発に使用するファイルを，開発マネージャから新規に作成できます。ファイルを新規に作成する手順を次に示します。

1. 開発マネージャのウィンドウの［ファイル］メニューから［新規作成］を選ぶ。

   ファイルの新規作成ダイアログボックスが表示されます。



2. 作成するファイルの種類を選んで，［OK］ボタンを選ぶ。

   指定したファイルの種類に関連づけられたツールが起動し，新規にファイルを作成できる状態になります。

注意事項

新規に作成したファイルは，そのままではプロジェクトに登録されません。登録したい場合は，作成したファイルを一度保存してから，「1.6.2　ファイルの登録と除外」の手順に従ってファイルをプロジェクトに登録してください。

1.6.2　ファイルの登録と除外

ここでは，COBOL プログラムの作成に必要なファイルをプロジェクトに登録する方法，および登録したファイルをプロジェクトから除外する方法について説明します。

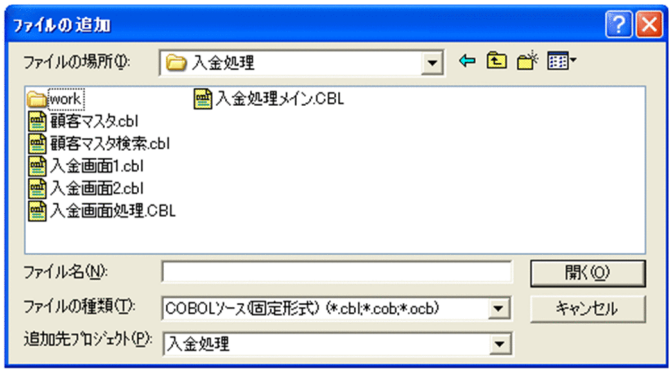
(1)　ソースファイル，リンケージ用ファイルの登録と除外

(a)　ファイルの登録

プロジェクトにソースファイル，およびリンケージ用ファイルを登録する方法について説明します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［ソースファイルの追加］を選ぶ。

ファイルの追加ダイアログボックスが表示されます。



2. 追加先プロジェクトドロップダウンリストで，ファイルを登録したいプロジェクトを指定する。

3. ファイルの種類ドロップダウンリストで，登録したいファイルの種類を指定する。

4. ファイル名を指定して［開く］ボタンを選ぶ。

   指定したファイルがプロジェクトに登録されます。

   登録したファイルは，開発マネージャのツリービューウィンドウで，プロジェクトのソースファイルノード，またはリンケージ用ファイルノードの下位に追加されます。

   実際にはないソースファイルも，存在を仮定してプロジェクトに登録できます。ファイル名の欄に仮定するファイル名を直接入力して［開く］ボタンを選んでください。存在を仮定してプロジェクトに登録するかを確認するダイアログボックスが表示されるので，［はい］ボタンを選んでプロジェクトに登録してください。

なお，次の定義ファイルを登録した場合，ソースファイル名に続けて出力先となる COBOL ソースファイル名を登録する必要があります。

- ファイル定義ファイル
- レコード定義ファイル
- 画面定義ファイル
- HTML ファイル
- XML データ定義ファイル

出力ファイルを登録するダイアログボックスが表示されるので，出力先となるソースファイル名を指定してください。

なお，XML データ定義ファイルを追加する場合の操作方法の詳細については，マニュアル「COBOL2002 XML 連携機能ガイド」を参照してください。

(b)　ファイルの除外

登録されているファイルをプロジェクトから除外する方法を，次に示します。

1. 開発マネージャのツリービューウィンドウで，除外したいファイルを選ぶ。

2. [ファイル] メニューの [削除] を選ぶ。

ファイルがプロジェクトから除外されます。

注意事項

- メインソースファイルをプロジェクトから除外した場合，残ったソースファイルのどれかを新たにメインソースファイルに指定する必要があります。メインソースファイルは，プロジェクト設定ダイアログボックスの [実行] タブを選び，表示されるページから「メインファイルを指定する」オプションを選んで設定します。
- 出力先を持つソースファイルをプロジェクトから除外した場合，それらに関連づけていた出力先のソースファイルも除外されます。
- プロジェクトからファイルを除外しても，ファイルの実体は削除されません。

(2) 依存ファイルの登録と除外

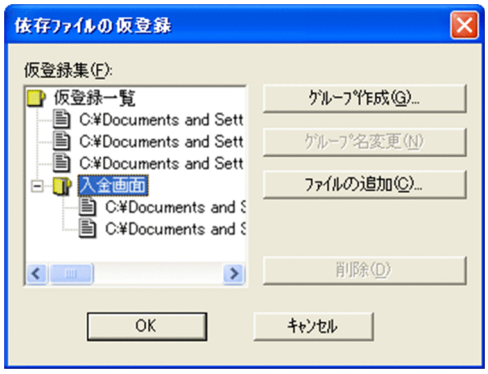
依存ファイルを登録する場合，まずプロジェクトマスタに仮登録し，それからソースファイルとの依存関係を設定します。

(a) 依存ファイルの仮登録

プロジェクトマスタで使用する依存ファイルを仮登録します。依存ファイルの仮登録には，依存ファイルの仮登録ダイアログボックスを使用します。依存ファイルの仮登録ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのウィンドウの [プロジェクト] メニューから [依存ファイルの仮登録] を選ぶ。

依存ファイルの仮登録ダイアログボックスが表示されます。



依存ファイルを仮登録する

[ファイルの追加] ボタンを選び，表示されるダイアログボックスでファイルを選択して仮登録します。

依存ファイルは，異なるグループに重複して登録できます。ただし，同じグループ内には，同じファイルを複数登録できません。

仮登録集にグループを作成する

仮登録集には，グループを作成できます。グループを作成すると，複数の依存ファイルの依存関係をまとめて設定できます。

グループを作成するには，［グループ作成］ボタンを選び，表示されるダイアログボックスにグループ名を指定します。グループ名は Windows のファイル名の規則に従った，40 バイト以内の文字列で指定する必要があります。

グループにファイルを追加する場合は，仮登録集ツリービューでグループを選んでから，［ファイルの追加］ボタンを選んでください。

また，すでに作成したグループ名を変更できます。グループ名を変更する場合は，仮登録集ツリービューでグループを選び，［グループ名変更］ボタンを選びます。グループ名が編集できる状態になるので，新しいグループ名を指定して［Enter］キーを押してください。

同じ名称のグループが複数あってはなりません。また，グループの下位にグループは作成できません。

仮登録集から除外する

仮登録からファイルを除外するには，仮登録集ツリービューから除外したいファイル，またはグループを選び，［削除］ボタンを選びます。

なお，グループを除外した場合は，グループに登録していた依存ファイルもすべて仮登録集から除外されます。

最上位の「仮登録集」グループは，除外できません。

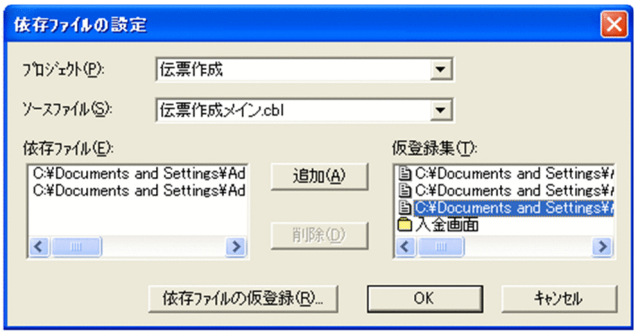
(b)　依存関係の設定

仮登録集に登録した依存ファイルを，ソースファイルと関連づけます。関連づけは，依存ファイルの設定ダイアログボックスで設定します。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［依存ファイルの設定］を選ぶ。

   依存ファイルの設定ダイアログボックスが表示されます。

また，ツリービューウィンドウで，依存ファイルを関連づけたいソースファイルやプロジェクトを右クリックし，表示されるメニューから［依存ファイルの設定］を選んでも，ダイアログボックスを表示できます。なお，ビルド対象外のファイルを選択した場合は，［依存ファイルの設定］メニューは選択できなくなります。



ソースファイルに依存ファイルを関連づける

1. 「プロジェクト」および「ソースファイル」で依存ファイルを関連づけたいソースファイルを選ぶ。

プロジェクト中の複数のソースファイルに依存ファイルを関連づけたい場合は，「ソースファイル」で「(選択したファイル)」を選び，表示されるダイアログボックスでファイルを選んでください。なお，ビルド対象外のファイルは関連づけられません。

プロジェクト中のすべてのソースファイルに依存ファイルを関連づけたい場合は，「ソースファイル」で「(すべてのファイル)」を選んでください。なお，ビルド対象外のファイルは関連づけられません。

2. 「仮登録集」から関連づける依存ファイルを選ぶ。

   依存ファイルは複数選択できます。グループを選んだ場合，グループに含まれるすべての依存ファイルを選択したことになります。

   仮登録集に関連づけたい依存ファイルがない場合，［依存ファイルの仮登録］ボタンを選んで，依存ファイルを仮登録してください。

3. ［追加］ボタンを選ぶ。

   2.で選択した依存ファイルが「依存ファイル」に追加されます。

4. ［OK］ボタンを選ぶ。

   ソースファイルと依存ファイルが関連づけられます。

ソースファイルから依存ファイルの関連づけを削除する

1. 「プロジェクト」および「ソースファイル」から関連づけを削除したいソースファイルを選ぶ。

   プロジェクト中の複数のソースファイルから関連づけを削除したい場合は，「ソースファイル」で「(選択したファイル)」を選び，表示されるダイアログボックスでファイルを選んでください。

   プロジェクト中のすべてのソースファイルから関連づけを削除したい場合は，「ソースファイル」で「(すべてのファイル)」を選んでください。

2. 「依存ファイル」から削除したい依存ファイルを選び，［削除］ボタンを選ぶ。

   選択した依存ファイルが「依存ファイル」から削除されます。

3. ［OK］ボタンを選ぶ。

   関連づけが削除されます。

また，ツリービューウィンドウでも依存ファイルを指定して削除できます。

1. 開発マネージャのツリービューウィンドウで，削除したい依存ファイル名を選ぶ。
2. ［ファイル］メニューから［削除］を選ぶ。

   選んだ依存ファイルがツリービュー上から削除されます。また，削除した依存ファイルとソースファイルとの関連づけも削除されます。

登録集原文を登録する場合の注意事項

登録集原文に登録集原文を関連づける指定は開発マネージャではできません。このような指定をする場合は，最終的に登録集原文を取り込む COBOL ソースに対して，すべての登録集原文を依存ファイルとして関連づけてください。

（例）

```
MAIN.CBL              COBOLソースファイル
└ COPY01.CBL          登録集原文
  └ COPY02.CBL        登録集原文
```

上の図のような関係の場合，登録集原文「COPY01.CBL」「COPY02.CBL」の両方をソースファイル「MAIN.CBL」の依存ファイルに指定してください。

補足事項

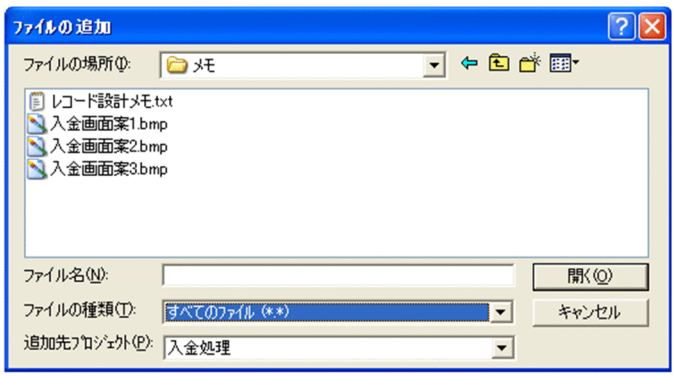
- ソースファイルに関連づけた依存ファイルは，ツリービューウィンドウの依存ファイルノードの下に表示されます。
- ソースファイルに関連づけた依存ファイルは，ファイルのプロパティダイアログボックスで確認できます。詳細は，「1.6.4　ファイルの情報の確認」を参照してください。
- 依存ファイルの設定ダイアログボックスでは，仮登録からのファイルの除外はできません。仮登録の除外は，依存ファイルの仮登録ダイアログボックスから実行してください。
- 関連づける依存ファイルと関連づけられるソースファイルに，同一のファイルは指定できません。
- COBOL ソースの依存ファイルとして登録されているファイル定義ファイル，またはレコード定義ファイルは，コンパイル，またはビルドによって，依存から削除されます。

(3)　その他のファイルの登録と除外

(a)　ファイルの登録

プロジェクトには，COBOL プログラムの開発に必要なファイル以外のファイルも登録できます。プロジェクトに関連するデータファイルや文書ファイルなどを登録しておくと便利です。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［ソースファイルの追加］を選ぶ。
   ファイルの追加ダイアログボックスが表示されます。



2. 追加先プロジェクトドロップダウンリストで，ファイルを登録したいプロジェクトを指定する。
3. ファイルの種類ドロップダウンリストで，「すべてのファイル（*.*)」を指定する。
4. ファイル名を指定して［開く］ボタンを選ぶ。
   指定したファイルがプロジェクトに登録されます。

登録したファイルは，開発マネージャのツリービューウィンドウで，プロジェクトのその他ファイルノードの下位に登録されます。

上記の方法で，実体のないファイルも一般ファイルとしてプロジェクトに登録できます。ただし，実体のないファイルは，開発マネージャから開けません。

(b)　ファイルの除外

ファイルの除外方法については，ソースファイル，およびリンケージ用ファイルの場合と同じです。「(1) ソースファイル，リンケージ用ファイルの登録と除外」の「(b)　ファイルの除外」を参照してください。

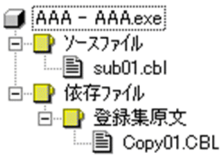
(4)　ファイルの登録と除外の規則

1. ファイルの重複について

- 追加するファイルと同一名称のファイルが，プロジェクトにすでに登録されている場合は，ファイルを登録できません。

（例 1）

次に示すプロジェクトに，「Copy01.cbl」を COBOL ソースとして登録できません（すでに登録集原文として登録されているため）。



- ソースファイルから生成される情報ファイル（オブジェクトファイルなど）の名称が重複してしまうようなファイルの組み合わせを，一つのプロジェクトの中に登録できません。

  例えば，次のようなファイルの組み合わせは登録できません。

  ・異なるフォルダにある同名の COBOL ソース

  ・ファイル名の基本部分が同一の COBOL 固定形式ソースと自由形式ソース（a.cob と a.cbf など）

（例 2）

（例 1）のプロジェクトに，別フォルダの COBOL ソース「sub01.cbl」は登録できません。また，COBOL ソース「sub01.cbf」も，コンパイルの結果出力される情報ファイルの名称が「sub01.cbl」から出力される情報ファイルの名称と同一になるので，登録できません。

- 追加するファイルから生成されるファイルと同一のファイルがプロジェクトにある場合は，ファイルを登録できません。

2. stdcall 呼び出し指示ファイルの登録について

- プロジェクト中の COBOL ソースファイル全体に有効な stdcall 呼び出し指示ファイルを指定する場合は，プロジェクトの設定で-StdCall および-StdCallFile コンパイラオプションを設定してください。この場合，指定した stdcall 呼び出し指示ファイルは，プロジェクトの「依存ファイル」－「stdcall 呼び出し指示ファイル」ノードに表示されます。

- プロジェクト中の特定の COBOL ソースファイルに対して stdcall 呼び出し指示ファイルを指定する場合は，ファイルの設定で-StdCall および-StdCallFile コンパイラオプションを設定してください。この場合，指定した stdcall 呼び出し指示ファイルは，プロジェクトの「依存ファイル」－「stdcall 呼び出し指示ファイル」－「ローカル」ノードに表示されます。
- 一つの COBOL ソースファイルに対して，stdcall 呼び出し指示ファイルは複数登録できません。また，プロジェクトに対して，stdcall 呼び出し指示ファイルは複数登録できません。
- stdcall 呼び出し指示ファイルの指定を解除する場合は，オプションダイアログボックスの-StdCall コンパイラオプションのチェックをオフにしてください。

3. HTML ファイルの登録について
   - HTML ファイルはプロジェクトの最終生成物の種類が「ダイナミックリンクライブラリ」の場合だけ開発マネージャに登録できます。
   - HTML から生成される COBOL ソースをメインプログラムとすることはできません。
   - HTML ファイルから生成される COBOL ソースは COBOL 固定形式ソースとなります。
   - HTML ファイルの出力先ファイルは，ファイルの登録後にも変更できます。変更手順は，定義ファイルの場合と同じです。

     HTML ファイルの出力先ファイルを変更した場合，元の出力先ファイルに指定されていたコンパイラオプション，環境変数は初期化されます。
4. アイコンファイルの登録について

   プロジェクトに対して，アイコンファイルは複数登録できません。
5. 環境変数 CBLFIX，CBLFREE の設定について

   開発マネージャのプロジェクト設定ダイアログボックスから環境変数 CBLFIX，または CBLFREE を設定した場合，次の点に注意してください。
   - 環境変数で設定した種類のファイルをプロジェクトに登録しようとすると，ファイルの種類を問い合わせるダイアログボックスが表示されます。環境変数 CBLFIX に設定した種類のファイルの場合「COBOL ソース（固定形式)」を，環境変数 CBLFREE に設定した種類のファイルの場合「COBOL ソース（自由形式)」を，それぞれ選択してください。

   なお，システムの環境変数に CBLFIX，または CBLFREE を設定しておくと，環境変数の設定に従ってファイルの種類が自動的に選択され，プロジェクトに登録されます。

1.6.3 ファイルを開く

開発マネージャに登録されているファイルは，関連づけられた編集ツールで編集できます。ファイルと編集ツールの関連づけは，次のどれかの方法で定義します。

1. 開発マネージャでの関連づけ

開発マネージャでは，あらかじめ幾つかのファイルと編集ツールの関連づけが定義されています。例えば，画面定義ファイル（.wdf）は画面定義に，ファイル定義ファイル（.flf）はファイル／レコード定義に，それぞれ関連づけられています。

2. ファイルの種類の登録による関連づけ

   標準では編集ツールが定義されていないファイルについては，ユーザがツールを指定できます。詳細は，「1.7.4　ファイルの種類の登録」を参照してください。

3. Windows での関連づけ

   Windows では，ファイルの種類ごとに編集に使用するツールを定義できます。上記 1.，2.のどちらにも編集ツールが定義されていないファイルの場合は，Windows で定義された編集ツールと関連づけられます。

   なお，Windows でのファイルの関連づけについては，Windows のマニュアルを参照してください。

1.～3.のどれかの方法で関連づけを指定していないファイルの種類は，開発マネージャから編集できません。また，3.の方法でファイルの種類が登録されている場合，ファイルをプロジェクトに仮定して登録したとき（ファイルの実体がないとき），開発マネージャから編集できません。

次に，開発マネージャからファイルを開く方法について説明します。

(1)　ツリービューウィンドウに表示されているファイルを開く

ツリービューウィンドウに表示されているファイルは，次のどれかの方法で開きます。

(a)　メニューによる方法

1. 開発マネージャのツリービューウィンドウで，開きたいファイルを選ぶ。
2. ［ファイル］メニューから［開く］を選ぶ。

   ツールが起動され，指定したファイルが開かれます。

(b)　マウスのダブルクリックによる方法

1. 開発マネージャのツリービューウィンドウで，開きたいファイルをダブルクリックする。

   ツールが起動され，指定したファイルが開かれます。

(c)　マウスの右クリックメニューによる方法

1. 開発マネージャのツリービューウィンドウで，開きたいファイルを右クリックして，表示されるメニューから［開く］を選ぶ。

   ツールが起動され，指定したファイルが開かれます。

(2) 出力先ファイルを開く

ファイル定義ファイル，画面定義ファイルなどの定義ファイルから出力された COBOL ソースファイルを直接開いて編集できます。編集方法を次に示します。

1. 開発マネージャのツリービューウィンドウで定義ファイルを選び，［ファイル］メニューから［出力ファイルの編集］を選ぶ。
   ツールが起動され，選択した定義ファイルの出力ファイルが開かれます。

(3) プロパティダイアログボックスからファイルを開く

プロジェクトのプロパティダイアログボックス，またはファイルのプロパティダイアログボックスから，出力ファイル，依存ファイルなどに指定されているファイルを開きます。ここでは例として，ファイルのプロパティダイアログボックスから，ファイルに関連づけられた登録集原文を開く方法を示します。

1. 開発マネージャのツリービューウィンドウで，登録集原文が関連づけられたソースファイルを選ぶ。

2. ［ファイル］メニューから［プロパティ］を選ぶ。
   ファイルのプロパティダイアログボックスが表示されます。

3. ［依存］タブを選ぶ。
   依存タブのページが表示されます。ページのファイル名の欄には，ソースファイルに関連づけられている登録集原文のファイル名が表示されます。

4. 登録集原文のファイル名をダブルクリックする。
   COBOL エディタが起動し，登録集原文ファイルが開かれます。

1.6.4 ファイルの情報の確認

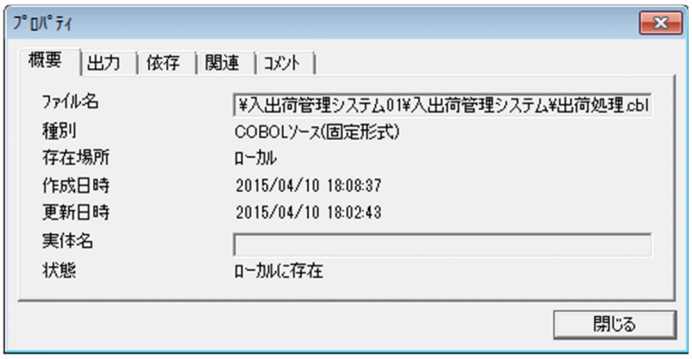
ファイルの更新日時や存在場所，設定したコメントなどの情報は，ファイルのプロパティダイアログボックスで確認できます。ファイルのプロパティダイアログボックスの表示手順を次に示します。

1. 開発マネージャのツリービューウィンドウで，情報を確認したいファイルを選ぶ。

2. ［ファイル］メニューから［プロパティ］を選ぶ。
   ファイルのプロパティダイアログボックスが表示されます。

次に，ファイルのプロパティダイアログボックスで確認できる内容について，各タブを選ぶと表示されるページごとに説明します。なお，［出力］および［依存］タブのページは，ビルド対象外のファイルの場合，表示されません。

(1) 概要



ファイル名

ファイルの絶対パス名が表示されます。

種別

ファイルの種別が表示されます。

存在場所

ファイルの実体のある場所が表示されます。

- ローカル

  ローカル（Windows）上にファイルがあります。

作成日時

ファイルを作成した日時が表示されます。

更新日時

ファイルを最後に更新した日時が表示されます。

実体名

何も表示されません。

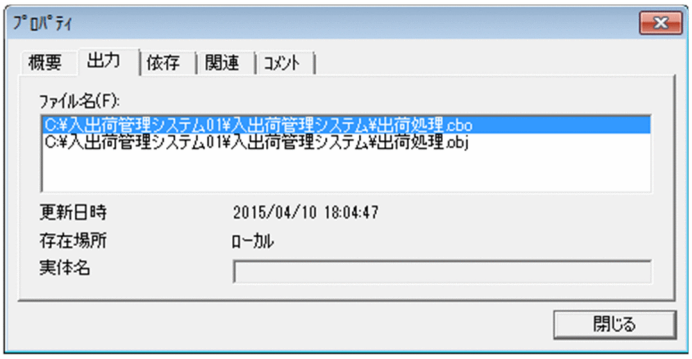
状態

現在のファイルの状態が表示されます。

- ローカルに存在

  ファイルは，ローカル（Windows）に格納されているか，ローカルで編集されています。

(2) 出力



ファイル名

選択中のファイルから生成されるファイル（オブジェクトファイルなど）が表示されます。

ファイルを起動するツールが関連づけられている場合，ファイル名をダブルクリックしてファイルを開くことができます。ファイルの種類とツールの関連づけについては，「1.6.3　ファイルを開く」を参照してください。

更新日時

出力ファイルを最後に更新した日時が表示されます。

存在場所

出力ファイルの実体のある場所が表示されます。

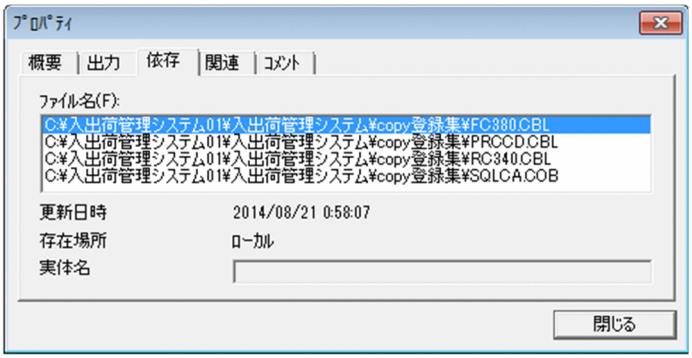
- ローカル

  ローカル（Windows）上にファイルがあります。

実体名

何も表示されません。

(3) 依存



ファイル名

選択中のファイルに依存しているファイル（登録集原文など）が表示されます。

ファイルを起動するツールが関連づけられている場合，ファイル名をダブルクリックしてファイルを開くことができます。ファイルの種類とツールの関連づけについては，「1.6.3　ファイルを開く」を参照してください。

更新日時

依存ファイルを最後に更新した日時が表示されます。

存在場所

依存ファイルの実体のある場所が表示されます。

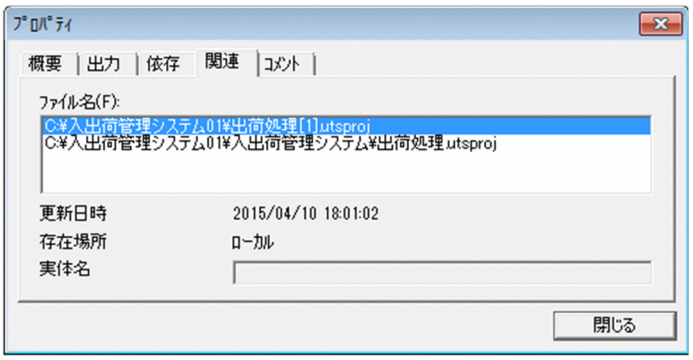
- ローカル

  ローカル（Windows）上にファイルがあります。

実体名

何も表示されません。

(4)　関連



ファイル名

選択中のファイルに関連しているファイルが表示されます。COBOL2002 Developer Professionalの単体テスト支援で，該当するソースファイルでテストプロジェクトを作成している場合に，テストプロジェクトファイル名が表示されます。

ファイルを起動するツールが関連づけられている場合，ファイル名をダブルクリックしてファイルを開くことができます。ファイルの種類とツールの関連づけについては，「1.6.3　ファイルを開く」を参照してください。

更新日時

関連ファイルを最後に更新した日時が表示されます。

存在場所

関連ファイルの実体のある場所が表示されます。

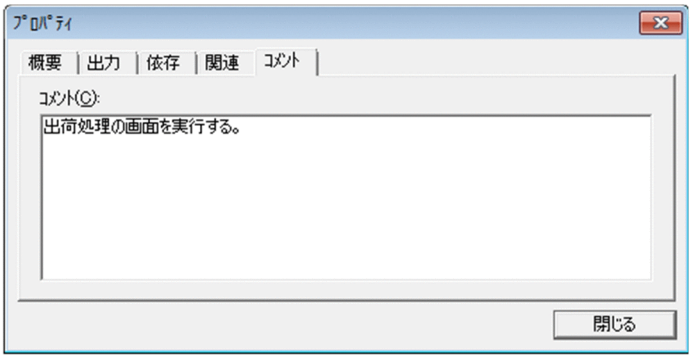
- ローカル

  ローカル（Windows）上にファイルがあります。

実体名

何も表示されません。

(5)　コメント



コメント

ファイルに対する説明，注釈などのコメントを設定します。

［コメント］タブからほかのタブにフォーカスを移動させた場合，または［閉じる］ボタンを選んだ場合，入力したコメントが設定されます。コメントは，動作オプション設定ダイアログボックスで，「ツリービューウィンドウに，コメントを表示する」が設定されている場合だけ，ツリービューウィンドウに表示されます。動作オプションの設定方法については，「1.7.1　動作オプションの設定」を参照してください。

コメントは，128 バイトまでツリービューウィンドウに表示されます。また，128 バイト未満で改行を入力した場合は，改行までの文字が表示されます。

1.6.5　ファイルの設定

開発マネージャでは，ソースファイル単位にコンパイラオプション，環境変数などを設定できます。

ソースファイル単位で指定する場合は，プロジェクト設定ダイアログボックスのプロジェクト一覧ツリービューでソースファイルを選んで設定します。

プロジェクト一覧ツリービューで複数のファイルを選択すると，同時にオプションを設定できます。このとき，オプションのチェックボックスの表示は次のようになります。

- チェックあり

  選択しているファイルのすべてでオプションが指定されていて，かつ引数とサブオプションが一致していることを示します。チェックありの場合だけ，プロジェクト設定ダイアログボックスの設定内容にオプションが表示されます。

- 網掛けのチェック

  次のどれか，またはすべてであることを示します。

  ・選択しているファイルの一部でオプションが指定されている

  ・オプションの引数がファイルごとに異なっている

  ・サブオプションが異なっている

- チェックなし

  選択しているファイルのすべてでオプションが指定されていないことを示します。

上記以外については，プロジェクト単位での設定と同じです。詳細は，「1.5.6　プロジェクトの設定」を参照してください。

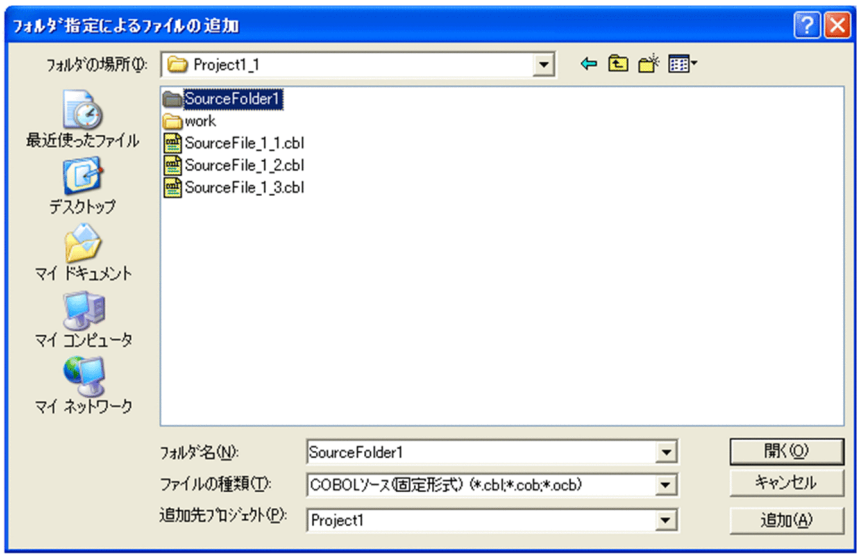
なお，画面定義ファイルやファイル／レコード定義ファイルなど，ファイルの設定ができないソースファイルもあります。

1.6.6　フォルダ指定によるファイルの追加

フォルダを指定してファイルを追加する手順を次に示します。

1. **開発マネージャのウィンドウの［プロジェクト］メニューから［ソースファイルの追加］－［フォルダ指定によるファイルの追加］を選ぶ。**

   フォルダ指定によるファイルの追加ダイアログボックスが表示されます。



2. **追加するファイルが格納されているフォルダ，追加するファイルの種類および追加先プロジェクトを指定し，［追加］ボタンを選ぶ。**

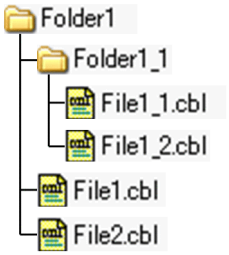
   指定したフォルダ下にある，該当する種類のファイルがすべて追加されます。

   フォルダ名テキストボックスが空のときは，「フォルダの場所」ドロップダウンリストで表示しているフォルダ下の該当するファイルが追加の対象になります。

   フォルダ名テキストボックスにフォルダ名が設定してあるときは，選択したフォルダ下の該当するファイルが追加の対象になります。

   (例)

   次に示すフォルダ構成のときの動作を説明します。



ダイアログボックスのリスト一覧に Folder1 を表示していて，フォルダ名テキストボックスが空のときは，ファイルの種類で COBOL ソースを指定し［追加］ボタンを選択すると，File1.cbl と File2.cbl が追加されます。

ダイアログボックスのリスト一覧で Folder1 を表示していて，Folder1_1 を選択し，フォルダ名テキストボックスに Folder1_1 と表示されているときに，ファイルの種類で COBOL ソースを指定し［追加］ボタンを選択すると，File1_1.cbl と File1_2.cbl が追加されます。

1.7　開発マネージャのカスタマイズ

開発マネージャでは，キー操作や画面の色，ファイルの編集に使用するエディタやツール，開発マネージャ自体の動作など，いろいろな機能をユーザが使いやすいように設定できます。ここでは，開発マネージャ自身の動作環境を設定する方法について説明します。

1.7.1　動作オプションの設定

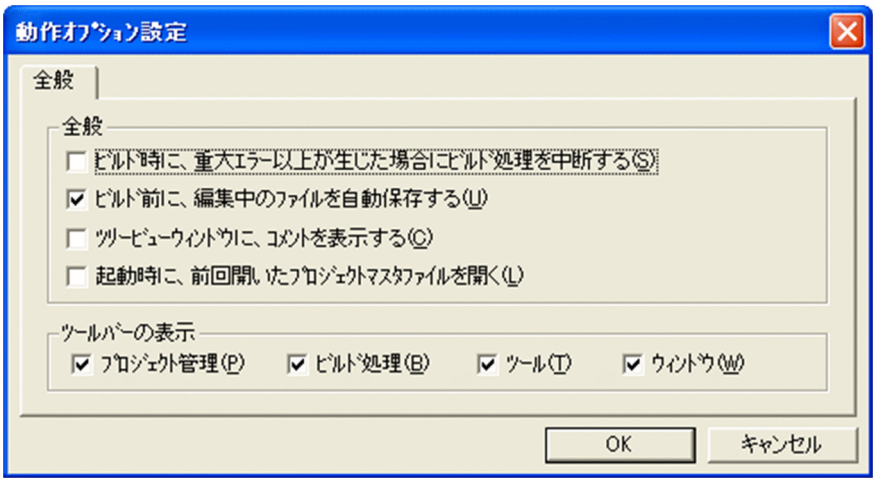
開発マネージャの動作オプションは，動作オプション設定ダイアログで設定します。動作オプションの設定方法を次に示します。

1. 開発マネージャのウィンドウの［ツール］メニューから［開発マネージャの動作オプション］を選ぶ。
   動作オプション設定ダイアログが表示されます。

2. 必要な動作オプションを設定し，［OK］ボタンを選ぶ。
   動作オプションが設定されます。

次に，このダイアログボックスで設定する内容について説明します。

(1)　全般



(a)　全般

ビルド時に，重大エラー以上が生じた場合にビルド処理を中断する

この項目を指定した場合，ビルド，クイックビルド，またはリビルド中に回復不能エラー，および重大エラーが発生したとき，ビルド処理が中断されます。

この項目を指定しない場合，回復不能エラー，および重大エラーが発生してもビルド処理を続行します。

ビルド前に，編集中のファイルを自動保存する

この項目を指定した場合，ビルド，クイックビルド，またはリビルドを実行すると，編集中のファイルをすべて自動保存してからビルド処理を開始します。

この項目を指定しない場合，ビルド，クイックビルド，またはリビルドを実行すると，編集中のファイルを保存するかを確認するダイアログボックスが表示されます。

ツリービューウィンドウに，コメントを表示する

この項目を指定した場合，プロジェクト，ソースファイル，または依存ファイルに設定したコメントが開発マネージャのツリービューウィンドウに表示されます。

この項目を指定しない場合，プロジェクト，ソースファイル，または依存ファイルにコメントが設定されていても，ツリービューウィンドウには表示されません。

起動時に，前回開いたプロジェクトマスタファイルを開く

この項目を指定した場合，開発マネージャを起動すると，自動的に前回終了時に開いていたプロジェクトマスタファイルが開かれます。

(b) ツールバーの表示

開発マネージャのツールバーの表示の有無を指定します。

ツールバーには次の 4 種類があります。表示の有無は，ツールバーごとに指定できます。

- プロジェクト管理
- ビルド処理
- ツール
- ウィンドウ

1.7.2 ユーザインタフェースの設定

開発マネージャでは，ショートカットキーの割り当てと，メッセージウィンドウの表示を変更できます。変更方法を次に示します。

1. 開発マネージャのウィンドウの［ツール］メニューから［開発マネージャのユーザカスタマイズ］の［ユーザインタフェース］を選ぶ。

   カスタマイズダイアログボックスが表示されます。

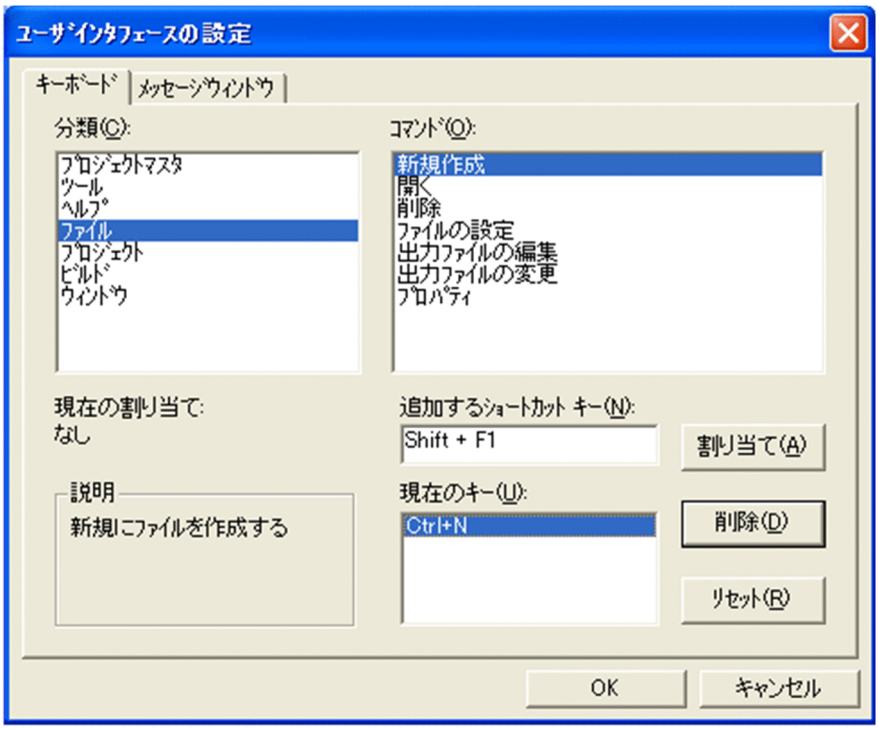
2. 変更したい項目のタブを選び，表示されるページで設定を変更し，［OK］ボタンを選ぶ。

   ダイアログボックスでの設定内容が開発マネージャに反映されます。

次に，各タブを選ぶと表示されるページごとに設定する内容について説明します。

(1)　ショートカットキーの設定

ショートカットキーは，［キーボード］タブを選んで表示されるページで設定します。



分類

コマンドの分類を選びます。

コマンド

キーを割り当てるコマンドを選びます。

現在の割り当て

「追加するショートカットキー」で指定したキーに，現在割り当てられている機能が表示されます。

追加するショートカットキー

ショートカットキーを指定します。

説明

「コマンド」で選んだコマンドの説明が表示されます。

現在のキー

「コマンド」で選んだコマンドに，現在登録されているショートカットキーが表示されます。

(a)　ショートカットキーの登録

1. 「分類」および「コマンド」から，ショートカットキーを登録したいコマンドを指定する。

   指定したコマンドに現在登録されているショートカットキーが「現在のキー」に表示されます。

2.「追加するショートカットキー」にフォーカスを合わせて，割り当てるショートカットキーをキーボードから入力する。

「追加するショートカットキー」に，入力したキーが表示されます。また，「現在の割り当て」には，そのキーに現在割り当てられているコマンドが表示されます。この時点では，まだショートカットキーが登録されていません。

3.［割り当て］ボタンを選ぶ。

ショートカットキーが登録されます。

規則

- 次のキーをショートカットキーとして使用できます。

ショートカットキーとして使用できるキー

- ［0］～［9］，［A］～［Z］，［F1］～［F12］キー
- 上記のキーと，［Shift］，［Ctrl］，または［Alt］キーの組み合わせ

- 一つのキーには，コマンドを一つだけ割り当てられます。すでにコマンドを割り当てているキーに，別のコマンドを割り当てた場合，新しい割り当てが有効となり，古い割り当ては削除されます。

(b) ショートカットキーの削除

1.「分類」および「コマンド」に，ショートカットキーを削除したいコマンドを指定する。

指定したコマンドに現在登録されているショートカットキーが「現在のキー」に表示されます。

2.削除したいショートカットキーを「現在のキー」から選び，［削除］ボタンを選ぶ。

ショートカットキーが削除されます。

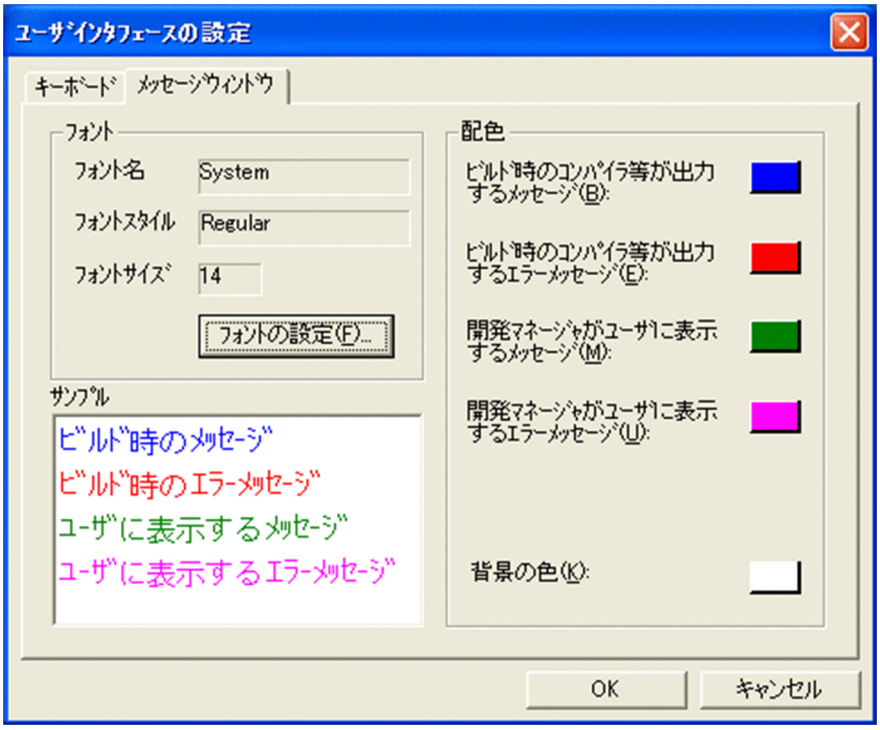
(c) ショートカットキーの初期化

1.［リセット］ボタンを選び，［OK］ボタンを選ぶ。

開発マネージャに登録したすべてのショートカットキーが初期化されます。

(2) メッセージウィンドウの表示の設定

開発マネージャのメッセージウィンドウの表示は，［メッセージウィンドウ］タブを選んで表示されるページで設定します。



フォント

メッセージウィンドウで使用する文字フォントを設定します。

配色

メッセージウィンドウに表示されるメッセージの色を，メッセージ種別ごとに設定します。

サンプル

現在設定されているフォント，配色での表示例が表示されます。

(a) フォントの設定方法

1. フォントの設定ボタンを選ぶ。

   フォントを設定するダイアログボックスが表示されます。

2. フォント名，スタイル，サイズ，および書体の種類を選択して［OK］ボタンを選ぶ。

   フォントが設定されます。

(b) 配色の設定方法

1. 配色の項目名横のボタンを選ぶ。

   色の設定ダイアログボックスが表示されます。

2. 色を選んで［OK］ボタンを選ぶ。

   なお，任意の色を指定するには，色の設定ダイアログボックスの［色の作成］ボタンを選び，拡張されたダイアログボックスで色を作成してください。

1.7.3　テキストエディタの設定

開発マネージャでは，ソースファイルの編集などで使用するテキストエディタを登録・変更できます。開発環境や COBOL ソースファイル，HTML ファイルなどのファイルの種類ごとに，それぞれ異なるエディタを設定できます。

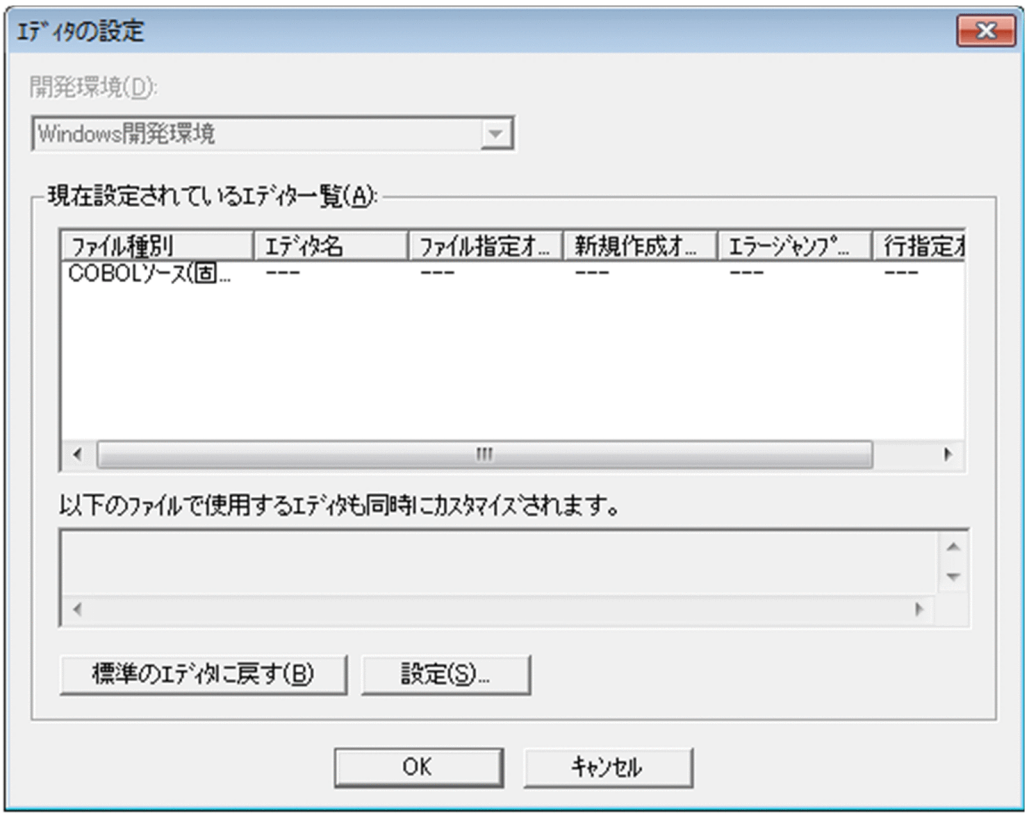
(1)　エディタの設定ダイアログボックスの表示

テキストエディタは，エディタの設定ダイアログボックスで登録します。エディタの設定ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのウィンドウの［ツール］メニューから［開発マネージャのユーザカスタマイズ］の［エディタ］を選ぶ。

   エディタの設定ダイアログボックスが表示されます。

(2)　エディタの設定ダイアログボックスの説明



開発環境

エディタを設定する開発環境が表示されます。

- Windows 開発環境

  Windows だけで COBOL プログラムを作成する開発環境です。

現在設定されているエディター一覧

ファイル種別ごとに登録されているエディタ，およびエディタの起動オプションの一覧が表示されます。

ファイル種別

開発環境で使用するファイルの種類です。

同じエディタを使用するファイルの種類は，一つのファイル種別としてまとめて表示されます。

ファイル種別を選択すると，そのファイル種別に連動してエディタの設定が変更されるファイル種別の一覧が下の欄に表示されます。

エディタ名

ファイル種別ごとに登録されているエディタ名が表示されます。

- COBOL2002 に内蔵されている COBOL エディタが登録されている場合，「---」と表示されます。
- COBOL エディタ以外のエディタが登録されている場合，エディタのプログラム名称が絶対パス名で表示されます。

ファイル指定オプション，新規作成オプション，エラージャンプオプション，行指定オプション

エディタの起動オプションが表示されます。設定が省略されている項目は，「---」と表示されます。

(3) エディタの登録

(a) 外部エディタを登録する場合

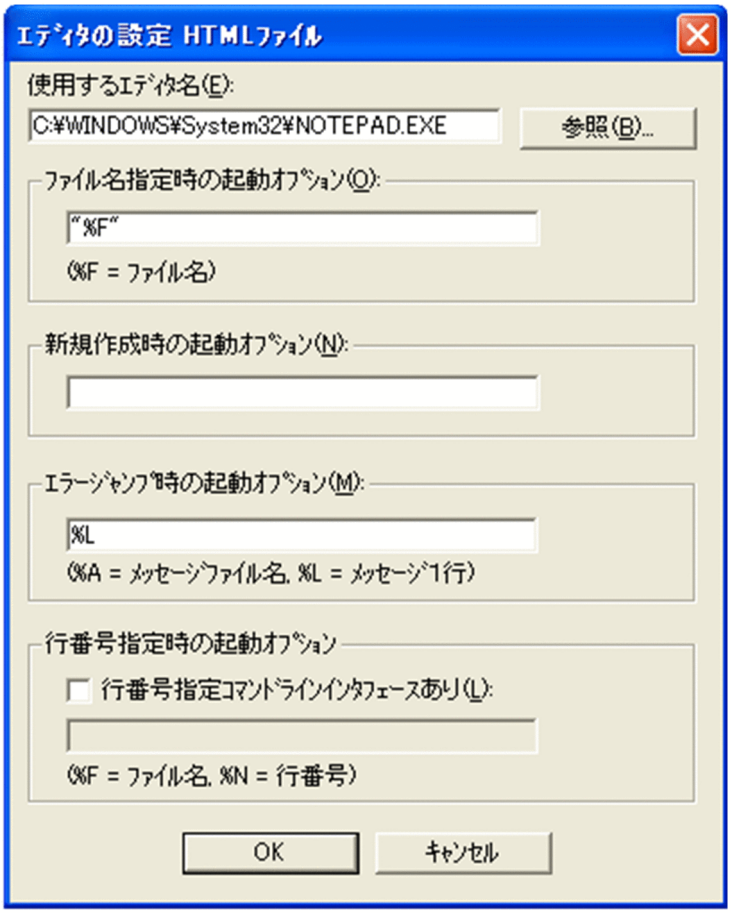
1. エディタの設定ダイアログボックスで，エディタを登録したいファイル種別を選ぶ。

   ファイル種別は複数選択できます。

2. ［設定］ボタンを選ぶ。

   次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に必要な項目を入力し，［OK］ボタンを選ぶと，エディタの設定が登録されます。



使用するエディタ名

テキストエディタのファイル名を絶対パスで指定します。

ファイル名指定時の起動オプション

ファイル名を指定してテキストエディタを起動した場合のオプションを設定します。

"%F"を含む文字列を指定する必要があります。

新規作成時の起動オプション

ファイルを新規に作成するためにテキストエディタを起動した場合のオプションを設定します。

エラージャンプ時の起動オプション

コンパイル時に出力されるエラーメッセージとエディタとの関連づけを設定します。エディタオプションには，"%L"または"%A"のどちらか一方を指定する必要があります。

メッセージウィンドウに表示されたメッセージをダブルクリックすると，ここで設定した方法でエディタが起動されます。エディタの起動方法には，次の 2 通りがあります。

%A（ダブルクリックされたメッセージを一時ファイルに書き出し，その一時ファイル名称をコマンドライン引数として渡す）

メッセージウィンドウ中のエラーメッセージには，エラーの発生したファイル名と行数がタグジャンプ情報の書式で出力されています。この方法では，エラーメッセージを一時タグファイルとして書き出し，そのファイル名をエディタに渡します。

一時タグファイルはダブルクリック 1 回につき 1 個，環境変数 TEMP で示されるフォルダに作成されます。このファイルは，開発マネージャの終了時に削除されます。

この方法で起動する場合には，次の両方の機能を持つエディタを使用してください。

- ファイル名をコマンドライン引数で受け取ってファイルを開く機能
- エディタで開いたタグファイルから，タグジャンプする機能

（例）

エディタに「fsed.exe」，オプションに「/error△%A」を指定した場合のコマンドラインは「fsed.exe /error 一時ファイル名」となります。

△は半角空白文字を表します。

%L（ダブルクリックされたメッセージの 1 行をコマンドライン引数として渡す）

この方法では，メッセージウィンドウでダブルクリックされたメッセージ行を，直接エディタにコマンドライン引数として渡します。

この方法で起動する場合には，次のどちらかの機能を持つエディタを使用してください。

- コンパイラの出力メッセージをコマンドライン引数で受け取って，直接タグジャンプする機能
- コマンドラインからのタグジャンプ機能は持たないが，コマンドラインの先頭作用対象をファイル名称として受け取り，ファイルを開く機能

（例）

エディタに「notepad.exe」，オプションに「%L」を指定した場合のコマンドラインは「notepad.exe ダブルクリックされた 1 行」となります。

行番号指定時の起動オプション

使用するテキストエディタが，行番号指定でファイルをオープンするコマンドラインオプションを提供している場合に，「行番号指定コマンドラインインタフェースあり」を指定し，そのコマンドラインオプションを指定します。オプションには，"%F"を含む文字列を指定する必要があります。

また，指定したコマンドラインは，テストデバッガからエディタを起動するときに使用されます。

（置換文字列の種類）

外部エディタ登録時のオプションに指定できる置換文字列を示します。

%F

編集対象となるファイル名（ドライブ名から始まる絶対パス）に置き換えられます。

%N

カーソルを位置づけたい行番号に置き換えられます。

%A

ダブルクリックされた時点のメッセージウィンドウの内容が一時ファイルに出力され，そのファイル名に置き換えられます。

%L

ダブルクリックされたメッセージウィンドウの 1 行と置き換えられます。

次に，エディタ設定ダイアログボックス上の設定項目に，これらの置換文字列をどのように指定するのか示します。

設定項目	%F	%N	%A	%L
起動時のオプション	◎	×	×	×
行番号指定エディタオプション	◎	△	×	×
メッセージウィンドウをダブルクリックしたときのエディタオプション	×	×	○	○

（凡例）

◎：常に指定する

○：どちらか一方を指定する

△：単独では指定できない

×：指定できない

（オプションの指定で使用する文字列についての注意事項）

オプションの文字列に「%」を使用したい場合は，「%%」と二つ続けて記述してください。

（例）

"%F" … 置換文字列とみなされ，編集対象となるファイル名に置換されます。

"%%F" … 「%F」という文字列とみなされます。

(b)　標準のエディタに設定を戻す場合

1. エディタの設定ダイアログボックスで，エディタの設定を戻したいファイル種別を選ぶ。

   ファイル種別は複数選択できます。

2. [標準のエディタに戻す] ボタンを選ぶ。

   指定したファイル種別に標準のエディタが設定されます。

1.7.4　ファイルの種類の登録

開発マネージャでは，標準で登録されているファイルの種類以外に，新しいファイルの種類としてファイルの名称や拡張子を登録，管理できます。新しく登録したファイルの種類は，ユーザ定義ファイルとして扱われます。新しいファイルの種類には，ユーザ定義ファイルから COBOL ソースを生成するツール（生成ツール）や，ユーザ定義ファイルを開いたときに起動するツール（定義ツール）を，それぞれ登録できます。

これによって，ユーザが任意のツールを使ってビルドを実行したり，ファイルの種類ごとに関連づけられたツールでファイルを編集したりできます。ここでは，新しいファイルの種類，およびファイルに関連づけるツールの登録方法について説明します。

なお，ここで登録したファイルの種類は，Windows 開発環境の場合だけで有効です。

(1)　生成ツールと定義ツール

ファイルには，生成ツールと定義ツールの 2 種類のツールを関連づけできます。

(a)　生成ツール

生成ツールとは，ユーザ定義ファイルから COBOL ソースまたは登録集原文を生成する際に使用するツールのことです。ユーザ定義ファイルは入力ファイル，COBOL ソースまたは登録集原文は出力ファイルとして扱われます。生成ツールは，入力ファイルであるユーザ定義ファイルがビルドされ，出力ファイルである COBOL ソースまたは登録集原文が生成されるタイミングで呼び出されます。

生成ツールとして登録できるプログラムは，次の条件をすべて満たしている必要があります。一つでも条件を満たしていないプログラムを生成ツールとして登録した場合，動作は保証しません。

- コンソールプログラム（Windows CUI プログラム）として呼び出せるインタフェースがある。
- コマンドラインから処理対象となるファイル名を引数として指定できる。
- 入力元のファイル，および生成される出力ファイルが，それぞれ一つである。
- メッセージは，すべて標準出力，または標準エラー出力に出力される。また，応答を求めるメッセージは出力されない。
- 処理の結果をリターンコードで判定できる。また，正常終了したときのリターンコードが 0 である。

また，コマンドファイルを使ってオプションを指定する場合には，次の条件も満たしている必要があります。

- コマンドファイルで渡される文字列の最後の改行を無視できる。

生成ツールには，次の二つの方法でオプションの引き渡しができます。

オプションを直接引数に指定する方法

入力ファイル名，出力ファイル名，オプションなどを，直接ツールの引数に指定する方法です。

この方法を使用する場合は，新しいファイルの種類の登録ダイアログボックスの［生成ツール］タブを選んで表示されるページで，「コマンドファイルインタフェースで起動する」チェックボックスをオフにします。新しいファイルの種類の登録ダイアログボックスについては「(2)　新しいファイルの種類の登録ダイアログボックス」を参照してください。

起動形式

```
”ツール名の絶対パス”
〔ファイルに指定されたオプション〕
〔入力オプション〕　入力ファイル名
〔出力オプション〕　出力ファイル名
```

使用例

［生成ツール］タブで「コマンドファイルインタフェースで起動する」チェックボックスをオフにして，生成ツール「gencobol.exe」を定義します。

- 入力コマンドの例

```
C:¥temp¥gencobol.exe -a -in Source.txt -out Source.cbl
```

コマンドファイルを使用する方法

コマンドファイルと呼ばれるファイルにオプションを指定し，コマンドファイル名を生成ツールの引数に指定する方法です。

この方法では，開発マネージャが，自動的に，［ファイル名の前に付加する起動オプション］に指定した入力ファイルと出力ファイルのオプションを，コマンドファイルに定義して出力します。ビルド時には，コマンドファイルの内容がコマンドライン上に展開されます。

この方法を使用する場合は，新しいファイルの種類の登録ダイアログボックスの［生成ツール］タブを選んで表示されるページで，「コマンドファイルインタフェースで起動する」チェックボックスをオンにします。新しいファイルの種類の登録ダイアログボックスについては「(2)　新しいファイルの種類の登録ダイアログボックス」を参照してください。

この方法は，OS でコマンドラインの文字列長に制限がある（文字列長が短い）場合に使用することをお勧めします。

起動形式

```
”ツール名の絶対パス”
〔@コマンドファイル名〕
```

コマンドファイルの形式

```
〔ファイルに指定されたオプション〕
〔入力オプション〕　入力ファイル名
〔出力オプション〕　出力ファイル名
```

使用例

［生成ツール］タブで「コマンドファイルインタフェースで起動する」チェックボックスをオンにして，生成ツール「gencobol.exe」を定義します。コマンドファイル「Sample.txt」は，開発マネージャによって自動的に出力されます。

- 入力コマンドの例

```
C:¥temp¥gencobol.exe @Sample.txt
```

- 展開後のコマンドラインの例

```
C:¥temp¥gencobol.exe -a -in Source.txt -out Source.cbl
```

(引き渡される値の説明)

入力オプション

ファイルの種類の登録で，入力ファイル名の前に付ける起動オプションに指定した文字列が設定されます。

入力ファイル名

生成ツールの入力ファイル名が設定されます。生成ツールでは，ユーザ定義ファイルを入力ファイルとして扱います。

ファイルに指定されたオプション

開発マネージャのプロジェクト設定ダイアログボックスで，ファイルに指定したオプションが格納されます。ファイルに指定されたオプションは，生成ツールで付加する起動オプション（入力オプションおよび出力オプション）より前に設定されます。

出力オプション

ファイルの種類の登録で，出力ファイル名の前に付ける起動オプションに指定した文字列が設定されます。

出力ファイル名

生成ツールの出力ファイル名が設定されます。生成ツールでは，COBOL ソースまたは登録集原文を出力ファイルとして扱います。

注意事項

- 生成ツールの起動時には，ファイルに設定したオプション情報だけがコマンドファイルで渡されます。プロジェクトに設定したオプション情報は，生成ツールに渡されません。
- オプションを直接引数に指定する方法の場合，コマンドラインの文字列長の制限は，使用している OS に依存します。

(b) 定義ツール

定義ツールとは，ユーザ定義ファイルを開いたときに起動するツールのことです。

定義ツールとして登録できるプログラムは，次の条件をすべて満たしている必要があります。一つでも条件を満たしていないプログラムを定義ツールとして登録した場合，動作は保証しません。

- コマンドラインで起動するとき，処理対象となるファイル名を引数として指定できる。
- ウィンドウプログラムである。

(2) 新しいファイルの種類の登録ダイアログボックス

新しいファイルの種類は，新しいファイルの種類の登録ダイアログボックスで設定します。

開発マネージャでは，次の表に示すファイルが標準で登録されています。

表 1-3 標準で登録されているファイル

拡張子名	ファイル名
.cbl;，cob;，ocb	COBOL ソース（固定形式）
.cbf;，ocf	COBOL ソース（自由形式）
.ecb	HiRDB 埋め込み型 UAP
.eoc	HiRDB 埋め込み型 UAP（OOCOBOL 移行用）
.cxd	XML データ定義ファイル

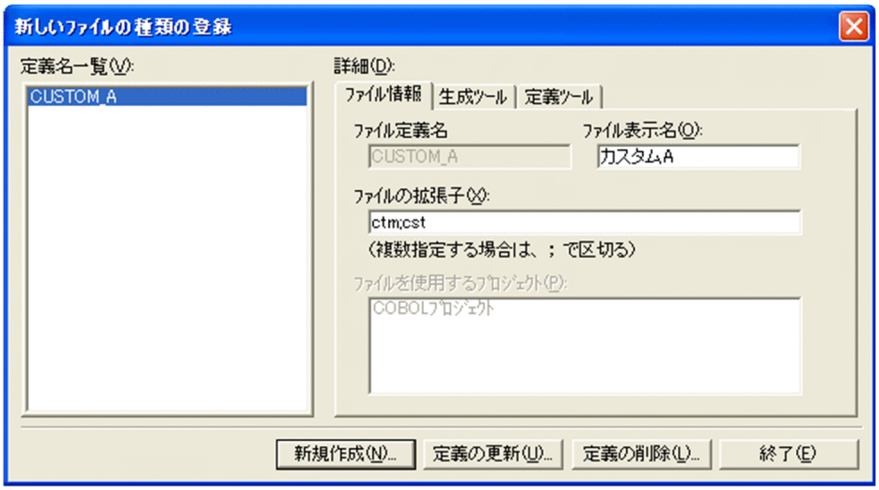
拡張子名	ファイル名
.ico	アイコンファイル
.obj	オブジェクトファイル
.flf	ファイル定義ファイル
.imp	マップ定義ファイル
.lib	ライブラリファイル
.rdf	レコード定義ファイル
.wdf	画面定義ファイル

新しいファイルの種類の登録ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのプロジェクトマスタを閉じた状態で，ウィンドウの［ツール］メニューから［開発マネージャのユーザカスタマイズ］の［新しいファイルの種類の登録］を選ぶ。
   新しいファイルの種類の登録ダイアログボックスが表示されます。

次に，このダイアログボックスについて，各タブを選ぶと表示されるページごとに説明します。

(a) ファイル情報の設定



定義名一覧

ファイル定義名の一覧が表示されます。

ファイル定義名

定義名一覧で選択しているファイルの種類の定義名，または新規に作成しているファイルの種類の定義名が表示されます。

ファイル表示名

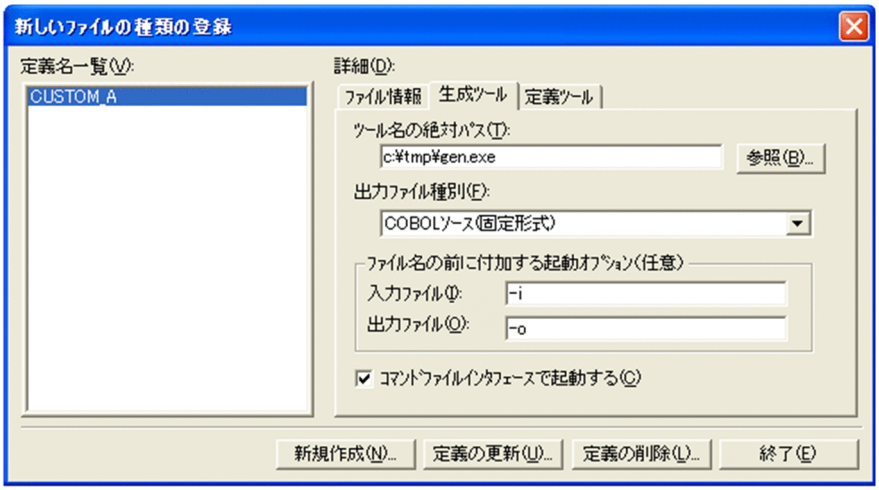
開発マネージャで扱うとき，ファイル種別として表示される名称を指定します。

ファイルの拡張子

登録するファイルの拡張子を指定します。ピリオド（.）を付けてはいけません。

複数の拡張子を指定する場合は，セミコロン（；）で区切って指定してください。拡張子は，五つまで指定できます。

(b)　生成ツールの設定



定義名一覧

ファイル定義名の一覧が表示されます。

ツール名の絶対パス

ユーザ定義ファイルから COBOL ソースまたは登録集原文を生成する際に使用するプログラムの絶対パス名を指定します。

出力ファイル種別

生成される出力ファイル（COBOL ソースまたは登録集原文）の種別を次の項目から選びます。

- COBOL ソース（固定形式）
- COBOL ソース（自由形式）
- 登録集原文（固定形式）
- 登録集原文（自由形式）

ファイル名の前に付加する起動オプション

ユーザ定義ファイルから COBOL ソースまたは登録集原文を生成する際に必要なオプションがある場合に，入力ファイル名または出力ファイル名の前に付けるオプションを指定します。

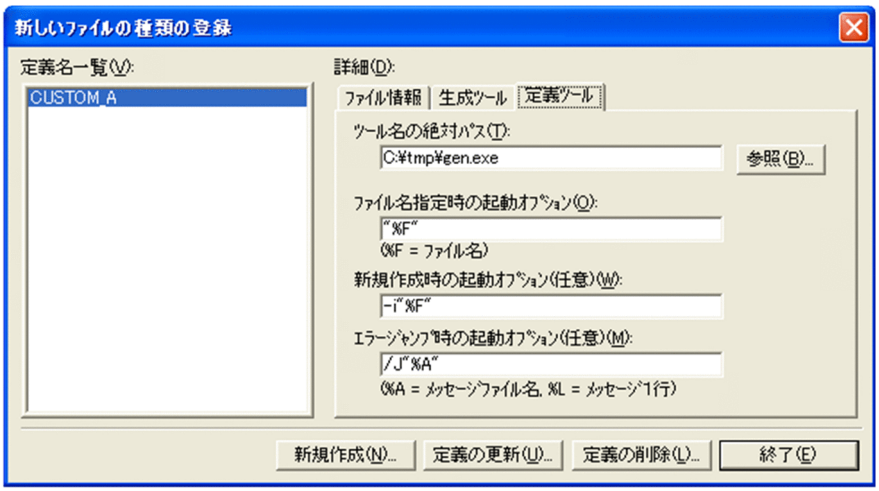
また，起動オプションは，新しいファイルの種類を登録したあとに，プロジェクト設定ダイアログボックスでユーザ定義ファイルを選んで指定できます。この場合はファイルに指定されたオプションとして扱われます。オプションの引き渡し方法の詳細は，「(1)　生成ツールと定義ツール」の「(a)　生成ツール」を参照してください。

オプションの指定が不要の場合は，何も指定しないでください。

コマンドファイルインタフェースで起動する

コマンドファイルを使用して生成ツールにオプションを引き渡す場合，このチェックボックスをオンにします。生成ツールの引数に直接オプションを指定する場合は，このチェックボックスをオフにします。

(c)　定義ツールの設定



定義名一覧

ファイル定義名の一覧が表示されます。

権限がないため編集できないファイル定義の場合，ファイル定義名の右側に「(編集不可)」と表示されます。詳細は，「(4)　登録したファイルの種類の情報の出力先と有効範囲」を参照してください。

ツール名の絶対パス

ユーザ定義ファイルを開いたときに起動するプログラムの絶対パス名を指定します。

ファイル名指定時の起動オプション

ファイル名を指定してツールを起動する場合のオプションを設定します。

"%F"を含む文字列を指定する必要があります。

新規作成時の起動オプション

ファイルを新規に作成するためにツールを起動する場合のオプションを設定します。指定しなかった場合は，オプションの指定なしでツールが起動します。

エラージャンプ時の起動オプション

生成ツール実行時に出力されるメッセージと定義ツールとの関連づけを設定します。オプションには，"%A"または"%L"のどちらか一方を指定します。指定しなかった場合は，オプションの指定なしでツールが起動します。

メッセージウィンドウに表示されたメッセージをダブルクリックすると，ここで設定した方法でツールが起動されます。ツールの起動方法には，次の 2 通りがあります。

%A（ダブルクリックされたメッセージを一時ファイルに書き出し，その一時ファイル名称をコマンドライン引数として渡す）

この方法では，ダブルクリックされたメッセージを一時ファイルとして書き出し，そのファイル名をツールに渡します。

一時ファイルはダブルクリック 1 回につき 1 個，環境変数 TEMP で示されるフォルダに作成されます。このファイルは，開発マネージャの終了時に削除されます。

この方法で起動する場合には，次の機能を持つツールを使用してください。

- ファイル名をコマンドライン引数で受け取る機能

（例）

ツールに「fsed.exe」，オプションに「/error△%A」を指定した場合のコマンドラインは「fsed.exe /error 一時ファイル名」となります。

△は半角空白文字を表します。

%L（ダブルクリックされたメッセージの 1 行をコマンドライン引数として渡す）

この方法では，メッセージウィンドウでダブルクリックされたメッセージ行を，直接ツールにコマンドライン引数として渡します。

この方法で起動する場合には，次のどちらかの機能を持つツールを使用してください。

- コンパイラの出力メッセージをコマンドライン引数で受け取って，直接タグジャンプする機能
- コマンドラインからのタグジャンプ機能は持たないが，コマンドラインの先頭作用対象をファイル名称として受け取り，ファイルを開く機能

（例）

ツールに「notepad.exe」，オプションに「%L」を指定した場合のコマンドラインは「notepad.exe ダブルクリックされた 1 行」となります。

（置換文字列の種類）

定義ツールのオプションに指定できる置換文字列を示します。

%F

編集対象となるファイル名（ドライブ名から始まる絶対パス）に置き換えられます。

%A

ダブルクリックされた時点のメッセージウィンドウの内容が一時ファイルに出力され，そのファイル名に置き換えられます。

%L

ダブルクリックされたメッセージウィンドウの 1 行と置き換えられます。

次に，定義ツールの設定項目に，これらの置換文字列をどのように指定するのか示します。

設定項目	%F	%A	%L
起動時のオプション	◎	×	×
メッセージウィンドウをダブルクリックしたときのオプション	×	○	○

（凡例）

◎：常に指定する

○：どちらか一方を指定する

×：指定できない

なお，メッセージウィンドウのダブルクリックと定義ツールの起動を連携させない場合，この項目には何も指定しないでください。

（オプションの指定で使用する文字列についての注意事項）

オプションの文字列に「%」を使用したい場合は，「%%」と二つ続けて記述してください。

（例）

"%F" … 置換文字列とみなされ，編集対象となるファイル名に置換されます。

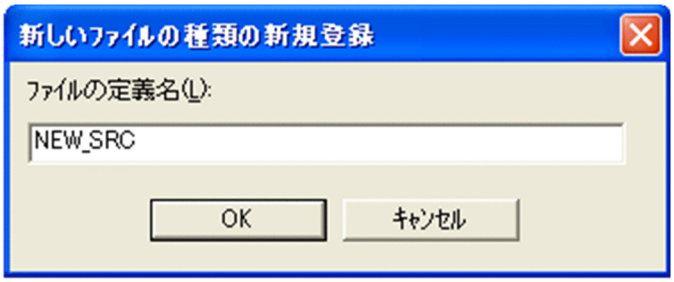
"%%F" … 「%F」という文字列とみなされます。

(3)　ファイルの種類の登録方法

(a)　ファイルの種類の新規登録

1. 新しいファイルの種類の登録ダイアログボックスで，［新規作成］ボタンを選ぶ。

新しいファイルの種類の新規登録ダイアログボックスが表示されます。



2. ファイルの種類を識別するための定義名を指定して，［OK］ボタンを選ぶ。

新しいファイルの種類の新規登録ダイアログボックスの「定義名一覧」に，指定した定義名が追加されます。

定義名は，英字で始まる半角英数字８文字以内で指定してください。

3. ［ファイル情報］タブで，ファイル表示名，ファイルの拡張子を入力する。

標準で登録されているファイルの拡張子については，「(2)　新しいファイルの種類の登録ダイアログボックス」を参照してください。

4. ［生成ツール］タブ，［定義ツール］タブを選んで表示されるページで，各ツールの設定を入力して，［終了］ボタンを選ぶ。

各タブの設定方法については，「(2)　新しいファイルの種類の登録ダイアログボックス」を参照してください。

(b)　ファイルの種類の登録情報の編集

1. 新しいファイルの種類の登録ダイアログボックスの「定義名一覧」から，情報を設定したいファイルの種類の定義名を選ぶ。

右側の「詳細」に，選んだファイルの種類の現在の登録情報が表示されます。

2.「詳細」にファイル情報，ツールの設定を入力して，［定義の更新］ボタンを選ぶ。

ファイルの種類の登録情報が更新されます。

(c) ファイルの種類の削除

1. 新しいファイルの種類の登録ダイアログボックスの「定義名一覧」から，削除したいファイルの種類の定義名を選ぶ。

2. ［定義の削除］ボタンを選ぶ。

削除してよいか確認するダイアログボックスが表示されます。ここで［OK］ボタンを選ぶと，ファイルの種類の登録情報が削除されます。

(d) ファイルの種類の保存

1. 新しいファイルの種類の登録ダイアログボックスで［終了］ボタンを選ぶ。

新しいファイルの種類の登録ダイアログボックスでファイルの種類の登録情報を変更していた場合，変更を保存するかを確認するダイアログボックスが表示されます。

2. ［はい］を選ぶ。

登録情報の変更が保存されます。

なお，新しいファイルの種類の登録ダイアログボックスでの設定内容を破棄したい場合は，［キャンセル］ボタンを選んでください。

(4) 登録したファイルの種類の情報の出力先と有効範囲

(a) ファイルの種類の情報の出力先

新しいファイルの種類の登録ダイアログボックスで登録した情報は，パッケージ情報ファイル（.pkg）に出力されます。パッケージ情報ファイル名の形式とパッケージ情報ファイルの出力先を次に示します。

パッケージ情報ファイル名の形式

_CSTM_定義名.pkg

定義名：

新しいファイルの種類の登録ダイアログボックスで指定した，ファイルの種類の定義名

パッケージ情報ファイルの出力先

- Windows(x86) COBOL2002 の場合
  環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002¥PLUGIN
- Windows(x64) COBOL2002 の場合
  環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002 64bit¥PLUGIN

環境変数 ALLUSERSPROFILE で指定されたフォルダに作成するファイルは，デフォルトではファイルを作成したユーザだけが更新できます。複数ユーザでパッケージ情報ファイルを更新したい場合は，必要に

応じて，パッケージ情報ファイルにアクセス権限を設定する，開発マネージャを管理者権限に昇格して起動し，更新するなど，対処してください。

(b)　ファイルの種類の情報の有効範囲

登録したファイルの種類の情報の有効範囲は，パッケージ情報ファイルの属性によって次のように異なります。

有効範囲	パッケージ情報ファイルの属性	
	読み込み権限だけ	読み込みと書き込み権限
開発マネージャでファイルの種類の登録情報を利用する	○	○
新しいファイルの種類の登録ダイアログボックスでファイルの種類を参照する		
新しいファイルの種類の登録ダイアログボックスでファイルの種類を変更する	×	

（凡例）

○：できる

×：できない

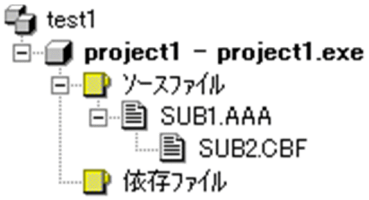
なお，パッケージ情報ファイルの定義情報に矛盾がある場合，パッケージ情報ファイルの属性に関係なく，定義情報は開発マネージャで利用することも，新しいファイルの種類の登録ダイアログボックスで参照／変更することもできません。

(5)　注意事項

- 一つのファイルの種類につき，生成ツール，定義ツールのどちらか一方，または両方の関連づけを登録する必要があります。
- ファイルの種類に関連づけた生成ツールでコンパイルする場合，プロジェクトやファイルに設定したオプション，および環境変数は無効となります。
- ファイルの種類を登録し，そのファイルに出力ファイルを設定した場合，開発マネージャの［ファイル］メニューの［出力ファイルの変更］では，出力ファイルを変更できません。出力ファイルを変更したい場合は，ファイルをプロジェクトから削除し，再度登録してください。
- ファイルの種類を登録し，そのファイルの種類を使って作成したプロジェクトをファイルの種類が登録されていない環境で開いた場合，メッセージウィンドウにエラーメッセージが表示され，ファイルは一般ファイルとして扱われます。

（例）

ファイルの種類「AAA」を登録した環境で次のようなプロジェクトを作成し，ファイルの種類「AAA」が未登録の環境で開こうとした場合



ファイルの種類が未登録の環境で開いた場合に表示されるメッセージ

```
===== プロジェクトマスタファイル入力時の警告メッセージ =====
現在の環境では使用できないファイル種別が指定されたか，または不正なフ
ァイル種別です。その他ファイルとして登録します。
"…¥test1.hmf",line 33: 'SUB2.CBF'

現在の環境では使用できないファイル種別が指定されたか，または不正なフ
ァイル種別です。その他ファイルとして登録します。
"…¥test1.hmf",line 44: 'SUB1.AAA'
```

1
2

1. 2のファイル（SUB1.AAA）がその他ファイルとして扱われるため，2のファイル（SUB1.AAA）の出力ファイルもその他ファイルとして扱われることを示しています。
2. ファイルの種類が登録されていないため，その他ファイルとして扱われることを示しています。

ファイルの種類が未登録の環境で開いた場合のプロジェクト構成



1.8　各種支援ツールでの環境設定

次に示す COBOL2002 の各種支援ツールでは，その中で使用するコマンドやユーザプログラムを実行する前に必要な環境変数を設定します。

- テストデバッガ
- カバレージ
- COBOL エディタ
- 開発マネージャ
- 実行支援

環境変数の設定については，マニュアル「COBOL2002　ユーザーズガイド」の「環境変数の設定」を参照してください。

注意事項

テストデバッガ，カバレージを連動実行した場合は，ユーザプログラムに対して，環境変数を設定しません。この場合，連動実行を開始する前に，ユーザプログラムに必要な環境変数を設定しておいてください。

なお，テストデバッガの連動実行については「10.17　プログラムからの連動実行」を，カバレージ情報の蓄積の連動実行については「15.2.2　プログラムからの連動実行による方法」を，カウント情報の表示の連動実行については「15.6.2　プログラムからの連動実行による方法」を，それぞれ参照してください。

1.9　プロジェクトマスタ情報の CSV ファイル入出力

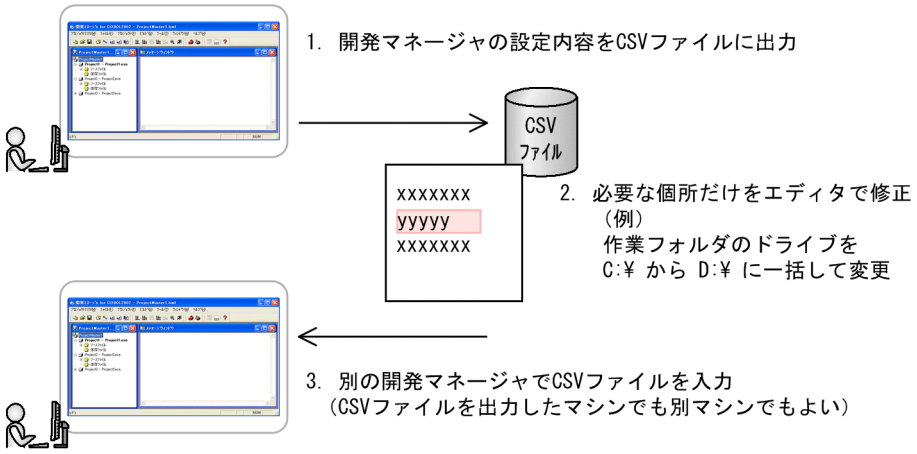
開発マネージャのプロジェクトマスタ情報を CSV ファイルに入出力できます。出力した CSV ファイルは，任意のエディタで編集できます。ファイルパスを一括変換するときは，CSV ファイルを編集すれば設定内容を変更できるので，それぞれの開発マネージャで設定し直す手間が省けます。

CSV ファイルは，Windows(x86) COBOL2002 と Windows(x64) COBOL2002 の開発マネージャで共用できます。Windows(x86) COBOL2002 と Windows(x64) COBOL2002 との仕様差で指定できないオプションが CSV ファイルで指定されている場合は，開発マネージャで CSV ファイルを入力したあとで，その旨のエラーメッセージがメッセージウィンドウに表示されます。

CSV ファイルの入力時に，プロジェクトマスタファイル（.hmf）を作成して保存します。このとき，保存先フォルダに同名のプロジェクトマスタファイル（.hmf）が存在した場合は，ダイアログボックスが表示されます。上書きするかキャンセルするかを選んでください。上書きを選択すると，プロジェクトマスタファイルが上書きされます。キャンセルを選択すると，CSV ファイルの入力がキャンセルされます。

プロジェクトマスタ情報の CSV ファイル入出力の例を次の図に示します。

図 1-5　プロジェクトマスタ情報の CSV ファイル入出力の例



CSV ファイルを開発マネージャに入力した場合にエラーになったときの対処

エラーメッセージを参照して，CSV ファイルを修正してください。

修正方法がわからない場合は，開発マネージャで，編集前の既存のプロジェクトマスタ情報を CSV ファイルに出力し，エラーメッセージが表示されたときに入力した CSV ファイルの内容と比較し，セクションの構成誤りやエントリ行の過不足がないか確認し，誤りがある個所を編集前の状態に戻してください。そのあとで，開発マネージャで，修正した CSV ファイルを入力してエラーメッセージが表示されないことを確認してください。

CSV ファイルの編集時には，編集後の内容に問題がないかを，随時，開発マネージャで入力して確認しながら編集することをお勧めします。

1.9.1　CSV ファイルの使用方法

プロジェクトマスタ情報の CSV ファイルの主な使い方について次に示します。セクションおよびエントリ行については，「1.9.3　プロジェクトマスタ情報の CSV ファイルの形式」を参照してください。

(1)　プロジェクトマスタ内のプロジェクトを複製する

1. 既存のプロジェクトマスタを CSV ファイルに出力する。
2. 任意のエディタで CSV ファイルを開き，複製したいプロジェクト情報のセクション（ProjectInf セクション）を複製する数だけコピーする。
3. プロジェクト名，プロジェクトフォルダ名およびプロジェクト作業フォルダ名のエントリ行の値を変更する。
   同一プロジェクトマスタ内に同一名のプロジェクトは複数登録できないため，プロジェクト名を変更します。パス構成はプロジェクトごとに変更してください。
4. プロジェクトに登録するソースファイルの情報を変更する。
5. 開発マネージャで，CSV ファイルを入力としてプロジェクトマスタファイル（.hmf）を作成して保存する。

(2)　プロジェクトマスタ内の複数プロジェクトのオプション設定内容をそろえる

1. 既存のプロジェクトマスタを CSV ファイルに出力する。
2. 任意のエディタで CSV ファイルを開き，基準とするプロジェクトのオプション設定情報のセクション（OptionDataInf セクション）を，異なるプロジェクトのオプション設定情報のセクションにコピーする。
3. 開発マネージャで，CSV ファイルを入力としてプロジェクトマスタファイル（.hmf）を作成して保存する。

(3)　プロジェクトに登録されたソースファイルを別プロジェクトに複製する

1. 既存のプロジェクトマスタを CSV ファイルに出力する。
2. 任意のエディタで CSV ファイルを開き，複製したいファイル情報のセクション（FileInf セクション）を追加先プロジェクトのプロジェクト情報のセクション（ProjectInf セクション）に複製する数だけコピーする。
3. ファイル名，ファイルフォルダ名のエントリ行の値を変更する。
   同一プロジェクトマスタ内に同一名のファイルは複数登録できないため，ファイル名は変更します。ファイルフォルダ名を変更する必要はありません。

4. 開発マネージャで，CSV ファイルを入力としてプロジェクトマスタファイル（.hmf）を作成して保存する。

(4) プロジェクトに登録された複数ソースファイルのオプション設定内容をそろえる

1. 既存のプロジェクトマスタを CSV ファイルに出力する。
2. 任意のエディタで CSV ファイルを開き，基準とするファイルのオプション設定情報のセクション（OptionDataInf セクション）を異なるファイルのオプション設定情報のセクションにコピーする。
3. 開発マネージャで，CSV ファイルを入力としてプロジェクトマスタファイル（.hmf）を作成して保存する。

(5) プロジェクトフォルダ，ファイルフォルダのパスを一括変換する

1. 既存のプロジェクトマスタを CSV ファイルに出力する。
2. 任意のエディタで CSV ファイルを開き，パスを置換する。
3. 開発マネージャで，CSV ファイルを入力としてプロジェクトマスタファイル（.hmf）を作成して保存する。

1.9.2 CSV ファイル編集時の規則

CSV ファイルを編集するときは，CSV ファイルおよび Excel の仕様で，次に示す規則に従ってください。

(1) オプションはダブルコーテーション（"）で囲む

サブオプションがあるオプションは，メインオプションとサブオプションをコンマで区切って表示するため，そのまま CSV ファイルに出力して Excel で開くと，そのオプションはメインとサブとでそれぞれ別のデータとして認識されて表示されます。

（例）

「-SrcList,OutputAll」は「=-SrcList」と「OutputAll」という別々のデータとして表示される。

上記を回避するため，開発マネージャはサブオプションの有無に関係なく，一律でオプションをダブルコーテーション（"）で囲んで CSV ファイルに出力します。CSV ファイルを編集してオプションを追加するときは，そのオプションをダブルコーテーション（"）で囲んで指定してください。

(2) オプションの先頭にはアポストロフィ（'）を付ける

オプションには先頭にハイフンが付いているため，そのまま CSV ファイルに出力して Excel で開くと，そのデータは数式の無効な名前として認識され，「#NAME?」と表示されます。

（例）

「"-SrcList,OutputAll"」は「=-SrcList,OutputAll」という数式の無効な名前として「#NAME?」と表示される。

上記を回避するため，開発マネージャは，オプションの先頭にアポストロフィ（'）を付けて CSV ファイルに出力します。CSV ファイルを編集してオプションを追加するときは，そのオプションの先頭にアポストロフィ（'）を付けてください。

(3) コメントを編集する場合は，テキストエディタを使用する

プロジェクトやソースファイルにコメントが設定されている場合は，Excel ではなくテキストエディタで編集してください。理由を次に示します。

- コメントには，Excel が自動で付与または削除する文字や，Excel 上で意図しない表記になる文字（ダブルコーテーション（"），アポストロフィ（'）またはコンマ（, ）区切り）が指定されることがあるため。
- CSV ファイル内のデータがダブルコーテーション（"）で囲まれている場合，Excel で開いて保存すると，Excel が先頭と末尾のダブルコーテーション（"）を削除して保存されるため。

同様の理由で，開発マネージャは，プロジェクトやソースファイルに設定されたコメントは，上記（1）（2）で示すデータ加工はしないで，設定されたままの文字列を CSV ファイルに出力します。

CSV ファイルでコメントを編集する場合は，上記（1）（2）に基づいたデータ加工はしないでください。指定した文字列がそのまま設定されます。

(4) 属性値にコンマ区切りの値を指定する場合，または属性名と同一の文字列を指定する場合は，指定する値をダブルコーテーション（"）で囲む

オプションの引数やユーザ定義オプション，環境変数の定義値は，ダブルコーテーション（"）で囲まれて CSV ファイルに出力されます。CSV ファイルで引数があるオプション，ユーザ定義オプション，または環境変数を追加するときには，指定値をダブルコーテーション（"）で囲んで指定してください。

なお，上記以外の属性値については，開発マネージャはデータ加工しないで CSV ファイルに出力しますが，上記以外の属性値についても，コンマ区切りの値を指定するときには，指定値をダブルコーテーション（"）で囲んで指定してください。

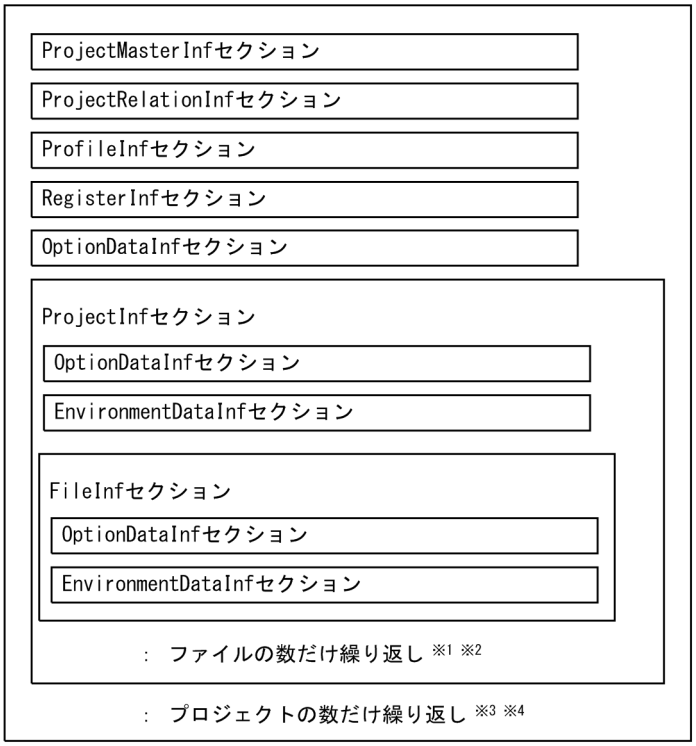
（例）

プロジェクト名に「Pro,je,ct1」と指定するときには，「プロジェクト名,"Pro,je,ct1"」と指定します。

1.9.3　プロジェクトマスタ情報の CSV ファイルの形式

プロジェクトマスタ情報の CSV ファイルの形式を次の図に示します。

図 1-6　プロジェクトマスタ情報の CSV ファイルの形式



注※1　登録できるソースファイルの上限は3,000個です。
ただし，3,000を超えてもエラーメッセージは表示されません。

注※2　出力順は，ソースファイルのファイル名の昇順になります。

注※3　プロジェクトマスタ中のプロジェクトの数の上限は1,000個です。
ただし，1,000を超えてもエラーメッセージは表示されません。

注※4　出力順は，プロジェクト名の昇順になります。

(1)　CSV ファイルのセクション

情報は，分類ごとに区分けされています。各区分をセクションといいます。

セクション名の一覧を次の表に示します。

表 1-4　セクション名の一覧

セクション名	指定できる数	定義内容
ProjectMasterInf	1	プロジェクトマスタの情報
ProjectRelationInf	0～1	プロジェクト間の依存関係の情報
ProfileInf	0～1	実行に関する情報
RegisterInf	0～1	仮依存ファイルの情報

セクション名	指定できる数	定義内容
ProjectInf	1～1000	プロジェクト情報
FileInf	0～3000	ファイル情報
OptionDataInf	0 以上	プロジェクトまたはファイルのオプション情報
EnvironmentDataInf	0 以上	プロジェクトまたはファイルの環境変数情報

セクションの規則を次に示します。

- セクションの開始行とセクションの終了行は，角括弧（[]）で囲まれたキーワードで構成されます（開始行は[セクション名_BEGIN]，終了行は[セクション名_END]）。
- ProjectMasterInf セクションは，指定が 1 個必要です。
- ProjectRelationInf セクション，ProfileInf セクション，RegisterInf セクションは，1 個だけ指定できます。
- ProjectInf セクションは，1 個以上指定が必要です。また，複数個指定できます。
- OptionDataInf セクションは，ProjectInf セクションまたは FileInf セクションに含めて，各セクション中に 1 個だけ指定できます。また，プロジェクトマスタのオプションを示す OptionDataInf セクションを 1 回だけ指定できます。
- EnvironmentDataInf セクションは，ProjectInf セクションおよび FileInf セクション内に 1 個だけ指定できます。

(2) CSV ファイルのエントリ行

セクションに指定する属性名と属性値の組み合わせをエントリ行といいます。エントリ行の先頭の属性名は，1 けた目から指定されている必要があります。また，それに続く属性値との間には，コンマ（,）が必要です。

1.9.4 CSV ファイルのセクションの形式

各セクションについて説明します。

注意

編集できない属性は，開発マネージャが生成した内容をそのまま使用してください。編集できる属性についても，開発マネージャが生成した内容をそのまま使用するか，別の正しい値を指定してください。

出力例（指定例）の属性値にある△は，半角空白を示します。

(1) ProjectMasterInf セクション

ProjectMasterInf セクションには，プロジェクトマスタの情報を定義します。ProjectMasterInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値	CSV 編集
CSV ファイル形式バージョン	1 ※1	×
開発マネージャ識別子	32bit/64bit ※2	◎
プロジェクトマスタ名	プロジェクトマスタの名前 ※3	◎
プロジェクトマスタフォルダ名	プロジェクトマスタの絶対パス ※3	◎
作業中プロジェクト	作業中プロジェクトの名前	◎

（凡例）

◎：編集できます。指定が必要な属性です。省略できません。

×：編集できません。CSV ファイルに出力した値のままにしてください。

注※1

CSV ファイルの形式のバージョンが出力されます。

注※2

Windows(x86) COBOL2002 の開発マネージャでは「32bit」，Windows(x64) COBOL2002 の開発マネージャでは「64bit」と出力されます。

注※3

CSV ファイルの入力時に，プロジェクトマスタフォルダ名の属性値とプロジェクトマスタ名の属性値に基づいてプロジェクトマスタファイル（.hmf）を作成して保存します。すでに同名のファイルが存在する場合は，上書きしてよいかどうかを確認するダイアログボックスが表示されます。

（例）

プロジェクトマスタフォルダ名が D:¥ProjectMaster1Folder1，プロジェクトマスタ名が ProjectMaster1 のとき，「D:¥ProjectMaster1Folder1¥ProjectMaster1.hmf」が作成されます。

開発マネージャによる出力例（指定例）

```
[ProjectMasterInf_BEGIN]
CSVファイル形式バージョン,1
開発マネージャ識別子,32bit
プロジェクトマスタ名,ProjectMaster1
プロジェクトマスタフォルダ名,D:¥ProjectMasterFolder1
作業中プロジェクト,Project1
[ProjectMasterInf_END]
```

(2) ProjectRelationInf セクション

ProjectRelationInf セクションには，プロジェクト間の依存関係の情報を定義します。ProjectRelationInf セクションには，次に示すエントリ行を複数個指定できます。ProjectRelationInf セクションを省略すると，登録されているプロジェクトすべてが最上位のプロジェクト（階層 1）として設定されます。

ProjectRelationInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値 1	属性値 2	CSV 編集
階層	階層の深さ	プロジェクト名	◎

（凡例）

◎：編集できます。指定が必要な属性です。省略できません。

開発マネージャによる出力例（指定例）

次に示すように，Project1 と Project3 が最上位のプロジェクトで，Project1 が Project2 に依存する関係の場合の出力例（指定例）を次に示します。




```
[ProjectRelationInf_BEGIN]
階層,1,Project1
階層,2,Project2
階層,1,Project3
[ProjectRelationInf_END]
```

(3) ProfileInf セクション

ProfileInf セクションには，実行に関する情報を定義します。［実行］メニューから［実行］ダイアログボックスを表示したときに，デフォルトで表示される情報です。ただし，作業中のプロジェクトを変更したタイミングでクリアされます。

ProfileInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値	CSV 編集
実行対象プロジェクト	実行の対象となったプロジェクト名	◎
実行可能ファイル名※1	実行可能ファイルの絶対パス	○
作業フォルダ※2	ユーザが指定したフォルダ	○
引数※3	ユーザが指定した引数	○

（凡例）

◎：編集できます。指定が必要な属性です。省略できません。

○：編集できます。指定が任意の属性です。省略できます。

注※1

実行可能ファイル名が指定されている場合に出力されます。

注※2

実行時の作業フォルダが指定されている場合に出力されます。

注※3

実行時の引数が指定されている場合に出力されます。

開発マネージャによる出力例（指定例）

```
[ProfileInf_BEGIN]
実行対象プロジェクト,Project1
実行可能ファイル名,D:¥ProjectMasterFolder1¥project1¥work¥Project1.exe
作業フォルダ,D:¥Folder1
引数,Param1
[ProfileInf_END]
```

(4) RegisterInf セクション

RegisterInf セクションには，仮依存ファイルの情報を定義します。RegisterInf セクションには，エントリ行を複数個指定できます。

RegisterInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値 1	属性値 2	CSV 編集
グループ指定開始 ※1	グループ名	-	○
グループ指定終了 ※1	グループ名	-	○
仮依存ファイル	仮依存ファイルの絶対パス	ファイルの種別 ※2	◎

（凡例）

-：該当の値はありません。

◎：編集できます。指定が必要な属性です。省略できません。

○：編集できます。指定が任意の属性です。省略できます。

注※1

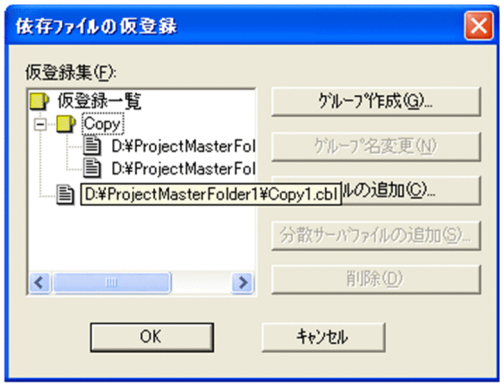
グループ指定開始とグループ指定終了で指定されている場合はグループ内のファイルであり，単独で指定されている場合はグループと同レベルのファイルを示します。

注※2

ファイルの種別については，「1.9.6　CSV ファイルに指定できるファイルの種別」を参照してください。

開発マネージャによる出力例（指定例）

次のように仮依存ファイルを登録する場合の出力例（指定例）を次に示します。




```
[RegisterInf_BEGIN]
仮依存ファイル,D:¥ProjectMasterFolder1¥Copy1.cbl,登録集原文(固定形式)
グループ指定開始,Copy
仮依存ファイル,D:¥ProjectMasterFolder1¥Copy1.cbl,登録集原文(固定形式)
仮依存ファイル,D:¥ProjectMasterFolder1¥Copy1.cbl,登録集原文(固定形式)
グループ指定終了,Copy
[RegisterInf_END]
```

(5)　ProjectInf セクション

ProjectInf セクションは，プロジェクト情報を定義します。

ProjectInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値 1	属性値 2	CSV 編集
プロジェクト名	プロジェクト名	-	◎
最終生成物の有無	最終生成物が存在するかどうか [0/1] ※1	-	◎
最終生成物の種別 ※2	最終生成ファイルの種別 ※3	-	◎
最終生成物フォルダ名 ※2	最終生成ファイルを作成するフォルダの絶対パス	-	◎
最終生成物名称 ※2	最終生成ファイル名	-	◎
プロジェクトの種別	COBOL プロジェクト	-	×
プロジェクトフォルダ名	プロジェクトフォルダの絶対パス	-	◎
プロジェクト作業フォルダ名 ※4	プロジェクト作業フォルダの絶対パス	-	◎
コメント ※5	コメント	-	○
リンク必須	リンケージが必要かどうか [0/1] ※6	-	◎
ビルド必須	ビルドが必要かどうか [0/1] ※7	-	◎

属性名	属性値 1	属性値 2	CSV 編集
ユーザ定義コンパイラオプション文字列 ※8	ユーザ定義コンパイラオプション文字列	-	○
ユーザ定義リンカオプション文字列 ※8	ユーザ定義リンカオプション文字列	-	○
ユーザ定義環境変数 ※8 ※9	環境変数名	環境変数の値	○

（凡例）

-：該当の値はありません。

◎：編集できます。指定が必要な属性です。省略できません。

○：編集できます。指定が任意の属性です。省略できます。

×：編集できません。CSV ファイルに出力した値のままにしてください。

注※1

最終生成物が存在する場合は 1，存在しない場合は 0 が出力されます。

注※2

最終生成物が存在しない場合は，このエントリ行は出力されません。

注※3

最終生成ファイルの種別については，「1.9.6　CSV ファイルに指定できるファイルの種別」を参照してください。

注※4

プロジェクトフォルダと作業フォルダが同じ場合でも，このエントリ行は出力されます。

注※5

コメントが設定されている場合に出力されます。

注※6

リンケージが必要な場合は 1，不要な場合は 0 が出力されます。プロジェクトで設定したリンカオプションを変更したときは，1 が出力されます。

注※7

ビルド（登録されているソースファイルのコンパイルとリンク）が必要な場合は 1，ビルドが不要な場合は 0 が出力されます。プロジェクト設定のコンパイラオプションまたは環境変数を変更したときは，1 が出力されます。

注※8

ユーザ設定タブで設定されている場合に出力されます。

注※9

このエントリ行は，複数個指定できます。

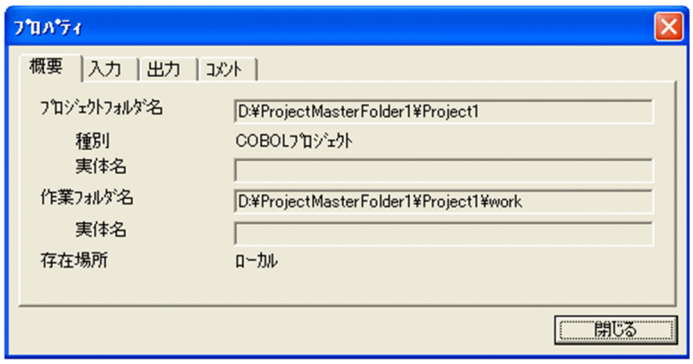
ProjectInf セクション中に定義されるセクション一覧を次に示します。

セクション名	説明
OptionDataInf	オプション情報
EnvironmentDataInf	環境変数情報
FileInf	プロジェクトに登録されているファイル情報

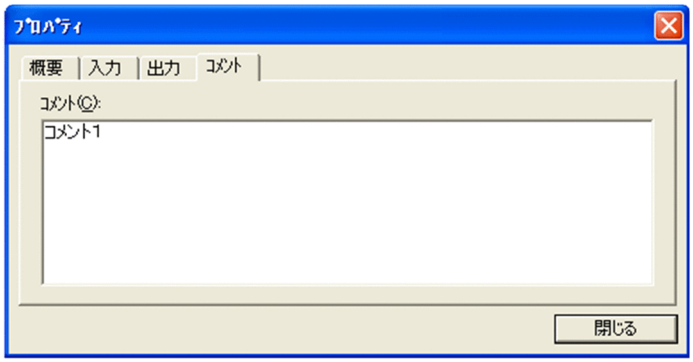
開発マネージャによる出力例（指定例）

次に示す設定の場合の出力例（指定例）を次に示します。

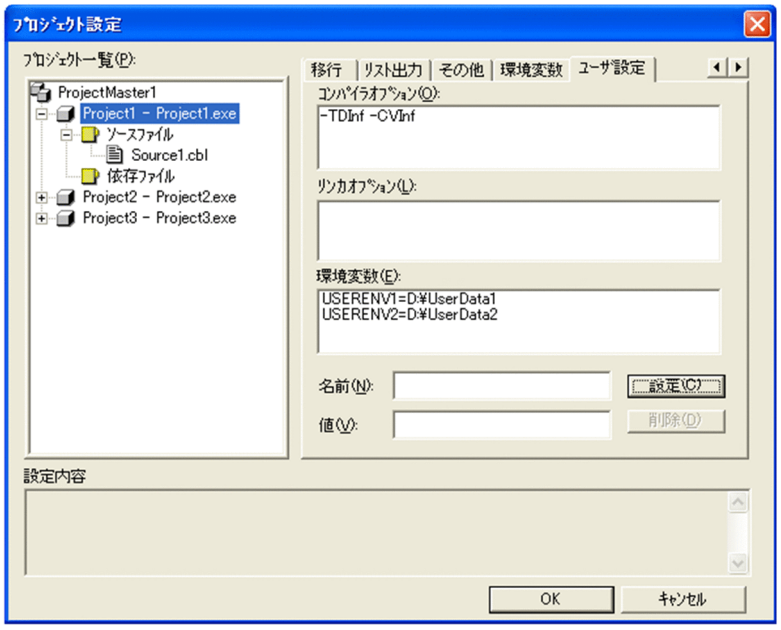
プロジェクトのプロパティ（概要）

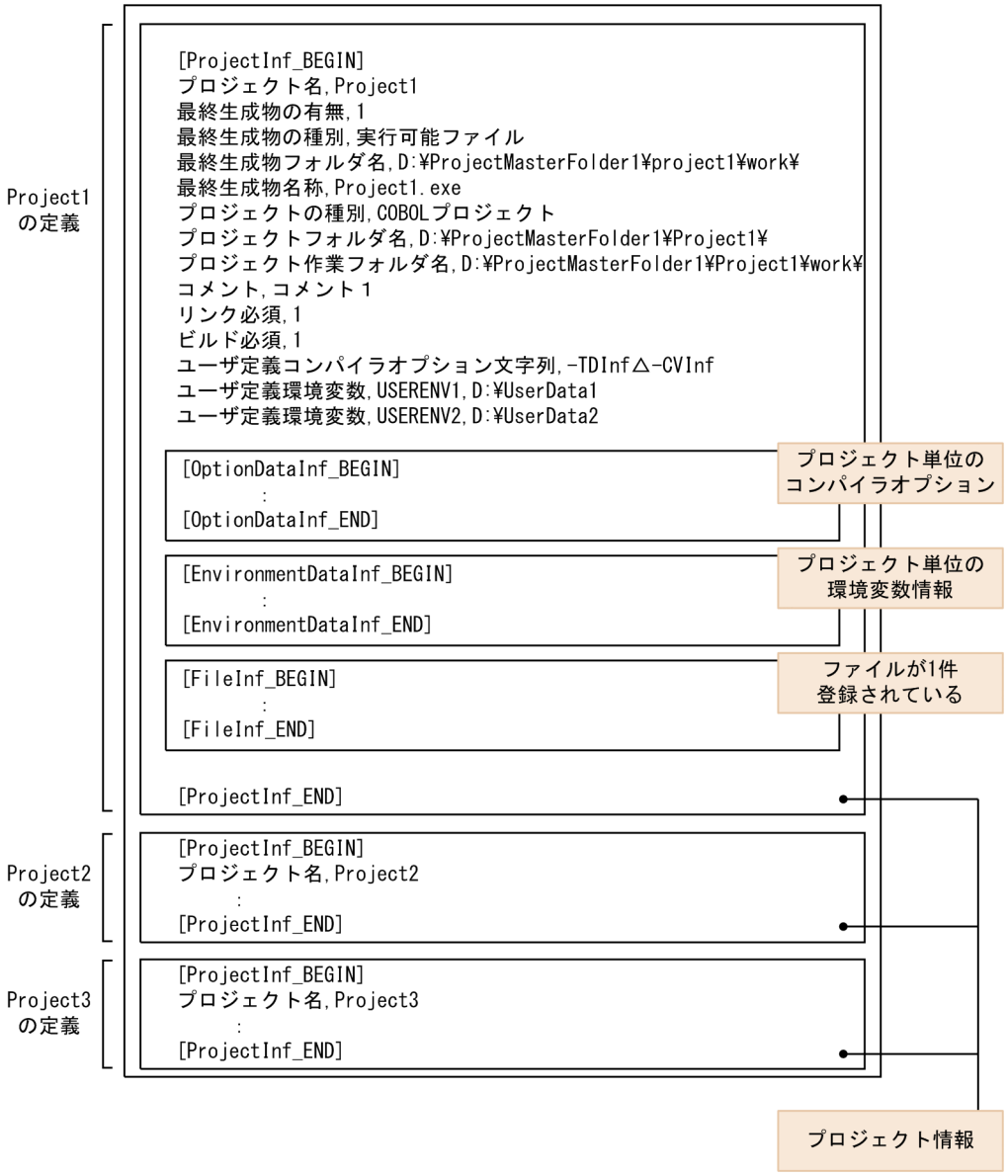


プロジェクトのプロパティ（コメント）



プロジェクト設定内容


プロジェクト設定
プロジェクト一覧(P):
ProjectMaster1
Project1 - Project1.exe
ソースファイル
Source1.cbl
依存ファイル
Project2 - Project2.exe
Project3 - Project3.exe
移行
リスト出力
その他
環境変数
ユーザ設定
コンパイラオプション(O):
-TDInf -CVInf
リンカオプション(L):
環境変数(E):
USERENV1=D:¥UserData1
USERENV2=D:¥UserData2
名前(N):
設定(C)
値(V):
削除(D)
設定内容
OK
キャンセル



(6) FileInf セクション

FileInf セクションは，ファイル情報を定義します。

FileInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値 1	属性値 2	CSV 編集
ファイルタイプ	ソースファイル/依存ファイル/出力ファイル/関連ファイル ※1	-	◎
ファイルの種別	ファイルの種別 ※2	-	◎
ファイルフォルダ名	ファイルフォルダの絶対パス	-	◎
ファイル名	ファイル名	-	◎

属性名	属性値 1	属性値 2	CSV 編集
自動登録	一般ファイルとして登録されたかどうか［0/1］※3	-	×
削除可能	ユーザによる削除が可能かどうか［0/1］※4	-	×
リンクファイル※5	ソースとしても登録が必要な出力ファイルかどうか［0/1］※5	-	×
コメント※6	コメント	-	○
コンパイル必須	コンパイルが必要かどうか［0/1］※7	-	◎
ユーザ定義コンパイラオプション文字列※8	ユーザ定義コンパイラオプション文字列	-	○
ユーザ定義環境変数※8※9	環境変数名	環境変数の値	○
チェックアウト中	チェックアウトされているかどうか［0/1］※10	-	◎

（凡例）

-：該当の値はありません。

◎：編集できます。指定が必要な属性です。省略できません。

○：編集できます。指定が任意の属性です。省略できます。

×：編集できません。CSV ファイルに出力した値のままにしてください。

注※1

ソースファイルの情報の場合は「ソースファイル」，依存ファイルの情報の場合は「依存ファイル」，出力ファイルの情報の場合は「出力ファイル」，関連ファイルの情報の場合は「関連ファイル」と出力されます。

依存ファイル，出力ファイル，関連ファイルの場合は，直前に定義されているソースファイルの FileInf セクションとの関係を示します。FileInf セクションが出力ファイルの場合で，直前にソースファイルの FileInf セクションが定義されていなければ，プロジェクトのメイン以外の最終生成物であることを示します。

注※2

ファイルの種別については，「1.9.6　CSV ファイルに指定できるファイルの種別」を参照してください。

注※3

開発マネージャで扱うファイル種別以外のファイルが一般ファイルとして強制的に登録された場合は 1，それ以外のファイルの場合は 0 が出力されます。ファイルタイプがソースファイルのときだけ出力します。ほかのファイルタイプのときには，指定されていても値を無視します。

注※4

ユーザが削除できる場合は 1，できない場合は 0 が出力されます。

ユーザによって登録されたものは削除可能です。

注※5

ファイルタイプが出力ファイルのときだけ出力されます。ほかのファイルタイプのときは，指定されていても無視されます。COBOL ソースファイルとしても登録されなければならないファイルの場合は 1，登録されていなくてよいファイルの場合は 0 が出力されます。

注※6

コメントが設定されている場合に出力されます。

注※7

コンパイルが必要な場合は 1，不要な場合は 0 が出力されます。ファイルを追加したとき，またはコンパイラオプションか環境変数を変更したときは，1 が出力されます。ファイルタイプがソースファイルのときだけ出力されます。ほかのファイルタイプのときは，指定していても無視されます。

注※8

ユーザ設定タブで設定されている場合に出力されます。

注※9

このエントリ行は，複数個指定できます。

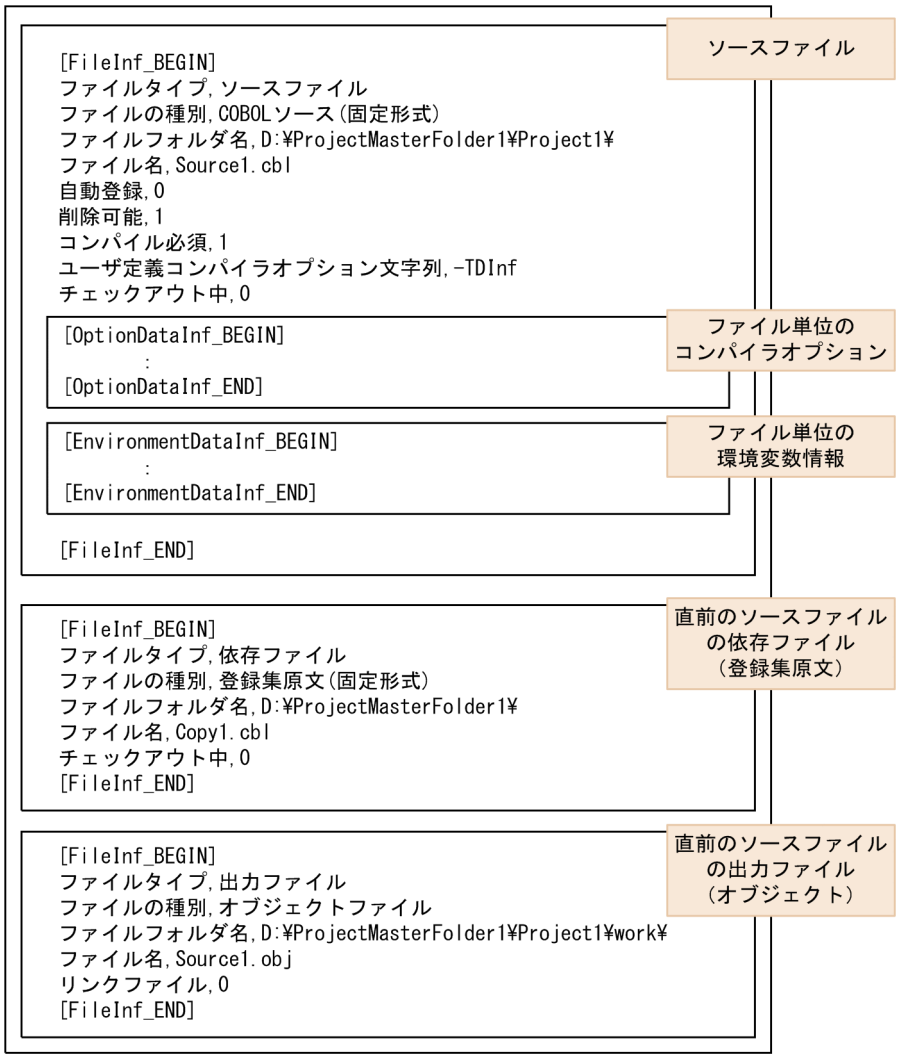
注※10

チェックアウトされている場合は 1，チェックアウトされていない場合は 0 が出力されます。ファイルタイプがソースファイルまたは依存ファイルのときに出力されます。ファイルタイプが出力ファイルのときは，指定していても無視されます。

FileInf セクション中に定義されるセクション一覧を次に示します。

セクション名	説明
OptionDataInf	オプション情報
EnvironmentDataInf	環境変数情報

開発マネージャによる出力例（指定例）



(7)　OptionDataInf セクション

OptionDataInf セクションは，プロジェクトまたはファイルのオプション情報を定義します。
OptionDataInf セクションの指定がない場合は，オプションの指定がないものとして扱われます。

プロジェクトの OptionDataInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値 1	属性値 2	属性値 3 以降	CSV 編集
プロジェクトの種類	プロジェクトの種類 ※1	-	-	◎
開発マネージャ固有オプション ※2	表示文字列	[,引数[[,引数]...]] ※3	-	○
オプション ※4 ※5	指定順序 ※6	表示文字列 ※7	[,引数[[,引数]...]] ※3	○

ファイルの OptionDataInf セクションに指定するエントリ行の詳細を次に示します。

属性名	属性値 1	属性値 2	属性値 3 以降	CSV 編集
オプション ※4 ※5	指定順序 ※6	表示文字列 ※7	[,引数[[,引数]...]] ※3	○
開発マネージャ固有オプション ※8	表示文字列	-	-	○

プロジェクトマスタの OptionDataInf セクションに記述するエントリの詳細を次に示します。

属性名	属性値 1	属性値 2	属性値 3 以降	CSV 編集
開発マネージャ固有オプション ※2	表示文字列	[,引数[[,引数]...]] ※3	-	○

（凡例）

-：該当の値はありません。

◎：編集できます。指定が必要な属性です。省略できません。

○：編集できます。指定が任意の属性です。省略できます。

注※1

プロジェクトの種類については，「1.9.5　CSV ファイルに指定できるプロジェクトの種類」を参照してください。

注※2

開発マネージャ固有オプションの指定に応じて，次に示す情報が出力されます。

プロジェクトのオプション

- メインファイルが指定されているときは「メインファイル指定情報」
- ライブラリが指定されているときは「ライブラリ指定情報」
- オプション「リンケージ処理を行わない」が指定されているときは「リンケージ処理対象外」
- Windows(x64) COBOL2002 の開発マネージャで作成した場合は「64bit プログラム指定情報」

プロジェクトマスタのオプション／プロジェクトのオプション

- 「データ影響波及分析用のデータベースを作成する」が指定されているときは「データ影響波及分析用データベース作成情報」
- 「サブシステム名の指定」が指定されているときは「サブシステム名指定情報」
- 「一意名呼び出し対応付けファイルの指定」が指定されているときは「一意名呼び出し対応付けファイル指定情報」
- 「データ領域番号の指定」が指定されているときは「データ領域番号の指定情報」

注※3

オプションの引数が，属性値 2 以降に出力されます。引数がない場合は出力されません。引数と引数の間はコンマ（,）で区切られます。

注※4

このエントリ行は，複数個指定できます。

注※5

表示文字列のアルファベットの昇順で出力されます。

注※6

コンパイラに渡されるオプションの順序が出力されます。サブオプションの指定順序は出力されません。同一オプションを異なる指定順序で指定した場合，あとに指定した順序が有効になります。

注※7

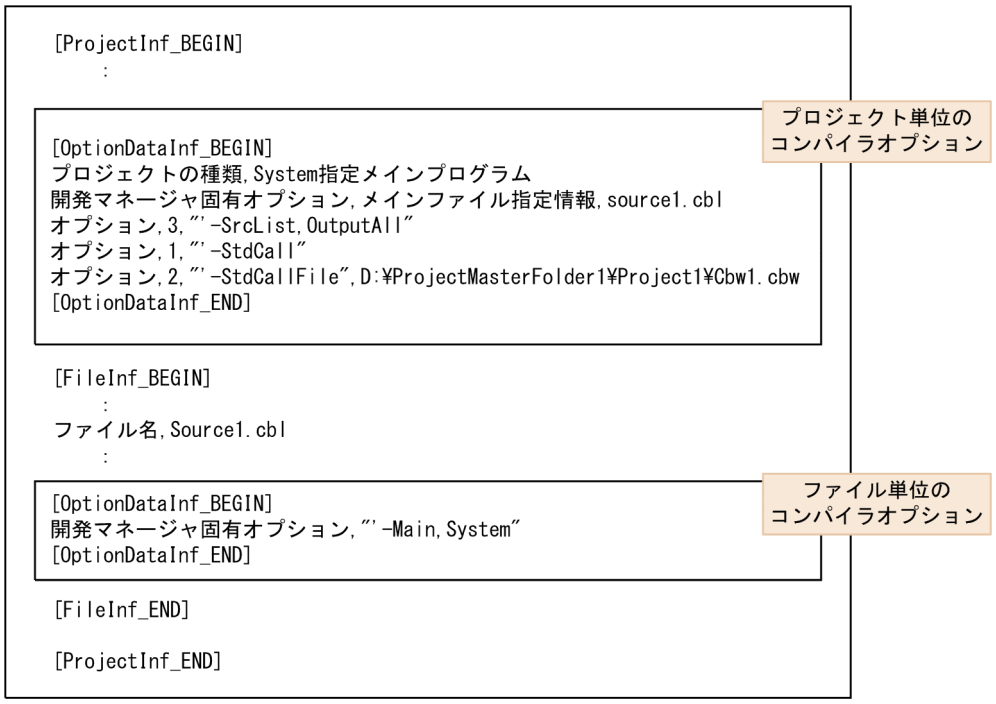
オプションの表示文字列を出力するとき，文字列の先頭にアポストロフィ（'）が付いて，文字列をダブルコーテーション（"）で囲んで出力されます。詳細は「1.9.2　CSV ファイル編集時の規則」を参照してください。

注※8

メインファイルに指定されている場合に出力します。メインファイルに指定するファイルを変更するときは，ファイルの OptionDataInf セクションに指定するエントリの指定も変更してください。

- System 指定メインプログラムのメインファイルに指定されているときは，「"'-Main,System"」を出力します。
- V3 指定メインプログラムのメインファイルに指定されているときは，「"'-Main,V3"」を出力します。

開発マネージャによる出力例と指定例



(8)　EnvironmentDataInf セクション

EnvironmentDataInf セクションは，プロジェクトまたはファイルの環境変数情報を定義します。
EnvironmentDataInf セクションの指定がない場合には，環境変数の指定がないものとして扱われます。

EnvironmentDataInf セクションに指定するエントリ行の詳細を次に示します。

定義値の数がない形式

属性名	属性値 1	属性値 2 以降	CSV 編集
環境変数 ※1	表示文字列	定義値[[,定義値]...] ※2	○

定義値の数がある形式

属性名	属性値 1	属性値 2	属性値 3	属性値 4 以降	CSV 編集
環境変数 ※1	表示文字列	ユーザ定義の環境変数名	定義値の数※3	定義値[[,定義値]...][,ユーザ定義の環境変数名,定義値の数,定義値[[,定義値]...]] ※3	○

(凡例)

○：編集できます。指定が任意の属性です。省略できます。

注※1

エントリ行は，複数個指定できます。

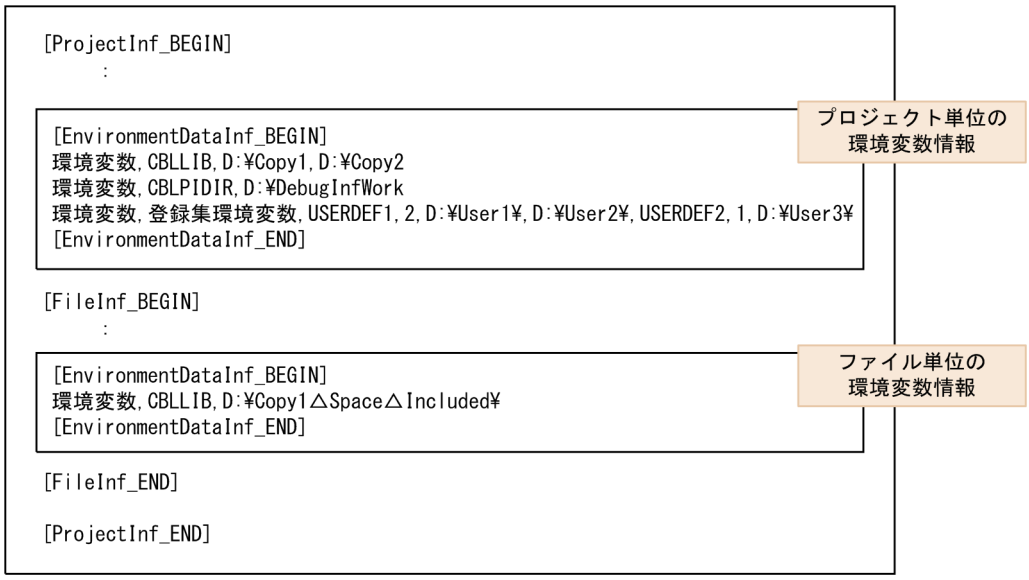
注※2

属性値 2 以降に定義値が出力されます。定義値と定義値の間はコンマ（,）で区切られます。

注※3

環境変数名を定義する環境変数で，定義値が可変個のものは，定義値の数が出力されます。このとき，定義値は属性値 4 以降に出力されます。定義値と定義値の間はコンマ（,）で区切られます。

開発マネージャによる出力例（指定例）



1.9.5　CSV ファイルに指定できるプロジェクトの種類

CSV ファイルに指定できるプロジェクトの種類を次の表に示します。

表 1-5　CSV ファイルに指定できるプロジェクトの種類

プロジェクトの種類	Windows(x86)	Windows(x64)
System 指定メインプログラム	○	○
V3 指定メインプログラム	○	○
メインプログラムなし	○	○
DLL プロジェクト ※	○	○
標準ライブラリ	○	○

（凡例）

○：CSV ファイルに指定できます。

注※

DLL プロジェクトを指定する場合は，次に示すオプションを指定してください。

Windows(x86) COBOL2002 のとき：「-Dll,Stdcall」または「-Dll,Cdecl」

Windows(x64) COBOL2002 のとき：「-Dll」

上記のオプションが指定されていない場合は，CSV ファイルの入力時にエラーになります。

1.9.6　CSV ファイルに指定できるファイルの種別

CSV ファイルに指定できるファイルの種別を次の表に示します。

表 1-6　CSV ファイルに指定できるファイルの種別

ファイルの種別	Windows(x86)	Windows(x64)
実行可能ファイル	○	○
DLL ファイル	○	○
モジュール定義ファイル	○	○
ライブラリファイル	○	○
オブジェクトファイル	○	○
アイコンファイル	○	○
一般ファイル	○	○
COBOL ソースファイル（固定形式）	○	○
COBOL ソースファイル（自由形式）	○	○
登録集原文（固定形式）	○	○

ファイルの種別	Windows(x86)	Windows(x64)
登録集原文（自由形式）	○	○
ファイル定義ファイル	○	○
レコード定義ファイル	○	○
画面定義ファイル	○	○
stdcall 呼び出し指示ファイル	○	－
実行環境ファイル	○	○
リポジトリファイル	○	○
マップ定義ファイル	○	－
HiRDB 埋め込み型 UAP	○	○
HiRDB 埋め込み型 UAP（OOCOBOL 移行用）	○	○
XML データ定義ファイル	○	○
XML 文書型定義ファイル	○	○
XML 文書型パラメタファイル	○	○
XML カタログファイル	○	○
COBOL ソース（固定形式）（XML）※1	○	○
COBOL ソース（固定形式）（HiRDB）※1	○	○
COBOL ソース（自由形式）（HiRDB）※1	○	○
単体テスト支援プロジェクトファイル※2	○	○

（凡例）

○：CSV ファイルに指定できます。

－：Windows(x64) COBOL2002 では指定できません。指定すると，一般ファイルとして登録されます。

注※1

COBOL2002 または関連プログラムプロダクトのプリプロセッサで生成される COBOL ソースを示します。

注※2

COBOL2002 Developer Professional の単体テスト支援を使用した場合に作成されます。

また，新しいファイルの種類の登録ダイアログボックスで追加したファイルの種別も指定できます。追加したファイルの種別の出力ファイルとして「COBOL ソース（固定形式)」または「COBOL ソース（自由形式)」を指定したとき，当該出力ファイルのファイルの種別は，「COBOL ソース（固定形式）(Custom：追加したファイルの種別のファイル表示名)」または「COBOL ソース（自由形式）(Custom：追加したファイルの種別のファイル表示名)」となります。

例

新しいファイルの種類の登録でファイル表示名「AAA」を追加し，出力ファイル種別を「COBOL ソース（固定形式）」としたとき，当該ファイルのファイルの種別は，「COBOL ソース固定形式（Custom:AAA）」と出力されます。

1.9.7　CSV ファイル入出力時のエラーチェック

CSV ファイルを入出力できないと，入出力に失敗したことを知らせるエラーメッセージがメッセージボックスに出力されます。

CSV ファイルの入出力時のエラーチェックには，CSV ファイルのアクセス権に関するチェックと，CSV ファイルの編集誤りのチェックがあります。

(1)　CSV ファイルのアクセス権に関するチェック

CSV ファイルのアクセス権に関するメッセージを次に示します。メッセージが表示された場合は，CSV ファイルのアクセス権を見直してください。

表 1-7　CSV ファイルのアクセス権に関するメッセージ

メッセージ	原因
"*CSV ファイルの絶対パス*"をオープンできません。	アクセス権がない CSV ファイルを入力した。
指定されたプロジェクトマスタ情報ファイルに書き込み権限がありません。	アクセス権がない CSV ファイルを出力ファイルに指定した。

(2)　CSV ファイルの内容に関するチェック

エラーチェックは CSV ファイルの先頭から開始し，最初のエラーを検出した時点で，エラーメッセージを表示して入力処理が終了します。

エラーチェックの項目を次に示します。

セクション

- セクション開始行キーワードとセクション終了行キーワードが対で指定されているか
- セクション間で指定されている順序が正しいか
- 指定されている数が正しいか
- セクション内のエントリ行に過不足はないか

エントリ行

- 属性名が正しいか
- 属性値が正しいか

CSV ファイルのエラーを検出した場合，メッセージボックスには次に示すエラーが表示されます。

プロジェクトマスタ情報ファイルの内容に誤りがあります。

CSVファイルの内容に関するメッセージ
“CSVファイルの絶対パス”，行番号：’エラー個所’

CSV ファイルの内容に関するメッセージを次に示します。メッセージが表示されたら，不具合を解消するように CSV ファイルを修正するか，CSV ファイルの出力からやり直してください。

表 1-8　CSV ファイルの内容に関するメッセージ（セクションに関するエラー）

メッセージ	原因
Section-name セクションがありません。※1※2	セクション開始行のキーワードがない。
[*Section-name*_END]がありません。※1※3	セクション終了行のキーワードがない。
セクション指定行が不正です。※2	セクション指定行がキーワードだけになっていない。
セクションの指定順序が不正です。※3	セクションの指定順序が間違っている。
Section-name セクションは 1 つだけ指定できます。※1※2	セクションの指定回数が規定数より多い。
エントリの指定が重複しています。※4	エントリ行が規定数より多い。
エントリの指定が不足しています。※3	エントリ行が規定数より少ない。

注※1

Section-name には，該当するセクション名が表示されます。

注※2

エラー個所には，エラー検出行の誤りの内容が表示されます。

（例）

セクション指定行が不正です。

"D:¥PM1¥PM1_Editing.CSV",line 2: 'AAA'

注※3

エラー個所には，該当するセクションのセクション開始行キーワードが表示されます。

注※4

エラー個所には，該当する属性名が表示されます。

表 1-9　CSV ファイルの内容に関するメッセージ（エントリ行に関するエラー）

メッセージ	原因
属性名が不正です。※1	属性名が間違っている。
エントリの指定が不正です。※2	指定が必須の属性値が指定されていない。
ファイル名の指定がない，無効な文字 ¥ / ： * ? " < > | @ を含むファイル名，ピリオドとスペースで終わるファイル名，または，256 文字以上のファイル名は，指定できません。※3	ファイル名やフォルダ名が間違っている。

メッセージ	原因
フォルダのパスの指定がないか，指定が 260 以上となっています。※3	フォルダのパスの指定がないか，260 文字以上指定されている。
属性値が不正です。※1	ファイル名やフォルダ名以外の属性値が間違っている。

注※1

エラー個所には，該当する属性名が表示されます。

注※2

エラー個所には，エラー検出行の誤りの内容が表示されます。

（例）

エントリの指定が不正です。

"D:¥PM1¥PM1_Editing.CSV",line 2: 'AAA'

注※3

エラー個所は表示されません。CSV ファイルの絶対パスと行番号だけが表示されます。

表 1-10　CSV ファイルの内容に関するメッセージ（セクションおよびエントリに共通するエラー）

メッセージ	原因
セクションまたはエントリが 1 けた目から始まっていません。※	1 けた目から指定されていない。

注※

メッセージに続けて，エラー検出行の誤りの内容が表示されます。

（例）

セクションまたはエントリが 1 けた目から始まっていません。

"D:¥PM1¥PM1_Editing.CSV",line 2: 'AAA'

1.10　構成管理ツールとの連携（COBOL2002 Developer Professional でだけ有効）

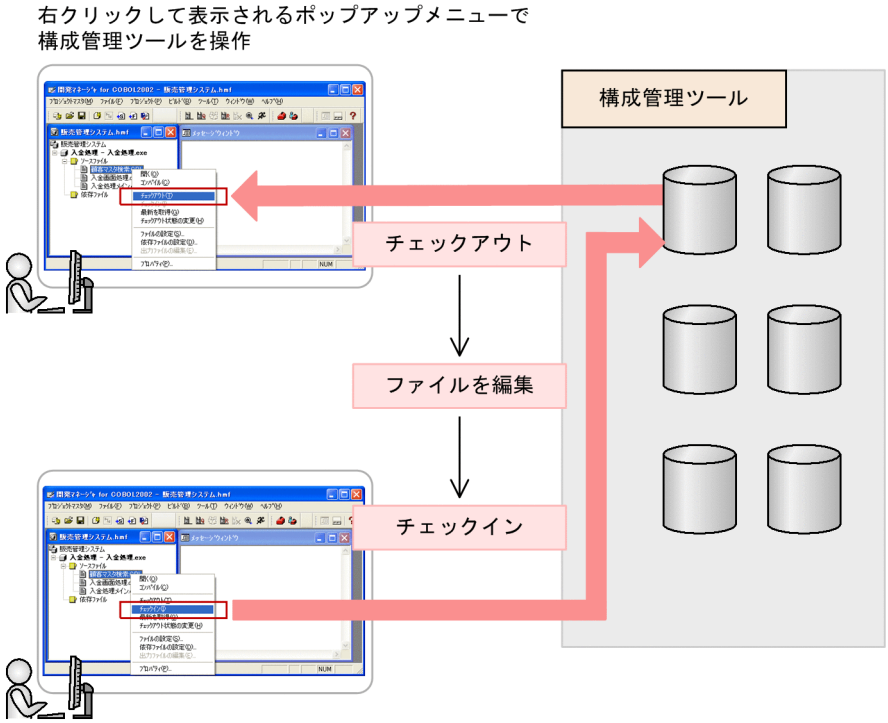
構成管理ツールとの連携は，COBOL2002 Developer Professional でだけ使用できます。COBOL2002 Net Developer だけインストールされている場合は，構成管理ツールを実行するメニューは表示されません。

プログラム開発チーム内で共通して使用するデータ（共通 COPY 文，共通サブルーチンなど）のバージョン管理の目的で，構成管理ツールを適用するケースがあります。開発マネージャと構成管理ツールを連携させると，開発マネージャ上で右クリックして表示されるポップアップメニュー，またはファイルメニューから該当するメニューを選んで構成管理ツールを操作でき，コマンドプロンプトなどから構成管理ツールを実行する手間が省けます。

構成管理ツールは，別途用意してください。

構成管理ツールとの連携を次の図に示します。

図 1-7　構成管理ツールとの連携



開発マネージャから実行できる構成管理ツールの機能は次のとおりです。

- チェックアウト
  該当するファイルを取り出します。
- チェックイン
  チェックアウトしたファイルを格納します。
- 最新を取得

最新バージョンのファイルを取得します。

- チェックアウト状態の変更

  フォルダ単位でチェックアウトするツールの場合に，構成管理ツールと開発マネージャの状態の同期を取るためにチェックアウト状態を変更します。

対象になるファイルは，開発マネージャのツリー上のノード（ソースファイルなど）下に表示されている，ソースファイルと依存ファイルです。

注意

- 事前に，使用する構成管理ツールの環境設定をしておいてください。
- 複数の開発マネージャを起動している場合，構成管理ツールの操作結果の情報は，設定した開発マネージャだけにしか反映されません。起動中のほかの開発マネージャは再起動してください。
- チェックアウトすると，チェックアウトの状態はプロジェクトマスタファイルに保存されます。構成管理ツールを設定していない環境で，チェックアウト状態のファイルを含むプロジェクトマスタファイルを開いた場合，メッセージウィンドウに警告メッセージが表示されます。チェックアウト中のファイルを含むプロジェクトマスタファイルをほかのマシンに移行しないでください。

1.10.1 構成管理ツールの仕様

構成管理ツールとは，プログラム開発チーム内で各種ファイルのバージョン管理に使用するプログラムのことです。

構成管理ツールとして登録できるプログラムは，次の条件をすべて満たしている必要があります。一つでも条件を満たしていないプログラムを構成管理ツールとして登録した場合，動作は保証しません。

- コンソールプログラム（Windows CUI プログラム）として呼び出せるインタフェースがある。
- コマンドラインから処理対象となるファイル名を引数として指定できる。
- 入力元のファイル，および生成される出力ファイルが，それぞれ一つである。
- メッセージは，すべて標準出力，または標準エラー出力に出力される。また，応答を求めるメッセージは出力されない。
- 処理の結果をリターンコードで判定できる。また，正常終了したときのリターンコードが 0 であり，0 以外は異常である。

1.10.2 構成管理ツールの操作

開発マネージャと構成管理ツールを連携させるため，構成管理ツールのコマンドラインを登録します。

プロジェクトマスタファイルを開いていると実行できません。プロジェクトマスタファイルを閉じてから実行してください。

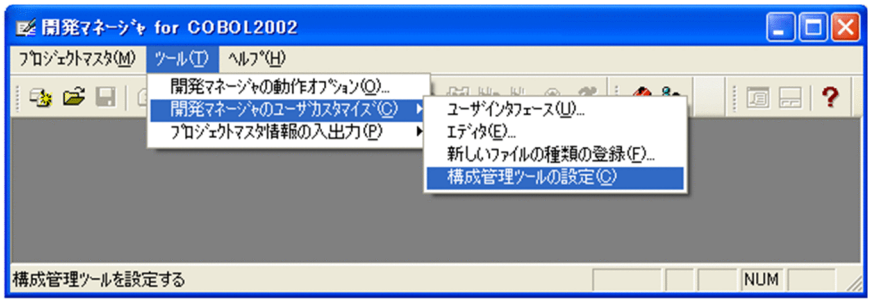
(1)　事前準備

開発チーム内の管理者が，連携する構成管理ツールのドキュメントでコマンドラインの仕様を調査し，要件を満たしているかどうかを確認してください。さらに，構成管理ツールを使用する上で必要な環境を設定してください。

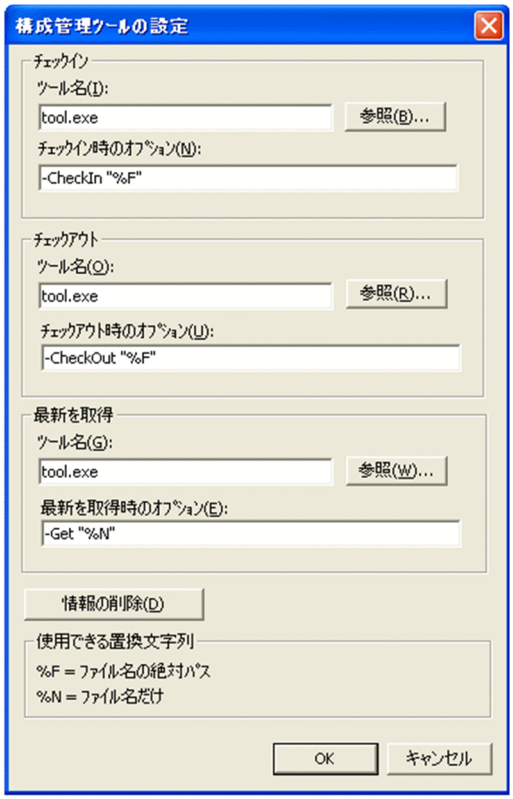
(2)　構成管理ツールの設定

開発チーム内の任意のマシンで，開発マネージャに構成管理ツールのコマンドラインを設定します。

1. 開発マネージャのプロジェクトマスタファイルを閉じている状態で，［ツール］－［開発マネージャのユーザカスタマイズ］－［構成管理ツールの設定］メニューを選択する。



2. ［構成管理ツールの設定］ダイアログボックスに，開発マネージャと連携する構成管理ツールとオプションを指定して［OK］ボタンを選ぶ。



「チェックアウト」「チェックイン」「最新を取得」で使用する構成管理ツールをそれぞれ指定します。

［構成管理ツールの設定］ダイアログボックスに入力する値については，「1.10.3　構成管理ツールの設定ダイアログボックス」を参照してください。

(3)　開発チーム内で同じ環境で作業をするためにパッケージ情報ファイルを配布する

(2) で設定した情報は，パッケージ情報ファイルとして出力されます。パッケージ情報ファイルをチーム内の担当者に配布します。

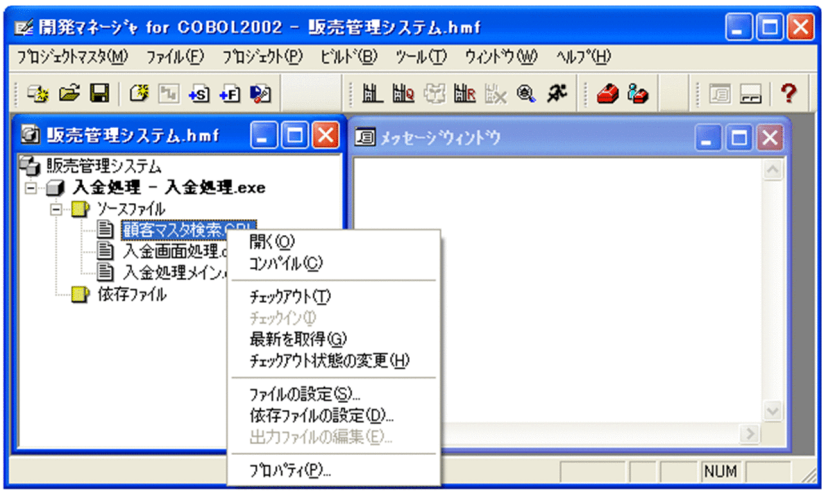
1. 構成管理ツールを設定したマシンのパッケージ情報ファイル「_CNFGMNG.PKG」を次の格納場所からコピーする。

   環境変数%ALLUSERSPROFILE%が示すフォルダ¥Hitachi¥COBOL2002¥PLUGIN

2. 開発チーム内の担当者の各マシンに，パッケージ情報ファイルを同じ場所に格納する。

(4)　開発チーム内でファイルを共有して開発

開発チーム内の各担当者が，それぞれファイルを編集します。



［チェックアウト］

該当するファイルを構成管理ツールから取り出します。複数プロジェクトで同一ファイルがある場合に，どれか一つのプロジェクトで任意のファイルをチェックアウトすると，ツリービュー上のアイコンは該当プロジェクトのファイルだけがチェックアウトのアイコンに切り替わります。

［チェックイン］

構成管理ツールにファイルを渡します。

［最新を取得］

最新のファイルを構成管理ツールから取り出します。

[チェックアウト状態の変更]

構成管理ツールと開発マネージャの状態の同期を取るために，コマンドラインを実行することなく，チェックアウト状態を変更します。チェックアウトされていない場合はチェックアウト状態に，チェックアウトされている場合はチェックアウトされていない状態に変更します。

コマンドラインを実行する際，開始，終了のメッセージと実行しているコマンドラインはメッセージウィンドウに表示されます。構成管理ツールが出力するメッセージは，メッセージウィンドウに表示されます。メッセージの表示を次に示します。

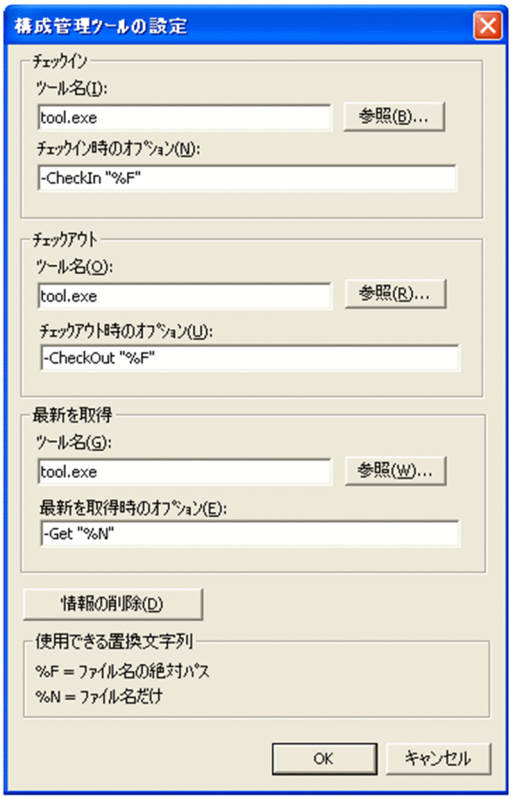
```
=== "ファイル名" のﾁｪｯｸｲﾝを開始します。===
"実行しているコマンドライン"
"構成管理ツールが出力したメッセージ"
=== "ファイル名" のﾁｪｯｸｲﾝ処理が終了しました。===

=== "ファイル名" のﾁｪｯｸｲﾝを開始します。===
"実行しているコマンドライン"
"構成管理ツールが出力したエラーメッセージ"
=== "ファイル名" のﾁｪｯｸｲﾝ処理でｴﾗｰが発生しました。===
```

ファイルを複数選択している場合，選択しているファイルの分だけ，コマンドラインの実行を繰り返します。

1.10.3　構成管理ツールの設定ダイアログボックス

構成管理ツールの設定ダイアログボックスを次に示します。



(1) ダイアログボックスの内容

各テキストボックスの初期値はすべて空です。情報を設定した状態でこのダイアログボックスを開くと，設定内容が表示されます。

ツール名テキストボックス

連携したい構成管理ツールを絶対パスまたは相対パスで指定します。該当するコマンドがない場合は，空白のままでもかまいません。

チェックイン時のオプションテキストボックス

チェックイン時にツールに指定するオプションを指定します。

チェックアウト時のオプションテキストボックス

チェックアウト時にツールに指定するオプションを指定します。

最新を取得時のオプションテキストボックス

最新を取得時にツールに指定するオプションを指定します。

実行例

上記のダイアログボックスどおりに設定した場合，ツリービュー上で選択しているファイルが「C:¥temp¥a.cbl」のときに，ファイルメニューから該当するメニューを実行すると，次に示すコマンドが実行されます。

「チェックイン」メニュー

```
tool.exe -CheckIn "C:¥temp¥a.cbl"
```

「チェックアウト」メニュー

```
tool.exe -CheckOut "C:¥temp¥a.cbl"
```

「最新を取得」メニュー

```
tool.exe -Get "a.cbl"
```

［情報の削除］ボタン

設定情報を保存しているパッケージ情報ファイルを削除します。［情報の削除］ボタンを選ぶと，各テキストボックスは空になります。

削除できなかった場合は，エラーメッセージが表示されます。削除に失敗する要因には，次のことが考えられます。

- 構成管理ツールのパッケージ情報ファイルまたはフォルダにアクセスする権限がない
- 別の開発マネージャで構成管理ツールのパッケージ情報ファイルにアクセスしている
- ダイアログボックスを表示した状態で，別の開発マネージャやエクスプローラなどから構成管理ツールのパッケージ情報ファイルを削除した

［OK］ボタン

［OK］ボタンを選ぶと，設定情報をパッケージ情報ファイルに出力します。すでにファイルが存在していた（情報を設定していた）場合には，すでに構成管理ツールのパッケージ情報ファイルが存在することを示す警告メッセージが表示されます。［はい］を選ぶと，パッケージ情報ファイルが上書きされます。

パッケージ情報ファイルへの出力が失敗した場合は，エラーメッセージが表示されます。出力に失敗する要因には，次のことが考えられます。

- 構成管理ツールのパッケージ情報ファイルまたはフォルダにアクセスする権限がない
- 別の開発マネージャで構成管理ツールのパッケージ情報ファイルにアクセスしている

(2) 使用できる置換文字列

オプションには，"%F"を含む文字列を指定する必要があります。「%英字 1 文字」を置換文字列と認識されます。

(置換文字列の種類)

指定できる置換文字列を次に示します。

%F

ツリー上で選択しているファイル名（ドライブ名から始まる絶対パス）に置き換えられます。

%N

ツリー上で選択しているファイル名（ファイル名だけ）に置き換えられます。

(オプションの指定で使用する文字列についての注意事項)

オプションの文字列に「%」を使用したい場合は，「%%」と二つ続けて記述してください。

(例)

"%F" … 置換文字列とみなされ，編集対象となるファイル名に置換されます。

"%%F" … 「%F」という文字列とみなされます。

(3) お勧めする使用方法

テキストボックスにはファイル名を直接入力できますが，基本的な操作として，開発マネージャに登録しているファイル自体をチェックアウトの対象とすることをお勧めします。

(4) 構成管理ツールのコマンドラインが設定されている場合

構成管理ツールのコマンドラインが設定されている（パッケージ情報ファイルが存在する）場合，構成管理ツールの設定ダイアログボックスは，各情報をテキストボックスに設定した状態で表示されます。パッケージ情報ファイルが存在する場合で入力ができなかったとき，エラーメッセージが表示されます。

(5) コマンドラインおよびオプションを設定するときの注意事項

- 「チェックイン」「チェックアウト」「最新を取得」をメニューで選択した場合，設定されたコマンドラインは，プロジェクトフォルダをカレントフォルダとして実行します。ファイルのパスが指定できないコマンドラインの場合は，対象とするファイルは，プロジェクトフォルダに格納してください。
- フォルダ単位でチェックアウトするツールの場合，「チェックアウト状態の変更」メニューを併用して，構成管理ツールと開発マネージャの状態の同期を取ってください。

- チェックアウトするフォルダを開発マネージャ上のファイルのパスと異なる場所にしたい場合は，置換文字列を使用してください。
- 実行後に応答が求められるコマンドラインまたはオプションを指定する場合は，応答の要求を抑止するオプションを一緒に指定してください。

1.11 外部ツールとの連携（COBOL2002 Developer Professional でだけ有効）

外部ツールとの連携は，COBOL2002 Developer Professional でだけ使用できます。COBOL2002 Net Developer だけインストールされている場合は，外部ツールとの連携は設定できません。

開発マネージャに登録しているファイルを対象にして，任意の外部ツールのコマンドを実行できます。開発マネージャ上で使用したいツールのコマンドライン，およびメニューの表示文字列を登録すると，メニューが追加されます。メニューを選択すると，登録したコマンドラインを実行します。また，登録した情報は，パッケージ情報ファイルとして保存し，ほかのマシンにも移行できます。

注意

- 複数の開発マネージャを起動している場合，外部ツールの操作結果の情報は，設定した開発マネージャだけにしか反映されません。起動中のほかの開発マネージャは再起動してください。

1.11.1 連携できる外部ツールの仕様

開発マネージャに登録できる外部ツールの仕様を次に示します。

これらの条件を一つでも満たしていないプログラムを外部ツールとして登録した場合，動作は保証しません。

ウィンドウプログラムの場合

- コマンドラインでの操作をサポートしている。
- GUI を持つウィンドウプログラムである。
- コマンドラインの引数としてファイル名を指定できる。

コンソールプログラムの場合

- コマンドラインでの操作をサポートしている。
- コンソールプログラムである。
- コマンドラインの引数としてファイル名を指定できる。
- エラーが発生してもユーザ応答を求めない。
  ただし，コマンドライン，またはオプションがユーザ応答を求める場合，ユーザ応答を求めないようにできるオプションがあれば同時に指定することで適用できる。
- 出力するメッセージは，標準出力または標準エラーに出力する。

1.11.2 外部ツールの操作

開発マネージャと外部ツールを連携させるため，外部ツールのコマンドラインを登録します。

プロジェクトマスタファイルを開いていると実行できません。プロジェクトマスタファイルを閉じてから実行してください。

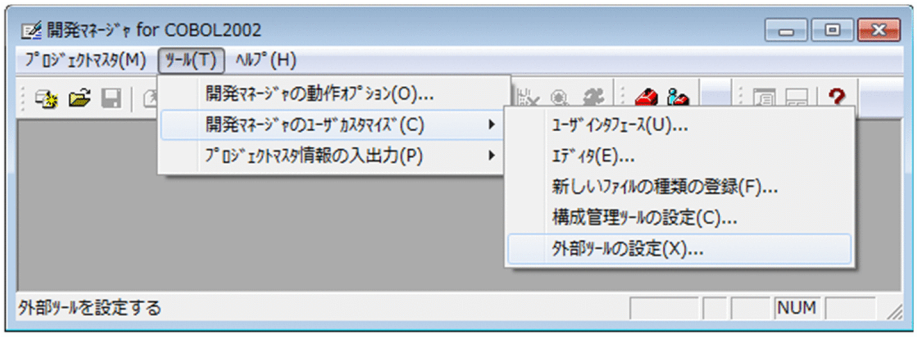
(1)　事前準備

開発チーム内の管理者が，連携する外部ツールのドキュメントでコマンドラインの仕様を調査し，要件を満たしているかどうかを確認してください。さらに，外部ツールを使用する上で必要な環境を設定してください。

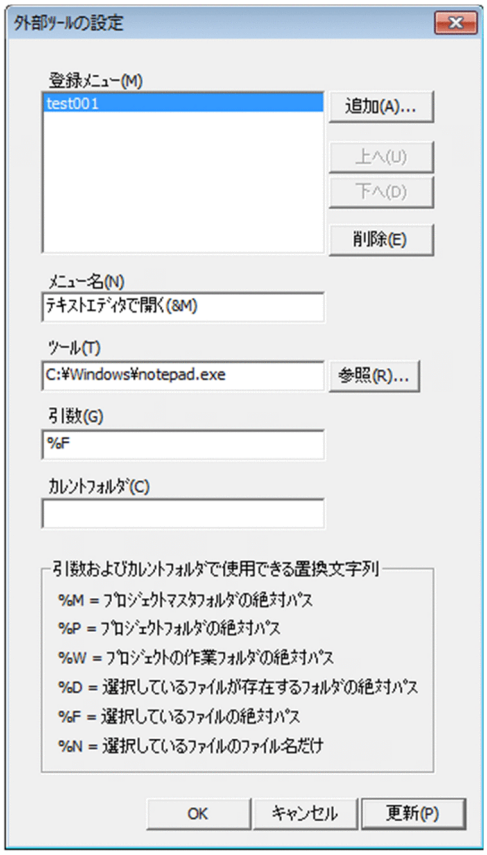
(2)　外部ツールの設定

開発チーム内の任意のマシンで，開発マネージャに外部ツールのコマンドラインを設定します。

1. 開発マネージャのプロジェクトマスタファイルを閉じている状態で，［ツール］－［開発マネージャのユーザカスタマイズ］－［外部ツールの設定］メニューを選択する。



2. ［外部ツールの設定］ダイアログボックスに，開発マネージャと連携する外部ツールとメニューの表示情報を指定して［OK］ボタンを選ぶ。



メニューの表示情報は，複数登録でき，［登録メニュー］にリスト表示されます。

［外部ツールの設定］ダイアログボックスに入力する値については，「1.11.3　外部ツールの設定ダイアログボックス」を参照してください。

(3)　開発チーム内で同じ環境で作業をするためにパッケージ情報ファイルを配布する

(2) で設定した情報は，パッケージ情報ファイルとして出力されます。パッケージ情報ファイルをチーム内の担当者に配布します。

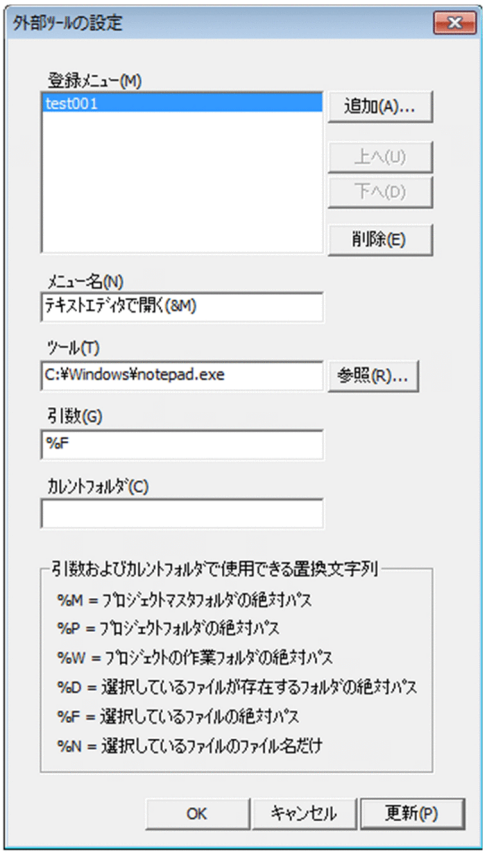
1. 外部ツールを設定したマシンのパッケージ情報ファイル「_FILEMENU.PKG」を次の格納場所からコピーする。

   環境変数%ALLUSERSPROFILE%が示すフォルダ¥Hitachi¥COBOL2002¥PLUGIN

2. 開発チーム内の担当者の各マシンに，パッケージ情報ファイルを同じ場所に格納する。

1.11.3　外部ツールの設定ダイアログボックス

外部ツールの設定ダイアログボックスを次に示します。



(1) ダイアログボックスの内容

各テキストボックスの初期値はすべて空です。情報を設定した状態でこのダイアログボックスを開くと，設定内容が表示されます。

登録メニューリストボックス

メニューとツールの情報を識別する識別名称の一覧です。

登録できる情報は，20 個までです。

このリストボックス上で識別名を選択すると，メニュー名，ツール，引数，およびカレントフォルダテキストボックスに情報が表示されます。情報を変更するには，各テキストボックスで内容を変更し，［更新］ボタンを選んでください。情報の追加，変更の途中で登録メニューの選択を変更すると，変更を保存するかを問い合わせるメッセージボックスが表示されます。なお，同時に複数のメニューを選択できません。

このリストボックスの順番で，［ファイル］メニューに表示されます。

［追加］ボタン

外部ツールと，外部ツールを呼び出すメニューの情報を新規で追加します。

［追加］ボタンを選ぶと，識別名を指定するテキストボックスだけのダイアログボックスが表示され，テキストボックスに識別名を入力できます。入力した識別名は，登録メニューリストボックスのいちばん下に追加されます。

識別名は，英字で始まる半角英数字 8 文字以内で入力してください。

［上へ］ボタン，［下へ］ボタン

登録メニューリストボックス内で，選択中の識別名の順番を変更します。

［削除］ボタン

登録メニューリストボックス内で，選択中の識別名を削除します。

［削除］ボタンを選ぶと，削除してよいか問い合わせるメッセージボックスが表示されます。

メニュー名テキストボックス

メニュー名として表示する文字列（アクセスキーを含む）を指定します。指定できる最大文字数は，32バイト（2バイト文字で16文字）です。

アクセスキーにしたい文字があるときは，その文字の直前にアンパサンド「&」を指定して，前後を丸括弧で囲んでください。アクセスキーに指定できる文字は半角英数字だけです。なお，アクセスキーが既存の開発マネージャのメニューと重複しているかどうかのチェックは実行されません。重複していると，正しくメニューを選択できません。

ここで指定したメニューは，［ファイル］メニューに表示されます。

必須項目のため，未入力の状態で［更新］ボタン，または［OK］ボタンを選ぶと，エラーメッセージが表示されます。

ツールテキストボックス

開発マネージャから起動したいツールを指定します。絶対パス，相対パスのどちらでも指定できます。相対パスを指定して［更新］ボタンをクリックすると，自動的に絶対パスに置き換わります。

［参照］ボタンを選ぶと，ファイルを選択できます。

ツールテキストボックスに指定できるのは，exeファイルだけです。

存在しないパスなどの不正パスを入力した場合，または未入力の状態で［更新］ボタン，または［OK］ボタンを選ぶと，エラーメッセージが表示されます。

引数テキストボックス

外部ツールに指定する引数を指定します。置換文字列を使用できます。

入力を省略できます。

カレントフォルダテキストボックス

外部ツールが起動する際のカレントフォルダを指定します。置換文字列を使用できます。

省略すると，プロジェクトフォルダが仮定されます。

［OK］ボタン

［OK］ボタンを選ぶと，設定情報をパッケージ情報ファイルに出力します。

すでにファイルが存在していた（情報を設定していた）場合には，すでに外部ツールのパッケージ情報ファイルが存在することを示す警告メッセージが表示されます。［はい］を選ぶと，パッケージ情報ファイルが上書きされます。

パッケージ情報ファイルへの出力が失敗した場合は，エラーメッセージが表示されます。出力に失敗する要因には，次のことが考えられます。

- 外部ツールのパッケージ情報ファイルまたはフォルダにアクセスする権限がない
- 別の開発マネージャで外部ツールのパッケージ情報ファイルにアクセスしている

［登録メニュー］に情報がない状態で［OK］ボタンを選ぶと，パッケージ情報ファイルが削除されます。

削除できなかった場合は，エラーメッセージが表示されます。削除に失敗する要因には，次のことが考えられます。

- 外部ツールのパッケージ情報ファイルまたはフォルダにアクセスする権限がない
- 別の開発マネージャで外部ツールのパッケージ情報ファイルにアクセスしている
- 外部ツールの設定ダイアログボックスを表示した状態で，別の開発マネージャやエクスプローラなどから外部ツールのパッケージ情報ファイルを削除した

［更新］ボタン

一つの識別名単位で情報の変更を確定します。

［更新］ボタンを選ぶと，追加した情報，または変更中の情報のエラーチェックが実行され，問題がなければ変更が確定します。

登録メニューを新規で追加した場合，または設定した内容を変更した場合に［更新］ボタンを選ばないで次に示す操作をすると，変更を保存するかを問い合わせるメッセージが表示されます。メッセージに従って操作してください。

- リストボックスの選択を変更
- ［追加］ボタンを選択
- ［上へ］ボタンを選択
- ［下へ］ボタンを選択
- ［OK］ボタンを選択

(2) 使用できる置換文字列

引数テキストボックスとカレントフォルダテキストボックスでは，置換文字列を指定できます。

(置換文字列の種類)

指定できる置換文字列を次に示します。

%M

プロジェクトマスタフォルダの絶対パスに置き換えられます。

%P

プロジェクトフォルダの絶対パスに置き換えられます。

%W

プロジェクトの作業フォルダの絶対パスに置き換えられます。

%D

ツリー上で選択しているファイルが存在するフォルダの絶対パスに置き換えられます。

%F

ツリー上で選択しているファイル名（ドライブ名から始まる絶対パス）に置き換えられます。

%N

ツリー上で選択しているファイル名（ファイル名だけ）に置き換えられます。

（オプションの指定で使用する文字列についての注意事項）

オプションの文字列に「%」を使用したい場合は，「%%」と二つ続けて記述してください。

（例）

"%F" … 置換文字列とみなされ，編集対象となるファイル名に置換されます。

"%%F" … 「%F」という文字列とみなされます。

（置換文字列の例）

次に示すプロジェクトで，ツリービュー上で選択しているファイルが a.cbl の場合

種類	パス
プロジェクトマスタフォルダ	c:¥ te△mp¥ProjectMaster
プロジェクトフォルダ	c:¥ te△mp¥ProjectMaster¥Project
プロジェクト作業フォルダ	c:¥ te△mp¥ProjectMaster¥Project¥Work
ソースファイルの存在場所	c:¥ te△mp¥ProjectMaster¥Project¥Source
ソースファイルの絶対パス	c:¥ te△mp¥ProjectMaster¥Project¥Source¥a.cbl

置換文字列と置換後の文字列の関係を次に示します。

置換文字列	置換後の文字列
%M	c:¥ te△mp¥ProjectMaster¥
"%M"	"c:¥ te△mp¥ProjectMaster¥"
%P	c:¥ te△mp¥ProjectMaster¥Project¥
%W	c:¥ te△mp¥ProjectMaster¥Project¥Work¥
%D	c:¥ te△mp¥ProjectMaster¥Project¥Source¥
%F	c:¥ te△mp¥ProjectMaster¥Project¥Source¥a.cbl
%N	a
"%D%N.txt"	"c:¥ te△mp¥ProjectMaster¥Project¥Source¥a.txt"

2 コンパイル

COBOL2002 では，コンパイル環境を設定してプログラムをコンパイルします。この章では，これらの設定方法，コンパイルするためのビルドやリビルドの方法について説明します。

2.1　概要

COBOL2002 によるプログラムのコンパイルには，次の四つの方法があります。

- ビルド
- クイックビルド
- リビルド
- ファイル単位のコンパイル

2.1.1　ビルドとは

ビルドとは，プロジェクトに登録したファイルをすべてコンパイルする方法です。

プロジェクトにファイル定義ファイル，レコード定義ファイル，HTML ファイルなど，ソースファイルや登録集原文を出力するファイルを持つファイルが登録されている場合は，ファイルからソース（登録集原文）が生成されてからコンパイルされます。

最終生成ファイル（実行可能ファイル，DLL，標準ライブラリなど）が未作成の場合は，すべてのプログラムがコンパイル，リンケージされ，最終生成ファイルが作成されます。また，ビルド対象のプロジェクトに子プロジェクトがある場合，必要に応じて子プロジェクトもビルドされます。

最終生成ファイル作成後にソースファイルや依存ファイルなどを追加・変更した場合は，コンパイルが必要なファイルだけを再コンパイルし，オブジェクトファイルを作り直します。このとき，オブジェクト指向機能または利用者定義関数を使用した COBOL プログラムの場合には，ファイル内容を解析して依存関係を再構築したあと，その依存関係に従ってコンパイルが必要なファイルだけを再コンパイルし，オブジェクトファイルを作り直します。その後，既存のオブジェクトファイルも含めて再リンケージし，最終生成ファイルを更新します。

2.1.2　クイックビルドとは

クイックビルドとは，オブジェクト指向機能または利用者定義関数を使用したプログラムのソースファイルを含むプロジェクトの場合に，ファイルの依存関係情報を再構築しないでビルドすることです。

ビルドに比べて依存関係再構築の時間を短縮できますが，依存関係に変更があった場合に正しくビルドできないことがあります。

2.1.3　リビルドとは

リビルドとは，ビルドと同様にプロジェクトに登録したファイルをすべてコンパイルする方法です。

ビルドと異なり，プログラムやコンパイラオプションなどに変更がなくても，無条件にすべてのファイルをコンパイル，リンケージし，最終生成ファイルを出力します。

2.1.4　ファイル単位のコンパイルとは

ファイル単位のコンパイルとは，ファイル単位にソース生成やコンパイルだけをする方法です。定義ファイルや HTML ファイルなどの出力ファイルが指定されているファイルをコンパイルするとソース（登録集原文）が生成されます。コンパイルするファイルに依存するファイルがあっても，そのファイルはコンパイルやソース生成の対象になりません。

COBOL ソースの場合は，ファイル単位のコンパイルによってオブジェクトファイルが作成されますが，リンケージはされません。最終生成ファイルを生成するには，ビルドまたはリビルドを実行する必要があります。

したがって，ファイル単位のコンパイルは，ファイルごとにコンパイルしてコンパイルエラーの有無を確認したり，定義ファイルからソースを生成したりするような場合に使用します。

2.2　コンパイラオプションの設定

この節では，コンパイラオプションの設定方法と各オプションの内容について説明します。

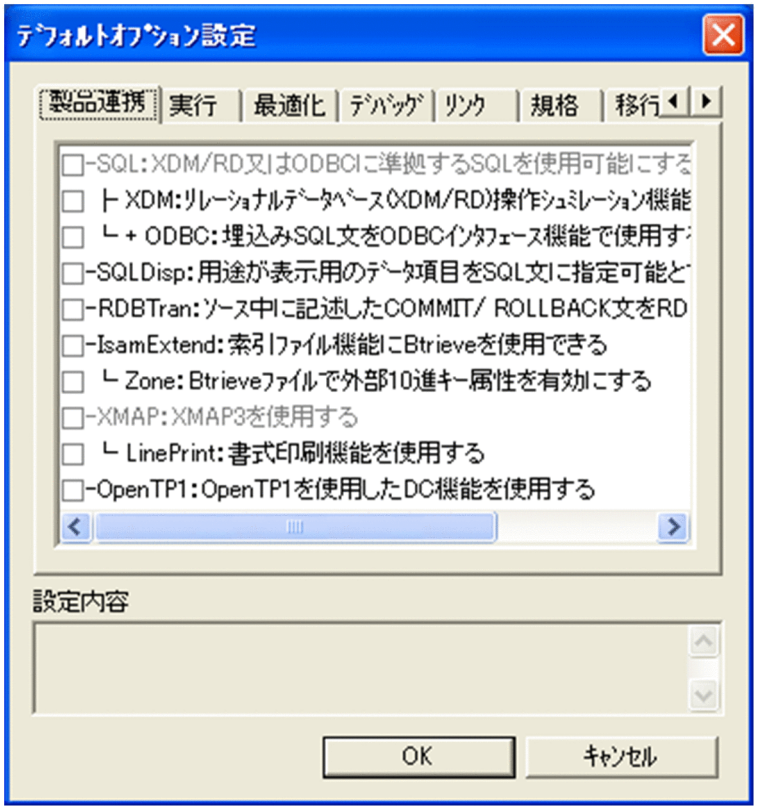
2.2.1　デフォルトオプションの設定

開発マネージャでは，プロジェクトを新規作成したときにデフォルトで設定するオプションをあらかじめ登録できます。

デフォルトオプションは，デフォルトオプション設定ダイアログボックスで設定します。ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのプロジェクトマスタを閉じた状態で，ウィンドウの［プロジェクトマスタ］メニューから［デフォルトオプションの設定］を選ぶ。

   デフォルトオプション設定ダイアログボックスが表示されます。



2.2.2　プロジェクト単位，またはファイル単位のオプションの設定

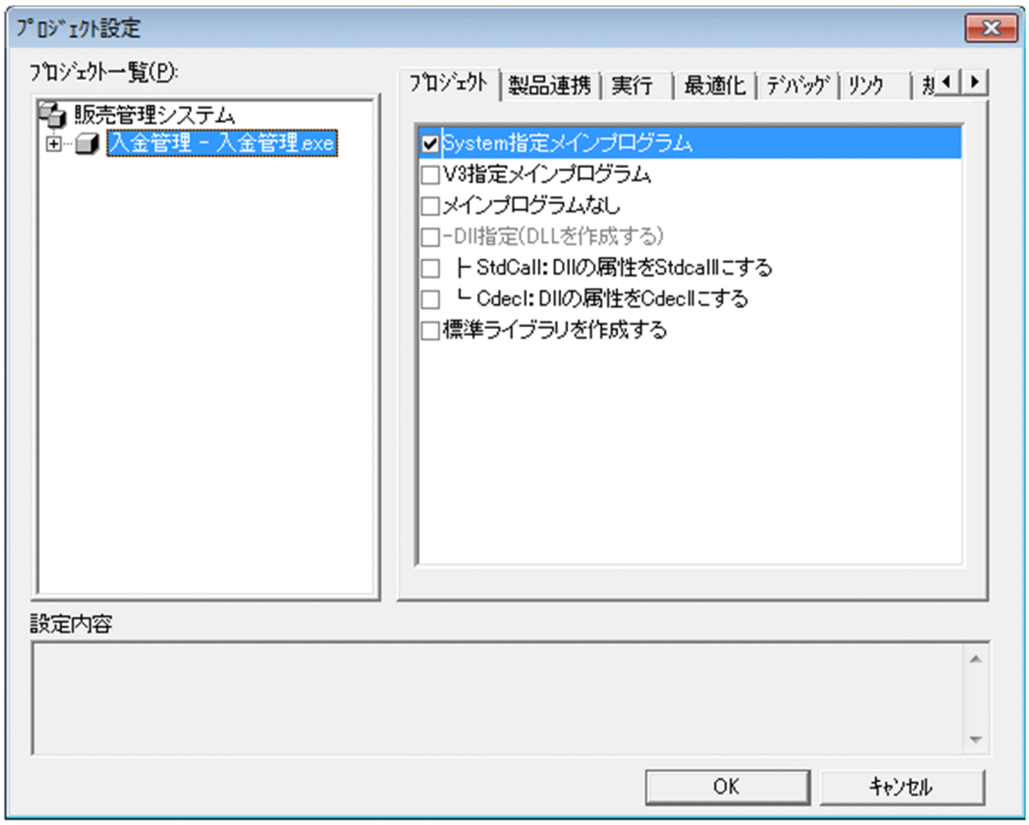
プロジェクト単位，またはソースファイル単位でコンパイラオプションを設定できます。

プロジェクト単位，またはソースファイル単位のコンパイラオプションの設定は，プロジェクト設定ダイアログボックスで設定します。ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのツリービューウィンドウから，コンパイラオプションを設定するプロジェクト，またはソースファイルを選ぶ。

2. [プロジェクト] メニューから［プロジェクトの設定］を選ぶ。または，［ファイル］メニューから［ファイルの設定］を選ぶ。

   プロジェクト設定ダイアログボックスが表示されます。



2.2.3　オプションの設定方法

デフォルトオプション設定ダイアログボックス，またはプロジェクト設定ダイアログボックスでコンパイル環境を設定する方法を次に示します。

コンパイラオプションの設定方法には，選択する方法と直接入力する方法の二つがあります。

選択する方法

プロジェクト設定ダイアログボックスの［プロジェクト］［製品連携］［実行］［最適化］［デバッグ］［リンク］［規格］［移行］［リスト出力］［その他］の各タブを選ぶと表示されるページのオプション一覧から，指定したいオプションを選びます。

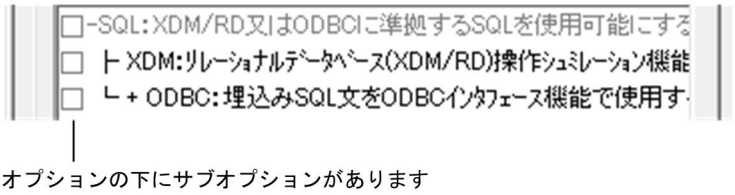
オプションの指定

1. 指定するオプションのチェックボックスを選ぶ。

   オプションが指定され，チェックボックスがオンになります。

なお，オプションに引数が必要な場合や，「メインファイルを指定する」オプションのようにファイル名を指定する必要がある場合は，引数を指定する必要があります。引数は，テキストボックス，またはオプションの引数の設定ダイアログボックスで指定します。

- オプションの左側に「+」が表示されている場合，オプションの下にサブオプションがあることを示しています。



サブオプションのチェックボックスをオンにした場合，サブオプションが含まれるオプションのチェックボックスもオンになります。

なお，サブオプションの下にサブオプションがある場合は，サブオプションのチェックボックスがオンになったときだけ表示されます。

オプションの引数の変更

1. 引数を変更するオプションのチェックボックスを選ぶ。

   現在のオプションの引数が，テキストボックス，またはオプションの引数の設定ダイアログボックスに表示されます。

2. 引数を変更する。

オプションの解除

1. 解除するオプションのチェックボックスを選ぶ。

   オプションが解除され，チェックボックスがオフになります。

   なお，オプションに引数が必要な場合や，「メインファイルを指定する」オプションのようにファイル名を指定する必要があるオプションの場合は，引数をすべて削除するとオプションが解除されます。

デフォルト値が設定されていないオプションの選択

SQL オプションなど，デフォルト値が設定されていないオプションは，サブオプションを先に選択してからでないと選択できません。

-Lib,GUI，-Optimize,1 のように，デフォルト値が設定されているオプションは，-Lib を押すと自動的にデフォルト値が設定されます。

オプションが設定されると，オプション名が設定内容の欄に表示されます。ただし，［プロジェクト］タブの「System 指定メインプログラム」「V3 指定メインプログラム」「メインプログラムなし」「標準ライブラリを作成する」，［リンク］タブの「ライブラリの指定」「リンケージを行わない」，［実行］タブの「メインファイルを指定する」を選んだ場合は，表示されません。また，［ユーザ設定］タブを選んで表示されるページでは，設定内容の欄は表示されません。

補足事項

プロジェクト設定ダイアログボックスの［ユーザ設定］タブを選び，コンパイラオプションの欄にコンパイラオプションを入力して［Enter］キー，または［Tab］キーを押すことで，直接オプションを入力できます。

複数のコンパイラオプションを指定する場合は，空白文字で区切って指定してください。また，［Ctrl］＋［Enter］キーで改行を入力してオプションを区切ることもできます。

ただし，この方法でオプションを入力した場合，オプションの背反や妥当性はチェックされません。

2.2.4　注意事項

- プロジェクト設定ダイアログボックスで指定したコンパイラオプションは，オプションのチェックボックスをオンにした順序でコンパイラに引き渡されます。コンパイラオプションの指定順序を意識する必要がある場合は注意してください。

  なお，コンパイラに引き渡されるオプションの順序は，プロジェクト設定ダイアログボックスの設定内容欄で確認できます。

- ソースファイルに設定したコンパイラオプションは，コンパイル方法（ビルド・リビルド・ファイル単位のコンパイルなど）に関係なく，設定したファイルで有効となります。

- プロジェクトに指定したオプションとソースファイルに指定したオプションでは，プロジェクトに指定したオプション，ソースファイルに指定したオプションの順でコンパイラに引き渡されます。背反となるオプションなどが指定されていた場合は，個々のオプションの仕様に依存します。詳細は，「2.3.3 コンパイラオプションの優先順位」の「(3)　同じオプションを重複して指定した場合の規則」－「(a) 背反関係にあるサブオプション同士を指定した場合」を参照してください。

- ソースファイルに指定したオプションに対して，プロジェクト設定ダイアログボックスの［ユーザ設定］タブにプリフィックス'no'の付いたコンパイラオプションを指定しても有効になりません。

- 背反となっているオプションの一方のチェックボックスをオンにすると，もう一方のオプションのチェックボックスはオフになります。また，オプションが背反する場合，一方のオプションに属するサブオプションのチェックボックスをオンにしても，もう一方のオプションのチェックボックスはオフになります。

- ソースファイルに設定したオプションは，リンク時には適用されません。

- プロジェクト単位で設定できるコンパイラオプションと，ファイル単位で設定できるコンパイラオプションには相違があります。詳細は，「2.3　コンパイラオプション」を参照してください。

2.3 コンパイラオプション

ここでは，ccbl2002 コマンドおよび開発マネージャに指定するコンパイラオプションについて説明します。詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「COBOL85 と COBOL2002 のコンパイラオプションの対応」を参照してください。

コンパイラオプションの表記について

これ以降のコンパイラオプションの説明では，次の表記を使用します。

- 「XXX オプション」，または単に「XXX」とオプション名を記載した場合
  XXX オプションについて，サブオプションの組み合わせを含む，すべての場合を指します。

（例）

「-SQL オプション」または「-SQL」と記載している場合，-SQL オプションのサブオプションの組み合わせすべて（-SQL,XDM／-SQL,ODBC／-SQL,ODBC,NoCont）のことを示します。

2.3.1 構文規則

コンパイラオプションの構文規則について説明します。

(1) オプション指定項

コマンドライン上で，コンパイラオプションを指定する部分をオプション指定項と呼びます。

次の例の下線部分は，それぞれオプション指定項です。

```
ccbl2002 -Option,SubOption -AnotherOption filename.cbl
```

(2) オプション指定項の構成

一つのオプション指定項を構成する要素，および構成要素の名称を次に示します。

```
-Option,SubOption Argument
   1.       2.        3.
```

- 1.をオプション名と呼びます。
- 2.をサブオプション名と呼びます。
- 3.をオプションの引数と呼びます。上記の例では，サブオプション名に引数が従属していますが，オプション名に直接引数が従属することもあります。
- 2.と 3.を合わせて，サブオプションと呼びます。
- 1.～3.を合わせて，オプションと呼びます。（オプションは，一つのオプション指定項全体を表します）

規則

- あるオプション Option に従属するオプションのことを「Option のサブオプション」と呼びます。また，説明の対象となるサブオプションがどのオプションに従属するか明確な場合には，主従関係を特定しないで単に「サブオプション」と呼びます。
- SubOption が Option のサブオプションである場合，構文規則およびオプション指定項では，次のように記述します。

  -Option,SubOption

  また，複数のサブオプションを持つオプションは，次のように記述します。

  -Option,SubOption1,SubOption2
- オプションとサブオプションの区切りには，コンマ（,）を使用します。また，サブオプション同士の区切りにもコンマを使用します。
- オプションには，サブオプションのほかに，ファイル名や数値などの値を引数として持つものもあります。

  オプション Option が引数 Argument を持つ場合は，次のように記述します。

  -Option Argument
- 引数は，オプション一つにつき一つだけしか指定できません。
- オプション名同士，およびオプション名と引数の区切りには，半角空白文字を使用します。

2.3.2 一般規則

コンパイラオプションの一般規則について説明します。

- 大文字小文字は，等価とみなされます。
- オプション指定項の先頭文字は，ハイフン（-）またはスラント（/）となります。
- オプションとサブオプションを区切るコンマの前後には，空白を入れてはいけません。
- コンパイラオプションの指定に構文誤りがある場合，コンパイル時に回復不能（U レベル）エラーとなり，コンパイルが中止されます。

2.3.3 コンパイラオプションの優先順位

複数のコンパイラオプションを指定した場合の，各オプションの優先順位を次に示します。

(1) 指定個所による優先順位

コンパイラオプションを指定した個所によって，次の順に優先順位が高くなります。

ccbl2002 コマンドの場合

次の順序で，コンパイラオプションの優先順位が高くなります。

表 2-1　指定個所による優先順位（ccbl2002 コマンドの場合）

優先度	指定個所	オプションの形式
1	ccbl2002 のコマンド行に指定したオプション	新形式
2	環境変数 CBLCOPT2002 に指定したオプション	新形式

ccbl コマンドの場合

次の順序で，コンパイラオプションの優先順位が高くなります。

表 2-2　指定個所による優先順位（ccbl コマンドの場合）

優先度	指定個所	オプションの形式
1	ccbl のコマンド行に指定したオプション	旧形式
2	環境変数 CBLCOPT に指定したオプション	旧形式
3	ccbl コマンドで暗黙的に仮定されるオプション	新形式

環境変数 CBLCOPT2002，CBLCOPT については，「2.4.3　コンパイラ環境変数の詳細」を参照してください。

(2)　オプション間の優先順位

コンパイラオプションには，オプション同士が背反の関係となっていたり，あるオプションを指定するとほかのオプションが仮定されたりするものがあります。このようなオプション間の優先順位を，次に示します。

(a)　あとに指定した方のオプションが有効となる場合

次のコンパイラオプションを同時に指定した場合，あとに指定した方のオプションが有効となります。

- -DigitsTrunc／-Comp5
- -V3Rec／-EquivRule
- -CompatiV3／-EquivRule

(b)　オプションを指定すると，ほかのオプションが無効となる場合

次のコンパイラオプションを指定した場合，無効となるオプションがあります。

表 2-3　オプションを指定すると，ほかのオプションが無効となる場合

指定したオプション	無効となるオプション
-MainNotCBL	• -Main

指定したオプション	無効となるオプション
-Dll	• -Main • -SimSub
-Compile	• -OutputFile • -ManifestFileExt
-Compile,CheckOnly	• -DebugInf • -DebugCompati • -DebugData • -TDInf • -CVInf • -DebugRange • -TestCmd
-StdMIA -Std85 -Std2002	• -V3Spec • -StdVersion • -CompatiM7 • -CompatiV3 • -H8Switch • -Cblctr • -IgnoreLCC • -JPN • -CmDol • -Comp5 • -NumAccept • -V3Rec • -V3RecFCSpace • -V3RecEased • -EquivRule,NotAny • -SQL • -SQLDisp • -BinExtend • -MaxDigits38 • -IntResult,DecFloat40 • -LiteralExtend
-StdMIA	• -EquivRule,StdCode • -Bin1Byte
-V3Spec	• -Comp5 • -CmAster • -BinExtend • -CBLVALUE • -Bin1Byte • -CompatiM7 • -NumAccept

指定したオプション	無効となるオプション
	• -CmDol • -MaxDigits38 • -IntResult,DecFloat40 • -LiteralExtend • -V3RecEased
-Compati85,Syntax	• -LiteralExtend
-Repository,Gen	• -Compile
-UniObjGen	• -JPN • -CompatiV3※ • -V3Rec※ • -V3RecFCSpace※ • -V3RecEased※
-IntResult,DecFloat40	• -Compati85,Power （-Compati85,All 指定した場合も含む）
-CompatiV3	• -LiteralExtend
-SimMain -SimSub -SimIdent	• -LiteralExtend

注※

コンパイラ環境変数 CBLV3UNICODE に YES を指定した場合，無効になりません。

(c)　仕様チェックオプションを複数指定した場合

-StdMIA オプション，-Std85 オプション，-Std2002 オプションは，同時に指定できません。同時に指定した場合，警告のメッセージが出力され，次の優先順位で有効となります。

1. -Std2002 オプション
2. -Std85 オプション
3. -StdMIA オプション

(d)　オプションを指定することによって，仮定されるオプション

次のコンパイラオプションを指定した場合，仮定されるオプションがあります。

表 2-4　オプションを指定することによって，仮定されるオプション

指定したオプション	仮定されるオプション
-V3Spec	• -V3Rec,Variable※1
-CompatiV3	• -JPN,Alnum※2，※3

指定したオプション	仮定されるオプション
	• -V3Rec,Variable
-DebugCompati	• -DebugInf
-DebugData	• -DebugInf
-DebugRange	• -DebugInf • -DebugCompati
-TestCmd	• -DebugInf • -TDInf
-TDInf	• -DebugInf
-TDInf と-Optimize,3 を同時に指定	• -Optimize,2 （-Optimize,3 を指定しても-Optimize,2 が仮定される）
-CVInf	• -DebugInf
-StdMIA,14	• -StdMIA,13
-MaxDigits38 および-IntResult,DecFloat40 と-Optimize,3 を同時に指定	• -Optimize,2 （-Optimize,3 を指定しても-Optimize,2 が仮定される）
-SpaceAsZero と-Optimize,3 を同時に指定	• -Optimize,2 （-Optimize,3 を指定しても-Optimize,2 が仮定される）

注※1

-V3Spec オプションと-UniObjGen オプションを同時に指定した場合，-V3Rec,Variable オプションは仮定されません。

注※2

コンパイラ環境変数 CBLV3UNICODE に YES を指定し，-UniObjGen オプションを指定した場合は，-CompatiV3 オプションを指定しても-JPN,Alnum オプションは仮定されません。

注※3

-CompatiV3 オプションと-JPN,V3JPN オプションまたは-JPN,V3JPNSpace オプションを同時に指定した場合，-JPN,Alnum オプションは仮定されません。

(e) ほかのオプションの指定を必要とするオプション

次のコンパイラオプションは，同時に指定する必要があるオプションを指定しない場合，無視されます。

表 2-5 同時に指定しないと，指定が無視されるオプション

指定したオプション	同時に指定する必要があるオプション
-SQLDisp	-SQL,ODBC
-UniEndian	-UniObjGen
-V3RecFCSpace	-V3Rec
-DllInit	-Dll
-V3RecEased	-V3Rec

指定したオプション	同時に指定する必要があるオプション
-CheckUninitData	-Compile,CheckOnly

次のコンパイラオプションは，同時に指定する必要があるオプションを指定しない場合，コンパイルエラーとなります。

表 2-6　同時に指定しないと，コンパイルエラーとなるオプション

指定したオプション	同時に指定する必要があるオプション
-IntResult,DecFloat40	-MaxDigits38
-MaxDigits38	-IntResult,DecFloat40
-SpaceAsZero	-Compati85,All
-VOSCBL,OccursKey	-CompatiV3
-VOSCBL,ReportControl	-CompatiV3
-VOSCBL,DataComm	-CompatiV3
-VOSCBL,RedefinesData	-CompatiV3
-VOSCBL,AssignDataToDevice	-CompatiV3
-VOSCBL,EvaluateWhenOther	-CompatiV3
-JPN,V3JPNSpace	-CompatiV3

(3)　同じオプションを重複して指定した場合の規則

同じオプションを重複して指定した場合，次の規則に従ってオプションが決定されます。なお，オプションごとのサブオプションの指定規則については，「2.3.4　コンパイラオプションの一覧」を参照してください。

(a)　背反関係にあるサブオプション同士を指定した場合

最後に指定されたオプションが有効となります。

（例）

-Compile，{CheckOnly ｜ NoLink}

に対して

-Compile,CheckOnly -Compile,NoLink

と指定した場合，-Compile,NoLink が有効となります。

ただし，次の場合は，特定のサブオプションが有効となります。

- -TestCmd,Full オプションと-TestCmd,Break オプション，または-TestCmd,Full オプションと-TestCmd,Sim オプションを重複して指定した場合は，-TestCmd,Full オプションが有効となります。

- -JPN,Alnum オプションと-JPN,V3JPN オプションを重複して指定した場合は，-JPN,V3JPN オプションが有効となります。
- -JPN,V3JPNSpace オプションと-JPN,Alnum オプションを重複して指定した場合は，-JPN,V3JPNSpace オプションが有効となります。
- -JPN,V3JPNSpace オプションと-JPN,V3JPN オプションを重複して指定した場合は，-JPN,V3JPNSpace オプションが有効となります。
- -JPN,Alnum オプションと-JPN,V3JPN オプションを重複して指定した場合は，-JPN,V3JPN オプションが有効となります。
- -Std85,High オプション，-Std85,Middle オプション，-Std85,Low オプションを重複して指定した場合は，次の優先順位で有効となります。

  1. -Std85,Low オプション
  2. -Std85,Middle オプション
  3. -Std85,High オプション

(b) 省略可能なサブオプション同士，または複数選択できるサブオプション同士を指定した場合

前に指定したサブオプションの指定を引き継ぎ，あとに指定したサブオプションの指定が追加で有効となります。

（例）

-BigEndian ｛,Bin ｜,Float｝＋

に対して

-BigEndian,Bin -BigEndian,Float

と指定した場合，Bin サブオプションと Float サブオプションの両方が有効となります。これは，「-BigEndian,Bin,Float」を指定した場合と同じです。

(4) オプションの打ち消し指定

デフォルト設定や環境変数などで指定済みのオプションを，コンパイル時に打ち消したい場合は，プリフィックス'no'の付いたコンパイラオプションを使用します。

例えば，環境変数 CBLCOPT2002 でオプション「-Details」が指定されている場合，コマンドラインで「ccbl2002 -noDetails …」と指定すると，環境変数 CBLCOPT2002 で設定済みのオプション「-Details」を打ち消せます。

規則

- オプションにデフォルト値の設定がある場合，プリフィックス'no'は指定できません。この場合，オプションのデフォルト値を指定することで，オプションを打ち消せます。

  例えば，環境変数 CBLCOPT2002 でオプション「-Lib,CUI」が指定されている場合，デフォルト値に戻したいときは，コマンドラインで「-Lib,GUI」を指定します。

- 通常のオプションと打ち消しのオプションを同時に指定した場合は，あとに指定したオプションが有効となります。
- 打ち消しのオプションは，コマンドラインや環境変数 CBLCOPT2002 に指定したオプションを打ち消し，オプション指定によって仮定されるオプションは打ち消しません。

注意事項

コンパイルリストに出力されるオプション一覧や，-Details オプションを指定した場合にコマンドラインへ出力されるコンパイラオプションの詳細情報表示では，no 指定のコンパイラオプションは表示されません。コンパイラオプションとして有効となったオプションだけが表示されます。

2.3.4　コンパイラオプションの一覧

コンパイラオプションの一覧を次に示します。

なお，打ち消しのオプション（-noXXX）を開発マネージャで指定する場合は，プロジェクト設定ダイアログボックスの［ユーザ設定］タブを使用してください。［ユーザ設定］タブの詳細は，「2.2.3　オプションの設定方法」を参照してください。

(1)　最終生成物の種類（プロジェクトの種類）

最終生成物※の種類（開発マネージャでは，プロジェクトの種類）を設定するコンパイラオプションを，次に示します。

注※

最終生成物とは，コンパイラが最終的に生成する実行可能ファイル，DLL，または標準ライブラリのことを示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.5　最終生成物の種類（プロジェクトの種類）の設定」の番号（(1)，(2)，…）と対応しています。

表 2-7　最終生成物の種類（プロジェクトの種類）を設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-Main，{System | V3} ファイル名	○	先頭の最外側プログラムを主プログラムとして作成する。 開発マネージャでは，次のオプションが該当する。 System 指定メインプログラム V3 指定メインプログラム	(1)　-Main オプション
2	−	○	メインプログラムなし	(2)　メインプログラムなし

項番	オプション	マネージャ	機能	参照先
3	-Dll，{Stdcall ｜ Cdecl} (Windows(x86) COBOL2002 で有効) -Dll(Windows(x64) COBOL2002 で有効) -noDll	○	DLL の形式を指定する。 開発マネージャでは，次のオプションが該当する。 最終生成物の種類がダイナミックリンクライブラリ StdCall：Dll の属性を Stdcall にする Cdecl：Dll の属性を Cdecl にする	(3)　-Dll オプション（最終生成物の種類がダイナミックリンクライブラリ）
4	－	○	標準ライブラリを作成する	(4)　標準ライブラリ

（凡例）

○：開発マネージャから指定できる。

－：コマンドライン（ccbl2002 コマンド）に対応するオプションが存在しない。

(2)　製品連携

他製品との連携を設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.6　他製品との連携の設定」の番号（(1)，(2)，…）と対応しています。

表 2-8　他製品との連携を設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-SQL，{XDM ｜ ODBC 〔,NoCont〕} -noSQL	○	埋め込み SQL 文を XDM/RD または ODBC インタフェース機能で使用できるようにする。	(1)　-SQL オプション
2	-SQLDisp -noSQLDisp	○	埋め込み SQL 文に用途（USAGE 句）が表示用（DISPLAY）のデータ項目を指定できるようにする。	(2)　-SQLDisp オプション
3	-RDBTran -noRDBTran	○	COMMIT 文／ROLLBACK 文を HiRDB による索引編成ファイルに対して適用する。	(3)　-RDBTran オプション
4	-IsamExtend〔,Zone〕 -noIsamExtend	○	Btrieve（Pervasive.SQL）による索引編成ファイルを使用する。	(4)　-IsamExtend オプション (Windows(x86) COBOL2002 で有効)
5	-XMAP,LinePrint -noXMAP	○	書式印刷機能を使用して，順編成ファイルをプリンタに出力する。	(5)　-XMAP オプション (Windows(x86) COBOL2002 で有効)

項番	オプション	マネージャ	機能	参照先
6	-OpenTP1 -noOpenTP1	○	OpenTP1 を使用したデータコミュニケーション機能を使用できるようにする。	(6)　-OpenTP1 オプション

（凡例）

○：開発マネージャから指定できる。

(3)　実行

実行時の動作を設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.7　実行の設定」の番号（(1)，(2)，…）と対応しています。

表 2-9　実行時の動作を設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-NumAccept -noNumAccept	○	ACCEPT 文に数字項目を指定できるようにする。	(1)　-NumAccept オプション
2	-NumCsv -noNumCsv	○	CSV 編成ファイルで，セルデータを数値として入出力できるようにする。	(2)　-NumCsv オプション
3	-MultiThread -noMultiThread	○	マルチスレッド対応 COBOL プログラムを作成する。	(3)　-MultiThread オプション
4	-MainNotCBL -noMainNotCBL	○	すべて副プログラムとして作成する。	(4)　-MainNotCBL オプション
5	-DllInit -noDllInit	○	DLL を初期化する。	(5)　-DllInit オプション
6	－	○	メインファイルを指定する （開発マネージャで，プロジェクトのメインプログラムにするソースファイルを指定する）	(6)　メインファイルを指定する

（凡例）

○：開発マネージャから指定できる。

－：コマンドライン（ccbl2002 コマンド）に対応するオプションが存在しない。

(4)　最適化

プログラムの最適化を設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)）は「2.3.8　プログラムの最適化の設定」の番号（(1)，(2)）と対応しています。

表 2-10　プログラムの最適化を設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-Optimize, {0 | 1 | 2 | 3}	○	コンパイル時の最適化のレベルを指定する。	(1)　-Optimize オプション
2	-ScreenSpeed -noScreenSpeed	○	画面の表示速度を重視する。	(2)　-ScreenSpeed オプション

(凡例)

○：開発マネージャから指定できる。

(5)　デバッグ

デバッグを設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.9　デバッグの設定」の番号（(1)，(2)，…）と対応しています。

表 2-11　デバッグを設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-DebugLine -noDebugLine	○	デバッグ行を有効にする。	(1)　-DebugLine オプション
2	-DebugInf〔,Trace〕 -noDebugInf	○	異常終了時，エラー要約情報を出力する。	(2)　-DebugInf オプション
3	-DebugCompati -noDebugCompati	○	実行時に次のチェックを行う。 • 添字，指標名の反復回数の範囲外チェック • プログラム間整合性チェック	(3)　-DebugCompati オプション
4	-DebugData〔,ValueHex〕 -noDebugData	○	データ例外を検出する。	(4)　-DebugData オプション
5	-TDInf -noTDInf	○	テストデバッグ情報を出力する。	(5)　-TDInf オプション
6	-CVInf -noCVInf	○	カバレージ情報を出力する。	(6)　-CVInf オプション
7	-DebugRange -noDebugRange	○	添字，指標名の反復回数について，次元ごとの範囲外チェックを行う。	(7)　-DebugRange オプション
8	-TestCmd, {Full | ,Break | ,Sim} ＋	○	TD コマンド格納ファイルに出力する情報の種類を指定する。	(8)　-TestCmd オプション

項番	オプション	マネージャ	機能	参照先
	-noTestCmd			
9	-SimMain プログラム名 -noSimMain	○	主プログラムをシミュレーションする。	(9)　-SimMain オプション
10	-SimSub プログラム名 -noSimSub	○	副プログラムをシミュレーションする。	(10)　-SimSub オプション
11	-SimIdent -noSimIdent	○	副プログラム（一意名 CALL 文）をシミュレーションする。	(11)　-SimIdent オプション

（凡例）

○：開発マネージャから指定できる。

(6)　リンク

リンクを設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.10　リンクの設定」の番号（(1)，(2)，…）と対応しています。

表 2-12　リンクを設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-StdCall -noStdCall	○	stdcall 呼び出し指示ファイルを有効にする。	(1)　-StdCall オプション (Windows(x86) COBOL2002 で有効)
2	-StdCallFile .cbw ファイル名 -noStdCallFile	○	stdcall 呼び出し指示ファイル名を指定する。	(2)　-StdCallFile オプション (Windows(x86) COBOL2002 で有効)
3	-Lib, {GUI | CUI}	○	GUI モード／CUI モードのどちらの実行時ライブラリをリンクするかを指定する。	(3)　-Lib オプション
4	−	○	ライブラリの指定	(4)　ライブラリの指定
5	−	○	リンケージを行わない	(5)　リンケージを行わない
6	-Compile, {CheckOnly | NoLink} -noCompile	×	コンパイルの処理範囲を指定する。	(6)　-Compile オプション

項番	オプション	マネージャ	機能	参照先
7	-DefFile .def ファイル名 -noDefFile	×	生成する.def ファイル名を指定する。	(7)　-DefFile オプション
8	-IconFile アイコンファイル名 -noIconFile	×	アイコンファイル名を指定する。	(8)　-IconFile オプション
9	-ResrcFile ファイル名 -noResrcFile	×	生成するリソース定義ファイル名を指定する。	(9)　-ResrcFile オプション
10	-OutputFile ファイル名 -noOutputFile	×	生成する実行可能ファイル名，または DLL ファイル名を指定する。	(10)　-OutputFile オプション
11	-Link オプションの並び -noLink	U	リンカに渡すオプションを指定する。	(11)　-Link オプション
12	-DynamicLink,Call -noDynamicLink	○	定数指定の CALL 文実行に動的リンク機能を使用するときに指定する。	(12)　-DynamicLink オプション
13	-ManifestFileExt -noManifestFileExt	○	マニフェストファイルを実行可能ファイルや DLL に埋め込まずに，外部マニフェストファイルとするときに指定する。	(13)　-ManifestFileExt オプション

（凡例）

○：開発マネージャから指定できる。

U：開発マネージャで使用する場合，［ユーザ設定］タブに指定する。

×：開発マネージャから指定できない。

－：コマンドライン（ccbl2002 コマンド）に対応するオプションが存在しない。

(7)　規格

規格仕様のチェックを設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.11　規格の設定」の番号（(1)，(2)，…）と対応しています。

表 2-13　規格仕様のチェックを設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-StdMIA {,13	,14} ＋ -noStdMIA※	○	MIA 仕様の範囲外チェックをする。	(1)　-StdMIA オプション
2	-Std85 {, {High	Middle	Low}	,Obso	,Report} ＋ -noStd85※	○	JIS 仕様をチェックする。	(2)　-Std85 オプション

項番	オプション	マネージャ	機能	参照先
3	-Std2002｛,OutRange｜,Obso｜,Archaic｝＋ -noStd2002	○	COBOL2002 規格仕様をチェックする。	(3)　-Std2002 オプション
4	-StdVersion,｛1｜2｝ -noStdVersion※	○	第 1 次規格／第 2 次規格の解釈でコンパイルする。	(4)　-StdVersion オプション

（凡例）

○：開発マネージャから指定できる。

注※

これらのオプションは，自由形式正書法で書かれた COBOL 原始プログラムとしてコンパイルするプログラムには指定できません。指定してコンパイルすると，エラーとなってコンパイルが中止されます。

(8)　移行

他システムとの移行を設定するコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.12　他システムとの移行の設定」の番号（(1)，(2)，…）と対応しています。

表 2-14　他システムとの移行を設定するコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-CompatiM7 -noCompatiM7	○	MIOS7 COBOL85 との互換機能を有効にする。	(1)　-CompatiM7 オプション (Windows(x86) COBOL2002 で有効)
2	-CompatiV3※ -noCompatiV3	○	VOS3 COBOL85 との互換機能を有効にする。	(2)　-CompatiV3 オプション
3	-Compati85｛,IoStatus｜,Linage｜,Call｜,Power｜,Syntax｜,IDParag｜,RsvWord｜,NoPropagate｜,All｝＋ -noCompati85	○	COBOL85 互換機能を有効にする。	(3)　-Compati85 オプション
4	-H8Switch -noH8Switch	○	HITAC8000 シリーズの仕様でコンパイルする。	(4)　-H8Switch オプション
5	-Cblctr -noCblctr	○	CBL-CTR 特殊レジスタを使用できるようにする。	(5)　-Cblctr オプション
6	-DigitsTrunc -noDigitsTrunc	○	転記文で上位けたを切り捨てる。	(6)　-DigitsTrunc オプション
7	-IgnoreLCC	○	行送り制御文字を無視する。	(7)　-IgnoreLCC オプション

項番	オプション	マネージャ	機能	参照先
	-noIgnoreLCC			
8	-CmAster -noCmAster	○	1 カラム目が'*'の行を注記行とする。	(8)　-CmAster オプション
9	-CmDol -noCmDol	○	7 カラム目が'$'の行を注記行とする。	(9)　-CmDol オプション
10	-Comp5 -noComp5	○	COMP-5 を指定できるようにする。	(10)　-Comp5 オプション
11	-V3Spec〔,CopyEased〕 -noV3Spec	○	VOS3 COBOL85 に対する COBOL2002 固有の構文をチェックする。	(11)　-V3Spec オプション
12	-V3ConvtName -noV3ConvtName	○	VOS3 COBOL85 からのソースファイル互換のため，COPY 文の原文名定数中の'¥'と'@'を変換する。	(12)　-V3ConvtName オプション
13	-Switch,｛EBCDIC ｜ EBCDIK｝〔,Unprintable〕〔,noApplyJpnItem〕 -noSwitch	○	照合順序および字類条件を EBCDIC コードまたは EBCDIK コードに切り替える。	(13)　-Switch オプション
14	-V3Rec,｛Fixed ｜ Variable｝※ -noV3Rec	○	メインフレーム（VOS3）の固定長または可変長レコード形式のプログラムを，VOS3 の日本語文字の扱いに合わせてコンパイルする。	(14)　-V3Rec オプション
15	-V3RecFCSpace -noV3RecFCSpace	○	空白に関する機能キャラクタの扱いをメインフレーム（VOS3）と同等にする。	(15)　-V3RecFCSpace オプション
16	-V3RecEased｛,QuoteCheck ｜,WordCheck｝＋ -noV3RecEased	○	-V3Rec オプション指定時の仕様チェックを緩和する。	(21)　-V3RecEased オプション
17	-DoubleQuote -noDoubleQuote	○	引用符（ " ）を分離符とみなしてコンパイルする。	(16)　-DoubleQuote オプション
18	-BigEndian｛,Bin ｜,Float｝＋ -noBigEndian	○	2 進データ項目または浮動小数点データ項目をビッグエンディアン形式で処理する。	(17)　-BigEndian オプション
19	-VOSCBL{,OccursKey ｜,ReportControl ｜,DataComm ｜,RedefinesData ｜,AssignDataToDevice ｜,EvaluateWhenOther}+ -noVOSCBL	○	メインフレーム互換機能を有効にする。	(18)　-VOSCBL オプション

項番	オプション	マネージャ	機能	参照先
20	-PortabilityCheck{,Literal | ,Numeric} ＋ -noPortabilityCheck	○	移行向けチェック機能を有効にする。	(19)　-PortabilityCheck オプション
21	-IgnoreAPPLY,FILESHARE -noIgnoreAPPLY	○	入出力管理記述項の APPLY FILE-SHARE 句を覚え書きとみなす。	(20)　-IgnoreAPPLY,FILESHARE オプション

（凡例）

○：開発マネージャから指定できる。

注※

これらのオプションは，自由形式正書法で書かれた COBOL 原始プログラムとしてコンパイルするプログラムには指定できません。指定してコンパイルすると，エラーとなってコンパイルが中止されます。

(9)　リスト出力

リスト出力の設定をするコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)）は「2.3.13　リスト出力の設定」の番号（(1)，(2)）と対応しています。

表 2-15　リスト出力の設定をするコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-SrcList，{OutputAll | CopyAll | CopySup | NoCopy}〔,NoFalsePath〕〔,DataLoc〕 -noSrcList	○	コンパイルリストを出力する。	(1)　-SrcList オプション
2	-ErrSup｛,I | ,W｝＋ -noErrSup	○	I レベルまたは W レベルエラーの出力を抑止する。	(2)　-ErrSup オプション

（凡例）

○：開発マネージャから指定できる。

(10)　その他

その他の設定をするコンパイラオプションを，次に示します。

表中の参照先の番号（(1)，(2)，…）は「2.3.14　その他の設定」の番号（(1)，(2)，…）と対応しています。

表 2-16　その他の設定をするコンパイラオプション

項番	オプション	マネージャ	機能	参照先
1	-Bin1Byte -noBin1Byte	○	1 バイトの 2 進項目を有効にする。(PICTURE 句の指定で 2 けたまでは 1 バイトとして扱う)	(1)　-Bin1Byte オプション
2	-JPN, {Alnum | V3JPN | V3JPNSpace} -noJPN	○	日本語項目の扱いを指定する。	(2)　-JPN オプション
3	-EquivRule, {NotExtend | NotAny | StdCode} -noEquivRule	○	文字の等価規則をどう変更するか指定する。	(3)　-EquivRule オプション
4	-UscoreStart -noUscoreStart	○	先頭が下線の CALL 定数を指定できるようにする。	(4)　-UscoreStart オプション
5	-BinExtend -noBinExtend	○	2 進データ項目に指定できる初期値を拡張する。	(5)　-BinExtend オプション
6	-MinusZero -noMinusZero	○	10 進項目で負の符号を持つゼロを正の符号を持つゼロに変換する。	(6)　-MinusZero オプション
7	-TruncCheck -noTruncCheck	○	転記でのデータ切り捨てをチェックする。	(7)　-TruncCheck オプション
8	-LowerAsUpper -noLowerAsUpper	○	定数指定の CALL に指定された英小文字を英大文字に変換してプログラムを呼び出す。	(8)　-LowerAsUpper オプション
9	-CBLVALUE -noCBLVALUE	○	環境変数 CBLVALUE を有効にする。	(9)　-CBLVALUE オプション
10	-Repository, {Gen | Sup} -noRepository	○	リポジトリファイルの生成時，強制的に出力するか，更新しないかを指定する。	(10)　-Repository オプション
11	-RepositoryCheck -noRepositoryCheck	○	同じソースファイル中の翻訳単位の定義と外部リポジトリ中の情報に相違があるかどうかをチェックする。	(11)　-RepositoryCheck オプション
12	-Define 翻訳変数名〔=値〕〔,翻訳変数名〔=値〕〕… -noDefine	○	コンパイル時に有効となる，翻訳変数名とその値を定義する。	(12)　-Define オプション
13	-Details -noDetails	○	コンパイラオプションの詳細情報を出力する。	(13)　-Details オプション
14	-OldForm "旧オプションの並び"	U	PC COBOL85 のオプションを指定する。	(14)　-OldForm オプション

項番	オプション	マネージャ	機能	参照先
15	-Help | -?	×	ccbl2002 コマンドのヘルプを出力する。	(15)　-Help オプション
16	-UniObjGen -noUniObjGen	○	シフト JIS の COBOL ソースから Unicode のオブジェクトを生成する。	(16)　-UniObjGen オプション
17	-UniEndian, {Little | Big} -noUniEndian	○	シフト JIS で記述された日本語文字定数を UTF-16LE，または UTF-16BE に変換する。	(17)　-UniEndian オプション
18	-MaxDigits38	○	数字項目，数字編集項目，および数字定数に指定できる最大けた数を 18 けたから 38 けたに拡張する。	(18)　-MaxDigits38 オプション（Windows(x64) COBOL2002 で有効）
19	-IntResult,DecFloat40	○	算術演算の中間結果の表現形式を 40 けたの 10 進浮動小数点形式とする。	(19)　-IntResult,DecFloat40 オプション（Windows(x64) COBOL2002 で有効）
20	-LiteralExtend,Alnum -noLiteralExtend	○	英数字定数と定数指定のプログラム名の長さを拡張する。	(20)　-LiteralExtend オプション
21	-SpaceAsZero -noSpaceAsZero	○	外部 10 進項目中に空白文字があるとき，ゼロとみなして比較，演算，転記を実行する。	(21)　-SpaceAsZero オプション
22	-CheckUninitData -noCheckUninitData	×	データ項目の初期化漏れをチェックする。	(22)　-CheckUninitData オプション
23	-FunctionECSup,CodeConvErr -noFunctionECSup	○	組み込み関数でコード変換エラーが発生しても例外条件を成立させない。	(23)　-FunctionECSup,CodeConvErr オプション

（凡例）

○：開発マネージャから指定できる。

U：開発マネージャで使用する場合，［ユーザ設定］タブに指定する。

×：開発マネージャから指定できない。

(11)　XML 連携（開発マネージャを使用する場合）

開発マネージャのツリービューで XML データ定義ファイルを選択したときに指定できるオプションです。

このページで指定するオプションの詳細については，マニュアル「COBOL2002　XML 連携機能ガイド」を参照してください。

2.3.5　最終生成物の種類（プロジェクトの種類）の設定

最終生成物※の種類（開発マネージャでは，プロジェクトの種類）を設定するコンパイラオプションについて，説明します。

注※

最終生成物とは，コンパイラが最終的に生成する実行可能ファイル，DLL，または標準ライブラリのことを示します。

(1)　-Main オプション

(a)　形式

```
-Main, {System | V3}　ファイル名
```

(b)　機能

最外側のプログラムをアプリケーションの主プログラムとしてコンパイルします。

ファイル中に複数の最外側のプログラムがあるときは，先頭の最外側のプログラムを主プログラムとしてコンパイルします。

-Main,System ファイル名（System 指定メインプログラム）

最外側のプログラムをアプリケーションの主プログラムとしてコンパイルします。

このとき，主プログラムが制御プログラムから受け取る引数の形式を，システム固有の argc，argv 形式に合わせます。

なお，開発マネージャでは，プロジェクトの種類で「System 指定メインプログラム」を選び，「メインファイルを指定する」オプションにメインファイルを指定すると，-Main,System オプションが指定されます。

-Main,V3 ファイル名（V3 指定メインプログラム）

最外側のプログラムをアプリケーションの主プログラムとしてコンパイルします。

このとき，主プログラムが制御プログラムから受け取る引数の形式を，メインフレーム（VOS3）に合わせます。

なお，開発マネージャでは，プロジェクトの種類で「V3 指定メインプログラム」を選び，「メインファイルを指定する」オプションにメインファイルを指定すると，-Main,V3 オプションが指定されます。

(c)　注意事項

- -Main,System オプションまたは-Main,V3 オプションの指定がない場合で，次の条件をすべて満たすときには，先頭の COBOL ソースファイルに-Main,System オプションを指定したものと仮定されます。
  1. -Compile オプションの指定がなく，かつ，オブジェクトファイル（.obj）の指定がない。
  2. -SimMain オプションの指定がない。

- -Main オプションと-MainNotCBL オプションまたは-Dll オプションを同時に指定した場合，-Main オプションが無効となります。

  （例）

  -MainNotCBL -Main,System

  と指定した場合，-MainNotCBL オプションが有効となり，-Main オプションは無効となります。
- -Main,System オプション，-Main,V3 オプションについては，マニュアル「COBOL2002　ユーザーズガイド」の記述も参照してください。
- このオプションは，直後に指定されたファイルをアプリケーションの主プログラムとして扱います。

```
ccbl2002 -Main,System main1.cbl sub1.cbl test1.obi
                      主ﾌﾟﾛｸﾞﾗﾑ    副ﾌﾟﾛｸﾞﾗﾑ
```

- -Main,V3 オプションを指定して作成した実行可能ファイルを実行するとき，受け取れるコマンド引数は一つ（空白まで）です。コマンド引数に空白を含むときは，引用符（ " ）で囲みます。

(2)　メインプログラムなし

開発マネージャ上でプロジェクトをビルドする場合に，メインの COBOL ソースファイルを指定しないとき，このオプションを指定します。「メインプログラムなし」を指定したプロジェクトは，メインプログラムのオブジェクトファイルまたはライブラリを，プロジェクトに登録する必要があります。

このコンパイル方式を選んだ場合で，作成しようとしている COBOL プログラムが次の二つの条件を満たすときは，-MainNotCBL オプションを指定する必要があります。

- 最初に呼び出される COBOL プログラムより前に，他言語で記述されたプログラムが実行される場合
- 最初に呼び出される COBOL プログラムから，呼び出し元プログラムに戻る場合

このプロジェクトの種類で指定できるオプションは，-Main,System オプション（System 指定メインプログラム）または-Main,V3 オプション（V3 指定メインプログラム）を指定して実行可能ファイルを生成する場合に準じますが，次のオプションは指定できません。

「メインファイルを指定する」

このオプションは，開発マネージャだけで指定できます。

(3)　-Dll オプション（最終生成物の種類がダイナミックリンクライブラリ）

(a)　形式

```
-Dll, {Stdcall | Cdecl}
-Dll
-noDll
```

(b) 機能

DLL 形式のオブジェクトファイルを出力します。-Dll オプションを指定しない場合は，実行可能ファイル（.exe）形式のオブジェクトファイルを出力します。

-Dll オプションを指定した場合，同時に指定した COBOL 原始プログラムは，すべて DLL 形式のオブジェクトファイルとなります。

なお，開発マネージャでは，最終生成物の種類に「ダイナミックリンクライブラリ」を選ぶと，-Dll オプションが指定されます。

-Dll,Stdcall（Dll の属性を stdcall にする）(Windows(x86) COBOL2002 で有効)

DLL の属性を stdcall にします。

なお，開発マネージャでは，「Dll の属性を Stdcall にする」を選ぶと，-Dll,Stdcall オプションが指定されます。

-Dll,Cdecl（Dll の属性を cdecl にする）(Windows(x86) COBOL2002 で有効)

DLL の属性を cdecl にします。

なお，開発マネージャでは，「Dll の属性を Cdecl にする」を選ぶと，-Dll,Cdecl オプションが指定されます。

-Dll（Dll の属性を fastcall にする）(Windows(x64) COBOL2002 で有効)

DLL の属性を fastcall にします。

開発マネージャでは，最終生成物の種類に「ダイナミックリンクライブラリ」を選びます。

-noDll

-Dll オプションの指定を打ち消します。

(c) 注意事項

- -Dll オプションを指定すると，-Main,System，-Main,V3，および-SimSub オプションが無効となります。
- Windows(x64) COBOL2002 の場合，-Dll,Stdcall オプション，または-Dll,Cdecl オプションを指定しても，-Dll オプションを指定したとみなされ，DLL の属性は fastcall になります。

(4) 標準ライブラリ

COBOL ソースから標準ライブラリを作成します。

プロジェクトの作成時に，最終生成物の種類で「標準ライブラリ」を選んだ場合，この項目が指定されます。

このプロジェクトによって生成された標準ライブラリをリンクする実行可能プログラムが次の条件を満たす場合，-MainNotCBL オプションを指定する必要があります。

- 最初に呼び出される COBOL プログラムより前に，他言語で記述されたプログラムが実行される場合
- 最初に呼び出される COBOL プログラムから，呼び出し元プログラムに戻る場合

このプロジェクトの種類を選んだ場合，プロジェクトに指定できるオプションは-Main,System または-Main,V3 オプションを指定した場合と同じです。ただし，次のオプションは指定できません。

-SimMain，-SimSub，-SimIdent，-Lib，「リンケージ処理を行わない」，「メインファイルを指定する」

このオプションは，開発マネージャだけで指定できます。

2.3.6　他製品との連携の設定

他製品との連携を設定するコンパイラオプションについて，説明します。

(1)　-SQL オプション

(a)　形式

```
-SQL, {XDM | ODBC 〔,NoCont〕 }
-noSQL
```

(b)　機能

埋め込み SQL 文を使用できるようにします。

-SQL,XDM

リレーショナルデータベース（XDM/RD）操作シミュレーション機能を使用する場合に指定します。このオプションを指定すると，VOS3 XDM/RD の SQL は覚え書きとみなされます。

-SQL,ODBC

埋め込み SQL 文を ODBC インタフェース機能で実行する場合に指定します。このオプションを指定すると，埋め込み SQL 文を使用して，ODBC インタフェース機能でデータベースにアクセスします。

-SQL,ODBC,NoCont

埋め込み SQL 文を ODBC インタフェース機能で実行する場合で，SQL 構文内の浮動継続指示子を有効としないときに指定します。

-noSQL

-SQL オプションの指定を打ち消します。

(c)　注意事項

- -SQL,XDM オプションと-SQL,ODBC オプションを同時に指定した場合，あとに指定したオプションが有効となります。

(2) -SQLDisp オプション

(a) 形式

```
-SQLDisp
-noSQLDisp
```

(b) 機能

-SQLDisp

埋め込み SQL 文に，用途が表示用のデータ項目を指定できるようにします。

このオプションを指定すると，データベースアクセス機能で，埋め込み変数の定義に次に示すデータ記述項を指定できます。埋め込み SQL 文では，これらの項目が占めるバイト数分と同じバイト数を占める，英数字項目の埋め込み変数として使用できます。

- 英字項目
- 〔SIGN IS〕 LEADING SEPARATE CHARACTER の指定がない外部 10 進形式の数字項目
- 英数字編集項目
- 数字編集項目
- 日本語項目
- 日本語編集項目
- 外部ブール項目
- 外部浮動小数点項目

-noSQLDisp

-SQLDisp オプションの指定を打ち消します。

(3) -RDBTran オプション

(a) 形式

```
-RDBTran
-noRDBTran
```

(b) 機能

-RDBTran

翻訳単位中に記述された COMMIT 文，ROLLBACK 文を，HiRDB による索引編成ファイルに対して適用します。詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「プログラムのコンパイルと実行」を参照してください。

なお，このオプションを指定するとデータコミュニケーション（DC）機能，および DC シミュレーション機能は使用できません。

また，このオプションの指定をしないと，COMMIT 文，ROLLBACK 文は，DC 機能に対して適用されます。

-noRDBTran

-RDBTran オプションの指定を打ち消します。

(4) -IsamExtend オプション(Windows(x86) COBOL2002 で有効)

(a) 形式

```
-IsamExtend [,Zone]
-noIsamExtend
```

(b) 機能

-IsamExtend

Btrieve（Pervasive.SQL）による索引編成ファイルを使用します。ISAM による索引編成ファイルを使用するときは，このオプションを指定しません。詳細は，マニュアル「COBOL2002 ユーザーズガイド」の「Btrieve（Pervasive.SQL）による索引編成ファイル（Windows(x86) COBOL2002 で有効)」を参照してください。

-IsamExtend,Zone

Btrieve（Pervasive.SQL）による索引編成ファイル使用する場合，レコードキーに外部 10 進項目が指定されたときに，外部 10 進属性として処理します。

-noIsamExtend

-IsamExtend オプションの指定を打ち消します。

(5) -XMAP オプション(Windows(x86) COBOL2002 で有効)

(a) 形式

```
-XMAP,LinePrint
-noXMAP
```

(b) 機能

-XMAP,LinePrint

XMAP3 を使用して，順編成ファイルをプリンタに出力するときに指定します。このとき，次の印刷機能も使用できます。

- 書式オーバレイ（APPLY FORMS-OVERLAY 句）
- 印刷制御付き（CHARACTER TYPE 句）

COBOL プログラム中に APPLY FORMS-OVERLAY 句または CHARACTER TYPE 句があっても，-XMAP,LinePrint オプションの指定がなければ，これらの句は覚え書きとなります。

注

XMAP3 を使用しないで CHARACTER TYPE 句を使用した印刷制御付きのプリンタ出力をする場合，GDI モード印刷機能を使用します。

これらの印刷機能の詳細については，マニュアル「COBOL2002　ユーザーズガイド」のプリンタへのアクセスの説明を参照してください。

-noXMAP

-XMAP オプションの指定を打ち消します。

(6)　-OpenTP1 オプション

(a)　形式

```
-OpenTP1
-noOpenTP1
```

(b)　機能

-OpenTP1

OpenTP1 を使用したデータコミュニケーション機能を使用する場合に指定します。ただし，テストデバッガの DC シミュレーション機能で，データコミュニケーション機能の単体テストをする場合は，このオプションを指定しないでください。

-noOpenTP1

-OpenTP1 オプションの指定を打ち消します。

2.3.7　実行の設定

実行の設定をするコンパイラオプションについて，説明します。

(1)　-NumAccept オプション

(a)　形式

```
-NumAccept
-noNumAccept
```

(b)　機能

-NumAccept

ACCEPT 文の一意名 1 に次の項目を指定できるようにします。

- 符号なし数字項目

- 符号あり数字項目
- 数字編集項目
- 外部浮動小数点数字項目
- 内部浮動小数点数字項目

数字項目を指定した場合，入力データは右詰めで格納され，残りの部分には左側に 0 が埋められます。数字編集項目を指定した場合，編集文字に合わせて右詰めで格納されます。浮動小数点数字項目を指定した場合，受け取りの有効けた数分の値が格納されます。

-noNumAccept

-NumAccept オプションの指定を打ち消します。

(2) -NumCsv オプション

(a) 形式

```
-NumCsv
-noNumCsv
```

(b) 機能

-NumCsv

CSV 編成ファイルで，セルのデータを数値として入出力したい場合に指定するオプションです。詳細は，マニュアル「COBOL2002 ユーザーズガイド」の「セルデータを数値として入出力する機能」を参照してください。

-noNumCsv

-NumCsv オプションの指定を打ち消します。

(3) -MultiThread オプション

(a) 形式

```
-MultiThread
-noMultiThread
```

(b) 機能

-MultiThread

COBOL プログラムをマルチスレッド環境下で動作させたい場合に指定します。

詳細は，マニュアル「COBOL2002 ユーザーズガイド」の「マルチスレッド対応 COBOL プログラムの生成」を参照してください。

-noMultiThread

-MultiThread オプションの指定を打ち消します。

(4) -MainNotCBL オプション

(a) 形式

```
-MainNotCBL
-noMainNotCBL
```

(b) 機能

-MainNotCBL

実行中のプロセスで最初に呼び出される COBOL プログラムを，副プログラムとします。他言語で作成したアプリケーションのメインプログラムから COBOL プログラムを呼び出す場合，このオプションを指定します。

-noMainNotCBL

-MainNotCBL オプションの指定を打ち消します。

(c) 注意事項

- -Main オプションと-MainNotCBL オプションを同時に指定した場合，-MainNotCBL オプションが有効となります。

(5) -DllInit オプション

(a) 形式

```
-DllInit
-noDllInit
```

(b) 機能

-DllInit

呼び出し時に DLL を初期状態にします。

-noDllInit

-DllInit オプションの指定を打ち消します。

(c) 注意事項

- -DllInit オプションを指定して，-Dll オプションを指定しなかった場合，-DllInit オプションは無視されます。この場合，-DllInit オプションが無効であることを示す警告メッセージが出力されます。
- -DllInit オプションを指定した場合，次に示す情報が初期化されます。

1. ファイルの状態
2. PERFORM 文に対する制御機構
3. ALTER 文で設定した GO TO 文
4. 作業場所節，ファイル節の内容

(6) メインファイルを指定する

プロジェクトのメインプログラムにするソースファイルを指定します。

メインファイルに指定していたファイルをプロジェクトから削除した場合などは，この項目でメインファイルを指定します。

このオプションは，開発マネージャだけで指定できます。

2.3.8 プログラムの最適化の設定

プログラムの最適化を設定するコンパイラオプションについて，説明します。

(1) -Optimize オプション

(a) 形式

```
-Optimize, {0 | 1 | 2 | 3}
```

(b) 機能

最適化のレベルを指定します。-Optimize オプションの詳細については，マニュアル「COBOL2002 ユーザーズガイド」の「最適化機能」を参照してください。

-Optimize,0
　最適化をしません。

-Optimize,1
　文の中で閉じた次の最適化をします。

- 命令レベルでの最適化（ピープホールによる不要命令の削除など）

　-Optimize オプションを指定しなかった場合，このオプションが仮定されます。

-Optimize,2
　広域的な次の最適化をします。

- 命令レベルでの最適化（ピープホールによる不要命令の削除など）
- 不変式のループ外への移動

- コピー伝播
- 定数の畳み込み
- 共通式の削除
- 演算の強さの軽減
- そと PERFORM 文のインライン展開

-Optimize,3

-Optimize,2 オプションでの最適化に加えて，10 進項目を 2 進項目に変換します。

(c) 注意事項

- 最適化オプションを指定すると，データ項目の使用状況の解析や文の順番の入れ替えなどをするため，コンパイル時間が増加します。-Optimize,0 指定時が最短で，-Optimize,3 指定時が最長となります。
- -Optimize,3 オプションと-TDInf オプションを同時に指定すると，-Optimize,3 オプションが無効になり，-Optimize,2 オプションが仮定されます。
- 最適化の内容については，マニュアル「COBOL2002　ユーザーズガイド」の「最適化機能」を参照してください。
- -Optimize,3 オプションを指定すると 10 進項目を 2 進項目化するため，アドレス操作機能を使用しているプログラムは正しく動作しないことがあります。
- Windows(x64) COBOL2002 の場合，-Optimize,3 オプション，-MaxDigits38 オプション，および-IntResult,DecFloat40 オプションを同時に指定したときは，-Optimize,3 オプションは無効となり，-Optimize,2 オプションが仮定されます。

(2) -ScreenSpeed オプション

(a) 形式

```
-ScreenSpeed
-noScreenSpeed
```

(b) 機能

-ScreenSpeed

画面節（WINDOW SECTION または SCREEN SECTION）による画面機能を使用するとき，表示の見栄えや機能よりも表示速度を重視するためのオプションです。このオプションを指定すると，表示速度は速くなりますが，次の不都合が生じます。

- 文字間隔が狭くなる，縦けい線と文字が一部重なるなど，表示の見栄えが悪くなる。
- データ項目が画面上で重なりあっている場合の表示は保証しない。
- 縦倍文字，横倍文字に対するけい線の表示は保証しない。

- 画面節（WINDOW SECTION）の THROUGH 指定の LINE NUMBER 句，COLUMN NUMBER 句を指定すると，ERASE（ATTRIBUTE）文が正しく実行されない場合がある。
- 表示する項目が 1 行に収まらない場合の表示は保証しない。

-noScreenSpeed

-noScreenSpeed オプションの指定を打ち消します。

(c) 注意事項

- -ScreenSpeed オプションを指定してコンパイルした画面機能を含むプログラムと，-ScreenSpeed オプションを指定しないでコンパイルしたプログラムを一つの実行可能ファイル，DLL，または標準ライブラリ中に混在させないでください。

2.3.9 デバッグの設定

デバッグを設定するコンパイラオプションについて，説明します。

(1) -DebugLine オプション

(a) 形式

```
-DebugLine
-noDebugLine
```

(b) 機能

-DebugLine

原始プログラム中のデバッグ行をコンパイルの対象とします。デバッグ行とは，原始プログラムの標識領域にデバッグ標識（「D」または「d」）が記述されているか，浮動デバッグ指示子（「>>D」）が記述されている行のことです。

このシステムでは，WITH DEBUGGING MODE 句は覚え書きとみなされるので，デバッグ行を有効にするためにはこのオプションの指定が必要です。

-noDebugLine

-DebugLine オプションの指定を打ち消します。

(2) -DebugInf オプション

(a) 形式

```
-DebugInf [,Trace]
-noDebugInf
```

(b) 機能

-DebugInf

実行時に異常終了した場合，または実行時エラーが発生した場合，異常終了時要約情報リストを出力します。詳細は，マニュアル「COBOL2002 ユーザーズガイド」の「異常終了時要約情報リスト」を参照してください。

-DebugInf,Trace

異常終了時要約情報リスト中にトレースバック情報を出力します。

-noDebugInf

-DebugInf オプションの指定を打ち消します。

(c) 注意事項

- -DebugInf オプション，-TDInf オプション，-CVInf オプション，-DebugRange オプション，-DebugCompati オプション，-DebugData オプション，および-TestCmd オプションを指定していない場合，エラー／デバッグ情報（異常終了時要約情報リストやデータ領域ダンプリスト）が出力されません。
- -DebugInf オプションと-Compile,CheckOnly オプションを同時に指定した場合，-DebugInf オプションが無効となります。
- -DebugInf,Trace オプションを指定すると，実行時の処理速度が遅くなる可能性があります。このため，-DebugInf,Trace オプションは，デバッグの目的でコンパイルするときだけ指定するようにしてください。

(3) -DebugCompati オプション

(a) 形式

```
-DebugCompati
-noDebugCompati
```

(b) 機能

-DebugCompati

次のチェックをします。なお，表要素または部分参照された一意名に OCCURS DEPENDING ON の指定がある場合は，実行時点の制御変数の値を用いてチェックします。

- 表操作で使用する添字または指標名が指す表要素が表の範囲内であるかどうか。
- 部分参照の指定が一意名の範囲内であるかどうか。
- プログラム間で整合性が取れているかどうか。

ただし，実行時環境変数 CBLPRMCHKW に NOCHK を指定したときは，プログラム間で整合性が取れていなくてもエラーとしないで，実行を継続します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「プログラム間整合性チェック」を参照してください。

-noDebugCompati

-DebugCompati オプションの指定を打ち消します。

(c)　注意事項

- -DebugCompati オプションと-Compile,CheckOnly オプションを同時に指定した場合，-DebugCompati オプションが無効となります。
- -DebugCompati オプションを指定すると，-DebugInf オプションが仮定されます。
- -DebugCompati オプションを指定すると，実行時の処理速度が遅くなる可能性があります。このため，-DebugCompati オプションは，デバッグの目的でコンパイルするときだけ指定するようにしてください。
- -DebugCompati オプションを指定すると，一部の例外名に対する TURN 指令が無効となります。デバッグオプション指定時に無効となる例外名については，マニュアル「COBOL2002　ユーザーズガイド」の「例外検出での注意事項」を参照してください。

(4)　-DebugData オプション

(a)　形式

```
-DebugData〔,ValueHex〕
-noDebugData
```

(b)　機能

-DebugData

外部 10 進項目または内部 10 進項目に対して，格納値がデータ項目の属性と矛盾している場合，データ例外エラーを検出します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「データ例外検出機能」を参照してください。

-DebugData,ValueHex

データ例外検知機能でデータ例外が検出された場合，出力されるエラーメッセージにデータ項目の属性と矛盾している格納値を 16 進数で表示します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「データ例外検出機能」を参照してください。

-noDebugData

-DebugData オプションの指定を打ち消します。

(c) 注意事項

- -DebugData オプションと-Compile,CheckOnly オプションを同時に指定した場合，-DebugData オプションが無効となります。
- -DebugData オプションを指定すると，-DebugInf オプションが仮定されます。
- -DebugData オプションを指定すると，実行時の処理速度が遅くなる可能性があります。このため，-DebugData オプションは，デバッグの目的でコンパイルするときだけ指定するようにしてください。
- ValueHex サブオプションを指定した場合，-noDebugData オプションの指定がないかぎり，ValueHex サブオプションの指定は有効となります。ValueHex サブオプションの指定を打ち消すときは，-noDebugData オプションを指定してください。

  （例）

```
ccbl2002 -DebugData -DebugData,ValueHex -DebugData …
 （コンパイル結果）
  -DebugData,ValueHex

ccbl2002 -DebugData,ValueHex -noDebugData -DebugData …
 （コンパイル結果）
  -DebugData
```

(5) -TDInf オプション

(a) 形式

```
-TDInf
-noTDInf
```

(b) 機能

-TDInf

デバッグ情報を含むプログラム情報ファイル（.cbp）を出力します。また，除算例外については，除算実行前に検出します。

-TDInf オプションを指定したプログラムをテストデバッガでデバッグする場合，-DebugCompati オプションの指定の有無に関係なく，プログラム間の整合性がチェックされます。プログラム間の整合性チェックについては，「(3) -DebugCompati オプション」を参照してください。

-noTDInf

-TDInf オプションの指定を打ち消します。

(c) 注意事項

- -TDInf オプションと-Compile,CheckOnly オプションを同時に指定した場合，-TDInf オプションが無効となります。
- -TDInf オプションを指定すると，-DebugInf オプションが仮定されます。

- -TDInf オプションと-Optimize,3 オプションを同時に指定すると，-Optimize,3 オプションが無効になり，-Optimize,2 オプションが仮定されます。
- コンパイルとリンクを別々に実行する場合で，-TDInf オプションを指定してコンパイルしたときは，ccbl2002 コマンドでリンクを実行するときにも，同じオプションを指定する必要があります。
- -TDInf オプションを指定すると，実行時の処理速度が遅くなる可能性があります。このため，-TDInf オプションは，デバッグの目的でコンパイルするときだけ指定するようにしてください。
- -TDInf オプションはテストデバッガのためにゼロによる除算チェックなど最適化に影響を与えるオブジェクトコードを生成します。これによって，-TDInf オプションを指定することで，異常終了時要約リストなどの行番号／欄の情報が変化することがあります。行番号／欄の情報を常に正しく保つためには，-Optimize,0 オプションを指定する必要があります。

(6) -CVInf オプション

(a) 形式

```
-CVInf
-noCVInf
```

(b) 機能

-CVInf

カバレージ情報を含むプログラム情報ファイル（.cbp）を出力します。

-noCVInf

-CVInf オプションの指定を打ち消します。

(c) 注意事項

- -CVInf オプションと-Compile,CheckOnly オプションを同時に指定した場合，-CVInf オプションが無効となります。
- -CVInf オプションを指定すると，-DebugInf オプションが仮定されます。
- コンパイルとリンクを別々に実行する場合で，-CVInf オプションを指定してコンパイルしたときは，ccbl2002 コマンドでリンクを実行するときにも，同じオプションを指定する必要があります。
- -CVInf オプションを指定すると，実行時の処理速度が遅くなる可能性があります。このため，-CVInf オプションは，デバッグの目的でコンパイルするときだけ指定するようにしてください。
- -CVInf オプションを指定してカバレージ情報を出力できるのは，ccbl2002 コマンドでコンパイルして出力したプログラム情報ファイルだけです。ccbl コマンドで出力したプログラム情報ファイルを使用して，-CVInf オプションを指定してコンパイルしたときは，プログラム情報ファイルにカバレージ情報を出力しません。

(7) -DebugRange オプション

(a) 形式

```
-DebugRange
-noDebugRange
```

(b) 機能

-DebugRange

表操作で使用する添字および指標名の値が，各次元ごとに OCCURS 句で指定した反復回数の範囲内であるかどうかを，実行時に調べます。

なお，表要素に OCCURS DEPENDING ON の指定がある場合は，実行時点の制御変数の値を用いて調べます。

-noDebugRange

-DebugRange オプションの指定を打ち消します。

(c) 注意事項

- -DebugRange オプションと-Compile,CheckOnly オプションを同時に指定した場合，-DebugRange オプションが無効となります。
- -DebugRange オプションを指定すると，-DebugInf オプションおよび-DebugCompati オプションが仮定されます。
- -DebugRange オプションを指定すると，実行時の処理速度が遅くなる可能性があります。このため，-DebugRange オプションは，デバッグの目的でコンパイルするときだけ指定するようにしてください。
- -DebugRange オプションを指定すると，一部の例外名に対する TURN 指令が無効となります。デバッグオプション指定時に無効となる例外名については，マニュアル「COBOL2002　ユーザーズガイド」の「例外検出での注意事項」を参照してください。

(8) -TestCmd オプション

(a) 形式

```
-TestCmd {,Full|,Break|,Sim} +
-noTestCmd
```

(b) 機能

TD コマンド格納ファイルに出力する情報の種類を指定します。

-TestCmd,Full

中断点情報，シミュレーション情報の TD コマンドを TD コマンド格納ファイル（.tdi）に出力します。

-TestCmd,Break

中断点情報の TD コマンドを TD コマンド格納ファイル（.tdi）に出力します。

-TestCmd,Sim

シミュレーション情報の TD コマンドを TD コマンド格納ファイル（.tds）に出力します。

-noTestCmd

-TestCmd オプションの指定を打ち消します。

(c) 注意事項

- -TestCmd,Full オプションと-TestCmd,Break オプション，または-TestCmd,Full オプションと-TestCmd,Sim オプションを重複して指定した場合，-TestCmd,Full オプションが有効となります。
- -TestCmd オプションを指定すると，-DebugInf および-TDInf オプションが仮定されます。
- コンパイルとリンクを別々にする場合で，コンパイル時に-TestCmd オプションを指定したときは，ccbl2002 コマンドでのリンク時に-TDInf オプションを指定する必要があります。

(9) -SimMain オプション

(a) 形式

```
-SimMain プログラム名
-noSimMain
```

(b) 機能

-SimMain プログラム名

テストデバッガの主プログラムシミュレーション機能を使用します。

プログラム名には，主プログラムシミュレーションの対象となる副プログラムの名称（最外側のプログラムのプログラム名段落で指定した名称）を指定します。

このオプションを指定すると，主プログラムシミュレーション用の一時的な COBOL ソースファイル（擬似主プログラム）が内部的に生成され，コンパイルされます。コンパイルによって生成されたオブジェクトファイルは，リンク時に取り込まれます。ここで生成されたソースファイル，およびオブジェクトファイルは，コンパイル終了時に消去されます。

また，テストデバッガで必要な擬似主プログラム用のプログラム情報ファイル（.cbs）が生成されます。テストデバッガの主プログラムシミュレーション機能を使用するとき，生成されたプログラム情報ファイル（.cbs）をテストデバッガ環境に指定してください。生成するプログラム情報ファイルの名称については，「(12) 主／副プログラムシミュレーションで生成する擬似プログラム用プログラム情報ファイル」を参照してください。

擬似主プログラムを生成する COBOL ソースファイルの情報は，プログラム情報ファイル（.cbp）から取得されます。このため，このオプションを指定する場合は，次のどちらかをする必要があります。

- -TDInf オプションを同時に指定してコンパイルする。

- 副プログラムを含む COBOL ソースファイルを-TDInf オプションを指定してコンパイルし，生成された.cbp ファイルを，環境変数 CBLPIDIR で指定したフォルダまたはカレントフォルダに入れておく。

Windows(x86) COBOL2002 の場合，DLL を呼び出す擬似主プログラムを作成するときは，次のオプションを同時に指定してコンパイルしてください。

- DLL の属性が stdcall 呼び出し規約の場合：-Dll,Stdcall
- DLL の属性が cdecl 呼び出し規約の場合：-Dll,Cdecl

このとき，-OutputFile オプションで実行可能ファイル名を指定しない場合は，生成される主プログラムの実行可能ファイル名は NONAME1.exe となります。

Windows(x64) COBOL2002 の場合，DLL を呼び出す擬似主プログラムを作成する場合は，-Dll オプションを同時に指定してコンパイルしてください。

このとき，-OutputFile オプションで実行可能ファイル名を指定しない場合は，生成される主プログラムの実行可能ファイル名は NONAME1.exe となります。

注意事項

下記に示すデータ項目を主プログラムシミュレーションの対象となる副プログラムの手続き部見出しに指定することはできません。

- プロパティ項目
- 関数一意名

-noSimMain

-SimMain オプションの指定を打ち消します。

(10) -SimSub オプション

(a) 形式

```
-SimSub プログラム名〔,…〕
-noSimSub
```

(b) 機能

-SimSub プログラム名〔,…〕

テストデバッガの副プログラムシミュレーション機能を使用します。

プログラム名には，副プログラムシミュレーションの対象となる副プログラムの名称（CALL 文で指定した名称）を指定します。ただし，副プログラムの呼び出し方法は，CALL 文の定数指定呼び出しでなければなりません。

このオプションを指定すると，副プログラムシミュレーション用の一時的な COBOL ソースファイル（擬似副プログラム）が内部的に生成され，コンパイルされます。コンパイルによって生成されたオブジェクトファイルは，リンク時に取り込まれます。ここで生成されたソースファイル，およびオブジェクトファイルは，コンパイル終了時に消去されます。

また，テストデバッガで必要な擬似副プログラム用のプログラム情報ファイル（.cbs）が生成されます。テストデバッガの副プログラムシミュレーション機能を使用するとき，生成されたプログラム情報ファイル（.cbs）をテストデバッガ環境に指定してください。生成するプログラム情報ファイルの名称については，「(12)　主／副プログラムシミュレーションで生成する擬似プログラム用プログラム情報ファイル」を参照してください。

擬似副プログラムを生成する COBOL ソースファイルの情報は，プログラム情報ファイル（.cbp）から取得されます。このため，このオプションを指定する場合は，次のどちらかをする必要があります。

- -TDInf オプションを同時に指定してコンパイルする。
- 副プログラムを呼び出している COBOL ソースファイルを-TDInf オプションを指定してコンパイルし，生成された.cbp ファイルを，環境変数 CBLPIDIR で指定したフォルダまたはカレントフォルダに入れておく。

注意事項

- このオプションは，-Dll オプションと同時に指定できません。また，呼び出し属性が stdcall となる副プログラムシミュレーションはできません。
- 下記に示すデータ項目を副プログラムシミュレーションの対象となるプログラムに指定した CALL 文の引数に指定することはできません。

  ・プロパティ項目

  ・関数一意名
- DLL に，擬似副プログラムを含めることはできません。

-noSimSub

-SimSub オプションの指定を打ち消します。

(11)　-SimIdent オプション

(a)　形式

```
-SimIdent
-noSimIdent
```

(b)　機能

-SimIdent

テストデバッガの副プログラムシミュレーション機能を使用します。この場合，副プログラムシミュレーションの対象となるのは，一意名呼び出しの CALL 文を指定したプログラムです。

このオプションを指定すると，副プログラムシミュレーション用の一時的な COBOL ソースファイルが内部的に生成され，コンパイルされます。コンパイルによって生成されたオブジェクトファイルは，リンク時に取り込まれます。ここで生成されたソースファイル，およびオブジェクトファイルは，コンパイル終了時に消去されます。

また，テストデバッガで必要な擬似副プログラム用のプログラム情報ファイル（.cbs）が生成されます。テストデバッガの副プログラムシミュレーション機能を使用するとき，生成されたプログラム情報ファイル（.cbs）をテストデバッガ環境に指定してください。生成するプログラム情報ファイルの名称については，「(12)　主／副プログラムシミュレーションで生成する擬似プログラム用プログラム情報ファイル」を参照してください。

擬似副プログラムを生成する COBOL ソースファイルの情報は，プログラム情報ファイル（.cbp）から取得されます。このため，このオプションを指定する場合は，次のどちらかをする必要があります。

- -TDInf オプションを同時に指定してコンパイルする。
- 副プログラムを呼び出している COBOL ソースファイルを-TDInf オプションを指定してコンパイルし，生成された.cbp ファイルを，環境変数 CBLPIDIR で指定したフォルダまたはカレントフォルダに入れておく。

注意事項

- このオプションは，-Dll オプションと同時に指定できません。また，呼び出し属性が stdcall となる副プログラムシミュレーションはできません。
- 下記に示すデータ項目を副プログラムシミュレーションの対象となるプログラムに指定した CALL 文の引数に指定することはできません。

  ・プロパティ項目

  ・関数一意名
- DLL に，擬似副プログラムを含めることはできません。

-noSimIdent

-SimIdent オプションの指定を打ち消します。

(12)　主／副プログラムシミュレーションで生成する擬似プログラム用プログラム情報ファイル

主／副プログラムシミュレーションでは，テストデバッガで必要な情報である擬似プログラム用プログラム情報ファイル（.cbs）が生成されます。擬似プログラム用プログラム情報ファイル（.cbs）は，カレントフォルダ，または環境変数 CBLPIDIR で指定したフォルダに出力されます。

生成される擬似プログラム用プログラム情報ファイル（.cbs）の名称規則を，次に示します。

- -SimMain オプションによって，擬似主プログラムシミュレーションをする場合
  SimMain@（-SimMain オプションで指定したプログラム名）.cbs
- -SimSub オプションによって，定数指定の CALL 文の擬似副プログラムシミュレーションをする場合
  SimSub@（-SimSub オプションで指定したプログラム名※）.cbs

  注※

  プログラム名を複数指定した場合，先頭のプログラム名が使用されます。
- -SimIdent オプションによって，一意名指定の CALL 文の擬似副プログラムシミュレーションをする場合

SimIdent@001.cbs

2.3.10　リンクの設定

リンクを設定するコンパイラオプションについて，説明します。

(1)　-StdCall オプション(Windows(x86) COBOL2002 で有効)

(a)　形式

```
-StdCall
-noStdCall
```

(b)　機能

-StdCall

stdcall 呼び出し指示ファイル（.cbw）を読み込むためのオプションです。stdcall 呼び出し指示ファイルは，CALL 文で最外側のプログラムを呼び出すときの属性が stdcall 呼び出し規約となるように指示するファイルです。したがって，cdecl 呼び出し規約で呼び出す最外側のプログラム名を，stdcall 呼び出し指示ファイルに指定できません。

このオプションを指定しない場合，stdcall 呼び出し指示ファイルは，読み込まれません。また，呼び出し属性は，CALL-CONVENTION 段落の指定に従います。

-noStdCall

-StdCall オプションの指定を打ち消します。

(2)　-StdCallFile オプション(Windows(x86) COBOL2002 で有効)

(a)　形式

```
-StdCallFile ファイル名
-noStdCallFile
```

(b)　機能

-StdCallFile ファイル名

-StdCall オプションで読み込む stdcall 呼び出し指示ファイル（.cbw）の名称を指定します。

このオプションを指定した場合，stdcall 呼び出し指示ファイルに指定したファイル名だけが参照され，「COBOL ソースファイル名.cbw」は参照されません。このオプションを指定しないときは，「COBOL ソースファイル名.cbw」だけが参照されます。

-noStdCallFile

-StdCallFile オプションの指定を打ち消します。

(3) -Lib オプション

(a) 形式

```
-Lib, {GUI | CUI}
```

(b) 機能

-Lib,GUI

GUI モードのユーザプログラムを作成します。

-Lib オプションを指定しなかった場合，このオプションが仮定されます。

-Lib,CUI

CUI モードのユーザプログラムを作成します。

(4) ライブラリの指定

インポートライブラリ，標準ライブラリを指定します。

このオプションは，開発マネージャだけで指定できます。

(5) リンケージを行わない

リンケージ処理をしない場合に指定します。

リンケージ処理をしない場合は，コンパイル処理によってオブジェクトファイルが生成されますが，リンケージ処理は行われません。

擬似主プログラム生成，擬似副プログラム生成を指定する場合は，この項目を指定しないでください。

このオプションは，開発マネージャだけで指定できます。

(6) -Compile オプション

(a) 形式

```
-Compile, {CheckOnly | NoLink}
-noCompile
```

(b) 機能

プログラムのコンパイル時，オブジェクトファイル（.obj），実行可能ファイル（.exe），および DLL（.dll）の生成を抑止します。

-Compile,CheckOnly

COBOL プログラムのコンパイル時にエラーチェックだけをします。オブジェクトファイル，実行可能ファイル，および DLL は出力しません。

このオプションを指定すると，実行可能ファイルを生成しないため，コンパイル速度が向上します。単にエラーチェックだけをしたい場合に指定します。

-Compile,NoLink

オブジェクトファイルは出力しますが，実行可能ファイル，DLL は生成しません。出力したオブジェクトファイルは，あとでリンカを使用して実行可能ファイル，または DLL にできます。

-noCompile

-Compile オプションの指定を打ち消します。

(c) 注意事項

- -Compile オプションを指定した場合，-OutputFile オプションの指定は無効となります。

  また，-Compile,CheckOnly オプションを指定した場合，アプリケーションデバッグ機能に関連するオプション（-DebugInf，-DebugCompati，-DebugData，-TDInf，-CVInf，-DebugRange または-Testcmd オプション）が指定されても，無効となります。
- このオプションは，コマンドライン（ccbl2002 コマンド）で指定できます。開発マネージャでは，指定できません。

(7) -DefFile オプション

(a) 形式

```
-DefFile ファイル名
-noDefFile
```

(b) 機能

-DefFile ファイル名

COBOL2002 が自動生成するモジュール定義ファイル（.def）の名称を指定します。

モジュール定義ファイルは，カレントフォルダに出力されるため，パス名を付けて指定してはいけません。パス名を付けて指定した場合，動作は保証しません。

このオプションを指定しない場合，生成されるモジュール定義ファイル名は，生成される DLL ファイルの名称に拡張子.def を付けた名称になります。

-noDefFile

-DefFile オプションの指定を打ち消します。

(c) 注意事項

- このオプションは，コマンドライン（ccbl2002 コマンド）で指定できます。開発マネージャでは，指定できません。

(8) -IconFile オプション

(a) 形式

```
-IconFile ファイル名
-noIconFile
```

(b) 機能

-IconFile ファイル名

生成する実行可能ファイルのアイコンを変更します。

このオプションを指定すると，COBOL2002 が自動生成するリソース定義ファイル（.rc）中に指定するアイコンを，オプションで指定したアイコンファイルに変更します。ファイル名は，パス名を付けて指定することもできます。

このオプションを指定しないときは，COBOL2002 が用意しているアイコンが設定されます。

-noIconFile

-IconFile オプションの指定を打ち消します。

(c) 注意事項

- このオプションは，コマンドライン（ccbl2002 コマンド）で指定できます。開発マネージャでは，指定できません。

(9) -ResrcFile オプション

(a) 形式

```
-ResrcFile ファイル名
-noResrcFile
```

(b) 機能

-ResrcFile ファイル名

COBOL2002 が自動生成するリソース定義ファイル（.rc）の名称を指定します。

リソース定義ファイルは，カレントフォルダに出力されるため，パス名を付けて指定してはいけません。パス名を付けて指定した場合，動作は保証しません。

このオプションを指定しない場合，生成されるモジュール定義ファイル名は，生成される実行可能ファイルの名称に拡張子.rc を付けた名称になります。

-noResrcFile

-ResrcFile オプションの指定を打ち消します。

(c) 注意事項

- このオプションは，コマンドライン（ccbl2002 コマンド）で指定できます。開発マネージャでは，指定できません。

(10) -OutputFile オプション

(a) 形式

```
-OutputFile ファイル名
-noOutputFile
```

(b) 機能

-OutputFile ファイル名

生成する実行可能ファイル，または DLL ファイルの名称を指定します。

ファイル名には，生成する実行可能ファイル，または DLL ファイルの名称を指定します。カレントフォルダに出力する場合は，ファイル名だけを指定します。また，任意のフォルダに出力する場合は，ドライブ名からの絶対パス名で指定します。

このオプションの指定がない場合，次の規則に従って実行可能ファイル，または DLL ファイル名が決定します。

（実行可能ファイルを生成する場合）

-Main オプションに指定したファイル名に拡張子（.exe）を付けたファイル名になります。-Main オプションの指定がない場合，ccbl2002 コマンド中で-Main が仮定されたソースファイル名（ccbl2002 コマンド中で最初にコンパイルされたソースファイル名）に拡張子.exe を付けた名称となります。このとき，出力先のフォルダも上記のソースファイルと同じになります。

（DLL ファイルを生成する場合）

ccbl2002 コマンド中で最初にコンパイルされたソースファイル名称に拡張子.dll を付けた名称となります。このとき，出力先のフォルダも上記のソースファイルと同じになります。

なお，-SimMain オプション，-SimSub オプション，または-SimIdent オプションを指定した場合については，-SimMain オプション，-SimSub オプション，-SimIdent オプションの説明をそれぞれ参照してください。

-OutputFile オプションと-Compile オプションを同時に指定した場合，-OutputFile オプションは，無効となります。

-noOutputFile

-OutputFile オプションの指定を打ち消します。

(c) 注意事項

- ccbl2002 コマンドは，-OutputFile に指定したファイル名の拡張子が.exe または.dll であるかどうかのチェックをしません。.exe または.dll 以外の拡張子を指定した場合，動作は保証しません。
- このオプションは，コマンドライン（ccbl2002 コマンド）で指定できます。開発マネージャでは，指定できません。

(11) -Link オプション

(a) 形式

```
-Link オプションの並び
-noLink
```

(b) 機能

-Link オプションの並び

リンカに渡すオプションの並びを指定します。オプションとオプションの間は，空白で区切ります。

-noLink

-Link オプションの指定を打ち消します。

(c) 注意事項

- -Link オプション以降の文字は，ほかのオプション名やファイル名なども含めすべてリンカに渡されます。したがって，このオプションはオプション群の最後に指定してください。
- コマンドファイル中に-Link オプションの指定がある場合は，その行の-Link オプション以降がリンカに渡され，次の行から別のオプションとして扱われます。
- -Link オプションは，-Details オプションでのコンパイラオプションの詳細情報表示の対象にはなりません。
- このオプションを開発マネージャで指定する場合，プロジェクト設定ダイアログボックスの［ユーザ設定］タブを使用してください。［ユーザ設定］タブの詳細は，「2.2.3　オプションの設定方法」を参照してください。
- オプションの並びに-OUT リンカオプションを指定して，生成する実行可能ファイル，または DLL ファイルの名称を指定しないでください。生成されたファイルの動作は保証しません。

(12) -DynamicLink オプション

(a) 形式

```
-DynamicLink,Call
-noDynamicLink
```

(b) 機能

-DynamicLink,Call

定数指定の CALL 文によってプログラムを呼び出す場合，呼び出されるプログラムが静的にリンクされていないときに，動的なリンク（ダイナミックリンク）が設定されます。

動的なリンク（ダイナミックリンク）については，マニュアル「COBOL2002　ユーザーズガイド」の「プログラムの呼び出し」を参照してください。

なお，COBOL2002 では，ccbl コマンドでコンパイルする場合に，-Bd オプションを指定できます。

-noDynamicLink

-DynamicLink オプションの指定を打ち消します。

(13) -ManifestFileExt オプション

(a) 形式

```
-ManifestFileExt
-noManifestFileExt
```

(b) 機能

-ManifestFileExt

マニフェストファイルを実行可能ファイル，または DLL に埋め込まずに，外部マニフェストファイルとします。

COBOL2002 コンパイラは，LINK コマンド実行時に生成されたマニフェスト情報※を，MT コマンドを使用して実行可能ファイルや DLL に埋め込みますが，このオプションを使用するとマニフェスト情報を外部マニフェストファイルとして出力できます。

外部マニフェストファイルの出力先フォルダは，生成する実行可能ファイル，または DLL ファイルと同じになります。

マニフェスト情報の追加，または変更をしたい場合，生成した外部マニフェストファイルを直接テキストエディタで編集するか，または追加情報のマニフェストファイルを作成して MT コマンドの機能によってマニフェスト情報をマージします。

なお，MT コマンドがマニフェスト情報の埋め込みに失敗して，COBOL2002 コンパイラがエラーとなることがありますが，このオプションを使用すると，MT コマンドの実行を抑止してエラーを回避できます。

注※

COBOL2002 が同梱する LINK コマンドは，使用する C 実行時ライブラリの情報や UAC 情報をマニフェスト情報として生成します。

なお，マニフェスト情報の種類や詳細については，Microsoft のリファレンスを参照してください。

-noManifestFileExt

-ManifestFileExt オプションの指定を打ち消します。

(c) 注意事項

- このオプションと-Compile オプションを同時に指定した場合，このオプションは無効となります。
- このオプションは，開発マネージャ上で出力した情報リストのコンパイルリストの情報リストのオプションには表示しません。
- このオプションを指定した場合，外部マニフェストファイル名は「実行可能ファイル名.manifest」，および「DLL ファイル名.manifest」となります。そのため，ファイルの出力先フォルダに同名のファイルがあると上書きされます。
- 実行可能ファイルに対応する外部マニフェストファイルは，必ず実行可能ファイルと同じフォルダに配置してください。実行可能ファイルと同じフォルダに配置されない場合，使用する C 実行時ライブラリのバージョンによっては実行可能ファイルが正しく動作しません。
- DLL ファイルを作成するとき，このオプションを指定して生成されるマニフェストファイルは DLL ファイルに埋め込んで使用しなければなりません。マニフェストの埋め込みについては，マニュアル「COBOL2002　ユーザーズガイド」の「コンパイルとリンクを別々に実行する方法」を参照してください。

  また，使用する C 実行時ライブラリのバージョンによってはマニフェストファイルは生成されないことがあります。

  生成されない場合は，マニフェストファイルがなくても動作に問題ありません。

2.3.11　規格の設定

規格との仕様チェックを設定するコンパイラオプションについて，説明します。

(1)　-StdMIA オプション

(a)　形式

```
-StdMIA {,13 | ,14} +
-noStdMIA
```

(b)　機能

仕様チェック機能として，MIA 仕様※の範囲外チェックをします。

仕様範囲外の記述があると，警告レベルのエラーメッセージが出力されます。

注※

MULTIVENDOR INTEGRATION ARCHITECTURE VERSION 1.3 および 1.4（日本電信電話株式会社）の仕様（JIS X 3002-1988　電子計算機プログラム COBOL の仕様が前提）

-StdMIA,13

MIA バージョン 1.3 仕様の範囲外チェックをします。

-StdMIA,14

MIA バージョン 1.4 仕様の範囲外チェックをします。

なお，このオプションを指定すると，-StdMIA,13 サブオプションを仮定します。

-noStdMIA

-StdMIA オプションの指定を打ち消します。

(c) 注意事項

- -StdMIA オプション，-Std85 オプション，-Std2002 オプションは，同時に指定できません。同時に指定した場合は，警告のメッセージが出力され，次の優先順位に従ってオプションが有効となります。
  1. -Std2002 オプション
  2. -Std85 オプション
  3. -StdMIA オプション
- 次のオプションは，-StdMIA オプションと背反となるため，同時に指定できません。同時に指定した場合は，-StdMIA オプションが有効となり，次のオプションが無効となります。

  -V3Spec -StdVersion -CompatiM7 -CompatiV3 -H8Switch

  -Cblctr -IgnoreLCC -JPN -CmDol -Comp5 -NumAccept

  -V3Rec -EquivRule,NotAny -SQL -BinExtend -MaxDigits38 -IntResult,DecFloat40 -LiteralExtend -V3RecFCSpace -SQLDisp

  -V3RecEased
- -StdMIA,13 オプションは，次のオプションと背反となるため，同時に指定できません。同時に指定した場合は，-StdMIA,13 オプションが有効となり，次のオプションが無効となります。

  -EquivRule,StdCode -Bin1Byte
- このオプションは，自由形式正書法で書かれた COBOL 原始プログラムをコンパイルする場合には指定できません。指定した場合，エラーとなってコンパイルが中止されます。

(2) -Std85 オプション

(a) 形式

```
-Std85 {, {High | Middle | Low } |,Obso |,Report} +
-noStd85
```

(b) 機能

仕様チェック機能として，JIS 仕様※の範囲外チェックや廃要素のチェックをします。

仕様範囲外の記述や廃要素があると，警告レベルのエラーメッセージが出力されます。

注※

JIS X 3002-1992　電子計算機プログラム COBOL の仕様

-Std85,High

JIS 仕様の範囲外チェックをします。

-Std85,Middle

JIS 仕様の中位集合範囲外（上位集合，JIS 仕様の範囲外）チェックをします。

-Std85,Low

JIS 仕様の下位集合範囲外（中位集合，上位集合，JIS 仕様の範囲外）チェックをします。

-Std85,Obso

JIS 仕様の廃要素のチェックをします。

-Std85,Report

JIS 仕様の報告書作成機能のチェックをします。

-noStd85

-Std85 オプションの指定を打ち消します。

(c) 注意事項

- -Std85 オプションは，次のオプションと背反となるため，同時に指定できません。同時に指定した場合は，-Std85 オプションが有効となり，次のオプションが無効となります。

  -V3Spec -StdVersion -CompatiM7 -CompatiV3 -H8Switch -Cblctr
  -IgnoreLCC -JPN -CmDol -Comp5 -NumAccept -V3Rec
  -EquivRule,NotAny -SQL -BinExtend -MaxDigits38 -IntResult,DecFloat40
  -LiteralExtend -V3RecFCSpace -SQLDisp -V3RecEased
- このオプションは，自由形式正書法で書かれた COBOL 原始プログラムをコンパイルする場合には指定できません。指定した場合，エラーとなってコンパイルが中止されます。

(3) -Std2002 オプション

(a) 形式

```
-Std2002 {,OutRange|,Obso|,Archaic} +
-noStd2002
```

(b) 機能

仕様チェック機能として，COBOL2002 仕様の範囲外チェックや廃要素のチェックをします。

仕様範囲外の記述や廃要素があると，警告レベルのエラーメッセージが出力されます。

-Std2002,OutRange

COBOL2002 仕様の範囲外チェックをします。

-Std2002,Obso

COBOL2002 仕様の廃要素のチェックをします。

-Std2002,Archaic

COBOL2002 仕様の古典的要素のチェックをします。

-noStd2002

-Std2002 オプションの指定を打ち消します。

(c) 注意事項

- -Std2002 オプションは，次のオプションと背反関係にあり同時に指定した場合は，-Std2002 オプションを有効とする。

  -V3Spec -StdVersion -CompatiM7 -CompatiV3 -H8Switch -Cblctr
  -IgnoreLCC -JPN -CmDol -Comp5 -NumAccept -V3Rec
  -EquivRule,NotAny -SQL -BinExtend -MaxDigits38 -IntResult,DecFloat40
  -LiteralExtend -V3RecFCSpace -SQLDisp -V3RecEased

(4) -StdVersion オプション

第 1 次または第 2 次規格の言語仕様の解釈でコンパイルするためのオプションです。このオプションを指定しないときは，第 3 次規格以降の解釈でコンパイルされます。

(a) 形式

```
-StdVersion, {1 | 2}
-noStdVersion
```

(b) 機能

-StdVersion,1

第 1 次規格（JIS72，ISO72，ANSI68）の解釈でコンパイルします。

-StdVersion,2

第 2 次規格（JIS80，ISO78，ANSI74）の解釈でコンパイルします。

-noStdVersion

-StdVersion オプションの指定を打ち消します。

このオプションを指定した場合，第 3 次規格以降の解釈※でコンパイルします。

注※

第 3 次規格以降の解釈とは，次の二つの規格の解釈を指します。

- 第 3 次規格（ISO85，ANSI85，JIS88，JIS92）
- 第 4 次規格（ISO/IEC 1989:2002，JIS X3002:2011）

-StdVersion オプション指定時，および未指定時の言語仕様の相違を，次に示します。

差異の生じる言語仕様	第 1 次規格	第 2 次規格	第 3 次規格以降
比較条件の略記法の NOT	略記法の条件の中の NOT が比較演算子の一部とも取れるときには論理演算子とみなす。	NOT の直後に GREATER，>，LESS，<，EQUAL，=，>=，<=のどれかが続くとき，NOT は比較条件の一部となる。これ以外の NOT は論理演算子とみなす。	
MOVE 文の位取り	送り出し側作用対象が位取りした整数項目（PICTURE 句の右端文字が P）で，受け取り側作用対象が英数字編集項目の場合，あとに続くゼロを切り捨てて空白とする。	左記の場合，あとに続くゼロを切り捨てない。	
UNSTRING 文の DELIMITED BY ALL 指定	複数の連続する区切り文字が現れた場合，区切り文字の受け取り領域に入るだけ区切り文字を移す。	左記の場合，区切り文字の受け取り領域には 1 個（1 組）の区切り文字しか移さない。	
COPY 文の展開	第 1 次規格の仕様に従った書き方は旧仕様として展開する。	第 2 次規格以降の仕様に従って COPY 文を展開する。	
JUST 句の処理	JUST 句の指定がある英数字，英字，英数字編集に VALUE 句で初期値を与える場合，JUST 句の機能を働かせる。	左記の場合，JUST 句の機能を働かせない。	
WRITE 文の ADVANCING の仮定	1 ファイルに対する複数個の WRITE 文で ADVANCING 指定があるものとないものを混用した場合，ADVANCING 指定のないものに対しては BEFORE ADVANCING 1 か AFTER ADVANCING 1 のどちらかを仮定する。どちらを仮定するかは WRITE 文の指定の内容による。	左記の場合，常に AFTER ADVANCING 1 を仮定する。	
PERFORM 文の VARYING AFTER 指定がある場合の反復制御変数を初期化する位置	• TEST AFTER 指定時 外側のループの反復制御変数を BY 指定の値で増加させてから，内側のループの反復制御変数を FROM 指定の変数で再び初期化する。 • TEST BEFORE 指定時 内側のループの反復制御変数 FROM 指定の変数で初期化してから，外側のループの反復制御変数を BY 指定の値で増加させる。		TEST AFTER 指定時も TEST BEFORE 指定時も，外側のループの反復制御変数を BY 指定の値で増加させてから，内側の反復制御変数を FROM 指定で再び初期化する。
SELECT OPTIONAL 句	覚え書きとする。	覚え書きとしない。	
CURRENCY SIGN 句に指定できる定数	"="，"/"が指定できる。	"="，"/"は指定できない。	

(c) 注意事項

- このオプションは，自由形式正書法で書かれた COBOL 原始プログラムをコンパイルする場合には指定できません。指定した場合，エラーとなってコンパイルが中止されます。

2.3.12　他システムとの移行の設定

他システムとの移行に関連するコンパイラオプションについて，説明します。

(1)　-CompatiM7 オプション(Windows(x86) COBOL2002 で有効)

(a)　形式

```
-CompatiM7
-noCompatiM7
```

(b)　機能

-CompatiM7

MIOS7 COBOL85 との互換機能を有効にします。

-noCompatiM7

-CompatiM7 オプションの指定を打ち消します。

(2)　-CompatiV3 オプション

(a)　形式

```
-CompatiV3
-noCompatiV3
```

(b)　機能

-CompatiV3

メインフレーム（VOS3）COBOL85 との互換を指定するオプションです。-CompatiV3 オプションは，VOS3 COBOL85 で開発した COBOL プログラムを Windows 環境に移行することを目的として提供しているオプションで，日立拡張機能の中で使用頻度の高いものを対象としています。

このオプションを指定した場合，指定しない場合と比べて次の点が異なります。

- ASSIGN 句で指定した外部装置名に関連する環境変数名が，次のように変更されます。

(書き方)

```
SYSnnn〔-装置クラス〕〔-装置名〕〔-編成〕
または，〔装置クラス-〕〔装置名-〕〔編成-〕ファイル定義名
```

- -CompatiV3 オプション指定時の環境変数名

  次のどちらかとなる。

  CBL_SYSnnn

  CBL_ファイル定義名

- -CompatiV3 オプション未指定時の環境変数名

  次のどちらかとなる。

  CBL_SYSnnn〔_装置クラス〕〔_装置名〕〔_編成〕

  CBL_〔装置クラス_〕〔装置名_〕〔編成_〕ファイル定義名

- OPEN I-O で開かれた順ファイルに対して WRITE 文が指定できます。このとき，次の条件をすべて満たすと，WRITE 文は REWRITE 文として扱われます。

(条件)

- -CompatiV3 オプションの指定がある。
- EXTERNAL 句が指定されているか，または OPEN I-O で開かれている。
- WRITE 文に ADVANCING 指定がない。
- LINAGE 句の指定がない。

- -CompatiV3 オプションを指定した場合，次のオプションが仮定されます。

(仮定されるオプション)

- -V3Rec,Variable

  可変長形式の原始プログラムのコンパイルを，メインフレームの日本語項目の扱いに合わせるオプションです。

  なお，固定長形式でコンパイルする場合，-CompatiV3 オプションと-V3Rec,Fixed オプションを指定する必要があります。

- -JPN,Alnum

  日本語項目，日本語編集項目，および日本語文字定数を，それぞれ英数字項目，英数字編集項目，および英数字定数として扱うオプションです。

  VOS3 COBOL85 で XCOBOL=(N)を指定したときと同じ動作となります。

- VOS3 COBOL85 上で使用できる報告書作成機能を記述したプログラムをコンパイルできます。

  なお，-CompatiV3 オプションを指定しない場合，PC COBOL85 と同じ言語仕様の報告書作成機能を使用します。

- 通信節による画面機能で，複数プログラムで通信文が実行された場合，一つの仮想端末を複数のプログラム間で共用し，送受信します。

- DIVIDE 文書き方 4 および書き方 5 の，除算の剰余を計算するのに使う商は，商を計算したときの中間結果と同じけた数，同じ小数点位置，同じ符号の有無を持ちます。

(例)

```
DIVIDE A BY B GIVING C REMAINDER D.
```

上記の例の DIVIDE 文は，下記のような連続した操作に変換されます（ただし，TMP1，TMP2，TMP3 はコンパイラによって作成された中間結果の記憶場所を表します)。

```
A/B → TMP1
TMP1 → TMP2
A - (B * TMP2) → TMP3
```

```
TMP1 → C
TMP3 → D
```

- -CompatiV3 オプションを指定したときは，TMP2 は TMP1 と同じけた数，同じ小数点位置，同じ符号になります。
- -CompatiV3 オプション未指定のときは，TMP2 は商（C）と同じけた数，同じ小数点位置，同じ符号になります。

- -CompatiV3 オプションを指定した場合，英数字定数の分離符には，アポストロフィ（'）だけが使用できます。
- -CompatiV3 オプションと-EquivRule オプションは，同時に指定できません。指定した場合，あとに指定したオプションが有効となります。
- Windows(x64) COBOL2002 の場合，-CompatiV3 オプションおよび-IntResult,DecFloat40 オプションを同時に指定したとき，算術式および算術文に対する-CompatiV3 オプションの仕様（DIVIDE 文書き方 4 および書き方 5）は，無効となります。この場合，-IntResult,DecFloat40 オプションの仕様で演算をします。-IntResult,DecFloat40 オプションについては，「2.3.14　その他の設定」の「(19)　-IntResult,DecFloat40 オプション（Windows(x64) COBOL2002 で有効)」を参照してください。

-noCompatiV3

-CompatiV3 オプションの指定を打ち消します。

-CompatiV3 オプションを指定した場合，-V3Rec,Variable オプションおよび-JPN,Alnum オプションが仮定されますが，-noCompatiV3 オプションを指定しても，仮定された-V3Rec,Variable オプションおよび-JPN,Alnum オプションは打ち消されません。

(c)　注意事項

- このオプションは，自由形式正書法で書かれた COBOL 原始プログラムをコンパイルする場合には指定できません。指定した場合，エラーとなってコンパイルが中止されます。
- このオプションを指定した場合，連結式を記述できません。
- コンパイラ環境変数 CBLV3UNICODE に YES を指定し，-UniObjGen オプションを指定すると，このオプションを指定しても-JPN,Alnum オプションは仮定されません。
- このオプションを指定した場合，特殊名段落の定数 10 が有効になります。ただし，コンパイラ環境変数 CBLV3UNICODE に YES を指定すると，-UniObjGen オプションとこのオプションの同時指定ができます。そのため，Unicode で多バイトになる文字を指定した場合，コンパイルエラーとなります。

(3)　-Compati85 オプション

(a)　形式

```
-Compati85 {,IoStatus|,Linage|,Call|,Power|,Syntax|,IDParag|,RsvWord|,NoPropagate|,A
ll} +
-noCompati85
```

(b) 機能

-Compati85,IoStatus

入出力状態 1x，2x に対する処理を PC COBOL85 と同様にします。

-Compati85,Linage

LINAGE 値が不正なときの処理を PC COBOL85 と同様にします。

-Compati85,Call

CALL 文の ON EXCEPTION，または ON OVERFLOW 指定の無条件文を実行するエラーの条件を PC COBOL85 と同様にします。

-Compati85,Power

べき乗演算の精度，エラー時の処理を PC COBOL85 と同様にします。

べき乗演算と-Compati85,Power オプションとの関係については，マニュアル「COBOL2002　ユーザーズガイド」の「例外チェックが無効な場合の動作」を参照してください。

-Compati85,Syntax

次の機能について，コンパイル時の解釈を PC COBOL85 と同様にします。

- COPY 文の REPLACING 指定での置換位置
- 見出し部注記項（-Compati85,IDParag オプション指定時と同等の解釈をする）
- 部の見出し，節名，段落名，END PROGRAM などの開始位置チェック
- 直前のピリオドが省略された A 領域に置かれた語の解釈
- COBOL 語の最大長（30 文字とする）
- 予約語（COBOL85 では予約語で，COBOL2002 では文脈依存語に変更されたものを予約語に戻す）

  なお，PC COBOL85 で予約語だったものが COBOL2002 で文脈依存語となっているのは，次の語です。

  ONLY　PREVIOUS　RECURSIVE
- プログラム名の長さ（30 バイトとする）
- 自由形式正書法の 1 行の長さ（80 文字とする）

-Compati85,IDParag

見出し部の構文チェックを緩和します。

このオプションを指定すると，見出し部に AUTHOR 段落，INSTALLATION 段落，DATE-WRITTEN 段落，DATE-COMPILED 段落，SECURITY 段落，および REMARKS 段落が記述できます。

-Compati85,RsvWord

予約語のうち COBOL2002 で新たに追加されたものを除外し，PC COBOL85 の予約語と同じ範囲でコンパイルします。

-Compati85,NoPropagate

CALL 文，INVOKE 文，および利用者定義関数で伝播を抑止します。

次のプログラムを呼び出す場合，例外が伝播することはないため，このオプションを指定するとオブジェクトファイルサイズを縮小できます。

- PROPAGATE 指令が指定されていないプログラム
- COBOL85 で作成したプログラム
- C プログラムなど，COBOL 言語以外の言語を使用したプログラム
- COBOL2002 が提供するサービスルーチン
- OpenTP1，XMAP3 などの関連プログラムが提供している関数など

ただし，条件によっては，自動的に呼び出し元プログラムへの伝播が抑止される場合があります。その場合は，このオプションを指定する必要はありません。詳細は，マニュアル「COBOL2002　ユーザーズガイド」の例外を受け取れないプログラムに例外を伝播させた場合の動作を参照してください。

-Compati85,All

-Compati85 オプションのすべてのサブオプションを仮定します。

-Compati85,All を指定した場合，-Compati85 オプションのすべてのサブオプションを指定したものとみなします。

-noCompati85

-Compati85 オプションの指定を打ち消します。

(c)　注意事項

- -Compati85 オプションは，単独で指定できません。必ずサブオプションを指定する必要があります。
- -Compati85,IDParag オプションを指定した場合，見出し部中に COPY 文，REPLACE 文，および INCLUDE 文を記述できません。注記項中に COPY 文，REPLACE 文，または INCLUDE 文を記述した場合，その COPY 文，REPLACE 文，または INCLUDE 文を注記とみなします。
- -Compati85,IDParag オプションを指定した場合，注記項中に継続行を表す継続標識（-）が現れたときは，コンパイルエラーとなります。
- -Compati85,IDParag オプションを指定した場合，見出し部には，COBOL85 で指定できる段落見出し（PROGRAM-ID，AUTHOR，INSTALLATION，DATE-WRITTEN，DATE-COMPILED，SECURITY，および REMARKS）以外を指定できません。
- -Compati85,Syntax オプションを指定した場合に適用される予約語は，COBOL2002 の予約語および ONLY，PREVIOUS，RECURSIVE の三つの語となります。

  -Compati85,RsvWord オプションを指定した場合に適用される予約語は，PC COBOL85 の予約語だけとなります。
- -Compati85,RsvWord オプションを指定した場合，COBOL2002 で新しく追加された構文が利用できなくなります。このため，既存プログラムに COBOL2002 の新機能を加えて拡張することが困難になります。既存プログラムに COBOL2002 の新しい予約語が使われている場合，新しい予約語とは異なる語に置き換えることを推奨します。COBOL2002 で追加された新しい予約語については，マニュアル「COBOL2002　言語　標準仕様編」を参照してください。

- -Compati85 オプションのうち次のオプションを指定した場合，COBOL ソース中に TURN 指令を指定できません。指定した場合，コンパイルエラーとなります。
  - -Compati85,IoStatus
  - -Compati85,Linage
  - -Compati85,Call
  - -Compati85,Power
  - -Compati85,NoPropagate
- -Compati85,NoPropagate オプションは，CALL 文のほかに INVOKE 文，および利用者定義関数の呼び出しの場合にも，伝播が抑止されます。なお，CALL 文，INVOKE 文，および利用者定義関数を使用していない場合，このオプションは意味を持ちません。
- -Compati85,Syntax オプションは，-LiteralExtend オプションと同時に指定すると，-Compati85,Syntax オプションが有効となり，-LiteralExtend オプションは無効となります。

(4) -H8Switch オプション

(a) 形式

```
-H8Switch
-noH8Switch
```

(b) 機能

-H8Switch

HITAC8000 シリーズの仕様でコンパイルするためのオプションです。

次の文で，送り出し側作用対象が数字項目（または固定小数点数字定数）で，受け取り側作用対象が英数字項目（または英数字編集項目）の場合，数字項目を英数字項目とみなして転記します。

- MOVE
- WRITE FROM
- REWRITE FROM
- RELEASE FROM
- ACCEPT FROM DAY
- ACCEPT FROM DATE
- ACCEPT FROM TIME
- ACCEPT FROM DAY-OF-WEEK

また，比較条件で，両辺の組み合わせが数字項目（または固定小数点数字定数）と英数字項目（または英数字編集項目，英字項目，英数字定数）の場合，数字項目を英数字項目とみなして比較します。

このオプションを指定しなかった場合，数字項目は数字項目として転記，比較されます。

-noH8Switch

-H8Switch オプションの指定を打ち消します。

(5) -Cblctr オプション

(a) 形式

```
-Cblctr
-noCblctr
```

(b) 機能

-Cblctr

報告書作成機能を使用するときに指定します。このオプションを指定すると，CBL-CTR 特殊レジスタが生成され，COBOL プログラムと報告書作成機能間で連絡ができるようになります。

報告書作成機能の拡張の詳細は，マニュアル「COBOL2002　言語　拡張仕様編」を参照してください。

-noCblctr

-Cblctr オプションの指定を打ち消します。

(6) -DigitsTrunc オプション

(a) 形式

```
-DigitsTrunc
-noDigitsTrunc
```

(b) 機能

-DigitsTrunc

原始プログラム中の転記文で受け取り側作用対象が 2 進項目の場合，送り出し側作用対象のけたが受け取り側作用対象よりも大きいときに，上位のけたを切り捨てます。

（例）次のプログラムの実行結果を示します。

```
       77 A PIC S9(4) USAGE COMP VALUE +123.
       77 B PIC S9(2) USAGE COMP.
           :
           MOVE A TO B.
```

- -DigitsTrunc オプションを指定した場合 …… +23
- -DigitsTrunc オプションを指定しない場合 … +123

このオプションを指定しなかった場合，原始プログラム中の転記文で受け取り側作用対象が 2 進項目のとき，送り出し側作用対象の有効けた数が受け取り側作用対象よりも大きいと，入り切らない部分が切り捨てられます。

-noDigitsTrunc

-DigitsTrunc オプションの指定を打ち消します。

(c) 注意事項

- -DigitsTrunc オプションと-Comp5 オプションを同時に指定した場合，あとに指定したオプションだけが有効となります。
- 例外名 EC-SIZE-TRUNCATION に対する TURN 指令が有効の場合で，送り出し側作用対象の有効けた数が受け取り側作用対象のけた数より大きいときは，このオプションの指定があっても，上位けたを切り捨てないで，EC-SIZE-TRUNCATION 例外を検出します。

(7) -IgnoreLCC オプション

(a) 形式

```
-IgnoreLCC
-noIgnoreLCC
```

(b) 機能

-IgnoreLCC

WRITE～ADVANCING／POSITIONING 文で書き出すレコードの先頭 1 バイトを出力しないことを指定します。

あらかじめ，レコード記述項で行送り制御文字用の 1 バイト分の領域を確保しておいたプログラムをコンパイルするときに指定します。ただし，次の場合には-IgnoreLCC オプションの指定があっても，レコードの先頭から出力されます。

- 順ファイル以外のファイルの場合
- 順ファイルの WRITE 文に行送りの指定（ADVANCING，POSITIONING）がない場合
- 順ファイルのファイル記述項に LINAGE 句の指定がある場合

-noIgnoreLCC

-IgnoreLCC オプションの指定を打ち消します。

(8) -CmAster オプション

(a) 形式

```
-CmAster
-noCmAster
```

(b) 機能

-CmAster

固定形式正書法のとき，行の先頭文字（1 カラム目の文字）が標準コードの星印（*）である行を注記行とします。

ただし，このオプションは固定形式正書法で記述された COBOL ソースファイルをコンパイルするときだけ有効となります。

-noCmAster

-CmAster オプションの指定を打ち消します。

(9) -CmDol オプション

(a) 形式

```
-CmDol
-noCmDol
```

(b) 機能

-CmDol

固定形式正書法のとき，行の先頭から 7 カラム目の文字が「$」記号である行を注記行とします。

ただし，このオプションは固定形式正書法で記述された COBOL ソースファイルをコンパイルするときだけ有効となります。

-noCmDol

-CmDol オプションの指定を打ち消します。

(10) -Comp5 オプション

(a) 形式

```
-Comp5
-noComp5
```

(b) 機能

-Comp5

COBOL プログラム中の USAGE 句の COMP-5 を使用できるようにするオプションです。

このオプションを指定すると，COMP-5 は，COMP を指定した場合と同様に処理されます。

COMP-5 を指定できる場所は，COMP を指定できる場所と同じです。また，COMP-5 は，COMP と同じ仕様となるため，PICTURE 句のけた数，データのバイト数，および実際に格納できる数値は次のとおりとなります。

表 2-17　COMP-5 で実際に格納できる数値

PICTURE 句のけた数	データのバイト数	実際に格納できる数値	
		符号あり	符号なし
1～4 けた	2 バイト	-2^{15}～$2^{15}-1$	0～$2^{15}-1$
5～9 けた	4 バイト	-2^{31}～$2^{31}-1$	0～$2^{31}-1$
10～18 けた	8 バイト	-2^{63}～$2^{63}-1$	0～$2^{63}-1$

ただし，COMP-5 は，-BigEndian,Bin オプションの対象にはなりません。

このオプションを指定しなかった場合，USAGE 句に COMP-5 が指定されているとエラーになります。

USAGE 句に COMP-5 を指定した項目に VALUE 句がある場合の指定できる値については，マニュアル「COBOL2002　言語　拡張仕様編」の「データ記述項」の「USAGE COMP-5 を指定した項目に設定できる初期値（VALUE 句の値）の拡張」を参照してください。

-noComp5

-Comp5 オプションの指定を打ち消します。

(c)　注意事項

- -Comp5 オプションと-DigitsTrunc オプションを同時に指定した場合，あとに指定したオプションだけが有効となります。

(11)　-V3Spec オプション

(a)　形式

```
-V3Spec [,CopyEased]
-noV3Spec
```

(b)　機能

-V3Spec

VOS3 COBOL85 でサポートされていない，COBOL2002 固有の言語仕様に対して構文チェックをするためのオプションです。

VOS3 COBOL85 09-00 を対象として，ほぼ同等の言語仕様で構文チェックをし，VOS3 COBOL85 ではサポートされていない COBOL2002 固有の言語仕様に対して警告メッセージを出力します。

出力するオブジェクトファイルは，このオプションの影響を受けません。

このオプションは，VOS3 COBOL85 互換機能です。

-V3Spec,CopyEased

-V3Spec オプションで指定された構文チェックの範囲を緩和し，COPY 文，REPLACE 文の一部のエラーチェックをしません。

このオプションは，VOS3 COBOL85 互換機能です。

-noV3Spec

-V3Spec オプションの指定を打ち消します。

-V3Spec オプションを指定した場合，-V3Rec,Variable オプションが仮定されますが，-noV3Spec オプションを指定しても，仮定された-V3Rec,Variable オプションは打ち消されません。

(c) 注意事項

- 日本語項目の部分参照については，VOS3 COBOL85 の XCOBOL=(-N)オプションを指定した状態を対象とします。
- -V3Spec オプションと-CompatiV3 オプションを同時に指定した場合，一部のチェック項目が重複するため，-CompatiV3 オプションのエラーメッセージが出力され，このオプションのエラーメッセージは出力されません。
- -V3Spec オプションを指定すると，-V3Rec,Variable オプションが仮定されます。このため，固定長形式でコンパイルする場合は-V3Rec,Fixed オプションを同時に指定する必要があります。
- -V3Spec オプションと次のオプションを同時に指定すると，-V3Spec オプションが有効になり，次のオプションは無効になります。

  -Comp5　-CmAster　-BinExtend　-CBLVALUE　-Bin1Byte

  -CompatiM7　-NumAccept　-CmDol　-MaxDigits38　-IntResult,DecFloat40

  -LiteralExtend　-V3RecEased
- 次の機能は，このオプションではチェックされません。
  - データコミュニケーション機能
  - 構造型データベース（XDM/SD）操作シミュレーション機能
  - リレーショナルデータベース（XDM/RD）操作シミュレーション機能
  - ODBC インタフェース機能
- -V3Spec オプションと-UniObjGen オプションを同時に指定した場合，-V3Rec,Variable オプションは仮定されません。-V3Rec オプションを有効とするときは，環境変数 CBLV3UNICODE を指定してください。

(12) -V3ConvtName オプション

(a) 形式

```
-V3ConvtName
-noV3ConvtName
```

(b) 機能

-V3ConvtName

原文名定数の「¥」を「_」に，「@」を「!」に変換した名称で登録集原文を検索する場合，このオプションを指定します。このオプションは，COPY 文および INCLUDE 文で有効になります。また，原

文名定数の末尾の拡張子を省略した場合は，「.cbl」が仮定されます。なお，原文名指定の場合は，このオプションの対象となりません。

（例 1）原文名定数に¥，@が含まれる場合

原文名定数中に¥，@が含まれる場合は，-V3ConvtName オプションを指定してコンパイルします。

- COPY '¥ABC'.
  _ABC.cbl を登録集原文として検索します。
- COPY '@ABC'.
  !ABC.cbl を登録集原文として検索します。

（例 2）原文名に¥，@が含まれる場合

¥，@，#は，COBOL2002 では原文名に使用できない文字のため，コンパイルエラーとなります。この場合は，¥，@を含む原文名を定数指定に変更してから-V3ConvtName オプションを指定してコンパイルします。※

- COPY ¥ABC.　→　COPY '¥ABC'.
  _ABC.cbl を登録集原文として検索します。
- COPY @ABC.　→　COPY '@ABC'.
  !ABC.cbl を登録集原文として検索します。

-V3ConvtName オプションを使用しない場合は，¥，@を¥，@，#以外の文字に変更してください。

注※

#を含む原文名は，原文名定数に変更するだけで使用できます。

COPY 文での登録集原文の利用については，マニュアル「COBOL2002　ユーザーズガイド」の「原始文操作機能」を参照してください。

このオプションは，VOS3 COBOL85 互換機能です。

-noV3ConvtName

-V3ConvtName オプションの指定を打ち消します。

(13)　-Switch オプション

(a)　形式

```
-Switch, {EBCDIC | EBCDIK} 〔,Unprintable〕 〔,noApplyJpnItem〕
-noSwitch
```

(b)　機能

照合順序および字類条件を EBCDIC コードまたは EBCDIK コードに切り替えるためのオプションです。

-Switch,EBCDIC

英数字の照合順序および字類条件を EBCDIC コードに切り替えます。ただし，1 バイトで印字できないコードは，照合順序として X'00'または X'FF'に割り付けられます。

-Switch,EBCDIK

英数字の照合順序および字類条件を EBCDIK コードに切り替えます。ただし，1 バイトで印字できないコードは，照合順序として X'00'または X'FF'に割り付けられます。

-Switch,xxxxx,Unprintable

英数字の照合順序を，xxxxx に指定したコードに切り替えます。

xxxxx は，EBCDIC，EBCDIK のどちらかを指定します。

また，1 バイトで印字できないコードも，照合順序として固有の文字位置に割り付けます。詳細は，「(c)　注意事項」を参照してください。

-Switch,xxxxx,noApplyJpnItem

日本語項目を-Switch オプションの適用対象外とします。

xxxxx は，EBCDIC，EBCDIK のどちらかを指定します。

-noSwitch

-Switch オプションの指定を打ち消します。

(c)　注意事項

字類条件の判断基準と文字の大小関係

プログラムに記述した字類条件に対するコードの判断基準と文字の大小関係を，次に示します。

表 2-18　字類条件の判断基準と文字の大小関係

条件		ASCII	EBCDIC	EBCDIK
字類の判断基準	ALPHABETIC	a～z，A～Z	A～Z，a～z	A～Z
	ALPHABETIC-UPPER	A～Z	A～Z	A～Z
	ALPHABETIC-LOWER	a～z	a～z	－
文字の大小関係		a～z ＞ A～Z	a～z ＜ A～Z	a～z ＜ A～Z
組み込み関数の結果	UPPER-CASE	a～z→A～Z	a～z→A～Z	a～z→A～Z
	LOWER-CASE	A～Z→a～z	A～Z→a～z	A～Z→a～z

（凡例）

-：字類条件が真になることはない。

＞：大きい（a ＞ b　a は b より大きい）

＜：小さい（a ＜ b　a は b より小さい）

→：置換（a→b　a を b に置換する）

なお，次に示すように PROGRAM COLLATING SEQUENCE 句との背反があった場合，コンパイル時にエラーとなります。

PROGRAM COLLATING SEQUENCE 句の符号系名	-Switch オプション	
	EBCDIC サブオプションを指定した場合	EBCDIK サブオプションを指定した場合
指定なし	EBCDIC コード	EBCDIK コード
EBCDIC	EBCDIC コード	S レベルエラー
EBCDIK	S レベルエラー	EBCDIK コード
EBCDIC，EBCDIK 以外	S レベルエラー	S レベルエラー

-Switch オプションの対象

-Switch オプションの対象を次に示します。

機能	対象の字類
比較条件	英字，英数字，日本語※1
整列併合機能※2	英字，英数字

注※1

noApplyJpnItem サブオプションを指定している場合，対象の字類が日本語のときは-Switch オプションの対象となりません。

注※2

-Switch オプション指定による照合順序は，索引ファイルの参照キーには適用されません。参照キーの比較は，固有文字集合の大小順序に従います。

照合順序

比較のときに使用する照合順序として，1 バイトの印字可能文字は対応する文字位置に割り付けられ，その他の印字できないコードはすべて X'FF'に割り付けられます。

ただし，Unprintable サブオプションが有効な場合は，印字できないコードに，印字可能文字が割り付けられていない固有の文字位置が割り付けられます。Unprintable サブオプションの有無での照合順序の違いを次に示します。

文字コード種別	Unprintable サブオプションの指定	
	なし	あり
印字可能文字	対応する文字の文字位置に割り付けられます。	
印字できないコード	すべて X'FF'に割り付けられます。	印字可能文字が割り付けられていない固有の文字位置に割り付けられます。 ただし，印字できないコードの大小関係は保証しません。

Unprintable サブオプション指定時に，JIS8 単位コードが，EBCDIC コードまたは EBCDIK コードのどの位置に対応づけられるかをそれぞれ次に示します。なお，表中の網掛けは，印字できないコードを示します。

図 2-1　EBCDIC コードと JIS8 単位コードの対応表

下4ビット	上4ビット															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	X'00'	X'10'	X'80'	X'90'	X'20'	X'26'	X'2D'	X'BA'	X'C3'	X'CA'	X'D1'	X'D8'	X'7B'	X'7D'	X'24'	X'30'
	NUL	DLE			空白	&	-						{	}	$	0
1	X'01'	X'11'	X'81'	X'91'	X'A0'	X'A9'	X'2F'	X'BB'	X'61'	X'6A'	X'7E'	X'D9'	X'41'	X'4A'	X'9F'	X'31'
	SOH	DC1					/		a	j	¯		A	J		1
2	X'02'	X'12'	X'82'	X'16'	X'A1'	X'AA'	X'B2'	X'BC'	X'62'	X'6B'	X'73'	X'DA'	X'42'	X'4B'	X'53'	X'32'
	STX	DC2		SYN					b	k	s		B	K	S	2
3	X'03'	X'13'	X'83'	X'93'	X'A2'	X'AB'	X'B3'	X'BD'	X'63'	X'6C'	X'74'	X'DB'	X'43'	X'4C'	X'54'	X'33'
	ETX	DC3							c	l	t		C	L	T	3
4	X'9C'	X'9D'	X'84'	X'94'	X'A3'	X'AC'	X'B4'	X'BE'	X'64'	X'6D'	X'75'	X'DC'	X'44'	X'4D'	X'55'	X'34'
									d	m	u		D	M	U	4
5	X'09'	X'0a'	X'85'	X'95'	X'A4'	X'AD'	X'B5'	X'BF'	X'65'	X'6E'	X'76'	X'DD'	X'45'	X'4E'	X'56'	X'35'
	HT	NL	LF						e	n	v		E	N	V	5
6	X'86'	X'08'	X'17'	X'96'	X'A5'	X'AE'	X'B6'	X'C0'	X'66'	X'6F'	X'77'	X'DE'	X'46'	X'4F'	X'57'	X'36'
		BS	ETB						f	o	w		F	O	W	6
7	X'7F'	X'87'	X'1B'	X'04'	X'A6'	X'AF'	X'B7'	X'C1'	X'67'	X'70'	X'78'	X'DF'	X'47'	X'50'	X'58'	X'37'
	DEL		ESC	EOT					g	p	x		G	P	X	7
8	X'97'	X'18'	X'88'	X'98'	X'A7'	X'B0'	X'B8'	X'C2'	X'68'	X'71'	X'79'	X'E0'	X'48'	X'51'	X'59'	X'38'
		CAN							h	q	y		H	Q	Y	8
9	X'8D'	X'19'	X'89'	X'99'	X'A8'	X'B1'	X'B9'	X'60'	X'69'	X'72'	X'7A'	X'E1'	X'49'	X'52'	X'5A'	X'39'
		EM						`	i	r	z		I	R	Z	9
A	X'8E'	X'92'	X'8A'	X'9A'	X'5B'	X'5D'	X'7C'	X'3A'	X'C4'	X'CB'	X'D2'	X'E2'	X'E8'	X'EE'	X'F4'	X'FA'
					[	]	|	:								
B	X'0B'	X'8F'	X'8B'	X'9B'	X'2E'	X'5C'	X'2C'	X'23'	X'C5'	X'CC'	X'D3'	X'E3'	X'E9'	X'EF'	X'F5'	X'FB'
	VT				.	¥	,	#								
C	X'0C'	X'1C'	X'8C'	X'14'	X'3C'	X'2A'	X'25'	X'40'	X'C6'	X'CD'	X'D4'	X'E4'	X'EA'	X'F0'	X'F6'	X'FC'
	FF	FS		DC4	<	*	%	@								
D	X'0D'	X'1D'	X'05'	X'15'	X'28'	X'29'	X'5F'	X'27'	X'C7'	X'CE'	X'D5'	X'E5'	X'EB'	X'F1'	X'F7'	X'FD'
	CR	GS	ENQ	NAK	(	)	_	'								
E	X'0E'	X'1E'	X'06'	X'9E'	X'2B'	X'3B'	X'3E'	X'3D'	X'C8'	X'CF'	X'D6'	X'E6'	X'EC'	X'F2'	X'F8'	X'FE'
	SO	RS	ACK		+	;	>	=								
F	X'0F'	X'1F'	X'07'	X'1A'	X'21'	X'5E'	X'3F'	X'22'	X'C9'	X'D0'	X'D7'	X'E7'	X'ED'	X'F3'	X'F9'	X'FF'
	SI	US	BEL	SUB	!	^	?	″								

図 2-2　EBCDIK コードと JIS8 単位コードの対応表

下4ビット	上4ビット															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	X'00'	X'10'	X'80'	X'90'	X'20'	X'26'	X'2D'	X'6A'	X'73'	X'BF'	X'77'	X'79'	X'7B'	X'7D'	X'24'	X'30'
	NUL	DLE			空白	&	-	j	s	ソ	w	y	{	}	$	0
1	X'01'	X'11'	X'81'	X'91'	X'A1'	X'AA'	X'2F'	X'6B'	X'B1'	X'C0'	X'7E'	X'7A'	X'41'	X'4A'	X'9F'	X'31'
	SOH	DC1			。	ｪ	/	k	ｱ	ﾀ	￣	z	A	J		1
2	X'02'	X'12'	X'82'	X'16'	X'A2'	X'AB'	X'62'	X'6C'	X'B2'	X'C1'	X'CD'	X'E0'	X'42'	X'4B'	X'53'	X'32'
	STX	DC2		SYN	「	ｫ	b	l	ｲ	ﾁ	ﾍ		B	K	S	2
3	X'03'	X'13'	X'83'	X'93'	X'A3'	X'AC'	X'63'	X'6D'	X'B3'	X'C2'	X'CE'	X'E1'	X'43'	X'4C'	X'54'	X'33'
	ETX	DC3			」	ｬ	c	m	ｳ	ﾂ	ﾎ		C	L	T	3
4	X'9C'	X'9D'	X'84'	X'94'	X'A4'	X'AD'	X'64'	X'6E'	X'B4'	X'C3'	X'CF'	X'E2'	X'44'	X'4D'	X'55'	X'34'
					、	ｭ	d	n	ｴ	ﾃ	ﾏ		D	M	U	4
5	X'09'	X'0a'	X'85'	X'95'	X'A5'	X'AE'	X'65'	X'6F'	X'B5'	X'C4'	X'D0'	X'E3'	X'45'	X'4E'	X'56'	X'35'
	HT	NL	LF		・	ｮ	e	o	ｵ	ﾄ	ﾐ		E	N	V	5
6	X'86'	X'08'	X'17'	X'96'	X'A6'	X'AF'	X'66'	X'70'	X'B6'	X'C5'	X'D1'	X'E4'	X'46'	X'4F'	X'57'	X'36'
		BS	ETB		ヲ	ｯ	f	p	ｶ	ﾅ	ﾑ		F	O	W	6
7	X'7F'	X'87'	X'1B'	X'04'	X'A7'	X'A0'	X'67'	X'71'	X'B7'	X'C6'	X'D2'	X'E5'	X'47'	X'50'	X'58'	X'37'
	DEL		ESC	EOT	ｧ		g	q	ｷ	ﾆ	ﾒ		G	P	X	7
8	X'97'	X'18'	X'88'	X'98'	X'A8'	X'B0'	X'68'	X'72'	X'B8'	X'C7'	X'D3'	X'E6'	X'48'	X'51'	X'59'	X'38'
		CAN			ｨ	-	h	r	ｸ	ﾇ	ﾓ		H	Q	Y	8
9	X'8D'	X'19'	X'89'	X'99'	X'A9'	X'61'	X'69'	X'60'	X'B9'	X'C8'	X'D4'	X'E7'	X'49'	X'52'	X'5A'	X'39'
		EM			ｳ	a	i	`	ｹ	ﾈ	ﾔ		I	R	Z	9
A	X'8E'	X'92'	X'8A'	X'9A'	X'5B'	X'5D'	X'7C'	X'3A'	X'BA'	X'C9'	X'D5'	X'DA'	X'E8'	X'EE'	X'F4'	X'FA'
					[	]	|	:	ｺ	ﾉ	ｭ	ﾚ				
B	X'0B'	X'8F'	X'8B'	X'9B'	X'2E'	X'5C'	X'2C'	X'23'	X'74'	X'75'	X'78'	X'DB'	X'E9'	X'EF'	X'F5'	X'FB'
	VT				.	¥	,	#	t	u	x	ﾛ				
C	X'0C'	X'1C'	X'8C'	X'14'	X'3C'	X'2A'	X'25'	X'40'	X'BB'	X'76'	X'D6'	X'DC'	X'EA'	X'F0'	X'F6'	X'FC'
	FF	FS		DC4	<	*	%	@	ｻ	v	ﾖ	ﾜ				
D	X'0D'	X'1D'	X'05'	X'15'	X'28'	X'29'	X'5F'	X'27'	X'BC'	X'CA'	X'D7'	X'DD'	X'EB'	X'F1'	X'F7'	X'FD'
	CR	GS	ENQ	NAK	(	)	_	'	ｼ	ﾊ	ﾗ	ﾝ				
E	X'0E'	X'1E'	X'06'	X'9E'	X'2B'	X'3B'	X'3E'	X'3D'	X'BD'	X'CB'	X'D8'	X'DE'	X'EC'	X'F2'	X'F8'	X'FE'
	SO	RS	ACK		+	;	>	=	ｽ	ﾋ	ﾘ	ﾞ				
F	X'0F'	X'1F'	X'07'	X'1A'	X'21'	X'5E'	X'3F'	X'22'	X'BE'	X'CC'	X'D9'	X'DF'	X'ED'	X'F3'	X'F9'	X'FF'
	SI	US	BEL	SUB	!	^	?	″	ｾ	ﾌ	ﾙ	ﾟ				

Unprintable サブオプションが有効でない場合の照合順序の対応表については，マニュアル「COBOL2002　言語　標準仕様編」の「付録 B 計算機文字集合」を参照してください。

-Switch オプションを指定したプログラムでの比較結果

-Switch オプションを指定したプログラムでは，特定のデータ種別の比較結果で注意が必要です。

次のように，1 バイトで印字できないコード（日本語データや数字データなど）を含む作用対象（データ項目や定数）を比較条件に指定している場合は，値が異なるのに等しいと解釈されたり，-Switch オプション未指定時とは大小関係が逆転したりすることがあります。

データ種別

- 日本語データ（英数字項目や日本語項目※に日本語文字が格納されている場合など）
- 数字データ（集団項目の従属項目に，用途（USAGE 句）が DISPLAY 以外の数字項目（2 進項目，内部 10 進項目，内部浮動小数点項目）がある場合など）

注※

noApplyJpnItem サブオプションが無効の場合です。

比較結果

- 日本語データや数字データで，1 バイトの印字できないコードでは，等しいと解釈されることがあります。

  ただし，Unprintable サブオプションが有効な場合は，コードの一致および不一致を判定できます。なお，Unprintable サブオプションが有効な場合でも，照合順序が切り替わることで，大小関係が-Switch オプション未指定時とは逆転することがあります。
- 日本語データや数字データ中の印字可能文字は，照合順序が切り替わることで，-Switch オプション未指定時とは大小関係が逆転することがあります。

(14) -V3Rec オプション

(a) 形式

```
-V3Rec, {Fixed | Variable}
-noV3Rec
```

(b) 機能

メインフレーム（VOS3）の固定長または可変長レコード形式のプログラムを，VOS3 COBOL85 の日本語文字の扱いに合わせてコンパイルするためのオプションです。

-V3Rec,Fixed

VOS3 COBOL85 の固定長レコード形式の COBOL 原始プログラムを，VOS3 COBOL85 の日本語文字の扱いに合わせてコンパイルします。

-V3Rec,Variable

VOS3 COBOL85 の可変長レコード形式の COBOL 原始プログラムを，VOS3 COBOL85 の日本語文字の扱いに合わせてコンパイルします。

-V3Rec,Variable オプションは，-CompatiV3 オプションと同時に指定できます。また，-CompatiV3 オプションを指定すると，-V3Rec,Variable オプションが仮定されます。

-noV3Rec

-V3Rec オプションの指定を打ち消します。

(c) 注意事項

- -V3Rec オプションと-EquivRule オプションを重複指定した場合，あとに指定したオプションを優先します。
- このオプションを指定した場合，このシステムの標準コードと拡張コードの等価規則は適用されません。また，日本語文字定数に標準コードの空白を書いてもエラーとはなりません。ただし，日本語文字定数に空白以外の標準コード文字を書けますが，警告レベルのエラーとなります。なお，日本語文字定数の標準コード文字数は偶数としてください。日本語文字定数の標準コード文字数が奇数の場合，警告レベルのエラーとし，日本語文字定数に標準コードの空白を追加します。
- 日本語空白は半角空白 2 個と等価とみなし，機能キャラクタが付加されたものとみなす対象としません。日本語空白を使用しているメインフレームの固定長形式の原始プログラムを-V3Rec,Fixed オプ

ションを指定してコンパイルする場合は，カラムずれなどの影響がないか確認する必要があります。機能キャラクタについては，マニュアル「COBOL2002　言語　拡張仕様編」の文字集合の説明を参照してください。また，日本語空白を拡張コードとして扱いたいときは，-V3Rec オプションと合わせて-V3RecFCSpace オプションを指定してください。-V3RecFCSpace オプションの詳細については，「(15)　-V3RecFCSpace オプション」を参照してください。

- VOS3 COBOL85 の固定長レコード形式の原始プログラムと可変長レコード形式の原始プログラムは，このオプションを指定したときだけコンパイルできます。

  したがって，COPY 文で複写した原始プログラムを含め，固定長レコード形式の原始プログラムと可変長レコード形式の原始プログラムは同時にコンパイルできません。
- このオプションを指定した場合，72 カラムより前に改行文字がある COBOL 原始プログラムは，改行文字から 72 カラムまで空白に置き換えてコンパイルされます。
- このオプションは，自由形式正書法で書かれた COBOL 原始プログラムをコンパイルする場合には指定できません。指定した場合，エラーとなってコンパイルが中止されます。
- このオプションを指定した場合，英数字定数の分離符には，アポストロフィ（'）だけが使用できます。

(15)　-V3RecFCSpace オプション

(a)　形式

```
-V3RecFCSpace
-noV3RecFCSpace
```

(b)　機能

-V3RecFCSpace

-V3Rec オプションでの空白に関する機能キャラクタの扱いを，VOS3 COBOL85 と同等にします。機能キャラクタが付加されたものとみなす条件については，「(d)　機能キャラクタが付加されたものとみなす条件」を参照してください。

-noV3RecFCSpace

-V3RecFCSpace オプションの指定を打ち消します。

(c)　注意事項

- -V3RecFCSpace オプションの指定は，-V3Rec オプションの指定が有効な場合にだけ有効となります。

(d)　機能キャラクタが付加されたものとみなす条件

このコンパイラは，-V3Rec オプションの指定が有効な場合に，-V3RecFCSpace オプションの有無で，空白に関する機能キャラクタの扱いが異なります。次の条件文中の太字の部分は，-V3RecFCSpace オプションの有無での相違点です。

なお，これ以降「KEIS コード開始」と「EBCDIK コード開始」を表す 2 バイトの機能キャラクタをそれぞれ，[漢]と[E]で表します。また，改行を[改行]で表します。

- -V3RecFCSpace オプション指定がない場合
  - **日本語空白を除く**日本語文字列の先頭文字の直前に[漢]が付加されたものとみなします。
  - 直前に[漢]が付加されたものとみなした日本語文字列のあとに，最初に現れた半角空白以外の半角文字の直前に[E]が付加されたものとみなします。
  - [漢]が付加されたものとみなしたあとは，[E]が付加されたものとみなすまで，同じ行には[漢]が付加されたものとみなしません。
- -V3RecFCSpace オプション指定がある場合
  - **日本語空白を含む**日本語文字列の先頭文字の直前に[漢]が付加されたものとみなします。
  - 直前に[漢]が付加されたものとみなした日本語文字列のあとに，最初に現れた半角空白以外の半角文字の直前に[E]が付加されたものとみなします。**また，日本語文字列の直後が改行か日本語文字列と改行との間が半角空白だけのとき，日本語文字列の直後に[E]が付加されたものとみなします。**
  - [漢]が付加されたものとみなしたあとは，[E]が付加されたものとみなすまで，同じ行には[漢]が付加されたものとみなしません。

-V3Rec オプションの指定が有効な場合に，-V3RecFCSpace オプションの有無での機能キャラクタが付加されたものとみなした状態を次の例に示します。

（例 1）日本語空白に関する機能キャラクタの扱い

COBOL 原始プログラム

```
000100 01△DATA1△PIC△N(5)△VALUE△N'▲あいう▲'.
```

-V3Rec,Fixed オプション指定ありで， -V3RecFCSpace オプション指定がない場合

```
000100 01△DATA1△PIC△N(5)△VALUE△N'▲[漢]あいう▲[E]'.
```

-V3Rec,Fixed オプション指定ありで，-V3RecFCSpace オプション指定がある場合

```
000100 01△DATA1△PIC△N(5)△VALUE△N'[漢]▲あいう▲[E]'.
```

（凡例）

▲：日本語空白

△：半角空白

-V3RecFCSpace オプション指定がない場合，日本語空白を日本語文字列の先頭文字として扱いません。このため，日本語空白以外の先頭文字「あ」の直前に[漢]が付加されたものとみなします。また，直前に[漢]が付加されたものとみなした日本語文字列「あいう」のあとで最初に現れた半角空白以外の半角文字「'」の直前に[E]が付加されたものとみなします。

-V3RecFCSpace オプション指定がある場合，日本語空白を日本語文字列の先頭文字と扱い，直前に[漢]が付加されたものとみなします。また，直前に[漢]が付加されたものとみなした日本語文字列「▲あいう▲」のあとで最初に現れた半角空白以外の半角文字「'」の直前に[E]が付加されたものとみなします。

（例 2）空白と改行に関する機能キャラクタの扱い

COBOL 原始プログラム

```
000100 01△DATA1△PIC△N(5)△VALUE△N'あいう△△[改行]
000200-'△△'.
```

-V3Rec,Fixed オプション指定ありで， -V3RecFCSpace オプション指定がない場合

```
000100 01△DATA1△PIC△N(5)△VALUE△N'[漢]あいう△△[改行]
000200-'△△'.
```

-V3Rec,Fixed オプション指定ありで，-V3RecFCSpace オプション指定がある場合

```
000100 01△DATA1△PIC△N(5)△VALUE△N'[漢]あいう[E]△△[改行]
000200-'△△'.
```

（凡例）

▲：日本語空白

△：半角空白

-V3RecFCSpace オプション指定がない場合，日本語文字列「あいう」のあとには，半角空白以外の半角文字がないため[E]が付加されたものとみなしません。

-V3RecFCSpace オプション指定がある場合，日本語文字列「あいう」と[改行]の間が半角空白だけであるため，日本語文字列「あいう」の直後に[E]が付加されたものとみなします。

(16) -DoubleQuote オプション

(a) 形式

```
-DoubleQuote
-noDoubleQuote
```

(b) 機能

-DoubleQuote

VOS3 COBOL85 互換に関連するオプションである-V3Rec，-CompatiV3 のどちらかを指定したときに，英数字定数の分離符として引用符（ " ）を用いることを指定するオプションです。このオプションを指定しないときは，アポストロフィ（ ' ）を分離符として用います。

-V3Rec オプション，-CompatiV3 オプションのどちらも指定していない場合は，-DoubleQuote オプションを指定しても無効となります。この場合は，引用符（ " ），アポストロフィ（ ' ）のどちらも分離符として使用できます。

ただし，SQL 文中の定数の分離符は，-DoubleQuote オプションの指定に関係なく，常にアポストロフィ（ ' ）を使用します。

また，QUOTE 表意定数は，-V3Rec オプション，-CompatiV3 オプションの指定に関係なく，-DoubleQuote オプション指定時は引用符（ " ）を意味し，-DoubleQuote オプション指定なしのときはアポストロフィ（ ' ）を意味します。

なお，このオプションは，翻訳指令に対しても有効となります。

-noDoubleQuote

-DoubleQuote オプションの指定を打ち消します。

(17) -BigEndian オプション

(a) 形式

```
-BigEndian {,Bin|,Float} +
-noBigEndian
```

(b) 機能

用途（USAGE 句）が 2 進または浮動小数点のデータ項目の形式を，ビッグエンディアン形式にするためのオプションです。

-BigEndian,Bin

用途が 2 進（BINARY，COMP，COMP-4）のデータ項目の形式を，ビッグエンディアン形式にします。

このオプションを指定した場合，2 進項目を演算などで参照すると，実行性能が劣化します。このため，必要のないかぎりこのオプションは指定しないでください。

-BigEndian,Float

用途が浮動小数点（COMP-1，COMP-2）のデータ項目の形式を，ビッグエンディアン形式にします。

このオプションを指定した場合，浮動小数点を演算などで参照すると，実行性能が劣化します。このため，必要のないかぎりこのオプションは指定しないでください。

-noBigEndian

-BigEndian オプションの指定を打ち消します。

(c) 注意事項

ビッグエンディアン形式とリトルエンディアン形式との違い

コンピュータで扱うバイナリ形式のデータ（2 進データ，内部浮動小数点数字データ）には，ビッグエンディアン形式とリトルエンディアン形式の 2 種類があります。

これは，マイクロプロセッサのアーキテクチャの違いによるもので，システムによって採用されるエンディアン形式が異なります。次の表にシステムごとのエンディアン形式を示します。

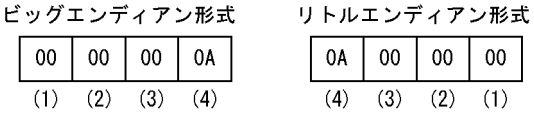
システム	エンディアン
Windows	リトルエンディアン
Linux	
HP-UX	ビッグエンディアン

システム	エンディアン
AIX	ビッグエンディアン
Solaris	

ビッグエンディアン形式とリトルエンディアン形式では，バイナリデータがメモリ上に配置されるときの順序が次の例のように逆になります。

（例）

10（=H'0000000A'）が４バイトのメモリに配置された場合の違い



ビッグエンディアン形式では左側から右側へ 1 バイトずつ配置されるのに対して，リトルエンディアン形式では右側から左側へ 1 バイトずつ配置されます。このため，ビッグエンディアン形式のシステムと，リトルエンディアン形式のシステムで同じデータを扱う場合でも，参照のしかたによっては違った結果になることがあります。-BigEndian オプションは，このような場合にデータを正しく処理できるようにするために使用します。

-BigEndian オプションの使い方

リトルエンディアン形式のシステム上で，2 進データをそのまま参照したり，作成したファイルを参照したりする場合は，このオプションを指定する必要はありません。しかし，REDEFINES 句などで英数字データを 2 進データとして参照したり，ビッグエンディアン形式のシステムで作られたファイル（可変長ファイルは対象外）を参照したりする場合は，ビッグエンディアン形式のシステムと，リトルエンディアン形式のシステムで 2 進データ，内部浮動小数点数字データの表現形式が異なるため，このオプションを指定する必要があります。このオプションは次のような場合に指定してください。

- ビッグエンディアン形式のシステム上で開発し，2 進データや内部浮動小数点数字データのメモリ上での配置を意識しているようなプログラムをリトルエンディアン形式のシステムに移植する場合
- ビッグエンディアン形式のシステム上で作成した 2 進データや内部浮動小数点数字データを含むファイルを，そのままリトルエンディアン形式のシステムへ移行し，プログラムで参照する場合
- ビッグエンディアン形式のシステム上で開発し，REDEFINES 句でほかの形式のデータ項目を 2 進データとして参照するようなプログラムを，リトルエンディアン形式のシステムに移植する場合

使用上の注意

-BigEndian オプションの使用上の注意を次に示します。

- C 言語と連携している場合，-BigEndian,Bin オプション指定時には，C プログラムとインタフェースを取る 2 進データは COMP-5 で定義してください。
- コマンド行に指定した引数を受け取る場合，-BigEndian,Bin 指定時には，連絡節の 2 進データは，COMP-5 で定義してください。なお，引数の受け取り方法については，マニュアル「COBOL2002

ユーザーズガイド」の「COBOL の実行単位」の「引数の受け取り方法（C 言語インタフェースに従った形式の場合)」または「引数の受け取り方法（VOS3 インタフェースに従った形式の場合)」の説明を参照してください。

- 2 進項目を引数とするサービスルーチン（OLE2 オートメーション機能で使用するサービスルーチンも含む）は，受け取る項目を COMP-5 で定義してください。
- CALL 文で LENGTH OF 一意名を指定した場合，呼び出し先プログラムは，受け取る項目を COMP-5 で定義してください。
- 開発マネージャでのビルド，リビルドや cblbuild2k コマンドを使用する場合，一つのプロジェクト中にこのオプションを指定したプログラムと指定しないプログラムを混在させないでください。詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「cblbuild2k コマンド」を参照してください。
- XMAP3 を使用する場合は，XMAP3 のオプションと合わせる必要があります。
- 索引ファイル，および整列併合機能で使用するキーデータ項目の 2 進データ項目は，リトルエンディアン形式でキーの大小比較を行います。このため，このオプションを指定したビッグエンディアン形式では，正常に動作できません。キーデータ項目の 2 進データ項目は COMP-5 で定義するか，またはリトルエンディアン形式データとしてください。

(18)　-VOSCBL オプション

メインフレーム互換機能を有効にします。

(a)　形式

```
-VOSCBL{,OccursKey|,ReportControl|,DataComm|,RedefinesData|,AssignDataToDevice|,Evaluat
eWhenOther}+
-noVOSCBL
```

(b)　機能

メインフレーム互換機能を有効にします。

-VOSCBL,OccursKey

OCCURS 句の KEY IS 指定のデータ名の名前の有効範囲を，その OCCURS 句のある記述項または，それに従属する記述項とします。

OCCURS 句の KEY IS 指定のデータ名は修飾してもその修飾は無視されます。

（例 1）

```
       WORKING-STORAGE SECTION.
         01 DAT0.
           02 DAT1 PIC X.   ・・・・・・・・・・・・・・・(1)
         01 DAT2.
           02 DAT3 OCCURS 10 ASCENDING KEY DAT1.   ・・・(2)
             03 DAT1  PIC X.       ・・・・・・・・・・・(3)
             03 DAT5  PIC X.
```

- -VOSCBL,OccursKey オプションの指定がある場合は，（2）の DAT1 は修飾しなくても，（2）の従属項目として定義された（3）の DAT1 が参照されます。
- -VOSCBL,OccursKey オプションの指定がない場合は，DAT1 が（1）と（3）にあるため，名前が一意になりません。一意に参照するためには，（2）の DAT1 を「OF DAT3」で修飾する必要があります。

（例 2）

```
      WORKING-STORAGE SECTION.
        01 DAT0.
          02 DAT1 PIC X.      ・・・・・・・・・・・・・・・・(1)
        01 DAT2.
          02 DAT3 OCCURS 10 ASCENDING KEY DAT1 OF DAT0.   ・・(2)
            03 DAT1  PIC X.       ・・・・・・・・・・・・・・(3)
            03 DAT5  PIC X.
```

- -VOSCBL,OccursKey オプションの指定がある場合は，（2）の DAT1 を修飾（OF DAT0）してもその修飾は無視されるため，（2）の DAT1 は，（2）の従属項目として定義された（3）の DAT1 が参照されます。
- -VOSCBL,OccursKey オプションの指定がない場合は，（2）の DAT1 の修飾（OF DAT0）を無視されないため，（1）の DAT1 が参照されます。しかし，KEY IS 指定のデータ名は，OCCURS 句のある記述項または，それに従属する記述項でなければなりません。

  したがって，（1）の DAT1 は（2）の従属項目でないため，（2）の修飾を「OF DAT3」に変更する必要があります。

-VOSCBL,ReportControl

制御脚書きの印刷中に改ページが起こったときに，ページの頭書きで SOURCE 句によって制御用データ項目を参照している場合，その制御用データ項目の値を元の値で参照されます。

-VOSCBL,DataComm

データコミュニケーション機能の通信記述項で，次のどちらかに該当するときの初期値を，メインフレーム(VOS3/VOS1)の XDM/DCCM データコミュニケーション機能を使用する場合の値とします。

- 通信記述項の句の指定を省略している。
- 通信文の実行前に，通信記述項の句に指定されたデータ名に初期値を設定していない。

なお，このオプションの指定がない場合は，TP1/Server Base を使用する場合の初期値とします。

-VOSCBL,RedefinesData

REDEFINES 句の右辺のデータ名が未定義のとき，直前の同一レベル番号のデータ名を REDEFINES 句の右辺として仮定します。

（例）

```
4               WORKING-STORAGE SECTION.
5                 01 DAT1.
6                   02 DAT2.
7                     03 DAT3  PIC XX.
8                   02 DAT4  REDEFINES UNDEF.
```

```
 9                   03 DAT5  PIC X.
10                   03 DAT6  PIC X.
```

この例では，8 行目の REDEFINES 句の右辺のデータ名 UNDEF が定義されていません。-VOSCBL,RedefinesData オプションが有効な場合，直前の同一レベル番号のデータ名 DAT2 を REDEFINES 句の右辺のデータ名として仮定します。-VOSCBL,RedefinesData オプションが無効な場合，8 行目の REDEFINES 句は無視されます。

-VOSCBL,AssignDataToDevice

ASSIGN 句の利用者定義語がデータ名と一致していても外部装置名とみなします。

（例）

```
 4       ENVIRONMENT         DIVISION.
 5        INPUT-OUTPUT       SECTION.
 6         FILE-CONTROL.
 7           SELECT  TEST-F  ASSIGN    SYS001.
 8       DATA                DIVISION.
 9        FILE               SECTION.
10         FD    TEST-F      LABEL  RECORD  STANDARD.
11          01   TEST-R      PIC  X(80).
12       WORKING-STORAGE    SECTION.
13          01   SYS001      PIC  X(16).
```

この例では，7 行目の ASSIGN 句でデータ名「SYS001」を定義しています。

-VOSCBL,AssignDataToDevice オプションが有効な場合は，「SYS001」を外部装置名とみなして，お知らせメッセージを出力します。-VOSCBL,AssignDataToDevice オプションが無効な場合は，「SYS001」をデータ名とみなします。

-VOSCBL,EvaluateWhenOther

EVALUATE 文の WHEN 指定と WHEN OTHER 指定の間の無条件文が省略された場合に，-VOSCBL,EvaluateWhenOther オプションが有効なときは，警告エラーメッセージを出力して CONTINUE 文を仮定します。-VOSCBL,EvaluateWhenOther オプションが無効なときは，重大エラーメッセージを出力します。

-noVOSCBL

-VOSCBL オプションの指定を打ち消します。

(c) 注意事項

- このオプションは，-CompatiV3 オプションと同時に指定してください。-CompatiV3 オプションの指定がない場合は，エラーメッセージが出力されます。
- 次のオプションは-CompatiV3 オプションを無効とするため，このオプションおよび-CompatiV3 オプションと次のオプションを同時に指定しないでください。-CompatiV3 オプションの指定が無効となり，U レベルのエラーになります。

  -StdMIA　-Std85　-Std2002

- このオプションと-UniObjGen オプション，および-CompatiV3 オプションを同時に指定する場合，-CompatiV3 オプションが有効となるよう，コンパイラ環境変数 CBLV3UNICODE を指定してください。

(19) -PortabilityCheck オプション

メインフレームから移行する場合，ユーザが注意すべき個所にお知らせメッセージ（I レベルメッセージ）を出力し，ユーザの移行作業を支援します。このオプションを指定すると，COBOL2002 とメインフレームで動作の異なる可能性がある個所を対象に，お知らせメッセージを出力します。

なお，このオプションは，S レベル以上のエラーがないプログラムに適用してください。S レベル以上のエラーがあるプログラムで適用した場合の動作は，保証しません。

(a) 形式

```
-PortabilityCheck{,Literal|,Numeric} +
-noPortabilityCheck
```

(b) 機能

-PortabilityCheck,Literal

- -UniObjGen オプション指定時，英数字定数および日本語文字定数中の日本語または半角かたかなにお知らせメッセージを出力します。
- 次に示す 16 進定数にお知らせメッセージを出力します。

  16 進英数字定数，16 進数字定数，16 進日本語文字定数

-PortabilityCheck,Numeric

- RECORD KEY 句，ALTERNATE RECORD KEY 句の 2 進項目，内部浮動小数点項目にお知らせメッセージを出力します。
- 浮動小数点の演算が発生する個所にお知らせメッセージを出力します。

  浮動小数点の演算が発生する個所は，次を対象としています。

1. 算術式，算術文に指定された浮動小数点項目や浮動小数点定数

   算術式が記述できる場所に書かれた単項の浮動小数点項目や浮動小数点定数も含みます。

   ただし，受け取り項目など，演算に関係しない浮動小数点項目は除外します。
2. べき乗の場合，べき数に指定されたデータ項目および算術式が小数点を持つ場合，または，べき数が小数点を含む数字定数である場合
3. 次の組み込み関数

   ACOS，ASIN，ATAN，COS，LOG，LOG10，MOD，RANDOM，REM，SIN，SQRT，STANDARD-DEVIATION，TAN，VARIANCE
4. 次のどれかが引数に指定されている組み込み関数

   浮動小数点項目※，べき乗が含まれる算術式，または除算が含まれる算術式

注※

ADDR 関数，および LENGTH 関数を除きます。

-noPortabilityCheck

-PortabilityCheck オプションの指定を打ち消します。

(20) -IgnoreAPPLY,FILESHARE オプション

入出力管理記述項の APPLY FILE-SHARE 句を覚え書きとみなします。覚え書きとみなしたときは，コンパイル時にお知らせメッセージを出力します。

(a) 形式

```
-IgnoreAPPLY,FILESHARE
-noIgnoreAPPLY
```

(b) 機能

-IgnoreAPPLY,FILESHARE

- 入出力管理記述項の APPLY FILE-SHARE 句を覚え書きとみなします。覚え書きとみなしたときは，コンパイル時にお知らせメッセージを出力します。

-noIgnoreAPPLY

-IgnoreAPPLY,FILESHARE オプションの指定を打ち消します。

(21) -V3RecEased オプション

(a) 形式

```
-V3RecEased{,QuoteCheck|,WordCheck}+
-noV3RecEased
```

(b) 機能

-V3RecEased,QuoteCheck

-CompatiV3 または-V3Rec オプションを指定したとき，定数の分離符のチェックを緩和します。

-V3RecEased,QuoteCheck オプションが有効な場合，-DoubleQuote オプションに関係なく，定数の分離符は，アポストロフィ（'）と引用符（"）のどちらでも指定できます。

-V3RecEased,WordCheck

-CompatiV3 または-V3Rec オプションを指定したとき，語または PICTURE 文字列の制限値のチェックを緩和します。

-V3RecEased,WordCheck オプションが有効な場合，語または PICTURE 文字列の制限値を超えたときのコンパイルエラーを，重大エラーから警告エラーに緩和できます。

-noV3RecEased

-V3RecEased オプションの指定を打ち消します。

(c) 注意事項

- -V3RecEased オプションと-V3Spec オプションを同時に指定したとき，-V3RecEased オプションの指定は無効になります。
- -V3RecEased オプションの指定は，-V3Rec オプションの指定が有効な場合にだけ有効です。
- -V3RecEased,QuoteCheck オプションの指定に関係なく，SQL 文中の定数の分離符はアポストロフィ(')だけ使用できます。
- -V3RecEased,QuoteCheck オプションは，翻訳指令に対しても有効です。

2.3.13 リスト出力の設定

リスト出力を設定するコンパイラオプションについて，説明します。

(1) -SrcList オプション

(a) 形式

```
-SrcList, {OutputAll | CopyAll | CopySup | NoCopy} 〔,NoFalsePath〕 〔,DataLoc〕
-noSrcList
```

(b) 機能

コンパイルリストの出力形式を指定するオプションです。このオプションを指定しなかった場合，コンパイルリストは，出力されません。

出力されるリストの内容，およびコンパイルリストに関連する翻訳指令については，マニュアル「COBOL2002 ユーザーズガイド」の「コンパイルリスト」および「コンパイルリストに関連する翻訳指令」を参照してください。

-SrcList,OutputAll

すべての情報を出力します。

原始プログラム中に，COPY 文の SUPPRESS 指定や LISTING OFF 指令がある場合でも，すべてのソースをコンパイルリストに展開します。

-SrcList,CopyAll

SUPPRESS 指定を無視して，すべての COPY 文を強制的に展開します。

-SrcList,CopySup

プログラムに SUPPRESS 指定があるときだけ，COPY 文の展開を抑止します。SUPPRESS 指定がない COPY 文は，すべて展開します。

-SrcList,NoCopy

すべての COPY 文の展開を抑止します。

-SrcList,xxxxx,NoFalsePath

条件翻訳の無効行を出力しません。

xxxxx には，CopyAll，CopySup，NoCopy のどれかを指定します。

-SrcList,OutputAll,NoFalsePath と指定した場合は，OutputAll サブオプションが有効となり，NoFalsePath サブオプションは無効になります。

-SrcList,xxxxx,DataLoc

コンパイルリスト（原始プログラムリスト）にデータ項目の相対位置と長さ（バイト）を 16 進数で表示します。相対位置は，データ部のファイル節／作業場所節／局所場所節の各節の先頭からの位置を表示します。

xxxxx には，OutputAll，CopyAll，CopySup，NoCopy のどれかを指定します。

S レベル／U レベルのコンパイルエラーが発生した場合は，指定があっても相対位置は表示しません。なお，相対位置の表示については，マニュアル「COBOL2002　ユーザーズガイド」の「リストの見方」の「相対位置表示時の原始プログラムリスト」を参照してください。

-noSrcList

-SrcList オプションの指定を打ち消します。

(c)　注意事項

NoFalsePath サブオプションまたは，DataLoc サブオプションを指定した場合，-noSrcList オプションの指定がないと，NoFalsePath サブオプション，および DataLoc サブオプションの指定は有効となります。NoFalsePath サブオプション，および DataLoc サブオプションの指定を打ち消すときは，-noSrcList オプションを指定してください。

（例 1）NoFalsePath サブオプション，および DataLoc サブオプション指定が打ち消されないケース

```
ccbl2002 -SrcList,OutputAll,NoFalsePath,DataLoc -SrcList,CopyAll ···

 （コンパイル結果）

-SrcList,CopyAll,NoFalsePath,DataLoc

ccbl2002 -SrcList,OutputAll,NoFalsePath -SrcList,CopySup -SrcList,CopyAll,DataLoc
                    -SrcList,CopySup -SrcList,CopyAll, ···

 （コンパイル結果）

-SrcList,CopyAll,NoFalsePath,DataLoc
```

（例 2）NoFalsePath サブオプション，および DataLoc サブオプション指定が打ち消されるケース

```
ccbl2002 -SrcList,CopySup,NoFalsePath,DataLoc -noSrcList -SrcList,CopyAll ···

 （コンパイル結果）
```

```
-SrcList,CopyAll
```

(2) -ErrSup オプション

(a) 形式

```
-ErrSup {,I|,W} +
-noErrSup
```

(b) 機能

コンパイル時にＩレベルまたはＷレベルメッセージの出力を抑止します。

-ErrSup,I

コンパイル時にＩレベル（お知らせ）メッセージの出力を抑止します。

-ErrSup,W

コンパイル時にＷレベル（警告）メッセージの出力を抑止します。

-noErrSup

-ErrSup オプションの指定を打ち消します。

(c) 注意事項

- -ErrSup,W オプションを指定した場合，利用者が注意する必要があるエラー（コンパイラによる解釈の変更や，中間結果けた数制限の適用など）の出力が抑止されます。このため，必要な場合以外は，-ErrSup,W オプションを指定しないようにしてください。
- -ErrSup オプションを指定してエラーメッセージの出力を抑止した場合，標準エラーおよびコンパイルリストにも，該当レベルのエラーが出力されません。
- 抑止されたエラーメッセージは，コンパイルリストに出力されるエラー件数にはカウントされません。
- コマンドライン上の記述誤りに対するＷレベルメッセージは，抑止されません。

2.3.14 その他の設定

その他のコンパイラオプションについて，説明します。

(1) -Bin1Byte オプション

(a) 形式

```
-Bin1Byte
-noBin1Byte
```

(b) 機能

-Bin1Byte

1 バイトの 2 進項目を有効にします。

-noBin1Byte

-Bin1Byte オプションの指定を打ち消します。

(2) -JPN オプション

(a) 形式

```
-JPN, {Alnum | V3JPN | V3JPNSpace}
-noJPN
```

(b) 機能

-JPN,Alnum

日本語項目，日本語編集項目，および日本語文字定数をそれぞれ英数字項目，英数字編集項目，および英数字定数として扱います。これは，VOS3 COBOL85 の LANGOPT=(-D)オプションと XCOBOL=(N)オプションを同時に指定したときと同じ動作になります。

なお，-CompatiV3 オプションを指定すると，このオプションが仮定されます。

-JPN,V3JPN

日本語項目，日本語編集項目，および日本語文字定数をそれぞれ英数字項目，英数字編集項目，および英数字定数として扱うようにするためのオプションです。ただし，LENGTH 関数の引数※，STRING 文，UNSTRING 文，または INSPECT 文では日本語項目，日本語編集項目，および日本語文字定数はそのままの属性として扱います。

また，日本語項目または日本語編集項目の部分参照は，英数字項目または英数字編集項目として扱いますが，最左端の文字位置と長さは日本語文字数を表します。これは，VOS3 COBOL85 の LANGOPT=(-D)オプションと XCOBOL=(-N)オプションを同時に指定したときと同じ動作になります。

なお，-JPN,V3JPN オプションは，VOS3 COBOL85 互換機能です。

注※

VOS3 COBOL85 では，LENGTH 関数の引数に日本語項目，日本語編集項目，および日本語文字定数は指定できません。

-JPN,V3JPNSpace

-CompatiV3 オプションと同時に指定した場合，日本語項目，日本語編集項目および日本語文字定数をそのままの属性で扱います。これは，VOS3 COBOL85 の LANGOPT=(D)オプションと XCOBOL=(-N)オプションを同時に指定したときと同じ動作になります。

-JPN,V3JPNSpace オプションは，VOS3 COBOL85 互換機能です。

VOS3 COBOL85 で LANGOPT=(D)オプションを指定した場合と COBOL2002 での動作の相違については，マニュアル「COBOL2002　言語　拡張仕様編」を参照してください。

このオプションを指定してコンパイルすると，日本語項目および日本語編集項目に対して次に示すとおりに扱います。

- 表意定数 SPACE（SPACES）の扱い

  日本語項目および日本語編集項目に対する表意定数 SPACE（SPACES）は，日本語空白（X'8140'）となります。

- 日本語項目および日本語編集項目同士の転記の扱い

  日本語項目および日本語編集項目同士の転記で，受け取り側のけた数が送り出し側のけた数より長い場合，受け取り側の右側に日本語空白（X'8140'）を補います。

- 日本語項目および日本語編集項目同士の比較の扱い

  日本語項目および日本語編集項目同士の比較で，作用対象のけた数が等しくない場合，短い方の作用対象の右側に，長い方の作用対象のけた数に等しくなるまで日本語空白（X'8140'）があるものとみなして比較します。

-noJPN

-JPN オプションの指定を打ち消します。

(c)　注意事項

- -JPN,Alnum オプションと-JPN,V3JPN オプションを重複して同時に指定した場合，-JPN,V3JPN オプションが有効になり，-JPN,Alnum オプションが無効になります。
- コンパイラ環境変数 CBLV3UNICODE に YES を指定し，かつ-UniObjGen を指定した場合，-CompatiV3 を指定しても-JPN,Alnum は仮定されません。
- -JPN,V3JPNSpace オプションを指定するとき，-CompatiV3 オプションを同時に指定する必要があります。

  -CompatiV3 オプションの指定がない場合，コンパイルエラーとなります。
- -JPN,V3JPNSpace オプションと-JPN,Alnum オプション，-JPN,V3JPN オプションを同時に指定すると，-JPN,Alnum オプション，-JPN,V3JPN オプションは無効となります。

  また，-JPN,V3JPNSpace オプションと-CompatiV3 オプションを同時に指定すると，-CompatiV3 オプションが仮定する-JPN,Alnum オプションは無効となります。
- -JPN,V3JPNSpace オプションと-UniObjGen オプションを同時に指定すると，-JPN,V3JPNSpace オプションは無効となります。
- -JPN,Alnum および-JPN,V3JPN オプションを指定するとき，動的長基本項目はすべて字類を英数字として扱います。また，PIC N が指定された動的長基本項目の LIMIT 指定の値は，その 2 倍の値が指定されたものと仮定します。

(3) -EquivRule オプション

(a) 形式

```
-EquivRule, {NotExtend | NotAny | StdCode}
-noEquivRule
```

(b) 機能

拡張コード文字と標準コード文字を等価とみなさないようにするためのオプションです。

標準コードおよび拡張コードについては，マニュアル「COBOL2002 言語 標準仕様編」を参照してください。

-EquivRule,NotExtend

拡張コード文字と標準コード文字を等価とみなしません。

-EquivRule,NotAny

拡張コード文字と標準コード文字を等価とみなしません。さらに，標準コードの英大文字と標準コードの英小文字も等価とみなしません。

この場合，予約語および文脈依存語はすべて標準コードの英大文字で記述しなければなりません。

-EquivRule,StdCode

拡張コード文字と標準コード文字を等価とみなしません。さらに，日本語文字定数中に標準コードの空白を書いてもエラーとしません。ただし，日本語文字定数中に空白以外の標準コード文字を書いた場合は，警告レベルのエラーとなります。

なお，日本語文字定数の標準コード文字数は偶数としてください。日本語文字定数の標準コード文字数が奇数の場合，警告レベルのエラーとし，日本語文字定数に標準コードの空白を追加します。

また，72 カラムより前に改行文字がある COBOL 原始プログラムは，改行文字から 72 カラムまで空白に置き換えてコンパイルします。

-noEquivRule

-EquivRule オプションの指定を打ち消します。

(4) -UscoreStart オプション

(a) 形式

```
-UscoreStart
-noUscoreStart
```

(b) 機能

-UscoreStart

先頭が下線の CALL 定数を指定できるようにします。

先頭が下線のプログラム名を CALL 定数で指定した場合，定数の先頭から'_'または'__'（下線 2 個）を除いた名称でプログラムを呼び出します。

-noUscoreStart

-UscoreStart オプションの指定を打ち消します。

(5) -BinExtend オプション

(a) 形式

```
-BinExtend
-noBinExtend
```

(b) 機能

-BinExtend

用途（USAGE 句）が 2 進（BINARY，COMP，COMP-4）のデータ項目に指定できる初期値（VALUE 句の値）を拡張します。

PICTURE 句で指定したけた数と VALUE 句に指定できる値との対応は次のとおりです。

PICTURE 句で指定したけた数	VALUE 句に指定できる値
1～4 けた	-2^{15}～$2^{15}-1$
5～9 けた	-2^{31}～$2^{31}-1$
10～18 けた	-999,999,999,999,999,999～999,999,999,999,999,999

-noBinExtend

-BinExtend オプションの指定を打ち消します。

(6) -MinusZero オプション

(a) 形式

```
-MinusZero
-noMinusZero
```

(b) 機能

符号付き内部 10 進項目または符号付き外部 10 進項目への転記で切り捨てが発生した結果，負の符号を持つゼロ（以降，-0 と表記します）が発生することがあります。この-0 は演算や比較で数値として参照する場合は，正の符号を持つゼロ（以降，+0 と表記します）と同じに扱われ，プログラムの動作には影響しません。しかし，DISPLAY 文で表示したり，上位の集団項目で参照したり，REDEFINES 句で別の属性として参照した場合に，実行結果が異なることがあります。-MinusZero オプションは，符号付き内部 10 進項目および符号付き外部 10 進項目に，ほかのデータ項目の値や演算結果を転記する場合に発生した-0 を

強制的に+0 に変換して，-0 の発生を防止するオプションです。他システムや他社の COBOL とのデータの互換性や移行性が向上します。

（例 1）

```
       01 A PIC S9(5) USAGE DISPLAY.
       01 B PIC S9(5) USAGE DISPLAY VALUE 0.
           COMPUTE A = B - 0.1.
```

右辺の式の結果は-0.1 になり，それを A に格納する際に，小数部が切り捨てられて-0 となる。※
-MinusZero オプションを指定すると，-0 が+0 に変換されて A に格納される。

注※

ただし，演算結果が必ず-0 になるとは限りません。

（例 2）

```
       01 A PIC S9(4) USAGE DISPLAY.
       01 B PIC S9(5) USAGE DISPLAY VALUE -10000.
           MOVE B TO A.
```

A に格納する際に，10,000 の最上位けたの 1 が切り捨てられて-0 となる。

-MinusZero オプションを指定すると，-0 が+0 に変換されて A に格納される。

-MinusZero

負の符号を持つゼロ（-0）を強制的に正の符号を持つゼロ（+0）に変換します。

-noMinusZero

-MinusZero オプションの指定を打ち消します。

(c) 注意事項

- -MinusZero オプションは，符号付き外部 10 進項目への転記および符号付き内部 10 進項目への転記に対してだけ有効です。したがって，-MinusZero オプションを指定していても，REDEFINES 句や集団項目名で，10 進項目を 10 進項目以外の項目として転記した場合，-0（例えば，内部 10 進数 3 けたの場合には X'000D'）がそのまま転記されます。

(7) -TruncCheck オプション

(a) 形式

```
-TruncCheck
-noTruncCheck
```

(b) 機能

-TruncCheck

転記での送り出し側作用対象のサイズをチェックします。

送り出し側作用対象のけた数が受け取り側作用対象のけた数より大きい場合は，お知らせメッセージが出力されます。

-noTruncCheck

-TruncCheck オプションの指定を打ち消します。

(c) -TruncCheck オプションのチェック対象

-TruncCheck オプションは，次の項目について，転記での送り出し側作用対象のサイズをチェックします。

チェック対象となる項目

- 外部 10 進項目
- 内部 10 進項目
- 2 進項目（1 バイト 2 進項目を含む）
- 数字定数（16 進数字定数を含む）
- 固定長集団項目
- 英数字項目
- 英数字定数（16 進英数字定数を含む）
- ALL 英数字定数
- 数字型，整数型，または英数字型の，組み込み関数※および利用者定義関数

注※

送り出し側が英数字型の組み込み関数で，かつ次に示す条件の場合はチェックしません。

- 引数が部分参照されていて，長さが可変である
  長さが可変になるのは次のどちらかのときです。
  ・部分参照の長さがデータ名指定である
  ・部分参照の開始位置がデータ名指定で，長さが省略されている
- 組み込み関数が MAX 関数または MIN 関数で，データ名の引数が複数指定されている
- 組み込み関数が TRIM 関数または SUBSTRING 関数である

チェック対象となる転記の個所

- VALUE 句
- 画面節（WINDOW SECTION）の SOURCE 句
- ACCEPT 文の書き方 2（日付と時刻を取得する ACCEPT 文）
- INITIALIZE 文の REPLACING 指定
- MOVE 文
- READ 文の INTO 指定
- RELEASE 文の FROM 指定

- RETURN 文の INTO 指定
- SET 文の書き方 4（条件設定の SET 文）
- WRITE 文の FROM 指定
- COMPUTE 文（送り出し側作用対象が単一の数字項目や数字定数の場合）
- 上記以外の算術文（中間結果のけた数※1 が受け取り側作用対象のけた数より大きい場合※2）

注※1

中間結果のけた数については，マニュアル「COBOL2002　ユーザーズガイド」の「演算の中間結果」を参照してください。

注※2

Windows(x64) COBOL2002 の場合，-MaxDigits38 オプションおよび-IntResult,DecFloat40 オプションを指定しているときは，常に中間結果のけた数の方が大きいと判定します。

受け取り側作用対象が英数字の場合

- 受け取り側作用対象が固定長集団項目のときはバイト数で比較されます。
- 送り出し側作用対象が数字で，受け取り側作用対象が英数字のときは，送り出し側作用対象はけた数（小数けたを含む）で比較します。
- -H8Switch オプションが指定してあり，送り出し側作用対象が数字で受け取り側作用対象が英数字のときは，数字項目はバイト数で比較されます。

受け取り側作用対象が数字の場合

- 整数けた数で比較します。
- 次のすべての条件を満たすときはチェックしません。

  ・-DigitsTrunc オプションの指定がある

  ・-MaxDigits38 オプションおよび-IntResult,DecFloat40 オプションを指定していない（Windows(x64) COBOL2002 の場合）

  ・受け取り側作用対象が 2 進項目である

  ・VALUE 句の転記ではない
- 送り出し側作用対象が 18 けた※を超える英数字項目のときはチェックしません。

注※

Windows(x64) COBOL2002 の場合，-MaxDigits38 オプションおよび-IntResult,DecFloat40 オプションを指定しているときは，38 けたとなります。

(8)　-LowerAsUpper オプション

(a)　形式

```
-LowerAsUpper
-noLowerAsUpper
```

(b)　機能

-LowerAsUpper

CALL 定数の英小文字を英大文字に変換した名称でプログラムを呼び出します。

-noLowerAsUpper

-LowerAsUpper オプションの指定を打ち消します。

(9)　-CBLVALUE オプション

(a)　形式

```
-CBLVALUE
-noCBLVALUE
```

(b)　機能

-CBLVALUE

環境変数 CBLVALUE を有効にするためのオプションです。

環境変数 CBLVALUE の詳細については，「(22)　CBLVALUE」を参照してください。

-noCBLVALUE

-CBLVALUE オプションの指定を打ち消します。

(10)　-Repository オプション

(a)　形式

```
-Repository, {Gen | Sup}
-noRepository
```

(b)　機能

-Repository,Gen

ソースファイルからリポジトリファイルを作成するときに指定します。この場合，オブジェクトファイルは作成されません。

なお，翻訳単位（プログラム定義を除く）が未完成でも，シグニチャと呼ばれるインタフェース部分が決まっていれば，リポジトリファイルを作成できます。詳細は，マニュアル「COBOL2002　ユーザーズガイド」のリポジトリファイルの単独生成の説明を参照してください。

注意事項

- クラス定義，インタフェース定義，および関数定義が一つも格納されていないファイルに-Repository,Gen オプションを指定した場合，オプションの指定が無効となります。

- -Repository,Gen オプションと-Compile オプションが同時に指定された場合，-Repository,Gen オプションが有効になり，-Compile オプションが無効になります。

-Repository,Sup

リポジトリファイルを更新しません。

ただし，リポジトリファイルが存在しない場合は，新規に作成します。

また，オブジェクトファイルは，生成されます。

-noRepository

-Repository オプションの指定を打ち消します。

(11) -RepositoryCheck オプション

(a) 形式

```
-RepositoryCheck
-noRepositoryCheck
```

(b) 機能

-RepositoryCheck

同じソースファイル中の翻訳単位の定義と外部リポジトリ中の情報に相違があるかどうかをチェックし，相違がある場合には警告メッセージを出力します。-RepositoryCheck オプションを指定した場合，リポジトリファイルは更新されません。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「リポジトリファイルの生成方法」を参照してください。

-noRepositoryCheck

-RepositoryCheck オプションの指定を打ち消します。

(12) -Define オプション

(a) 形式

```
-Define 翻訳変数名〔=値〕〔,翻訳変数名〔=値〕〕…
-noDefine
```

(b) 機能

-Define 翻訳変数名〔=値〕〔,翻訳変数名〔=値〕〕…

翻訳変数名（条件翻訳で，ソース行の取り込みや読み飛ばしを制御する変数の名称）を定義します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「条件翻訳の利用」を参照してください。

-noDefine

-Define オプションの指定を打ち消します。

(c) 注意事項

- 翻訳変数名は，31 文字まで指定できます。32 文字以上の文字列を指定した場合，コンパイル時に警告メッセージが出力され，先頭の 31 文字だけが翻訳変数名として有効となります。
- 翻訳変数名は，-Define オプションに指定した文字列がそのまま使われます。この文字列には，COBOL 語の等価変換が適用されません。

  -Define オプションで定義した英小文字の翻訳変数名を COBOL プログラム中で参照するには，-EquivRule,NotAny オプションの指定が必要です。
- 翻訳変数の値は，160 バイトまで指定できます。161 バイト以上の文字列を指定した場合，コンパイル時に警告メッセージが出力され，先頭の 160 バイトだけが翻訳変数の値として有効となります。
- 翻訳変数名で使用できる文字は，COBOL の語で使用できる文字と同じです。ただし，コンパイラオプションの区切り文字となるイコール（=），コンマ（,）および半角空白文字は，指定できません。
- 翻訳変数の値を指定する場合の注意事項を，次に示します。

  1. 翻訳変数に指定した値は，すべて英数字定数として扱われます。

     （例 1）

     コンパイラオプションに，-Define VER=3 を指定したとき，ソース中に

     >>IF VER = '3'

     という記述があれば，上記条件が真となる。

     （例 2）

     コンパイラオプションに，-Define VER=5 を指定したとき，ソース中に

     >>IF VER = '3'

     という記述があれば，上記条件が偽となる。
  2. コンパイラオプションの区切り文字となるイコール（=），コンマ（,）は，指定できません。
  3. 文字列内に区切り文字（空白）を指定するときは，値全体をダブルコーテーション（"）で囲む必要があります。このとき，ダブルコーテーションは値として扱われません。

     （例）

     -Define DEF01="aaa bbb"

     翻訳変数名：DEF01

     値　　：aaa bbb

(13) -Details オプション

(a) 形式

```
-Details
-noDetails
```

(b) 機能

-Details

コンパイラオプションの詳細情報を出力します。

また，環境変数 CBLLINKER，環境変数 CBLMANIFESTTOOL，および環境変数 CBLRESOURCECOMPILER を指定した場合，環境変数に指定されたコマンドが起動するときに，コマンドのパス情報を出力します。

形式

ccbl2002 コマンドの絶対パス名: ccbl2002 オプション群　COBOL ソースファイル名群

ccbl2002 オプション群

ccbl2002 コマンドが認識するオプションがすべて出力されます。ほかのオプションを指定したために仮定されたオプションも出力されます。反対に，ほかのオプションを指定したために無視されたオプションは出力されません。オプションの引数を指定した場合は，オプションの引数も出力されます。

COBOL ソースファイル名群

コンパイル対象の COBOL ソースファイル名がすべて出力されます。

（例）

入力

```
ccbl2002 -TDInf -SrcList,CopyAll -SimSub PROG1 -Details sample1.cbl
```

出力

```
COBOL2002 インストールフォルダ¥bin¥ccbl2002: ccbl2002 -DebugInf -TDInf
-SrcList,CopyAll -Details -Optimize,1 -Lib,GUI -SimSub PROG1 -Main,System sample1.cbl
```

-noDetails

-Details オプションの指定を打ち消します。

(14) -OldForm オプション

(a) 形式

```
-OldForm ”旧オプションの並び”
```

(b) 機能

-OldForm "旧オプションの並び"

旧形式のオプション（PC COBOL85 用のコンパイラオプション）を指定できるようにします。

(c) 注意事項

- -OldForm オプションに指定する旧オプションの並びに，旧オプションの引数ではない単独ファイル名を指定した場合，そのファイル名は無視されます。ただし，旧オプションの引数として指定したファイル名については，有効となります。

  指定例を，次に示します。

  ```
  -OldForm ”-Mw sample.cbl subsample.cbl”
  ```

  sample.cbl は，-Mw オプションの引数として有効となりますが，subsample.cbl は旧オプションの引数でない単独のファイル名のため，無視されます。

- -OldForm オプションに指定する旧オプションの並びの中に，さらにダブルコーテーション（"）が存在する場合，ダブルコーテーションを「¥"」に置き換えて指定してください。指定例を，次に示します。

  ```
  -OldForm ”-Fw ¥”c:temp¥sample.cbw¥” -Mw sample.cbl”
  ```

- このオプションを開発マネージャで指定する場合，プロジェクト設定ダイアログボックスの［ユーザ設定］タブを使用してください。［ユーザ設定］タブの詳細は，「2.2.3　オプションの設定方法」を参照してください。ただし，プロジェクト構成のコンパイル方法に影響する次のオプションを指定できません。指定した場合，動作は保証しません。

  -Mw　-Mh　-Cs　-Dl　-Vb　-Ad　-Ax　-Dc

(15) -Help オプション

(a) 形式

```
-Help | -?
```

(b) 機能

ccbl2002 コマンドのヘルプを表示します。

(c) 注意事項

- -Help オプションを指定した場合，ほかのオプションやファイル名を指定しても無視されます。
- このオプションは，コマンドライン（ccbl2002 コマンド）で指定できます。開発マネージャでは，指定できません。

(16) -UniObjGen オプション

(a) 形式

```
-UniObjGen
-noUniObjGen
```

(b) 機能

-UniObjGen

シフト JIS で記述された COBOL ソースプログラムから，英数字定数を UTF-8 に，日本語文字定数を UTF-16LE，または UTF-16BE に変換したオブジェクトファイルを生成します。

-noUniObjGen

-UniObjGen オプションの指定を打ち消します。

(c) 注意事項

- 日本語文字定数の文字コードのバイトオーダは，-UniEndian オプションの指定に従います。
- このオプションを指定する場合，次のオプションは同時に指定できません。同時に指定した場合，このオプションが有効となり，次のオプションは無効になります。

  -JPN　-CompatiV3　-V3Rec　-V3RecFCSpace　-V3RecEased

  ただし，コンパイラ環境変数 CBLV3UNICODE に YES を指定すると，このオプションを指定しても-CompatiV3，-V3Rec，-V3RecFCSpace，および-V3RecEased オプションは有効となります。

(17) -UniEndian オプション

(a) 形式

```
-UniEndian,{Little | Big}
-noUniEndian
```

(b) 機能

-UniEndian,Little

日本語文字定数を UTF-16LE に変換したオブジェクトファイルを生成します。

-UniEndian,Big

日本語文字定数を UTF-16BE に変換したオブジェクトファイルを生成します。

-noUniEndian

-UniEndian オプションの指定を打ち消します。

(c) 注意事項

- このオプションは，-UniObjGen オプションが指定された場合だけ有効になります。
- -UniObjGen オプションが指定されていて，-UniEndian オプションが指定されていない場合，日本語文字定数は UTF-16LE のコードに変換します。

(18) -MaxDigits38 オプション（Windows(x64) COBOL2002 で有効）

(a) 形式

```
-MaxDigits38
-noMaxDigits38
```

(b) 機能

-MaxDigits38

数字項目，数字編集項目，および数字定数に指定できる最大けた数を 18 けたから 38 けたに拡張します。外部 10 進形式および内部 10 進形式の数字項目と固定小数点数字定数で，けた数を拡張できます。

詳細については，マニュアル「COBOL2002　ユーザーズガイド」，およびマニュアル「COBOL2002　言語　拡張仕様編」の数字項目のけた拡張機能を参照してください。

-noMaxDigits38

-MaxDigits38 オプションの指定を打ち消します。

(c) 注意事項

- -IntResult,DecFloat40 オプションと同時に指定してください。

  -IntResult,DecFloat40 オプションの指定がない場合，エラーメッセージが出力されます。
- -MaxDigits38 オプション，-IntResult,DecFloat40 オプション，および-Optimize,3 オプションを同時に指定した場合，-Optimize,3 オプションは無効となり，-Optimize,2 オプションが仮定されます。
- -MaxDigits38 オプションを指定しても，19～38 けたの数字項目および数字定数が指定できない機能や文などがあります。詳細については，マニュアル「COBOL2002　ユーザーズガイド」，およびマニュアル「COBOL2002　言語　拡張仕様編」の数字項目のけた拡張機能を参照してください。

(19) -IntResult,DecFloat40 オプション（Windows(x64) COBOL2002 で有効）

(a) 形式

```
-IntResult,DecFloat40
-noIntResult
```

(b) 機能

-IntResult,DecFloat40

算術演算の中間結果の表現形式を 40 けた 10 進浮動小数点形式とします。詳細については，マニュアル「COBOL2002　ユーザーズガイド」の数字項目のけた拡張機能を参照してください。

-noIntResult

-IntResult オプションの指定を打ち消します。

(c) 注意事項

- -MaxDigits38 オプションと同時に指定してください。-MaxDigits38 オプションの指定がない場合，エラーメッセージが出力されます。
- -IntResult,DecFloat40 オプション，-MaxDigits38 オプション，および-Optimize,3 オプションを同時に指定した場合，-Optimize,3 オプションは無効となり，-Optimize,2 オプションが仮定されます。
- -IntResult,DecFloat40 オプション，-MaxDigits38 オプション，および-Compati85,Power オプション（-Compati85,All 指定による仮定時を含む）を同時に指定した場合，-Compati85,Power オプションは無効となります。
- -IntResult,DecFloat40 オプションおよび-CompatiV3 オプションを同時に指定した場合，-CompatiV3 オプションの仕様が一部無効となります。詳細は，「2.3.12　他システムとの移行の設定」の「(2)　-CompatiV3 オプション」を参照してください。

(20) -LiteralExtend オプション

(a) 形式

```
-LiteralExtend,Alnum
-noLiteralExtend
```

(b) 機能

-LiteralExtend,Alnum

定数長拡張機能は，英数字定数の最大長を拡張します。拡張する項目と制限値を次に示します。

表 2-19　最大長を拡張する項目と制限値

最大長を拡張する項目	制限値および限界値	
	-LiteralExtend,Alnum なし	-LiteralExtend,Alnum あり
英数字定数の長さ	1～160 文字（バイト）	1～8,191 文字（バイト）
STOP 文の定数に指定した英数字定数の長さ	1～160 文字（バイト）	1～8,191 文字（バイト）
定数指定する場合のプログラム名，メソッド名の長さ	1～160 文字（バイト）	1～1,024 文字（バイト）

ただし，次に示す項目では英数字定数の最大長は拡張されません。

表 2-20　最大長を拡張しない項目

最大長を拡張しない項目	制限値および限界値（-LiteralExtend の有無に関係なく同じ）
連結式で連結した英数字定数の長さ	2～1,024 文字（バイト）
指定できる文字位置のけた数が個別に規定されている構文	各構文の規定に従う
COPY 文，REPLACE 文の構文中に指定する原文語の長さ	1～322 文字（バイト）
翻訳指令行で使用する英数字定数の長さ	1～160 文字（バイト）
SQL 文中で使用する英数字定数の長さ	1～160 文字（バイト）

-noLiteralExtend

-LiteralExtend オプションの指定を打ち消します。

(c)　注意事項

- -LiteralExtend オプションは，次のオプションと背反関係にあり，同時に指定した場合は，-LiteralExtend オプションが無効となり，次のオプションが有効となります。

  -StdMIA　-Std85　-Std2002　-V3Spec　-Compati85,Syntax　-CompatiV3

  -SimMain　-SimSub　-SimIdent
- COBOL プログラム中に COPY 文または REPLACE 文がある場合，322 バイトを超える英数字定数に対して，原文語の長さが 322 文字を超えている旨の警告エラーを出力することがありますが，プログラムの動作には影響しません。

(21)　-SpaceAsZero オプション

(a)　形式

```
-SpaceAsZero
-noSpaceAsZero
```

(b)　機能

-SpaceAsZero

外部 10 進項目に空白文字（X'20'）データがあるとき，ゼロ（X'30'）とみなして比較，演算，転記を実行します。このオプションを指定すると，約 1.5 倍の実行性能劣化となります。このため，必要のないかぎり，このオプションは指定しないでください。

-noSpaceAsZero

-SpaceAsZero オプションの指定を打ち消します。

(c)　注意事項

- -SpaceAsZero オプションは，-Compati85,All オプションと同時に指定してください。-Compati85,All オプションの指定がない場合，エラーメッセージが出力されます。なお，-

Compati85,All オプションと同じ意味となる-Compati85 のサブオプションが指定された場合，-SpaceAsZero オプションの指定が有効になります。

- -SpaceAsZero オプションと-Optimize,3 オプションを同時に指定した場合，-Optimize,3 オプションは無効となり，-Optimize,2 オプションが仮定されます。
- -DebugData オプションが指定されている場合，比較，演算，転記の外部 10 進項目に含まれる空白文字（X'20'）は，-SpaceAsZero オプションを指定しても，不当なデータ（データ例外）として検出されます。
- 外部 10 進項目に空白文字（X'20'）データがあるとき，比較，演算，転記以外の実行結果（画面節（SCREEN SECTION および WINDOW SECTION），索引ファイルのキー項目，整列・併合処理のキー指定，DISPLAY 文，組み込み関数，SQL などの実行結果）は，-SpaceAsZero オプションを指定しても保証しません。

(22) -CheckUninitData オプション

(a) 形式

```
-CheckUninitData
-noCheckUninitData
```

(b) 機能

-CheckUninitData

ソースに閉じた初期化漏れチェック機能を有効にします。ソースに閉じた初期化漏れチェック機能については，マニュアル「COBOL2002 ユーザーズガイド」の「初期化漏れチェック機能」を参照してください。

-noCheckUninitData

-CheckUninitData オプションの指定を打ち消します。

(c) 注意事項

- このオプションは，-Compile,CheckOnly オプションと同時に指定する必要があります。-Compile,CheckOnly オプションを指定していない場合，または-Compile,CheckOnly オプションと，-Compile,NoLink オプションもしくは-Repository,Gen オプションを同時に指定した場合，-CheckUninitData オプションは無効となります。

(23) -FunctionECSup,CodeConvErr オプション

(a) 形式

```
-FunctionECSup,CodeConvErr
-noFunctionECSup
```

(b)　機能

-FunctionECSup,CodeConvErr

組み込み関数の DISPLAY-OF 関数または NATIONAL-OF 関数で，変換前の文字に対応する変換後の文字を持たない文字が引数中にある場合，文字変換を行わないで，EC-ARGUMENT-FUNCTION／EC-ARGUMENT-IMP 例外が成立しないようにします。

組み込み関数の DISPLAY-OF 関数または NATIONAL-OF 関数のエラー情報の取得には，CBLCNVERRORINFO サービスルーチンを使用します。CBLCNVERRORINFO サービスルーチンについては，マニュアル「COBOL2002　ユーザーズガイド」の「CBLCNVERRORINFO」を参照してください。

なお，このオプションの指定がない場合は，EC-ARGUMENT-FUNCTION／EC-ARGUMENT-IMP 例外が成立します。

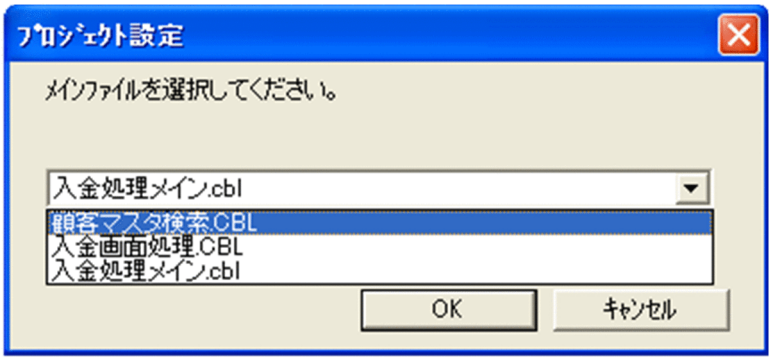
-noFunctionECSup

-FunctionECSup オプションの指定を打ち消します。

2.3.15　プロジェクトの種類の設定時の注意事項

プロジェクト設定ダイアログボックスの［プロジェクト］タブを選んで表示されるページで設定する内容について説明します。

「System 指定メインプログラム」，「V3 指定メインプログラム」を選んだ場合，メインファイル名も指定する必要があります。メインファイル名を選ぶダイアログボックスが表示されますので，プロジェクトのメインファイルに指定するソースファイル名を選んで［OK］ボタンを選んでください。



2.4　環境変数の設定

この節では，コンパイル時に使用する環境変数の設定方法，および機能について説明します。

2.4.1　環境変数の設定方法

(1)　システムに従った環境変数の設定方法

ccbl2002 コマンドのコンパイラ環境は，環境変数で設定できます。環境変数は，コマンドプロンプトまたは Windows の環境変数で設定します。

形式

環境変数名=環境変数の値

注意事項

空白を含むフォルダ名やファイル名を環境変数の値に指定するときは，引用符（"）で囲まないでください。

(2)　コンパイラ環境変数の規則

環境変数に YES を設定する場合，大文字と小文字は等価とみなされます。

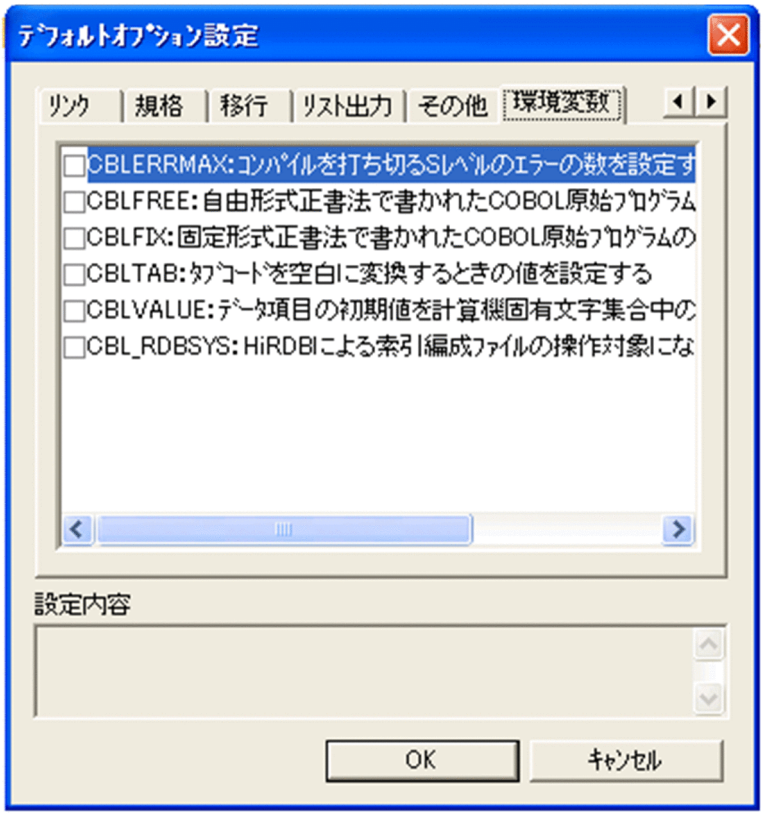
(3)　デフォルトの環境変数の設定方法

開発マネージャでは，プロジェクトを新規作成したときにデフォルトで設定する環境変数をあらかじめ登録できます。

デフォルトの環境変数は，デフォルトオプション設定ダイアログボックスで設定します。ダイアログボックスの表示方法を次に示します。

1. **開発マネージャのプロジェクトマスタを閉じた状態で，ウィンドウの［プロジェクトマスタ］メニューから［デフォルトオプションの設定］を選ぶ。**

   デフォルトオプション設定ダイアログボックスが表示されます。



(4)　プロジェクト単位，またはファイル単位の環境変数の設定

プロジェクト単位，またはソースファイル単位で環境変数を設定できます。

プロジェクト単位，またはソースファイル単位の環境変数の設定は，プロジェクト設定ダイアログボックスで設定します。ダイアログボックスの表示方法を次に示します。

1. 開発マネージャのツリービューウィンドウから，環境変数を設定するプロジェクト，またはソースファイルを選ぶ。

2. ［プロジェクト］メニューから［プロジェクトの設定］を選ぶ。または，［ファイル］メニューから［ファイルの設定］を選ぶ。

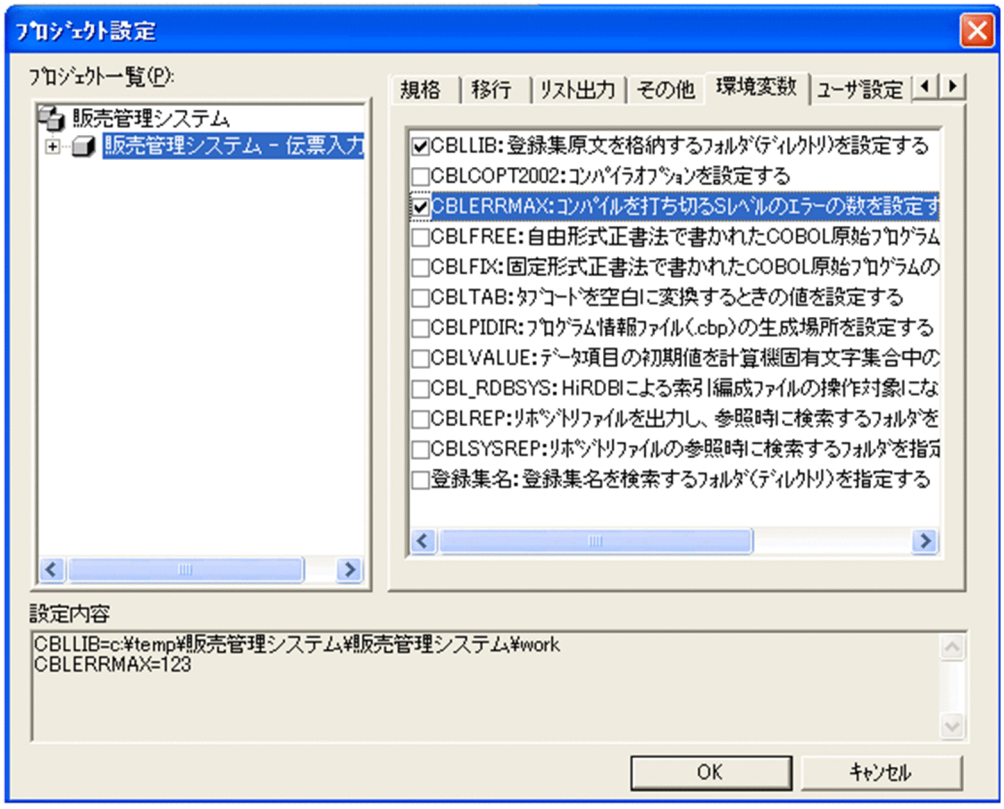
   プロジェクト設定ダイアログボックスが表示されます。

(5)　環境変数の設定方法

(a)　環境変数を一覧から選択する方法

1. ［環境変数］タブを選び，表示されるページで環境変数を設定して［OK］ボタンを選ぶ。

   環境変数が設定されます。また，設定された環境変数の内容が「設定内容」欄に表示されます。



［環境変数］タブを選んで表示されるページでの設定方法を次に示します。

環境変数の登録

登録する環境変数のチェックボックスを選びます。

環境変数の値を入力するテキストボックス，または環境変数設定ダイアログボックスが表示されます。値を指定すると，環境変数が登録されます。

環境変数が登録されると，チェックボックスがオンになります。

環境変数の変更

変更する環境変数のチェックボックスを選びます。

現在の環境変数の値が，テキストボックス，または環境変数設定ダイアログボックスが表示されます。値を変更してください。

環境変数の削除

削除する環境変数のチェックボックスを選びます。

環境変数の値をすべて削除すると，環境変数の指定も削除されます。

環境変数が解除されると，チェックボックスがオフになります。

(b)　環境変数を直接入力する方法

次の方法で指定します。指定した環境変数は，［ユーザ設定］タブを選んで表示されるページの「環境変数」の欄に表示されます。

環境変数の追加

1. [ユーザ設定] タブを選んで表示されるページの「名前」の欄に環境変数名，「値」の欄に環境変数に指定する値をそれぞれ指定する。
2. [設定] ボタンを選ぶ。

   環境変数が追加されます。

環境変数の変更

1. [ユーザ設定] タブを選んで表示されるページの「環境変数」の欄から変更したい環境変数を選ぶ。

   「名前」および「値」の欄に，選んだ環境変数の名前と現在の値が表示されます。
2. 値を修正して［設定］ボタンを選ぶ。

   環境変数が変更されます。

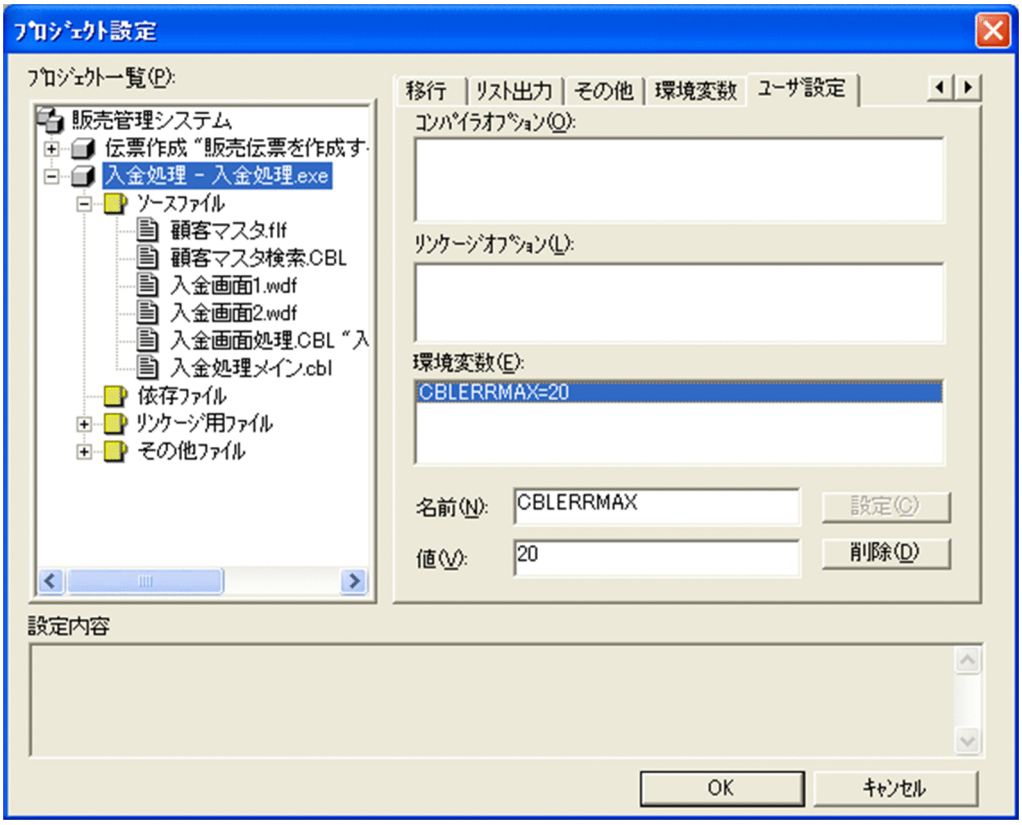
環境変数の削除

1. [ユーザ設定] タブを選んで表示されるページの「環境変数」の欄から削除したい環境変数を選ぶ。

   「名前」および「値」の欄に，選んだ環境変数の名前と現在の値が表示されます。
2. [削除] ボタンを選ぶ。

   環境変数が削除されます。

環境変数が設定されると，環境変数名と指定した値が設定内容の欄に表示されます。ただし，［ユーザ設定］タブを選んで表示されるページでは，設定内容の欄は表示されません。



(6) 注意事項

- ソースファイルに設定した環境変数は，そのソースファイルだけで有効です。また，プロジェクトに設定した環境変数は，プロジェクト内だけに有効です。
- システムの環境変数と同じものについては，ここで設定した内容だけが有効になります。
- ［環境変数］タブを選んで表示されるページと，［ユーザ設定］タブを選んで表示されるページの両方で同じ環境変数を指定した場合，後者が有効となります。
- プロジェクト設定ダイアログで設定した環境変数は，ビルド，リビルド，クイックビルド，ファイル単位のコンパイル，依存関係の更新，および開発マネージャから起動したツールで有効になります。
  ［ビルド］－［実行］メニューから実行するプログラムの起動時には無効になります。ただし，実行するプログラムが，実行支援で環境変数を設定されている場合は実行支援の環境変数が有効になります。
  実行時環境変数は，実行支援で設定できます。詳細は，「3.3.2　実行環境の設定」を参照してください。
  また，システム環境変数やユーザ環境変数に指定することを指示されている環境変数は，開発マネージャ上では指定しないでください。

2.4.2　コンパイラ環境変数の一覧

コンパイラ環境変数の一覧を，次に示します。

表 2-21　コンパイラ環境変数の一覧

項番	環境変数	マネージャ	設定内容
1	CBL_RDBSYS	○	HiRDB による索引編成ファイルで操作対象となるデータベースシステムの種別
2	CBL_UNINITDATA_BREAKOFF	×	初期化漏れチェック機能使用時，初期化漏れチェック処理を打ち切るかどうか
3	CBLCOPT	×	ccbl コマンドに指定する旧形式のオプション列
4	CBLCOPT2002	○	ccbl2002 コマンドに指定する新形式のオプション列
5	CBLERRMAX	○	コンパイルを打ち切る S レベルのエラーの数
6	CBLFIX	○	固定形式正書法の COBOL ソースファイルの拡張子
7	CBLFIXEDFORMLINE	×	固定形式正書法の 1 行の長さの制限値
8	CBLFREE	○	自由形式正書法の COBOL ソースファイルの拡張子
9	CBLINITVALUE	○	初期化属性プログラムが呼ばれたときの VALUE 句の指定がないデータ項目の初期値（X'00'）
10	CBLLIB	○	登録集原文の検索フォルダ
11	CBLLINKER	○	コンパイラが使用するリンカの絶対パス名
12	CBLLINKINTERVAL	○	対象ファイルへのアクセス処理の前に空ける間隔（ミリ秒単位）

項番	環境変数	マネージャ	設定内容
13	CBLMANIFESTTOOL	○	コンパイラが使用するマニフェストツールの絶対パス名
14	CBLPIDIR	○	プログラム情報ファイル（.cbp）の生成先フォルダ
15	CBLREP	○	リポジトリファイルの出力先のフォルダ，およびリポジトリ段落で指定した翻訳単位名を含むリポジトリファイルを検索するフォルダ
16	CBLRESOURCECOMPILER	○	コンパイラが使用するリソースコンパイラの絶対パス名
17	CBLSYSREP	○	リポジトリファイルの参照時に検索するフォルダ
18	CBLTAB	○	タブコードを空白に変換するときの値
19	CBLUNINITDATA_EXITPROGRAM	×	初期化漏れチェック機能使用時，「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムの EXIT PROGRAM 文を，GOBACK 文とみなすかどうか
20	CBLUNINITDATA_OUTRESULTLIST	×	初期化漏れチェック機能使用時，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧を出力するかどうか
21	CBLUNINITDATA_OUTRESULTLISTDIR	×	初期化漏れチェック機能使用時，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧の出力先フォルダ
22	CBLVALUE	○	VALUE 句の指定のないデータ項目の初期値
23	CBLV3UNICODE	○	-UniObjGen オプション指定時に，-CompatiV3 オプションおよび-V3Rec オプションの指定を有効とするかどうか
24	登録集環境変数	○	登録集原文の検索フォルダ

（凡例）

○：開発マネージャから指定できる。

×：開発マネージャから指定できない。

2.4.3　コンパイラ環境変数の詳細

(1)　CBL_RDBSYS

HiRDB による索引編成ファイルの操作対象になるデータベースシステムを設定します。指定できる値は，HiRDB だけです。この環境変数の指定がない場合，および指定した値に誤りがある場合は，HiRDB が仮定されます。

（例）

```
CBL_RDBSYS=HiRDB
```

(2)　CBL_UNINITDATA_BREAKOFF

初期化漏れチェック処理を打ち切るかどうかを指定します。

この環境変数の値に NO を指定すると，初期化漏れチェック処理は打ち切られません。プログラムを実行したときに通る可能性のある経路（制御ブロック）をすべて走査するまで，初期化漏れチェック処理を続行します。

次の場合は，この環境変数の指定は無効となり，初期化漏れチェック処理で走査した制御ブロックの数が内部的な上限を超えると，初期化漏れチェック処理は打ち切られます。

- この環境変数の指定がない，または環境変数の値に NO 以外を指定している

（例）

```
CBL_UNINITDATA_BREAKOFF=NO
```

(3) CBLCOPT

ccbl コマンドに指定するオプション列（コンパイラオプションの並び）を設定します。この環境変数に設定しておけば，ccbl のコマンドラインにオプションを指定する必要がなくなります。ただし，この環境変数は COBOL85 からの移行を目的とする場合にだけご使用ください。

環境変数 CBLCOPT にファイル名を指定した場合は，ファイル名が無視されます。

各オプションは空白で区切って指定します。

（例）

```
CBLCOPT=-S1 -T4 -Ek
```

旧形式のオプションを開発マネージャで指定する場合，プロジェクト設定ダイアログボックスの［ユーザ設定］タブのコンパイラオプション欄で，-OldForm オプションを使用してください。［ユーザ設定］タブの詳細は，「2.2.3　オプションの設定方法」参照してください。

(4) CBLCOPT2002

ccbl2002 コマンドに指定するオプション列（コンパイラオプションの並び）を設定します。この環境変数に設定しておけば，ccbl2002 のコマンドラインにオプションを指定する必要がなくなります。

環境変数 CBLCOPT2002 にファイル名を指定した場合は，ファイル名が無視されます。また，開発マネージャを使用する場合は，プロジェクト構成のコンパイル方法に影響する次のオプションを指定できません。指定した場合，動作は保証しません。

-Main　-MainNotCBL　-Dll

各オプションは空白で区切って指定します。

（例）

```
CBLCOPT2002=-StdVersion,1 -DebugData -Switch,EBCDIK
```

(5) CBLERRMAX

コンパイルを打ち切るSレベルのエラーの数を設定します。設定した個数分のエラーが発生すると，メッセージが出力され，コンパイルが打ち切られます。

設定できる範囲は0～999,999で，省略時は30が仮定されます。コンパイルを続行し，すべてのエラーメッセージを出力したい場合は0を設定します。

なお，コンパイルリストを出力する場合（-SrcListオプションを指定した場合），この環境変数は無効となります。

（例）

```
CBLERRMAX=15
```

(6) CBLFIX

固定形式正書法で書かれたCOBOL原始プログラムとしてコンパイルするCOBOLソースファイルの拡張子を設定します。ただし，拡張子.cbl，.cob，.ocbの付いたファイルは，ここで設定しなくても固定形式正書法で書かれたCOBOL原始プログラムとしてコンパイルされます。

拡張子は，先頭のピリオド（.）と3文字以内の英数字で指定します。複数の拡張子を設定する場合は，それぞれの拡張子を半角空白文字で区切って指定します。

（例）

```
CBLFIX=.aaa .bbb .ccc
```

(7) CBLFIXEDFORMLINE

固定形式正書法の1行の長さの制限値（バイト単位）を80または255で指定します。次の場合は，80を仮定します。

- この環境変数の指定がない場合
- この環境変数に80または255以外を指定している場合
- 次のコンパイラオプションのどれかが有効である場合
  -V3Spec，-V3Rec,Fixed，-TDInf，-CVInf

（例）

```
CBLFIXEDFORMLINE=255
```

注意事項

- この環境変数に255を指定した場合でも，固定形式正書法のプログラム原文領域は，72カラムまでです。

- この環境変数の指定値（制限値）を超える部分は無視されます。コンパイルリストの原始プログラムリストに，制限値を超える部分は表示されません。

(8) CBLFREE

自由形式正書法で書かれた COBOL 原始プログラムとしてコンパイルする COBOL ソースファイルの拡張子を設定します。ただし，拡張子.cbf，.ocf の付いたファイルは，ここで設定しなくても自由形式正書法で書かれた COBOL 原始プログラムとしてコンパイルされます。

拡張子は，先頭のピリオド（.）と 3 文字以内の英数字で指定します。複数の拡張子を設定する場合は，それぞれの拡張子を半角空白文字で区切って指定します。

（例）

```
CBLFREE=.aaa .bbb .ccc
```

(9) CBLINITVALUE

初期化属性プログラムの作業場所節にある，VALUE 句の指定がないデータ項目の初期値を NULL（X'00'）に設定するときに，環境変数 CBLINITVALUE に NULL を設定します。初期化属性プログラムが呼ばれるたびに，初期値が設定されます。

環境変数 CBLINITVALUE に NULL 以外の値を設定した場合は，環境変数 CBLINITVALUE は無視されます。

（例）

```
CBLINITVALUE=NULL
```

環境変数 CBLINITVALUE は，次のデータ項目には適用されません。

- VALUE 句の指定があるデータ項目
- ADDRESSED 句の指定があるデータ項目
- EXTERNAL 句の指定があるデータ項目

注意事項

初期化属性プログラムの作業場所節に対する初期化は，環境変数 CBLVALUE と環境変数 CBLINITVALUE が両方とも有効な場合と，どちらか片方だけが有効な場合とで初期値が異なります。詳細については，マニュアル「COBOL2002　ユーザーズガイド」の「プログラム属性」の「初期化属性プログラム」を参照してください。また，環境変数 CBLVALUE については，「(22) CBLVALUE」を参照してください。

(10) CBLLIB

登録集原文を検索するフォルダを設定します。フォルダを複数指定する場合は，セミコロン（;）で区切って指定します。

（例）

```
CBLLIB=¥usr¥user¥copylib;c:¥source¥copy
```

なお，登録集環境変数が設定してある場合，登録集環境変数で設定したフォルダの方が環境変数 CBLLIB で指定したフォルダよりも優先します。フォルダによる検索順序を次に示します。

1. 登録集環境変数で設定したフォルダ
2. 環境変数 CBLLIB で設定したフォルダ
3. カレントフォルダ

開発マネージャを使用した場合の登録集原文の検索順序

開発マネージャからビルドを実行する場合，上記の 2.，3.については，開発マネージャによって次の検索順序が仮定されます。

1. プロジェクト設定ダイアログボックスで，ファイルに対して設定した環境変数 CBLLIB に指定されているフォルダ
2. プロジェクト設定ダイアログボックスで，プロジェクトに対して設定した環境変数 CBLLIB に指定されているフォルダ
3. プロジェクトの作業フォルダ
4. プロジェクトに登録されている登録集原文が格納されているすべてのフォルダ
5. プロジェクトに登録されている COBOL ソースファイルが格納されているすべてのフォルダ
6. システムに設定されている環境変数 CBLLIB に指定されているフォルダ

開発マネージャは上記 1.～6.のフォルダを仮定するため，1.の方法で明示的に検索フォルダを設定してください。

(11) CBLLINKER

コンパイラが呼び出すリンカを変更したい場合に LINK コマンド（LINK.exe）の絶対パス名を指定します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「リンカパスの切り替え機能」を参照してください。

（例）

```
CBLLINKER=C:¥Program Files¥…¥BIN¥LINK.exe
```

注意事項

- LINK コマンドを相対パスで指定しないでください。指定した場合の動作は保証しません。

- この製品が同梱する LINK コマンドより古いバージョンの LINK コマンドを指定しないでください。バージョンは，LINK コマンドファイルのプロパティからファイルバージョンを参照してください。古いバージョンの LINK コマンドを指定した場合の動作は保証しません。
- この環境変数を指定する場合，環境変数 PATH と LIB には，環境変数に指定した LINK コマンドが必要とするパスをこの製品のパスより優先して指定してください。指定しなかった場合の動作は保証しません。
- この環境変数を指定する場合，環境変数 CBLRESOURCECOMPILER および環境変数 CBLMANIFESTTOOL を同時に指定してください。同時に指定しなかった場合の動作は保証しません。
- この環境変数に指定する値の長さは 256 バイト以内でなければなりません。

  256 バイトを超える値を指定した場合，エラーメッセージが出力されて，コンパイルが中止されます。このとき，生成対象のファイル（実行可能ファイルまたは DLL ファイル）が残っている場合は削除します。
- COBOL2002 コンパイラは LINK コマンドの起動に失敗した場合，エラーメッセージが出力されて，コンパイルが中止されます。このとき，生成対象のファイル（実行可能ファイルまたは DLL ファイル）が残っている場合は削除します。

(12) CBLLINKINTERVAL

ccbl2002 コマンドまたは ccbl コマンドで，実行可能ファイルまたは DLL を作成する際に，対象ファイルへのアクセス処理の前に空ける間隔をミリ秒単位で指定します。指定できる範囲は，1～60,000 であり，省略時および範囲外の値の場合は，この環境変数の指定を無効とします。

なお，環境変数 CBLLINKINTERVAL の指定が有効な場合は，対象ファイルが占有可能かどうかも合わせてチェックします。

（例）対象ファイルへのアクセス間隔を 200 ミリ秒とする場合

```
CBLLINKINTERVAL=200
```

ccbl2002 コマンドまたは ccbl コマンドから呼び出す LINK コマンドや MT コマンドは，出力ファイルが OS やウィルスチェッカなどから，一時的に共有されることが原因でエラーになる場合があります。

この場合，時間をおいて再コンパイルしてください。時間をおいても問題が解決しないときは，環境変数 CBLLINKINTERVAL で対象ファイルへのアクセス間隔を変更した状態で再コンパイルすることで対処してください。

注意事項

- 環境変数 CBLLINKINTERVAL で指定する値は，開発環境で使用されるディスク装置，マシン環境，およびシステムの仕様に依存するため，実際の試行（検証）によって最適な値を使用してください。
- 環境変数 CBLLINKINTERVAL の指定は，実行可能ファイルまたは DLL を作成する際のファイルアクセスごとに有効となります。また，ファイルアクセスごとに，ファイルの占有ができるかを最

大で５回チェックします。このため，環境変数 CBLLINKINTERVAL を指定すると，指定した値以上にコンパイル時間が遅くなります。

- 環境変数 CBLLINKINTERVAL が有効な場合，開発マネージャでのビルド，クイックビルド，リビルドの際にもそれぞれの処理時間が遅くなります。また，それぞれの処理でのリンクの実行中は，リンクが完了するまでビルドの中止がすぐにできないため，環境変数 CBLLINKINTERVAL を指定する場合は注意が必要です。

(13) CBLMANIFESTTOOL

コンパイラが呼び出すマニフェストツールを変更したい場合に MT コマンド（MT.exe）の絶対パス名を指定します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「リンカパスの切り替え機能」を参照してください。

（例）

```
CBLMANIFESTTOOL=C:¥Program Files¥…¥bin¥MT.exe
```

注意事項

- MT コマンドを相対パスで指定しないでください。指定した場合の動作は保証しません。
- この環境変数を指定する場合，環境変数 PATH には環境変数に指定した MT コマンドが必要とするパスを Windows SDK の bin へのパスより優先して指定してください。指定しなかった場合の動作は保証しません。
- この環境変数を指定する場合，環境変数 CBLLINKER および環境変数 CBLRESOURCECOMPILER を同時に指定してください。同時に指定しなかった場合の動作は保証しません。
- この環境変数に指定する値の長さは 256 バイト以内でなければなりません。256 バイトを超える値を指定した場合，エラーメッセージが出力され，コンパイルが中止されます。このとき，生成対象のファイル（実行可能ファイルまたは DLL ファイル）が残っている場合は削除します。
- COBOL2002 コンパイラは MT コマンドの起動に失敗した場合，エラーメッセージが出力され，コンパイルが中止されます。このとき，生成対象のファイル（実行可能ファイルまたは DLL ファイル）が残っている場合は削除します。

(14) CBLPIDIR

プログラム情報ファイル（.cbp）を任意のフォルダに生成したい場合に設定します。プログラム情報ファイルは，テストデバッガを使用するときに必要となるファイルです。設定できるフォルダは一つだけです。

（例）

```
CBLPIDIR=c:¥temp
```

(15) CBLREP

リポジトリファイル（.rep）を任意のフォルダに生成，または更新したい場合に，出力先のフォルダを指定します。指定されたフォルダは，リポジトリ段落で指定された名前の翻訳単位（関数定義，クラス定義，またはインタフェース定義）を含むリポジトリファイル参照時の検索対象となります。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「定義別のコンパイル方法とリポジトリファイル」についての説明を参照してください。

規則

- 指定するフォルダが複数ある場合は，各フォルダをセミコロン（;）で区切って指定します。
- 指定したフォルダ群に同じ名称のリポジトリファイルが複数存在する場合，先に指定したフォルダに含まれるリポジトリファイルを優先します。
- 生成または更新時に，同じ名称のリポジトリファイルが存在しない場合，最初に指定されたフォルダにリポジトリファイルを新規に生成します。
- リポジトリファイル検索時に同じ名称のリポジトリファイルが見つからなければ，カレントフォルダが検索されます。また，環境変数 CBLREP の指定がない場合，リポジトリファイルの生成，更新，および検索は，カレントフォルダが対象となります。

（例）

フォルダとして¥usr¥user¥replib を指定します。

```
CBLREP=¥usr¥user¥replib
```

(16) CBLRESOURCECOMPILER

コンパイラが呼び出すリソースコンパイラを変更したい場合に RC コマンド（RC.exe）の絶対パス名を指定します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「リンカパスの切り替え機能」を参照してください。

（例）

```
CBLRESOURCECOMPILER=C:¥Program Files¥…¥bin¥RC.exe
```

注意事項

- RC コマンドを相対パスで指定しないでください。指定した場合の動作は保証しません。
- この環境変数を指定する場合，環境変数 PATH には環境変数に指定した RC コマンドが必要とするパスを Windows SDK の bin へのパスより優先して指定してください。指定しなかった場合の動作は保証しません。
- この環境変数を指定する場合，環境変数 CBLLINKER および環境変数 CBLMANIFESTTOOL を同時に指定してください。同時に指定しなかった場合の動作は保証しません。

- 環境変数に指定する値の長さは 256 バイト以内でなければなりません。256 バイトを超える値を指定した場合，エラーメッセージが出力されて，コンパイルが中止されます。このとき，生成対象のファイル（リソースファイル）が残っている場合は削除します。
- COBOL2002 コンパイラは RC コマンドの起動に失敗した場合，エラーメッセージを出力し，コンパイルを中止します。このとき，生成対象のファイル（リソースファイル）が残っている場合は削除します。

(17) CBLSYSREP

リポジトリ段落で指定された名前の翻訳単位（関数定義，クラス定義，またはインタフェース定義）を含むリポジトリファイルを検索するフォルダを指定します。

環境変数 CBLSYSREP に指定したフォルダに格納されたリポジトリファイルは，リポジトリファイルに情報が格納されている翻訳単位と同じ名称の翻訳単位を作成してコンパイルした場合でも，上書きされることはありません。このため，環境変数 CBLSYSREP には，主に（DLL とリポジトリファイルだけが提供されている場合など）生成元ソースファイルのないリポジトリファイルの検索フォルダを指定する場合に使用します。

詳細は，マニュアル「COBOL2002　ユーザーズガイド」の定義別のコンパイル方法とリポジトリファイルについての説明を参照してください。

規則

- 指定するフォルダが複数ある場合は，各フォルダをセミコロン（;）で区切って指定します。
- リポジトリファイルの検索順序は，カレントフォルダが優先されます。

（例）

フォルダとして c:¥usr¥user¥sysreplib と d:¥users¥lib¥rep を指定します。

```
CBLSYSREP=c:¥usr¥user¥sysreplib;d:¥users¥lib¥rep
```

(18) CBLTAB

COBOL 原始プログラム中のタブコードを空白に変換するときの，空白の数を設定します。指定できる範囲は 1～72 で，省略時は 8 が仮定されます。

（例）

```
CBLTAB=4
```

(19) CBLUNINITDATA_EXITPROGRAM

初期化漏れチェック機能使用時，「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムの EXIT PROGRAM 文を，GOBACK 文とみなすかどうかを指定します。

この環境変数に GOBACK を指定すると，「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムの EXIT PROGRAM 文が，GOBACK 文とみなされます。

「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムの詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「初期化漏れチェック機能」を参照してください。

次の場合は，この環境変数の指定は無効となり，「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムの EXIT PROGRAM 文が，CONTINUE 文とみなされます。

- この環境変数の指定がない，または環境変数の値に GOBACK 以外を指定している
- -CheckUninitData オプションを指定していない

（例）

```
CBLUNINITDATA_EXITPROGRAM=GOBACK
```

注意事項

この環境変数の指定を有効にした場合，初期化漏れチェック機能使用時，「COBOL 主プログラムとも COBOL 副プログラムとも判断できる」プログラムの EXIT PROGRAM 文を，GOBACK 文とみなしてチェックします。このため，プログラム実行時に COBOL 主プログラムとして動作するプログラムであっても，EXIT PROGRAM 文を GOBACK 文とみなしてチェックします。該当するプログラムの EXIT PROGRAM 文を，CONTINUE 文とみなしたい場合，次のどちらかの方法で対応してください。

- 該当するプログラムが，COBOL ソースファイル内の，先頭の最外側のプログラムである場合，-Main オプションを明示的に指定して初期化漏れチェック機能でチェックしてください。

  （例）

  ・コンパイルする際のコマンドライン：「ccbl2002 main.cbl」

  main.cbl に-Main,System が仮定されます。

  ・初期化漏れチェック機能でチェックする際のコマンドライン：「ccbl2002 -Main,System main.cbl -Compile,CheckOnly -CheckUninitData」

  -Main,System を明示的に指定します。
- 該当するプログラムが，COBOL ソースファイル内の，二つ目以降の最外側のプログラムである場合，該当するプログラムを別の COBOL ソースファイルに分離します。分離した COBOL ソースファイルは，この環境変数の指定なしで，初期化漏れチェック機能でチェックしてください。

(20)　CBLUNINITDATA_OUTRESULTLIST

初期化漏れチェック機能使用時，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧を出力するかどうかを指定します。

この環境変数に ON を指定すると，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧が出力されます。初期化漏れ確認結果一覧および初期化漏れ従属項目一覧の詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「初期化漏れチェック機能」を参照してください。

次の場合は，この環境変数の指定は無効となり，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧が出力されません。

- この環境変数の指定がない，または環境変数の値に ON 以外を指定している
- -CheckUninitData オプションを指定していない

（例）

```
CBLUNINITDATA_OUTRESULTLIST=ON
```

(21) CBLUNINITDATA_OUTRESULTLISTDIR

初期化漏れチェック機能使用時，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧を任意のフォルダに出力したい場合に，その出力先フォルダを 255 バイト以内の絶対パス名で指定します。この環境変数の指定がない場合，初期化漏れ確認結果一覧および初期化漏れ従属項目一覧の出力先フォルダには，カレントフォルダが仮定されます。

注意事項

- 出力先フォルダは絶対パスで指定してください。絶対パスで指定していない場合の動作は保証しません。
- この環境変数に指定する値の長さは 255 バイト以内でなければなりません。255 バイトを超える値を指定した場合，255 バイトまでの値を有効とします。
- 次の場合は，この環境変数の指定は無効となります。
  - 環境変数 CBLUNINITDATA_OUTRESULTLIST に ON を指定していない
  - -CheckUninitData オプションを指定していない

（例）

```
CBLUNINITDATA_OUTRESULTLISTDIR=C:¥Work
```

(22) CBLVALUE

サブスキーマ節，作業場所節，画面節（WINDOW SECTION／SCREEN SECTION），報告書節，ファイル節で定義しているデータ項目の VALUE 句の指定のない初期値を，このシステムで採用している計算機文字集合である JIS8 単位コードの順序位置（1～256）で指定します。JIS8 単位コードについては，マニュアル「COBOL2002　言語　標準仕様編」の「付録 B 計算機文字集合」を参照してください。

実行単位でプログラムが最初に呼び出された場合だけ，この環境変数で指定した初期値をデータ項目に設定します。

この環境変数は，-CBLVALUE オプションを指定した場合だけ有効です。また，-CBLVALUE オプションを指定している場合で，この環境変数の指定がないとき，および指定した値に誤りがあるときは，1 が仮定されます。

（例）

```
CBLVALUE=33
```

VALUE 句のないデータ項目の初期値に，空白文字（JIS8 単位コードで 33 番目の文字）を指定します。

なお，この環境変数は次のデータ項目には適用されません。

- VALUE 句の指定のあるデータ項目
- ADDRESSED 句の指定のあるデータ項目
- EXTERNAL 句の指定のあるデータ項目
- DYNAMIC LENGTH 句のあるデータ項目

注意事項

初期化属性プログラムの作業場所節に対する初期化は，環境変数 CBLVALUE と環境変数 CBLINITVALUE が両方とも有効な場合と，どちらか片方だけが有効な場合とで初期値が異なります。詳細については，マニュアル「COBOL2002　ユーザーズガイド」の「プログラム属性」の「初期化属性プログラム」を参照してください。また，環境変数 CBLINITVALUE については，「(9) CBLINITVALUE」を参照してください。

(23)　CBLV3UNICODE

-UniObjGen オプション指定時，-CompatiV3 オプションおよび-V3Rec オプションの指定を有効としたい場合に設定します。

ただし，この環境変数を設定した場合，-CompatiV3 オプションは-JPN,Alnum オプションを仮定しません。

（例）

```
CBLV3UNICODE=YES
```

なお，この環境変数は次の場合は無効となります。

- この環境変数に YES を指定していない場合
- -UniObjGen オプションを指定していない場合

この環境変数とコンパイラオプションの組み合わせによるメインフレーム（VOS3）COBOL85 移行用オプションである，-CompatiV3 オプション，-JPN オプション，および-V3Rec オプションの有効／無効の関係を次に示します。

表 2-22　環境変数 CBLV3UNICODE と移行用オプションの組み合わせ

環境変数 CBLV3UNICODE	-UniObjGen オプションと同時に指定するオプション		
	-CompatiV3	-JPN	-V3Rec
YES	○※	×	○
YES 以外	×	×	×

（凡例）

○：有効となる

×：無効となる

注※

-V3Rec,Variable オプションは仮定されますが，-JPN,Alnum オプションは仮定されません。

注意事項

- -JPN,Alnum オプションが仮定されないことで，日本語項目，日本語編集項目，および日本語文字定数をそれぞれ英数字項目，英数字編集項目，および英数字定数として扱うことができないため，日本語項目と英数字項目間での比較および転記ができません。また，VOS3 COBOL85 と日本語機能に差異があるため，日本語項目または英数字項目のどちらか一方に合わせる必要があります。VOS3 COBOL85 と COBOL2002 との日本語機能の差異を次に示します。

表 2-23　VOS3 COBOL85 と COBOL2002 との日本語機能の差異

項目	VOS3 COBOL85	COBOL2002※
日本語項目と英数字項目間の転記／比較	LANGOPT=(-D)の場合，転記または比較できる。	転記または比較できない。
INITIALIZE 文（日本語項目の初期設定）	• REPLACING 指定がないと半角空白で初期化する。 • REPLACING ALPHANUMERIC, ALPHANUMERIC-EDITED 指定時，一意名 2，定数 1 の値で初期化する。	• REPLACING 指定がないと全角空白で初期化する。 • NATIONAL,NATIONAL-EDITED 以外の REPLACING 指定の場合，初期化されない。
INSPECT，STRING，UNSTRING 文の日本語項目	XCOBOL=(N)オプションのときは 1 バイト単位で処理する。	日本語文字単位で処理する。
日本語項目パディング	LANGOPT=(-D)の場合，パディング文字に半角空白(X'40')を使用する。	パディング文字に全角空白(X'3000')を使用する。

注※

環境変数 CBLV3UNICODE=YES，-UniObjGen オプションおよび-CompatiV3 オプション指定あり

- -V3Rec,Variable オプションが指定できることで，日本語文字定数に標準コード文字が指定できます。日本語文字定数が UTF-16 となり，標準コード文字も 2 バイトとなります。このことから，転記先のけた数を増やす必要があり，その日本語項目または日本語編集項目を参照している個所も修正が必要となる場合があります。例を次に示します。

（転記先のけた数の見直しの例）

01 ABC PIC N(2).

01 DEF PIC N(1).

：

MOVE N'あ AB' TO ABC.

MOVE ABC(2:1) TO DEF.

日本語文字定数は 3 けた（6 バイト）になることから，転記先の ABC けた数を 3 けたに修正する必要があります。

また，標準コード文字が 2 バイトになることから，DEF に'AB'が格納されていることを期待する場合，標準コード文字'AB'を参照する部分参照の長さを 1 から 2 に修正する必要があり，その転記先となる DEF のけた数も 1 から 2 に修正する必要があります。

（日本語文字定数の半角空白補完による転記先のけた数の見直しの例）

01 ABC PIC N(3).

：

MOVE N'あ abc' TO ABC.

日本語文字定数の標準コード文字数が奇数になる場合は，標準コード文字数が偶数になるように日本語文字定数の終端に半角空白が補完されます。これによって，補完された半角空白を含み，日本語文字定数は 5 けた（10 バイト）になることから，転記先となる ABC のけた数を 5 けたに修正する必要があります。

- -CompatiV3 オプションが指定できることによって，報告書作成機能が拡張され，特殊名段落の定数 10 が指定可能となりますが，1 バイト文字の指定が前提であるため，UTF-8 で多バイトとなる半角かたかなは指定できません。このことから，定数 10 に UTF-8 で多バイトとなる文字を指定している場合，UTF-8 で 1 バイトとなる文字に変更する必要があります。

(24)　登録集環境変数

登録集原文を検索するフォルダを設定します。フォルダを複数指定する場合は，セミコロン（;）で区切って指定します。

（例）

```
登録集名=¥usr¥user¥copylib
```

登録集名は，英大文字と数字から成る 8 文字以内の任意の文字列で指定します。

ここで指定したフォルダは，環境変数 CBLLIB で指定したフォルダよりも優先します。詳細については，マニュアル「COBOL2002　ユーザーズガイド」の原始文操作機能についての説明を参照してください。

2.5　リンカオプションの設定

開発マネージャのビルドを使用して COBOL プログラムをリンケージした場合，必要なリンカオプションが自動的に設定されます。ここでは，通常設定される以外のリンカオプションを特別に設定する方法について説明します。

2.5.1　リンカオプションの設定方法

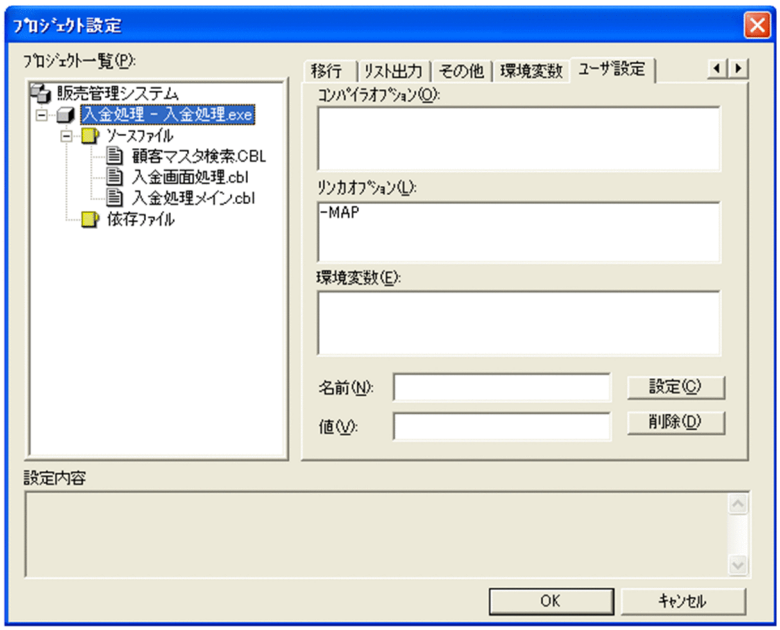
リンカオプションは，プロジェクト単位で設定します。リンカオプションを設定する方法を説明します。設定できるリンカオプションの詳細については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

リンクを行う場合，開発マネージャは ccbl2002 コマンドを呼び出し，その延長で LINK コマンドが呼ばれます。そのため，「リンカオプション」欄には，ccbl2002 コマンドのオプションも指定できます。また，プロジェクトの種類が「標準ライブラリ」の場合は，リンカオプションに指定したオプションは，LIB コマンドに渡ります。

1. 開発マネージャのウィンドウの［プロジェクト］メニューから［プロジェクトの設定］を選ぶ。
   プロジェクト設定ダイアログボックスが表示されます。

2. プロジェクト設定ダイアログボックスの「プロジェクト一覧」から，リンカオプションを設定するプロジェクトを選ぶ。

3. ［ユーザ設定］タブを選ぶ。



4. 「リンカオプション」の欄にリンカオプションを直接入力して［Enter］キー，または［Tab］キーを押す。
   リンカオプションが設定されます。

複数のリンカオプションを指定する場合は，空白文字で区切って指定してください。また，［Ctrl］＋［Enter］キーで改行を入力してオプションを区切ることもできます。

2.6　コンパイルの方法

コンパイルの方法には，ビルド，クイックビルド，リビルド，ファイル単位のコンパイルの四つの方法があります。また，ビルド，クイックビルド，リビルドは，すべてのプロジェクトに対しても実行できます。

2.6.1　ビルド

(1)　再コンパイル，再リンケージの規則

ビルドによってプログラムが再コンパイル，再リンケージされるときの規則について示します。

(a)　再コンパイルされる場合

プログラムが再コンパイルされるケースとその対象を次に示します。

表 2-24　ビルド時に再コンパイルされるケースとその対象

コンパイルの対象となるケース	再コンパイルの対象となるプログラム
プロジェクトを構成するファイルを更新／追加した場合	変更したファイル。
コンパイラオプションを変更した場合	• プロジェクトのコンパイラオプションを変更した場合は，オプションを変更したプロジェクトに対応するすべてのファイル。 • ファイルのコンパイラオプションを変更した場合は，コンパイラオプションを変更したファイル。
メインプログラムの指定を変更した場合	メインプログラム指定を解除したプログラム，および新しくメインプログラムの指定をしたプログラム。
• ソースファイルに対応する出力ファイルがない場合 • 出力ファイルを持つソースファイルまたは依存ファイルを更新した場合※	出力ファイルの生成元となるファイル。
前回のビルド処理で，リビルド中に中断した場合	ビルド処理が実行されなかったファイル。

注

ソースファイルがない場合は，メッセージウィンドウに警告のメッセージが出力され，そのままビルド処理が継続します。

注※

更新時間が取得できないため，更新されたかどうかわからないファイルは，必ず再コンパイルされます。

(b)　再リンケージされる場合

再リンケージされるのは，「2.6.1　ビルド」でプログラムが再コンパイルされる場合と次の場合です。

- 最終生成物がない場合
  （最終生成物を削除した場合，またはほかの場所に移動した場合）
- 最終生成物の名称を変更した場合

- オブジェクトファイルが，最終生成物よりも新しい場合※
- リンカオプションを変更した場合
- プロジェクトからファイルを削除した場合

注※

更新時間が取得できないため，更新されたかどうかわからないファイルは，必ず再リンケージされます。

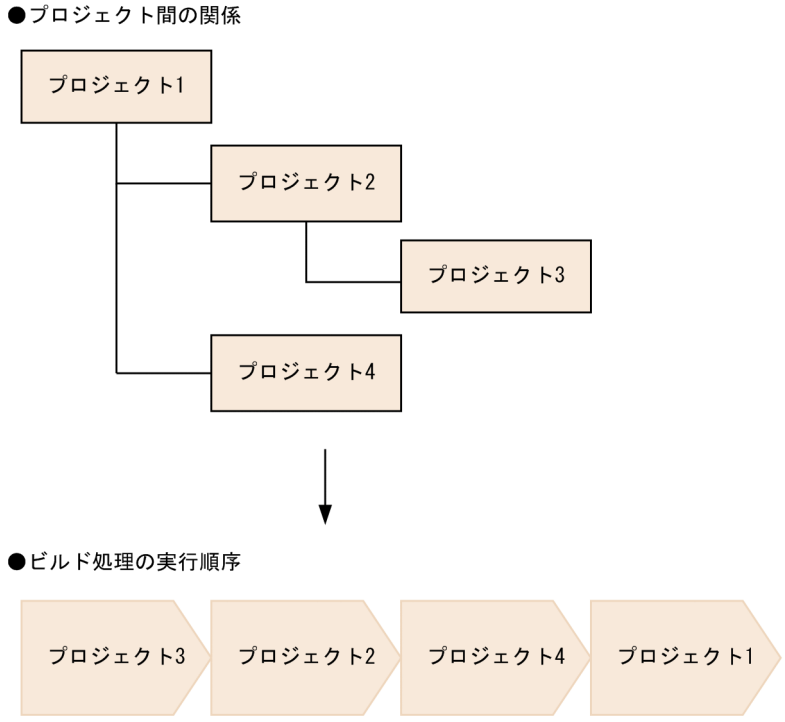
(c)　注意事項

- ファイル単位のコンパイルで作成したファイルをプロジェクトに登録した場合，ビルドをすると，結果が保証されないことがあります。このため，最初の 1 回はリビルドをしてください。
- 実行ファイルが実行されている状態でビルドをすると，実行ファイルに対する書き込みエラーになります。このような場合は，実行ファイルの実行が終了してからビルドをしてください。

(2)　ビルドの実行順序

開発マネージャでビルドを実行した場合，作業中プロジェクトがビルドされます。このとき，作業中プロジェクトに子プロジェクトがある場合は，最も深い階層の子プロジェクトから順にビルドが実行されます。

図 2-3　プロジェクト間の関係とビルドの実行順序



(3)　ビルドの方法

作業中プロジェクトのビルドを実行する方法

1. 開発マネージャのウィンドウの［ビルド］メニューから［ビルド］を選ぶ。

作業中プロジェクトのビルドが実行されます。

選んだプロジェクトのビルドを実行する方法

1. 開発マネージャのツリービューウィンドウで，ビルドを実行したいトップレベルプロジェクトを右クリックし，表示されるメニューから［選択中のプロジェクトのビルド］を選ぶ。

   選んだプロジェクトのビルドが実行されます。

ビルドを実行すると，再コンパイル，再リンケージの規則に従ってプログラムがコンパイル，リンケージされ，メッセージウィンドウにメッセージが表示されます。また，ビルドの対象に定義ファイルやHTMLファイルなど，出力ファイルを指定したファイルがある場合は，出力ファイルが生成されてからコンパイルされます。

(4) ビルド処理の中止

1. 開発マネージャのウィンドウの［ビルド］メニューから［ビルドの中止］を選ぶ。

   実行中のビルド処理が中止され，メッセージウィンドウにメッセージが表示されます。

(5) コンパイルエラーの修正

1. メッセージウィンドウから，修正したいファイル名を含む1行をダブルクリックする。

   「1.7.3 テキストエディタの設定」で設定したエディタが起動され，指定したファイルが開かれます。

補足説明

コンパイルエラーが発生した場合に，ビルドまたはリビルドを中断するか続行するかを設定できます。

開発マネージャのウィンドウの［ツール］メニューから［開発マネージャの動作設定］を選び，表示されたダイアログボックスで設定してください。

2.6.2 クイックビルド

(1) ビルドとクイックビルドの相違

クイックビルドとは，オブジェクト指向機能または利用者定義関数を使用するプロジェクトに有効となる生成手段です。オブジェクト指向機能または利用者定義関数を使用しないプロジェクトの場合は，ビルドと同じ処理になります。

ビルドでは，タイムスタンプのチェックによってオブジェクト指向機能または利用者定義関数を使用したプログラムのソース間の依存関係を必ず再構成します。これに対して，クイックビルドでは，既存の依存関係情報に従ってタイムスタンプをチェックします。したがって，他ファイルとの依存関係が変化しないようなオブジェクト指向機能または利用者定義関数を使用したプログラムのソースファイルの修正をした場合，ビルドの代わりにクイックビルドを使用すると処理時間を短縮できます。しかし，他ファイルとの依存関係が変化するようなソース修正をした場合，クイックビルドでは正しくビルドできません。

(2) クイックビルドの方法

1. 開発マネージャのウィンドウの［ビルド］メニューから［クイックビルド］を選ぶ。

   プロジェクトは，以前にビルドまたはリビルドをしたものでないと，ビルド処理と同じになります。

   既存の依存関係情報に従ってビルドされます。

   オブジェクト指向機能または利用者定義関数を使用しない COBOL プログラムだけで構成されているプロジェクトに対して実行した場合，普通のビルドと同じ処理になります。

(3) 依存関係の更新

1. 開発マネージャのウィンドウの［ビルド］メニューから［依存関係の更新］を選ぶ。

   オブジェクト指向機能または利用者定義関数を使用したプロジェクトのファイル間の依存関係が再構築されます。

   オブジェクト指向機能または利用者定義関数を使用したプロジェクトでは，依存関係を更新したあとにクイックビルドすると，ビルドと同じ処理になります。

   オブジェクト指向機能または利用者定義関数を使用しないプロジェクトに対して実行しても，何も処理されません。

2.6.3 リビルド

(1) リビルドの方法

1. 開発マネージャのウィンドウの［ビルド］メニューから［リビルド］を選ぶ。

   プロジェクトに属するすべてのプログラムがコンパイル，リンケージされ，メッセージウィンドウにメッセージが表示されます。また，プロジェクト中に定義ファイルや HTML ファイルなど，出力ファイルを指定したファイルがある場合は，出力ファイルが生成されてからコンパイルされます。

2.6.4 ファイル単位のコンパイル

(1) ファイル単位のコンパイルの方法

1. 開発マネージャのツリービューウィンドウからコンパイルしたいソースファイル名を選ぶ。

2. ［ビルド］メニューから［コンパイル］を選ぶ。

   指定したプログラムがコンパイルされ，メッセージウィンドウにメッセージが表示されます。定義ファイルや HTML ファイルなど，出力ファイルを指定したファイルをコンパイルした場合は，出力ファイルが生成されます。

2.6.5　すべてのプロジェクトに対する操作

通常のビルド処理は，選んだプロジェクトだけに対して実行されますが，すべてのプロジェクトを対象にビルド処理を実行することもできます。

すべてのプロジェクトに対して実行できるのは，ビルド，リビルド，クイックビルド，および依存関係の更新です。

1. **開発マネージャの［ビルド］メニューから［すべてのプロジェクトに対する操作］を選び，サブメニューから［ビルド］［リビルド］［クイックビルド］［依存関係の更新］のどれかを選ぶ。**

   選んだ処理がプロジェクトマスタ中のすべてのプロジェクトに対して実行されます。

注意事項

一つのプロジェクトが複数のプロジェクトに依存するようなプロジェクト間の依存関係を設定していた場合，ビルドが重複して実行されます。

3 実行

実行可能ファイルとして生成されたプログラムは，必要な実行環境を設定して実行します。

ここでは，プログラムの実行方法と実行環境の設定方法について説明します。

3.1　概要

プログラムの実行には，実行環境を設定する方法と設定しない方法があります。実行環境は，実行支援を使用して設定できます。

ここでは，実行方法の種類，および実行支援を使用してプログラムを実行するときに使用するファイルについて説明します。

3.1.1　実行方法の種類

プログラムの実行方法には，次の二つがあります。

- **実行環境を設定する方法**
  実行支援を使用して，環境変数や入出力ファイルなどを設定し，設定した環境に従ってプログラムを実行します。
- **実行環境を設定しない方法**
  既存の実行環境に従ってプログラムを実行するか，または実行環境を使用しないでそのまま実行します。

3.1.2　実行支援で使用するファイル

- **実行可能ファイル（.exe）**
  コンパイル・リンケージをして実行可能になったプログラムが格納されているファイルです。
- **プログラム別実行環境ファイル（.cbr）**
  プログラム別に設定した実行環境を格納するファイルです。このファイルは，実行可能ファイルと同じフォルダに，同じ名称（拡張子は除く）で保存する必要があります。
- **共通実行環境ファイル（.cbr）**
  システム中，すべてのプログラムで共通する実行環境を格納するファイルです。このファイル名称（拡張子は除く）は自由に指定できます。

3.2　実行の開始と終了

3.2.1　実行の開始

(1)　実行環境を設定する場合

実行環境を設定するには，実行支援を起動します。

- 開発マネージャのウィンドウの［ビルド］メニューから［実行支援］を選ぶ。

  作業中プロジェクトに対する実行支援ウィンドウが表示されます。

  作業中プロジェクトの最終生成物にダイナミックリンクライブラリ，標準ライブラリが指定されている場合，またはプロジェクトの設定の［リンク］タブで，「リンケージ処理を行わない」が指定されている場合は，実行支援ウィンドウのタイトルバーには「無題」が表示されます。リンケージ処理をしないオプションについては「2.3.10　リンクの設定」の「(5)　リンケージを行わない」を参照してください。

- Windows のプログラム一覧にある［COBOL2002］下のメニューから［実行支援 for COBOL2002］を選ぶ。

(2)　プログラムを実行する場合

- 開発マネージャのウィンドウの［ビルド］メニューから［実行］を選ぶ。

  実行ダイアログボックスが表示されます。



実行可能ファイル名

実行するファイル名を指定します。

デフォルトでは，作業中プロジェクトの出力ファイル名称が絶対パスで表示されます。

作業中プロジェクトの最終生成物にダイナミックリンクライブラリ，標準ライブラリが指定されている場合，またはプロジェクトの設定の［リンク］タブで，「リンケージ処理を行わない」が指定されている場合は，実行可能ファイル名は表示されません。リンケージ処理をしないオプションについては「2.3.10　リンクの設定」の「(5)　リンケージを行わない」を参照してください。

作業フォルダ

プログラムを実行するフォルダを指定します。デフォルトでは，前回実行ダイアログボックスで指定した内容が表示されます。ただし，作業中プロジェクトが変更された場合は，初期化されます。

作業フォルダを指定しなかった場合は，実行可能ファイル名のカレントフォルダが仮定されます。

引数

プログラムの引数を指定します。デフォルトでは，前回実行ダイアログボックスで指定した内容が表示されます。ただし，作業中プロジェクトが変更された場合は，初期化されます。

作業フォルダ，実行可能ファイル名，および引数は，以前に指定した内容をドロップダウンリストから選ぶこともできます。なお，ドロップダウンリストの内容は，プロジェクトマスタを閉じると初期化されます。

また，開発マネージャのツリービューウィンドウでプロジェクトを選び，右クリックで表示されるメニューから［実行］を選んでも，実行ダイアログボックスを表示できます。この場合，「実行可能ファイル名」には，選んだプロジェクトに対応するプログラム名が仮定されます。

- **あらかじめアイコン化しておいた実行可能ファイルをダブルクリックする。**
- **エクスプローラから実行可能ファイルをダブルクリックする。**

  指定した実行可能ファイルと同じ名称の実行環境ファイルが，同じフォルダにあれば，その実行環境に従ってプログラムが実行されます。同じ名称の実行環境ファイルがなければ，実行環境なしで実行されます。

3.2.2 実行の終了

実行支援を使用してプログラムを実行した場合の終了方法を説明します。

1. **実行支援ウィンドウの［ファイル］メニューから［終了］を選ぶ。**

   設定した環境情報を保存していない場合には，ファイルを保存するかどうかを問い合わせるダイアログボックスが表示されるので，ファイルを保存してください。

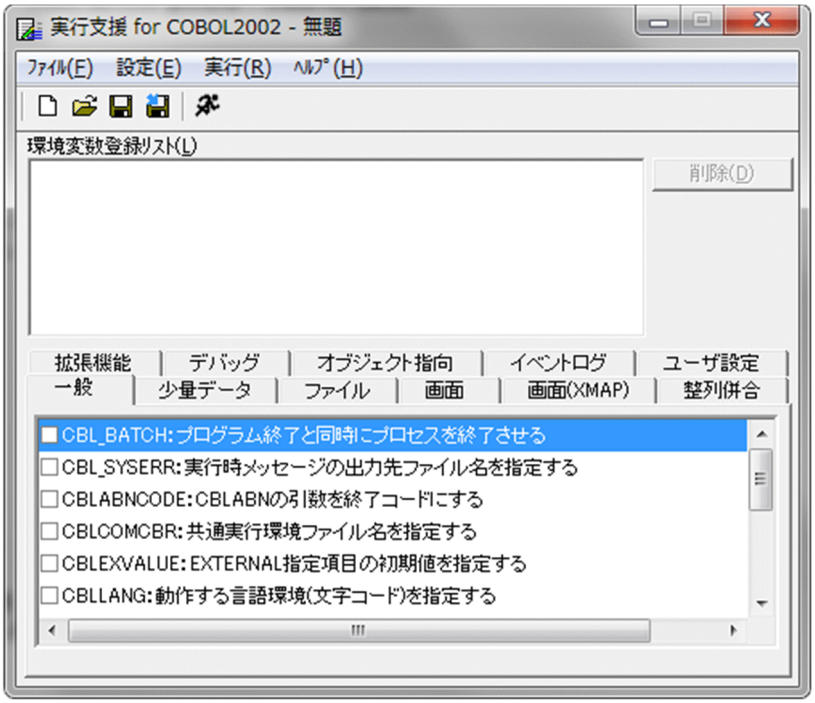
3.3　実行支援ウィンドウでの操作

実行支援ウィンドウで実行環境を設定する操作，および設定した実行環境に従ってプログラムを実行する操作について説明します。

3.3.1　実行支援ウィンドウの構成

実行支援ウィンドウを次に示します。

図 3–1　実行支援ウィンドウ



(1)　メニューバー

実行支援ウィンドウのメニューの種類と機能を次に示します。

表 3–1　実行支援ウィンドウのメニュー一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	新しく実行環境ファイルを作成する。
	開く(O)	既存の実行環境ファイルを開く。
	上書き保存(S)	設定した環境情報を作業中の実行環境ファイルに保存する。
	名前を付けて保存(A)	設定した環境情報を実行環境ファイルの名前を付けて保存する。
	終了(X)	実行支援ウィンドウを閉じる。

メニューバー	メニュー項目	機能
設定(E)	画面環境(S)	画面機能の実行環境を設定する。
	印刷書式(P)	印刷時の書式や出力先のプリンタを設定する。 [デフォルト(D)]：デフォルトとして使用する印刷書式を設定する。 [システム(S)]：システムで使用する印刷書式を設定する。
	オプション(O)	実行環境ファイルで設定した画面環境，および印刷書式（デフォルト）を，実行時に無効にする。
実行(R)	現在の内容で実行(R)	実行環境設定中の実行可能ファイルを実行する。
	他の実行可能ファイル(P)	実行環境設定中の実行可能ファイル以外の実行可能ファイルを実行する。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

(2) ツールバー

ツールバーのボタンを選ぶとメニューと同じ操作ができます。ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [ファイル] メニューの [新規作成]
2. [ファイル] メニューの [開く]
3. [ファイル] メニューの [上書き保存]
4. [ファイル] メニューの [名前を付けて保存]
5. [実行] メニューの [現在の内容で実行]

3.3.2 実行環境の設定

(1) 環境変数の登録

環境変数の登録・変更・削除の操作，および設定できる環境変数について説明します。

環境変数は，次の形式で環境変数登録リストに表示されます。

形式

環境変数名=値

値には，メタキャラクタを使用した「%環境変数名%」の形式が指定できます。詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「実行時環境変数の設定方法」を参照してください。

(a)　登録

1. 実行支援ウィンドウの下部にあるタブから，設定する環境変数の種類を選ぶ。

   選んだ種類に関連する環境変数の説明の一覧が表示されます。

2. 設定する環境変数の説明のチェックボックスを選ぶ。

   ここから先の操作は，環境変数ごとに異なります。

指定できる値が 1 種類だけの環境変数の場合

「環境変数名=YES」のように，指定できる値が 1 種類だけの環境変数の場合，自動的に環境変数に値が設定されます。

変数名に任意の情報を含む環境変数の場合

CBL_外部装置名の「外部装置名」のように，変数名に任意の情報を含む環境変数の場合，環境変数の設定ダイアログボックスが表示されます。



「変数名」の欄に「CBL_」まで入力されているので，外部装置名の部分を入力し，「値」の欄に値を設定して［登録］ボタンを選んでください。

なお，次に示す操作によって，環境変数に物理ファイルを割り当てることもできます。

ファイルを割り当てる場合

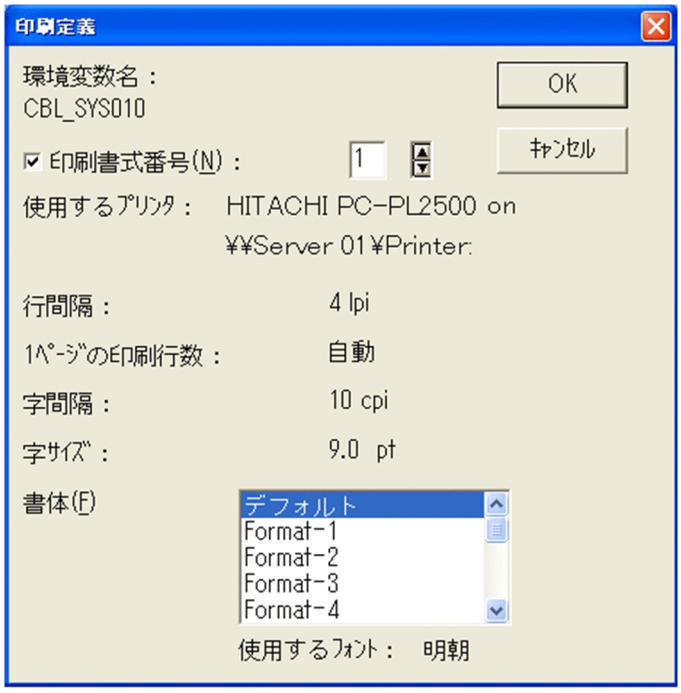
外部装置名を入力して［参照］ボタンを選び，表示されたダイアログボックスからファイルを指定します。指定したファイル名が値に設定されます。

環境変数によっては，複数のファイル名を指定できるものもあります。

GDI モード印刷機能で出力したいプリンタを割り当てる場合

外部装置名を入力して［印刷定義］ボタンを選びます。

次に示す印刷定義ダイアログボックスが表示されます。



印刷書式番号を指定します。各印刷書式番号には，あらかじめ出力先や出力形式が設定されています（「3.3.4　印刷書式の設定」参照）。チェックボックスがチェックされていない場合は，デフォルトが使用されます。

ESC／P モード印刷機能で出力したいプリンタを割り当てる場合

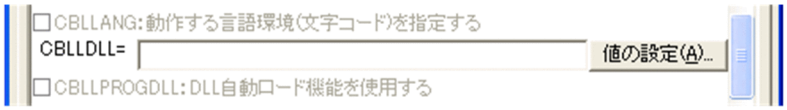
「値」の欄に"SYSPRT"を指定します。

GDI モード印刷機能，および ESC／P モード印刷機能での出力の詳細については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

環境変数の設定ダイアログボックスの登録リストに環境変数を設定したら，［OK］ボタンを選んで環境変数の設定ダイアログボックスを閉じてください。なお，［キャンセル］ボタンを選ぶと，環境変数の登録を中止できます。

上記以外の環境変数の場合

値を設定する欄が表示されます。設定欄の例を次に示します。



値をテキストボックスに直接入力するか，ドロップダウンリスト，スピンボタン，［参照］ボタン，［設定値選択］ボタン，［値の設定］ボタンなどで選んで［Enter］キーを押してください。

なお，値をテキストボックスに直接入力中に登録を中止する場合は，［Esc］キーを押してください。また，ほかのウィンドウをアクティブにした場合も入力が中止されます。

環境変数が登録されると，環境変数の説明のチェックボックスがオンになり，「環境変数登録リスト」の欄に設定内容が表示されます。

注意事項

- ユーザ指定の環境変数を登録する場合は，［ユーザ設定］タブを選んで「変数名」「値」の欄にそれぞれ設定したい環境変数名と値を直接入力し，［登録］ボタンを選びます。
- 「環境変数名=」のように値のない環境変数を登録する場合は，［ユーザ設定］タブを選んで「変数名」の欄に環境変数名だけを入力し，［登録］ボタンを選びます。
- 各環境変数に設定する値の詳細については，マニュアル「COBOL2002　ユーザーズガイド」の「プログラムの実行環境の設定」についての説明を参照してください。

(b)　変更

1. 実行支援ウィンドウの下部にあるタブから，設定する環境変数の種類を選ぶ。

   選んだ種類に関連する環境変数の説明の一覧が表示されます。

2. 設定する環境変数の説明のチェックボックスを選ぶ。

   ここから先の操作は，環境変数ごとに異なります。

変数名に任意の情報を含む環境変数の場合

環境変数の設定ダイアログボックスが表示されます。次の手順で環境変数の値を変更してください。

1. 変更したい環境変数を登録リストからダブルクリックする。

   設定欄に現在の値が表示されます

2. 値を変更して，［登録］ボタンを選ぶ。

   環境変数の変更が登録リストに反映されます。

3. ［OK］ボタンを選ぶ。

   環境変数の設定ダイアログボックスが閉じ，環境変数の値が変更されます。

   なお，「キャンセル」ボタンを選ぶと，環境変数の変更を中止できます。

上記以外の環境変数の場合

選んだ環境変数の現在の設定値が表示されます。値を変更して［Enter］キーを押すと，環境変数が変更されます。

なお，値をテキストボックスに直接入力中に変更を中止する場合は，［Esc］キーを押してください。

(c)　削除

1. 実行支援ウィンドウの下部にあるタブから，環境変数の種類を選ぶ。

   選んだ種類の環境変数の一覧が表示されます。

2. 変更する環境変数名のチェックボックスを選ぶ。

   ここから先の操作は，環境変数ごとに異なります。

指定できる値が 1 種類だけの環境変数の場合

環境変数が削除され，チェックボックスがオフになります。

変数名に任意の情報を含む環境変数の場合

環境変数の設定ダイアログボックスが表示されます。次の手順で環境変数を削除してください。

1. 削除したい環境変数を登録リストから選ぶ。
2. [削除] ボタンを選ぶ。

   登録リストから環境変数が削除されます。
3. [OK] ボタンを選ぶ。

   環境変数の設定ダイアログボックスが閉じます。環境変数の説明のチェックボックスがオフになり，環境変数の値が削除されます。

   なお，［キャンセル］ボタンを選ぶと，環境変数の削除を中止できます。

上記以外の環境変数の場合

値を設定する欄が表示されます。値をすべて削除して空欄の状態で［Enter］キーを押すと，チェックボックスがオフになり，環境変数登録リストから指定した環境変数が削除されます。

また，次の方法でも環境変数の設定を削除できます。

1. 実行支援ウィンドウの環境変数登録リストから削除したい環境変数を選ぶ。
2. [削除] ボタンを選ぶ。

   環境変数が削除されます。

(2) 環境変数一覧

COBOL2002 の環境変数の一覧を次に示します。環境変数の詳細な内容，および指定の詳細については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

実行時環境変数の一覧を，次に示します。

(a) 一般

表 3-2　環境変数一覧（一般）

環境変数名	指定する内容
CBLABNCODE	CBLABN サービスルーチンの引数を COBOL2002 アプリケーションの終了コードにするかどうか
CBLCOMCBR	システム共通の実行環境でプログラムを実行するときの実行環境ファイル名
CBLEXVALUE	EXTERNAL 句の指定のあるデータ項目の初期値を指定
CBLLANG	動作する言語環境（文字コード）を指定
CBLLDLL	動的なリンクで実行時にダイナミックリンクする DLL の名称
CBLLPROGDLL	DLL 自動ロード機能を使うかどうか
CBLPGMSEARCHTRC	プログラム検索トレースファイル名を指定

環境変数名	指定する内容
CBLPGMSEARCHTRC_SIZE	プログラム検索トレースファイル名を切り替えるサイズを指定
CBLPRELOAD	プレロードリストファイル名を指定
CBLUNIENDIAN	用途が NATIONAL の項目に対する Unicode のバイトオーダを指定
CBLUPSI	外部スイッチの状態
CBL_BATCH	COBOL2002 アプリケーションの終了と同時にプロセスを終了させるかどうか
CBL_SYSERR	実行時エラーメッセージの出力先ファイル名

(b) 少量データ

表 3-3 環境変数一覧（少量データ）

環境変数名	指定する内容
CBLDATE	ACCEPT 文，CURRENT-DATE 関数，MOVE 文（日付と時刻用）でシステムから受け取る西暦年月日
CBLDAY	ACCEPT 文でシステムから受け取る通算日付
CBL_STOPNOADV	CUI モードで STOP　定数文を実行したとき，メッセージ ID と定数 1 の直後の改行文字を出力するかどうか
CBL_SYSIN	FROM SYSIN 指定の ACCEPT 文での入力ファイル名
CBL_SYSOUT	UPON SYSOUT 指定の DISPLAY 文での出力ファイル名
CBL_SYSPUNCH	UPON SYSPUNCH 指定の DISPLAY 文での出力ファイル名
CBL_SYSSTD	FROM SYSSTD 指定の ACCEPT 文での入力ファイル名

(c) ファイル

表 3-4 環境変数一覧（ファイル）

環境変数名	指定する内容
CBLCSVCHAR	-NumCsv オプションを指定して CSV 編成ファイルを数値として読み込むとき，無視する文字列
CBLCSVINIT	CSV 編成ファイルの READ 文実行時に，セルと対応しない未使用の基本項目を初期化するかどうか
CBLD_ファイル名	ファイル単位に入出力を指示するオプション
CBLEUDCFUNC	外字を有効にするため内部的に発行している EnableEUDC 関数の扱いの指定
CBLFSYNC	ファイルクローズ時のディスク書き込み保証を適用するかどうか
CBLF_ファイル名	印刷機能を利用したとき，スプールに登録される印刷文書名称の指定

環境変数名	指定する内容
CBLGDIINTERVAL	GDI モード印刷で INTERVAL 指定がないとき，実行支援の印刷書式に設定した字間隔を使用するかどうか
CBLGDIWMSG	GDI モード印刷機能を使用する場合に，警告メッセージの出力を抑止するかどうか
CBLIOMESSAGE	ファイル入出力文でのエラー情報出力機能を使用するかどうか
CBLISAMDL	既存の索引ファイルに対して OPEN OUTPUT を実行したとき，旧ファイルを削除後，新規に作成するかどうか
CBLISAMLARGE※	実行単位中のすべての ISAM による索引編成ファイルに対してラージファイル形式を適用するかどうか
CBLPRTEXCHR	CBLP_ファイル名を設定して ESC／P モード印刷機能を利用するプリンタへ外字を出力するとき，ベンダ定義文字を外字として出力する
CBLP_ファイル名	ESC／P モード印刷機能を利用するプリンタへ，Windows 上で作成した外字を登録する
CBLRDBDATAERR	HiRDB による索引編成ファイルで，レコード中の保証されないデータをエラーとして検出するかどうか
CBLRDBILWAIT	HiRDB による索引編成ファイルで，内部的に発行される SELECT に対して排他オプションを付けるかどうか
CBLRDBOPURGE	HiRDB による索引編成ファイルの全データを削除する際に，PURGE TABLE を使用するかどうか
CBLRDBROWVALCONSTRUCTOR	HiRDB による索引編成ファイルで，内部発行 SELECT で行値構成子を使用する「内部発行される SQL 文で行値構成子を使用する機能」を使用するかどうか
CBLTEXTSUPPRESSBOM	テキスト編成ファイルでの Unicode シグニチャ出力を切り替えるかどうか
CBLTEXTWRITESPACE	テキスト編成ファイルの WRITE 文，REWRITE 文で，レコード末尾の半角空白文字をファイルに書き出すかどうか
CBLX_外部装置名※	書式印刷をするときの印刷サービス名称
CBL_BTRPGSZ※	Btrieve（Pervasive.SQL）による索引編成ファイルのページサイズ
CBL_RDBCOMMIT	HiRDB による索引編成ファイルのトランザクション管理をするかどうか
CBL_RECLOCKMAX	相対編成ファイルで LOCK MODE 句に MANUAL を指定したとき同時に施錠できるレコード数
CBL_SYSCSVIN	FROM SYSCSV 指定の ACCEPT 文での入力ファイル名
CBL_SYSCSVOUT	UPON SYSCSV 指定の DISPLAY 文での出力ファイル名
CBL_外部装置名	ファイル入出力での入出力ファイル名

注※

Windows(x86) COBOL2002 で有効です。

(d) 画面

表 3-5 環境変数一覧（画面）

環境変数名	指定する内容
CBLACTWIN	COBOL アプリケーションの出力するコンソールおよび画面機能のウィンドウを DISPLAY 文のメッセージ出力のタイミングでアクティブにするかどうか
CBLATTRIBUTE	画面節（WINDOW SECTION）の ERASE ATTRIBUTE 文で指定したフィールドに重ねて表示してあるけい線，および属性を消去するかどうか
CBLAUTOCLEAR	画面機能でのデータ入力時の動作変更
CBLENTERCHK	ACCEPT 文や REPLY 文実行時，画面節（WINDOW SECTION）の ENTER-CHECK 句指定項目がデータ未入力の場合，WINDOW-STATUS 特殊レジスタへの値設定
CBLFEP	-JPN,Alnum，または-JPN,V3JPN オプション指定時に，カーソル移動によって自動的に日本語入力に切り替えるかどうか
CBLIMEPOS	画面機能で，日本語をフィールドに直接入力するかどうか
CBLJCPOPENDKEY	JCPOPUP サービスルーチンの終了キーの拡張
CBLM7ENDKEY	MIOS7 の終了キーを，［Alt］＋［F1］～［Alt］＋［F8］から［Shift］＋［F1］～［Shift］＋［F8］に変更する
CBLNOCLOSE	COBOL プログラム実行中の強制中断を抑止するかどうか
CBLONLYNUM	環境変数 CBLAUTOCLEAR での画面動作の変更を，画面節（WINDOW SECTION）の数字項目，および数字編集項目だけに限定するかどうか
CBLOVERFLOW	画面節（WINDOW SECTION）入力時に，自動カーソル位置づけを抑止するかどうか，およびオーバフローをチェックするかどうか
CBLSETFIELD	画面節（WINDOW SECTION）の SET 文でフィールドの属性を変更するとき，それまで SET 文で変更していた属性を初期化するかどうか
CBLUNDERDOT	画面機能で，入力および入出力フィールドにピリオド（.）を表示するかどうか
CBLUPDOWNMOVE	画面機能（画面節（WINDOW SECTION））での ACCEPT 文，または REPLY 文実行時，［↑］キー，［↓］キーの動作を変更するかどうか

(e) 画面（XMAP）

表 3-6 環境変数一覧（画面（XMAP））

環境変数名	指定する内容
CBLPRNTID	画面機能での送信先プリンタに対する仮想端末名
CBLPRNT_xxx	画面機能で送信先がプリンタのときの仮想端末名
CBLTERMID	画面機能での送信先ディスプレイに対する仮想端末名
CBLTERMSHAR	複数プログラムでの仮想端末共有
CBLTERM_xxx	画面機能で送信先がディスプレイのときの仮想端末名

(f)　整列併合

表 3-7　環境変数一覧（整列併合）

環境変数名	指定する内容
CBLSORTSIZE	整列処理で使用するメモリサイズ
CBLSORTWORK	整列処理用の作業用ファイルのフォルダ名

(g)　拡張機能

表 3-8　環境変数一覧（拡張機能）

環境変数名	指定する内容
CBLOPS	MIOS7 COBOL85 互換のために複数の環境変数を指定する
CBLSQLCOMMOD	ODBC インタフェースを使用してデータベースに接続する場合のコミットモードの設定
CBLSQLCURUSE	ODBC インタフェース機能を使用した場合にカーソルオプションの設定を変更する
CBLSQLDYNAMIC	ODBC インタフェース機能の動的 SQL で内部的に発行する ODBC API を変更する
CBLSQLLOGINTIMEOUT	ODBC インタフェースを使用した場合に ODBC オプションの SQL_LOGIN_TIMEOUT の値を指定する
CBLSQLQUERYTIMEOUT	ODBC インタフェースを使用した場合に ODBC オプションの SQL_QUERY_TIMEOUT の値を指定する
CBLSQLROWCOUNT	ODBC インタフェースを使用した場合に ODBC インタフェースで影響行数が 0 の場合に SQLCODE に 100 を設定する
CBLSQLSUPPRESSMSG	ODBC インタフェースを使用した場合に ODBC インタフェースでの実行時メッセージの出力を抑止するかどうか
CBLSQLWMSG	ODBC インタフェースを使用した場合に警告メッセージの出力を抑止するかどうか

(h)　デバッグ

表 3-9　環境変数一覧（デバッグ）

環境変数名	指定する内容
CBLABNLST	異常終了時要約情報リストの出力先
CBLDDUMP	異常終了時のデータ領域ダンプの出力先
CBLDATADUMPFILE	CBLDATADUMP サービスルーチンによるデータ領域ダンプの出力先
CBLEXCEPT	プログラム実行中に例外が発生した時の動作
CBLPRMCHKW	-DebugCompati オプション指定時，またはテストデバッグ時のプログラム間整合性チェックを緩和するかどうか
CBLTDEXEC	プログラム実行時にテストデバッガを起動するかどうか
CBL_FLSRVDUMP	COBOL 入出力サービスルーチンのデバッグ情報を出力するファイル名

(i) イベントログ

表 3-10 環境変数一覧（イベントログ）

環境変数名	指定する内容
CBLSYSLOG	イベントログファイル出力機能を使用するかどうか
CBLSYSLOGLVL	イベントログファイル出力機能での出力対象イベントの種類
CBLSYSLOGSRV	イベントログファイル出力機能でのイベント出力先のコンピュータ名

(j) オブジェクト指向

表 3-11 環境変数一覧（オブジェクト指向）

環境変数名	指定する内容
CBLGCINTERVAL	前回のガーベジコレクションを終了してから，次のガーベジコレクションを開始するまでのメモリ使用量（ガーベジコレクションの実行間隔）
CBLGCSTART	ガーベジコレクタの開始条件となる，インスタンスオブジェクトの生成によるメモリ使用量の累積値

(3) 共通実行環境の設定

プログラム別実行環境ファイルに共通実行環境ファイルを取り込む環境変数の形式を次に示します。

```
CBLCOMCBR=共通実行環境ファイル名
```

- 共通実行環境ファイル名には，実行支援で作成した.cbr 形式のファイル名を絶対パスで指定します。
- 環境変数 CBLCOMCBR に指定できる共通実行環境ファイルは一つだけです。
- 環境変数 CBLCOMCBR に指定した実行環境ファイル中の，CBLCOMCBR 環境変数は無視します。
- 共通実行環境ファイルとして使用するファイル名は，拡張子が.CBR であれば，自由な名称を使用できます。
- プログラム実行時，共通実行環境ファイルは実行可能ファイルと同じフォルダにある必要はありません。
- 共通実行環境ファイルは実行時環境変数だけ有効となります。コンパイラ環境変数が指定されても無視します。

(4) 実行時環境変数の優先順位

実行時環境変数を複数の個所で指定した場合，それらの優先順位は次のようになります。

1. プログラム別実行環境ファイル
2. 共通実行環境ファイル
3. コマンドプロンプトで入力する SET コマンドや Windows システムで設定した環境変数

例えば，プログラム別実行環境ファイルで「CBLABNLST=C:¥ABNLST.TXT」を，共通実行環境ファイルで「CBLABNLST=C:¥USR¥ERRLST.TXT」をそれぞれ指定している場合，プログラム別実行環境ファイルで指定されている「CBLABNLST=C:¥ABNLST.TXT」が有効となります。

3.3.3　画面環境の設定

画面節（SCREEN SECTION および WINDOW SECTION）による画面機能実行時の文字フォントや，画面の色，および制御キーに対するキーの割り当てなどを設定できます。また，設定した画面環境を無効にできます。

(1)　画面環境を設定する

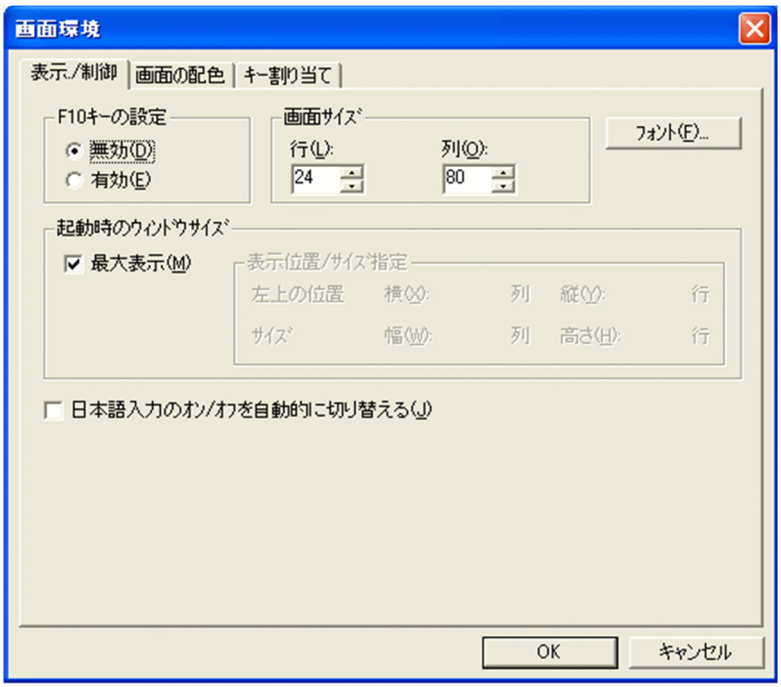
1. [設定] メニューから［画面環境］を選ぶ。

   画面環境を設定するダイアログボックスが表示されます。[表示／制御] タブ，[画面の配色] タブ，または［キー割り当て］タブを選んで，表示されるページに必要事項を指定します。

2. [OK] ボタンを選ぶ。

   次に，このダイアログボックスで設定する内容について説明します。

(a)　表示／制御の設定



F10 キーの設定

[F10] キーをシステムキーとして割り当てるかユーザキーとして割り当てるかを設定します。[F10] キーは，通常メニューバーのアクティブ化を切り替えるためのシステムキーとして割り当てられています。これを COBOL プログラム中で使用するためには，ユーザキーに割り当てる必要があります。

画面サイズ

表示画面のサイズを行数とけた数で設定します。設定範囲を次に示します。

24 ≦ 行数 ≦ 120

80 ≦ けた数 ≦ 160

デフォルトのサイズは 24 行，80 けたです。

フォント

［フォント］ボタンを選ぶと表示されるダイアログボックスで，画面に表示される文字のフォント（フォント名およびフォントサイズ）を設定します。また，ここでの設定内容は，コンソールウィンドウの文字フォントにも反映されます。なお，デフォルトのフォントは，フォント名が「MS　ゴシック」，フォントサイズが「12」です。

注意事項

- システムで表示できない，または日本語表示ができないフォントは設定できません。
- システムで表示できない，または日本語表示ができないフォントがすでに設定されている場合，画面環境を設定するダイアログボックスを開いたときに，デフォルトのフォントに置き換えてよいかを確認するダイアログボックスが表示されます。このダイアログボックスで「はい」を選ぶと，デフォルトのフォントに置き換えられます。

起動時のウィンドウサイズ

起動時のウィンドウを最大表示するか，位置・サイズを指定して表示するかを設定します。表示位置・サイズは画面に文字が表示されるときの行数と列数で設定します。設定範囲を次に示します。

左上の位置：

0 ≦ 縦 ≦ 画面サイズで指定した行数

0 ≦ 横 ≦ 画面サイズで指定した列数

サイズ：

1 ≦ 高さ ≦ 画面サイズで指定した行数

1 ≦ 幅 ≦ 画面サイズで指定した列数

デフォルトは最大表示です。

日本語入力のオン／オフを自動的に切り替える

日本語入力のオン／オフを自動的に切り替えるかどうかを設定します。自動的に切り替える場合の日本語入力（FEP）の状態については補足事項を参照してください。

デフォルトでは，自動的な切り替えを行いません。

補足事項

日本語入力のオン／オフを自動的に切り替える場合の，日本語入力（FEP）の状態を次に示します。

項目の種類	FEP
数字項目	オフ
英字項目	オフ

項目の種類	FEP
英数字項目	オフ
日本語項目	オン
数字編集項目	オフ
英数字編集項目	オフ
日本語編集項目	オン
サービスルーチンの入力	オフ

(b) 画面の配色の設定

COBOL2002 のターミナルウィンドウ（COBOL TERMINAL ウィンドウ），エラーメッセージウィンドウ，および JCPOPUP ウィンドウの配色を設定できます。

- COBOL TERMINAL ウィンドウ
  画面の背景色，デフォルトの文字の色（COBOL プログラム中で文字の色指定がないときの表示色），および罫線の色を設定できます。
- エラーメッセージウィンドウ※
  画面の背景色，および文字の色を設定できます。
- JCPOPUP ウィンドウ
  文字の色，および罫線の色を設定できます。

注※

エラーメッセージウィンドウとは，画面機能の実行時にけたあふれなどによってエラーが発生した場合に表示されるポップアップウィンドウのことです。

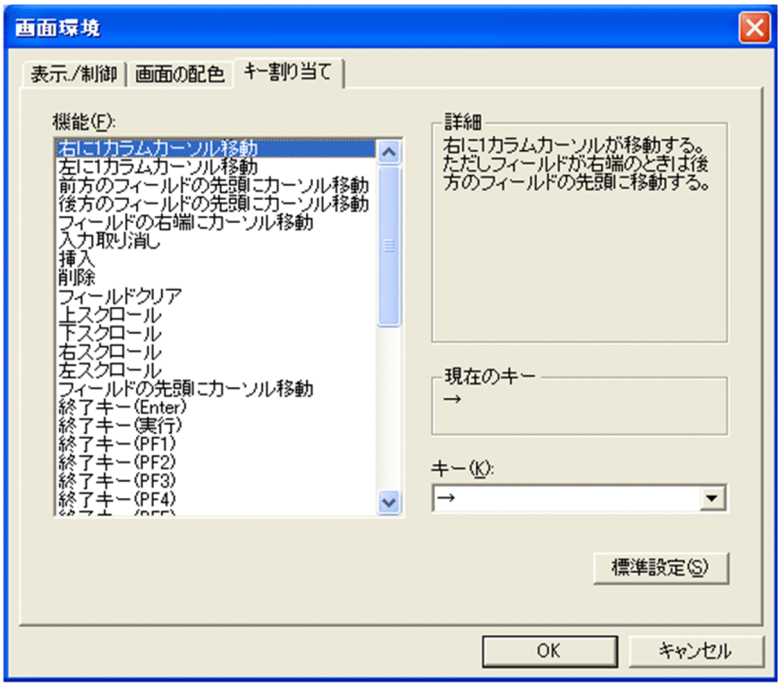
色の設定方法

色を設定するには，項目名横のボタンを選び，色の設定ダイアログボックスで設定します。任意の色を指定するには色の設定ダイアログボックスの「システムの設定」チェックボックスをオフにし，基本色，または作成した色から任意の色を選びます。

注意事項

Windows の画面のプロパティで設定しているカラーパレットの範囲外の色を選んだ場合，その色は正しく表示されません。

(c)　キー割り当ての設定



カーソル移動，挿入，削除などの機能に対して割り当てられたキーを変更できます。機能に対して割り当てられているデフォルトのキーについては，マニュアル「COBOL2002　ユーザーズガイド」の画面入出力機能の説明を参照してください。

キーの割り当てを変更するには，リストから機能を選び，さらに割り当てたいキーをリストから選択します。このとき，次の事項に注意してください。

- 割り当てるキーは重複してはなりません。
- ある機能に［Alt］キーを割り当て，さらに［Alt］＋［F1］キー（ほかのキーでも同様）を別の機能に割り当てた場合，まず［Alt］キーが割り当てられている機能が動作し，次に［Alt］＋［F1］キーが割り当てられている機能が動作します。［Shift］キー，［Ctrl］キーについても同様です。
- 割り当てたキーは，画面節（WINDOW SECTION），画面節（SCREEN SECTION）だけに有効となります。
- 終了キーをほかのキーに割り当てたあと，元のキーを CBLADTRM サービスルーチンで終了キーに追加すると，どちらのキーも終了キーとして有効になります。

  （例）

  次の 1.2.の両方の設定をした場合，［F1］キー，［Shift］＋［F1］キーのどちらも終了キーとして有効になります。

  1. 実行支援のキー割り当てで「終了キー（PF1）」に［Shift］＋［F1］キーを設定する。
  2. CBLADTRM サービスルーチンで［F1］キー（X'80D1'）を終了キーに追加する。

(2)　設定されている画面環境を無効にする

1. [設定] メニューから［オプション］を選ぶ。
   実行支援のオプションダイアログボックスが表示されます。

2. 「画面環境の設定」チェックボックスをオンにする。

3. [OK] ボタンを選ぶ。

なお，プログラム別実行環境ファイルおよび共通実行環境ファイルを利用している場合，それぞれのファイルに対して「画面環境の設定」チェックボックスのオンまたはオフの組み合わせによって有効になるファイルが異なります。「画面環境の設定」チェックボックスのオンまたはオフの組み合わせによって有効になる実行環境ファイルを次に示します。

プログラム別実行環境ファイル	共通実行環境ファイル	有効となる実行環境ファイル
オフ	オン	プログラム別実行環境ファイル
オン	オフ	共通実行環境ファイル
オン	オン	デフォルトの設定※
オフ	オフ	プログラム別実行環境ファイル

注※
画面環境ダイアログボックスで何も設定していない状態のことです。

3.3.4　印刷書式の設定

GDI モード印刷機能を利用してプリンタへ出力するときの印刷書式を設定します。設定できる印刷書式にはシステム／デフォルトがあります。システムとして設定した書式は，印刷書式番号で識別されます。また，デフォルトとして設定した印刷書式を無効にすることもできます。なお，実行環境ファイルを作成しない場合のデフォルトについては，「(2)　設定されている印刷書式を無効にする」の「**補足事項**」を参照してください。

注意事項

システム共通の印刷書式情報ファイル（.prt），および印刷書式を含む実行環境ファイル（.cbr）は，印刷書式の設定をした OS とマシン固有の情報を含むため，OS とマシンの互換はありません。

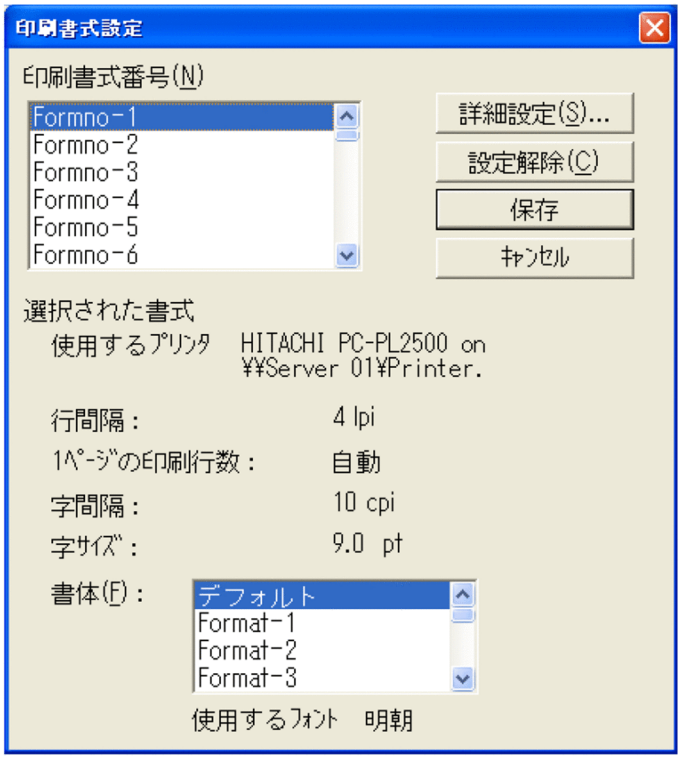
実行環境ファイル（.cbr）を別のマシンに複写した場合は，実行支援を起動し，印刷書式を再設定してください。

(1)　印刷書式を設定する

(a)　システムで使用する印刷書式の設定

1. [設定] メニューから［印刷書式］を選び，さらに［システム］を選ぶ。

次に示すダイアログボックスが表示されます。



2. 設定する印刷書式番号を選ぶ。

印刷書式設定ダイアログボックスに，選択された書式の情報が表示されます。

3. [詳細設定] ボタンを選ぶ。

印刷書式の詳細を設定するダイアログボックスが表示されます。[余白／行／文字] タブまたは [書体] タブを選んで，表示されるページに必要事項を指定します。

4. [OK] ボタンを選ぶ。

5. 印刷書式設定ダイアログボックスで [保存] ボタンを選ぶ。

なお，設定した印刷書式を解除するには，印刷書式設定ダイアログボックスで [設定解除] ボタンを選び，[保存] ボタンを選びます。

[保存] ボタンを選ぶことで設定した印刷書式，または解除した印刷書式の情報が，次のフォルダの印刷書式情報ファイル（.prt）※に格納されます。

- Windows(x86) COBOL2002 の場合
  環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002¥bin
- Windows(x64) COBOL2002 の場合
  環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002 64bit¥bin

システムが環境変数 ALLUSERSPROFILE で指定されたフォルダに作成するファイルは，デフォルトでは，それを作成したユーザでしか更新できません。

複数ユーザで更新したい場合は，必要に応じて，印刷書式情報ファイル（.prt）にアクセス権限を設定してください。または，実行支援を管理者権限に昇格後，起動し印刷書式の設定を行うことで印刷書式情報ファイル（.prt）を更新するなどの対処を行ってください。

注※

印刷書式情報ファイルは，使用しているシステム環境情報を含みます。このファイルは，アンインストール時に削除されません。COBOL の使用を取りやめる場合やシステムの変更時には，エクスプローラなどから削除してください。

次に，詳細設定ダイアログボックスで設定する内容について説明します。

(i) プリンタの設定

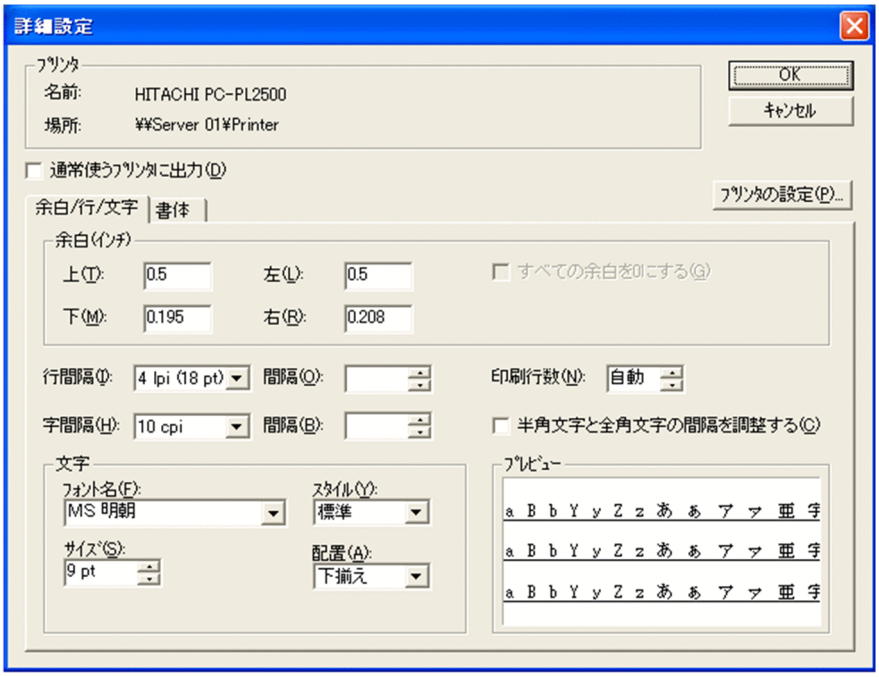
[プリンタの設定] ボタンを選ぶとプリンタの設定ダイアログボックスが表示され，使用するプリンタを選択できます。

また，「通常使うプリンタに出力」チェックボックスをオンにすると，選択したプリンタを無視して，Windows で設定した通常使うプリンタに印刷できます。

注意事項

- 実行支援からプリンタの設定ダイアログボックスでプリンタの情報を変更した場合，有効となるのは「用紙のサイズ」，「印刷の向き」，および［給紙方法］だけです。その他の情報はプリンタマネージャでの設定が有効となります。
- 「通常使うプリンタに出力」チェックボックスをオンにした場合，通常使うプリンタとプリンタの設定ダイアログボックスで設定したプリンタの，プリンタドライバが同じでなければなりません。プリンタドライバが異なる場合は，実行時にエラーとなります。また，プリンタドライバのバージョンが異なるために指定した「用紙のサイズ」や「印刷の向き」に対応できない場合，そのプリンタドライバのデフォルト値が仮定されて印刷されます。
- 使用するプリンタが選択されていない場合は，Windows で通常使うプリンタに設定したプリンタで印刷されます。

(ii) 余白／行／文字の設定



余白

ページの上下左右に余白を設定します。用紙の端からの長さ（単位=インチ）をそれぞれのテキストボックスに入力します。デフォルトを次に示します。

上：0.5 インチ

左：0.5 インチ

下：使用しているプリンタドライバの最小余白幅

右：使用しているプリンタドライバの最小余白幅

上下左右すべての余白に 0 インチを指定した場合は，印刷時の余白はデフォルトの値となります。

すべての余白を 0 にする

上下左右すべての余白を 0 インチにして印刷したいときは，このチェックボックスをオンにします。

なお，このチェックボックスは，上下左右すべての余白に 0 が指定できるプリンタを使用している場合だけ利用できます。

印刷行数

1 ページに印刷する行数を自動的に設定するか，または任意の行数を入力します。自動的に設定する場合は，現在選択されている行間隔と用紙サイズから印刷できる最大行数が設定されます。任意の行数を入力する場合は，印刷できる最大行数までを指定でき，印刷できる行数を超えた値を設定するとエラーメッセージが表示されます。なお，デフォルトは自動となります。

行間隔

ドロップダウンリストから行間隔を選択するか，またはユーザ設定を選んで任意の間隔（1～72 ポイント）を入力します。デフォルトは 4LPI です。

字間隔

印刷制御をしない場合，出力する行データの字間隔を設定できます。ドロップダウンリストから字間隔を選択するか，またはユーザ設定を選んで任意の間隔（1～72 ポイント）を入力します。デフォルトは 10CPI です。

また，半角文字と全角文字の間隔を調整することもできます。この項目をチェックすると，半角文字と全角文字の間隔が 1:2 に調整されます。

フォント名

ドロップダウンリストから使用するフォントを選択します。デフォルトはプリンタフォント※です。

スタイル

ドロップダウンリストから使用するスタイルを選択します。

サイズ

印刷制御をしない場合，出力する行データの字サイズを設定できます。字サイズは，任意のポイント数（現在選択されている行間隔によって指定できる範囲が異なる）を入力します。デフォルトは，現在選択されている行間隔によって 12，9，または 7 ポイントとなります。

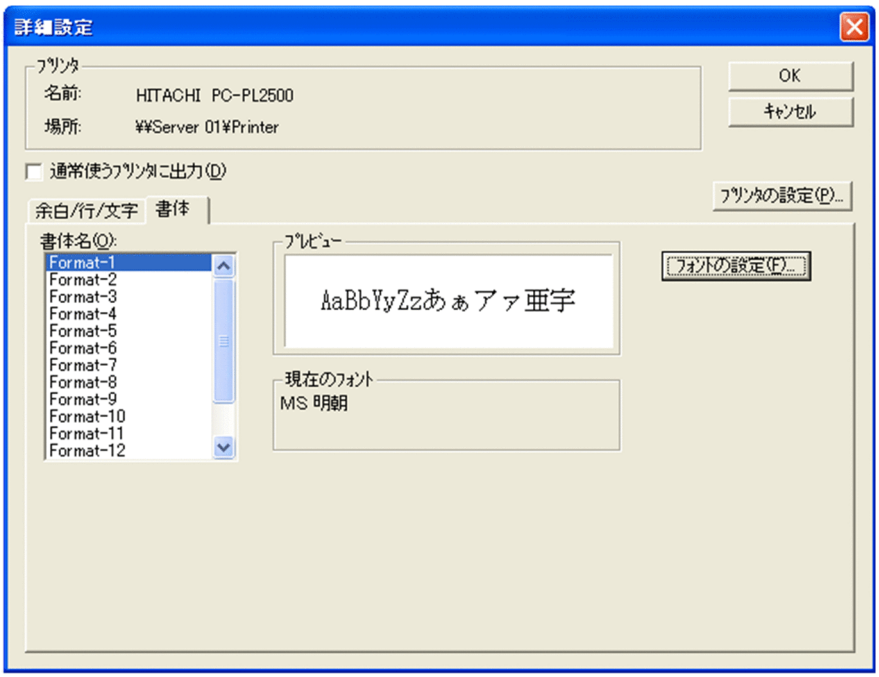
配置

ドロップダウンリストから文字の配置を選択します。デフォルトは下揃えです。

注※

プリンタフォント使用時，字間隔や字サイズは無効となることがあります。また，選択できるフォントは固定ピッチだけです。

(iii) 書体の設定



印刷制御で指定した書体にフォントを割り当てるには，リストから書体を選び，［フォントの設定］ボタンを選びます。表示されたダイアログボックスから割り当てたいフォントを選択します。なお，書体にフォントが割り当てられていない場合は，プリンタフォント※が使用されます。

注※

プリンタフォント使用時，字間隔や字サイズは無効となることがあります。また，選択できるフォントは固定ピッチだけです。

(b)　デフォルトとして使用する印刷書式の設定

1. [設定] メニューから［印刷書式］を選び，さらに［デフォルト］を選ぶ。

   表示された印刷書式の詳細を設定するダイアログボックスで，印刷書式の情報を設定します。この詳細設定ダイアログボックスで設定する内容については，「(1)　印刷書式を設定する」の「(a)　システムで使用する印刷書式の設定」の(i)～(iii)を参照してください。

2. [OK] ボタンを選ぶ。

(2)　設定されている印刷書式を無効にする

1. [設定] メニューから［オプション］を選ぶ。

   実行支援のオプションダイアログボックスが表示されます。

2. 「印刷書式（デフォルト）の設定」チェックボックスをオンにする。

3. [OK] ボタンを選ぶ。

なお，プログラム別実行環境ファイルおよび共通実行環境ファイルを利用している場合，それぞれのファイルに対して「印刷書式（デフォルト）の設定」チェックボックスのオンまたはオフの組み合わせによって有効になるファイルが異なります。「印刷書式（デフォルト）の設定」チェックボックスのオンまたはオフの組み合わせによって有効になる実行環境ファイルを次に示します。

プログラム別実行環境ファイル	共通実行環境ファイル	有効となる実行環境ファイル
オフ	オン	プログラム別実行環境ファイル
オン	オフ	共通実行環境ファイル
オン	オン	デフォルトの設定※
オフ	オフ	プログラム別実行環境ファイル

注※

印刷書式ダイアログボックスで何も設定していない状態のことです。

補足事項

実行環境ファイルを作成しない場合の印刷書式のデフォルトを次に示します。

出力先	プリンタマネージャで設定している通常使うプリンタ
余白	ページの上と左に最小 1／2 インチの余白をとる
行間隔	4LPI
1 ページの印刷行数	自動的に設定される

出力先	プリンタマネージャで設定している通常使うプリンタ
字間隔	10CPI，半角，全角文字の字間隔は調整されない
字サイズ	9 ポイント
書体	出力先プリンタに依存する

3.3.5　実行方法

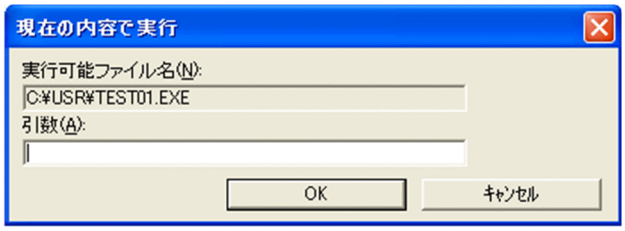
実行支援ウィンドウでの実行には，実行環境設定中の実行可能ファイルを実行する方法と，別の実行可能ファイルを実行する方法があります。

(1)　実行環境を設定したあと，続けて実行可能ファイルを実行する場合

1. [実行] メニューから［現在の内容で実行］を選ぶ。

   設定した環境情報が保存されていない場合は，ファイルを保存するかどうかを問い合わせるダイアログボックスが表示されるので，ファイルを保存してください。ファイルが保存されるまでは，プログラムを実行できません。

   次に示すダイアログボックスが表示されます。



   設定中の実行環境に該当する実行可能ファイル名が表示されます。引数の欄に引数を指定できます。

2. [OK] ボタンを選ぶ。

   実行環境設定中の実行可能ファイルが実行されます。

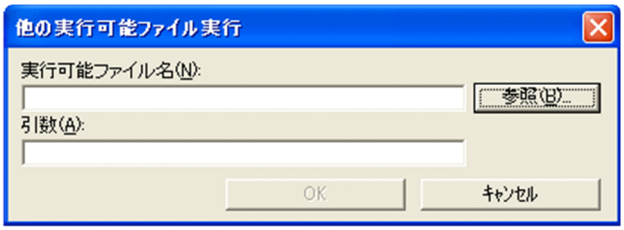
注意事項

実行環境ファイルは，実行支援ウィンドウのタイトルバーに表示されている名称で保存されます。タイトルバーが「無題」の場合は，ファイル名を設定するダイアログボックスが表示されるので，ファイル名を設定して［OK］ボタンを選びます。このとき，ファイルの名称は，設定した環境で実行したい実行可能ファイル名称と同じ（拡張子は除く）にする必要があります。

(2)　ほかの実行可能ファイルを実行する場合

1. [実行] メニューから［他の実行可能ファイル］を選ぶ。

   次に示すダイアログボックスが表示されます。



実行可能ファイル名称を絶対パスで入力，または［参照］ボタンで表示させた一覧の中からファイルを選びます。引数の欄に引数を指定できます。

2. ［OK］ボタンを選ぶ。

   指定した実行可能ファイルが実行されます。

3.3.6　実行環境ファイルの操作

(1)　実行環境ファイルの作成

実行環境ファイルを新規に作成します。

1. ［ファイル］メニューから［新規作成］を選ぶ。

   無題の実行支援ウィンドウが別に表示されます。

(2)　実行環境ファイルのオープン

既存の実行環境ファイルを開きます。

1. ［ファイル］メニューから［開く］を選ぶ。

   ファイルを指定するダイアログボックスが表示されます。

2. ファイルを指定し，［開く］ボタンを選ぶ。

   実行支援ウィンドウに表示された環境情報を更新している場合は，元から表示されているウィンドウとは別の実行支援ウィンドウに，指定した実行環境ファイルの内容が表示されます。

   表示された環境情報を更新していない場合は，元から表示されているウィンドウに，指定した実行環境ファイルの内容が表示されます。

(3)　実行環境ファイルの保存

(a)　上書き保存

1. ［ファイル］メニューから［上書き保存］を選ぶ。

   設定した環境情報が既存の実行環境ファイルにそのまま保存されます。

(b) 名前を付けて保存

1. [ファイル] メニューから [名前を付けて保存] を選ぶ。

   ファイル名を設定するダイアログボックスが表示されるので，ファイル名を設定して［保存］ボタンを選びます。このとき，指定するファイル名は，設定した環境で実行したい実行可能ファイルの名称と同じ（拡張子は除く）にしてください。

3.3.7 実行支援のエラーメッセージ表示について

実行支援の操作によって，メッセージボックスで出力されるエラーメッセージ中に「エラー値 = *****」で示されるエラーコードについては，Microsoft Visual C++のヘルプや MSDN(Microsoft Developer Network)など，Microsoft の C／C++に関するマニュアルを参照してください。

4 COBOL エディタ

COBOL2002 では，COBOL 編集に適したエディタを使用できます。この章では，COBOL エディタの操作方法について説明します。

4.1　概要

ここでは，COBOL エディタの特長について説明します。

4.1.1　COBOL 編集に適した機能

(1)　COBOL 構文によるテキストの色分け

予約語，英数字定数，サービスルーチン名などの COBOL のキーワードに，それぞれ異なる色を指定し，色分けして表示できます。指定した色で表示されない場合は，入力誤りがあることがわかります。

また，COBOL の語以外の語をユーザキーワードとして登録し，色分けすることもできます。例えば，COBOL ソースファイルを作成する上で何回も使用するデータ項目名を登録し，色分けして表示できます。

(2)　マーカによるインデント

マーカは，ソースウィンドウに表示される縦の直線です。マーカに沿ってテキストを入力すると，COBOL のコーディングに必要なインデントを揃えることができます。また，テキスト入力の際，キー操作によってテキストカーソルを前後のマーカに移動できます。

(3)　COBOL ソースの形式

ソースファイルの形式を固定形式正書法，または自由形式正書法に指定できます。固定形式正書法を指定した場合は，コーディングシートの形式に合わせてソースウィンドウの 7，8，および 73 カラムの前にマーカを表示できます。自由形式正書法を指定した場合は，256 カラムの前にマーカを表示できます。また，一連番号を生成し，ソースウィンドウに表示できます。

(4)　プログラムテンプレート

プログラムテンプレートは，部や節の見出しなどの COBOL プログラムの形式があらかじめ入力されているひな型です。このテンプレートを使うことによって，コーディング量を少なくできます。また，ユーザが作成したファイルをテンプレートとして登録することもできるため，各ユーザに合った環境でコーディングできます。

(5)　構文テンプレート

構文テンプレートは，COBOL2002 や VOS3 COBOL85 で使用する構文が入力されているひな型です。このテンプレートから使いたい構文を選び，条件や文を入力してファイルに貼り付けるため，構文エラーを少なくできます。また，ユーザがテンプレートファイルを作成することもできるため，各ユーザに合った環境でコーディングできます。

(6) 構文チェック

構文チェックは，COBOL エディタで編集中の COBOL 原始プログラムを，COBOL の言語規則に基づいてチェックする機能です。このため，プログラムをコンパイルする前に COBOL 原始プログラムの構文の誤りを発見し，修正できます。

(7) キーワード補完

キーワード補完は，途中まで入力された COBOL のキーワードに，自動的に残りを補ってキーワードを完成する機能です。このため，キーワードを入力するときに，つづりを確かめる手間が省け，また入力誤りも少なくできます。

(8) タグジャンプ

タグジャンプは，タグファイルからファイル名，および行番号を取得して，該当するファイルの該当行へカーソルを位置づける機能です。構文チェック時などにエラーが発生した場合，タグジャンプでエラー個所を開き，COBOL ソースを修正できます。

(9) 一連番号，標識領域，および見出し領域の編集

複数行の標識領域，見出し領域を一括して編集できます。また，一連番号の付け直しができます。これらの機能を使用すると，固定形式の COBOL ソースファイルを効率良く編集できます。

(10) キーボードマクロ，文字列のシフト

繰り返しの多い編集作業をする場合，キーボードマクロを使うとキー操作の回数を減らすことができます。また，文字列を左右にシフトする機能を使って，複数行のインデントを一括して編集できます。

(11) 機能キャラクタの表示

メインフレーム（VOS3）コンピュータの日本語文字処理では，日本語文字列の先頭と末尾に機能キャラクタが必要になります。そのため，Windows 上で機能キャラクタを意識しないで作成した COBOL ソースをメインフレームに転送すると，日本語を含む行が機能キャラクタの分だけ長くなり，1 行の文字列が 72 カラムを超えてしまうおそれがあります。

このような問題を避けるため，COBOL エディタには機能キャラクタを文字として表示する機能があります。この機能を使うと，日本語文字列の開始と終了の機能キャラクタをソース中に表示できます。これによって，メインフレームに転送したあとの文字数が COBOL エディタの画面上で把握でき，行の長さを確認しながらコーディングできます。

(12) 行指定の背景色分け表示の設定

特定のコメント行の間の背景色を，指定した色で表示できます。これによって，ソースファイル中の特定の個所を部分的に強調でき，効率的に編集できます。

4.1.2　その他の機能

(1)　検索・置換

文字列を検索または置換できます。一度検索または置換の対象に指定した文字列は記憶され，ドロップダウンリストに表示されます。文字列を検索する場合には，検索した文字列を含むすべての行にブックマークを設定し，あとでブックマークを使って検索することもできます。

(2)　ファイルの管理

自動的にファイルのバックアップを取ったり，編集中のファイルの内容を定期的に別のファイルに保存したりできます。

(3)　カスタマイズ情報の外部ファイル保存機能

COBOL エディタで設定したオプションなどのカスタマイズ情報を，外部ファイルにエクスポートできます。別の COBOL エディタで，エクスポートした外部ファイルをインポートすると，同じ設定を使用できます。

(4)　影響波及の調査

COBOL エディタで開いているソースプログラム中のデータ項目を選択して，影響波及調査を実行できます。この機能は，COBOL2002 Developer Professional でだけ使用できます。

4.2 COBOL エディタの起動と終了

4.2.1 COBOL エディタの起動

COBOL エディタを起動するには，新規にファイルを作成して編集する方法と，既存のファイルを編集する方法があります。

また，COBOL エディタはテストデバッガからも起動できます。詳細は「10.3.2 ソーステキストウィンドウ」を参照してください。

(1) 新規にファイルを作成して編集する方法

次のどちらかの方法で，COBOL エディタを起動できます。

- Windows のプログラム一覧にある［COBOL2002］下のメニューから［COBOL エディタ for COBOL2002］を選ぶ。

  COBOL エディタが起動されます。
- 開発マネージャのウィンドウの［ファイル］メニューから［新規作成］を選び，COBOL エディタに関連づけられたファイルの種類（COBOL ソースファイルなど）を選ぶ。

  COBOL エディタが起動され，指定した種類の新規ファイルが開かれます。

(2) 既存のファイルを編集する方法

次のどれかの方法で，COBOL エディタを起動できます。

- 開発マネージャのツリービューウィンドウから COBOL エディタに関連づけられたファイル（COBOL ソースファイルなど）を選び，［ファイル］メニューから［開く］を選ぶ。
- 開発マネージャのツリービューウィンドウから COBOL エディタに関連づけられたファイル（COBOL ソースファイルなど）をダブルクリックする。
- エクスプローラから COBOL エディタに関連づけられたファイル（COBOL ソースファイルなど）をダブルクリックする。

  COBOL エディタが起動され，指定したファイルが表示されます。

4.2.2 COBOL エディタのプロセス

COBOL エディタをサインインユーザとは別のユーザとして実行する機能を使用して複数のユーザアカウントで起動する場合，アカウントごとに一つのプロセスが起動します。また，Windows のユーザアカウント制御（UAC）が有効な OS では，管理者として実行した COBOL エディタと，サインインユーザが起動した COBOL エディタは別のプロセスとして起動します。

4.2.3　COBOL エディタの終了

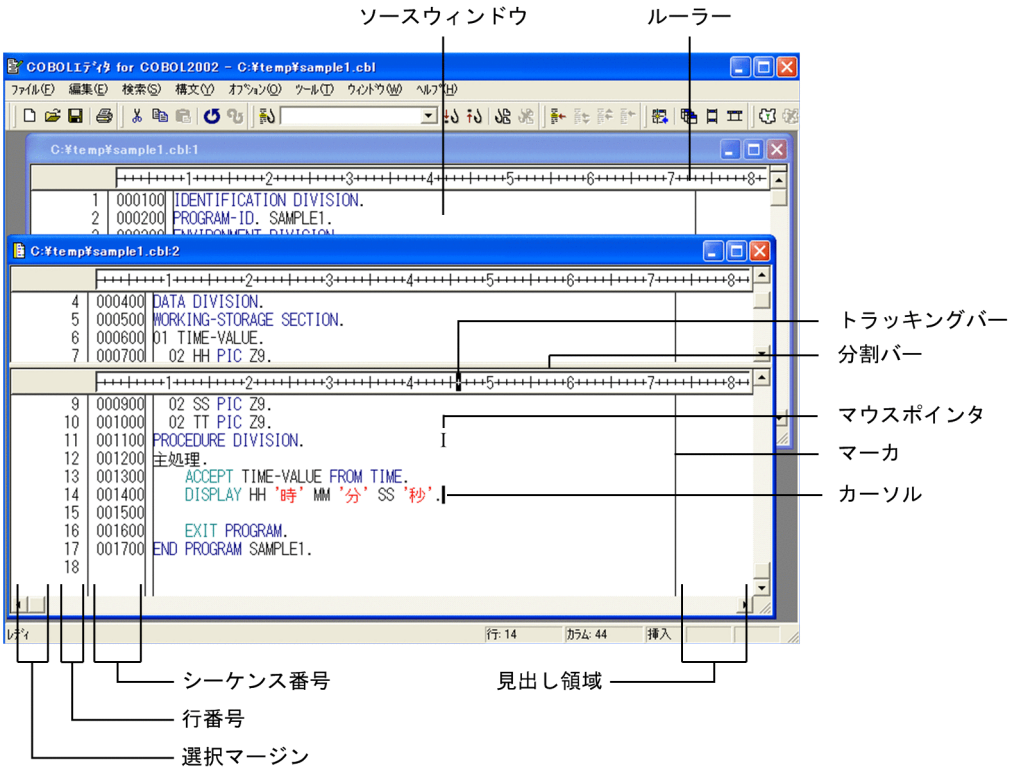
1. COBOL エディタウィンドウの［ファイル］メニューから［COBOL エディタの終了］を選ぶ。
   COBOL エディタウィンドウが閉じます。

4.3 COBOL エディタウィンドウ

4.3.1 ウィンドウの構成

COBOL エディタウィンドウの構成を次に示します。

図 4-1 COBOL エディタウィンドウ



複数のソースウィンドウがカスケード状に表示されている場合は，［ウィンドウ］メニューから［次のウィンドウ］または［前のウィンドウ］を選ぶと，最前面のウィンドウが切り替わります。

COBOL エディタでは，［ウィンドウ］メニューから［新しいウィンドウ］を選ぶと，一つのソースファイルの内容が複数のソースウィンドウに表示されます。また，分割バーを下方向にドラッグすると，ソースウィンドウが二つに分割されます。そのため，容量が大きいソースファイルを編集する場合でも，スクロールの手間が省けます。

4.3.2 メニューバー

COBOL エディタウィンドウのメニューの種類と機能を次に示します。

表 4-1　COBOL エディタウィンドウのメニュー一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	ファイルを新規に作成し，ウィンドウに表示する。
	開く(O)	ファイルを開き，ウィンドウに表示する。
	閉じる(C)	ウィンドウを閉じる。
	上書き保存(S)	編集中のファイルを更新する。
	名前を付けて保存(A)	編集中のファイルに名前を付けて保存する。
	すべて保存(L)	開いているファイルすべてを保存する。
	ページレイアウトの設定(U)	印刷時のページレイアウトを設定する。
	印刷プレビュー(V)	編集中のファイルの内容をプレビュー表示し，印刷する。
	印刷(P)	編集中のファイルの内容を印刷する。
	プロパティ(R)	編集中のファイルの情報を表示する。
	ファイル一覧（開いたファイル名を新しいものから 8 個まで表示される）	指定したファイルを開く。
	COBOL エディタの終了(X)	COBOL エディタを終了する。
編集(E)	元に戻す(U)	直前の編集操作を取り消す。
	やり直し(R)	「元に戻す」で取り消した操作を取り消す。
	切り取り(T)	選択した文字列を切り取って一時的なエリアに保存する。
	コピー(C)	選択した文字列をコピーして一時的なエリアに保存する。
	貼り付け(P)	一時的なエリアの文字列を選択した位置に貼り付ける。
	削除(D)	選択した文字列を削除する。
	すべてを選択(S)	ファイル全体を選択する。
	大文字に変換(E)	選択範囲の英文字を大文字に変換する。
	小文字に変換(L)	選択範囲の英文字を小文字に変換する。
	タブを空白文字に置換(O)	選択範囲のタブ文字を空白文字に置換する。
	ファイルの挿入(I)	指定したファイルのすべての内容をテキストカーソル位置に挿入する。
検索(S)	文字列の検索(F)	指定した文字列を検索する。
	置換(R)	指定した文字列を新しい文字列に置換する。
	ファイルから検索(I)※	指定した文字列をファイルから検索する。
	ファイルから検索を中止(I)※	ファイルからの検索を中止する。
	ブックマークの設定／解除(B)	ブックマークを設定または解除する。

メニューバー	メニュー項目	機能
	次のブックマーク(X)	次のブックマークを検索する。
	前のブックマーク(V)	前のブックマークを検索する。
	ブックマークをすべて解除(C)	すべてのブックマークを解除する。
	ジャンプ(G)	指定した行番号へジャンプする。
	一連番号を検索(S)	指定した一連番号を検索する。
	不当な一連番号を検索(N)	不当な一連番号を検索する。
	80 バイトを超える行を検索(L)	1 行の文字が 80 バイトを超えている行を検索する。
構文(Y)	構文チェック(S)	編集ファイルの構文をチェックする。
	構文チェックの中止(C)	実行中の構文チェックを中止する。
	構文チェック　オプション(O)	構文チェックの環境を設定する。
	構文テンプレート(B)	構文テンプレートをウィンドウに挿入する。
	キーワード補完(E)	COBOL キーワードを補完する。
	ユーザキーワード補完(K)	ユーザキーワードを補完する。
オプション(O)	カスタマイズ(C)	COBOL エディタ，カラー，およびフォントのオプションを設定する。
	エクスポート(E)	カスタマイズ情報を外部ファイルに保存する。
	インポート(I)	カスタマイズ情報を外部ファイルから読み込む。
	ツールバーの表示(S)	[標準(D)]： 　標準用ツールバーの表示／非表示を指定する。 [編集(E)]： 　編集用ツールバーの表示／非表示を指定する。 [検索(S)]： 　検索用ツールバーの表示／非表示を指定する。 [ブックマーク(B)]： 　ブックマーク用ツールバーの表示／非表示を指定する。 [ウィンドウ(W)]： 　ウィンドウ用ツールバーの表示／非表示を指定する。 [構文チェック(C)]： 　構文チェック用ツールバーの表示／非表示を指定する。 [正書法(F)]： 　正書法用ツールバーの表示／非表示を指定する。
ツール(T)	キーボードマクロ登録開始(S)※	キー入力の記録を開始する。
	キーボードマクロ登録終了(S)※	キー入力の記録を終了する。
	キーボードマクロ実行(R)	記録したキー操作を再生する。

メニューバー	メニュー項目	機能
	選択行の右シフト(I)	選択した行を右方向にシフトする。
	選択行の左シフト(E)	選択した行を左方向にシフトする。
	標識領域の指定(A)	[注記(O):*]： 選択範囲の標識領域をアスタリスク(*)に置換する [注記(M):/]： 選択範囲の標識領域を斜線(/)に置換する [継続(N):-]： 選択範囲の標識領域をハイフン(-)に置換する [デバッグ(B):D]： 選択範囲の標識領域を「D」に置換する [空白(A):]： 選択範囲の標識領域を空白に置換する
	見出し領域の置換(L)	見出し領域を，指定した文字列や現在の日付と置換する。
	一連番号の指定(N)	[すべての一連番号を付け直す(A)]： 一連番号領域のすべての一連番号を付け直す [不当な一連番号を付け直す(I)]： 一連番号領域の不当な一連番号を付け直す [一連番号領域を空白に置換(R)]： 一連番号領域を空白に置換する
	COBOL ソース表示形式の切り替え(W)	[固定形式正書法(X)]： マーカと COBOL 構文の色分け表示を固定形式正書法で表示する [自由形式正書法(R)]： マーカと COBOL 構文の色分け表示を自由形式正書法で表示する
ウィンドウ(W)	新しいウィンドウ(N)	表示中のファイルを新しいウィンドウに表示する。
	次のウィンドウ(X)	次のウィンドウをアクティブ状態にする。
	前のウィンドウ(P)	前のウィンドウをアクティブ状態にする。
	重ねて表示(C)	ウィンドウを重ねて表示する。
	上下に並べて表示(H)	ウィンドウを上下に並べて表示する。
	左右に並べて表示(V)	ウィンドウを左右に並べて表示する。
	すべてアイコン化(I)	ウィンドウをすべてアイコン表示する。
	すべて元のサイズに戻す(R)	アイコン表示されたウィンドウをすべて元のサイズに戻す。
	すべて閉じる(A)	ウィンドウをすべて閉じる。
	開いているウィンドウ	指定したウィンドウを表示する。

メニューバー	メニュー項目	機能
	その他のウィンドウ（開いているウィンドウが 10 個以上の場合に表示される）	指定したウィンドウを表示する。
ヘルプ(H)	バージョン情報(A)	COBOL エディタについてのシステム情報，バージョン情報，著作権情報を表示する。

注※　機能の開始，終了でメニューが切り替わります。

4.3.3　ツールバー

ツールバーには，標準用と編集用を含み全部で 7 種類があります。これらは，通常は一つのツールバーとして表示されていますが，それぞれをマウスで選んで移動させることができます。

ツールバーは，前回 COBOL エディタを終了したときと同じ状態で表示されます。なお，COBOL エディタを初めて起動した場合は，正書法用ツールバーを除き，メニューの下に標準用，編集用，検索用，ブックマーク用，ウィンドウ用，構文チェック用のツールバーが表示されます。

また，これらのツールバーは，［オプション］メニューの［ツールバーの表示］で表示するかしないかを指定できます。

(1)　標準用ツールバー

標準用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. ［ファイル］メニューの［新規作成］
2. ［ファイル］メニューの［開く］
3. ［ファイル］メニューの［上書き保存］
4. ［ファイル］メニューの［印刷］

(2)　編集用ツールバー

編集用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [編集] メニューの [切り取り]
2. [編集] メニューの [コピー]
3. [編集] メニューの [貼り付け]
4. [編集] メニューの [元に戻す]
5. [編集] メニューの [やり直し]

(3) 検索用ツールバー

検索用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [検索] メニューの [文字列の検索]
2. 検索文字列入力領域
3. [検索] メニューの [次を検索]
4. [検索] メニューの [前を検索]
5. [検索] メニューの [ファイルから検索]
6. [検索] メニューの [ファイルから検索を中止]

(4) ブックマーク用ツールバー

ブックマーク用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [検索] メニューの [ブックマークの設定／解除]
2. [検索] メニューの [次のブックマーク]
3. [検索] メニューの [前のブックマーク]
4. [検索] メニューの [ブックマークをすべて解除]

(5) ウィンドウ用ツールバー

ウィンドウ用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [ウィンドウ] メニューの [新しいウィンドウ]
2. [ウィンドウ] メニューの [重ねて表示]
3. [ウィンドウ] メニューの [上下に並べて表示]
4. [ウィンドウ] メニューの [左右に並べて表示]

(6) 構文チェック用ツールバー

構文チェック用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [構文] メニューの [構文チェック]
2. [構文] メニューの [構文チェックを中止]

(7) 正書法用ツールバー

正書法用ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。

COBOL エディタを初めて起動した場合，正書法用ツールバーは表示しません。



1. [ツール] メニューの [固定形式正書法]
2. [ツール] メニューの [自由形式正書法]

4.3.4 ポップアップメニュー

COBOL エディタのソースウィンドウを右クリックすると，ポップアップメニューが表示されます。

次に，ソースウィンドウのポップアップメニューのメニュー項目と機能を示します。

表 4-2　ポップアップメニュー

メニュー項目	機能
切り取り	選択した文字列を切り取って一時的なエリアに保存する。
コピー	選択した文字列をコピーして一時的なエリアに保存する。
貼り付け	一時的なエリアの文字列を選択した位置に貼り付ける。
削除	選択した文字列を削除する。
標識領域の指定	標識領域にある文字を標識領域で指定できる文字に置換する。
見出し領域の置換	見出し領域を，指定した文字列や現在の日付と置換する。
影響波及調査	選択したデータ項目の影響波及を調査する。※1
構文テンプレート	構文テンプレートをウィンドウに挿入する。
キーワード補完	COBOL キーワードを補完する。
ユーザキーワード補完※2	ユーザキーワードを補完する。
ルーラーの表示／非表示	ルーラーを表示するかしないかを指定する。
プロパティ	編集中のファイルの情報を表示する。

注※1

データ影響波及分析については，マニュアル「COBOL2002 Professional Tool Kit データ影響波及分析ガイド」を参照してください。

注※2

ユーザキーワード補完を使用するには，カスタマイズダイアログボックスの［予約語ファイル］タブを選び，使用するシステム属性に合わせて，ユーザキーワードを登録しなければなりません。

4.3.5　ルーラー

ルーラーは，ソースウィンドウ上のカラム位置を示します。ルーラーには，現在のマウスポインタの位置を示すトラッキングバーが表示されています。また，ルーラーからマーカを設定して，インデントを揃えたいカラム位置を表示できます。

トラッキングバーやマーカを操作するには，ルーラーポップアップメニューを使用します。

(1)　ルーラーポップアップメニューの表示

1. ルーラー上にマウスポインタを位置づける。

   マウスポインタが十字型に変わります。

2. マウスの右ボタンを押す。

   ルーラーポップアップメニューが表示されます。

ルーラーが表示されていない場合は，マウスポインタをソースウィンドウ中に位置づけ，マウスの右ボタンを押すと，エディットポップアップメニューが表示されます。ポップアップメニュー中の［ルーラーの表示］を選んで，ルーラーを表示した上で，1.の操作をしてください。

なお，表示のオプションで，ルーラーを非表示に指定している場合は，エディットポップアップメニューに［ルーラーの表示］は表示されません。ルーラーを表示したい場合は，オプションを設定し直してください。表示のオプションについては，「4.10.2　表示の設定」を参照してください。

(2)　ルーラーポップアップメニューの項目と機能

次に，ルーラーポップアップメニューのメニュー項目と機能を示します。

表 4-3　ルーラーポップアップメニューのメニュー項目と機能

メニュー項目	機能
トラッキングバーの表示／非表示	トラッキングバーを表示するかしないかを指定する。
マーカの表示／非表示	マーカを表示するかしないかを指定する。
マーカ設定	マウスポインタ位置にマーカを設定する。
マーカ解除	マウスポインタ位置のマーカを解除する。
すべてのマーカの解除	すべてのマーカを解除する。
ルーラーの非表示	ルーラーの非表示を指定する。

(3)　マーカの操作

- **ルーラーポップアップメニューを使用する場合**

  ルーラーポップアップメニューから［マーカ設定］［マーカ解除］または［すべてのマーカの解除］を選ぶと，マーカは設定または解除されます。

- **マウスを使用する場合**

  ルーラー上のマーカを設定したい位置をマウスでクリックするとマーカが設定され，ダブルクリックすると解除されます。

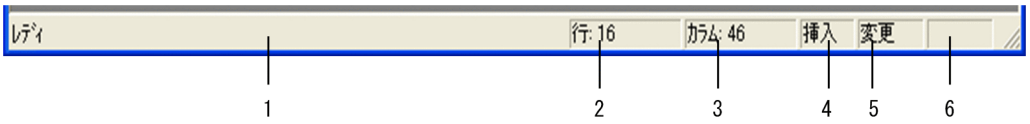
また，次の操作でカーソルをマーカへ移動させることができます。

表 4-4　マーカの操作（マウスを使用する場合）

キー操作	カーソルの移動先
［Ctrl］＋［Tab］キー	次のマーカへ
［Ctrl］＋［Shift］＋［Tab］キー	前のマーカへ

4.3.6　ステータスバー

ステータスバーには，カーソルの位置（行，カラム）やメッセージ，入力モードなどが表示されます。ステータスバーの表示と内容を次に示します。



1. メッセージ

   メニューの機能，メッセージジャンプのコンパイラメッセージ，および処理中のメッセージが表示されます。

2. 行

   カーソルの行番号が表示されます。

3. カラム

   カーソルのカラム位置が表示されます。

4. 入力モード

   文字の入力モードが表示されます。

   - 挿入：挿入モードを表します。
   - 上書：上書きモードを表します。

5. 編集表示

   ファイルの更新状態が表示されます。

   - 変更：ファイルが更新されていることを表します。
   - 書禁：ファイルが書き込み禁止属性であることを表します。

   なお，何も表示されていない場合は，ファイルが更新されていないことを表します。

6. 通信状態

   開発マネージャから COBOL エディタを起動した場合，開発マネージャと COBOL エディタとの間での通信状況が表示されます。

   - 接続：通信が接続中であることを表します。
   - LOCK：COBOL エディタが，開発マネージャからファイルを編集できないようにロックしていることを表します。

   なお，何も表示されていない場合は，通信機能が接続されていない状態であることを表します。

4.3.7　カーソルの形状

挿入モードの場合，縦線のカーソルが表示されます。

上書きモードの場合，横線のカーソルが表示されます。

4.3.8　メニューバーのアイコンとファイル種別

編集しているファイルの種別によって，ソースウィンドウのメニューバーの左上に表示されるアイコンが変化します。ファイル種別とアイコンとの対応を，次に示します。

 　固定形式ファイル

 　自由形式ファイル

 　その他のファイル

4.4　ファイルの操作

COBOL エディタのファイルに関する操作を説明します。

4.4.1　ファイルの作成

1. COBOL エディタウィンドウの［ファイル］メニューから［新規作成］を選ぶ。
   新規作成ダイアログボックスが表示されます。

2. ファイルの種類を選ぶ。
   必要であれば，拡張子やテンプレートを指定してください（「4.5.1　プログラムテンプレート」参照）。

3. ［OK］ボタンを選ぶ。
   新規にファイルが作成され，ソースウィンドウに表示されます。
   テンプレートを指定した場合は，指定したテンプレートの内容で新規にファイルが作成されます。
   なお，ツールバーの［新規作成］ボタンでは，新規作成ダイアログボックスでデフォルトに設定されている内容で新規にファイルが作成されます。

4.4.2　ファイルのオープン

(1)　開く

1. COBOL エディタウィンドウの［ファイル］メニューから［開く］を選ぶ。
   ファイルを指定するダイアログボックスが表示されます。

2. リストボックスから目的のファイルを選ぶ。または，ファイル名テキストボックスにファイル名を入力する。

3. ［開く］ボタンを選ぶ。
   指定したファイルが開かれ，ソースウィンドウに表示されます。

(2)　特殊文字を含んだファイルを開いた場合

COBOL エディタは，特殊文字（NULL や垂直タブなど）を含んだファイルを開けません。このようなファイルを開こうとした場合，特殊文字を削除するかしないかを問い合わせるメッセージが表示されます。特殊文字を取り除いてファイルを開きたい場合は，［はい］を選んでください。特殊文字を削除したくない場合は，［いいえ］を選んでファイルを開く処理を中止してください。

COBOL エディタが特殊文字と認識する文字コードを，次に示します。

- X'00'～X'08'，X'0B'，X'0E'～X'1F'，X'7F'，X'FD'～X'FF'

シフト JIS のマルチバイト文字として正しくないコードやシフト JIS の範囲内でも文字として割り当てられていないコードは，不当文字とみなして，削除対象になります。文字として割り当てられていないコードを次に示します。

表 4-5　文字として割り当てられていないコード

前半バイト	後半バイト
X'81'	X'AD'，X'AE'，X'AF'，X'B0'～X'B7'，X'C0'～X'C7'，X'CF'～X'D9'，X'E9'～X'EF'，X'F8'～X'FB'
X'82'	X'40'～X'4E'，X'59'～X'5F'，X'7A'～X'7E'，X'80'，X'9B'～X'9E'，X'F2'～X'FC'
X'83'	X'97'～X'9E'，X'B7'～X'BE'，X'D7'～X'FC'
X'84'	X'61'～X'6F'，X'92'～X'9E'，X'BF'～X'FC'
X'85'	X'40'～X'7E'，X'80'～X'FC'
X'86'	X'40'～X'7E'，X'80'～X'FC'
X'87'	X'5E'，X'76'～X'7D'，X'9D'～X'FC'
X'88'	X'40'～X'7E'，X'80'～X'9E'
X'98'	X'73'～X'7E'，X'80'～X'9E'
X'EA'	X'A5'～X'FC'
X'EB'	X'40'～X'7E'，X'80'～X'FC'
X'EC'	X'40'～X'7E'，X'80'～X'FC'
X'EE'	X'ED'～X'EE'
X'EF'	X'40'～X'7E'，X'80'～X'FC'
X'FC'	X'4C'～X'7E'，X'80'～X'FC'

(3)　注意事項

- COBOL エディタでは，文字コードにシフト JIS を使用している日本語テキストファイルだけを開きます。EUC，Unicode などのシフト JIS 以外の文字コードを使用しているテキストファイルは，使用できません。
- 開発マネージャから COBOL エディタを起動した場合で，ファイル中に特殊文字が含まれているとき，特殊文字を削除するかしないかの問い合わせのダイアログボックスに長時間応答しないと，COBOL エディタの起動がキャンセルされます。
- HiRDB 埋め込み型 UAP（*.ecb，*.eoc）から生成される COBOL ソースを COBOL エディタで開いたときに，COBOL エディタは読み取り専用で開きます。また，COBOL ソースが存在しない場合，COBOL ソースは開かれないで"HiRDB のプリプロセッサの出力ソースが存在しません。"のメッセージが表示されます。

4.4.3　ファイルの保存

(1)　上書き保存

1. COBOL エディタウィンドウの［ファイル］メニューから［上書き保存］を選ぶ。
   ファイルの内容が，既存のファイルにそのまま保存されます。

(2)　名前を付けて保存

1. COBOL エディタウィンドウの［ファイル］メニューから［名前を付けて保存］を選ぶ。
   ファイル名を設定するダイアログボックスが表示されるので，ファイル名を入力してください。

2. ［保存］ボタンを選ぶ。
   指定した名でファイルが作成されます。

(3)　すべて保存

1. COBOL エディタウィンドウの［ファイル］メニューから［すべて保存］を選ぶ。
   現在ソースウィンドウに表示されているファイルがすべて保存されます。新規にファイルを作成した場合は，ファイル名を設定するダイアログボックスが表示されるので，ファイル名を入力してください。

4.4.4　ファイルのクローズ

1. COBOL エディタウィンドウの［ファイル］メニューから［閉じる］を選ぶ。
   ファイルの内容を更新した場合は，ファイルを書き換えるかどうかを問い合わせるダイアログボックスが表示されます。

2. ［はい］ボタンを選ぶ。
   新規にファイルを作成した場合は，ファイル名を設定するダイアログボックスが表示されるので，ファイル名を入力してください。
   ソースウィンドウが閉じられます。

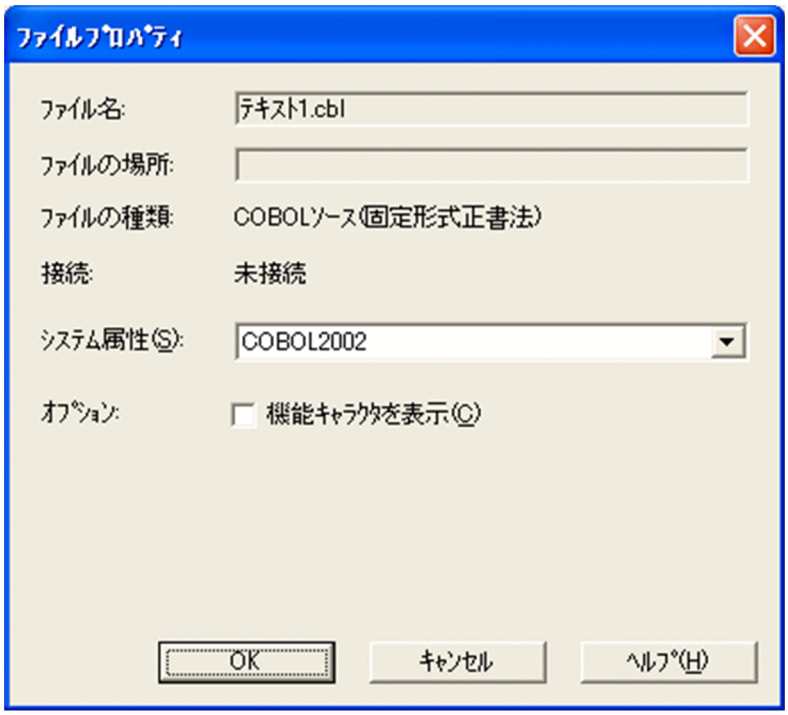
4.4.5　ファイル情報の参照

編集中のソースファイルがどこに格納されているのか，どのシステムに対応した COBOL プログラムなのか，といったファイル情報を，COBOL エディタ上で確認できます。

選択中のソースウィンドウのファイル情報を表示する方法を次に示します。

1. COBOL エディタウィンドウの［ファイル］メニューから［プロパティ］を選ぶ。

ファイルプロパティダイアログボックスが表示されます。



ファイル名

ファイル名が表示されます。

ファイルの場所

ファイルが格納されているフォルダの絶対パス名が表示されます。新規に作成したファイルの場合は，何も表示されません。

ファイルの種類

ファイルの種類が表示されます。表示されるファイルの種類と意味を，次に示します。

- COBOL ソース（固定形式正書法）
  固定形式正書法で書かれた COBOL 原始プログラムを格納したファイル。
- COBOL ソース（自由形式正書法）
  自由形式正書法で書かれた COBOL 原始プログラムを格納したファイル。
- テキストファイル
  上記以外の形式のファイル。

接続

COBOL エディタと接続しているアプリケーションが表示されます。

- 未接続
  通信が接続されていないことを表します。
- 開発マネージャ
  開発マネージャとの通信が接続されていることを表します。

システム属性

設定されたシステムの種類を表示します。

開発マネージャから COBOL エディタを起動した場合は，システム属性が自動的に設定されます。それ以外の方法で起動した場合は，システムの種類を選択できます。

システムの種類

- COBOL2002
  COBOL2002 向けの COBOL プログラムを表します。
- COBOL85
  COBOL85 向けの COBOL プログラムを表します。
- VOS3 COBOL85
  VOS3　COBOL85 向けの COBOL プログラムを表します。

システム属性に設定したシステムの種類に従って COBOL 予約語，構文，サービスルーチンの色分け表示，キーワード補完，ユーザキーワード補完，構文テンプレートの表示を切り替えます。 各システムの COBOL 予約語，COBOL サービスルーチン，ユーザキーワードの設定は，「4.10.3　色の設定」を参照してください。また，構文テンプレートについては「4.5.2　構文テンプレート」を参照してください。

オプション

機能キャラクタを表示するかしないかを選択します。

なお，機能キャラクタとして表示される文字は，カスタマイズダイアログボックスの［表示］タブを選んで表示されるページで設定します。詳細は，「4.10.2　表示の設定」を参照してください。

4.5　テンプレート機能

COBOL エディタでは，より快適な環境でコーディングできるようにテンプレート機能を提供しています。テンプレート機能には，プログラムテンプレートと構文テンプレートがあります。

4.5.1　プログラムテンプレート

プログラムテンプレートとは，部や節の見出しなどの COBOL プログラムの形式（書き方）があらかじめ入力されているプログラムのひな型です。COBOL エディタでは次のようなテンプレートを提供しています。

COBOL エディタが提供しているプログラムテンプレート

- COBOL2002 プログラム用
- COBOL2002 入出力機能プログラム用
- COBOL2002 整列機能プログラム用
- COBOL2002 報告書作成機能プログラム用
- COBOL2002 画面機能プログラム用
- COBOL2002 データベースアクセス機能プログラム用

プログラムテンプレートの格納場所

```
COBOL2002インストールフォルダ¥template
```

[カスタマイズ]ダイアログボックスで，[予約語]タブの[テンプレートフォルダ]を変更して，この格納場所を変更できます。

また，ユーザが作成したファイルをテンプレートとして登録することもできます。このため，ユーザ独自のテンプレートを作成し，各ユーザに合った環境でプログラムをコーディングできます。

(1)　テンプレートを使用してファイルを新規作成する

1. ［ファイル］メニューから［新規作成］を選ぶ。
   新規作成ダイアログボックスが表示されます。
2. 作成したいファイルの種類を選ぶ。
   必要ならば，ファイルの拡張子も選びます。
3. ファイルの種類に対応するプログラムテンプレートの［...］ボタンを選び，表示されるダイアログボックスからテンプレートファイルを選ぶ。
4. プログラムテンプレートのチェックボックスをオンにする。
5. 2.～4.で設定した内容を次回以降のデフォルトにしたい場合は，［設定値を登録］ボタンを選ぶ。

6. [OK] ボタンを選ぶ。

新規にファイルが作成され，テンプレートの内容がソースウィンドウに表示されます。

次に，新規作成ダイアログボックスの内容について説明します。



ファイルの種類

新規に作成するファイルの種類を選びます。

- COBOL ソース（固定形式）

  固定形式正書法の COBOL 原始プログラムを作成します。

- COBOL ソース（自由形式）

  自由形式正書法の COBOL 原始プログラムを作成します。

- テキスト，その他のファイル

  テキストファイルや，上記以外の形式のファイルを作成します。

拡張子

作成するファイルの拡張子が表示されます。また，COBOL ソース（固定形式）および COBOL ソース（自由形式）の場合，ドロップダウンリストからファイルの拡張子を選択できます。

プログラムテンプレート

ファイルの種類ごとに使用したいテンプレートを登録します。テンプレートは，[...] ボタンで表示されるダイアログボックスから選びます。

登録したテンプレートを使用する場合は，プログラムテンプレートの左にあるチェックボックスをオンにしてください。チェックボックスをオフにした場合，プログラムテンプレートが指定してあっても，テンプレートを使用しないでファイルを新規作成します。

設定値を登録

[設定値を登録] ボタンを選ぶと，このダイアログボックスで指定したファイルの種類，拡張子，およびプログラムテンプレートを登録します。設定値を登録すると，ファイルを新規作成するときに，前回登録した内容をデフォルトとして使用できます。

注意事項

設定した値を登録する場合は，まず［設定値を登録］ボタンを選んでから，［OK］ボタンを選んでください。［設定値を登録］ボタンを選ばないと，設定した値は登録されません。

(2)　テンプレートとして登録する

1. [ファイル] メニューから［名前を付けて保存］を選ぶ。
   ファイル名を指定するダイアログボックスが表示されます。

2. ファイルの拡張子に「.cet」を指定して［保存］ボタンを選ぶ。
   編集中のファイルがプログラムテンプレートとして保存されます。

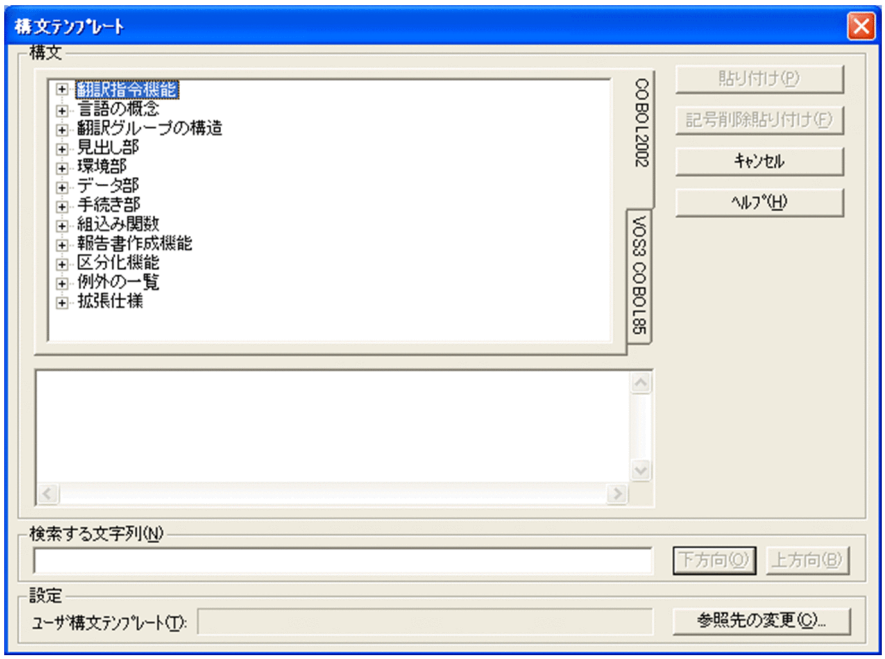
4.5.2　構文テンプレート

構文テンプレートとは，COBOL2002 で使用する構文が入力されているひな型です。このテンプレートから使いたい構文を選び，条件や文を入力してソースウィンドウに貼り付けることができます。このため，コーディングの誤りや構文エラーを少なくできます。また，ユーザが独自のテンプレートファイルを作成することもできるため，各ユーザに合った環境でコーディングできます。

(1)　構文テンプレートの使用方法

1. ソースウィンドウ上の構文を挿入したい位置にカーソルを位置づける。

2. [構文] メニューから［構文テンプレート］を選ぶ。
   または
   [Ctrl] ＋ [B] キーを押す。
   構文テンプレートダイアログボックスが表示されます。



構文

構文ツリー欄（上の欄）と構文形式欄（下の欄）から構成されています。

構文ツリー欄には，COBOL の構文の名称が階層表示されます。[COBOL2002] タブを選択したときは COBOL2002 用の構文テンプレートを，[VOS3 COBOL85] タブを選択したときは VOS3 COBOL85 用の構文テンプレートを表示します。

1.の操作でカーソルが構文上のキーワードの先頭から末尾の間にある場合は，そのキーワードに該当する構文名称が選択状態になります。ツリーをたどって階層を下げていくと，構文形式欄に構文の形式が表示されます。

構文形式欄には，構文ツリーから選択した構文の形式が表示されます。構文形式欄で条件や文を入力したあと，[貼り付け] ボタンで構文形式欄の内容をソースウィンドウに貼り付けます。また，[記号削除貼り付け] ボタンでは，構文形式欄上の’ {}’,’ []’,・(中点)，｜（ストローク)，および／（スラント）を削除して貼り付けできます。

検索する文字列

構文テンプレートで検索する文字列を指定します。

検索する文字列を指定したあと，[下方向] ボタンを選ぶと，構文ツリーで現在表示している場所から下方向に文字列が検索されます。同様に，[上方向] ボタンを選ぶと，上方向に文字列が検索されます。

ユーザ構文テンプレート

ユーザ構文テンプレートファイルの参照先を表示します。右側の [参照先の変更...] ボタンを押して，ユーザ構文テンプレートファイルの参照先を変更できます。

3. **使用したい構文を選ぶ。**

構文ツリー欄をたどってクリックし，使用したい構文の形式を表示させます。

（例）

「手続き部」→「手続き部の文」→「ADD 文」→「ADD （一意名)」

4. **構文形式欄で条件や文を入力する。**

5. **[貼り付け] または [記号削除貼り付け] ボタンを選ぶ。**

構文形式欄の内容がソースウィンドウのカーソル位置に貼り付けられます。ソースウィンドウで文字列が選択されている場合は，選択文字と置き換えて貼り付けられます。

(2) 構文テンプレートファイルの作成

COBOL エディタでは，ユーザ独自のテンプレートファイルを作成できます。構文テンプレートファイルを作成するときは，「user.syn」または「userv.syn」の名称で作成してください。作成した構文テンプレートファイルの既定の参照先は，COBOL2002 のインストールフォルダ下の「bin」フォルダです。

この参照先は，構文テンプレートダイアログボックスで変更または確認できます。

表 4-6 構文テンプレートファイル名

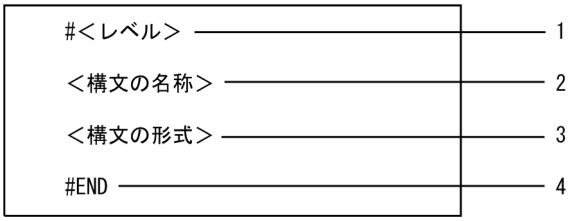
ファイル名	登録内容
user.syn	COBOL2002 の構文テンプレートファイル名

ファイル名	登録内容
userv.syn	VOS3 COBOL85 の構文テンプレートファイル名

ユーザが作成したテンプレートファイルの内容は，構文テンプレートダイアログボックスに，COBOL エディタが提供するテンプレートのあとに続いてツリー表示されます。

次に，構文テンプレートファイルに記述する内容について説明します。

構文テンプレートファイルでは，一つの構文テンプレートを次のような形式で表します。



1.には 1～9 の数字を記述します。これが構文テンプレートダイアログボックスの構文ツリーでの階層になります。階層が複雑にならないように，レベルは 4 までとすることを推奨します。

2.には構文の名称を 1 行で記述します。これが構文テンプレートダイアログボックスの構文ツリーに表示される名称になります。

3.には構文の形式を記述します。ここには複数行記述できます。この構文の形式を構文テンプレートダイアログボックスの構文形式欄で編集し，ソースウィンドウに貼り付けることになります。そのため，構文形式欄での編集が容易になるような記述を推奨します。

4.は一つの構文テンプレートの終わりを表します。

このような構文テンプレートを必要なだけ記述して，ユーザ独自のテンプレートファイルを作成してください。なお，テンプレートファイルに記述した順序で構文テンプレートダイアログボックスに階層表示されるので，それも考慮に入れて記述してください。

注意事項

テンプレートファイル中で文字「#」を使用できるのは，「#＜レベル＞」と「#END」だけです。その他の部分では，「#」を使用できません。

構文テンプレートファイルの例

次に，テンプレートファイルと構文テンプレートダイアログボックスでの構文ツリーの例を示します。

図 4-2　構文テンプレートファイルの例

```
#1
コーディング規則
#END

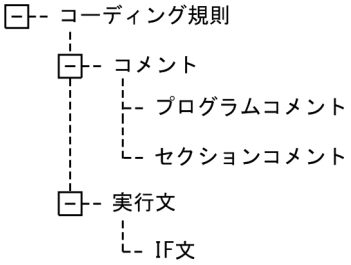
#2
コメント
#END

#3
プログラムコメント
*****************************************************
*  入り口点名称  :
*  機能概略  :
*  作成者    :
*  作成日付  :
*  更新日付  :
*****************************************************
#END
#3
セクションコメント
*****************************************************
*  セクション名  :
*  機能概略  :
*  更新日付  :
*****************************************************
#END

#2
実行文
#END

#3
IF文
IF   条件
  THEN
     実行文
  ELSE
     実行文
END-IF
#END
```

図 4-3　構文ツリーの例



4.6　文字列の編集

4.6.1　基本操作

文字列を編集するための基本的な操作について説明します。

(1)　文字の入力

特殊な文字の入力

特殊な文字の入力方法を次に示します。

表 4-7　特殊な文字の入力方法

キー操作	入力される文字
[Enter] キー	改行文字
[Ctrl] ＋ [Enter] キー	改ページ文字と改行文字
[Tab] キー	タブ文字※

注※

「空白に置換」が指定されていない場合に入力されます。「空白に置換」は，カスタマイズダイアログボックスの［編集］タブを選ぶと表示されるページで指定します。詳細は，「4.10.1　編集動作の設定」を参照してください。

(2)　カーソルの移動

(a)　マウス操作の場合

1. 移動先にマウスポインタを位置づけクリックする。

   指定した位置にカーソルが移動します。

(b)　キー操作の場合

キーによってカーソルがどのように移動するかを次に示します。

表 4-8　キーによるカーソルの移動

キー操作	移動先
[←] キー	1 文字左へ
[→] キー	1 文字右へ
[↑] キー	1 行上へ
[↓] キー	1 行下へ

キー操作	移動先
[Ctrl] ＋ [←] キー	前の文字列の先頭へ※
[Ctrl] ＋ [→] キー	次の文字列の先頭へ
[Ctrl] ＋ [↑] キー	1 行上にスクロールする
[Ctrl] ＋ [↓] キー	1 行下にスクロールする
[Page Up] キー	前ページへ
[Page Down] キー	次ページへ
[Ctrl] ＋ [Page Up] キー	左ページへ
[Ctrl] ＋ [Page Down] キー	右ページへ
[Home] キー	行の先頭へ
[End] キー	行の末尾へ
[Ctrl] ＋ [Home] キー	ファイルの先頭へ
[Ctrl] ＋ [End] キー	ファイルの末尾の行の先頭へ
[Ctrl] ＋ [Tab] キー	次のマーカへ
[Shift] ＋ [Ctrl] ＋ [Tab] キー	前のマーカへ

注※

カーソルが文字列中にある場合は，その文字列の先頭へ移動します。

(3) 文字列の選択

(a) マウス操作の場合

(i) 語選択

1. 選択したい語をダブルクリックする。

   指定した語が選択されます。

(ii) 行選択

1. 選択したい行の選択マージン（左余白），または行番号を選ぶ。

   指定した行が選択されます。

(iii) 範囲選択

1. 選択したい最初の位置を選び，最後の位置までドラッグする。

   指定した範囲がすべて選択されます。

(iv) 行の範囲選択

1. 選択したい最初の行の選択マージン（左余白）を選び，最後の行の選択マージンまでドラッグする。
   通常の行番号が表示されている場合は，行番号も選べます。
   指定した範囲のすべての行が選択されます。

(v) ブロック選択

1. [Alt] キーを押したまま，選択したいブロックの最初の位置を選び，目的のサイズになるまでドラッグする。
   指定したブロックが選択されます。また，[Alt] ＋ [Shift] キーを押したまま，選択したいブロックの最初の位置を選び，次に選択したいブロックの最後の位置を選んでも，ブロック選択ができます。

(b) キー操作の場合

キーによって文字列を選択する操作を次に示します。

表 4-9 キーによる文字列の選択

キー操作	選択対象（カーソルの位置を基準とする）
[Shift] ＋ [←] キー	1 文字左まで
[Shift] ＋ [→] キー	1 文字右まで
[Shift] ＋ [↑] キー	1 行上のカーソルの位置まで
[Shift] ＋ [↓] キー	1 行下のカーソルの位置まで
[Shift] ＋ [Ctrl] ＋ [←] キー	前の文字列の先頭まで
[Shift] ＋ [Ctrl] ＋ [→] キー	次の文字列の先頭まで
[Shift] ＋ [Page Up] キー	前ページまで
[Shift] ＋ [Page Down] キー	次ページまで
[Shift] ＋ [Home] キー	行の先頭まで
[Shift] ＋ [End] キー	行の末尾まで
[Shift] ＋ [Ctrl] ＋ [Home] キー	ファイルの先頭まで
[Shift] ＋ [Ctrl] ＋ [End] キー	ファイルの末尾まで

(c) メニュー操作の場合

1. COBOL エディタウィンドウの［編集］メニューから［すべてを選択］を選ぶ。
   編集中のソースファイル全体が選択状態になります。

4.6.2 編集操作

文字列を編集する操作について説明します。

(1) 移動

1. 移動したい文字列を選び，［編集］メニューから［切り取り］を選ぶ。
2. 移動先を選び，［編集］メニューから［貼り付け］を選ぶ。
   また，移動したい文字列を選び，移動先までドラッグしても移動できます。

(2) 複写

1. 複写したい文字列を選び，［編集］メニューから［コピー］を選ぶ。
2. 複写先を選び，［編集］メニューから［貼り付け］を選ぶ。
   また，複写したい文字列を選び，［Ctrl］キーを押したまま，複写先までドラッグしても複写できます。

(3) 削除

1. 削除したい文字列を選び，［編集］メニューから［削除］を選ぶ。
   また，［Delete］キー，または［Back Space］キーを使って，文字列を次のように削除できます。

表 4-10 ［Delete］キー，または［Back Space］キーによる，文字列の削除

キー操作	機能
［Back Space］キー	カーソルの左 1 文字を削除
［Delete］キー	カーソルの右 1 文字を削除
［Ctrl］＋［Back Space］キー	カーソルの左 1 語を削除
［Ctrl］＋［Delete］キー	カーソルの右 1 語を削除
文字列を指定して［Back Space］キー	指定した文字列を削除
文字列を指定して［Delete］キー	指定した文字列を削除

(4) 元に戻す

1. ［編集］メニューから［元に戻す］を選ぶ。
   直前の編集操作を取り消し，その操作をする前の状態に戻ることができます。

(5) やり直し

1. ［編集］メニューから［やり直し］を選ぶ。
   ［元に戻す］で取り消した操作を取り消すことができます。［元に戻す］の操作の直後にだけ有効です。

(6) 標識領域の指定

選択した標識領域にある文字を，指定した文字に置換します。

1. 置換したい標識領域の文字列を選択状態にし，［ツール］メニューの［標識領域の指定］から［注記:＊］［注記:／］［継続:－］［デバッグ:D］［空白:　］のどれかを選ぶ。

   標識領域の文字が，指定した文字に置換されます。標識領域にタブ文字，または全角空白文字がある場合，半角空白文字に変換されます。

   また，置換したい標識領域の文字列を選択状態にし，右クリックで表示されるポップアップメニューから［標識領域の指定］を選んでも標識領域を指定できます。

(7)　見出し領域の置換

選択した行の見出し領域を，指定した文字に置換します。

1. 見出し領域を置換したい行を選択状態にし，［ツール］メニューから［見出し領域の置換］を選ぶ。

   見出し領域の文字が，指定した文字に置換されます。

   また，見出し領域を置換したい行を選択状態にし，右クリックで表示されるポップアップメニューから［見出し領域の置換］を選んでも見出し領域を置換できます。

なお，置換する文字列は，固定形式ソースダイアログボックスで指定します。詳細は，「4.10.8　COBOLソースの設定」を参照してください。

(8)　大文字に変換／小文字に変換

英文字を大文字，または小文字に変換します。

大文字に変換する場合

1. 変換したい文字列を選び，［編集］メニューから［大文字に変換］を選ぶ。

   選択した文字列に含まれる英文字の部分が大文字に変換されます。

小文字に変換する場合

1. 変換したい文字列を選び，［編集］メニューから［小文字に変換］を選ぶ。

   選択した文字列に含まれる英文字の部分が小文字に変換されます。

(9)　インデント

複数の行を選択して，一括してインデントを設定できます。

インデントの設定方法を次に示します。

1. インデントを設定したい行を選ぶ。

   複数の行を選択できます。

2. ［ツール］メニューから［選択行の右シフト］，または［選択行の左シフト］を選ぶ。

   タブ幅分だけ行が左右に移動します。

また，［オプション］メニューの［カスタマイズ］を選ぶと表示されるカスタマイズダイアログボックスの［編集］タブを選び，「選択行に対するタブによるインデント」チェックボックスをオンにしておくと，次のキー操作でもインデントを設定できます。

表 4-11　［Tab］キーによるインデント設定

キー操作	機能
［Tab］キー	選択行の右シフト
［Shift］＋［Tab］キー	選択行の左シフト

詳細は，「4.10.1　編集動作の設定」を参照してください。

補足事項

インデントは，各行の開始位置に半角空白文字が挿入，または削除されることで設定されます。なお，行の開始位置は，固定形式正書法の場合 7 カラム目，自由形式正書法の場合 1 カラム目です。

(10)　ファイルの挿入

1. ［編集］メニューから［ファイルの挿入］を選ぶ。

   挿入するソースファイルを指定するダイアログボックスが表示されるので，ファイルを指定してください。

   指定したソースファイルのすべての内容が，カーソルの位置に挿入されます。

注意事項

特殊文字を含んだファイルは挿入できません。このようなファイルを挿入しようとした場合，エラーメッセージが表示されます。

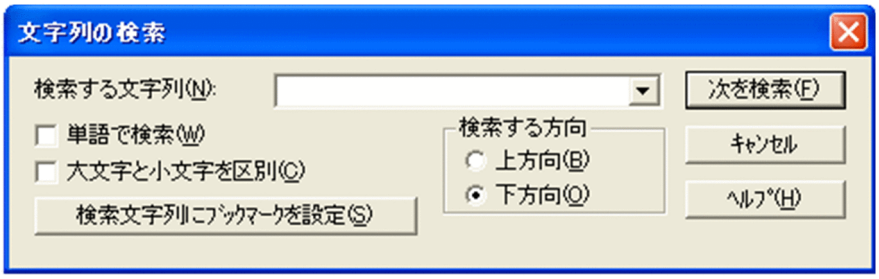
4.7　文字列の検索

4.7.1　検索

(1)　編集中のファイルから検索する

編集中のソースファイルから文字列を検索する操作について説明します。

1. COBOL エディタウィンドウの［検索］メニューから［文字列の検索］を選ぶ。
   文字列の検索ダイアログボックスが表示されます。



2. 検索する文字列を入力する。
   以前に検索対象に指定した文字列をドロップダウンリストから選ぶこともできます。

3. 次の項目を指定する。

   単語で検索

   単語すべてが一致するものだけを検索したいとき指定します。

   大文字と小文字を区別

   大文字と小文字を区別して検索したいとき指定します。

   検索文字列にブックマークを設定

   検索した文字列にブックマークを設定したいとき指定します。

   検索する方向

   上下どちらの方向に検索するのかを指定します。

4. ［次を検索］ボタンを選ぶ。
   編集中のファイルが検索され，指定した検索対象と一致する最初の文字列が選択状態になります。一致する文字列がない場合は，メッセージダイアログボックスが表示されます。

検索対象と一致する最初の文字列が検索されたあと，ツールバーの［次を検索］ボタンを選ぶと，次に一致する文字列が検索されます。

(2)　複数のファイルから検索する

複数のソースファイルから文字列を検索する操作について説明します。

1. COBOL エディタウィンドウの［検索］メニューから［ファイルから検索］を選ぶ。

   ファイルから検索ダイアログボックスが表示されます。



2. 検索する文字列を入力する。

   以前に検索対象に指定した文字列をドロップダウンリストから選ぶこともできます。

3. ファイルの種類を入力する。

   以前に検索対象に指定したファイルの種類をドロップダウンリストから選ぶこともできます。

   複数の種類のファイルから検索したい場合は，セミコロンで区切ってください。

4. フォルダを入力する。

   検索を開始するフォルダを指定します。フォルダのパスを直接入力するか，[...]ボタンからフォルダを選びます。また，以前に指定したフォルダをドロップダウンリストから選ぶこともできます。

5. 次の項目を指定する。

   サブフォルダも検索

   サブフォルダを検索したいとき指定します。

   単語で検索

   単語すべてが一致するものだけを検索したいとき指定します。

   大文字と小文字を区別

   大文字と小文字を区別して検索したいとき指定します。

6. ［検索］ボタンを選ぶ。

   指定したフォルダから指定した種類のファイルが検索され，検索結果がファイルから検索結果ウィンドウに表示されます。

   なお，検索を中断したい場合には，［検索］メニューから［ファイルから検索を中止］を選ぶか，ツールバーの［ファイルから検索を中止］ボタンを押してください。

   指定した検索対象と一致する文字列がある場合は，「ファイル名（行番号）：メッセージテキスト」の形式で表示されます。一致する文字列がない場合は，メッセージが表示されます。

また，ファイルから検索結果ウィンドウ上で開きたいファイル名が記述されている行をダブルクリックすると，タグジャンプできます。タグジャンプの詳細については，「4.7.8　タグジャンプ」を参照してください。

注意事項

複数のファイルの種類を指定して検索を実行する場合，拡張子が短いものから順に指定してください。拡張子が長いものから指定すると，あとに指定したファイルの種類が検索されないことがあります。

（例）.c，.cpp の 2 種類のファイルを検索したい場合

＜正しい指定例＞

```
*.c;*.cpp
```

＜誤った指定例＞

```
*.cpp;*.c
```

上記の誤った指定例の場合，拡張子.c のファイルが検索されません。

4.7.2　置換

編集中のソースファイルの文字列を置換する操作について説明します。

1. COBOL エディタウィンドウの［検索］メニューから［置換］を選ぶ。

   置換ダイアログボックスが表示されます。



2. 検索する文字列と置換後の文字列を入力する。

   以前に検索対象に指定した文字列をドロップダウンリストから選ぶこともできます。

3. 次の項目を指定する。

   単語で検索

   単語すべてが一致するものだけを検索したいとき指定します。

   大文字と小文字を区別

   大文字と小文字を区別して検索したいとき指定します。

検索場所

選択された範囲，およびファイル全体のどちらを検索するのかを指定します。

4. ［次を検索］ボタンで置換先を検索したあと，［置換］ボタンを選ぶ。

   または，［すべて置換］ボタンを選ぶ。

   ただし，「検索場所」で「選択された範囲」を指定した場合は，［すべて置換］ボタンだけを選べます。

4.7.3　ブックマークの設定

ブックマークとは，ソースウィンドウ中の選択マージンに表示される目印です。編集中にブックマークを設定したり，検索した文字列すべてにブックマークを設定したりしておけば，あとでブックマークをたどって検索できます。

ブックマークは，一つのソースファイルに何個でも設定できますが，COBOL エディタの終了後は保存されません。また，COBOL エディタの動作のオプションによって，ブックマークを表示しないで，その行に色を付けることもできます。COBOL エディタのオプションについては，「4.10.2　表示の設定」を参照してください。

次にブックマークの操作について説明します。

(1)　ブックマークの設定と解除

1. COBOL エディタウィンドウの［検索］メニューから［ブックマークの設定／解除］を選ぶ。

   カーソルがある行にブックマークが設定されていない場合は，ブックマークが設定されます。ブックマークが設定されている場合は，ブックマークが解除されます。

   また，［検索］メニューから［ブックマークをすべて解除］を選ぶと，設定されているブックマークをすべて解除できます。

(2)　ブックマークによる検索

1. COBOL エディタウィンドウの［検索］メニューから［次のブックマーク］または［前のブックマーク］を選ぶ。

   次のブックマークまたは前のブックマークがある行にテキストカーソルが移動します。

4.7.4　ジャンプ

指定した行にジャンプする操作について説明します。

ジャンプ機能では，ジャンプ先を行番号，または現在の行番号からの差分で指定します。この行番号を表示するかしないかは，表示のオプションで設定します。表示のオプションについては，「4.10.2　表示の設定」を参照してください。

1. COBOL エディタウィンドウの［検索］メニューから［ジャンプ］を選ぶ。
   ジャンプ先を指定するダイアログボックスが表示されます。

2. ジャンプ先の行番号を入力する。
   - 符号なしの行番号を指定した場合，指定した行番号の行に移動します。
   - 符号付きの行番号を指定した場合，現在の行から指定した行番号の分だけ，行が移動します。

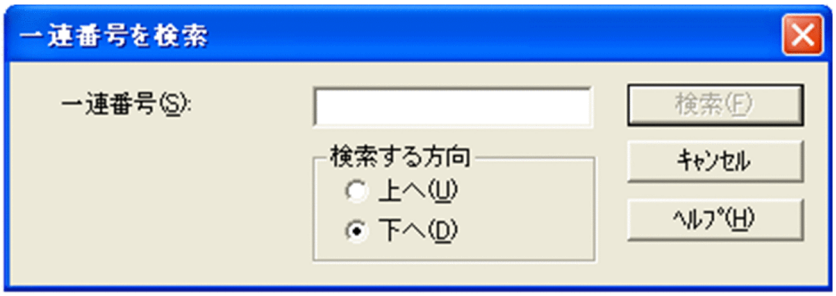
   （例）行番号 300 にテキストカーソルが位置している場合
   ・ジャンプ先に 500 を指定
   行番号 500 に移動します。
   ・ジャンプ先に+500 を指定
   行番号 800 に移動します。

ジャンプ後は，行の先頭にテキストカーソルが移動します。また，入力した行番号は，次回のジャンプ時に表示されます。

4.7.5　一連番号の検索

現在編集中のソースファイルで，指定した一連番号を検索する操作について説明します。

1. COBOL エディタウィンドウの［検索］メニューから［一連番号を検索］を選ぶ。
   一連番号を検索するダイアログボックスが表示されます。



2. 検索する行番号を入力し，検索する方向を指定する。

3. ［検索］ボタンを選ぶ。
   編集中のファイルの一連番号が検索され，一致した文字列が選択状態になります。

4.7.6　不当な一連番号の検索

現在編集中のソースファイルから，不当な一連番号を検索する操作について説明します。

下方向に検索する場合

- COBOL エディタウィンドウの［検索］メニューから［不当な一連番号を検索］を選ぶ。または［F9］キーを押す。

規則

次のどちらかの場合は，不当な一連番号とみなされます。

- 一連番号が昇順になっていない場合
- 一連番号領域に数字以外の文字列が記述されている場合

4.7.7　80 バイトを超える行の検索

現在編集中のソースファイルから，80 バイトを超える行を検索する操作について説明します。

下方向に検索する場合

- COBOL エディタウィンドウの［検索］メニューから［80 バイトを超える行を検索］を選ぶ。または［F10］キーを押す。

上方向に検索する場合

- ［Shift］＋［F10］キーを押す。

編集中のファイルが検索され，1 行の長さが 80 バイトを超える行にカーソルが位置づけられます。

4.7.8　タグジャンプ

タグジャンプの操作について説明します。

1. タグジャンプできる書式で記述されているタグファイル（.tag）を開く。

   タグファイルウィンドウが表示されます。

   タグファイルウィンドウの代わりに，ファイルから検索結果ウィンドウを使ってタグジャンプすることもできます。

   タグジャンプできる書式を次に示します。

```
絶対パスファイル名,line 行番号:メッセージテキスト
"絶対パスファイル名",line 行番号:メッセージテキスト
絶対パスファイル名(行番号):メッセージテキスト
```

2. タグファイルウィンドウ上で，開きたいファイル名が記述されている行をダブルクリックする。

   ファイルが開かれ，指定した行にカーソルが位置づけられます。

   なお，タグファイルウィンドウでは次に示す操作ができます。

表 4-12　タグファイルウィンドウでの操作

操作	機能
ダブルクリック	タグジャンプできる書式のファイルを開く
ポップアップメニューの［タグジャンプ］	タグジャンプできる書式のファイルを開く
［Enter］キー	タグジャンプできる書式のファイルを開く
［↑］キー	選択行を上に移動する
［↓］キー	選択行を下に移動する

4.8　キーボードマクロ

COBOL エディタでは，キーボードからの編集操作の組み合わせをマクロとして登録しておき，あとで何度でも実行できます。この機能を使うと，繰り返しの多い編集作業を効率良く実行できます。

ここでは，キーボードマクロの使用方法について説明します。

4.8.1　キーボードマクロの登録

キーボード操作をキーボードマクロとして登録する方法を，次に示します。

1. ［ツール］メニューから［キーボードマクロ登録開始］を選ぶ。
   キーボードマクロの登録が開始します。
   キーボードマクロの登録中は，COBOL エディタウィンドウのステータスバーに「マクロ登録中・・・」のメッセージが表示されています。
2. マクロに記録したいキーボード操作を順に実行する。
3. ［ツール］メニューから［キーボードマクロ登録終了］を選ぶ。
   キーボードマクロの登録が終了します。

注意事項

- キーボードマクロの登録中に，ほかのソースウィンドウに編集を切り替えた場合は，キーボードマクロの登録が中止されます。この場合，中止された時点までのキーボード操作がキーボードマクロとして登録されます。
- キーボードマクロの登録を開始すると，以前に登録したキーボードマクロは消去されます。
- COBOL エディタを終了すると，登録したキーボードマクロは消去されます。

4.8.2　キーボードマクロの実行

登録したキーボードマクロを実行する方法を，次に示します。

1. ［ツール］メニューから［キーボードマクロ実行］を選ぶ。
   登録したキーボードマクロが実行されます。

4.8.3　キーボードマクロに登録できるキー操作

キーボードマクロに登録できるキー操作を，次に示します。

なお，マウスを使った操作やダイアログボックスを表示する操作は，キーボードマクロに登録できません。

(1) 編集に使用するキー

- 文字入力に使用するキー
- 日本語入力に使用するキー

また，次の編集に使用するキーを登録できます。

[Enter]，[Tab]，[BackSpace]，[Delete]，[Insert]

(2) 移動キー

次の移動キーを登録できます。

[←]，[→]，[↑]，[↓]，[Ctrl] ＋ [←]，[Ctrl] ＋ [→]，[Ctrl] ＋ [↑]，[Ctrl] ＋ [↓]，[PageUp]，[PageDown]，[Ctrl] ＋ [PageUp]，[Ctrl] ＋ [PageDown]，[Home]，[End]，[Ctrl] ＋ [Home]，[Ctrl] ＋ [End]

(3) 選択キー

次の選択キーを登録できます。

[Shift] ＋ [←]，[Shift] ＋ [→]，[Shift] ＋ [↑]，[Shift] ＋ [↓]，[Shift] ＋ [Ctrl] ＋ [←]，[Shift] ＋ [Ctrl] ＋ [→]，[Shift] ＋ [PageUp]，[Shift] ＋ [PageDown]，[Shift] ＋ [Ctrl] ＋ [PageUp]，[Shift] ＋ [Ctrl] ＋ [PageDown]，[Shift] ＋ [Home]，[Shift] ＋ [End]，[Shift] ＋ [Ctrl] ＋ [Home]，[Shift] ＋ [Ctrl] ＋ [End]

(4) ショートカットキー，ファンクションキー

次のショートカットキー，およびファンクションキーを登録できます。

表 4-13 ショートカットキー，およびファンクションキーの機能

キー操作	機能（対応するメニュー）
[Ctrl] ＋ [X] キー	切り取り
[Ctrl] ＋ [C] キー	コピー
[Ctrl] ＋ [V] キー	貼り付け
[Ctrl] ＋ [U] キー	大文字に変換
[Ctrl] ＋ [L] キー	小文字に変換
[F2] キー	次のブックマーク
[Shift] ＋ [F2] キー	前のブックマーク
[F3] キー	次を検索
[Shift] ＋ [F3] キー	前を検索

4.9　構文の入力を支援する機能

COBOL エディタには，次のような構文の入力を支援する機能が用意されています。

構文チェック

構文が正しく使用されているかをチェックします。

キーワード補完

途中まで入力したキーワードから COBOL の予約語，サービスルーチン名，ユーザキーワードなどに対して，残りのスペルを補います。

ここでは，これらの機能について説明します。

4.9.1　構文チェック

現在編集しているソースファイルの構文を COBOL の言語規則に基づいてチェックします。

1. 構文をチェックしたいソースファイルをアクティブにする。
2. ［構文］メニューから［構文チェック］を選ぶ。

   ソースファイルの構文がチェックされ，構文チェック結果ウィンドウに結果が表示されます。

   また，構文チェック結果ウィンドウ上のエラーメッセージをダブルクリックすると，該当する行にタグジャンプができます。タグジャンプの詳細は，「4.7.8　タグジャンプ」を参照してください。

   なお，構文チェック中に処理を中止したいときは，［構文］メニューから［構文チェックの中止］を選ぶか，ツールバーの［構文チェックの中止］ボタンを選びます。

注意事項

- 構文をチェックできるのは，ccbl2002 コマンドが対象にしているソースファイルだけです。
- 構文チェック時には，リポジトリファイルを参照しないため，実際のコンパイル時とメッセージの表示が変わる場合があります。

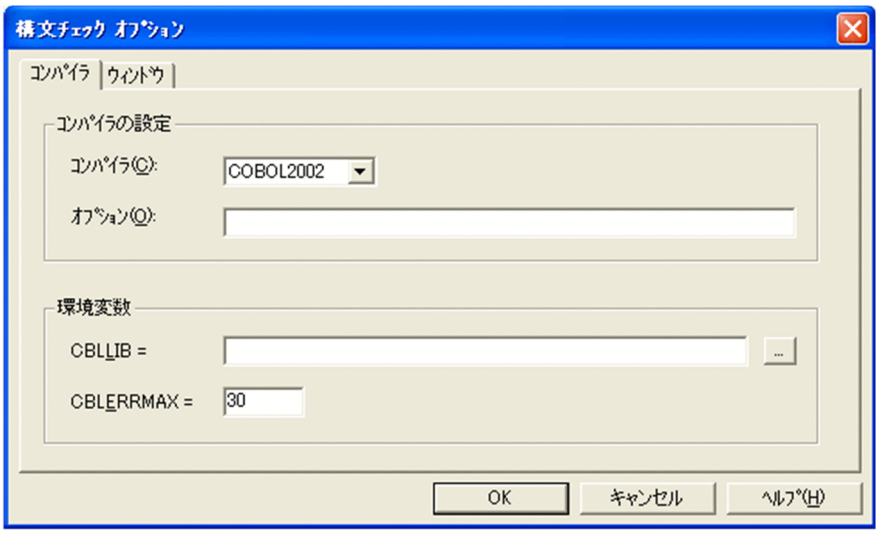
4.9.2　構文チェックの環境設定

構文をチェックするときのコンパイル環境や，構文チェック機能のウィンドウなどを設定します。

1. ［構文］メニューから［構文チェック オプション］を選ぶ。

   構文チェック オプションダイアログボックスが表示されます。

(1)　構文チェックに適用するコンパイラや環境変数を設定する



(a)　コンパイラの設定

コンパイラ

構文チェック時に使用する COBOL2002 コンパイラを示します。

オプション

構文チェック時に仮定するコンパイラオプションを指定します。コンパイラオプションについては，「2.3　コンパイラオプション」を参照してください。

複数のコンパイラオプションを指定する場合は，空白文字で区切って指定してください。

なお，-Main,System，-Main,V3 オプションを指定する場合は，いちばん最後に指定してください。

注意事項

コンパイラオプションの指定が誤っていた場合，正しく構文チェックできないことがあります。

(b)　環境変数の設定

CBLLIB

構文をチェックしたいソースファイルが参照する，登録集原文を検索するフォルダを指定します。複数のフォルダを指定する場合は，セミコロンで区切って指定してください。

「%CBLLIB%」を指定すると，システム環境変数に登録しておいた CBLLIB の値を引用できます。

CBLERRMAX

構文チェックを打ち切る S レベルエラーの数を指定します。指定できる値の範囲は 0～999,999 です。

注意事項

構文チェック時には，次のように環境変数の値が仮定されます。各環境変数の詳細については，「2.4.3 コンパイラ環境変数の詳細」を参照してください。

- CBLTAB

カスタマイズダイアログボックスの［編集］タブを選んで表示されるページで設定されたタブ幅が仮定されます。

- CBLFIX

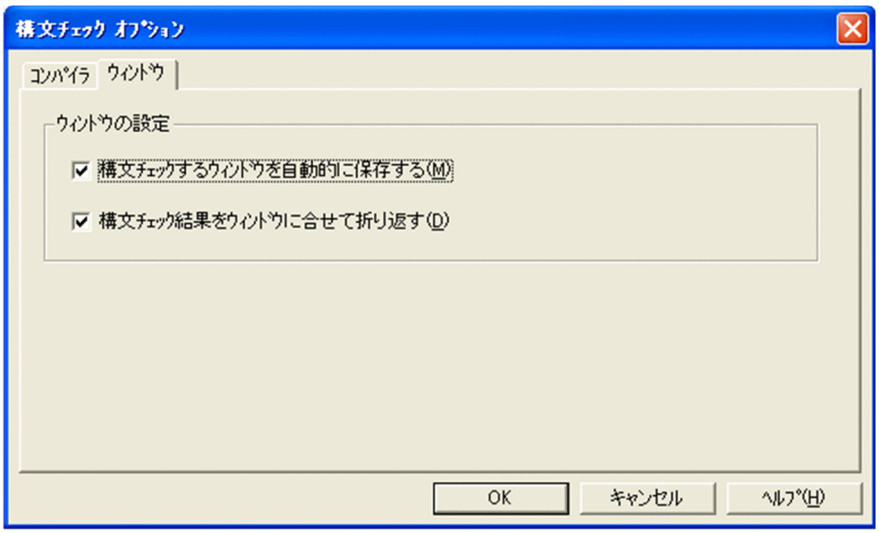
  カスタマイズダイアログボックスの［ファイルの種類］タブを選んで表示されるページの「COBOLソース（固定形式正書法）」に登録された内容が仮定されます。

- CBLFREE

  カスタマイズダイアログボックスの［ファイルの種類］タブを選んで表示されるページの「COBOLソース（自由形式正書法）」に登録された内容が仮定されます。

なお，上記以外の環境変数については，COBOL エディタを起動した時点での設定内容がコンパイラに渡されます。

(2)　ソースファイルの保存やチェック結果のウィンドウを設定する



構文チェックするウィンドウを自動的に保存する

編集中のソースファイルを構文チェックするとき，自動的にソースファイルを保存したいとき指定します。なお，このチェックボックスをオフにしてからソースファイルを編集した場合，編集内容を保存してファイルの内容を更新しないと，構文チェックは実行できません。

構文チェック結果をウィンドウに合せて折り返す

構文チェックの結果を，ウィンドウ幅に合わせて行を折り返して表示したいときに指定します。

4.9.3　キーワード補完

キーワード補完とは，途中まで入力された COBOL のキーワードに，自動的に残りを補ってキーワードを完成する機能です。このとき，入力したテキストだけでキーワードを特定できない場合は，複数の候補がリストに表示されます。また，何も入力しないでこの機能を使用すると，COBOL の予約語（ただし，アスタリスクなどの特殊文字や文脈依存語は含まない），COBOL サービスルーチン，ユーザキーワードのすべての COBOL のキーワードがリストに表示されます。

COBOL エディタには，次の 2 種類のキーワード補完機能があります。

- キーワード補完

  すべての COBOL のキーワードを対象にしたキーワード補完機能です。

- ユーザキーワード補完

  ユーザキーワードだけを対象にしたキーワード補完機能です。なお，ユーザキーワード補完を使用するには，カスタマイズダイアログの［予約語ファイル］タブを選び，使用するシステム属性に合わせて，ユーザキーワードを登録しなければなりません。

キーワード補完

キーワード補完の操作方法を次に示します。

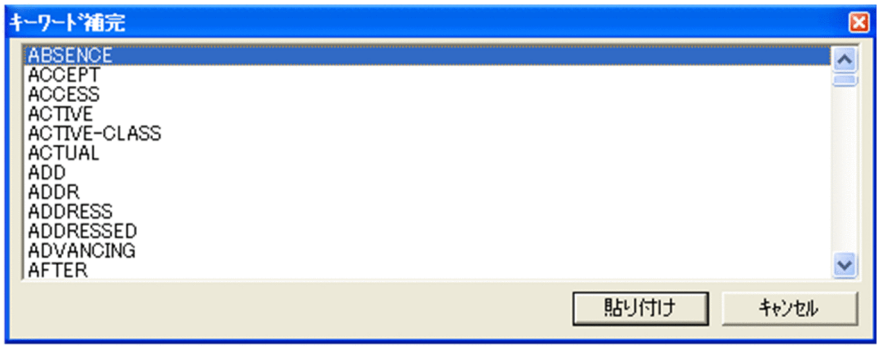
1. **COBOL のキーワードを途中まで入力する。**

   カーソルの前後が空白のときは，何も入力されていないとみなされます。

2. **[Ctrl] ＋ [E] キーを押す。**

   ［構文］メニューから［キーワード補完］を選ぶこともできます。

   複数の候補がある場合は，次のダイアログボックスが表示されます。



3. **目的のキーワードを選ぶ。**

   キーワードが完成されます。

ユーザキーワード補完

ユーザキーワード補完の操作方法を次に示します。

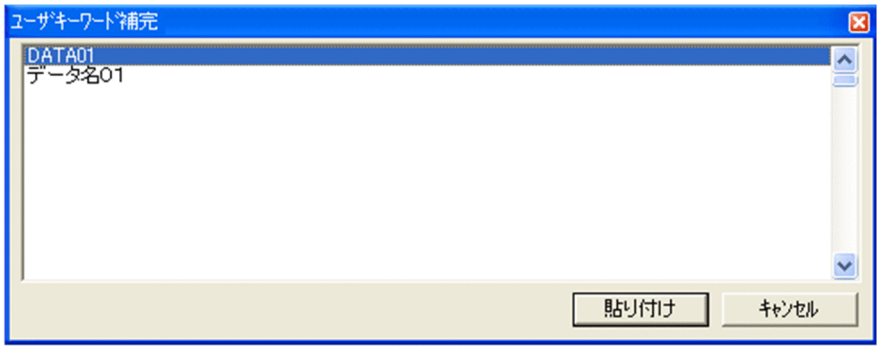
1. **ユーザキーワードを途中まで入力する。**

   カーソルの前後が空白のときは，何も入力されていないとみなされます。

2. **[Ctrl] ＋ [K] キーを押す。**

   ［構文］メニューから［ユーザキーワード補完］を選ぶこともできます。

   複数の候補がある場合は，次のダイアログボックスが表示されます。



3. 目的のキーワードを選ぶ。

   キーワードが完成されます。

補足事項

COBOL のキーワードとは，COBOL の予約語，サービスルーチン，およびユーザが設定したキーワードのことです。これらのキーワードは，カスタマイズダイアログボックスの「予約語ファイル」タブを選んで表示されるページで設定できます。詳細は，「4.10.7　予約語の設定」を参照してください。

4.10　オプション

COBOL エディタでは，編集時の操作や表示についてオプションを設定できます。これらのオプションは，カスタマイズダイアログボックスで，設定したい内容に合わせて［編集］［表示］［色］［画面フォント］［ファイルの種類］［保存］［予約語］［COBOL ソース］のタブを選び，表示されるページで設定します。

次に，オプションを設定する操作を説明します。

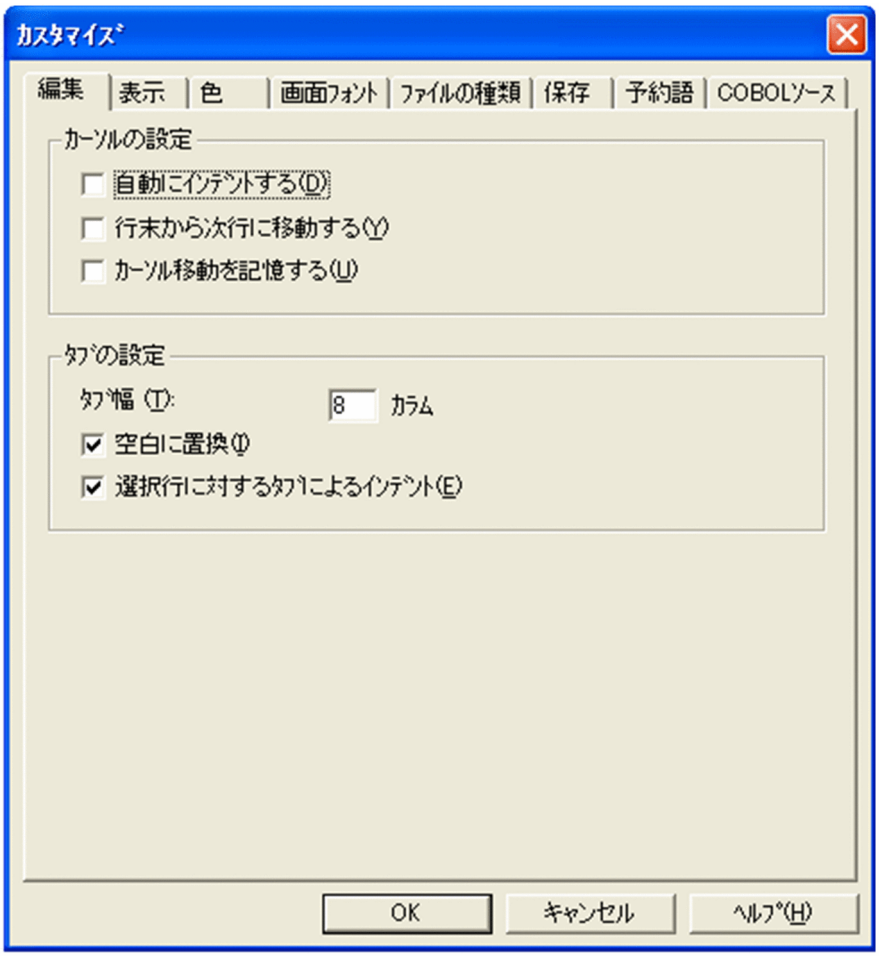
1. COBOL エディタウィンドウの［オプション］メニューから［カスタマイズ］を選ぶ。
   カスタマイズダイアログボックスが表示されます。

2. 設定したい内容に関連するタブを選び，表示されるページでオプションを設定し，［OK］ボタンを選ぶ。

次に，各タブを選んで表示されるページから設定するオプションについて説明します。

4.10.1　編集動作の設定

［編集］タブを選んで表示されるページを次に示します。ここでは，カーソル，およびタブについて設定します。



(1)　編集タブの説明

(a) カーソルの設定

自動にインデントする

自動的にインデントを揃えたいとき指定します。自動インデントを指定した場合は，新しい行は，前の行と同じカラム位置から始まります。

行末から次行に移動する

カーソルを行末から次の行頭まで移動したいとき指定します。チェックボックスをオンに指定した場合，改行文字より右側の位置にカーソルがあるとき［→］キーを押すと次の行の行頭に，行頭にカーソルがあるとき［←］キーを押すと前の行の行末に，カーソルが移動します。

カーソル移動を記憶する

カーソルの移動を，［元に戻す］による取り消し操作の対象にしたいとき指定します。

(b) タブの設定

タブ幅

タブ幅を入力します。1～80 の値で指定してください。

空白に置換

入力したタブを，空白に変換する場合に指定します。

選択行に対するタブによるインデント

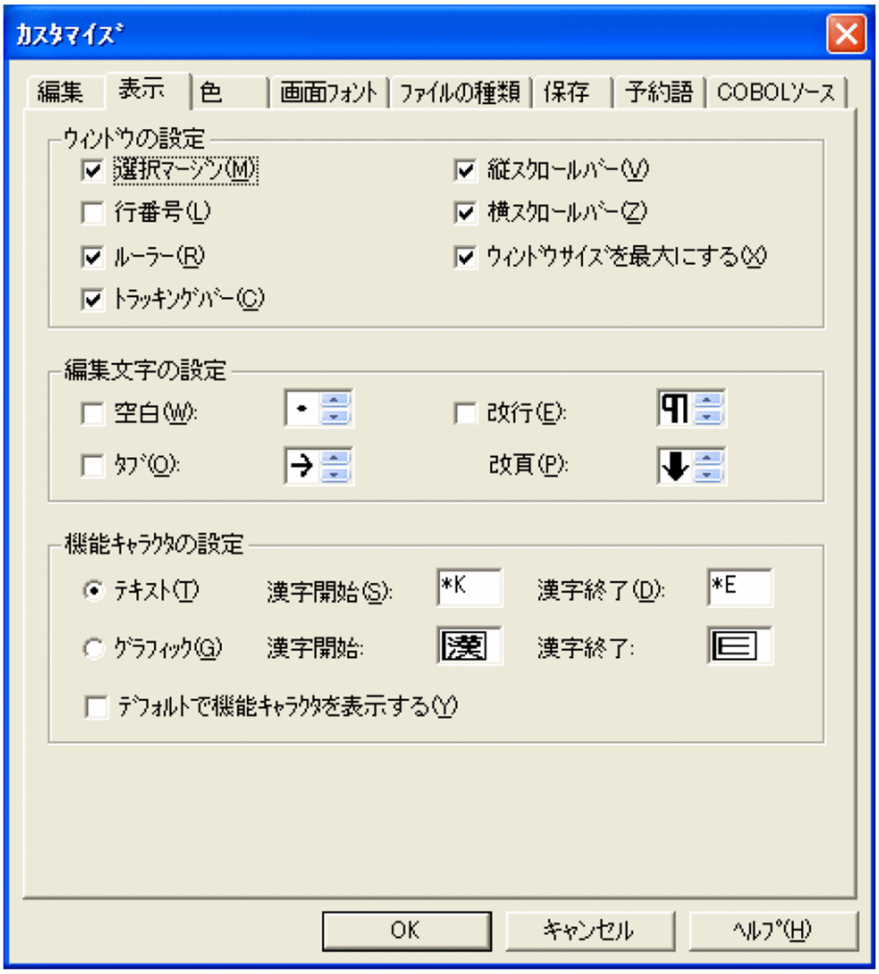
［Tab］キーを使って選択した行にインデントを設定する機能を使用する場合，指定します。インデントの詳細については，「4.6.2　編集操作」の「(9)　インデント」を参照してください。

注意事項

VOS3 COBOL85 では，ソースファイル中にタブ文字を使用できません。VOS3 COBOL85 用のソースファイルを編集する場合，「空白に置換」を必ず指定してください。

4.10.2　表示の設定

［表示］タブを選んで表示されるページを次に示します。ここでは，ウィンドウ，および編集文字の表示について設定します。



(1)　表示タブの説明

(a) ウィンドウの設定

選択マージン

選択マージン（行の左余白）を表示したいとき指定します。

選択マージンを表示した場合，ブックマークの設定されている行の選択マージンに四角いマークが表示されます。

選択マージンを非表示にした場合，ブックマークの設定されている行が，カラーオプションで指定した色で表示されます。

行番号

行番号を表示したいとき指定します。行番号は，選択マージンの右に 6 けたで表示されます。

ルーラー

ルーラーを表示したいとき指定します。

トラッキングバー

トラッキングバーを表示したい場合に指定します。ただし，ルーラーが指定されていない場合は，トラッキングバーを指定できません。

縦スクロールバー

縦スクロールバーを表示したいとき指定します。

横スクロールバー

横スクロールバーを表示したいとき指定します。

ウィンドウサイズを最大にする

ファイルを新規作成したり，開いたりした場合に，ソースウィンドウを自動的に最大化したいときに指定します。

ただし，タグファイル，ファイルからの検索結果，または構文チェック結果のウィンドウを開いている場合は，最大化されません。

(b) 編集文字の設定

空白

空白を表示したいとき指定します。表示形式はリストから選べます。

タブ

タブを表示したいとき指定します。表示形式はリストから選べます。

改行

改行コードを表示したいとき指定します。表示形式はリストから選べます。

改頁

改頁コードの表示形式をリストから選びます。

(c) 機能キャラクタの設定

テキスト，グラフィック

機能キャラクタの「漢字開始」および「漢字終了」を表す文字を，2バイトの文字列またはグラフィック文字で指定します。デフォルトでは，「漢字開始」に「*K」，「漢字終了」に「*E」が設定されています。

なお，機能キャラクタの表示／非表示は，ファイルのプロパティダイアログボックスで確認できます。詳細は，「4.4.5　ファイル情報の参照」を参照してください。

（機能キャラクタの表示規則）

機能キャラクタを表す文字は，次の規則に従って表示されます。

- 機能キャラクタは，2バイト文字（日本語文字）の前後に表示されます。
- 「漢字開始」「漢字終了」を表す文字は，2バイトで表示されます。
- 「漢字開始」を表す文字は，2バイト文字列の先頭文字の直前に表示されます。
- 「漢字終了」を表す文字は，2バイト文字列の最後の文字の直後に表示されます。ただし，2バイト文字のあとに半角空白文字，タブ文字，または改行文字が続いている場合，それらの文字の直後に表示されます。

- 半角空白文字，またはタブ文字は，全角文字の終了とみなされません。したがって，2 バイト文字の間に半角空白文字，またはタブ文字がある場合，これらの文字の間に機能キャラクタは表示されません。
- 2 バイト文字のあと，改行文字の前に半角空白文字がある場合，2 バイト文字の直後に「漢字終了」を表す文字が表示されます。

デフォルトで機能キャラクタを表示する

機能キャラクタを表示させたいとき指定します。Windows 上の COBOL ファイルを VOS3 COBOL85 のファイルとして開きたいとき，このオプションを指定しておくことで機能キャラクタを表示できます。

なお，アクティブ状態のソースウィンドウに対して，このオプションは有効になりません。

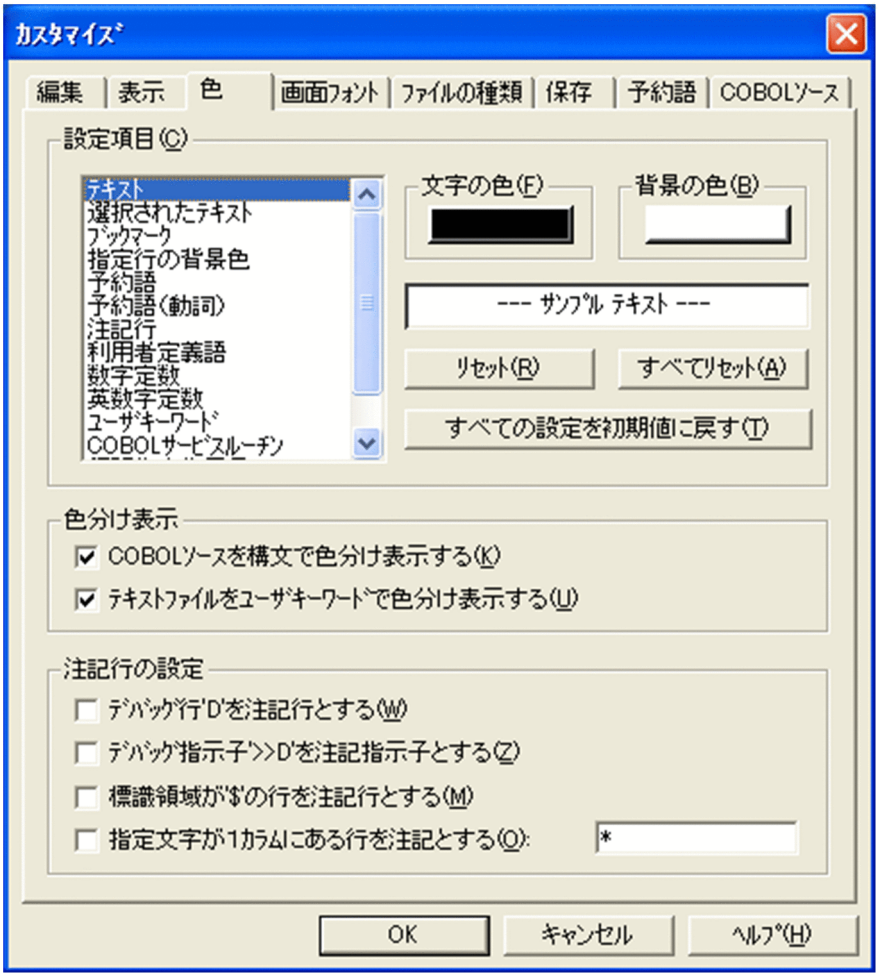
アクティブ状態のソースウィンドウに対してシステム属性を指定したい場合は，ファイルのプロパティダイアログボックスで指定します。

詳細は，「4.4.5　ファイル情報の参照」を参照してください。

デフォルトは，機能キャラクタが表示されない状態です（チェックなし）。

4.10.3　色の設定

［色］タブを選んで表示されるページを次に示します。ここでは，ソースウィンドウ中の項目の色を設定します。色を設定できる項目は，標準のテキスト，選択されたテキスト，ブックマーク，および COBOL 構文上の用語です。ただし，色分け表示できるのは，編集ファイルが COBOL の形式か，ファイルタイプオプションで設定した拡張子のときだけです。



(1)　色タブの説明

1. 設定項目をリストから選ぶ。

   COBOL の語を色分け表示するためには，「COBOL ソースを構文で色分け表示する」を指定して，リストに COBOL 構文上の用語を表示させる必要があります。テキストファイル中の，COBOL2002 ユーザキーワードに登録した文字列と一致する文字列をユーザキーワードに指定した色で表示するためには，「テキストファイルをユーザキーワードで色分け表示する」を指定する必要があります。

2. [文字の色] ボタンまたは [背景の色] ボタンを選ぶ。

   色の設定ダイアログボックスが表示されます。

3. 文字および背景の色をパレットから選ぶ。

   サンプルの表示が指定した色に変わります。

   なお，[リセット] ボタンまたは [すべてリセット] ボタンを使うと，設定した色を無効にできます。また，[すべての設定を初期値に戻す] ボタンで，設定した色を無効にしてデフォルトに戻せます。

(a) 設定項目

文字の色

任意の色から文字の色を設定します。

背景の色

任意の色から背景の色を設定します。

(b) 色分け表示

COBOL ソースを構文で色分け表示する

COBOL ソースを COBOL 言語の構文に合わせて色分け表示をします。

テキストファイルをユーザキーワードで色分け表示する

テキストファイル中の，COBOL2002 ユーザキーワードに登録した文字列と一致する文字列をユーザキーワードに指定した色で表示します。区切り文字は COBOL ソースの色分け表示に従います。

(c) 注記行の設定

デバッグ行やデバッグ指示子，標識領域（7 カラム目）の'$'文字，注記行とする 1 カラム目の指定文字のそれぞれを注記行とみなし，色分け表示するかどうかを指定します。

デバッグ行'D'を注記行とする

固定形式正書法の COBOL ソースの標識領域（7 カラム目）にデバッグ行'D'または'd'のある行を注記行とするときに指定します。このオプションによって-DebugLine コンパイラオプションを指定した場合と同じ解釈で，注記行が色分け表示されます。

デバッグ指示子'>>D 'を注記指示子とする

COBOL ソースのデバッグ指示子'>>D 'を注記指示子とするときに指定します。このオプションによって-DebugLine コンパイラオプションを指定した場合と同じ解釈で，注記行が色分け表示されます。

標識領域が'$'の行を注記行とする

固定形式正書法の COBOL ソースの標識領域（7 カラム）に'$'文字がある行を注記行とするときに指定します。このオプションによって，-CmDol コンパイラオプションを指定した場合と同じ解釈で，注記行が色分け表示されます。

指定文字が 1 カラムにある行を注記とする

固定形式正書法の COBOL ソースの一連番号領域である 1 カラム目に指定した文字がある行を注記とするときに指定します。この項目を指定した場合，-CmAster コンパイラオプションを指定した場合と同じ解釈で，注記行が色分け表示されます。この項目を指定した場合，指定文字が 1 カラム目にある行を注記とみなし，その行を一連番号の指定の対象としません。

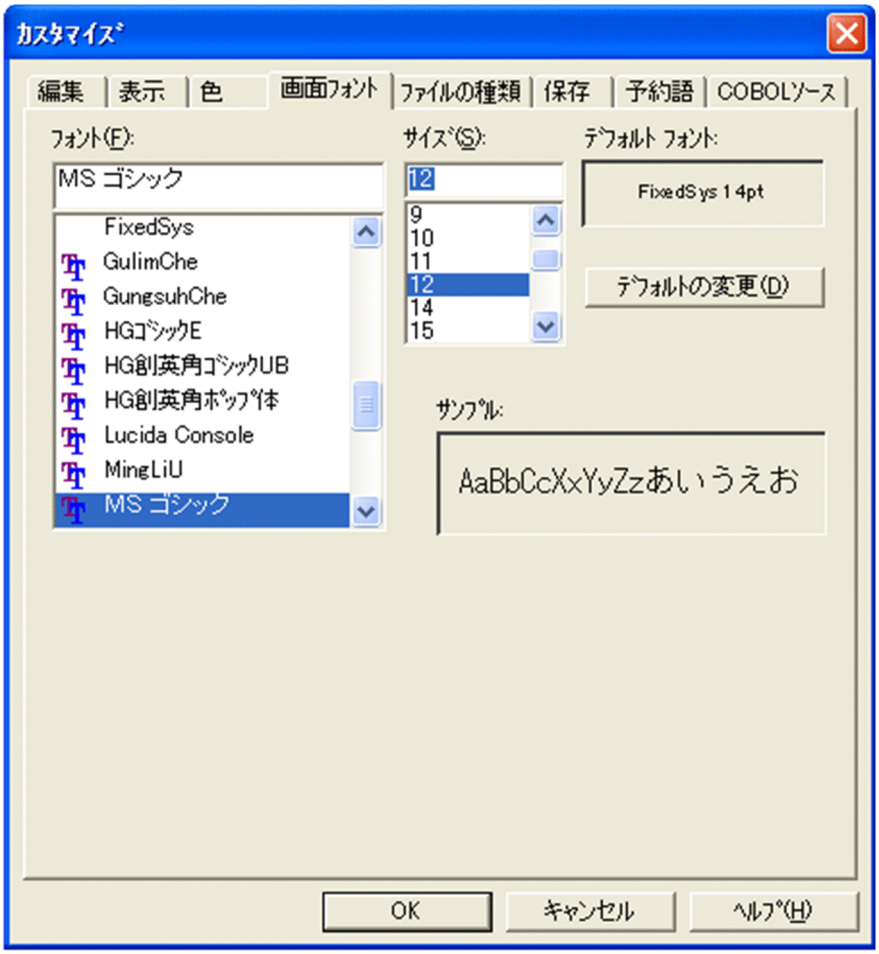
指定文字には初めて COBOL エディタを起動した場合，'*'が設定されます。指定文字には 1 バイトの文字を 10 文字まで指定できます。英小文字を入力した場合は，自動的に英大文字に変換する。半角空白，タブ文字および全角文字を入力してはなりません。

注意事項

注記の設定の指定をしても構文チェックには有効となりません。構文チェックのコンパイラオプションは構文チェックオプションダイアログボックスの「コンパイラ」ページの「コンパイラの設定」にある「オプション」に COBOL2002 コンパイラのオプションを指定してください。

4.10.4　画面フォントの設定

[画面フォント] タブを選んで表示されるページを次に示します。ここでは，ソースウィンドウ中のテキストのフォントおよびサイズを設定します。ここで設定したフォントオプションは，アクティブ状態のソースウィンドウだけに有効です。



(1)　画面フォントタブの説明

1. フォントリストから設定したいフォントを選ぶ。

   指定したフォントで表示できるサイズがサイズリストに表示されます。

   なお，日本語を持たないフォントを指定したときは日本語が正しく表示されません。

2. サイズリストから設定したいサイズを選ぶ。

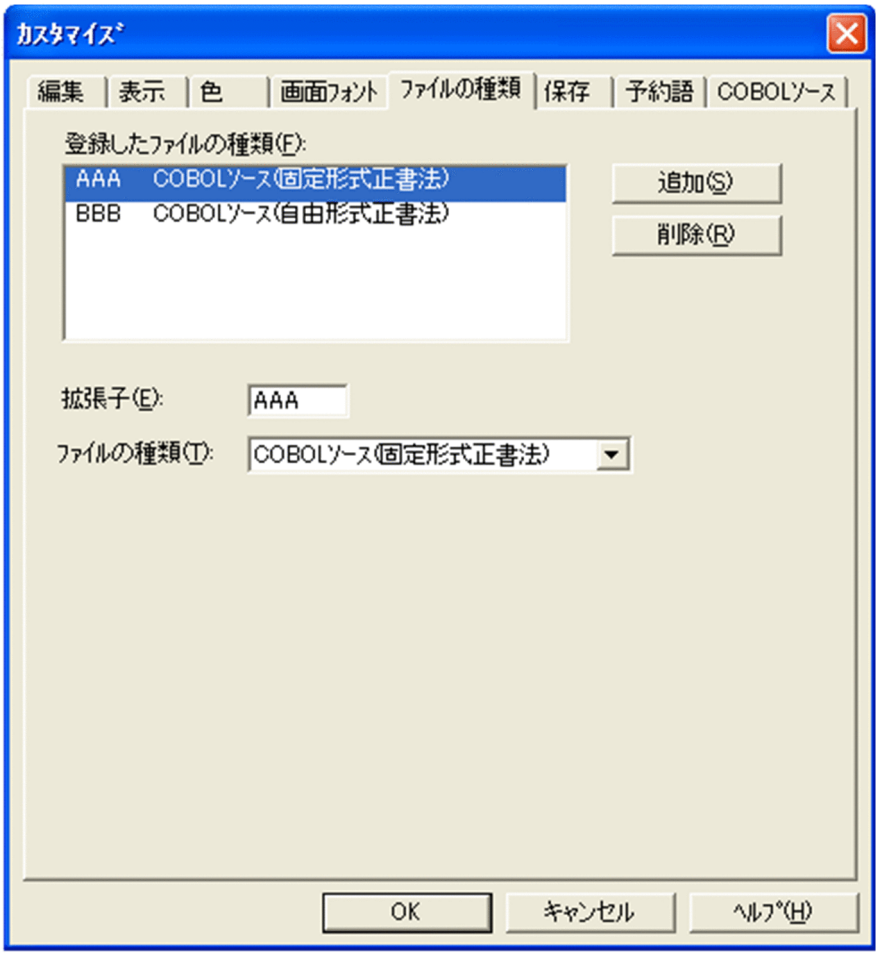
   サンプルの表示が指定したフォントおよびサイズに変わります。

   なお，[デフォルトの変更] ボタンで，指定したフォントをデフォルトにできます。デフォルトは，新しく開くすべてのソースウィンドウに有効です。

4.10.5　ファイルの種類の設定

［ファイルの種類］タブを選んで表示されるページを示します。ここでは，COBOL ソースの形式として認識させたい拡張子を登録します。

COBOL エディタでは，編集ファイルが COBOL ソースの形式の場合だけ，テキストを色分けして表示できます。ただし，ユーザが指定した拡張子でも COBOL ソースの形式として認識させることによって，テキストの色分け表示などの COBOL ソース固有の表示ができます。



(1)　ファイルの種類タブの説明

1. 任意の拡張子を入力する。

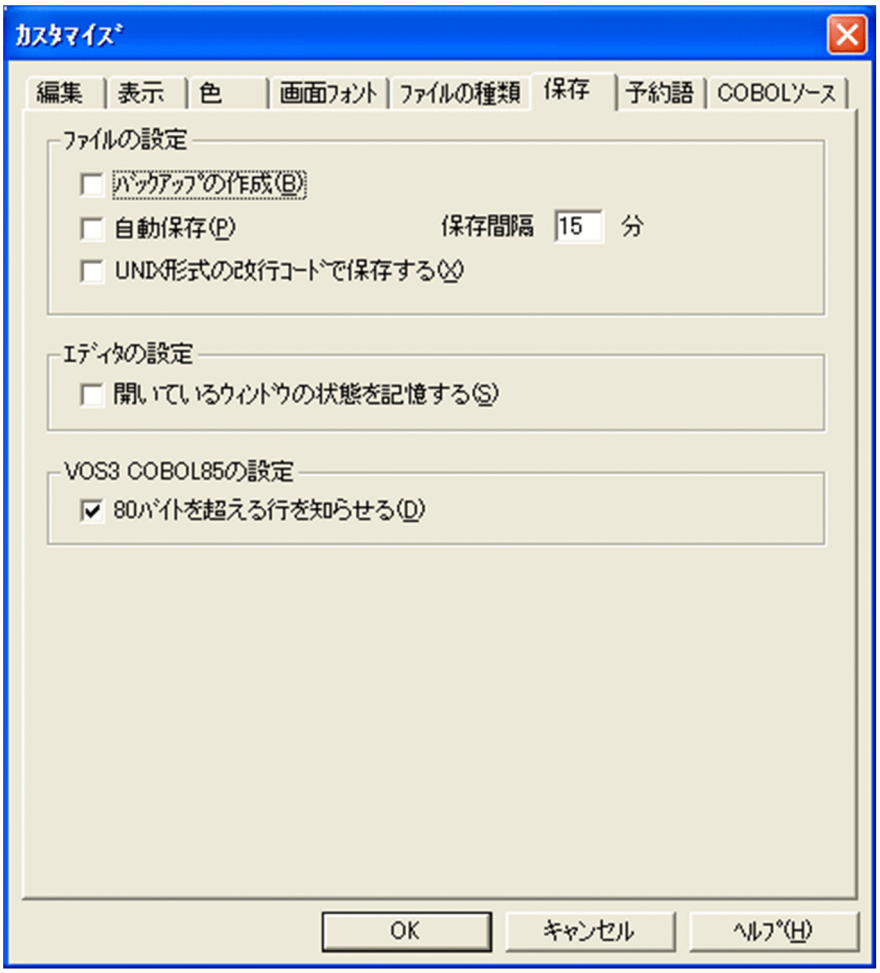
2. ファイルの種類から固定形式か自由形式かを選び，［追加］ボタンを選ぶ。

   指定した拡張子が，COBOL ソースの形式として認識されます。また，指定したファイルの種類（固定／自由形式）に応じてソースウィンドウの表示が変わります。

   なお，COBOL ソースの形式として登録されている拡張子を選び，［削除］ボタンを選ぶと，登録を解除できます。

4.10.6　保存の設定

［保存］タブを選んで表示されるページを次に示します。ここでは，ファイルの保存方法について設定します。



(1)　保存タブの説明

(a) ファイルの設定

バックアップの作成

バックアップを作成したいとき指定します。バックアップは，ソースファイルのオープン時に，同じフォルダにあるバックアップファイル（同じファイル名に拡張子（.bak）を付けたファイル）に保存されます。すでにバックアップファイルがある場合は，そのファイルを上書きします。

自動保存

自動的に保存したいとき指定します。保存する場合は，保存間隔を指定できます。保存先は，同じフォルダにある，元のファイル名の拡張子の最後に「~」（チルダ）を付けたファイルです。元の拡張子が 3 文字の場合は，最後の文字を「~」（チルダ）にしたファイルに保存されます。

UNIX 形式の改行コードで保存する

UNIX 用のソースファイルで保存したいときに指定します。

この項目を指定した場合は，UNIX で一般的な改行コード（X'0A'）が使用されます。指定しない場合は，MS-DOS，Windows で一般的な改行コード（X'0D0A'）が使用されます。

(b) エディタの設定

開いているファイルを記憶する

COBOL エディタの終了時に，開いているファイルの状態を記録しておきたいときに設定します。ファイルの状態を記録しておくと，次に COBOL エディタを起動したときに，前回編集していたファイルが自動的に開かれます。

(c) VOS3 COBOL85 の設定

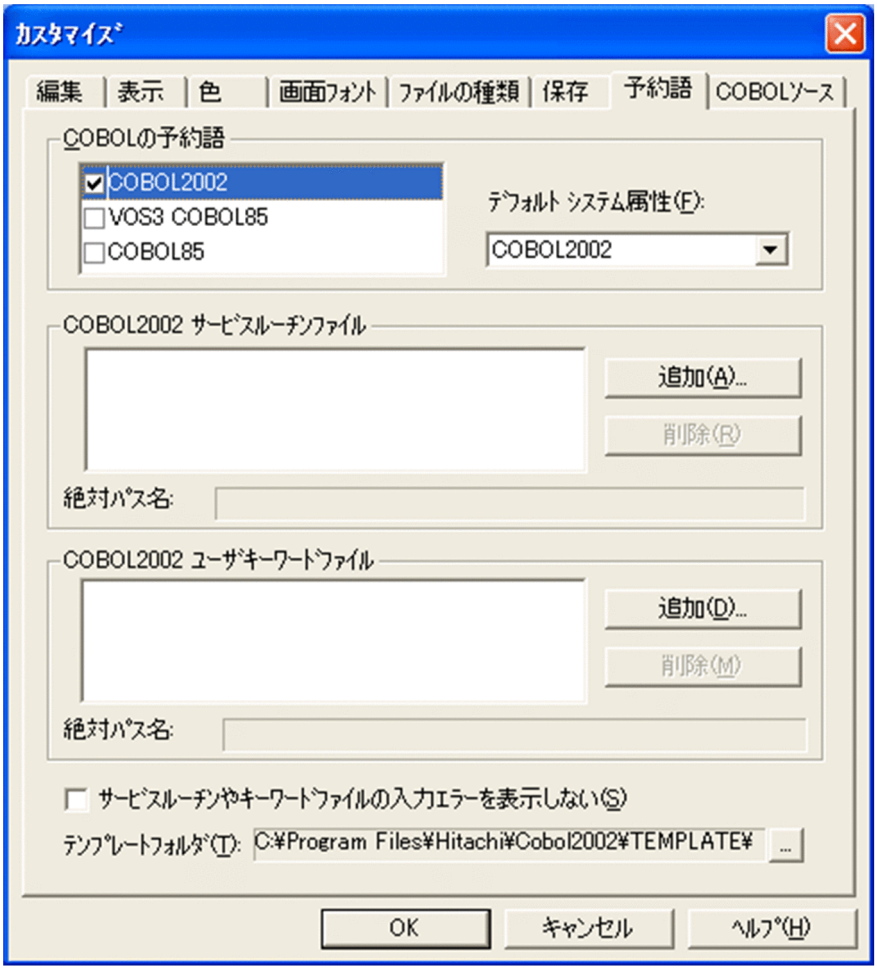
80 バイトを超える行を知らせる

ファイルプロパティダイアログボックスのシステム属性が「VOS3 COBOL85」の場合，COBOL 用のソースファイルの保存時に，80 バイトを超える行をエラーとして検出したいときに指定します。

この項目を指定した場合は，80 バイトを超える行を含んだ COBOL 用ソースファイルを保存しようとしたときに警告のダイアログボックスが表示され，保存するかしないかを指定できます。

4.10.7　予約語の設定

［予約語］タブを選んで表示されるページを次に示します。ここでは，色分け表示やキーワード補完の対象となる COBOL の予約語，COBOL サービスルーチン，およびユーザキーワードの種類を設定します。



(1)　予約語タブの説明

COBOL の予約語

システムごとの COBOL 予約語を使用するかどうかを指定します。この項目をチェックした場合，予約語の色分け表示やキーワード補完ができます。チェックしない場合，予約語は登録されないため利用者定義語として表示されます。チェックを外すことで COBOL エディタの起動時間を短縮したり，メモリ消費量を減らしたりできるため，COBOL 予約語を使用しないシステムはチェックしないことを推奨します。

- COBOL2002
  COBOL2002 の予約語を有効にします。
- VOS3 COBOL85
  メインフレーム（VOS3） COBOL85 の予約語を有効にします。
- COBOL85
  COBOL85 の予約語を有効にします。

デフォルトでは，「COBOL2002」が選択されています。

デフォルト システム属性

ソースウィンドウを開いた場合に選択されるシステム属性のデフォルト値を指定します。開発マネージャから開かれたファイルのときは，開発マネージャの指定が有効となります。指定できる値は COBOL の予約語のシステム属性値です。これによって，COBOL ソースファイルを VOS3 などの COBOL85 言語で記述したファイルとして開くことができます。このオプションはアクティブなウィンドウに対して有効にはなりません。アクティブなウィンドウに対してシステム属性を指定したい場合は，ファイルプロパティダイアログボックスで設定します。

COBOL2002 サービスルーチンファイル

COBOL エディタでキーワードとして有効にするサービスルーチンを，COBOL サービスルーチンファイルの形式で登録します。

COBOL サービスルーチンファイルは，対象となる COBOL の予約語ごとに設定します。対象となる COBOL の予約語は，「COBOL 予約語」で選択してください。

なお，COBOL エディタでは，表 4-14 に示す COBOL サービスルーチンファイルを提供しています。ただし，システム属性が COBOL85 の場合の COBOL サービスルーチンファイルは提供していません。

表 4-14　COBOL サービスルーチンファイル

システム属性	ファイル名	登録内容
COBOL2002	COBOL2002 サービスルーチン Windows 用.svw	COBOL2002 のサービスルーチン名
VOS3 COBOL85	COBOL85 サービスルーチン VOS3 用.svw	メインフレーム（VOS3）COBOL85 のサービスルーチン名

また，上記のファイル以外に，ユーザ独自の COBOL サービスルーチンファイルを作成して COBOL エディタに登録できます。作成方法の詳細は，「(3)　COBOL サービスルーチンファイル，ユーザキーワードファイルの作成」を参照してください。

COBOL2002 ユーザキーワードファイル

COBOL エディタで，予約語，サービスルーチン名以外でキーワードとする語句を，ユーザキーワードファイルの形式で登録します。

ユーザキーワードファイルは，対象となる COBOL の予約語ごとに設定します。対象となる COBOL の予約語は，「COBOL 予約語」で選択してください。

ユーザキーワードファイルは，ユーザが独自に作成して登録します。作成方法の詳細は，「(3)　COBOL サービスルーチンファイル，ユーザキーワードファイルの作成」を参照してください。

COBOL エディタでは，ユーザキーワードファイルのサンプルとして次のファイルを提供しています。

ファイル名	登録内容
ユーザキーワード.usw	ユーザキーワードのサンプル （「DATA01」「データ名０１」の２語をユーザキーワードとして登録する）

サンプルを参考にして，ユーザキーワードファイルを作成してください。

サービスルーチンやキーワードファイルの入力エラーを表示しない

COBOL サービスルーチンファイル，またはユーザキーワードファイルの入力時に発生したエラーを無視したいときに指定します。

この項目を指定しなかった場合は，COBOL サービスルーチンファイル，またはユーザキーワードファイルの入力時にエラーが発生すると，予約語ファイルエラー結果ダイアログボックスにエラー内容が表示されます。

テンプレートフォルダ

COBOL サービスルーチンファイル，ユーザキーワードファイルおよびユーザ独自のプログラムテンプレートファイルの格納場所を表示します。

デフォルトの格納場所

```
COBOL2002インストールフォルダ¥template
```

ユーザは右側の[...]ボタンを押して，この格納場所を変更できます。

ユーザがこの格納場所を変更するとき，変更前の格納場所にあるすべての COBOL サービスルーチンファイル，ユーザキーワードファイルおよびユーザ独自のプログラムテンプレートファイルで選択しているファイルを，変更後の格納場所に移動またはコピーしてください。

(2) COBOL サービスルーチンファイル，ユーザキーワードファイルの登録

ここでは，COBOL サービスルーチンファイル，およびユーザキーワードファイルを登録する方法について説明します。

(a) ファイルを登録する

1. 「COBOL 予約語」から，ファイルを登録したいシステムを選ぶ。

   ファイル一覧の見出しが選んだシステム名に変更されます。

   例えば，「COBOL の予約語」から「COBOL2002」を選んだ場合は，「COBOL2002 サービスルーチンファイル」，「COBOL2002 ユーザキーワードファイル」に変更されます。

2. 「COBOL2002 サービスルーチンファイル」または「COBOL2002 ユーザキーワードファイル」の欄の［追加］ボタンから，ファイルを選択する。

   ファイルが一覧に登録されます。

なお，COBOL サービスルーチンファイル，ユーザキーワードファイルは，システムごとにそれぞれ 10 ファイルずつ登録できます。

(b) ファイルを登録から除外する

1. 除外する COBOL サービスルーチンファイル，またはユーザキーワードファイルを一覧から選んで［削除］ボタンを選ぶ。

   ファイルが一覧から除外されます。

(c) ファイルを一時的に無効にする

1. 無効にする COBOL サービスルーチンファイル，またはユーザキーワードファイルを一覧から選び，チェックボックスをオフにする。

   ファイルに登録された語句が，一時的にキーワードとして無効になります。再度有効にする場合は，チェックボックスをオンにしてください。

(3) COBOL サービスルーチンファイル，ユーザキーワードファイルの作成

COBOL エディタでは，ユーザ独自の COBOL サービスルーチンファイルやユーザキーワードファイルを作成できます。次に，COBOL サービスルーチンファイル，ユーザキーワードファイルに記述する内容を説明します。

(a) ファイルの形式

COBOL サービスルーチンファイルの形式

COBOL サービスルーチンファイルは，次の形式で作成してください。

```
* COMMENT              ―― 1
S サービスルーチン名    ―― 2
** END                 ―― 3
```

1. コメント

   ＊で始まる行は，コメントを表します。

2. サービスルーチン名

   「S 」（S のあとに半角空白文字）で始まる行は，サービスルーチン名を表します。S に続く文字列が，サービスルーチンとして登録されます。

3. ファイルの終わり

   COBOL サービスルーチンファイルの終わりを表します。

ユーザキーワードファイルの形式

ユーザキーワードファイルは，次の形式で作成してください。

```
* COMMENT              ―― 1
U ユーザキーワード名    ―― 2
** END                 ―― 3
```

1. コメント

   ＊で始まる行は，コメントを表します。

2. ユーザキーワード名

   「U 」（U のあとに半角空白文字）で始まる行は，ユーザキーワード名を表します。U に続く文字列が，ユーザキーワードとして登録されます。

3. ファイルの終わり

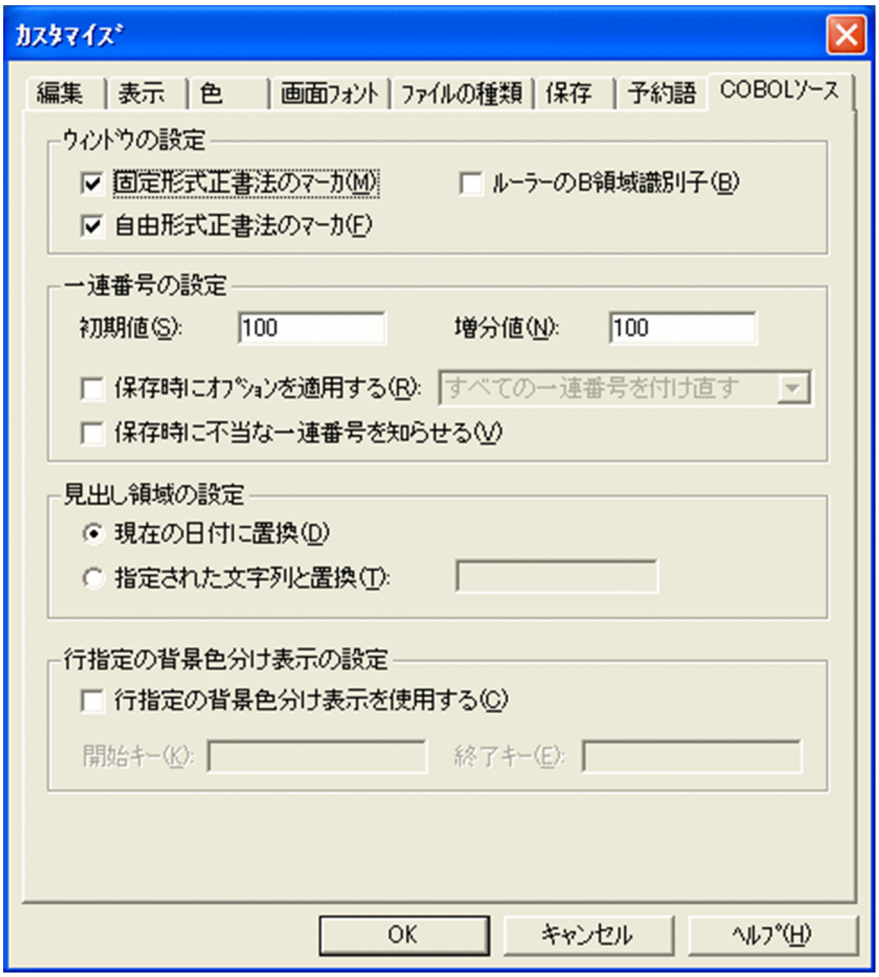
   ユーザキーワードファイルの終わりを表します。

(b)　ファイルの作成規則

- サービスルーチン名，およびユーザキーワード名は，60 バイト以内で指定してください。
- サービスルーチン名，およびユーザキーワード名には，次の文字列を使用しないでください。
  ( ) = + * / . , : ; ” ’ [ ] { } < > 空白文字　タブ文字　数字
- 予約語，サービスルーチン名，ユーザキーワード名で，語の重複があってはなりません。

4.10.8　COBOL ソースの設定

[COBOL ソース] タブを選んで表示されるページを次に示します。ここでは，COBOL ソースの正書法に従ったマーカの表示，一連番号領域や見出し領域を編集する場合の初期値などの設定，注記行を利用し特別なキーを設定するすることで異なる背景色を設定します。



(1)　COBOL ソースタブの説明

(a)ウィンドウの設定

COBOL ソースの正書法に従ったマーカやルーラーの B 領域識別子の表示を設定します。

固定形式正書法のマーカ

COBOL ソース（固定形式正書法）のソースウィンドウの 7，8，および 73 カラムの前にマーカを表示したいとき指定します。チェックボックスをオンにして［OK］ボタンを選ぶと，マーカを表示します。

自由形式正書法のマーカ

COBOL ソース（自由形式正書法）のソースウィンドウの 256 カラムの前にマーカを表示したいときに指定します。チェックボックスをオンにして［OK］ボタンを選ぶと，マーカを表示します。

ルーラーの B 領域識別子

COBOL85 の COBOL ソース（固定形式正書法）の B 領域を示すため，ルーラーの 12 カラムに'B'を表示したいとき指定します。チェックボックスをオンにして［OK］ボタンを選ぶと，ルーラーに’B’を表示します。

B 領域とは，COBOL85 で使用する固定形式正書法のコーディング用紙の形式の 12 カラムから 72 カラムの領域のことです。COBOL85 コーディング用紙の形式の領域は次のとおりです。

なお，数字はカラムを示します。



(b)一連番号の設定

一連番号の設定では，一連番号を付け直す場合の初期値，増分値を設定します。または COBOL ソース（固定形式正書法）の保存時に，一連番号を付け直すかどうかを指定できます。一連番号を付け直すときは，［ツール］メニューの［一連番号の設定］の［すべての一連番号を付け直す］，［不当な一連番号を付け直す］のどちらかを選択します。

行数が多く一連番号の付け直しに時間を要する場合は，メッセージボックスに「一連番号を付け直し中・・・」が表示されます。

初期値

初期値に指定できる値は 0～999,999 の最大 6 けたの数値です。数字以外の文字を入力した場合はエラーメッセージを出力します。

増分値

増分値に指定できる値は 1～100,000 の最大 6 けたの数値です。数字以外，または範囲外の数値を入力した場合はエラーメッセージを出力します。

保存時にオプションを適用する

保存時に COBOL ソース（固定形式正書法）の一連番号領域について一連番号を付け直したいとき，または空白に置換したいときに指定します。このオプションがチェックありの場合，「保存時に不当な一連番号を知らせる」オプションは無効となります。

表 4-15　保存時に適用するオプション

保存時に適用するオプション	保存時の動作
すべての一連番号を付け直す	すべての一連番号を生成し直す

保存時に適用するオプション	保存時の動作
不当な一連番号を付け直す	数字でない，または昇順でない一連番号だけを生成し直す
一連番号領域を空白に置換	一連番号領域を半角空白文字に置換する

注意事項

- カスタマイズダイアログボックスの「色」ページの「注記行の設定」で，「指定文字が 1 カラムにある行を注記行とする」を有効にして，指定文字が 1 カラムにある行を注記行とみなし，一連番号の生成や空白に置換の対象としません。
- プログラム中で，翻訳指令によって自由形式正書法を指定している場合（">>SOURCE FORMAT IS FREE"の記述がある場合），［ツール］メニューの［一連番号の設定］は選択できません。

保存時に不当な一連番号を知らせる

COBOL ソース（固定形式正書法）の保存時に不当な一連番号があればメッセージを出力し，保存を中止します。

(c)見出し領域の設定

見出し領域の設定では，見出し領域を編集する文字列を設定します。［ツール］メニューの［見出し領域の置換］を選ぶと，カーソルが位置する行，または選択した行の見出し領域に見出し領域の設定で指定した現在の日付，または指定した文字列を追加または置換できます。

見出し領域とは，COBOL85 で使用する固定形式正書法のコーディング用紙の形式の 73 カラムから 80 カラムの領域のことです。COBOL85 コーディング用紙の形式の領域は次のとおりです。

なお，数字はカラムを示します。



現在の日付に置換

現在の日付を文字列として見出し領域に設定します。現在の日付は”yyyymmdd”（yyyy は西暦の年，mm は月，dd は日）の 8 けたの形式で設定します。

指定された文字列と置換

見出し領域に追加，または置換する文字列を設定します。見出し領域の文字列は最大 8 バイトです。指定文字列が 8 バイトに満たない場合は，8 バイトになるように 1 バイト空白文字を補います。

(d)行指定の背景色分け表示の設定

行指定の背景色分け表示を使用する

行指定の背景色分け表示機能を使用したいとき指定します。

開始キー，終了キー

COBOL ソース中で，色分け表示したい区間を表すコメント文字を，30 バイト以内の文字列で指定します。

コメント（注記）行を表す星印（＊），斜線（/），または注記指示子（*>）に続けて開始キー，および終了キーを記述すると，開始キーの行と終了キーの行で囲まれた部分の背景色が，指定した色で表示されます。

開始キーを「@@START-KEY」，終了キーを「@@END-KEY」に設定した場合の COBOL エディタのソースウィンドウの表示の例を次に示します。

```
000510     DISPLAY 'Special Key'.
000520*@@START-KEY  COMMENT
000530     IF TEST01 = 'START' THEN
000540         DISPLAY 'TEST01'
000550     END-IF.
000560*@@END-KEY    COMMENT
000570     IF TEST02 = 'START' THEN
000580         DISPLAY 'TEST02'
000590     END-IF.
```

（行指定の背景色分け表示の規則）

- 開始キーと終了キーに空白文字は使用できません。
- 開始キーと終了キーに同じ文字列は使用できません。
- 開始キーと終了キーに，それぞれの先頭からの文字が一致した文字列は登録できません。

  例えば，「KEY」と「KEY1」という文字列では，「KEY」の部分が重複するので，この組み合わせを開始キーと終了キーには指定できません。
- 開始キーと終了キーも指定した 1 バイトの英文字は，大文字／小文字が等価に扱われます。ただし，2 バイトの英文字は大文字／小文字が区別されます。

  例えば，開始キーに 1 バイト文字の「START」を指定した場合，「Start」も開始キーとみなされます。
- 固定形式正書法の場合，コメント行の標識領域の星印（＊）または斜線（/）に続けて開始キー，または終了キーを記述しなければなりません。また，自由形式正書法の場合，行の最初の文字位置から星印（＊）または斜線（/）に続けて開始キー，または終了キーを記述しなければなりません。
- 開始キー，または終了キーに続く同一行の文字列は，通常のコメント文字として扱われます。
- 開始キーだけがあり，終了キーがない場合は，開始キーから最終行までの背景色が色分け表示されます。
- 終了キーだけがあり，開始キーがない場合は，終了キーは無視され，色分け表示されません。

  背景が色分け表示された範囲内では，予約語や動詞などに指定した背景色は無視されます。

4.11　カスタマイズ情報の外部ファイル保存機能

外部ファイルを使用して，COBOL エディタで設定したオプションなどのカスタマイズ情報をエクスポートまたはインポートできます。

外部ファイルに保存すれば，あるユーザが COBOL エディタで設定したカスタマイズ情報を，ほかのユーザの COBOL エディタでも使えるようになり，同じ設定で COBOL ソースファイルを扱うことができます。

例えば，次の場合に役に立ちます。

- 製品をアンインストールして再度インストールする場合

  注

  　カスタマイズ情報は，アンインストール前にエクスポートする必要があります。
- 別のコンピュータやサインインアカウントで同じ設定を使用したい場合

4.11.1　保存できるカスタマイズ情報

カスタマイズ情報のうち，外部ファイルに保存できる情報を次に示します。

- カスタマイズダイアログボックスで設定したオプション情報
- 構文チェックオプションダイアログボックスで設定した情報
- 構文テンプレートダイアログボックスで設定したユーザ構文テンプレートフォルダ

4.11.2　カスタマイズ情報のエクスポート方法

カスタマイズ情報を COBOL エディタ設定ファイル（.cex）にエクスポートする方法を次に示します。

1. **［オプション］メニューから［エクスポート］を選ぶ。**
   保存先のファイル名を設定するダイアログが表示されるので，ファイル名を入力します。

2. **［保存］ボタンを選ぶ。**
   指定した名前でファイルが作成されます。

注意事項

- エクスポートした COBOL エディタ設定ファイル（.cex）を編集しないでください。編集したファイルをインポートした場合の動作は保証しません。
- エクスポートを実行したとき，存在するファイルは上書きされます。必要なファイルは，事前に退避してからエクスポートを実行してください。

4.11.3　カスタマイズ情報のインポート方法

カスタマイズ情報を COBOL エディタ設定ファイル（.cex）からインポートする方法を次に示します。

1. ［オプション］メニューから［インポート］を選ぶ。

   読み込むファイル名を設定するダイアログが表示されるので，目的のファイルを選ぶか，またはファイル名を入力します。

2. ［開く］ボタンを選ぶ。

   指定したファイルが読み込まれ，COBOL エディタのカスタマイズ情報に反映されます。

注意事項

インポートが正常に終了している場合でも，一部の設定は現在の COBOL エディタにはすぐには反映されていないことがあります。このため，インポート終了後に COBOL エディタを再起動することを推奨します。

4.12　印刷

ここでは，印刷に関する操作を説明します。

4.12.1　ページレイアウトの設定

印刷時のページレイアウトの設定方法を次に示します。

1. COBOL エディタウィンドウの［ファイル］メニューから［ページレイアウトの設定］を選ぶ。

   ページレイアウトの設定ダイアログボックスが表示されます。



2. 次の項目を指定する。

   ヘッダとフッタ

   表題のほかに次に示すマクロ文字を使ってファイル名やページ番号などの情報をヘッダやフッタ領域に印刷できます。また，揃え操作のマクロ文字を利用して，ヘッダやフッタに印刷する文字列のレイアウトを自由に設定できます。

   これらのマクロ文字は，［▶］ボタンを押すと表示される印刷項目や揃え操作を選んで指定します。

印刷項目	マクロ文字
ファイル名	&F
ページ番号	&P
現在時刻	&T
日付	&D

揃え操作	マクロ文字
左端揃え	&L
中央	&C
右端揃え	&R

   例えば，ファイル名が「KINRI.CBL」の場合，ヘッダに「&L&F&C 金利計算&R ページ&P」と指定すれば，次のように印刷されます。

```
KINRI.CBL              金利計算              ページ1
```

なお，マクロ文字の指定によっては，ヘッダやフッタ領域の文字が重なって印刷されることもあります。

余白

［上］，［下］，［左］，［右］ボックスに余白の値（単位はインチ，またはミリメートル）を入力します。また，二段組みを指定した場合は，段間隔の値を設定できます。

行番号

行番号を印刷したいときに指定します。

行末空白削除

行末に余分な空白があっても，その空白を印刷しない場合に指定します。

折り返し

印刷する用紙からはみ出す行を折り返して，全体を印刷したいときに指定します。

段組

段数を指定します。

印刷フォント

［印刷フォント］ボタンを選ぶと表示されるダイアログボックスで，印刷時に使用するフォント，スタイル，およびサイズを設定します。

プリンタの設定

［印刷フォント］ボタンを選ぶと表示されるダイアログボックスで，印刷時に使用するプリンタ，用紙サイズ，および印刷の向きなどを指定します。

初期値へ戻す

［印刷フォント］ボタンを選ぶと，上記の設定が初期値に戻ります。

なお，［印刷プレビュー］ボタンで印刷イメージを確認できます。

3. **［OK］ボタンを選ぶ。**

指定したページレイアウトが設定されます。

補足事項

次のどちらかの場合は，［印刷プレビュー］ボタンが表示されません。

- ソースファイルを開いていない場合
- 印刷プレビューから，ページレイアウトの設定ダイアログボックスを表示した場合

4.12.2 印刷プレビュー

印刷プレビューを表示して印刷イメージを確認したあと，印刷します。

1. **COBOL エディタウィンドウの［ファイル］メニューから［印刷プレビュー］を選ぶ。**

アクティブ状態のソースウィンドウに表示中の内容が，印刷イメージで表示されます。

2. 印刷イメージを確認し，［印刷］ボタンを選ぶ。

表示された印刷ダイアログボックスに必要事項を入力して印刷します。詳細は，「4.12.3　印刷」を参照してください。

4.12.3　印刷

印刷の範囲や部数を設定したあと，印刷します。

1. COBOL エディタウィンドウの［ファイル］メニューから［印刷］を選ぶ。

印刷ダイアログボックスが表示されます。

2. 印刷範囲，部数，および印刷品質などを指定し，［OK］ボタンを選ぶ。

アクティブ状態のソースウィンドウに表示されているソースファイルの内容が印刷されます。

5 ODBC レコード定義生成

COBOL2002 を使って開発したプログラムから，SQL を使ってデータソースをアクセスし，表を参照・更新できます。そのためには，ODBC（Open Database Connectivity）レコード定義生成機能を使って，あらかじめ表の定義からレコード定義を生成しておき，原始プログラムから COPY 文で取り込む必要があります。

この章では，表の定義からレコード定義を生成する方法について説明します。

5.1　概要

ODBC（Open Database Connectivity）レコード定義生成とは，各種データベースによって生成されたデータソース中から表に関する情報を取得し，これを基に COBOL2002 のレコード定義を生成する機能です。

この機能を使用して，データベースにアクセスできる ODBC 対応のアプリケーションプログラムを作成できます。

つまり，ユーザは生成されたレコード定義を COPY 文で取り込み，原始プログラム中に展開すれば，COBOL2002 のプログラムから SQL を使って表を参照したり更新したりできます。

5.1.1　データソースとは

データソースとは，ユーザがアクセスするデータとそのデータを取得するために必要な情報です。

例えば，SQL Server ドライバから見ると，SQL Server のデータベース，SQL Server データベースが保存されているサーバ，そのサーバへアクセスするために使われるネットワークがデータソースになります。

5.1.2　前提条件

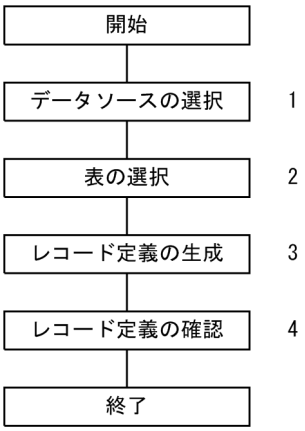
ODBC レコード定義を生成する前に，あらかじめ ODBC ドライバの組み込みとデータソースのセットアップが必要です。

データソースのセットアップについては，マニュアル「COBOL2002　ユーザーズガイド」のデータソースの管理についての記述を参照してください。

5.1.3　レコード定義生成の流れ

ODBC レコード定義生成の流れを次に示します。

図 5-1　レコード定義生成の流れ



1. データソース群から該当のデータソースを選択します。

2. データソースからレコード定義を生成したい表を選択します。

3. 生成したレコード定義を格納するファイル名などを指定します。

4. 生成されたレコード定義を確認します。

5.2　レコード定義生成の規則

ODBC レコード定義生成機能は，データソースに依存する列情報を，いったんデータソースに依存しない ODBC の SQL データ型に変換します。そして，変換した SQL データ型を基に COBOL2002 のレコード定義を生成します。

ここでは，その規則について説明します。

5.2.1　データソース中の表から求められる列の情報

データソース中の表から求められる列情報の意味，およびその列情報がどのような値をとるのかを次に示します。

表 5-1　データソース中の表から求められる列情報と変換される SQL データ型

表の列情報（型名）	SQL データ型	列情報の意味
表名(TABLE_NAME)	Varchar(128) not NULL	表の識別子。
列名(COLUMN_NAME)	Varchar(128) not NULL	列の識別子。
データ型 (DATA_TYPE)	Smallint not NULL	SQL のデータ型。 ODBC SQL，またはドライバ特有の SQL のデータ型である。
型名(TYPE_NAME)	Varchar(128) not NULL	データソースに依存するデータ型の名称。 例えば，"CHAR"，"VARCHAR"，"MONEY"，"LONG VARBINARY"，または"CHAR ( ) FOR BIT DATA"など。
精度(PRECISION)	Integer	列の精度。
長さ(LENGTH)	Integer	列の長さ。 SQL_C_DEFAULT が指定されていると SQLGetData，または SQLFetch に送られたデータのバイト長。 数値データについてのサイズは，データソース上に格納されたデータのサイズと異なる場合がある。 この値は，文字またはバイナリデータ時には，列の精度と同じになる。
スケール(SCALE)	Smallint	列のスケール。 スケールが適用できないデータ型には，NULL が返される。

注

　データ型の詳細については，各 ODBC ドライバのマニュアルを参照してください。

5.2.2 COBOL2002 のレコード定義を生成するときの規則

COBOL2002 のレコード定義を生成するとき，01 レベルのデータ名称は表名になります。02 レベル以降の変換規則については，次に示す SQL のデータ型から生成される COBOL2002 のレコード定義に従います。

表 5-2 SQL のデータ型から生成される COBOL2002 のレコード定義

SQL データ型	符号付き／符号なし	生成される COBOL2002 のレコード定義
CHAR(n)	−	02 列名 PIC X(n).
VARCHAR(n)	−	02 列名. 03 ODBC-length PIC S9(9) USAGE BINARY. 03 ODBC-char PIC X(n).
LONG VARCHAR	−	02 列名. 03 ODBC-length PIC S9(9) USAGE BINARY. 03 ODBC-char PIC X(2000).
DECIMAL(p，s)	−	02 列名 PIC S9(p-s)V9(s) SIGN LEADING SEPARATE.
NUMERIC(p，s)	−	02 列名 PIC S9(p-s)V9(s) SIGN LEADING SEPARATE.
SMALLINT	符号付き	02 列名 PIC S9(4) USAGE BINARY.
	符号なし	02 列名 PIC 9(4) USAGE BINARY.
INTEGER	符号付き	02 列名 PIC S9(9) USAGE BINARY.
	符号なし	02 列名 PIC 9(9) USAGE BINARY.
REAL	−	02 列名 USAGE COMP-1.
FLOAT	−	02 列名 USAGE COMP-2.
DOUBLE PRECISION	−	02 列名 USAGE COMP-2.
BIT	−	02 列名. 03 FILLER PIC 1(7) USAGE BIT VALUE ZERO. 03 ODBC-bit PIC 1(1) USAGE BIT.
TINYINT	−	02 列名 PIC X(1).
BIGINT	−	02 列名 PIC X(20).

SQL データ型	符号付き／符号なし	生成される COBOL2002 のレコード定義
BINARY(n)	−	02 列名 PIC X(n).
VARBINARY(n)	−	02 列名. 03 ODBC-length PIC S9(9) USAGE BINARY. 03 ODBC-binary PIC X(n).
LONG VARBINARY	−	02 列名. 03 ODBC-length PIC S9(9) USAGE BINARY. 03 ODBC-binary PIC X(2000).
DATE	−	02 列名. 03 ODBC-year PIC S9(4) USAGE BINARY. 03 ODBC-month PIC 9(4) USAGE BINARY. 03 ODBC-day PIC 9(4) USAGE BINARY.
TIME	−	02 列名. 03 ODBC-hour PIC 9(4) USAGE BINARY. 03 ODBC-minute PIC 9(4) USAGE BINARY. 03 ODBC-second PIC 9(4) USAGE BINARY.
TIMESTAMP	−	02 列名. 03 ODBC-year PIC S9(4) USAGE BINARY. 03 ODBC-month PIC 9(4) USAGE BINARY. 03 ODBC-day PIC 9(4) USAGE BINARY. 03 ODBC-hour PIC 9(4) USAGE BINARY. 03 ODBC-minute PIC 9(4) USAGE BINARY. 03 ODBC-second PIC 9(4) USAGE BINARY. 03 ODBC-fraction PIC 9(9) USAGE BINARY.

（凡例）

n：長さ（LENGTH）

p：精度（PRECISION）

s：位取り（SCALE）

−：該当しない

注

「表 5-2　SQL のデータ型から生成される COBOL2002 のレコード定義」以外の SQL データ型には対応していません。

5.3　レコード定義生成の操作

5.3.1　操作方法

ODBC レコード定義生成の操作方法を説明します。

(1)　生成の開始

1. 開発マネージャのウィンドウの［ツール］メニューから［ODBC 定義］を選ぶ。または Windows のプログラム一覧にある［COBOL2002］下のメニューから［ODBC レコード定義生成 for COBOL2002］を選ぶ。

   データソースを選択するダイアログボックスが表示されます。

(2)　データソースの選択

データソース群から該当のデータソースを選択します。



1. データソースのリストから，使用したいデータソースを選び，［OK］ボタンを選ぶ。またはデータソースのリストから，使用したいデータソースをダブルクリックする。

   指定したデータソースの表情報が取得され，表を選択するダイアログボックスが表示されます。

注意事項

1. データソースのリスト中に使用したいデータソースがない場合は，データソースを追加する必要があります。データソースを追加する方法については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。
2. データソースを選択したあと，データソースの内容によっては，さらにデータソースの詳細を問い合わせるダイアログボックスが表示される場合があります。その場合は，ダイアログボックスの指示に従って操作してください。

(3)　表の選択

データソースから，レコード定義を生成する表を選択します。



指定したデータソース中にあるすべての表名が，右のリストボックスに表示されます。

1. 右のリストボックスから一つ以上の表名を選び，［選択］ボタンを選ぶ。

   選択した表名が，左のリストボックスに表示されます。

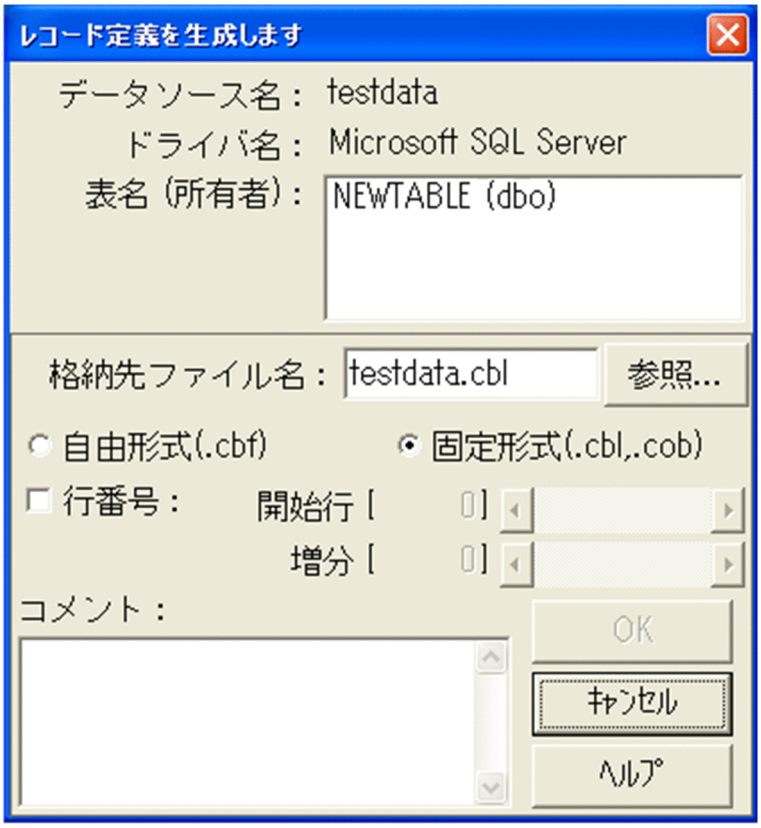
   なお，左のリストボックスから表名を一つ以上選び［削除］ボタンを選ぶか，またはダブルクリックすると，表名が消え選択が取り消されます。

2. ［OK］ボタンを選ぶ。

   指定した表の列情報が取得され，レコード定義を生成するダイアログボックスが表示されます。

(4)　レコード定義の生成

レコード定義の生成先，および COBOL ソースの形式を指定します。また，ソース中に挿入するコメントを入力します。



- 固定形式と自由形式はどちらか一つを指定します。
- 自由形式を指定した場合，行番号は指定できません。
- コメントは，レコード定義生成時に，ソースの先頭に「COBOL2002 注記」として展開されます。

1. レコード定義の格納先ファイル名，自由／固定形式，およびソース行番号の有無を指定する。

2. ソース中に挿入するコメントを入力する。（任意）

3. [OK] ボタンを選ぶ。

   指定した内容に従って，レコード定義が生成され，ファイルに格納されます。ファイルは，カレントドライブのカレントフォルダに保存され，ファイル名には形式で指定した拡張子が自動的に付きます。

   生成されたレコード定義を確認するダイアログボックスが表示されます。

注意事項

レコード定義生成ダイアログボックスで指定した形式の（　）内の拡張子と格納するファイルの拡張子は，同じにする必要があります。

拡張子が異なる場合，このファイルを COPY 展開するときに正しく展開されません。ただし，これはデフォルトの拡張子であるため，コンパイル時に環境変数 CBLFIX または CBLFREE で拡張子を設定，登録すれば，正しく展開されます。

(5)　レコード定義の確認

格納先ファイルを開いて，生成されたレコード定義を確認します。

レコード定義確認ダイアログボックスの［表示］ボタンを選んで，生成されたレコード定義を確認してください。

5.4　レコード定義の生成例

ある SQL Server ドライバのデータソースの表から，COBOL2002 のレコード定義を生成した場合について説明します。

このときの，基となるデータソースの表情報を「表 5-3　データソースの表から求められた列情報」に示します。また，生成された COBOL2002 のレコード定義を「図 5-2　生成された COBOL2002 のレコード定義」に示します。

表 5-3　データソースの表から求められた列情報

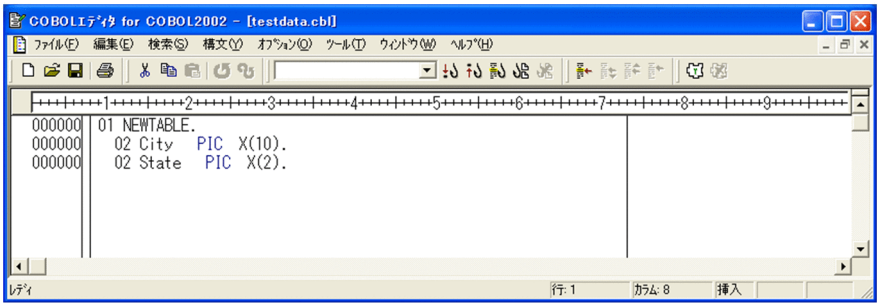
列情報項目	列情報 1	列情報 2
表名(TABLE_NAME)	NEWTABLE	NEWTABLE
列名(COLUMN_NAME)	City	State
データ型(DATA_TYPE)	1(CHAR)	1(CHAR)
型名(TYPE_NAME)	char	char
精度(PRECISION)	10	2
長さ(LENGTH)	10	2
位取り(SCALE)	<Null>	<Null>

注

列情報項目については，「表 5-1　データソース中の表から求められる列情報と変換される SQL データ型」を参照してください。

以上の情報を基に，固定形式，行番号あり，開始行 0，増分 0，コメント未入力と指定すると，次のような COBOL2002 のレコード定義が生成されます。

図 5-2　生成された COBOL2002 のレコード定義



注 1

01 レベルのデータ名称は表名になります。

注 2

02 レベル以降の変換規則は，「表 5-2　SQL のデータ型から生成される COBOL2002 のレコード定義」に従います。

注 3

コメントについては，次の規則で生成します。

- 固定形式の場合
  72 バイト（2 バイト文字が 72 バイトと 73 バイトにわたる場合は 73 バイト）で改行を挿入します。
- 自由形式の場合
  80 バイト（2 バイト文字が 80 バイトと 81 バイトにわたる場合は 81 バイト）で改行を挿入します。

6 画面定義

COBOL2002 では，COBOL プログラムで使用する画面を定義できます。また，定義した画面から画面節（SCREEN SECTION または WINDOW SECTION）のデータ部の COBOL ソースを生成できます。

この章では，画面定義，および COBOL ソース生成の操作について説明します。

画面定義ファイルは，Windows(x86) COBOL2002 と Windows(x64) COBOL2002 間で共用できます。

6.1　概要

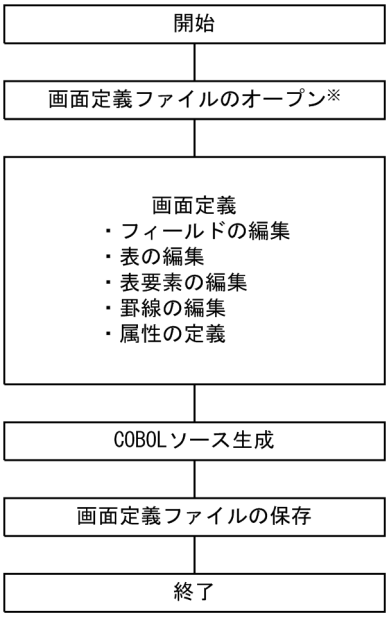
画面定義とは，COBOL プログラムで使用する画面を作成するツールのことです。

画面定義を使うと，GUI を使って画面を作成できます。また，作成した画面を基に，画面節（SCREEN SECTION または WINDOW SECTION）のデータ部の COBOL ソースファイルを生成できます。生成された COBOL ソースファイルは，COBOL プログラム中に COPY 文で取り込みます。

6.1.1　画面定義の流れ

画面定義の流れを次に示します。

図 6-1　画面定義の流れ



注※

画面定義ファイルが開かれていない場合でも，画面を定義したり，COBOL ソースを生成したりできます。

6.1.2　使用するファイル

(1)　画面定義ファイル（.wdf）

画面定義で設定した情報を格納するファイルです。

(2) COBOL データ定義ファイル（.cbl，.cbf，.cob，.ocb，.ocf）

画面節（SCREEN SECTION および WINDOW SECTION）の定義部の COBOL ソースファイルです。これらのソースファイルは，画面定義ファイルが自動生成したファイルであり，COBOL の COPY 文で取り込む必要があります。

6.1.3 予約語の扱い

画面定義では，COBOL の予約語として COBOL コンパイラと同等の予約語を認識します。

6.2　画面定義の開始と終了

画面定義の開始と終了の方法について説明します。

6.2.1　画面定義の開始

画面定義を開始するには，画面定義ファイルを指定して開始する方法と，定義ファイルを指定しないで開始する方法があります。

(1)　画面定義ファイルを指定する方法

次のどれかの方法で，画面定義を開始できます。

- 開発マネージャのツリービューウィンドウから，画面定義ファイルを選び，［ファイル］メニューから［開く］を選ぶ。
- 開発マネージャのツリービューウィンドウから，画面定義ファイルをダブルクリックする。
- エクスプローラから画面定義ファイル（.wdf）をダブルクリックする。
  指定した画面定義ファイルが開かれ，画面定義ウィンドウが表示されます。

(2)　画面定義ファイルを指定しない方法

次のどれかの方法で，画面定義を開始できます。

- 開発マネージャのウィンドウの［ファイル］メニューから［新規作成］を選び，ファイルの種類から「画面定義ファイル」を選ぶ。
- Windows のプログラム一覧にある［COBOL2002］下のメニューから［画面定義 for COBOL2002］を選ぶ。
  画面定義ウィンドウが表示されます。

6.2.2　画面定義の終了

1. 画面定義ウィンドウの［ファイル］メニューから［画面定義の終了］を選ぶ。
   設定した画面情報を保存していない場合には，ファイルを保存するかどうかを問い合わせるダイアログボックスが表示されるので，ファイルを保存してください。
   なお，新規に設定した画面情報を保存しないで終了した場合，設定した情報は失われます。「6.7　定義ファイルの操作」を参照し，ファイル名を付けて保存してください。

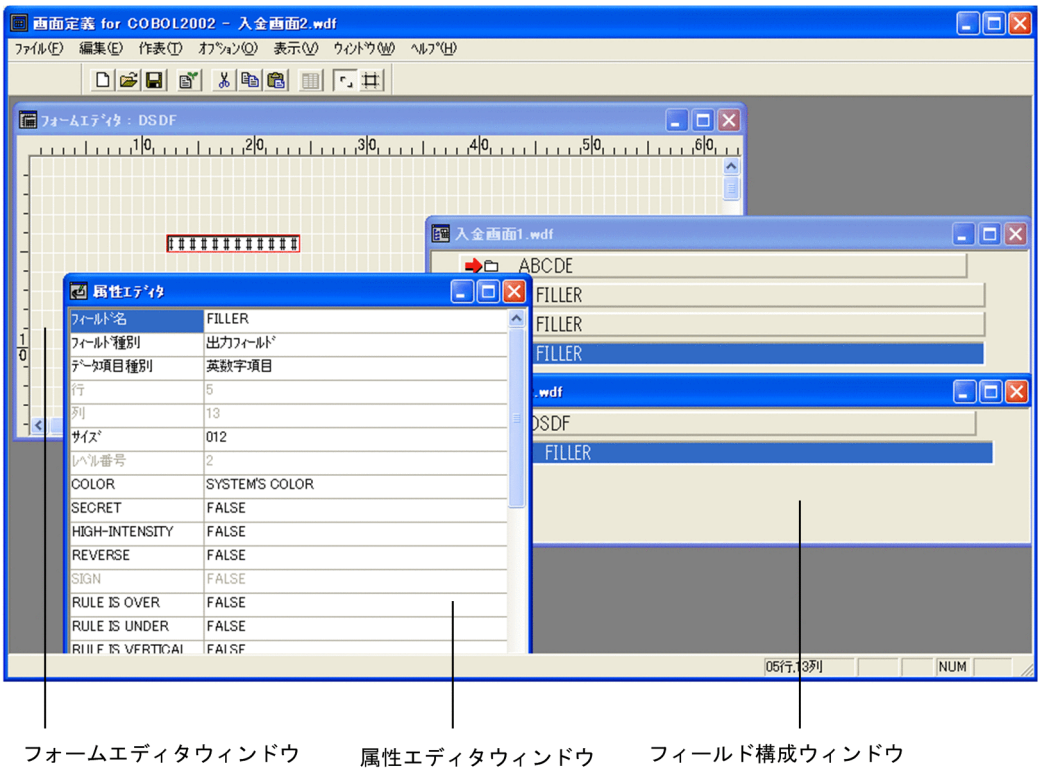
6.3　画面定義ウィンドウ

画面定義ウィンドウについて説明します。

6.3.1　画面定義ウィンドウの構成

画面定義ウィンドウを次に示します。

図 6-2　画面定義ウィンドウ



画面定義ウィンドウには，フォームエディタウィンドウと属性エディタウィンドウはそれぞれ 1 個だけ表示されます。フィールド構成ウィンドウは，開かれている画面定義ファイル（.wdf）の数に対応して表示されます。また，複数の画面定義ファイルを開いて画面を定義することもできます。

(1)　メニューバー

画面定義ウィンドウのメニューの種類と機能を次に示します。

表 6-1　画面定義ウィンドウのメニュー一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	新しく画面定義ファイルを作成する。
	開く(O)	既存の画面定義ファイルを開く。
	閉じる(C)	既存の画面定義ファイルを閉じる。

メニューバー	メニュー項目	機能
	上書き保存(S)	設定した画面情報を作業中の画面定義ファイルに保存する。
	名前を付けて保存(A)	設定した画面情報を画面定義ファイルの名前を付けて保存する。
	COBOL 生成(G)	COBOL ソースファイルを生成する。
	ファイル情報(I)	作業中のファイル(.wdf)の情報を表示する。
	画面定義の終了(X)	画面定義ウィンドウを閉じる。
編集(E)	切り取り(T)	選択した範囲を切り取って一時的なエリアに保存する。※1
	コピー(C)	選択した範囲をコピーして一時的なエリアに保存する。※1
	貼り付け(P)	一時的なエリアの内容をカーソルのある位置に貼り付ける。※2
	削除(D)	選択された範囲を削除する。
	集団項目作成(G)	集団項目を作成する。
作表(T)	表作成(G)	表を作成する。
	表要素の追加(A)	表の要素を追加する。
	表要素の削除(D)	表の要素を削除する。
	表要素の切り取り(T)	表の要素を切り取って一時的なエリアに保存する。
	表要素の貼り付け(P)	表の要素を一時的なエリアから貼り付ける。
	表要素のコピー(C)	表の要素をコピーして一時的なエリアに保存する。
オプション(O)	画面定義環境設定(E)	画面定義の実行環境を設定する。
	COBOL 生成許可(A)	COBOL ソースファイルの生成を許可する。
	フィールド作成モード(F)	フィールドの作成モードにする。
	罫線作成モード(R)	罫線の作成モードにする。
表示(V)	ツールバー(T)	ツールバーの表示を制御する。
	ステータスバー(S)	ステータスバーの表示を制御する。
ウィンドウ(W)	重ねて表示(C)	画面を重ねて表示する。
	並べて表示(T)	画面を並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	フォームエディタを開く(F)	フォームエディタウィンドウを開く。
	属性エディタを開く(B)	属性エディタウィンドウを開く。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※1

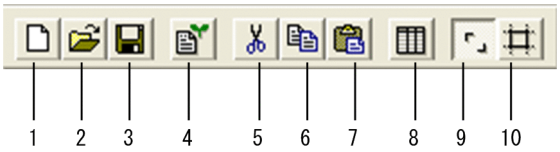
属性エディタの場合，キーボードから直接値を入力できる項目に対してだけ，［切り取り］［コピー］が実行できます。また，これらのコマンドで取得した内容は，クリップボードに保存されます。

注※2

属性エディタの場合，キーボードから直接値を入力できる項目に対してだけ，［貼り付け］が実行できます。また，［貼り付け］によって出力されるのは，クリップボードの内容になります。

(2)　ツールバー

ツールバーのボタンを選ぶとメニューと同じ操作ができます。ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。

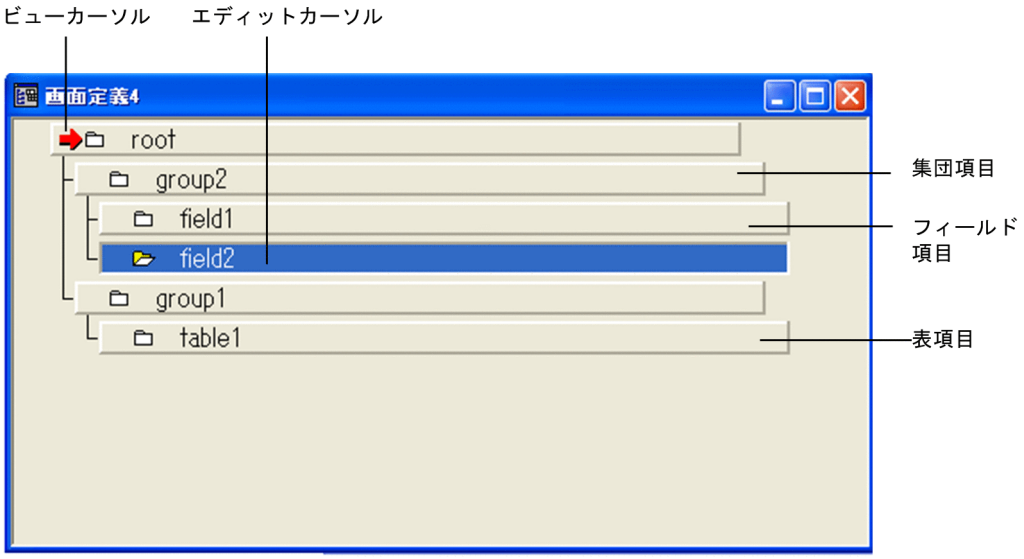


1. ［ファイル］メニューの［新規作成］
2. ［ファイル］メニューの［開く］
3. ［ファイル］メニューの［上書き保存］
4. ［ファイル］メニューの［COBOL 生成］
5. ［編集］メニューの［切り取り］
6. ［編集］メニューの［コピー］
7. ［編集］メニューの［貼り付け］
8. ［作表］メニューの［表作成］
9. ［オプション］メニューの［フィールド作成モード］
10. ［オプション］メニューの［罫線作成モード］

6.3.2　フィールド構成ウィンドウ

フィールド構成ウィンドウを次に示します。

図 6-3　フィールド構成ウィンドウ



フィールド構成ウィンドウは，画面に表示するフィールドと表の階層構造を表します。

ウィンドウには，フィールド項目，表項目，および集団項目が表示されます。フィールド項目は，画面の 1 フィールドに対応します。表項目は，定義した表に対応します。集団項目は，フィールド項目と表項目を複数まとめて一つの集団とするための項目です。

また，ウィンドウにはビューカーソルとエディットカーソルが表示されています。ビューカーソルは，集団項目だけを移動できます。エディットカーソルは，すべての項目を移動できます。ビューカーソルがある集団項目以下のフィールド項目と表項目が，フォームエディタウィンドウに表示されます。エディットカーソルがある項目の属性が，属性エディタウィンドウに表示されます。

次に，カーソルを移動する操作を示します。

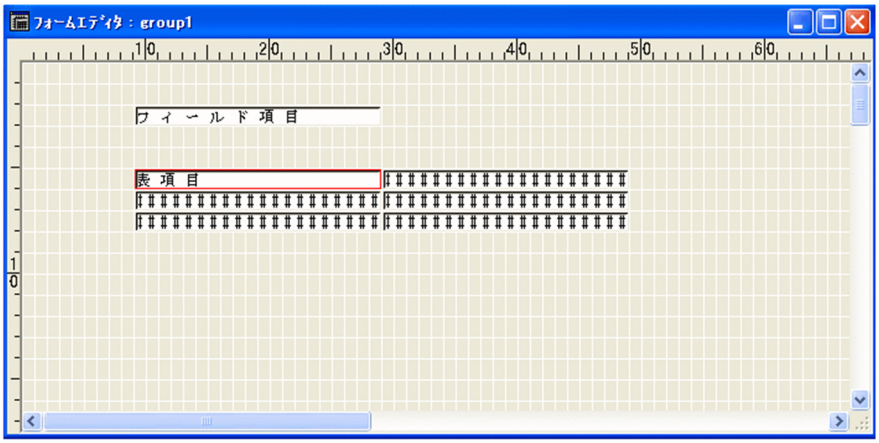
表 6-2　ビューカーソルおよびエディットカーソルの移動操作

	ビューカーソル	エディットカーソル
マウス操作	[Shift] ＋ [Ctrl] キーを押したままクリック	クリック
キー操作	[Ctrl] ＋矢印キー	矢印キー

6.3.3　フォームエディタウィンドウ

フォームエディタウィンドウを次に示します。

図 6-4　フォームエディタウィンドウ



フォームエディタウィンドウは，画面に表示するフィールドと表の位置関係を表します。

ウィンドウには，フィールド構成ウィンドウ上のビューカーソルがある集団項目以下のすべてのフィールド項目と表項目が，フィールドと表として表示されます。また，このウィンドウでは，最大 120 行 160 カラムの範囲を表示できます。

ウィンドウ中のフィールドと表の要素は，フィールド種別によって，次に示す文字で埋められています。

- 入力フィールド　：　@
- 出力フィールド　：　＃
- 更新フィールド　：　&

  なお，初期値を設定した場合は，その値が左詰めで表示されます。

6.3.4　属性エディタウィンドウ

属性エディタウィンドウには，フィールド構成ウィンドウ上のエディットカーソルがある項目の属性が表示され，フォームエディタウィンドウやフィールド構成ウィンドウでは設定できない，詳細な属性情報が表示されます。また，属性の設定時に，COBOL のシンタックスレベルのチェックができます。

このウィンドウで属性を設定，または変更した場合，内容はフォームエディタウィンドウに反映されます。

属性エディタウィンドウと，ウィンドウの各部の名称を次に示します。

図 6-5　属性エディタウィンドウ



属性エディタに表示されているリストは，図のようにリスト 1，リスト 2 に分かれています。リスト 1 には，フィールドの属性の種類が表示されます。リスト 2 には，その属性に設定されている値が表示されます。属性の設定方法については，「6.4.7　属性の設定」を参照してください。

属性エディタに表示される属性項目は，選択しているフィールドの種類によって異なります。また，COBOL の文法や，フィールド種別，データ項目種別の規則によって指定できない属性は，リスト中の項目は変更できません。

次に，属性エディタで設定できる項目について，フィールド項目ごとに説明します。

(1)　集団項目を選択した場合の属性情報

集団項目を選択した場合の属性項目について説明します。

グループ名

グループ名が表示され，そのまま COBOL ソース生成時のデータ名として使用されます。デフォルトは「FILLER」で，キー入力で変更できます。

レベル番号

COBOL ソース生成時のレベル番号が表示されます。

罫線表示

フォームエディタウィンドウ上で罫線を引くと，「指定あり」と表示されます。デフォルトは「指定なし」です。

(2) フィールド項目を選択した場合の属性情報

フィールド項目を選択した場合の属性情報について説明します。

フィールド項目を選択した場合，画面機能が WINDOW SECTION か SCREEN SECTION かによって，設定できる属性項目が異なります。

(a) 基本的な項目

フィールド名，項目の位置やサイズなど，基本的な項目について説明します。ここで挙げる項目は，WINDOW SECTION，および SCREEN SECTION で共通です。

フィールド名

フィールド名が表示され，そのまま COBOL ソース生成時のデータ名として使用されます。デフォルトは「FILLER」で，変更できます。

フィールド種別

フィールド項目選択時には，フィールド種別（入力フィールド，出力フィールド，更新フィールド）が表示されます。デフォルトは「出力フィールド」で，変更できます。

データ項目種別

フィールド項目選択時には，データ項目の種別（数字，英数字，英字，日本語，数字編集，英数字編集，日本語編集，定数項目）が表示されます。デフォルトは「英数字項目」で，変更できます。

行，列

項目の位置が表示されますが，変更はできません。

サイズ

データのバイト数が表示され，変更できます。

レベル番号

COBOL ソース生成時のレベル番号が表示されますが，変更はできません。

(b) 詳細な属性項目

ここでは，詳細な属性項目について説明します。詳細な属性項目は，次の六つの内容に分類できます。

- 表示属性
- 表示前動作
- 入力編集
- 入力チェック
- COBOL 属性
- HELP 属性

これらの属性項目の中には，WINDOW SECTION と SCREEN SECTION で項目名や機能が異なるものがあります。各項目の項目名，および機能の説明で，( ) が付いていない部分は WINDOW SECTION，( ) が付いている部分は SCREEN SECTION に関するものです。

表示属性

フィールドの視覚的な属性に関する情報です。

表 6–3　表示属性

名称	機能
COLOR (FOREGROUND-COLOR)	フィールドの文字の色を変えて表示する。 (フィールドの前景色を指定する。)
SECRET (SECURE)	フィールドの入出力データを表示しない。 (カーソルがこの句を指定した入力フィールド内にある間，オペレータがキーボードから入力したデータを画面上に表示しない。)
HIGH-INTENSITY (HIGHLIGHT)	フィールドの文字を高輝度で表示する。 (フィールドを画面上に最高明度で表示する。)
REVERSE (REVERSE-VIDEO)	フィールドの背景の色と文字の色を反転して表示する。 (デフォルトの前景色と背景色，または指定された前景色と背景色を入れ換えた状態でフィールドを表示する。)
SIGN (SIGN)	演算記号の位置を指定する。 (演算符号の表示の位置とモードを明示的に記述する必要がある場合，それらを指定する。)
RULE IS OVER (OVERLINE)	フィールドに上罫線を引く。 (フィールドの各文字の上側に横線を引く。)
RULE IS UNDER (UNDERLINE)	フィールドに下罫線を引く。 (フィールドが画面上に表示されるとき，そのフィールドの各文字に下線を引く。)
RULE IS VERTICAL	フィールドに縦罫線を引く。
RULE IS SQUARE	フィールドを罫線で囲む。
LETTER（表示用）	フィールドの文字の種類を指定する。 WIDE：横 2 倍 LONG：縦 2 倍 DOUBLE：横 2 倍×縦 2 倍
(BACKGROUND-COLOR)	(フィールドの背景色を指定する。)
(LOWLIGHT)	(フィールドを画面上に最低明度で表示する。2 段階の明度しか使用できない場合，LOWLIGHT 句による明度と通常の明度とは同一になる。)
(GRID)	(フィールドが画面上に表示されるとき，そのフィールドの各文字の左側に縦線を引く。)
(LEFTLINE)	(フィールドが画面上に表示されるとき，そのフィールドの最左端の文字の左側に縦線を引く。)

表示前動作

フィールドを表示する前の表示動作に関する情報です。

表 6-4　表示前動作

名称	機能
USING	数字項目の表現形式を指定する。
(BLANK)	（フィールドが表示されるとき，画面の 1 行または画面全体をクリアする。）
(ERASE)	（行の一部またはカーソル位置以降の画面をクリアする。）
(PROMPT)	（ACCEPT 文実行中にフィールドの空白の文字位置に印を付ける。）
(BELL)	（フィールドが表示開始されるとき，警告音を鳴らす。）

入力編集

フィールドの内容を編集する属性です。

表 6-5　入力編集

名称	機能
JUST (JUST)	項目にデータを収めるときに標準的でないけた寄せを指定する。 （フィールドにデータが転記されるとき，標準的でないけた寄せが行われる。）
BLANK WHEN ZERO (BLANK WHEN ZERO)	項目の値が 0 のときにそれを空白文字にする。 （データ項目の値が 0 のとき，空白文字を表示する。）
RESET	画面からの入力データを右に寄せて収めることを指定する。
(ZERO-FILL)	（フィールド中のユーザ入力のなかった文字位置の値が空白の場合，0 で置き換える。）

入力チェック

フィールドに入力する際にチェックが入る情報です。

表 6-6　入力チェック

名称	機能
ENTER-CHECK (REQUIRED)	入力または入出力フィールドにオペレータがデータを入力したかどうかを検査する。 （入力または更新フィールドに，少なくとも一つの文字を入力しなければならないことを指定する。）
LETTER（入力チェック用）	フィールドの文字の種類を指定する。 FREE： 日本語フィールドに全角，半角文字の混在入力および表示ができる。FREE 句を指定しないと，その日本語フィールドに半角文字を入力できない。 ALPHANUMERIC： フィールドに全角入力はできない。また，挿入，削除などは半角単位となる。
(AUTO)	（ACCEPT 文の実行中に，入力または更新フィールドの最後の文字が入力されたときカーソルを自動的に次のフィールドに移動させる。）

名称	機能
(FULL)	(フィールドを完全に空白のままにしておくか，またはそのフィールドをデータで満たすかのどちらかでなければならないことを指定する。)

COBOL 属性

COBOL 固有の言語仕様に従う情報です。

表 6-7　COBOL 属性

名称	機能
PICTURE (PICTURE)	基本データ項目の性質と編集の形式を示す。 (基本フィールドの一般的な性質と編集要求を記述する。)
SOURCE	フィールドが画面に表示されるときに内容が表示されるデータ項目を指定する。
(FROM)	(SOURCE 句と同様。)
(TO)	(フィールドに入力，または更新されたときに内容が格納されるデータ項目を指定する。)
VALUE(VALUE)	データ項目の初期値を定める。

HELP 属性

WINDOW SECTION のヘルプウィンドウ機能を規定する情報です。

表 6-8　HELP 属性

名称	機能
HELP	この句を書いたフィールドに対して，ユーザポップアップ HELP 画面名称，HELP キー，HELP 画面の表示位置および表示範囲を指定する。

(3)　表項目を選択した場合の属性情報

表項目を選択した場合の属性情報について説明します。

(a)　基本的な項目

表名，表項目の繰り返し回数，罫線の表示など，基本的な項目について説明します。

表名

表名が表示され，そのまま COBOL ソース生成時のデータ名として使用されます。この表名は変更できます。

(表) 行，(表) 列

表の位置が表示されますが，変更はできません。

繰り返し回数

表項目の繰り返し回数が表示されます。繰り返し回数は「REPEAT LINE」の繰り返しの数です。また，この繰り返し回数は変更できます。

行間

行間が表示されますが，変更はできません。

（表）レベル番号

COBOL ソース生成時のレベル番号が表示されますが，変更はできません。

罫線表示

表の罫線の有無が表示されますが，変更はできません。

要素名

表要素名が表示され，そのまま COBOL ソース生成時のデータ名として使用されます。デフォルトは「FILLER」で，変更できます。

次の属性については，フィールド項目を選択した場合の属性情報と同じです。詳細は，「(2)　フィールド項目を選択した場合の属性情報」の「(a)　基本的な項目」を参照してください。

なお，これらの属性は，現在選択状態にある表要素に対して表示されます。

フィールド種別

データ項目種別

（項目）行，（項目）列

サイズ

（項目）レベル番号

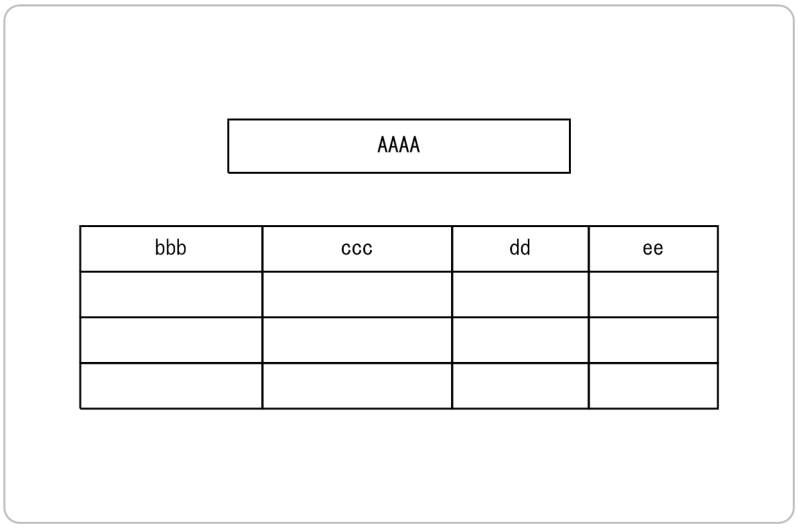
(b)　詳細な属性項目

詳細な属性項目については，WINDOW SECTION のフィールド項目を選択した場合と同じです。詳細は，「(2)　フィールド項目を選択した場合の属性情報」の「(b)　詳細な属性項目」を参照してください。

6.3.5　画面定義の手順例

ここでは，画面定義ウィンドウで画面を定義する手順を説明します。なお，詳細な操作については，「6.4 画面定義の操作」を参照してください。

次のような画面を定義するときの手順を説明します。



表を作成する場合は，新規作成ダイアログボックスで WINDOW SECTION を指定する必要があります。新規作成ダイアログボックスは，新規に画面定義を開始したあと，最初にフィールドを作成するか，または［ファイル］メニューから［新規作成］を選んだときに表示されます。

(1) フィールドを作成する

1. フォームエディタウィンドウ上で，フィールドの起点となる位置をマウスで選び，目的のサイズになるまでドラッグする。

   属性エディタウィンドウに，作成したフィールドの属性が表示されます。また，フィールド構成ウィンドウにはフィールド項目が表示されます。

2. 属性エディタウィンドウで，フィールドの属性を設定する。

   フィールド名，フィールド種別，データ項目の種別，および初期値などを設定します。例では，初期値に「AAAA」を設定します。設定したフィールド名は，フィールド構成ウィンドウ上のフィールド項目の名称になります。また，初期値は，フォームエディタウィンドウ上のフィールドに表示されます。

(2) 表を作成する

(a) 繰り返しのない表を作成する

1. フォームエディタウィンドウ上にフォーカスがあるとき，［作表］メニューから［表作成］を選ぶ。

   表示されたダイアログボックスに表の開始位置，繰り返し回数，および表要素の個数などを指定します。例では，繰り返し回数「1」，表要素数「4」を設定します。

2. ［OK］ボタンを選びます。

   属性エディタウィンドウに，作成した表の属性が表示されます。また，フィールド構成ウィンドウには表項目が表示されます。

3. 属性エディタウィンドウで，表の属性を設定する。

表名，要素名，フィールド種別，データ項目の種別，および初期値などを設定します。例では，各表要素の初期値に「bbb」「ccc」「dd」「ee」を設定します。

設定した表名は，フィールド構成ウィンドウ上の表項目の名称になります。

また，初期値は，フォームエディタウィンドウ上の表に表示されます。

(b) 繰り返しのある表を作成する

1. (a) と同様に，［作表］メニューから［表作成］で表を作成する。

   例では，繰り返し回数「3」，表要素数「4」を設定します。

2. (a) と同様に，属性エディタウィンドウで，表の属性を設定する。

   例では，初期値を設定しません。

なお，フィールド構成ウィンドウは，関連のあるフィールドや表を COBOL の手続き部でまとめて扱うための集団項目を定義するために使用します。

6.4　画面定義の操作

6.4.1　基本的な編集操作

ここでは，編集時などの基本的な操作について説明します。

(1)　移動

マウス操作の場合

1. 選択されている対象物を移動先までドラッグする。
2. マウスのボタンを離す。

メニュー操作の場合

1. 移動したい対象物を選ぶ。
2. [編集] メニューから［切り取り］を選ぶ。
3. 移動先を選ぶ。
4. [編集] メニューから［貼り付け］を選ぶ。

(2)　複写

マウス操作の場合

1. [Ctrl] キーを押したまま，選択されている対象物を複写したい位置までドラッグする。
2. マウスのボタンを離す。

メニュー操作の場合

1. 複写したい対象物を選ぶ。
2. [編集] メニューから［コピー］を選ぶ。
3. 複写先を選ぶ。
4. [編集] メニューから［貼り付け］を選ぶ。

(3)　削除

キー操作の場合

1. 削除したい対象物を選ぶ。
2. [Delete] キーを押す。

メニュー操作の場合

1. 削除したい対象物を選ぶ。

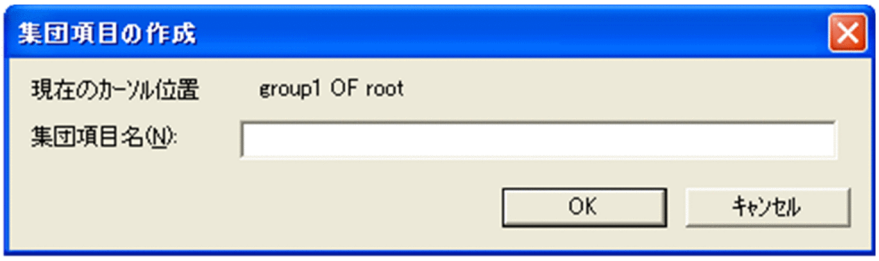
2. [編集] メニューから [削除] を選ぶ。

6.4.2 集団項目の編集

(1) 集団項目の作成

1. [編集] メニューから [集団項目作成] を選ぶ。

   次に示すダイアログボックスが表示されます。



2. 集団項目の名称を入力する。

   集団項目名に何も指定しなかった場合，「FILLER」という名称が仮定されます。

   集団項目は，フィールド構成ウィンドウ上のビューカーソルがある集団項目の下に作成されます。

（例）

現在のビューカーソルが，group1 of root の場合，集団項目名を sample と入力すると，sample of group1 of root が作成されます。

(2) 集団項目の選択

集団項目は，フィールド構成ウィンドウで選択します。

(a) 単数選択

1. 任意の集団項目をクリックする。または矢印キーで，任意の集団項目にエディットカーソルを移動させる。

(b) 複数選択

追加選択

1. [Ctrl] キーを押したまま，任意の集団項目をクリックする。

選択の取り消し

1. [Ctrl] キーを押したまま，選択状態の集団項目を再度クリックする。

範囲選択

範囲選択できるのは，すでに選択されている項目と同じ集団項目に属する項目だけです。

1. 選択する最初の集団項目をクリックする。
2. [Shift] キーを押したまま，選択する最後の集団項目をクリックする。または［Shift］＋矢印キーで，選択する最後の集団項目を選ぶ。
   最初から最後までの，同じ集団項目に属するすべての項目が選択状態になります。

(3) 集団項目の移動

集団項目は，フィールド構成ウィンドウで移動します。

移動の操作は，「6.4.1 基本的な編集操作」を参照してください。

注意事項

- マウスを使用する場合は，移動先に集団項目以外を指定すると，操作は無効になります。また，集団項目を指定する場合は，移動先は指定する集団項目の下になります。
- ［編集］メニューの［切り取り］および［貼り付け］を使用する場合は，移動先は，ビューカーソルがある集団項目の下になります。

(4) 集団項目の複写

集団項目は，フィールド構成ウィンドウで複写します。

複写の操作は，「6.4.1 基本的な編集操作」を参照してください。

注意事項

- マウスを使用する場合は，移動先に集団項目以外を指定すると，操作は無効になります。また，集団項目を指定する場合は，移動先は指定する集団項目の下になります。
- ［編集］メニューの［コピー］および［貼り付け］を使用する場合は，複写先は，ビューカーソルがある集団項目の下になります。

(5) 集団項目の削除

集団項目は，フィールド構成ウィンドウで削除します。

削除の操作は，「6.4.1 基本的な編集操作」を参照してください。

6.4.3 フィールドの編集

(1) フィールドの作成

フォームエディタウィンドウで，任意のサイズのフィールドを作成できます。

1. フィールドの起点となる位置をマウスで選び，目的のサイズになるまでドラッグする。

2. マウスのボタンを離す。

選択状態のフィールドが指定したサイズで作成されます。

また，フィールド構成ウィンドウにフィールド項目が作成されます。

(2) フィールドの選択

フォームエディタウィンドウでフィールドを選択，またはフィールド構成ウィンドウでフィールド項目を選択します。

(a) 単数選択

(i) フォームエディタウィンドウでの操作

1. 任意のフィールドをクリックする。

(ii) フィールド構成ウィンドウでの操作

1. 任意のフィールド項目をクリックする。または矢印キーで，任意のフィールド項目にエディットカーソルを移動させる。

(b) 複数選択

(i) フォームエディタウィンドウでの操作

追加選択

1. [Ctrl] キーを押したまま，任意のフィールドをクリックする。

選択の取り消し

1. [Ctrl] キーを押したまま，選択状態のフィールドを再度クリックする。

範囲選択

1. [Shift] キーを押したまま，ドラッグする。

矩形枠に触れているフィールド，および表はすべて選択状態になります。

(ii) フィールド構成ウィンドウでの操作

追加選択

1. [Ctrl] キーを押したまま，任意のフィールド項目をクリックする。

選択の取り消し

1. [Ctrl] キーを押したまま，選択状態のフィールド項目を再度クリックする。

範囲選択

範囲選択できるのは，すでに選択されている項目と同じ集団項目に属する項目だけです。

1. 選択する最初のフィールド項目をクリックする。
2. [Shift] キーを押したまま，選択する最後のフィールド項目をクリックする。または［Shift］＋矢印キーで，選択する最後のフィールド項目を選ぶ。

最初から最後までの，同じ集団項目に属するすべての項目が選択状態になります。

(3)　フィールドのサイズ変更

フィールドは，フォームエディタウィンドウ，および属性エディタウィンドウでサイズを変更します。最小サイズは日本語項目では２バイト，その他の項目では１バイトで，最大は画面の右端です。

フィールドを複数選択している場合は，サイズを変更できません。

(a)　フォームエディタウィンドウでのサイズ変更

1. 選択状態のフィールドの右端をマウスで選び，目的のサイズになるまでドラッグする。
2. マウスのボタンを離す。
   フィールドのサイズが変更されます。

(b)　属性エディタウィンドウでのサイズ変更

1. サイズテキストボックスにサイズ数（カラム単位）を入力する。
   フィールドのサイズが変更されます。

(4)　フィールドの移動

フォームエディタウィンドウでは，表示位置を移動します。フィールド構成ウィンドウでは，論理的な階層位置を移動します。

移動の操作は，「6.4.1　基本的な編集操作」を参照してください。

注意事項

(i) フォームエディタウィンドウでの操作

- マウスを使用する場合は，フィールドの右端以外を選んでください。
- ［編集］メニューの［切り取り］および［貼り付け］を使用する場合は，選択状態のフィールドが画面内の任意の位置に貼り付けられます。マウスの操作で目的の位置まで移動させてください。

(ii) フィールド構成ウィンドウでの操作

- マウスを使用する場合は，移動先に集団項目以外を指定すると，操作は無効になります。また，集団項目を指定する場合は，移動先は指定する集団項目の下になります。
- ［編集］メニューの［切り取り］および［貼り付け］を使用する場合は，移動先は，ビューカーソルがある集団項目の下になります。

(5)　フィールドの複写

フィールドは，フォームエディタウィンドウ，およびフィールド構成ウィンドウで複写します。

移動の操作は，「6.4.1　基本的な編集操作」を参照してください。

注意事項

(i) フォームエディタウィンドウでの操作

- マウスを使用する場合は，フィールドの右端以外を選んでください。
- ［編集］メニューの［コピー］および［貼り付け］を使用する場合は，選択状態のフィールドが画面内の任意の位置に貼り付けられます。マウスの操作で目的の位置に複写してください。

(ii) フィールド構成ウィンドウでの操作

- マウスを使用する場合は，移動先に集団項目以外を指定すると，操作は無効になります。また，集団項目を指定する場合は，移動先は指定する集団項目の下になります。
- ［編集］メニューの［コピー］および［貼り付け］を使用する場合は，複写先は，ビューカーソルがある集団項目の下になります。

(6)　フィールドの削除

フィールドは，フォームエディタウィンドウ，およびフィールド構成ウィンドウで削除します。

削除の操作は，「6.4.1　基本的な編集操作」を参照してください。

6.4.4　表の編集

(1)　表の作成

表を作成するには，フォームエディタウィンドウにフォーカスがある必要があります。「REPEAT LINE ～」形式の表を作成できます。

1. ［作表］メニューから［表作成］を選ぶ。
   次に示すダイアログボックスが表示されます。



2. 必要項目を定義し，［OK］ボタンを選ぶ。

フォームエディタウィンドウに表が作成され，フィールド構成ウィンドウには表項目が作成されます。

(2) 表の選択

フォームエディタウィンドウで表を選択，またはフィールド構成ウィンドウで表項目を選択します。

(a) 単数選択

(i) フォームエディタウィンドウでの操作

1. 表内の任意の要素をクリックする。

表全体が選択状態になり，指定した表要素がほかの要素と異なる色で強調され，表要素としての選択状態になります。

(ii) フィールド構成ウィンドウでの操作

1. 任意の表項目をクリックする。または矢印キーで，任意の表項目にエディットカーソルを移動させる。

(b) 複数選択

(i) フォームエディタウィンドウでの操作

追加選択

1. ［Ctrl］キーを押したまま，任意の表をクリックする。

選択の取り消し

1. ［Ctrl］キーを押したまま，選択状態の表を再度クリックする。

範囲選択

1. ［Shift］キーを押したまま，ドラッグする。

矩形枠に触れている表，およびフィールドはすべて選択状態になります。

(ii) フィールド構成ウィンドウでの操作

追加選択

1. [Ctrl] キーを押したまま，任意の表項目をクリックする。

選択の取り消し

1. [Ctrl] キーを押したまま，選択状態の表項目を再度クリックする。

範囲選択

範囲選択できるのは，すでに選択されている項目と同じ集団項目に属する項目だけです。

1. 選択する最初の表項目をクリックする。
2. [Shift] キーを押したまま，選択する最後の表項目をクリックする。または [Shift] ＋矢印キーで，選択する最後の表項目を選ぶ。

   最初から最後までの，同じ集団項目に属するすべての項目が選択状態になります。

(3) 表のサイズ変更

表のサイズ変更には，表項目の繰り返し数を変更する方法と，表要素長を変更する方法があります。

なお，表中の要素の DOUBLE，WIDE，LONG などの表示文字サイズは，属性エディタウィンドウで変更します。

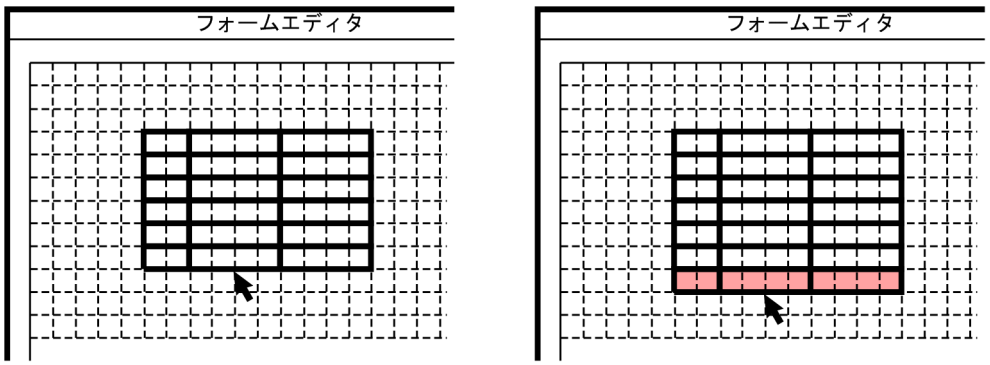
(a) 表項目の繰り返し数の変更

繰り返し数は，縦方向（REPEAT LINE）に変更できます。最小の繰り返し数は 1 であり，最大は画面に収まる範囲までです。

1. 表の外枠の下罫線を選び，目的のサイズになるまで上下にドラッグする。
2. マウスのボタンを離す。

   表項目の繰り返し数は変更され，表のサイズが変更されます。

（例）



(b) 表要素長の変更

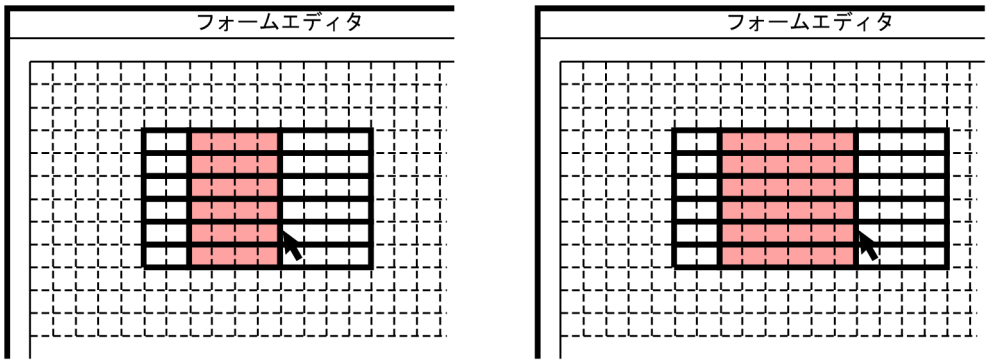
最小のサイズは，日本語項目では 2 バイト，その他の項目では 1 バイトまでで，最大は画面の右端です。

1. 表中の該当する要素に接する縦罫線を選び，目的のサイズになるまで左右にドラッグする。

2. マウスのボタンを離す。

   表要素長は変更され，表のサイズが変更されます。

（例）



注意事項

表要素長のサイズ変更では，外枠の上罫線または左罫線を指定できません。

また，サイズを変更する表要素の右側にある表要素は，サイズは変わらないで，カラム番号だけが変更されます。

(4)　表の移動

フォームエディタウィンドウでは，表示位置を移動できます。フィールド構成ウィンドウでは，論理的な階層位置を移動できます。

移動の操作は，「6.4.1　基本的な編集操作」を参照してください。

注意事項

(i) フォームエディタウィンドウでの操作

- ［編集］メニューの［切り取り］および［貼り付け］を使用する場合は，選択状態の表が画面内の任意の位置に貼り付けられます。マウスの操作で目的の位置まで移動させてください。

(ii) フィールド構成ウィンドウでの操作

- マウスを使用する場合は，移動先に集団項目以外を指定すると，操作は無効になります。また，集団項目を指定する場合は，移動先は指定する集団項目の下になります。
- ［編集］メニューの［切り取り］および［貼り付け］を使用する場合は，移動先は，ビューカーソルがある集団項目の下になります。

(5)　表の複写

表は，フォームエディタウィンドウ，およびフィールド構成ウィンドウで複写します。

複写の操作は，「6.4.1　基本的な編集操作」を参照してください。

注意事項

(i) フォームエディタウィンドウでの操作

- ［編集］メニューの［コピー］および［貼り付け］を使用する場合は，選択状態の表が画面内の任意の位置に貼り付けられます。マウスの操作で目的の位置まで移動させてください。

(ii) フィールド構成ウィンドウでの操作

- マウスを使用する場合は，移動先に集団項目以外を指定すると，操作は無効になります。また，集団項目を指定する場合は，移動先は指定する集団項目の下になります。
- ［編集］メニューの［コピー］および［貼り付け］を使用する場合は，移動先は，ビューカーソルがある集団項目の下になります。

(6)　表の削除

表は，フォームエディタウィンドウ，およびフィールド構成ウィンドウで削除します。

削除の操作は，「6.4.1　基本的な編集操作」を参照してください。

6.4.5　表要素の編集

表要素は，フォームエディタウィンドウで編集します。表要素を編集するには，対象となる表要素が選択状態である必要があります。表要素を選択するには，フォームエディタウィンドウで表を選択します。マウスでポイントした表要素が，ほかの表要素と異なる色で強調され，選択状態になります。

なお，表を複数選択している場合は，表要素を編集できません。

(1)　表要素の追加

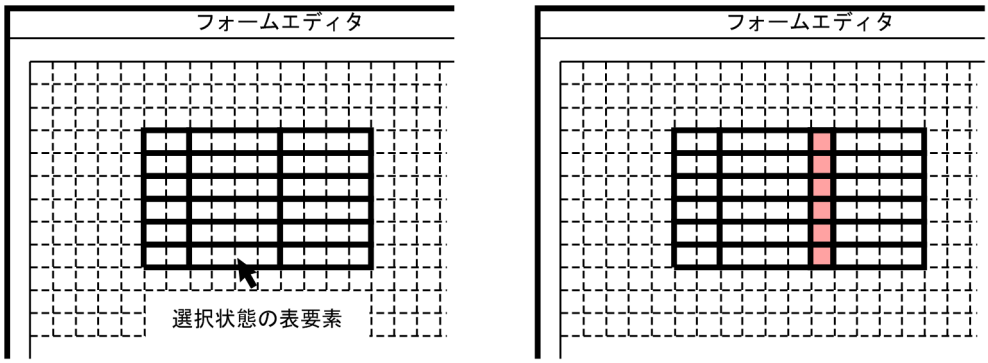
表要素は，フォームエディタウィンドウで追加します。

表を複数選択している場合には，表要素を追加できません。

1. 表要素を追加したい表を選ぶ。
   指定した表要素が選択状態になります。

2. ［作表］メニューから［表要素の追加］を選ぶ。
   指定した表要素の後ろに，1 バイトの新しい表要素が追加されます。

（例）



(2)　表要素の移動

表要素は，フォームエディタウィンドウで移動します。

表を複数選択している場合には，表要素を移動できません。

1. 移動したい表要素を選ぶ。
2. ［作表］メニューから［表要素の切り取り］を選ぶ。
   選択状態の表要素は削除され，一時的に記憶されます。
3. 移動先を選ぶ。
4. ［作表］メニューから［表要素の貼り付け］を選ぶ。
   指定した表要素の後ろに，一時的に記憶されていた表要素が貼り付けられます。

(3)　表要素の複写

表要素は，フォームエディタウィンドウで複写します。

表を複数選択している場合には，表要素を複写できません。

1. 複写したい表要素を選ぶ。
2. ［作表］メニューから［表要素のコピー］を選ぶ。
   選択状態の表要素は一時的に記憶されます。
3. 複写先を選ぶ。
4. ［作表］メニューから［表要素の貼り付け］を選ぶ。
   指定した表要素の後ろに，一時的に記憶されていた表要素が貼り付けられます。

(4)　表要素の削除

表要素は，フォームエディタウィンドウで削除します。

表を複数選択している場合には，表要素を削除できません。

1. 削除したい表要素を選ぶ。

2. [作表] メニューから [表要素の削除] を選ぶ。
   削除を確認するダイアログボックスが表示されます。

3. [OK] ボタンを選ぶ。
   選択状態の表要素は削除されます。削除された表要素の後続の表要素は前詰めに移動します。

6.4.6 罫線の編集

罫線は，フォームエディタウィンドウで編集します。

(1) 罫線を編集するための操作

罫線を編集するには，フォームエディタウィンドウを罫線作成モードに切り替える必要があります。

1. [オプション] メニューから [罫線作成モード] を選ぶ。
   フォームエディタウィンドウが，フィールド作成モードから罫線作成モードに切り替わります。

(2) 罫線の作成

罫線は，フォームエディタウィンドウで作成します。

1. 起点となる個所をマウスで選ぶ。

2. 縦および横にドラッグする。または，斜めにドラッグする。
   縦横にドラッグした場合は，罫線が引けます。また，斜めにドラッグした場合は，矩形が引けます。
   なお，作成した罫線は，フィールド構成ウィンドウ上のビューカーソルがある集団項目の属性情報となります。

(3) 罫線の選択

罫線は，フォームエディタウィンドウで選択します。

(a) 単数選択

1. 任意の罫線をクリックする。

(b) 複数選択

追加選択

1. [Ctrl] キーを押したまま，任意の罫線をクリックする。

範囲選択

1. [Shift] キーを押したまま，マウスをドラッグする。
   矩形枠に触れているすべての罫線が選択状態になります。

(4) 罫線の移動

罫線は，フォームエディタウィンドウで移動します。

移動の操作は，「6.4.1 基本的な編集操作」を参照してください。

注意事項

- ［編集］メニューの［切り取り］および［貼り付け］を使用する場合は，選択状態の罫線が画面内の任意の位置に貼り付けられます。マウスの操作で目的の位置まで移動させてください。

(5) 罫線の複写

罫線は，フォームエディタウィンドウで複写します。

複写の操作は，「6.4.1 基本的な編集操作」を参照してください。

注意事項

- ［編集］メニューの［コピー］および［貼り付け］を使用する場合は，選択状態の罫線が画面内の任意の位置に貼り付けられます。マウスの操作で目的の位置まで移動させてください。

(6) 罫線の削除

罫線は，フォームエディタウィンドウで削除します。

削除の操作は，「6.4.1 基本的な編集操作」を参照してください。

6.4.7 属性の設定

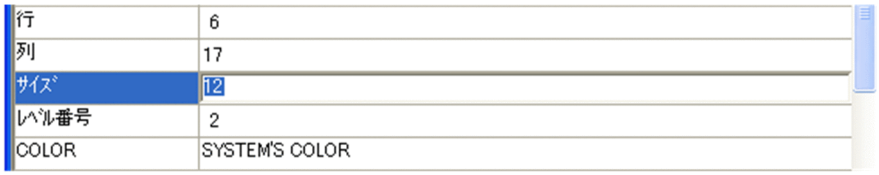
(1) 属性の設定方法

属性エディタウィンドウを使って属性に値を設定します。

属性項目は，次に示す 4 種類のどれかの方法で設定できます。

(a) キーボードから値を入力する方法

1. 属性エディタウィンドウで，設定したい属性項目のリスト 2 の部分を選ぶ。
   次のように，項目が選択され，リスト 2 の部分が入力状態になります。



次の操作方法でも，入力状態になります。

- 項目のリスト 1 の部分をダブルクリックする
- カーソルキーなどで項目を選択し，［Tab］キーを押す

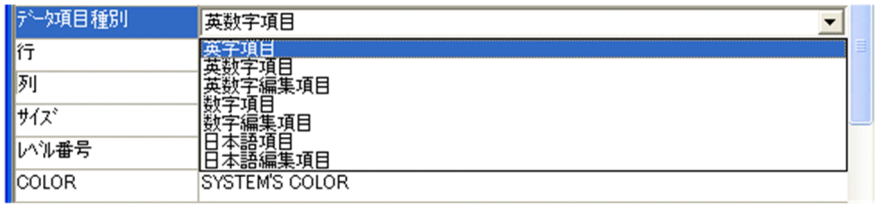
2. リスト 2 の部分に，キーボードで値を入力する。

入力された内容に従って，値が設定，または変更されます。

(b)　ドロップダウンリストから値を選択する方法

1. 属性エディタウィンドウで，設定したい属性項目のリスト 2 の部分を選ぶ。

次のように，項目が選択され，リスト 2 の部分にドロップダウンリストが表示されます。



次の操作でも，ドロップダウンリストを表示できます。

- カーソルキーなどで項目を選択し，［Tab］キーを押す

2. ドロップダウンリストから値を選ぶ。

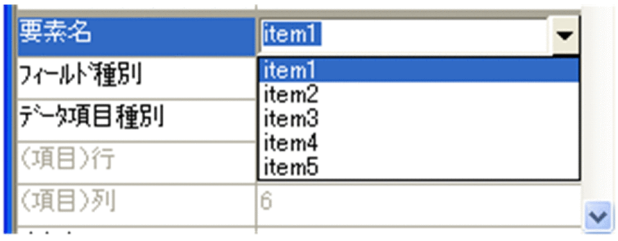
選んだ内容に従って，値が設定，または変更されます。

なお，ドロップダウンリストから値を選ぶ項目では，項目を選択した状態でリスト 1 の部分をダブルクリックすると，ドロップダウンリストの選択肢が順に表示・設定できます。設定したい値が表示されるまで，ダブルクリックしてください。

(c)　キーボードからの入力，またはリストからの選択によって値を設定する方法

1. 属性エディタウィンドウで，設定したい属性項目のリスト 2 の部分を選ぶ。

次の図のように，項目が選択され，リスト 2 の部分にドロップダウンリストが表示されます。また，リスト 2 の部分は，キーボードから属性を入力できる状態になります。



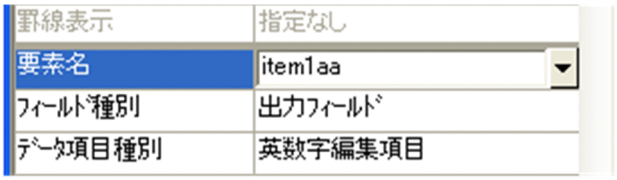
次の操作でも，ドロップダウンリストを表示できます。

- カーソルキーなどで項目を選択し，［Tab］キーを押す

2. ドロップダウンリストから値を選ぶか，キーボードで値を入力する。

選んだ内容，または入力した内容に従って，値が設定，または変更されます。

次の図は，キーボードから値を入力する例です。



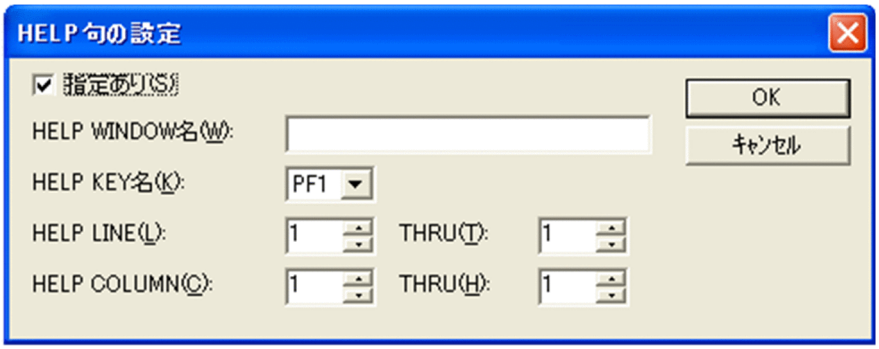
なお，ドロップダウンリストで値を設定する項目では，項目を選択した状態でリスト 1 の部分をダブルクリックすると，ドロップダウンリストの選択肢が順に表示・設定できます。設定したい値が表示されるまで，ダブルクリックしてください。

(d) ダイアログボックスを使って設定する方法

1. 属性エディタウィンドウで，設定したい属性項目のリスト 2 の部分を選ぶ。

ダイアログボックスが表示されます。

次の図は，HELP 句の設定ダイアログボックスが表示された例です。



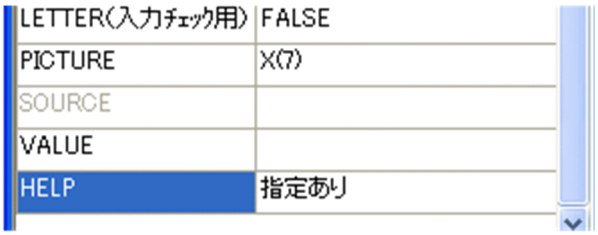
2. ダイアログボックス中の「指定あり」チェックボックスをオンにする。

ダイアログボックス内の項目が設定できる状態になります。

3. ダイアログボックスで属性を設定して，［OK］ボタンを選ぶ。

属性が設定されます。

ダイアログボックスを使って値を設定すると，設定された項目のリスト 2 の部分に，次のように「指定あり」と表示されます。



ダイアログボックスで設定した内容を無効にしたい場合は，もう一度ダイアログボックスを開き，「指定あり」チェックボックスをオフにします。

(2) 属性項目の編集

属性エディタウィンドウに表示される属性のうち，キーボードから直接値を入力できる項目では，テキストの切り取り，コピー，貼り付け，および入力のキャンセルができます。テキストの切り取り，コピー，および貼り付けは，クリップボードを利用して実行されます。

6.5　COBOL ソースファイルの生成

定義した画面から，WINDOW SECTION または SCREEN SECTION 用のデータ部の COBOL ソースファイルを生成する方法を説明します。COBOL ソースファイルは，固定形式および自由形式のどちらでも出力できます。

画面定義は，自由形式の COBOL ソースを次の形式で生成します。

- 1 行の最大長は 80 バイトです。
- コメント行は，行中の最初の文字位置に「*」が生成されます。
- 定数の連結演算子には「&」が生成されます。

1. ［ファイル］メニューから［COBOL 生成］を選ぶ。

   生成先を指定するダイアログボックスが表示されます。

2. ファイル名テキストボックスに生成先のファイル名を入力する。

   ファイルの種類が固定形式（＊.cbl，＊.cob，＊.ocb）の場合は［オプション］ボタンで一連番号を付けるように指定できます。一連番号の指定方法については，「6.6.1　環境設定」を参照してください。

3. ［保存］ボタンを選ぶ。

   COBOL ソースファイルが生成されます。

   拡張子が指定されていない場合は，（.cbl）または（.cbf）が仮定されます。

注意事項

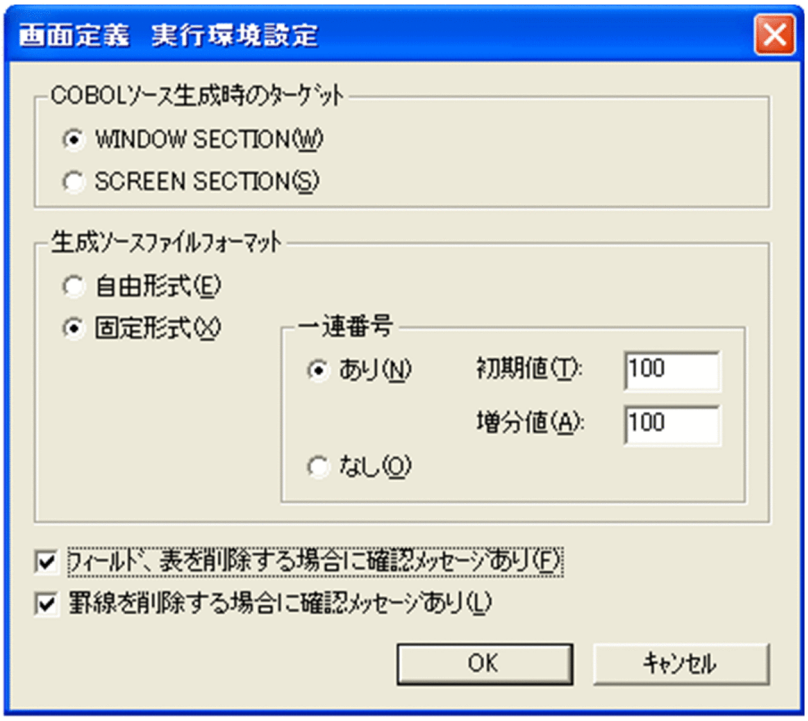
エラーは，1 フィールド内の属性間の範囲だけでチェックされます。このため，生成した COBOL 原始プログラムには，他フィールドや他データ項目間のエラー（SOURCE 句に指定したデータ項目がないなど）が残ることがあります。

6.6　画面定義時のオプション機能

6.6.1　環境設定

1. [オプション] メニューから [画面定義環境設定] を選ぶ。

   次に示すダイアログボックスで環境を設定します。



- COBOL ソース生成時のターゲット

  [ファイル] メニューの [新規作成] を選んで表示される，新規作成ダイアログボックスの [COBOL 生成時のターゲット] の標準値を設定します。

- 生成ソースファイルフォーマット

  [ファイル] メニューの [COBOL 生成] を選んで表示される，COBOL ソースファイルの生成先を指定するダイアログボックスの [ファイルの種別] の標準値を指定します。

- フィールド，表を削除する場合に確認メッセージあり

  このチェックボックスをオンにすると，フィールドや表を削除するときに削除するかどうかを確認するメッセージを表示します。デフォルトはオンです。

- 罫線を削除する場合に確認メッセージあり

  このチェックボックスをオンにすると，罫線を削除するときに削除するかどうかを確認するメッセージを表示するします。デフォルトはオンです。

6.6.2　COBOL ソースファイル生成の許可の設定

COBOL ソースファイルの生成を許可するかどうか設定できます。COBOL ソース生成の許可の設定を切り替える操作を説明します。

1. ［オプション］メニューから［COBOL 生成許可］を選ぶ。

   COBOL ソースの生成が許可されているときは，［COBOL 生成許可］の前に" ∨ "が表示され，［ファイル］メニューの［COBOL 生成］を選べます。

6.6.3　ファイル情報の表示

1. ［ファイル］メニューから［ファイル情報］を選ぶ。

   現在作業中の画面定義ファイル（.wdf）に関する情報が表示されます。

6.7　定義ファイルの操作

6.7.1　定義ファイルの作成

画面定義ファイルを新規に作成して開く操作を説明します。画面定義ウィンドウでは，複数の定義ファイルを開いて画面を定義できます。

1. ［ファイル］メニューから［新規作成］を選ぶ。
   次に示すダイアログボックスが表示されます。



2. 必要な項目を指定する。

   - 画面名
     画面の名称を入力します。画面の名称は必ず指定する必要があります。
   - COBOL 生成時のターゲット
     ターゲットを WINDOW SECTION にするか，SCREEN SECTION にするか選びます。
   - PICTURE 句情報
     CURRENCY SIGN…PICTURE 句で，通貨編集用文字を表す 1 文字の英数字定数を指定します。デフォルトは「¥」です。
     DECIMAL-POINT…PICTURE 句の文字列，および数字定数でのコンマとピリオドの機能を入れ替えたい場合，COMMA を指定します。
   - ［概要設定］ボタン
     画面定義ファイルに関するコメントを入力します。

3. ［OK］ボタンを選ぶ。
   新規の画面定義ファイルが開きます。

6.7.2　定義ファイルのオープン

既存の画面定義ファイルを開く操作を説明します。画面定義ウィンドウでは，複数の定義ファイルを開いて画面を定義できます。

(1)　メニュー操作の場合

1. [ファイル] メニューから［開く］を選ぶ。
   ファイルを指定するダイアログボックスが表示されます。

2. リストボックスから開きたい定義ファイルを選ぶ。またはファイル名テキストボックスにファイル名を入力する。

3. [開く] ボタンを選ぶ。
   指定した画面定義ファイルが開きます。

(2)　マウス操作の場合

- エクスプローラで，開きたい画面定義ファイル（.wdf）をドラッグし，画面定義ウィンドウ中にドロップする。
  指定した画面定義ファイルが開きます。

6.7.3　定義ファイルの保存

画面定義ファイルを保存する操作を説明します。

(1)　上書き保存

1. [ファイル] メニューから［上書き保存］を選ぶ。
   定義した画面情報が，既存の画面定義ファイルにそのまま保存されます。

(2)　名前を付けて保存

1. [ファイル] メニューから［名前を付けて保存］を選ぶ。
   ファイル名を設定するダイアログボックスが表示されるので，ファイル名を入力してください。

2. [保存] ボタンを選ぶ。
   指定した名でファイルが作成されます。ファイル名には拡張子（.wdf）が自動的に付きます。

7 ファイル／レコード定義

COBOL2002 では，COBOL プログラムで使用するファイル／レコード仕様を定義できます。また，定義したファイル／レコード仕様から FILE SECTION（ファイル節），WORKING-STORAGE SECTION（作業場所節），および LOCAL-STORAGE SECTION（局所場所節）のデータ部の COBOL ソースを生成できます。

この章では，ファイル／レコード仕様の定義，および COBOL ソース生成の操作について説明します。

ファイル／レコード定義では，ラージファイル形式の ISAM ファイルは作成できません。

ファイル定義ファイル，およびレコード定義ファイルは，Windows(x86) COBOL2002 と Windows(x64) COBOL2002 間で共用できます。

7.1 概要

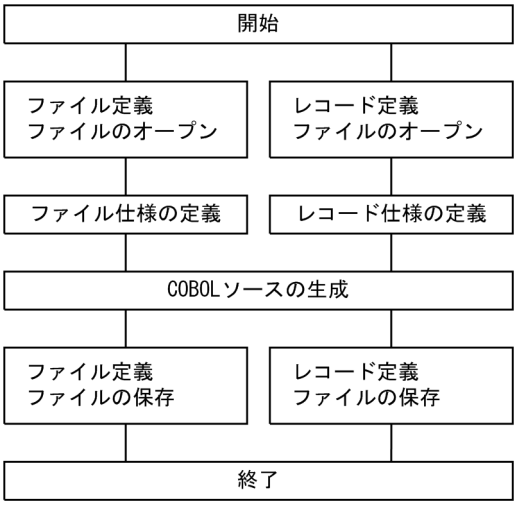
ファイル／レコード定義とは，COBOL プログラムで使用するファイル仕様，およびレコード仕様を設定するツールのことです。

ファイル／レコード定義を使うと，ファイル仕様やレコード仕様を GUI を使って設定できます。また，設定した仕様を基に，FILE SECTION（ファイル節），WORKING-STORAGE SECTION（作業場所節），および LOCAL-STORAGE SECTION（局所場所節）のデータ部の COBOL ソースファイルを生成できます。生成された COBOL ソースファイルは，COBOL プログラム中に COPY 文で取り込みます。

7.1.1 ファイル／レコード定義の流れ

ファイル／レコード定義の流れを次に示します。

図 7-1 ファイル／レコード定義の流れ



ファイル仕様を定義する場合はファイル定義ファイルを，レコード仕様を定義する場合はレコード定義ファイルをオープンします。また，ファイル仕様とレコード仕様を同時に定義する場合は，ファイル定義ファイルとレコード定義ファイルをオープンします。

7.1.2 使用するファイル

(1) ファイル定義ファイル

ファイル定義ファイル（.flf）は，設定したファイル定義の情報を格納するファイルです。

(2) レコード定義ファイル

レコード定義ファイル（.rdf）は，設定したレコード定義の情報を格納するファイルです。

(3) COBOL データ定義ファイル（.cbl，.cbf，.cob，.ocb，.ocf）

ファイル節，作業場所節，および局所場所節の定義部の COBOL ソースファイルです。これらのソースファイルはファイル／レコード定義が自動生成したファイルであり，COBOL の COPY 文で取り込む必要があります。

7.1.3 予約語の扱い

ファイル／レコード定義では，COBOL 予約語として COBOL コンパイラと同等の予約語を認識します。

7.2　ファイル／レコード定義の開始と終了

ファイル／レコード定義の開始と終了の方法について説明します。

7.2.1　ファイル／レコード定義の開始

ファイル／レコード定義を開始するには，ファイル定義ファイル，またはレコード定義ファイルを指定して開始する方法と，定義ファイルを指定しないで開始する方法があります。

(1)　ファイル／レコード定義ファイルを指定する方法

次のどれかの方法で，ファイル／レコード定義を開始できます。

- 開発マネージャのツリービューウィンドウから，ファイル定義ファイル（.flf），またはレコード定義ファイル（.rdf）をダブルクリックする。
- 開発マネージャのツリービューウィンドウから，ファイル定義ファイル（.flf），またはレコード定義ファイル（.rdf）を選び，［ファイル］メニューから［開く］を選ぶ。
- エクスプローラから，ファイル定義ファイル（.flf），またはレコード定義ファイル（.rdf）をダブルクリックする。

  指定した定義ファイルが開かれ，ファイル／レコード定義ウィンドウが表示されます。

(2)　ファイル／レコード定義ファイルを指定しない方法

次のどれかの方法で，ファイル／レコード定義を開始できます。

- 開発マネージャのウィンドウの［ファイル］メニューから［新規作成］を選び，ファイルの種類から「ファイル定義ファイル」または「レコード定義ファイル」を選ぶ。
- Windows のプログラム一覧にある［COBOL2002］下のメニューから［ファイル・レコード定義 for COBOL2002］を選ぶ。

  ファイル／レコード定義ウィンドウが表示されます。

なお，ファイル／レコード仕様を定義するには，ファイル／レコード定義ウィンドウで定義ファイルを開く必要があります。「7.4　定義ファイルの操作」を参照してください。

7.2.2　ファイル／レコード定義の終了

1. ファイル／レコード定義ウィンドウの［ファイル］メニューから［ファイル／レコード定義の終了］を選ぶ。

   設定したファイル／レコード仕様情報を保存していない場合には，ファイルを保存するかどうかを問い合わせるダイアログボックスが表示されるので，ファイルを保存してください。

なお，新規に設定したファイル／レコード仕様情報を保存しないで終了した場合，設定した情報は失われます。「7.4　定義ファイルの操作」を参照し，ファイル名を付けて保存してください。

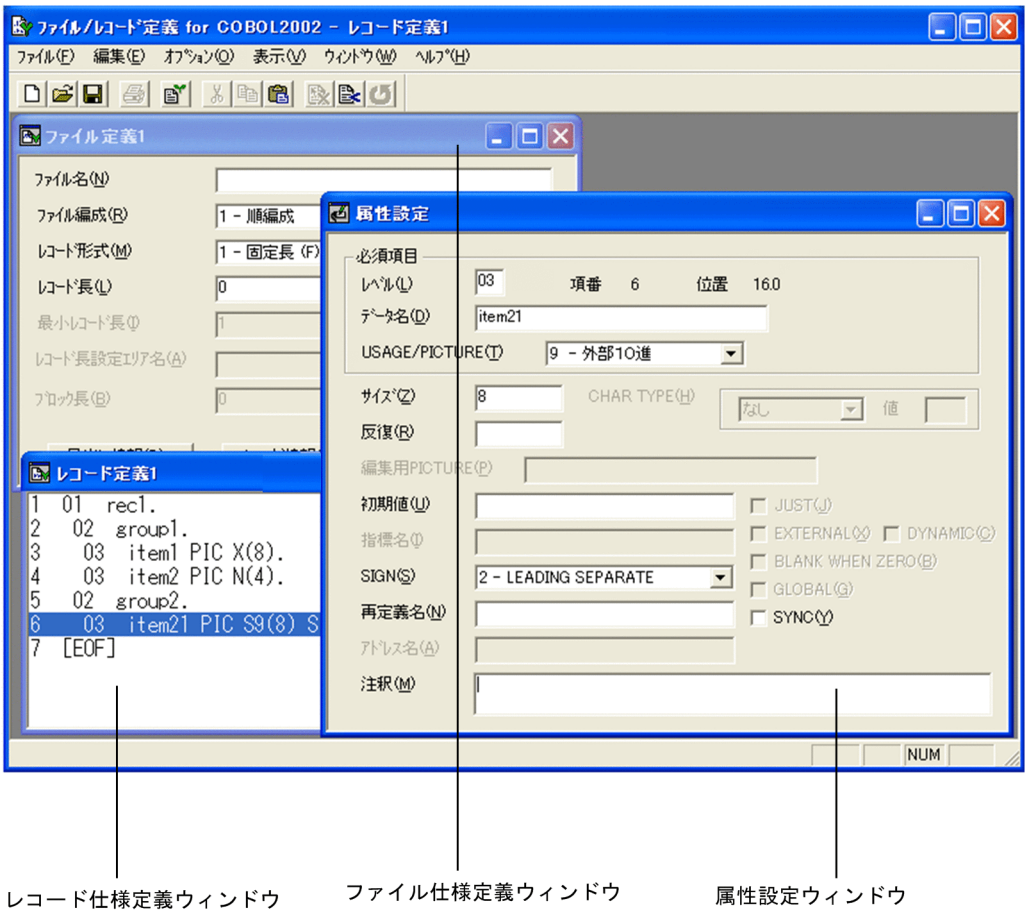
7.3　ファイル／レコード定義ウィンドウ

ファイル／レコード定義ウィンドウについて説明します。

7.3.1　ファイル／レコード定義ウィンドウの構成

ファイル／レコード定義ウィンドウの例を次に示します。

図 7-2　ファイル／レコード定義ウィンドウ



(1)　メニューバー

ファイル／レコード定義ウィンドウのメニューの種類と機能を次に示します。

表 7-1　ファイル／レコード定義ウィンドウのメニュー一覧

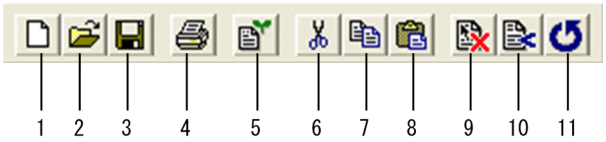
メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	新しくファイル／レコード定義ファイルを作成する。
	開く(O)	既存のファイル／レコード定義ファイルを開く。
	閉じる(C)	既存のファイル／レコード定義ファイルを閉じる。

メニューバー	メニュー項目	機能
	上書き保存(S)	設定した情報を作業中のファイル／レコード定義ファイルに保存する。
	名前を付けて保存(A)	設定した情報をファイル／レコード定義ファイルの名前を付けて保存する。
	ファイル情報(I)	作業中のファイル（.flf または.rdf）の情報を表示する。
	ページの設定(U)	印刷の設定を変更する。
	印刷プレビュー(V)	ページ全体を印刷時のイメージで表示する。
	印刷(P)	作業中のファイル（.flf または.rdf）を印刷する。
	COBOL 生成(G)	COBOL ソースファイルを生成する。
	ISAM ファイル生成(E)	ISAM ファイルを生成する。
	CSV 形式で出力(T)	作業中のファイル（.flf または.rdf）を CSV 形式で出力する。
	ファイル／レコード定義の終了(X)	ファイル／レコード定義ウィンドウを閉じる。
編集(E)	切り取り(T)	選択したデータ項目または文字列を切り取って一時的なエリアに保存する。
	コピー(C)	選択したデータ項目または文字列をコピーして一時的なエリアに保存する。
	貼り付け(P)	一時的なエリアのデータ項目または文字列を選択した位置に貼り付ける。
	挿入(A)	選択した位置の直前にデータ項目を挿入する。
	削除(D)	選択されたデータ項目を削除する。
	取消(U)	直前の挿入／削除を取り消す。
	検索(F)	指定した文字列を検索する。
	置換(E)	指定した文字列を新しい文字列に変更する。
	位置付け(L)	指定した項番を選択状態にする。
オプション(O)	ファイル／レコード定義環境設定(E)	ファイル／レコード定義の実行環境を設定する。
	COBOL 生成許可(A)	COBOL ソースファイルの生成を許可する。
表示(V)	ツールバー(T)	ツールバーの表示を制御する。
	ステータスバー(S)	ステータスバーの表示を制御する。
ウィンドウ(W)	重ねて表示(C)	画面を重ねて表示する。
	並べて表示(T)	画面を並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	属性設定を開く(B)	属性設定ウィンドウを開く。

メニューバー	メニュー項目	機能
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

(2)　ツールバー

ツールバーのボタンを選ぶとメニューと同じ操作ができます。ツールバーの構成と各ボタンに割り当てられているメニューを次に示します。



1. [ファイル] メニューの [新規作成]
2. [ファイル] メニューの [開く]
3. [ファイル] メニューの [上書き保存]
4. [ファイル] メニューの [印刷]
5. [ファイル] メニューの [COBOL 生成]
6. [編集] メニューの [切り取り]
7. [編集] メニューの [コピー]
8. [編集] メニューの [貼り付け]
9. [編集] メニューの [削除]
10. [編集] メニューの [挿入]
11. [編集] メニューの [取消]

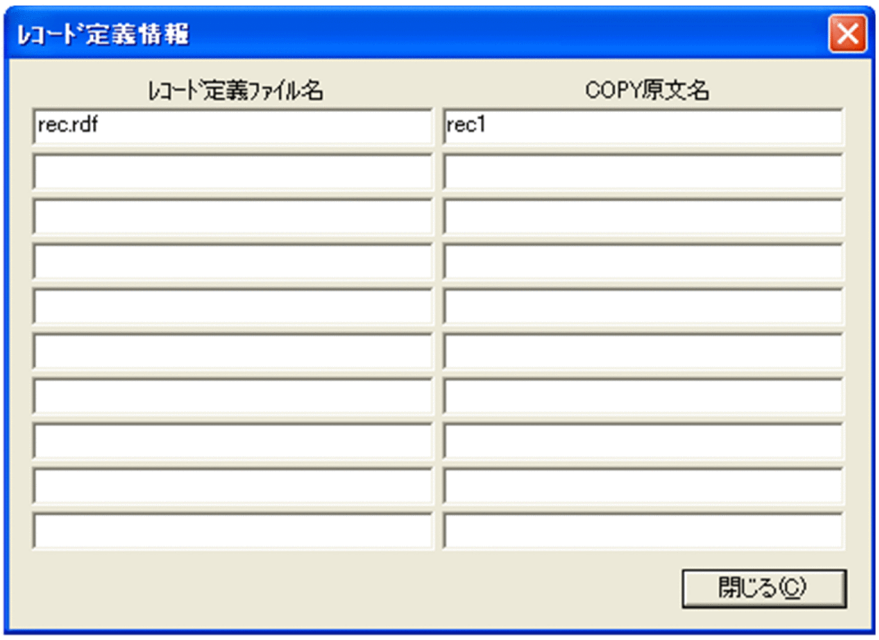
7.3.2　ファイル仕様定義ウィンドウ

ファイル仕様を定義するウィンドウを次に示します。ファイル仕様定義ウィンドウは，ファイル定義ファイル（.flf）の数と同じ数だけ表示されます。

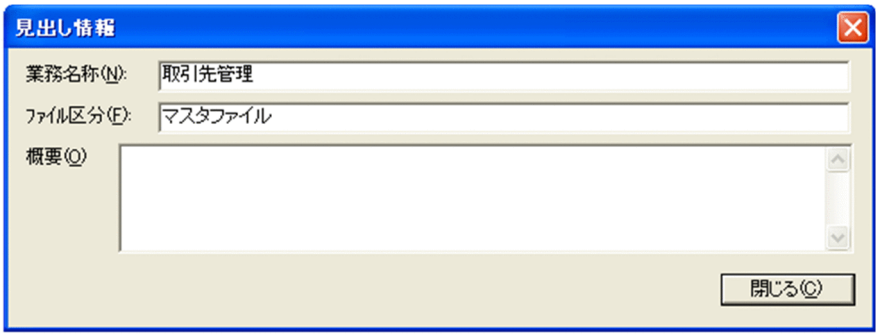
図 7-3　ファイル仕様定義ウィンドウ



ファイル仕様定義ウィンドウの［レコード情報］ボタンを選ぶと，次に示すダイアログボックスが表示されます。



ファイル仕様定義ウィンドウの［見出し情報］ボタンを選ぶと，次に示すダイアログボックスが表示されます。



ファイル仕様定義ウィンドウの［ISAM 情報］ボタンを選ぶと，ISAM ファイルを作成するためのダイアログボックスが表示されます。詳細は「7.6　ISAM 情報」を参照してください。

次に，ファイル仕様定義で定義する項目について説明します。

項目は，キーおよびマウスで定義できます。また，ファイル仕様の定義時には，COBOL のシンタックスレベルのエラーがチェックされます。

ファイル名

COBOL プログラム内で使用するファイル名称を指定します。

ファイル編成

ファイル編成（順編成，相対編成，索引順編成，テキスト編成）を指定します。

レコード形式

次の形式からレコードの形式を指定します。

```
1  F      固定長
2  FB     ブロック化固定長
3  V      可変長
4  VB     ブロック化可変長
5  S      スパンド
6  U      不定長
```

レコード長

レコード形式が，F，FB，V，VB のときだけ指定します。

ISAM ファイル作成時には，必ず指定する必要があります。

最小レコード長

レコード形式が，V，VB のときだけ指定します。

ISAM ファイル作成時には，必ず指定する必要があります。

EXTERNAL

外部属性の有無を指定します。デフォルトは，チェックボックスがオフになっています。

GLOBAL

大域属性の有無を指定します。デフォルトは，チェックボックスがオフになっています。

レコード長設定エリア名

レコード形式が，V，VB のときだけ有効です。COBOL プログラムの実行時に，可変長レコードのサイズを設定するデータ項目名称を記入します。

ブロック長

レコード形式が，FB，VB のときだけ指定します。

レコード定義ファイル名，COPY 原文名

このファイル仕様に関連するレコード仕様を定義したレコード定義ファイル（.rdf）の名称，およびそこから生成される COBOL ソースファイル名を 10 個まで設定できます。

レコード定義ファイル名には，「.rdf」以外の拡張子は指定できません。

COPY 原文名には，「.cbl」「.cob」「.ocb」「.cbf」「.ocf」以外の拡張子は指定できません。

レコード定義ファイル名または，COPY 原文名に誤りがある場合は，メッセージボックスに次のエラーメッセージが表示されます。

レコード定義ファイルの場合

　レコード定義ファイル名として正しい文字を入れてください。

COPY 原文名の場合

　COPY 原文名として正しい文字を入れてください。

また，これらの項目は省略できます。COPY 原文名を指定しなかった場合，レコード定義ファイル名の拡張子を（.cbl）にしたものが仮定されます。

なお，レコード定義ファイル名をダブルクリックすると，そのレコード仕様定義を開くことができます。

業務名称

対象業務の名称を設定します。

ファイル区分

マスタ，トランザクション，蓄積ファイルなどファイルの種別を設定します。

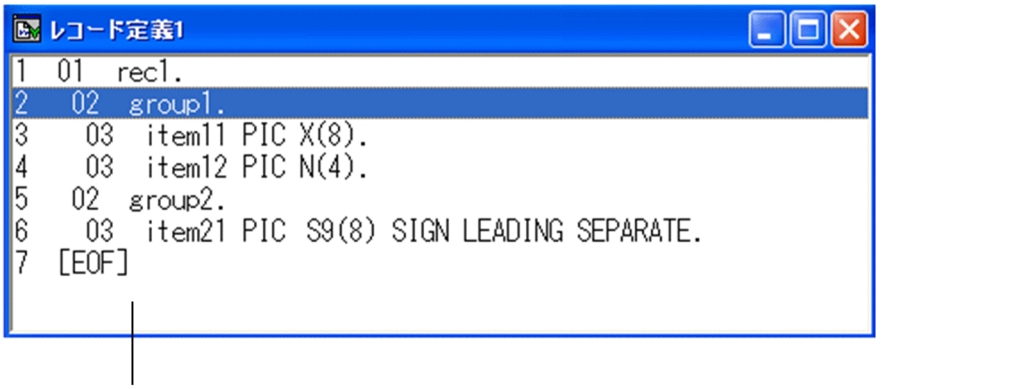
概要

ファイルに関する概要を設定します。

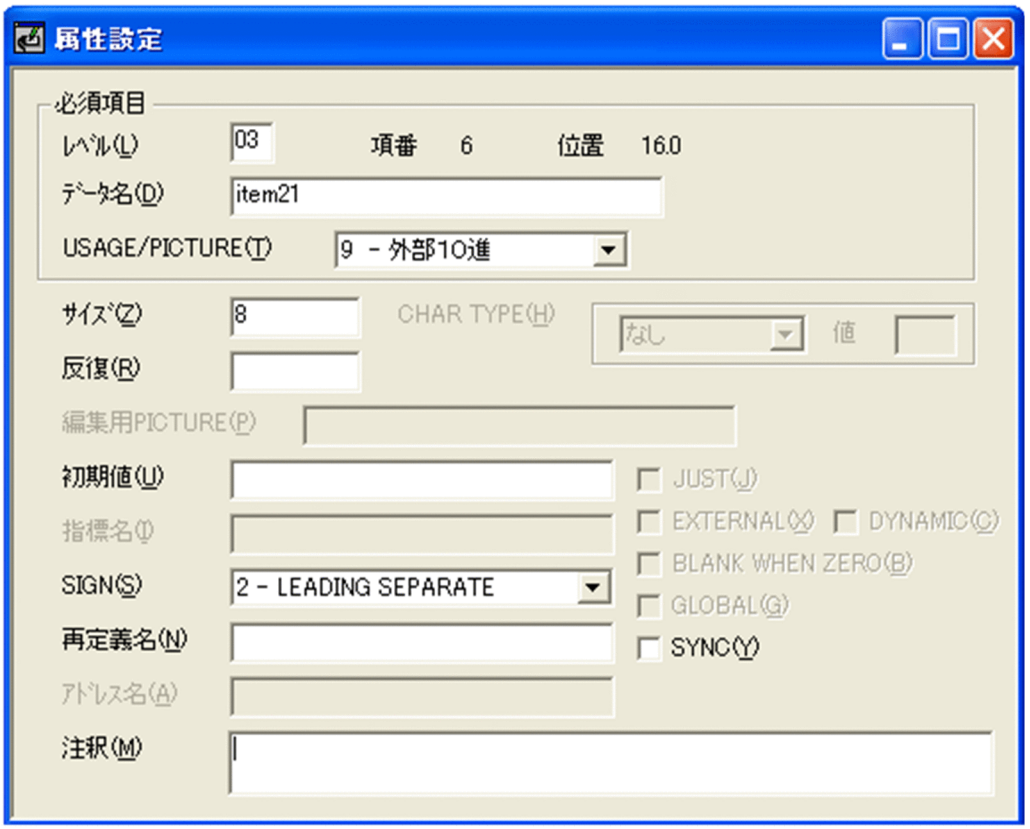
7.3.3　レコード仕様定義ウィンドウ

レコード仕様を定義するウィンドウを次に示します。レコード仕様定義ウィンドウは，レコード定義ファイル（.rdf）の数と同じ数だけ表示されます。

図 7-4　レコード仕様定義ウィンドウ



レコード定義ウィンドウ



レコード仕様定義ウィンドウには，レコード定義ウィンドウと属性設定ウィンドウが表示されます。属性設定ウィンドウでレコード仕様を定義すると，データ項目が自動的にレコード定義ウィンドウに表示されます。

(1)　レコード定義ウィンドウ

レコード定義ウィンドウには，属性設定ウィンドウで定義した情報が表示されます。また，このウィンドウでは，データ項目を編集できます。データ項目の編集の操作は，「7.5　データ項目の編集」を参照してください。

(2) 属性設定ウィンドウ

属性設定ウィンドウでは，レコード仕様をキーおよびマウスで定義します。レコード仕様の定義時には，COBOL のシンタックスレベルのエラーがチェックされます。

次に，属性設定ウィンドウの項目について説明します。

レベル

1 から 49 までのレベル番号を設定します。

項番

レコード定義ウィンドウ中の対応するデータ項目の左端の数字と対応しています。

位置

データ項目のレコード先頭からのバイト数が表示されます。

ただし，サイズに最大値を指定した場合は，位置の表示は保証しません。

データ名

データ項目の名称を設定します。デフォルトは「FILLER」です。

USAGE/PICTURE

次の属性の中から，データ項目のタイプを指定します。

```
X  英数字
E  外部浮動
A  英字
1  内部ブール
N  日本語
8  外部ブール
G  集団項目
XE 英数字編集
B  2進
Z  数字編集
P  内部10進
NE 日本語編集
9  外部10進
T  アドレス
F  単精度浮動
I  インデックス
D  倍精度浮動
```

サイズ

データ項目に必要なサイズを設定します。設定方法は，「USAGE/PICTURE」ドロップダウンリストで選択した項目によって異なります。

・英数字，英字，日本語：文字数を指定する

・2 進，内部 10 進，外部 10 進：けた数を指定する

（p.q または p,q：p は整数部，q は小数部）

・内部ブール，外部ブール：ブール文字数を指定する

・その他：設定できない

反復

項目の反復回数を設定します。

編集用 PICTURE

データ項目のタイプが編集項目（英数字編集，数字編集，日本語編集）と外部浮動のとき PICTURE 文字列を設定します。

データ項目のタイプは，USAGE/PICTURE で設定します。

初期値

データ項目の初期値を設定します。

指標名

反復するデータ項目に指標を付けたいときに指標名を設定します。

SIGN

次の情報の中から符号情報を指定します。

```
1  SIGN なし
2  SIGN あり
3  SIGN IS LEADING SEPARATE
4  SIGN IS TRAILING SEPARATE
5  SIGN IS LEADING
6  SIGN IS TRAILING
```

再定義名

データ領域を再定義する場合のデータ名を設定します。

アドレス名

データをアドレス操作する場合，アドレス名を設定します。

CHAR TYPE

レコードを出力するときに，データ項目の前に機能キャラクタを自動挿入することを指定します。次の指定ができます。

```
KEIS
POINT-l          値  0～127
GAP-m            値  0～511
FORMAT-n         値  0～127
INTERVAL-i       値  0～127
WIDE
```

JUST

右にけた寄せをするかどうかを指定します。デフォルトは，チェックボックスがオフになっています。

BLANK WHEN ZERO

項目の値が 0 のときに空白にするかどうかを指定します。デフォルトは，チェックボックスがオフになっています。

EXTERNAL

外部属性の有無を指定します。ありを指定した場合，DYNAMIC かどうかも指定します。デフォルトは，チェックボックスがオフになっています。

GLOBAL

大域属性の有無を指定します。デフォルトは，チェックボックスがオフになっています。

SYNC

計算機の固有の境界に従った項目の配置をするかどうかを指定します。デフォルトは，チェックボックスがオフになっています。

注釈

データ項目に関する注釈を設定します。

7.4　定義ファイルの操作

ファイル／レコード仕様を定義するには，ファイル定義ファイル，またはレコード定義ファイルが開いている必要があります。既存の定義ファイルを指定してファイル／レコード定義を開始した場合は，定義ファイルは自動的に開きます。それ以外の方法で定義を開始した場合は，ファイル／レコード定義のウィンドウで定義ファイルを開く必要があります。

定義ファイルを開くには，新規に作成する方法と既存のファイルを開く方法があります。

また，ファイル／レコード定義のウィンドウでは，複数の定義ファイルを開いて，同時にファイル仕様とレコード仕様を定義できます。

7.4.1　定義ファイルの作成

1. ［ファイル］メニューから［新規作成］を選ぶ。

   次に示すダイアログボックスが表示されます。



2. 必要な項目を指定する。

   - 定義の種別

     定義ファイルをファイル仕様にするか，レコード仕様にするか選びます。

   - PICTURE 句情報

     CURRENCY SIGN…PICTURE 句で，通貨編集用文字を表す 1 文字の英数字定数を指定します。デフォルトは「¥」です。

     DECIMAL-POINT…PICTURE 句の文字列，および数字定数でのコンマとピリオドの機能を入れ替えたい場合，COMMA を指定します。

3. ［OK］ボタンを選ぶ。

   新規の定義ファイルが開きます。

7.4.2　定義ファイルのオープン

(1)　メニュー操作の場合

1. [ファイル] メニューから［開く］を選ぶ。
   ファイルを指定するダイアログボックスが表示されます。

2. リストボックスから目的の定義ファイルを選ぶ。またはファイル名テキストボックスに定義ファイル名を入力する。

3. [開く] ボタンを選ぶ。
   指定した定義ファイルが開きます。

(2)　マウス操作の場合

1. エクスプローラで，ファイル定義ファイル（.flf）またはレコード定義ファイル（.rdf）をドラッグし，ファイル／レコード定義のウィンドウ中にドロップする。

7.4.3　定義ファイルの保存

(1)　上書き保存

1. [ファイル] メニューから［上書き保存］を選ぶ。
   定義したファイル／レコード仕様情報が，既存の定義ファイルにそのまま保存されます。

(2)　名前を付けて保存

1. [ファイル] メニューから［名前を付けて保存］を選ぶ。
   ファイル名を設定するダイアログボックスが表示されるので，ファイル名を入力してください。

2. [保存] ボタンを選ぶ。
   指定した名でファイルが作成されます。ファイル名には拡張子（.flf または.rdf）が自動的に付きます。

7.5　データ項目の編集

レコード定義ウィンドウでデータ項目を編集する操作を説明します。

7.5.1　データ項目の選択

(1)　単数選択

1. 任意の項目をクリックする。または［↑］キー，または［↓］キーでカーソルを移動させる。

(2)　複数選択（範囲選択）

1. 選択する最初のデータ項目をクリックする。
2. [Shift] キーを押したまま，最後のデータ項目をクリックする。または［Shift］＋矢印キーで最後のデータ項目を選ぶ。
   最初から最後の項番までのすべての項目が選択状態になります。

7.5.2　データ項目の挿入

1. 挿入したい位置の項目を選ぶ。
2. [編集] メニューから［挿入］を選ぶ。または，[Insert] キーを押す。

指定したデータ項目の直前に新しいデータ項目が挿入されます。

挿入されるデータ項目の初期値は，"レベル番号 FILLER PIC X(1)."です。レベル番号は，指定したデータ項目のレベル番号と同じになります。

7.5.3　データ項目の属性の編集

データ項目の属性を編集するには，レコード定義ウィンドウから編集するデータ項目を選び，属性設定ウィンドウで属性を定義します。

1. レコード定義ウィンドウから編集するデータ項目を選ぶ。
   選んだデータ項目の属性が属性設定ウィンドウに表示されます。
2. 属性設定ウィンドウでデータ項目の属性を設定する。
3. [Tab] キーを押す。

設定した属性情報が指定したデータ項目に反映されます。

7.5.4　文字列の複写

データ項目の属性を編集するとき，データ名や初期値など，既存のデータ項目の一部を複写して編集できます。ただし，この場合，複写先の該当する文字列は上書きされるので注意してください。

1. レコード定義ウィンドウから複写元のデータ項目を選ぶ。
   選んだデータ項目の属性が属性設定ウィンドウに表示されます。
2. 属性設定ウィンドウから複写したい文字列を選ぶ。
   データ名，初期値などの欄から，複写したい文字列の一部またはすべてを選びます。
3. [編集] メニューの [コピー] を選ぶ。
4. 複写先のデータ項目を選ぶ。
5. [編集] メニューの [貼り付け] を選ぶ。
6. [Tab] キーを押す。
   複写先のデータ項目中のデータ名，初期値などが指定した文字列で上書きされます。

7.5.5　データ項目の移動

移動の操作は，「6.4.1　基本的な編集操作」を参照してください。

注意事項

(i) マウス操作の場合

- データ項目の項番をマウスでポイントします。移動先は，マウスのボタンを離した位置のデータ項目の直前になります。

(ii) メニュー操作の場合

- データ項目を移動するには，データ項目を選びます。
- 文字列を移動するには，データ名称を選びます。

7.5.6　データ項目の複写

複写の操作は，「6.4.1　基本的な編集操作」を参照してください。

注意事項

- 複写は，［編集］メニューの［コピー］および［貼り付け］を使用して操作します。
- データ項目を複写するには，データ項目を選びます。
- 文字列を複写するには，データ名称を選びます。

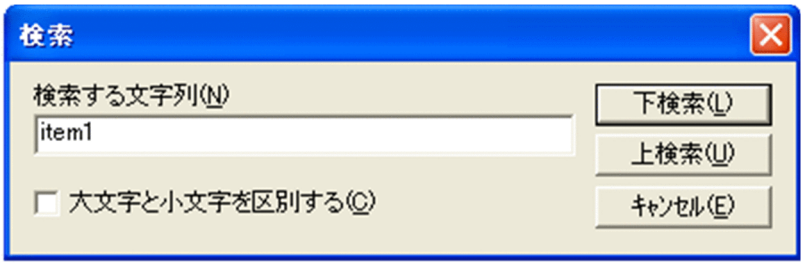
7.5.7　データ項目の削除

削除の操作は，「6.4.1　基本的な編集操作」を参照してください。

7.5.8　データ項目の検索

検索できるのは，文字列を値に持つデータ名称だけです。

1. ［編集］メニューから［検索］を選ぶ。
   次に示すダイアログボックスが表示されます。

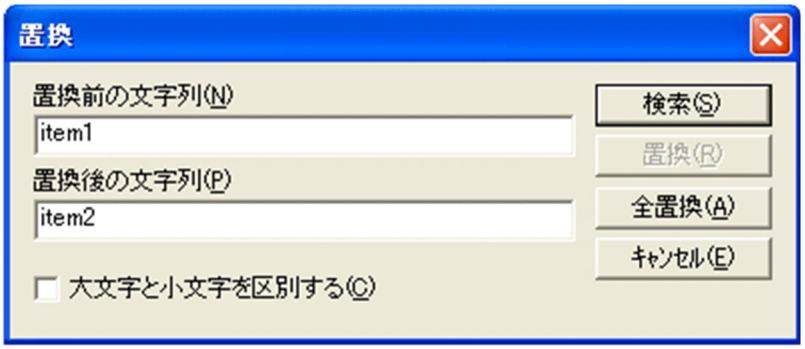


2. 検索したい文字列を入力し，大文字と小文字を区別して検索するかどうかを指定する。
3. ［下検索］または［上検索］ボタンを選ぶ。
   - ［下検索］
     カーソルがある位置から下方向に検索し，見付かった文字列の先頭にカーソルが移動します。
   - ［上検索］
     カーソルがある位置から上方向に検索し，見付かった文字列の先頭にカーソルが移動します。

7.5.9　データ項目の置換

置換名称だけです。

1. ［編集］メニューから［置換］を選ぶ。
   次に示すダイアログボックスが表示されます。



2. 置換前の文字列と置換後の文字列を入力し，大文字と小文字を区別して検索するかどうかを指定する。

3. [検索] ボタンを選ぶ。

   カーソルがある位置から下方向に検索し，見付かった文字列の先頭にカーソルが移動します。

4. [置換] または [全置換] ボタンを選ぶ。

   - [置換]

     文字列を置換後，カーソルがある位置から下方向に検索し，次に見付かった文字列の先頭にカーソルが移動します。

   - [全置換]

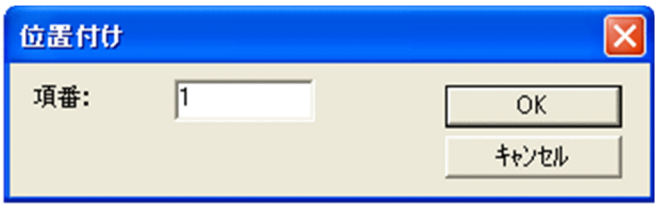
     置換前の文字列と一致するすべての文字列を置換して，最後に置換した位置にカーソルが移動する。

7.5.10　データ項目の位置付け

レコード仕様定義ウィンドウの先頭に表示されるデータ項目を変更できます。

1. [編集] メニューから [位置付け] を選ぶ。

   次に示すダイアログボックスが表示されます。



2. [項番] に表示したいデータ項目の項番を入力する。

3. [OK] ボタンを選ぶ。

   指定した項番のデータ項目がウィンドウの先頭に表示されます。

   指定した項番のデータ項目がウィンドウに表示されていない場合は，ウィンドウに表示される位置まで画面がスクロールします。指定した項番のデータ項目がウィンドウの先頭に表示できる場合は，スクロールをして先頭に表示します。ウィンドウの先頭に表示できない場合は，スクロールが可能な位置まで画面をスクロールできます。

7.5.11　取消

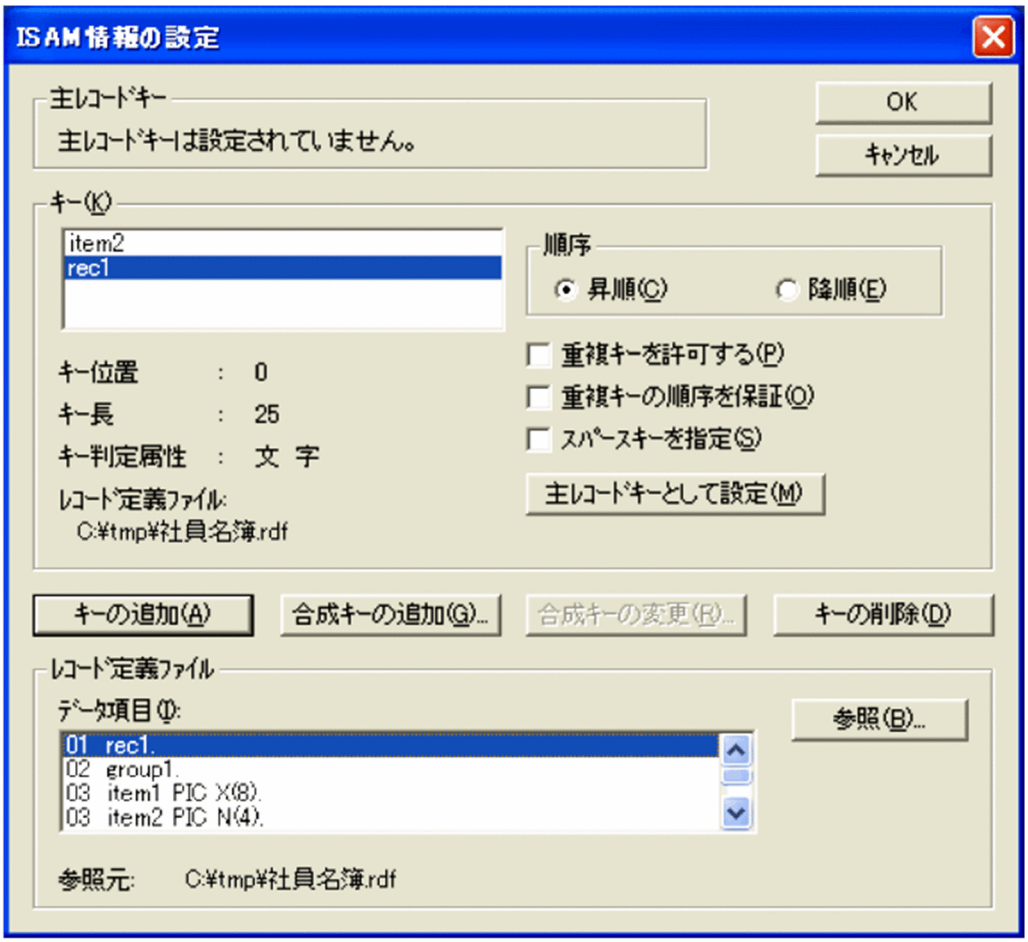
1. [編集] メニューから [取消] を選ぶ。
   直前の挿入または削除の操作が取り消されます。

7.6　ISAM 情報

ここでは，ISAM ファイルを作成するための情報を設定する方法，および ISAM ファイル作成の操作について説明します。

7.6.1　ISAM 情報の設定

ファイル仕様定義ウィンドウの［ISAM 情報］ボタンを選ぶと，次に示す ISAM 情報の設定ダイアログボックスが表示されます。



このダイアログボックスで定義する情報について説明します。

主レコードキー

主レコードキーとして登録されているキーの名称が表示されます。

キー

登録されているキーの一覧が表示されます。目的のキーを選んで，キー情報の変更，主レコードキーとして設定，および削除などの操作ができます。

キー位置

キー一覧で選択したキーの開始位置が表示されます。

キー長

キー一覧で選択したキーの長さが表示されます。

キー判定属性

キー一覧で選択したキーの判定属性が表示されます。

レコード定義ファイル

キー一覧で選択したキーが定義されているレコード定義ファイルの名称が表示されます。

順序

キー一覧で選択したキーのキー順序を指定します。昇順を指定してください。降順を指定した場合，OPEN 文でエラーとなります。

重複キーを許可する

キー一覧で選択したキーの重複キーを許可するとき指定します。

重複キーの順序を保証

キー一覧で選択したキーの重複キーの順序を保証するとき指定します。

スパースキーを指定

キー一覧で選択したキーのスパースキーを指定するかどうかを指定します。指定する場合には，スパース文字を 16 進数 2 けたで指定します。

[主レコードキーとして設定] ボタン

このボタンを選ぶと，キー一覧で選択したキーを主レコードキーとして登録できます。

[キーの追加] ボタン

データ項目一覧に表示されているデータ項目の中から目的のデータ項目を選び，このボタンを選ぶと，選択したデータ項目をキーとして登録できます。

注意事項

次の項目はキーとして登録できません。

- 8 バイトの 2 進項目
- 内部ブール
- 外部ブール
- 英数字編集
- 日本語編集
- アドレス
- インデックス

［合成キーの追加］ボタン

このボタンを選ぶと，合成キーの追加ダイアログボックスが表示されます。このダイアログボックスでは，新規の合成キーが登録できます。

［合成キーの変更］ボタン

キー一覧に表示されているキーの中から目的のキーを選び，このボタンを選ぶと，合成キーの変更ダイアログボックスが表示されます。このダイアログボックスでは，合成キーを変更できます。

［キーの削除］ボタン

キー一覧に表示されているキーの中から目的のキーを選び，このボタンを選ぶと，選択したキーをキー一覧から削除できます。

データ項目

参照しているレコード定義ファイルに定義されているデータ項目の一覧が表示されます。

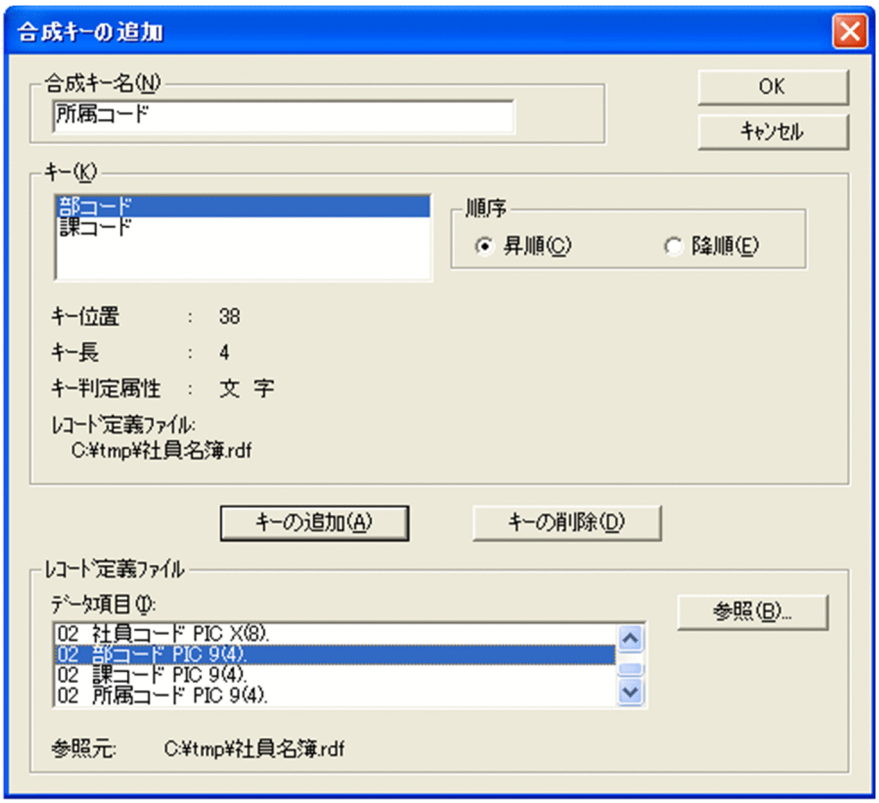
［参照］ボタンを選ぶと表示されるダイアログボックスによって，データ項目一覧に表示する内容を変更できます。

参照元

参照しているレコード定義ファイルの名称が表示されます。

7.6.2　合成キーの登録

ISAM 情報の設定ダイアログボックスの［合成キーの追加］ボタンを選ぶと，次に示す合成キーの追加ダイアログボックスが表示されます。



このダイアログボックスで定義する情報について説明します。

合成キー名

新規に登録する合成キーの名前を入力します。

キー

登録されているキーの一覧が表示されます。目的のキーを選び，キー情報の変更および削除などの操作ができます。

キー位置

キー一覧で選択したキーの開始位置が表示されます。

キー長

キー一覧で選択したキーの長さが表示されます。

キー判定属性

キー一覧で選択したキーの判定属性が表示されます。

レコード定義ファイル

キー一覧で選択したキーが定義されているレコード定義ファイルの名称が表示されます。

順序

キー一覧で選択したキーのキー順序を指定します。昇順を指定してください。降順を指定した場合，OPEN 文でエラーとなります。

[キーの追加] ボタン

データ項目一覧に表示されているデータ項目の中から目的のデータ項目を選び，このボタンを選ぶと，選択したデータ項目をキーとして登録できます。

注意事項

次の項目はキーとして登録できません。

- 8 バイトの 2 進項目
- 内部ブール
- 外部ブール
- 英数字編集
- 日本語編集
- アドレス
- インデックス

[キーの削除] ボタン

キー一覧に表示されているキーの中から目的のキーを選び，このボタンを選ぶと，選択したキーをキー一覧から削除できます。

データ項目

参照しているレコード定義ファイルに定義されているデータ項目の一覧が表示されます。

［参照］ボタンを選ぶと表示されるダイアログボックスによって，データ項目一覧に表示する内容を変更できます。

参照元

参照しているレコード定義ファイルの名称が表示されます。

7.6.3　合成キー情報の設定

ISAM 情報の設定ダイアログボックスの［合成キーの変更］ボタンを選ぶと，合成キーの変更ダイアログボックスが表示されます。

このダイアログボックスで定義する情報は，合成キーの追加ダイアログボックスと共通です。詳細は，「7.6.2　合成キーの登録」を参照してください。

7.6.4　ISAM ファイルの生成

定義した ISAM 情報から ISAM ファイルを生成する操作について説明します。

1. **［ファイル］メニューから［ISAM ファイル生成］を選ぶ。**

   ファイル名を指定するダイアログボックスが表示されます。

   この直前に ISAM ファイルが作成されている場合は，その ISAM ファイルのあるフォルダがリストボックスに表示されます。

   また，ISAM ファイルを新規に作成するときには，［オプション］ボタンで，障害発生時にデータファイルの内容を保証するかどうか，および削除レコードの再利用を抑止するかどうかを指定できます。

2. **ファイル名テキストボックスにファイル名を入力する。**

3. **［保存］ボタンを選ぶ。**

   ISAM ファイルが生成されます。

7.7 COBOL ソースファイルの生成

定義したファイル／レコード仕様から，FILE SECTION，WORKING-STORAGE SECTION または LOCAL-STORAGE SECTION 用の COBOL ソースファイルを生成する方法を説明します。

ファイル／レコード定義は，自由形式の COBOL ソースを次の形式で生成します。

- 1 行の最大長は 80 バイトです。
- コメント行は，行中の最初の文字位置に「*」が生成されます。
- 定数の連結演算子には「&」が生成されます。

1. **［ファイル］メニューから［COBOL 生成］を選ぶ。**
   生成先のファイル名を指定するダイアログボックスが表示されます。

2. **ファイル名テキストボックスに生成先のファイル名を入力する。**
   ファイルの種類が固定形式（*.cbl，*.cob，*.ocb）の場合は，［オプション］ボタンで一連番号を付けるように指定できます。一連番号の指定方法については，「7.9.1　環境設定」を参照してください。

3. **［保存］ボタンを選ぶ。**
   COBOL ソースファイルが生成されます。
   拡張子が指定されていない場合は，（.cbl）または（.cbf）が仮定されます。

注意事項

- エラーは，1 データ項目内の属性間の範囲だけでチェックされます。このため，生成した COBOL 原始プログラムには，ほかのデータ項目間のエラー（再定義名に指定したデータ項目がないなど）が残ることがあります。
- ［生成ソースファイルフォーマット］の設定がされていない場合，ファイルの種類には「COBOL ソース（*.cbf;*.ocf)」が表示されます。
- 生成した COBOL ソースがファイル節で参照される場合，制限値エラーとなることがあります。レコード定義では「作業場所節の制限値」を採用するため，制限値が異なるほかの節ではエラーとなります。

7.8　ファイル／レコード定義の印刷

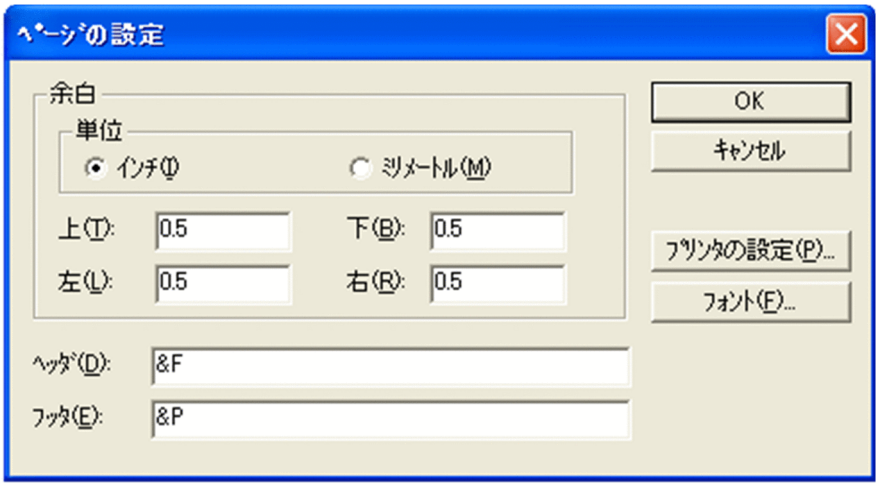
定義したファイル／レコード仕様の内容を印刷する方法について説明します。

7.8.1　印刷ページの設定

印刷するページの余白，およびページのヘッダとフッタを設定する方法を説明します。

1. [ファイル] メニューから［ページの設定］を選ぶ。

   次に示すページの設定ダイアログボックスが表示されます。必要事項を指定します。



2. [OK] ボタンを選ぶ。

   次に，このダイアログボックスで設定する内容について説明します。

余白

［上］［下］［左］［右］ボックスに余白の値（単位はインチ，またはミリメートル）を入力します。デフォルトは，上下左右すべて，0.5 インチです。

余白には，プリンタの最小余白幅よりも大きな値を指定してください。

ヘッダとフッタ

表題のほかに次に示すマクロ文字を使ってファイル名やページ番号などの情報を印刷できます。ヘッダに指定した文字列はページの上端から 0.25 インチの位置に，フッタに指定した文字列はページの下端から 0.25 インチの位置に，それぞれ印刷されます。

また，揃え操作のマクロ文字を利用して，ヘッダやフッタに印刷する文字列のレイアウトを設定できます。揃え操作のマクロ文字を何も指定しなかった場合，ヘッダやフッタは中央揃えで印刷されます。

次に，ヘッダとフッタで使用できるマクロ文字の一覧を示します。

印刷項目	マクロ文字
ファイル名	&F
ページ番号	&P
現在時間	&T
日付	&D

印刷項目	マクロ文字
左揃え	&L
中央揃え	&C
右揃え	&R

例えば，ファイル名が「SAMPLE.FLF」の場合，ヘッダに「&L&F&C 定義&R ページ&P」と指定すれば，次のように印刷されます。

```
SAMPLE.FLF                    定義                    ページ1
```

なお，マクロ文字の指定によっては，ヘッダやフッタ領域の文字が重なって印刷されることもあります。

フォント

［フォント］ボタンを選ぶと表示されるフォントの指定ダイアログボックスで，印刷に使用するフォント，スタイル，サイズなどを設定します。固定ピッチのフォントだけが選択できます。

プリンタの設定

［プリンタの設定］ボタンを選ぶと表示されるプリンタの設定ダイアログボックスで，出力先のプリンタ，用紙サイズ，用紙の向きなどを設定します。

注意事項

ページの設定ダイアログボックスで設定した内容は，表示中の印刷プレビューには反映されません。

7.8.2　印刷プレビュー

印刷のプレビューを表示して印刷イメージを確認したあと，印刷します。

1. **［ファイル］メニューから［印刷プレビュー］を選ぶ。**

   印刷イメージが表示されます。

2. **印刷イメージを確認し，［印刷］ボタンを選ぶ。**

   表示されたダイアログボックスに必要事項を入力して印刷します。詳細は，「7.8.3　印刷」を参照してください。

注意事項

属性設定ウィンドウにフォーカスがあるときは，印刷のプレビューができません。

7.8.3　印刷

現在作業中のファイル（.flf または.rdf）を印刷する方法を説明します。

1. [ファイル] メニューから [印刷] を選ぶ。

印刷ダイアログボックスが表示されます。必要事項を指定します。

2. [OK] ボタンを選ぶ。

次に，このダイアログボックスで設定する内容について説明します。

ファイル情報の印刷

このチェックボックスをオンにすると，作業中のファイルのファイル情報が印刷されます。デフォルトはオンです。

COBOL ソース情報の印刷

このチェックボックスをオンにすると，作業中のファイルの COBOL ソース情報が印刷されます。デフォルトはオンです。

PICTURE 句情報の印刷

このチェックボックスをオンにすると，作業中のファイルの PICTURE 句情報が印刷されます。デフォルトはオンです。

ファイル仕様定義の印刷規則

- ファイル仕様定義を印刷するには，半角 65 文字以上の印字幅が必要です。
- ファイル仕様定義は，左寄せの行形式で印刷されます。ただし，COPY 原文名は，レコード定義ファイル名の項目に続いて 36 文字目から印刷されます。
- 印刷される項目の長さが用紙の印刷幅を超える場合，用紙の幅に収まらなかった部分は印刷されません。ただし，次の項目については，行が折り返されて印刷されます。

  見出し情報の概要

  文字列の長さが半角 30 文字を超える場合，印刷幅に応じて調節されて折り返されます。

  レコード情報のレコード定義ファイル名

  半角 30 文字で折り返されます。
- ファイル情報，COBOL ソース情報，および PICTURE 句情報は，改ページされて印刷されます。

レコード仕様定義の印刷規則

- レコード仕様定義を印刷するには，半角 137 文字以上の印字幅が必要です。
- レコード仕様定義は，表形式で印刷されます。次に，印刷される表のフォーマットを示します。

レコード仕様定義の項目	カラム	印字幅	備考
項番	2 文字	5 文字	
レベル	8 文字	4 文字	
データ名	13 文字	30 文字	
USAGE/PICTURE	44 文字	4 文字	項目のタイトルは「タイプ」とする
サイズ	49 文字	10 文字	
反復	60 文字	10 文字	

レコード仕様定義の項目	カラム	印字幅	備考
SYNC	71 文字	4 文字	
その他	76 文字	32 文字	印字幅で折り返される
注釈	109 文字	28 文字	印字幅で折り返される

注

カラムは，左端から数えた文字数を表します。

- 印刷される項目の長さが用紙の印刷幅を超える場合，用紙の幅に収まらなかった部分は印刷されません。ただし，その他項目と注釈項目は印字幅で折り返されて印刷されます。
- ファイル情報，COBOL ソース情報，および PICTURE 句情報は，改ページされて印刷されます。

印刷例

ファイル／レコード仕様定義の印刷例を，次に示します。

- ファイル仕様定義の印刷例

図 7-5　ファイル仕様定義の印刷例

```
                                        aaa. flf

[ファイル定義]
ファイル名                  :   MST-FILE
ファイル編成                :   索引順編成
レコード形式                :   ブロック化可変長（VB）
レコード長                  :   10
最小レコード長              :   4
レコード長設定エリア名      :   ABCDE-NAME
ブロック長                  :   100
EXTERNAL                    :   指定あり
GLOBAL                      :   指定なし

[見出し情報]
業務名称                    :   このファイルは，・・・・・・・・・・である。
ファイル区分                :   マスタ
概要                        :   このファイルは，・・・・・・・・・・・・・・・・・・・・
                                ・・・（折り返し）・・である。

[レコード情報]
   レコードファイル名                                          COPY原文名
 1  C:￥abc.rdf                                               COPY1
 2  def.rdf                                                   COPY2
 3  ggggggggg.rdf                                             COPY3
 4  hihihihih.rdf                                             COPY4
 5  japan.rdf                                                 COPY5
 6  def.rdf                                                   COPY6
 7  kkk.rdf                                                   COPY7
 8  HITACHI. rdf                                              COPY8
 9  C:¥aaaaa¥bbbb¥(折り返し)¥ffff¥gggg¥hhh                    COPY9
    hhhh¥abc.rdf
10  long.rdf                                                  COPY10
```

```
                                        aaa.flf

[ファイル情報]
種別                        :   ファイル仕様
作成日付                    :   2003/01/01          09:00
更新日付                    :   2003/01/01          09:00

[COBOLソース情報]
生成の許可                  :   許可あり
ソースファイル名            :   C:¥USER¥SAMPLE.cbl
生成日付                    :   2003/01/01          09:00

[PICTURE句情報]
CURRENCY  SIGN              :   ￥
DECIMAL-POINT               :   PERIOD
```

- レコード仕様定義の印刷例

図 7-6　レコード仕様定義の印刷例

aaa.rdf

項番	レベル	データ名	タイプ	サイズ	反復	SYNC	その他	注釈
01	01	MAST-REC	G				・・・・・・・・・(折り返し)・	・・・・・・・・・(折り返し)・
02	03	A-KEY	I					
03	03	B-GRP						
04	03	APPLE						
05	03	BOY						
06	03	CAT						
07	03	DIAMOND						
08	03	ELEPHANT						
09	03	FUU・（折り返し）・・U						
10	03	A-KEY						

凡例：
G：集団　　I：インデクス

```
                                aaa.rdf

[ファイル情報]
種別                    :   レコード仕様
作成日付                :   2003/01/01        09:00
更新日付                :   2003/01/01        09:00

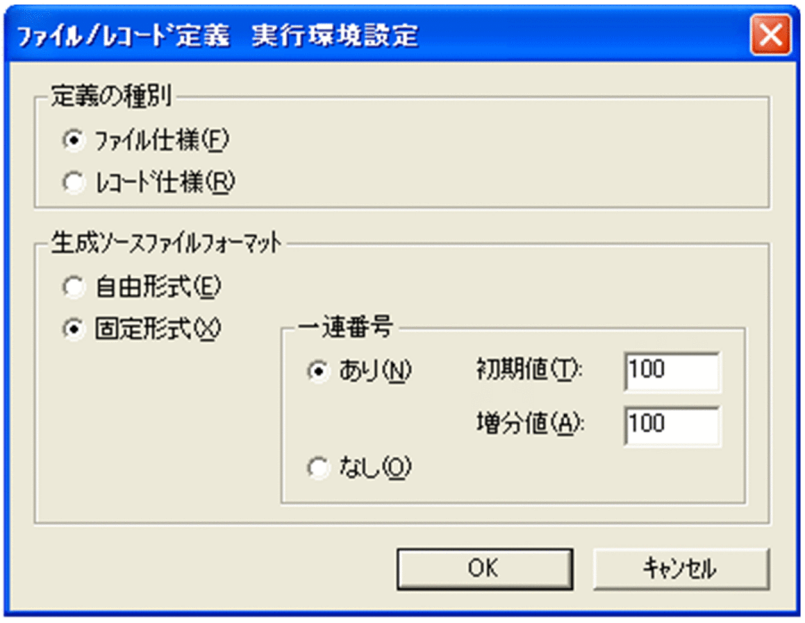
[COBOLソース情報]
生成の許可              :   許可あり
ソースファイル名        :   C:¥USER¥SAMPLE.cbl
生成日付                :   2003/01/01        09:00

[PICTURE句情報]
CURRENCY  SIGN          :   ￥
DECIMAL-POINT           :   PERIOD
```

7.9　ファイル／レコード定義時のオプション機能

7.9.1　環境設定

1. [オプション] メニューから［ファイル／レコード定義環境設定］を選ぶ。
   次に示すダイアログボックスで環境を設定します。



7.9.2　COBOL ソース生成の許可の設定

COBOL ソースファイルの生成を許可するかどうかを設定できます。COBOL ソース生成の許可の設定を切り替える操作を説明します。

1. [オプション] メニューから［COBOL 生成許可］を選ぶ。
   COBOL ソースの生成が許可されているときは，［COBOL 生成許可］の前に" ✓ "が表示され，［ファイル］メニューの［COBOL 生成］を選べます。

7.9.3　ファイル情報の表示

1. [ファイル] メニューから［ファイル情報］を選ぶ。
   現在作業中の定義ファイル（.flf または.rdf）に関する情報が表示されます。

7.9.4 CSV ファイルへの出力

現在作業中の定義ファイル（.flf または.rdf）の内容を，CSV 形式のファイルに出力できます。

1. [ファイル] メニューから［CSV 形式で出力］を選ぶ。
   CSV 形式で出力ダイアログボックスが表示されます。

2. [ファイル名] に，出力する CSV ファイルの名称を指定する。

3. [データの引用符] に，引用符の有無を指定する。
   「"」を指定した場合，データがダブルコーテーションで囲まれて出力されます。
   「なし」を指定した場合は，ダブルコーテーションで囲まれません。デフォルトは，「"」（ダブルコーテーションで囲んで出力）です。

4. [保存] ボタンを選ぶ。
   作業中のファイルの内容が，CSV 形式でファイルに出力されます。

出力される CSV ファイルのフォーマット

出力される CSV ファイルのフォーマットを，次に示します。

""で囲まれた部分は固定文字列で，ユーザが定義した値は（）で囲まれた部分に格納されます。

(a) ファイル仕様定義を出力したときのフォーマット

```
(定義ファイルの名称)
”種別”,”ファイル仕様”
”作成日付”,(作成の日時)
”更新日付”,(更新の日時)
”COBOLソース生成”,(COBOLソース生成の許可)
”COBOLソースファイル名”,(COBOLソースファイルの名称)
”COBOLソース生成日付”,(COBOLソース生成の日時)
”CURRENCY SIGN”,(通貨文字)
”DECIMAL-POINT”,(種別)※

”ファイル名”,(ファイル名)
”ファイル編成”,(ファイル編成)
”レコード形式”,(レコード形式)
”レコード長”,(レコード長)
”最小レコード長”,(最小レコード長)
”レコード長設定エリア名”,(レコード長設定エリア名)
”ブロック長”,(ブロック長)
”EXTERNAL”,(EXTERNAL指定)
”GLOBAL”,(GLOBAL指定)

”業務名称”,(業務名称)
”ファイル区分”,(ファイル区分)
”概要”,(概要)

”レコード定義ファイル名”,”COPY原文名”
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
```

```
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
(レコード定義ファイル名),(COPY原文名)
```

注※

（種別）は，「PERIOD」「COMMA」のどちらかになります。

(b) レコード仕様定義を出力したときのフォーマット

```
(定義ファイルの名称)
”種別”,”レコード仕様”
”作成日付”,(作成の日時)
”更新日付”,(更新の日時)
”COBOLソース生成”,(ソース生成の許可)
”COBOLソースファイル名”,(COBOLソースファイルの名称)
”COBOLソース生成日付”,(COBOLソース生成の日時)
”CURRENCY SIGN”,(通貨文字)
”DECIMAL-POINT”,(種別)※

”項番”, ”レベル”, ”データ名”, ”タイプ”, ”サイズ”, ”反復”, ”編集用PICTURE”, ”初期値”, ”指
標名”, ”SIGN”, ”再定義名”, ”アドレス名”, ”CHAR TYPE”, ”JUST”, ”EXTERNAL”, ”GLOBAL”, ”SYNC
”, ”BLANK WHEN ZERO”, ”注釈”,
(項番), (レベル), (データ名), (タイプ), (サイズ), (反復), (編集用PICTURE), (初期値), (指
標名), (SIGN), (再定義名), (アドレス名), (CHAR TYPE), (JUST), (EXTERNAL), (GLOBAL), (SYNC
), (BLANK WHEN ZERO), (注釈)
(項番), (レベル), (データ名), (タイプ), (サイズ), (反復), (編集用PICTURE), (初期値), (指
標名), (SIGN), (再定義名), (アドレス名), (CHAR TYPE), (JUST), (EXTERNAL), (GLOBAL), (SYNC
), (BLANK WHEN ZERO), (注釈)
 :
定義されたレコード数分出力する
 :
```

注

データ項目のタイプは，USAGE/PICTURE で設定します。

注※

（種別）は，「PERIOD」「COMMA」のどちらかになります。

8 テストデバッガ，カバレージ，TD コマンド生成機能，および Unicode 機能の概要

テストデバッガ，カバレージ，TD コマンド生成機能，および Unicode 機能の概要について説明します。

8.1　テストデバッガとは

テストデバッガは，TD コマンドなどでプログラムの実行を制御しながらプログラムをテストしたり，デバッグしたりする機能です。テストデバッガで，プログラムの実行を実行文単位に中断し，データ項目に対して値を表示，代入，比較してデバッグできます。また，連動するプログラムやファイルなどの実行環境が整わない段階でも，それらの処理をシミュレーションできる機能もあるため，単体でプログラムをテストすることもできます。テストデバッガを使用するときは，-TDInf コンパイラオプションを指定します。

テストデバッガの主な機能について説明します。

8.1.1　デバッグ機能

(1)　実行プログラムの中断

プログラムの実行を実行文単位に中断し，データの代入などができます。機能の概要については「9.2.1 プログラムの中断」を参照してください。

- 中断点の設定による中断
- データ監視条件の設定による中断
- 割り込みによる中断
- 実行時エラーが発生した場合の中断

(2)　実行の制御

設定した中断点などに基づき実行できます。機能の概要については「9.2.2　プログラムの実行」を参照してください。

- 連続実行
- ステップイン
- ステップオーバー
- ジャンプ
- ジャンプ実行
- 実行の強制終了
- 実行の開始と開始時にできる指定

(3)　データの操作

COBOL プログラムのデータに対して，値の表示，値の代入，比較ができます。

Unicode 機能を使用する場合のデータの操作については，「8.4.2　機能の詳細」を参照してください。
Unicode 機能を使用しない場合のデータの操作については「9.2.5　データの操作」を参照してください。

- データ値の表示
- データ値の代入
- データ値の比較

8.1.2　テスト機能

(1)　単体テスト

連動するプログラムやファイルなどの実行環境が整わない段階でも，それらの処理をシミュレーションして，プログラムを単体でテストできます。呼び出し元のプログラム，呼び出し先のプログラム，ファイル，または DC（データコミュニケーション）を，TD コマンドによってシミュレーションできます。

機能の概要については「9.2.7　プログラムの単体テスト」を参照してください。

- 主プログラムシミュレーション
- 副プログラムシミュレーション
- ファイル入出力文のシミュレーション
- DC シミュレーション

8.1.3　GUI モードとバッチモード

テストデバッガには，GUI モードとバッチモードの二つの方法があります。

- GUI モード

  ウィンドウに表示されたプログラムの動きを確認しながら，マウスやキーボードの操作によって，対話的にテストしたり，デバッグしたりできます。TD コマンドも入力できます。

  GUI モードでのテストデバッグの手順については，「10.1.1　テストデバッグの手順」を参照してください。

- バッチモード

  あらかじめ，実行したい TD コマンドをファイルにまとめて記述しておき，そのファイルを起動することによって，一括してテストデバッグを実行する方法です。コマンドプロンプトに，TD コマンドを記述したファイルを指定した起動コマンドを入力すると，ファイルに記述された TD コマンドを一括して実行します。一度実行を開始すれば，利用者の操作を必要としないので，効率良く大量のプログラムがテストできます。

  バッチモードでのテストデバッグの手順については，「11.2.1　テストの手順」を参照してください。

8.1.4　テストデバッガの機能一覧

テストデバッガの機能一覧と GUI モード，バッチモードでの機能の有無を次に示します。

表 8-1　テストデバッガの機能一覧と GUI モード，バッチモードでの機能の有無

機能の概要	機能の詳細		GUI モード	バッチモード
プログラムの中断	中断点の設定による中断	中断点の設定／解除	○	○
		中断点の一覧表示	○	○
	データ監視条件の設定による中断	データ監視条件の設定／解除	○	○
		データ監視条件の一覧表示	○	×
	割り込みによる中断		○	×
	実行時エラーが発生した場合の中断		○	×
プログラムの実行	連続実行		○	○
	ステップイン		○	○
	ステップオーバー		○	○
	ジャンプ		○	○
	ジャンプ実行		○	○
	実行の強制終了		○	○
プログラムの実行の追跡	トレース表示		○※	○
	フロー情報の蓄積と表示		○※	○
	アニメーション表示		○	×
	バックトレース表示		○	×
	現在位置の表示		○	○
データの操作	データ値の表示	データ値の表示	○	○
		バリアント値の表示	○	○
	オブジェクトデータ値の表示	オブジェクト参照データ値の表示	○	○
	OLE プロパティの表示		○	○
	データ値の代入		○	○
	データ値の比較		○※	○
領域の操作	領域の確保／解放		○※	○
プログラムの単体テスト	主プログラムシミュレーション		○※	○

機能の概要	機能の詳細		GUI モード	バッチモード
	副プログラムシミュレーション		○※	○
	ファイルシミュレーション		○※	○
	ファイル入出力条件発生のシミュレーション	ファイル終了条件	○※	○
		ページ終了条件	○※	○
		無効キー	○※	○
		入出力誤り	○※	○
	DC シミュレーション		○※	○
バッチによるテスト	テストケースの指定		×	○
テストデバッグのための準備	TD コマンド格納ファイルの TD コマンドの実行		○	○
	ユーザプログラムの変更		○	×
	テストデバッグ機能の初期化		○	×
	等価規則の変更		○	○
	結果のファイル出力	結果出力先の指定	○※	○
		テストデバッガの表示内容のファイル出力	○	×
ヘルプ	バージョン情報		○	×
その他	ソースファイルの編集		○	×
	テスト結果への注釈の表示		○※	○

(凡例)

○：機能を使用できる

×：機能を使用できない

注※

GUI モードでは，TD コマンド入力ウィンドウに TD コマンドを入力することで使用できます。

8.2　カバレージとは

カバレージを使用して，テスト終了後，C0 メジャー，C1 メジャー，未実行文情報などのカバレージ情報を集計して表示できます。カバレージ情報を確認することでテストの進捗状況や性能を数値で把握できます。カバレージには，テストの進捗状況を定量的に表すカバレージ情報，文の実行回数をカウントするカウント情報の表示機能があります。

カバレージを使用するときは，プログラムのコンパイル時に-CVInf コンパイラオプションを指定します。テストデバッガ上でカバレージを使用するときは，-TDInf コンパイラオプションも指定します。

カバレージの手順については，「14.　GUI モードによるカバレージ」および「15.　バッチモードによるカバレージ」を参照してください。

8.2.1　カバレージ情報

カバレージ情報とは，テストによって実行された文の割合を表示したもので，テストの進捗状況を確認するのに使用します。

カバレージ情報を表示できるリストには，カバレージ情報一覧，まとめ表示，全ソース表示などがあります。また，カバレージ情報を 0%化したり，カバレージ情報ファイルをマージしたりする操作もできます。

8.2.2　カウント情報

カウント情報とは，プログラムごとに文の実行回数を表示したもので，プログラムの実行時の性能を評価するのに使用します。

8.2.3　GUI モードとバッチモード

カバレージには，GUI モード，バッチモードの二つの方法があります。

- GUI モード

  ウィンドウのメニューからカバレージの機能を操作できます。

  GUI モードでのカバレージの手順については，「14.1.1　カバレージの手順」を参照してください。

- バッチモード

  コマンドプロンプトから入力するコマンドの指示によって，カバレージの機能を操作できます。プログラムからカバレージを連動実行させ，一括してカバレージの蓄積およびカウント情報の表示をすることもできます。一度実行を開始すれば，利用者の操作を必要としないので，効率良く大量のプログラムのカバレージができます。

バッチモードでのカバレージの手順については，「15.　バッチモードによるカバレージ」を参照してください。

8.2.4　カバレージの機能一覧

カバレージの機能一覧と GUI モード，バッチモードでの機能の有無を次に示します。

表 8-2　カバレージの機能一覧と GUI モード，バッチモードでの機能の有無

機能の概要	機能の詳細		GUI モード	バッチモード
カバレージ情報の蓄積	カバレージ情報の蓄積		○	○
カバレージ情報の表示	テキスト形式のファイルの場合	C0,C1,S1 メジャーの表示	○	○
		未実行の文の表示	○	○
		差分カバレージ情報の表示	○	○
	CSV 形式のファイルの場合	C0,C1,S1 メジャーの表示	×	○
		未実行の文の表示	×	○
		差分カバレージ情報の表示	×	○
カバレージ情報の操作	カバレージ情報の 0%化		○	○
	差分カバレージ情報の 0%化		○	○
	差分カバレージ情報のクリア		○	○
	カバレージ情報のマージ		○	○
カウント情報の表示	実行回数の表示		○	○

（凡例）

○：機能を使用できる

×：機能を使用できない

8.3　TD コマンド生成機能とは

TD コマンド生成機能とは，テスト対象の COBOL ソースファイルから，テストデバッガで使用する TD コマンド群を生成する機能です。TD コマンド群とは，テストデバッガの実行を制御するための TD コマンドの集まりです。生成されたファイルを利用し，必要に応じて修正することで，TD コマンド群を効率良く作成できます。

生成される TD コマンド群には，中断点情報，プログラムのシミュレーション情報，およびファイルのシミュレーション情報の 3 種類があります。TD コマンド生成機能の詳細は，「16.　TD コマンド生成機能」を参照してください。

8.3.1　TD コマンド生成機能を使用するときに指定するコンパイラオプション

TD コマンド群は，-TestCmd コンパイラオプションを指定してコンパイルしたときに生成されます。

8.3.2　TD コマンド生成機能の対象となるプログラム

TD コマンド生成機能の対象となるプログラムは，コンパイルしたときすべてのプログラムで，S レベル，U レベルのコンパイルエラーのなかった原始プログラムファイルだけです。

8.4 Unicode 機能

8.4.1 機能の概略

Unicode をデータ値として扱う COBOL プログラムをデバッグするための機能です。

テストデバッガは，データを表示するときに，Unicode からシフト JIS への変換を行い，代入や比較をするときは，シフト JIS から Unicode への変換を行います。そのため，ユーザはデータの操作をシフト JIS でできます。

この機能は，カバレージ情報の蓄積およびカウント情報の表示は対象にしません。ただし，カバレージ情報の蓄積またはカウント情報の表示を実行した場合は，この機能を使用しないで実行した場合と同じ動作をします。

Unicode 機能のその他の機能については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

(1) 使用するコード変換ライブラリ

Unicode 機能を利用するには，コード変換ライブラリをインストールしておく必要があります。次のプログラムプロダクト（以降これらのプログラムプロダクトをコード変換ライブラリと表記します）のどちらかをインストールしてください。

Windows(x86) COBOL2002 の場合

- Hitachi Code Converter - Server Runtime for C/COBOL
- Hitachi Code Converter - Client Runtime for C/COBOL

Windows(x64) COBOL2002 の場合

- Hitachi Code Converter - Client Runtime for C/COBOL(64)
- Hitachi Code Converter - Server Runtime for C/COBOL(64)

また，COBOL プログラムから直接コード変換ライブラリを呼び出す場合は，上記のプログラムプロダクトに加えて次のプログラムプロダクトのどれかが必要です。

Windows(x86) COBOL2002 の場合

- Hitachi Code Converter - Development Kit for C/COBOL
- Hitachi Code Converter - Development Kit for C/COBOL(64)

Windows(x64) COBOL2002 の場合

- Hitachi Code Converter - Development Kit for C/COBOL(64)

コード変換ライブラリが次のどれかの場合で，テストデバッガを起動したときはエラーメッセージを出力してテストデバッガは終了します。

- コード変換ライブラリをインストールしていない。
- 正しくインストールされていない。
- 前提バージョンより前のバージョンがインストールされている。

「KCCC0202T-U　コード変換ライブラリが正しくインストールされていません。」

外字を表示するための外字マッピングファイルは，コード変換ライブラリのインストールディレクトリにあるファイルを使用します。外字マッピングファイル名は，インストール時のファイル名にします。

(2)　コンパイラオプションと環境変数について

Unicode 機能を利用してデバッグをするときに必要なコンパイラオプションと環境変数を次に示します。

なお，コンパイラオプションおよび環境変数の詳細は，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

Unicode 機能に必要なコンパイラオプションを表 8-3 に，環境変数を表 8-4 に示します。

表 8–3　Unicode 機能に必要なコンパイラオプション

コンパイラオプション	説明
-UniObjGen	デバッグ対象プログラムは，必ずこのコンパイラオプションを指定してコンパイルしなければならない。
-UniEndian	UTF-16 のバイトオーダを指定したいときだけ指定する。

表 8–4　Unicode 機能に必要な環境変数

環境変数	説明
CBLLANG	必ず設定しなければならない。 設定は，テストデバッガを起動する前に設定しなければならない。
CBLUNIENDIAN	UTF-16 のバイトオーダを指定したいときだけ設定する。 設定は，テストデバッガを起動する前に設定しなければならない。

Unicode 機能に必要なコンパイラオプションの指定と環境変数を設定した場合，テストデバッガは，次の規則に従い動作します。

1. すべてのデバッグ対象プログラムに-UniObjGen コンパイラオプションを指定してコンパイルする必要があります。また，環境変数 CBLLANG に UNICODE を指定した環境でデバッグをしてください。それ以外の環境での動作は保証しません。
2. -UniEndian コンパイラオプションで指定したバイトオーダと環境変数 CBLUNIENDIAN で指定したバイトオーダを一致させてください。一致しない場合の動作は保証しません。
3. 環境変数の値は，テストデバッガの起動前に設定してください。テストデバッガを起動中に環境変数の値を設定または変更しても，有効になりません。

4. 環境変数の設定がテストデバッガ（GUI モード）に有効になる設定個所を表 8-5 に示します。連動実行時の環境変数の設定の場合も，表 8-5 のとおりです。

表 8–5　環境変数 CBLLANG，CBLUNIENDIAN の設定個所

テストデバッガ(GUI モード)起動方法	環境変数設定個所		
	システム	開発マネージャ[プロジェクト]-[プロジェクトの設定]-[ユーザ設定]タブ	実行支援
開発マネージャから起動	○	○	×
[COBOL2002] 下のメニューから起動	○	×	×

（凡例）
○：有効　　×：無効

(3)　デバッグ手順

Unicode 機能を利用したデバッグの手順を次に示します。

1. デバッグ対象の COBOL プログラムをコンパイルするときに，-UniObjGen コンパイラオプションを指定します。
2. 環境変数 CBLLANG に値として UNICODE を設定します。
3. テストデバッガを起動してデバッグをします。

用途が NATIONAL の項目（UTF-16）に対してバイトオーダを指定する場合は，上記手順に合わせて，次の手順になります。

1. デバッグ対象の COBOL プログラムをコンパイルするときに，-UniEndian コンパイラオプションで UTF-16 のバイトオーダを指定します。
2. 環境変数 CBLUNIENDIAN に値として BIG または LITTLE を設定して，UTF-16 のバイトオーダを指定します。
3. テストデバッガを起動して，デバッグをします。

バイトオーダの指定については，「8.4.3　その他の注意事項」の「(1)　バイトオーダ」を参照してください。

8.4.2　機能の詳細

(1)　データ値の表示

データ値をデータ属性表示するときは，テストデバッガが文字コードを Unicode（UTF-8 または UTF-16）からシフト JIS に変換して，変換後の値を表示します。

また，データ値を 16 進数で表示するときは，文字コードの変換は行わず，Unicode（UTF-8 または UTF-16）の 16 進数で表示します。UTF-16 の 16 進数での表示については，「8.4.3　その他の注意事項」の「(1)　バイトオーダ」を参照してください。

データ値の表示は，［データの操作］メニューから［データ値の表示］または［常時データ表示に設定］を選ぶか，または DISPLAY DATA コマンド，SET TRACE コマンド，DISPLAY OBJECT コマンド，DISPLAY FACTORY コマンドで，データの値をデータ属性または 16 進数で表示できます。

GUI モードでのデータの表示については，「10.13.1　データ値の表示」を参照してください。

DISPLAY DATA コマンド，SET TRACE コマンド，DISPLAY OBJECT コマンド，DISPLAY FACTORY コマンドの詳細については，「12.4.8　DISPLAY DATA（データの値表示）」，「12.4.10　DISPLAY OBJECT／DISPLAY FACTORY（オブジェクトのデータ値の表示）」，「12.4.24　SET TRACE／RESET TRACE（トレース表示の開始と中止）」を参照してください。

データ値を表示するときの規則を次に示します。

1. データ値をデータ属性表示するときに，用途が DISPLAY の項目および用途が NATIONAL の項目を変換して表示します。
   - 用途が DISPLAY の項目の場合は，UTF-8 からシフト JIS に変換して表示します。
   - 用途が NATIONAL の項目の場合は，UTF-16 からシフト JIS に変換して表示します。

   各データ項目について変換するかどうかを表 8-6 に示します。

表 8-6　シフト JIS へ変換するデータ項目

表示するデータ項目の種類	Unicode からシフト JIS への変換
英数字集団項目	○
英字項目	−
英数字項目	○
英数字編集項目	○
数字編集項目	_※1
整数項目※2	−
非整数項目※2	−
COMP－X 項目	−
外部浮動小数点項目	−
内部浮動小数点項目	−
指標名	−
指標データ項目	−
外部ブール項目	−

表示するデータ項目の種類	Unicode からシフト JIS への変換
内部ブール項目	－
日本語集団項目	●
日本語項目	●
日本語編集項目	●
アドレス名	－
アドレスデータ項目	－
ポインタ項目	－
オブジェクト参照データ項目	－
OLE オブジェクト参照データ項目	－
バリアントデータ項目	－

（凡例）

○：UTF-8 からシフト JIS へ変換する

●：UTF-16 からシフト JIS へ変換する

－：変換の必要がないため変換しない

注※1

文字がある場合は，UTF-8 からシフト JIS に変換します。

注※2

整数，非整数項目はそれぞれ外部 10 進項目，内部 10 進項目，および 2 進項目を含みます。

2. 表示機能を使用してデータ値を 16 進数で表示する場合は，変換しないで，Unicode（UTF-8 または UTF-16）の 16 進数で表示します。データ属性表示で表示エラーが発生した場合も変換しないで，Unicode （UTF-8 または UTF-16）の 16 進数で表示します。
3. Unicode（UTF-8 または UTF-16）からシフト JIS へ変換できないコードがある場合は，そのコードを置き換えて表示します。
   - 用途が DISPLAY の項目の場合は，半角ピリオドに置き換えて表示します。
   - 用途が NATIONAL の項目の場合は，全角' ＝ '（げた記号）（シフト JIS:X'81AC'）に置き換えて表示します。
4. Unicode（UTF-8 または UTF-16）で多バイトとして扱われる文字の途中で終了している場合は，そのバイトを半角ピリオドに置き換えて表示します。
5. コード変換中にエラーが発生したときは，データ属性表示で表示エラーになった場合と同様に変換しないで，Unicode（UTF-8 または UTF-16）の 16 進数で表示します。

(2) データ値の代入

データ名に文字定数（英数字定数，日本語文字定数，ALL 定数）を代入する場合，テストデバッガが，文字定数の文字コードをシフト JIS から Unicode（UTF-8 または UTF-16）に変換して，データに代入するので，ユーザは文字定数をシフト JIS で指定できます。

また，16 進英数字定数または 16 進日本語文字定数を使用すれば，文字コードを直接指定した代入ができます。16 進日本語文字定数での代入については，「8.4.3　その他の注意事項」の「(1)　バイトオーダ」を参照してください。

データ値の代入は，［データの操作］メニューから［データ値の代入］を選ぶか，または ASSIGN DATA コマンドで，データに値を代入して，値を変更できます。

GUI モードでのデータ値の代入については，「10.13.2　データ値の代入」を参照してください。

ASSIGN DATA コマンドの詳細については，「12.4.4　ASSIGN DATA（データの値代入）」を参照してください。

代入操作の規則を次に示します。

1. 代入する値に文字定数を指定した場合は次のように変換して代入します。
   - 英数字定数の場合は，シフト JIS から UTF-8 に変換して代入します。
   - 日本語文字定数の場合は，シフト JIS から UTF-16 に変換して代入します。

   代入は変換後のサイズで行われるため，変換によってデータのサイズが受け取り側のデータ項目のサイズを超えた場合，超えた部分は代入されません。
2. 16 進英数字定数または 16 進日本語文字定数を指定した場合は，文字コードの変換はしないでデータに代入します。
3. 16 進英数字定数または 16 進日本語文字定数の場合，次のように指定します。
   - 代入先の用途が DISPLAY の場合は，UTF-8 のコード値を指定します。
   - 代入先の用途が NATIONAL の場合は，UTF-16 のコード値を指定します。
4. 集団項目を除く，用途が DISPLAY の項目と，用途が NATIONAL の項目との間では代入はできません。
5. 用途が NATIONAL の項目を含む集団項目と，用途が DISPLAY の項目間の代入はしないでください。また，用途が DISPLAY の項目を含む集団項目と，用途が NATIONAL の項目間の代入はしないでください。代入をした場合は，代入後の値を保証しません。
6. 作用対象の用途が DISPLAY と，用途が NATIONAL の代入規則（Unicode 機能）を表 8-7 に示します。なお，表 8-7 に示されていないデータ項目の組み合わせについては，「9.3.1　データの比較・代入規則」の「表 9–10　代入規則」を参照してください。
7. コード変換中に未定義コードを検出するなどのエラーが発生したときは，次のエラーメッセージを出力して代入処理を中止します。

   「KCCC4302T-E　値が不正なため処理できません。値（XXXXX）」

表 8-7　代入規則（Unicode 機能）

送り出し側＼受け取り側	英数字集団項目	英字項目	英数字項目	英数字編集項目	数字編集項目	外部10進項目	整数※6	非整数※6	外部浮動小数点※7	内部浮動小数点※8	指標名	日本語集団項目	日本語項目	日本語編集項目
英数字集団項目	※1	※4	※4	※4	※4	※4	○	○	※4	※4		※5	※5	※5
英字項目		○	○	○										
英数字項目	※2	○	○	○	○	○	○	○	○	○				
英数字編集項目	※2	○	○	○										
数字編集項目	※2		○	○	○	○	○	○	○	○				
HIGH-VALUE	○	○	○	○								○	○	○
LOW-VALUE	○	○	○	○								○	○	○
SPACE	○	○	○	○								○	○	○
ZERO	○		○	○	○	○	○	○	○	○	○	○	○	○
英数字定数	※2		○	○										
英数字定数（数字のみ）	※2	○	○	○	○	○	○	○						
16進英数字定数	※2	○	○	○	○	○	○	○						
整数定数※6	○	○	○	○	○	○	○	○	○	○	※9			
非整数定数※6	○		○	○	○	○	○	○	○	○				
外部10進項目	※2		○	○	○	○	○	○	○	○	○			
整数※6	○		○	○	○	○	○	○	○	○	○			
非整数※6	○		○	○	○	○	○	○	○	○				
16進数字定数	※2		○	○	○	○	○	○	○	○	○			
外部浮動小数点※7	※2		○		○	○	○	○	○	○				
内部浮動小数点※8	※2		○		○	○	○	○	○	○				
浮動小数点数字定数	※2		○		○	○	○	○	○	○				
指標名						○	○				○			
日本語集団項目	※3											○	○	○
日本語項目	※3											○	○	○
日本語編集項目	※3											○	○	○
日本語文字定数	※3											○	○	○
16進日本語文字定数	※3											○	○	○
拡張16進定数	○	○	○	○	○	○	○	○	○	○		○	○	○

（凡例）

○：代入できる　　空白：代入できない

注※1

送り出し側の集団項目に用途が NATIONAL の項目を含んでいて，受け取り側の集団項目に用途が DISPLAY の項目を含んでいる場合は，代入後の値は保証しません。また，送り出し側の集団項目に用途が DISPLAY の項目を含んでいて，受け取り側の集団項目に用途が NATIONAL の項目を含んでいた場合も，代入後の値は保証しません。

注※2

受け取り側の集団項目に用途が NATIONAL の項目を含んでいた場合は，代入後の値は保証しません。

注※3

受け取り側の集団項目に用途が DISPLAY の項目を含んでいた場合は，代入後の値は保証しません。

注※4

送り出し側の集団項目に用途が NATIONAL の項目を含んでいた場合は，代入後の値は保証しません。

注※5

送り出し側の集団項目に用途が DISPLAY の項目を含んでいた場合は，代入後の値は保証しません。

注※6

整数は，次のデータ項目で小数点けたを持たない指定です。非整数は，次のデータ項目で小数点けたを持つ指定です。

- 内部 10 進項目
- 2 進項目（COMP-X も含む）

注※7

外部浮動小数点形式の数字項目です。

注※8

内部浮動小数点形式の数字項目です。

注※9

正の整数（0 を含むが，-0 は含まない）だけ代入できます。

(3)　比較条件式の設定

比較条件式に文字定数（英数字定数，日本語文字定数，ALL 定数）を指定した場合，テストデバッガが，文字定数の文字コードをシフト JIS から Unicode（UTF-8 または UTF-16）に変換して比較するため，文字定数をシフト JIS で指定できます。

また，16 進英数字定数または 16 進日本語文字定数を使用できるため，文字コードを直接比較条件式に設定できます。16 進日本語文字定数での比較については，「8.4.3　その他の注意事項」の「(1)　バイトオーダ」を参照してください。

比較は，バイト単位で比較します。大小順もバイト単位の大小順になります。

比較条件式は，［データの操作］メニューから［データ監視条件の設定］を選ぶか，IF コマンド，または SET WATCH コマンドを使用して設定できます。

GUI モードでの比較条件式の設定については，「10.9.1　データ監視条件の設定」を参照してください。

IF コマンド，SET WATCH コマンドの詳細については，「12.4.15　IF（データの値比較）」，「12.4.25 SET WATCH／RESET WATCH（データ監視条件の設定と解除）」を参照してください。

比較の規則を次に示します。

1. 比較条件式に文字定数を指定した場合，文字定数の文字コードの比較は変換後の値で比較します。
   - 英数字定数の場合は，UTF-8 に変換して比較します。
   - 日本語文字定数の場合は，UTF-16 に変換して比較します。
2. 16 進英数字定数または 16 進日本語文字定数を指定した場合は，文字コードを変換しないで比較します。
3. 16 進英数字定数または 16 進日本語文字定数を指定する場合，次のように指定してください。
   - 用途が DISPLAY の項目と比較する場合は，UTF-8 を指定します。
   - 用途が NATIONAL の項目と比較する場合は，UTF-16 を指定します。
4. コード変換中にエラーが発生したときは，エラーメッセージを出力して比較処理を中止します。
   「KCCC4302T-E　値が不正なため処理できません。値（XXXXX）」

8.4.3　その他の注意事項

(1)　バイトオーダ

テストデバッガは，文字コードの変換だけではなく，UTF-16 のバイトオーダの変換をします。

UTF-16 のバイトオーダの変換の規則を次に示します。

1. シフト JIS から UTF-16，または UTF-16 からシフト JIS に変換するときの UTF-16 のバイトオーダは，環境変数 CBLUNIENDIAN の値に従います。環境変数 CBLUNIENDIAN の値と仮定するバイトオーダの関係を表 8-8 に示します。
2. データ名への代入，および比較条件式を設定するときの 16 進日本語文字定数は UTF-16BE を指定してください。16 進日本語文字定数に指定された UTF-16BE のコード値は，環境変数 CBLUNIENDIAN の指定に従い，テストデバッガが自動的にバイトオーダの変換をします。
3. 用途が NATIONAL の項目を 16 進数で表示する場合，およびデータ属性表示で表示エラーが発生した場合に表示する 16 進数の値は，UTF-16BE の 16 進数で表示します。
4. データ名に値を代入する場合，拡張 16 進定数を使用してコード値を直接指定したときは，バイトオーダの変換はしません。
5. 比較条件式を設定する場合，拡張 16 進定数を使用してコード値を直接指定したときは，バイトオーダの変換はしません。

表 8-8　環境変数 CBLUNIENDIAN の値と仮定するバイトオーダ

データ項目	環境変数 CBLUNIENDIAN の値		
	LITTLE	BIG	LITTLE，BIG 以外 (指定なしも含む)
用途が NATIONAL	UTF-16LE※1 を 仮定する	UTF-16BE※2 を 仮定する	UTF-16LE※1 を仮定する

注※1

UTF-16LE: UTF-16 をリトルエンディアンで表現します。

注※2

UTF-16BE: UTF-16 をビッグエンディアンで表現します。

(2)　テストデバッガとカバレージが入出力するファイル

TD コマンド格納ファイルはシフト JIS で作成してください。ただし，外字を指定する場合など文字のコード値を直接指定したい場合は，16 進英数字定数，16 進日本語文字定数を使用して，指定したいコード値を Unicode で指定します。

テストデバッガまたはカバレージが出力するエラーメッセージおよび次に示すファイルはシフト JIS で出力します。

- ログ出力ファイル

- 結果出力ファイル
- 結果蓄積ファイル
- 実行結果出力ファイル
- カバレージ情報リストファイル
- カウント情報リストファイル
- カバレージ統計情報 CSV ファイル
- ソースカバレージ情報 CSV ファイル

(3) 表意定数の扱い

空白文字，表意定数 SPACE の文字コードは次のようになります。

- 用途が DISPLAY の場合は，UTF-8 の半角空白文字（X'20'）になります。
- 用途が NATIONAL の場合は，環境変数 CBLUNIENDIAN の設定に従い，UTF-16LE の全角空白文字（X'0030'）または UTF-16BE の全角空白文字（X'3000'）になります。仮定するバイトオーダについては「表 8-8　環境変数 CBLUNIENDIAN の値と仮定するバイトオーダ」を参照してください。

9 テストデバッガの概要と規則

テストデバッガの前提条件，概要，設定値や使用例などについて説明します。

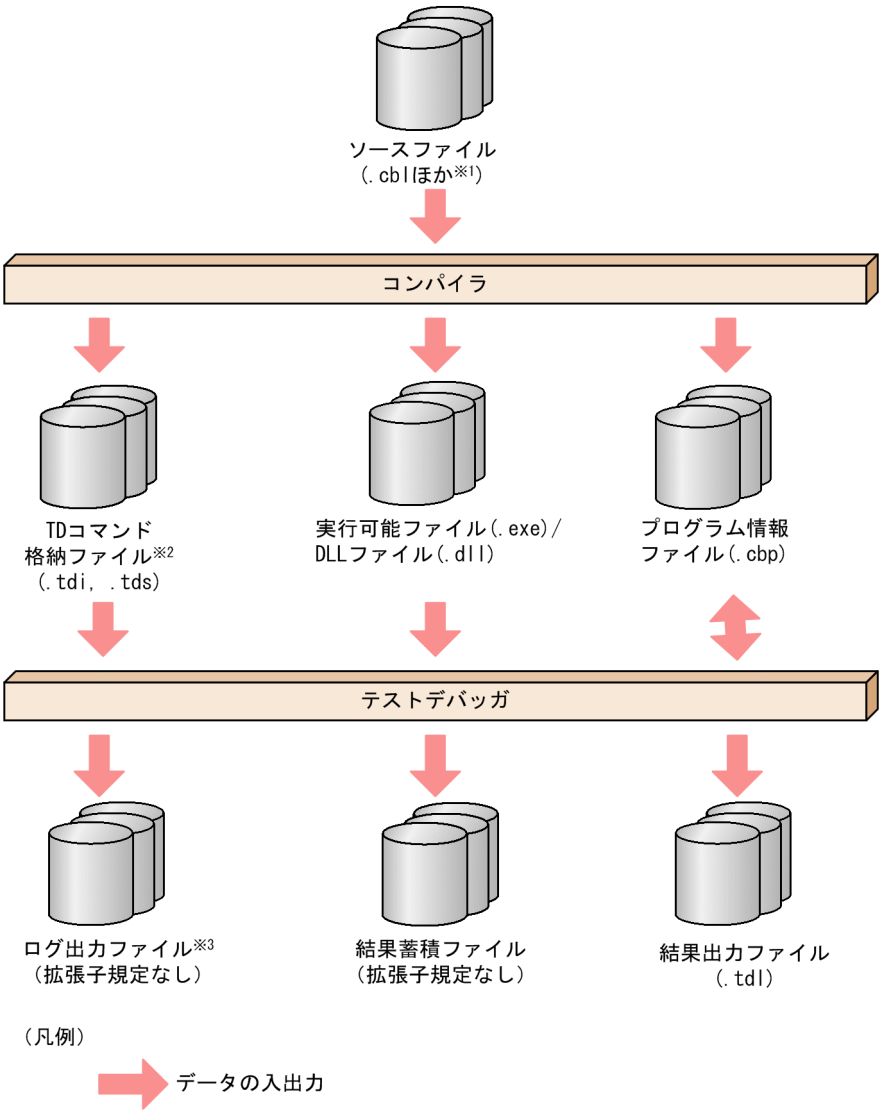
9.1　テストデバッガの前提条件

9.1.1　テストデバッガの入出力構成と使用するファイル

(1)　入出力構成

テストデバッガを使ったプログラムのテスト時の入出力構成を次に示します。

図 9-1　テスト時の入出力構成



注※1

目的に応じて次の拡張子を使用します。

・固定形式正書法で書かれた原始プログラムをコンパイルする場合

.cbl，.cob，.ocb，または環境変数 CBLFIX で指定した拡張子

・自由形式正書法で書かれた原始プログラムをコンパイルする場合

.cbf，.ocf，または環境変数 CBLFREE で指定した拡張子

注※2

コンパイラによる生成と，利用者による作成ができます。

#INCLUDE コマンドで指定するときは，拡張子は任意に指定できます。

注※3

GUI モードの場合だけ使用できます。

(2) 使用するファイル

テストデバッガを使ったテストで使用するファイルについて説明します。

- ソースファイル

  テストデバッガから，テキストエディタを使用して開かれるファイルです。

- 実行可能ファイル

  テストデバッグの対象となる実行可能ファイル（.exe）です。

- DLL ファイル

  テストデバッグの対象となる DLL ファイル（.dll）です。

- プログラム情報ファイル

  テストデバッガで使用する，プログラムの情報を格納したファイルです。テストデバッグのために，-TDInf コンパイラオプションを指定してコンパイルすると，ソースファイル名の拡張子を「.cbp」に変えたプログラム情報ファイルを出力します。

  シミュレーションのために，-SimMain コンパイラオプション，-SimSub コンパイラオプション，または-SimIdent コンパイラオプションを指定してコンパイルすると，コンパイラは拡張子に「.cbs」をつけたプログラム情報ファイルを出力します。プログラム情報ファイルの詳細については，マニュアル「COBOL2002 ユーザーズガイド」の「主／副プログラムシミュレーションで生成する擬似プログラム用プログラム情報ファイル」を参照してください。

- TD コマンド格納ファイル

  TD コマンドを記述するテキストファイルです。記述した TD コマンドに従って，テストが実行されます。

  バッチモードの場合，起動コマンドで指定した TD コマンド格納ファイル（.tdi）に記述された TD コマンドに従ってテストが実行されるため，このファイルは必ず用意する必要があります。#INCLUDE コマンドで，TD コマンド格納ファイルの中に別の TD コマンド格納ファイルを取り込むこともできます。

  GUI モードの場合，ファイル名と拡張子は任意に指定できます。[TD コマンド格納ファイルの入力] メニューを選択し，ファイル選択ダイアログボックスで実行する TD コマンドが登録されているファイルを選択するか，TD コマンド入力ウィンドウに#INCLUDE コマンドを入力して，TD コマンド格納ファイルを選択できます。

#INCLUDE コマンドで入力する TD コマンド格納ファイルに，絶対パスのつかないファイル名をしたときは，検索の順序は次のようになります。また，#INCLUDE コマンドで相対パスが指定されたときは，検索の順序で示しているフォルダからの相対パスとします。

検索の順序

1. 起動コマンド-IncludePath オプションの指定があれば，そのフォルダ，-IncludePath オプションの指定がなければ，-Input オプションの指定するフォルダ
2. 環境変数 CBLPIDIR で指定したフォルダ
3. カレントフォルダ

- 結果出力ファイル

  バッチモードの場合に，起動コマンドで指定した TD コマンド格納ファイル中の TD コマンド操作によるメッセージおよび結果を出力するファイルです。

- ログ出力ファイル

  GUI モードの場合に，モニタウィンドウに表示される内容を出力するファイルです。ファイル名称は，任意に指定できます。オプションダイアログボックスでファイル名を指定します。

- 結果蓄積ファイル

  TD コマンドの実行結果を出力するファイルです。ファイル名称は，任意に指定できます。TD コマンド入力ウィンドウで SET PRINT コマンドでファイルを指定します。PRINT オペランドを持つ TD コマンドで PRINT を指定すると，実行結果がファイルに出力されます。

注意事項

- プログラム情報ファイルは，実行可能ファイルまたは DLL ファイルのあるフォルダを参照します。任意のフォルダのプログラム情報ファイルを参照するときは，環境変数 CBLPIDIR にフォルダ名を指定してください。
- プログラム情報ファイルは，使用するテストデバッガと同じバージョンのコンパイラで作成されたものでなければなりません。異なる場合は，エラーメッセージが出力されます。
- 環境変数 CBLPIDIR を指定した場合のプログラム情報ファイルの検索順序を，次に示します。検索した結果，見つからないときは，プログラム情報ファイルに該当するプログラムはテストデバッグの対象となりません。

  検索順序

  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. 実行可能ファイルに含まれるプログラムは，実行可能ファイルのあるフォルダ

     DLL に含まれるプログラムは，DLL のあるフォルダ

- 次に示すファイルに，複数のテストデバッガから同時に書き込みを行ったとき，あとからファイルをオープンする処理がエラーとなる場合があります。オープンエラーが起きた場合は，「表 9-1　オープンエラーが起きた場合の対処方法」の対処をしてください。

表 9-1　オープンエラーが起きた場合の対処方法

ファイル名	オープンエラーが発生する可能性があるとき	対処方法
ログ出力ファイル	• SET LOG コマンド入力時 • 「オプション」「モニタ」で「モニタウィンドウの内容をファイルに出力する」を選択したとき	ログ出力ファイル名を変更する。
結果蓄積ファイル	SET PRINT コマンド入力時	結果出力ファイル名を変更する。
結果出力ファイル	バッチモードの開始時	• 結果出力ファイル名を変更する。 • 時間を置いて，再度バッチを起動する。

- ファイルの相対パス名を指定したときの起点となるフォルダ，およびパスの付かないファイル名を指定したときのフォルダは，「表 9-2　相対パス名の起点となるフォルダ」に示すとおりです。

表 9-2　相対パス名の起点となるフォルダ

ファイル名	GUI モード	バッチモード
TD コマンド格納ファイル	「テストデバッグの設定ダイアログ」で指定した作業フォルダ	起動コマンドを入力したフォルダ
ログ出力ファイル		−
結果蓄積ファイル		起動コマンドを入力したフォルダ
結果出力ファイル	−	

(凡例)

−：使用できません。

9.1.2　プログラムのコンパイル

テストデバッガを使用するためには，プログラムのコンパイル時に，コンパイラオプションとして-TDInf コンパイラオプションを必ず指定します。また，必要に応じて-SimMain コンパイラオプション，-SimSub コンパイラオプション，または-SimIdent コンパイラオプションも指定します。テストデバッガの機能を使用するために指定する COBOL2002 のコンパイラオプションを次に示します。コンパイラオプションの詳細については，「2.3　コンパイラオプション」を参照してください。

表 9-3　テストデバッガの機能とコンパイラオプション

テストデバッガの機能	コンパイラオプション
テストデバッガを使用する	-TDInf
主プログラムシミュレーションを使用する	-SimMain（擬似主プログラムを生成する）
副プログラムシミュレーションを使用する	-SimSub，-SimIdent（擬似副プログラムを生成する）

注意事項

- リンカオプションに-DEBUG は指定しないでください。指定した場合のテストデバッガの動作は保証しません。
- -Optimize,2 コンパイラオプションによる最適化の指定をして作成したプログラムでは，文が最適化されることがあり，プログラムの動きを保証できない場合があります。また，中断点で参照できるデータ名が，最適化によって参照できない場合があります。

9.1.3　テストしたいプログラムの実行

テストデバッグの対象となるプログラムは，一つの実行可能ファイルと複数の DLL ファイルで構成されます。

テストデバッグ対象のプログラムの実行環境は，COBOL2002 のプログラムの実行環境に基づきます。実行環境の詳細については，「3.3.2　実行環境の設定」を参照してください。

注意事項

- DLL のロードは，COBOL プログラムの実行時の動作に従います。

  指定した DLL がテスト実行時にロードされた場合，テストデバッグの対象になります。ロードされなかった場合，テストデバッグの対象になりません。
- テストデバッガでは，DLL のロード時に以下のチェックを行い，指定された DLL であるかを判断しています。チェックで指定された DLL とは異なると判断された場合は，テストデバッグの対象になりません。

  ・DLL ファイル内で保持している作成時間が等しい。

  ・DLL ファイル内で保持しているファイル名が等しい(フォルダ名は含みません)。
- Windows 以外で作成された実行可能ファイル，または DLL ファイル，もしくは，それ以外のファイルの拡張子を実行可能ファイル，または DLL ファイルの拡張子に変更したものをテストデバッグ対象に指定された場合は，KCCC4200T-S のエラーメッセージを出力し，実行を中止します。
- Windows の別のシステム(32bit Windows の場合は 64bit Windows，64bit Windows の場合は 32bit Windows)で作成された実行可能ファイル，または DLL ファイルがテストデバッガの機能の対象に指定された場合は，KCCC4202T-S のエラーメッセージを出力し，実行を中止します。ただし，64bit Windows 上でも，32bit 用として作成された実行可能ファイル，または DLL ファイルは 32bit のシステムで作成されたものとして扱います。
- 同じ Windows のシステムで作成された COBOL85 の実行可能ファイル，または，DLL ファイルがデバッグ対象に指定された場合は，エラーメッセージは出力しないで，デバッグ対象から外します。

9.1.4　テストプログラムの構成

テストデバッグの対象となるプログラムの構成，プログラム定義，クラス定義，および関数定義の詳細については，マニュアル「COBOL2002　ユーザーズガイド」の翻訳グループについての説明を参照してください。

9.1.5　文字コード

画面やテキストファイルへ入出力する文字コードには，シフト JIS を使用します。

9.1.6　その他の注意事項

(1)　ファイルの書き込み時の制約

- 複数の開発マネージャまたは ccbl2002 コマンドによって，同時にプログラムがコンパイルされた場合，コンパイラが生成する次のファイルは，あとから生成されたファイルが有効になります。
  - プログラム情報ファイル
  - TD コマンド格納ファイル
- 同時に複数のカバレージ情報の蓄積をした場合，プログラムの終了後に，プログラム情報ファイルに書き込みを同時にすると，エラーになるときがあります。

(2)　テストデバッガで動作させるプログラムについての注意事項

テストデバッガは，デバッグ対象プログラムを自身と同じ権限で動作させます。

このため，テストデバッガを標準権限で起動させた場合と，管理者権限で起動させた場合とで，権限の違いによって実行結果が変わることがあります。また，テストデバッガは，デバッグ対象プログラムを実行させたい権限で実行しなければなりません。特に，デバッグ対象プログラムに管理者権限を設定している場合は，必ずテストデバッガを管理者権限で実行しなければなりません。標準権限で実行した場合は，デバッグ対象プログラムの起動に失敗します。

デバッグ対象プログラムの実行権限について，次に示します。

表 9-4　デバッグ対象プログラムの実行権限

テストデバッガの実行権限	デバッグ対象プログラムの設定	
	標準権限	管理者権限
標準権限	標準権限で動作する	起動に失敗する
管理者権限	管理者権限で動作する	管理者権限で動作する

(3)　プログラムからの連動実行についての注意事項

テストデバッガでプログラムからの連動実行をする場合，OS の「グローバルオブジェクトの作成」権限のあるユーザでプログラムを実行してください。権限のないユーザで実行した場合はエラーになります。

9.2.1　プログラムの中断

テスト中のプログラムの実行を任意に中断させることができます。中断させた状態で，次の操作をし，プログラムを調査できます。

- データの値を表示，変更する。
- プログラムの実行経路を表示する。

プログラムを中断させるには，次の方法があります。

- 中断点を設定する。
- データ監視条件を設定する。
- 割り込みによる中断をする。

また，GUI モードでは，次の場合にもプログラムを中断させます。

- プログラムで実行時エラーが発生した。

(1)　中断点の設定による中断

プログラムの実行を，任意の個所で中断させるには，中断点を設定します。中断点を設定すると，実行中のプログラムが中断点に達したときに，中断状態になります。中断中は，中断状態で実行できる TD コマンドだけを指定できます。処理が繰り返し実行されるような個所に中断点を設定する場合は，スキップ回数を指定できます。この指定によって，不要な中断回数を減らすことができ，効率良くプログラムをテストできます。

設定した中断点は，一時的に無効な状態にできます。無効な状態の中断点では，プログラムは中断しません。中断点の有効・無効をすばやく切り替えながらテストデバッグできます。設定した中断点は，一覧を表示して確認できます。中断点の一覧表示については，「10.8.3　中断点一覧の表示」を参照してください。

GUI モードの場合，［中断］メニューから［中断点の設定/解除］を選ぶと，中断点が設定できます。設定した中断点に達したときに，プログラムが中断します。GUI モードでの中断点設定の詳細については，「10.8　中断点の設定と解除」を参照してください。

TD コマンドの場合，SET BREAK コマンドや RESET BREAK コマンドで，中断点を設定したり，解除したりできます。SET BREAK コマンドおよび RESET BREAK コマンドの詳細については，「12.4.19 SET BREAK／RESET BREAK（中断点の設定と解除）」を参照してください。

(2)　データ監視条件の設定による中断

データの値を監視し，その状態によってプログラムを中断できます。監視したいデータの値に条件式を指定することもできます。

GUI モードの場合，［データの操作］メニューから［データ監視条件の設定］を選ぶと，データ監視条件が設定できます。設定した条件が成立したときに，プログラムが中断します。GUI モードでのデータ監視条件設定の詳細については，「10.9　データ監視条件の設定と解除」を参照してください。

TD コマンドの場合，SET WATCH コマンドや RESET WATCH コマンドで，データ監視条件を設定したり，解除したりできます。SET WATCH コマンドおよび RESET WATCH コマンドの詳細については，「12.4.25　SET WATCH／RESET WATCH（データ監視条件の設定と解除）」を参照してください。

データ監視条件の設定による中断の詳細は次のとおりです。

- データ値の変化の監視

  データ値が次の変化をしたときに，プログラムの実行が中断されます。

  - データ値が更新され変化した。
  - アドレス名によって参照されるデータの場合，アドレス不正（アドレスが設定されていない，不正なアドレスが設定されている）によって参照できない状態から，参照できる状態に変化した。または，参照できる状態から，アドレス不正（アドレスが設定されていない，不正なアドレスが設定されている）によって参照できない状態に変化した。

- 比較条件式の監視

  比較条件式の判定結果が次のとき，プログラムの実行が中断されます。比較条件式の評価については，比較の規則（「9.3.1　データの比較・代入規則」の「(2)　データの比較規則」）を参照してください。

  - 比較条件式が，不成立の状態から，成立の状態に変わった（ただし，成立の状態からデータが変更されたとき，さらに成立の状態が継続しても，中断の状態とはならない）。
  - アドレス名によって参照されるデータの場合，アドレス不正（アドレスが設定されていない，不正なアドレスが設定されている）によって参照できない状態から，参照できる状態に変化したときに，比較条件式が成立した。
  - 左辺・右辺の両方のデータが参照できるソース要素に制御が渡ったときに，比較条件式が成立した。

  （例）

  次のプログラムに，（A ＞ B）の比較条件式のデータ監視をすると一連番号 3000 の行で中断します。一連番号 7000 の行で比較条件式は成立しますが，すでに一連番号 3000 の行で成立していて，不成立な状態から成立な状態に変化したわけではないので中断されません。再び比較条件式が不成立な状態から成立の状態になり，中断されるのは一連番号 11000 の行となります。

プログラム

```
001000     MOVE  10 TO A.
002000     MOVE   5 TO B.
003000     ～
005000     ～
```

```
006000     MOVE  11 TO A.
007000     ～.
008000     MOVE   1 TO A.
009000     ～
010000     MOVE  10 TO A.
011000     ～
```

データの監視範囲については，「9.3.2　データ名・ファイル名が参照できる範囲」を参照してください。

注意事項

- COBOL プログラムの連絡節や局所場所節に定義されたデータ項目および特殊レジスタは，次に示す実行単位ごとに，それぞれ異なるデータとして監視されます。

  ・オブジェクト指向のインスタンス単位

  ・シングルスレッドやマルチスレッドなどのスレッド単位

  ・再帰できるソース要素※の再帰単位

注※

再帰できるソース要素とは次の COBOL プログラムを示します。

- RECURSIVE 指定を伴うプログラム
- 利用者定義関数
- メソッド

- 左辺のデータがファクトリ定義（FACTORY）またはインスタンス定義（OBJECT）に定義されたデータのときは，右辺のデータには同じファクトリ定義またはインスタンス定義に定義されたデータが指定できます。
- プログラムの中断時に，データ監視条件を設定した場合，監視対象データが参照できればその位置から，参照できなければ参照できる実行時要素に制御が渡ったときにデータの監視が開始されます。

(3)　割り込みによる中断

［中断］メニューから［割り込み中断］を選ぶと，プログラムの実行を強制的に中断できます。実行の強制中断の詳細については，「10.10.8　実行の割り込み中断」を参照してください。

注意事項

- プログラムが次の状態にあるとき，割り込みによる中断はできません。

  ・-TDInf コンパイラオプションを付けずにコンパイルした COBOL2002 の翻訳単位を実行している間。

  ・COBOL2002 以外の言語で記述されたプログラムを実行している間。

  ・COBOL2002 のソース要素の最後の出口から，プログラムの実行が終了するまでの間。

  これらの状態のプログラムを途中で終了させるときは，タスクマネージャなどの Windows のプロセス管理プロブラムから，プログラムを強制終了させてください。

- ACCEPT 文などによる画面入力の要求中は，画面への入力が行われたあとにプログラムが中断します。

(4)　実行時エラーが発生した場合の中断

GUI モードの場合，実行時エラーが発生したときプログラムが中断します。実行時エラーが発生した文で，プログラムの状態を調べることができます。この中断のための設定は特に必要ありません。

次の条件がすべて当てはまるときに，プログラムが中断します。

- 実行時エラーのメッセージのレベルが S または U のとき，または，CBLABN サービスルーチンによってユーザプログラムが終了したとき（KCCC0900R-I のメッセージが表示されます）。
- プログラムが，実行時エラーによって中断したあとに，データ参照などデバッガの操作の対象にできる状態であるとき。
- -TDInf コンパイラオプションでコンパイルした翻訳単位で実行時エラーが発生したとき。

実行時エラーによってユーザプログラムが中断したときは，次の手順でユーザプログラムを続行できます。

1. データの値を変えるなどのデバッガの操作によって実行時エラーの原因を解決する。
2. 実行時エラーの原因を解決したあとに，ジャンプまたはジャンプ実行によって，実行時エラーが発生した文を再び実行させる。

ただし，実行時エラーの種類によって，ジャンプまたはジャンプ実行ができない場合があります。ジャンプまたはジャンプ実行をしないで，ユーザプログラムを続行したときは，ユーザプログラムが終了します。

9.2.2　プログラムの実行

プログラムの実行を制御する機能には次の種類があります。

(1)　連続実行

テストデバッグウィンドウの［プログラムの制御］メニューから［実行］を選ぶか，または GO コマンドで，プログラムを連続して実行できます。

GUI モードでの連続実行については，「10.10.2　連続実行」を参照してください。

GO コマンドについては「12.4.13　GO（実行の開始／再開）」を参照してください。

(2)　ステップイン

テストデバッグウィンドウの［プログラムの制御］メニューから［ステップイン］を選ぶか，または STEP IN コマンドで，プログラムを 1 文ずつ実行して中断できます。CALL 文，INVOKE 文および関数呼び出しを行う文は，呼び出したプログラムに制御が渡って中断されます。

GUI モードでのステップインについては，「10.10.3　ステップイン」を参照してください。

STEP IN コマンドについては「12.4.26　STEP IN（ステップイン実行の開始／再開）」を参照してください。

(3)　ステップオーバー

テストデバッグウィンドウの［プログラムの制御］メニューから［ステップオーバー］を選ぶか，またはSTEP OVER コマンドで，プログラムを 1 文ずつ実行して中断できます。ただし，ステップインと異なり，CALL 文，INVOKE 文および関数呼び出しを行う文は，呼び出したプログラムに制御が渡って中断されることはありません。

GUI モードでのステップオーバーについては，「10.10.4　ステップオーバー」を参照してください。

STEP OVER コマンドについては「12.4.27　STEP OVER（ステップオーバー実行の開始／再開）」を参照してください。

注意事項

- 呼び出し先プログラムで，中断点の設定があるとき，データ監視の条件が成立したとき，割り込みが実行されたとき，および実行時エラーが発生したときは，これらの条件が起きた文でプログラムが中断されます。
- USE 手続きにプログラムの制御が渡る文でステップオーバーした場合は，USE 手続きの文では中断されないで，USE 手続きを終了した次の文で中断されます。
- プログラムの入口を示すものとして，PROCEDURE または ENTRY を使用します。プログラムの出口を示すものとして，END PROGRAM または END METHOD を使用します。省略された場合は空白行が挿入され，使用されます。

(4)　ジャンプ

ソーステキストウィンドウのポップアップメニューから［ジャンプ］を選ぶか，または STEP TO コマンドで，中断点と異なる文に，中断位置をジャンプできます。次にプログラムの実行を再開したときは，移動した中断位置からプログラムが実行されます。

GUI モードでのジャンプについては「10.10.5　ジャンプ」を参照してください。

STEP TO コマンドについては「12.4.28　STEP TO（ステップツー実行の再開）」を参照してください。

注意事項

- 中断点が属するソース要素の文へジャンプできます。手続き部に宣言部分があるときにできるジャンプは次のとおりです。
  ・宣言部分以外で中断しているとき
  宣言部分以外にジャンプできる。宣言部分にはジャンプできない。
  ・宣言部分で中断しているとき
  中断点が属する節の中の文にジャンプできる。
- ソース要素の入口・出口および ENTRY 文へのジャンプはできません。

- -Optimize,2 コンパイラオプションによる最適化を指定してコンパイルしたプログラムでは文が最適化されることがあります。このため，文を選択してもジャンプ，ジャンプ実行ができないことがあります。また，プログラムの実行順序を変更するジャンプ操作をすると，その後のプログラムの動作を保証しない場合があります。

(5)　ジャンプ実行

ソーステキストウィンドウのポップアップメニューから［ジャンプ実行］を選ぶか，または GO コマンドで，中断点と異なる文から，プログラムの実行を再開できます。

GUI モードでのジャンプ実行については「10.10.6　ジャンプ実行」を参照してください。

GO コマンドについては「12.4.13　GO（実行の開始／再開）」を参照してください。

(6)　実行の強制終了

［プログラムの制御］メニューから［プログラムの実行を終了］を選ぶか，または STOP コマンドで，プログラムの実行を強制的に終了できます。

GUI モードでの実行の強制終了については「10.10.7　実行の中止」を参照してください。

STOP コマンドについては，「12.4.29　STOP（プログラムの強制終了）」を参照してください。

注意事項

-Lib,GUI コンパイラオプションを指定してコンパイルしたプログラムは，プログラムの実行結果をコンソールウィンドウに表示します。プログラムの終了時には，コンソールウィンドウを閉じる代わりに，テストデバッグウィンドウのメニューからプログラムを終了できます。

(7)　実行の開始と開始時にできる指定

プログラムの実行は，次の操作で開始できます。

- 連続実行
- ステップイン
- ステップオーバー

プログラムが終了したあとも，上記の操作で，再びプログラムを開始させることができます。

プログラムの開始時には，次の指定ができます。

- プログラムへ渡す引数
- カバレージ情報の蓄積

プログラムへ渡す引数は，文字列を指定します。指定された文字列は，空白の区切りによって，複数の引数に分けられます。空白を含む文字列を一つの引数として指定するときは，引用符（"）で囲みます。

実行の開始と開始時にできる指定の詳細については「12.4.13　GO（実行の開始／再開）」を，ステップインについては「12.4.26　STEP IN（ステップイン実行の開始／再開）」を，ステップオーバーについては「12.4.27　STEP OVER（ステップオーバー実行の開始／再開）」を参照してください。

［ツール］メニューから［オプション］を選ぶと，GUI モードでのテストデバッグのためのオプションが設定できます。詳細については，「10.18　テストデバッグのオプション設定」を参照してください。

カバレージ情報の蓄積を指定すると，プログラムの実行中にカバレージ情報が採取され，プロセスが終了したときに，採取されたカバレージ情報がプログラム情報ファイルに蓄積されます。蓄積されたカバレージ情報は，カバレージの表示機能で表示できます。

カバレージウィンドウの［カバレージ情報］メニューから［表示］を選ぶと，蓄積されたカバレージ情報を表示できます。詳細については，「14.3.3　カバレージ情報」の「(2)　表示」を参照してください。バッチモードでのカバレージ情報の表示については，「15.3　カバレージ情報の表示（テキスト形式）」を参照してください。

注意事項

プログラムへ渡す引数は，C インタフェース形式（-Main,System コンパイラオプションまたは C 言語のプログラムから呼ばれるプログラム），または VOS3 形式（-Main,V3 コンパイラオプション）の二つがあります。

プログラムへ渡す引数については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

使用例 1

C インタフェース形式の引数に，GO コマンドで値を設定します。

COBOL プログラム

```
01 ARGC PIC 9(8) USAGE COMP.
01 ARGV.
  02 ARGV1 ADDRESS.
  02 ARGV2 ADDRESS.
  02 ARGV3 ADDRESS.
PROCEDURE DIVISION USING BY VALUE ARGC BY REFERENCE ARGV.
             .
             .
             .
```

TD コマンド

- ARGC に 3，ARGV1 にプログラム名のアドレス，ARGV2 に文字列 APPLE のアドレス，ARGV3 に文字列 ORANGE のアドレスを設定します。

```
   GO PARAMETER('APPLE ORANGE')
```

- ARGC に 2，ARGV2 に文字列 APPLE ORANGE のアドレスを設定します。

```
   GO PARAMETER('"APPLE ORANGE"')
```

使用例 2

VOS3 形式の引数に，GO コマンドで値を設定します。

COBOL プログラム

```
01 PARM.
  02 PLEN  PIC S9(4) USAGE COMP.
  02 PCHAR PIC X(30).
PROCEDURE DIVISION USING PARM.
             .
             .
             .
```

TD コマンド

- PLEN に 5，PCHAR に APPLE を設定します。

```
   GO PARAMETER('APPLE ORANGE')
```

- PLEN に 12，PCHAR に APPLE ORANGE を設定します。

```
   GO PARAMETER('"APPLE ORANGE"')
   GO PARAMETER("""APPLE ORANGE""")
```

9.2.3　プログラムの状態の表示

プログラムの状態を表示します。

(1)　呼び出し順序

［ウィンドウ］メニューから［呼び出し一覧ウィンドウ］を選ぶと，ソース要素の呼び出しの順序が表示できます。

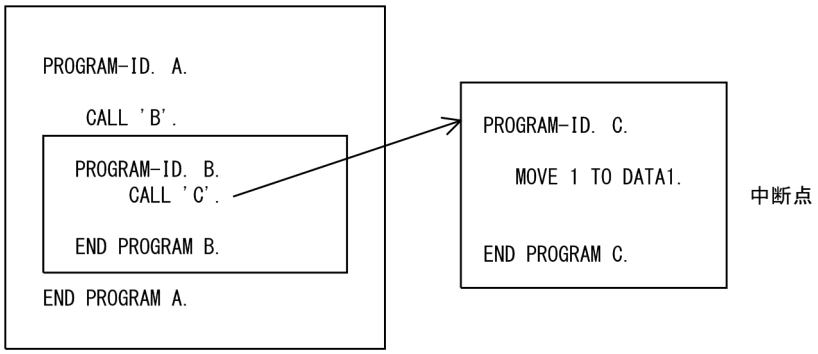
呼び出し一覧ウィンドウについては，「10.3.4　呼び出し一覧ウィンドウ」を参照してください。

(2)　呼び出し順序番号

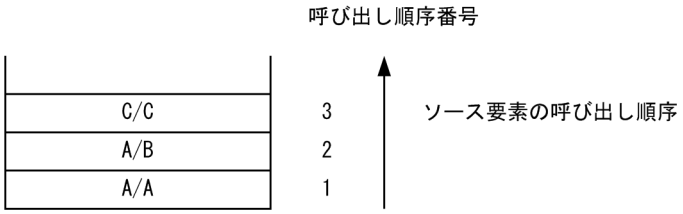
呼び出し順の番号を，呼び出し順序番号とします。

呼び出し順序番号は，最初に制御が渡る実行時要素を 1 として実行時要素が生成されるたびに一つずつ増加します。呼び出し元に戻るときは一つ減少します。

（例）



上記の順でプログラムが呼び出された場合，呼び出し順序番号は次のとおりになります。



(3)　現在位置の表示

［編集/表示］メニューから［現在位置の表示］を選ぶか，または，DISPLAY POINT コマンドでプログラムの中断位置のソーステキストが表示できます。操作対象の実行時要素を変更したあとに，現在位置の表示を行うと，操作対象の実行時要素も中断位置の実行時要素に戻ります。詳細については，「10.14　現在位置の表示」を参照してください。

DISPLAY POINT コマンドの詳細については「12.4.11　DISPLAY POINT（現在位置の表示）」を参照してください。

9.2.4　プログラムの実行の追跡

プログラムの実行経過を追跡して，その経路を表示します。

(1)　アニメーション表示

［ツール］メニューから［オプション］を選んで表示されるオプションダイアログボックスのソーステキストタブで「アニメーションを設定する」のチェックボックスをオンにすると，プログラムの実行中に，ソーステキストの画面で通過点を順次に表示できます。プログラムの実行速度を遅くすることもできます。アニメーションの詳細については「10.11.1　アニメーション」を参照してください。

アニメーションを指定しているときは，常時表示されたデータの値が，実行に伴い変更されます。

(2)　バックトレース表示

［編集/表示］メニューから［バックトレース］を選んで表示されるサブメニューを選択して，中断点に至るプログラムの実行経路を蓄積し，プログラムの中断時にソーステキストの画面上で実行経路を順次表示できます。バックトレース表示については，「10.11.4　バックトレース」を参照してください。

(3)　トレース表示

SET TRACE コマンド，または RESET TRACE コマンドで，プログラムの実行を追跡して，通過点を表示します。同時に，ソース要素の引数のデータ名の値を表示することもできます。

SET TRACE コマンドおよび RESET TRACE コマンドについては「12.4.24　SET TRACE／RESET TRACE（トレース表示の開始と中止）」を参照してください。

(4)　フロー情報の表示

SET FLOW コマンド，または RESET FLOW コマンドでプログラムの実行経路を示す通過点の情報を追跡して，プログラムの中断時にまとめて表示します。プログラムの実行時エラーが発生したときにも，実行時エラー発生までのプログラムの実行経路を表示できます。

SET FLOW コマンドおよび RESET FLOW コマンドについては「12.4.20　SET FLOW／RESET FLOW（フロー情報の蓄積開始と終了）」を参照してください。

9.2.5　データの操作

COBOL プログラムのデータに対して，データの表示，データの代入，データの比較ができます。データの表示によって，プログラムの実行状態を確認できます。また，値を変更することによって，状態を変更してのテストができます。

データの代入規則については「9.3.1　データの比較・代入規則」の「(1)　データの代入規則」を，データの比較規則については「9.3.1　データの比較・代入規則」の「(2)　データの比較規則」を参照してください。

(1)　データの表示

［データの操作］メニューから［データ値の表示］を選ぶか，または DISPLAY DATA コマンドで，データの値をデータ属性または 16 進数で表示できます。

GUI モードでのデータの表示については，「10.13.1　データ値の表示」を参照してください。

DISPLAY DATA コマンドについては，「12.4.8　DISPLAY DATA（データの値表示）」を参照してください。

(a) データ属性表示

データの属性に従い値を表示します。データの値が属性で表示できないときは，16 進数で表示されます。

(表示例)

```
Windows(x86) COBOL2002の場合

  名　称      データ1
  値        123

  名　称      データ2
  エラー
  *001E87C0  0000

Windows(x64) COBOL2002の場合

  名　称      データ1
  値        123

  名　称      データ2
  エラー
  *00000000004172F8  0000
```

(b) 16 進数表示

メモリに位置するアドレスとともに，データの値を 16 進数で表示します。アドレスは，先頭に”*”を付けて表示され，表示できないときは,”*-------- ” が表示されます。データの値は，4 バイト（8 けた）単位に区切って表示されます。

(表示例)

```
Windows(x86) COBOL2002の場合

  名　称      データ3
  *001E87C0  31323334 353637

Windows(x64) COBOL2002の場合

  名　称      データ3
  *00000000004172F8  31323334 353637
```

注意事項

表示されるデータのサイズは，32,767 バイトまでです。これを超えるときは，データの先頭から 32,767 バイトを表示します。32,768 バイト以降のデータは，DISPLAY DATA コマンドにデータ名の部分参照を指定することによって表示できます。バリアントデータは，この制限の対象となりません。

(2) データの代入

[データの操作] メニューから［データ値の代入］を選ぶか，または ASSIGN DATA コマンドで，データに値を代入して，値を変更できます。

GUI モードでのデータ値の代入の詳細については，「10.13.2　データ値の代入」を参照してください。

ASSIGN DATA コマンドの詳細については，「12.4.4　ASSIGN DATA（データの値代入）」を参照してください。

(3)　データの比較

IF コマンドを使用して，比較条件式によって，データとデータの比較，データと定数の比較ができます。比較条件式によって，TD コマンドの実行の条件を指定できます。また，比較条件式は，データ監視条件の設定にも使用できます。

IF コマンドの詳細については，「12.4.15　IF（データの値比較）」を参照してください。

9.2.6　領域の操作

領域を確保し，そのアドレスをアドレス名に設定できます。

アドレスデータ項目またはポインタ項目にも設定できます。

また，任意のデータ項目のアドレスをアドレスデータ項目またはポインタ項目に設定できます。

- 領域の確保と解放
- 任意のデータ項目のアドレスの取得

(1)　領域の確保と解放

アドレス名によって参照されるデータ項目と同じサイズの領域を確保して，アドレス名に設定できます。また，指定したサイズの領域を確保して，アドレスデータ項目，ポインタ項目に設定できます。

アドレス名に設定されているアドレスの領域が不要となったときは，これを解放できます。領域を解放したときは，アドレス名に 0 が設定されます。

主プログラムシミュレーションの引数や副プログラムシミュレーションの RETURNING に，アドレス名を使用するときに，領域を確保して値を設定できます。

なお，アドレスデータ項目，ポインタ項目も同様です。

ALLOCATE AREA コマンドおよび FREE AREA コマンドの詳細については，「12.4.1　ALLOCATE AREA／FREE AREA（領域の確保と解放）」を参照してください。

注意事項

FREE AREA コマンドが解放する領域は，ALLOCATE AREA コマンドで確保した領域です。そのため，FREE AREA コマンドには，ALLOCATE AREA コマンドで確保した領域のアドレスが設定されているアドレス名を指定する必要があります。なお，アドレスデータ項目，ポインタ項目も同様です。

(2) 任意のデータ項目のアドレスの取得

任意のデータのアドレスをアドレス名，アドレスデータ項目，またはポインタ項目に設定できます。

アドレス名で参照されたデータ名のアドレスを取得した場合は，そのアドレス名が指すアドレスが設定されます。

ASSIGN ADDRESS コマンドの詳細については，「12.4.2 ASSIGN ADDRESS（アドレスの取得）」を参照してください。

9.2.7 プログラムの単体テスト

テストデバッガには，主・副プログラムシミュレーション機能，ファイルシミュレーション機能，DC シミュレーション機能などの単体テスト機能があります。単体テスト機能を使用すると，プログラムやファイルなどの実行環境が整わない段階でも，作成したプログラムを単体でテストできます。

テストデバッガでは，プログラムの単体テストを行うために，次のシミュレーションをする機能があります。

- 主プログラムシミュレーション
- 副プログラムシミュレーション
- ファイルシミュレーション
- DC シミュレーション

(1) 主プログラムシミュレーション

SIMULATE MAIN コマンドで，ソース要素を呼び出す主プログラムをシミュレーションします。ソース要素に渡す引数に，TD コマンドで値を設定するなどの操作ができます。呼び出されるソース要素は，プログラム定義です。

SIMULATE MAIN コマンドの詳細については，「12.4.32 SIMULATE MAIN（主プログラムシミュレーションの設定）」を参照してください。

主プログラムシミュレーションは，次の手順で行います。

1. -SimMain コンパイラオプションを指定して，プログラムをコンパイルする。
2. テストデバッガを起動する。
3. テストデバッグ対象のソース要素を開始するときに必要な設定を SIMULATE MAIN コマンドで設定する。テストデバッグ対象のソース要素に渡す引数に値の設定ができる。
4. プログラムを実行して，テストデバッグの操作を開始する。

注意事項

SIMULATE MAIN コマンドで入口名を指定したとき，テストデバッグ対象プログラム中に SIMULATE MAIN コマンドで指定したプログラム名と同じ名前の別のプログラムがある場合，別のプログラムにも同じ入口名が指定された ENTRY 文が必要です。同じ入口名を持つ ENTRY 文がない場合は，エラーメッセージを出力して，主プログラムシミュレーションは設定されません。

主プログラムシミュレーションを設定する場合は，同じ名前を持ち，主プログラムシミュレーションの対象としないプログラムを次のどちらかの方法でデバッグ対象から外してください。

1. 主プログラムシミュレーションの対象としないプログラムは-TDInf コンパイラオプションを付けないでコンパイルしてください。
2. 主プログラムシミュレーションの対象としないプログラムが属する DLL ファイルを，-Library オプションで指定しないでください。

主プログラムシミュレーションの使用例を次に示します。

使用例 1

プログラム SUB1 の開始時に，データ名 PARAM01・PARAM02 に値を設定します。

```
SIMULATE MAIN( #PROG( SUB1 ) )
  ASSIGN DATA( PARAM01 ) VALUE( 1 )
  ASSIGN DATA( PARAM02 ) VALUE(100)
ENDSIMULATE
GO    *>プログラムの実行を開始する。
```

使用例 2

ENTRY 文 'ENTRYSUB'からプログラムの実行を開始します。

```
SIMULATE  MAIN( #PROG( SUB1 ) 'ENTRYSUB' )
  ASSIGN DATA( PARAM01 ) VALUE( 2 )
ENDSIMULATE
GO    *>プログラムの実行を開始する。
```

(2)　副プログラムシミュレーション

SIMULATE SUB コマンドで，テストデバッグ対象のソース要素が呼び出す副プログラムのインタフェースを TD コマンドでシミュレーションします。副プログラムに渡される引数の値を表示して確認したり，副プログラムが値を返す引数に値を設定したりできます。シミュレーションする副プログラムのソース要素は，プログラム定義です。

SIMULATE SUB コマンドの詳細については，「12.4.33　SIMULATE SUB（副プログラムシミュレーションの設定)」を参照してください。

副プログラムシミュレーションをする手順を次に示します。

1. 呼び出す副プログラムがない場合は，次のコンパイラオプションを指定してコンパイルする。

・-SimSub コンパイラオプションで呼び出すプログラム名を指定する。

・一意名を指定した CALL 文によって副プログラムを呼び出す場合は，-SimIdent コンパイラオプションを指定する。

2. テストデバッガを起動する。

3. 副プログラムがする処理を SIMULATE SUB コマンドで設定する。副プログラムが，引数や RETURNING で返すデータ項目に値の設定ができる。

4. プログラムを実行して，テストデバッグの操作を開始する。

呼び出す副プログラムのソース要素がある場合は，-SimSub コンパイラオプションを指定してコンパイルする必要はありません。存在する副プログラムの処理を，SIMULATE SUB コマンドの設定によって，TD コマンドの処理に置き換えて実行させることができます。

CALL 文の引数と RETURNING のデータ項目は，記号名を割り当ててシミュレーションの手続きで参照できます。複数の CALL 文で異なるデータ項目の引数や RETURNING が指定されている場合でも，記号名を割り当てることで，同一の名称によって，シミュレーションの手続きで参照できます。

副プログラムシミュレーションの実行回数を，カウンタ変数によってカウントし，参照できます。

注意事項

- -SimSub コンパイラオプションまたは-SimIdent コンパイラオプションを指定してコンパイルするとき，-Dll コンパイラオプションと同時に指定できません。また，Windows(x86) COBOL2002 の場合，stdcall 呼び出し規約のプログラムのシミュレーションは指定できません。
- 再帰属性の付いたプログラムが自プログラムを呼び出している場合，そのプログラム中に中断している状態で，自プログラムに対しての副プログラムシミュレーションは設定できません。
- SIMULATE SUB コマンドの TD コマンド群中に書かれた ASSIGN DATA コマンドによって，特殊レジスタ RETURN-CODE に設定した値は，シミュレーション対象の副プログラムに RETURNING 指定がない場合だけ，副プログラムを呼び出すプログラムの RETURN-CODE に戻されます。
- ソース要素がある副プログラムにシミュレーションを設定しても，呼び出すプログラムと副プログラムの間の引数および返却項目に矛盾があると，COBOL プログラムが異常終了します。異常終了を避けるためには，環境変数 CBLPRMCHKW を設定してください。
- DLL ファイルから呼び出すプログラムが存在しない場合，擬似副プログラムを作成するコンパイラオプションを指定した副プログラムシミュレーションは設定できません。この場合，呼び出すプログラムのダミープログラムを作成してください。

副プログラムシミュレーションの使用例を次に示します。

使用例 1

プログラム PROG1 をシミュレーションします。

引数 X，Y に記号名 A，B を対応させます。

COBOL プログラム

- データ定義

```
01 X  PIC X(10).
01 Y.
    02 Y1  PIC X(10).
    02 Y2  PIC X(10).
01 RTN.
    02 R-NAME.
       03 N-DATA  PIC X(10).
```

- 呼び出し文

```
CALL  'PROG1'  USING  X,Y  RETURNING  RTN.
```

TD コマンド

```
 SIMULATE SUB( #PROGRAM(PROG1) )  USING(A,B)  RETURNING(R)
   DEFINE
     01 A
     01 B
       02 B1
       02 B2
     01 R
       02 R1
         03 R11
   ENDDEFINE
   IF CONDITION(A=1)
     ASSIGN DATA( B1 )   VALUE ('APPLE')
     ASSIGN DATA( B2 )   VALUE ('PANDA')
     ASSIGN DATA( R11 )  VALUE ('NORMAL')
   ELSE
     DISPLAY  DATA ( A )
     ASSIGN DATA( R11 )  VALUE ('ABNORMAL')
   ENDIF
 ENDSIMULATE
```

(3)　ファイルシミュレーション

SIMULATE FILE コマンドで，プログラムから入出力するファイルがないときに，入出力文によるインタフェースをシミュレーションできます。

SIMULATE FILE コマンドの詳細については，「12.4.31　SIMULATE FILE（ファイルシミュレーションの設定）」を参照してください。

(a)　ファイル入出力文のシミュレーション

入力文のシミュレーションでは，ファイルのレコード領域への値の設定ができます。出力文のシミュレーションでは，レコード領域の内容を表示できます。また，入出力文で，擬似的に入出力条件を発生させることができます。

ファイルシミュレーションは，ファイル結合子とオープンモード（INPUT，OUTPUT，I-O，EXTEND）の単位に，シミュレーションで行う手続きを指定します。さらに，入出力文ごとに手続きを指定できます。

ファイルシミュレーションの実行回数を，カウンタ変数によってカウントし，参照できます。

注意事項

- ファイルシミュレーションは，実体ファイルが割り当てられているときでも指定できます。シミュレーションの手続きが実行されると，実体ファイルへのアクセスは行われません。
- INTO の指定がある READ 文は，シミュレーションの手続きの実行後に，READ 文によるデータが転記されます。レコード領域に代入した値が書き換えられるため，注意が必要です。
- SORT 文，MERGE 文の USING 指定，GIVING 指定で指定するファイルは，READ 文，WRITE 文としてシミュレーション対象となります。ただし，USING 指定に指定されたファイルは GO END コマンドを指定しないかぎりシミュレーションが終わりません。
- ファイルシミュレーションは，シミュレーションを設定したあとの OPEN 文で開始します。ファイルを OPEN したあとにファイルシミュレーションを指定した場合は，ファイルが CLOSE されたあと，再度，OPEN 文が実行されたときにシミュレーションは開始します。
- ファイルシミュレーションの設定と，ファイル名とオープンモードが合致する OPEN 文で，ファイルシミュレーションは開始します。EXTERNAL 句指定のあるファイルでは，同じシミュレーション手続きが実行されます。EXTERNAL 句指定がないファイルでは，ファイルごとに別のシミュレーション手続きが実行されます。
- シミュレーションが設定されているファイルに対して，再度シミュレーションを設定した場合は，設定されていた TD コマンド群を置き換えます。このとき，カウンタ変数および REPEAT 回数は初期値に戻ります。
- EXTERNAL 句指定のファイルのシミュレーションでは，OPEN 文と入出力文が記述された翻訳単位の両方を-TDInf コンパイラオプションでコンパイルします。OPEN 文または入出力文のどちらか記述された翻訳単位が-TDInf コンパイラオプションでコンパイルされていないときは，シミュレーションをしません。

使用例 1

カウンタ変数を使用して，カウンタ変数の値ごとにレコードの値を設定できます。

```
SET QUALIFICATION( #PROG( 社員/社員 ) )
 SIMULATE FILE( 給与ファイル ) OPENMODE( IO ) COUNTER(CNT)
   IF CONDITION(CNT = 1)
      ASSIGN DATA(基本給) VALUE(100000)
   ENDIF
   IF CONDITION(CNT=2)
      ASSIGN DATA(基本給) VALUE(101000)
   ENDIF
   IF CONDITION(CNT=3)
      ASSIGN DATA(基本給) VALUE(104000)
   ENDIF
 ENDSIMULATE
```

同じレコードの値を連続して設定する場合は，反復回数指定を使用できます。

使用例 2

繰り返し指定を使用して，同じレコードの値を連続して設定できます。

```
SET QUALIFICATION( #PROG( 社員/社員 ) )
SIMULATE FILE( 給与ファイル ) OPENMODE(IO)
  REPEAT  TIMES(2)
     ASSIGN DATA(基本給) VALUE(100000)
  ENDREPEAT
  REPEAT
     ASSIGN DATA(基本給) VALUE(101000)
  ENDREPEAT
  REPEAT  TIMES(3)
     ASSIGN DATA(基本給) VALUE(104000)
  ENDREPEAT
ENDSIMULATE
```

繰り返し指定（REPEAT）は，指定された反復回数（TIMES）の回数分を入出力文に対して実行します。指定した反復回数分の実行が終了すると，次の繰り返し指定に移ります。

- 反復回数を省略した場合は，TIMES(1)が仮定されます。
- 繰り返し指定が終了しても，まだ，入出力文が現れた場合は，最後の繰り返し指定が実行されます。

OPEN 文のオープンモードが I-O の場合の入出力文は，複数あります。この場合は入出力文単位にレコード値を設定できる入出力文選択指定を使用すると便利です。

使用例 3

入出力文を指定します。

```
SET QUALIFICATION( #PROG( 社員/社員 ) )
SIMULATE FILE( 給与ファイル )
    OPENMODE(IO)
  SELECT ACTION(DELETE)
     REPEAT TIMES(2)
       DISPLAY DATA(名前)
     ENDREPEAT
  ENDSELECT
  SELECT ACTION(READ)
     REPEAT
       ASSIGN DATA(基本給) VALUE(104000)
     ENDREPEAT
     REPEAT TIMES(3)
       ASSIGN DATA(基本給) VALUE(108000)
   ENDREPEAT
ENDSELECT
ENDSIMULATE
```

文種別単位に処理を繰り返すことができます。文種別指定は SELECT を使用して指定します。

- SELECT 指定のない入出力文で制御が渡った場合，この入出力文に対してはコマンドが実行されません。例えば上の例のようにシミュレーション手続きを実行した場合，START 文の文種別指定がないのでコマンド実行はなく続行します。
- 各文種別単位に反復回数がなくなったときは，使用例 2 のように最後の REPEAT を繰り返します。

- 同じ SELECT 指定が現れた場合は，繰り返し（REPEAT）が設定順に追加されます。

(b)　入出力条件のシミュレーション

ファイルシミュレーションの手続き中で，次の入出力条件を擬似的に発生させることができます。

表 9-5　シミュレーション対象入出力条件

コマンド	シミュレーション対象入出力条件	入出力文
GO END	ファイル終了条件 (END OF FILE)	READ
GO EOP	ページ終了条件 (END OF PAGE)	WRITE
GO INVALID	無効キー条件 (INVALID KEY)	READ, WRITE, REWRITE, START, DELETE
GO ERROR	入出力誤り	READ, WRITE, REWRITE, START, DELETE

(c)　論理誤りチェックと入出力状態の値

ファイルシミュレーションの手続きで，入出力状態の値（FILE STATUS 句に指定されたデータ名の値）に任意の値を設定できます。次の場合は，入出力状態の値にテストデバッガが値を設定します。この場合も，ASSIGN DATA コマンドによって値を変更できます。

ファイルの入出力処理の論理誤りをチェックして，エラーがあった場合，シミュレーションは実行しません。シミュレーション対象ファイルに入出力状態の値（FILE STATUS 句に指定されたデータ名の値）の指定があるときは，シミュレーションの終了状態によって「表 9-6　論理誤りチェック条件」の値を入出力状態の値に設定します。

また，シミュレーション手続きによって「表 9-5　シミュレーション対象入出力条件」で示したコマンドが実行されたときは，「表 9-7　論理誤りチェック以外の設定条件」で示した値が設定されます。

シミュレーション手続きで指定がない場合は，「表 9-8　成功完了の設定条件」の成功完了時の入出力状態の値が設定されます。

表 9-6　論理誤りチェック条件

入出力状態の値	内容
41	開かれているファイル結合子に対して OPEN 文を実行しようとした。

入出力状態の値	内容
46	GO END コマンド実行後に再度 READ 文を実行しようとした。
47	入力モード（INPUT）や入出力モード（I-O）で開かれていないファイル結合子に対して READ 文や START 文を実行しようとした。
48	正しい OPEN モードでは開かれていないファイル結合子に対して WRITE 文を実行しようとした。順アクセス法の場合，ファイル結合子が出力モード（OUTPUT）や拡張モード（EXTEND）で開かれていない。動的アクセス法や乱アクセス法の場合，ファイル結合子が出力モード（OUTPUT）や入出力モード（I-O）で開かれていない。
49	入出力モード（I-O）で開かれていないファイル結合子に対して REWRITE 文や DELETE 文を実行しようとした。

表 9-7　論理誤りチェック以外の設定条件

入出力状態の値	内容
10	GO END コマンド実行による終了。
23	GO INVALID コマンド実行による終了（相対ファイル，索引ファイル）。
34	GO INVALID コマンド実行による終了（順ファイル）。
90	GO ERROR コマンド実行による終了。または，論理誤りが発生した。

表 9-8　成功完了の設定条件

入出力状態の値	内容
00	正常終了。一般のコマンド（入出力状態の値を設定するコマンドは除く）で終了。
07	REEL/UNIT の CLOSE 文を実行した（CLOSE 処理を行わない。OPEN 状態を残す）（順ファイル）。

注意事項

入出力状態の値は，ファイルシミュレーションが開始されたとき，00 に初期化されます。

(4)　記号名

記号名は，次のシミュレーションでデータ項目の名前を置き換えるときに使用します。

- 副プログラムシミュレーション

  CALL 文の呼び出し文の引数に対して，USING と RETURNING で指定するデータ項目を記号名の名前で置き換えて参照できます。

- ファイルシミュレーション

  レコード記述項のデータ項目を記号名の名前で置き換えて参照できます。ファイル記述項に EXTERNAL 句を指定し，複数の翻訳単位から参照されるファイルのシミュレーションで，翻訳単位ごとにレコード記述項の名前に違いがあるときでも，記号名を対応させることによって同一の名称でレコード領域を参照できます。

記号名で置き換えるデータ項目が集団項目の場合は，DEFINE オペランドによって，基本項目をレベル番号と構造定義で指定できます。基本項目を参照しないなど，必要がないときは，DEFINE オペランドの指定は不要です。

レベル番号の指定については，「12.4.37　レベル番号（記号名構造指定）」を参照してください。

注意事項

- 記号名の構造定義は参照が必要な部分までを記述すればよく，原始プログラムのデータ項目の残りのレベル番号は，指定しなくてもかまいません。ただし，途中の引数，レコード記述項を省略して定義することはできません。
- 記号名の指定を省略した場合，データ値の表示や，エラーが発生したときのメッセージ中に記号名を表示できません。このため記号名の代わりに「*****」と表示します。記号名の表示が必要なときは，記号名を記述してください。

（例）

A1 までが記号名に対応し，A11 以降が無視されます。

COBOL プログラム

- データ定義

```
01 A.
   02 A1.
      03 A11 PIC X(10).
      03 A12 PIC X(10).
   02 A2.
```

- 呼び出し文

```
CALL 'SUB1' USING A.
```

TD コマンド

```
SIMULATE SUB(#PROGRAM(SUB1)) USING(P)
  DEFINE
    01 P          *>  Aに対応する。
      02 P1       *>  A1に対応する。
  ENDDEFINE
  :
ENDSIMULATE
```

- 記号名の定義は，01 から始まらなければなりません。構造が原始プログラムと合っていれば，レベル番号を一致させる必要はありません。

（例）

P1 は A1 に対応づけられます。

COBOL プログラム

- データ定義

```
01 A.
   02 A1.
```

```
      03 A11 PIC X(10).
      03 A12 PIC X(10).
   02 A2 PIC X(10).
```

- 呼び出し文

```
CALL 'SUB2' USING A.
```

TD コマンド

```
SIMULATE SUB(#PROGRAM(SUB2)) USING(P)
  DEFINE
    01 P          *>  Aに対応する。
      04 P1       *>  A1に対応する。
  ENDDEFINE
     :
ENDSIMULATE
```

- データ項目の数より記号名の数が多いとき，対応するデータ項目のない記号名を使用した TD コマンドは実行できません。

（例）

P2 に対応するデータ項目の定義がありません。

COBOL プログラム

- データ定義

```
01 A.
   02 A1 PIC X(10).
```

- 呼び出し文

```
CALL 'SUB3' USING A.
```

TD コマンド

```
SIMULATE SUB(#PROGRAM(SUB3)) USING(P)
  DEFINE
   01 P             *>  Aに対応する。
     02 P1          *>  A1に対応する。
     02 P2          *>  対応するデータ項目がない。
  ENDDEFINE
    :
 DISPLAY DATA(P)     *> 実行される。
 DISPLAY DATA(P1)    *> 実行される。
 DISPLAY DATA(P2)    *> 実行できない。
       :
ENDSIMULATE
```

- 記号名を対応づけるデータ項目に TYPE 句または SAME AS 句が指定されているときは，記号名の構造定義には，TYPE 句または SAME AS 句によって展開された構造による指定をします。

（例）

P3 は A1 に，P4 は A2 に対応づけられます。

COBOL プログラム

- データ定義

```
01 TYPE1 IS TYPEDEF.
  02 A1  PIC X(1).
  02 A2  PIC X(1).

01 U1.
  02 PRM1  PIC X(1).
  02 PRM2  TYPE TO TYPE1.
```

- 呼び出し文

```
CALL 'SUB4' USING U1.
```

TD コマンド

```
SIMULATE SUB(#PROGRAM(SUB4)) USING(P)
DEFINE
01 P             *>  U1に対応する。
  02 P1          *>  PRM1に対応する。
  02 P2          *>  PRM2(TYPE1)に対応する。
    03 P3        *>  A1に対応する。
    03 P4        *>  A2に対応する。
ENDDEFINE
  :
ENDSIMULATE
```

使用例 1

副プログラムシミュレーションの引数に記号名を設定します。

COBOL プログラム

- データ定義

```
01 X.
  02 X1 OCCURS 5.
    03 X11 PIC X(2).
  02 X2 PIC X(2).
01 Y  PIC 9.
01 RTN.
  02 R OCCURS 5 PIC X(10).
```

- 呼び出し文

```
CALL  'PROG1'  USING  X,X11(2), Y   RETURNING  R(1).
```

TD コマンド

```
SIMULATE SUB(#PROGRAM(PROG1)) USING(A,B,C)  RETURNING(R)
  DEFINE
     01 A              *>  Xに対応する。
      02 A1            *>  X1に対応する。
       03 A11          *>  X11に対応する。
      02 A2            *>  X2に対応する。
  ENDDEFINE
```

```
  IF CONDITION( C=1 )
    ASSIGN DATA(A11(1)) VALUE('OK')
    ASSIGN DATA(R) VALUE('normal')
  ELSE
    ASSIGN DATA(B) VALUE('NG')
    ASSIGN DATA(R) VALUE('abnormal')
  ENDIF
ENDSIMULATE
```

使用例 2

ファイルシミュレーションのレコードに記号名を設定します。

COBOL プログラム

```
FD 給与ファイル DATA RECORD IS 形式1.
01 形式1.
 02 社員コード PIC X(9).
 02 氏名.
  03 名字  PIC N(10).
  03 名前  PIC N(10).
01 形式2.
 02 契約コード.
  03 種別 PIC X(9).
  03 コード PIC X(40).
01 形式3 PIC X(49).
```

TD コマンド

```
SIMULATE FILE(給与ファイル)  OPENMODE(IO)  RECORD(A,B,C)
  DEFINE
    01 A
     02 A1
     02 A2
      03 A21
      03 A22
    01 B
     02 B1
      03 B11
      03 B12
  ENDDEFINE
     :
ENDSIMULATE
```

使用例 3

ファイルシミュレーションのレコードに記号名を設定します。

COBOL プログラム

```
FD 給与ファイル.
01 形式1.
 02 社員コード PIC 9(8).
 02 氏名.
   03 名字 PIC X(30).
   03 名前 PIC X(30).
01 形式2.
 02 契約コード.
   03 種別 PIC 99.
```

```
   03 コード PIC 9(4).
01 形式3 PIC X(80).
01 形式4 PIC X.
```

TD コマンド

```
SIMULATE FILE(給与ファイル) OPENMODE(IO) RECORD(A,B,C)
   DISPLAY DATA(B)
   :
ENDSIMULATE
```

RECORD 指定で対応づけをします。使用例 2 および使用例 3 の場合，形式 1 が A，形式 2 が B，形式 3 が C に対応します。対応づけた記号化名称をコマンド中でデータの代わりに使用できます。

- B 指定だけを省略するなど，途中の省略はできません。後ろは省略できます。
- レコードが構造定義のとき，構造内を参照するために DEFINE～ENDDEFINE を使用して記号名を定義できます。

(5)　DC シミュレーション

SIMULATE DC コマンドによって，次の DC 文をシミュレーションします。DC 文を使用した OpenTP1 のインタフェースをシミュレーションすることもできます。SIMULATE DC コマンドの詳細については，「12.4.30　SIMULATE DC（DC シミュレーションの設定）」を参照してください。

- COMMIT 文
- ENABLE 文
- ROLLBACK 文
- DISABLE 文
- RECEIVE 文
- SEND 文

対象とする DC 文の種別と通信記述名を指定して，シミュレーションで実行する手続きを TD コマンドで記述します。手続きでは，メッセージファイル領域の参照，値の設定ができます。また，状態コード（STATUS KEY 句で指定したデータ項目）に対して，値を設定できます。

DC シミュレーションの実行回数を，カウンタ変数によってカウントし，参照することもできます。

注意事項

DC シミュレーションは，プログラムの実行開始前でも実行中でも指定できます。DC シミュレーション手続き設定後の DC 文からシミュレーションが実行されます。

使用例 1

DISABLE 文のシミュレーションをします。

```
SIMULATE DC( DISABLE )  CDNAME(CD1)
  ASSIGN DATA( STATUS-KEY ) VALUE( 1 )
ENDSIMULATE
```

使用例 2

COMMIT 文のシミュレーションをします。

```
SIMULATE  DC( COMMIT )
  ASSIGN DATA( STATUS-KEY ) VALUE( 2 )
ENDSIMULATE
```

9.2.8　バッチによるテスト

(1)　テストケース

バッチモードでは，TD コマンドによる処理を，テストケースに分けて記述できます。記述した複数のテストケースは，順番に実行されます。次の方法によって，テストケースを選択して実行することもできます。

- ケース識別子の指定
- ケースコードによるケース選択条件の判定

TEST コマンドは，テストケースの集まりを定義します。CASE コマンドは，ケース識別子による名前と，一連の処理を実行するための TD コマンドの集まりによるテストケースを指定します。

TEST コマンドについては「12.4.34　TEST（テストケースの選択）」を，CASE コマンドについては「12.4.5　CASE（テストケースの指定）」を参照してください。

一つのテストケースの実行が終了したとき，次に示すテストデバッガの機能が設定されていた場合，テストケースの設定が解除されます。解除されたテストケースの設定は，次のテストケースには引き継がれません。プログラムが実行中のときは，強制的に終了されます。テストケースを複数実行している場合でも，データなどは次のテストケースに引き継がれません。

- 中断点
- データ監視条件
- トレース表示
- フロー情報の蓄積
- ファイルシミュレーション
- DC シミュレーション
- 主プログラムシミュレーション
- 副プログラムシミュレーション
- SET QUALIFICATION

(2) ケース識別子の指定

ケース識別子の指定によって，テストケースを選択して実行できます。ケース識別子の末尾に*を付けて指定すると，*より前の文字列が一致するケース識別子のテストケースが選択されて，実行されます。

ケース識別子が指定されている場合にケース識別子の指定がないテストケースは実行されません。

使用例 1

指定されたテストケースから，TEST1 と TEST3 を選んで実行します。テストケースは CASE コマンドが指定された順に実行されます。TEST コマンドの CASEID オペランドに指定したケース識別子の順序は，テストケースの実行順序にはかかわりません。

```
TEST CASEID( TEST1, TEST3 )
  CASE ID(TEST1)               *> ケース識別子TEST1のテストケース
    SET BREAK ST(100)          *> 実行される
    GO
    DISPLAY DATA( RETURN-CODE )
    IF CONDITION(RETURN-CODE NOT= 0)
      ASSIGN CASECODE( 8 )
    ENDIF
    GO
  ENDCASE

  CASE ID(TEST2)               *> ケース識別子TEST2のテストケース
    GO                         *> TESTコマンドで指定されていないため,
 ENDCASE                       *> 実行されない

  CASE ID(TEST3)               *> ケース識別子TEST3のテストケース
    SET TRACE                  *> 実行される
    GO
  ENDCASE


  CASE                         *> ケース識別子の指定がないテストケース
    GO                         *> 実行されない
  ENDCASE

ENDTEST
```

使用例 2

CASEID オペランドに TEST*を指定することによって，TEST で始まるケース識別子のテストケースを実行します。

```
  TEST CASEID( TEST* )
    CASE ID(TEST1)             *> ケース識別子TEST1のテストケース
      ASSIGN CASECODE( 0 )     *> 実行される
    ENDCASE

    CASE ID(TEST2)             *> ケース識別子TEST2のテストケース
      :                        *> 実行される
    ENDCASE

    CASE ID(SAMP1)             *> ケース識別子SAMP1のテストケース
      :                        *> 実行されない
    ENDCASE
  ENDTEST
```

(3) ケースコードによる条件判定

ASSIGN CASECODE コマンドによって，テストケースの実行結果をケースコードに設定できます。ケース選択条件は，このケースコードの値を条件判定して，あとに続くテストケースの実行を決めます。

ASSIGN CASECODE コマンドについては「12.4.3 ASSIGN CASECODE（ケースコードの設定）」を参照してください。

ケース選択条件は，次のどちらかを，<，>，<=，>=，=，NOT= の比較演算子によって比較します。

- 最後に設定されたケースコード（#LASTCODE）と定数
- ケース識別子のケースコードと定数

ケース選択条件に複数の条件式を指定すると，すべての条件を満たしたときに，テストケースを実行します。

使用例 1

最後に設定されたケースコードによって，テストケースを実行します。

```
TEST
  CASE ID(TEST1)
   :
  IF CONDITION(RETURN-CODE NOT= 0)
    ASSIGN CASECODE( 8 )                  *> ケースコードに８を設定する
    ENDIF
  ENDCASE

  CASE ID(TEST2) CONDITION( #LASTCODE=0 )  *> TEST1のケースコードが
                                           *> ８のときは，実行されない。
   :
  ENDCASE

  CASE ID(TEST3) CONDITION( #LASTCODE=8 )  *> TEST1のケースコードが
                                           *> ８のとき，実行される
   :
  ENDCASE
ENDTEST
```

使用例 2

直前に実行されたテストケースがケースコードに値を設定しないとき，ケースコードによるケース選択条件が指定されたテストケースは実行されません。

```
TEST
  CASE ID(TEST1)              *> テストケースを実行する
    ASSIGN CASECODE( 8 )      *> ケースコードに８を設定する
  ENDCASE

  CASE ID(TEST2)                *> 実行する
     :                          *> ケースコードに値を設定しない
  ENDCASE

  CASE ID(TEST3) CONDITION ( #LASTCODE = 8 )        *> TEST2が
                                                    *> ケースコードの
                                                    *> 設定をしないため
     :                                              *> 実行されない
  ENDCASE
ENDTEST
```

使用例３

最初のテストケースにケースコードによるケース選択条件を指定すると，そのテストケースは実行されません。

```
TEST
  CASE ID(TEST1) CONDITION( #LASTCODE = 8 )      *> 実行されない
    ASSIGN CASECODE( 8 )
  ENDCASE
ENDTEST
```

使用例４

同一のケース識別子を持つテストケースが複数あるときは，最後に実行されたテストケースのケースコードを判定します。ケース選択条件に指定されたケース識別子のテストケースがまだ実行されていないときは，判定が偽となり，実行されません。

```
TEST
  CASE ID(TEST1)                        *> 実行される
    ASSIGN CASECODE( 4 )                *> ケースコードに４を設定する
  ENDCASE

  CASE ID(TEST1)                        *> 実行される
    ASSIGN CASECODE( 8 )                *> ケースコードに８を設定する    ★1
  ENDCASE

  CASE ID(TEST2) CONDITION(TEST1 = 8)   *> ★1で設定されたケースコードを判定
         :                              *> 実行される
    ASSIGN CASECODE( 0 )                *> ケースコードに０を設定する    ★2
  ENDCASE

  CASE ID(TEST2) CONDITION(TEST3 = 8)   *> TEST3はまだ実行されていない
    ASSIGN CASECODE( 4 )                *> 実行されない
  ENDCASE

  CASE ID(TEST3) CONDITION(TEST2 = 0)   *> ★2で設定されたケースコードを判定する
         :                              *> 実行される
    ASSIGN CASECODE( 3 )                *> ケースコードに３を設定する    ★3
  ENDCASE

  CASE CONDITION(TEST3 = 3)             *> ★3で設定されたケースコードを判定する
         :                              *> 実行される
    ASSIGN CASECODE( 5 )                *> ケースコードに５を設定する    ★4
  ENDCASE

  CASE CONDITION(#LASTCODE = 5)         *> ★4で設定されたケースコードを判定する
         :                              *> 実行される
  ENDCASE
ENDTEST
```

使用例５

TEST コマンドで選択され，かつ，ケース選択条件の判定が真のときに，テストケースは実行されます。

```
TEST  CASEID(TEST1,TEST3)     *> TEST1, TEST3の実行を指定する ★1
  CASE ID(TEST1)              *> 実行される
    :
  ENDCASE

  CASE ID(TEST2)              *> 実行されない
    :
  ENDCASE

  CASE ID(TEST1)              *> 実行される
    ASSIGN CASECODE( 0 )      *> ケースコードに0を設定する  ★2
  ENDCASE

  CASE ID(TEST2) CONDITION( #LASTCODE = 0 )  *> ★2で設定されたケースコードを判定する
    :                                        *> ケース選択条件は成立するが,
    :                                        *> ★1のTESTコマンド選択されていないため,
  ENDCASE                                    *> 実行されない

  CASE ID(TEST3) CONDITION( #LASTCODE = 4 )  *> ★2で設定されたケースコードを判定する
    :                                        *> ケース選択条件は成立しない
    :                                        *> ★1のTESTコマンド選択されているが
  ENDCASE                                    *> 実行されない
ENDTEST
```

注意事項

TEST コマンドを複数記述したとき，前の TEST コマンドのテストケースで設定されたケースコードは，次の TEST コマンドのケース選択条件では参照されません。

9.2.9　テストデバッグのための設定

テストデバッグをするための設定について説明します。

(1)　テストデバッグ対象とする実行可能ファイルと DLL ファイルの変更

[テストデバッガの制御] メニューから [テストデバッグ対象の設定] を選ぶと，テストデバッガを終了させないで，テストデバッグ対象とする実行可能ファイルと DLL ファイルを変更できます。変更時には，プログラムの実行を終了させておく必要があります。

詳細については，「10.4　テストデバッグ対象とする実行可能ファイルと DLL ファイルの変更」を参照してください。

注意事項

実行可能ファイルと DLL ファイルを変更した結果，元と同じファイルとなった場合でも，初期化の対象とならない機能を除いて，テストデバッガの設定は初期化されます。

(2)　テストデバッガの初期化

[テストデバッガの制御] メニューから [初期化] を選ぶと，テストデバッガで設定した次の機能を解除できます。プログラムが実行中の場合は，プログラムを終了させます。詳細については，「10.5　テストデバッグ機能の初期化」を参照してください。

表 9-9　初期化できるテストデバッガの機能一覧

機能	初期化状態
中断点	解除
データ監視条件	解除
トレース表示	解除
フロー情報の蓄積	解除
すべてのシミュレーションの設定	解除
ユーザパラメタ	解除（オプションダイアログだけ）
ソーステキストウィンドウ	閉じる
ソーステキストウィンドウの，実行済みの文の「*」マーク	解除
常時データ表示ウィンドウのデータ	解除
オプションの「モニタウィンドウの表示内容をファイルに出力する」	解除
アニメーション抑止	解除
データ監視条件設定ダイアログボックスで設定したときに生成される監視識別子	解除

(3)　入力する文字の扱い（等価規則）

［ツール］メニューから［オプション］を選ぶか，または#OPTION コマンドで，入力する文字の扱い（等価規則）を変更できます。詳細については，「10.18　テストデバッグのオプション設定」，または「12.4.36　#OPTION（オプションの変更）」を参照してください。

TD コマンドに使用される文字コードでは，英数字文字は 1 バイトの標準コードに，拡張文字は 2 バイトの拡張コードになります。標準コードの文字を英数字文字（例えば大文字 A と小文字 a）と呼び，拡張コードの文字を拡張文字（例えば大文字Ａと小文字ａ）と呼びます。

英大文字と英小文字，英数字文字と拡張文字を，同じに扱うか区別するかの規則が等価規則です。等価規則の指定がない場合は，同じ文字を表す英大文字と英小文字，英数字文字と拡張文字は同じものとして扱われます。例えば，次に示す，すべての形でコマンドが実行できます。このとき，データ名はすべて英数字文字の大文字 ABC として扱われます。

使用例 1

#OPTION NOCASE を指定した場合は，次の指定はすべて同じ結果となります。

```
DISPLAY DATA(ABC)
display data(abc)
ＤＩＳＰＬＡＹ　ＤＡＴＡ（ＡＢＣ）
```

使用例 2

TD コマンドのオペランド#PROGRAM の#は必ず英数字文字で記述します。

データ名　＃ＤＡＴＡ＠０１は，拡張文字の＃が使用できます。

```
ＤＩＳＰＬＡＹ　ＤＡＴＡ（#ＰＲＯＧＲＡＭ(Ｐ／Ｐ）＃ＤＡＴＡ@０１）
```

等価規則は，次の入力に適用されます。また，アポストロフィ（'）やダブルコーテーション（"）で囲まれた文字列は，等価規則の対象となりません。

- TD コマンドとオペランドで指定するプログラム名，データ名
- ウィンドウから入力するプログラム名，データ名

(4)　TD コマンド格納ファイルの読み込み

［テストデバッガの制御］メニューから［TD コマンド格納ファイルの入力］を選ぶか，#INCLUDE コマンドで，ファイルに格納した TD コマンドを入力し，実行させることができます。シミュレーション手続きなど，テストデバッグのたびに使用する手続きを TD コマンドで記述しファイルに保存することによって，繰り返し実行できます。

詳細については，「10.6　TD コマンド格納ファイルの入力」，または「12.4.35　#INCLUDE（TD コマンド格納ファイルの取り込み）」を参照してください。

TD コマンド格納ファイルは，利用者がテキストエディタによって作成できます。また，-TestCmd,Full，-TestCmd,Break，-TestCmd,Sim コンパイラオプションを指定してコンパイルすることによって自動生成し，必要に応じて TD コマンドの修正や追加をすることで作成できます。

コンパイラが自動生成する TD コマンド格納ファイルについては，「16.　TD コマンド生成機能」を参照してください。

(5)　テスト結果の出力

(a)　結果出力先の指定

［ツール］メニューから［オプション］を選ぶか，SET PRINT コマンド，RESET PRINT コマンド，SET LOG コマンド，または RESET LOG コマンドで，結果出力の出力先を変更できます。GUI モードでは，モニタウィンドウへ表示する結果を，ファイルへ出力できます。バッチモードでは，結果出力ファイル以外のファイルへ出力先を変更できます。詳細については，「10.18　テストデバッグのオプション設定」，または「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」と「12.4.21　SET LOG／RESET LOG（モニタウィンドウ表示内容のファイル出力の設定と解除）」を参照してください。

(b)　注釈の表示

DISPLAY COMMENT コマンドで，任意の文字列をテスト結果に表示できます。テスト結果に説明を追加することによって，わかりやすくできます。詳細については，「12.4.7　DISPLAY COMMENT（注釈の表示）」を参照してください。

9.2.10　GUI の操作のための機能

GUI モードのテストデバッガには，テストデバッグの効率を上げるための，次の機能があります。

(1)　メニューとウィンドウ

GUI モードのテストデバッガには，次のウィンドウと，テストデバッグの機能を指定するメニュー，ツールバー，ステータスバーがあります。

- ソーステキストウィンドウ
- テストデバッグ一覧ウィンドウ
- 呼び出し一覧ウィンドウ
- 常時データ表示ウィンドウ
- モニタウィンドウ
- TD コマンド入力ウィンドウ

詳細については，「10.3.1　テストデバッグウィンドウ」，「10.3.2　ソーステキストウィンドウ」，「10.3.3　テストデバッグ一覧ウィンドウ」，「10.3.4　呼び出し一覧ウィンドウ」，「10.3.5　常時データ表示ウィンドウ」，「10.3.6　モニタウィンドウ」または「10.3.7　TD コマンド入力ウィンドウ」を参照してください。

(2)　指定行の表示

［編集/表示］メニューから［指定行の表示］を選ぶと，ソーステキストウィンドウに指定された行番号の原始プログラムを表示できます。詳細については，「10.7.6　指定行の表示」を参照してください。

(3)　ソーステキストの編集

［編集/表示］メニューから［ソースファイルの編集］を選ぶと，原始プログラムのファイルをテキストエディタで開くことができます。詳細については「10.15　ソースファイルの編集」を参照してください。

(4)　文字列の検索

次のウィンドウでは，［編集/表示］メニューから［文字列の検索］を選ぶと，指定された文字列を検索できます。詳細については「10.16.1　ウィンドウ内の文字列検索」を参照してください。

- ソーステキストウィンドウ
- TD コマンド入力ウィンドウ
- モニタウィンドウ

9.3　データ操作の規則

9.3.1　データの比較・代入規則

(1)　データの代入規則

テストデバッガの代入規則は，原則として COBOL2002 言語の転記規則に従います。COBOL2002 言語の転記規則との差異を次に示します。

COBOL2002 言語の転記規則についてはマニュアル「COBOL2002 言語　標準仕様編」の MOVE 文についての説明を参照してください。

(a)　けたあふれ・不正なデータ

次の代入は，COBOL プログラムの実行時にされる同様の転記と代入される値が異なる場合があります。

- 整数・非整数のデータへの代入でけたあふれが生じる。
- データ項目の属性に適さない値を保持するデータ項目から，整数・非整数にデータを代入する。
- 外部浮動小数点・内部浮動小数点の代入の精度が異なる。

(b)　コンパイラオプション

次のコンパイラオプションは適用されません。

-StdVersion,1，-StdVersion,2，-H8Switch，-MinusZero，-V3Rec,Fixed，-V3Rec,Variable，-EquivRule,StdCode，-V3Rec,Fixed，-V3Rec,Variable，-EquivRule,StdCode，-SpaceAsZero

(c)　拡張 16 進定数

テストデバッガ独自の定数であり，COBOL 言語の 16 進英数字定数・16 進数字定数・16 進日本語定数が代入できないデータ項目へ，16 進数で指定した値を代入できます。

- 受け取り側作用対象が次の項目のとき，送り出し側作用対象の 16 進のけた数が 2 の倍数でなければなりません。
  - 英数字集団項目
  - 英字項目
  - 英数字項目
  - 英数字編集項目
  - 数字編集項目
- 受け取り側作用対象が次の項目のとき，送り出し側作用対象の 16 進のけた数が 4 の倍数でなければなりません。

- 日本語集団項目
- 日本語項目
- 日本語編集項目

- 受け取り側作用対象が次の項目のとき，送り出し側作用対象と受け取り側作用対象のサイズが等しくなければなりません。
  - 整数の数字項目
  - 非整数の数字項目
  - 外部浮動小数点の数字項目
  - 内部浮動小数点の数字項目

(d) 代入規則

代入規則を表 9-10 に示します。

表 9-10 代入規則

送り出し側作用対象＼受け取り側作用対象	英数字集団項目	英字項目	英数字項目	英数字編集項目	数字編集項目	整数※4	非整数※4	外部浮動小数点※5	内部浮動小数点※6	指標名	指標データ項目	外部ブール項目	内部ブール項目	日本語集団項目※2	日本語項目※2	日本語編集項目※2	アドレス名	アドレスデータ項目	ポインタ項目	オブジェクト参照データ項目	OLEオブジェクト参照データ項目	バリアントデータ項目
英数字集団項目	○	○	○	○	○	○	○	○	○					○	○	○						
英字項目	○	○	○	○																		
英数字項目	○	○	○	○	○	○	○	○	○			○	○									
英数字編集項目	○	○	○	○																		
数字編集項目	○		○	○	○	○	○	○	○													
HIGH-VALUE	○	○	○	○										○	○	○						
LOW-VALUE	○	○	○	○										○	○	○						
SPACE	○	○	○	○										○	○	○						
QUOTE	○	○	○	○																		
ZERO	○		○	○	○	○	○	○	○	○		○	○	○	○	○	○	○	○			
英数字定数	○	○	○	○																		
英数字定数(数字だけ)	○	○	○	○	○	○	○															
16進英数字定数	○	○	○	○	○	○	○															
整数定数※4	○		○	○	○	○	○	○	○	※1												
非整数定数※4	○		○	○	○	○	○	○	○													
整数※4	○		○	○	○	○	○	○	○	○												
非整数※4	○		○	○	○	○	○	○	○													
16進数字定数	○		○	○	○	○	○	○	○	○												
外部浮動小数点※5	○		○		○	○	○	○	○													
内部浮動小数点※6	○		○		○	○	○	○	○													
浮動小数点数字定数	○		○		○	○	○	○	○													
指標名						○				○	○											
指標データ項目										○	○											
外部ブール項目			○									○	○									
内部ブール項目			○									○	○									
ブール定数			○									○	○									
日本語集団項目※2	○													○	○	○						
日本語項目※2	○													○	○	○						
日本語編集項目※2	○													○	○	○						
日本語文字定数	○													○	○	○						
16進日本語文字定数	○													○	○	○						
アドレス名																	○	○	○			
アドレスデータ項目																	○	○	○			
ポインタ項目																	○	○	○			
NULL																	○	○	○	○		
オブジェクト参照データ項目																				○		
既定義オブジェクト※3																				○		
拡張16進定数	○	○	○	○	○	○	○	○	○					○	○	○						

（凡例）

○：代入できる　空白：代入できない

注※1

正の整数（0 を含むが，-0 は含まない）だけ代入できます。

注※2

-JPN コンパイラオプションを指定してコンパイルされたプログラム内データ項目の場合，日本語項目／日本語編集項目は，英数字項目／英数字編集項目の代入規則に従います。

日本語集団項目も同様に，英数字集団項目の代入規則に従います。ただし，-JPN コンパイラオプションのうち，-JPN,V3JPNSpace コンパイラオプションは除きます。

注※3

既定義オブジェクトは，SELF，EXCEPTION-OBJECT とします。

注※4

整数は，次のデータ項目で小数点けたを持たない指定です。非整数は，次のデータ項目で小数点けたを持つ指定です。

・外部 10 進項目

・内部 10 進項目

・2 進項目（COMP-X も含む）

注※5

外部浮動小数点形式の数字項目です。

注※6

内部浮動小数点形式の数字項目です。

注意事項

- 次のデータ項目に NULL または不正な値を代入すると，その後，領域が解放されないで残る場合があります。

  NULL または不正な値が代入される場合とは，これらのデータ項目が所属する英数字集団項目に基本項目の属性を考慮しないで値を設定したときなどです。

  ・OLE オブジェクト参照データ項目

  ・オブジェクト参照データ項目

  ・バリアントデータ項目

- テストデバッガでは，TYPEDEF 句の STRONG 指定があるデータ項目への代入は，STRONG 指定がないデータ項目と同様に扱われ代入できます。
- 一般に浮動小数点数は，数値を正確に表現したものではなく，概略の値を表現するもので誤差を含むことが多いです。このため，代入元の値と代入後の値が一致しないことがあります。
- 連絡節で定義した LIMIT 指定がない動的長基本項目には代入できません。KCCC3433T-E のメッセージを出力し，処理を中止します。ただし，主プログラムシミュレーション中で，SIMULATE

MAIN TD コマンドに指定した副プログラム内でだけ代入できます。また，この動的長基本項目は主プログラム内で LIMIT 1024 を定義されているものと仮定されます。

値を代入すると，代入した値の長さが最大長となります。それ以降は，代入した値の長さ以下しか代入できなくなります。

(2) データの比較規則

テストデバッガの比較条件の組み合わせは，次の場合を除いて COBOL 言語の比較規則に従います。

COBOL 言語の比較規則については，マニュアル「COBOL2002 言語　標準仕様編」の「いろいろな式」の「条件式（Conditional expressions)」を参照してください。

(a) けたあふれ・不正なデータ

次の比較は，COBOL プログラムの実行時にされる同様の比較と結果が異なる場合があります。

- データ項目の属性に適さない値を保持するデータ項目の整数・非整数を比較する。
- 外部浮動小数点・内部浮動小数点の比較の精度が異なる。

(b) コンパイラオプション

次のコンパイラオプションは適用されません。

-StdVersion,1，-StdVersion,2，-H8Switch，-MinusZero，-V3Rec,Fixed，-V3Rec,Variable，-EquivRule,StdCode，-V3Rec,Fixed，-V3Rec,Variable，-EquivRule,StdCode，-SpaceAsZero

(c) 拡張 16 進定数

テストデバッガ独自の定数であり，COBOL 言語の 16 進英数字定数・16 進数字定数・16 進日本語定数とは比較できないデータ項目と，16 進数で指定した値で比較ができます。

- 左辺または右辺が次の項目のとき，けた数が 2 の倍数でなければなりません。
  - 英数字集団項目
  - 英字項目
  - 英数字項目
  - 英数字編集項目
  - 数字編集項目
- 左辺または右辺が次の項目のとき，けた数が 4 の倍数でなければなりません。
  - 日本語集団項目
  - 日本語項目
  - 日本語編集項目
- 左辺または右辺が次の項目のとき，両辺のサイズは等しくなければなりません。

- 整数の数字項目
- 非整数の数字項目
- 外部浮動小数点の数字項目
- 内部浮動小数点の数字項目
- アドレス名
- アドレスデータ項目
- ポインタ項目

(d)　比較規則

比較規則を表 9-11 に示します。

表 9-11　比較規則

左辺／右辺	英数字集団項目	英字項目	英数字項目	英数字編集項目	数字編集項目	HIGH-VALUE／LOW-VALUE	QUOTE	SPACE	ZERO	英数字定数	16進数字定数	整数定数※4	非整数定数※4	16進数字定数	外部10進項目	内部10進項目	2進項目・COMP-X	外部浮動小数点数字定数	浮動小数点数字定数	内部浮動小数点※6	指標名	指標データ項目	外部ブール項目	内部ブール項目	ブール定数	日本語集団項目※2	日本語項目※2	日本語編集項目※2	日本語文字定数	16進日本語文字定数	アドレス名	アドレスデータ項目	ポインタ項目	NULL	OLEオブジェクト参照データ項目	バリアントデータ項目	オブジェクト参照データ項目	既定義オブジェクト	拡張16進定数
英数字集団項目	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○									○
英字項目	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			○	○																				○
英数字項目	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			○	○																				○
英数字編集項目	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			○	○																				○
数字編集項目	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			○	○																				○
HIGH-VALUE/LOW-VALUE	○	○	○	○	○										○			○								○	○	○											
QUOTE	○	○	○	○	○										○			○																					
SPACE	○	○	○	○	○										○			○								○	○	○											
ZERO	○	○	○	○	○										○	○	○	○		○	○		○	○		○	○	○			○	○	○		○	○			
英数字定数	○	○	○	○	○										○			○																					
16進英数字定数(X'xxxx')	○	○	○	○	○										○			○																					
整数定数※4	○	○	○	○	○										○	○	○	○		○	○																		
非整数定数※4	○	○	○	○	○										○	○	○	○		○																			
16進数字定数(H'xxxx')	○	○	○	○	○										○	○	○	○		○	○																		
外部10進項目	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	※1																	○
内部10進項目	○								○			○	○	○	○	○	○	○	○	○	※1																	○
2進項目・COMP-X	○								○			○	○	○	○	○	○	○	○	○	※1																	○
外部浮動小数点※5	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○																		○
浮動小数点数字定数	○	○	○	○	○										○	○	○			○																			
内部浮動小数点※6	○								○			○	○	○	○	○	○	○	○	○																			○
指標名									○			○		○	※1	※1	※1					○	○															
指標データ項目																						○	○															
外部ブール項目※7									○															○	○	○												
内部ブール項目※7									○															○	○	○												
ブール定数																								○	○													
日本語集団項目※2	○					○		○	○																	○	○	○	○	○									○
日本語項目※2	○					○		○	○																	○	○	○	○	○									○
日本語編集項目※2	○					○		○	○																	○	○	○	○	○									○
日本語文字定数	○																									○	○	○											
16進日本語文字定数	○																									○	○	○											
アドレス名									○																					○	○	○	○						○
アドレスデータ項目									○																					○	○	○	○						○
ポインタ項目									○																					○	○	○	○						○
NULL																														○	○	○		○	○	○	○	
OLEオブジェクト参照データ項目※7									○																								○	○				
バリアントデータ項目※7									○																								○		○			
オブジェクト参照データ項目※7																																	○			○	○	
既定義オブジェクト※3																																	○			○	○	
拡張16進定数	○	○	○	○	○										○	○	○	○		○						○	○	○			○	○	○						

（凡例）

○：比較できる　空白：比較できない

注※1

整数の場合は比較できます。非整数の場合は比較できません。

注※2

-JPN コンパイラオプションを指定してコンパイルされたプログラム内データの場合，日本語項目／日本語編集項目は，英数字項目／英数字編集項目の比較規則に従います。

日本語集団項目も同様に，英数字集団項目の比較規則に従います。

ただし，-JPN コンパイラオプションのうち，-JPN,V3JPNSpace コンパイラオプションは除きます。

注※3

既定義オブジェクトは，SELF，EXCEPTION-OBJECT とします。

注※4

整数は，次のデータ項目で小数点けたを持たない指定です。非整数は，次のデータ項目で小数点けたを持つ指定です。

- 外部 10 進項目
- 内部 10 進項目
- 2 進項目（COMP-X も含む）

注※5

外部浮動小数点形式の数字項目です。

注※6

内部浮動小数点形式の数字項目です。

注※7

比較条件式は，次のデータ項目を除いて，>，<，>=，<=，=，NOT=の比較演算子を用います。次のデータ項目では=，NOT=の 2 種類の比較演算子を用います。

・ブール項目

・OLE オブジェクト参照データ項目

・バリアントデータ項目

・オブジェクト参照データ項目

・既定義オブジェクト

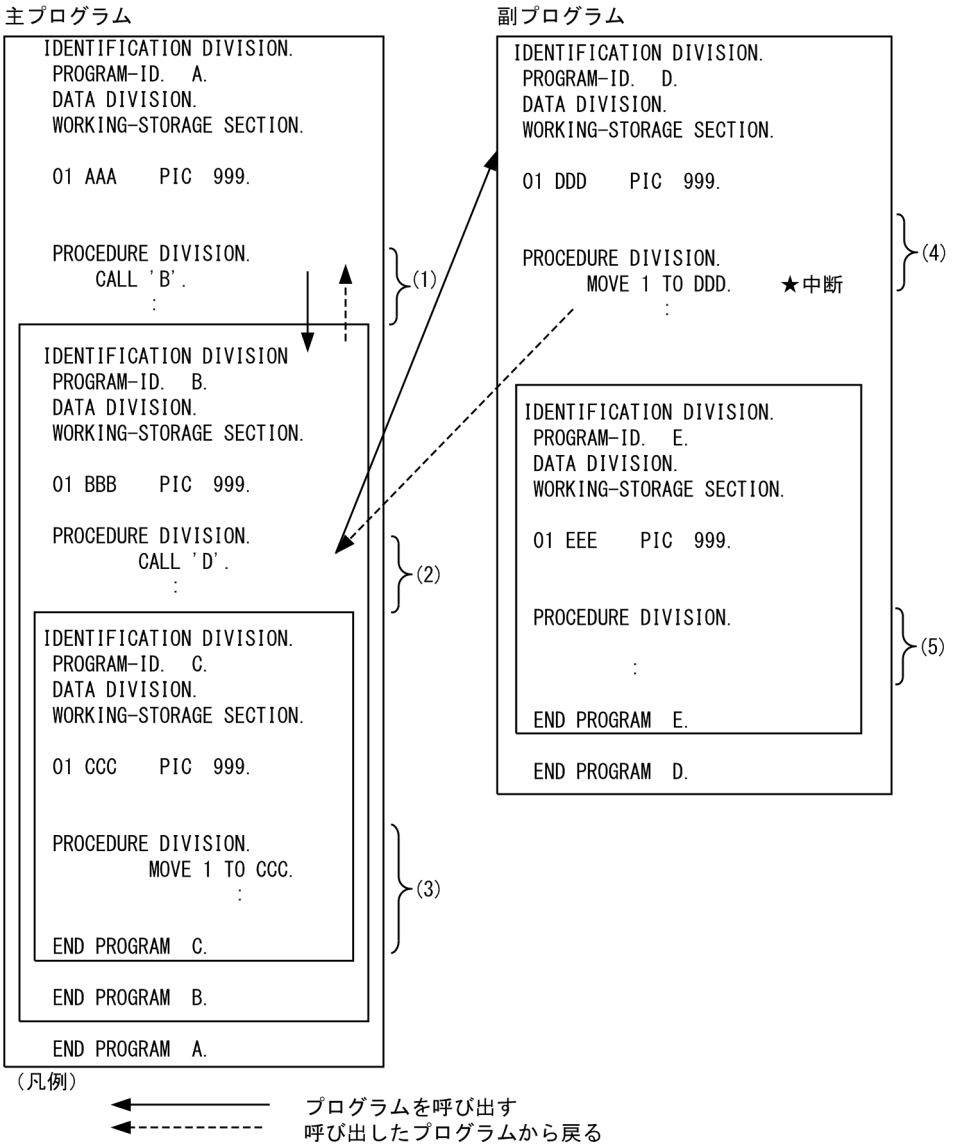
注意事項

- 一般に浮動小数点数は，数値を正確に表現したものではなく，概略の値を表現するもので誤差を含むことが多いです。このため，DISPLAY DATA TD コマンドなどで表示されたデータ値と比較しても等しくならないなど，比較条件の真理値が一致しない場合があります。

9.3.2　データ名・ファイル名が参照できる範囲

テストデバッガで，データに対する操作ができる範囲を次に示します。

(1)　プログラム定義



(1)→(2)→(4)の順でプログラムが呼び出され，★で中断しているとき，表示・代入・比較ができるデータ名は，(1)(2)(4)の各プログラムで参照できる，次のデータです。

(1) データ名 AAA　(2) データ名 BBB　(4) データ名 DDD

データ監視条件で指定した監視対象データは，データの定義のあるプログラムに制御が渡ってから呼び出し元に戻るまでの間，監視されます。プログラムが(1)→(2)→(4)の順で呼ばれ，(4)→(2)→(1)の順で戻り，監視対象のデータがBBBのとき，データは(2)(4)のプログラム実行中に監視されます。
また，左辺・右辺がデータ名の比較条件式の監視では，左辺・右辺両方のデータが参照できる間に監視されます。

9.3.3　プログラムの実行環境の文字コード

データに対する操作は，プログラムの実行環境の文字コードに従います。

(1)　シフト JIS コード

データを表示するときに，次の変換をします。

- 印刷不能文字は，半角ピリオドに置き換える。
- 外字は Windows に登録されている文字を表示する。

(2)　Unicode

Unicode については，「8.4　Unicode 機能」を参照してください。

9.3.4　ファクトリオブジェクトとインスタンスオブジェクト

クラス定義のファクトリ定義（FACTORY）とインスタンス定義（OBJECT）のデータの操作について説明します。

(1)　オブジェクト

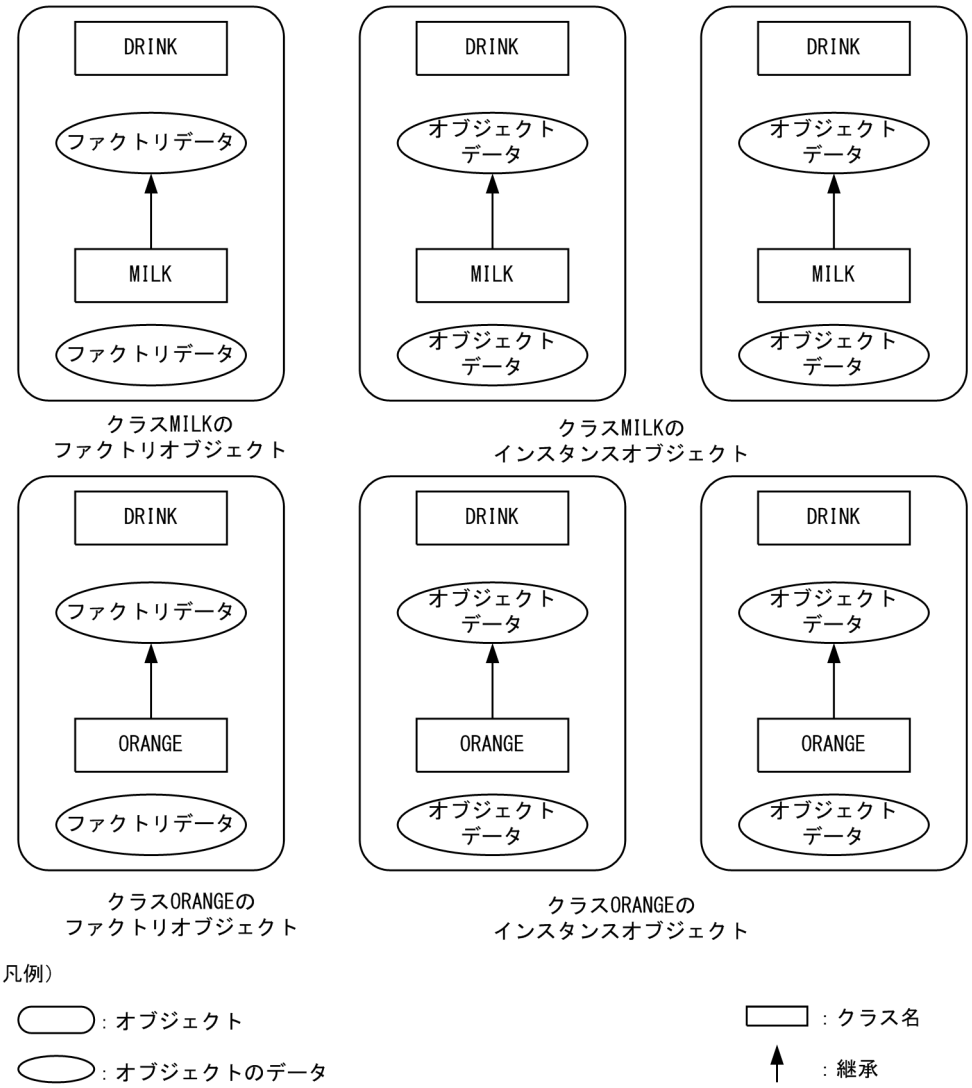
ファクトリオブジェクトは，ファクトリ定義のデータの集まりと，そのデータに対する操作の集まりです。インスタンスオブジェクトは，クラスのインスタンス定義のデータと，そのデータに対する操作の集まりです。クラス定義からは，一つのファクトリオブジェクトと，複数のインスタンスオブジェクトが生成されます。

ファクトリオブジェクトとインスタンスオブジェクトを合わせて，オブジェクトと呼びます。オブジェクトは，オブジェクト参照データ項目または既定義オブジェクトの値によって，一意となります。ファクトリオブジェクトまたはインスタンスオブジェクトに定義されるデータを，オブジェクトデータとします。

ファクトリオブジェクトとインスタンスオブジェクトの例を次に示します。

図 9-2　ファクトリオブジェクトとインスタンスオブジェクト

クラスDRINKを，クラスMILKとクラスORANGEが継承する。
クラスMILKとクラスORANGEのインスタンスオブジェクトがそれぞれ2個ずつ生成された図を示す。



(2)　オブジェクトデータの表示

[データの操作] メニューから [データ値の表示] を選ぶか，DISPLAY OBJECT／DIPLAY FACTORY コマンドで，次のオブジェクトデータが表示できます。

- データ名を指定

  ほかのソース要素のデータと同様に，中断点から参照できるデータを表示できます。

  詳細については，「9.2.5　データの操作」の「(1)　データの表示」を参照してください。

- オブジェクト参照データ項目・既定義オブジェクトを指定

  [データの操作] メニューから [データ値の表示] を選ぶか，DIPLAY OBJECT コマンドで，オブジェクト参照データ項目または既定義オブジェクト SELF，EXCEPTION-OBJECT を指定して，参照するオブジェクトのデータの値を表示できます。

詳細については，「12.4.10　DISPLAY OBJECT／DISPLAY FACTORY（オブジェクトのデータ値の表示）」および「10.13.1　データ値の表示」を参照してください。

- クラス名を指定

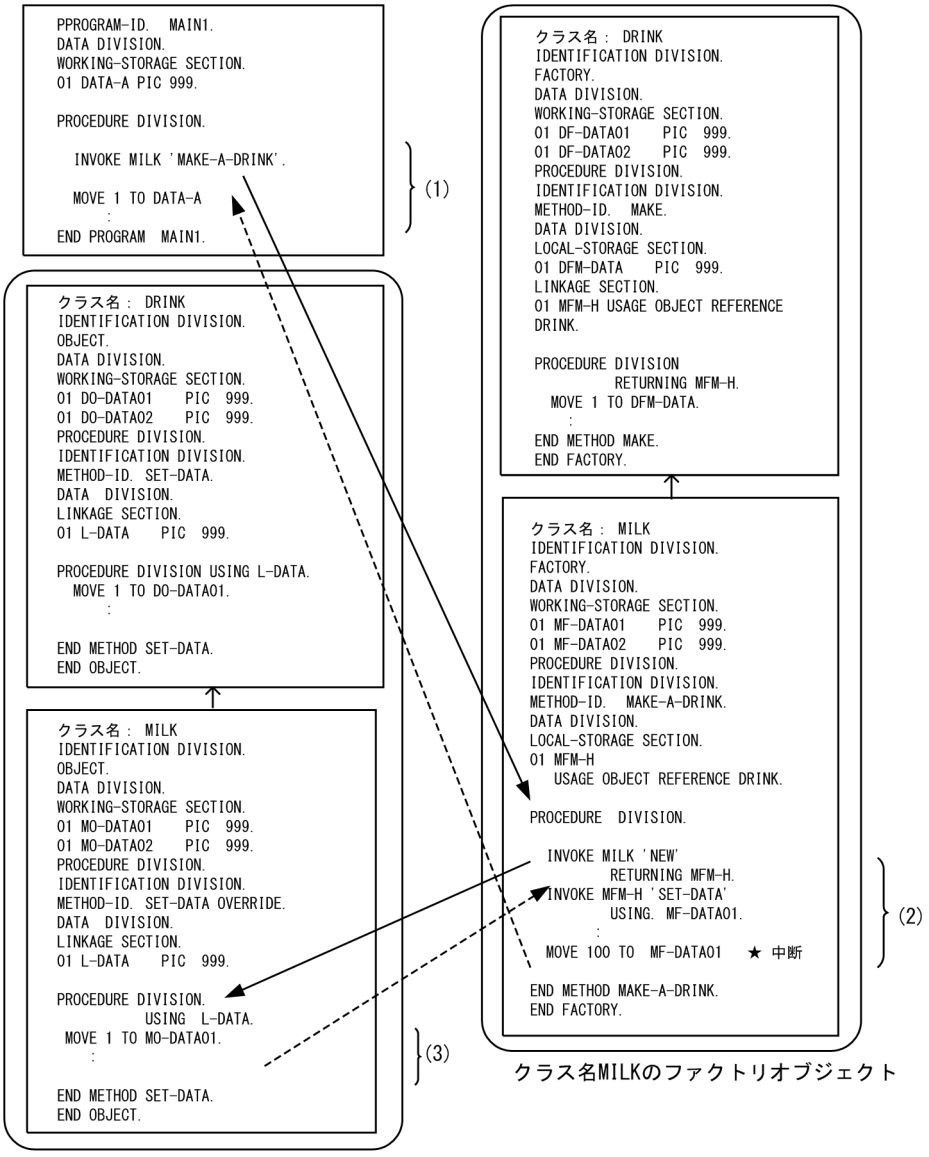
  DISPLAY FACTORY コマンドで，クラス名を指定して，ファクトリオブジェクトのデータの値を表示できます。

  詳細については，「12.4.10　DISPLAY OBJECT／DISPLAY FACTORY（オブジェクトのデータ値の表示）」を参照してください。

注意事項

- オブジェクト生成時のクラスのスーパクラスのデータを表示できます。表示をするときは，スーパクラスも，-TDInf コンパイラオプションを指定してコンパイルします。
- 集団項目に属し，OCCURS 句に DEPENDING の指定のあるデータ項目よりあとにあるデータ項目は，オブジェクトが生成されても，そのメソッドが呼ばれるまで値の表示ができません。

中断点から参照できるデータ項目の例を次に示します。



(1)→(2)→(3)の順にメソッドが呼び出され，(3)のメソッドから戻った後に★で中断しているとき，オブジェクト参照データ項目MFM-Hを使用して参照できるデータ項目は，次のインスタンスオブジェクトのデータ項目です。

・DRINKクラスのDO-DATA01　DO-DATA02
・MILKクラスのMO-DATA01　MO-DATA02

また，同じ★で中断しているとき，既定義オブジェクトSELFを用いて参照できるデータ項目は，ファクトリオブジェクトの次のデータ項目です。

・DRINKクラスのDF-DATA01　DF-DATA02
・MILKクラスのMF-DATA01　MF-DATA02

(3)　オブジェクトデータの監視

ファクトリオブジェクト・インスタンスオブジェクトのデータは，オブジェクトごとのデータの変化を監視します。また，一つのオブジェクトのデータだけの監視を指定できます。一つのオブジェクトのデータの監視を指定したとき，監視の対象とするオブジェクトデータは次のとおりです。

- 指定されたクラスのオブジェクトデータが中断位置で参照できるときは，参照できるオブジェクトデータ

- プログラムの起動前または指定されたオブジェクトデータが中断位置で参照できないときは，データ監視の設定後に，指定されたオブジェクトデータが参照できるメソッドに制御が渡ったときに参照できたオブジェクトデータ（GUI モードでスレッドおよび呼び出し位置を切り替えた場合は，その位置で参照できるオブジェクトデータ）。

データ監視条件の詳細については，「10.9　データ監視条件の設定と解除」を参照してください。

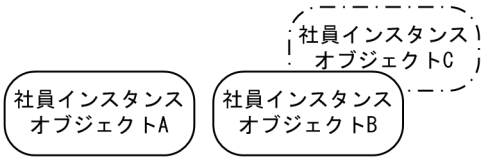
(4)　ファイルシミュレーション

一つのファイル定義で複数のファイル結合子がある場合は，ファイル結合子単位にシミュレーション手続きが実行されます。

（例）

```
ID DIVISION.
CLASS-ID. 社員 INHERITS BASE.
  ⋮
ID DIVISION.
OBJECT.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT  給与ファイル   ASSIGN TO SYS001.
DATA DIVISION.
FILE SECTION.
FD 給与ファイル.
 01 給与レコード.
  02  コード PIC 999.
  02  基本給 PIC 9(8).
  ⋮
METHOD-ID. 計算.
  ⋮
  OPEN INPUT給与ファイル.
  READ 給与ファイル.
  ⋮
END METHOD 計算.
            ⋮
```

```
SET QUALIFICATION( #CLASS （社員/#OBJECT） )
SIMULATE FILE(給与ファイル) OPENMODE( INPUT ) COUNTER( CNT )
  IF CONDITION(CNT = 1)
     ASSIGN DATA(コード) VALUE(1)
     ASSIGN DATA(基本給) VALUE(100000)        (1)
  ENDIF
  IF CONDITION(CNT=2)
     ASSIGN DATA(コード) VALUE(2)
     ASSIGN DATA(基本給) VALUE(101000)
  ENDIF
  IF CONDITION(CNT=3)
     ASSIGN DATA(コード) VALUE(3)
     ASSIGN DATA(基本給) VALUE(104000)
  ENDIF
ENDSIMULATE
```



社員インスタンスオブジェクト A，B…は，それぞれファイル結合子を持つため，一つのファイルシミュレーション定義が個々に有効となります。

社員インスタンスオブジェクト A の計算の READ に対して(1)が実行され，CLOSE 前に社員インスタンスオブジェクト B の計算の READ が実行されても個々にファイルシミュレーション定義が有効なので(1)が実行されます。

カウンタ変数または繰り返し指定の反復回数はファイル結合子単位に実行されるシミュレーション単位にカウントされます。

9.4　その他の機能

9.4.1　マルチスレッド対応 COBOL プログラムのテストデバッグ

マルチスレッド対応 COBOL プログラムのデバッグのための機能と注意事項を説明します。

マルチスレッド対応 COBOL プログラムは，プロセス内で動作するすべての COBOL プログラムを，-MultiThread コンパイラオプションを使ってコンパイルしないと，実行時の動作は保証しません。

マルチスレッド対応 COBOL プログラムの詳細については，マニュアル「COBOL2002　ユーザーズガイド」の，マルチスレッド環境での実行についての記述を参照してください。

(1)　プログラムの実行

マルチスレッド対応 COBOL プログラムを実行するときの仕様は次のとおりです。

- 中断点の設定による中断

  スレッドに依存しないで，中断点に達すれば中断します。
- データ監視条件による中断

  マルチスレッド対応 COBOL プログラムでは，スレッドごとにデータ値の変化を監視します。どれかのスレッドでデータの値が変化するか，または，比較条件式が成立したときに中断します。
- 特定スレッドに対応するデータの値だけを監視する設定もできます。この場合，設定時のプログラムによって，監視するデータは次のとおりになります。
  - プログラムの中断時

    中断しているスレッドの指定ソース要素のデータ
  - プログラムの起動前

    プログラムの起動後に，最初に生成された指定ソース要素を実行するスレッドのデータ
- ステップイン

  ステップインが実行されると，スレッドが切り替わるタイミングに関係なく，次に実行される文で中断します。つまり，ステップインを実行した直後にスレッドの切り替えが発生すると，ステップイン実行前の中断位置とステップイン実行後の中断位置は別のスレッドで実行されるプログラムとなります。
- ステップオーバー

  ステップオーバーを実行したスレッドと同じスレッドで，CALL 文，INVOKE 文および関数呼び出しを行う文を 1 文として実行して中断します。中断点の設定，データ監視，実行時エラー，割り込みによる中断がなければ，ステップオーバーを実行したスレッド以外で中断しません。
- ジャンプ，ジャンプ実行

  ジャンプ，ジャンプ実行したスレッドで，指定された文にジャンプします。ただし，ジャンプ，ジャンプ実行の直後に，別スレッドに切り替わり，そのプログラムで中断点の設定・データ監視・実行時エラー・割り込みなどによって中断した場合は，ジャンプ，ジャンプ実行は行われません。

- 呼び出し順序番号

  マルチスレッドで実行するプログラムの場合は，スレッドごとに呼び出し順序番号の増加と減少が行われます。

(2)　中断時のスレッド

プログラムが中断したとき，すべてのスレッドは停止しています。［ウィンドウ］メニューから［呼び出し一覧ウィンドウ］を選ぶと，COBOL プログラムが動くスレッド ID の一覧を表示できます。詳細については「10.3.4　呼び出し一覧ウィンドウ」を参照してください。

(3)　データの操作

GUI モードでは，中断点のスレッドだけでなく，停止しているほかの COBOL プログラムが動くスレッドのデータも表示できます。ただし，指定したデータに対する転記の処理の途中でスレッドが停止した場合は，正しく値を表示できない可能性があります。ほかのスレッドのデータへ，値の代入はできません。

(4)　ファイルシミュレーション

ファイルシミュレーションは，COBOL2002 言語のファイル結合子ごとに行います。スレッドごとにファイル結合子が生成されるファイルでは，スレッドごとにファイルシミュレーションを行います。

(5)　使用できない機能

次の機能は，マルチスレッド対応 COBOL プログラムでは使用できません。

- DC シミュレーション
- アニメーション
- プロパティ値の表示
- 領域の確保と解放

9.4.2　OLE2 オートメーションのテストデバッグ

［データの操作］メニューから［データ値の表示］を選ぶか，DISPLAY DATA コマンドで，OLE2 オートメーション機能のためのデータに対して，次の操作ができます。

詳細については，「10.13.1　データ値の表示」，または「12.4.8　DISPLAY DATA（データの値表示）」を参照してください。

(1)　バリアントデータ項目の値の表示

バリアントデータ項目は，データの型と VARIANT 値を表示できます。

VARIANT 値は，OLE2（Object Linking and Embedding 2）の VARIANTARG 構造体の vt の値を表示するものです。バリアントデータの型を次に示します。バリアントデータの型がブールのときは，TRUE，FALSE を表示します。バリアントデータの型がその他のときは，VARIANT 値を 16 進で表示します。

表 9-12　データの型と VARIANTARG 構造体の vt の値の対応

データの型	VARIANTARG 構造体の vt の値
整数型	2
長整数型	3
単精度内部浮動小数点型	4
倍精度内部浮動小数点型	5
通貨型	6
日付型	7
文字列型	8
ブール型	11
EMPTY	0
NULL	1
オブジェクト型	9
その他	上記以外（16 進で表示）

注意事項

データ監視条件にバリアントデータ項目を指定したときは，バリアントデータ項目の値を監視します。VARIANT 値は監視しません。

(2)　OLE プロパティの値の表示

OLE プロパティは，OLE オブジェクトのプロパティです。OLE オブジェクトは，次の方法で指定します。

- OLE オブジェクトを参照する OLE オブジェクト参照データ項目を指定する。

OLE プロパティは，バリアントデータの属性を持つため，バリアントデータの型と VARIANT 値を表示できます。

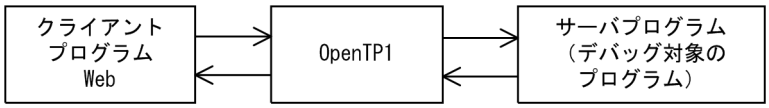
注意事項

データ監視条件に OLE オブジェクト参照データ項目を指定したときは，OLE オブジェクト参照データ項目の値を監視します。OLE プロパティの値は監視しません。

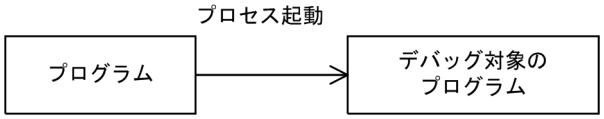
9.4.3　プログラムからの連動実行

ほかのプロセスから起動されることを前提にするプログラムのテストデバッグをするときは，プログラムからの連動実行を使用できます。次の環境で動作するプログラムのテストデバッグをするときに有効です。詳細については，「10.17　プログラムからの連動実行」を参照してください。

- OpenTP1 から起動されるサーバプログラム



- ほかのプログラムからプロセス起動されるプログラム



プログラムからの連動実行を使用する手順は次のとおりです。

1. デバッグ対象のプログラムの実行環境に，環境変数 CBLTDEXEC=TD を指定する。
   環境変数は，システムが提供する方法，または実行支援で指定します。
2. テストデバッガを起動する。
   テストデバッグの対象とする実行可能ファイルと DLL を指定します。
   実行可能ファイルだけ，または DLL だけの指定もできます。
3. ［プログラムの制御］メニューから［連動実行の待機］を選び，テストデバッガを待機状態にする。
4. プログラムを実行する。
   起動されたプロセスの実行可能ファイル名を確認するメッセージボックスが表示されるので，確認して応答します。

サーバプログラムなど，同時に複数プロセスが起動され，そのそれぞれについてデバッグをしたい場合は，必要な数のテストデバッガを待機状態にしておく必要があります。

9.4.4　再帰によるソース要素の実行

プログラムのソース要素が再帰によって実行され，一つのソース要素から複数の実行時要素が生成されたときに，［ウィンドウ］メニューから［呼び出し一覧ウィンドウ］を選ぶと，実行時要素を指定してデータの表示ができます。詳細については，「10.3.4　呼び出し一覧ウィンドウ」を参照してください。

データ監視では，連絡節，局所場所節に定義されているデータ，特殊レジスタは，実行時要素ごとに監視します。カウンタ変数も実行時要素ごとにカウントされます。

9.4.5　共通例外を発生させるプログラムのテストデバッグ

プログラムで共通例外が発生したあとのプログラムの動きは，COBOL の言語仕様に従います。

共通例外の詳細については，マニュアル「COBOL2002　ユーザーズガイド」の共通例外処理の説明を参照してください。

共通例外が発生し伝播が起きることによって，共通例外が発生した実行時要素を呼び出した実行時要素の手続き部の宣言部分が実行される場合があります。このとき，共通例外が発生した実行時要素の出口を通過してから，手続き部の宣言部分が実行されます。次の機能が設定されているときは，実行時要素の出口では対応する機能がそれぞれ動作します。

- 出口指定の中断点による中断
- データ監視条件による中断
- トレース
- フロー
- ステップイン・ステップオーバーによる中断
- アニメーション

注意事項

テストデバッグの対象となるプログラムで実行時エラーが発生して KCCC0015R-S のメッセージを表示した場合，KCCC1207T-I で示される行番号は，KCCC0015R-S のメッセージが示す共通例外を引き起こした文の行番号となります。また，そのときに実行部の表示で強調表示される個所は，共通例外を引き起こした文となります。

9.4.6　数字項目のけた拡張機能（Windows(x64) COBOL2002 で有効）

ここでは，数字項目のけた拡張機能を有効にしたユーザプログラムのテストデバッグについて説明します。数字項目のけた拡張機能の詳細については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

数字項目のけた拡張機能を有効にしたユーザプログラムをテストデバッグの対象にした場合，次に示すデータ名や定数を TD コマンドや GUI モードで使用できます。

- 19～38 けたまで定義したデータ名
- 38 けたまでの固定小数点定数

上記のデータ名，定数が使用できる TD コマンドを「表 9-13　数字項目のけた拡張機能が有効となる TD コマンド」に示します。また，数字項目のけた拡張機能が有効となる GUI モードについては，「表 9-14 数字項目のけた拡張機能が有効となる GUI モード」に示します。

表 9-13　数字項目のけた拡張機能が有効となる TD コマンド

項番	TD コマンド	使用可否	指定できるオペランド
1	SET BREAK	×	−
2	RESET BREAK	−	−
3	SET WATCH	○	CONDITION
4	RESET WATCH	−	−
5	GO	○	PARAMETER
6	STEP IN	○	PARAMETER
7	STEP OVER	○	PARAMETER
8	STEP TO	−	−
9	STOP	−	−
10	SET TRACE	△※1	INTERFACE
11	RESET TRACE	−	−
12	SET FLOW	×	−
13	RESET FLOW	−	−
14	DISPLAY FLOW	−	−
15	DISPLAY BREAK	−	−
16	DISPLAY POINT	−	−
17	DISPLAY DATA	○	DATA
18	DISPLAY OBJECT・DISPLAY FACTORY	×	−
19	GETPROPERTY	×	−
20	ASSIGN DATA	○	VALUE
21	IF	○	CONDITION
22	ALLOCATE AREA	×	−
23	FREE AREA	×	−
24	SIMULATE MAIN	−	−
25	SIMULATE SUB	△※2	記号名
26	SIMULATE FILE	△※2	記号名
27	SELECT ACTION	−	−
28	GO END・GO INVALID・GO EOP・GO ERROR	−	−
29	レベル番号	△※2	記号名

項番	TD コマンド	使用可否	指定できるオペランド
30	REPEAT	×	−
31	SIMULATE DC	−	−
32	TEST	−	−
33	CASE	×	−
34	ASSIGN CASECODE	×	−
35	QUIT	−	−
36	DISPLAY COMMENT	−	−
37	SET QUALIFICATION	−	−
38	RESET QUALIFICATION	−	−
39	SET PRINT	−	−
40	RESET PRINT	−	−
41	SET LOG	−	−
42	RESET LOG	−	−
43	#OPTION	−	−
44	#INCLUDE	−	−
45	ASSIGN DEVICE	−	−
46	!	−	−

（凡例）

○：使用できる。

×：使用できない。

△：データ名または数字定数は指定できないが，影響を受ける。

−：該当しない（データ名または数字定数を指定できない）。

注※1

INTERFACE オペランドを指定し，プログラム引数に数字項目のけた拡張機能が有効になったデータ名を指定した場合でも，データ値が表示されます。

注※2

数字項目のけた拡張機能が有効になったデータ名と記号名を関連づけられます。

表 9-14　数字項目のけた拡張機能が有効となる GUI モード

項番	GUI メニュー	指定個所
1	連続実行	ユーザパラメタ
2	ステップイン	ユーザパラメタ
3	ステップオーバー	ユーザパラメタ
4	データ値の表示	データ名

項番	GUI メニュー	指定個所
5	データ値の代入	代入値
6	常時データ表示に設定	データ名
7	データ監視条件の設定	データ名

注意事項

- 添字，部分参照の指定では 10 けたを超える数字定数は指定できません。詳細は，「12.　TD コマンド」を参照してください。
- 数字項目のけた拡張機能が有効な場合でも，16 進数字定数に 8 文字を超える文字列を指定できません。
- 数字項目のけた拡張機能が有効な場合でも，浮動小数点数字定数の仮数部に 16 けたを超える数字定数を指定できません。

9.4.7　英数字定数の定数長拡張機能

英数字定数の定数長を拡張した場合の，テストデバッガの動作について説明します。

英数字定数の定数長を拡張したユーザプログラムをデバッグ対象にした場合，テストデバッガでも，拡張した項目を使用できます。ただし，#OPTION TD コマンドで 85SYNTAX を指定するか，または GUI モードのオプション設定ダイアログボックスの指定規則タブで「TD コマンドで使用する利用者定義語の最大長」を「COBOL85 の規則と同等にする」にした場合は使用できません。

また，拡張 16 進定数は拡張しません。

9.5　ソース要素・中断点・通過点の表示形式

テストデバッガの画面，およびメッセージに表示する中断点・通過点の形式を説明します。

9.5.1　ソース要素の表示形式

ソース要素は，次の形式で表示します。

(1)　プログラム定義

{ 外部プログラム名/外部プログラム名
外部プログラム名/内部プログラム名 }

(2)　関数定義

```
関数名
```

(3)　クラス定義

{ クラス名/#FACTORY
クラス名/#OBJECT
クラス名/メソッド名 }

9.5.2　入口・出口・文

ソース要素の入口，出口，文は，次の形式で表示します。△は一つ以上の空白を表します。

(1)　ソース要素の入口

```
#ENTRY△＜ ソース要素の表示形式 ＞
```

(2)　ソース要素の出口

```
#EXIT△＜ プログラム識別子 ＞
```

(3)　文

```
文番号△＜ プログラム識別子 ＞
文番号△手続き名△＜ プログラム識別子 ＞
```

手続き名は，文番号の文が手続き名のときに表示します。

9.5.3　スレッド ID

マルチスレッドで実行するプログラムの場合は，各スレッドを識別するスレッド ID を表示します。

10 GUI モードによるテストデバッグ

GUI モードによるテストデバッグについて説明します。GUI モードでは，プログラムを画面上に表示し，確認しながらテストデバッグできます。

10.1　GUI モードによるテストデバッグの概要

GUI モードによって対話形式にテストデバッグする方式について説明します。GUI モードによるテストデバッグには，次の特長があります。

- 実行されたプログラムの追跡

  ソーステキストに実行の経路を表示します。テストプログラムの実行状態を目視できるため，テスト状況を把握しやすくなります。プログラムの実行に合わせてリアルタイムに実行文を表示するアニメーションと，プログラムの実行後に実行された経路を表示するバックトレースがあります。

- プログラムの中断

  ソーステキストから文を指定して，中断点を設定できます。プログラムの実行を中断して，コーディングを検証したいときなどに有効です。設定されている中断点・データ監視条件を，ダイアログボックスの一覧で確認できます。また，設定した中断点を解除することもできます。

- データの確認

  プログラムを中断時に，ソーステキストからデータ名を選択して値の表示とデータ監視条件の設定ができます。これによって，特定の値になる時点まで，プログラムを実行させることができます。また，ウィンドウにデータを登録し，プログラムの中断時に最新の値を自動的に表示させることができます。

- プログラムの構成

  テストデバッグ対象のソース要素の一覧を表示します。ソース要素を選択し，ソーステキストの表示，プログラムの入口および出口に中断点を設定できます。

- シミュレーション

  TD コマンド入力ウィンドウに TD コマンドを入力して，主プログラムシミュレーション，副プログラムシミュレーション，ファイルシミュレーション，ＤＣシミュレーションが実行できます。

- カバレージ

  プログラムを実行しカバレージ情報を蓄積できます。

- プログラムの連動実行

  ほかのアプリケーションから起動されるプログラムのテストデバッグができます。

10.1.1　テストデバッグの手順

プログラムをテストデバッグする作業の流れを説明します。

1. コンパイラオプションを指定して，コンパイルする。

   プログラムのコンパイル時には，コンパイラオプションとして-TDInf コンパイラオプションを必ず指定します。また，必要に応じて-CVInf コンパイラオプション，-SimMain コンパイラオプション，-SimSub コンパイラオプション，または-SimIdent コンパイラオプションも指定します。

   コンパイラオプションの内容を次に示します。

   - -TDInf

テストデバッガを使用する場合に指定する。

- -CVInf
  カバレージを使用する場合に指定する。
- -SimMain
  主プログラムシミュレーション機能を使用する場合に指定する。
- -SimSub または-SimIdent
  副プログラムシミュレーション機能を使用する場合に指定する。

2. 環境変数を設定する。

プログラムの実行に必要な環境変数と，テストデバッガに必要な環境変数があります。環境変数については，「10.2.1　テストデバッガの起動」の「(1)　開発マネージャからの開始」を参照してください。

3. テストデバッガを起動する。

起動方法については，「10.2.1　テストデバッガの起動」の「(1)　開発マネージャからの開始」または「(2)　Windows のプログラム一覧にある［COBOL2002］メニューからの開始」を参照してください。

4. プログラムの実行前の準備を行う。

テストデバッグの目的に応じて，次の操作を行います。

- プログラムを中断させるときは，ソーステキストウィンドウを開き中断点を設定します。
- プログラムにパラメタを渡すときはオプション設定のダイアログボックスで，「ユーザパラメタ」にテストデバッグ対象プログラムに渡すパラメタの値を設定します。ただし，テストデバッグのオプション設定で，実行環境が「実行開始開始時に毎回設定する」になっているときは，実行開始時に設定します。
- カバレージ情報の蓄積をするときは，オプション設定のダイアログボックスで，「カバレージ情報を取得する」をチェックします。
- シミュレーションを行うときは，TD コマンドによって指定します。

5. プログラムの実行を開始する。

［連続実行］，［ステップイン］，［ステップオーバー］のどれかのメニューを選ぶと，プログラムの実行が開始します。連動実行の場合については，［連動実行の待機］を選び，ほかのアプリケーションなどからプログラムを実行します。

6. プログラムを中断する。

中断点を設定することによって，中断点でプログラムが中断します。実行時エラーが発生した場合，割り込みの操作でも，プログラムは中断します。プログラムの中断状態で，データの値の確認や変更ができます。

7. プログラムの実行を再開する。

中断している文から，プログラムの実行が再開します。プログラムが再開する文を，別の文に変更することもできます。

8. プログラムを再実行する。

   プログラムの実行が終了したあと，再度プログラムの実行を開始できます。

9. テストデバッグ対象のプログラムを変更する。

   必要であれば，テストデバッグ対象の実行可能ファイルと DLL ファイルを変更できます。

10. テストデバッガを終了する。

10.2　起動と終了

10.2.1　テストデバッガの起動

テストデバッガを起動するには，次の二つの方法があります。

- 開発マネージャの［ビルド］メニューから［デバッガ］を選ぶ。
- Windows のプログラム一覧にある［COBOL2002］下のメニューから［テストデバッガ for COBOL2002］を選ぶ。

(1)　開発マネージャからの開始

開発マネージャからテストデバッガを起動した場合，開発マネージャで作業中プロジェクトに設定されている実行可能ファイル（.exe），または DLL ファイルをテストデバッグ対象とします。また，作業中プロジェクトがほかのプロジェクトと関連を持っている場合には，条件によって，その関連プロジェクトもテストデバッグの対象となります。開発マネージャの関連プロジェクトについては，「(c)　開発マネージャのテストデバッグ対象プロジェクトについて」を参照してください。

(a)　環境変数の指定

CBLPIDIR

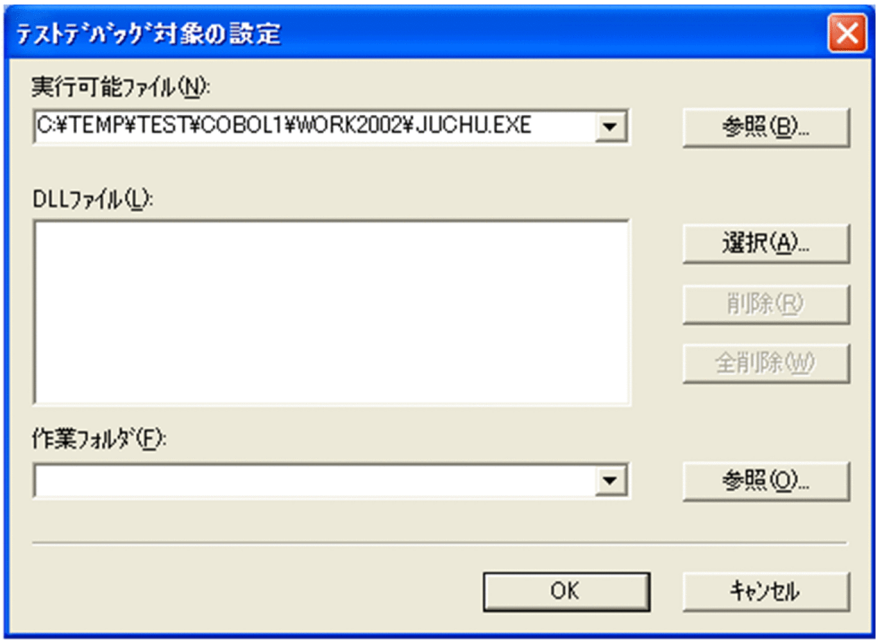
プログラム情報ファイルのあるフォルダ名を指定します。

注意事項

環境変数 CBLPIDIR は開発マネージャで指定できます。ユーザが指定しなかった場合は，作業フォルダが指定されています。開発マネージャから起動されるときに，テストデバッガにこの環境変数は引き継がれます。

(b)　テストデバッガの起動

開発マネージャで［デバッガ］メニューを選択して起動します。テストデバッグ対象の設定ダイアログボックスが表示され，テストデバッグ対象とする実行可能ファイルと DLL ファイルを選択します。実行可能ファイル名として作業中プロジェクトで生成される実行可能ファイル名が表示されます。ただし，最終生成物が DLL の場合は，前回の実行可能ファイル名が表示されます。最終生成物が標準ライブラリの場合は，エラーメッセージが表示され，テストデバッガは起動しません。



1. 実行可能ファイル名を指定する。

   [参照] ボタンで実行可能ファイル選択用のダイアログボックスを表示し，ファイル名を指定できます。

   直接，実行可能ファイルのパス名を指定することもできます。

   以前に選択した実行可能ファイルの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

2. DLL ファイル名を指定する。

   DLL ファイル一覧に表示されるファイル名がテストデバッグの対象になります。DLL ファイルは次の方法で指定します。

   [選択]

   テストデバッグ対象の DLL ファイルを追加します。[選択] ボタンを選ぶと，DLL ファイル選択のダイアログボックスが表示され，DLL ファイル名を指定できます。

   [削除]

   選択した DLL ファイルをテストデバッグの対象から外します。DLL ファイルを選択し，[削除] ボタンを選びます。

   [全削除]

   DLL ファイル一覧に表示されたファイルを，すべてテストデバッグの対象から外します。

3. 作業フォルダを設定する。

   ダイアログボックス表示直後は，何も設定されていません。何も設定しなければ，実行可能ファイルがあるフォルダが作業フォルダとなります。

   変更したい場合は，[参照] ボタンでダイアログボックスを表示し，フォルダ名を指定します。選択したフォルダは，作業フォルダの領域に表示されます。作業フォルダを直接入力することもできます。

   以前に選択した作業フォルダの履歴を 10 回分まで表示し，そこから選択して設定することもできます。

4. [OK] ボタンを選ぶ。

テストデバッグウィンドウが表示されます。

注意事項

- 作業フォルダは，ユーザが設定しなければ，実行可能ファイルのあるフォルダが設定されます。ユーザが設定した場合は，その後，実行可能ファイルを変更しても，作業フォルダは変更されません。実行可能ファイルのあるフォルダを仮定したい場合は，作業フォルダの設定を削除してください。
- 実行可能ファイルを指定しない場合は，必ず DLL ファイルの指定をしなければなりません。
- 実行可能ファイルを指定しないで，かつ，作業フォルダに何も設定しない場合，テストデバッガの起動時のカレントフォルダが，作業フォルダとなります。

(c) 開発マネージャのテストデバッグ対象プロジェクトについて

開発マネージャでの作業中プロジェクトの種類によって，次のような動作となります。

- 開発マネージャから起動される場合，開発マネージャの作業中プロジェクトが実行形式プログラムのプロジェクト（以降，EXE プロジェクトと記述します）またはダイナミックリンクライブラリのプロジェクト（以降，DLL プロジェクトと記述します）の場合，テストデバッグ対象となります。
- 標準ライブラリのプロジェクト（以降，LIB プロジェクトと記述します）または「リンケージ処理を行わない」オプション指定のあるプロジェクトが作業中プロジェクトの場合はテストデバッグ対象としません。

開発マネージャでの作業中プロジェクトが EXE プロジェクトまたは DLL プロジェクトであり，かつ，ほかのプロジェクトと関係設定されている場合，テストデバッグ対象となるプロジェクト，ファイルは次のとおりです。

- 関係設定されているプロジェクト（作業中プロジェクトも含みます）の中に DLL プロジェクトがないとき

  開発マネージャに登録されている実行可能ファイルをテストデバッグ対象とします。前回，テストデバッガを起動したときに設定された DLL ファイルはテストデバッグ対象にしません。
- 関係設定されているプロジェクト（作業中プロジェクトも含みます）の中に EXEL プロジェクトがないとき

  開発マネージャに登録されている DLL ファイル，および前回，テストデバッガを起動したときに設定された EXE ファイル名をテストデバッグ対象とします。
- 関係設定されているプロジェクト（作業中プロジェクトも含みます）の中に DLL プロジェクトが複数あるとき

  すべての DLL ファイルをテストデバッグ対象とします。
- 作業中プロジェクトが EXE プロジェクトのとき

  関連プロジェクト内にほかの EXE があってもテストデバッグ対象とはしません。対象外とする EXE プロジェクトとの下に関連を持っている DLL プロジェクトはテストデバッグ対象とします。
- DLL プロジェクトが作業中プロジェクトであり，関連プロジェクト内に EXE プロジェクトが複数あるとき

次の条件を満たす EXE プロジェクト一つをテストデバッグ対象とし，それ以外の EXE プロジェクトはテストデバッグ対象としません。

- 関連のレベルが上位のもの（作業中プロジェクトに近いもの）。
- 同レベルに複数ある場合は，開発マネージャのプロジェクト設定ダイアログボックスのプロジェクト一覧ツリービューで，上側に書かれているもの。

注意事項

- 複数プロジェクトがテストデバッグ対象となっている場合は，EXE ファイルと DLL ファイルが同一フォルダにある，または環境変数 PATH や環境変数 CBLLDLL を指定するなどの実行環境が整っていなければなりません。実行環境の詳細については，マニュアル「COBOL2002　ユーザーズガイド」のプログラムの実行についての説明を参照してください。
- テストデバッグ対象の設定ダイアログボックスでテストデバッグ対象 DLL ファイルを削除し，再び同じファイルを追加した場合は，テストデバッグ情報ファイルが見つからないため，テストデバッグ対象とならないことがあります。その場合は，一度，テストデバッガを終了し，開発マネージャから，再度，テストデバッガを起動する必要があります。
- 標準ライブラリのデバッグを行いたい場合は，プログラム情報ファイル（.cbp）を，標準ライブラリを取り込んでいる実行可能ファイルまたは DLL ファイルのプログラム情報ファイルが出力されているフォルダに格納してください。

(2)　Windows のプログラム一覧にある［COBOL2002］メニューからの開始

(a)　環境変数の指定

「(1)　開発マネージャからの開始」を参照してください。

(b)　テストデバッガの起動

Windows のプログラム一覧にある［COBOL2002］下のメニューから［テストデバッガ for COBOL2002］を選ぶと，テストデバッグ対象の設定ダイアログボックスが表示されます。テストデバッグ対象の設定ダイアログボックスで，テストデバッグ対象とする実行可能ファイルと DLL ファイル，および作業フォルダを選択します。

テストデバッグ対象の設定ダイアログボックスの操作については，「(1)　開発マネージャからの開始」を参照してください。

10.2.2　テストデバッガの終了

テストデバッガを終了するには，次の二つの方法があります。

- **［テストデバッガの終了］メニューを選ぶ。**

- システムメニューから［閉じる］を選ぶ。

テストデバッガのメニューで指定することによって，テストデバッグウィンドウが閉じられ終了します。

注意事項

プログラムの実行中にテストデバッガを終了させると，テストデバッグ対象のプログラムは強制終了されます。［テストデバッガの終了］メニューを選択すると，「テストデバッグ対象プログラムの実行を終了します」というメッセージが表示されます。［OK］ボタンを選択すると，テストデバッガの終了とともに，プログラムが強制終了されます。

10.3　テストデバッグで使用するウィンドウ

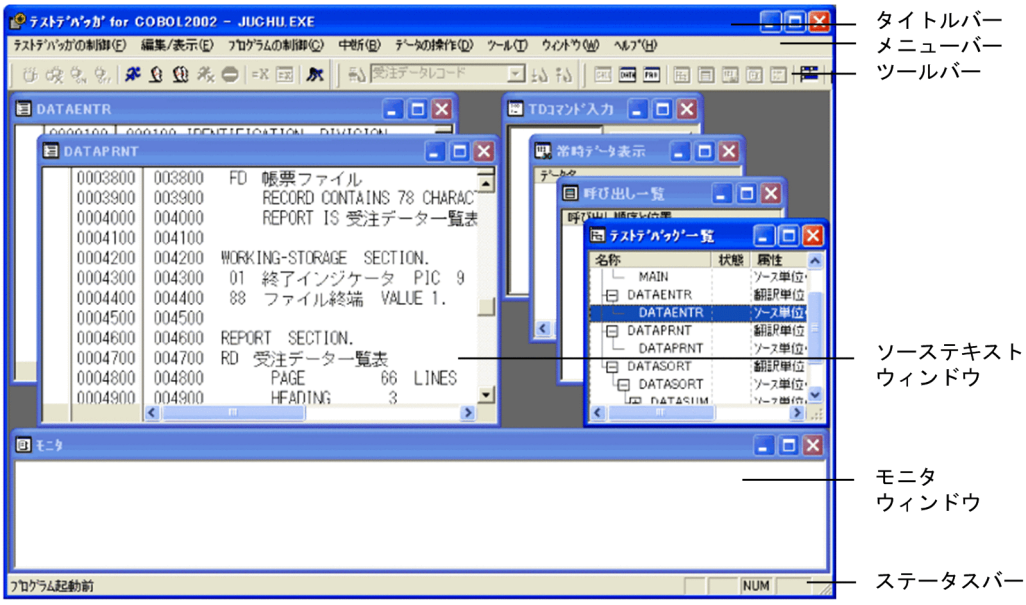
テストデバッグウィンドウは，各種ウィンドウと，メニューおよびツールバーなどから構成されています。これらを使用して，テストデバッグをします。

注意事項

- テストデバッガの起動時，ソーステキストウィンドウおよびテストデバッガ終了時に非表示にしていたウィンドウは，次のテストデバッガ起動時には表示されません。
- 表示中のどれかのウィンドウを最大化してテストデバッガを終了すると，次のテストデバッガ起動時には，最前面に表示されるウィンドウは最大化します。ただし，ソーステキストウィンドウを最前面に最大化で表示してテストデバッガを終了した場合，次のテストデバッガ起動時には，ソーステキストウィンドウは表示されないで，その他のウィンドウは元のサイズで表示されます。

10.3.1　テストデバッグウィンドウ

図 10–1　テストデバッグウィンドウ



(1)　タイトルバー

現在，テストデバッグ対象の実行可能ファイル名が表示されます。デバッグ対象の実行可能ファイル名を指定しなかった場合は，「実行可能ファイル指定なし」と表示されます。

(2) メニューバー

表 10–1 メニューバー（テストデバッグウィンドウ）

メニューバー	メニュー項目	機能
テストデバッガの制御(F)	テストデバッグ対象の設定(T)	テストデバッグ対象プログラムを指定する。
	TD コマンド格納ファイルの入力(I)	TD コマンド格納ファイルの内容を実行する。
	初期化(N)	テスト環境を初期状態にする。
	テストデバッガの終了(X)	テストデバッガを終了する。
編集/表示(E)	切り取り(T)	文字列を切り取る。
	コピー(C)	文字列をクリップボードにコピーする。
	貼り付け(P)	文字列をクリップボードからカーソル位置に貼り付ける。
	すべて選択(A)	すべての文字列を選択する。
	クリア(L)	すべての文字列を消去する。
	ソースファイルの編集(E)	テキストエディタを起動し，ソースファイルを表示する。
	文字列の検索(F)	文字列を検索する。
	指定行の表示(O)	指定した行を表示する。
	現在位置の表示(R)	現在の中断個所の実行文を表示する。
	バックトレース(B)	[進む(F)]： バックトレース位置を一つ先に進める。 [戻る(B)]： バックトレース位置を一つ前に戻す。 [先頭(T)]： 蓄積されているバックトレース情報の中で，最初にプログラム実行された文にバックトレース位置を表示する。 [終了(E)]： バックトレース位置の表示を終了する。
プログラムの制御(C)	連続実行(G)	連続実行をする。
	ステップイン(S)	呼び出すプログラムを対象として 1 ステップ実行する。

メニューバー	メニュー項目	機能
	ステップオーバー(T)	呼び出すプログラムを対象外として 1 ステップ実行する。
	連動実行の待機(W)	連動実行されるまで待機状態にする。
	プログラムの実行を終了(E)	プログラムの実行を終了する。
中断(B)	中断点の設定/解除(B)	中断点の設定/解除を行う。
	中断点の一覧(P)	中断点の一覧を表示する。
	すべての中断点の解除(R)	すべての中断点を解除する。
	割り込み中断(I)	実行を中断する。
データの操作(D)	データ値の表示(D)	データ名の値を表示する。
	データ値の代入(V)	データ名に値を代入する。
	常時データ表示に設定(T)	常時データ表示ウィンドウにデータ名を設定する。
	データ監視条件の設定(S)	データ監視条件を設定する。
	データ監視条件の一覧(W)	データ監視条件の一覧を表示する。
ツール(T)	オプション(O)	各種オプションを設定する。
	カスタマイズ(U)	ウィンドウの文字フォントや色を設定する。
ウィンドウ(W)※	重ねて表示(C)	ウィンドウを重ねて表示する。
	並べて表示(T)	ウィンドウを並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	標準配置(S)	ウィンドウを標準位置に表示する。
	ソーステキストウィンドウ(R)	[呼び出し位置の表示(C)]： ソーステキストの呼び出し位置を表示する。 [定義部の表示(F)]： ソーステキストの定義部を表示する。 [実行部の表示(M)]： ソーステキストの実行部を表示する。 [すべて閉じる(W)]： 開いているソーステキストウィンドウをすべて閉じる。
	テストデバッグ一覧ウィンドウ(U)	テストデバッグ一覧ウィンドウを表示する。
	呼び出し一覧ウィンドウ(K)	呼び出し一覧ウィンドウを表示する。
	常時データ表示ウィンドウ(D)	常時データ表示ウィンドウを表示する。

メニューバー	メニュー項目	機能
	モニタウィンドウ(M)	モニタウィンドウを表示する。
	TD コマンド入力ウィンドウ(I)	TD コマンド入力ウィンドウを表示する。
	ステータスバー(B)	ステータスバーの表示／非表示を切り替える。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※

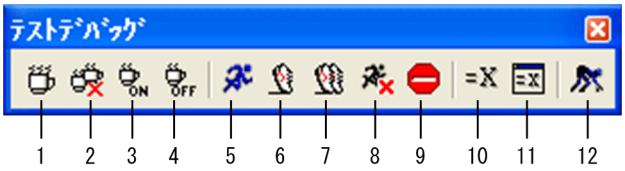
ウィンドウとは，ソーステキストウィンドウ，呼び出し一覧ウィンドウ，常時データ表示ウィンドウ，TD コマンド入力ウィンドウ，モニタウィンドウのことをいいます。

(3)　ツールバー

ツールバーのボタンを選ぶとメニューと同じ操作ができます。ツールバーは，カスタマイズダイアログボックスで表示するかどうかを指定できます。ツールバーの種類は，次のとおりです。カスタマイズダイアログボックスについては，「10.3.9　テストデバッグで使用するウィンドウのカスタマイズ」を参照してください。

(a)　テストデバッグ用ツールバー

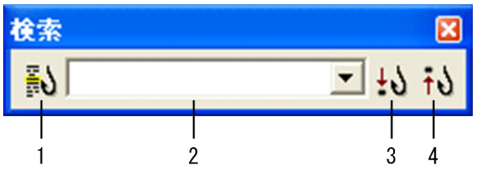
テストデバッグ用ツールバーの構成と各ボタンに割り当てられている機能を次に示します。



1. 中断点の設定/解除
2. すべての中断点の解除
3. 中断点の有効
4. 中断点の無効
5. 連続実行
6. ステップイン
7. ステップオーバー
8. プログラムの実行を終了
9. 割り込み中断
10. データ値の代入
11. データ値の表示
12. 連動実行の待機

(b)　検索用ツールバー

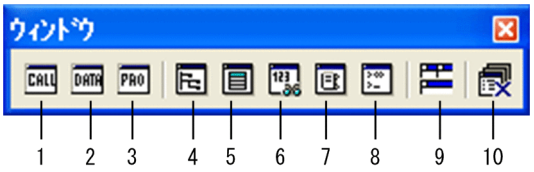
検索用ツールバーの構成と各ボタンに割り当てられている機能を次に示します。



1. 文字列の検索
2. 検索文字列入力領域
3. 次を検索
4. 前を検索

(c)　ウィンドウ用ツールバー

ウィンドウ用ツールバーの構成と各ボタンに割り当てられている機能を次に示します。



1. 呼び出し位置の表示
2. 定義部の表示
3. 実行部の表示
4. テストデバッグ一覧ウィンドウの表示
5. 呼び出し一覧ウィンドウの表示
6. 常時データ表示ウィンドウの表示
7. モニタウィンドウの表示
8. TD コマンド入力ウィンドウの表示
9. 標準配置
10. ソーステキストウィンドウをすべて閉じる

注意事項

- 検索文字列入力領域にフォーカスが位置づけられている場合に Enter キーを押すと［次を検索］と同じ動作が行われます。
- 検索文字列入力領域にフォーカスが位置づけられている場合に使用できるショートカットキーは，切り取り（[Ctrl] ＋ [X]），コピー（[Ctrl] ＋ [C]），貼り付け（[Ctrl] ＋ [V]），すべて選択（[Ctrl] ＋ [A]），次を検索（[F3]），前を検索（[Shift] ＋ [F3]）だけです。

(d)　バックトレース用ツールバー

バックトレース用ツールバーの構成と各ボタンに割り当てられている機能を次に示します。



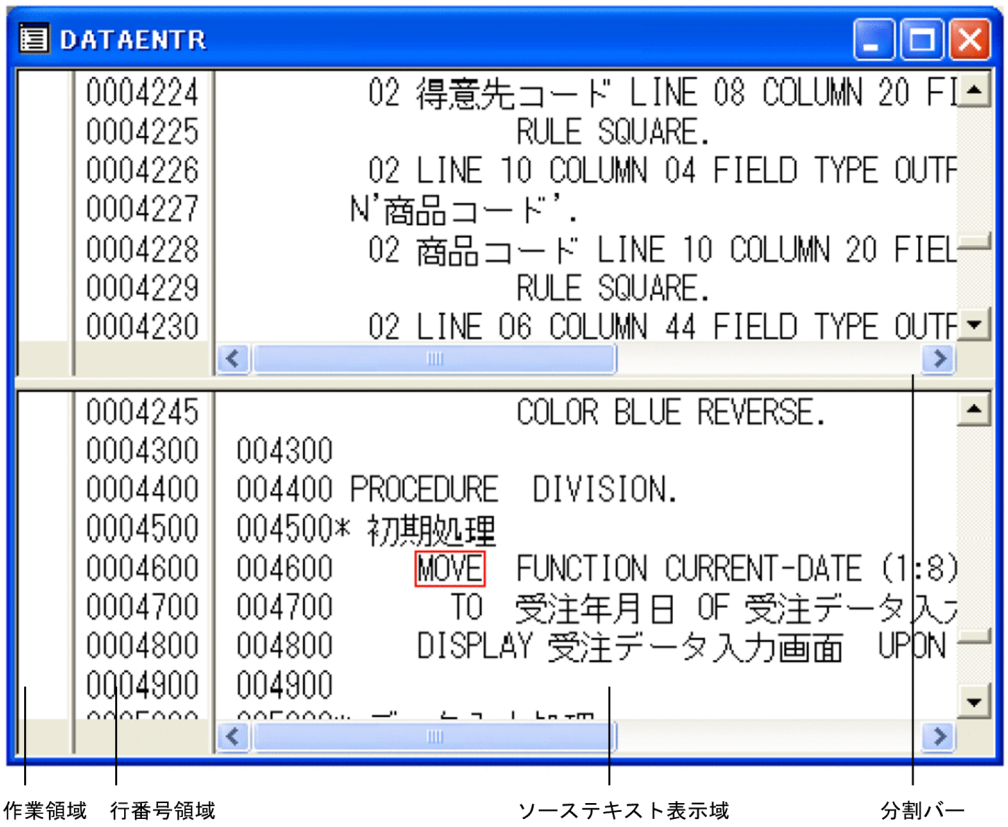
1. 進む
2. 戻る
3. 先頭
4. 終了

10.3.2　ソーステキストウィンドウ

ソーステキストウィンドウは，行番号とソーステキストを表示します。プログラムの実行を確認し，テストデバッガに対する操作を指示できます。

ソーステキストウィンドウは，原始プログラムの単位でソーステキストを表示し，複数のソーステキストウィンドウを同時に表示できます。ただし，一つのソース要素のソーステキストウィンドウを複数開くことはできません。

図 10-2　ソーステキストウィンドウ



- 作業領域

  オプションダイアログボックスで，「実行履歴を表示する」をオンにしている場合，実行済みの文には，「＊」が表示されます。1 文が複数行に渡っているときは，先頭行だけに「＊」が表示されます。また，1 行が複文になっているときは，1 文でも実行されていれば「＊」が表示されます。

  中断点の設定と解除ができます。中断点の設定と解除については，「10.8　中断点の設定と解除」を参照してください。

- 行番号領域

  翻訳後の行番号が 7 けたの数字で表示されます。行番号領域は表示を消すことができます。行番号領域の表示，非表示の変更については，「10.18　テストデバッグのオプション設定」を参照してください。

- ソーステキスト表示域

  ソーステキスト表示域の語を選択し，中断点の設定やデータの表示ができます。語を選択して行う操作については，「10.3.1　テストデバッグウィンドウ」を参照してください。

  語の選択は，次のどちらかの方法で行います。

  - マウスで選択したい最初の位置を選び，最後の位置までドラッグします。
  - ［Shift］キーを押したまま，選択したい最初の位置から最後の位置までカーソルを移動します。

  また，語をダブルクリックすることによって，次の操作ができます。

  - 文キーワードをダブルクリックします。

    中断点を設定します。

- データ名をダブルクリックします。

  モニタウィンドウへデータを表示します。

中断点が設定されると，文キーワードは，次のとおりに表示されます。

- 中断点が有効状態のとき，実線の矩形で囲む。
- 中断点が無効状態のとき，破線の矩形で囲む。

中断点の有効，無効については，「10.8　中断点の設定と解除」を参照してください。

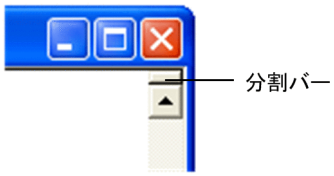
プログラムが中断したとき，文キーワードは，カスタマイズされた色の矩形で強調表示されます。ソーステキストウィンドウのカスタマイズについては，「10.3.9　テストデバッグで使用するウィンドウのカスタマイズ」を参照してください。

ソーステキスト表示域内にカーソルを位置づけ，［ソースファイルの編集］メニューを選択すると，テキストエディタを起動し，カーソルの位置づいている行を含むソーステキストを表示します。ソースファイルの編集などで使用するテキストエディタは，開発マネージャで設定できます。デフォルトではCOBOL エディタが設定されています。詳細については，「1.7.3　テキストエディタの設定」を参照してください。

- 分割バー

  同じソーステキストの別の部分を，同じウィンドウに同時に表示できます。

  分割されていないウィンドウを分割するには，垂直スクロールバー上部にある分割バーをポイントし，マウスポインタの形が変わったら，分割したい位置までドラッグします。



  ウィンドウの分割を解除するには，分割バーをダブルクリックします。

  また，テストデバッグ一覧ウィンドウからの定義部表示や実行部表示を行った場合，すでに対象のソーステキストが表示されているときは，画面が分割されて定義部や実行部が表示されます。また，ソーステキストウィンドウからの定義データ行の表示でも同様に画面が分割されます。

注意事項

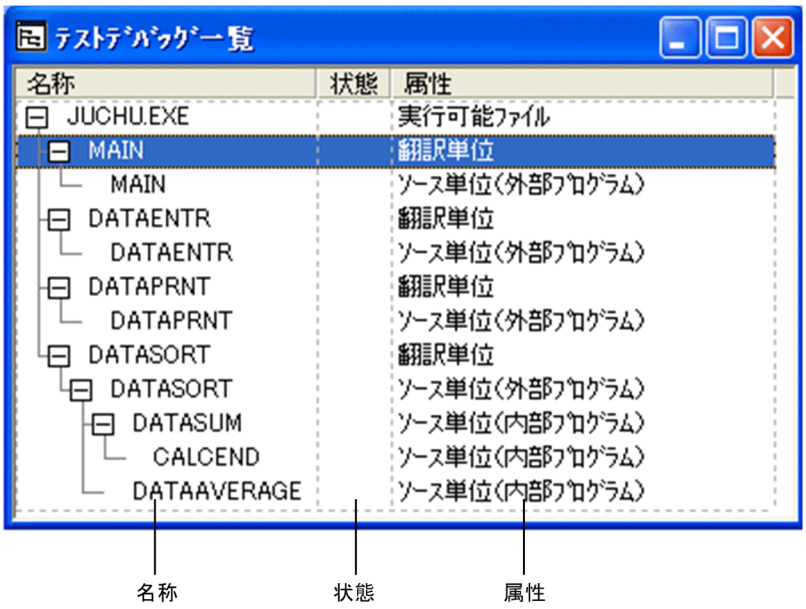
- ソーステキストウィンドウに表示するソーステキストは，原始プログラムに書かれた COPY 文，REPLACE 文などの登録集原文の展開をコンパイラが行ったあとのテキストです。LISTING 指令，条件翻訳によってコンパイラが無視する文も表示します。
- 行番号領域に表示される行番号は，コンパイラが翻訳時に振り直した番号です。
- ソーステキストウィンドウを開ける最大数は 25 個です。25 個を超えた場合は，使用順位のいちばん古いソーステキストウィンドウが自動的に閉じられます。
- 表示するソーステキスト中のプログラムの最後を示す「END PROGRAM」が省略されている場合は，プログラムの出口を表示するための空白の 1 行を追加して表示します（出口中断挿入行）。

10.3.3　テストデバッグ一覧ウィンドウ

テストデバッグ一覧ウィンドウには，プログラムの構成を一覧で表示します。ソース単位を選択して，次の操作ができます。

- 定義部または実行部のソーステキストを表示します。
- プログラムの入口および出口に中断点を設定します。

図 10-3　テストデバッグ一覧ウィンドウ



- 名称

  次の一覧を，次の順番の階層のツリー構造で表示します。

  - 実行可能ファイル名と DLL ファイル名
  - 翻訳単位名
  - ソース単位名

- 状態

  入口中断，出口中断，アニメーション抑止の表示を行います。

  各機能が設定されると，左側から，入口中断，出口中断，アニメーション抑止の順で「*」が表示されます。

  入口中断と出口中断で中断設定が無効になると「－」と表示されます。

  (例)

  入口中断，アニメーション抑止が設定され，出口中断が未設定の場合，「＊△△＊」と表示されます。

  入口中断，出口中断が設定され，出口中断は無効の場合，「＊△－△△」と表示されます。

  △は一つ以上の空白を表します。

- 属性

  ソース要素の種別を表示します。

注意事項

- -TDInf コンパイラオプションでコンパイルした翻訳単位のソース要素を表示します。
- 複数行を選択した場合，最後に選択した行（点線で囲まれている行）に対するメニューが選択できます。例えば，複数のソース単位または翻訳単位を選択した状態で［入口中断点の設定］を選ぶと，選択したすべての行に中断点が設定されます。この場合，中断点が設定できない属性（実行ファイルなど）の行が含まれているときは，その行には中断点は設定されません。

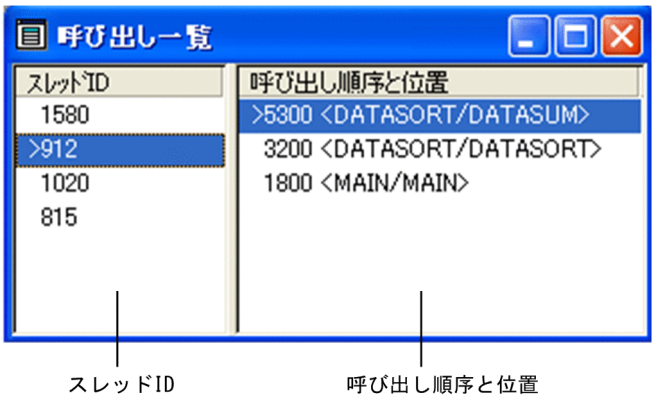
10.3.4　呼び出し一覧ウィンドウ

スレッドの一覧と実行時要素の呼び出し順序を表示します。

呼び出し一覧は，アニメーションを使用しないときはプログラムの実行が中断するごとに，そのときの状態が表示されます。

アニメーションを使用しているとき（アニメーション抑止中も含む）は 1 文が実行されるごとに，そのときの状態が表示されます。

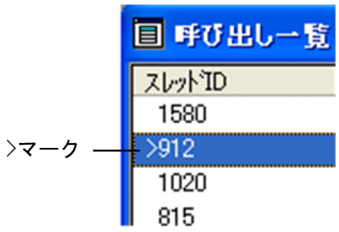
図 10–4　呼び出し一覧ウィンドウ



- スレッド ID

  実行中のスレッドの一覧を表示します。スレッド ID を選ぶと，「呼び出し順序と位置」に，スレッドごとの呼び出し順序を表示します。ただし，マルチスレッドのプログラムでない場合は，スレッドを区別する必要がないため，表示されません。

  実行中のプログラムが中断した直後は中断したスレッドのスレッド ID の左側に「>」マークが表示されます。プログラムの中断中にユーザの操作によって実行時要素が切り替えられた場合，その実行時要素が含まれるスレッド ID の左側に，「>」マークが表示されます。



- 呼び出し順序と位置

  呼び出し順序は次の位置が表示されます。

  ・中断位置の文

  ・CALL 文・INVOKE 文・利用者定義関数でほかのソース要素を呼び出した文

  「スレッド ID」で選択されたスレッド ID の呼び出し順序の一覧を表示します。1 番上が中断位置の文であり，2 番目以降は，一つ前のソース要素を呼び出した CALL 文・INVOKE 文・利用者定義関数が表示されます。操作する対象をクリックで選択します。呼び出し順序のソーステキストを表示できます。

  実行中のプログラムが中断した直後は，中断した実行時要素の左側に，次のように「>」マークが表示されます。プログラムの中断中にユーザにより実行時要素が切り替えられた場合，その切り替えた実行時要素の左側に「>」マークが表示されます。

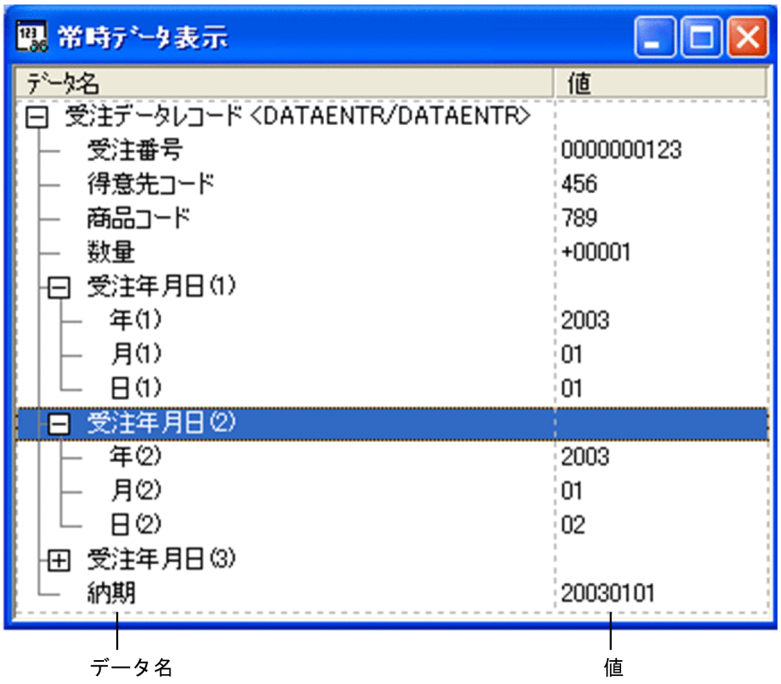
  MAIN プログラム 1,800 行目から，DATASORT プログラムが呼び出され，さらに DATASORT プログラムの 3,200 行目から内部プログラム DATASUM が呼び出され，現在は DATASUM プログラム中の 5,300 行目が実行中であるときの例を次に示します。



10.3.5　常時データ表示ウィンドウ

値を確認する頻度の高いデータ名を常時データ表示ウィンドウに設定することで，プログラムの中断ごとに値を確認できます。最大 25 個のデータ名を設定できます。

図 10-5　常時データ表示ウィンドウ



- データ名

  設定したデータ名にプログラム名が付き，表示されます。下位データのある場合はツリー構造で表示されます。OCCURS 句が指定されているデータ名は，データ名のあとに添字を付けて表示されます。データ名は 128 バイトまでを表示します。選択状態にある項目のデータ名をクリックするか，Tab キーを押すと，データ名の変更ダイアログボックスが表示され，設定中のデータ名の編集ができます。データ名をダブルクリックしても，データ名の変更ダイアログボックスが表示されます。

- 値

  データ名に対応するデータの値が表示されます。データがない場合や二重定義，添字の数不正など，値が参照できないデータの場合は，”???”が表示されます。値表示の対象となるのは基本項目のデータで，指定の表示属性で表示されます。初期表示時はデータ属性で表示されます。ただし，値をデータ属性で表示できないデータは 16 進で表示されます。値は 255 バイトまでを表示します。

注意事項

- 値には 255 バイトまでの値が表示されます。255 バイトを超える値を確認したい場合は，TD コマンドを使用してモニタ画面に表示してください。データの値の表示については，「12.4.8　DISPLAY DATA（データの値表示）」または「12.4.10　DISPLAY OBJECT／DISPLAY FACTORY（オブジェクトのデータ値の表示）」を参照してください。
- プログラムの実行前など，値の取得ができない場合は，値域に”???”が表示されます。
- データ名を登録するとき，値はデータの属性で表示されます。ただし，取得したデータが不当で，値をデータ属性で表示できない場合は，16 進で表示されます。表示属性の設定については，「(4) 表示属性の設定」を参照してください。
- DISPLAY DATA コマンドを使用したときに値に「エラー」と表示されるデータの場合，常時データ表示ウィンドウでは値に「*エラー」と表示されます。

(1)　データ名の設定

値を表示するデータ名をデータ表示一覧ウィンドウに設定します。

データ名を登録すると，アニメーションを使用しないときはプログラムの実行が中断するごとに，そのときの値が表示されます。アニメーションを使用しているとき（アニメーション抑止中も含む）は，1 文が実行されるごとに，そのときの値が表示されます。

1. **ソーステキストウィンドウでデータ名を選ぶ。**

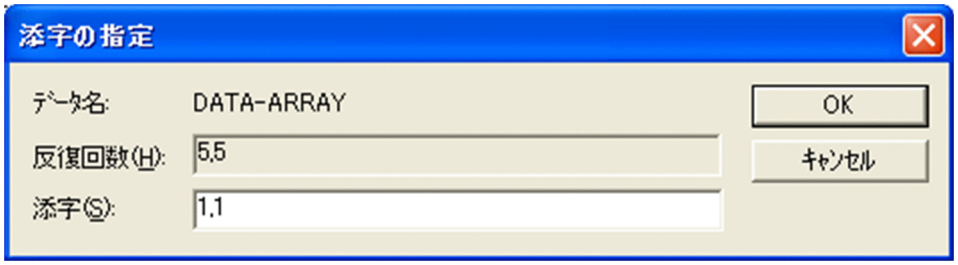
   選んだデータ名が選択状態になります。

2. **［データの操作］メニュー，またはポップアップメニューから［常時データ表示に設定］を選ぶ。**

   常時データ表示ウィンドウにデータ名が設定され，値が表示されます。

   データ名が正しくない場合は，エラーメッセージが表示され，データ名は登録されません。

   データに添字の指定が必要な場合は，添字の指定ダイアログボックスが表示されます。



(a)　添字の設定方法

反復回数には，指定できる添字の最大値が表示されます。

1. 添字を入力する。

   添字の入力領域に，添字を入力します。反復回数を超えない値を，コンマ（,）または空白文字で区切って入力してください。

2. [OK] ボタンを選ぶ。

   添字の指定ダイアログボックスが閉じます。添字の指定方法に誤りがある場合は，エラーメッセージボックスが表示され，ダイアログボックスは閉じません。

(2)　データ名の変更

必要に応じて，登録したデータ名を変更します。

1. 常時データ表示ウィンドウでデータ名を選ぶ。

   選んだデータ名が選択状態になります。

2. ポップアップメニューから［データ名の変更］を選ぶ。

   データ名の変更ダイアログボックスが表示されます。



   現在設定中のデータ名とデータが定義されているプログラム名が表示されます。

   データ名域をクリックしても，データ名の変更ダイアログボックスが表示されます。

3. データ名の変更ダイアログボックスのデータ名を変更し，[OK] ボタンを選ぶ。

   常時データ表示ウィンドウにデータ名が設定され，値が表示されます。存在しないデータ名を入力するなど，データ名の変更処理がエラーになった場合は，エラーメッセージが表示され，変更したデータ名は無効となります。

(3)　データの設定解除

不要となったデータ名の設定を解除します。

1. 常時データ表示ウィンドウでデータ名を選ぶ。
   選んだデータ名が選択状態になります。

2. ポップアップメニューから［データの設定解除］を選ぶ。
   常時データ表示ウィンドウからデータ名の設定が解除されます。
   ［Delete］キーを押して，データ名の設定を解除することもできます。

(4)　表示属性の設定

表示の形式を設定します。データ名の属性と 16 進を指定できます。

1. 常時データ表示ウィンドウで表示属性を設定したいデータ名を選ぶ。
   選んだデータ名が選択状態になります。

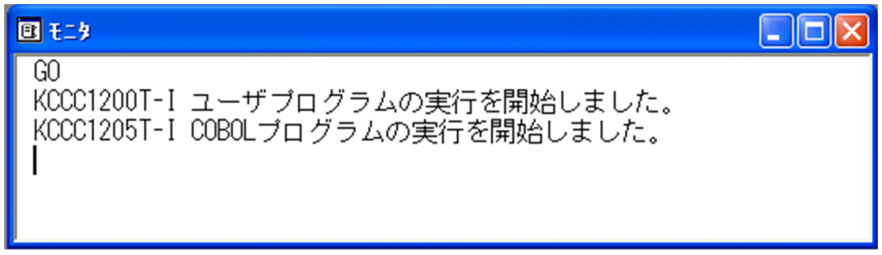
2. ポップアップメニューから［データ属性/16 進］を選ぶ。
   値がデータ属性または 16 進で表示されます。

10.3.6　モニタウィンドウ

次の情報を表示します。TD コマンドによる結果は，すべてモニタウィンドウに表示されます。モニタウィンドウを次に示します。

- プログラムの動作（プログラムの開始・終了・中断など）
- TD コマンドの実行結果
- テストデバッグ環境不正（プログラム情報ファイル（.cbp）の入力エラーなど）

図 10-6　モニタウィンドウ



- モニタウィンドウ

  編集はできません。カーソルは，モニタが活性状態にあるときに存在します。カーソルの操作方法はシステムの基本的な操作方法に従います。モニタ領域全体が選択できる領域です。選択の操作方法はシステムの基本的な操作方法に従います。ダブルクリックでの語の選択はできません。選択した文字列はコピーできます。

注意事項

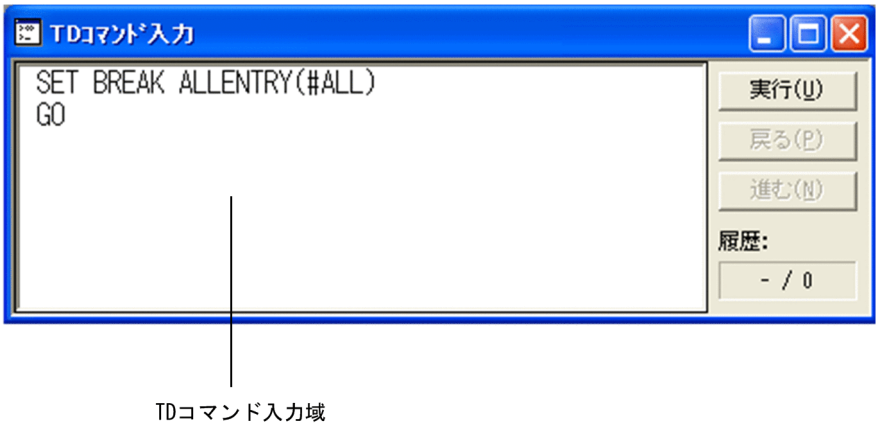
- カーソルの位置が最下行以外にあっても，文字列が新規に表示されるたびに画面はスクロールします。

- 画面が表示されていない場合，文字列が新規に表示されると，画面を表示します。ただし，アイコン化されているときは元のサイズには戻しません。
- 最大 5,000 行が表示されます。5,000 行を超えて表示しようとすると，古い行から削除されます。5,000 行以上の表示内容を見たい場合は，オプション設定か SET LOG コマンドで，モニタウィンドウの内容をファイルに出力させます。
- テストデバッガを終了すると，履歴はクリアされます。

10.3.7　TD コマンド入力ウィンドウ

TD コマンドを入力するウィンドウです。複数の TD コマンドを 1 度に実行できます。また，以前に入力した TD コマンドが保存され，再実行できます。

図 10–7　TD コマンド入力ウィンドウ



- TD コマンド入力域

  TD コマンドを入力します。

- ［実行］ボタン

  TD コマンド入力域の TD コマンドを実行します。Enter キー，または Shift+Enter キーで操作することもできます。キーの切り替えについては，「10.3.9　テストデバッグで使用するウィンドウのカスタマイズ」の「(2)　TD コマンド入力ウィンドウの設定」を参照してください。

- ［戻る］ボタン，［進む］ボタン

  以前に実行した TD コマンドを表示します。

- ［履歴］

  ［戻る］ボタン，［進む］ボタンによって表示された TD コマンド入力域の内容の順序番号と，保存されている TD コマンドの合計数を表示します。

注意事項

- TD コマンド入力域に，タブキーによる入力はできません。

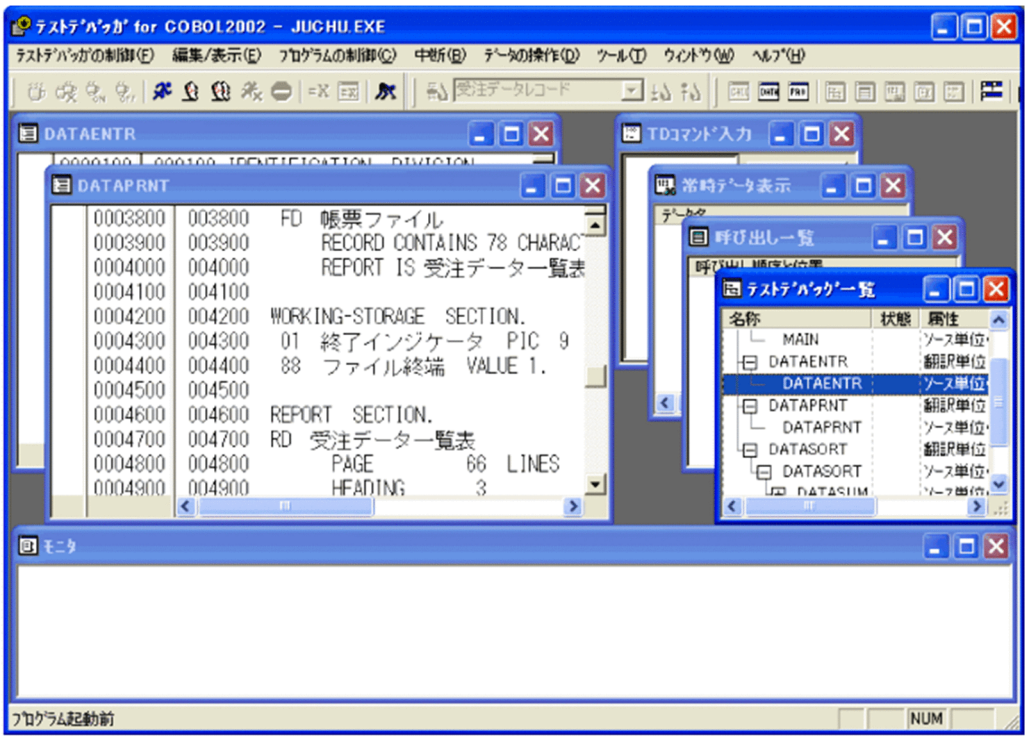
- 履歴は 999 回の TD コマンドの入力を保存します。999 件を超えた場合は，いちばん古い履歴を削除し，最新の 999 回分の入力を，履歴として保存します。

10.3.8　標準配置

現在表示中のウィンドウを，デフォルト位置へ配置します。ただし，非表示中のウィンドウは，表示しません。

- モニタウィンドウは画面下側に配置します。
- ソーステキスト，モニタウィンドウを除くウィンドウは画面右上に重ねて表示します。
- ソーステキストウィンドウは画面左上に重ねて表示します。

図 10-8　テストデバッグウィンドウの標準配置



注意事項

重ねて表示されるウィンドウは，直前までの操作で操作された順に表示されます。つまり，最後に操作された画面が最前面に表示されます。

10.3.9　テストデバッグで使用するウィンドウのカスタマイズ

画面の表示，ツールバーの表示を変更できます。

表示の変更ができる画面はソーステキストウィンドウ，モニタウィンドウ，TD コマンド入力ウィンドウです。

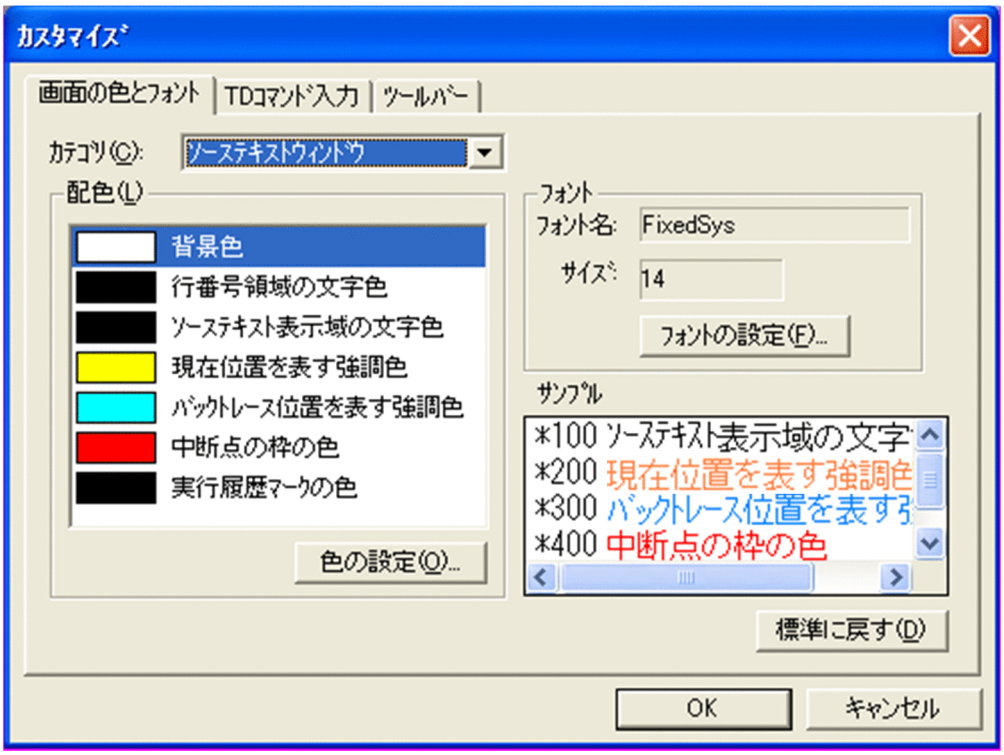
1. ［ツール］メニューから［カスタマイズ］を選ぶ。
   カスタマイズダイアログボックスが表示されます。
   ソーステキストウィンドウ，モニタウィンドウ，TD コマンドウィンドウ上，またはポップアップメニューから［カスタマイズ］を選んでも，カスタマイズダイアログボックスが表示されます。

2. 設定したい項目のタブを選び，表示されるページで設定を変更し，［OK］ボタンを選ぶ。
   ダイアログボックスでの設定が反映されます。

次に，このダイアログボックスで設定する内容について説明します。

(1)　画面表示の設定

画面の表示は，［画面の色とフォント］タブを選んで表示されるページで設定します。



- カテゴリ
  色とフォントを設定するウィンドウ名を選びます。
- 配色
  色の設定ができる項目と現在の配色が表示されます。各項目の色が変更できます。
  色の設定ができる項目と現在の配色が表示されます。
  色の設定ができる項目とデフォルト色を次に示します。

表 10–2　画面のデフォルト色

項目	項目	デフォルト色
ソーステキストウィンドウ	背景色	システム

項目	項目	デフォルト色
	行番号領域の文字色	システム
	ソーステキスト表示域の文字色	システム
	現在位置を表す強調色	黄
	バックトレース位置を表す強調色	シアン
	中断点の枠の色	赤
	実行履歴マークの色	システム
モニタウィンドウ	背景色	システム
	文字	システム
TD コマンド入力ウィンドウ	背景色	システム
	文字	システム

- フォント

  画面で使用している現在の文字フォントが表示されます。文字フォントが変更できます。デフォルトの設定は「FixedSys」，サイズ「14」です。

- サンプル

  現在設定されているフォント，配色での表示例が表示されます。

(a) 配色の設定方法

1. カテゴリから，色を設定したいウィンドウ名を選ぶ。

   ドロップダウンリストで選択したウィンドウ名が表示されます。

2. 配色から，色を設定したい項目を選ぶ。

   選んだ項目が選択状態になります。複数の項目を選ぶことができます。

3. [色の設定] ボタンを選ぶ。

   色の設定ダイアログボックスが表示されます。

4. 色の設定ダイアログボックスで色を選択して，[OK] ボタンを選ぶ。

   なお，任意の色を指定するには，色の設定ダイアログボックスの［色の作成］ボタンを選び，拡張されたダイアログボックスで色を作成します。

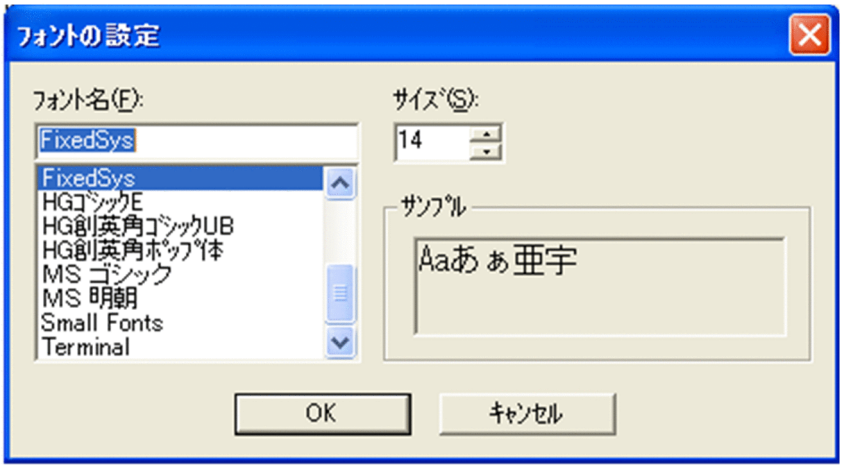
(b) フォントの設定方法

1. カテゴリから，フォントを設定したいウィンドウ名を選ぶ。

   ドロップダウンリストで選択したウィンドウ名が表示されます。

2. [フォントの設定] ボタンを選ぶ。

   フォントの設定ダイアログボックスが表示されます。



3. フォントの設定ダイアログボックスでフォント名とサイズを選択して，［OK］ボタンを選ぶ。
   フォントが設定されます。

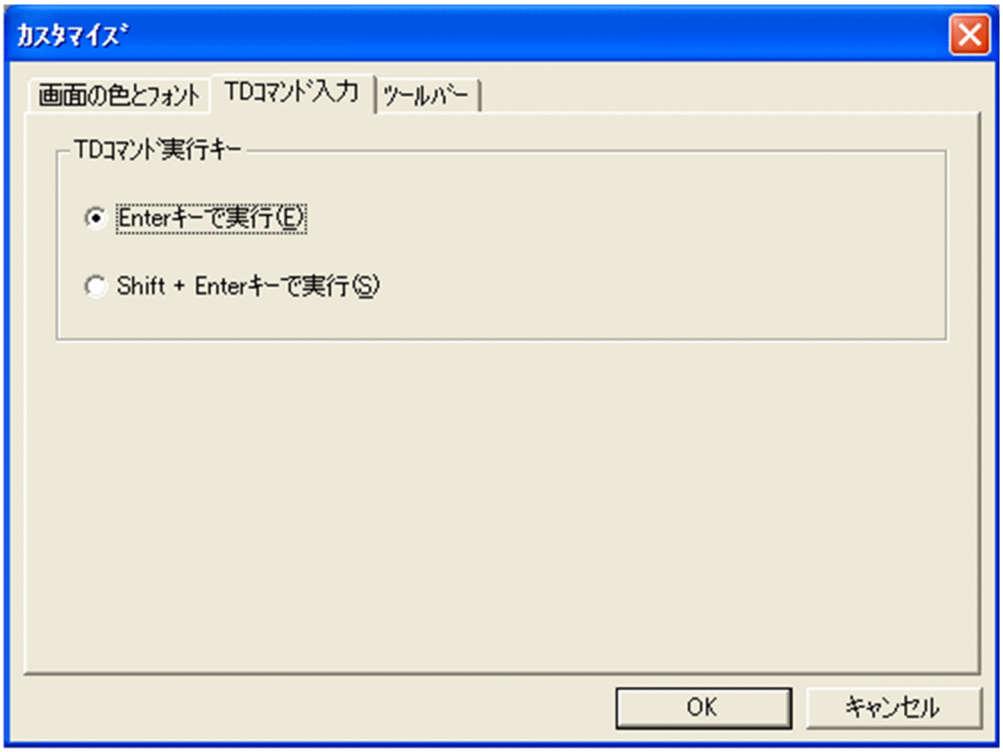
(c)　色とフォントの初期化

1. ［標準に戻す］ボタンを選び，［OK］ボタンを選ぶ。
   設定した色とフォントが初期化されます。

(2)　TD コマンド入力ウィンドウの設定

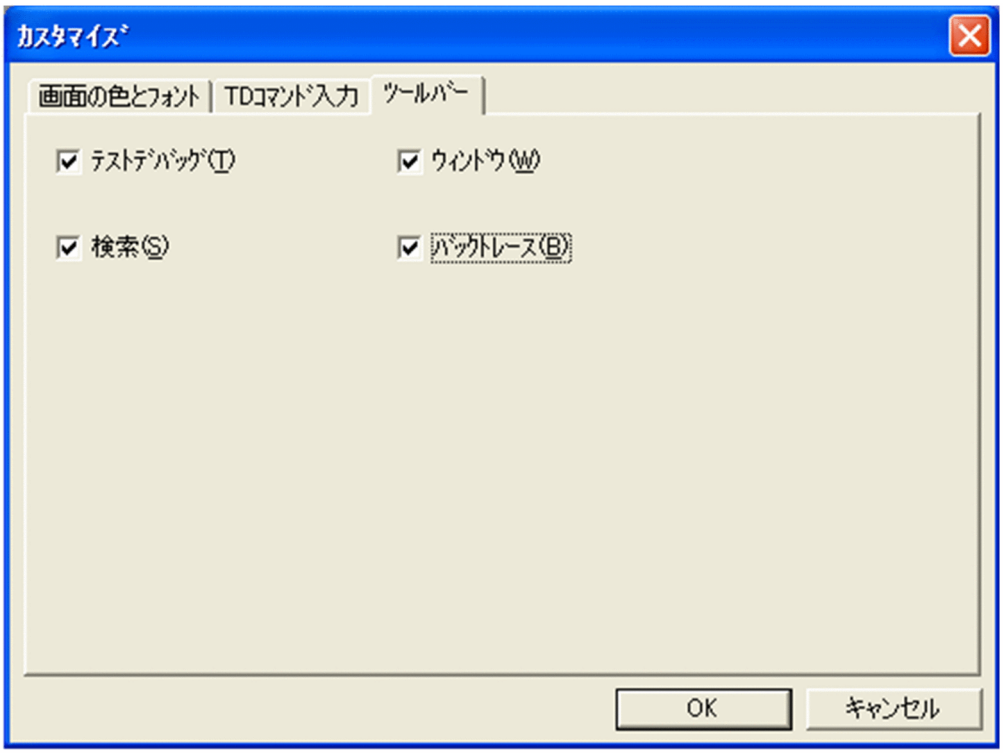
TD コマンド入力ウィンドウのキー操作を設定します。

「Enter キーで実行」と，「Shift ＋ Enter キーで実行」を選択できます。



(3)　ツールバーの設定

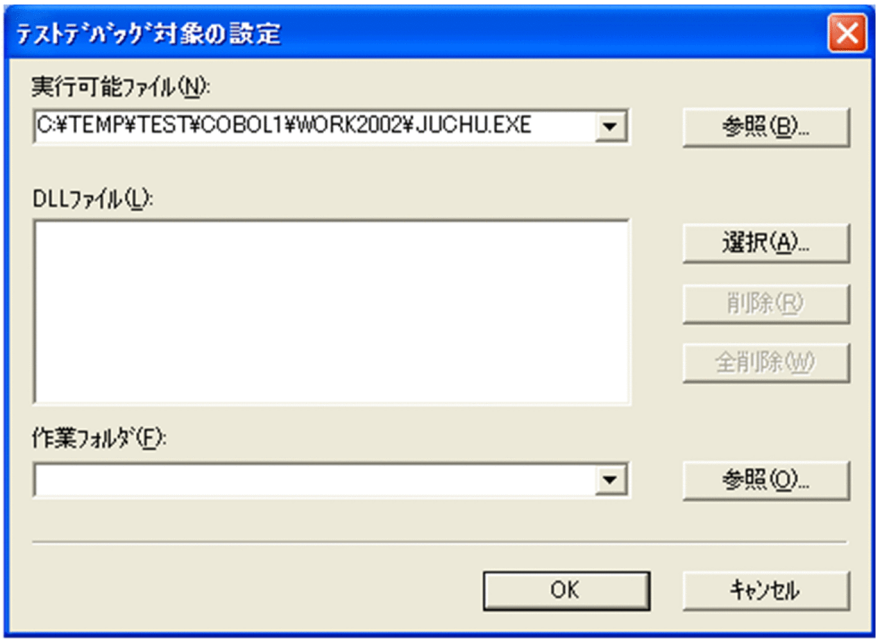
ツールバーの表示および非表示を設定します。



10.4　テストデバッグ対象とする実行可能ファイルと DLL ファイルの変更

テストデバッグの対象とする実行可能ファイルと DLL ファイルを変更します。

1. ［テストデバッガの制御］メニューから［テストデバッグ対象の設定］を選ぶ。
   テストデバッグ対象のプログラムを設定するダイアログボックスに，現在設定中の内容が表示されます。



操作方法については，「10.2.1　テストデバッガの起動」「(1)　開発マネージャからの開始」－「(b)　テストデバッガの起動」を参照してください

注意事項

- プログラムが実行中の場合については，「テストデバッグ対象プログラムの実行を終了します。」というメッセージが表示されます。［OK］ボタンを選択するとプログラムの実行が終了したあと，テストデバッグ対象の設定ダイアログボックスが表示されます。［キャンセル］ボタンを選択すると，元の状態に戻ります。
- テストデバッガの実行に関する環境変数を変更する場合は，テストデバッガを終了させ，環境変数を変更したあと，再度テストデバッガを起動します。
- このダイアログボックスでテストデバッグ対象を変更した場合，変更前の設定状態は初期化されます。設定状態の初期化については，「9.2.9　テストデバッグのための設定」の「(2)　テストデバッガの初期化」を参照してください。モニタウィンドウの表示は変更前と変わらないので，必要がない場合はモニタウィンドウのポップアップメニューから［クリア］を選び，モニタウィンドウをクリアします。

10.5　テストデバッグ機能の初期化

テストデバッガで設定した，中断点，データ監視条件，トレース表示などを解除できます。解除できる機能については，「9.2.9　テストデバッグのための設定」の「(2)　テストデバッガの初期化」を参照してください。

1. [テストデバッガの制御] メニューから [初期化] を選ぶ。

   確認のためのダイアログボックスが表示されます。



2. [OK] ボタンを選ぶ。

   テストデバッグの機能が解除されます。また，テストデバッグ対象プログラムの中断時に初期化を行うとプログラムは終了します。

10.6　TD コマンド格納ファイルの入力

中断点，シミュレーションなどを TD コマンド形式でファイルに登録しておき実行します。TD コマンド格納ファイルに格納する TD コマンドの詳細については，「12.4　TD コマンドの詳細」を参照してください。

1. ［テストデバッガの制御］メニューから［TD コマンド格納ファイルの入力］を選ぶ。
   ファイル選択ダイアログボックスが表示されます。

2. 実行する TD コマンドが登録されているファイルを選び，［開く］ボタンを選ぶ。
   登録されている TD コマンドが実行されます。TD コマンドの実行結果についてはモニタウィンドウに表示されます。

10.7　ソーステキストの表示

テストデバッグ対象プログラムの起動前や中断中に，任意のソーステキストを表示できます。ソーステキストウィンドウは分割バーを境に上下で分割して表示でき，同じソーステキストの異なる個所を同時に表示できます。

また，ソーステキストの手続き部に書かれたデータ名からそのデータ名を選ぶと，そのデータが定義されている部分のソーステキストを表示できます。

ここでは，ソーステキストを表示する方法と，現在開いているソーステキストウィンドウをすべて閉じる方法について説明します。

10.7.1　テストデバッグ一覧ウィンドウからの表示

(1)　翻訳単位の先頭を表示する

1. テストデバッグ一覧ウィンドウから，ソーステキストを表示する翻訳単位を選ぶ。
   翻訳単位が選択状態になります。

2. [ウィンドウ] メニューから［ソーステキストウィンドウ］を選び，サブメニューから［定義部の表示］を選ぶ。または，ポップアップメニューから［定義部の表示］を選ぶ。
   ソーステキストウィンドウに選択した翻訳単位のソーステキストが先頭行から表示されます。

(2)　定義部の先頭を表示する

1. テストデバッグ一覧ウィンドウから，ソーステキストを表示するソース単位を選ぶ。
   ソース単位が選択状態になります。

2. [ウィンドウ] メニューから［ソーステキストウィンドウ］を選び，サブメニューから［定義部の表示］を選ぶ。またはポップアップメニューから［定義部の表示］を選ぶ。
   ソーステキストウィンドウに選択したソース単位のソーステキストが定義部の先頭行から表示されます。

(3)　実行部の先頭を表示する

1. テストデバッグ一覧ウィンドウから，ソーステキストを表示するソース単位を選ぶ。
   ソース単位が選択状態になります。

2. [ウィンドウ] メニューから［ソーステキストウィンドウ］を選び，サブメニューから［実行部の表示］を選ぶ。またはポップアップメニューから［実行部の表示］を選ぶ。
   ソーステキストウィンドウに選択したソース単位のソーステキストが実行部の先頭行から表示されます。

注意事項

- 定義部のない翻訳単位を選択した場合，［定義部の表示］は選択できません。
- 定義部のないソース単位を選択した場合，［定義部の表示］は選択できません。
- 実行部のないソース単位を選択した場合，［実行部の表示］は選択できません。

10.7.2　呼び出し一覧ウィンドウからの表示

呼び出し一覧ウィンドウ上の呼び出し文をソーステキストウィンドウに表示します。

1. 呼び出し一覧ウィンドウからソーステキストを表示する呼び出し位置を選ぶ。
   選択した呼び出し位置が選択状態になります。
2. ［ウィンドウ］メニューから［ソーステキストウィンドウ］を選び，サブメニューから［呼び出し位置の表示］を選ぶ。またはポップアップメニューから［呼び出し位置の表示］を選ぶ。
   ソーステキストウィンドウに，選択した呼び出し文の行が表示されます。

10.7.3　ソーステキストウィンドウからの表示

ソーステキストウィンドウ上の任意のデータ名を選んで，そのデータが定義されている行を表示します。

1. ソーステキストウィンドウから実行部のデータ名を選ぶ。
   データ名が選択状態になります。
2. ポップアップメニューから［定義データ行の表示］を選ぶ。
   ソーステキストウィンドウに選択したデータの定義されている行が表示されます。

10.7.4　登録集原文の表示／非表示

ソーステキストウィンドウ上に登録集原文を表示するかどうかを指定できます。

1. ［ツール］メニューから［オプション］を選ぶ。
   オプションダイアログボックスが表示されます。
   オプションダイアログボックスについては，「10.18　テストデバッグのオプション設定」を参照してください。
2. ［ソーステキスト］タブを選び，「登録集原文を表示する」のチェックボックスをオンにする。
   登録集原文を表示しない場合は，チェックボックスをオフにします。
3. ［OK］ボタンを選ぶ。

設定した内容が有効になります。

登録集原文の非表示時の登録集原文内へ対する，各機能の動作の詳細を次に示します。

表 10–3　登録集原文非表示時の動作

機能	表示
中断点の設定（TD コマンド）	中断点は設定されます（登録集原文を展開すれば，設定した文は中断点を表す枠で囲まれています）。
現在位置の表示	該当登録集原文を定義している「COPY」文を表示します。
中断位置の表示（中断点） 中断位置の表示（データ条件中断が成立） 中断位置の表示（実行時エラーが発生）	該当登録集原文を定義している「COPY」文を表示します（表示中に登録集原文を非展開から展開に切り替えられた場合は，COPY 文から該当文に，展開から非展開に切り替えられた場合は，該当文を COPY 文に表示を切り替えます）。
中断位置の表示（ステップイン，ステップオーバー実行後） アニメーション	登録集原文内の文は対象としません。登録集原文以降に最初に表示できる文に位置づきます。 実行履歴は表示されません。
指定行に位置づけて表示 文字列の検索	登録集原文内は対象となりません。
バックトレースの取得	取得： 登録集原文内の文も対象とします。 表示： 「COPY 文」を表示します（表示中に登録集原文の非展開から展開に切り替えられた場合は，COPY 文を該当文に，展開から非展開に切り替えられた場合は，該当文を COPY 文に表示を切り替えます）。
トレース フロー シミュレーション	登録集原文内の文も対象とします。

10.7.5　中断時のソーステキストの表示

テストデバッグ対象プログラムが中断したときに中断した文がソーステキストウィンドウに表示されます。中断した文は，背景色が通常と違った色で表示されます。ソーステキストウィンドウが表示されていない場合は，ソーステキストウィンドウを新たに開いて中断文を表示します。

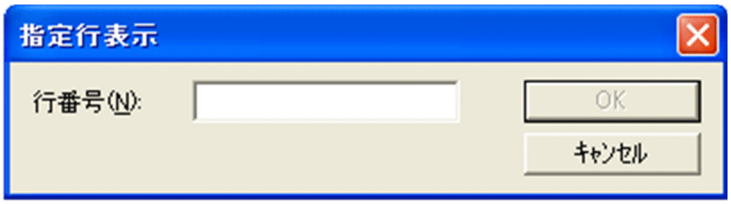
注意事項

- 登録集原文が非表示の場合，登録集原文内で中断したときは，COPY 文が中断文となります。
- 実行部の入口で中断している場合は，PROCEDURE または ENTRY 文が表示されます。
  実行部の出口で中断している場合は，END PROGRAM，END METHOD，または END FUNCTION の「END」が強調表示されます。ただし，END PROGRAM が省略された場合，最下行の空白行 1 行が強調表示されます。

10.7.6　指定行の表示

ソーステキストウィンドウの行番号領域に表示されている行番号を指定してソーステキストを表示します。

1. 指定行を表示したいソーステキストウィンドウにフォーカスを位置づける。
2. ［編集/表示］メニューから［指定行の表示］を選ぶ。
   指定行表示ダイアログボックスが表示されます。



3. 任意の行番号を入力する。
4. ダイアログボックスの［OK］ボタンを選ぶ。
   指定した行が画面の 1 行目に表示され，行番号の左端に”＞”マークが付けられます。表示されている画面内に指定の行がある場合は，指定行の行番号の左側に”＞”マークが付くだけとなります。

注意事項

- ソーステキストウィンドウにフォーカスを位置づけていない場合，［指定行の表示］は選択できません。
- 指定した行番号がない場合は，行番号に最も近く，行番号より小さい数の行番号のソーステキストが表示されます。ただし，先頭行よりも小さい行番号を指定したときは，先頭の行が表示されます。
- ソーステキストウィンドウに行番号領域が表示されていなくても，表示の対象となる行番号は，行番号領域に表示される行番号です。

10.7.7　すべてのソーステキストウィンドウを閉じる

［ウィンドウ］メニューから［すべて閉じる］を選ぶと，現在開いているソーステキストウィンドウをすべて閉じます。ただし，設定してある中断点が解除されたり，実行履歴がクリアされたりすることはありません。

10.8　中断点の設定と解除

実行文に対して中断点を設定，解除する方法について説明します。

10.8.1　中断点の設定

中断点はソーステキストウィンドウとテストデバッグ一覧ウィンドウから設定できます。中断点の設定方法について説明します。

(1)　ソーステキストウィンドウからの設定

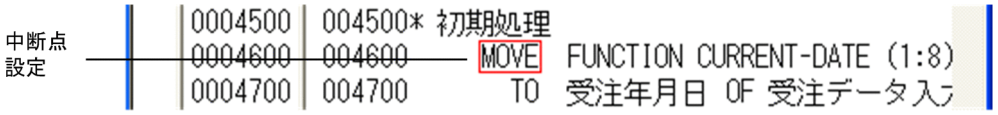
ソーステキストウィンドウ中の文に対して中断点を設定します。ソーステキストウィンドウからの中断点の設定には，次の 3 とおりの方法があります。

(a)　中断点を設定する文の文キーワードを選択，または文キーワード上にカーソルを位置づける方法

1. 中断点を設定する文（節・段落名を含む）の文キーワードを選択する。または，文キーワード上にカーソルを位置づける。

2. ［中断］メニューから［中断点の設定/解除］を選ぶ。またはポップアップメニューから［中断点の設定/解除］を選ぶ。

   カーソルを位置づけた文に対して中断点が設定され，文キーワードがカスタマイズダイアログボックスで設定した色で強調表示されます。カスタマイズダイアログボックスについては，「10.18　テストデバッグのオプション設定」を参照してください。



(b)　中断点を設定する文の文キーワード上をダブルクリックする方法

1. 中断点を設定する文（節・段落名を含む）の文キーワード上をダブルクリックする。

   ダブルクリックした文に対して中断点が設定され，文キーワードが強調表示されます。

(c)　中断点を設定する文がある行の作業領域をクリックする方法

1. 中断点を設定する文（節・段落名を含む）がある行の作業領域をクリックする。

   クリックした行にある文に対して中断点が設定され，文キーワードが強調表示されます。

   ただし，1 行に複数の文がある場合は，行内の先頭の文キーワードに対して中断点が設定されます。

注意事項

ソーステキストウィンドウの作業領域からの設定では，文キーワードを含まない行の上での操作は無効となります。

(2) テストデバッグ一覧ウィンドウからの設定

テストデバッグ一覧ウィンドウ中のソース単位に対して入口または出口中断点を設定します。翻訳単位などに対して入口または出口中断点を設定した場合には，下位の全項目に中断点が設定されます。

1. 入口，または出口中断点を設定するソース単位または翻訳単位を選ぶ。

   選択したソース単位または翻訳単位が選択状態になります。

2. ポップアップメニューから，［入口中断点の設定］または［出口中断点の設定］を選ぶ。

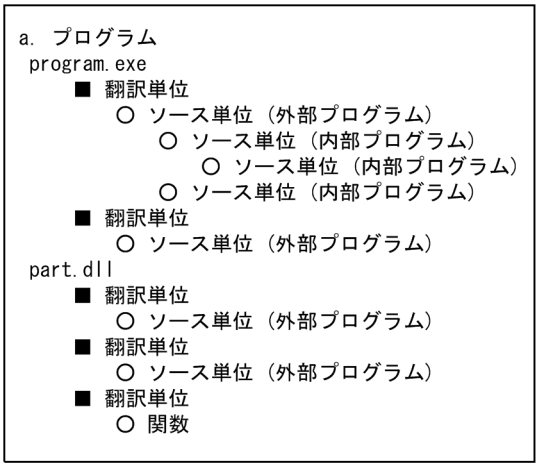
   選択したソース単位の状態欄に「*」マークが表示されます。

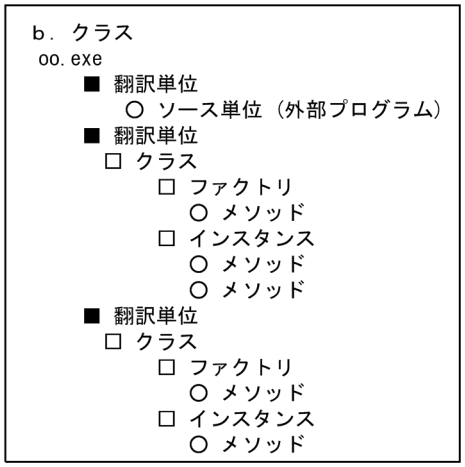
   翻訳単位などを選択した場合は，下位の全ソース単位の状態欄に「*」マークが表示されます。

注意事項

入口または出口中断点設定の対象範囲は次のとおりです。

○マークの項目に対して入口または出口中断点を設定すると，○マークの項目自体に中断点が設定されます。■マークと□マークの項目に対して入口または出口中断点を設定すると，下位のすべての項目が中断点設定の対象となります。





10.8.2 中断点の解除

ソーステキストウィンドウとテストデバッグ一覧ウィンドウから中断点を解除する方法について説明します。

(1) ソーステキストウィンドウからの解除

文に対して設定されている中断点を解除します。

ソーステキストウィンドウからの解除には，次の 3 とおりの方法があります。

(a) 中断点を解除する文の文キーワードを選択，または文キーワード上にカーソルを位置づける方法

1. 中断点を解除する文（節・段落名を含む）の文キーワードを選択する。または，文キーワード上にカーソルを位置づける。

2. [中断] メニューから [中断点の設定/解除] メニュー，またはポップアップメニューから [中断点の設定/解除] を選ぶ。

   カーソルを位置づけた文に対して中断点が解除され，文キーワードの強調表示が解除されます。

(b) 中断点を解除する文の文キーワード上をダブルクリックする方法

1. 中断点を解除する文（節・段落名を含む）の文キーワード上をダブルクリックする。

   ダブルクリックした文に対して中断点が解除され，文キーワードの強調表示が解除されます。

(c) 中断点が設定されている行の作業領域をクリックする方法

1. 中断点が設定されている行の作業領域をクリックする。

   クリックした行に設定されている中断点が解除され，文キーワードの強調表示が解除されます。

   ただし，1 行に複数の文がある場合は，行内の先頭の文キーワードに対して解除されます。

(2) テストデバッグ一覧ウィンドウからの解除

ソース単位または翻訳単位の，入口または出口に対して設定されている中断点を解除します。

1. 入口または出口中断点を解除する，ソース単位または翻訳単位を選ぶ。

   選択したソース単位または翻訳単位が選択状態になります。

2. ポップアップメニューから，[入口中断点の解除] または [出口中断点の解除] を選ぶ。

   選択したソース単位の状態欄に表示されている「*」マークが非表示になります。翻訳単位を選択した場合は，その翻訳単位に含まれるソース単位の状態欄に表示されている「*」マークが非表示になります。

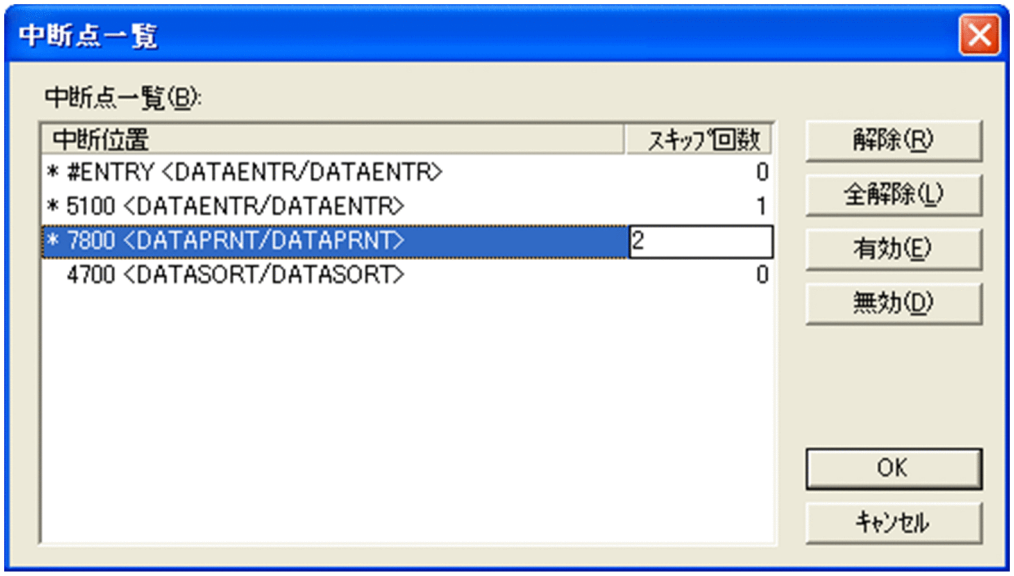
10.8.3 中断点一覧の表示

設定されている中断点の一覧を表示します。また，この一覧から既存の中断点を有効化，無効化，または解除できます。ここでは，中断点一覧の表示方法と中断点の解除，無効化の方法について説明します。

(1) 一覧の表示

1. [中断] メニューから [中断点の一覧] を選ぶ。

中断点一覧ダイアログボックスが表示されます。ここでは，画面上で中断点を設定したものだけでなく，TD コマンドで設定した情報も反映されます。



- 中断点一覧

  中断位置とスキップ回数が表示されます。中断点が有効な場合は，中断位置の左端に「*」マークが表示されます。処理が繰り返し実行されるような個所に中断点を設定する場合は，スキップ回数を設定し，中断点で止まらずに通過させる回数を設定できます。不要な中断回数を減らすことで，効率良くプログラムをテストできます。

  スキップ回数の設定方法について説明します。

  1. 設定したい中断点のスキップ回数を選ぶ。

     入力域が表示されます。

  2. スキップ回数を入力します。

     [Enter] キーを押すか，ダイアログボックス内のフォーカスが別のところへ移動した場合に入力が確定します。

     入力した値が不正な場合はエラーメッセージが表示され，入力した値は無効となります。

- 解除

  中断点一覧から中断点を選び［解除］ボタンを選ぶと，選んだ中断点が解除されます。

- 全解除

  中断点一覧に表示されているすべての中断点を解除します。

- 有効

  無効になっている中断点を中断点一覧から選び，［有効］ボタンを選ぶと，選んだ中断点が有効化します。中断点一覧の中断位置の左端にアスタリスク（*）が表示されます。

- 無効

  有効になっている中断点を中断点一覧から選び，［無効］ボタンを選ぶと，選んだ中断点が，一時的に中断しないように無効化されます。中断点一覧の中断位置の左端にあるアスタリスク（*）が消えます。

(2) 中断点の解除

1. ダイアログボックスの一覧の中から解除したい中断点を選ぶ。

   選択した中断点が選択状態になります。複数の中断点を選ぶこともできます。

2. [解除] ボタンを選ぶ。

   選択した中断点が中断点一覧から消去されます。中断点を複数選んだ場合，一度に複数の中断点を解除できます。[Delete] キーを押しても，選択した中断点が一覧から消去され，中断点を解除できます。

   全中断点を解除する場合は，[全解除] ボタンを選びます。

3. [OK] ボタンを選ぶ。

   設定した内容が有効になります。

   表示中のソーステキストウィンドウの強調表示が解除されます。また，入口や出口中断点が解除された場合は，テストデバッグ一覧ウィンドウの状態欄の表示が非表示になります。

(3) 中断点の無効化

中断点を設定したまま，一時的に中断しないように無効化できます。

1. ダイアログボックスの一覧の中から無効にしたい中断点を選ぶ。

   選択した中断点が選択状態になります。複数の中断点を選ぶこともできます。

2. [無効] ボタンを選ぶ。

   選択した中断点の左端の「*」マークが非表示になります。中断点を複数選んだ場合，一度に複数の中断点を無効にできます。

   無効にした中断点を有効にする場合は，有効にしたい中断点を選び，[有効] ボタンを選ぶと，中断位置の左端に「*」マークが表示されます。

3. [OK] ボタンを選ぶ。

   設定した内容が有効になります。

   表示中のソーステキストウィンドウの強調表示が矩形表示となります。また，入口や出口中断点が無効になった場合は，テストデバッグ一覧ウィンドウの状態欄の表示が「*」マークから「−」マークになります。

10.8.4 中断点の全解除

中断点の設定をすべて解除します。

1. [中断] メニューから [すべての中断点の解除] を選ぶ。

   すべての中断点が解除され，ソーステキストウィンドウ上の強調表示とテストデバッグ一覧ウィンドウ上の中断点設定欄の表示が解除されます。

10.9　データ監視条件の設定と解除

データ監視条件を設定すると，条件が成立した時点でプログラムの実行が中断されます。データ名または領域に対してデータ監視条件を指定できます。

10.9.1　データ監視条件の設定

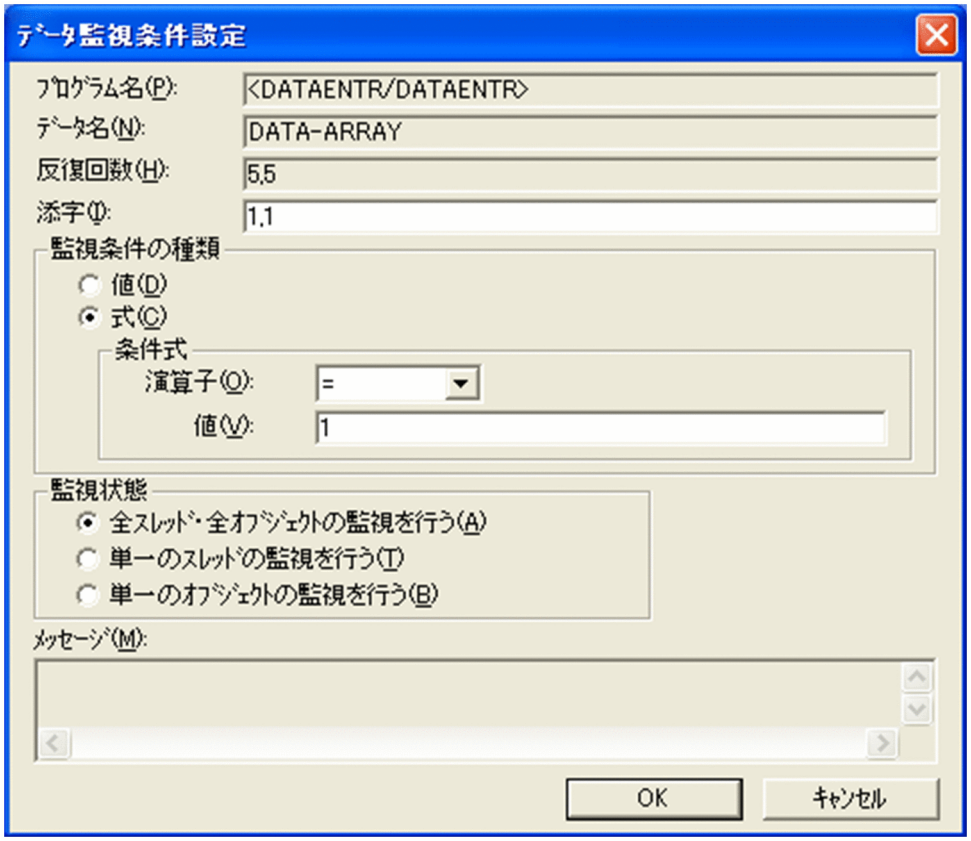
データ監視条件には 2 とおりあり，データ名に対して式か値で設定します。式で設定すると条件式が成立した時点で中断し，値で設定するとデータ名の値が変化した時点で無条件に中断します。データ監視条件の設定方法を説明します。

1. ソーステキストウィンドウからデータ監視条件を設定するデータ名を選ぶ

   選択されたデータ名が選択状態になります。

2. [データの操作] メニュー，またはポップアップメニューから［データ監視条件の設定］を選ぶ

   データ監視条件設定ダイアログボックスが表示されます。



   なお，データ監視条件設定ダイアログボックスで設定の解除はできません。解除する場合は，データ監視条件一覧ダイアログボックス，または TD コマンドで行います。

3. データ監視条件一覧ダイアログボックス内の必要な項目を設定する

4. [OK] ボタンを選ぶ。

設定した内容によりデータ監視を設定します。設定にエラーがあった場合は，「メッセージ」にエラーの詳細が表示され，ダイアログボックスは閉じません。

(1) データ監視条件一覧ダイアログボックスで設定する内容

プログラム名

ソーステキストウィンドウで選択されたデータが定義されているプログラム名が表示されます。プログラム名は変更できません。

データ名

条件中断を設定したいデータ名を設定します。

ダイアログボックス表示時は，ソーステキストウィンドウで選択されたデータ名が表示されます。データ名の変更はできません。

選んだデータ名が表要素の場合は，添字付きで指定します。添字の指定方法については，「12.2.12 ソース要素指定」を参照してください。

反復回数

ソーステキストウィンドウで選択されたデータが表要素の場合，添字の反復回数の最大値が表示されます。

反復回数の変更はできません。

添字

ソーステキストウィンドウで選択されたデータが表要素の場合，添字を入力できます。表要素ではない場合は入力できません。

監視条件の種類

監視条件の種類で，値または式を指定します。

- 値の設定

  値を選択した場合は，条件式は設定できません。

- 式の設定

  式を選択した場合は，条件式が設定できます。

条件式の設定

中断の条件となる演算子と値を指定します。

選択できる演算子は「>」，「<」，「>=」，「<=」，「=」，「NOT=」です。

- 値に定数を入力する場合

  定数を入力する場合の書式は，「12.2.7　定数」を参照してください。

- 値にデータ名を入力する場合

  ・データ名だけ入力した場合は，ダイアログボックス上部のプログラム名領域に表示されているプログラム内のデータ名として扱います。

  ・別のプログラムのデータ名を指定したい場合は，データ名をプログラム名で修飾します。プログラム名による修飾の詳細については，「12.2.12　ソース要素指定」を参照してください。

監視状態

監視の実行状態を設定します。

- 全スレッド・全オブジェクトの監視を行う

  全スレッド・全オブジェクトのプログラム名・データ名に該当するデータの監視をします。単一スレッド，オブジェクトの監視を行うとき以外は，マルチスレッドプログラム，オブジェクト指向機能を使用した COBOL プログラムも含み，どのプログラムでも，この設定で監視をします。

- 単一のスレッドの監視を行う

  単一のスレッドのデータを監視します。プログラムの起動前に指定すると，プログラムの開始後に最初のスレッドのデータを監視します。中断点で指定すると，中断点のスレッドのデータを監視します。マルチスレッドプログラムのテストデバッグをしていて，すべてのスレッドのデータを監視したくない場合に選択します。

- 単一のオブジェクトの監視を行う

  単一のオブジェクトのデータを監視します。プログラムの起動前に指定すると，最初に参照できたオブジェクトのデータを監視します。中断点で指定すると，中断位置から最初に参照できたオブジェクトのデータを監視します。オブジェクト指向機能を使用した COBOL プログラムのテストデバッグをしていて，すべてのオブジェクトのデータを監視したくない場合に選択します。

メッセージ

［OK］ボタンを選んだ場合，エラーが発生したときに，メッセージの詳細が表示されます。エラーが発生したとき，ダイアログボックスは閉じません。

(2) 注意事項

- ［OK］ボタンを選んだ場合，エラーが発生したときは，メッセージ欄にメッセージを出力し，［OK］ボタンを選ぶ前の状態に戻ります。
- ［OK］ボタンを選んで警告が表示された場合，警告を表示したダイアログボックスで監視を設定するかどうかをユーザに問い合わせ，ユーザが［OK］を選んだときは監視を設定します。ユーザが［キャンセル］を選んだときは，［OK］ボタンを選ぶ前の状態に戻ります。ここでいう警告とは，監視条件式の値設定で，データ名長が最大長より長い場合に，制限長だけ有効にしても監視の設定ができる場合などのことです。
- データ監視条件設定ダイアログボックスでデータ監視条件を設定すると，先頭が＃で，2 カラム以降が 7 文字の数字の監視識別子が設定されます。
- このダイアログボックスで設定したデータ監視条件は，個別には TD コマンドでは解除できません。解除する場合は，データ監視条件一覧ダイアログボックスで行います。ただし，全解除は TD コマンドからできます。

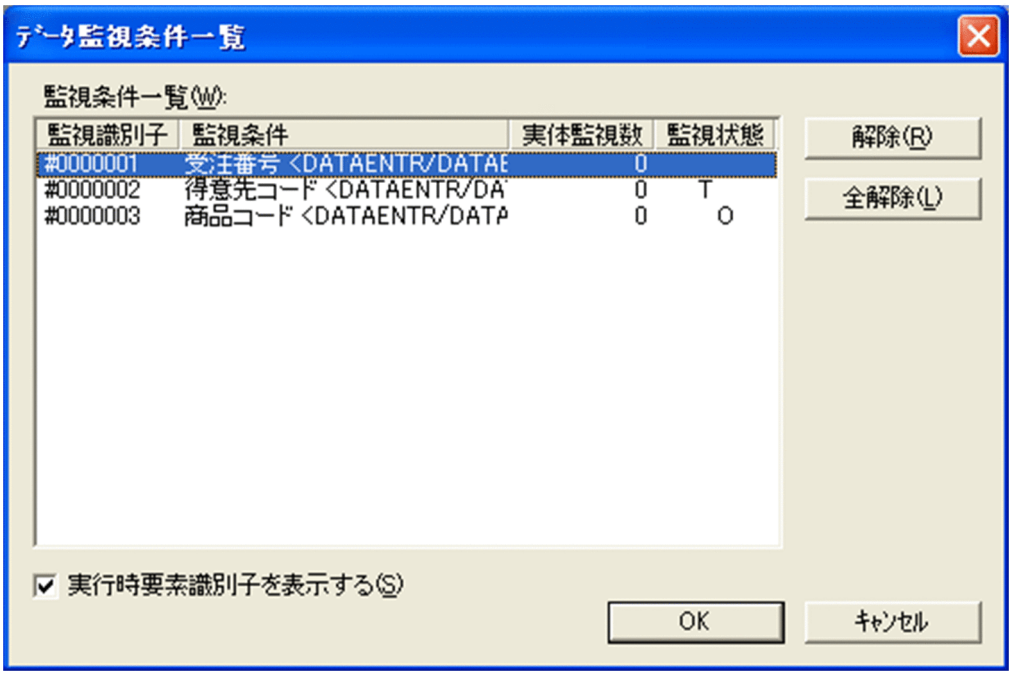
10.9.2　データ監視条件一覧の表示と解除

設定されているデータ監視条件の一覧を表示します。一覧からデータ監視条件を解除できます。データ監視条件一覧の表示方法とデータ監視条件の解除の操作を説明します。

(1)　一覧の表示

1. [データの操作] メニューから［データ監視条件の一覧］を選ぶ。

   データ監視条件一覧ダイアログボックスが表示されます。



(a)　データ監視条件一覧ダイアログボックスで設定する内容

監視条件一覧

- 監視識別子

  監視条件情報の識別子を表示します。

- 監視条件

  データ監視条件を次の形式で表示します。ただし，「実行時要素識別子を表示する」が選択されていない場合は，実行時要素識別子は表示しません。

「値」の指定による監視

　データ名 < 実行時要素識別子 >

「条件式」の指定による監視

- 右辺が定数の場合

  データ名 < 実行時要素識別子 > 演算子 定数

- 右辺がデータ名の場合

データ名 < 実行時要素識別子 > 演算子 データ名 < 実行時要素識別子 >

左辺と右辺のデータ名の実行時要素が等しい場合は，右辺の実行時要素は表示されません。

- 実体監視数

  監視対象のデータが複数存在する場合，中断位置での指定されたデータの存在する数を表示します。データが複数存在するとは次のような場合があります。

  ・マルチスレッドプログラムで複数のスレッドが実行している。

  ・再帰プログラムが再帰的に呼び出されているとき，そのプログラムの局所場所節に定義されているデータ。

  ・複数のインスタンスが生成されているインスタンスオブジェクトに定義されているデータ。

- 監視状態

  単一のスレッドの監視を行う場合「T」が表示されます。

  単一のオブジェクトの監視を行う場合「O」が表示されます。

  「T」および「O」のどちらも表示されてない場合は，すべてのスレッドおよびオブジェクトに対して監視が行われます。単一スレッド，オブジェクトの監視を行うとき以外はすべてのプログラムがこの設定です。

解除

監視条件を解除するのに使用します。解除したい監視条件を監視条件一覧から選び，［解除］ボタンを選ぶと，選んだ監視条件が解除されます。

全解除

監視条件一覧に表示されているすべての監視条件を解除します。

実行時要素識別子を表示する

監視条件一覧の監視条件に実行時要素識別子を表示します。

(2) 監視条件の解除

1. データ監視条件一覧ダイアログボックスの一覧の中から解除したい監視条件を選ぶ。

   選択した監視条件が選択状態になります。

   複数の監視条件を選ぶこともできます。

2. ［解除］ボタンを選ぶ。

   選択した監視条件が一覧から消去されます。監視条件を複数選んだ場合，一度に複数のデータ条件中断を解除できます。［Delete］キーを押した場合も，選択した監視条件を解除できます。

   全データ条件中断を解除したい場合は，［全解除］ボタンを選びます。

3. ［OK］ボタンを選ぶ。

   設定した内容が有効になります。

10.10　プログラムの実行制御

テストデバッグ対象のプログラムの実行を制御する基本操作について説明します。制御に関する注意事項については，「9.2.1　プログラムの中断」を参照してください。

10.10.1　プログラムの実行開始

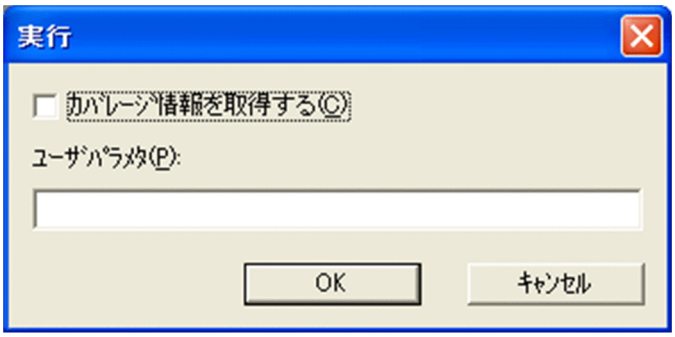
プログラムの実行開始前に，必要に応じて次の項目を設定します。

- カバレージ情報の取得
- ユーザパラメタ

設定の方法には，次の 2 とおりの方法があります。［ツール］メニューから［オプション］を選び，オプションダイアログボックスで切り替えを設定します。オプションダイアログボックスの詳細については，「10.18　テストデバッグのオプション設定」を参照してください。

(1)　毎回設定

プログラムが実行を開始するたびに，次のダイアログボックスを表示し，カバレージ情報を取得するかどうか，およびユーザパラメタを設定します。



最初に設定したあと，テストデバッグ対象を変更するまで，プログラムの実行を開始するときには前回の内容が表示されます。

(2)　固定設定

あらかじめカバレージ情報を取得するかどうか，およびユーザパラメタの設定をしておき，プログラムの実行を開始するときに自動的に適用します。固定設定の方法は，次のとおりです。

1. ［ツール］メニューから［オプション］を選ぶ。
   オプションダイアログボックスが表示されます。
   オプションダイアログボックスについては，「10.18　テストデバッグのオプション設定」を参照してください。

2. 「次の属性で実行開始する」を選ぶ

3. カバレージ情報を蓄積するときは，「カバレージ情報を取得する」チェックボックスをオンにする。

4. テストデバッグ対象プログラムに渡すパラメタの値を設定するときは，ユーザパラメタに設定する。

5. [OK] ボタンを選ぶ。
   設定した内容が有効になります。

10.10.2　連続実行

中断が発生するまで実行します。中断点が設定されていないでエラーが発生しない場合は，最後まで実行します。

1. [プログラムの制御] メニューから [連続実行] を選ぶ。
   連続して実行します。

注意事項

アニメーションの設定がされている場合は，実行文の背景色がカスタマイズダイアログボックスで指定された色で強調表示されます。カスタマイズダイアログボックスについては，「10.18　テストデバッグのオプション設定」を参照してください。また，アニメーションの設定がされていない場合は，ステータスバーに「実行中」というメッセージが表示されます。

10.10.3　ステップイン

呼び出すプログラムを対象として 1 ステップ実行します。ほかのプログラムを呼び出す文で実行した場合，呼び出されるプログラムに制御が渡り中断します。呼び出されるプログラムも 1 文ずつ実行されます。

1. [プログラムの制御] メニューから [ステップイン] を選ぶ。
   1 ステップだけ実行されます。

注意事項

次に実行される文が登録集原文中であり，かつ登録集原文が非表示の場合の詳細については，「10.7.4 登録集原文の表示／非表示」を参照してください。

10.10.4　ステップオーバー

呼び出すプログラムを対象外として 1 ステップ実行します。ほかのプログラムを呼び出す文で実行した場合でも，呼び出されるプログラムに制御は渡らず，呼び出し文の次の文までまとめて実行されます。

1. [プログラムの制御] メニューから [ステップオーバー] を選ぶ。

1 ステップだけ実行されます。呼び出されるプログラムの中に中断文が存在した場合や，エラーが発生した場合には，プログラムは中断します。

注意事項

次に実行される文が登録集原文中であり，かつ登録集原文が非表示の場合の詳細については，「10.7.4 登録集原文の表示／非表示」を参照してください。

10.10.5　ジャンプ

プログラムの一部を実行しないで飛び越したいときに使用します。すでに実行した文に戻ることもできます。

ジャンプできる範囲については，「12.4.28　STEP TO（ステップツー実行の再開）」を参照してください。

1. **ソーステキストウィンドウから，ジャンプしたい先の文キーワードを選ぶ。**
   選択した文キーワードが選択状態になります。
2. **ポップアップメニューから［ジャンプ］を選ぶ。**
   選択した文まで飛んで中断します。

10.10.6　ジャンプ実行

プログラムの一部を実行したくない場合に，その部分を飛ばして実行できます。または，すでに実行した文に戻って実行することもできます。ジャンプとの違いは，ジャンプ後，そのままプログラムの実行を開始することです。

1. **ソーステキストウィンドウから，次に実行したい文キーワードを選ぶ。**
   選択した文キーワードが選択状態になります。
2. **ポップアップメニューから［ジャンプ実行］を選ぶ。**
   選択した文から実行が再開します。

10.10.7　実行の中止

実行中のプログラムを強制的に終了させます。

1. **［プログラムの制御］メニューの［プログラムの実行を終了］を選ぶ。**
   プログラムの実行が終了します。
   アニメーション表示状態の場合も，背景色の強調表示は解除されます。

注意事項

テストデバッグ中の CUI モードのプログラムを強制的に終了するときは，プログラムの実行によって開かれたコマンドプロンプトのメニューを操作して画面を閉じてはいけません。［プログラムの実行を終了］メニューによって，プログラムを強制的に終了させてください。

10.10.8　実行の割り込み中断

実行中のプログラムを強制的に中断させます。

1. ［中断］メニューから［割り込み中断］を選ぶ。

   割り込みが発生した文でプログラムの実行が中断します。中断した文の背景色はカスタマイズダイアログボックスで指定した色で強調表示されます。カスタマイズダイアログボックスについては，「10.18 テストデバッグのオプション設定」を参照してください。

10.11　プログラムの実行追跡

10.11.1　アニメーション

アニメーションは，プログラムの実行状況をソースウィンドウに追跡して表示する機能です。アニメーションを指定してプログラムを実行すると，実行位置の文の背景色が順次変化します。実行位置の文が開いているソーステキストウィンドウにない場合は，ソーステキストウィンドウを新しく開き，その文を表示します。

1. [ツール] メニューから［オプション］を選ぶ。
   オプションダイアログボックスが表示されます。
   オプションダイアログボックスについては，「10.18　テストデバッグのオプション設定」を参照してください。

2. [ソーステキスト] タブを選び，「アニメーションを設定する」のチェックボックスをオンにする。
   必要に応じて，実行速度を設定します。
   アニメーションを解除する場合は，チェックボックスをオフにします。

3. [OK] ボタンを選ぶ。
   設定した内容が有効になります。

注意事項

- アニメーション指定をしてプログラムを起動または続行した場合，ソーステキストウィンドウは最大 25 個までしか開きません。このため，26 個目を開くときに，使用順位のいちばん古いソーステキストウィンドウが自動的に閉じられてから，新たにウィンドウが開かれます。使用順位のいちばん古いウィンドウとは，最初に開いたウィンドウです。ユーザがウィンドウを操作するかアニメーション表示によってウィンドウがアクティブになると，このウィンドウの使用順位はいちばん新しくなります。現在の中断位置を表示する場合は，メニューから「現在位置の表示」を選択してください。
- 登録集原文中であり，かつ登録集原文が非表示の場合の文に対するアニメーションの詳細については，「10.7.4　登録集原文の表示／非表示」を参照してください。
- アニメーションの設定を解除すると，プログラムの実行速度は速くなります。
- アニメーションの指定は，マルチスレッドプログラムでは無効になります。

10.11.2　アニメーション抑止の設定

特定のソース要素に対して，アニメーションをしないように抑止します。

1. テストデバッグ一覧ウィンドウで，アニメーションを抑止したい翻訳単位またはソース要素を選択する。

2. ポップアップメニューから「アニメーション抑止の設定」を選ぶ。

   翻訳単位を選択した場合は，その下位のソース要素すべてにアニメーション抑止が設定されます。ソース要素を選択した場合は，そのソース要素だけにアニメーション抑止が設定されます。

注意事項

- アニメーションが設定されていない場合，この設定は無効です。ただし，アニメーションが設定されていない場合でも，アニメーション抑止を設定できます。
- アニメーション抑止が設定されている場合，アニメーションの実行速度の設定は無効となります。

10.11.3　アニメーション抑止の解除

アニメーション抑止を解除します。

1. テストデバッグ一覧ウィンドウで，アニメーション抑止を解除したい翻訳単位またはソース要素を選択する。

2. ポップアップメニューから「アニメーション抑止の解除」を選ぶ。

注意事項

- アニメーション抑止が設定されていないソース要素を選択した場合，解除はできません。
- 翻訳単位を選択した場合，その下位のソース要素に一つもアニメーション抑止が設定されていない場合，解除はできません。

10.11.4　バックトレース

中断点およびプログラム終了に至るまでの実行経路を表示します。プログラムの実行後に蓄積されたバックトレース情報を基に，プログラムの実行経路を 1 文ずつ表示できます。表示方法は，現在の位置からさかのぼって表示していくことも，実際に実行した順に従って表示することもできます。バックトレース蓄積数の設定方法については，「10.18　テストデバッグのオプション設定」を参照してください。

1. [編集/表示] メニューから［バックトレース］を選び，次の四つのサブメニューからどれかを選択する。

- 進む

  バックトレース位置を一つ先に進めます。

- 戻る

  バックトレース位置を一つ前に戻します。

- 先頭

  蓄積されているバックトレース情報の中で，最初にプログラム実行された文にバックトレース位置を表示します。

- 終了

バックトレース位置の表示を終了します。

ソーステキストウィンドウに表示されるバックトレース位置は，カスタマイズダイアログボックスで設定した色で強調表示されます。カスタマイズダイアログボックスについては，「10.18　テストデバッグのオプション設定」を参照してください。

注意事項

- バックトレース中にデータの参照をする場合，そのデータの内容は，プログラムの中断時点のデータです。バックトレースによって表示されている文が実行された時点のものではありません。
- バックトレース位置を含むソーステキストがソーステキストウィンドウに分割して表示されていた場合，下側のウィンドウに表示されるバックトレース位置だけがバックトレース位置を表す色で強調表示され，上側のウィンドウに表示されているバックトレース位置の表示は変わりません。
- 次に表示される文が登録集原文中であり，かつ，登録集原文が非表示の場合の詳細については，「10.7.4　登録集原文の表示／非表示」を参照してください。
- バックトレースによって実行時要素が変更になった場合，その実行時要素が呼び出し一覧ウィンドウ上に表示されているときは，実行時要素の切り替えが行われます。呼び出し一覧ウィンドウ上に表示されていない実行時要素の場合，実行時要素の切り替えは行われません。したがって，その実行時要素のデータなどは正しく表示されません。
- バックトレース位置を含むソーステキストがソーステキストウィンドウに分割して表示されていた場合と同様に，スレッドが変更になったとき，呼び出し一覧ウィンドウ上に表示されているスレッドの場合，スレッドの切り替えが行われます。

10.12　実行時要素の切り替え

実行時要素の切り替えとは，中断したプログラムとは異なるプログラムを操作の対象にすることです。中断したプログラム以外のプログラムにあるデータ値を参照できます。実行時要素の切り替えは，マルチスレッドプログラム，オブジェクト指向機能のメソッド，または再帰的プログラムの呼び出しで，同一のプログラムが複数同時に実行されているとき，特定のデータの参照をするための機能です。

1. 呼び出し一覧ウィンドウから操作対象とする実行時要素を選ぶ。
2. ポップアップメニューから［実行時要素の切り替え］を選ぶ。

   選んだスレッド ID と実行時要素に”>”マークが表示されます。

   また，常時データ表示ウィンドウに登録されているデータ名の値が，操作対象となったプログラムのデータ名の値に書き換わります。

   実行時要素上をダブルクリックしても，操作対象にできます。

マルチスレッドプログラムの場合は，上の操作の前に，次の操作をします。

1. 呼び出し一覧ウィンドウでスレッド ID を選ぶ。

   選択したスレッドに対応する呼び出し順序と位置が表示されます。

注意事項

- プログラムを続行すると切り替えは解除されます。
- ［編集/表示］メニューから［現在位置の表示］を選んで現在中断中の文を表示した場合は，中断文のあるプログラムに戻ります。

10.13　データ値の表示と代入

データ名の値を表示し，データの動きを追いながらテストデバッグできます。また，データ名に値を代入して，結果を確認できます。

10.13.1　データ値の表示

プログラムの中断時に，データの値を表示します。

データ値の表示には，次の 2 とおりの方法があります。

(1)　ソーステキストウィンドウから値を表示したいデータ名を選ぶ方法

1. ソーステキストウィンドウから値を表示したいデータ名を選ぶ。

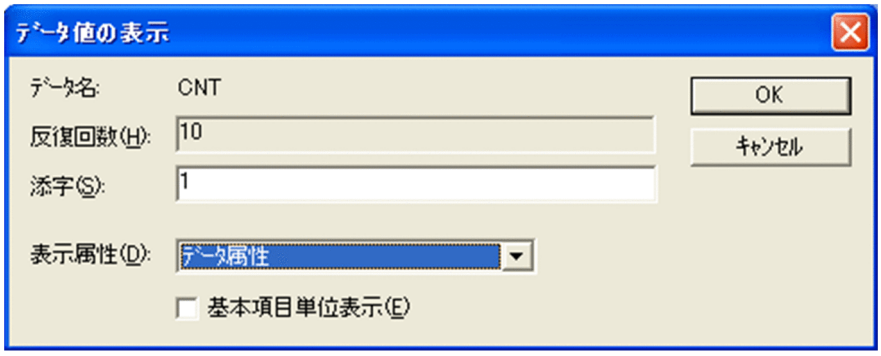
   選んだデータ名が選択状態になります。

2. [データの操作] メニュー，またはポップアップメニューから［データ値の表示］を選ぶ。

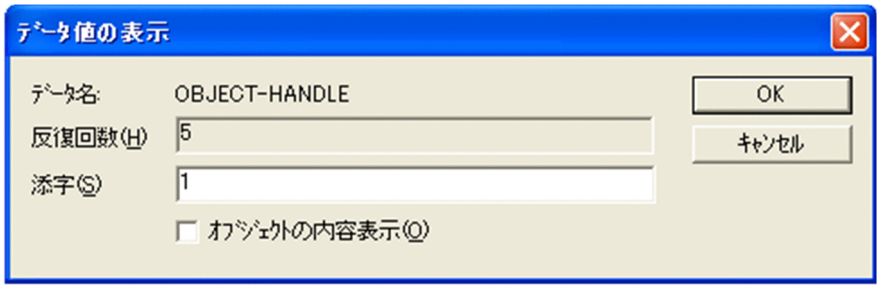
   選択したデータ名の型によって，次のどれかのデータ値の表示ダイアログボックスが表示され，手順 1.で選択した文字列が表示されます。

   添字が必要なデータ名の場合は，添字を入力するための領域が表示されます。添字が複数個ある場合はコンマ（,）または空白文字で区切って入力します。

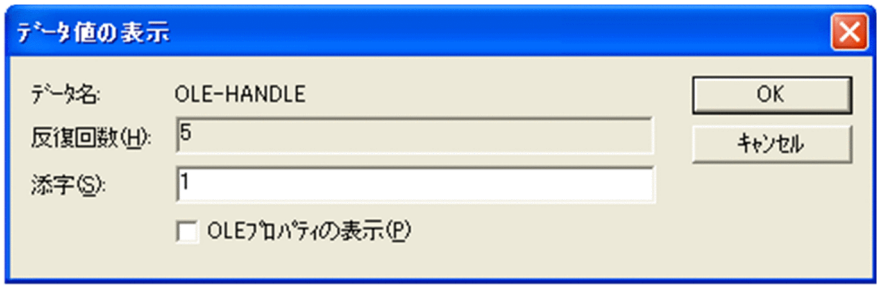
   - 通常のデータ，またはバリアントデータの場合

   - オブジェクト参照データの場合

   - OLE オブジェクト参照データの場合



3. 表示形式を選択して，［OK］ボタンを選ぶ

   モニタウィンドウに値が表示されます。モニタウィンドウが非表示のときは，モニタウィンドウが表示されます。

注意事項

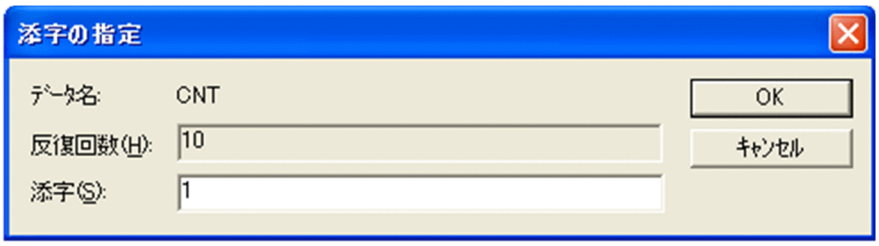
- 添字を入力する領域には，数字，空白文字，コンマ（,）以外は入力できません。
- 部分参照を表示することはできません。
- データ値の表示ダイアログボックスのデータ名の編集領域には，ソーステキストウィンドウで選択した文字列がそのまま表示されます。添字を含んで選択することはできません。
- 表示形式は，オプション設定のダイアログボックスによって，あらかじめ設定できます。

(2) ソーステキストウィンドウで値を表示したいデータ名をダブルクリックする方法

1. ソーステキストウィンドウで値を表示したいデータ名をダブルクリックする。

   モニタウィンドウに，ソーステキストウィンドウで選択したデータ名の値が表示されます。

   データに添字の指定が必要な場合は，添字の指定ダイアログボックスが表示されます。



(3) 添字の設定方法

反復回数には，指定できる添字の最大値が表示されます。

1. 添字を入力する。

   添字の入力領域に，添字を入力します。添字が複数個ある場合はコンマ（,）または空白文字で区切って入力します。

2. ［OK］ボタンを選ぶ。

添字の指定ダイアログボックスが閉じます。添字の指定方法に誤りがある場合は，エラーメッセージが表示され，ダイアログボックスは閉じません。

注意事項

- 「添字」には数字，空白文字，コンマ（,）以外の値は入力できません。
- 部分参照は指定できません。
- ソーステキストウィンドウ上でデータ名をダブルクリックしたときは，オプション設定のダイアログボックスで設定した表示属性で表示されます。詳細は「10.18.5　データ表示」を参照してください。

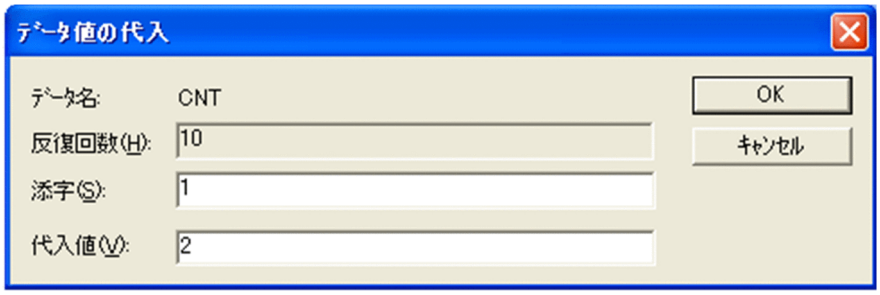
10.13.2　データ値の代入

プログラムの中断時に，データの値を代入します。

データの値を代入する方法は，次のとおりです。

1. ソーステキストウィンドウから値を代入したいデータ名を選ぶ。
   選んだデータ名が選択状態になります。

2. [データの操作] メニュー，またはポップアップメニューから［データ値の代入］を選ぶ。
   データ値の代入ダイアログボックスが表示されます。
   選んだデータ名が表要素の場合，添字を入力するための領域が追加されます。



3. 代入したい値を入力する。
   入力の書式については，「9.3.1　データの比較・代入規則」の「(1)　データの代入規則」を参照してください。

4. [OK] ボタンを選ぶ。
   添字の指定方法に誤りがある場合は，エラーメッセージが表示され，ダイアログボックスは閉じません。

注意事項

値の代入は，操作対象が実行中断したプログラムのスレッドと同一のスレッドの場合だけできます。呼び出し一覧ウィンドウで実行中断したプログラムのスレッド以外の実行時要素を操作対象に切り替えた場合は，メニューの「データ値の代入」は選択できなくなり，値の変更はできません。

10.14　現在位置の表示

中断状態となった中断文を表示します。

1. ［編集/表示］メニューから［現在位置の表示］を選ぶ。

   中断状態となった中断文のあるソーステキストウィンドウが表示され，その中の中断した文がウィンドウ内に表示されます。

   実行時要素が中断した実行時要素とは別の実行時要素に切り替わっている場合は，現在位置の表示を行うと，中断した実行時要素に戻ります。切り替えられていたスレッド ID とプログラムに表示されていた”＞”マークが，戻ったプログラムのスレッド ID とプログラムに表示されます。常時データ表示ウィンドウに登録されているデータ値も，戻ったプログラムでの値が表示されます。

注意事項

- 登録集原文中に中断している場合の詳細については，「10.7.4　登録集原文の表示／非表示」を参照してください。
- バックトレース実行中の場合，バックトレースは終了します。

10.15　ソースファイルの編集

ソーステキストウィンドウからテキストエディタを起動します。

1. ソーステキストウィンドウから編集したい行を選ぶ。

2. ［編集/表示］メニュー，またはポップアップメニューから［ソースファイルの編集］を選ぶ。
   開発マネージャで設定したテキストエディタが起動され，指定した行を含むソーステキストが表示されます。なお，ソーステキストを表示したままでテストデバッグを続けることもできます。

注意事項

- ソースファイルが，コンパイル時と絶対パスが同一のフォルダにない場合は，ソースファイルの編集はできません。
- 編集したソースファイルは，ビルドし直すまで，テストデバッガには反映されません。編集したソースファイルでテストデバッグをしたい場合は，一度，テストデバッガを終了し，ビルドし直してから，再度テストデバッグをしてください。

10.16　文字列の検索

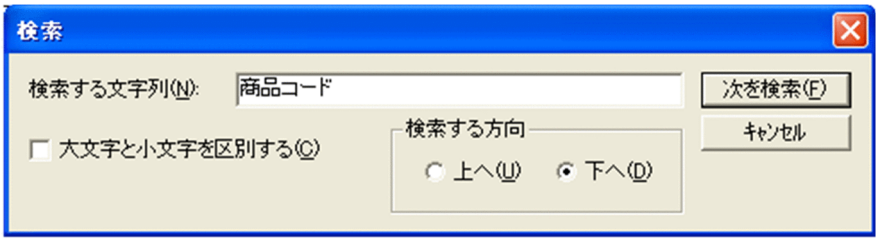
10.16.1　ウィンドウ内の文字列検索

ソーステキストウィンドウ，またはモニタウィンドウの文字列を検索します。

1. 文字列の検索をするウィンドウにフォーカスを位置づける。

2. ［編集/表示］メニュー，またはポップアップメニューから［文字列の検索］を選ぶ。

   検索ダイアログボックスが表示されます。



3. 検索する文字列を入力する。

   大文字と小文字を区別して検索するときは，「大文字と小文字を区別する」のチェックボックスをオンにします。検索する方向を設定するときは，「上へ」または「下へ」を選択します。

4. ［次を検索］ボタンを選ぶ。

   検索する文字列と一致する最初の文字列が表示されます。

   検索が行われたあとは，ツールバーの［検索文字列入力域］に検索文字列が登録され，［下を検索］ボタンまたは［上を検索］ボタンを選ぶと，引き続き同じ文字列を検索できます。

注意事項

- 文字列の検索は行単位で行います。文字列が 2 行にわたる場合は一つの文字列とはみなしません。
- ソーステキストウィンドウで文字列を選択し，［文字列の検索］によって検索ダイアログボックスを表示すると，［検索する文字列］に選択された文字列が表示されます。ただし，複数行を選択したときは，はじめの 1 行目だけを［検索する文字列］に設定します。

10.17　プログラムからの連動実行

10.17.1　操作手順

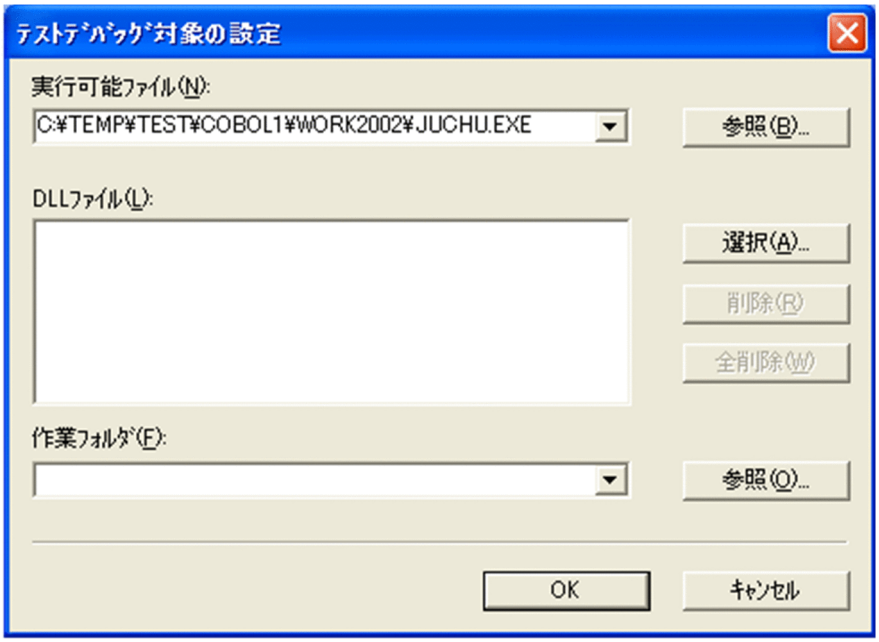
連動実行で，ほかのプロセスから起動されるプログラムのテストデバッグを開始する場合については，「10.1.1　テストデバッグの手順」を参照してください。

連動実行でテストデバッグを行う場合の操作の詳細について説明します。

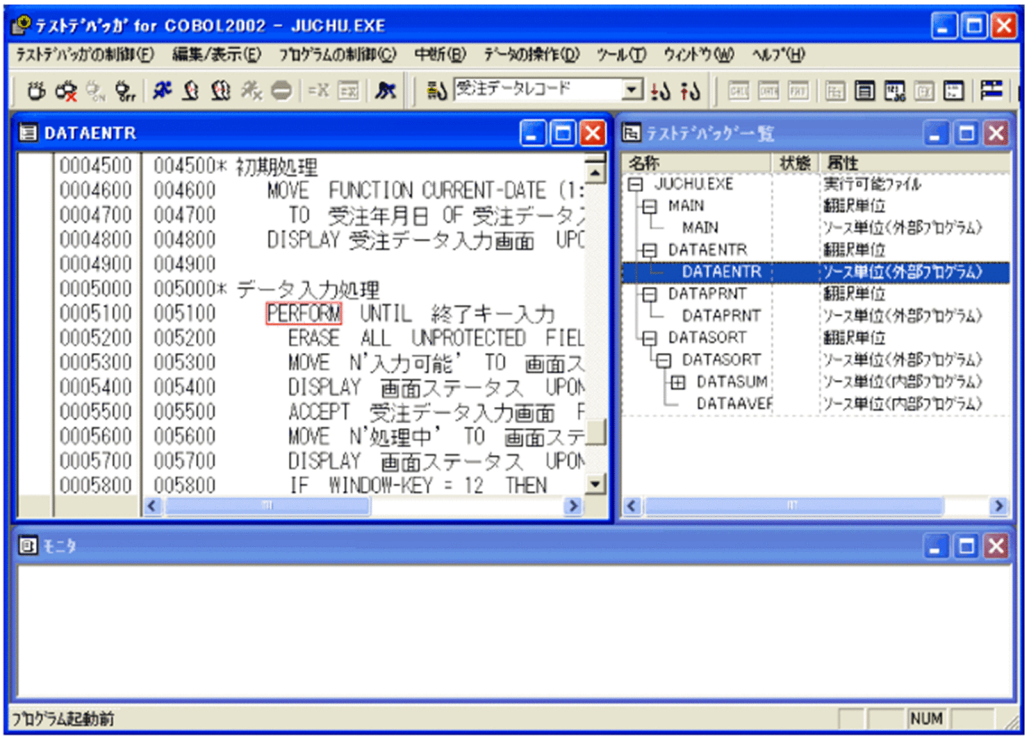
1. Windows のプログラム一覧にある［COBOL2002］下のメニュー，または開発マネージャからテストデバッガを起動する。

   起動方法の詳細については，「10.2.1　テストデバッガの起動」を参照してください。

2. テストデバッグ対象の設定ダイアログボックスで，実行可能ファイル(.exe)および DLL ファイルを設定する。



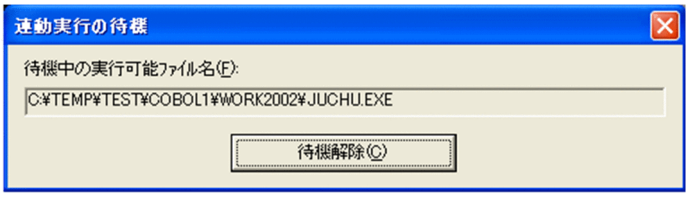
3. 中断点などを設定し，テストデバッグの準備をする。



4. [プログラムの制御] メニューから [連動実行の待機] を選ぶ。

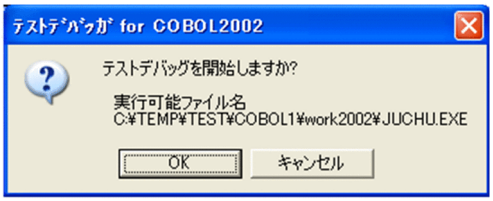
   連動実行の待機ダイアログボックスが表示され，テストデバッグで使用するウィンドウはアイコン化されます。待機中の実行可能ファイル名については，テストデバッグ対象の設定ダイアログボックスで指定した実行可能ファイル名が表示されます。

   連動実行を中止する場合には，「待機解除」ボタンを選び，手順 3.の状態に戻ります。



5. テストデバッグ対象のプログラムを実行する。

   環境変数 CBLTDEXEC=TD が指定された実行環境で実行された実行可能ファイルのプログラムに制御が渡ると，次のダイアログボックスが表示されます。



6. ダイアログボックスに応答する。

   - [OK] ボタンを選ぶ。

ダイアログボックスと連動実行の待機ダイアログボックスが閉じて，テストデバッグウィンドウがアイコンから元のサイズに戻り，ユーザが操作できる状態となります。ただし，この段階では，実行プログラムは待機状態のままです。

- ［キャンセル］ボタンを選ぶ。

  連動実行の待機状態に戻ります。

7. テストデバッグを開始する。

テストデバッグの対象となった実行可能ファイルと DLL ファイル中のプログラムが実行されます。

実行可能ファイルと DLL ファイル中のプログラムが最後まで実行され，プログラムが終了すると，手順 3.の状態になります。再度，連動実行によるテストデバッグを開始したい場合は，手順 4.以降の操作をします。

注意事項

この注意事項で使用されるプログラムとは，環境変数 CBLTDEXEC=TD の指定がある環境で実行される連動実行の対象となるプログラムのことを意味します。

同様に，待機状態のテストデバッガとは，手順 4.の状態から連動実行待機ダイアログボックスの「待機解除」ボタンを選ぶまでの状態，および手順 4.の状態から手順 5.のテストデバッグを開始するかどうかを確認するダイアログボックスで［OK］ボタンを選ぶ直前までの状態であるテストデバッガを意味します。

プログラム起動時の注意事項

プログラムが起動されたときに，次のどれかの条件を満たす場合，手順 5.のテストデバッグを開始するかどうかを確認するダイアログボックスを表示しないでプログラムが続行されます。

- プログラムが起動されたタイミングで，待機状態のテストデバッガがないとき
- プログラム名と，テストデバッグで指定した実行可能ファイル名（パスも含みます）が一致する待機状態のテストデバッガがないとき

連動実行の待機中の注意事項

- テストデバッグを開始するかどうかを確認するダイアログボックスは，待機状態になった順番に表示します。対象となる待機中のテストデバッガとは，次のとおりです。

  ・同一のプログラムをテストデバッグ対象とするテストデバッガ。

  ・DLL ファイルだけをテストデバッグ対象にしているテストデバッガ。

  DLL ファイルだけの連動実行については「10.17.2　DLL ファイルだけの連動実行について」を参照してください。

- テストデバッグを開始するかどうかを確認するダイアログボックスで［OK］ボタンを選んでも，連続実行やステップインなどでプログラムを実行しなければ，プログラムは待機状態となったままです。

  プログラムを実行しない状態で，テストデバッガを終了させようとしたときは，プログラムが続行されてから，テストデバッガが終了します。

  同様に，プログラムを実行しない状態で，テストデバッグ対象の設定ダイアログボックスでテストデバッグの対象を切り替えた場合も，プログラムが続行されます。

待機状態のテストデバッガが複数ある場合の注意事項

- 同一のプログラムをテストデバッグするための待機状態のテストデバッガが複数ある状態で，同一のプログラムが複数起動された場合には，次の動作となります。

  ・テストデバッグを開始するかどうかを確認するダイアログボックスにユーザが応答しないでいる場合，対象となるプログラムも，また，あとから起動された同一のプログラムも待ち状態となります。

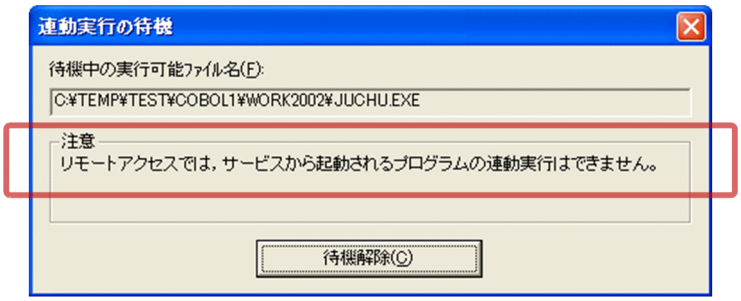
  ・待機している複数のテストデバッガの中に DLL ファイルだけをテストデバッグ対象にしているテストデバッガが含まれている場合，テストデバッグを開始するかどうかを確認するダイアログボックスで，［キャンセル］を応答したときに，テストデバッグを開始するかどうかを確認するダイアログボックスが表示される順番は，テストデバッガを待機状態にした順番と異なることがあります。

- テストデバッグを開始するかどうかを確認するダイアログボックスがほかのウィンドウの後ろに隠れてしまうことがあります。［Alt］＋［Tab］キーやタスクバーのアイコンを操作して，ダイアログボックスを前面に表示してください。

リモートアクセスしたクライアントでデバッグするときの注意事項

- Windows のサービス機能から起動するユーザープログラムを，連動実行でデバッグできません。デバッグする場合は，リモートアクセスしたクライアントからではなく，サーバ上でデバッグ操作をする必要があります。リモートアクセスしたクライアントで連動実行の待機をした場合は，環境変数 CBLTDEXEC を指定していた場合でも，手順 5.のテストデバッグを開始するかどうかを確認するダイアログボックスは表示されません。また，エラーメッセージも表示されません。

- リモートアクセスしたクライアントで連動実行の待機をした場合，手順 4.では，代わりに，次に示すダイアログボックスが表示されます。



  このダイアログボックスを表示するかどうかは，連動実行の待機をしたときの状態で判断します。連動実行の待機をした状態のまま，リモートアクセスに切り替えても，逆にリモートアクセスをやめても，注意事項の表示/非表示は切り替わりません。そのため，例えば連動実行の待機をした状態のままでリモートアクセスに切り替えた場合，上記の注意事項は表示されていませんが，サービスから起動されるプログラムの連動実行はできません。

その他の注意事項

- テストデバッガを終了した時点でプログラムは強制終了されます。

- OpenTP1 から起動されるプログラムなど，プログラムが Windows のサービス機能から起動された場合，ファイルに対する参照権限の設定によっては，プログラム情報ファイル（.cbp）お

よびソースファイルが参照できない場合があります。この場合，プログラム情報ファイル（.cbp）およびソースファイルに，テストデバッガを起動するユーザ ID で参照できるように参照権限を設定しておいてください。参照権限がない場合，プログラム情報ファイルがデバッグ対象から外れたり，ソースファイルがエディタで開けなかったりします。

- 連動実行する場合は，テストデバッガ，およびデバッグ対象プログラムの起動時の権限に注意が必要です。詳細は，「付録 A.1　管理者権限についての注意事項」を参照してください。
- OS の「グローバルオブジェクトの作成」権限がない場合は，［連動実行の待機］メニューが非活性になり，手順 4.の連動実行の待機にする操作ができません。

10.17.2　DLL ファイルだけの連動実行について

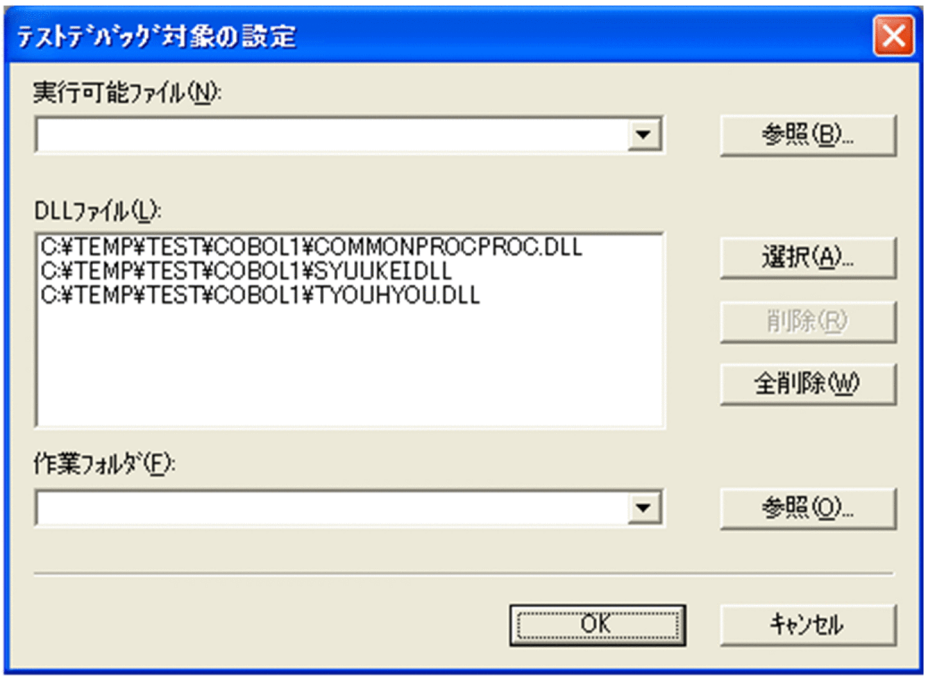
DLL ファイルだけテストデバッグしたい場合や，実行可能ファイルは不明でも，テストデバッグ対象の DLL ファイルがわかる場合には，次の方法によってテストデバッグをします。

1. Windows のプログラム一覧にある［COBOL2002］下のメニュー，または開発マネージャからテストデバッガを起動する。

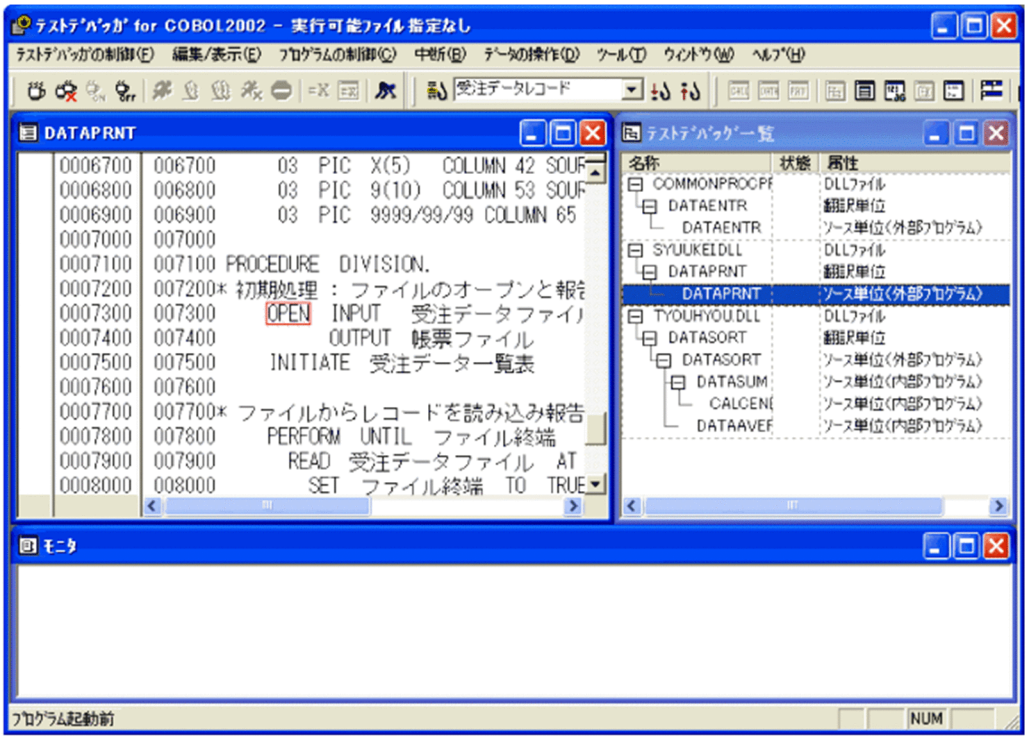
   起動方法の詳細については，「10.2.1　テストデバッガの起動」を参照してください。

2. テストデバッグ対象の設定ダイアログボックスで，DLL ファイルを設定する。

   実行可能ファイルを設定する必要はありません。



3. 中断点などを設定し，テストデバッグの準備をする。



4. [プログラムの制御] メニューから [連動実行の待機] を選ぶ。

   連動実行の待機ダイアログボックスが表示され，テストデバッグウィンドウはアイコン化されます。待機中の実行可能ファイル名には何も表示されません。

   待機状態を解除したい場合は，「待機解除」ボタンを選ぶと，手順 3.の状態に戻ります。



5. テストデバッグ対象のプログラムを実行する。

   環境変数 CBLTDEXEC=TD が指定された実行環境で実行された実行可能ファイルのプログラムに制御が渡るとテストデバッガに制御が移り，次のダイアログボックスが表示されます。



6. ダイアログボックスに応答する。

   - [OK] ボタンを選ぶ。

ダイアログボックスと連動実行の待機ダイアログボックスを閉じて，テストデバッグウィンドウがアイコンから元のサイズに戻り，ユーザが操作できる状態となります。ただし，この段階では，実行プログラムは待機状態のままです。

- ［キャンセル］ボタンを選ぶ。

  連動実行の待機状態に戻ります。

7. テストデバッグを開始する。

DLL ファイルが実行されます。テストデバッグの対象となるのは，手順 2 で設定した DLL ファイルだけです。実行可能ファイルはテストデバッグの対象となりません。

テストデバッガで対象となった DLL ファイル中のプログラムが最後まで実行され，プログラムが終了すると，手順 3.の状態になります。再度，連動実行によるテストデバッグを開始したい場合は，手順 4.以降の操作をします。

注意事項

- プログラムが起動されたとき，次の条件を満たす場合，手順 5.のダイアログボックスを表示しないでプログラムが続行されます。

  ・実行可能ファイル名を指定していない待機状態のテストデバッガがない場合。

  このほかの注意事項については，「10.17.1　操作手順」を参照してください。

10.17.3　連動実行のエラーについて

連動実行で，モニタ画面などの画面上にエラーメッセージを出力できないときは，次のとおりにメッセージを出力します。

(1)　イベントログが使用できる Windows

イベントログにメッセージを出力します。何らかの理由でイベントログを出力できなかった場合は，「(2) イベントログが使用できない Windows およびイベントログに出力ができないとき」のとおりとなります。

(2)　イベントログが使用できない Windows およびイベントログに出力ができないとき

イベントログが出力できなかった場合は，temp フォルダに結果ファイルを出力します。

また，この結果ファイルにも出力できないときは，標準エラー出力に出力します。ただし，-Lib,GUI コンパイラオプションを指定してコンパイルしたプログラムの場合は，メッセージは表示されません。

temp フォルダ

環境変数 TMP，TEMP，または USERPROFILE に設定されているフォルダです。環境変数が設定されていない場合は，次のフォルダとなります。

- Windows のフォルダ（C:¥Windows など）

結果ファイル名

TDO*****.tmp となります。*****には任意の文字が入ります。

10.18　テストデバッグのオプション設定

テストデバッグ操作のためのオプション設定をする方法について説明します。

1. [ツール] メニューから［オプション］を選ぶ。
   オプション設定のダイアログボックスが表示されます。設定したいタブを選択し，表示されるページに必要事項を指定します。

2. [OK] ボタンを選ぶ。
   設定した内容が有効になります。

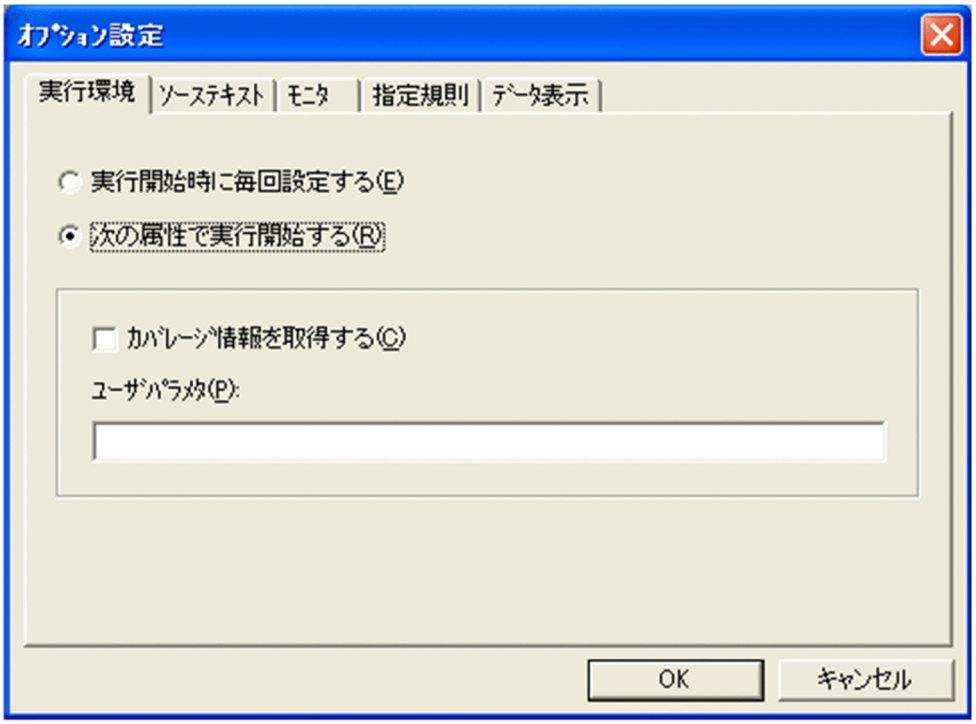
注意事項

テストデバッガを起動し，オプション設定したあとさらにテストデバッガを起動すると，前に起動されたテストデバッガによって設定されたオプションを反映して動作します。

オプション設定のダイアログボックスで設定できる内容について説明します。

10.18.1　実行環境

プログラムの実行環境を設定できます。



- 「実行開始時に毎回設定する」
  プログラムの実行開始のたびに，ユーザパラメタとカバレージ情報を取得するかどうかのダイアログボックスを表示するときに選択します。

- 「次の属性で実行開始する」
  プログラムの実行開始のたびに，次の設定した情報でプログラムの実行を開始します。
  - カバレージ情報を取得する
    カバレージ情報を蓄積するときは，チェックボックスをオンにします。
  - ユーザパラメタ
    テストデバッグ対象プログラムに渡すパラメタの値を設定します。

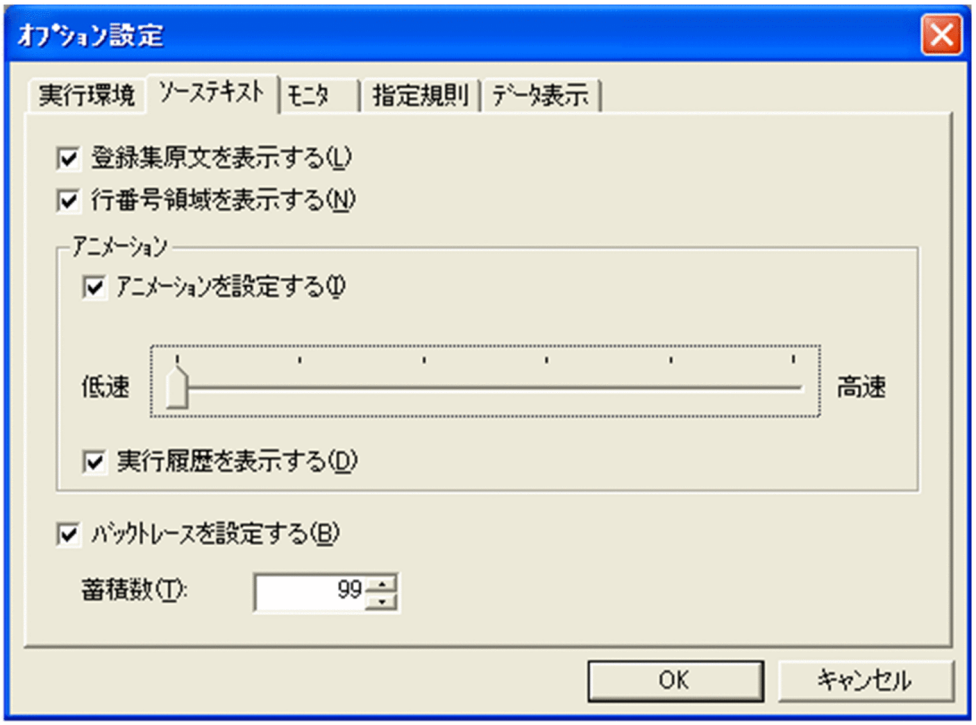
注意事項

次の場合，ユーザパラメタで指定した文字列は無視されます。

- テストデバッグ対象のプログラムが主プログラムシミュレーション指定（-SimMain）でコンパイルされているとき
- 連動実行のとき

10.18.2　ソーステキスト

ソーステキストウィンドウの表示内容やアニメーションの設定ができます。



- 登録集原文を表示する
  ソーステキストウィンドウに登録集原文を表示したい場合は，チェックボックスをオンにします。
- 行番号領域を表示する
  ソーステキストウィンドウの行番号領域を表示したい場合は，チェックボックスをオンにします。
- アニメーションを設定する

アニメーションを設定したい場合は，チェックボックスをオンにします。

アニメーションを設定した場合は，プログラムの実行速度を設定することもできます。

アニメーションを設定してプログラムを実行すると，順次，実行された文を強調表示します。

- 実行履歴を表示する

  実行履歴を表示したい場合は，チェックボックスをオンにします。ただし，アニメーションの設定を解除した場合は，チェックボックスがオンになっていても，実行履歴の表示は行いません。

- バックトレースを設定する

  バックトレースを設定したい場合は，チェックボックスをオンにします。

  バックトレースを設定した場合は，実行経路を蓄積しておく最大数を設定することもできます。

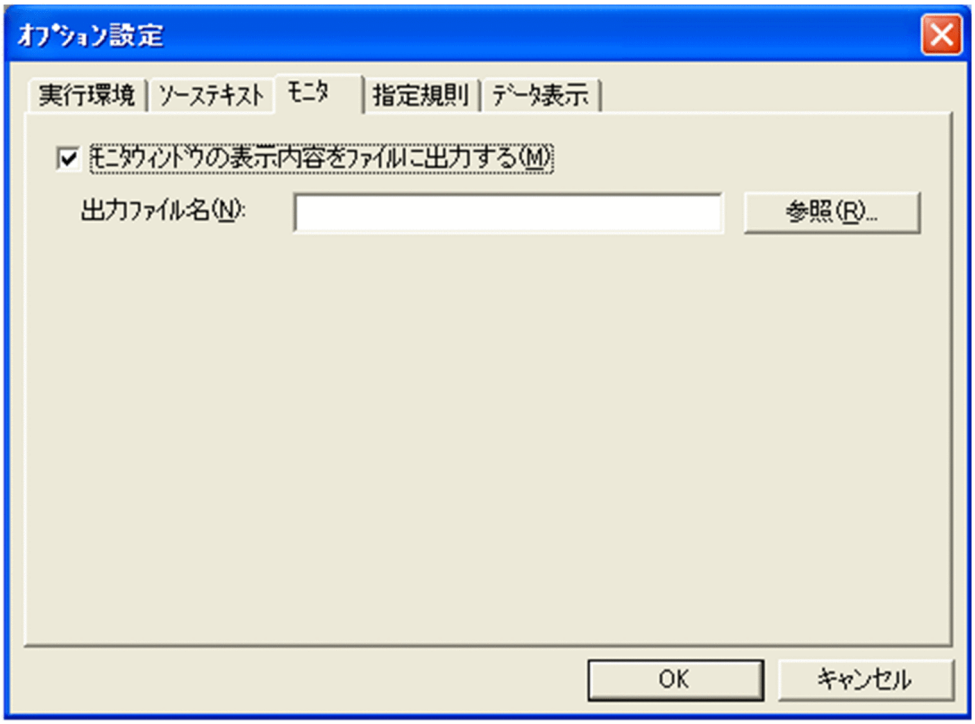
  蓄積数には 1～500 の値が指定できます。初期値には 99 が設定されています。

注意事項

バックトレース蓄積数を変更した場合，バックトレースのカウンタはクリアされ，ソーステキストウィンドウに現在表示されている強調表示は解除されます。

10.18.3　モニタ

モニタウィンドウの表示内容をファイルに出力できます。



- モニタウィンドウの表示内容をファイルに出力する

  モニタウィンドウに表示する内容をファイルに出力したい場合は，チェックボックスをオンにします。

  このチェックボックスをオンにした場合は，ファイル名を指定しなければなりません。

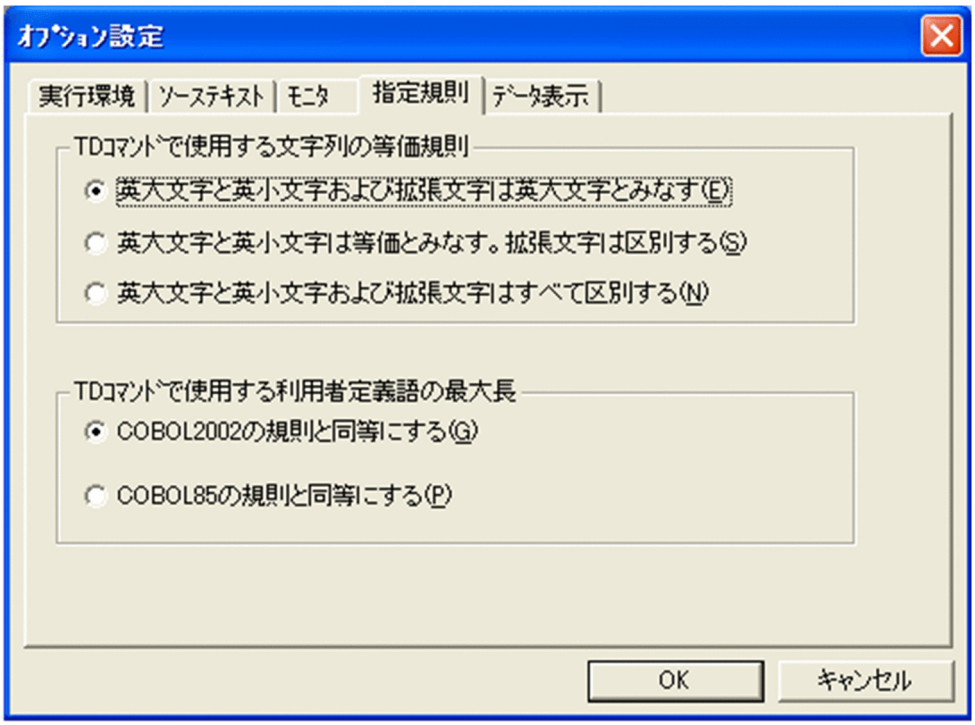
注意事項

SET LOG コマンドによって出力ファイル名が変更された場合も，オプション設定のダイアログボックスに反映されます。SET LOG コマンドについては，「12.4.21　SET LOG／RESET LOG（モニタウィンドウ表示内容のファイル出力の設定と解除）」の「(1)　SET LOG（モニタウィンドウ表示内容のファイル出力の設定）」を参照してください。

10.18.4　指定規則

等価規則と利用者定義語の最大長を設定できます。指定した結果は，次の操作に反映されます。

- TD コマンド入力ウィンドウと TD コマンド格納ファイルから入力した TD コマンド
- データ表示ウィンドウと常時データ表示ウィンドウで入力するデータ名
- データ監視条件の設定ダイアログボックスで入力するデータ名



注意事項

- #OPTION コマンドによって等価規則や利用者定義語の最大長が変更された場合も，オプション設定のダイアログボックスに反映されます。#OPTION コマンドについては，「12.4.36　#OPTION（オプションの変更）」を参照してください。
- TD コマンドのオペランドに指定する利用者定義語の有効文字列長を COBOL2002 または COBOL85 の言語仕様どちらかに合わせてチェックできます。利用者定義語の詳細については，マニュアル「COBOL2002　言語　標準仕様編」を参照してください。

  TD コマンドのオペランドに指定する利用者定義語には，次のものがあります。

  ・プログラム名

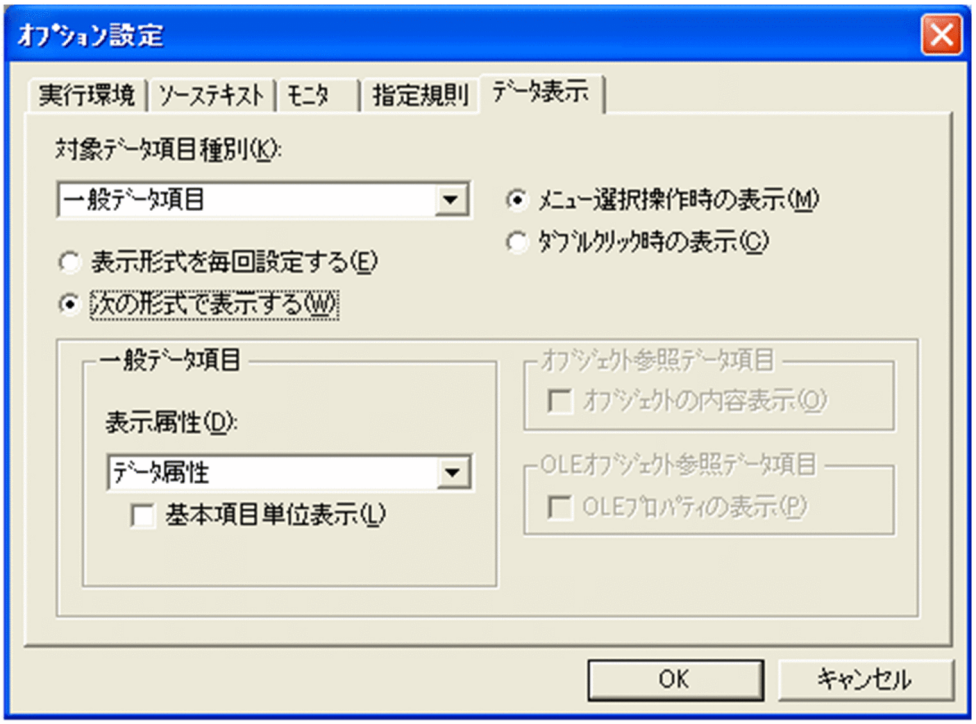
・手続き名（段落名および節名）
・クラス名
・メソッド名
・データ名
・通信記述名
・ファイル名
・利用者定義関数名

10.18.5　データ表示

[データ値の表示] メニューを選択したとき，またはソーステキストウィンドウ上でデータ名をダブルクリックしたときのデータの表示形式を設定できます。

各データの種類（一般データ項目，オブジェクト参照データ項目，OLE オブジェクト参照データ項目）に対し，メニューを選択したとき，またはダブルクリックしたときのそれぞれに対応した表示形式を設定できます。

「表示形式を毎回設定する」を選択すると，メニューの「データ値の表示」選択時に表示形式を問い合わせるダイアログボックスが表示されます。「次の形式で表示する」を選択すると，メニューの「データ値の表示」選択時に表示形式を問い合わせるダイアログボックスは表示されないで，このオプション設定で設定した形式でデータが表示されます。



11 バッチモードによるテストデバッグ

バッチモードによるテストデバッグの方法について説明します。バッチモードは，テスト内容をファイルに登録しておき一括して実行させる方式です。ファイルに登録した TD コマンドを一括して実行させることで，大量のプログラムのテストが効率良く実施できます。

11.1　バッチモードによるテストデバッグの概要

あらかじめ，実行したい TD コマンドをファイルにまとめて記述しておき，そのファイルを起動することで，一括してテストデバッグを実行する方法です。コマンドプロンプトに，TD コマンドを記述したファイルを指定した起動コマンドを入力すると，ファイルに記述された TD コマンドを一括して実行します。一度実行を開始すれば，利用者の操作を必要としないので，大量のプログラムのテストが効率良く実施できます。

11.2　バッチモードでのテストの方法

11.2.1　テストの手順

バッチモードでプログラムをテストするときの作業の流れを説明します。

1. コンパイラオプションを指定し，テストしたいプログラムをコンパイルする。
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。

2. テストに必要なファイルを準備する。
   テスト内容をバッチモードの入力となる TD コマンド格納ファイルに記述します。

3. テストデバッガ，またはプログラムの実行のための環境変数を設定する。
   コマンドプロンプトで必要な環境変数を設定します。

4. cbltd2k コマンドをコマンドプロンプトで指定して実行する。

5. 結果出力ファイルおよび結果蓄積ファイルを開き，結果を確認する。
   カバレージ情報の採取の結果は，カバレージ情報の表示を使用して確認します。
   カバレージ情報の表示については，「13.2　カバレージ情報の表示と操作」を参照してください。

11.2.2　環境変数の指定

テストデバッグに必要な環境変数を指定します。システム，またはバッチモードを実行するコマンドプロンプトに設定します。

(1)　CBLPIDIR

作成または更新したプログラム情報ファイルのあるフォルダ名を指定します。テストデバッガは，次の順番でプログラム情報ファイルを検索します。

1. 環境変数 CBLPIDIR で指定したフォルダ

2. 実行可能ファイルまたは DLL ファイルのあるフォルダ

プログラムの実行に関するそのほかの環境変数については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

注意事項

バッチモードの起動後のカレントフォルダは，起動コマンドを実行したフォルダです。

11.3 cbltd2k コマンドの指定形式

形式

```
cbltd2k  〔-?〕
      -Input TDコマンド格納ファイル名
    〔 -Syntax Only〕〔 -Output 結果出力ファイル名 〕
    〔 -I nclude Path TDコマンド格納ファイルフォルダ名〕
    〔 -Library  DLLファイル名 〔,… 〕  〕
      -Execute  実行可能ファイル名
```

-Input

TD コマンド格納ファイル名（.tdi）を指定します。

-SyntaxOnly

TD コマンドの構文解析だけを行い，実行しません。

-Output

結果出力ファイル名（.tdl）を指定します。省略時は，-Input で指定した TD コマンド格納ファイル名の拡張子を「.tdl」に変更したファイル名とします。

-IncludePath

#INCLUDE コマンドで指定する TD コマンド格納ファイルのフォルダを指定します。省略時は，cbltd2k コマンドで指定した TD コマンド格納ファイルのフォルダを仮定します。cbltd2k コマンドで指定する TD コマンド格納ファイルには適用しません。

-Library

テストデバッグ対象とする DLL ファイル名（.dll）を指定します。

-Execute

テストデバッグ対象プログラムを起動するための実行可能ファイル名（.exe）を指定します。

-?

cbltd2k コマンドの構文を表示します。-?を指定すると，ほかの引数はすべて無視されます。

注意事項

- 引数の指定順序は任意です。ただし，-Execute オプションは，必ず最後に指定します。
- -Library オプションは，引数に複数のファイル名を指定できます。ファイル名を区切るコンマ(,)の前後に空白文字を指定してはいけません。
- 起動後のカレントフォルダは，cbltd2k コマンドを投入したフォルダです。
- cbltd2k コマンドのメッセージは，標準エラー出力に出力します。
- -SyntaxOnly オプションが指定されていても，#INCLUDE コマンドは TD コマンド格納ファイルの取り込みをし，取り込んだ TD コマンド格納ファイル内の TD コマンドの構文解析が行われます。また，#OPTION コマンドの指定を有効にします。

- コマンド名，引数は，英大文字，英小文字のどちらでも指定できます。引数の始まりは，ハイフン(-）の代わりにスラント（/）も使用できます。
- コマンド名だけを指定した場合は，コマンドの構文を表示します。
- 引数の区切り記号は空白文字とタブです。空白文字とタブを区切り記号としたくないときは，引数をダブルコーテーション（"）で囲みます。例えば，空白文字を含むファイル名はファイル名をダブルコーテーション（"）で囲んで指定します。
- 「@コマンドファイル名」の形でコマンドファイルが指定できます。コマンドファイルの内容は，実行時にコマンドラインに変換されます。コマンドファイルはコマンド名よりあとに指定します。
- 同じ引数を複数指定した場合は，あとに指定した引数のパラメタを有効とします。
- 引数にパスの付かないファイル名だけを指定した場合は，カレントフォルダのファイルとします。フォルダ名だけを指定した場合は，カレントフォルダ下にあるフォルダとします。相対パスで指定された場合は，カレントフォルダを基準に検索します。
- コマンドが返すリターン値は，次のとおりです

リターン値	内容
0	正常終了
1	エラー発生による終了
2	キー操作の割り込みによる終了

- -？引数が指定された場合のコマンドの構文は，標準出力に出力されます。それ以外のメッセージは，標準エラー出力に出力されます。

12 TD コマンド

TD コマンドはテストデバッグの指示を行うコマンドです。TD コマンドは，バッチモードで使用する TD コマンド格納ファイルや，TD コマンドで指定したファイルなどに定義します。また，GUI モードの TD コマンド入力ウィンドウでも使用できます。

この章では，TD コマンドの指定方法，オペランドの指定方法，TD コマンドの詳細について説明します。

12.1　TD コマンドの指定方法

12.1.1　TD コマンドの形式

TD コマンドは次の形式で指定します。TD コマンド名とオペランド，オペランドとオペランドの間は，一つ以上の空白，またはタブを記述します。

形式

TD コマンド名 必須オペランド 〔 任意オペランド ・・・ 〕

指定する内容

TD コマンド名

テストデバッグの操作を示す，TD コマンドを指定します。TD コマンドの種類については，「12.3 TD コマンドの一覧」を参照してください。

必須オペランド

TD コマンドの操作対象や詳細を指定します。必須オペランドは決められた順で必ず指定します。重複して指定することはできません。

任意オペランド

TD コマンドの操作対象や詳細を指定します。任意オペランドは省略できます。指定順序は任意です。

同一オペランドのグループに属し，効果が背反するオペランドを重複して指定した場合は，最後に指定したものが有効になります。

12.1.2　TD コマンドの行

一つの TD コマンドを，複数の行にわたって記述できます。1 行に記述できる TD コマンドおよび注釈の長さは，11,264 バイト以内です。

12.1.3　空白と括弧を含む文字列の指定方法

空白と括弧を文字列に使用する場合は，アポストロフィ（'）またはダブルコーテーション（"）で囲みます。アポストロフィ（'）またはダブルコーテーション（"）で，2 行にわたる文字列を囲むことはできません。アポストロフィ（'）またはダブルコーテーション（"）で囲んだ文字列を使用できる文字列には，次のものがあります。

- #INCLUDE コマンドのファイル名
- DISPLAY COMMENT コマンドのコメント文字列
- SET PRINT コマンドのファイル名

- SET LOG コマンドのファイル名
- GO コマンドの引数の文字列
- STEP IN コマンドの引数の文字列
- STEP OVER コマンドの引数の文字列
- 定数

アポストロフィ（'）で囲んだ文字列の中にアポストロフィ（'）を指定する場合は，連続した 2 文字のアポストロフィ（'）で指定します。ダブルコーテーション（"）で囲んだ文字列の中にダブルコーテーション（"）を指定する場合は，連続した 2 文字のダブルコーテーション（"）で指定します。

使用例

ファイル名 DON'T の TD コマンド格納ファイルを入力します。

```
#INCLUDE INFILE('DON''T')
```

12.1.4 英大文字と英小文字，英数字文字と拡張文字を区別しての文字列の指定方法

#OPTION コマンドが有効にならないように文字列を指定する場合は，アポストロフィ（'）またはダブルコーテーション（"）で囲みます。アポストロフィ（'）またはダブルコーテーション（"）で，2 行にわたる文字列を囲むことはできません。

アポストロフィ（'）またはダブルコーテーション（"）で囲んだ文字列を使用できる文字列は，「12.1.3 空白と括弧を含む文字列の指定方法」を参照してください。

12.1.5 注釈

「*>」から行の終わりまでを注釈とします。

使用例

```
*> プログラムの実行開始前の指定
*> 中断点を設定する
SET BREAK PROCEDURE(#PROGRAM(決算処理/決算処理) #ENTRY) *> プログラムの開始で中断
```

12.1.6 等価規則

TD コマンド名とオペランドの補助語で使用する#は，等価規則に関係なく必ず英数字文字で指定しなければなりません。その他のオペランドには，等価規則が適用されます。

等価規則の詳細については，「9.2.9　テストデバッグのための設定」の「(3)　入力する文字の扱い（等価規則）」を参照してください。

12.2.1　ソース要素の名称

(1)　外部プログラム名と内部プログラム名

外部プログラム名は，プログラム定義の最外のプログラムである外部プログラムのプログラム名段落（PROGRAM-ID）で指定した名称です。内部プログラム名は，内部プログラムのプログラム名段落（PROGRAM-ID）で指定した名称です。記述規則は，COBOL の言語仕様に従って，変換規則も適用します。変換後の名称を TD コマンドのオペランドに指定できます。

(2)　クラス名とメソッド名

クラス名は，クラス定義のクラス名段落（CLASS-ID）で指定した名称です。メソッド名は，メソッド定義のメソッド名段落（METHOD-ID）で指定した名称です。記述規則は，COBOL の言語仕様に従って，変換規則も適用します。変換後の名称を TD コマンドのオペランドに指定できます。

(3)　関数名

利用者による関数定義の関数名段落(FUNCTION-ID)で指定した名称です。記述規則は，COBOL の言語仕様に従って，変換規則も適用します。変換後の名称を TD コマンドのオペランドに指定できます。

12.2.2　行番号（＝一連番号）

原始プログラムの一連番号領域に記述してある番号です。原始プログラムの一連番号領域に番号の記述がない場合，順序が昇順でない場合，または自由形式正書法の場合は，コンパイラが自動的に振り直す番号が行番号となります。

12.2.3　文番号

COBOL 言語の文を指定する番号です。行番号と位置番号をピリオドでつなぎます。行の先頭文を位置番号の 1 とします。位置番号の 1 は省略できます。

使用例

```
001000  READ A-FILE AT END GO TO P2.
          1000/1000.1      1000.2
```

文番号1000および1000.1は，READ文を表します。
文番号1000.2は，GO TO文を表します。

12.2.4　入口名

外部プログラム名および ENTRY 文で指定される入口点の名前を，入口名とします。

12.2.5　手続き名

COBOL 言語の手続き名です。手続き名には，段落名と節名があります。手続き名も一つの文として文番号が割り当てられます。

使用例

```
001000  P1.       READ
        1000/1000.1
002000                A-FILE AT END GO TO P2.
```

文番号1000および1000.1は，手続きP1を表します。

12.2.6　データ名

テストデバッグで指定できる COBOL 言語のデータは次のとおりです。

- データ名
- 指標名
- アドレス名
- 特殊レジスタ

データ名は，COBOL 言語の仕様に準拠した形で，修飾，添字付けおよび部分参照の指定ができます。COBOL 言語との指定方法の違いは次のとおりです。

(1)　表の参照

表の属性を持つデータ名を参照するときは，添字を付けて指定します。添字は，整数定数，添字の付かない数字項目，指標名の指定ができます。ただし，データ監視条件では，整数定数でなければなりません。

次の項目は，添字に指定できません。

- ALL
- 算術式（相対添字付け）
- ほかの表に割り当てた指標名

また，整数定数を，ピリオドを 2 個（..）続けてつなぎ，添字に指定すると，表の一部の範囲を一度に参照できます。

(2) 部分参照

データの部分を参照するときは，最左端位置と長さを指定します。最左端位置と長さは，整数定数を指定します。部分参照には，次の制限があります。

- 最左端位置と長さに，算術式は指定できません。
- 添字付けの範囲指定と部分参照は同時に指定できません。

注意事項

- 報告書節のデータ項目および画面節（SCREEN SECTION）の画面名は参照できません。
- 最適化オプション（-Optimize,2）でコンパイルしたソース要素をテストデバッグするときは，最適化されたデータ項目を参照できないことがあります。
- TYPEDEF 句の付いたデータ項目およびその下位項目のデータ項目は参照できません。
- DISPLAY OBJECT コマンド，DISPLAY FACTORY コマンドによって，オブジェクトまたはファクトリオブジェクト内に定義されている表を参照するときの添字の指定は，整数定数でなければなりません。

12.2.7 定数

テストデバッガで使用できる定数には，COBOL の言語仕様に準拠した形式と，拡張 16 進定数があります。COBOL の言語仕様に準拠した形式では，次の定数を代入，比較で使用できます。

(1) COBOL 言語の仕様に準拠した定数

記号文字・定数名は使用できません。ALL は，英数字定数・ブール定数・日本語文字定数・表意定数に指定できます。

定数を囲む一対の区切り記号には，アポストロフィ（'）またはダブルコーテーション（"）が使用できます。区切り記号としてアポストロフィ（'）またはダブルコーテーション（"）を使用する場合の指定方法については「12.1 TD コマンドの指定方法」を参照してください。

表 12-1 COBOL 言語の仕様に準拠した定数一覧

定数の種類		例
英数字定数	英数字	'ABC'
	16 進英数字	X'0AE2'
数字定数	固定小数点数字	123 123.45
	浮動小数点数字	123.E+02
	16 進数字	H'B0EF'長さ 8 文字まで指定できます。 2 の倍数長である必要はありません。

定数の種類		例
ブール定数	ブール	B'1010'
日本語文字定数	日本語文字	N'日本語' , NC'日本語' , ND'日本語'
	16 進日本語文字	NX'81568134'
表意定数		ALL 'A', ALL N'日本語', ALL B'1010' , 〔ALL〕HIGH-VALUE, 〔ALL〕HIGH-VALUES, 〔ALL〕LOW-VALUE, 〔ALL〕LOW-VALUES, 〔ALL〕ZERO, 〔ALL〕ZEROS, 〔ALL〕ZEROES, 〔ALL〕SPACE, 〔ALL〕SPACES, 〔ALL〕QUOTE, 〔ALL〕QUOTES, 〔ALL〕NULL, 〔ALL〕NULLS

(2)　拡張 16 進定数

#X で始まり，一対のアポストロフィ（'）またはダブルコーテーション（"）で囲んだ 0～9，A～F の文字列です。文字数は，2 の倍数で，160 文字までの長さです。ALL による定数の繰り返しも指定できます。

拡張 16 進定数は，COBOL2002 の定数で代入，比較ができないデータ項目にも，代入，比較ができます。代入，比較の規則の詳細については，「9.3.1　データの比較・代入規則」の「(1)　データの代入規則」，「(2)　データの比較規則」を参照してください。

構文規則

```
〔ALL〕　#X’ ・・・・ ’
〔ALL〕　#X” ・・・・ ”
```

使用例

```
#X’313233’
ALL　#X’FF’
```

12.2.8　既定義オブジェクト名

テストデバッガで使用できる COBOL 言語の既定義オブジェクトは次のとおりです。

- SELF
- EXCEPTION-OBJECT

12.2.9　オペランドの補助語

オペランドの内容を指定するときに，補助語を使用します。補助語の先頭にはシャープ（#）が付きます。オペランドの補助語を次に示します。

表 12–2　オペランドの補助語

補助語	内容
#ENTRY	ソース要素の入口を指定します。
#EXIT	ソース要素の出口を指定します。
#PROGRAM	プログラム名を指定します。
#CLASS	クラス名を指定します。
#FACTORY	ファクトリオブジェクトを示します。
#OBJECT	インスタンスオブジェクトを示します。
#FUNCTION	利用者定義関数名を指定します。
#ALL	すべてという意味で指定します。
#>	参照するオブジェクトの特定のデータ名を示します。
#LASTCODE	直前の ASSIGN CASECODE コマンドで設定した値を示します。

12.2.10　TD 利用者定義語

利用者が任意に指定する語です。先頭が英字で始まる英数字で，長さは 31 バイト以内です。

表 12–3　TD 利用者定義語

種類	用途
カウンタ変数	TD コマンド群の実行回数をカウントします。
記号名	シミュレーションでデータ名を識別します。
監視識別子	データ条件中断の設定を識別します。
ケース識別子	テストケースを識別します。

TD 利用者定義語は，TD コマンドで使用する COBOL2002 の予約語（OF，IN，WITH など）と同じ名称を指定できません。

12.2.11　翻訳単位指定

翻訳単位を特定します。

翻訳単位指定が指定されていない場合は，次の順で翻訳単位指定を仮定します。SET QUALIFICATION コマンドの詳細については，「12.4.23　SET QUALIFICATION／RESET QUALIFICATION（翻訳単位・ソース要素の指定と解除）」の「(1)　SET QUALIFICATION（翻訳単位・ソース要素の指定）」を参照してください。

- プログラムの起動前
  1. 事前に実行された SET QUALIFICATION コマンドで指定した翻訳単位指定
  2. -Main コンパイラオプションでコンパイルされた翻訳単位
     -Main コンパイラオプションでコンパイルされた翻訳単位がない場合は，翻訳単位指定または事前に SET QUALIFICATION コマンドを実行しなければなりません。
- プログラムの中断状態
  1. 事前に実行された SET QUALIFICATION コマンドで指定した翻訳単位指定
  2. バックトレース実行中で，現在表示されている翻訳単位
  3. 現在中断している翻訳単位

(1)　プログラム指定

プログラムを特定するときに指定します。

形式

#PROGRAM（外部プログラム名）

(2)　クラス指定

クラスを特定するときに指定します。

形式

#CLASS（クラス名）

(3)　関数指定

関数を特定するときに指定します。

形式

#FUNCTION（関数名）

12.2.12　ソース要素指定

ソース要素を特定します。

ソース要素指定が指定されていない場合は，次の順でソース要素指定を仮定します。

SET QUALIFICATION コマンドの詳細については，「12.4.23　SET QUALIFICATION／RESET QUALIFICATION（翻訳単位・ソース要素の指定と解除）」の「(1)　SET QUALIFICATION（翻訳単位・ソース要素の指定）」を参照してください。

- プログラムの起動前
  1. 事前に実行された SET QUALIFICATION コマンドで指定したソース要素指定
  2. -Main コンパイラオプションでコンパイルされた翻訳単位の外部プログラム

     -Main コンパイラオプションでコンパイルされた翻訳単位がない場合は，ソース要素指定または事前に SET QUALIFICATION コマンドを指定しなければなりません。
- プログラムの中断状態
  1. 事前に実行された SET QUALIFICATION コマンドで指定したソース要素指定
  2. バックトレース実行中で，現在表示されているソース要素
  3. 中断状態のソース要素

プログラム定義のソース要素を指定します。

形式

#PROGRAM（{外部プログラム名/外部プログラム名 / 外部プログラム名/内部プログラム名}）

クラス定義のソース要素を特定します。

形式

#CLASS（{クラス名/#FACTORY / クラス名/#OBJECT / クラス名/メソッド名}）

関数を指定します。

形式

#FUNCTION（関数名）

12.2.13　文番号指定

文を文番号で指定します。

形式

```
{ 文番号
  翻訳単位指定 文番号 }
```

12.2.14　手続き名指定

手続き名を指定します。

形式

```
{ 手続き名
  ソース要素指定　手続き名 }
```

12.2.15　入口，出口指定

ソース要素の入口，出口を指定します。ソース要素の入口とはソース要素の最初の文を実行する前の状態です。ソース要素の出口とはソース要素の最後の文を実行したあとの状態です。

ソース要素の入口を#ENTRY，ソース要素の出口を#EXIT で表します。

形式

```
{ #ENTRY
  ソース要素指定 #ENTRY
  #EXIT
  ソース要素指定 #EXIT }
```

使用例

```
#ENTRY                  *> 入口
#PROGRAM(A/A)#ENTRY     *> 外部プログラムAの入口
#EXIT                   *> 出口
#PROGRAM(A/B)#EXIT      *> 外部プログラムAに属する
                        *> 内部プログラムBの出口
```

注意事項

ソース要素指定をするとき，次の指定はできません。

```
#CLASS(クラス名/#FACTORY)
#CLASS(クラス名/#OBJECT)
```

12.2.16　データ名指定

操作する対象となるデータを一意にするときに指定します。

形式

```
{ データ名
  ソース要素指定　データ名
  カウンタ変数
  記号名 }
```

12.2.17　ファイル名指定

COBOL 言語の原始プログラムのファイル記述項のファイル名を指定します。

形式

```
{ ファイル名
  ソース要素指定 ファイル名 }
```

12.2.18　TD コマンド群

一つにまとまった処理をするための TD コマンドの集まりを TD コマンド群として指定できます。TD コマンド群は，次の TD コマンドで指定できます。

- SET BREAK コマンド，SET WATCH コマンドで，中断時に実行するコマンド
- IF コマンドで，成立または不成立時に実行するコマンド
- シミュレーションの手続き
- TEST コマンドのテストケース

使用例

```
SET BREAK STATEMENT(100) DO
 DISPLAY DATA(品名)
 SET BREAK STATEMENT(500) DO  *> ネスト
   DISPLAY DATA(番号)
 ENDDO
ENDDO
```

データ監視条件の設定，または中断点の設定による中断が一つの文で同時に成立したときは，次の順番で TD コマンド群が実行されます。

1.「データ監視条件の設定」の TD コマンド群

2.「中断点の設定」の TD コマンド群

さらに，データ監視条件の設定によって，一つの文で複数のデータ監視条件が同時に成立したときは，監視識別子の昇順に TD コマンド群が実行されます。

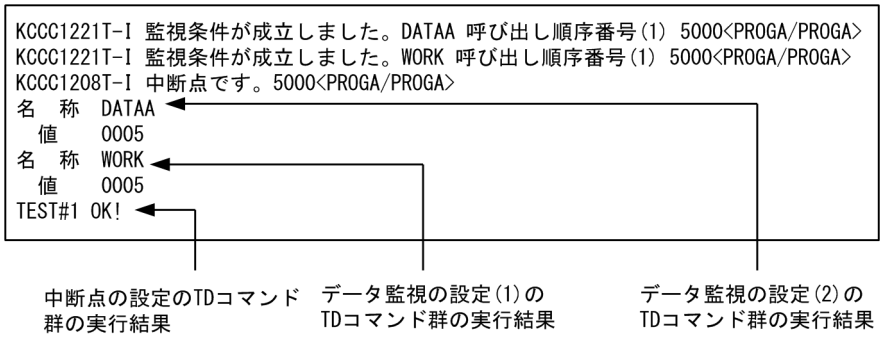
使用例

次の TD コマンドを実行し，一つの文で同時に中断が成立すると，「データ監視の設定(SET WATCH(DATAA))」「データ監視の設定(SET WATCH(WORK))」「中断点の設定」の順番で TD コマンド群が実行されます。

TD コマンド群

```
SET BREAK STATEMENT(5000)                *> 中断点の設定
DO
  DISPLAY COMMENT("TEST#1 OK!")
ENDDO
SET WATCH(WORK) CONDITION(WORK NOT = 0) *> データ監視の設定(1)
DO
  DISPLAY DATA(WORK)
ENDDO
SET WATCH(DATAA) CONDITION(DATAA NOT = 0) *> データ監視の設定(2)
DO
  DISPLAY DATA(DATAA)
ENDDO
```

実行結果



次の，プログラムを続行する TD コマンドを TD コマンド群に指定すると，プログラムの実行が再開されてしまうために，それ以降の TD コマンドは実行されません。TD コマンド群が複数ある場合，最初に実行される TD コマンド群にプログラムを続行する TD コマンドが指定されていたときは，それ以降の TD コマンド群は実行されません。

- GO
- STEP IN
- STEP OVER
- STOP

TD コマンド群が実行されなかったとき，カウンタ変数は加算されません。中断点を設定したときのスキップ回数のカウントは加算されます。

使用例

次の TD コマンドを実行した結果，文番号 6000 で比較条件式 A ＝ 50 が成立したとします。SET WATCH コマンドで指定された TD コマンド群が実行され，GO コマンドによってプログラムが続行されます。SET BREAK コマンドで指定された無条件中断の TD コマンド群の DISPLAY DATA コマンドは実行されません。カウンタ変数 C の値も変化しません。

```
SET WATCH(WATCH50)  CONDITION(A = 50)
DO
  GO
ENDDO

SET BREAK STATEMENT(6000) COUNTER(C)
DO
  DISPLAY DATA(B)
ENDDO
```

注意事項

- 「データ監視条件の設定」と「中断点の設定」が成立しても，TD コマンド群でプログラムを続行するコマンドが実行されたときは，入力待ちにならないで，実行が継続されます。
- ステップイン，ステップオーバーまたはジャンプで中断した文で，同時に「データ監視条件の設定」と「中断点の設定」が成立しても，TD コマンド群は一切実行しません。
- 中断点のスキップ回数は，TD コマンド群の実行には関係なく，中断点を通過した場合には必ず更新されます。
- 中断点に達した場合のメッセージおよびデータ監視条件が成立した場合のメッセージは，TD コマンド群の実行に関係なく表示されます。
- GUI モードでは，TD コマンド群の実行を強制的に中止できます。コマンド実行中に割り込みを指示した場合は，その時点で実行している TD コマンド以降のコマンドは実行されないで中断します。
- シミュレーション手続きの場合は，シミュレーション対象入出力文の次の文でプログラムが中断します。

12.2.19　カウンタ変数

カウンタ変数は，TD コマンド群の実行回数を参照するための変数です。カウンタ変数を定義した TD コマンド群の中でだけ参照できます。2 進項目のデータとして扱われ，1 から始まり，TD コマンド群が実行されるたびに 1 ずつ加算されます。最大値は 2,147,483,647 で，最大値を超えた場合は，最大値のまま更新されません。

カウンタ変数は，表示，代入，比較に使用できます。ただし，添字・部分参照には使用できません。

- 表示

  2 進項目のデータ属性，または 16 進で表示できます。
- 代入

送り出し側作用対象に指定して，データ項目へ値を代入できます。このとき，2 進項目の代入規則に従います。カウンタ変数へ値を代入することはできません。

代入については，「9.3.1　データの比較・代入規則」の「(1)　データの代入規則」を参照してください。

- 比較

  2 進項目の比較規則に従い，データ項目，定数と比較できます。

  比較については，「9.3.1　データの比較・代入規則」の「(2)　データの比較規則」を参照してください。

TD コマンド群が実行される場合，カウンタ変数と同じ名前のデータ名が参照範囲にあるときは，カウンタ変数が指定できるオペランドであれば，カウンタ変数とみなします。カウンタ変数が指定できないオペランドであれば，データ名とみなします。

使用例

カウンタ変数によって，実行される TD コマンドを変えます。

```
SET QUALIFICATION (#PROG (給与計算))
SET BREAK  ST(1350)  COUNTER(CNT)
DO
    IF  C (CNT = 1)
      DISPLAY DATA(合計)                *> 一度目に実行される
    ENDIF
    IF  C (CNT = 2)
        DISPLAY DATA(総合計)            *> 二度目に実行される
    ENDIF
ENDDO
```

12.2.20　記号名

記号名は，副プログラムシミュレーション・ファイルシミュレーションのために定義する名前です。記号名については，「9.2.7　プログラムの単体テスト」の「(4)　記号名」を参照してください。

カウンタ変数と同じ名前は指定できません。ソース要素中のデータ名と同じ名前を記号名として定義したときは，記号名とみなします。記号名に対応するデータ名が表の場合は，添字を必要とします。部分参照できるデータ名であれば，部分参照の指定ができます。

使用例

ファイルシミュレーションのレコード定義に記号名を割り当てます。

```
    SIMULATE FILE(OUTPUT-FILE) OPENMODE(OUTPUT) RECORD(SYMBOL1)
       DEFINE
        01 SYMBOL1
          02 SYMBOL11
           03 SYMBOL111
       ENDDEFINE
          DISPLAY DATA (SYMBOL111(1, J)(1:5))
    ENDSIMULATE
```

12.2.21　繰り返し指定

次のシミュレーションで，反復回数と TD コマンド群を指定します。「12.4.17　REPEAT（繰り返し指定）」を参照してください。

- 副プログラムシミュレーションで副プログラムが呼ばれる
- ファイルシミュレーション入出力文が実行される
- DC シミュレーションで DC 文が実行される

繰り返し指定の使用例については，「9.2.7　プログラムの単体テスト」の「(3)　ファイルシミュレーション」の「(a)　ファイル入出力文のシミュレーション」の使用例 2 を参照してください。

繰り返し指定を複数記述することで，シミュレーションが行われるごとに実行される TD コマンド群を変更できます。反復回数分のシミュレーションで TD コマンドを実行されると，次のシミュレーションから次の繰り返し指定の TD コマンド群が実行されます。すべての繰り返し指定を実行したあとに，シミュレーションが実行されるときは，最後の繰り返し指定の TD コマンド群を実行します。

12.2.22　入出力文選択指定

シミュレーションする次の入出力文に対応させて，ファイルシミュレーションの手続きを実行させます。「12.4.18　SELECT ACTION（入出力文選択指定）」を参照してください。

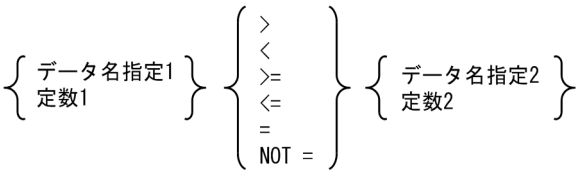
使用例については「9.2.7　プログラムの単体テスト」の「(3)　ファイルシミュレーション」の「(a)　ファイル入出力文のシミュレーション」の使用例 3 を参照してください。

- READ
- WRITE
- REWRITE
- START
- DELETE

12.2.23　比較条件式

条件を指定します。

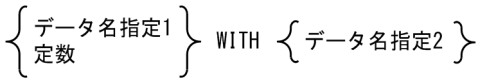
形式



12.2.24　OLE プロパティ指定

OLE プロパティを特定します。

形式



オペランドの詳細

- データ名指定 1

  OLE プロパティの名称が設定された英字項目，英数字項目，日本語項目または集団項目を指定します。

- 定数

  OLE プロパティの名称を指定する英数字定数，または日本語定数を指定します。

- データ名指定 2

  OLE オブジェクト参照データ項目のデータ名を指定します。

注意事項

データ名指定 1 およびデータ名指定 2 には，添字の範囲指定はできません。

使用例

```
'Caption' WITH OLE-OBJ1
     *> OLE-OBJ1のOLEオブジェクトに属するOLEプロパティ'Caption'
PROP1 WITH OLE-OBJ2
      *> OLE-OBJ2のOLEオブジェクトに属し
      *> データ名PROP1に設定された名称のOLEプロパティ
```

12.3 TD コマンドの一覧

12.3.1 プログラムの実行の制御と追跡

テストプログラムの実行を制御したり，実行状態を追跡したりする TD コマンドの一覧を次に示します。

表 12–4 実行を制御したり，実行状態を追跡したりする TD コマンドの一覧

目的	コマンド名
プログラムの実行を中断させる中断点を設定する。	SET BREAK
SET BREAK コマンドで設定した中断点を解除する。	RESET BREAK
設定されている中断点の一覧を表示する。	DISPLAY BREAK
プログラムの実行を中断する条件を設定する。	SET WATCH
SET WATCH コマンドで設定したデータ監視条件を解除する。	RESET WATCH
プログラムが起動前状態のときは，実行を開始する。中断状態のときは，実行を再開する。	GO
プログラムを 1 文実行して中断する。	STEP IN
プログラムを 1 文実行して中断する。ただし，CALL 文，関数呼び出しを持つ文および INVOKE 文は 1 文として実行する。	STEP OVER
プログラムの中断時に，実行を再開する位置を変更する。	STEP TO
実行中のプログラムを強制的に終了する。	STOP
プログラムの実行を追跡し，指定した単位で通過点を表示する。	SET TRACE
SET TRACE コマンドで開始したトレース表示を中止する。	RESET TRACE
フロー情報の蓄積を開始する。	SET FLOW
フロー情報の蓄積を中止する。	RESET FLOW
蓄積されているフロー情報を表示する。	DISPLAY FLOW
プログラムが中断している現在位置を表示する。	DISPLAY POINT

12.3.2 データの操作

テストプログラムで扱われているデータの値を表示したり，任意の値を代入したりする TD コマンドの一覧を次に示します。

表 12–5　データの値を表示したり，任意の値を代入したりする TD コマンドの一覧

目的	コマンド名
データの値を指定された形式で表示する。	DISPLAY DATA
オブジェクト参照データ項目または既定義オブジェクト SELF，EXCEPTION-OBJECT の指定によって，その参照するオブジェクトのオブジェクト参照の値，クラス名，インスタンスオブジェクト・ファクトリオブジェクトの種別，データの値を表示する。DISPLAY FACTORY コマンドは，クラス名の指定によって，ファクトリオブジェクトのデータの値を表示する。	DISPLAY OBJECT DISPLAY FACTORY
OLE2 の OLE オブジェクトが持つ OLE プロパティの値を表示する。	GETPROPERTY OLE
データに値を代入する。	ASSIGN DATA
条件が成立したときは，ELSE の前の TD コマンド群を実行する。 条件が成立しないときは，ELSE のあとの TD コマンド群を実行する。	IF
領域を確保して，アドレス名にアドレスを設定する。	ALLOCATE AREA
アドレス名の示す領域を解放する。	FREE AREA
アドレスを取得する。	ASSIGN ADDRESS

12.3.3　単体テスト

未完成のプログラムを単体でテストする TD コマンドの一覧を次に示します。

表 12–6　単体でテストする TD コマンドの一覧

目的	コマンド名
主プログラムシミュレーションで実行する手続きを設定する。	SIMULATE MAIN
副プログラムシミュレーションで実行する手続きを設定する。	SIMULATE SUB
ファイルのシミュレーションで実行する手続きを設定する。	SIMULATE FILE
入出力条件を擬似的に発生させる。	GO END GO EOP GO INVALID GO ERROR
記号名に対応するデータ名が集団項目のとき，集団項目に属するデータ名に対応する記号名を指定する。	レベル番号
シミュレーションで一度に実行する手続きと，繰り返す回数を指定する。	REPEAT
TD コマンドで手続きを設定して DC シミュレーションを実行する。	SIMULATE DC

12.3.4　バッチによるテスト

バッチによるテストで使用する TD コマンドの一覧を次に示します。

表 12–7　バッチによるテストで使用する TD コマンドの一覧

目的	コマンド名
複数のテストケースをまとめて定義して，その中から実行するテストケースを指定する。	TEST
テストケースの手続きを設定する。	CASE
テストケースの終了状態（ケースコード）を設定する。	ASSIGN CASECODE
テストデバッグを終了する。	QUIT
指定した文字を表示する。	DISPLAY COMMENT

12.3.5　テストデバッグの環境設定

入力する文字の扱いを変更したり，指定したファイルから TD コマンドを入力したりする TD コマンドの一覧を次に示します。

表 12–8　入力する文字の扱いを変更したり，指定したファイルから TD コマンドを入力したりする TD コマンドの一覧

目的	コマンド名
テストデバッグの対象とする翻訳単位またはソース要素を変更する。	SET QUALIFICATION
翻訳単位指定またはソース要素指定を解除する。	RESET QUALIFICATION
実行結果の出力先を指定する。	SET PRINT
実行結果の出力先の指定を解除する。	RESET PRINT
モニタウィンドウへ出力されるテストデバッグの実行結果をファイルへ出力する。	SET LOG
テストデバッグの実行結果をモニタウィンドウへ出力するように戻す。	RESET LOG
入力する文字の扱い（等価規則）を変更する。	#OPTION
指定したファイルから TD コマンドを入力する。	#INCLUDE

12.4 TD コマンドの詳細

12.4.1 ALLOCATE AREA／FREE AREA（領域の確保と解放）

(1) ALLOCATE AREA（領域の確保）

アドレス名によって参照されるデータ項目と同じサイズの領域を確保して，アドレス名にアドレスを設定します。

指定されたサイズの領域を確保して，アドレスデータ項目またはポインタ項目にアドレスを設定します。

形式

ALLOCATE AREA（ データ名指定 1 ） LENGTH （ 領域長 ）

ALLOCATE AREA（ データ名指定 2 ）

- AREA（データ名指定 1）

  ポインタ項目またはアドレスデータ項目を指定します。

  LENGTH オペランドに指定されたサイズを確保して，指定されたポインタ項目またはアドレスデータ項目に，確保した領域のアドレスを設定します。

- LENGTH（領域長）

  指定された領域長分の領域を確保します。指定できる長さは，1～2,147,483,647 バイトです。なお，データ名を使った指定はできません。

- AREA（データ名指定 2）

  アドレス名を指定します。アドレス名によって参照されるデータ項目と同じサイズの領域を確保して，指定されたアドレス名に確保した領域のアドレスを設定します。

注意事項

- 確保した領域は，X'00'で初期化されます。
- 確保した領域は，次の場合に解放されます。

  ・FREE AREA コマンドを実行したとき。

  ・プログラムの実行が終了したとき。
- マルチスレッドプログラムでは使用できません。
- LENGTH オペランドを指定して確保した領域を使用する場合に，確保した領域を超える部分にアクセスしたときの動作は保証しません。

使用例

領域破壊を起こす例を示します。

次の条件でテストした場合，ADDRESS_NAME には英数字 3 文字分の領域にしか割り当たりません。

DATA2 へ代入するのは'ABC'の 3 文字ですが，定義は 10 文字なので残りの 7 文字分の埋め字処理で領域破壊が起きます。

・COBOL ソースでのデータ定義

```
01 DATA1 ADDRESSED BY ADDRESS_NAME.
  02 DATA2 PIC X(10).
01 ADDRESS_DATA USAGE ADDRESS.
```

・デバッガで実行する TD コマンド

```
ALLOCATE AREA( ADDRESS_DATA ) LENGTH( 3 )
ASSIGN DATA( ADDRESS_NAME ) VALUE( ADDRESS_DATA )
ASSIGN DATA( DATA2 ) VALUE( 'ABC' ) *> 領域破壊を起こす。
```

(2) FREE AREA（領域の解放）

アドレス名の示す領域を解放します。

アドレスデータ項目またはポインタ項目も指定できます。

形式

```
FREE AREA（ データ名指定 ）
```

- AREA（データ名指定）

  確保した領域を解放するアドレス名を指定します。

  または，確保した領域を解放するアドレスデータ項目，もしくはポインタ項目を指定します。

注意事項

- ALLOCATE AREA コマンドで確保された領域以外は解放できません。
- 領域の解放後，アドレス名に 0 を設定します。アドレスデータ項目またはポインタ項目を指定した場合も同様です。
- マルチスレッドプログラムでは使用できません。

12.4.2 ASSIGN ADDRESS（アドレスの取得）

データ名のアドレスを取得します。

形式

```
ASSIGN ADDRESS( データ名指定1 ) DATA( データ名指定2 )
```

- ADDRESS（データ名指定 1）

  アドレス名，アドレスデータ項目，またはポインタ項目を指定します。

  指定したデータ名に，DATA オペランドで指定したデータ名のアドレスが設定されます。

- DATA（データ名指定 2）

アドレスを取得したいデータ名を指定します。

注意事項

- データ名指定 2 には，次のデータ項目は指定できません。
  ・指標名
  ・アドレス名
  ・動的長基本項目
- データ名指定 2 にブール項目を指定した場合，バイト境界になっていればアドレスを取得できます。バイト境界になっていない場合は，エラーになります。
- アドレスを取得したあとは，取得したアドレスが指すデータが参照できるかどうかは確認しません。参照できなくなっていた場合，そのアドレスを使ってデータ操作したときの動作は保証しません。
- データ名指定 1 にアドレス名を指定したとき，指定したアドレス名で参照されるデータとデータ名指定 2 に指定したデータの構造のチェックはしません。二つの構造が違うことによる領域破壊などの動作は保証しません。

使用例

領域破壊を起こす例を示します。

次の条件で，データ名指定 1 に ADDRS，データ名指定 2 に SRC_DATA1 を指定した場合，DATA1（アドレス名 ADDRS で参照されるデータ）と SRC_DATA1 は，構造およびサイズとも異なりますが，ASSIGN ADDRESS コマンドは成功します。

ただし，DATA3 OF DATA1 の部分は SRC_DATA1 の範囲外になるため，その値を更新すると，領域破壊になります。

・COBOL ソースでのデータ定義

```
01 DATA1 ADDRESSED BY ADDRS.
  02 DATA2 PIC X(10).
  02 DATA3 PIC X(10).
01 SRC_DATA1 PIC X(10).
```

・デバッガで実行する TD コマンド

```
ASSIGN ADDRESS( ADDRS ) DATA( SRC_DATA1 )
ASSIGN DATA( DATA3 OF DATA1) VALUE( ”SAMPLE” ) *> 領域破壊になる。
```

- データ名指定 2 にアドレス名で参照されたデータ名を指定した場合，アドレス名が指すアドレスが設定されます。

12.4.3 ASSIGN CASECODE（ケースコードの設定）

テストケースの終了状態（ケースコード）を設定します。

形式

ASSIGN CASECODE（ ケースコード ）

- CASECODE(ケースコード)
  指定できる範囲は，0～2,147,483,647 の整数です。

12.4.4 ASSIGN DATA（データの値代入）

データに値を代入します。

形式

```
ASSIGN DATA（ データ名指定1 ）VALUE（ データ名指定2 | 定数 ）
```

- DATA（データ名指定 1）
  値が代入されるデータ名を指定します。
- VALUE（データ名指定 2）
  代入する値を持つデータ名を指定します。
- VALUE（定数）
  代入する定数を指定します。

注意事項

- データ名指定には，添字の範囲指定はできません。
- カウンタ変数を指定できるのは，データ名指定 2 だけです。

12.4.5 CASE（テストケースの指定）

テストケースの手続きを設定します。

形式

```
CASE 〔 ID（ ケース識別子 ）〕
     〔 TOSTEP（ ステップ数 ）〕
     〔 CONDITION（ ケース選択条件〔 AND … 〕）〕
     〔 TDコマンド群 〕
ENDCASE
```

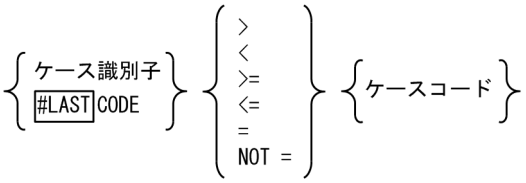
- ID（ケース識別子）
  テストケースの名称をケース識別子として定義します。英字で始まる 31 文字以内の英数字文字列で指定します。
- TOSTEP（ステップ数）
  プログラムの実行を開始してから終了するまでのステップ数を指定します。最初の文を実行したときのステップ数を 1 とします。

指定できる範囲は，1～2,147,483,647 の整数です。実行した文が指定された数に達すると，実行が打ち切られます。

- CONDITION（ケース選択条件 〔AND … 〕）

  テストケースを実行する条件を指定します。指定したすべての条件を満たしたときにテストケースが実行されます。

  ケース選択条件は，次の条件式で指定します。



  ケース識別子は，先に実行したテストケースのケース識別子を指定します。#LASTCODE を指定した場合は，直前のテストケースが仮定されます。

  ケースコードは，先行するテストケース内で ASSIGN CASECODE コマンドを使用して設定した値です。

- TD コマンド群

  テスト手続きを TD コマンドで指定します。

注意事項

- ステップ数は，スレッドに依存しないで，実行した文をカウントします。

12.4.6 DISPLAY BREAK（中断点の表示）

設定されている中断点の一覧を表示します。

形式



- BREAK(#ALL | 翻訳単位指定)

  #ALL を指定すると，すべての中断点を表示します。翻訳単位指定を指定すると，指定した翻訳単位の中断点を表示します。

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除)」の「(1)　SET PRINT（テスト結果蓄積先の設定)」を参照してください。

使用例と表示形式

使用例 1

プログラム PROGRAM_A の入口に中断点が設定されています。

```
DISPLAY BREAK(#ALL)
```

```
***** 中断点一覧 *****
#ENTRY <PROGRAM_A/PROGRAM_A> スキップ回数（0）
```

使用例 2

次の中断点が設定されています。

- プログラム PROGRAM_B の入口・出口
- 内部プログラム SUB1 の入口・出口
- プログラム PROGRAM_B の行番号 1000 の文
- 内部プログラムの手続き名 P1（行番号 1600 の文）・行番号 1800 の文

```
DISPLAY BREAK(#PROGRAM(PROGRAM_B))
```

```
***** 中断点一覧 *****
#ENTRY <PROGRAM_B/PROGRAM_B> スキップ回数（0）
1000 <PROGRAM_B/PROGRAM_B> スキップ回数（0）
#ENTRY <PROGRAM_B/SUB1> スキップ回数（0）
1600 P1 <PROGRAM_B/SUB1> スキップ回数（0）
1800 <PROGRAM_B/SUB1> スキップ回数（1）
#EXIT <PROGRAM_B/SUB1> スキップ回数（0）
#EXIT <PROGRAM_B/PROGRAM_B> スキップ回数（0）
```

12.4.7 DISPLAY COMMENT（注釈の表示）

指定した文字を表示します。

形式

DISPLAY COMMENT（ コメント文字列 ）〔 PRINT 〕

- COMMENT（コメント文字列）
  表示する文字列を指定します。文字列に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1 TD コマンドの指定方法」を参照してください。
- PRINT
  SET PRINT コマンドで指定した出力先に結果を表示します。

12.4.8 DISPLAY DATA（データの値表示）

データの値を指定された形式で表示します。バリアントデータ項目は，VARIANT 値も表示されます。

形式

```
DISPLAY DATA（ データ名指定 | #ALL ）
        〔 ATTRIBUTE | HEX | ALLTYPE 〕
        〔 GROUP | ELEMENT           〕
        〔 PRINT                     〕
```

- DATA（データ名指定 | #ALL）

  表示するデータを指定します。

  ・データ名指定：データ名を指定します。

  ・#ALL：現在中断している翻訳単位で参照できるすべてのデータの値を表示します。

- ATTRIBUTE | HEX | ALLTYPE

  データの表示形式を指定します。

  ・ATTRIBUTE：データ名の属性で表示します。

  ・HEX：16 進で表示します。

  ・ALLTYPE：ATTRIBUTE・HEX の両方の形式で表示します。

- GROUP | ELEMENT

  集団項目の表示形式を指定します。

  ・GROUP：英数字集団項目を英数字項目とみなし全体を先頭から表示します。または，日本語集団項目を日本語項目とみなし全体を先頭から表示します。

  ・ELEMENT：基本項目単位に表示します。

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」の「(1)　SET PRINT（テスト結果蓄積先の設定）」を参照してください。

注意事項

- GROUP オペランド，および ELEMENT オペランドは，データ名が集団項目でないときは無効となります。
- #ALL を指定したとき，ELEMENT オペランドは無効となります。集団項目は GROUP オペランドの形式で表示されます。
- ELEMENT オペランドは，集団項目を部分参照で表示するときは無効となります。
- HEX オペランドおよび ALLTYPE オペランドは，データ名が内部ブール項目のときは無効となり，ATTRIBUTE オペランド指定時同様に 0 と 1 から成る文字列が表示されます。内部ブール項目が集団項目の下位項目として存在し，集団項目を ELEMENT 指定で表示するときも HEX オペランドおよび ALLTYPE オペランドは無効となります。
- データの値がデータの属性で表示できないときは，「エラー」と表示した上で，値を 16 進数で表示します。
- DATA（#ALL）を指定したときは，特殊レジスタ RETURN-CODE と COBOL2002 の原始プログラムの手続き部で参照されている特殊レジスタも表示されます。

- ELEMENT オペランドが有効なとき，集団項目の再命名項目※に対する表示は名称だけになります。

  注※　集団項目の再命名項目を含む集団項目も該当します。

使用例と表示形式

使用例 1 について，Windows(x86) COBOL2002 の場合と Windows(x64) COBOL2002 の場合の表示例を示します。使用例 2 以降で，明記されていない場合については，Windows(x86) COBOL2002 の場合だけを示します。

Windows(x86) COBOL2002 の場合と Windows(x64) COBOL2002 の場合の相違は，16 進表示個所でアドレスを表示している部分が 8 けたから 16 けたになっていることです。また，表示するデータによってはデータサイズが変更になるものもあります。変更になるデータは，マニュアル「COBOL2002　ユーザーズガイド」の「アドレス系データを表現するデータ項目」を参照してください。

使用例 1

基本項目を属性と 16 進で表示します。

```
DISPLAY DATA(基本項目01)　ALLTYPE
```

Windows(x86) COBOL2002の場合

```
名　称　　基本項目01
　値　　 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789
          0123456789
*001E8FAC   30313233  34353637  38393031   32333435  36373839  30313233  34353637  38393031
*001E8FCC   32333435  36373839  30313233   34353637  38393031  32333435  36373839  30313233
*001E8FEC   34353637  38393031  32333435   36373839  30313233  34353637  3839
```

Windows(x64) COBOL2002の場合

```
名　称　　基本項目01
　値　　 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789
          0123456789
*00011000001E8FAC     30313233  34353637  38393031  32333435  36373839  30313233  34353637  38393031
*00011000001E8FCC     32333435  36373839  30313233  34353637  38393031  32333435  36373839  30313233
*00011000001E8FEC     34353637  38393031  32333435  36373839  30313233  34353637  3839
```

使用例 2

表の要素の 1～20 番目を，属性と 16 進で表示します。

```
DISPLAY DATA (表基本03-1 (1..20))　ALLTYPE
```

- すべての要素が正しく表示された。

```
名　称　　表基本03-1(1..9)
　値　　 +0111    +0222    +0333    +0444    +0555    +0666    +0777    +0888    +0999
*00405F30  00111C  *00405F33  00222C  *00405F36  00333C  *00405F39  00444C
*00405F3C  00555C  *00405F3F  00666C  *00405F42  00777C  *00405F45  00888C
*00405F48  00999C
名　称　　表基本03-1(10..18)
　値　　 +0000    +1111    +1222    +1333    +1444    +1555    +1666    +1777    +1888
*00405F4B  00000C  *00405F4E  01111C  *00405F51  01222C  *00405F54  01333C
*00405F57  01444C  *00405F5A  01555C  *00405F5D  01666C  *00405F60  01777C
*00405F63  01888C
名　称　　表基本03-1(19..20)
　値　　 +1999    +1000
*00405F66  01999C  *00405F69  01000C
```

- 4～6 番目と 8 番目の要素が，データ属性で表示できない値であった。

```
名　称      表基本03-1(1..3)
  値        +0111     +0222      +0333
*00405FC0  00111C     *00405FC3    00222C     *00405FC6     00333C
名　称      表基本03-1(4..6)
エラー
*00405FC9  004441     *00405FCC    414141     *00405FCF     414120
名　称      表基本03-1(7)
  値         +0777
*00405FD2   00777C
名　称      表基本03-1(8)
エラー
*00405FD5   004141
名　称      表基本03-1(9..17)
  値        +0999      +0000    +1111    +1222    +1333    +1444    +1555    +1666    +1777
*00405FD8  00999C      *00405FDB    00000C     *00405FDE     01111C     *00405FE1    01222C
*00405FE4  01333C      *00405FE7    01444C     *00405FEA     01555C     *00405FED    01666C
*00405FF0   01777C
名　称      表基本03-1(18..20)
  値        +1888      +1999       +1000
*00405FF3  01888C     *00405FF6      01999C      *00405FF9     01000C
```

使用例 3～使用例 6 の TD コマンドが表示する COBOL プログラムのデータ定義

```
01 集団01.
  02 基本02-1 PIC X(5) VALUE ALL 'A'.
  02 集団02-1.
    03 基本03-1  PIC X(5) VALUE ALL 'B'.
    03 集団03-1.
      04 基本04-1  PIC X(5) VALUE ALL 'C'.
      04           PIC X(5) VALUE ALL 'D'.
    03 基本03-2  PIC X(5) VALUE ALL 'E'.
  02 集団02-2.
    03 集団03-2 OCCURS 10.
      04 基本03-3  PIC 9(2) VALUE ZERO.
  02 基本02-2 PIC X(5) VALUE ALL 'F'.
```

使用例 3

集団項目を GROUP 指定で表示します。全体を英数字項目として表示されます。GROUP オペランドは省略できます。

```
DISPLAY DATA（集団01）  GROUP  ATTRIBUTE
```

```
名　称  集団01
  値    AAAAABBBBBCCCCCDDDDDEEEEE00000000000000000000FFFFF
```

使用例 4

集団項目を 16 進数で表示します。全体を英数字項目とした上で，16 進数で表示されます。

```
DISPLAY DATA（集団01）  HEX
```

```
名　称      集団01
*001E8FAB  41414141  41424242  42424343  43434344  44444444  45454545  45303030  30303030
*001E8FCB  30303030  30303030  30303030  30464646  4646
```

使用例 5

集団項目を，属性と 16 進で表示します。

```
DISPLAY DATA（集団01）  ALLTYPE
```

```
名　称      集団01
  値        AAAAABBBBBCCCCCDDDDDEEEEE00000000000000000000FFFFF
 *001E8FAB  41414141  41424242  42424343  43434344  44444444  45454545  45303030  30303030
 *001E8FCB  30303030  30303030  30303030  30464646  4646
```

使用例 6

集団項目を指定して，集団項目の所属するすべての基本項目を，属性と 16 進で表示します。

```
DISPLAY DATA (集団01)   ELEMENT ALLTYPE
```

```
名　称       集団01
名　称       基本02-1
  値         AAAAA
*001E8FAB  41414141  41
名　称       集団02-1
名　称       基本03-1
  値         BBBBB
*001E8FAB  42424242  42
名　称       集団03-1
名　称       基本04-1
  値         CCCCC
*001E8FAB  43434343  43
名　称       FILLER
  値         DDDDD
*001E8FAB  44444444  44
名　称       基本03-2
  値         EEEEE
*001E8FAB  45454545  45
名　称       集団02-2
名　称       集団03-2
名　称       基本03-3(1..10)
  値         00    00    00    00    00    00    00    00    00    00
*004036C9  3030  *004036CB  3030  *004036CD  3030  *004036CF  3030
*004036D1  3030  *004036D3  3030  *004036D5  3030  *004036D7  3030
*004036D9  3030  *004036DB  3030

名　称       基本02-2
  値         FFFFF
*001E8FAB  46464646  46
```

使用例 7

バリアントデータ項目の表の要素の 1～3 番目を，属性と 16 進で表示します。

```
DISPLAY DATA (VARIANT-DATA(1..3))   ALLTYPE
```

```
バリアントデータ名 :  VARIANT-DATA(1)
  値       +000112345
*004037A8  0001B6D9
  型       整数型 : 2
  値       123456
*--------  0001E240
バリアントデータ名 :  VARIANT-DATA(2)
  値       +000112678
*004037AC  0001B826
  型       EMPTY : 0
バリアントデータ名 :  VARIANT-DATA(3)
  値       +000113890
*004037B0  0001BCE2
  型       その他 : X'000c'
*--------  00000000  0014DEFC
```

バリアントデータ項目の値

バリアントデータ項目が参照するバリアント値

使用例 8

カウンタ変数を基本項目と属性と 16 進で表示します。

```
DISPLAY DATA (Counter01)   ALLTYPE
```

```
名　称       #Counter01
  値         1
*--------    01000000
```

使用例 9

記号名を基本項目と属性と 16 進で表示します。

```
DISPLAY DATA (Return01)   ALLTYPE
```

```
名　称<記号名>      Return01
  値         0
*001E8FAC  0000
```

使用例 10

内部プログラムで中断したときに，すべてのデータを表示します。最外レベル(01,77)単位に表示されます。

```
DISPLAY DATA (#ALL)   ALLTYPE
```

```
プログラム名 : COBOL2002_P2/INTER_P2
名　称       RETURN-CODE
  値        +000000000
*00403884    00000000                                          RETURN-CODEと手続き部で
          :                                                    参照される特殊レジスタが
名　称  LOCALデータ項目                                        表示される
  値         abcdefjhijabcdefjhijabcdefjhijabcdefjhijabcdefjhijabcdefjh
             ijabcdefjhijabcdefjhijabcdefjhij
*00403BB0    61626364  65666A68  696A6162  63646566  6A68696A  61626364  65666A68  696A6162
*00403BD0    63646566  6A68696A  61626364  65666A68  696A6162  63646566  6A68696A  61626364
*00403BF0    65666A68  696A6162  63646566  6A68696A  61626364  65666A68  696A6162  63646566
*00403C10     6A68696A


プログラム名 : COBOL2002_P2/COBOL2002_P2
名　称       RETURN-CODE
  値        +000000000
*004034A4    00000000                                          RETURN-CODEと手続き部で
          :                                                    参照される特殊レジスタ
                                                               が表示される
名　称  GLOBALデータ項目
  値         012345678901234567890123456789012345678901234567890123456789012345678901234567
             8901234567890123456789012345678 9
*00403B40    30313233  34353637  38393031  32333435  36373839  30313233  34353637  38393031
*00403B60    32333435  36373839  30313233  34353637  38393031  32333435  36373839  30313233
*00403B80    34353637  38393031  32333435  36373839  30313233  34353637  38393031  32333435
*00403BA0    36373839   30313233   34353637  3839
```

使用例 11

データ名の整数 01，整数 02 を 16 進数で表示されます。

```
DISPLAY DATA (整数01)   HEX
DISPLAY DATA (整数02)   HEX
```

```
名　称       整数01
*001E8FAC    67452301
名　称       整数02
*001E8FB4    0D0C0B0A
```

12.4.9　DISPLAY FLOW（フロー情報の表示）

蓄積されているフロー情報を表示します。フロー情報とは，プログラムの実行経路を示す通過点の情報です。フロー情報は，最後に通過した文から，プログラムの実行順と逆の順序に表示されます。

形式

```
DISPLAY FLOW 〔 PRINT 〕
```

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除)」の「(1)　SET PRINT（テスト結果蓄積先の設定)」を参照してください。

表示例

ソース要素の表示形式については，「9.5　ソース要素・中断点・通過点の表示形式」を参照してください。なお，マルチスレッドプログラムでないときは，スレッド ID は表示されません。

```
>>>>> 手続きフロー情報
スレッドID(123) #EXIT <飲み物管理/飲み物管理>
スレッドID(123) #EXIT <牛乳/牛乳>
スレッドID(123) #EXIT <牛乳/日付管理>
スレッドID(123) 41700 USE-CL1 OF SITUATION-1 <牛乳/日付管理>
スレッドID(123) 41200 USE-IN1 OF SITUATION-1 <牛乳/日付管理>
スレッドID(123) #EXIT <牛乳/製造管理>
スレッドID(123) 52600 USE-S-CL1 <牛乳/製造管理>
スレッドID(123) 52100 USE-S-IN1 <牛乳/製造管理>
スレッドID(123) #ENTRY <牛乳/製造管理 >
スレッドID(123) 40600 MAIN-START OF SITUATION-1 <牛乳/日付管理>
スレッドID(123) 40500 SITUATION-1 <牛乳/日付管理>
スレッドID(123) 40300 主処理 <牛乳/日付管理>
スレッドID(123) #ENTRY <牛乳/日付管理>
スレッドID(123) #ENTRY <牛乳/牛乳>
スレッドID(123) 4100 PARAM-GET <飲み物管理/飲み物管理>
スレッドID(123) 4000 主処理 <飲み物管理/飲み物管理>
スレッドID(123) #ENTRY <飲み物管理/飲み物管理>
<<<<<

>>>>> 文フロー情報

スレッドID(123) #EXIT <飲み物管理/飲み物管理>
スレッドID(123) 6300 <飲み物管理/飲み物管理>
スレッドID(123) #EXIT <牛乳/牛乳>
スレッドID(123) 20800 <牛乳/牛乳>
スレッドID(123) 20200 <牛乳/牛乳>
スレッドID(123) 20100 <牛乳/牛乳>
スレッドID(123) #ENTRY <牛乳/牛乳>
スレッドID(123) 4200 <飲み物管理/飲み物管理>
スレッドID(123) 4100 PARAM-GET <飲み物管理/飲み物管理>
スレッドID(123) 4000 主処理 <飲み物管理/飲み物管理>
スレッドID(123) #ENTRY <飲み物管理/飲み物管理>

<<<<<

>>>>> プログラムフロー情報

スレッドID(123) #EXIT <飲み物管理/飲み物管理>
スレッドID(123) #EXIT <牛乳/牛乳>
スレッドID(123) #EXIT <牛乳/日付管理>
スレッドID(123) #EXIT <牛乳/製造管理>
スレッドID(123) #ENTRY <牛乳/製造管理 >
スレッドID(123) #ENTRY <牛乳/日付管理>
スレッドID(123) #ENTRY <牛乳/牛乳>
スレッドID(123) #ENTRY <飲み物管理/飲み物管理>

<<<<<
```

12.4.10 DISPLAY OBJECT／DISPLAY FACTORY（オブジェクトのデータ値の表示）

DISPLAY OBJECT コマンドは，オブジェクト参照データ項目または既定義オブジェクト SELF，EXCEPTION-OBJECT の指定によって，その参照するオブジェクトの次の情報を表示します。

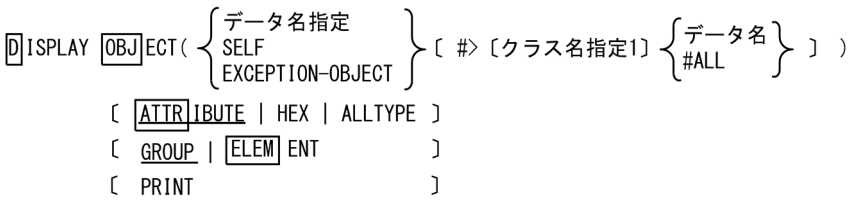
- オブジェクト参照の値
- クラス名
- インスタンスオブジェクト・ファクトリオブジェクトの種別
- データの値

DISPLAY FACTORY コマンドは，クラス名の指定によって，ファクトリオブジェクトの次の情報を表示します。

- データの値

形式

オブジェクト参照を指定してオブジェクトの情報を表示する



- データ名指定

  オブジェクトを参照するオブジェクト参照データ項目を指定します。

- SELF，EXCEPTION-OBJECT

  オブジェクトを参照する既定義オブジェクト名を指定します。

- #>

  データ名指定，SELF または EXCEPTION-OBJECT で参照するオブジェクトの特定のデータ名を指すために指定します。

- クラス名指定 1

  データ名が所属するクラス名を指定します。クラス名は，オブジェクトを定義するクラスまたはそのスーパクラスの名称です。

- データ名

  表示するデータ名を指定します。

- #ALL

  クラス名指定 1 が指定されないときは，オブジェクトを定義するクラスとそのスーパクラスに所属するデータをすべて表示します。クラス名指定 1 が指定されたときは，指定されたクラスに所属す

るデータをすべて表示します。ただし，データが定義されていないスーパクラスは何も表示されません。

- ATTRIBUTE | HEX | ALLTYPE

  データの表示形式を指定します。

  ・ATTRIBUTE：データ名の属性で表示します。

  ・HEX：16 進で表示します。

  ・ALLTYPE：ATTRIBUTE，HEX の両方の形式で表示します。

- GROUP | ELEMENT

  集団項目の表示形式を指定します。

  ・GROUP：英数字集団項目を英数字項目とみなし全体を先頭から表示します。または，日本語集団項目を日本語項目とみなし全体を先頭から表示します。

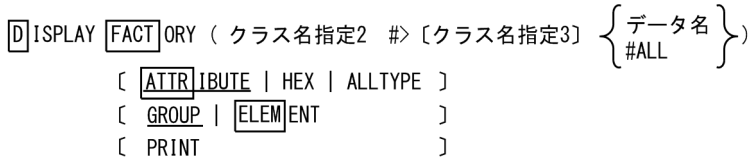
  ・ELEMENT：基本項目単位に表示します。

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」の「(1)　SET PRINT（テスト結果蓄積先の設定）」を参照してください。

クラス名を指定してファクトリオブジェクトの情報を表示する



- クラス名指定 2

  ファクトリオブジェクトのクラス名を指定します。

- #>

  参照するオブジェクトの特定のデータ名を指すために指定します。

- クラス名指定 3

  データ名が定義されているクラス名を指定します。クラス名は，オブジェクトを作成したクラスまたはそのスーパクラスの名称です。

- データ名

  表示するデータ名を指定します。

- #ALL

  クラス名指定 3 が指定されないときは，オブジェクトを定義するクラスとそのスーパクラスに所属するデータをすべて表示します。クラス名指定 3 が指定されたときは，指定されたクラスに所属するデータをすべて表示します。ただしデータが定義されていないスーパクラスは何も表示されません。

- ATTRIBUTE | HEX | ALLTYPE

  データの表示形式を指定します。

・ATTRIBUTE：データ名の属性で表示します。

・HEX：16 進で表示します。

・ALLTYPE：ATTRIBUTE，HEX の両方の形式で表示します。

- GROUP | ELEMENT

  集団項目の表示形式を指定します。

  ・GROUP：集団項目を英数字項目とみなし全体を先頭から表示します。

  ・ELEMENT：基本項目単位に表示します。

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」の「(1)　SET PRINT（テスト結果蓄積先の設定）」を参照してください。

注意事項

- GROUP オペランド，および ELEMENT オペランドは，データ名が集団項目でないときは無効となります。
- #ALL を指定したとき，ELEMENT オペランドは無効となります。集団項目は GROUP オペランドの形式で表示されます。
- 既定義オブジェクトの EXCEPTION-OBJECT は，COBOL プログラムの手続きで使用されているときに指定できます。既定義オブジェクトの SELF は，メソッド内では常に指定できます。
- ELEMENT オペランドは，集団項目を部分参照で表示する場合は無効となります。
- HEX | ALLTYPE オペランドは，データ名が内部ブール項目のときは無効となります。
- 表示するデータ名がオブジェクト参照データ項目の場合，データの値を表示します。オブジェクト参照データ項目が参照するオブジェクトの情報とデータ名は表示しません。
- クラス名指定 1・クラス名指定 2・クラス名指定 3 で指定するクラスは，-TDInf コンパイラオプションを指定してコンパイルする必要があります。データ名・#ALL 指定によってデータを表示するスーパクラスも，-TDInf コンパイラオプションを指定してコンパイルしたクラスだけです。
- データ名に，特殊レジスタは指定できません。
- データ名が表のとき，添字および部分参照の最左端位置と長さは整数定数を指定する必要があります。
- データ名，#ALL の指定を行わないとき，データ名の値は表示しません。DISPLAY OBJECT コマンドはオブジェクトの種別とクラス名だけを，DISPLAY FACTORY コマンドはクラス名だけを表示します。

使用例と表示形式

使用例 1 について，Windows(x86) COBOL2002 の場合と Windows(x64) COBOL2002 の場合の表示例を示します。使用例 2 以降で，明記されていない場合については，Windows(x86) COBOL2002 の場合だけを示します。

Windows(x86) COBOL2002 の場合と Windows(x64) COBOL2002 の場合の相違は，16 進表示個所でアドレスを表示している部分が 8 けたから 16 けたになっていることです。また，表示するデータ

によってはデータサイズが変更になるものもあります。変更になるデータは，マニュアル「COBOL2002　ユーザーズガイド」の「アドレス系データを表現するデータ項目」を参照してください。

次の COBOL プログラムのクラス定義に従って，TD コマンドの使用例を示します。

クラス MILK は，クラス DRINK を継承します。

クラス DRINK

```
IDENTIFICATION DIVISION.
CLASS-ID. DRINK INHERITS BASE.
      :
IDENTIFICATION DIVISION.
FACTORY.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 F-DATA01 PIC S9(9) COMP.
01 F-DATA02.
 02 F-DATA03 PIC X(25).
 02 F-DATA04 PIC 9(18).
 02 F-DATA05 PIC X(5).
01 F-DATA06 PIC X(5).
 :

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 O-DATA01 PIC S9(9) COMP.
01 O-DATA02 PIC S9(9) COMP.
 :
END CLASS DRINK.
```

クラス MILK

```
IDENTIFICATION DIVISION.
CLASS-ID. MILK INHERITS DRINK.
      :
IDENTIFICATION DIVISION.
FACTORY.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 F-DATA01 PIC S9(9) COMP.
01 F-DATA11 PIC S9(9) COMP.
01 F-DATA12 PIC S9(9) COMP.
 :

OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 O-DATA01 PIC S9(9) COMP.
01 O-DATA11 PIC S9(9) COMP.
 :

END CLASS MILK.
```

使用例 1

既定義オブジェクト SELF の値が，クラス MILK から生成されたファクトリオブジェクトを参照するときに，クラス MILK のスーパクラス DRINK に定義されたデータ名 F-DATA01 を属性と 16 進で表示します。

```
DISPLAY OBJECT (SELF #> #CLASS(DRINK) F-DATA01)  ALLTYPE
```

Windows(x86) COBOL2002の場合

```
オブジェクト参照データ名 : SELF
  値         +0000112345
*001E87F0   0001B6D9
オブジェクトを定義するクラス名 : MILK
オブジェクト種別 : ファクトリ
オブジェクト情報
クラスDRINKで定義されたデータ項目と値:
名 称       F-DATA01
  値         +000000000
*001E88F0   00000000
```

Windows(x64) COBOL2002の場合

```
オブジェクト参照データ名 :  SELF
  値          +00000000000000112345
*00011000001E8EAC   00000000  0001B6D9
オブジェクトを定義するクラス名 : MILK
オブジェクト種別 : ファクトリ
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名 称       F-DATA01
  値         +000000000
*00011000001E8FAC   00000000
```

使用例 2

既定義オブジェクト SELF の値がクラス MILK から生成されたインスタンスオブジェクトを参照するとき，クラス MILK のスーパクラス DRINK で定義されたデータ名 O-DATA01 を属性と 16 進で表示します。

```
DISPLAY OBJECT (SELF #> #CLASS(DRINK) O-DATA01)  ALLTYPE
```

```
オブジェクト参照データ名 :  SELF
  値        +0000112345
*001E87F0  0001B6D9
オブジェクトを定義するクラス名 :  MILK
オブジェクト種別: インスタンス
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名 称       O-DATA01
  値        +000000000
*001E88F0  00000000
```

使用例 3

既定義オブジェクト SELF の値が，クラス MILK から生成されたインスタンスオブジェクトを参照するとき，クラス名を指定しないで，データ名 O-DATA01 を属性と 16 進で表示します。

```
DISPLAY OBJECT (SELF #> O-DATA01)  ALLTYPE
```

```
オブジェクト参照データ名 : SELF
  値         +0000112345
*001E87F0    0001B6D9
オブジェクトを定義するクラス名 : MILK
オブジェクト種別: インスタンス
オブジェクト情報
クラスMILKで定義されたデータ項目と値 :
名　称       O-DATA01
  値         +000000000
*001E88F0    00000000
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名　称       O-DATA01
  値         +000000001
*001EC1F0    01000000
```

使用例 4

既定義オブジェクト SELF の値が，クラス MILK から生成されたファクトリオブジェクトを参照するとき，クラス MILK のスーパクラス DRINK のすべてのファクトリデータを，属性と 16 進で表示します。

```
DISPLAY OBJECT (SELF #> #CLASS(DRINK) #ALL)  ALLTYPE
```

```
オブジェクト参照データ名 : SELF
  値         +0000112345
*001E87F0  0001B6D9
オブジェクトを定義するクラス名 : MILK
オブジェクト種別: ファクトリ
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名　称       F-DATA01
  値         +000000000
*001E88F0  00000000
名　称       F-DATA02
  値         AAAAABBBBBCCCCCDDDDDEEEEE0102030405060708090FFF

*001E8930  41414141  41424242  42424343  43434344  44444444  45454545  45303130  32303330
*001E8950  34303530  36303730  38303930  30464646

名　称       F-DATA06
  値         AAAAA
*001E96F0  41414141  41
```

使用例 5

オブジェクト参照データ項目 A-DRINK の値が，クラス MILK から生成されたファクトリオブジェクトを参照するとき，オブジェクトのすべてのファクトリデータを，属性で表示します。

```
DISPLAY OBJECT (A-DRINK #> #ALL)  ATTRIBUTE
```

```
オブジェクト参照データ名 : A-DRINK
  値         +0000112345
オブジェクトを定義するクラス名 : MILK
オブジェクト種別: ファクトリ
オブジェクト情報
クラスMILKで定義されたデータ項目と値 :
名　称       F-DATA01
  値         +000000001
名　称       F-DATA11
  値         +000000002
名　称       F-DATA12
  値         +000000003

オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名　称       F-DATA01
  値         +000000000
名　称       F-DATA02
  値         AAAAABBBBBCCCCCDDDDDEEEEE0102030405060708090FFF
名　称       F-DATA06
  値         AAAAA
```

使用例 6

クラス名 MILK を指定して，ファクトリオブジェクトのスーパクラス DRINK のデータ名 F-DATA01 を属性と 16 進で表示します。

```
DISPLAY FACTORY (#CLASS(MILK) #> #CLASS(DRINK) F-DATA01)  ALLTYPE
```

```
ファクトリオブジェクト名 : MILK
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名　称      F-DATA01
  値        +000000000
*001E87F0  00000000
```

使用例 7

クラス名 MILK を指定して，ファクトリオブジェクトのデータ名 F-DATA01 を属性で表示します。

ファクトリオブジェクトのスーパクラス名は指定しないで，該当するデータ名をすべて表示します。

```
DISPLAY FACTORY (#CLASS(MILK) #> F-DATA01)  ATTRIBUTE
```

```
ファクトリオブジェクト名 : MILK
オブジェクト情報
クラスMILKで定義されたデータ項目と値 :
名　称    F-DATA01
  値      +000000000
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名　称    F-DATA01
  値      +000000001
```

使用例 8

クラス名 MILK を指定して，ファクトリオブジェクトのスーパクラス DRINK のすべてのデータを属性と 16 進で表示します。

```
DISPLAY FACTORY (#CLASS(MILK) #> #CLASS(DRINK) #ALL)  ALLTYPE
```

```
ファクトリオブジェクト名 : MILK
オブジェクト情報
クラスDRINKで定義されたデータ項目と値 :
名　称      F-DATA01
  値        +000000000
*001E88F0  00000000
名　称      F-DATA02
  値        AAAAABBBBBCCCCCDDDDDEEEEE0102030405060708090FFF
*001E8930  41414141  41424242  42424343  43434344  44444444  45454545  45303130  32303330
*001E8950  34303530  36303730  38303930  30464646
名　称      F-DATA06
  値        AAAAA
*001E96F0  41414141  41
```

使用例 9

オブジェクト参照データ項目 B-DRINK がクラス MILK のインスタンスオブジェクトを参照するとき，オブジェクト参照の値，クラス名，オブジェクトの種別を表示します。

```
DISPLAY OBJECT(B-DRINK)
```

```
オブジェクト参照データ名 : B-DRINK
  値          +0000112345
オブジェクトを定義するクラス名 : MILK
オブジェクト種別 : インスタンス
```

12.4.11　DISPLAY POINT（現在位置の表示）

プログラムが中断している現在の位置を表示します。

形式

```
DISPLAY POINT 〔 PRINT 〕
```

- PRINT
  SET PRINT コマンドで指定した出力先に結果を表示します。
  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」の「(1)　SET PRINT（テスト結果蓄積先の設定）」を参照してください。

使用例と表示形式

使用例 1

マルチスレッドプログラムのプログラム PROGRAM_A の入口で中断しています。

```
DISPLAY POINT
```

```
KCCC3326T-I　＜中断点＞ 呼び出し順序番号（1）　#ENTRY〈PROGRAM_A/PROGRAM_A〉　スレッドID（20）
```

使用例 2

プログラム PROGRAM_B の内部プログラム SUB1 の 1600 行の手続き名 P1 で中断しています。

```
DISPLAY POINT
```

```
KCCC3326T-I　＜中断点＞ 呼び出し順序番号（2）　1600 P1〈PROGRAM_B/SUB1〉
```

12.4.12　GETPROPERTY（プロパティの値表示）

OLE2 の OLE オブジェクトが持つ OLE プロパティの値を表示します。

形式

OLEオブジェクトを指定して表示する

```
GETPROPERTY OLE（OLEプロパティ指定）
            〔 ATTRIBUTE | HEX | ALLTYPE 〕
            〔 PRINT 〕
```

OLEオブジェクトを指定して所有するすべてのOLEプロパティを表示する

```
GETPROPERTY  OLE （  データ名指定1  #> #ALL）
            〔 ATTRIBUTE | HEX | ALLTYPE 〕
            〔 PRINT 〕
```

- OLE（OLE プロパティ指定）
  OLE プロパティを指定します。
- OLE（データ名指定 1 #> #ALL）

・データ名指定 1：OLE オブジェクトを参照する OLE オブジェクト参照データ項目を指定します。

- ATTRIBUTE | HEX | ALLTYPE

  データの表示形式を指定します。

  ・ATTRIBUTE：プロパティの属性で表示します。

  ・HEX：16 進で表示します。

  ・ALLTYPE：ATTRIBUTE，HEX の両方の形式で表示します。

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」の「(1)　SET PRINT（テスト結果蓄積先の設定）」を参照してください。

注意事項

引数を持つ OLE プロパティの値を表示するときは，OLE プロパティの名称に引数を付けて指定します。引数を指定しない場合は，名称のあとに"(*)"を表示して，値は表示されません。

使用例と表示形式

使用例 1 について，Windows(x86) COBOL2002 の場合と Windows(x64) COBOL2002 の場合の表示例を示します。使用例 2 以降で，明記されていない場合については，Windows(x86) COBOL2002 の場合だけを示します。

Windows(x86) COBOL2002 の場合と Windows(x64) COBOL2002 の場合の相違は，16 進表示個所でアドレスを表示している部分が 8 けたから 16 けたになっていることです。また，表示するデータによってはデータサイズが変更になるものもあります。変更になるデータは，マニュアル「COBOL2002　ユーザーズガイド」の「アドレス系データを表現するデータ項目」を参照してください。

使用例 1

OLE オブジェクト参照データ項目 OBJ-REF の値が参照する OLE オブジェクトのプロパティ'OrganizationName'の値を属性と 16 進で表示します。

```
GETPROPERTY OLE ('OrganizationName' WITH OBJ-REF)   ALLTYPE
```

Windows(x86) COBOL2002の場合

```
OLEオブジェクト参照データ名 : OBJ-REF
  値          +0000112345
*00406490     0001B6D9
名　称        OrganizationName
  型          文字列型 : 8
  値          (株)日立製作所
*--------     28002A68  2900E565  CB7AFD88  5C4F4062
```

Windows(x64) COBOL2002の場合

```
OLEオブジェクト参照データ名 : OBJ-REF
  値          +00000000000000112345
*0000000000406490    00000000  0001B6D9
名　称        OrganizationName
  型          文字列型 : 8
  値          (株)日立製作所
*----------------  28002A68  2900E565  CB7AFD88  5C4F4062
```

使用例 2

OLE オブジェクト参照データ項目 OBJ-REF の値が参照する OLE オブジェクトのすべてのプロパティの値を属性で表示します。

```
GETPROPERTY OLE (OBJ-REF #> #ALL)
```

```
OLEオブジェクト参照データ名：OBJ-REF
  値        +000013294
名　称      Application
  型        オブジェクト型：9
  値        54491400
名　称      OnWindow
  型        EMPTY：0
```

使用例 3

引数を持つ OLE プロパティ Name に引数 3 を渡して値を表示します。

```
GETPROPERTY OLE ('Name(3)' WITH OBJ-REF)   ALLTYPE
```

```
OLEオブジェクト参照データ名 :  OBJ-REF
  値        +0000112345
名　称       Name(3)
  型        文字列型 :  8
  値         John
```

12.4.13　GO（実行の開始／再開）

プログラムが起動前状態のときは，実行を開始します。中断状態のときは，実行を再開します。

形式

ユーザプログラム起動前の形式

```
GO 〔 PARAMETER （ 引数の文字列 ）〕
   〔 COVERAGE 〕
```

ユーザプログラム中断状態の形式

```
GO 〔 STATEMENT （ 文番号 ）  〕
   〔 PARAGRAPH （ 手続き名 ）〕
```

- PARAMETER（パラメタの文字列）
  起動時にプログラムへ渡すパラメタを指定します。
- COVERAGE
  カバレージ情報を蓄積します。
- STATEMENT（文番号）・PARAGRAPH（手続き名）
  プログラムの実行を再開させる位置を変更するときに，再開する文または手続きを指定します。

注意事項

- PARAMETER オペランドおよび COVERAGE オペランドは，プログラムの中断状態で指定したときは無効となります。

- PARAMETER オペランドは，次の場合は指定できません。

  ・プログラムの連動実行を行っている。

  ・-SimMain コンパイラオプションでプログラムをコンパイルして主プログラムシミュレーションを行っている。
- 文字列に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1　TD コマンドの指定方法」を参照してください。
- COVERAGE オペランドによってカバレージ情報を蓄積する翻訳単位は，コンパイル時に-CVInf コンパイラオプションを指定します。
- STATEMENT オペランドおよび PARAGRAPH オペランドは，プログラムの起動前に指定したときは無効となります。
- STATEMENT オペランドおよび PARAGRAPH オペランドは，中断しているソース要素の文または手続きを指定します。宣言手続きで中断しているときは，その宣言手続きの中の文または手続きを指定します。宣言手続きの外で中断しているときは，宣言手続きの外の文または手続きを指定します。
- STATEMENT オペランドに ENTRY 文の文番号は指定できません。

12.4.14　GO END・GO INVALID・GO EOP・GO ERROR（入出力条件シミュレーション）

入出力条件を擬似的に発生させます。このとき，入出力状態に値を設定します。ファイルシミュレーションの手続きにだけ指定できます。

ファイルシミュレーションについては，「9.2.7　プログラムの単体テスト」の「(3)　ファイルシミュレーション」を参照してください。

形式

```
GO END
GO INVALID
GO EOP
GO ERROR
```

注意事項

- GO END コマンド実行後のプログラムの制御は，次のような関係となります。

	USE 指定あり	USE 指定なし
AT END 指定あり	AT END 指定の無条件文に制御が移る	AT END 指定の無条件文に制御が移る
AT END 指定なし	USE 手続きが実行される	入出力文の終わりに制御が移る

- GO INVALID コマンド実行後のプログラムの制御は，次のような関係となります。

	USE 指定あり	USE 指定なし
INVALID KEY 指定あり	INVALID KEY 指定の無条件文に制御が移る	INVALID KEY 指定の無条件文に制御が移る
INVALID KEY 指定なし	USE 手続きが実行される	入出力文の終わりに制御が移る

- GO EOP コマンド実行後のプログラムの制御は，次のような関係となります。

	USE 指定あり	USE 指定なし
END-OF-PAGE 指定あり	END-OF-PAGE 指定の無条件文に制御が移る	END-OF-PAGE 指定の無条件文に制御が移る
END-OF-PAGE 指定なし	入出力文の終わりに制御が移る	入出力文の終わりに制御が移る

- GO ERROR コマンド実行後のプログラムの制御は，次のような関係となります。

USE 指定あり	USE 指定なし
USE 手続きが実行される	入出力文の終わりに制御が移る

12.4.15　IF（データの値比較）

条件が成立したときは，ELSE の前の TD コマンド群を実行します。条件が成立しないときは，ELSE のあとの TD コマンド群を実行します。

形式

```
IF CONDITION( 比較条件式 )
    〔 TDコマンド群 〕
    〔 ELSE 〔 TDコマンド群 〕〕
ENDIF
```

- CONDITION（比較条件式）

  比較については，「9.3.1　データの比較・代入規則」の「(2)　データの比較規則」を参照してください。

- TD コマンド群

  条件が成立したときと，成立しなかったときに実行する TD コマンド群を指定します。

使用例

文番号 8700 で中断し，R-CODE（I）の値によって，TD コマンドを実行します。

```
SET QUALIFICATION (#PROG(社員))
SET BREAK  ST(8700) DO
  IF CONDITION(R-CODE(I) = 0)
    DISPLAY COM('**TEST OK **')
  ELSE
    IF CONDITION(R-CODE(I) NOT= 8)
      DISPLAY COMMENT ('??? TEST NG')
      DISPLAY DATA(R-CODE(I))  HEX
      ASSIGN DATA(R-CODE(I)) VALUE(8)
```

```
    ELSE
      DISPLAY COMMENT ('**ERROR TEST OK**')
    ENDIF
  ENDIF
  GO
ENDDO
GO
```

12.4.16　QUIT（テストデバッグの終了）

テストデバッグを終了します。

形式

```
QUIT
```

12.4.17　REPEAT（繰り返し指定）

次のシミュレーションで一度に実行する手続きと，繰り返す回数を指定します。

- 副プログラムシミュレーションで副プログラムが呼ばれる
- ファイルシミュレーション入出力文が実行される
- DC シミュレーションで DC 文が実行される

ファイルシミュレーションについては，「9.2.7　プログラムの単体テスト」の「(3)　ファイルシミュレーション」を参照してください。

形式

```
REPEAT〔 TIMES（ 反復回数 ）〕
    〔 TDコマンド群 〕
ENDREPEAT
```

- TIMES（反復回数）

  シミュレーションが実行されることで，指定された TD コマンド群を繰り返し実行する回数を指定します。指定できるのは 1～2,147,483,647 の整数で，省略時は 1 が仮定されます。

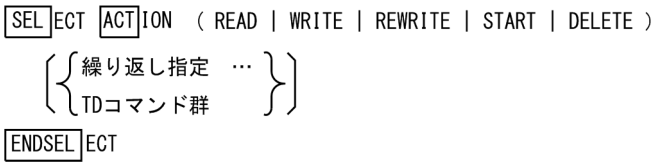
- TD コマンド群

  一度に実行するシミュレーション手続きを指定します。

12.4.18　SELECT ACTION（入出力文選択指定）

擬似実行する入出力文に対応させて，ファイルシミュレーションの手続きを実行させます。ファイルシミュレーションについては，「9.2.7　プログラムの単体テスト」の「(3)　ファイルシミュレーション」を参照してください。

形式



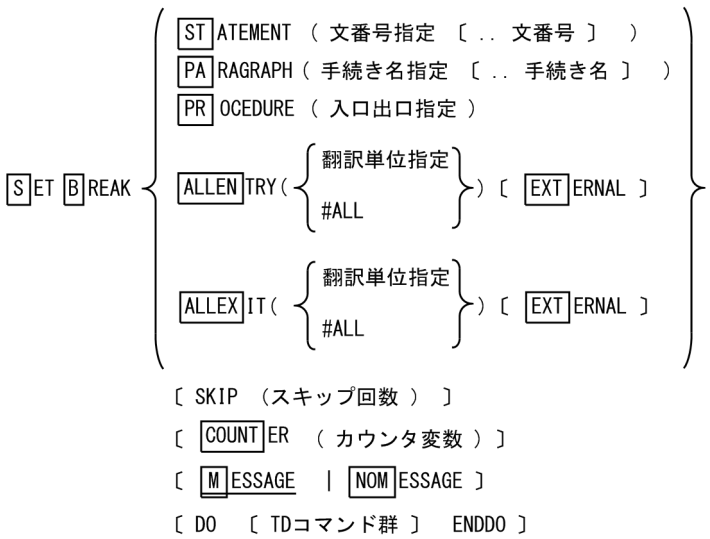
- ACTION（READ | WRITE | REWRITE | START | DELETE）
  対象とする入出力文を指定します。
- 繰り返し指定・TD コマンド群
  指定した入出力文で実行するシミュレーション手続きを指定します。
  繰り返し指定の詳細については，「12.4.17　REPEAT（繰り返し指定）」を参照してください。

12.4.19　SET BREAK／RESET BREAK（中断点の設定と解除）

(1)　SET BREAK（中断点の設定）

プログラムの実行を中断させる中断点を設定します。中断したときに実行する TD コマンドの指定もできます。

形式



- STATEMENT（文番号指定）
  文へ中断点を設定します。

- STATEMENT（文番号指定.. 文番号）
  指定した範囲内のすべての文および手続き名に中断点を設定します。
- PARAGRAPH（手続き名指定）
  手続きへ中断点を設定します。
- PARAGRAPH（手続き名指定 .. 手続き名）
  指定した範囲内のすべての文および手続き名に中断点を設定します。
- PROCEDURE（入口，出口指定）
  指定したソース要素の入口または出口に中断点を設定します。
- ALLENTRY（翻訳単位指定 | #ALL）
  次のソース要素の入口に中断を設定します。
  ・翻訳単位指定：指定した翻訳単位に属するソース要素
  ・＃ALL：プログラムに属するすべてのソース要素
- ALLEXIT（翻訳単位指定 | #ALL）
  次のソース要素の出口に中断を設定します。
  ・翻訳単位指定：指定した翻訳単位に属するソース要素
  ・＃ALL：プログラムに属するすべてのソース要素
- EXTERNAL
  ALLENTRY オペランドの指定があるときは，ほかの翻訳単位の呼び出し文から呼び出される入口だけに中断点を設定します。ALLEXIT オペランドの指定があるときは，ほかの翻訳単位に戻る出口だけに中断点を設定します。
  中断点を設定する入口と出口は，次のとおりになります。

入口または出口に中断点を設定するソース要素

- プログラム
  外部プログラム
- クラス
  メソッド
- 利用者定義関数

入口または出口に中断点を設定しないソース要素

- プログラム
  内部プログラム

- SKIP（スキップ回数）
  設定した中断点で中断する周期を指定できます。中断してから次に中断するまで，スキップ回数分，中断点で止まらずに通過させることができます。指定できるのは，1～2,147,483,647 の整数です。スキップ回数のカウントは，プログラムの開始時に 0 が設定されます。
- COUNTER（カウンタ変数）

カウンタ変数は，プログラムの開始時に 0 が設定され，DO～ENDDO オペランドで指定した TD コマンド群が実行されるたびに 1 ずつ値が増加します。

- MESSAGE | NOMESSAGE

  中断を知らせるメッセージを表示するかどうかを指定します。

- DO 〔 TD コマンド群 〕 ENDDO

  中断時に実行する TD コマンド群を指定します。

注意事項

- 中断点が設定されている文または手続きに，再度，中断点の設定をしたときは，前の設定が解除され，新しい指定が有効となります。指定方法が異なる場合でも，同じ文に対する設定のときは，次のように，新しい指定が有効になります。

使用例

SET BREAK PARAGRAPH (P1) SKIP (10)の指定で，SET BREAK STATEMENT(100) SKIP(1) は解除されます。

```
       SET BREAK STATEMENT (100)  SKIP(1)
       SET BREAK PARAGRAPH (P1)   SKIP(10)
```

```
100  P1.
200    MOVE  A  TO  B.
```

- 中断点が設定されている入口または出口に，再度中断点を設定したときは，前の設定が解除され，新しい指定が有効となります。指定方法が異なる場合でも，同じ入口または出口に対する設定のときは，次のように，新しい指定が有効になります。

使用例

プログラム「社員」の入口に設定された中断点の SKIP 指定は 20 となります。

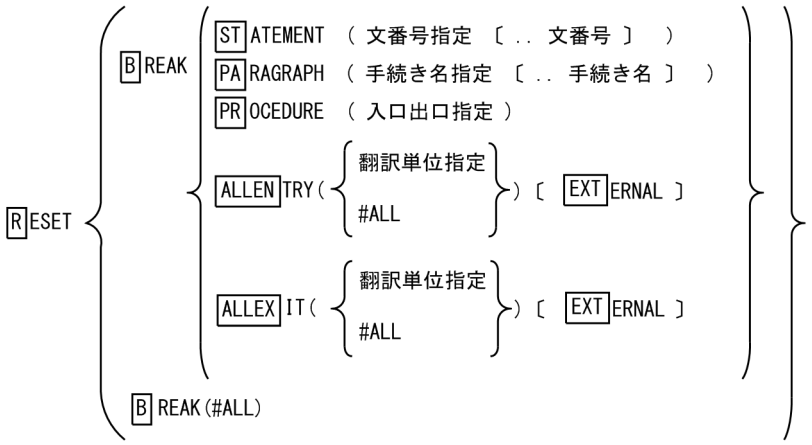
```
SET BREAK  ALLENTRY (#ALL) EXTERNAL  SKIP (10)
SET BREAK  PROCEDURE (#PROGRAM (社員/社員) #ENTRY)  SKIP (20)
```

- STATEMENT オペランドと PARAGRAPH オペランドの範囲指定，ALLENTRY オペランド，ALLEXIT オペランドは，複数の中断点を一度に設定します。カウンタ変数とスキップ回数は，設定された個々の中断点ごとにカウントされます。
- 中断点を設定できる文が存在しない文番号を指定した場合は，その文番号よりあとで最も近い中断点を設定できる文に中断点を設定します。ただし，指定した文番号よりあとに中断点を設定できる文がないときは，翻訳単位の最後の文を仮定します。
- STATEMENT オペランドで範囲指定を行ったとき，最後の文番号に中断点を設定できる文がないときは，指定した文番号より前で最も近い文にまで，中断点を設定します。
- STATEMENT オペランドと PARAGRAPH オペランドの範囲指定では，終了位置が開始位置より前にあってはなりません。

(2) RESET BREAK（中断点の解除）

SET BREAK コマンドで設定した中断点を解除します。

形式



- STATEMENT（文番号指定）
  文の中断点を解除します。
- STATEMENT（文番号指定 .. 文番号）
  指定した範囲内のすべての文および手続き名の中断点を解除します。
- PARAGRAPH（手続き名指定）
  手続きの中断点を解除します。
- PARAGRAPH（手続き名指定 .. 手続き名）
  指定した範囲内のすべての文および手続き名の中断点を解除します。
- PROCEDURE（入口，出口指定）
  指定したソース要素の入口または出口の中断点を解除します。
- ALLENTRY（翻訳単位指定 | #ALL）
  次のソース要素の入口の中断点を解除します。
  ・翻訳単位指定：指定した翻訳単位に属するソース要素
  ・＃ALL：プログラムに属するすべてのソース要素
- ALLEXIT（翻訳単位指定 | #ALL）
  次のソース要素の出口の中断点を解除します。
  ・翻訳単位指定：指定した翻訳単位に属するソース要素
  ・＃ALL：プログラムに属するすべてのソース要素
- EXTERNAL
  ALLENTRY オペランドの指定があるときは，ほかの翻訳単位の呼び出し文から呼び出される入口の中断点を解除します。ALLEXIT オペランドの指定があるときは，ほかの翻訳単位に戻る出口の中断点を解除します。

EXTERNAL オペランド指定時に解除される入口，出口の中断点は，「(1)　SET BREAK（中断点の設定)」の入口または出口に中断点を設定するソース要素と同様となります。

- BREAK（#ALL）

  すべての中断点を解除します。

注意事項

中断点を設定したときの SET BREAK コマンドの指定方法と同一の指定方法でなくても解除できます。

使用例

すべてのソース要素の入口に中断点を設定したあと，プログラム PGM01 と PGM02 の，外部プログラムの入口の中断点だけを解除します。

```
SET   BREAK   ALLENTRY (#ALL)   EXTERNAL

RESET BREAK   ALLENTRY (#PROGRAM(PGM01))    EXTERNAL
RESET BREAK   ALLENTRY (#PROGRAM(PGM02))    EXTERNAL
```

12.4.20　SET FLOW／RESET FLOW（フロー情報の蓄積開始と終了）

(1)　SET FLOW（フロー情報の蓄積開始）

フロー情報は，プログラムの実行時に通過した文の実行経路です。SET FLOW コマンドで，フロー情報の蓄積を開始します。蓄積したフロー情報は，DISPLAY FLOW コマンドで表示できます。また，バッチモードでは，実行時エラーが発生したときに，フロー情報を表示します。

形式

```
SET FLOW 〔 PROCEDURE | PARAGRAPH | STATEMENT 〕
         〔 STACK ( 蓄積数 ) 〕
```

- PROCEDURE | PARAGRAPH | STATEMENT

  蓄積する通過点の単位を指定します。

  PROCEDURE：ソース単位の入口，出口の情報を蓄積します。

  PARAGRAPH：ソース単位の入口，出口と手続き名の情報を蓄積します。

  STATEMENT：ソース単位の入口，出口と手続き名および文の情報を蓄積します。

- STACK（蓄積数）

  蓄積する通過点の数を指定します。指定できる数は 1～500 の整数で，省略時は 99 が仮定されます。

(2)　RESET FLOW（フロー情報の蓄積中止）

フロー情報の蓄積を中止します。

形式

```
RESET FLOW
```

12.4.21　SET LOG／RESET LOG（モニタウィンドウ表示内容のファイル出力の設定と解除）

(1)　SET LOG（モニタウィンドウ表示内容のファイル出力の設定）

モニタウィンドウへ出力されるテストデバッグの実行結果をファイルへ出力します。

形式

```
SET LOG OUTFILE（ ファイル名 ）
```

- OUTFILE（ファイル名）

  実行結果を出力するファイル名を指定します。Windows ファイル名に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1　TD コマンドの指定方法」を参照してください。

注意事項

- SET LOG でファイルがオープンされ，RESET LOG でファイルがクローズされます。ファイルがオープンされている間は，ほかのアプリケーションからの書き込みはできません。
- 指定されたファイルがすでにある場合は，上書きとなります。
- SET LOG によって出力先が指定されていた場合は，先に指定されたファイルをクローズしてから指定されたファイルをオープンします。
- オープンされているファイルを，ファイル名に指定することはできません。したがって，SET PRINT コマンドで指定されたファイルも，指定できません。
- 大文字・小文字を区別するため，#OPTION コマンドの指定がファイル名に効かないようにさせたいときは，文字列を指定します。指定方法については，「12.1.4　英大文字と英小文字，英数字文字と拡張文字を区別しての文字列の指定方法」を参照してください。
- ファイル名に正規表現を指定しても適用されません。

(2)　RESET LOG（モニタウィンドウ表示内容のファイル出力の解除）

テストデバッグの実行結果をモニタウィンドウへ出力するように戻します。

形式

```
RESET LOG
```

12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）

(1)　SET PRINT（テスト結果蓄積先の設定）

PRINT オペランドが指定された TD コマンドによる実行結果の出力先を指定します。SET PRINT の指定がない場合，GUI モードではモニタウィンドウへ，バッチモードでは結果出力ファイルへ，実行結果を出力します。

形式

```
SET PRINT OUTFILE（ファイル名 ）
```

- OUTFILE（ファイル名）

  実行結果を出力するファイル名を指定します。Windows ファイル名に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1　TD コマンドの指定方法」を参照してください。

注意事項

- SET PRINT でファイルがオープンされ，RESET PRINT でファイルがクローズされます。ファイルがオープンされている間は，ほかのアプリケーションからの書き込みを行うと，ファイルのオープンでエラーとなります。
- 指定されたファイルがすでにある場合は，上書きとなります。
- SET PRINT コマンドによって出力先が指定されていた場合は，先に指定されたファイルをクローズしてから指定されたファイルをオープンします。
- すでにオープンされているファイルをファイル名に指定できません。したがって，SET LOG コマンドで指定されたファイルも指定できません。
- 大文字・小文字を区別するため，#OPTION コマンドの指定がファイル名に効かないようにさせたいときは，文字列を指定します。指定方法については，「12.1.4　英大文字と英小文字，英数字文字と拡張文字を区別しての文字列の指定方法」を参照してください。
- ファイル名に正規表現を指定しても適用されません。

(2)　RESET PRINT（テスト結果蓄積先の解除）

実行結果の出力先を戻します。

形式

```
RESET PRINT
```

12.4.23　SET QUALIFICATION／RESET QUALIFICATION（翻訳単位・ソース要素の指定と解除）

(1)　SET QUALIFICATION（翻訳単位・ソース要素の指定）

テストデバッグの対象とする翻訳単位またはソース要素を変更します。SET QUALIFICATION コマンドで翻訳単位指定をすると，その後の文番号に翻訳単位を指定したことになります。SET QUALIFICATION コマンドでソース要素指定をすると，その後のソース要素指定ができるオペランドに指定したことになります。

形式

SET QUALIFICATION（ 翻訳単位指定 | ソース要素指定 ）

- QUALIFICATION（翻訳単位指定）
  仮定する翻訳単位を指定します。
- QUALIFICATION（ソース要素指定）
  仮定するソース要素を指定します。

注意事項

- SET QUALIFICATION コマンドは，プログラムの開始時および再開時に解除されます。また，プログラムの実行中止（STOP コマンド）でも解除されます。
- 翻訳単位指定は，文番号だけに有効です。その他にはソース要素指定をしなくてはなりません。
  SET QUALIFICATION コマンド以降，文番号以外も対象とする場合は，ソース要素指定を指定します。

(2)　RESET QUALIFICATION（翻訳単位・ソース要素の指定の解除）

翻訳単位指定またはソース要素指定を解除します。

形式

RESET QUALIFICATION

12.4.24　SET TRACE／RESET TRACE（トレース表示の開始と中止）

(1)　SET TRACE（トレース表示の開始）

プログラムの実行を追跡し，指定した単位で通過点を表示します。

形式

```
SET TRACE
〔 PROCEDURE | PARAGRAPH | STATEMENT 〕
〔 INTERFACE 〕
〔 FROMSTATEMENT(文番号指定)|FROMPARAGRAPH(手続き名指定) 〕
〔 TOSTATEMENT(文番号指定)|TOPARAGRAPH(手続き名指定)| TOSTEP(ステップ数)〕
〔 PRINT 〕
```

- PROCEDURE | PARAGRAPH | STATEMENT

  表示単位を指定します。

  PROCEDURE：ソース単位の入口，出口を表示します。

  PARAGRAPH：ソース単位の入口，出口および手続き名を表示します。

  STATEMENT：ソース単位の入口，出口，手続き名および文を表示します。

- INTERFACE

  入口では引数を表示し，出口では戻り値を表示します。

- FROMSTATEMENT（文番号指定）|FROMPARAGRAPH（手続き名指定）

  表示を開始する位置を指定します。

- TOSTATEMENT（文番号指定）|TOPARAGRAPH（手続き名指定）

  表示を終了する位置を指定します。

- TOSTEP（ステップ数）

  表示を終了するステップ数を指定します。最初に表示したときのステップ数を 1 とします。指定できる範囲は，1～2,147,483,647 の整数です。

- PRINT

  SET PRINT コマンドで指定した出力先に結果を表示します。

  SET PRINT コマンドについては，「12.4.22　SET PRINT／RESET PRINT（テスト結果蓄積先の設定と解除）」の「(1)　SET PRINT（テスト結果蓄積先の設定）」を参照してください。

注意事項

- プログラムを再開始した場合，FROMSTATEMENT および FROMPARAGRAPH オペランドの指定がある場合はオペランドで指定した位置を通過したときに表示が開始されます。指定がない場合は，プログラム開始時に表示が開始されます。
- FROMSTATEMENT オペランドの文番号指定に該当する文または手続き名がない場合は，指定した文番号以降の最も近い文または手続き名を仮定します。

  また，TOSTATEMENT オペランドを指定した文番号に該当する文または手続き名がない場合は，指定した文番号以前の最も近い文または手続き名を仮定します。
- FROMSTATEMENT および FROMPARAGRAPH オペランドと TOSTATEMENT および TOPARAGRAPH オペランドの指定がある場合は，開始する位置と終了する位置に同じ位置は指定できません。

(2)　RESET TRACE（トレース表示の中止）

SET TRACE コマンドで開始したトレース表示を中止します。

形式

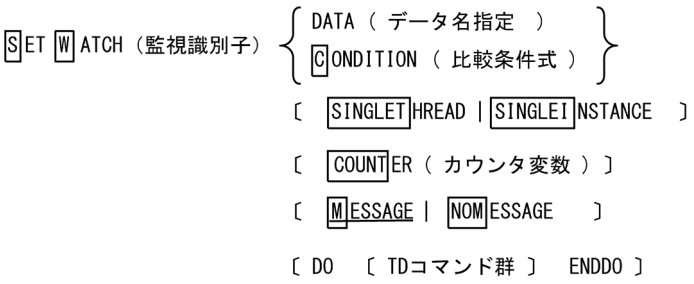


12.4.25　SET WATCH／RESET WATCH（データ監視条件の設定と解除）

(1)　SET WATCH（データ監視条件の設定）

プログラムの実行を中断する条件を設定します。

形式



- WATCH（監視識別子）

  データ監視条件を識別するための名称を指定します。

- DATA（データ名指定）

  監視対象とするデータ名を指定します。値の変化を監視します。詳細については，「9.2.1　プログラムの中断」を参照してください。

- CONDITION（比較条件式）

  監視対象となるデータの値に対しての比較条件式を指定します。条件式の評価を監視します。

  比較条件式の指定の詳細については，「9.3.1　データの比較・代入規則」の「(2)　データの比較規則」を参照してください。

  条件式の監視については，「9.2.1　プログラムの中断」を参照してください。

- SINGLETHREAD

  単一のスレッドのデータの監視を行います。

- SINGLEINSTANCE

  単一のオブジェクトのデータの監視を行います。

- COUNTER（カウンタ変数）

  カウンタ変数は，プログラムの開始時に 0 が設定され，DO～ENDDO オペランドで指定した TD コマンド群が実行されるたびに 1 ずつ値が増加します。

- MESSAGE | NOMESSAGE
  中断を知らせるメッセージを表示するかどうかを指定します。
- DO 〔 TD コマンド群 〕 ENDDO
  中断時に実行する TD コマンド群を指定します。

注意事項

- データ名指定または比較条件式のデータ名に，カウンタ変数は指定できません。
- データ名指定または比較条件式のデータ名に添字付きのデータ名を指定するときの添字は，整数定数でなければなりません。また，添字の範囲指定はできません。
- データ名指定または比較条件式のデータ名に部分参照のデータ名を指定するとき，最左端位置と長さ指定は整数定数でなければなりません。
- すでに設定された監視識別子を指定して，再度，データ監視条件を設定した場合は，以前の設定が無効となり，新しい設定が有効となります。
- オブジェクトに属するデータ名の領域がオブジェクトごとに生成されるときは，生成されたオブジェクトの領域ごとに監視が行われます。カウンタ変数は，オブジェクトのデータごとにカウントされます。SINGLEINSTANCE オペランドは，その一つのデータを指定して監視できます。
- データ名の領域が再帰によって複数生成されるときは，生成された領域ごとに監視が行われます。カウンタ変数は，データごとにカウントされます。
- マルチスレッドで実行するプログラムで，指定したデータ名の領域がスレッドごとに生成されるときは，生成されたスレッドの領域ごとに監視が行われます。カウンタ変数は，スレッドのデータごとにカウントされます。SINGLETHREAD オペランドは，その一つのデータを指定して監視できます。
- 連絡節で定義した LIMIT 指定のない動的長基本項目への部分参照指定は，プログラム開始前には指定できません。指定した場合は，KCCC3434T-E のメッセージを出力して処理を中止します。

(2) RESET WATCH（データ監視条件の解除）

SET WATCH コマンドで設定したデータ監視条件を解除します。

形式

RESET WATCH（ 監視識別子 | #ALL ）

- WATCH（監視識別子）
  データ監視条件を識別するための名称を指定します。
- WATCH（#ALL）
  すべてのデータ監視条件を解除します。

注意事項

GUI モードで設定したデータ監視条件は，RESET WATCH コマンドで解除できません。

12.4.26　STEP IN（ステップイン実行の開始／再開）

プログラムが起動前のときは，実行を開始します。中断状態のときは，実行を再開します。プログラムを１文実行して中断します。

形式

ユーザプログラムが起動前の形式

```
STEP IN 〔PARAMETER（ 引数の文字列 ）〕
        〔COVERAGE  〕
```

ユーザプログラムが中断状態の形式

```
STEP IN
```

- PARAMETER（パラメタの文字列）

  起動時にプログラムへ渡すパラメタを指定します。

- COVERAGE

  カバレージ情報を蓄積します。

注意事項

- PARAMETER オペランドおよび COVERAGE オペランドは，プログラムの中断状態で指定したときは無効となります。
- PARAMETER オペランドは，次の場合は指定できません。

  ・プログラムの連動実行を行っている。

  ・-SimMain コンパイラオプションでプログラムをコンパイルして主プログラムシミュレーションを行っている。

  文字列に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1　TD コマンドの指定方法」を参照してください。
- COVERAGE オペランドによってカバレージ情報を蓄積する翻訳単位は，コンパイル時に-CVInf コンパイラオプションを指定します。

12.4.27　STEP OVER（ステップオーバー実行の開始／再開）

プログラムが起動前のときは，実行を開始します。中断状態のときは，実行を再開します。プログラムを１文実行して中断します。ただし，CALL 文，関数呼び出しを持つ文および INVOKE 文は 1 文として実行します。

形式

ユーザプログラムが起動前の形式

```
STEP OVER 〔 PARAMETER（引数の文字列 ）〕
          〔 COVERAGE 〕
```

ユーザプログラムが中断状態の形式

```
STEP OVER
```

- PARAMETER（パラメタの文字列）

  起動時にプログラムへ渡すパラメタを指定します。

- COVERAGE

  カバレージ情報を蓄積します。

注意事項

- PARAMETER オペランド，COVERAGE オペランドは，プログラムの中断状態で指定したときは無効となります。
- PARAMETER オペランドは，次の場合は指定できません。

  ・プログラムの連動実行を行っている。

  ・-SimMain コンパイラオプションでプログラムをコンパイルして主プログラムシミュレーションを行っている。

  文字列に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1　TD コマンドの指定方法」を参照してください。
- COVERAGE オペランドによってカバレージ情報を蓄積する翻訳単位は，コンパイル時に-CVInf コンパイラオプションを指定します。

12.4.28　STEP TO（ステップツー実行の再開）

プログラムの中断時に，実行を再開する位置を変更します。

形式

```
STEP TO { STATEMANT（ 文番号 ） }
        { PARAGRAPH（ 手続き名 ） }
```

- STATEMENT（文番号）・PARAGRAPH（手続き名）

  プログラムを再開させるときに，再開させる文または手続き名を指定します。

注意事項

- STATEMENT オペランド，PARAGRAPH オペランドは，中断しているソース要素の文または手続き名を指定します。宣言手続きで中断しているときは，その宣言手続きの中の文または手続きを指定します。宣言手続きの外で中断しているときは，宣言手続きの外の文または手続き名を指定します。

- STATEMENT オペランドに ENTRY 文の文番号は指定できません。

12.4.29　STOP（プログラムの強制終了）

実行中のプログラムを強制的に終了します。

形式

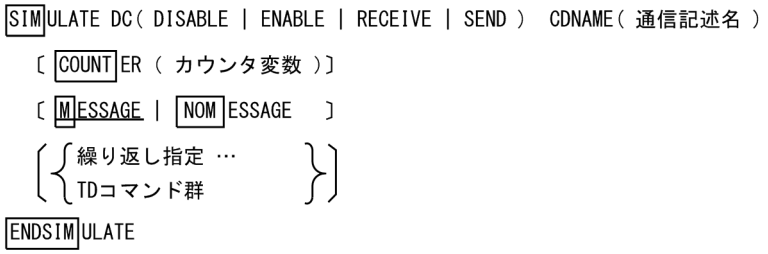
```
STOP
```

12.4.30　SIMULATE DC（DC シミュレーションの設定）

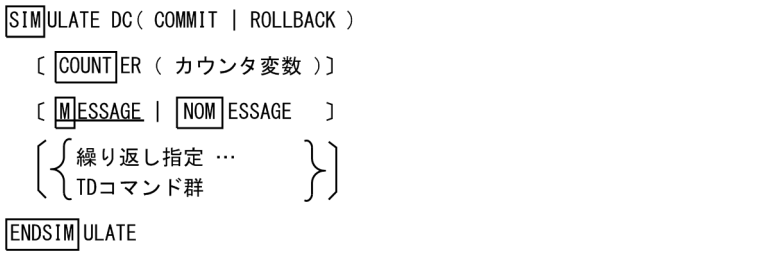
DC シミュレーションを，TD コマンドで手続きを設定して実行させます。詳細については，「9.2.7　プログラムの単体テスト」の「(5)　DC シミュレーション」を参照してください。

形式

操作文がDISABLE，ENABLE，RECEIVE，SENDのとき



操作文がCOMMIT，ROLLBACKのとき



- DC（操作文種別）

  シミュレーションの対象とする DC 文を指定します。

  指定できる操作文は，COMMIT，DISABLE，ENABLE，RECEIVE，ROLLBACK，SEND 文です。

- CDNAME（通信記述名）

  シミュレーションの対象とする通信記述名を指定します。操作文種別と通信記述名の組み合わせでシミュレーション対象が決まります。

- COUNTER（カウンタ変数）

カウンタ変数は，プログラムの開始時に 0 が設定され，シミュレーションが実行されるたびに 1 ずつ値が増加します。

- MESSAGE | NOMESSAGE

  シミュレーションの開始を知らせるメッセージを表示するかどうかを指定します。

- 繰り返し指定，TD コマンド群

  DC 文で実行する手続きを TD コマンドで指定します。

  繰り返し指定の詳細については，「12.4.17　REPEAT（繰り返し指定）」を参照してください。

注意事項

マルチスレッドプログラムでは，使用できません。

12.4.31　SIMULATE FILE（ファイルシミュレーションの設定）

ファイルのシミュレーションで実行する手続きを指定します。詳細については，「9.2.7　プログラムの単体テスト」の「(3)　ファイルシミュレーション」を参照してください。

形式

```
SIMULATE FILE（ ファイル名指定 ） OPENMODE（ INPUT | OUTPUT | IO | EXTEND ）
  〔 RECORD（ 記号名 〔,… 〕 ） 〕
  〔 COUNTER（ カウンタ変数 ） 〕
  〔 MESSAGE | NOMESSAGE 〕
  〔 DEFINE〔 記号名構造定義指定 … 〕 ENDDEFINE 〕
  ｛ 入出力文選択指定 …
    繰り返し指定 …
    TDコマンド群 ｝
ENDSIMULATE
```

- FILE（ファイル名指定）

  シミュレーション対象とするファイル名を指定します。

- OPENMODE（INPUT | OUTPUT | IO | EXTEND）

  ファイルのオープンモードを指定します。

- RECORD（記号名）

  シミュレーション手続き中で使用する記号名を定義します。記号名は，該当するファイルのレコード名定義に対応します。

- COUNTER（カウンタ変数）

  カウンタ変数は，プログラムの開始時に 0 が設定され，シミュレーションが実行されるたびに 1 ずつ値が増加します。

- MESSAGE | NOMESSAGE

  シミュレーションの開始を知らせるメッセージを表示するかどうかを指定します。

- DEFINE 〔 記号名構造定義指定 ・・・ 〕 ENDDEFINE

RECORD オペランドで指定した記号名に対応するデータ名が集団項目のとき，集団項目に所属するデータ名を指定します。詳細については，「12.4.37　レベル番号（記号名構造指定）」を参照してください。

- 入出力文選択指定
  入出力文で実行する手続きを TD コマンドで指定します。
  繰り返し指定の詳細は，「12.4.18　SELECT ACTION（入出力文選択指定）」を参照してください。
- 繰り返し指定，TD コマンド群
  入出力文で実行する手続きを TD コマンドで指定します。
  繰り返し指定の詳細については，「12.4.17　REPEAT（繰り返し指定）」を参照してください。

12.4.32　SIMULATE MAIN（主プログラムシミュレーションの設定）

主プログラムシミュレーションで実行する手続きを指定します。詳細については，「9.2.7　プログラムの単体テスト」の「(1)　主プログラムシミュレーション」を参照してください。

形式

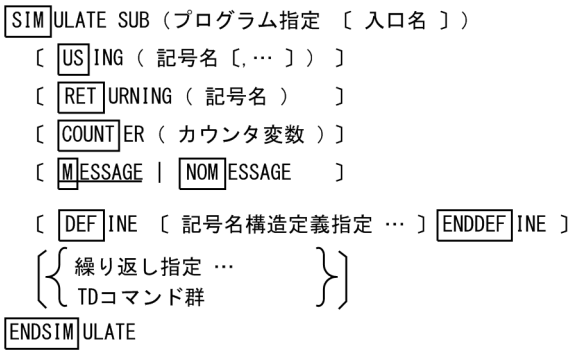
```
SIMULATE MAIN（ プログラム指定 〔 入口名 〕）
  〔 MESSAGE | NOMESSAGE〕
  〔 TDコマンド群 〕
ENDSIMULATE
```

- MAIN（プログラム指定）
  主プログラムから呼び出される副プログラムの入口名を指定します。
- 入口名
  ENTRY 文で指定される入口点の名前を指定します。
- MESSAGE | NOMESSAGE
  シミュレーションの開始を知らせるメッセージを表示するかどうかを指定します。
- TD コマンド群
  テストデバッグ対象のプログラムへ渡すインタフェースの設定をする手続きを TD コマンドで指定します。

12.4.33　SIMULATE SUB（副プログラムシミュレーションの設定）

副プログラムシミュレーションで実行する手続きを指定します。詳細については，「9.2.7　プログラムの単体テスト」の「(2)　副プログラムシミュレーション」を参照してください。

形式



- SUB（プログラム指定）

  シミュレーション対象とするプログラム定義の名前を指定します。

- 入口名

  ENTRY 文で指定される入口点の名前を指定します。

- USING（記号名〔,・・・〕）

  CALL 文の USING で指定したデータ名をシミュレーションの手続きで参照するための記号名を指定します。USING で指定したデータ名の順番と USING オペランドに指定した記号名の順番が対応します。

- RETURNING（記号名）

  シミュレーション手続きで RETURNING に値を設定するための記号名を指定します。

- COUNTER（カウンタ変数）

  カウンタ変数は，プログラムの開始時に 0 が設定され，シミュレーションが実行されるたびに 1 ずつ値が増加します。

- MESSAGE | NOMESSAGE

  シミュレーションの開始を知らせるメッセージを表示するかどうかを指定します。

- DEFINE 〔 記号名構造定義指定 ・・・〕 ENDDEFINE

  USING または RETURNING オペランドで指定した記号名に対応するデータ名が集団項目のとき，集団項目に所属するデータ名を指定します。詳細については，「12.4.37　レベル番号（記号名構造指定）」を参照してください。

- 繰り返し指定，TD コマンド群

  プログラムの呼び出しをシミュレーションで実行する手続きを TD コマンドで指定します。

  繰り返し指定の詳細については，「12.4.17　REPEAT（繰り返し指定）」を参照してください。

12.4.34　TEST（テストケースの選択）

複数のテストケースをまとめて定義して，その中から実行するテストケースを指定します。

形式

```
TEST 〔CASEID （ ケース識別子 〔,…〕 ）〕
    〔〔 テストケース | コマンド群 〕… 〕
ENDTEST
```

- CASEID（ケース識別子）

  定義されたテストケースの中から，実行するテストケースを指定します。このオペランドを指定しなかったときは，TEST コマンドの中で定義されたすべてのテストケースを実行します。末尾に*を付けて指定すると，*より前の文字が一致するケース識別子を持つすべてのテストケースを実行します。

- テストケース

  テストケースとして実行する TD コマンドを指定します。

  テストケースの詳細については，「12.4.5　CASE（テストケースの指定）」を参照してください。

- コマンド群

  QUIT，#INCLUDE，#OPTION コマンドが指定できます。

12.4.35　#INCLUDE（TD コマンド格納ファイルの取り込み）

指定したファイルから TD コマンドを入力します。

形式

```
#INCLUDE INFILE（ファイル名 ）〔 { MESSAGE | NOMESSAGE } … 〕
```

- INFILE（ファイル名）

  TD コマンドを格納しているファイル名を指定します。Windows ファイル名に空白，括弧，アポストロフィまたはダブルコーテーションを使用する場合の指定方法については，「12.1　TD コマンドの指定方法」を参照してください。

- MESSAGE | NOMESSAGE

  入力した TD コマンドを表示するかどうかを指定します。

注意事項

- #INCLUDE コマンドで指定したファイルの中に，再度同じファイル名を指定した#INCLUDE コマンドは記述できません。
- 大文字・小文字を区別するため，#OPTION コマンドの指定がファイル名に効かないようにさせたいときは，文字列を指定します。指定方法については，「12.1.4　英大文字と英小文字，英数字文字と拡張文字を区別しての文字列の指定方法」を参照してください。
- バッチモードの cbltd2k コマンドで-SyntaxOnly オプションの指定の有無に関係なく，指定された TD コマンド格納ファイルの取り込みをします。バッチモードの cbltd2k コマンドで-SyntaxOnly オプションが指定されていた場合は，取り込んだ TD コマンド格納ファイル内の TD コマンドの構文解析が行われます。

- ファイル名に正規表現を指定しても適用されません。

12.4.36 #OPTION（オプションの変更）

TD コマンドの解析方法を指定します。

形式

#OPTION 〔 NOCASE | CASE1 | CASE2 〕|〔 2002SYNTAX | 85SYNTAX 〕

下記の説明で，特殊文字とは次の文字です。

＋ － ＊ ／ ＝ ￥ ' " ； ． ， （） ＜ ＞ ＆ ：

- NOCASE

  同じ文字を表す英大文字と英小文字，英数字文字と拡張文字を同じに扱います。

  ・COBOL2002 の予約語と特殊文字は，英数字文字と拡張文字のどちらでも指定できます。

  ・データ名の拡張文字と英小文字は，英数字文字の英大文字に変換されます。

- CASE1

  同じ文字を表す英数字文字の英大文字と英小文字を同じに扱います。英数字文字と拡張文字は区別します。

  ・COBOL2002 の予約語と特殊文字は，英数字文字で指定します。

  ・データ名の英数字文字の英小文字は英大文字に変換されます。拡張文字のデータ名は変換しません。

- CASE2

  同じ文字を表す英大文字と英小文字，英数字文字と拡張文字を区別します。

  ・COBOL2002 の予約語と特殊文字は，英数字文字の英大文字で指定します。

  ・データ名は変換しません。

- 2002SYNTAX

  TD コマンドのオペランドに指定する利用者定義語の最大長を COBOL2002 言語の規則と同等にします。利用者定義語に関する詳細については，マニュアル「COBOL2002　言語　標準仕様編」を参照してください。

- 85SYNTAX

  TD コマンドのオペランドに指定する利用者定義語の最大長を COBOL85 言語の規則と同等にします。

注意事項

バッチモードの起動コマンドで-SyntaxOnly オプションの指定の有無に関係なく，指定は有効となります。

使用例

DISPLAY DATA コマンドのデータ名 data1 および data2 は，英小文字のまま扱われます。

```
#OPTION  NOCASE
SET BREAK STATEMENT(#PROGRAM( PROG ) 100) DO
      #OPTION  CASE2
      DISPLAY DATA( data1 )
ENDDO
DISPLAY DATA( data2 )
```

12.4.37　レベル番号（記号名構造指定）

次のオペランドで指定する記号名に対応するデータ名が集団項目のとき，集団項目に属するデータ名に対応する記号名を指定します。指定された記号名によって，集団項目に属するデータ名をシミュレーションの手続きで参照できます。

- ファイルシミュレーション RECORD オペランド
- 副プログラムシミュレーションの USING オペランドおよび RETURNING オペランド

形式

```
レベル番号　記号名
```

- レベル番号

  レベル番号を指定します。

- 記号名

  集団項目に属するデータ名に対応させる記号名を指定します。

注意事項

- DEFINE オペランドと ENDDEFINE オペランドの間でだけ指定できます。
- 対応するデータ名に TYPE 句または SAME AS 句の指定がある場合，レベル番号は展開後の構造で指定します。
- レベル番号の 01～09 の 0 は省略できます。

13 カバレージ機能の概要

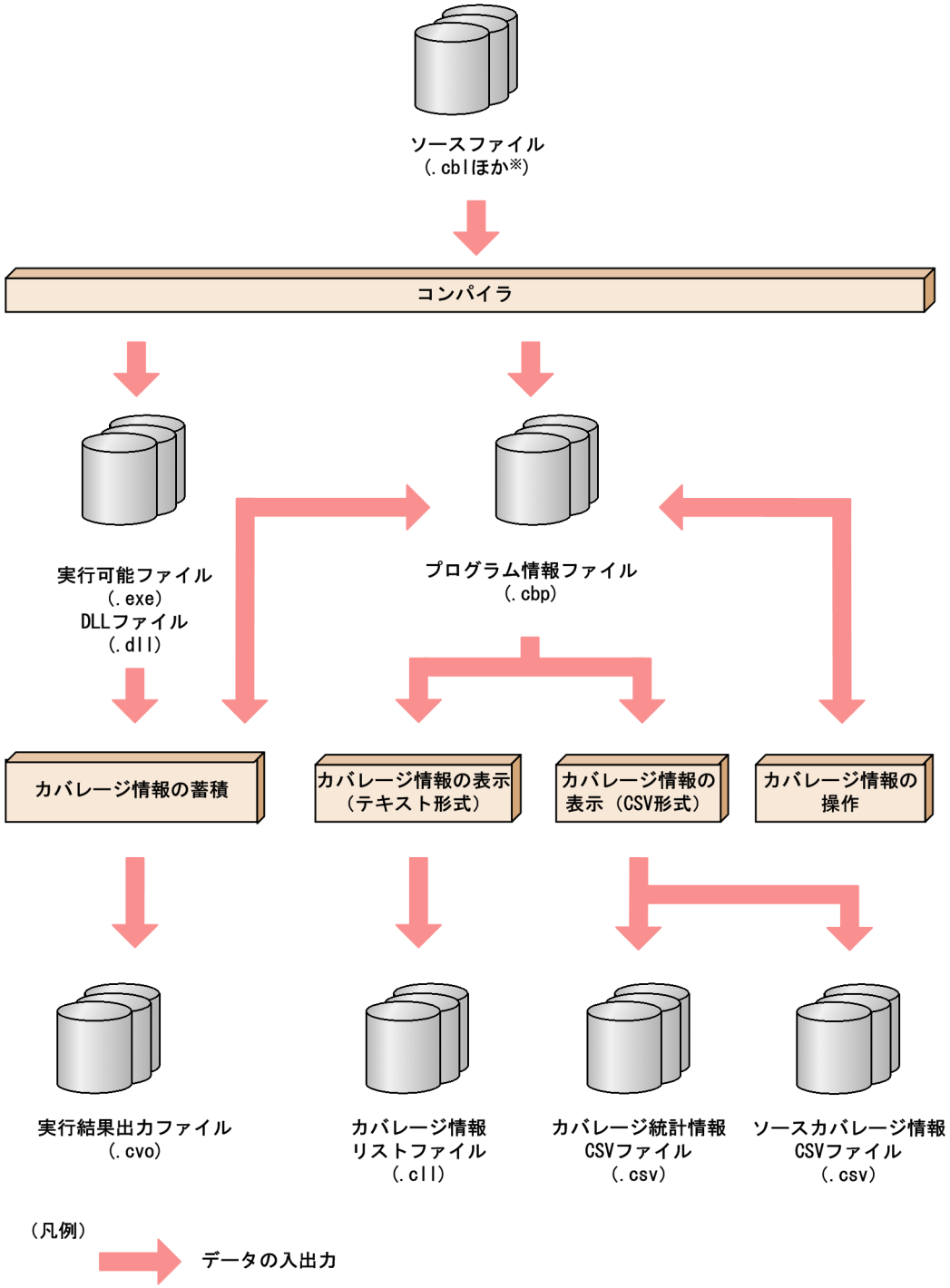
カバレージ機能を使うと，プログラムの品質を評価したり，プログラムの性能を分析したりできます。この章では，これらの情報の内容と取得，表示などの方法について説明します。

13.1.1 カバレージ機能の入出力構成と使用するファイル

(1) 入出力構成

カバレージ機能の入出力構成を示します。矢印は，入力を示します。

図 13-1　カバレージ情報の蓄積・表示・操作



td130010.vsd／D:¥SVN_COBOL¥01_ソースファイル¥3021-3-601¥VSD¥／／2018/11/21／12:36／

注※

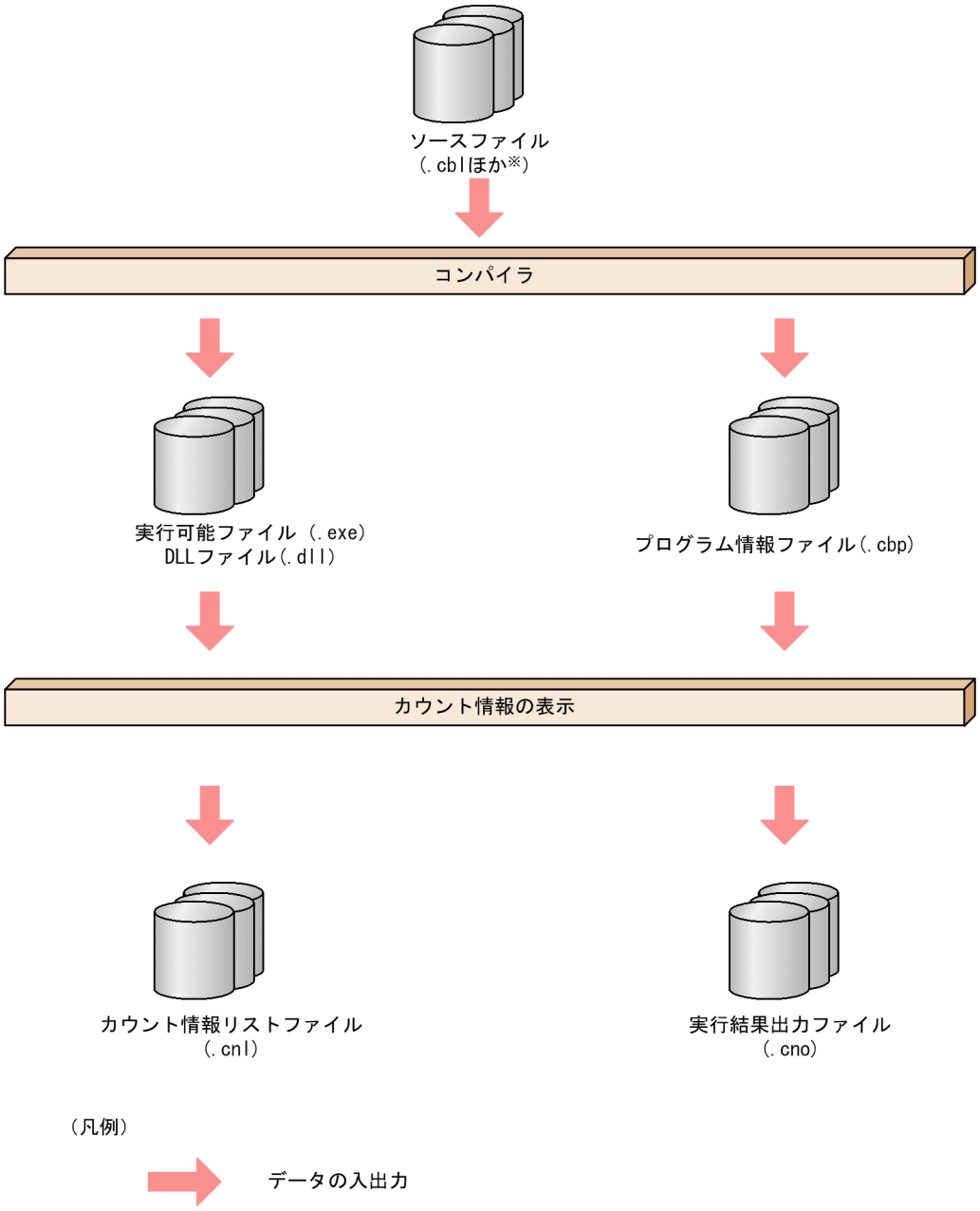
目的に応じて次の拡張子を使用します。

・固定形式正書法で書かれた原始プログラムをコンパイルする場合
　.cbl，.cob，.ocb，または環境変数 CBLFIX で指定した拡張子

・自由形式正書法で書かれた原始プログラムをコンパイルする場合

.cbf，.ocf，または環境変数 CBLFREE で指定した拡張子

図 13-2　カウント情報の表示



注※

目的に応じて次の拡張子を使用します。

・固定形式正書法で書かれた原始プログラムをコンパイルする場合

.cbl，.cob，.ocb，または環境変数 CBLFIX で指定した拡張子

・自由形式正書法で書かれた原始プログラムをコンパイルする場合

.cbf，.ocf，または環境変数 CBLFREE で指定した拡張子

(2)　使用するファイル

カバレージ機能で使用するファイルを次に示します。

- 実行可能ファイル

  カバレージ情報の蓄積対象，またはカウント情報の表示対象となる実行可能ファイル（.exe）です。

- DLL ファイル

  カバレージ情報の蓄積対象，またはカウント情報の表示対象となる DLL ファイル（.dll）またはライブラリファイルです。

- プログラム情報ファイル

  カバレージ情報の蓄積，またはカウント情報の表示に必要とするプログラムの情報を格納するファイルです。また，蓄積，取得したカバレージ情報を格納するためのファイルでもあります。-CVInf コンパイラオプションを指定してコンパイルしたときに出力されます。

  カバレージ情報の操作時は，プログラム情報ファイルに格納されている情報に対してカバレージ情報の操作をします。

- カバレージ情報リストファイル（カバレージ情報表示時）

  カバレージ表示の実行結果を格納するファイルです。ファイルはテキストファイルです。

- カウント情報リストファイル（カウント情報表示時）

  カウント情報の表示結果を格納するファイルです。ファイルはテキストファイルです。

- カバレージ統計情報 CSV ファイル（カバレージ情報表示時）

  ソースファイル単位または翻訳単位に，カバレージ統計情報を出力するファイルです。複数のプログラム情報ファイルに対するカバレージ統計情報を一つのファイルにまとめて出力します。ファイルは CSV ファイルです。

- ソースカバレージ情報 CSV ファイル（カバレージ情報表示時）

  原始プログラム 1 行ごとのカバレージ情報を出力するファイルです。一つのプログラム情報ファイルに対して，ソースカバレージ情報 CSV ファイルを一つ出力します。ファイルは CSV ファイルです。

- 実行結果出力ファイル（連動実行）

  カバレージ情報の蓄積およびカウント情報の表示で，実行中のメッセージの出力は実行結果出力ファイルに出力されます。次の理由で，実行結果出力ファイルに出力できない場合については，「15.2.2　プログラムからの連動実行による方法」の「(3)　実行結果出力ファイルにメッセージを出力できないときの処理」を参照してください。

  - 環境変数 CBLTDEXEC の値の指定に誤りがあった場合
  - 結果出力ファイルにエラーが発生し，実行結果出力ファイルが出力できなかった場合
  - 実行環境が整っていなかったためカバレージ情報の蓄積またはカウント情報の表示ができなかった場合

注意事項

- プログラム情報ファイルは，実行可能ファイルまたは DLL ファイルのあるディレクトリを参照します。任意のディレクトリのプログラム情報ファイルを参照するときは，環境変数 CBLPIDIR にディレクトリ名を指定します。
- プログラム情報ファイルは，使用するカバレージと同じバージョンのコンパイラで作成されたものでなければなりません。異なる場合は，エラーメッセージが出力されます。

- 環境変数 CBLPIDIR を指定した場合のプログラム情報ファイルの検索順序は，次の順序で検索します。検索した結果，見つからないときは，プログラム情報ファイルに該当するプログラムはカバレージの対象となりません。

検索順序

1. 環境変数 CBLPIDIR で指定したフォルダ
2. 実行可能ファイルに含まれるプログラムは，実行可能ファイルのあるフォルダ
   DLL に含まれるプログラムは，DLL のあるフォルダ

- 次に示すファイルに，複数のカバレージから同時に書き込みを行ったとき，あとからファイルをオープンする処理がエラーとなる場合があります。オープンエラーが起きた場合は，表 13-1　オープンエラーが起きた場合の対処方法の対処をしてください。

表 13-1　オープンエラーが起きた場合の対処方法

ファイル名	オープンエラーが発生する可能性があるとき	対処方法
プログラム情報ファイル	カバレージ情報の蓄積時	再度カバレージ蓄積を行う。
ログ出力ファイル	• SET LOG コマンド投入時 •「オプション」「モニタ」で「モニタウィンドウの内容をファイルに出力する」を選択したとき	ログ出力ファイル名を変更する。
結果蓄積ファイル	SET PRINT コマンド投入時	結果出力ファイル名を変更する。
結果出力ファイル	バッチモードの開始時	• 結果出力ファイル名を変更する。 • 時間を置いて，再度バッチを起動する。

- ファイルの相対パス名を指定したときの起点となるフォルダ，およびパスのつかないファイル名を指定したときのフォルダは，表 13-2　相対パス名の起点となるフォルダに示すとおりです。

表 13-2　相対パス名の起点となるフォルダ

ファイル名	GUI モード	バッチモード
TD コマンド格納ファイル	「テストデバッグの設定ダイアログボックス」で指定した作業フォルダ	起動コマンドを投入したフォルダ
ログ出力ファイル		−
結果蓄積ファイル		起動コマンドを投入したフォルダ
結果出力ファイル	−	

(凡例)

−：使用できません。

13.1.2　プログラムのコンパイル

カバレージを使用するためには，プログラムのコンパイル時に，コンパイラオプションとして-CVInf コンパイラオプションを必ず指定します。カバレージの機能を使用するために指定する COBOL2002 のコンパイラオプションを次に示します。コンパイラオプションの詳細については，「2.3　コンパイラオプション」を参照してください。

表 13–3　カバレージの機能とコンパイラオプション

カバレージの機能	コンパイラオプション
カバレージを使用する	-CVInf

注意事項

- リンカオプションに-DEBUG は指定しないでください。指定した場合のカバレージの動作は保証しません。
- -Optimize,2 コンパイラオプションによる最適化の指定をして作成したプログラムでは，文が最適化されることがあり，プログラムの動きを保証できない場合があります。また，中断点で参照できるデータ名が，最適化によって参照できない場合があります。

13.1.3　テストしたいプログラムの実行

カバレージの対象となるプログラムは，一つの実行可能ファイルと複数の DLL ファイルで構成されます。

カバレージ対象のプログラムの実行環境は，COBOL2002 のプログラムの実行環境に基づきます。実行環境の詳細については，「3.3.2　実行環境の設定」を参照してください。

注意事項

- DLL のロードは，COBOL プログラムの実行時の動作に従います。

  指定した DLL がテスト実行時にロードされた場合，カバレージの対象になります。ロードされなかった場合，カバレージの対象になりません。
- カバレージでは，DLL のロード時に以下のチェックを行い，指定された DLL であるかを判断しています。チェックで指定された DLL とは異なると判断された場合は，カバレージの対象になりません。
  - ・DLL ファイル内で保持している作成時間が等しい。
  - ・DLL ファイル内で保持しているファイル名が等しい(フォルダ名は含みません)。
- Windows 以外で作成された実行可能ファイル，または DLL ファイル，あるいは，それ以外のファイルの拡張子を実行可能ファイル，または DLL ファイルの拡張子に変更したものをカバレージ対象に指定された場合は，KCCC4200T-S のエラーメッセージを出力し，実行を中止します。
- Windows の別のシステム（32bit Windows の場合は 64bit Windows，64bit Windows の場合は 32bit Windows）で作成された実行可能ファイル，または DLL ファイルがカバレージの機能の

対象に指定された場合は，KCCC4202T-S のエラーメッセージを出力し，実行を中止します。ただし，64bit Windows 上でも，32bit 用として作成された実行可能ファイル，または DLL ファイルは 32bit のシステムで作成されたものとして扱います。

- 同じ Windows のシステムで作成された COBOL85 の実行可能ファイル，または，DLL ファイルがカバレージの機能の対象に指定された場合は，エラーメッセージを出力し，実行を中止します。

13.1.4　テストプログラムの構成

カバレージの対象となるプログラムの構成，およびプログラム定義，クラス定義，関数定義の詳細については，マニュアル「COBOL2002　ユーザーズガイド」の翻訳グループについての説明を参照してください。

13.1.5　カバレージ情報の表示と操作

カバレージ情報は，テストによって実行された手続き文の割合です。計画したテスト内容に従って，プログラムに記述した手続き文がテスト実行されたかどうかを把握でき，テスト工程の進捗度を判定できます。

実行されたテストプログラムの文はカバレージ情報としてプログラム情報ファイルに記憶されます。プログラム情報ファイルに蓄積されたカバレージ情報は，次の形式で表示できます。

- 翻訳単位ごとに実行された文の割合を，文・分岐・呼び出し文ごとに示す。
- ソース原文を表示して実行が済んだ文と実行されていない文を示す。
- 原始プログラムの修正によって変更された差分に対するカバレージ情報を示す。

さらに，蓄積したカバレージ情報をクリアする，別の環境で蓄積したカバレージ情報をマージするなどの機能があります。

13.1.6　カウント情報の表示

テスト実行させたプログラムの文の実行回数を表示します。繰り返し実行される文の実行回数や，呼び出すプログラムの実行回数も計測できるため，COBOL プログラムの実行性能を把握できます。このカウント情報を基にして，改善の施策を検討できます。

13.1.7　その他の注意事項

(1)　ファイルの書き込み時の制約

- 複数の開発マネージャまたは ccbl2002 コマンドによって，同時にビルドをした場合は，コンパイラが生成する次のファイルは，あとから生成されたファイルが有効になります。
  - プログラム情報ファイル
- 同時に複数のカバレージ情報の蓄積をした場合，プログラムの終了後に，プログラム情報ファイルに書き込みを同時にすると，エラーになるときがあります。

(2)　プログラム情報ファイルのエラーについて

- COBOL2002 以外で作成されたプログラム情報ファイルをカバレージ情報の表示，またはカバレージ情報の操作の対象に指定した場合は，KCCC4416T-E のエラーメッセージを出力し，カバレージ情報の表示，またはカバレージ情報の操作の対象にしません。
- COBOL2002 ではあるが，別のシステムで作成されたプログラム情報ファイルをカバレージ情報の表示，またはカバレージ情報の操作の対象に指定した場合は，KCCC4437T-E のエラーメッセージを出力し，カバレージ情報の表示，またはカバレージ情報の操作の対象にしません。
- 別のシステムで作成されたプログラム情報ファイルをカバレージ情報の表示，またはカバレージ情報の操作の対象に指定した場合は，エラーメッセージを出力し，カバレージ情報の表示，またはカバレージ情報の操作の対象にしません。

(3)　カバレージで動作させるプログラムについての注意事項

カバレージは，デバッグ対象プログラムを自身と同じ権限で動作させます。

このため，カバレージを標準権限で起動させた場合と，管理者権限で起動させた場合とで，権限の違いによって実行結果が変わることがあります。また，カバレージは，デバッグ対象プログラムを実行させたい権限で実行しなければなりません。特に，デバッグ対象プログラムに管理者権限を設定している場合は，必ずカバレージを管理者権限で実行しなければなりません。標準権限で実行した場合は，デバッグ対象プログラムの起動に失敗します。

デバッグ対象プログラムの実行権限について，次に示します。

表 13–4　デバッグ対象プログラムの実行権限

カバレージの実行権限	デバッグ対象プログラムの設定	
	標準権限	管理者権限
標準権限	標準権限で動作する	起動に失敗する
管理者権限	管理者権限で動作する	管理者権限で動作する

13.2　カバレージ情報の表示と操作

13.2.1　カバレージ情報

カバレージ機能が表示するカバレージ情報は，次のとおりです。どれも，翻訳単位ごとに割合を求め，小数点第 2 位以下を切り捨てた値を表示します。

C0 メジャー

実行した文の割合を表す情報です。

C0 メジャー＝（実行が済んだ実行文の数）/（実行文の数）×100（%）

C1 メジャー

処理が分岐する個所で，実行した分岐先の数の割合を表す情報です。

C1 メジャー＝（実行が済んだ分岐先の数）/（実行できる分岐先の数）×100（%）

S1 メジャー

実行した呼び出し文の数の割合を表す情報です。

S1 メジャー＝（実行が済んだ呼び出し文の数）/（実行できる呼び出し文の数）×100（%）

なお，プログラムを修正したあと，修正個所に着目してカバレージ情報を採取することもできます。

差分 C0 メジャー

修正個所中の，実行した文の割合を表す情報です。

差分 C0 メジャー＝（実行が済んだ実行文の数）/（実行文の数）×100（%）

差分 C1 メジャー

修正個所中の，処理が分岐する個所で実行した分岐先の数の割合を表す情報です。

差分 C1 メジャー＝（実行が済んだ分岐先の数）/（実行できる分岐先の数）×100（%）

差分 S1 メジャー

修正個所中の，実行した呼び出し文の数の割合を表す情報です。

差分 S1 メジャー＝（実行が済んだ呼び出し文の数）/（実行できる呼び出し文の数）×100（%）

注意事項

- カバレージ情報は，ブロックごとに蓄積します。カバレージのブロックは，順次実行される無条件文の集まりで，条件文の条件句の直前までが原則として一つのブロックとなります。ブロックの先頭の文を実行したときにブロック全体が実行されたと判断します。ブロック中の文で実行時エラーが発生した場合も，そのブロックは実行されたと判断します。

ブロックを分ける一般規則と例

- それ自身でブロックとなる文または句

(1)制御を 2 方向以上に分岐させる文

・IF 文・条件指定のある文

・PERFORM 文（TIMES 指定，UNTIL 指定（VARYING 指定の UNTIL を含む）の PERFORM 文）

・EVALUATE 文

・SEARCH 文

・GO TO 文（DEPENDING ON 指定つきの文，DEPENDING ON 指定はこのブロックに含まれる）

- 直前でブロックを区切る文または句

(1)2 方向以上に分岐

・THEN

・ELSE

・条件指定

・手続き名〔THRU 手続き名〕 TIMES 指定，UNTIL 指定（VARYING 指定の UNTIL を含む）の PERFORM 文

・WHEN

・WHEN OTHER

・AT END（SEARCH 文）

・手続き名（DEPENDING ON 指定つきの GO TO 文）

(2)単独の制御の開始

・手続き部分の最初の文

・宣言部分の節名

・手続き部分の節名

・段落名

・ENTRY 文

(3)制御の合流の直前

・範囲句（END－動詞）

- 直後でブロックを区切る文または句

(1)制御を 1 方向に分岐させる文

・CALL 文（条件指定がない文）

・PERFORM 文（TIMES 指定，UNTIL 指定（VARYING 指定の UNTIL を含む）がない文）

・SORT 文（INPUT PROCEDURE または OUTPUT PROCEDURE がある文）

・MERGE 文（OUTPUT PROCEDURE がある文）

・OLE メソッド操作を除く INVOKE 文（条件指定がない文）

・INITIALIZE 文（関数一意名指定あり）

・INSPECT 文（同上）

・MOVE 文（同上）

・COMPUTE 文（同上）

・STRING 文（同上）

・UNSTRING 文（同上）

(2)制御の終わり

・STOP 文

・EXIT PROGRAM 文

・GOBACK 文

・EXIT METHOD 文

・EXIT FUNCTION 文

・EXIT 文

・GO TO 文（DEPENDING ON 指定なし）

・EXIT PERFORM 文

・EXIT USE 文

・IF 文，SERCH 文の NEXT SENTENCE

・RESUME 文

(3)制御の終わりになる可能性のある文

・OPEN 文

・CLOSE 文

・READ 文（条件指定なし）

・DELETE 文（条件指定なし）

・WRITE 文（条件指定なし）

・REWRITE 文（条件指定なし）

・START 文（条件指定なし）

・SORT 文（USING または GIVING 指定がある文）

・MERGE 文（GIVING 指定がある文）

・RAISE 文

- C0 メジャーは，次の文を除く文を対象とします。

  ・原始文操作

  COPY 文

  REPLACE 文

  INCLUDE 文

  ・宣言文

  USE 文

  ・覚え書きの文（ENTER 文，うち PERFORM 文の外にある EXIT PERFORM 文）

- C0 メジャーは，次の文を対象とします。

  ・上記以外の文（データベースアクセス機能で用いる文，データコミュニケーション機能で用いる文，EXEC SQL 文も含む）

  ・NEXT SENTENCE 指定（IF 文，SEARCH 文）

- C1 メジャーの対象となる文は，次のとおりです。絶対に制御が渡らない分岐も対象にします。

文	分岐方向	備考
IF 文	2	・THEN ・ELSE
EVALUATE 文	WHEN の数※1	・WHEN への分岐 ・WHEN OTHER
GO TO 文（DEPENDING ON 指定つき）	手続き名の数＋ 1	・手続き名への分岐 ・手続き名への分岐なし
PERFORM 文（TIMES 指定，UNTIL 指定（VARYING 指定の UNTIL を含む））	2	・ループする ・ループしない
SEARCH 文	WHEN の数＋ 1	・WHEN への分岐 ・AT END への分岐
条件指定のある文※2	2	・条件～への分岐 ・NOT 条件～への分岐

注※1　無条件文がない WHEN 指定の直後に WHEN 指定が続く場合，明示的または暗黙的無条件文がある WHEN 指定までを 1 つの分岐方向と数えます。

（例）

```
EVALUATE　AAA
WHEN A DISPLAY A *> 分岐方向 1
WHEN B *>（無条件文なし）*> 分岐方向 2
WHEN C DISPLAY C
WHEN D *>（無条件文なし）*> 分岐方向 3
WHEN E *>（無条件文なし）
WHEN F DISPLAY F
WHEN G DISPLAY G *> 分岐方向 4
WHEN OTHER DISPLAY O *> 分岐方向 5
```

この例では，EVALUATE 文は WHEN 指定を 7 つと，WHEN OTHER 指定を持ちます。しかし，無条件文を持たない WHEN 指定は，後続の無条件文を持つ WHEN 指定と合わせて 1 つと数えるため，「WHEN B」「WHEN C」で分岐方向 1 つ，「WHEN D」「WHEN E」「WHEN F」で分岐方向 1 つとなり，WHEN の数は合計 8 つありますが，分岐方向の合計は 5 つとなります。

注※2　条件指定のある文を次に示します。

条件指定	文
AT END ,NOT AT END	READ,RETURN
ON SIZE ERROR ,NOT ON SIZE ERROR	ADD,COMPUTE,DIVIDE,MULTIPLY,SUBTRACT
ON OVERFLOW ,NOT ON OVERFLOW	STRING,UNSTRING
ON EXCEPTION ,NOT ON EXCEPTION	CALL,ACCEPT
AT EOP ,NOT AT EOP	WRITE

条件指定	文
INVALID KEY ,NOT INVALID KEY	DELETE,READ,REWRITE,START,WRITE
ON OVERFLOW	CALL

- S1 メジャーの対象となる呼び出し文は，次のとおりです。

  ・CALL 文

  ・ファクトリおよびインスタンスオブジェクトのメソッド呼び起こしで指定した INVOKE 文（OLE メソッドに対する INVOKE 文は除く）

  GO TO 文の直後等のように制御が渡らない文も対象にします。CALL 文，INVOKE 文の条件指定の有り無し，呼び出し先プログラムの実行可，不可には依存しません。

13.2.2　カバレージ情報表示の手順

カバレージ情報を表示する手順を次に示します。

1. プログラムのコンパイル

   -CVInf コンパイラオプションを指定して，原始プログラムをコンパイルします。

2. プログラムの実行

   テストによるプログラムの実行を繰り返し行い，カバレージ情報を蓄積します。また，テストデバッガでも，カバレージの蓄積ができます。

3. カバレージ情報の表示

   蓄積したカバレージ情報を表示します。カバレージ情報は，テキスト形式または CSV 形式のファイルへ出力できます。

   テキスト形式の場合，カバレージ情報の表示には次の種類があります。

表 13-5　カバレージ情報の表示（テキスト形式）

表示の種別	内容
翻訳単位の一覧表示	指定した翻訳単位の C0 メジャー，C1 メジャーを一覧表示します。
まとめ表示	指定した翻訳単位の C0 メジャー，C1 メジャー，S1 メジャーを表示します。
ソース表示※	原始プログラムを表示し，変更行，変更影響行，実行が済んだ実行文（C0 メジャーの対象），実行が済んだ分岐先（C1 メジャーの対象）に印を付けます。
差分ソース表示※	修正によって生じた原始プログラムの差分を表示し，差分に含まれる変更行，変更が影響する行，実行が済んだ実行文（差分 C0 メジャーの対象），実行が済んだ分岐先（差分 C1 メジャーの対象）に印を付けます。
未実行ソース表示※	原始プログラムの未実行の文を表示します。
差分未実行ソース表示※	修正による原始プログラムの差分に含まれる未実行の文を表示します。
呼び出し文ソース表示※	原始プログラムの S1 カバレージの対象になる呼び出し文を表示します。

注※

これらの表示を指定すると，まとめ情報を同時に表示します。

CSV 形式の場合，カバレージ情報の表示には次の種類があります。

表 13–6　カバレージ情報の表示（CSV 形式）

表示の種別	内容
カバレージ統計情報	指定したソースファイル単位または翻訳単位の C0 メジャー，C1 メジャー，S1 メジャーを出力します。
ソースカバレージ情報	原始プログラム，原始プログラムの変更による差分，C0 メジャーおよび C1 メジャーの実行が済んだ状態を出力します。また，差分ソース，未実行ソース，差分未実行ソース，または呼び出し文ソースのどれに該当する行であるかを識別できます。

4. カバレージ情報の操作

ファイルに蓄積したカバレージ情報に対して，次の操作ができます。

表 13–7　カバレージ情報の操作

操作の種別	内 容
カバレージ情報の０%化	実行が済んだ文をすべて未実行文に戻します。すべてのカバレージ情報が 0%になります。
差分カバレージ情報の０%化	差分の実行が済んだ文をすべて未実行文に戻します。
差分カバレージ情報のクリア	修正による原始プログラムの差分を，差分とみなさないようにします。
カバレージ情報のマージ	別々の環境で蓄積した同一翻訳単位のカバレージ情報をマージします。

13.2.3　プログラムのコンパイル

カバレージ情報を取得するプログラムは，-CVInf コンパイラオプションを指定してコンパイルします。コンパイラオプションについては，「13.1.2　プログラムのコンパイル」を参照してください。

コンパイラは，プログラム情報ファイルの有無によって，次のとおりにプログラム情報ファイルを出力します。

表 13–8　コンパイラによるプログラム情報ファイルの生成

プログラム情報ファイルの有無	コンパイラが出力するプログラム情報ファイル
無し	プログラム情報ファイルを新たに作成します。
有り	原始プログラムの修正部分に対する「差分」情報をプログラム情報ファイルへ出力します。

13.2.4　カバレージ情報の蓄積

次のどれかの方法で，プログラムを実行して，カバレージ情報を，プログラム情報ファイルへ蓄積します。

- Windows のプログラム一覧にある［COBOL2002］下のメニューから［カバレージ for COBOL2002］を選び，カバレージウィンドウからプログラムを実行する。
- コマンドプロンプトから cblcv2k コマンドを使用して，プログラムを実行する。
- 「プログラムからの連動実行」の設定を行い，プログラムを実行する。

また，次のどちらかの方法で，テストデバッグによるプログラムの実行時にも，カバレージ情報が蓄積できます。

- Windows のプログラム一覧にある［COBOL2002］下のメニューから［テストデバッガ for COBOL2002］を選び，カバレージ情報の蓄積を指定してプログラムを実行する。
- TD コマンドのプログラムの実行（GO コマンド）で，COVERAGE オペランドを指定する。

注意事項

- マルチスレッドプログラムは，COBOL プログラムの全スレッドがカバレージ情報蓄積の対象になります。
- 実行可能ファイル・DLL に含まれる翻訳単位とプログラム情報ファイルは，コンパイルによって同時に作成されていなければなりません（同一のコンパイル時間である必要があります）。
- プログラムの実行が終了したときに，蓄積したカバレージ情報をプログラム情報ファイルへ保存します。プログラムが実行時エラーによって終了した場合も，終了までに蓄積したカバレージ情報を保存します。
- カバレージ情報の蓄積は，ブロックごとに蓄積します。カバレージのブロックは，順次実行される無条件文の集まりで，条件文の条件句の直前までが原則として一つのブロックとなります。ブロックの先頭の文を実行したときにブロック全体が実行されたと判断します。ブロック中の文で共通例外や実行時エラーが発生した場合も，そのブロックは実行されたと判断します。

13.2.5　カバレージ情報の表示（テキスト形式）

次のどちらかの方法で，プログラム情報ファイルに蓄積されたカバレージ情報を，テキスト形式のカバレージ情報リストファイルへ出力できます。

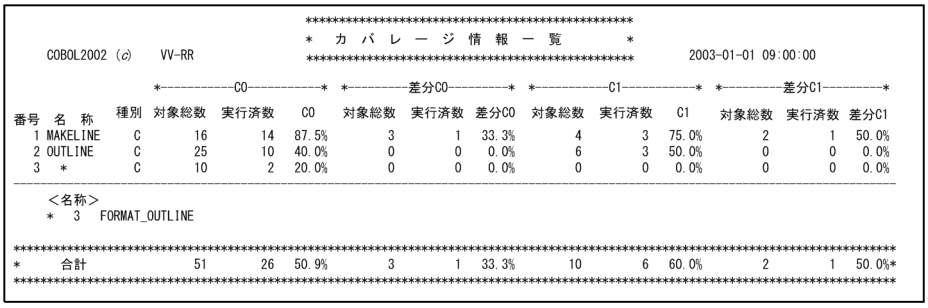
- Windows のプログラム一覧にある［COBOL2002］下のメニューから［カバレージ for COBOL2002］を選び，カバレージウィンドウからカバレージ表示を指定する。
- コマンドプロンプトから cblcl2k コマンドを実行して，カバレージ情報を表示する。

カバレージウィンドウからカバレージ情報をテキスト形式のファイルに出力する方法については，「14.3.3 カバレージ情報」を参照してください。コマンドを使用してカバレージ情報をテキスト形式のファイルに出力する方法については，「15.3　カバレージ情報の表示（テキスト形式）」を参照してください。

カバレージ情報の表示による出力リストの形式は次のとおりです。表示される順序は，選択されたプログラム情報ファイルの順に依存します。スタックコンパイルされている場合は，-Main,System コンパイラオプションを指定したプログラムを，そのプログラム情報ファイル単位内の先頭に表示します。

(1) 翻訳単位の一覧表示

指定された翻訳単位の C0 メジャー，差分 C0 メジャー，C1 メジャー，差分 C1 メジャーを一覧に表示します。



```
                                        ************************************************
                                        *  カ バ レ ー ジ 情 報 一 覧          *
    COBOL2002 (c)    VV-RR              ************************************************      2003-01-01 09:00:00

                    *----------C0----------*  *---------差分C0---------*  *----------C1----------*  *---------差分C1---------*
番号  名  称   種別 対象総数  実行済数    C0   対象総数  実行済数  差分C0  対象総数  実行済数    C1   対象総数  実行済数  差分C1
   1 MAKELINE    C       16        14 87.5%          3         1  33.3%         4         3 75.0%          2         1  50.0%
   2 OUTLINE     C       25        10 40.0%          0         0   0.0%         6         3 50.0%          0         0   0.0%
   3  *          C       10         2 20.0%          0         0   0.0%         0         0  0.0%          0         0   0.0%
------------------------------------------------------------------------------------------------------------------------------

     <名称>
     *  3   FORMAT_OUTLINE

******************************************************************************************************************************
*      合計                51        26  50.9%          3         1  33.3%        10         6 60.0%          2         1  50.0%*
******************************************************************************************************************************
```

表 13-9　プログラムの一覧表示

項目	内容			
翻訳単位	名称		翻訳単位の名称 9 文字以上の名称は，アスタリスク（*）を表示し，＜名称＞のあとに表示する。	
	種別		翻訳単位の種別 P：プログラム C：クラス T：利用者定義関数	
	C0	対象総数	翻訳単位	実行文の数
		実行済数		実行が済んだ実行文の数
		C0		C0 メジャー※
	差分 C0	対象総数	差分	実行文の数
		実行済数		実行が済んだ実行文の数
		差分 C0		差分 C0 メジャー※
	C1	対象総数	翻訳単位	実行できる分岐先の数
		実行済数		実行が済んだ分岐先の数
		C1		C1 メジャー※
	差分 C1	対象総数	差分	実行できる分岐先の数
		実行済数		実行が済んだ分岐先の数
		差分 C1		差分 C1 メジャー※

項目	内容			
合計	C0	対象総数	一覧の全翻訳単位	実行文の数の合計
		実行済数		実行が済んだ実行文の数の合計
		C0		C0 メジャー※
	差分 C0	対象総数	一覧の全翻訳単位の差分	実行文の数の合計
		実行済数		実行が済んだ実行文の数の合計
		差分 C0		差分 C0 メジャー※
	C1	対象総数	一覧の全翻訳単位	実行できる分岐先の数の合計
		実行済数		実行が済んだ分岐先の数の合計
		C1		C1 メジャー※
	差分 C1	対象総数	一覧の全翻訳単位の差分	実行できる分岐先の数の合計
		実行済数		実行が済んだ分岐先の数の合計
		差分 C1		差分 C1 メジャー※

注※

小数点第 2 位以下を切り捨てます。

(2) まとめ情報の表示

指定した翻訳単位の C0 メジャー，差分 C0 メジャー，C1 メジャー，差分 C1 メジャー，S1 メジャー，差分 S1 メジャーを表示します。

```
                              ****************************
                              *      カバレージ情報      *
    COBOL2002 (c)  VV-RR      ****************************      2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :      1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数  :      2

*************************************************************************************************
*                    <C0>    <差分C0>    <C1>    <差分C1>    <S1>    <差分S1>     *
*     対象総数         16           3          4          2          5          0     *
*     実行済数         14           1          3          1          5          0     *
*     未実行数          2           2          1          1          0          0     *
*     カバレージ率   87.5%       33.3%      75.0%      50.0%     100.0%       0.0%     *
*************************************************************************************************
```

表 13–10　まとめ情報の表示の説明

項目		内容	
翻訳単位名の種別		プログラム名・クラス名・利用者定義関数名	
翻訳単位名		翻訳単位の名称	
コンパイル日時		翻訳単位を含む原始プログラムをコンパイルした日時	
テスト日時		翻訳単位を含む原始プログラムのカバレージ情報を最後に蓄積した日時	
変更回数		翻訳単位を含む原始プログラムを変更した回数	
テスト回数		翻訳単位を含む原始プログラムを実行した回数 32,766 を超えた場合は*****	
＜ C0 ＞	対象総数	翻訳単位	実行文の数
	実行済数		実行が済んだ実行文の数
	未実行数		実行していない文の数
	カバレージ率		C0 メジャー※
＜差分 C0 ＞	対象総数	差分	実行文の数
	実行済数		実行が済んだ実行文の数
	未実行数		実行していない実行文の数
	カバレージ率		差分 C0 メジャー※
＜ C1 ＞	対象総数	翻訳単位	実行できる分岐の数
	実行済数		実行が済んだ分岐の数
	未実行数		実行していない分岐の数
	カバレージ率		C1 メジャー※
＜差分 C1 ＞	対象総数	差分	実行できる分岐の数
	実行済数		実行が済んだ分岐の数
	未実行数		実行していない分岐の数
	カバレージ率		差分 C1 メジャー※
＜ S1 ＞	対象総数	翻訳単位	実行できる呼び出し文の数
	実行済数		実行が済んだ呼び出し文の数
	未実行数		実行していない呼び出し文の数
	カバレージ率		S1 メジャー※
＜差分 S1 ＞	対象総数	差分	実行できる呼び出し文の数
	実行済数		実行が済んだ呼び出し文の数

項目		内容	
	未実行数		実行していない呼び出し文の数
	カバレージ率		差分 S1 メジャー※

注※

小数点第 2 位以下を切り捨てます。

注意事項

C0 メジャー，C1 メジャー，差分 C0 メジャー，差分 C1 メジャー，S1 メジャーおよび差分 S1 メジャーのそれぞれの対象総数が 99,999,999 を超えた場合，表示は「********」となります。

(3) 全ソース表示

原始プログラムを表示し，原始プログラムの変更による差分と C0・C1 の実行済みの状態を表示します。

```
                              ******************************
                              *     カバレージ情報         *
    COBOL2002 (c)  VV-RR      ******************************      2003-01-01 09:30:00
------------------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :     1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数  :     2

-----------------------------------<全ソース表示>-----------------------------------------------
メソッド名    :INIT_MAKELINE_F
変更<C0 > <C1 >-----------------------------------------------------------------------------------
              0000021 PROCEDURE  DIVISION.
   *          0000022     INVOKE SUPER  'INIT-COLORS-F'.
   *          0000023     CALL 'CBLEXEC' USING EXEC-NAME-LEN EXEC-NAME EXEC-PARM.
   *          0000024     EXIT  METHOD.
              0000025 END  METHOD  INIT-MAKELINE-F.
              0000026
              0000027 END FACTORY.

メソッド名    :INIT_MAKELINE_O
変更<C0 > <C1 >-----------------------------------------------------------------------------------
              0000042 PROCEDURE  DIVISION.
   *          0000043     INVOKE  SUPER  'INIT-COLORS-O'.
   *          0000044     COMPUTE  MSGCOUNT  =  0.
   *          0000045     EXIT  METHOD.
              0000046 END  METHOD  INIT-MAKELINE-O.

メソッド名    :DRAWLINE
変更<C0 > <C1 >-----------------------------------------------------------------------------------
              0000056 PROCEDURE  DIVISION  USING  I-COLOR.
   *          0000057     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE.
   *          0000058     EVALUATE  MINE
        @     0000059       WHEN  I-COLOR
   *          0000060         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR
              0000061                                    RETURNING  IRO
              0000062         END-INVOKE
   *          0000063           DISPLAY  IRO  'の線を書きました'
        @     0000064       WHEN  OTHER
   *          0000067         COMPUTE  MSGCOUNT  =  MSGCOUNT  +  1
   *          0000068         DISPLAY  '同色の絵の具がパレットにありません'
              0000069     END-EVALUATE.
 Y  *    .    0000070     IF  MSGCOUNT  >  20  THEN
 #  .         0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .         0000072       COMPUTE  MSGCOUNT  =  0
              0000073     END-IF.
 #       @
   *          0000074     EXIT  METHOD.
              0000075 END METHOD  DRAWLINE.
              0000077 END OBJECT.
              0000079 END CLASS MAKELINE.

************************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         16            3          4           2          5           0      *
*     実行済数         14            1          3           1          5           0      *
*     未実行数          2            2          1           1          0           0      *
*     カバレージ率   87.5%       33.3%      75.0%       50.0%      100.0%        0.0%      *
************************************************************************************************
```

表 13-11　全ソース表示の説明

項目	内容
翻訳単位名の種別	プログラム名・クラス名・利用者定義関数名
翻訳単位名	翻訳単位の名称
コンパイル日時	翻訳単位を含む原始プログラムをコンパイルした日時
テスト日時	翻訳単位を含む原始プログラムのカバレージ情報を最後に蓄積した日時
変更回数	翻訳単位を含む原始プログラムを変更した回数

項目	内容
テスト回数	翻訳単位を含む原始プログラムを実行した回数
ソース要素名の種別	プログラム名・メソッド名・利用者定義関数名
ソース要素名	ソース要素の名称
変更	Y：原始プログラムの変更された行
	#：原始プログラムの変更によって影響がある行
<C0 >	*：実行が済んだ文
	.：実行していない文
<C1 >	@：実行が済んだ分岐
	.：実行していない分岐
行番号	コンパイラが振り直した行番号
ソーステキスト	原始プログラムの行

まとめ情報の項目は，「(2)　まとめ情報の表示」を参照してください。

注意事項

- C1 に該当する明示的な文がないときは，空白行を表示して C1 の情報を表示します。

（例 1）ELSE がない IF 文

ELSE がない IF 文は，IF 文の下に空白行を示します。IF 文がネストするときは，最外の IF 文の下に C1 に該当する空白行を示します。

(a)は 111200 行の IF 文の「省略された ELSE」の C1 を示します。

(b)は 111000 行の IF 文の「省略された ELSE」の C1 を示します。

```
変更<C0 > <C1 >-----------------------------------------------------------------
    .          0111000     IF FLAG-1 = SPACE
         .     0111100       THEN
    .          0111200         IF FLAG-2 = SPACE
         .     0111300           THEN
    .          0111400             PERFORM 処理１
    .          0111500             MOVE '10' TO 番号 OF 集計レコード
    .          0111600             PERFORM 処理２
    .          0111700             MOVE 1 TO FLAG-3
               0111800         END-IF
               0111900     END-IF.
(a)      .
(b)      .
```

（例 2）うち PERFORM 文

UNTIL 指定のある「うち PERFORM 文」には「ループする」と「ループしない」の 2 方向の分岐があり，両方の C1 を表示します。

(a)は 0112000 行の PERFORM 文の「ループする」ときの分岐先の C1 の行です。

(b)は 0112000 行の PERFORM 文の「ループしない」ときの分岐先の C1 を空白行で示します。

```
変更<C0 > <C1 >-------------------------------------------------------------------------
    .          0112000     PERFORM WITH TEST BEFORE
               0112100       UNTIL リターンコード OF テーブルパラメタ NOT = ZERO
(a) .    .     0112200         MOVE  保存キー TO テーブルキー
    .          0112300         PERFORM 処理３
               0112400     END-PERFORM.
(b)        .
```

（例 3）そと PERFORM 文

UNTIL 指定のある「そと PERFORM 文」は「ループしない」ときの C1 を PERFORM 文の下に表示します（「ループする」の分岐先の C1 は PERFORM 文で実行される文に表示します）。

(a)は 0113000 行の PERFORM 文の「ループしない」ときの分岐先の C1 を示します。

```
変更<C0 > <C1 >-------------------------------------------------------------
    .    .     0113000    PERFORM 処理４ WITH TEST
(a)        .
                0113100    AFTER UNTIL (情報フラグ = C-ON) OR
                0113200                (終了フラグ = C-ON)
```

(4)　差分ソース表示

原始プログラムの修正によって生じた差分の実行文に対して，変更行，変更影響行，C0 実行済み，C1 実行済みを明示します。

```
                              ****************************
                              *     カバレージ情報       *
    COBOL2002 (c)  VV-RR      ****************************        2003-01-01 09:30:00
-------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :     1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数  :     2

-----------------------------------<差分表示>----------------------------------------------
メソッド名    : DRAWLINE
変更<C0 > <C1 >-------------------------------------------------------------------------
 Y  *     .    0000070     IF  MSGCOUNT  >  20  THEN
 #  .          0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .          0000072       COMPUTE  MSGCOUNT  =  0
               0000073     END-IF.
 #       @

*******************************************************************************************
*                    <C0>    <差分C0>    <C1>    <差分C1>    <S1>    <差分S1>     *
*     対象総数         16           3          4          2          5          0     *
*     実行済数         14           1          3          1          5          0     *
*     未実行数          2           2          1          1          0          0     *
*     カバレージ率   87.5%       33.3%      75.0%      50.0%     100.0%       0.0%     *
*******************************************************************************************
```

各項目の内容については，「(3)　全ソース表示」を参照してください。

(5)　未実行表示

原始プログラムの実行していない文を表示します。

```
                              ******************************
                              *       カバレージ情報       *
     COBOL2002 (c)  VV-RR     ******************************          2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数     :      1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数   :      2

-----------------------------------<未実行表示>----------------------------------------
メソッド名    : DRAWLINE
変更<C0 > <C1 >-------------------------------------------------------------------------------
 Y  *     .     0000070     IF  MSGCOUNT  >  20  THEN
 #  .           0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .           0000072       COMPUTE  MSGCOUNT  =  0

*************************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         16            3          4            2          5            0      *
*     実行済数         14            1          3            1          5            0      *
*     未実行数          2            2          1            1          0            0      *
*     カバレージ率   87.5%       33.3%       75.0%        50.0%      100.0%         0.0%     *
*************************************************************************************************
```

各項目の内容については，「(3)　全ソース表示」を参照してください。

(6)　差分未実行ソース表示

原始プログラムの差分未実行ソースを表示します。

```
                              ******************************
                              *       カバレージ情報       *
     COBOL2002 (c)  VV-RR     ******************************          2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数     :      1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数   :      2

-----------------------------------<差分未実行表示>----------------------------------------
メソッド名    : DRAWLINE
変更<C0 > <C1 >-------------------------------------------------------------------------------
 Y  *     .     0000070     IF  MSGCOUNT  >  20  THEN
 #  .           0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .           0000072       COMPUTE  MSGCOUNT  =  0

*************************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         16            3          4            2          5            0      *
*     実行済数         14            1          3            1          5            0      *
*     未実行数          2            2          1            1          0            0      *
*     カバレージ率   87.5%       33.3%       75.0%        50.0%      100.0%         0.0%     *
*************************************************************************************************
```

各項目の内容については，「(3)　全ソース表示」を参照してください。

(7)　呼び出し文ソース表示

原始プログラムの S1 カバレージ対象の呼び出し文を表示します。

```
                              ********************************
                              *      カバレージ情報          *
     COBOL2002 (c)  VV-RR     ********************************      2003-01-01 09:30:00
------------------------------------------------------------------------------------------------------------
  クラス名        : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                    変更回数    :     1
  テスト日時      : 2003-01-01 09:15:00                                  テスト回数  :     2

-----------------------------------<呼び出し文表示>-----------------------------------------
メソッド名    : INIT_MAKELINE_F
変更<C0 > <C1 >------------------------------------------------------------------------------------------
     *          0000022     INVOKE  SUPER  'INIT-COLORS-F'.
     *          0000023     CALL 'CBLEXEC' USING EXEC-NAME-LEN EXEC-NAME EXEC-PARM.

メソッド名    : INIT_MAKELINE_O
変更<C0 > <C1 >------------------------------------------------------------------------------------------
    *          0000043     INVOKE  SUPER  'INIT-COLORS-O'.

メソッド名    : DRAWLINE
変更<C0 > <C1 >------------------------------------------------------------------------------------------
    *          0000057     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE.
    *     @    0000060         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR


************************************************************************************************
*                    <C0>    <差分C0>    <C1>    <差分C1>    <S1>    <差分S1>     *
*     対象総数         16           3          4          2          5          0     *
*     実行済数         14           1          3          1          5          0     *
*     未実行数          2           2          1          1          0          0     *
*     カバレージ率    87.5%       33.3%      75.0%      50.0%     100.0%       0.0%     *
************************************************************************************************
```

各項目の内容については，「(3)　全ソース表示」を参照してください。

13.2.6　カバレージ情報の表示（CSV 形式）

次の方法で，プログラム情報ファイルに蓄積されたカバレージ情報を，カバレージ統計情報 CSV ファイル，およびソースカバレージ情報 CSV ファイルへ出力できます。

- コマンドプロンプトから cblcc2k コマンドを実行して，カバレージ情報を表示する。

カバレージ情報を CSV ファイルに出力する方法については，「15.4　カバレージ情報の表示（CSV 形式）」を参照してください。

カバレージ情報の表示による出力ファイル（CSV）の形式は次のとおりです。

注意事項

項目および出力内容は，ダブルコーテーション（"）で囲んで CSV ファイルに出力します。出力内容にダブルコーテーション（"）を含む場合は 2 個続けて（""）出力します。出力内容が何もない場合は，ダブルコーテーション（"）も出力しません。

（例）

出力内容が「XXX"YYY」の場合，CSV ファイルには「"XXX""YYY"」と出力します。

(1)　カバレージ統計情報の出力

指定されたソースファイル単位または翻訳単位の C0 メジャー，C1 メジャー，S1 メジャーをカバレージ統計情報 CSV ファイルに出力します。

```
"バージョン","出力日時"                                                
```

項目		出力内容
		翻訳単位の場合 　翻訳単位の名称
	コンパイル日時	翻訳単位を含む原始プログラムをコンパイルした日時
	テスト日時	ソースファイル単位の場合 　-（ハイフン） 翻訳単位の場合 　翻訳単位を含む原始プログラムのカバレージ情報を最後に蓄積した日時 　カバレージ情報を蓄積していない場合は-（ハイフン）
	変更回数	ソースファイル単位の場合 　-（ハイフン） 翻訳単位の場合 　翻訳単位を含む原始プログラムを変更した回数 　32,767 を超えた場合は*
	テスト回数	ソースファイル単位の場合 　-（ハイフン） 翻訳単位の場合 　翻訳単位を含む原始プログラムを実行した回数 　32,766 を超えた場合は*
	種別	ソースファイル単位の場合 　-（ハイフン） 翻訳単位の場合 　翻訳単位の種別 　P：プログラム 　C：クラス 　T：利用者定義関数
	C0 対象総数	実行文の数
	C0 実行済数	実行が済んだ実行文の数
	C0 カバレージ率	C0 メジャー※ C0 対象総数が 0 の場合は-（ハイフン）
	差分 C0 対象総数	差分に含まれる実行文の数
	差分 C0 実行済数	差分に含まれる実行が済んだ実行文の数
	差分 C0 カバレージ率	差分 C0 メジャー※ 差分 C0 対象総数が 0 の場合は-（ハイフン）
	C1 対象総数	実行できる分岐の数
	C1 実行済数	実行が済んだ分岐の数

項目		出力内容
	C1 カバレージ率	C1 メジャー※ C1 対象総数が 0 の場合は-（ハイフン）
	差分 C1 対象総数	差分に含まれる実行できる分岐の数
	差分 C1 実行済数	差分に含まれる実行が済んだ分岐の数
	差分 C1 カバレージ率	差分 C1 メジャー※ 差分 C1 対象総数が 0 の場合は-（ハイフン）
	S1 対象総数	実行できる呼び出し文の数
	S1 実行済数	実行が済んだ呼び出し文の数
	S1 カバレージ率	S1 メジャー※ S1 対象総数が 0 の場合は-（ハイフン）
	差分 S1 対象総数	差分に含まれる実行できる呼び出し文の数
	差分 S1 実行済数	差分に含まれる実行が済んだ呼び出し文の数
	差分 S1 カバレージ率	差分 S1 メジャー※ 差分 S1 対象総数が 0 の場合は-（ハイフン）

注※

小数点第 2 位以下を切り捨てます。

注意事項

- C0 メジャー，C1 メジャー，差分 C0 メジャー，差分 C1 メジャー，S1 メジャーおよび差分 S1 メジャーのそれぞれの対象総数，または実行済数が 99,999,999 を超えた場合，出力内容は「*」となります。
- プログラム情報ファイルを指定した場合，カバレージ統計情報 CSV ファイルには，指定されたプログラム情報ファイルの順に出力します。プログラム情報ファイルに複数の翻訳単位があるときは，COBOL ソースファイルに記述された順に出力します。
- プログラム情報ファイル格納フォルダパスを指定した場合，カバレージ統計情報 CSV ファイルには，次の順序で出力します。

  ・同一フォルダ内のプログラム情報ファイルおよびサブフォルダは，ファイル名およびフォルダ名の文字コードの昇順で出力します。

  ・同一フォルダにプログラム情報ファイルとサブフォルダが混在する場合，フォルダ直下にあるプログラム情報ファイルを先に出力します。

  ・サブフォルダ下にあるプログラム情報ファイルは，上位フォルダから順に出力します。

  ・プログラム情報ファイルに複数の翻訳単位がある場合は，COBOL ソースファイルに記述された順に出力します。
- 翻訳単位のカバレージ統計情報とソースファイル単位のカバレージ統計情報を同時に出力する場合，同一ソースファイルに対するカバレージ統計情報は，翻訳単位のカバレージ統計情報のあとにソースファイル単位のカバレージ統計情報を出力します。

(2) ソースカバレージ情報の出力

原始プログラム，原始プログラムの変更による差分，C0 と C1 の実行済みの状態をソースカバレージ情報 CSV ファイルに出力します。また，差分ソース，未実行ソース，差分未実行ソースまたは呼び出し文ソースのどれかに該当する行を識別します。

```
"バージョン","出力日時","コンパイル日時","プログラム情報ファイル格納先","プログラム情報ファイル名"                                                
```

表 13–13　ソースカバレージ情報 CSV ファイルの各項目の出力内容

項目		出力内容
ヘッダ	バージョン	COBOL2002 のバージョン情報 「COBOL2002 （c） VV-RR」の形式で出力する。 COBOL2002：COBOL2002 であることの記述 （c）：COBOL2002 の識別記号 COBOL2002 の識別記号については「付録 F.2　このマニュアルでの表記」を参照 VV-RR：COBOL2002 のバージョン番号
	出力日時	ソースカバレージ情報 CSV ファイルに出力した日時
	コンパイル日時	翻訳単位を含む原始プログラムをコンパイルした日時
	プログラム情報ファイル格納先	プログラム情報ファイルが格納されている場所の絶対パス
	プログラム情報ファイル名	プログラム情報ファイル名
ソースカバレージ情報	翻訳単位名	翻訳単位の名称
	ソース要素種別	ソース要素の種別 PPROG：一番外側のプログラム IPROG：入れ子プログラム METHOD：ファクトリメソッドまたはオブジェクトメソッド FUNC：利用者定義関数名
	ソース要素名	ソース要素の名称
	未実行ソース	UNEX：原始プログラムの実行していない文の行 該当しない場合は何も出力されない
	差分ソース	DIFF：原始プログラムの修正によって生じた差分の行 該当しない場合は何も出力されない
	差分未実行ソース	UNEXDIFF：原始プログラムの修正によって生じた差分に含まれる，実行していない文の行 該当しない場合は何も出力されない
	呼び出し文ソース	ACT：原始プログラムの S1 カバレージ対象の呼び出し文の行 該当しない場合は何も出力されない
	コメント行	COMMENT：コメント行（注記行） 該当しない場合は何も出力されない
	変更	Y：原始プログラムの変更された行
		#：原始プログラムの変更によって影響がある行
	C0	*：実行が済んだ文
		.：実行していない文
	C1	@：実行が済んだ分岐

項目		出力内容
		.：実行していない分岐
	行番号	コンパイラが振り直した行番号
	ソーステキスト	原始プログラムの行

注意事項

- 原始プログラムの 1 行に複数の実行可能文が含まれる場合，C0，C1 には，実行済みかどうかを示す記号を実行可能文の個数分出力します。
- 原始プログラムの末尾の半角空白文字は，ソーステキストとして出力しません。
- IF 文の ELSE 側を省略した場合など，暗黙的に生成される C1 メジャーの対象となるときは，行番号は空になります。行番号が空になる行の条件は，「13.2.5　カバレージ情報の表示（テキスト形式）」の「(3)　全ソース表示」を参照してください。

13.2.7　カバレージ情報の操作

次のどちらかの方法で，プログラム情報ファイルのカバレージ情報を操作できます。

- Windows のプログラム一覧にある［COBOL2002］下のメニューから［カバレージ for COBOL2002］を選び，カバレージウィンドウからカバレージの操作を指定する。
- コマンドプロンプトから cblca2k コマンドを実行して，カバレージ情報を操作する。

カバレージウィンドウからカバレージ情報を操作する方法については，「14.3.3　カバレージ情報」を参照してください。コマンドを使用してカバレージ情報を操作する方法については，「15.5　カバレージ情報の操作」を参照してください。

カバレージ情報の操作には次の機能があります。

(1)　カバレージ情報の０％化

実行が済んだ文を未実行文に戻します。再度のテストが必要なときに，蓄積したカバレージ情報を０％に戻すときに使用できます。

次のカバレージ情報が０％になります。

- C0 メジャー
- 差分 C0 メジャー
- C1 メジャー
- 差分 C1 メジャー
- S1 メジャー

- 差分 S1 メジャー

(2)　差分カバレージ情報の０％化

差分に含まれる実行が済んだ文をすべて未実行文に戻します。次のカバレージ情報が０％になります。

- 差分 C0 メジャー
- 差分 C1 メジャー
- 差分 S1 メジャー

(3)　差分カバレージ情報のクリア

差分の対象となる文を差分とみなさないようにします。次の情報が０になります。

- 差分の実行文の数
- 差分の実行できる分岐の数
- 差分の実行できる呼び出し文の数

(4)　カバレージ情報のマージ

複数のプログラム情報ファイルのカバレージ情報を，一つのプログラム情報ファイルへマージします。実行環境をコピーして複数のテスト環境を作成して別々に蓄積したカバレージ情報を，一つのプログラム情報ファイルにマージするときに使用できます。マージするプログラム情報ファイルのコンパイル時間は，すべて同じである必要があります。

注意事項

カバレージ情報の操作は，プログラム情報ファイルの単位で行います。プログラム情報ファイルに複数の翻訳単位がある場合は，すべてを対象にします。

13.3　カウント情報の表示

13.3.1　カウント情報

プログラムを実行させたときの各文の実行回数です。

実行回数は，ブロックごとにカウントします。カバレージのブロックは，順次実行される無条件文の集まりで，条件文の条件句の直前までが原則として一つのブロックとなります。ブロックの先頭の文を実行したときにブロック全体が実行されたと判断します。ブロック中の文で実行時エラーが発生した場合も，そのブロックは実行されたとします。

13.3.2　カウント情報表示の手順

カウント情報を表示する手順を次に示します。

1. プログラムのコンパイル

   カウント情報を表示するプログラムは，-CVInf コンパイラオプションを指定してコンパイルします。コンパイラオプションについては，「2.3　コンパイラオプション」を参照してください。

2. プログラムの実行

   次の方法でプログラムを実行して，カウント情報をテキスト形式のカウント情報リストファイルへ出力します。

   - Windows のプログラム一覧にある［COBOL2002］下のメニューから［カバレージ for COBOL2002］を選び，カバレージウィンドウからプログラムを実行する。
   - コマンドプロンプトから cblcn2k コマンドを使用して，プログラムを実行する。
   - 「プログラムからの連動実行」の設定を行い，プログラムを実行する。

注意事項

- マルチスレッドプログラムは，COBOL プログラムの全スレッドがカウント情報表示の対象になります。
- 実行可能ファイル，DLL に含まれる翻訳単位とプログラム情報ファイルは，コンパイルで同時に作成されていなければなりません（同一のコンパイル時間である必要があります）。
- プログラムの実行が終了したとき，カウントした実行回数を表示します。プログラムが実行時エラーで終了した場合も，終了までにカウントした実行回数を表示します。

13.3.3　カウント情報の表示

カウント情報の表示によって，ファイルへ出力される出力リストを説明します。

表示される順序は，実行可能ファイル，DLL ファイル（DLL ファイル名でソート）の順です。実行可能ファイル，DLL ファイル内の翻訳単位はソートされて表示されます。ただし，-Main 指定のある翻訳単位は，最初に表示されます。翻訳単位内の表示順は，プログラムに書かれている順です。

カバレージウィンドウからカウント情報を表示する方法については，「14.3.5　カウント情報」を参照してください。コマンドまたはプログラムの連続実行を使用してカウント情報を表示する方法については，「15.6　カウント情報の表示」を参照してください。

```
                          ****************************
                          *       カウント情報       *
    COBOL2002 (c)  VV-RR  ****************************          2003-01-01 09:30:00
------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00
  実行日時       : 2003-01-01 09:15:00

------------------------------------------------------------------------------------------
メソッド名    : INIT_MAKELINE_F
  実行回数      ----------------------------------------------------------------------------
                0000021 PROCEDURE  DIVISION.
        1       0000022     INVOKE  SUPER  'INIT-COLORS-F'.
        1       0000023     EXIT  METHOD.
                0000024 END  METHOD  INIT-MAKELINE-F.
                0000025
                0000026 END  FACTORY.

メソッド名    : INIT_MAKELINE_0
  実行回数      ----------------------------------------------------------------------------
                0000042 PROCEDURE  DIVISION.
        2       0000043     INVOKE  SUPER  'INIT-COLORS-0'.
        2       0000044     COMPUTE  MSGCOUNT  =  0.
                0000045     EXIT  METHOD.
                0000046 END  METHOD  INIT-MAKELINE-0.

メソッド名    : DRAWLINE
  実行回数      ----------------------------------------------------------------------------
                0000056 PROCEDURE  DIVISION  USING  I-COLOR.
        6       0000057     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE.
        6       0000058     EVALUATE  MINE
                0000059       WHEN  I-COLOR
        4       0000060         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR
                0000061                                    RETURNING IRO
        4       0000063         DISPLAY  IRO  'の線を書きました'
                0000064         WHEN  OTHER
        2       0000067           COMPUTE  MSGCOUNT  =  MSGCOUNT  +  1
                0000068           DISPLAY  '同色の絵の具がパレットにありません'
                0000069     END-EVALUATE.
        6       0000070     IF  MSGCOUNT  >  20  THEN
        0       0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
                0000072       COMPUTE  MSGCOUNT  =  0
                0000073     END-IF.
        6       0000074     EXIT  METHOD.
                0000075 END  METHOD  DRAWLINE.
                0000077 END  OBJECT.
                0000079 END  CLASS  MAKELINE.
```

表 13-14　カウント情報の表示

項目	備考
翻訳単位名の種別	プログラム名・クラス名・利用者定義関数名
翻訳単位名	翻訳単位の名称
コンパイル日時	翻訳単位を含む原始プログラムをコンパイルした日時
実行日時	翻訳単位を含むプログラムを実行した日時

項目	備考
ソース要素名の種別	プログラム名・メソッド名・利用者定義関数名
ソース要素名	ソース要素の名称
行番号	コンパイラが振り直した行番号
実行回数	文のブロックが実行された回数。0～2,147,483,647 の範囲で表示する 2,147,483,647 を超えた場合は，実行回数に代えて"**********"を表示する
ソーステキスト	原始プログラムの行

14 GUI モードによるカバレージ

GUI モードによるカバレージでは，必要な操作をウィンドウで指定できます。GUI モードによるカバレージについて説明します。

14.1　概要

カバレージは，テスト工程の管理作業をサポートします。カバレージには，テストの進捗状況を定量的に表すカバレージ情報，文の実行回数をカウントするカウント情報の表示機能があります。

GUI モードによるカバレージでは，必要な操作をウィンドウで指定できます。GUI モードによるカバレージについて説明します。

14.1.1　カバレージの手順

GUI モードのカバレージでの作業の流れを説明します。

1. -CVInf コンパイラオプションを指定して，コンパイルする。
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。
2. カバレージ情報の採取，およびプログラムの実行のための環境変数を設定する。
3. カバレージを起動する。
4. カバレージの機能を使用する。
   メニューから機能を選択して実行します。次の機能が選択できます。
   - カバレージ情報の蓄積
   - カバレージ情報の表示
   - カバレージ情報のマージ
   - カバレージ情報の操作
   - カウント情報の表示
5. カバレージを終了する。

14.1.2　環境変数の指定

(1)　CBLPIDIR

作成または更新したプログラム情報ファイルのあるフォルダを指定します。次の順番でプログラム情報ファイルを検索します。

1. 環境変数 CBLPIDIR で指定したフォルダ
2. 実行可能ファイルまたは DLL ファイルのあるフォルダ

プログラムの実行に関するその他の環境変数については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

14.2　開始と終了

14.2.1　開発マネージャからの開始

開発マネージャで［カバレージ］メニューを選択します。

14.2.2　［COBOL2002］下のメニューからの開始

Windows のプログラム一覧にある［COBOL2002］下のメニューから［カバレージ for COBOL2002］を選び，カバレージを起動します。

14.2.3　終了

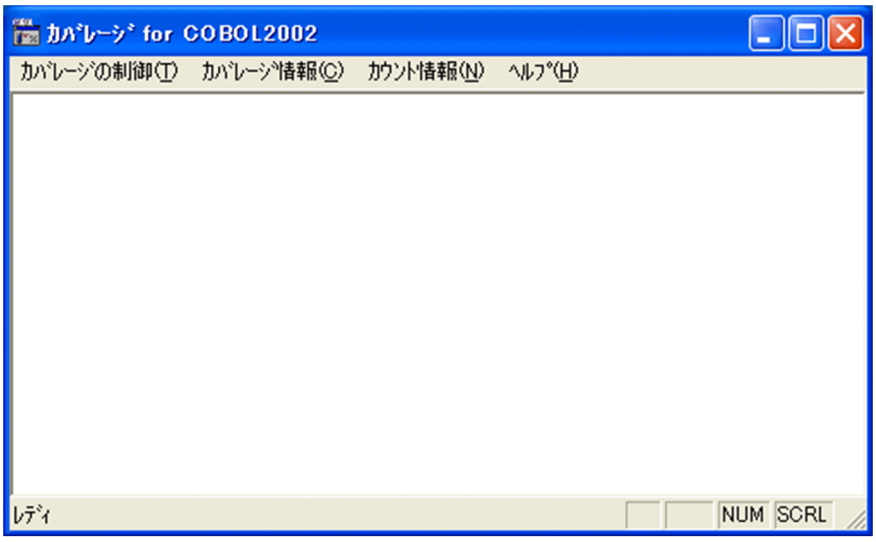
カバレージのメニューで指定することで，カバレージウィンドウが閉じられ終了します。

14.3　カバレージのウィンドウ

14.3.1　ウィンドウ

カバレージを操作するウィンドウは次のとおりです。操作の結果のメッセージがウィンドウへ表示されます。

図 14-1　カバレージのウィンドウ



14.3.2　メニュー

カバレージウィンドウのメインメニューは次のとおりです。

表 14-1　カバレージウィンドウのメニュー一覧

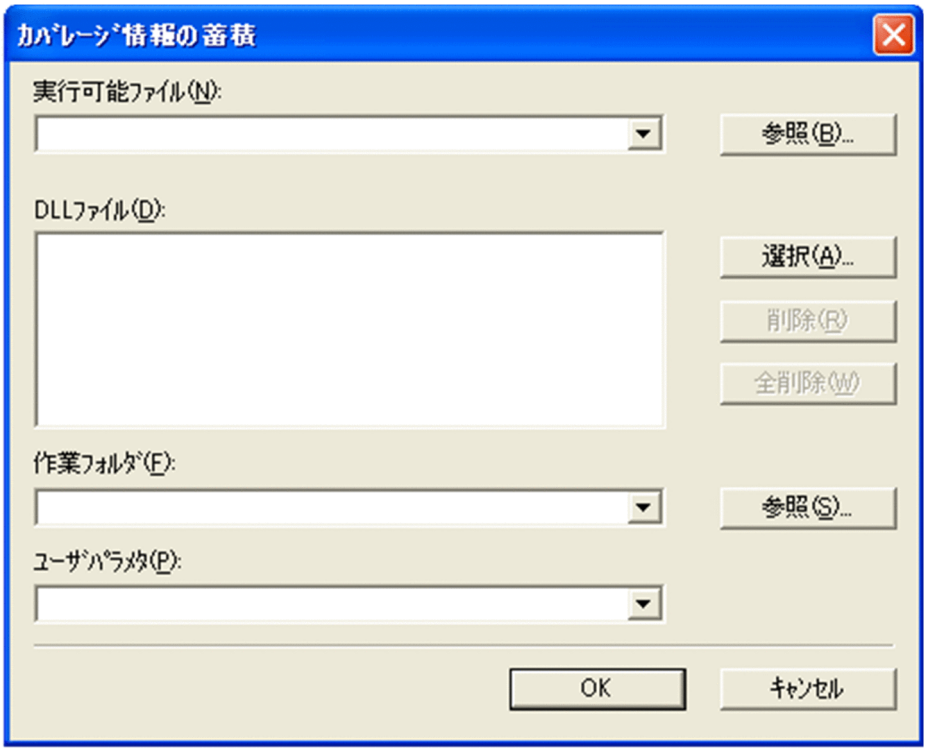
メニューバー	メニュー項目	機能
カバレージの制御(T)	カバレージの終了(X)	カバレージを終了する。
カバレージ情報(C)	蓄積(A)	カバレージ情報を蓄積する。
	表示(D)	カバレージ情報を表示する。
	マージ(M)	カバレージ情報をマージする。
	操作(T)	カバレージ情報を操作する。
カウント情報(N)	表示(D)	カウント情報を表示する。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

14.3.3　カバレージ情報

(1)　蓄積

1. [カバレージ情報] メニューから [蓄積] を選ぶ。

   カバレージ情報の蓄積ダイアログボックスが表示されます。



2. 実行可能ファイル名を指定する。

   [参照] ボタンで実行可能ファイル選択用のダイアログボックスを表示し，ファイル名を指定できます。

   直接，実行可能ファイルのパス名を指定することもできます。

   以前に選択した実行可能ファイルの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

3. DLL ファイル名を指定する。

   「DLL ファイル」の一覧に表示されるファイル名がカバレージ情報蓄積の対象になります。

   DLL ファイルは次の方法で指定します。

   [選択]

   カバレージ情報蓄積の対象の DLL ファイルを追加します。[選択] ボタンを選ぶと，DLL ファイル選択のダイアログボックスが表示され，DLL のファイル名を指定できます。

   [削除]

   DLL ファイルをテストデバッグの対象から外します。DLL ファイルを選択し，[削除] ボタンを選びます。

［全削除］

「DLL ファイル」の一覧に表示されたファイルを，すべてカバレージ情報蓄積の対象から外します。

4. 作業フォルダを設定する。

ダイアログボックスの表示直後は，何も設定されていません。何も設定しなければ，実行可能ファイルがあるフォルダが作業フォルダとなります。

作業フォルダを設定したい場合は，［参照］ボタンでダイアログボックスを表示し，フォルダ名を指定します。選択したフォルダは，作業フォルダの領域に表示されます。直接入力することもできます。

以前に選択した作業フォルダの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

5. ユーザパラメタを設定する。

実行可能ファイルに渡すパラメタを設定します。

以前に選択したユーザパラメタの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

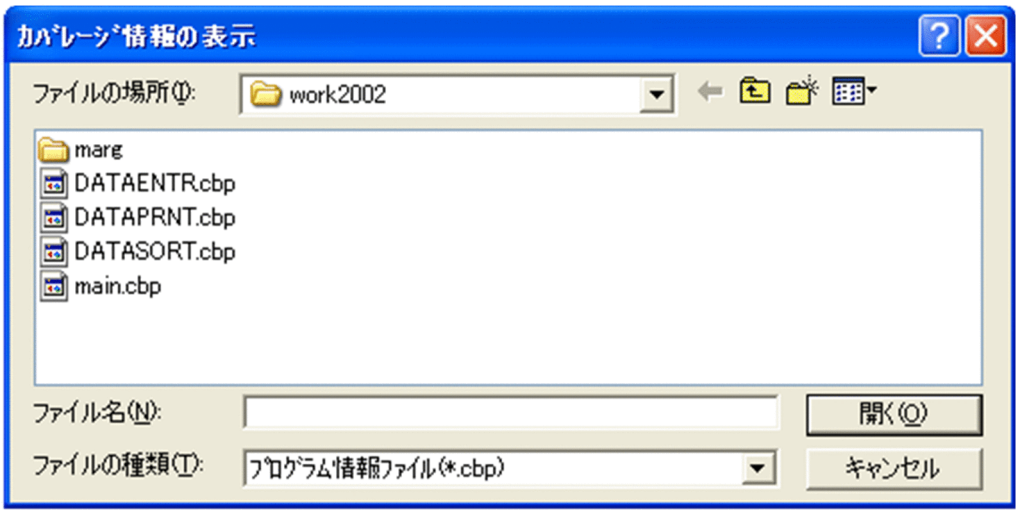
6. ［OK］ボタンを選ぶ。

プログラムの実行を開始します。カバレージ情報の蓄積の結果はカバレージウィンドウへ出力します。

(2) 表示

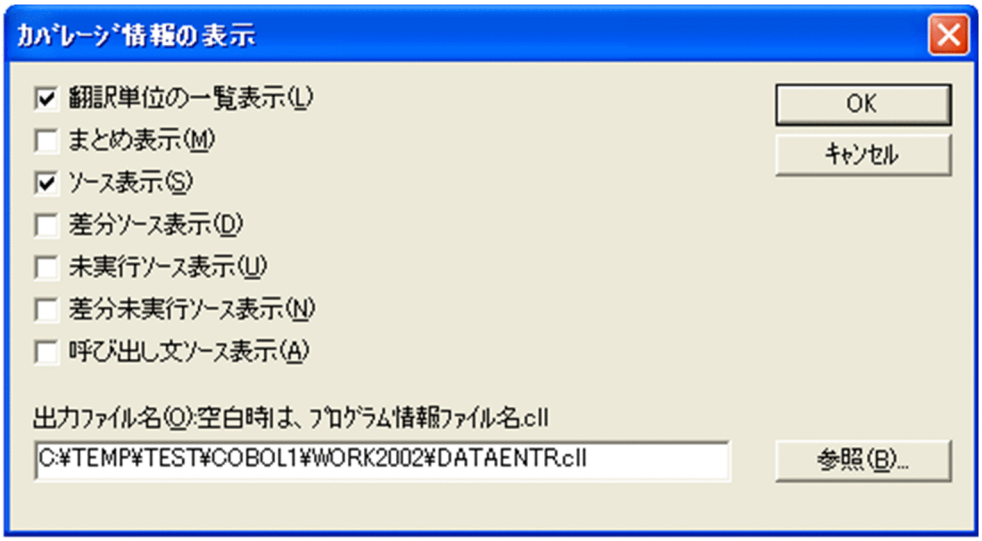
1. ［カバレージ情報］メニューから［表示］を選ぶ。

プログラム情報ファイルの選択ダイアログボックスが表示されます。プログラム情報ファイルの選択ダイアログボックスから，カバレージ情報の表示を行うプログラム情報ファイルを選択します。



2. ［開く］ボタンを選ぶ。

カバレージ情報の表示ダイアログボックスが表示されます。



カバレージ情報の表示の種類と出力ファイルを選択します。

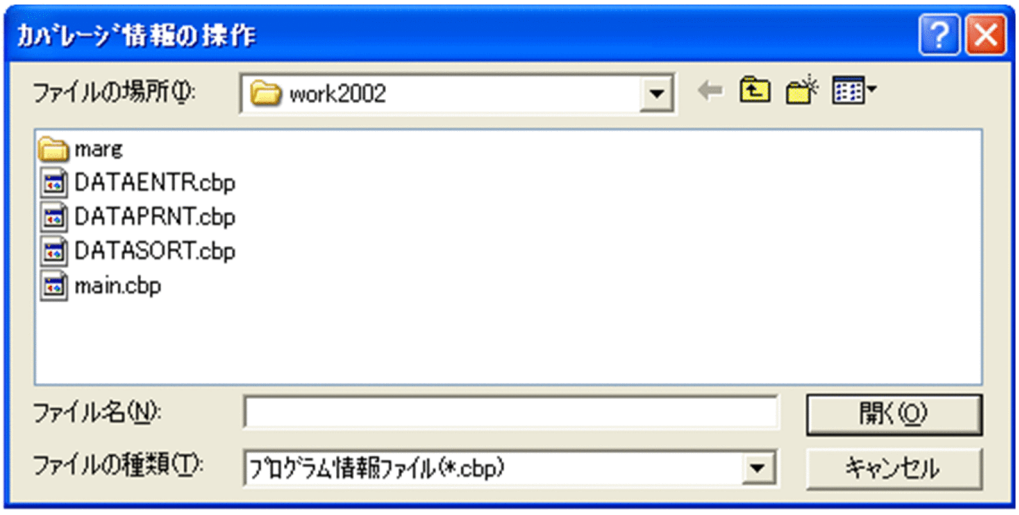
3. [OK] ボタンを選ぶ。

操作の結果がカバレージウィンドウに表示されます。カバレージ情報の表示結果は，指定した出力ファイルへ出力されます。

(3) 操作

1. [カバレージ情報] メニューから [操作] を選ぶ。

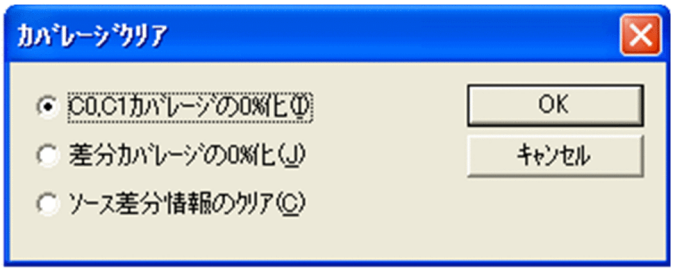
プログラム情報ファイルの選択ダイアログボックスが表示されます。



プログラム情報ファイルの選択ダイアログボックスから，カバレージ情報の操作を行うプログラム情報ファイルを選択します。

2. [開く] ボタンを選ぶ。

カバレージクリアダイアログボックスが表示されます。



カバレージクリアダイアログボックスで，カバレージ情報の操作の種類を選択します。

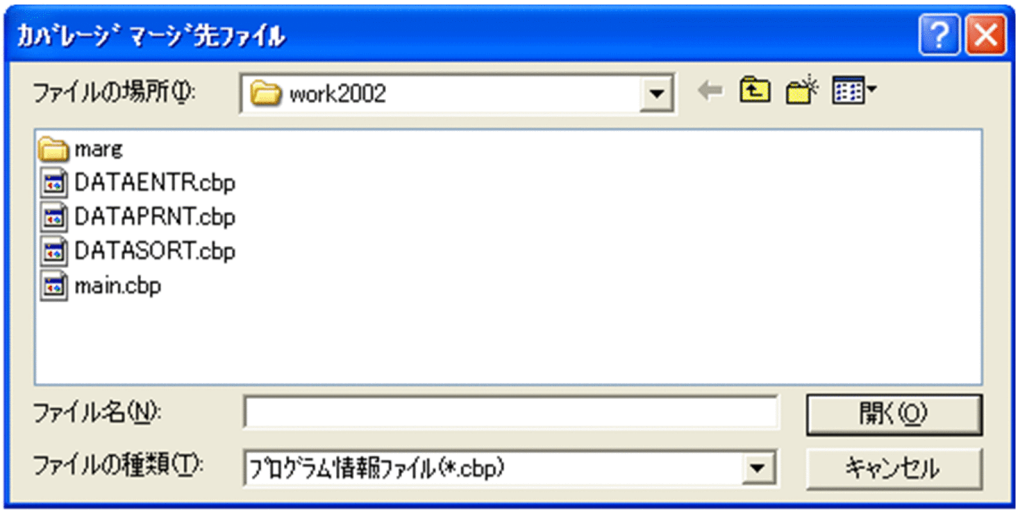
3. [OK] ボタンを選ぶ。

操作の結果がカバレージウィンドウに表示されます。

14.3.4　カバレージ情報のマージ

1. [カバレージ情報] メニューから [マージ] を選ぶ。

カバレージ情報のマージで，マージの結果を出力するプログラム情報ファイルの選択ダイアログボックスが表示されます。



プログラム情報ファイルを選択します。

2. [開く] ボタンを選ぶ。

カバレージ情報のマージで，入力するプログラム情報ファイルの選択ダイアログボックスが表示されます。

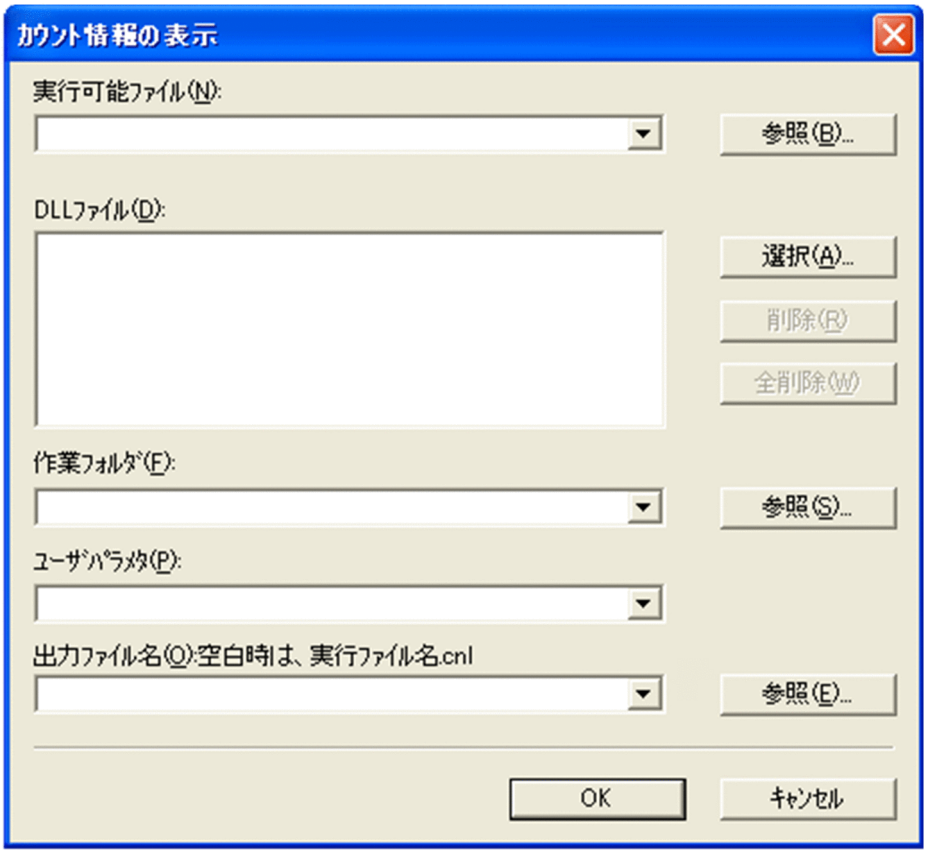


プログラム情報ファイルを選択します。

3. [開く] ボタンを選ぶ。
   操作の結果がカバレージウィンドウに表示されます。

14.3.5 カウント情報

1. [カウント情報] メニューから [表示] を選ぶ。
   カウント情報の表示ダイアログボックスが表示されます。



2. 実行可能ファイル名を指定する。

［参照］ボタンで実行可能ファイル選択用のダイアログボックスを表示し，ファイル名を指定できます。

直接，実行可能ファイルのパス名を指定することもできます。

以前に選択した実行可能ファイルの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

3. DLL ファイル名を指定する。

「DLL ファイル」の一覧に表示されるファイル名がテストデバッグの対象になります。

DLL ファイルは次の方法で指定します。

［選択］

カウント情報表示の対象の DLL ファイルを追加します。［選択］ボタンを選ぶと，DLL ファイル選択のダイアログボックスが表示され，DLL のファイル名を指定できます。

［削除］

DLL ファイルをカウント情報表示の対象から外します。DLL ファイルを選択し，［削除］ボタンを選びます。

［全削除］

「DLL ファイル」の一覧に表示されたファイルを，すべてカウント情報表示の対象から外します。

4. 作業フォルダを設定する。

ダイアログボックス表示直後は，何も設定されていません。何も設定しなければ，実行可能ファイルがあるフォルダが作業フォルダとなります。

作業フォルダを設定するときは，［参照］ボタンでダイアログボックスを表示し，フォルダ名を指定します。選択したフォルダは，作業フォルダの領域に表示されます。直接入力することもできます。

以前に選択した作業フォルダの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

5. ユーザパラメタを設定する。

実行可能ファイルに渡すユーザパラメタを設定します。

以前に選択したユーザパラメタの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

6. 出力ファイル名を設定する。

カウントの結果を格納するカウント情報リストのパス名を設定します。

［参照］ボタンでダイアログボックスを表示し，フォルダを選択し，ファイル名を指定します。設定したファイルのパス名は，出力ファイル名の領域に表示されます。直接入力することもできます。

出力ファイル名に何も指定しなかったとき，実行可能ファイル名の拡張子を.cnl にしたパス名で出力されます。

以前に選択した出力ファイルの履歴を 10 回分まで表示し，そこから選択して実行することもできます。

7. ［OK］ボタンを選ぶ。

操作の結果がカバレージウィンドウに表示されます。カウント情報は，指定した出力ファイルへ出力します。

15 バッチモードによるカバレージ

バッチモードによるカバレージの方法について説明します。

カバレージは，テスト工程の管理作業をサポートします。カバレージには，テストの進捗状況を定量的に表すカバレージ情報，文の実行回数をカウントするカウント情報の表示機能があります。

バッチモードでは，一括してカバレージを実行させることができます。

15.1 概要

コマンドプロンプトから入力するコマンドの指示によって，カバレージの機能を操作できます。プログラムからカバレージを連動実行させ，一括してカバレージの蓄積およびカウント情報の表示をすることもできます。一度実行を開始すれば，利用者の操作を必要としないので，効率良く大量のプログラムのカバレージが実行できます。

15.2.1　コマンドによる方法

コマンドプロンプトからコマンドを入力して，カバレージ情報を蓄積します。

カバレージ情報の詳細については，「13.2.1　カバレージ情報」を参照してください。

(1)　カバレージ情報の蓄積の手順

バッチモードでカバレージ情報を蓄積するときの作業の流れを説明します。

1. -CVInf コンパイラオプションを指定して，プログラムをコンパイルする。
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。
2. カバレージ情報の蓄積または，カバレージ情報の蓄積対象プログラムの実行に必要な環境変数を設定する。
   プログラムの実行に必要な環境変数とカバレージに必要な環境変数を設定します。カバレージに必要な環境変数については，「15.2.1　コマンドによる方法」の「(3)　環境変数の指定」を参照してください。プログラムの実行に必要な環境変数については,マニュアル「COBOL2002　ユーザーズガイド」を参照してください。
3. cblcv2k コマンドを指定して，実行する。
4. 実行結果をコマンドプロンプトに出力されるメッセージから確認する。カバレージ情報の蓄積結果は，カバレージ表示機能を使用して確認する。

カバレージ表示機能については，「13.2　カバレージ情報の表示と操作」を参照してください。

(2)　cblcv2k コマンドの指定形式

形式

```
cblcv2k　〔-?〕

    〔-Library　DLLファイル名　〔,…〕　　〕
     -Execute 実行可能ファイル名　〔　プログラムへ渡すパラメタ　〕
```

-Library

　カバレージ情報の蓄積をする DLL ファイル名(.dll)を指定します。

-Execute

　カバレージ情報の蓄積をする実行可能ファイル名(.exe)を指定します。

プログラムへ渡すパラメタ

　カバレージ情報の蓄積をするプログラムへ渡す引数を指定します。

-?

cblcv2k コマンドの構文を表示します。-?を指定すると，ほかの引数はすべて無視されます。

注意事項

- -Library オプションは，複数のファイル名を指定できます。
- -Execute オプションは，必ず，最後に指定してください。-Execute オプション以外のオプションの指定は任意です。
- 実行可能ファイル名のあとに指定されたものすべてをカバレージ情報の蓄積対象プログラムへ渡すパラメタとみなします。
- 起動後のカレントフォルダは，cblcv2k コマンドを実行したフォルダです。
- cblcv2k コマンドのメッセージは，標準エラー出力へ出力します。
- コマンド名，引数は，英大文字，英小文字のどちらでも指定できます。引数の始まりは，ハイフン(-）の代わりにスラント（/）も使用できます。
- コマンド名だけを指定した場合は，コマンドの構文を表示します。
- 引数の区切り記号は空白文字とタブです。空白文字とタブを区切り記号としたくないときは，引数をダブルコーテーション（"）で囲みます。例えば，空白文字を含むファイル名はファイル名をダブルコーテーション（"）で囲んで指定します。
- 「@コマンドファイル名」の形でコマンドファイルが指定できます。コマンドファイルの内容は，実行時にコマンドラインに変換されます。コマンドファイルはコマンド名よりあとに指定します。
- 同じ引数を複数指定した場合は，あとに指定した引数のパラメタを有効とします。
- 引数にパスの付かないファイル名だけを指定した場合は，カレントフォルダのファイルとします。フォルダ名だけを指定した場合は，カレントフォルダ下にあるフォルダとします。相対パスで指定された場合は，カレントフォルダを基準に検索します。ただし，プログラム情報ファイルを指定した場合は，環境変数 CBLPIDIR の指定が有効となります。
- コマンドが返すリターン値は，次のとおりです

リターン値	内容
0	正常終了
1	エラー発生による終了

- -？引数が指定された場合のコマンドの構文は，標準出力に出力されます。それ以外のメッセージは，標準エラー出力に出力されます。

(3)　環境変数の指定

カバレージ情報の蓄積に必要な環境変数を指定します。システム，またはバッチモードを実行するコマンドプロンプトに設定します。

(a)　CBLPIDIR

作成または更新したプログラム情報ファイルのあるフォルダを指定します。

カバレージ情報の蓄積対象であるプログラムの実行に関するその他の環境変数については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

注意事項

- プログラム情報ファイルを次の順序で検索します。検索した結果，プログラム情報ファイルが見つからないときは，プログラム情報ファイルに該当するプログラムはカバレージ情報の蓄積の対象となりません。
  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. 実行可能ファイルに含まれるプログラムは，実行可能ファイルのあるフォルダ
     DLL ファイルに含まれるプログラムは，DLL ファイルのあるフォルダ
  3. カレントフォルダ
- プログラムが-CVInf コンパイラオプションを指定してコンパイルされていない場合は，カバレージ情報の蓄積の対象となりません。
- カバレージ情報の蓄積の対象となるプログラム情報ファイルが一つもない場合は，カバレージ情報の蓄積は行いません（カバレージ情報の蓄積対象プログラムの実行も行いません）。

15.2.2　プログラムからの連動実行による方法

カバレージ情報の蓄積対象プログラムを実行することで，自動的にカバレージ情報を蓄積します。

(1)　カバレージ情報の蓄積の手順

連動実行でカバレージ情報を蓄積するときの作業の流れを説明します。

1. **-CVInf コンパイラオプションを指定して，プログラムをコンパイルする。**
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。

2. **連動実行のための環境変数を指定する。**
   環境変数 CBLTDEXEC を指定します。詳細については，「15.2.2　プログラムからの連動実行による方法」の「(2)　環境変数の指定」を参照してください。

3. **カバレージ情報の蓄積，またはカバレージ情報の蓄積対象プログラムの実行に必要な環境変数を設定する。**
   プログラムの実行に必要な環境変数とカバレージに必要な環境変数を設定します。カバレージに必要な環境変数については，「15.2.1　コマンドによる方法」の「(3)　環境変数の指定」を参照してください。プログラムの実行に必要な環境変数については,マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

4. カバレージ情報の蓄積対象の実行可能ファイルを実行する。

   カバレージ情報が蓄積されます。

5. 実行結果を実行結果出力ファイルによって確認する。カバレージ情報の蓄積結果は，カバレージ表示機能を使用して確認する。

   カバレージ表示機能については，「13.2　カバレージ情報の表示と操作」および「(3)　実行結果出力ファイルにメッセージを出力できないときの処理」を参照してください。

(2)　環境変数の指定

プログラムの開始と同時にカバレージ情報蓄積を連動実行するには，環境変数 CBLTDEXEC に「CV」を指定します。

形式

```
CBLTDEXEC=CV
〔-OutFile 実行結果出力ファイル名〕〔-Add〕〔-Library DLLファイル名〔,…〕〕
```

-OutFile

カバレージ情報の蓄積の実行結果およびトラブルシュート情報※を出力するファイル名を指定します。-OutFile オプション省略時は，実行可能ファイル名の拡張子を「.cvo」に変更したファイル名を，カレントフォルダに出力します。

注※

当社保守員が調査するときに使用する情報です。DLL ファイルのカバレージ情報が蓄積されないなど，連動実行が動作しない場合には，当社保守員に連絡してください。

-Add

結果出力ファイルに追加書きで結果を出力します。指定しなかった場合，同名の実行結果出力ファイルがあったときは，上書きされます。

-Library

カバレージ情報蓄積対象とする DLL ファイル名を指定します。

注意事項

- 環境変数名，引数は，英大文字，英小文字のどちらで指定しても同じ扱いとなります。
- -OutFile でユーザが指定したファイル名は拡張子が「.cvo」以外でも有効とし，ユーザが指定したとおりのファイル名で出力します。
- カレントフォルダは，プログラムが実行されたフォルダです。
- プログラム情報ファイルを次の順序で検索します。検索した結果，プログラム情報ファイルが見つからないときは，プログラム情報ファイルに該当するプログラムはカバレージ情報の蓄積の対象となりません。

  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. 実行可能ファイルに含まれるプログラムは，実行可能ファイルのあるフォルダ

DLL ファイルに含まれるプログラムは，DLL ファイルのあるフォルダ

3. カレントフォルダ

- プログラムが-CVInf コンパイラオプションを指定してコンパイルされていない場合は，カバレージ情報の蓄積の対象となりません。
- カバレージ情報の蓄積の対象となるプログラム情報ファイルが一つもない場合は，カバレージ情報の蓄積は行いません（カバレージ情報の蓄積対象プログラムの実行も行いません）。

(3) 実行結果出力ファイルにメッセージを出力できないときの処理

実行結果出力ファイルにメッセージを出力できない場合，次のようにメッセージを出力します。

(a) イベントログが使用できる Windows

イベントログにメッセージを出力します。何かの理由でイベントログを出力できなかった場合は，「(b) イベントログが使用できない Windows およびイベントログに出力ができないとき」のとおりとなります。

(b) イベントログが使用できない Windows およびイベントログに出力ができないとき

temp フォルダに結果ファイルを出力します。

また，この結果ファイルにも出力できないときは，標準エラー出力に出力します。ただし，-Lib,GUI コンパイラオプションを指定してコンパイルしたプログラムの場合は，メッセージは表示されません。

temp フォルダとは

環境変数 TMP，TEMP，または USERPROFILE に設定されているフォルダです。環境変数が設定されていない場合は，次のフォルダとなります。

- Windows のフォルダ（C:¥Windows など）

結果ファイル名

カバレージ情報蓄積時は，CVO*****.tmp というファイル名となります。カウント情報取得時は，CNO*****.tmp というファイル名となります。*****には任意の文字が入ります。

15.3　カバレージ情報の表示（テキスト形式）

コマンドプロンプトから，コマンドを入力してカバレージ情報を表示します。

カバレージ情報の表示については，「13.2.5　カバレージ情報の表示（テキスト形式）」を参照してください。

15.3.1　カバレージ情報の表示の方法（テキスト形式）

コマンドプロンプトから，コマンドを入力してカバレージ情報を表示します。

カバレージ情報の表示については，「13.2.5　カバレージ情報の表示（テキスト形式）」を参照してください。

(1)　カバレージ情報の表示の手順

バッチモードでカバレージ情報を表示するときの作業の流れを説明します。

1. -CVInf コンパイラオプションを指定して，プログラムをコンパイルする。
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。
2. cblcl2k コマンドを指定して，実行する。
3. コマンドに指定したカバレージ情報リストファイルをテキストエディタで開き，内容を確認する。

(2)　cblcl2k コマンドの指定形式

形式

```
cblcl2k 〔-?〕
  〔-List〕〔 -Sum mary 〕〔 -S ou rc e〕〔 -Diff erence 〕
  〔 -Unex ecuted 〕〔 -Unex ecuted Diff erence 〕
  〔 -Act ivate 〕 〔-All〕
  〔 -Out Put カバレージ情報リストファイル名 〕
    -In put プログラム情報ファイル名 〔,… 〕
```

-List
　すべてのプログラムのカバレージ情報一覧を表示します。

-Summary
　カバレージ情報のまとめを表示します。

-Source
　すべてのソース情報を表示します。

-Difference
　差分ソース情報を表示します。

-Unexecuted

未実行ソース情報を表示します。

-UnexecutedDifference

差分未実行ソース情報を表示します。

-Activate

呼び出し文情報を表示します。

-All

すべての情報を表示します。次の引数をすべて指定したのと同じです。

-List，-Summary，-Source，-Difference，-Unexecuted，-UnexecutedDifference，-Activate

-All が指定されたときは，ほかの引数は無視されます。

-Output

カバレージ情報リストファイルのパス名を指定します。省略時はカレントフォルダに，最初に指定したプログラム情報ファイル名に拡張子「.cll」を付けたファイル名で出力します。

-Input

プログラム情報ファイルのパス名を指定します。

-?

cblcl2k コマンドの構文を表示します。-?を指定すると，ほかの引数はすべて無視されます。

出力されるカバレージ情報リストの詳細については，「13.2.5　カバレージ情報の表示（テキスト形式)」を参照してください。

注意事項

- -List，-Summary，-Source，-Difference，-Unexecuted，-UnexecutedDifference，-Activate または-All オプションを何も指定しなかった場合，-All オプションを仮定します。
- cblcl2k コマンドのメッセージは標準エラー出力に出力します。
- コマンド名，引数は，英大文字，英小文字のどちらでも指定できます。引数の始まりは，ハイフン(-）の代わりにスラント（/）も使用できます。
- コマンド名だけを指定した場合は，コマンドの構文を表示します。
- 引数の区切り記号は空白文字とタブです。空白文字とタブを区切り記号としたくないときは，引数をダブルコーテーション（"）で囲みます。例えば，空白文字を含むファイル名はファイル名をダブルコーテーション（"）で囲んで指定します。
- 「@コマンドファイル名」の形でコマンドファイルが指定できます。コマンドファイルの内容は，実行時にコマンドラインに変換されます。コマンドファイルはコマンド名よりあとに指定します。
- 同じ引数を複数指定した場合は，あとに指定した引数のパラメタを有効とします。
- 引数にパスの付かないファイル名だけを指定した場合は，カレントフォルダのファイルとします。フォルダ名だけを指定した場合は，カレントフォルダ下にあるフォルダとします。相対パスで指定された場合は，カレントフォルダを基準に検索します。ただし，プログラム情報ファイルを指定した場合は，環境変数 CBLPIDIR の指定が有効となります。

- コマンドが返すリターン値は，次のとおりです

リターン値	内容
0	正常終了
1	エラー発生による終了

- -？引数が指定された場合のコマンドの構文は，標準出力に出力されます。それ以外のメッセージは，標準エラー出力に出力されます。

(3) 環境変数の指定

カバレージ表示に必要なファイルを環境変数に指定します。

(a) CBLPIDIR

作成または更新したプログラム情報ファイルのあるフォルダを指定します。

注意事項

- プログラム情報ファイルがパスの付かないファイル名または，相対パスで指定された時は，次のフォルダを検索します。相対パスを指定した時は，次のフォルダからの相対パスを検索します。絶対パスが指定されたときは，次に示すフォルダの検索は行いません。

  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. カレントフォルダ

- プログラムが-CVInf コンパイラオプションでコンパイルされていない場合，カバレージ情報の表示，およびカウント情報の表示の対象となりません。

  カバレージ情報の表示では対象外とするエラーメッセージを出力します。

- カバレージ情報の表示の対象となるプログラム情報ファイルが一つもない場合は，カバレージの表示を行いません（カバレージ情報リストファイルも出力しません）。

15.4　カバレージ情報の表示（CSV 形式）

コマンドプロンプトから，コマンドを入力してカバレージ情報を表示します。

カバレージ情報の表示については，「13.2.6　カバレージ情報の表示（CSV 形式）」を参照してください。

15.4.1　カバレージ情報の表示の方法（CSV 形式）

コマンドプロンプトから，コマンドを入力してカバレージ情報を表示します。

カバレージ情報の表示については，「13.2.6　カバレージ情報の表示（CSV 形式）」を参照してください。

(1)　カバレージ情報の表示の手順

バッチモードでカバレージ情報を表示するときの作業の流れを説明します。

1. -CVInf コンパイラオプションを指定して，プログラムをコンパイルする。
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。
2. cblcc2k コマンドを指定して，実行する。
3. 出力された CSV ファイルをエディタや Excel などの表計算プログラムで開き，内容を確認する。

(2)　cblcc2k コマンドの指定形式

形式

```
cblcc2k 〔 -? 〕
        〔 -List 〕 〔 -Source 〕 〔 -All 〕 〔 -NoHeader 〕
        〔 -ListSourceFile 〕 〔 -ListCompilationUnit 〕 〔 -ListAll 〕 〔 -Comment 〕
        〔 -ListFileName カバレージ統計情報CSVファイル名 〕
        〔 -OutputPath 出力先フォルダ名 〕 〔 -Recursive 〕
        { -Input プログラム情報ファイル名〔 , … 〕 | -InputPath 入力フォルダ名 }
```

-List
　カバレージ統計情報 CSV ファイルを出力します。

-Source
　ソースカバレージ情報 CSV ファイルを出力します。

-All
　カバレージ統計情報 CSV ファイルとソースカバレージ情報 CSV ファイルの両方を出力します。
　-List と-Source を同時に指定したのと同じです。

-NoHeader
　ヘッダカラム名とヘッダを出力しません。

この引数を指定した場合，カバレージ統計情報 CSV ファイルとソースカバレージ情報 CSV ファイルには，それぞれ次の内容を出力します。

- カバレージ統計情報 CSV ファイルのとき

  カバレージ統計情報カラム名とカバレージ統計情報

- ソースカバレージ情報 CSV ファイルのとき

  ソースカバレージ情報カラム名とソースカバレージ情報

カバレージ統計情報 CSV ファイルおよびソースカバレージ情報 CSV ファイルの出力内容については，「13.2.6　カバレージ情報の表示（CSV 形式）」を参照してください。

-ListSourceFile

カバレージ統計情報 CSV ファイルにソースファイル単位の統計情報を出力します。カバレージ統計情報 CSV ファイルを出力しない場合は無視されます。

-ListCompilationUnit

カバレージ統計情報 CSV ファイルに翻訳単位の統計情報を出力します。カバレージ統計情報 CSV ファイルを出力しない場合は無視されます。

-ListAll

カバレージ統計情報 CSV ファイルに，ソースファイル単位と翻訳単位の両方の統計情報を出力します。

-ListSourceFile と-ListCompilationUnit を同時に指定したのと同じです。カバレージ統計情報 CSV ファイルを出力しない場合は無視されます。

-Comment

ソースに記述されたコメント行（注記行）をソースカバレージ情報 CSV ファイルに出力します。ソースカバレージ情報 CSV ファイルを出力しない場合は無視されます。

省略時は，ソースに記述されたコメント行（注記行）をソースカバレージ情報 CSV ファイルに出力しません。

-ListFileName

出力するカバレージ統計情報 CSV ファイルのファイル名（.csv）を指定します。ファイル名には，カバレージ統計情報 CSV ファイルを格納するフォルダのパスを含めないで，ファイル名だけを指定してください。

省略時は，「summary_list.csv」を仮定します。カバレージ統計情報 CSV ファイルを出力しない場合は無視されます。

-OutputPath

生成する CSV ファイルの出力先フォルダのパスを指定します。出力先フォルダ名には，存在するフォルダを指定してください。省略時は，出力先フォルダをカレントフォルダとします。

- カバレージ統計情報 CSV ファイルの場合

  -OutputPath オプションで指定したフォルダに，-ListFileName オプションで指定したファイル名で出力します。

- ソースカバレージ情報 CSV ファイルの場合

-OutputPath オプションで指定したフォルダに，プログラム情報ファイル名（拡張子を除く）に「_source.csv」を付けたファイル名で出力します。例えば，プログラム情報ファイル名が「makeline.cbp」の場合は，ソースカバレージ情報 CSV ファイルは「makeline_source.csv」で出力します。

-Recursive

-InputPath で指定したフォルダと，そのサブフォルダにあるすべてのプログラム情報ファイルを出力対象とします。また，-OutputPath に指定した出力先フォルダに，-InputPath で指定したフォルダ下と同じフォルダ構成で出力します。

省略時は，-InputPath で指定したフォルダの直下にあるプログラム情報ファイルだけを出力対象とし，-OutputPath に指定した出力先フォルダの直下に出力します。

この引数は，-InputPath を指定した場合にだけ有効になります。

-Input

出力対象のプログラム情報ファイル名を指定します。

プログラム情報ファイル名は，コンマ（,）区切りで複数指定できます。ファイル名が同じプログラム情報ファイル名は同時に指定できません。絶対パスや相対パスで指定した場合も，同じパスではなく，同じプログラム情報ファイル名は同時に指定できません。

-InputPath

出力対象のプログラム情報ファイルを格納しているフォルダのパスを指定します。

-?

cblcc2k コマンドの構文を表示します。-?を指定すると，ほかの引数はすべて無視されます。

出力されるカバレージ情報リストの詳細については，「13.2.6　カバレージ情報の表示（CSV 形式）」を参照してください。

注意事項

- -List，-Source または-All のうち何も指定がなかった場合，-All オプションを仮定します。
- -ListSourceFile，-ListCompilationUnit または-ListAll のうち何も指定がなかった場合，-ListAll オプションを仮定します。
- cblcc2k コマンドのメッセージは標準エラー出力に出力します。
- 複数のファイルを出力する場合に，処理中にエラーが発生したときは，それ以降のファイルの出力を中止します。それより前に出力したファイルは削除されません。
- コマンド名，引数は，英大文字，英小文字のどちらでも指定できます。引数の始まりは，ハイフン（-）の代わりにスラント（/）も使用できます。
- コマンド名だけを指定した場合は，コマンドの構文を表示します。
- 引数の区切り記号は空白文字とタブです。空白文字とタブを区切り記号としたくないときは，引数をダブルコーテーション（"）で囲みます。例えば，空白文字を含むファイル名はファイル名をダブルコーテーション（"）で囲んで指定します。
- 「@コマンドファイル名」の形でコマンドファイルが指定できます。コマンドファイルの内容は，実行時にコマンドラインに変換されます。コマンドファイルはコマンド名よりあとに指定します。

- 同じ引数を複数指定した場合は，あとに指定した引数のパラメタを有効とします。
- 引数にパスの付かないファイル名だけを指定した場合は，カレントフォルダのファイルとします。フォルダ名だけを指定した場合は，カレントフォルダ下にあるフォルダとします。相対パスで指定された場合は，カレントフォルダを基準に検索します。ただし，プログラム情報ファイルを指定した場合は，環境変数 CBLPIDIR の指定が有効となります。
- コマンドが返すリターン値は，次のとおりです

リターン値	内容
0	正常終了
1	エラー発生による終了

- -？引数が指定された場合のコマンドの構文は，標準出力に出力されます。それ以外のメッセージは，標準エラー出力に出力されます。

(3) 環境変数の指定

カバレージ表示に必要なファイルを環境変数に指定します。

(a) CBLPIDIR

作成または更新したプログラム情報ファイルのあるフォルダを指定します。

注意事項

- プログラム情報ファイルがパスの付かないファイル名または，相対パスで指定された時は，次のフォルダを検索します。相対パスを指定した時は，次のフォルダからの相対パスを検索します。絶対パスが指定されたときは，次に示すフォルダの検索は行いません。

  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. カレントフォルダ

- プログラムが-CVInf コンパイラオプションでコンパイルされていない場合，カバレージ情報の表示，およびカウント情報の表示の対象となりません。

  カバレージ情報の表示では対象外とするエラーメッセージを出力します。

- カバレージ情報の表示の対象となるプログラム情報ファイルが一つもない場合は，カバレージの表示を行いません（カバレージ統計情報 CSV ファイルも出力しません）。

15.5　カバレージ情報の操作

蓄積したカバレージ情報に対して次の操作ができます。

- カバレージ全情報を 0%にする
- カバレージ差分情報を 0%にする
- ソース差分情報をクリアする
- カバレージ情報をマージする

15.5.1　コマンドによる方法

コマンドプロンプトからコマンドを入力して，カバレージ情報を操作できます。

カバレージ情報の操作については，「13.2.7　カバレージ情報の操作」を参照してください。

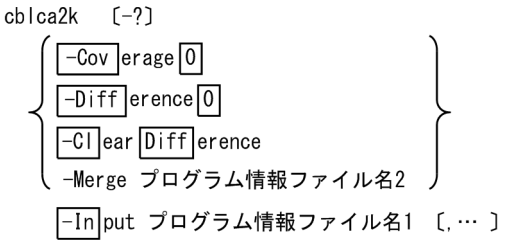
(1)　カバレージ情報の操作の方法

バッチモードでカバレージ情報を操作するときの作業の流れを説明します。

1. cblca2k コマンドに操作したい機能を指定して，実行する。

(2)　cblca2k コマンドの指定形式

形式



-Coverage0

　カバレージ全情報を 0%にします。

-Difference0

　カバレージ差分情報を 0%にします。

-ClearDifference

　ソース差分情報をクリアします。

-Merge

プログラム情報ファイル名 1 のカバレージ情報をプログラム情報ファイル名 2 へマージします。プログラム情報ファイル名 1 は，パスを除いたファイル名がプログラム情報ファイル 1 と同じでなければなりません。

-Input

プログラム情報ファイルを指定します。

-?

cblca2k コマンドの構文を表示します。-?を指定すると，ほかの引数はすべて無視されます。

注意事項

- コマンド名，引数は英大文字，英小文字のどちらで指定しても同じ扱いとなります。
- -Merge オプションを指定した場合，プログラム情報ファイル名 1 のパスを除いたファイル名はすべて同じでなければなりません。
- -Coverage0，-Difference0，-ClearDifference，-Merge オプションを複数指定すると，最後に指定されたオプションを有効とします。
- cblca2k コマンドのメッセージは標準エラー出力に出力します。
- コマンド名，引数は，英大文字，英小文字のどちらでも指定できます。引数の始まりは，ハイフン（-）の代わりにスラント（/）も使用できます。
- コマンド名だけを指定した場合は，コマンドの構文を表示します。
- 引数の区切り記号は空白文字とタブです。空白文字とタブを区切り記号としたくないときは，引数をダブルコーテーション（"）で囲みます。例えば，空白文字を含むファイル名はファイル名をダブルコーテーション（"）で囲んで指定します。
- 「@コマンドファイル名」の形でコマンドファイルが指定できます。コマンドファイルの内容は，実行時にコマンドラインに変換されます。コマンドファイルはコマンド名よりあとに指定します。
- 同じ引数を複数指定した場合は，あとに指定した引数のパラメタを有効とします。
- 引数にパスの付かないファイル名だけを指定した場合は，カレントフォルダのファイルとします。フォルダ名だけを指定した場合は，カレントフォルダ下にあるフォルダとします。相対パスで指定された場合は，カレントフォルダを基準に検索します。ただし，プログラム情報ファイルを指定した場合は，環境変数 CBLPIDIR の指定が有効となります。
- コマンドが返すリターン値は，次のとおりです

リターン値	内容
0	正常終了
1	エラー発生による終了

- -？引数が指定された場合のコマンドの構文は，標準出力に出力されます。それ以外のメッセージは，標準エラー出力に出力されます。

(3) 環境変数の指定

(a) CBLPIDIR

プログラム情報ファイルのあるフォルダを指定します。

注意事項

- プログラム情報ファイルがパスの付かないファイル名または，相対パスで指定されたときは，次のフォルダを検索します。相対パスを指定したときは，次に示すフォルダからの相対パスを検索します。絶対パスが指定されたときは，次に示すフォルダの検索は行いません。

  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. カレントフォルダ

- プログラムが-CVInf コンパイラオプションを指定してコンパイルされていない場合，カバレージ情報の操作，およびカウント情報の表示の対象となりません。

  カバレージ情報の操作では対象外とするエラーメッセージを出力します。

- カバレージ情報の操作の対象となるプログラム情報ファイルが一つもない場合は，カバレージの操作を行いません。

15.6　カウント情報の表示

カウント情報は，テスト実行させたプログラムの文の実行回数を表示します。COBOL プログラムの実行性能改善の施策を検討するときに，各文の実行回数を測定し，COBOL プログラムの性能を分析するために必要な情報を取得し，表示できます。

15.6.1　コマンドによる方法

コマンドプロンプトからコマンドを入力して，カウント情報を表示できます。

(1)　カウント情報の表示の手順

バッチモードでカウント情報を取得するときの作業の流れを説明します。

1. -CVInf コンパイラオプションを指定して，プログラムをコンパイルする。
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。

2. カウント情報の取得またはカウント情報の取得対象プログラムの実行に必要な環境変数を設定する。
   プログラムの実行に必要な環境変数とカウント情報の取得に必要な環境変数を設定します。カウント情報の取得に必要な環境変数については，「15.6.1　コマンドによる方法」の「(3)　環境変数の指定」を参照してください。プログラムの実行に必要な環境変数については,マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

3. cblcn2k コマンドを指定して，実行する。

4. 実行結果を画面に出力されるメッセージから確認する。カウント情報の取得結果は，カウント情報リストファイルをテキストエディタで開いて確認する。

(2)　cblcn2k コマンドの指定形式

コマンドプロンプトから，コマンドを入力してカウント情報を取得します。

COBOL プログラムを実行してカウント情報を取得し，結果をファイルに出力します。

形式

```
cblcn2k 〔-?〕
 〔 -Library DLLファイル名 〔,… 〕〕〔 -Output カウント情報リストファイル名 〕
   -Execute 実行可能ファイル名 〔 プログラムへ渡すパラメタ 〕
```

-Library
　カウント情報の表示対象とする DLL ファイル名を指定します。

-Output

カウント情報リストファイル名を指定します。省略時は，カレントフォルダに，実行可能ファイル名に拡張子「.cnl」を付けて出力します。

-Execute

実行可能ファイル名を指定します。

プログラムへ渡すパラメタ

カウント情報の取得をするプログラムへ渡す引数を指定します。

-?

cblcn2k コマンドの構文を表示します。-?を指定すると，ほかの引数はすべて無視されます。

注意事項

- 実行可能ファイル名は必須です。
- 実行可能ファイルのパス名のあとに指定されたものすべてをカウント情報の取得対象のプログラムへ渡すパラメタとみなします。
- -Library オプションは，複数のファイル名を指定できます。
- -Execute オプションは，必ず，最後に指定してください。-Execute オプション以外のオプションの指定は任意です。
- 起動後のカレントフォルダは，cblcn2k コマンドを実行したフォルダです。
- コマンドライン上には，コマンド名以降であればどこにでもコマンドファイルを指定できます。コマンドファイル名の先頭には@を付けて指定します。
- 引数の区切り記号は空白文字とタブです。空白文字とタブを区切り記号としたくないときは，引数をダブルコーテーション（"）で囲みます。例えば，空白文字を含むファイル名はファイル名をダブルコーテーション（"）で囲んで指定します。
- コマンド名，引数は英大文字，英小文字のどちらで指定しても同じ扱いとなります。
- 同じ引数を複数指定した場合，最後に指定した内容が有効となります。ただし-Execute オプションは繰り返して指定することはできません。
- 引数を何も指定しない場合，コマンドの構文を表示します。
- パラメタに複数のファイル名を指定するとき，ファイル名を区切るコンマ（,）の前後に空白文字を指定してはいけません。
- cblcn2k コマンドのメッセージは標準エラー出力に出力します。
- コマンドが返すリターン値は，次のとおりです。

リターン値	内容
0	正常終了
1	エラー発生による終了

(3)　環境変数の指定

(a)　CBLPIDIR

作成または更新したプログラム情報ファイルのあるフォルダを指定します。

プログラムの実行に関するその他の環境変数については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

注意事項

- プログラム情報ファイルを次に示す順序で検索します。検索した結果，見つからないときは，プログラム情報ファイルに該当するプログラムはカウント情報表示の対象となりません。
  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. 実行可能ファイルに含まれるプログラムは，実行可能ファイルのあるフォルダ
     DLL ファイルに含まれるプログラムは，DLL ファイルのあるフォルダ
  3. カレントフォルダ
- プログラム情報ファイルがパスの付かないファイル名または，相対パスで指定されたときは，以下のフォルダを検索する。相対パスを指定したときは，以下のフォルダからの相対パスを検索します。絶対パスが指定されたときは，以下の検索はしません。
  1. 環境変数 CBLPIDIR で指定したフォルダ
  2. カレントフォルダ
- プログラムが-CVInf コンパイラオプションを指定してコンパイルされていない場合，カウント情報の表示の対象となりません。
- 対象となるプログラム情報ファイルが一つもない場合は，カウント情報の表示のためのプログラムは実行しません（カウント情報リストファイルも出力しません）。

15.6.2　プログラムからの連動実行による方法

カウント情報の取得対象プログラムを実行することで，自動的にカウント情報を取得できます。

(1)　カウント情報の表示の手順

プログラムからの連動実行で，プログラムのカウント情報を表示するときの作業の流れを説明します。

1. **-CVInf コンパイラオプションを指定して，プログラムをコンパイルする。**
   コンパイラオプションについては，「9.1.2　プログラムのコンパイル」を参照してください。

2. **連動実行の環境変数を指定する。**
   環境変数 CBLTDEXEC を指定します。詳細については，「15.6.2　プログラムからの連動実行による方法」の「(2)　環境変数の指定」を参照してください。

3. カウント情報の取得またはプログラムの実行に必要な環境変数を設定する。

   プログラムの実行に必要な環境変数とカウント情報の取得に必要な環境変数を設定します。カウント情報の取得に必要な環境変数については，「15.6.1　コマンドによる方法」の「(3)　環境変数の指定」を参照してください。プログラムの実行に必要な環境変数については,マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

4. カウント情報の取得対象の実行可能ファイルを実行する。

   カウント情報が取得されます。

5. 実行結果を実行結果出力ファイルで確認する。取得したカウント情報はカウント情報リストファイルをテキストエディタで開いて確認する。

(2)　環境変数の指定

プログラムの開始と同時にカウント情報の取得を連動実行するには，環境変数 CBLTDEXEC に「CN」を指定します。

形式

```
CBLTDEXEC=CN
        〔-Output カウント情報リストファイル名〕
        〔-OutFile 実行結果出力ファイル名　　〕
        〔-Add〕〔-Library　DLLファイル名 〔,…〕　〕
```

-Output

カウント情報リストファイル名を指定します。省略時は，カレントフォルダに，実行可能ファイル名に拡張子「.cnl」を付けて出力します。

-OutFile

カウントの実行結果およびトラブルシュート情報※を出力するファイル名を指定します。指定しなかった場合は，実行可能ファイル名の拡張子を「.cno」に変更したファイル名を，カレントフォルダに出力します。

注※

当社保守員が調査するときに使用する情報です。DLL ファイルのカウント情報が表示されないなど，連動実行が動作しない場合には，当社保守員に連絡してください。

-Add

実行結果出力ファイルに追加書きで結果を出力します。指定しなかった場合，同名の実行結果出力ファイルがあった場合は，上書きされます。

-Library

カウント情報取得の対象とする DLL ファイル名を指定します。

注意事項

- 環境変数名，引数は英大文字，英小文字のどちらで指定しても同じ扱いとなります。

- -OutFile でユーザが指定したファイル名は拡張子が「.cno」以外でも有効とし，ユーザが指定したとおりのファイル名で出力します。
- カレントフォルダはプログラムが実行されたフォルダです。
- 各ファイル名を相対パスで記述した場合，検索開始位置は起動コマンドが実行されたフォルダです。
- パラメタに複数のファイル名を指定するとき，ファイル名を区切るコンマ（,）の前後に空白文字を指定してはいけません。

(3)　実行結果出力ファイルにメッセージを出力できないときの処理

実行結果出力ファイルにメッセージを出力できない場合のメッセージの出力方法については，「15.2.2　プログラムからの連動実行による方法」の「(3)　実行結果出力ファイルにメッセージを出力できないときの処理」を参照してください。

16 TD コマンド生成機能

TD コマンド生成機能とは，テストデバッグで使用する TD コマンドをコンパイル時にファイルに生成する機能です。この章では，この機能によって生成される TD コマンドの内容について説明します。

16.1　概要

16.1.1　機能の概要

TD コマンド生成機能とは，COBOL ソースファイルを解析して，テストデバッガで使用する TD コマンドをファイルに生成する機能です。

生成される TD コマンドには，中断点情報，プログラムのシミュレーション情報，およびファイルのシミュレーション情報の 3 種類があります。

これらの TD コマンド群は，-TestCmd オプションを指定してコンパイルしたときに生成されます。生成された TD コマンド群に，必要に応じて TD コマンドの修正や追加をすることで，TD コマンド格納ファイルを作成できます。

また，TD コマンド生成機能の対象となるプログラムは，コンパイルしたときすべてのプログラムで，S レベル，U レベルのコンパイルエラーのなかった原始プログラムファイルだけです。

16.1.2　TD コマンド格納ファイルの構成と出力先

(1)　TD コマンド格納ファイルの構成

TD コマンド格納ファイルは，COBOL ソースファイル単位に生成されます。指定するコンパイラオプションと，生成される TD コマンド格納ファイル（.tdi または.tds）との関係を次に示します。

表 16–1　コンパイラオプションと TD コマンド格納ファイルとの関係

指定するコンパイラオプション	生成されるファイルの名称とその内容
• -TestCmd,Full • -TestCmd,Full　-TestCmd,Break • -TestCmd,Full　-TestCmd,Sim • -TestCmd,Full　-TestCmd,Break　-TestCmd,Sim	a.tdi（中断点情報，シミュレーション情報）
-TestCmd,Break	a.tdi（中断点情報）
-TestCmd,Sim	a.tds（シミュレーション情報）
-TestCmd,Break　-TestCmd,Sim	a.tdi（中断点情報） a.tds（シミュレーション情報）

注 1

-TestCmd,Full オプションと同時に指定された-TestCmd,Break，および-TestCmd,Sim オプションは無効となります。

注 2

生成されるファイルの名称は，COBOL ソースファイル名を a.cbl としたときの例で示しています。

注 3

-TestCmd,Break，および-TestCmd,Sim オプションを同時に指定したときに出力される.tdi ファイルには，シミュレーション情報を一括入力するための #INCLUDE コマンドが追加になります。

注 4

TD コマンドは COBOL 原始プログラム単位に作成します。

(2) TD コマンド格納ファイルの出力先

TD コマンド格納ファイルの出力先は，コンパイラ環境変数 CBLPIDIR で指定します。環境変数の指定がない場合は，カレントフォルダに出力されます。

16.2　中断点情報とシミュレーション情報

TD コマンド生成機能で生成されるそれぞれの TD コマンド群について説明します。

16.2.1　中断点情報

(1)　注釈行（中断点情報の開始）

形式

```
*>*************************************************************************
*>**    SET BREAK TD COMMAND START                                       ***
*>*************************************************************************
*>*************************************************************************
*>**    TEST PROGRAM ==> プログラム名                                    ***
*>*************************************************************************
```

機能

中断点情報の TD コマンドの設定開始を表示します。

生成条件

原始プログラムごとに必ず生成されます。

(2)　SET QUALIFICATION コマンド

形式

```
SET QUALIFICATION(
      { #PROGRAM( '最外側のプログラム名'/'最外側のプログラム名' ) }
      { #CLASS(クラス名)                                          } )
      { #FUNCTION(関数名)                                         }
```

機能

この TD コマンド以降に指定する TD コマンドを最外側のプログラム名，クラス名または関数名で修飾します。内側のプログラム，メソッドについては生成されません。このため，以降の TD コマンドで内部プログラムやメソッドのデータ名や節名を参照するときは，内側のプログラム名，メソッド名による修飾が必要となります。

生成条件

- tdi ファイル上では，最外側のプログラムの SET BREAK コマンドの前ごとに生成されます。
- tds ファイル上では，複数の最外側のプログラムがあるときだけシミュレーション TD コマンドの前ごとに生成されます。

(3) #INCLUDE コマンド

形式

```
#INCLUDE INFILE('ファイル名') NOMESSAGE
```

機能

ファイル名で示すシミュレーション情報の TD コマンド格納ファイルを入力します。

生成条件

SET BREAK コマンドのあとに生成されます。環境変数 CBLPIDIR が指定されている場合，環境変数で指定したフォルダ名がファイル名に付けられます。

(4) SET BREAK コマンド（節名）

形式

```
SET BREAK PARAGRAPH( [
   { #PROGRAM( '最外側のプログラム名'/'内側のプログラム名' )
   { #CLASS(クラス名/'メソッド名')                          } ] 節名 ) DO
   { #FUNCTION(関数名)
GO
ENDDO
```

機能

プログラムが実行中に一時停止する中断点を指定します。GO コマンドがあるので，実行のトレース情報を知るのに役立ちます。

生成条件

プログラム内で一意となる節名があるときに生成されます。

入れ子のプログラムのときは内側のプログラム名で修飾します。

(5) SET BREAK コマンド（文）

形式

```
SET BREAK STATEMENT(xxxxxxx [.n] )  DO
  GO
ENDDO
```

機能

プログラムが実行中に一時停止する中断点を指定します。GO コマンドによって，実行のトレース情報を参照できます。

生成条件

条件文（IF 文，EVALUATE 文，PERFORM 文，SEARCH 文，および条件指定のある文）または出口文（GOBACK 文，EXIT PROGRAM 文，EXIT METHOD 文，EXIT FUNCTION 文，および STOP RUN 文）のある行に対して生成されます。

文番号（xxxxxxx）は常に 7 けたとし，7 けたに満たない場合は前に 0 を入れます。

条件文や出口文が 1 行に 2 文以上ある場合，2 文目からは行番号に位置番号（.2，.3，……）が付きます。

(6) 注釈行（中断点情報の終了）

形式

```
*>**    SET BREAK TD COMMAND ENDED                          ***
```

機能

中断点情報の TD コマンドの設定終了を表示します。

生成条件

中断点情報 TD コマンド格納ファイルには，最外側のプログラムごとに必ず生成されます。

(7) SET BREAK コマンド（プログラム名）

形式

```
SET BREAK PROCEDURE( 〔
    {#PROGRAM( '最外側のプログラム名'/'最外側のプログラム名' )
    {#CLASS(クラス名/'メソッド名')                              〕#ENTRY ) DO
    {#FUNCTION(関数名)
  〔 ASSIGN DATA(データ名) VALUE(データ値) 〕
  〔 GO 〔COVERAGE 〕〕
ENDDO
```

機能

最外側のプログラム，メソッドまたは関数の実行を開始する入口点を指定します。また，USING で指定されたデータに対して ASSIGN DATA コマンドでデータ値を設定します。設定するデータ値については，「16.2.3　データ設定値」を参照してください。

生成条件

最外側のプログラム，メソッドまたは関数の PROCEDURE DIVISION に対して生成されます。

"ASSIGN DATA(データ名) VALUE(データ値)"は，USING 指定があるときに生成されます。なお，ASSIGN DATA コマンドは，USING に指定された左側の引数から生成されます。

"GO"は，最後のプログラムにだけ生成されます。また，"COVERAGE"は，-CVInf オプションを指定してコンパイルしたときだけ生成されます。

16.2.2　シミュレーション情報

(1)　注釈行（シミュレーション情報の開始）

形式

```
*>*************************************************************************
*>**    SIMULATE TD COMMAND START                                       ***
*>*************************************************************************
```

機能

シミュレーションの TD コマンドの設定開始を表示します。

生成条件

シミュレーション情報を含む TD コマンド格納ファイルには必ず生成されます。

(2)　注釈行（主プログラム）

形式

```
*>*************************************************************************
*>**    MAIN PROGRAM SIMULATE ==> プログラム名                          ***
*>*************************************************************************
```

機能

主プログラムシミュレーションの設定開始を表示します。

生成条件

-SimMain オプションを指定したプログラムに対して生成されます。

(3)　SIMULATE MAIN コマンド

形式

```
SIMULATE MAIN(#PROGRAM('最外側のプログラム名'))
          [ ASSIGN DATA(データ名) VALUE(データ値)  ]
          [           :                           ]
ENDSIMULATE
```

機能

主プログラムをシミュレーションします。

生成条件

-SimMain オプションを指定したプログラムに対して生成されます。USING 指定があるときは，データ値が設定されます。なお，USING 指定があるとき生成される ASSIGN DATA コマンドは，指定された左側の引数から生成されます。

設定されるデータ値については，「16.2.3　データ設定値」を参照してください。

(4) 注釈行（副プログラム）

形式

```
*>*************************************************************************
*>**    SUB PROGRAM SIMULATE ==> プログラム名                            ***
*>*************************************************************************
```

機能

副プログラムシミュレーションの設定開始を表示します。

生成条件

-SimSub オプションを指定したプログラムを定数指定の CALL 文で呼び出しているときに生成されます。

注意事項

-SimIdent オプションを指定した場合，一意名指定の CALL 文に対する注釈行（副プログラム）は，生成されません。

(5) SIMULATE SUB コマンド

形式

```
SIMULATE SUB(#PROGRAM('最外側のプログラム名')
    〔 USING(記号名1  〔,… 〕 ) 〕
    〔 RETURNING(記号名2)〕
    ┌ DEFINE                  ┐
    │    01 記号名1           │
    │          :              │
    │ ENDDEFINE               │
    │ DISPLAY DATA(記号名1)   │
    │          :              │
    └                         ┘
    〔 ASSIGN DATA(記号名2) VALUE(データ値)〕
ENDSIMULATE
```

機能

呼び出されるプログラムをシミュレーションします。

生成条件

-SimSub オプションを指定した最外側のプログラムを定数指定の CALL 文で呼び出しているときに生成されます。

CALL 文の USING 指定があるときは，データ名の内容を表示します。なお，USING 指定があるとき生成される DISPLAY DATA コマンドは，指定された左側の引数から生成されます。また，CALL 文の RETURNING 指定があるときは，データ値を設定します。このとき生成する記号名は，文字列"PARM"に 8 けたの 1 から始まる数字を付加したものとします。また，設定されるデータ値については，「16.2.3　データ設定値」を参照してください。

注意事項

-SimIdent オプションを指定した場合，一意名指定の CALL 文に対する SIMULATE SUB コマンドは，生成されません。-SimIdent オプションを指定して副プログラムシミュレーションをする場合は，一意名指定の CALL 文で呼び出す副プログラム名に対する SIMULATE SUB コマンドを，TD コマンド格納ファイルに定義して使用してください。

(6) 注釈行（ファイル）

形式

```
*>*************************************************************************
*>**    FILE   SIMULATE ==> ファイル名                                   ***
*>*************************************************************************
```

機能

ファイルシミュレーションの設定開始を表示します。

生成条件

ファイルに対して OPEN 文のあるときに生成されます。

(7) SIMULATE FILE コマンド（INPUT／IO モード）

形式

```
SIMULATE FILE( [
  { #PROGRAM( '最外側のプログラム名'/'内側のプログラム名')
    #CLASS(クラス名/#FACTORY)
    #CLASS(クラス名/#OBJECT)
    #FUNCTION(関数名)                                          } ] ファイル名 )
        OPENMODE( { INPUT } ) COUNTER(c1)
                  { IO    }

  IF CONDITION(c1=1)
    [ASSIGN DATA(データ名) VALUE(データ値)  ]
    [          :                            ]
  ELSE
      {GO END     }
      {GO INVALID }
  ENDIF
ENDSIMULATE
```

機能

OPEN INPUT または OPEN I-O モードで開かれた入出力ファイルに対するシミュレーションをします。

1 回目のシミュレーションでデータを設定し，2 回目以降のシミュレーションでファイル終了シミュレーションまたは無効キーシミュレーションをします。

生成条件

FD 項で定義したファイルが OPEN INPUT モードで開かれたときは SIMULATE FILE ～ OPENMODE(INPUT)が生成されます。OPEN I-O モードで開かれたときは SIMULATE FILE ～ OPENMODE(IO)コマンドが生成されます。

修飾対象がファクトリ定義のデータ部で定義されているときは#FACTORY で，インスタンス定義のデータ部で定義されているときは#OBJECT で，関数定義のデータ部で定義されているときは #FUNCTION で修飾されます。

また，順アクセスのファイルには GO END が，乱アクセス，動的アクセスのファイルには GO INVALID が設定されます。

(8) SIMULATE FILE コマンド（OUTPUT／EXTEND モード）

形式

```
SIMULATE FILE( [
  { #PROGRAM( '最外側のプログラム名'/'内側のプログラム名' )
    #CLASS(クラス名/#FACTORY)
    #CLASS(クラス名/#OBJECT)
    #FUNCTION(関数名)                                      } ] ファイル名 )
        OPENMODE( { OUTPUT
                  { EXTEND } )
   DISPLAY DATA(データ名)
ENDSIMULATE
```

機能

OPEN OUTPUT または OPEN EXTEND モードで開かれた出力ファイルに対するシミュレーションをします。

生成条件

FD 項で定義したファイルが OPEN OUTPUT モードで開かれたときは SIMULATE FILE ～ OPENMODE(OUTPUT)コマンドが生成されます。OPEN EXTEND モードで開かれたときは SIMULATE FILE ～ OPENMODE(EXTEND)コマンドが生成されます。

修飾対象がファクトリ定義のデータ部で定義されているときは#FACTORY で，インスタンス定義のデータ部で定義されているときは#OBJECT で，関数定義のデータ部で定義されているときは #FUNCTION で修飾されます。

(9) 注釈行（シミュレーション情報の終了）

形式

```
>***        SIMULATE SUB COMMAND ENDED        ***
```

機能

ファイルシミュレーションの TD コマンドの終了を表示します。

生成条件

シミュレーション情報を含む TD コマンド格納ファイルには必ず生成されます。

16.2.3 データ設定値

TD コマンド格納ファイル中の ASSIGN DATA コマンドで設定されるデータ値を次に示します。

表 16-2　データ設定値

データ属性			設定内容	生成例	
				定義	設定値 ※1 ※2
英数字集団項目			半角空白を設定する。配下の集団項目，基本項目についても，規則に従って設定する。	SYU1. SYU2. KI1 PIC XX. KI2 PIC 99.	’△’ ’△’ ’AB’ 12
英字項目			A で始まる n けたの文字を埋め込む。	PIC A(10)	’ABCDEFGHIJ’
英数字項目				PIC X(4)	’ABCD’
英数字編集項目				PIC XBXBX	’ABC’
数字項目	外部 10 進項目		n けたの数字を埋め込む。小数点の位置も合わせる。P に対しては 0 を埋め込む。	PIC S9V99	1.23
	内部 10 進項目			PIC 9V9 USAGE COMP-3	1.2
	2 進項目			PIC 99PP USAGE COMP	1200
	外部浮動小数点		仮数部に n けたの数字を埋め込む。小数点の位置も合わせる。指数部には E12 を埋め込む。	PIC +9.9E-99	1.2E12
	内部浮動小数点	COMP-1	1.2E34 を埋め込む。	USAGE COMP-1	1.2E34
		COMP-2	2.3E45 を埋め込む。	USAGE COMP-2	2.3E45
ブール項目			n けたの数字を埋め込む。	PIC 1(8) USAGE DISPLAY	B’11111111’
				PIC 1(4) USAGE BIT	B’1111’
数字編集項目			n けたの数字を埋め込む。	PIC ¥¥¥¥¥¥9	1234567
日本語集団項目			全角空白を設定する。配下の集団項目，基本項目についても，規則に従って設定する。	SYU1 GROUP-USAGE NATIONAL. SYU2. KI1 PIC N. KI2 PIC NN.	N’▲’ N’▲’ N’１’ N’１２’
日本語項目			n けたの全角数字を埋め込む。	PIC N(4)	N’１２３４’
日本語編集項目				PIC NBN	N’１２’
指標データ項目			設定しない。	USAGE INDEX	－
アドレスデータ項目				USAGE ADDRESS	－
ポインタ項目				USAGE POINTER	－
オブジェクト参照データ項目				USAGE OBJECT REFERENCE	－
OLE オブジェクト参照データ項目				USAGE OBJECT REFERENCE OLE	－

データ属性	設定内容	生成例	
		定義	設定値 ※1 ※2
バリアントデータ項目		USAGE VARIANT	－
動的長基本項目	連絡節での定義で，LIMIT 指定がある場合，LIMIT に指定されたけた数を上限値として，基本項目の規則に従って設定する。	PIC X DYNAMIC LENGTH C-STRING LIMIT 4. PIC N DYNAMIC LENGTH C-STRING LIMIT 4.	'ABCD' N'１２３４'
	連絡節での定義で，LIMIT 指定がない場合，1,024 けたが仮定されているものとして，基本項目の規則に従って設定する。※3	PIC X DYNAMIC LENGTH C-STRING. PIC N DYNAMIC LENGTH C-STRING.	'ABCDE･･･' N'１２３４･･･'
その他	FILLER 項目は設定しない。	FILLER PIC X(4)	－

(凡例)

－：設定されない

注※1

埋め込めるけた数の最大は 28 けたです。

注※2

OCCURS 句が付くときは添字(1)だけを設定します。

注※3

-SimMain オプションを指定した場合の主プログラムシミュレーションのときだけ生成されます。

16.3　TD コマンドの生成例

次の「被テスト COBOL 原始プログラム」から生成される次のファイルの内容を示します。

- 中断点情報ファイル単独作成のときの中断点情報ファイル
- シミュレーション情報ファイルと同時作成時の中断点情報ファイル
- シミュレーション情報ファイル
- 中断点情報・シミュレーション情報ファイル

図 16-1　被テスト COBOL 原始プログラム（ファイル名：FLI0010.cbl）

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. FLI0010.
000300 ENVIRONMENT DIVISION.
000400 INPUT-OUTPUT SECTION.
000500 FILE-CONTROL.
000600     SELECT URIAGE-FILE
000700          ASSIGN TO SYS001.
000800     SELECT HATTCHUU-FILE
000900          ASSIGN TO SYS002.
001000 DATA DIVISION.
001100 FILE SECTION.
001200 FD  URIAGE-FILE
001300           RECORDING MODE IS F
001400           LABEL RECORD IS STANDARD.
001500 01  URIAGE-REC.
001600     03  URIAGE-CODE       PIC X(10).
001700     03  URIAGE-DATE       PIC X(6).
001800     03  URIAGE-DATA.
001900       05  URIAGE-HINMOKU    OCCURS 50.
002000         07  URIAGE-NAME       PIC X(10).
002100         07  URIAGE-SUURYO     PIC 9(07)  COMP-3.
002200         07  URIAGE-TANKA      PIC 9(07)  COMP-3.
002300         07  URIAGE-GOOKEI     PIC 9(07)  COMP-3.
002400 FD  HATTCHUU-FILE
002500           RECORDING MODE IS F
002600           LABEL RECORD IS STANDARD.
002700 01  HATTCHUU-REC.
002800     03  HATTCHUU-CODE     PIC X(10).
002900     03  HATTCHUU-DATE     PIC X(6).
003000     03  HATTCHUU-DATA.
003100       05  HATTCHUU-HINMOKU  OCCURS 50.
003200         07  HATTCHUU-NAME     PIC X(10).
003300         07  HATTCHUU-SUURYO   PIC 9(07)  COMP-3.
003400         07  HATTCHUU-GENKA    PIC 9(07)  COMP-3.
003500         07  HATTCHUU-GOOKEI   PIC 9(07)  COMP-3.
```

```
003600 WORKING-STORAGE SECTION.
003700 01  I                        PIC S9(4)   COMP.
003800 01  J                        PIC S9(4)   COMP.
003900 01  W-CODE                   PIC X(10).
004000 01  W-HATTCHUU.
004100     03  W-HINMOKU         OCCURS 50.
004200       07  W-NAME             PIC X(10).
004300       07  W-SUURYO           PIC 9(07)  COMP-3.
004400       07  W-GENKA            PIC 9(07)  COMP-3.
004500       07  W-GOOKEI           PIC 9(07)  COMP-3.
004600       07  W-FUSOKU-SW        PIC X(01).
004700 01  HATTCHUU-FLAG            PIC X(01).
004800 LINKAGE SECTION.
004900 01  W-YYMMDD                 PIC X(6).
005000 PROCEDURE DIVISION USING W-YYMMDD.
005100 MAIN      SECTION.
005200 MAIN-START.
005300     OPEN  INPUT  URIAGE-FILE.
005400     OPEN  OUTPUT HATTCHUU-FILE.
005500 URIAGE-FILE-READ.
005600     MOVE  SPACE  TO  HATTCHUU-FLAG.
005700     MOVE  ZERO   TO  J.
005800     READ  URIAGE-FILE  AT END  GO  TO  OWARI.
005900     MOVE  URIAGE-CODE  TO  W-CODE.
006000     MOVE  URIAGE-DATA  TO  W-HATTCHUU.
006100     CALL  'FLIO020'  USING  W-CODE  W-HATTCHUU.
006200     PERFORM  HATTCHUU-CHECK
006300              VARYING I FROM 1 BY 1 UNTIL (I > 50).
006400     IF  HATTCHUU-FLAG = '1'
006500         WRITE  HATTCHUU-REC.
006600     GO  TO  URIAGE-FILE-READ.
006700 OWARI.
006800     CLOSE  URIAGE-FILE.
006900     CLOSE  HATTCHUU-FILE.
007000 MAIN-EXIT.
007100     GOBACK.
007200 HATTCHUU-CHECK   SECTION.
007300 HATTCHUU-CHECK-START.
007400     IF  W-FUSOKU-SW (1) = '1'
007500         MOVE  '1'  TO  HATTCHUU-FLAG
007600         ADD   1    TO  J
007700         MOVE W-CODE     TO HATTCHUU-CODE
007800         MOVE W-YYMMDD   TO HATTCHUU-DATE
007900         MOVE W-NAME (I)   TO HATTCHUU-NAME (J)
008000         MOVE W-SUURYO (I) TO HATTCHUU-SUURYO (J)
008100         MOVE W-GENKA (I)  TO HATTCHUU-GENKA (J)
008200         MOVE W-GOOKEI (I) TO HATTCHUU-GOOKEI (J).
008300 HATTCHUU-CHECK-EXIT.
008400     EXIT.
```

図 16-2　中断点情報ファイル単独作成時の中断点情報ファイル

```
*>*****************************************************************************
*>**    SET BREAK TD COMMAND START                                           ***
*>*****************************************************************************
*>*****************************************************************************
*>**    TEST PROGRAM ==> FLI0010                                             ***
*>*****************************************************************************
SET QUALIFICATION(#PROGRAM('FLI0010'/'FLI0010'))
SET BREAK PARAGRAPH(MAIN) DO
 GO
ENDDO
SET BREAK STATEMENT(0005800) DO
 GO
ENDDO
SET BREAK STATEMENT(0006200) DO
 GO
ENDDO
SET BREAK STATEMENT(0006400) DO
 GO
ENDDO
SET BREAK STATEMENT(0007100) DO
 GO
ENDDO
SET BREAK PARAGRAPH(HATTCHUU-CHECK) DO
 GO
ENDDO
SET BREAK STATEMENT(0007400) DO
 GO
ENDDO
*>**    SET BREAK TD COMMAND ENDED                                           ***
SET BREAK PROCEDURE(#PROGRAM('FLI0010'/'FLI0010')#ENTRY) DO
    ASSIGN DATA(W-YYMMDD) VALUE('ABCDEF')
    GO
ENDDO
GO
```

図 16-3　シミュレーション情報ファイルと同時作成の中断点情報ファイル

```
*>*************************************************************************
*>**    SET BREAK TD COMMAND START                                       ***
*>*************************************************************************
*>*************************************************************************
*>**    TEST PROGRAM ==> FLIO010                                         ***
*>*************************************************************************
SET QUALIFICATION(#PROGRAM('FLIO010'/'FLIO010'))
#INCLUDE INFILE(C:¥PRJ¥FLIO010.tds) NOMESSAGE
SET BREAK PARAGRAPH(MAIN) DO
 GO
ENDDO
SET BREAK STATEMENT(0005800) DO
 GO
ENDDO
SET BREAK STATEMENT(0006200) DO
 GO
ENDDO
SET BREAK STATEMENT(0006400) DO
 GO
ENDDO
SET BREAK STATEMENT(0007100) DO
 GO
ENDDO
SET BREAK PARAGRAPH(HATTCHUU-CHECK) DO
 GO
ENDDO
SET BREAK STATEMENT(0007400) DO
 GO
ENDDO
*>**    SET BREAK TD COMMAND ENDED                                       ***
SET BREAK PROCEDURE(#PROGRAM('FLIO010'/'FLIO010')#ENTRY) DO
    ASSIGN DATA(W-YYMMDD) VALUE('ABCDEF')
    GO
ENDDO
GO
```

図 16-4　シミュレーション情報ファイル

```
*>*************************************************************************
*>**    SIMULATE TD COMMAND START                                        ***
*>*************************************************************************
*>*************************************************************************
*>**    MAIN PROGRAM SIMULATE ==> FLI0010                                ***
*>*************************************************************************
SIMULATE MAIN(#PROGRAM('FLI0010'))
 ASSIGN DATA(W-YYMMDD) VALUE('ABCDEF')
ENDSIMULATE
*>*************************************************************************
*>**    SUB PROGRAM SIMULATE ==> FLI0020                                 ***
*>*************************************************************************
SIMULATE SUB(#PROGRAM('FLI0020'))
            USING(PARM00000001,PARM00000002)
  DEFINE
    01 PARM00000001
    01 PARM00000002
  ENDDEFINE
  DISPLAY DATA(PARM00000001)
    DISPLAY DATA(PARM00000002)
  ENDSIMULATE
*>*************************************************************************
*>**    FILE   SIMULATE ==> URIAGE-FILE                                  ***
*>*************************************************************************
SIMULATE FILE(#PROGRAM('FLI0010'/'FLI0010') URIAGE-FILE) OPENMODE(INPUT)
COUNTER(C1)
  IF CONDITION(C1=1)
               ASSIGN DATA(URIAGE-REC) VALUE(' ')
               ASSIGN DATA(URIAGE-CODE) VALUE('ABCDEFGHIJ')
               ASSIGN DATA(URIAGE-DATE) VALUE('ABCDEF')
               ASSIGN DATA(URIAGE-DATA) VALUE(' ')
               ASSIGN DATA(URIAGE-HINMOKU(1)) VALUE(' ')
               ASSIGN DATA(URIAGE-NAME(1)) VALUE('ABCDEFGHIJ')
               ASSIGN DATA(URIAGE-SUURYO(1)) VALUE(1234567)
               ASSIGN DATA(URIAGE-TANKA(1)) VALUE(1234567)
               ASSIGN DATA(URIAGE-GOOKEI(1)) VALUE(1234567)
             ELSE
               GO END
             ENDIF
ENDSIMULATE
*>*************************************************************************
*>**    FILE   SIMULATE ==> HATTCHUU-FILE                                ***
*>*************************************************************************
SIMULATE FILE(#PROGRAM('FLI0010'/'FLI0010') HATTCHUU-FILE) OPENMODE(OUTPUT)
            DISPLAY DATA(HATTCHUU-REC)
ENDSIMULATE
*>**    SIMULATE TD COMMAND ENDED                                        ***
```

図 16-5　中断点情報・シミュレーション情報ファイル

```
*>*****************************************************************************
*>**    SET BREAK TD COMMAND START                                           ***
*>*****************************************************************************
*>*****************************************************************************
*>**    TEST PROGRAM ==> FLI0010                                             ***
*>*****************************************************************************
SET QUALIFICATION(#PROGRAM('FLI0010'/'FLI0010'))
SET BREAK PARAGRAPH(MAIN) DO
 GO
ENDDO
SET BREAK STATEMENT(0005800) DO
 GO
ENDDO
SET BREAK STATEMENT(0006200) DO
 GO
ENDDO
SET BREAK STATEMENT(0006400) DO
 GO
ENDDO
SET BREAK STATEMENT(0007100) DO
 GO
ENDDO
SET BREAK PARAGRAPH(HATTCHUU-CHECK) DO
 GO
ENDDO
SET BREAK STATEMENT(0007400) DO
 GO
ENDDO
*>**    SET BREAK TD COMMAND ENDED                                           ***
*>*****************************************************************************
*>**    SIMULATE TD COMMAND START                                            ***
*>*****************************************************************************
*>*****************************************************************************
*>**    MAIN PROGRAM SIMULATE ==> FLI0010                                    ***
*>*****************************************************************************
SIMULATE MAIN(#PROGRAM('FLI0010'))
 ASSIGN DATA(W-YYMMDD) VALUE('ABCDEF')
ENDSIMULATE
*>*****************************************************************************
*>**    SUB PROGRAM SIMULATE ==> FLI0020                                     ***
*>*****************************************************************************
SIMULATE SUB(#PROGRAM('FLI0020'))
             USING(PARM00000001,PARM00000002)
  DEFINE
    01 PARM00000001
    01 PARM00000002
  ENDDEFINE
  DISPLAY DATA(PARM00000001)
    DISPLAY DATA(PARM00000002)
  ENDSIMULATE
```

```
*>*************************************************************************
*>**    FILE   SIMULATE ==> URIAGE-FILE                                  ***
*>*************************************************************************
SIMULATE FILE(#PROGRAM('FLI0010'/'FLI0010') URIAGE-FILE) OPENMODE(INPUT)
COUNTER(C1)
               IF CONDITION(C1=1)
                 ASSIGN DATA(URIAGE-REC) VALUE(' ')
                 ASSIGN DATA(URIAGE-CODE) VALUE('ABCDEFGHIJ')
                 ASSIGN DATA(URIAGE-DATE) VALUE('ABCDEF')
                 ASSIGN DATA(URIAGE-DATA) VALUE(' ')
                 ASSIGN DATA(URIAGE-HINMOKU(1)) VALUE(' ')
                 ASSIGN DATA(URIAGE-NAME(1)) VALUE('ABCDEFGHIJ')
                 ASSIGN DATA(URIAGE-SUURYO(1)) VALUE(1234567)
                 ASSIGN DATA(URIAGE-TANKA(1)) VALUE(1234567)
                 ASSIGN DATA(URIAGE-GOOKEI(1)) VALUE(1234567)
               ELSE
                 GO END
               ENDIF
ENDSIMULATE
*>*************************************************************************
*>**    FILE   SIMULATE ==> HATTCHUU-FILE                                ***
*>*************************************************************************
SIMULATE FILE(#PROGRAM('FLI0010'/'FLI0010') HATTCHUU-FILE) OPENMODE(OUTPUT)
               DISPLAY DATA(HATTCHUU-REC)
ENDSIMULATE
*>**    SIMULATE TD COMMAND ENDED                                        ***
SET BREAK PROCEDURE(#PROGRAM('FLI0010'/'FLI0010')#ENTRY) DO
    ASSIGN DATA(W-YYMMDD) VALUE('ABCDEF')
    GO
ENDDO
GO
```

17 COBOL2002 情報抽出ツール

この章では，COBOL2002 情報抽出ツールについて説明します。

17.1　概要

17.1.1　COBOL2002 情報抽出ツールとは

COBOL2002 情報抽出ツールは，バイナリファイル中の COBOL2002 の情報を抽出するツールです。COBOL2002 の COBOL プログラムで障害が発生した場合に，このツールを使用することで調査に有効な資料が抽出でき，障害調査の効率が向上します。

COBOL2002 情報抽出ツールの入力ファイルと抽出できる情報の対応を次の表に示します。

表 17–1　COBOL2002 情報抽出ツールの入力ファイルと抽出できる情報の対応

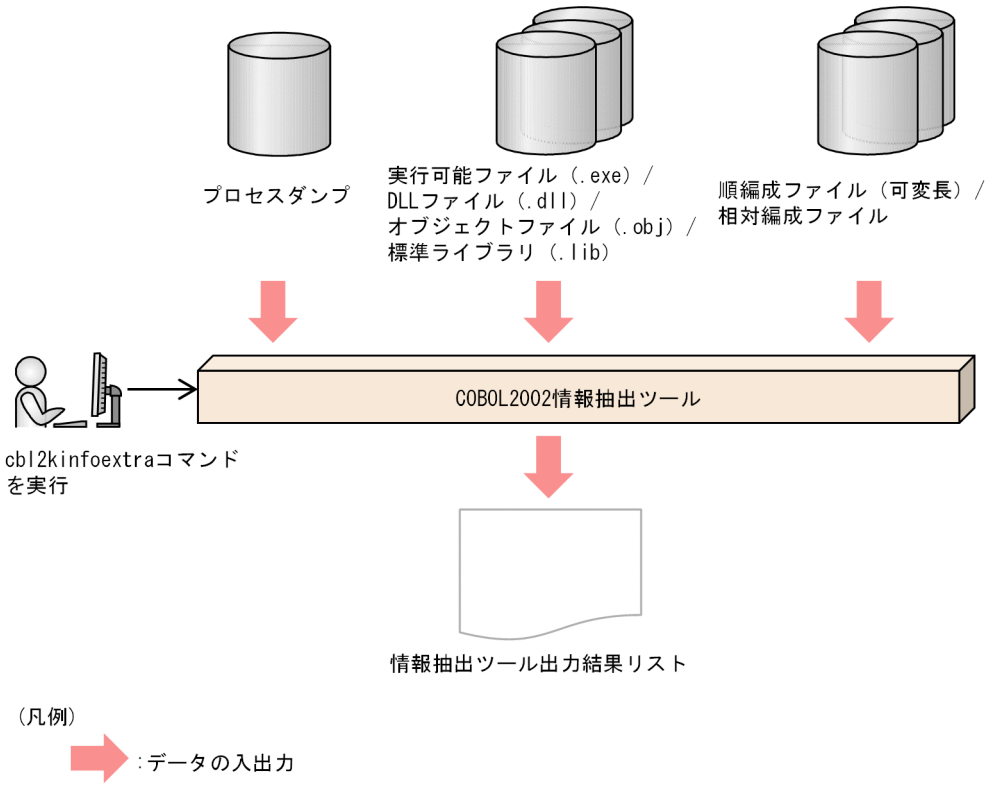
入力ファイル	抽出できる情報
プロセスダンプ	• COBOL 製品のバージョン • 最終実行 COBOL プログラム • 最終実行文の行番号 • 実行中プログラム • 実行済みプログラム • システムトレースバック • データ領域のダンプ
• 実行可能ファイル（.exe） • DLL ファイル（.dll）	• COBOL 製品のバージョン • コンパイル日時 • コンパイラオプション • シンボル情報 • 関連 DLL 一覧
• オブジェクトファイル（.obj） • 標準ライブラリ（.lib）	• COBOL 製品のバージョン • コンパイル日時 • コンパイラオプション
• 順編成ファイル（可変長） • 相対編成ファイル	• COBOL 製品のバージョン • ファイル編成 • ファイル作成日時 • 最大レコード長，最小レコード長 • 相対編成の最大スロット番号

17.1.2　COBOL2002 情報抽出ツールの入出力構成と使用するファイル

(1)　入出力構成

COBOL2002 情報抽出ツールの入出力構成を次に示します。

図 17-1　COBOL2002 情報抽出ツールの入出力構成



(2)　使用するファイルと出力リスト

COBOL2002 情報抽出ツールで使用するファイルと出力リストを次に示します。

- プロセスダンプ

  COBOL2002 の情報の抽出対象となるプロセスダンプです。プロセスダンプの採取方法については，マニュアル「COBOL2002　ユーザーズガイド」の「トラブルシュートに有効な資料」を参照してください。
- 実行可能ファイル

  COBOL2002 の情報の抽出対象となる実行可能ファイル（.exe）です。
- DLL ファイル

  COBOL2002 の情報の抽出対象となる DLL ファイル（.dll）です。
- オブジェクトファイル

  COBOL2002 の情報の抽出対象となるオブジェクトファイル（.obj）です。
- 標準ライブラリ

  COBOL2002 の情報の抽出対象となる標準ライブラリ（.lib）です。
- 順編成ファイル（可変長）

  COBOL2002 の情報の抽出対象となる可変長レコード形式の順編成ファイルです。
- 相対編成ファイル

COBOL2002 の情報の抽出対象となる相対編成ファイルです。

- 情報抽出ツール出力結果リスト

  COBOL2002 情報抽出ツール（cbl2kinfoextra コマンド）の実行結果です。情報抽出ツール出力結果リストは，標準出力（stdout）に出力されます。

17.2　COBOL2002 情報抽出ツールの使用方法

コマンドプロンプトからコマンドを入力して COBOL2002 情報抽出ツールを起動し，情報抽出ツール出力結果リストを出力します。

17.2.1　COBOL2002 情報抽出ツールの使用手順

COBOL2002 情報抽出ツールを使用するときの作業の流れを説明します。

1. COBOL2002 の情報を抽出したいバイナリファイルを用意する。

   抽出対象のバイナリファイルについては，「17.1.2　COBOL2002 情報抽出ツールの入出力構成と使用するファイル」を参照してください。

2. 手順 1.で用意したバイナリファイルを指定して，cbl2kinfoextra コマンドを実行する。

   cbl2kinfoextra コマンドについては，「17.2.2　cbl2kinfoextra コマンドの指定形式」を参照してください。

3. 標準出力に出力された情報抽出ツール出力結果リストを確認する。

   情報抽出ツール出力結果リストについては，「17.3　情報抽出ツール出力結果リストの出力形式」を参照してください。

17.2.2　cbl2kinfoextra コマンドの指定形式

形式

```
cbl2kinfoextra { 入力ファイル名 …| [{ -Help|-? }] }
```

入力ファイル名

COBOL2002 情報を抽出するプロセスダンプや，COBOL2002 情報の抽出対象となるファイル名を指定します。入力ファイルには，次のファイルが指定できます。

- プロセスダンプ
- 実行可能ファイル（.exe）
- DLL ファイル（.dll）
- オブジェクトファイル（.obj）
- 標準ライブラリ（.lib）
- 順編成ファイル（可変長）
- 相対編成ファイル

-Help または-?

cbl2kinfoextra コマンドの構文が標準出力に出力されます。-Help または-?を指定すると，ほかの引数はすべて無視されます。

注意事項

- 先頭文字がハイフン（-）の引数は，オプションになります。ハイフン（-）の代わりにスラント（/）も使用できます。
- オプションの先頭文字に続く大文字と小文字は，等価とみなします。
- 引数の区切り記号は，一つ以上の空白文字またはタブです。ただし，引数に空白文字を含むファイル名を指定する場合は，ファイル名をダブルコーテーション（"）で囲んでください。
- 入力ファイル名にはワイルドカード（*，?など）を指定できません。指定した場合は，ファイル名としてそのまま使用されます。
- 入力ファイル名は，Windows のファイル名の規則に従って指定します。
- コマンド名だけを指定した場合は，コマンドの構文が標準出力に出力されます。
- -Help または-?の指定がなく，指定できないオプションがある場合は，メッセージを出力して cbl2kinfoextra コマンドを終了します。
- コマンドが返すリターン値は，次のとおりです。

リターン値	内容
0	正常終了
1	エラー発生による終了

- COBOL2002 情報を抽出できなかった場合や，抽出した COBOL2002 情報に対して補足情報がある場合は，情報抽出ツール出力結果リストにメッセージが出力されます。出力されるメッセージについては，「17.5　COBOL2002 情報抽出ツールのメッセージ」を参照してください。

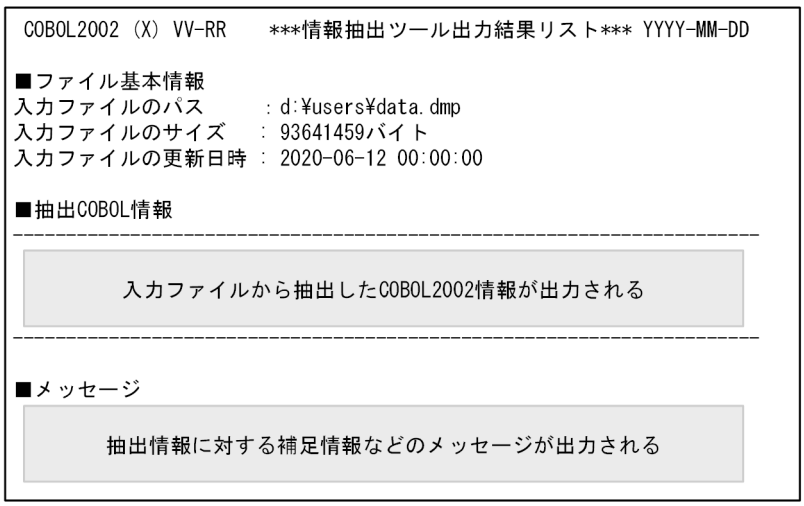
17.3　情報抽出ツール出力結果リストの出力形式

COBOL2002 情報抽出ツールで出力される情報抽出ツール出力結果リストの形式を説明します。

出力形式

情報抽出ツール出力結果リストの出力形式を次に示します。

cbl2kinfoextra コマンドで入力ファイル名を複数指定した場合は，各入力ファイルの抽出情報が，指定したファイルの順序で繰り返し出力されます。



情報抽出ツール出力結果リストの説明

情報抽出ツール出力結果リストに出力される項目を次に示します。

項目		出力内容
ヘッダ	(x)	COBOL2002 の識別記号。COBOL2002 の識別記号については，「付録 F.2　このマニュアルでの表記」を参照
	VV-RR	COBOL2002 情報抽出ツールのバージョン番号
	YYYY-MM-DD	COBOL2002 情報抽出ツールを実行した日付（年-月-日）
ファイル基本情報※	入力ファイルのパス	入力ファイルの絶対パス名
	入力ファイルのサイズ	入力ファイルのサイズ（バイト）
	入力ファイルの更新日時	入力ファイルを最後に更新した日時（年-月-日 時:分:秒）
抽出 COBOL 情報※		入力ファイルから抽出した COBOL2002 情報
メッセージ		COBOL2002 情報抽出ツールが出力するメッセージ。次の場合に出力される。 • 抽出した COBOL2002 情報に対して補足情報がある場合 • COBOL2002 情報が抽出できなかった場合

注※

COBOL2002 情報が抽出できなかった場合，ファイル基本情報や抽出 COBOL 情報が出力されないことがあります。

出力例

```
  COBOL2002 (X) 04-40    ***情報抽出ツール出力結果リスト*** 2021-08-26

■ファイル基本情報
  入力ファイルのパス        :D:¥temp¥sample.obj
  入力ファイルのサイズ      :2576バイト
  入力ファイルの更新日時    :2021-01-15 19:24:34

■抽出COBOL情報
------------------------------------------------------------------------

■オブジェクト情報
  コンパイル日時            :2021-01-15 19:29:29
  COBOL製品バージョン       :Windows COBOL85 06-05
  コンパイラオプション      :-I4 -O1 -Mw
  プロダクトID              :P-2634-2154
------------------------------------------------------------------------

■メッセージ
  [INF] :COBOL85で作成されたファイルです。COBOL2002で再作成することを
          推奨します。
```

17.3.1 プロセスダンプからの抽出情報

入力ファイル名にプロセスダンプを指定した場合,「抽出 COBOL 情報」には，プロセスダンプ中の COBOL2002 情報を抽出した結果（プロセス情報と，スレッドごとのスレッド情報）が出力されます。

出力形式

```
■プロセス情報
  実行環境バージョン       :＜COBOL2002のバージョン＞
  モジュール一覧           :
    D:¥temp¥sample1.exe
            :
  シンボル名一覧           :
    D:¥temp¥sample1.exe
      @compile@info@SAMPLE1
            :
  環境変数情報             :
    =D:=D:¥temp
            :
■スレッド情報
  スレッドID               :＜スレッドID＞
  最終実行COBOLプログラム  :＜プログラム名＞
  最終実行文の行番号       :＜行番号＞
  実行中プログラム         :
    プログラム名           行番号
    ＜プログラム名＞       ＜行番号＞ ＜行番号＞ ...
    ＜プログラム名＞       ＜行番号＞ ＜行番号＞ ...
            :
  実行済みプログラム       :
    ＜プログラム名＞  ＜プログラム名＞ ...
            :
  システムトレースバック   :
    SUB1+0x2e3 [D:¥temp¥sub1.dll]
    SAMPLE1+0x4be [D:¥temp¥sample1.exe]
    SAMPLE2+0x62c [D:¥temp¥sample1.exe]
            :
  データ領域ダンプ         :
    プログラム名 ＝ ＜プログラム名＞
    ＜FILE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    ＜WORKING-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................
            :
    ＜LOCAL-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................
            :
    プログラム名 ＝ ＜プログラム名＞
    ＜WORKING-STORAGE SECTION＞
            :
    EXTERNAL句を指定したファイルのレコード領域
    ファイル名 ＝ ＜ファイル名＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................
            :
    EXTERNAL句を指定したデータ領域
    データ名 ＝ ＜データ名＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................
            :
■スレッド情報
  スレッドID               :＜スレッドID＞
            :
```

プロセスダンプからの抽出情報の説明

情報抽出ツール出力結果リストの「抽出 COBOL 情報」に出力される項目を次に示します。

項目		出力内容
プロセス情報	実行環境バージョン	COBOL プログラム実行時に動作していた COBOL2002 実行時ライブラリのバージョン
	モジュール一覧	プログラム実行時にロードされていた実行可能ファイルや DLL ファイルなどのモジュール一覧
	シンボル名一覧	モジュール一覧のモジュールでエクスポートされているシンボル名の一覧
	環境変数情報	プログラム実行時に設定されていた環境変数情報
スレッド情報	スレッド ID	COBOL プログラムが動作していたスレッドのスレッド ID
	最終実行 COBOL プログラム	プロセスダンプ出力時に実行されていた COBOL プログラムのうち，最後に動作していたプログラム名 COBOL プログラムが実行中でなかった場合は何も出力されない
	最終実行文の行番号	最終実行 COBOL プログラムの最終実行文の行番号 -DebugInf オプションの指定がない場合は何も出力されない
	実行中プログラム	プロセスダンプ出力時に実行されていた COBOL プログラムのプログラム名およびトレースバック情報 -DebugInf,Trace オプションに指定した COBOL プログラムの情報（EXIT PROGRAM 文を実行した COBOL プログラムを除く）が，直近に実行した順序で出力される。
	実行済みプログラム	プロセスダンプ出力時に実行中の COBOL プログラムおよび実行済み COBOL プログラムのプログラム名 COBOL プログラムが一度も実行していない場合は何も出力されない
	システムトレースバック	プロセスダンプ出力時点の，C プログラムやシステム関数を含むトレースバック
	データ領域ダンプ	実行中 COBOL プログラムで定義していた作業場所節，局所場所節，ファイル節，および EXTERNAL 句が指定されたデータ項目のダンプ

出力例

```
  COBOL2002 (X) 04-40     ***情報抽出ツール出力結果リスト*** 2021-08-26

■ファイル基本情報
  入力ファイルのパス         :D:¥temp¥sample1.DMP
  入力ファイルのサイズ       :40262940バイト
  入力ファイルの更新日時     :2021-02-08 20:10:52

■抽出COBOL情報
--------------------------------------------------------------------

■プロセス情報
  実行環境バージョン         :04-30
  モジュール一覧             :
    D:¥temp¥sample1.exe
    C:¥Windows¥SysWOW64¥ntdll.dll
    C:¥Windows¥SysWOW64¥kernel32.dll
    C:¥Windows¥SysWOW64¥KERNELBASE.dll
    C:¥Windows¥SysWOW64¥sysfer.dll
```

```
  C:¥Windows¥SysWOW64¥ucrtbase.dll
  C:¥Windows¥SysWOW64¥VCRUNTIME140.dll
  D:¥temp¥sub1.dll
  D:¥Program Files (x86)¥Hitachi¥COBOL2002¥BIN¥CBL85RT.DLL
  C:¥Windows¥SysWOW64¥user32.dll
  C:¥Windows¥SysWOW64¥win32u.dll
  C:¥Windows¥SysWOW64¥gdi32.dll
  C:¥Windows¥SysWOW64¥gdi32full.dll
  C:¥Windows¥SysWOW64¥msvcp_win.dll
  C:¥Windows¥SysWOW64¥version.dll
  C:¥Windows¥SysWOW64¥msvcrt.dll
  C:¥Windows¥SysWOW64¥comdlg32.dll
  C:¥Windows¥SysWOW64¥combase.dll
  C:¥Windows¥SysWOW64¥rpcrt4.dll
  C:¥Windows¥SysWOW64¥SHCore.dll
  C:¥Windows¥SysWOW64¥shlwapi.dll
  C:¥Windows¥SysWOW64¥winspool.drv
  C:¥Windows¥SysWOW64¥shell32.dll
  C:¥Windows¥SysWOW64¥advapi32.dll
  C:¥Windows¥SysWOW64¥sechost.dll
  C:¥Windows¥WinSxS¥x86_microsoft.windows.common-controls_6595b64144
    ccf1df_5.82.19041.488_none_89e6152f0b32762e¥comctl32.dll
  C:¥Windows¥SysWOW64¥ole32.dll
  D:¥Program Files (x86)¥Hitachi¥COBOL2002¥BIN¥cblucrt2k.dll
  C:¥Windows¥SysWOW64¥imm32.dll
  C:¥Windows¥SysWOW64¥kernel.appcore.dll
  C:¥Windows¥SysWOW64¥windows.storage.dll
  C:¥Windows¥SysWOW64¥wldp.dll
  C:¥Windows¥SysWOW64¥profapi.dll
シンボル名一覧          :
  D:¥temp¥sample1.exe
    @compile@info@SAMPLE1
    @compile@info@SAMPLE2
    SAMPLE1@TD@CBL02
    SAMPLE2@TD@CBL02
    SAMPLE1
    SAMPLE2
  C:¥Windows¥SysWOW64¥ntdll.dll
    RtlDispatchAPC
    RtlActivateActivationContextUnsafeFast
    RtlDeactivateActivationContextUnsafeFast
    RtlInterlockedPushListSList
    RtlUlongByteSwap
    RtlUlonglongByteSwap
    RtlUshortByteSwap
    A_SHAFinal
    A_SHAInit
    A_SHAUpdate
    AlpcAdjustCompletionListConcurrencyCount
     (省略)
  C:¥Windows¥SysWOW64¥kernel32.dll
    BaseThreadInitThunk
    InterlockedPushListSList
    Wow64Transition
    AcquireSRWLockExclusive
    AcquireSRWLockShared
     (省略)
  C:¥Windows¥SysWOW64¥wldp.dll
```

```
    WldpIsAppApprovedByPolicy
    WldpQueryWindowsLockdownMode
    WldpQueryWindowsLockdownRestriction
    WldpSetWindowsLockdownRestriction
    WldpAddDeveloperCertificateForDynamicCodeTrust
    WldpCheckDeviceEncryptionNotStarted
    WldpCheckRetailConfiguration
    WldpCheckSecurityWatermarkState
    WldpCheckWcosDeviceEncryptionSecure
    WldpDisableDeveloperMode
    WldpEnableDeveloperMode
    WldpGetLockdownPolicy
    WldpIsAllowedEntryPoint
    WldpIsClassInApprovedList
    WldpIsDebugAllowed
    WldpIsDynamicCodePolicyEnabled
    WldpQueryDynamicCodeTrust
    WldpQuerySecurityPolicy
    WldpResetSecurityWatermarkState
    WldpSetDynamicCodeTrust2
    WldpSetDynamicCodeTrust
  C:¥Windows¥SysWOW64¥profapi.dll
環境変数情報            :
  =::=::¥
  =D:=D:¥temp
  =ExitCode=00000000
  ALLUSERSPROFILE=C:¥ProgramData
  APPDATA=C:¥Users¥20165857¥AppData¥Roaming
  CBLEXCEPT=THROW
  CBLLIB=D:¥Program Files (x86)¥Hitachi¥COBOL2002¥Copy;D:¥Program Fi
    les¥Hitachi¥COBOL2002 64bit¥copy;D:¥Program Files (x86)¥Hitachi¥
    COBOL2002¥copy
  CLIENTNAME=GUJPVP163234048
  CommonProgramFiles=C:¥Program Files¥Common Files
  CommonProgramFiles(x86)=C:¥Program Files (x86)¥Common Files
  CommonProgramW6432=C:¥Program Files¥Common Files
  COMPUTERNAME=OOBA
  ComSpec=C:¥WINDOWS¥system32¥cmd.exe
  DevEnvDir=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥Comm
    on7¥IDE¥
  DriverData=C:¥Windows¥System32¥Drivers¥DriverData
  Framework40Version=v4.0
  FrameworkDir=C:¥Windows¥Microsoft.NET¥Framework¥
  FrameworkDIR32=C:¥Windows¥Microsoft.NET¥Framework¥
  FrameworkVersion=v4.0.30319
  FrameworkVersion32=v4.0.30319
  HOMEDRIVE=C:
  HOMEPATH=¥Users¥20165857
  INCLUDE=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥VC¥INC
    LUDE;D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥VC¥ATLM
    FC¥INCLUDE;C:¥Program Files (x86)¥Windows Kits¥10¥include¥10.0.1
    6299.0¥ucrt;C:¥Program Files (x86)¥Windows Kits¥NETFXSDK¥4.6.1¥i
    nclude¥um;C:¥Program Files (x86)¥Windows Kits¥10¥include¥10.0.16
    299.0¥shared;C:¥Program Files (x86)¥Windows Kits¥10¥include¥10.0
    .16299.0¥um;C:¥Program Files (x86)¥Windows Kits¥10¥include¥10.0.
    16299.0¥winrt;
  Lib=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥VC¥LIB;D:¥
    Program Files (x86)¥Microsoft Visual Studio 14.0¥VC¥ATLMFC¥LIB;C
```

```
  :¥Program Files (x86)¥Windows Kits¥10¥lib¥10.0.16299.0¥ucrt¥x86;
  C:¥Program Files (x86)¥Windows Kits¥NETFXSDK¥4.6.1¥lib¥um¥x86;C:
  ¥Program Files (x86)¥Windows Kits¥10¥lib¥10.0.16299.0¥um¥x86;D:¥
  Program Files (x86)¥Hitachi¥COBOL2002¥Lib;D:¥Program Files (x86)
  ¥Hitachi¥COBOL2002¥Lib¥SDK;C:¥Program Files (x86)¥Windows Kits¥1
  0¥lib¥10.0.16299.0¥ucrt¥x86;C:¥Program Files (x86)¥Windows Kits¥
  10¥lib¥10.0.16299.0¥um¥x86;D:¥Program Files¥Hitachi¥ISAM64¥Lib;D
  :¥Program Files (x86)¥Hitachi¥ISAM¥Lib;D:¥Program Files¥Hitachi¥
  COBOL2002 64bit¥lib;D:¥Program Files¥Hitachi¥COBOL2002 64bit¥lib
  ¥SDK;C:¥Program Files (x86)¥Windows Kits¥10¥lib¥10.0.16299.0¥ucr
  t¥x64;C:¥Program Files (x86)¥Windows Kits¥10¥lib¥10.0.16299.0¥um
  ¥x64;D:¥Program Files (x86)¥Hitachi¥COBOL2002¥lib;D:¥Program Fil
  es (x86)¥Hitachi¥COBOL2002¥lib¥SDK;C:¥Program Files (x86)¥Window
  s Kits¥10¥lib¥10.0.16299.0¥ucrt¥x86;C:¥Program Files (x86)¥Windo
  ws Kits¥10¥lib¥10.0.16299.0¥um¥x86
LIBPATH=C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319;D:¥Program F
  iles (x86)¥Microsoft Visual Studio 14.0¥VC¥LIB;D:¥Program Files
  (x86)¥Microsoft Visual Studio 14.0¥VC¥ATLMFC¥LIB;C:¥Program File
  s (x86)¥Windows Kits¥10¥UnionMetadata;C:¥Program Files (x86)¥Win
  dows Kits¥10¥References;¥Microsoft.VCLibs¥14.0¥References¥Common
  Configuration¥neutral;
LOCALAPPDATA=C:¥Users¥20165857¥AppData¥Local
LOGONSERVER=¥¥OOBA
NETFXSDKDir=C:¥Program Files (x86)¥Windows Kits¥NETFXSDK¥4.6.1¥
NUMBER_OF_PROCESSORS=12
OneDrive=C:¥Users¥HTC-Admin¥OneDrive
OS=Windows_NT
Path=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥Common7¥I
  DE¥CommonExtensions¥Microsoft¥TestWindow;C:¥Program Files (x86)¥
  MSBuild¥14.0¥bin;C:¥Program Files (x86)¥MSBuild¥14.0¥bin;D:¥Prog
  ram Files (x86)¥Microsoft Visual Studio 14.0¥Common7¥IDE¥;D:¥Pro
  gram Files (x86)¥Microsoft Visual Studio 14.0¥VC¥BIN;D:¥Program
  Files (x86)¥Microsoft Visual Studio 14.0¥Common7¥Tools;C:¥Window
  s¥Microsoft.NET¥Framework¥v4.0.30319;D:¥Program Files (x86)¥Micr
  osoft Visual Studio 14.0¥VC¥VCPackages;C:¥Program Files (x86)¥HT
  ML Help Workshop;C:¥Program Files (x86)¥HTML Help Workshop;D:¥Pr
  ogram Files (x86)¥Microsoft Visual Studio 14.0¥Team Tools¥Perfor
  mance Tools;C:¥Program Files (x86)¥Windows Kits¥10¥bin¥x86;C:¥Pr
  ogram Files (x86)¥Microsoft SDKs¥Windows¥v10.0A¥bin¥NETFX 4.6.1
  Tools¥;C:¥Program Files (x86)¥Common Files¥Hitachi;D:¥Program Fi
  les (x86)¥Hitachi¥COBOL2002¥Bin;D:¥Program Files (x86)¥Hitachi¥C
  OBOL2002¥Bin¥SDK;C:¥Program Files (x86)¥Windows Kits¥10¥bin¥10.0
  .16299.0¥x86;C:¥WINDOWS¥system32;C:¥WINDOWS;C:¥WINDOWS¥System32¥
  Wbem;C:¥WINDOWS¥System32¥WindowsPowerShell¥v1.0¥;C:¥WINDOWS¥Syst
  em32¥OpenSSH¥;C:¥Program Files (x86)¥Intel¥Intel(R) Management E
  ngine Components¥DAL;C:¥Program Files¥Intel¥Intel(R) Management
  Engine Components¥DAL;C:¥Program Files (x86)¥Windows Kits¥8.1¥Wi
  ndows Performance Toolkit¥;C:¥Program Files (x86)¥Windows Kits¥1
  0¥Windows Performance Toolkit¥;D:¥Program Files¥TortoiseSVN¥bin;
  C:¥Program Files¥Common Files¥HITACHI;D:¥Program Files¥Hitachi¥I
  SAM64¥Bin;C:¥Program Files (x86)¥Common Files¥HITACHI;D:¥Program
   Files (x86)¥Hitachi¥ISAM¥Bin;D:¥Program Files (x86)¥Hitachi¥Scr
  ipt¥Bin;d:¥win32app¥hitachi¥hirdb_s¥bin;d:¥win32app¥hitachi¥hird
  b_s¥client¥utl;d:¥Program Files¥Git¥cmd;D:¥Program Files¥Hitachi
  ¥COBOL2002 64bit¥bin;D:¥Program Files¥Hitachi¥COBOL2002 64bit¥bi
  n¥SDK;C:¥Program Files (x86)¥Windows Kits¥10¥bin¥10.0.16299.0¥x6
  4;D:¥Program Files (x86)¥Hitachi¥COBOL2002 Professional Tool Kit
  ¥BIN;D:¥Program Files (x86)¥Hitachi¥COBOL2002¥bin;D:¥Program Fil
```

```
      es (x86)¥Hitachi¥COBOL2002¥bin¥SDK;C:¥Program Files (x86)¥Window
      s Kits¥10¥bin¥10.0.16299.0¥x86;C:¥Users¥20165857¥AppData¥Local¥M
      icrosoft¥WindowsApps;E:¥work¥_tool;¥¥cobol2002¥tool¥si;¥¥cobol20
      02¥tool¥Perl64¥bin;D:¥Program Files (x86)¥Hitachi¥COBOL2002 Prof
      essional Tool Kit¥DBUtility¥bin;
    PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
    PDCONFPATH=d:¥win32app¥hitachi¥hirdb_s¥CONF
    PDDIR=d:¥win32app¥hitachi¥hirdb_s
    PDUXPLDIR=d:¥win32app¥hitachi¥hirdb_s¥UXPLDIR
    PROCESSOR_ARCHITECTURE=AMD64
    PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 158 Stepping 10, Genui
      neIntel
    PROCESSOR_LEVEL=6
    PROCESSOR_REVISION=9e0a
    ProgramData=C:¥ProgramData
    ProgramFiles=C:¥Program Files
    ProgramFiles(x86)=C:¥Program Files (x86)
    ProgramW6432=C:¥Program Files
    PROMPT=$P$G
    PSModulePath=C:¥Program Files¥WindowsPowerShell¥Modules;C:¥WINDOWS
      ¥system32¥WindowsPowerShell¥v1.0¥Modules;C:¥Program Files (x86)¥
      Microsoft Azure Information Protection¥Powershell
    PUBLIC=C:¥Users¥Public
    SESSIONNAME=RDP-Tcp#2
    SystemDrive=C:
    SystemRoot=C:¥WINDOWS
    TEMP=C:¥Users¥20165857¥AppData¥Local¥Temp
    TMP=C:¥Users¥20165857¥AppData¥Local¥Temp
    TZ=JST-9
    UCRTVersion=10.0.16299.0
    UniversalCRTSdkDir=C:¥Program Files (x86)¥Windows Kits¥10¥
    USERDOMAIN=OOBA
    USERDOMAIN_ROAMINGPROFILE=OOBA
    USERNAME=20165857
    USERPROFILE=C:¥Users¥20165857
    VCINSTALLDIR=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥V
      C¥
    VisualStudioVersion=14.0
    VS140COMNTOOLS=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0
      ¥Common7¥Tools¥
    VSINSTALLDIR=D:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥
    windir=C:¥WINDOWS
    WindowsLibPath=C:¥Program Files (x86)¥Windows Kits¥10¥UnionMetadat
      a;C:¥Program Files (x86)¥Windows Kits¥10¥References
    WindowsSdkDir=C:¥Program Files (x86)¥Windows Kits¥10¥
    WindowsSDKLibVersion=10.0.16299.0¥
    WindowsSDKVersion=10.0.16299.0¥
    WindowsSDK_ExecutablePath_x64=C:¥Program Files (x86)¥Microsoft SDK
      s¥Windows¥v10.0A¥bin¥NETFX 4.6.1 Tools¥x64¥
    WindowsSDK_ExecutablePath_x86=C:¥Program Files (x86)¥Microsoft SDK
      s¥Windows¥v10.0A¥bin¥NETFX 4.6.1 Tools¥

■スレッド情報
  スレッドID              :4868
  最終実行COBOLプログラム  :SUB1
  実行中プログラム         :
    プログラム名          行番号
      SAMPLE1             000019
```

```
  実行済みプログラム       :
    SUB1        SAMPLE1
  システムトレースバック   :
    SUB1+0x2e3 [D:¥temp¥sub1.dll]
    SAMPLE1+0x4be [D:¥temp¥sample1.exe]
    SAMPLE2+0x62c [D:¥temp¥sample1.exe]
    _o_iswdigit+0x5f [C:¥Windows¥SysWOW64¥ucrtbase.dll]
    BaseThreadInitThunk+0x19 [C:¥Windows¥SysWOW64¥kernel32.dll]
    RtlGetAppContainerNamedObjectPath+0xe4 [C:¥Windows¥SysWOW64¥ntdll.
      dll]
    RtlGetAppContainerNamedObjectPath+0xb4 [C:¥Windows¥SysWOW64¥ntdll.
      dll]
  データ領域ダンプ         :
    プログラム名 ＝ SUB1
    ＜WORKING-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    ＜LOCAL-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    プログラム名 ＝ SAMPLE1
    ＜FILE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................
               LINE 00000010 SAME AS ABOVE
    00000020   00000000 00000000                      ........

    ＜WORKING-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    ＜LOCAL-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    EXTERNAL句を指定したファイルのレコード領域
    ファイル名 ＝ FILE1
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    EXTERNAL句を指定したデータ領域
    データ名 ＝ DATA1
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

■スレッド情報
  スレッドID               :464
  最終実行COBOLプログラム  :SAMPLE2
  最終実行文の行番号       :000020
  実行中プログラム         :
    プログラム名           行番号
      SAMPLE2              000020 000020 000020 000020 000020 000020
                           000020 000020 000020 000020 000020 000020
                           000020 000020 000020 000020 000020 000020
                           000020 000020 000020 000020 000020 000020
                           000020 000020 000020 000020 000020 000020
```

```
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
                        000020 000020 000020 000020 000020 000020
実行済みプログラム      :
  SAMPLE2
システムトレースバック  :
  CLS181a+0x3c [D:¥Program Files (x86)¥Hitachi¥COBOL2002¥BIN¥CBL85RT
    .DLL]
  SAMPLE2+0x3f6 [D:¥temp¥sample1.exe]
  SAMPLE2+0x655 [D:¥temp¥sample1.exe]
  _o_iswdigit+0x5f [C:¥Windows¥SysWOW64¥ucrtbase.dll]
  BaseThreadInitThunk+0x19 [C:¥Windows¥SysWOW64¥kernel32.dll]
  RtlGetAppContainerNamedObjectPath+0xe4 [C:¥Windows¥SysWOW64¥ntdll.
    dll]
  RtlGetAppContainerNamedObjectPath+0xb4 [C:¥Windows¥SysWOW64¥ntdll.
    dll]
データ領域ダンプ        :
  プログラム名 ＝ SAMPLE2
  ＜FILE SECTION＞
  位置----   内容---- -------- -------- --------    ----------------
  00000000   00000000 00000000 00000000 00000000    ................
             LINE 00000010 SAME AS ABOVE
  00000020   00000000 00000000                       ........

  ＜WORKING-STORAGE SECTION＞
  位置----   内容---- -------- -------- --------    ----------------
```

```
    00000000   00000000 00000000 00000000 00000000    ................

    ＜LOCAL-STORAGE SECTION＞
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    EXTERNAL句を指定したファイルのレコード領域
    ファイル名 ＝ FILE1
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................

    EXTERNAL句を指定したデータ領域
    データ名 ＝ DATA1
    位置----   内容---- -------- -------- --------    ----------------
    00000000   00000000 00000000 00000000 00000000    ................
-------------------------------------------------------------------------
```

注

シンボル名一覧の（省略）は，出力リストを省略していることを示します。

17.3.2 実行可能ファイル，DLL ファイルからの抽出情報

入力ファイル名に実行可能ファイル，DLL ファイルを指定した場合，「抽出 COBOL 情報」には，実行可能ファイル，DLL ファイル中の COBOL2002 情報を抽出した結果（ソースファイル，プログラム情報）が出力されます。

出力形式

```
■ソースファイル           : sample_main.CBL
  COBOL製品バージョン     : Windows(x86) COBOL2002 04-30
  コンパイル日時          : YYYY-MM-DD HH:MM:SS
  コンパイラオプション    : -JPN,Alnum -Optimize,1 -Main,System

■ソースファイル           : sample_sub.CBL
  COBOL製品バージョン     : Windows(x86) COBOL2002 04-30
  コンパイル日時          : YYYY-MM-DD HH:MM:SS
  コンパイラオプション    : -MainNotCBL -JPN,Alnum -Optimize,2

■プログラム情報
  シンボル情報            :
    @compile@info@TEST32_MAIN
    @compile@info@TEST32_SUB
    TEST32_MAIN@TD@CBL02
    TEST32_SUB@TD@CBL02
    CblMain
    TEST32_SUB
  関連DLL一覧             :
    cbl85rt.dll
    KERNEL32.dll
    VCRUNTIME140.dll
    api-ms-win-crt-runtime-l1-1-0.dll
    api-ms-win-crt-math-l1-1-0.dll
    api-ms-win-crt-stdio-l1-1-0.dll
    api-ms-win-crt-locale-l1-1-0.dll
    api-ms-win-crt-heap-l1-1-0.dll
```

実行可能ファイル，DLL ファイルからの抽出情報の説明

情報抽出ツール出力結果リストの「抽出 COBOL 情報」に出力される項目を次に示します。

項目		内容
ソースファイル	－	ソースファイル名（ファイル格納フォルダパスは含まれない） COBOL85 で作成されたオブジェクトの場合はソースファイル名を取得できないため「***」と出力される
	COBOL 製品バージョン	COBOL 製品のバージョン。「COBOL 製品の名称△COBOL 製品のバージョン」の形式で出力される（△は半角空白） COBOL 製品の名称には次のどれかが出力される • Windows(x86) COBOL2002 • Windows(x64) COBOL2002 • Windows COBOL85 (例) Windows(x86) COBOL2002 01-01
	コンパイル日時	コンパイルした日時
	コンパイラオプション	入力ファイルに含まれるオブジェクトが COBOL2002 で作成されていた場合 COBOL2002 形式のコンパイラオプション※ 入力ファイルに含まれるオブジェクトが COBOL85 で作成されていた場合 COBOL85 形式のコンパイラオプション※
プログラム情報	シンボル情報	シンボル情報（COBOL プログラム以外の関連するプログラム情報を含む）
	関連 DLL 一覧	実行可能ファイル，DLL ファイルに直接依存している DLL の一覧（COBOL プログラム以外の関連するプログラム情報を含む） 動的なリンクで動的にリンクする DLL は出力されない

注※

COBOL プログラムのオブジェクトから抽出した情報がそのまま出力されます。COBOL プログラムのオブジェクトには，コンパイル時に指定したオプションの一部の情報しか格納されていないため，情報抽出ツール出力結果リストに出力されたコンパイラオプションを指定しても，同じオブジェクトを作成できないことがあります。

また，コンパイル時に COBOL2002 が仮定するオプションが出力されることがあります。COBOL2002 が仮定するオプションについては，「2.3　コンパイラオプション」を参照してください。

出力例

```
  COBOL2002 (X) 04-40     ***情報抽出ツール出力結果リスト*** 2021-08-26

■ファイル基本情報
  入力ファイルのパス        : D:¥temp¥sample_main.exe
  入力ファイルのサイズ      : 13824バイト
  入力ファイルの更新日時    : 2021-02-08 20:27:38

■抽出COBOL情報
------------------------------------------------------------------------

■ソースファイル            : sample_main.CBL
  COBOL製品バージョン       : Windows(x86) COBOL2002 04-30
  コンパイル日時            : 2021-02-08 20:27:38
  コンパイラオプション      : -JPN,Alnum -Optimize,1 -Main,System

■ソースファイル            : sample_sub.CBL
  COBOL製品バージョン       : Windows(x86) COBOL2002 04-30
```

```
  コンパイル日時          : 2021-02-08 20:26:21
  コンパイラオプション    : -MainNotCBL -JPN,Alnum -Optimize,2

■プログラム情報
  シンボル情報            :
    @compile@info@TEST32_MAIN
    @compile@info@TEST32_SUB
    TEST32_MAIN@TD@CBL02
    TEST32_SUB@TD@CBL02
    CblMain
    TEST32_SUB
  関連DLL一覧             :
    cbl85rt.dll
    KERNEL32.dll
    VCRUNTIME140.dll
    api-ms-win-crt-runtime-l1-1-0.dll
    api-ms-win-crt-math-l1-1-0.dll
    api-ms-win-crt-stdio-l1-1-0.dll
    api-ms-win-crt-locale-l1-1-0.dll
    api-ms-win-crt-heap-l1-1-0.dll
----------------------------------------------------------------------
```

17.3.3 オブジェクトファイル，標準ライブラリからの抽出情報

入力ファイル名にオブジェクトファイル，標準ライブラリを指定した場合，「抽出 COBOL 情報」には，COBOL コンパイラで生成したオブジェクトファイル，標準ライブラリ中の COBOL2002 情報を抽出した結果（オブジェクト情報）が出力されます。標準ライブラリに複数の COBOL プログラムのオブジェクトが含まれる場合は，COBOL プログラムのオブジェクト数分のオブジェクト情報が出力されます。

出力形式

```
■オブジェクト情報
  オブジェクトファイル      : test.obj
  コンパイル日時            : YYYY-MM-DD HH:MM:SS
  COBOL製品バージョン       : Windows(x86) COBOL2002 04-30
  コンパイラオプション      : -JPN,Alnum -Optimize,1 -Main,System
  プロダクトID              : P-2636-2344
```

オブジェクトファイル，標準ライブラリからの抽出情報の説明

情報抽出ツール出力結果リストの「抽出 COBOL 情報」に出力される項目を次に示します。

項目		内容
オブジェクト情報	オブジェクトファイル	入力ファイルがオブジェクトファイルの場合 　何も出力されない 入力ファイルが標準ライブラリの場合 　標準ライブラリ作成時，LIB コマンドに指定したオブジェクトファイル名（パスを含む）
	コンパイル日時	コンパイルした日時

項目		内容
	COBOL 製品バージョン	COBOL 製品のバージョン。「COBOL 製品の名称△COBOL 製品のバージョン」の形式で出力される（△は半角空白） COBOL 製品の名称には次のどれかが出力される • Windows(x86) COBOL2002 • Windows(x64) COBOL2002 • Windows COBOL85 (例) Windows(x86) COBOL2002 01-01
	コンパイラオプション	入力ファイルに含まれるオブジェクトが COBOL2002 で作成されていた場合 COBOL2002 形式のコンパイラオプション※ 入力ファイルに含まれるオブジェクトが COBOL85 で作成されていた場合 COBOL85 形式のコンパイラオプション※
	プロダクト ID	COBOL 製品の形名

注※

COBOL プログラムのオブジェクトから抽出した情報がそのまま出力されます。COBOL プログラムのオブジェクトには，コンパイル時に指定したオプションの一部の情報しか格納されていないため，情報抽出ツール出力結果リストに出力されたコンパイラオプションを指定しても，同じオブジェクトを作成できないことがあります。

また，コンパイル時に COBOL2002 が仮定するオプションが出力されることがあります。COBOL2002 が仮定するオプションについては，「2.3　コンパイラオプション」を参照してください。

出力例

```
  COBOL2002 (X) 04-40    ***情報抽出ツール出力結果リスト*** 2021-08-26

■ファイル基本情報
  入力ファイルのパス        : D:¥temp¥test.lib
  入力ファイルのサイズ      : 3002バイト
  入力ファイルの更新日時    : 2021-02-08 20:43:21

■抽出COBOL情報
------------------------------------------------------------------------

■オブジェクト情報
  オブジェクトファイル      : test.obj
  コンパイル日時            : 2021-02-08 20:41:24
  COBOL製品バージョン       : Windows(x86) COBOL2002 04-30
  コンパイラオプション      : -JPN,Alnum -Optimize,1 -Main,System
  プロダクトID              : P-2636-2344
------------------------------------------------------------------------
```

17.3.4　順編成ファイル（可変長），相対編成ファイルからの抽出情報

入力ファイル名に順編成ファイル（可変長），相対編成ファイルを指定した場合，「抽出 COBOL 情報」には，順編成ファイル（可変長），相対編成ファイル中の COBOL2002 情報を抽出した結果（ファイル情報）が出力されます。

出力形式

```
■ファイル情報
  ファイル編成          : 相対編成ファイル（可変長レコード）
  COBOLシステム名       : COBOL2002
  バージョン番号        : 04-30
  作成日時              : YYYY-MM-DD HH:MM:SS
  最大レコード長        : 1024
  最小レコード長        : 11
  最大スロット番号      : 0
```

順編成ファイル（可変長），相対編成ファイルからの抽出情報の説明

情報抽出ツール出力結果リストの「抽出 COBOL 情報」に出力される項目を次に示します。

項目		内容
ファイル情報	ファイル編成	順編成ファイル（可変長）の場合 順編成ファイル（可変長レコード） 相対編成ファイル（可変長）の場合 相対編成ファイル（可変長レコード） 相対編成ファイル（固定長）の場合 相対編成ファイル（固定長レコード）
	COBOL システム名	COBOL2002 で作成したファイルの場合 COBOL2002 COBOL85 で作成したファイルの場合 COBOL85 COBOL2002，COBOL85 以外の製品ユティリティやツールなどで作成したファイルの場合 UNKNOWN
	バージョン番号	ファイルを作成した COBOL 製品のバージョン COBOL2002，COBOL85 以外の製品ユティリティやツールなどで作成したファイルの場合は，「UNKNOWN」と出力される
	作成日時	ファイル作成日時
	最大レコード長	最大レコード長 最大レコード長よりも最小レコード長の方が大きいときは不正な値のため「***」と出力される
	最小レコード長	最小レコード長 最大レコード長よりも最小レコード長の方が大きいときは不正な値のため「***」と出力される
	最大スロット番号	最大スロット番号。相対編成ファイルの場合だけ出力される 最大スロット番号が制限値よりも大きいとき，または最大レコード長よりも最小レコード長の方が大きいときは不正な値のため「***」と出力される

出力例

```
 COBOL2002 (X) 04-40     ***情報抽出ツール出力結果リスト*** 2021-08-26

■ファイル基本情報
```

```
  入力ファイルのパス        :D:¥temp¥file1
  入力ファイルのサイズ      :128バイト
  入力ファイルの更新日時    :2021-02-08 21:26:28

■抽出COBOL情報
------------------------------------------------------------------------

■ファイル情報
  ファイル編成              :相対編成ファイル（可変長レコード）
  COBOLシステム名           :COBOL2002
  バージョン番号            :04-30
  作成日時                  :2021-02-08 21:26:28
  最大レコード長            :1024
  最小レコード長            :11
  最大スロット番号          :0
------------------------------------------------------------------------
```

17.4　COBOL2002 情報抽出ツール使用時の注意事項

- プロセスダンプでファイルの先頭４バイトが「MDMP」以外の場合，プロセスダンプは解析できません。
- COBOL2002 情報抽出ツールでは，指定されたファイルをメモリ上にマッピングします。そのため，サイズの大きいファイルを指定した場合，メモリ不足が発生するおそれがあります。

17.5 COBOL2002 情報抽出ツールのメッセージ

17.5.1 メッセージレベル

COBOL2002 情報抽出ツールでは，メッセージテキストに[xxx]の形式でメッセージレベルを出力します。メッセージレベルには「ERR」と「INF」があります。それぞれの意味を次に示します。

- ERR

  エラーメッセージです。すべての入力ファイル，またはエラーが発生した入力ファイルの COBOL2002 情報の抽出を中止します。情報抽出ツール出力結果リストの「抽出 COBOL 情報」は出力されません。

- INF

  お知らせのメッセージです。情報抽出ツール出力結果リストの「抽出 COBOL 情報」に出力した内容の補足情報です。

17.5.2 メッセージレベルが「ERR」のメッセージの一覧

メッセージレベルが「ERR」のメッセージを次に示します。メッセージ中の可変の埋め字部分は*** n ***（n は数字）で示します。

メッセージテキスト	メッセージの説明
[ERR] ：COBOL2002が正しくインストールされていません。 [対処] COBOL2002情報抽出ツールを起動できる環境が整っていませんので，COBOL2002を再インストールしてください。	COBOL2002 がインストールされていない環境で cbl2kinfoextra コマンドを実行した場合に出力されます。
[ERR] ：オプション"**** 1 ****"は指定できません。 ＜cbl2kinfoextraコマンドの一般形式＞ cbl2kinfoextra 入力ファイル名 ･･･ または， cbl2kinfoextra [{-Help | -?}] [対処] コマンドの一般形式に示す正しいオプションを指定してください。	指定できないオプションを指定した場合に出力されます。 **** 1 **** 指定されたオプション名
[ERR] ：入力ファイルは抽出対象ファイルではありません。または壊れてい るおそれがあります。 [対処] 入力ファイル名には，以下のどれかのファイルを指定してくださ い。指定している場合は，ファイルが壊れていないか確認してく ださい。 ・プロセスダンプ ・オブジェクトファイル(.obj)/標準ライブラリ(.lib) ・実行可能ファイル(.exe)/DLLファイル(.dll) ・COBOL2002の入出力機能で作成したファイル	入力ファイルが COBOL2002 情報抽出ツールの対象ファイルでない場合に出力されます。

メッセージテキスト	メッセージの説明
[ERR]：必要なメモリが確保できません。 [対処] 不要なプログラムを終了後に，コマンドを再実行してください。	抽出処理をするために必要なメモリが確保できない場合に出力されます。
[ERR]：入力ファイルにアクセスできません。 [対処] 以下の情報をもとに，エラーの原因を調査してください。 **** 1 ****	入力ファイルにアクセスできない場合に出力されます。 **** 1 **** • オープンエラーの場合：OPEN Error = nnnnnnnn • 読み込みエラーの場合：READ Error = nnnnnnnn • 位置づけエラーの場合：SEEK Error = nnnnnnnn nnnnnnnn：GetLastError()のエラー番号
[ERR]：入力ファイルのファイル情報が取得できません。 [対処] 以下の情報をもとに，エラーの原因を調査してください。 **** 1 ****	入力ファイルのファイル情報が取得できない場合に出力されます。 **** 1 **** • 入力ファイルが存在しないか，パスが誤っています。 • 入力ファイルにフォルダが指定されています。 • ファイル情報へのアクセスが許可されていません。Error = nnnnnnnn nnnnnnnn：Errno のエラー番号

17.5.3 メッセージレベルが「INF」のメッセージの一覧

メッセージレベルが「INF」のメッセージを次に示します。メッセージ中の可変の埋め字部分は*** n ***（n は数字）で示します。

メッセージテキスト	メッセージの説明
[INF]：COBOL2002情報は含まれていません。	入力ファイルは COBOL2002 情報抽出ツールの対象ファイルであるが，ファイルの中に COBOL2002 情報が含まれていない場合に出力されます。
[INF]：COBOL85で作成されたファイルです。COBOL2002で再作成することを 推奨します。	入力ファイルの OBJ／LIB／EXE／DLL に COBOL85 のオブジェクトが含まれている場合に出力されます。
[INF]：入力ファイルから正しい最大レコード長，最小レコード長が取得で きません。エンディアンがWindowsと異なるOSで作られたファイルのおそれがあります。	順編成ファイルの場合に，ビッグエンディアン形式のファイルを入力ファイルに指定したときに出力されます。

メッセージテキスト	メッセージの説明
[INF]：入力ファイルから正しい最大レコード長，最小レコード長，最大ス ロット番号が取得できません。エンディアンがWindowsと異なるOSで作られたファイルのおそれがあります。	相対編成ファイルの場合に，ビッグエンディアン形式のファイルを入力ファイルに指定したときに出力されます。
[INF]：領域破壊を検知しました。抽出情報は正しくないおそれがあります。 スレッドID = **** 1 **** 領域破壊されたプログラム名 = **** 2 **** 領域破壊したおそれのあるプログラム名 = **** 3 ****	領域が壊れているため，COBOL2002 情報が抽出できなくなった場合に出力されます。 **** 1 **** 領域破壊されたプログラムが動作していたスレッド ID。 「抽出 COBOL 情報」の「スレッド情報」に出力されたスレッド ID に対応しています。 **** 2 **** 領域破壊された領域を使用していたプログラム名。 ただし，プログラム名が特定できない場合は「***」と出力されます。 **** 3 **** 領域破壊したおそれのあるプログラム名。 ただし，領域破壊したおそれのあるプログラム名が不明の場合は「***」と出力されます。 なお，**** 1 ****プログラムから呼び出されている別のプログラムで領域破壊した場合も「***」と出力されます。

18 64bit アプリケーションの開発

この章では，Windows(x64) COBOL2002 について説明します。

18.1　Windows(x64) COBOL2002 について

Windows(x64) COBOL2002 では，Windows(x86) COBOL2002 と比べて呼び出し規約の変更，およびポインタサイズの 64bit 化に対応しています。ここでは，Windows(x86) COBOL2002 と Windows(x64) COBOL2002 について，相違を説明します。

18.1.1　使用できない機能

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 とでは，使用できる機能が異なります。ここでは，Windows(x86) COBOL2002 の機能のうち，Windows(x64) COBOL2002 で使用できない機能について説明します。

(1)　使用できない機能

Windows(x64) COBOL2002 では使用できない機能を次に示します。

表 18–1　Windows(x64) COBOL2002 で使用できない機能

機能名	説明
XMAP3 を使用して書式印刷する機能	書式と行データを重ね合わせる印刷（書式オーバレイ印刷）や，印刷制御付きの行データの印刷を行う。
リモートファイルアクセス機能	接続されているほかの Windows や UNIX 上に存在する，ISAM を使用する索引編成ファイルにアクセスする。
Visual Basic 連携機能	Visual Basic プログラムから，DLL として作成された COBOL プログラムを呼び出す。

(2)　ファイル編成

Windows(x64) COBOL2002 では使用できないファイル編成を次に示します。

- Btrieve（Pervasive.SQL）による索引編成ファイル

(3)　MIOS7 COBOL85 互換機能

Windows(x64) COBOL2002 では，MIOS7 COBOL85 との互換機能を有効にする-CompatiM7 オプションは使用できません。使用できる機能を次に示します。

(a)　言語仕様

-CompatiM7 オプションを指定しなくても使用できる言語仕様を，次に示します。

環境部の入出力管理記述項(I-O-CONTROL)

- APPLY FILE-SHARE 句

環境部のファイル管理記述項

- 相対編成ファイルに対する SYMBOLIC KEY 句
- ISAM による索引編成ファイルに対する SYMBOLIC KEY 句

(b)　実行時環境変数

-CompatiM7 オプションを指定しなくても使用できる実行時環境変数を，次に示します。

- CBLM7ENDKEY
- CBLOPS

18.1.2　使用できないコンパイラオプション

ここでは，Windows(x64) COBOL2002 では使用できないコンパイラオプションについて説明します。

(1)　最終生成物の種類（プロジェクトの種類）のコンパイラオプション

最終生成物※の種類（開発マネージャでは，プロジェクトの種類）を設定するコンパイラオプションのうち，Windows(x64) COBOL2002 では使用できないコンパイラオプションを次に示します。プロジェクトの種類については，「18.2.5　作成できるプログラムの相違」，「18.2.6　プロジェクト設定の相違」を参照してください。

注※

最終生成物とは，コンパイラが最終的に生成する実行可能ファイル，DLL，または標準ライブラリのことを示します。

表 18–2　Windows(x64) COBOL2002 で使用できない最終生成物の種類のコンパイラオプション

コンパイラオプション	説明
-Dll，{Stdcall ｜ Cdecl}	DLL 形式のオブジェクトファイルを出力する。DLL 属性は stdcall，または cdecl である。

(2)　製品連携のコンパイラオプション

他製品との連携を設定するコンパイラオプションのうち，Windows(x64) COBOL2002 では使用できないコンパイラオプションを次に示します。

表 18–3　Windows(x64) COBOL2002 で使用できない製品連携のコンパイラオプション

コンパイラオプション	説明
-IsamExtend〔,Zone〕	Btrieve（Pervasive.SQL）による索引編成ファイルを使用する。

コンパイラオプション	説明
-XMAP,LinePrint	書式印刷機能を使用して，順編成ファイルをプリンタに出力する。

(3) リンクのコンパイラオプション

リンクを設定するコンパイラオプションのうち，Windows(x64) COBOL2002 では使用できないコンパイラオプションを次に示します。

Windows(x64) COBOL2002 では，呼び出し規約が fastcall に限定されます。ほかの呼び出し規約を指定することはできません。fastcall 呼び出し規約については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

表 18–4　Windows(x64) COBOL2002 で使用できないリンクのコンパイラオプション

コンパイラオプション	説明
-StdCall	stdcall 呼び出し指示ファイルを有効にする。
-StdCallFile .cbw ファイル名	stdcall 呼び出し指示ファイル名を指定する。

(4) 移行のコンパイラオプション

他システムへの移行を設定するコンパイラオプションのうち，Windows(x64) COBOL2002 では使用できないコンパイラオプションを次に示します。

表 18–5　Windows(x64) COBOL2002 で使用できない移行のコンパイラオプション

コンパイラオプション	説明
-CompatiM7	MIOS7 COBOL85 との互換機能を有効にする。

18.1.3 使用できない環境変数

ここでは，Windows(x64) COBOL2002 では使用できない実行時環境変数について説明します。

(1) ファイル

ファイルの実行時環境変数のうち，Windows(x64) COBOL2002 では使用できない実行時環境変数を次に示します。

表 18–6　Windows(x64) COBOL2002 で使用できないファイルの実行時環境変数

実行時環境変数	指定する内容
CBLX_外部装置名	書式印刷をするときの印刷サービス名称
CBL_BTRPGSZ	Btrieve による索引編成ファイルのページサイズ

18.1.4　実行可能ファイルと DLL ファイル

Windows(x64) COBOL2002 での実行可能ファイルと DLL の作成の詳細については，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

Windows(x86) COBOL2002 で作成した，DLL ファイルおよび実行可能ファイルの注意事項を次に示します。

注意事項

- Windows(x64) COBOL2002 で作成した実行可能ファイルまたは DLL ファイルから，Windows(x86) COBOL2002 で作成した DLL ファイルを呼び出すことはできません。
- Windows(x64) COBOL2002 で作成した実行可能ファイルまたは DLL ファイルから，Windows(x86) COBOL2002 で作成した実行可能ファイルを，CALL 文または CBLEXEC サービスルーチンを使用して，別プロセスとして起動できます※。

注※

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 がインストールされた環境で実行してください。

18.1.5　オブジェクトファイルとライブラリファイル

Windows(x64) COBOL2002 でのオブジェクトファイルとライブラリファイルの注意事項を次に示します。

注意事項

- Windows(x64) COBOL2002 で作成したオブジェクトファイルと，Windows(x86) COBOL2002 で作成したオブジェクトファイルおよびライブラリファイルはリンクできません。

18.1.6　入出力ファイル

Windows(x86) COBOL2002 で作成した順編成ファイル，テキスト編成ファイル，CSV 編成ファイル，相対編成ファイル，索引編成ファイルの入出力ファイルは，Windows(x64) COBOL2002 でそのまま使用できます。

ただし，次に示す項目をファイル節のレコード定義に指定した場合，ファイルの互換性はありません。必要に応じて再コンパイルおよびファイルを再作成してください。

- アドレスデータ項目
- ポインタ項目
- 指標データ項目

- 指標名

18.1.7　プログラム情報ファイル

Windows(x86) COBOL2002 で作成したプログラム情報ファイルは互換性がありません。再コンパイルして情報を蓄積し直してください。

18.1.8　リポジトリファイル

Windows(x86) COBOL2002 で作成したリポジトリファイルは互換性がありません。再コンパイルしてリポジトリファイルを生成し直してください。

18.2　Windows(x64) COBOL2002 開発マネージャの機能

ここでは，Windows(x86) COBOL2002 と Windows(x64) COBOL2002 について，相違を説明します。

なお，この節では，表示されるメッセージの半角かたかな,”､”を全角かたかな,”,”で表記します。表示されたメッセージをこのマニュアルで検索する場合は，メッセージを全角かたかな,”,”に変えてください。

18.2.1　表示の相違

ここでは，プラットフォームに依存する表示の相違について説明します。

- Windows(x64) COBOL2002 と Windows(x86) COBOL2002 での相違は，タイトルだけです。
- Windows のプログラム一覧にある［COBOL2002］下のメニューも相違があります。Windows(x64) COBOL2002 の場合は，このマニュアルの表記を「開発マネージャ for COBOL2002 64bit」に読み替えてください。

18.2.2　プロジェクトマスタファイルの相違

Windows(x64) COBOL2002 では，開発マネージャ自身が作成するプロジェクトマスタファイルのほかに，次のプロジェクトマスタファイルを開くことができます。

- Windows(x86) COBOL2002 で使用できるプロジェクトマスタファイル(.hmf)および COBOL プロジェクトファイル(.pmi)

Windows(x86) COBOL2002 で使用できるプロジェクトマスタファイル(.hmf)および COBOL プロジェクトファイル(.pmi)については，「1.4.2　プロジェクトマスタを開く」の「(1)　メニューから開く」を参照してください。

プロジェクトマスタファイルの移行については，「18.2.7　ユーザ資産の移行」の「(1)　プロジェクトマスタファイルの移行」を参照してください。

18.2.3　操作できるファイルの扱いの相違

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 では，次に示すファイルの扱いに相違があります。

扱いに相違のあるファイル一覧を次の表に示します。ファイルの移行については，「18.2.7　ユーザ資産の移行」の「(3)　ファイルの移行」を参照してください。

表 18–7　扱いに相違のあるファイル一覧

ファイル種別	拡張子	相違
ライブラリファイル	.lib	最終生成物が Windows(x64) COBOL2002 用になる。
実行可能プログラム	.exe	
ダイナミックリンクライブラリ	.dll	最終生成物が Windows(x64) COBOL2002 用になる。
stdcall 呼び出し指示ファイル	.cbw	登録できない。

18.2.4　ファイルメニューの相違

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 で相違のある次のファイルメニューの各項目について説明します。

- 新規作成
- 開く

(1)　新規作成メニューの相違

Windows(x64) COBOL2002 では，次に示すファイルは［新規作成］メニューには表示されません。

- stdcall 呼び出し指示ファイル

(2)　開くメニューの相違

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 で［開く］メニューに相違があります。

［開く］メニューで扱うファイルの相違を次の表に示します。ファイルの移行については，「18.2.7　ユーザ資産の移行」の「(3)　ファイルの移行」を参照してください。

表 18–8　開くメニューで扱うファイルの相違

ファイル種別	編集ツール	相違
stdcall 呼び出し指示ファイル※	COBOL エディタ	編集できない。

注※

一般ファイルとして扱われるため，編集対象にはなりません。

18.2.5　作成できるプログラムの相違

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 では作成できるプログラムに相違があります。Windows(x64) COBOL2002 で作成できるプログラムを次に示します。

- Windows(x64) COBOL2002 で作成できるプロジェクトの種類
  すべて，Windows(x64) COBOL2002 用になります。
  - System 指定メインプログラム
  - V3 指定メインプログラム
  - メインプログラムなし
  - 標準ライブラリ

注意事項

最終生成物の種類を指定するダイアログボックスでダイナミックリンクライブラリを選択した場合は，DLL の属性を選択する［DLL 呼び出し規約選択］ダイアログボックスは表示されません。出力ファイル名を指定するダイアログボックスが表示されます。

プロジェクトの移行については，「18.2.7　ユーザ資産の移行」の「(2)　プロジェクトの移行」を参照してください。

18.2.6　プロジェクト設定の相違

GUI 定義ファイルは一般ファイルとして扱われ，オプションの設定はできません。

(1)　プロジェクトの相違

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 では，プロジェクトの設定に相違があります。

最終生成物の種類を指定するダイアログボックスでダイナミックリンクライブラリを選択した場合は，DLL の属性を選択する［DLL 呼び出し規約選択］ダイアログボックスは表示されません。Windows(x64) COBOL2002 用の生成物を作成するプロジェクトになります。

プロジェクトの移行については，「18.2.7　ユーザ資産の移行」の「(2)　プロジェクトの移行」を参照してください。

(2)　Windows(x64) COBOL2002 だけで使用できるコンパイラオプション

Windows(x64) COBOL2002 だけで使用できるコンパイラオプションを次に示します。

表 18–9　Windows(x64) COBOL2002 だけで使用できるコンパイラオプション

プロジェクト設定の分類	使用できるコンパイラオプション	相違
その他	-MaxDigits38	表示する。

プロジェクト設定の分類	使用できるコンパイラオプション	相違
	（数字項目，数字編集項目，および数字定数の最大けた数を 38 けたとする）	
	-IntResult,DecFloat40 （中間結果を 40 けた 10 進浮動小数点形式で保持する）	

(3)　Windows(x64) COBOL2002 では使用できないコンパイラオプション

Windows(x64) COBOL2002 と Windows(x86) COBOL2002 では，コンパイラオプションに相違があります。Windows(x64) COBOL2002 では使用できないコンパイラオプションを次に示します。

コンパイラオプションの移行については，「18.2.7　ユーザ資産の移行」の「(4)　コンパイラオプションの移行」を参照してください。

表 18–10　Windows(x64) COBOL2002 で使用できないコンパイラオプション

プロジェクト設定の分類	使用できないコンパイラオプション	相違
製品連携	-IsamExtend （Btrieve（Pervasive.SQL）による索引編成ファイルを使用する）	表示しない。
	-IsamExtend,Zone （Btrieve（Pervasive.SQL）による索引編成ファイルを使用する場合，レコードキーに外部 10 進項目が指定されたときに，外部 10 進属性として処理する）	
	-XMAP （XMAP3 を使用する）	
	-XMAP,LinePrint （書式印刷機能を使用する）	
リンク	-StdCall （stdcall 呼び出し指示ファイルを有効にする）	
	-StdCallFile （stdcall 呼び出し規約の指示ファイルを指定する）	
移行	-CompatiM7 （MIOS7 COBOL85 との互換機能を有効にする）	

18.2.7　ユーザ資産の移行

Windows(x64) COBOL2002 でのユーザ資産の移行について，説明します。

(1) プロジェクトマスタファイルの移行

プロジェクトマスタファイルとプロジェクトマスタファイルに登録されている情報の移行について説明します。

- Windows(x86) COBOL2002 で扱うことができるプロジェクトマスタファイル（.hmf）および COBOL プロジェクトファイル（.pmi）は，Windows(x64) COBOL2002 に移行できます。
- COBOL85 Version 6，Version 7，Windows(x86) COBOL2002 で作成したプロジェクトマスタファイル(.hmf)を開いた場合には，次の警告メッセージが表示されます。

  「「開発マネージャ for COBOL2002 64bit」以外で作成されたプロジェクトマスタファイルです。」

  このメッセージが表示された場合に，このままプロジェクトマスタの保存を行うと，移行前のプロジェクト形式と異なるプロジェクト形式に変換されることがあります。移行前のプロジェクトマスタファイルのバックアップを採っていない場合は，プロジェクトマスタの保存を行わずに開発マネージャを終了し，バックアップを採ってから作業を行うことを推奨します。

(2) プロジェクトの移行

Windows(x64) COBOL2002 で開いたプロジェクトマスタファイル中に，使用できない機能を含んだプロジェクト種別が含まれていた場合，プロジェクト種別部分は不確定（プロジェクト種別の設定がない状態）のプロジェクトになります。

また，COBOL85 Version 6，Version 7，Windows(x86) COBOL2002 でエクスポートしたプロジェクト情報ファイル（.hmp）に，使用できない機能を含むプロジェクト種別が含まれていた場合，プロジェクト種別部分は不確定のプロジェクトになります。

プロジェクトの移行を次に示します。

表 18–11　プロジェクトの移行

プロジェクトで指定するオプション	移行時の扱い
-Dll 指定 （DLL を作成する）	移行できる。ただし，Windows(x64) COBOL2002 用の生成物を作成するプロジェクトとして動作する。
Dll の属性を stdcall または cdecl にする-Dll 指定	読み込み時に警告※1 を表示する。設定は解除される。
GUI プログラム	読み込み時に警告※2 を表示する。設定は解除され，不確定のプロジェクトとなる。

注※1

次のメッセージを表示します。

「このシステムでは使用できないオプションです。設定を無視します。」

注※2

次のメッセージを表示します。

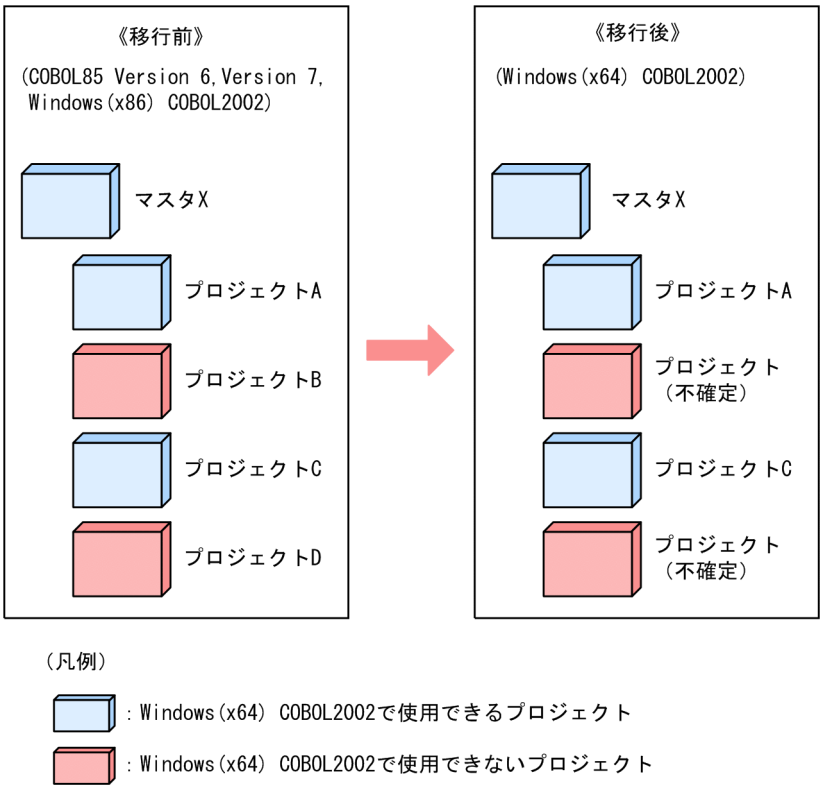
「必要な連携機能または，プログラムプロダクトがインストールされていないか，未サポート等により認識できません。以下のオプションは無効です。」

使用できない機能を含むプロジェクト種別としては，「18.2.5　作成できるプログラムの相違」に記載しているもののほかに，COBOL85 Version 6，Version 7 で作成した，次のプロジェクト種別があります。

- CORBA サーバプログラム（COBOL85）
- CORBA サーバプログラム（OOCOBOL）
- CORBA クライアントプログラム（EXE）
- CORBA クライアントプログラム（DLL）

使用できない機能を含んだプロジェクトマスタの移行例を次に示します。

図 18-1　使用できない機能を含むプロジェクトのプロジェクトマスタの移行例



使用できない機能を含むプロジェクトマスタは，Windows(x64) COBOL2002 で開かれた場合，不確定のプロジェクト（プロジェクト種別が選択されていない状態）になります。多くの場合，このプロジェクトマスタを使用することは難しいですが，次の対応によって利用できるものもあります。その際には，移行前のプロジェクトファイルを退避してから再利用してください。

DLL を作成するプロジェクトについては，呼び出し規約のサブオプションが無効になりますが，特に変更操作をすることなく移行できます。

- 不確定のプロジェクトを修正し，プロジェクトを再構築する。
- 不確定のプロジェクトを破棄し，プロジェクトを新規に作成する。
- 不確定のプロジェクトを破棄し，有効なプロジェクトだけを使用する。

(3)　ファイルの移行

Windows(x64) COBOL2002 で使用できないファイルの移行について説明します。使用できないファイル種別については，「18.2.3　操作できるファイルの扱いの相違」を参照してください。

Windows(x64) COBOL2002 開発マネージャへの移行時の扱いを次に示します。

表 18–12　Windows(x64) COBOL2002 開発マネージャへの移行時の扱い

ファイル種別	拡張子	移行時の扱い
stdcall 呼び出し指示ファイル	.cbw	読み込み時にエラーメッセージ※1 を表示し，プロジェクトから削除対象となる。
GUI 定義ファイル	.gcb	読み込み時に警告※2 を表示し，構成を変更して，一般ファイルとして扱う。
ブラウズ情報ファイル	.brd	読み込み時にエラーメッセージ※1 を表示し，プロジェクトから削除対象となる。

注※1

次のエラーメッセージを表示します。

「このシステムでは使用できないファイルです。ファイルをプロジェクトから削除します。」

注※2

移行の際には，「その他ファイル」に登録され，次のメッセージを表示します。

「必要な連携機能または，プログラムプロダクトがインストールされていないか，未サポート等により認識できないファイル種別です。一般ファイルとして登録します。」

(4)　コンパイラオプションの移行

Windows(x64) COBOL2002 で開いたプロジェクトマスタ中に，使用できないコンパイラオプションがある場合は次のように扱います。

- 使用できないツールや使用できない製品に依存するオプション含む場合は，該当するオプションを削除して移行します。そのほかの使用できないオプションは該当するオプションを削除して移行します。
- Windows(x64) COBOL2002 への移行によって，オプションが変更になったもの（例えば，-Dll,Cdecl オプションは-Dll オプションに変換）は，開発マネージャでもコンパイラのオプション変更に合わせたオプションの置き換えを行います。
- プロジェクトマスタファイル，プロジェクトファイルの読み込み時には，エラーメッセージを表示します。

コンパイラオプションの移行について次に示します。

表 18–13　コンパイラオプションの移行

プロジェクト設定の分類	コンパイラオプション	移行時の扱い
製品連携	-IsamExtend （Btrieve（Pervasive.SQL）による索引編成ファイルを使用する）	読み込み時にエラーメッセージ※を表示し，設定は解除する。
	-IsamExtend,Zone （Btrieve（Pervasive.SQL）による索引編成ファイルを使用する場合，レコードキーに外部 10 進項目が指定されたときに，外部 10 進属性として処理する）	
	-XMAP （XMAP3 を使用する）	
	-XMAP,LinePrint （書式印刷機能を使用する）	
リンク	-StdCall （stdcall 呼び出し指示ファイルを有効にする）	
	-StdCallFile （stdcall 呼び出し規約の指示ファイルを指定する）	
移行	-CompatiM7 （MIOS7 COBOL85 との互換機能を有効にする）	

注※

次のメッセージを表示します。

「このシステムでは使用できないオプションです。設定を無視します。」

ただし，関連製品がインストールされていないなどの場合，非表示となるオプションは次のメッセージを表示して無効とします。

- 関連製品がインストールされていないなどの場合

  「必要な連携機能または，プログラムプロダクトがインストールされていないか，未サポート等により認識できません。」

- オプションの場合

  「以下のオプションは無効です。」

なお，表示されるオプションに，開発マネージャ内部で使用しているオプション名称が表示されることがあります。表示された場合には，製品のインストール状況やオプションのサポート状況を確認してください。

使用できないオプションについては，「18.2.6　プロジェクト設定の相違」の「(3)　Windows(x64) COBOL2002 では使用できないコンパイラオプション」を参照してください。

18.3　Windows(x64) COBOL2002 の実行支援の機能

ここでは，Windows(x64) COBOL2002 で使用できない機能を説明します。

18.3.1　実行支援で使用できない機能

Windows(x64) COBOL2002 で使用できない実行時環境変数の扱いについて説明します。

(1)　実行時環境変数の表示

Windows(x64) COBOL2002 で使用できない実行時環境変数を実行支援で指定することはできません。使用できない実行時環境変数は「18.1.3　使用できない環境変数」を参照してください。

(2)　実行環境ファイルからの読み込み

Windows(x86) COBOL2002 または PC COBOL85 で作成した，Windows(x64) COBOL2002 では使用できない実行時環境変数の情報が含まれる実行環境ファイルを読み込んだ場合，実行支援では次のように扱います。

- 使用できない実行時環境変数の指定はそのまま，登録リストに表示します。
- 登録リストで使用できない実行時環境変数をダブルクリックした場合，［ユーザ設定］タブに実行時環境変数，および値を表示します。
- 環境情報の保存で，使用できない実行時環境変数が登録リストにあっても，実行環境ファイルに保存します。

なお，使用できない実行時環境変数を削除する場合は，削除したい実行時環境変数を登録リストから削除して，実行環境ファイルを保存してください。

付録

付録 A　Windows OS 固有の注意事項

Windows で COBOL2002 を使用するときの注意事項を次に示します。

付録 A.1　管理者権限についての注意事項

UAC 機能が有効な場合，管理者ユーザとしてサインインしても，アプリケーションの実行権限は標準権限に下げられます。例えば，管理者権限を必要とするアプリケーションを起動しようとすると，システムは，昇格ダイアログボックスを表示して，ユーザの確認を求めます。この場合，ユーザが実行を許可すると，アプリケーションは管理者権限で実行されます。または，管理者権限で実行するためにプログラムやコマンドのプロパティを変更するか，「管理者として実行」で実行するなどの操作が必要です。

管理者権限についての注意事項を次に示します。

(1)　プログラムを管理者権限で実行する場合の注意事項

管理者権限でプログラムを実行する場合，起動時に Windows のユーザアカウント制御（UAC）の昇格ダイアログボックスが表示されます。また，ユーザが作成したバッチファイル内など，呼び出し元で設定した環境変数は，呼び出し先の管理者権限で動作するプロセスには引き継がれません。

これらを回避する場合は，呼び出し元のプログラムを管理者権限で実行してください。その際，次の点に注意してください。

- 呼び出し元から呼ばれるすべてのプログラムは管理者権限で動作することになります。セキュリティの面で問題がないかどうか確認してください。
- 標準権限のコマンドプロンプトから管理者権限のプログラムを起動した場合，管理者権限のプログラムから標準権限のコマンドプロンプトで設定した環境変数にアクセスできません。コマンドプロンプトも管理者権限で起動してください。バッチファイルを使用する場合は，特に注意してください。
- 開発マネージャや COBOL エディタなどを管理者権限で実行させている場合，標準権限で動作するエクスプローラからファイルをドラッグ＆ドロップできません。

(2)　テストデバッガおよびカバレージ使用時の注意事項

- テストデバッガおよびカバレージで，次に示す操作を標準権限（Administrators グループのユーザ含む）で実行する場合，注意が必要です。
  - 連動実行のテストデバッガでデバッグする場合
  - 管理者権限を設定したプログラムをデバッグ対象にする場合

  次に示す方法で実行してください。
  - 連動実行のテストデバッガでデバッグする場合

Administrators グループのユーザの場合は，テストデバッガおよびデバッグ対象プログラムを管理者権限で実行してください。

それ以外のユーザの場合は，テストデバッガおよびデバッグ対象プログラムを，管理者権限で実行するか，または「グローバルオブジェクトの作成」権限を追加してから標準権限で実行してください。

「グローバルオブジェクトの作成」権限は，コントロールパネルの［管理ツール］から［ローカルセキュリティポリシー］を使用して，実行するユーザに追加してください。

- 管理者権限を設定したプログラムをデバッグ対象にする場合

  テストデバッガおよびカバレージを管理者権限で実行してください。

  テストデバッガおよびカバレージを開発マネージャから呼び出す場合は，開発マネージャを管理者権限で実行してください。テストデバッガ，カバレージまたは開発マネージャを標準権限（Administrators グループのユーザ含む）で実行すると，デバッグ対象プログラムの起動に失敗します。

- テストデバッガおよびカバレージを管理者権限で実行した場合は，デバッグ対象プログラムも管理者権限で動作します。標準権限で実行した場合は，デバッグ対象プログラムも標準権限で動作します。しかし，テストデバッガおよびカバレージを標準権限で実行した場合は，Windows リソース保護（WRP）の影響を受けるため，管理者権限で実行したときと実行結果が変わることがあります。そのため，運用時と同じ権限でテストデバッガおよびカバレージを実行してください。

(3) ISAM による索引編成ファイル使用時の注意事項

プログラムで ISAM による索引編成ファイルを使用する場合は，管理者権限で実行してください。管理者権限がない場合，ISAM による索引編成ファイルにアクセスすると実行時エラーになります。

(4) ISAM ユティリティに関する注意事項

使用する ISAM ユティリティによって，管理者権限での実行が必要かどうかが変わります。ISAM ユティリティと管理者権限について説明します。

(a) 管理者権限での実行が必要な ISAM ユティリティ

管理者権限での実行が必要な ISAM ユティリティを次に示します。

Windows(x86) COBOL2002 の場合

- オプション情報の設定ユティリティ
- islckext コマンド
- islckclear コマンド
- islckclearL コマンド

Windows(x64) COBOL2002 の場合

- オプション情報の設定ユティリティ
- islckext64 コマンド

- islckclear64 コマンド

(b) 原則，管理者権限での実行が必要な ISAM ユティリティ

原則，管理者権限での実行が必要な ISAM ユティリティを次に示します。

Windows(x86) COBOL2002 の場合

- 定義ファイルの作成ユティリティ
- レコード内容の表示ユティリティ
- iscpy コマンド
- isers コマンド
- isren コマンド
- isprt コマンド

Windows(x64) COBOL2002 の場合

- 定義ファイルの作成ユティリティ
- レコード内容の表示ユティリティ
- iscpy64 コマンド
- isers64 コマンド
- isren64 コマンド
- isprt64 コマンド

ただし，セキュリティ上の問題で，管理者権限で実行できない場合は，コントロールパネルの［管理ツール］から［ローカルセキュリティポリシー］を使用して，標準権限にグローバルオブジェクトの作成権限を追加したユーザで実行してください。

グローバルオブジェクトの作成権限がないユーザで，これらの ISAM ユティリティを標準権限で実行した場合，定義ファイルの作成ユティリティ以外の ISAM ユティリティはエラーとなります。表示されるエラーメッセージを次の表に示します。

項番	ISAM ユティリティ	エラーメッセージ
1	• レコード内容の表示ユティリティ • isprt コマンド	KAIU001-E ＩＳＡＭアクセス関数エラーが発生しました。(isopen)
2	• iscpy コマンド • isers コマンド	KAIU001-E ＩＳＡＭアクセス関数エラーが発生しました。(normal,5)
3	isren コマンド	KAIU001-E ＩＳＡＭアクセス関数エラーが発生しました。(isrename,1010)

注意

これらのメッセージは，ISAM ファイルにアクセス権がない場合にも表示されます。

また，コマンドプロンプトでもエラーの原因を確認できます。次に示すコマンドを実行し，表示される関数エラー情報を確認してください。

Windows(x86) COBOL2002 の場合

```
ismdmpntコマンド（ラージファイル形式の場合ismdmpntLコマンド）
```

Windows(x64) COBOL2002 の場合

```
ismdmpnt64コマンド
```

該当する時間に iserrno=ESYSERR,issyserr=5,syserrknd=13 のエラー情報が表示されている場合，グローバルオブジェクトの作成権限がないユーザで，これらの ISAM ユティリティを標準権限で実行したことを示します。

ismdmpnt コマンドまたは ismdmpnt64 コマンドの表示例

```
[01]13/10/02; 11:22:42; procno = 2760; thrdno = 2392（ISAMファイル名）
(ISAM関数): iserrno=ESYSERR,iserrcd = 00000113 ,issyserr = 00000005,syserrknd = 00000013
```

なお，定義ファイルの作成ユティリティは，ISAM ファイルの作成先にすでに ISAM ファイルが存在する場合，アクセス権などの属性情報が引き継がれます。

(c) 標準権限で実行できる ISAM ユティリティ

標準権限で実行できる ISAM ユティリティを次に示します。

Windows(x86) COBOL2002 の場合

- キーの追加／削除／再構築ユティリティ
- ファイルの検証ユティリティ
- ファイルの圧縮ユティリティ
- ファイルの変換ユティリティ
- ファイルの抽出ユティリティ
- ファイルの移行ユティリティ
- キー定義情報の表示ユティリティ
- iskeydef コマンド
- iskeymnt コマンド
- isconv コマンド
- ischk コマンド
- isext コマンド
- isinfo コマンド
- iscond コマンド

- isshift コマンド
- isgendef コマンド

Windows(x64) COBOL2002 の場合

- キーの追加／削除／再構築ユティリティ
- ファイルの検証ユティリティ
- ファイルの圧縮ユティリティ
- ファイルの変換ユティリティ
- ファイルの抽出ユティリティ
- ファイルの移行ユティリティ
- キー定義情報の表示ユティリティ
- iskeydef64 コマンド
- iskeymnt64 コマンド
- isconv64 コマンド
- ischk64 コマンド
- isext64 コマンド
- isinfo64 コマンド
- iscond64 コマンド
- isshift64 コマンド

なお，ファイル／レコード定義は，ISAM ファイルを生成する際に ISAM ユティリティを使用しますが，ISAM ファイルは標準権限で生成できるため，ファイル／レコード定義は，標準権限，管理者権限のどちらでも起動できます。

(d)　ISAM ユティリティを管理者権限だけで使用できるようにする方法

ISAM ユティリティを<ISAM のインストールフォルダ>※¥Bin¥Compat フォルダに格納されている ISAM ユティリティと置き換えることで，ISAM ユティリティを管理者権限だけで使用できます。なお，置き換え前の ISAM ユティリティに戻すことができるように，作業前に置き換え前の ISAM ユティリティをバックアップしておくことを推奨します。

ISAM ユティリティを置き換える手順を次に示します。

1. 管理者権限のあるユーザであることを確認する。
2. コマンドプロンプトを管理者権限に権限昇格して起動する。
3. <ISAM のインストールフォルダ>※¥Bin¥Compat フォルダに移動する。

```
cd <ISAMのインストールフォルダ>※¥Bin¥Compat
```

4. ISAM ユティリティを実行していないことを確認する。

5. xcopy コマンドで Compat フォルダ内の exe ファイルを直上の Bin フォルダへコピーする。

```
xcopy *.exe ..¥ /y
```

6. サインアウトする。

注※

ISAM のインストールフォルダは，デフォルトでは次のとおりです。

32bit 版：

```
C:¥Program Files¥Hitachi¥ISAM
```

64bit 版：

```
C:¥Program Files¥Hitachi¥ISAM64
```

付録 A.2　ファイル格納先についての注意事項

(1)　ファイルダイアログボックスで格納する場合

開発マネージャ，COBOL エディタ，実行支援などで編集したファイルを格納する場合，インストール直後の状態では，ファイルダイアログボックスで表示される格納先は COBOL2002 インストールフォルダに位置付けられています。そのまま格納しようとすると Windows リソース保護（WRP）によって失敗※しますので，別の場所を指定してください。Windows リソース保護（WRP）については，各 OS のヘルプなどを参照してください。

注※

管理者権限で実行した場合を除きます。

付録 A.3　文字コードについての注意事項

COBOL2002 のコンポーネント（コンパイラ，実行時ライブラリ，開発ツール）や COBOL2002 または COBOL85 で作成したプログラムで使用するフォルダ名，ファイル名，プログラムへの入力文字列，および環境変数に指定できる文字は，シフト JIS の範囲だけです。JIS X0213 の第 3 水準漢字，および第 4 水準漢字を含む Unicode の文字は使用できません。

第 3 水準漢字，および第 4 水準漢字の文字を含んだフォルダやファイル名などを入力した場合，システムが第 3 水準漢字，および第 4 水準漢字の文字を”?”などに変換し，意図しない値になりますので，注意してください。

付録 A.4　実行可能ファイルの実行権限についての注意事項

Windows(x86) COBOL2002 02-00 以降または Windows(x64) COBOL2002 で作成したアプリケーションは，標準権限，管理者権限のどちらでも実行できます。

ただし，アプリケーションを実行する場合の実行権限について，次の注意事項があります。

- COBOL プログラムから CALL 文または CBLEXEC サービスルーチンで実行可能ファイルを実行する場合は，呼び出し元と同じ権限で実行します。例えば，標準権限のアプリケーションから管理者権限のアプリケーション（実行可能ファイル）を CALL 文で呼び出した場合，KCCC0137R-S のメッセージが出力され，システムから返されたエラー番号に 740（管理者権限のないユーザから呼び出された）が示されます。このとき，プログラム互換性アシスタント（PCA）に登録され，再度実行した場合はエラーにならない場合があります※。ただし，プログラム互換性アシスタント（PCA）に登録された標準権限のアプリケーションから管理者権限のアプリケーションの呼び出しを COBOL2002 では保証しません。
- ISAM による索引ファイル入出力機能を使用する場合，管理者権限で実行してください。セキュリティ上の問題で管理者権限で実行できない場合は，コントロールパネルの［管理ツール］から［ローカルセキュリティポリシー］を使用して，標準権限にグローバルオブジェクトの作成権限を追加したユーザで実行してください。

注※

プログラム互換性アシスタント（PCA）が自動的に回避策を適用する機能は無効となっていて，継続的にエラーとなることがあります。

アプリケーションの権限の設定方法，およびプログラム互換性アシスタント（PCA）については，各 OS のヘルプなどを参照してください。

管理者権限を設定する例

実行可能ファイルのリンク時，-MANIFESTUAC リンカオプションで実行可能ファイルのアクセス許可レベルの値を指定します。

```
ccbl2002 -Main,System test01.cbl -Link -MANIFESTUAC:level='requireAdministrator'
```

付録 A.5　実行時の注意事項

実行時の注意事項を次に示します。

- Windows(x86) COBOL2002 02-00 以降または Windows(x64) COBOL2002 で作成したアプリケーションは，デフォルトで内部マニフェストを埋め込んでいます。

  外部マニフェストファイル※よりも内部マニフェストの情報が優先されるため，内部マニフェストを埋め込んだアプリケーションに対して，外部マニフェストファイルの設定内容は有効となりません。

注※

実行可能ファイル名の後ろに「.manifest」を付加したファイルで，権限属性などを定義できます。

- Windows(x86) COBOL2002 02-00 以降または Windows(x64) COBOL2002 で作成したアプリケーションを，「プロパティ」ダイアログボックスで実行権限を設定したあと，別のフォルダに移動した場合は，再度実行可能ファイルの「プロパティ」ダイアログボックスで，管理者権限を設定する必要があります。詳細は，各 OS のヘルプなどを参照してください。
- アプリケーションでファイル入出力機能などを使用する場合，Windows リソース保護（WRP）によって，Windows リソース（OS ファイル，フォルダなど）が保護されるので，ファイルの出力先の指定に注意が必要です。Windows リソース保護（WRP）については，各 OS のヘルプなどを参照してください。
- リンカのインストール先フォルダについては，マニュアル「COBOL2002　ユーザーズガイド」を参照してください。

付録 A.6　開発マネージャの使用上の注意事項

開発マネージャの使用上の注意事項を次に示します。

なお，この節では，表示されるメッセージの半角かたかなを全角かたかなで表記します。表示されたメッセージをこのマニュアルで検索する場合は，メッセージを全角かたかなに変えてください。

(1)　パッケージ情報ファイル（.pkg）の移行

開発マネージャを使用する前に，パッケージ情報ファイルを移行してください。

移行方法

＜移行元フォルダ＞のパッケージ情報ファイルを，＜移行先フォルダ＞にエクスプローラなどでコピーしてください。

同一マシンに，COBOL2002 をインストールする場合は，あらかじめ，パッケージ情報ファイルを退避し，COBOL2002 をインストールしたあとで，＜移行先フォルダ＞にエクスプローラなどでコピーしてください。

＜移行元フォルダ＞

次のどれかのフォルダ

- インストールフォルダ¥PLUGIN
- 環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002¥PLUGIN
- 環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002 64bit¥PLUGIN

＜移行先フォルダ＞

- Windows(x86) COBOL2002 の場合

環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002¥PLUGIN

- Windows(x64) COBOL2002 の場合
  環境変数 ALLUSERSPROFILE が示すフォルダ¥Hitachi¥COBOL2002 64bit¥PLUGIN

(2) 旧バージョンで作成したプロジェクトマスタファイルを移行した場合の注意事項

サポートしていない機能の情報が含まれているプロジェクトマスタファイルを開いた場合，次に示す動作になります。

- 該当するプロジェクトは，プロジェクト種別部分が不確定（プロジェクト種別の設定がない状態）のプロジェクトになります。プロジェクトのインポートについても同様に，プロジェクト種別部分が不確定のプロジェクトになります。
- プロジェクトに使用できないファイルが含まれている場合，次に示すメッセージを表示し，ファイルはツリー上の「その他ファイル」フォルダに格納されます。
  「このシステムでは使用できないファイルです。一般ファイルとして登録します。」
- プロジェクトに使用できないオプションが含まれている場合，次に示すメッセージを表示し，設定を無視します。
  「このシステムでは使用できないオプションです。設定を無視します。」

(3) 開発マネージャを標準権限で起動した場合の注意事項

開発マネージャを標準権限で起動した場合の注意事項を次に示します。

- 開発マネージャからテストデバッガやカバレージを起動する場合，昇格ダイアログボックスが表示されます。この場合，必要な情報を入力して操作を続行できます。昇格ダイアログボックスでキャンセルを選択した場合，メッセージ※1 が表示されます。
- 開発マネージャから管理者権限のユーザプログラムを起動する場合，昇格ダイアログボックスが表示されます。この場合，必要な情報を入力して操作を続行できます。昇格ダイアログボックスでキャンセルを選択した場合，メッセージ※1 が表示されます。
- 開発マネージャから起動するツール※2 が管理者権限であった場合は起動できません。メッセージ※1 が表示されます。
  このメッセージが表示されたとき，起動に失敗したツールの権限が管理者権限であるかを見直し，管理者権限であった場合は，開発マネージャも管理者権限で起動してください。ツールの権限が管理者権限でなかった場合は，ツールの存在有無など環境について確認してください。
  なお，ビルド処理で起動に失敗した場合，ビルドは中断します。
- cblbuild2k コマンドから起動するツール※2 が管理者権限であった場合は起動できません。メッセージ※1 が表示されます。
  このメッセージが表示されたとき，起動に失敗したツールの権限が管理者権限であるかを見直し，管理者権限であった場合は，cblbuild2k コマンドを管理者権限で起動するか，管理者権限で起動した

「COBOL コマンドプロンプト for COBOL2002」で cblbuild2k コマンドを実行してください。管理者権限でなかった場合は，ツールの存在有無など環境について確認してください。

なお，起動に失敗した場合，ビルドは中断し，リターンコード 1 を返します。

- 一般ファイルとして登録したファイルに関連づいているツールが管理者権限の場合，一般ファイルを開く際に昇格ダイアログボックスが表示されます。この場合，必要な情報を入力して操作を続行できます。昇格ダイアログボックスでキャンセルを選択した場合，メッセージ※1 が表示されます。

注※1

表示されるメッセージは，次のとおりです。

- ユーザプログラムの起動に失敗した場合

  「実行に失敗しました。

  "xxx"を起動できません。」

- それ以外の場合

  「xxx の起動に失敗しました。」

xxx：開発マネージャや cblbuild2k コマンドから起動されるツール名またはユーザプログラム名が表示されます。

注※2

開発マネージャや cblbuild2k コマンドから起動されるツールを次に示します。

- テストデバッガやカバレージを除く，開発マネージャから呼び出される COBOL2002 のコンポーネント
- 「新しいファイルの種類の登録」で登録した任意のツール
- エディタの設定ダイアログボックスで指定した任意のツール
- 開発マネージャから呼び出される関連製品のツール

付録 B　メニュー項目一覧

開発マネージャウィンドウ，実行支援ウィンドウ，テストデバッグウィンドウ，カバレージウィンドウ，画面定義ウィンドウ，ファイル／レコード定義ウィンドウ，および COBOL エディタウィンドウのメニューの一覧を次に示します。

付録 B.1　開発マネージャウィンドウのメニュー項目一覧

開発マネージャウィンドウのメニュー項目一覧を次に示します。

表 B-1　開発マネージャウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
プロジェクトマスタ(M)	新規作成(N)	プロジェクトマスタを新規に作成する。
	開く(O)	編集やコンパイルの対象となるプロジェクトマスタを開く。
	閉じる(C)	編集中のプロジェクトマスタを閉じる。
	保存(S)	開発マネージャで編集中のプロジェクトマスタ情報を，プロジェクトマスタファイルに保存する。
	デフォルトオプションの設定(D)※1	プロジェクトを作成したときに仮定されるオプションを設定する。
	プロパティ(P)	プロジェクトマスタの情報を表示する。
	最新のファイル（最新のプロジェクトマスタファイル名が４個まで表示される）	指定したプロジェクトマスタを開く。
	開発マネージャの終了(X)	開発マネージャを終了し，ウィンドウを閉じる。
ファイル(F)	新規作成(N)	プロジェクトに登録するファイルを新規に作成する。
	開く(O)	ツリービューウィンドウで選択しているファイルを，関連づけられたツールで開く。
	削除(D)	ツリービューウィンドウで選択しているファイルを削除する。
	チェックアウト(T)※2	構成管理ツールと連携して，ファイルをチェックアウトする。
	チェックイン(I)※2	構成管理ツールと連携して，ファイルをチェックインする。
	最新を取得(G)※2	構成管理ツールと連携して，最新のファイルを取得する。
	チェックアウト状態の変更(H)※2	構成管理ツールと連携して，チェックアウトされていない場合は，チェックアウト状態にする。また，チェックアウトされている場合は，チェックアウトされていない状態にする。
	ファイルの設定(S)	コンパイラオプション，およびコンパイラ環境変数を，ファイルごとに設定する。

メニューバー	メニュー項目	機能
	出力ファイルの編集(E)	ツリービューウィンドウで選択しているファイルの出力ファイルを編集する。
	出力ファイルの変更(U)	定義ファイルをコンパイルしたときに出力先となるファイルを指定する。
	外部ツールを操作するメニュー	外部ツールとの連携で登録されたメニューを表示する。 COBOL2002 Professional Tool Kit がインストールされていない，または［外部ツールの設定］ダイアログボックスにメニュー名が指定されていない場合は，このメニューは表示されない。
	プロパティ(P)	ツリービューウィンドウで選択しているファイルの情報を表示する。
プロジェクト(P)※3	プロジェクトの作成(A)	プロジェクトを新規に作成し，編集中のプロジェクトマスタに追加する。
	プロジェクト名の変更(N)	プロジェクトの名称を変更する。
	プロジェクトのエクスポート(X)	プロジェクトの情報をファイルに出力する。
	プロジェクトのインポート(I)	プロジェクトの情報をファイルから取り込む。
	作業中プロジェクトの設定(W)※4	作業中プロジェクトを切り替える。
	プロジェクトの削除(E)※4	プロジェクトを削除する。
	プロジェクトの関係設定(R)※4	プロジェクトの親子関係を定義する。
	ソースファイルの追加(F)	［ファイルの追加(C)］： プロジェクトにファイルを追加する。 ［フォルダ指定によるファイルの追加(D)］： フォルダを指定して，プロジェクトにファイルを追加する。
	プロジェクトの設定(S)	コンパイラオプション，およびコンパイラ環境変数を，プロジェクトごとに設定する。
	依存ファイルの設定(D)	ソースファイルに対する依存ファイルを，依存ファイルの仮登録から選ぶ。
	依存ファイルの仮登録(T)	依存ファイルをプロジェクトマスタに登録する。
	カバレージ(O)	カバレージを起動する。
	プロパティ(P)	プロジェクトの情報を表示する。
ビルド(B)※3	ビルド(B)	作業中プロジェクト，およびその子プロジェクトに属するソースファイルをコンパイル，リンケージする。また，定義ファイルからはソースを生成する。このとき，コンパイルの不要なプログラムや生成の不要な定義ファイルはビルドの対象としない。 オブジェクト指向機能または利用者定義関数を使用したプログラムのプロジェクトの場合，ファイルの依存関係を再構築する。

メニューバー	メニュー項目	機能
	クイックビルド(Q)	ビルドと同じ処理をする。ただし，オブジェクト指向機能または利用者定義関数を使用したプロジェクトの場合，ファイルの依存関係を再構築しない。
	コンパイル(C)	ツリービューウィンドウでソースファイルを選択している場合，そのソースファイルをコンパイルする。 ツリービューウィンドウで定義ファイルを選択している場合，その定義ファイルからソースを生成する。
	リビルド(R)	ビルドと同様にプロジェクトに登録したファイルをすべてコンパイル，リンケージする。 ビルドと異なり，プログラムやコンパイラオプションなどに変更がなくても，無条件にすべてのファイルをコンパイル，リンケージし，最終生成ファイルを出力する。
	すべてのプロジェクトに対する操作(P)※4	[ビルド(B)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，ビルドを実行する。 [クイックビルド(Q)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，クイックビルドを実行する。 [リビルド(R)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，リビルドを実行する。 [依存関係更新(A)]： プロジェクトマスタに登録されているすべてのプロジェクトに対して，依存関係の更新を実行する。
	ビルドの中止(S)	ビルド，クイックビルド，またはリビルドを中止する。
	依存関係更新(A)	作業中プロジェクト，およびその子プロジェクトに登録されているすべてのソースファイルについて，依存関係を更新する。
	デバッガ(D)	テストデバッガを起動する。
	実行(E)	実行可能ファイルを，実行環境を設定しないでそのまま実行する。
	実行支援(L)	実行支援を起動する。
ツール(T)※3	ODBC 定義(D)	ODBC レコード定義生成を起動する。
	開発マネージャの動作オプション(O)	開発マネージャの動作，およびツールバーの表示を設定する。
	開発マネージャのユーザカスタマイズ(C)	[ユーザインタフェース(U)]： キー操作，およびメッセージウィンドウの文字フォントや色を設定する。 [エディタ(E)]： テキストファイルを編集するエディタを設定する。

メニューバー	メニュー項目	機能
		［新しいファイルの種類の登録(F)］※1： 新しいファイルの種類を登録する。 ［構成管理ツールの設定(C)］※1 ※2： 構成管理ツールの操作方法を登録する。 ［外部ツールの設定(X)］※1 ※2： 外部ツールを登録する。
	プロジェクトマスタ情報の入出力(P)	［CSV 入力(C)］： CSV ファイルを入力する。 ［CSV 出力(S)］： CSV ファイルを出力する。
	COBOL ソース解析(A)※2	COBOL2002 Developer Professional の COBOL ソース解析の機能で使用するメニュー。※5 ［ソース解析プロジェクトの使用開始(T)］： ソース解析プロジェクト名を設定し，ソース解析プロジェクトを開始する。 ソース解析プロジェクトが使用中の場合，メニュー名の後ろに，使用中のソース解析プロジェクト名を表示する。 ［ソース解析プロジェクトの使用終了(F)］： 使用中のソース解析プロジェクトを終了する。 ［ソース解析プロジェクトの削除(D)］： 使用中のソース解析プロジェクトを削除する。 ［解析の実行(E)］： COBOL ソースファイルの解析を実行する。 ［解析の中止(P)］： 実行中の COBOL ソースファイルの解析を中止する。 ［設定(S)］： ［解析生成オプション設定］画面を表示する。 ［解析結果(HTML)の表示(H)］： HTML 形式の解析結果のプロジェクトサマリ画面（index.html）を表示する。 ［解析結果(CSV)の表示(C)］： CSV 形式の解析結果を格納したフォルダを表示する。
	データ影響波及分析(I)※2	COBOL2002 Developer Professional のデータ影響波及分析の機能で使用するメニュー。※5 ［設定(S)］： ［設定］ダイアログボックスを表示する。 ［データ影響波及分析用 DB 作成(D)］： 開発マネージャで開いているプロジェクトのデータ影響波及分析用データベースを作成する。

メニューバー	メニュー項目	機能
		[すべてのプロジェクトの DB 作成(A)]： プロジェクトマスタにあるすべてのプロジェクトのデータ影響波及分析用データベースを作成する。
	単体テスト支援(T)※2	COBOL2002 Developer Professional の単体テスト支援の機能で使用するメニュー。※5 [単体テスト支援(N)]： 選択した COBOL ソースファイルの単体テスト支援を実行する。
ウィンドウ(W)	新しいウィンドウを開く(N)	ツリービューウィンドウを新しいウィンドウに表示する。
	重ねて表示(C)	ツリービューウィンドウ，およびメッセージウィンドウを重ねて表示する。
	並べて表示(T)	ツリービューウィンドウ，およびメッセージウィンドウを並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	メッセージウィンドウ(M)	メッセージウィンドウを表示する。
	ステータスバー(S)	ステータスバーの表示／非表示を切り替える。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※1

プロジェクトマスタを閉じている状態のときだけ表示されるコマンドです。

注※2

COBOL2002 Developer Professional でだけ表示されるメニュー項目です。

注※3

開発マネージャに登録されているツールがほかにある場合は，メニュー項目が追加されます。

注※4

編集中のプロジェクトマスタに複数のプロジェクトが登録されている場合に使用できます。

注※5

使用方法については，次に示すマニュアルを参照してください。

「COBOL2002 Professional Tool Kit COBOL ソース解析ガイド」

「COBOL2002 Professional Tool Kit データ影響波及分析ガイド」

「COBOL2002 Professional Tool Kit 単体テスト支援ガイド」

付録 B.2　実行支援ウィンドウのメニュー項目一覧

実行支援ウィンドウのメニュー項目一覧を次に示します。

表 B-2　実行支援ウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	新しく実行環境ファイルを作成する。

メニューバー	メニュー項目	機能
	開く(O)	既存の実行環境ファイルを開く。
	上書き保存(S)	設定した環境情報を作業中の実行環境ファイルに保存する。
	名前を付けて保存(A)	設定した環境情報を実行環境ファイルの名前を付けて保存する。
	終了(X)	実行支援ウィンドウを閉じる。
設定(E)	画面環境(S)	画面機能の実行環境を設定する。
	印刷書式(P)	印刷時の書式や出力先のプリンタを設定する。 [デフォルト(D)]： デフォルトとして使用する印刷書式を設定する。 [システム(S)]： システムで使用する印刷書式を設定する。
	オプション(O)	実行環境ファイルで設定した画面環境，および印刷書式（デフォルト）を，実行時に無効にする。
実行(R)	現在の内容で実行>(R)	実行環境設定中の実行可能ファイルを実行する。
	他の実行可能ファイル(P)	実行環境設定中の実行可能ファイル以外の実行可能ファイルを実行する。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

付録 B.3　テストデバッグウィンドウのメニュー項目一覧

テストデバッグウィンドウのメニュー項目一覧を次に示します。

表 B-3　テストデバッグウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
テストデバッガの制御(F)	テストデバッグ対象の設定(T)	テストデバッグ対象プログラムを指定する。
	TD コマンド格納ファイルの入力(I)	TD コマンド格納ファイルの内容を実行する。
	初期化(N)	テスト環境を初期状態にする。
	テストデバッガの終了(X)	テストデバッガを終了する。
編集/表示(E)	切り取り(T)	文字列を切り取る。
	コピー(C)	文字列をクリップボードにコピーする。
	貼り付け(P)	文字列をクリップボードからカーソル位置に貼り付ける。
	すべて選択(A)	すべての文字列を選択する。

メニューバー	メニュー項目	機能
	クリア(L)	すべての文字列を消去する。
	ソースファイルの編集(E)	テキストエディタを起動し，ソースファイルを表示する。
	文字列の検索(F)	文字列を検索する。
	指定行の表示(O)	指定した行を表示する。
	現在位置の表示(R)	現在の中断個所の実行文を表示する。
	バックトレース(B)	[進む(F)]： バックトレース位置を一つ先に進める。 [戻る(B)]： バックトレース位置を一つ前に戻す。 [先頭(T)]： 蓄積されているバックトレース情報の中で，最初にプログラム実行された文にバックトレース位置を表示する。 [終了(E)]： バックトレース位置の表示を終了する。
プログラムの制御(C)	連続実行(G)	連続実行をする。
	ステップイン(S)	呼び出すプログラムを対象として 1 ステップ実行する。
	ステップオーバー(T)	呼び出すプログラムを対象外として 1 ステップ実行する。
	連動実行の待機(W)	連動実行されるまで待機状態にする。
	プログラムの実行を終了(E)	プログラムの実行を終了する。
中断(B)	中断点の設定/解除(B)	中断点の設定/解除を行う。
	中断点の一覧(P)	中断点の一覧を表示する。
	すべての中断点の解除(R)	すべての中断点を解除する。
	割り込み中断(I)	実行を中断する。
データの操作(D)	データ値の表示(D)	データ名の値を表示する。
	データ値の代入(V)	データ名に値を代入する。
	常時データ表示に設定(T)	常時データ表示ウィンドウにデータ名を設定する。
	データ監視条件の設定(S)	データ監視条件を設定する。

メニューバー	メニュー項目	機能
	データ監視条件の一覧(W)	データ監視条件の一覧を表示する。
ツール(T)	オプション(O)	各種オプションを設定する。
	カスタマイズ(U)	ウィンドウの文字フォントや色を設定する。
ウィンドウ(W)※	重ねて表示(C)	ウィンドウを重ねて表示する。
	並べて表示(T)	ウィンドウを並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	標準配置(S)	ウィンドウを標準位置に表示する。
	ソーステキストウィンドウ(R)	[呼び出し位置の表示(C)]： ソーステキストの呼び出し位置を表示する。 [定義部の表示(F)]： ソーステキストの定義部を表示する。 [実行部の表示(M)]： ソーステキストの実行部を表示する。 [すべて閉じる(W)]： 開いているソーステキストウィンドウをすべて閉じる。
	テストデバッグ一覧ウィンドウ(U)	テストデバッグ一覧ウィンドウを表示する。
	呼び出し一覧ウィンドウ(K)	呼び出し一覧ウィンドウを表示する。
	常時データ表示ウィンドウ(D)	常時データ表示ウィンドウを表示する。
	モニタウィンドウ(M)	モニタウィンドウを表示する。
	TD コマンド入力ウィンドウ(I)	TD コマンド入力ウィンドウを表示する。
	ステータスバー(B)	ステータスバーの表示／非表示を切り替える。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※

ウィンドウとは，ソーステキストウィンドウ，呼び出し一覧ウィンドウ，常時データ表示ウィンドウ，TD コマンド入力ウィンドウ，モニタウィンドウのことをいいます。

付録 B.4　カバレージウィンドウのメニュー項目一覧

カバレージウィンドウのメニュー項目一覧を次に示します。

表 B-4　カバレージウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
カバレージの制御(T)	カバレージの終了(X)	カバレージを終了する。
カバレージ情報(C)	蓄積(A)	カバレージ情報を蓄積する。
	表示(D)	カバレージ情報を表示する。
	マージ(M)	カバレージ情報をマージする。
	操作(T)	カバレージ情報を操作する。
カウント情報(N)	表示(D)	カウント情報を表示する。
ヘルプ(H)	バージョン情報(A)...	バージョン情報を表示する。

付録 B.5　画面定義ウィンドウのメニュー項目一覧

画面定義ウィンドウのメニュー項目一覧を次に示します。

表 B-5　画面定義ウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	新しく画面定義ファイルを作成する。
	開く(O)	既存の画面定義ファイルを開く。
	閉じる(C)	既存の画面定義ファイルを閉じる。
	上書き保存(S)	設定した画面情報を作業中の画面定義ファイルに保存する。
	名前を付けて保存(A)	設定した画面情報を画面定義ファイルの名前を付けて保存する。
	COBOL 生成(G)	COBOL ソースファイルを生成する。
	ファイル情報(I)	作業中のファイル(.wdf)の情報を表示する。
	画面定義の終了(X)	画面定義ウィンドウを閉じる。
編集(E)	切り取り(T)	選択した範囲を切り取って一時的なエリアに保存する。※1
	コピー(C)	選択した範囲をコピーして一時的なエリアに保存する。※1
	貼り付け(P)	一時的なエリアの内容をカーソルのある位置に貼り付ける。※2
	削除(D)	選択された範囲を削除する。

メニューバー	メニュー項目	機能
	集団項目作成(G)	集団項目を作成する。
作表(T)	表作成(G)	表を作成する。
	表要素の追加(A)	表の要素を追加する。
	表要素の削除(D)	表の要素を削除する。
	表要素の切り取り(T)	表の要素を切り取って一時的なエリアに保存する。
	表要素の貼り付け(P)	表の要素を一時的なエリアから貼り付ける。
	表要素のコピー(C)	表の要素をコピーして一時的なエリアに保存する。
オプション(O)	画面定義環境設定(E)	画面定義の実行環境を設定する。
	COBOL 生成許可(A)	COBOL ソースファイルの生成を許可する。
	フィールド作成モード(F)	フィールドの作成モードにする。
	罫線作成モード(R)	罫線の作成モードにする。
表示(V)	ツールバー(T)	ツールバーの表示を制御する。
	ステータスバー(S)	ステータスバーの表示を制御する。
ウィンドウ(W)	重ねて表示(C)	画面を重ねて表示する。
	並べて表示(T)	画面を並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	フォームエディタを開く(F)	フォームエディタウィンドウを開く。
	属性エディタを開く(B)	属性エディタウィンドウを開く。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※1

属性エディタに対して［切り取り］［コピー］メニューを実行した場合，選択した内容はクリップボードに保存されます。

注※2

属性エディタに対して［貼り付け］メニューを実行した場合，クリップボードの内容が貼り付けられます。

付録 B.6　ファイル／レコード定義ウィンドウのメニュー項目一覧

ファイル／レコード定義ウィンドウのメニュー項目一覧を次に示します。

表 B-6　ファイル／レコード定義ウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	新しくファイル／レコード定義ファイルを作成する。
	開く(O)	既存のファイル／レコード定義ファイルを開く。

メニューバー	メニュー項目	機能
	閉じる(C)	既存のファイル／レコード定義ファイルを閉じる。
	上書き保存(S)	設定した情報を作業中のファイル／レコード定義ファイルに保存する。
	名前を付けて保存(A)	設定した情報をファイル／レコード定義ファイルの名前を付けて保存する。
	ファイル情報(I)	作業中のファイル（.flf または.rdf）の情報を表示する。
	ページの設定(U)	印刷の設定を変更する。
	印刷プレビュー(V)	ページ全体を印刷時のイメージで表示する。
	印刷(P)	作業中のファイル（.flf または.rdf）の情報を表示する。
	COBOL 生成(G)	COBOL ソースファイルを生成する。
	ISAM ファイル生成(E)	ISAM ファイルを生成する。
	CSV 形式で出力(T)	作業中のファイル（.flf または.rdf）を CSV 形式で出力する。
	ファイル／レコード定義の終了(X)	ファイル／レコード定義ウィンドウを閉じる。
編集(E)	切り取り(T)	選択したデータ項目または文字列を切り取って一時的なエリアに保存する。
	コピー(C)	選択したデータ項目または文字列をコピーして一時的なエリアに保存する。
	貼り付け(P)	一時的なエリアのデータ項目または文字列を選択した位置に貼り付ける。
	挿入(A)	選択した位置の直前にデータ項目を挿入する。
	削除(D)	選択されたデータ項目を削除する。
	取消(U)	直前の挿入／削除を取り消す。
	検索(F)	指定した文字列を検索する。
	置換(E)	指定した文字列を新しい文字列に変更する。
	位置付け(L)	指定した項番のデータ項目を先頭に表示して，選択状態にする。
オプション(O)	ファイル／レコード定義環境設定(E)	ファイル／レコード定義の実行環境を設定する。
	COBOL 生成許可(A)	COBOL ソースファイルの生成を許可する。
表示(V)	ツールバー(T)	ツールバーの表示を制御する。
	ステータスバー(S)	ステータスバーの表示を制御する。
ウィンドウ(W)	重ねて表示(C)	画面を重ねて表示する。
	並べて表示(T)	画面を並べて表示する。
	アイコンの整列(A)	アイコンを整列させる。
	属性設定を開く(B)	属性設定ウィンドウを開く。

メニューバー	メニュー項目	機能
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

付録 B.7　COBOL エディタウィンドウのメニュー項目一覧

COBOL エディタウィンドウのメニュー項目一覧を次に示します。

表 B-7　COBOL エディタウィンドウのメニュー項目一覧

メニューバー	メニュー項目	機能
ファイル(F)	新規作成(N)	ファイルを新規に作成し，ウィンドウに表示する。
	開く(O)	ファイルを開き，ウィンドウに表示する。
	閉じる(C)	ウィンドウを閉じる。
	上書き保存(S)	編集中のファイルを更新する。
	名前を付けて保存(A)	編集中のファイルに名前を付けて保存する。
	すべて保存(L)	開いているファイルすべてを保存する。
	ページレイアウトの設定(U)	印刷時のページレイアウトを設定する。
	印刷プレビュー(V)	編集中のファイルの内容をプレビュー表示し，印刷する。
	印刷(P)	編集中のファイルの内容を印刷する。
	プロパティ(R)	編集中のファイルの情報を表示する。
	ファイル一覧（開いたファイル名を新しいものから 8 個まで表示される）	指定したファイルを開く。
	COBOL エディタの終了(X)	COBOL エディタを終了する。
編集(E)	元に戻す(U)	直前の編集操作を取り消す。
	やり直し(R)	「元に戻す」で取り消した操作を取り消す。
	切り取り(T)	選択した文字列を切り取って一時的なエリアに保存する。
	コピー(C)	選択した文字列をコピーして一時的なエリアに保存する。
	貼り付け(P)	一時的なエリアの文字列を選択した位置に貼り付ける。
	削除(D)	選択した文字列を削除する。
	すべてを選択(S)	ファイル全体を選択する。
	大文字に変換(E)	選択範囲の英文字を大文字に変換する。
	小文字に変換(L)	選択範囲の英文字を小文字に変換する。
	タブを空白文字に置換(O)	選択範囲のタブ文字を空白文字に置換する。

メニューバー	メニュー項目	機能
	ファイルの挿入(I)	指定したファイルのすべての内容をテキストカーソル位置に挿入する。
検索(S)	文字列の検索(F)	指定した文字列を検索する。
	置換(R)	指定した文字列を新しい文字列に置換する。
	ファイルから検索(I)※	指定した文字列をファイルから検索する。
	ファイルから検索を中止(I)※	ファイルからの検索を中止する。
	ブックマークの設定／解除(B)	ブックマークを設定または解除する。
	次のブックマーク(X)	次のブックマークを検索する。
	前のブックマーク(V)	前のブックマークを検索する。
	ブックマークをすべて解除(C)	すべてのブックマークを解除する。
	ジャンプ(G)	指定した行番号へジャンプする。
	一連番号を検索(S)	指定した一連番号を検索する。
	不当な一連番号を検索(N)	不当な一連番号を検索する。
	80 バイトを超える行を検索(L)	1 行の文字が 80 バイトを超えている行を検索する。
構文(Y)	構文チェック(S)	編集ファイルの構文をチェックする。
	構文チェックの中止(C)	実行中の構文チェックを中止する。
	構文チェック　オプション(O)	構文チェックの環境を設定する。
	構文テンプレート(B)	構文テンプレートをウィンドウに挿入する。
	キーワード補完(E)	COBOL キーワードを補完する。
	ユーザキーワード補完(K)	ユーザキーワードを補完する。
オプション(O)	カスタマイズ(C)	COBOL エディタ，カラー，およびフォントのオプションを設定する。
	エクスポート(E)	カスタマイズ情報を外部ファイルに保存する。
	インポート(I)	カスタマイズ情報を外部ファイルから読み込む。
	ツールバーの表示(S)	[標準(D)]： 　標準用ツールバーの表示／非表示を指定する。 [編集(E)]： 　編集用ツールバーの表示／非表示を指定する。 [検索(S)]： 　検索用ツールバーの表示／非表示を指定する。 [ブックマーク(B)]： 　ブックマーク用ツールバーの表示／非表示を指定する。

メニューバー	メニュー項目	機能
		[ウィンドウ(W)]： ウィンドウ用ツールバーの表示／非表示を指定する。 [構文チェック(C)]： 構文チェック用ツールバーの表示／非表示を指定する。 [正書法(F)]： 正書法用ツールバーの表示／非表示を指定する。
ツール(T)	キーボードマクロ登録開始(S)※	キー入力の記録を開始する。
	キーボードマクロ登録終了(S)※	キー入力の記録を終了する。
	キーボードマクロ実行(R)	記録したキー操作を再生する。
	選択行の右シフト(I)	選択した行を右方向にシフトする。
	選択行の左シフト(E)	選択した行を左方向にシフトする。
	標識領域の指定(A)	[注記(O):*]： 選択範囲の標識領域をアスタリスク(*)に置換する [注記(M):/]： 選択範囲の標識領域を斜線(/)に置換する [継続(N):-]： 選択範囲の標識領域をハイフン(-)に置換する [デバッグ(B):D]： 選択範囲の標識領域を「D」に置換する [空白(A):]： 選択範囲の標識領域を空白に置換する
	見出し領域の置換(L)	見出し領域を，指定した文字列や現在の日付と置換する。
	一連番号の指定(N)	[すべての一連番号を付け直す(A)]： 一連番号領域のすべての一連番号を付け直す [不当な一連番号を付け直す(I)]： 一連番号領域の不当な一連番号を付け直す [一連番号領域を空白に置換(R)]： 一連番号領域を空白に置換する
	COBOL ソース表示形式の切り替え(W)	[固定形式正書法(X)]： マーカと COBOL 構文の色分け表示を固定形式正書法で表示する [自由形式正書法(R)]： マーカと COBOL 構文の色分け表示を自由形式正書法で表示する
ウィンドウ(W)	新しいウィンドウ(N)	表示中のファイルを新しいウィンドウに表示する。
	次のウィンドウ(X)	次のウィンドウをアクティブ状態にする。
	前のウィンドウ(P)	前のウィンドウをアクティブ状態にする。

メニューバー	メニュー項目	機能
	重ねて表示(C)	ウィンドウを重ねて表示する。
	上下に並べて表示(H)	ウィンドウを上下に並べて表示する。
	左右に並べて表示(V)	ウィンドウを左右に並べて表示する。
	すべてアイコン化(I)	ウィンドウをすべてアイコン表示する。
	すべて元のサイズに戻す(R)	アイコン表示されたウィンドウをすべて元のサイズに戻す。
	すべて閉じる(A)	ウィンドウをすべて閉じる。
	開いているウィンドウ	指定したウィンドウを表示する。
	その他のウィンドウ（開いているウィンドウが 10 個以上の場合に表示される）	指定したウィンドウを表示する。
ヘルプ(H)	バージョン情報(A)	バージョン情報を表示する。

注※　機能の開始，終了でメニューが切り替わります。

付録 C　制限値，限界値

付録 C.1　コンパイラの制限値，限界値

コンパイラの制限値，限界値を次に示します。

表 C-1　コンパイラの制限値，限界値

区分	項目	制限値，限界値
全般規定	語の長さ	31 文字
	英数字定数，日本語文字定数の長さ	1～160 文字※16
	固定小数点数字定数のけた数	1～18 けた※11
	浮動小数点数字定数のけた数	仮数部 1～16 けた 指数部 2 けた
	単精度浮動小数点数字定数の値の範囲（絶対値）	約 $10^{-37.9}$ 以上約 $10^{38.5}$ 以下，または 0
	倍精度浮動小数点数字定数の値の範囲（絶対値）	約 10^{-308} 以上約 10^{308} 以下，または 0
	連結式で連結できる定数の長さ	2～1,024 文字
基本機能	作業場所節のデータ項目の長さ	16,777,215 バイト（クラス定義） 1,073,741,823 バイト（上記以外）
	局所場所節のデータ項目の長さ	1,073,741,823 バイト
	ファイル節のデータ項目の長さ	16,777,191 バイト※1（クラス定義のファイル節 FD 記述項） 1,073,741,799 バイト※1（上記以外のファイル節 FD 記述項） 65,535 バイト（ファイル節 SD 記述項）
	連絡節のデータ項目の長さ	16,777,215 バイト（BY VALUE,RETURNING 項目） 1,073,741,823 バイト（上記以外）
	上記以外の節のデータ項目の長さ	16,777,215 バイト
	1 レコードに含まれる可変長項目の制御変数	100 個
	PICTURE 文字列の長さ	30 文字※15
	作業場所節の英字項目のけた数	16,777,215 けた（クラス定義） 1,073,741,823 けた（上記以外）
	局所場所節の英字項目のけた数	1,073,741,823 けた
	ファイル節の英字項目のけた数	16,777,191 バイト※1（クラス定義のファイル節 FD 記述項）

区分	項目	制限値，限界値
		1,073,741,799 バイト※1（上記以外のファイル節 FD 記述項） 65,535 バイト（ファイル節 SD 記述項）
	連絡節の英字項目のけた数	16,777,215 けた（BY VALUE,RETURNING 項目） 1,073,741,823 けた（上記以外）
	上記以外の英字項目のけた数	16,777,215 けた
	外部 10 進，内部 10 進のけた数	18 けた※12
	2 進項目のけた数	18 けた
	外部浮動小数点数字項目のけた数	仮数部 1～16 けた 指数部 2 けた
	作業場所節の英数字，英数字編集項目のけた数	16,777,215 けた（クラス定義） 1,073,741,823 けた（上記以外）
	局所場所節の英数字，英数字編集項目のけた数	1,073,741,823 けた
	ファイル節の英数字，英数字編集項目のけた数	16,777,191 バイト※1（クラス定義のファイル節 FD 記述項） 1,073,741,799 バイト※1（上記以外のファイル節 FD 記述項） 65,535 バイト（ファイル節 SD 記述項）
	連絡節の英数字，英数字編集項目のけた数	16,777,215 けた（BY VALUE,RETURNING 項目） 1,073,741,823 けた（上記以外）
	上記以外の英数字，英数字編集項目のけた数	16,777,215 けた
	数字編集項目のけた数	249 けた (数字は 18 けた以内※14)
	ブール項目のけた数	2,034 けた
	日本語項目のけた数	16,383 けた
	動的長基本項目の LIMIT に指定できる値	英数字項目の場合 • 16,777,214 けた 日本語項目の場合 • 16,382 けた
	外部浮動小数点数字項目の値の範囲（絶対値）	約 $10^{-37.9}$ 以上約 $10^{38.5}$ 以下，または 0
	条件文の入れ子	1,000 重
	うち PERFORM 文の入れ子	100 重
	PERFORM 文の VARYING 指定で変化させられる一意名の数	7 個（AFTER 指定の数は 6 個以内）

区分	項目	制限値，限界値
	EVALUATE 文の入れ子	50 重
	GO TO DEPENDING 文の一意名のけた数	9 けた
	STOP 文の定数の長さ	160 文字※16
	ACCEPT 文で一度に転送できるデータのサイズ	114 バイト（FROM CONSOLE 指定時）
	COPY 文の入れ子	20 重
	原文名の長さ	31 文字※2
	仮原文の原文語	1～322 文字
	演算結果を保証する算術式の中間結果のけた数	30 けた※13
	算術式に含まれる算術演算子	100 個
	作業場所節※3 の節の合計サイズ	16,777,215 バイト（クラス定義） 1,073,741,823 バイト（上記以外）
	局所場所節の節の合計サイズ	1,073,741,823 バイト
	ファイル節※3※4 の節の合計サイズ	16,777,215 バイト（クラス定義） 1,073,741,823 バイト（上記以外）
	その他の節（連絡節を除く）の合計サイズ	16,777,215 バイト
	算術式中に含まれる算術演算子	100 個
表操作機能	作業場所節の OCCURS 句の整数 2 の値，添字の値，部分参照の値	16,777,215（クラス定義） 1,073,741,823（上記以外）
	局所場所節の OCCURS 句の整数 2 の値，添字の値，部分参照の値	1,073,741,823
	ファイル節の OCCURS 句の整数 2 の値，添字の値，部分参照の値	16,777,191（クラス定義のファイル節 FD 記述項） 1,073,741,799（上記以外のファイル節 FD 記述項） 65,535（ファイル節 SD 記述項）
	連絡節の OCCURS 句の整数 2 の値，添字の値，部分参照の値	16,777,215（BY VALUE,RETURNING 項目）※5 1,073,741,823（上記以外）
	上記外の節の OCCURS 句の整数 2 の値，添字の値，部分参照の値	16,777,215
	OCCURS 句に指定できる指標名の数	12 個
	OCCURS 句に定義した指標名に設定できる最大値	10 けた（クラス定義以外の作業場所節，クラス定義以外のファイル節 FD 記述項，局所場所節，BY VALUE／RETURNING 項目以外の連絡節）

区分	項目	制限値，限界値
		9 けた（BY VALUE／RETURNING 項目の連絡節および上記以外の節）
	OCCURS 句の次元数	7 次元
	OCCURS KEY の個数	12 個
	SEARCH 文の入れ子	15 重
入出力機能	RECORD CONTAINS CHARACTERS 句で指定できる値	16,777,191※1 （クラス定義のファイル節 FD 記述項） 1,073,741,799※1 （上記以外のファイル節 FD 記述項） 65,535 （ファイル節 SD 記述項）
	BLOCK CONTAINS 句で指定できる値	16,777,191※1 （クラス定義のファイル節 FD 記述項） 1,073,741,799※1 （上記以外のファイル節 FD 記述項）
	WRITE ADVANCING で指定できる整数	0～99
	RECORD KEY 句で指定できるキーの最大長	255 バイト
	ALTERNATE RECORD KEY 句で指定できるキーの最大長	255 バイト
	ALTERNATE RECORD KEY 句で指定できるキーの個数	98 個
	LINAGE 句で指定できる整数	1～32,767
	LINAGE FOOTING で指定できる整数	0～32,767
	LINAGE TOP で指定できる整数	0～32,767
	LINAGE BOTTOM で指定できる整数	0～32,767
	RECORD IS VARYING 句の FROM に指定できる整数	1～16,777,191※1 （クラス定義のファイル節 FD 記述項） 1～1,073,741,799※1 （上記以外のファイル節 FD 記述項） 1～65,535 （ファイル節 SD 記述項）
	RECORD IS VARYING 句の TO に指定できる整数	1～16,777,191※1 （クラス定義のファイル節 FD 記述項） 1～1,073,741,799※1 （上記以外のファイル節 FD 記述項） 1～65,535 （ファイル節 SD 記述項）
	ファイルのブロック長に指定できる値	1～16,777,191※1

区分	項目	制限値，限界値
		（クラス定義） 1～1,073,741,799 （上記以外）
	CSV 編成ファイルの READ／WRITE 文で扱える基本項目の最大数	Windows(x86) COBOL2002 の場合 • 2,729 個（-NumCsv オプション指定ありの場合） • 5,459 個（-NumCsv オプション指定なしの場合） Windows(x64) COBOL2002 の場合 • 2,047 個（-NumCsv オプション指定ありの場合） • 4,094 個（-NumCsv オプション指定なしの場合）
	ASSIGN 句のデータ名 1 に指定できるデータ項目の最大長	32,767 バイト
プログラム間連絡機能	プログラム名，利用者定義関数名，クラス名，インタフェース名，メソッド名の長さ	31 文字（定数指定でない場合）
	プログラム名，メソッド名を定数指定する場合の長さ	160 バイト※17
	-Main,V3 オプションで引数を受け取るときに指定できる最大長	100 バイト
	FD EXTERNAL 句の数（プログラム単位，実行時単位）	255 個
	01 EXTERNAL 句の数（外部属性）	実行単位内で 32,767 個 プログラム単位で 32,767 個※6
	入れ子のプログラムの個数	65,535 個
	入れ子のプログラムのネストの深さ	255 レベル
	ENTRY 文の個数	32,767 個
	stdcall 呼び出し規約にするプログラムの個数（CALL 一意名がある場合）	1,700 個※7
	CALL 文，INVOKE 文の BY VALUE，BY CONTENT，RETURNING に指定されたデータ項目のサイズ	16,777,215 バイト
	利用者定義関数の引数として指定されたデータ項目のサイズ（ただし，対応する仮引数に BY VALUE が指定または仮定される場合，および返却値だけ）	16,777,215 バイト
整列併合機能	SORT 文，MERGE 文の入力ファイル数	12 個
	整列併合用キーの指定個数	64 個
	整列併合用キーのサイズの合計	4,080 バイト
	整列併合用レコードのサイズ	65,535 バイト

区分	項目	制限値，限界値
データベースアクセス機能	SQL 文の長さ	65,528 バイト※9
	SQL 文で参照する埋め込み変数の長さ	COBOL2002 のデータ項目と同じ
	SQL 文で記述する語，定数の長さ	COBOL2002 の語，定数と同じ
オブジェクト指向機能	クラス定義中に記述可能なメソッドの数	65,530※8
その他	コンパイル可能なソース行数	999,999 行
	手続き名の個数	1,048,575 個
	ファイルの数	255 個
	自由形式正書法での 1 行のバイト数	255 バイト
	COPY 文 PREFIXING 指定および SUFFIXING 指定に指定できる語の長さ	30 文字
	一つの実行単位が使用できるスタックサイズの上限	システムのスタックサイズの設定値に依存
	COBOL2002 で扱えるファイル名の長さの上限	パス名を含めて 255 バイト※10

注

ソース行数以外の制限値は独立していないため，すべての制限値が同時に適用されるとは限りません。

注※1

索引ファイルは，割り当てるファイルによって次のようになります。

- ISAM を使用する場合：65,503 バイト
- Btrieve（Pervasive.SQL）による索引編成ファイルを使用する場合：ページサイズに依存する。
  詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「使用できる機能と制限事項」を参照してください。
- HiRDB による索引編成ファイルを使用する場合：1,073,741,799 バイト
  ただし，CREATE TABLE 定義の指定に依存する。
  詳細は，マニュアル「COBOL2002　ユーザーズガイド」の「HiRDB による索引編成ファイル」を参照してください。

注※2

-V3Rec または-CompatiV3 オプション（VOS3 COBOL85 互換のオプション）を指定した場合，先頭の 8 文字だけが有効となります。

注※3

-MultiThread オプション指定時，ファイル節，作業場所節の合計の上限は，16,777,215 バイトとなります。

注※4

ファイル節の合計サイズは，次のように計算します。

Σ（各記述項のレコード長＋ 24+α）

α：レコード間の境界調整で生じる遊びバイト

レコード間は８バイト境界調整される。

```
FILE SECTION.
FD FILE1.
01 A1.                  }
  02 A2 PIC X(10).     }レコード長10
                       }遊びバイト6
SD FILE2.
01 B1.                  }
  02 B2 PIC X(10).     }レコード長10


ファイル節の合計サイズ=（A1のレコード長+24+遊びバイト）+（B1のレコード長+24）
                      =40＋34
                      =74バイト
```

注※5

添字の値，および部分参照の値の場合，上限は 1,073,741,823 となります。

注※6

ただし，プログラム単位では，アドレスデータ項目の個数や連絡節の 01 レベルデータ項目の個数によって，上記の最大数まで指定できないことがあります。

注※7

次のプログラム数の合計値です。

- 外部プログラム節で，STDCALL または PASCAL に関連づけたプログラム
- stdcall 呼び出し指示ファイル(.cbw)に指定したプログラム
- stdcall 呼び出し規約の DLL 中の内側のプログラム

注※8

クラス定義に含まれるファクトリ定義，オブジェクト定義のデータ記述項に PROPERTY 句が指定されている場合は次のようになります。

クラス定義中に記述可能なメソッドの数

＝ 65,530 - （SET,GET 指定がない PROPERTY 句の数）×2 - （SET,GET 指定がある PROPERTY 句の数）

注※9

SQL 文の長さは，次の項目の合計値になります。

- SQL 文を構成する語，定数の長さ（「EXEC SQL」と「END-EXEC」は含めない）
- 語と定数の間は，分離符の空白文字 1 バイトを加算する。

  （語と定数の間に，注記行や複数の空白文字があっても，常に空白文字 1 バイトとして扱う）

ただし，動的 SQL の長さは，コンパイラではチェックしません。

注※10

パス名を含めないファイル名の長さの上限も 255 バイトです。

使用できるファイル名の長さが機能によって個別に規定されている場合は，その長さが上限となります。また，指定されたファイル名は内部的に絶対パス名に変換することがありますが，このとき内部的に処理できる絶対パス名の長さの上限は，260 バイトです。

注※11

-MaxDigits38 オプションを指定した場合，けた数は 1～38 けたとなります。

注※12

-MaxDigits38 オプションを指定した場合，けた数の上限は 38 けたとなります。

注※13

-IntResult,DecFloat40 オプションを指定した場合，けた数は 40 けたとなります。

注※14

-MaxDigits38 オプションを指定したとき，数字編集項目の数字は 38 けた以内となります。

注※15

-MaxDigits38 オプションを指定したとき，PICTURE 文字列の長さは 60 文字となります。

注※16

-LiteralExtend,Alnum オプションを指定したとき，英数字定数の長さは 1～8,191 文字（バイト）となります。

注※17

-LiteralExtend,Alnum オプションを指定した場合，プログラム名およびメソッド名を定数指定するときの長さは，1～1,024 文字（バイト）となります。

付録 C.2　開発マネージャの制限値，限界値

開発マネージャの制限値，限界値を次に説明します。

表 C-2　開発マネージャの制限値，限界値

項目	制限値，限界値
プロジェクトマスタ中のプロジェクトの数	1,000
一度に登録できるソースファイルの数	ファイル名の長さ＝ 10 バイトの場合：300 ファイル名の長さ＝ 20 バイトの場合：150
登録できるソースファイルの数	3,000
「フォルダ指定によるファイルの追加」ダイアログ内で一度に追加できるファイルの数	3,000
登録できる依存ファイルの数	3,000
ソースとリポジトリの関係のように，ビルドの順番を決定する必要があるファイルのビルドソースと依存関係の階層の深さ	2,000

項目	制限値，限界値
一度にエクスポートできるプロジェクトの数	1,000
一度にインポートできるプロジェクトの数	999
一度に削除できるソースファイルの数	3,000
一度に削除できる依存ファイルの数	3,000
作成できる依存グループの数	50
コンパイラ環境変数 CBLLIB に設定できる値の数	50
コンパイラ環境変数 CBLREP に設定できる値の数	50
コンパイラ環境変数 CBLSYSREP に設定できる値の数	50
ファイルのプロパティのコメントのバイト数	10,000
新しい種類のファイルの登録	50
一つのメニューに対して割り当てられるキーの数	10
開けるツリービューウィンドウの数	30
メッセージウィンドウに表示できるメッセージの件数※	400

注※

1 回のコンパイルまたはリンク単位で，コンパイラ，リンカおよび「新しいファイルの種類の登録」で登録した生成ツールが出力するメッセージおよびエラーメッセージの件数。

付録 C.3　実行支援の制限値，限界値

実行支援の制限値，限界値を次に説明します。

- 環境変数名とパス名の合計バイト数は，1,024 バイトを超えてはいけません。1,024 バイトを超える指定をした場合，1,025 バイト以降の文字は無視されます。
- 実行可能ファイルへの引数を記述する場合，引数の欄に記述できるバイト数は 255 バイトまでです。

付録 C.4　画面定義の制限値，限界値

画面定義の制限値，限界値を次に説明します。

表 C-3　画面定義の制限値，限界値

項目	制限値，限界値
サイズ	160※
行	120

項目	制限値，限界値
けた	160
COBOL の語	30 バイト
PICTURE 文字列	PICTURE 句に指定できる文字数は 30 文字，PICTURE 文字列中で表現できる数字のけた数は 18 けた

注※

MIOS7 から移行した画面定義ファイルの場合，1,920 バイトまで指定できます（ただし，表要素のサイズを除きます）。

付録 C.5　ファイル／レコード定義の制限値，限界値

ファイル／レコード定義の制限値，限界値を次に説明します。

表 C-4　ファイル／レコード定義の制限値，限界値

分類	項目	制限値，限界値
ファイル定義	レコード長	1,073,741,799
	最小レコード長	1,073,741,799
	ブロック長	1,073,741,799
	COBOL の語	30 バイト
	ISAM のキー長	255
	ISAM のレコード長	65,503
レコード定義	サイズ	1,073,741,823
	反復回数	1,073,741,823
	COBOL の語	30 バイト
	PICTURE 文字列	PICTURE 句に指定できる文字数は 30 文字，PICTURE 文字列中で表現できる数字のけた数は 18 けた

付録 C.6　テストデバッガの制限値，限界値

テストデバッガの制限値，限界値を次に示します。

表 C-5　テストデバッガの制限値，限界値

分類	項目	制限値，限界値
テストデバッガ	表示するデータのサイズ	32,767 バイト

分類	項目	制限値，限界値
	起動ダイアログボックスの実行可能ファイルの履歴	10
	起動ダイアログボックスの作業フォルダの履歴	10
	ソーステキストウィンドウの個数	25 個
	常時データ表示ウィンドウのデータ名の最大登録数	25
	常時データ表示ウィンドウのデータ名域に表示される文字列長	128 バイト
	常時データ表示ウィンドウの値表示	255 バイト
	TD コマンド入力ウィンドウのコマンド履歴	999
	「指定行の表示」で指定する行番号	7 けた
	中断点一覧ダイアログボックスで指定するスキップ回数	0～2,147,483,647
	データ監視条件設定ダイアログボックスの添字入力域の次元数	7
	データ監視条件設定ダイアログボックスの添字入力域に指定できる整数値	10 けた以内
	データ監視条件設定ダイアログボックスで指定できる条件数	7 けた以内
	データ値の表示ダイアログボックスの添字入力域に指定できる整数値	10 けた以内
	データ値の代入ダイアログボックスの添字入力域に指定できる整数値	10 けた以内
	オプションダイアログボックスの「ソーステキスト」に指定するバックトレース蓄積数に指定できる整数値	1～500
	オプションダイアログボックスの「ソーステキスト」に指定するアニメーション速度の切り替え	0.1～1.1 秒
	カスタマイズダイアログボックスの「フォントの設定」で指定するサイズ	6～99 ポイント
	モニタウィンドウで表示できる最大行数	5,000 行
TD コマンド	1 行に記述できる TD コマンド文字列長	11,264 バイト以内
	文番号指定の行番号に指定できる整数値	7 けた以内
	文番号指定の位置番号に指定できる整数値	2 けた以内
	添字に指定する次元数	7
	添字に指定する整数定数	10 けた以内
	添字の範囲指定で指定する整数定数	10 けた以内
	部分参照の最左端位置に指定する整数定数	10 けた以内
	部分参照の長さに指定する整数定数	10 けた以内
	拡張 16 進定数の文字列長	160 文字

分類	項目	制限値，限界値
	TD 利用者定義語であるカウンタ変数名，記号名，監視識別子，ケース識別子の文字列長	31 文字
	カウンタ変数に設定される値	1～2,147,483,647
	SET　BREAK コマンドの SKIP オペランドに指定される整数値	1～2,147,483,647
	SET　TRACE コマンドの TOSTEP オペランドに指定される整数値	1～2,147,483,647
	SET　FLOW コマンドの STACK オペランドに指定される整数値	1～500
	データの値表示の表示形式	見出し域 12 バイト 値域 80 バイト
	記号名構造指定のレベル番号に指定される整数値	2 けた以内
	REPEAT コマンドの TIMES オペランドに指定される整数値	1～2,147,483,647
	CASE コマンドの TOSTEP オペランドに指定される整数値	1～2,147,483,647
	ASSIGN CASECODE コマンドで設定できる値	0～2,147,483,647
	TD コマンド群のネスト数	128 以内
カバレージ	カバレージ情報の表示形式：対象総数，実行済数，未実行数	8 けた
	カバレージ情報の表示形式：変更回数	0～32,767
	カバレージ情報の表示形式：テスト回数	0～32,766
	カバレージ情報の表示形式：実行回数	0～2,147,483,647
	カバレージ情報の蓄積ダイアログボックスの実行可能ファイルの履歴	10
	カバレージ情報の蓄積ダイアログボックスの作業フォルダの履歴	10
	カバレージ情報の蓄積ダイアログボックスのユーザパラメタ入力域の履歴	10
	カウント情報の表示ダイアログボックスの実行可能ファイルの履歴	10
	カウント情報の表示ダイアログボックスの作業フォルダの履歴	10
	カウント情報の表示ダイアログボックスのユーザパラメタ入力域の履歴	10

付録 D　テストデバッガの例題

付録 D.1　例題プログラム

(1)　主プログラム（AVERAGE）

入力ファイル inputdata.txt から English，Japanese，Mathematics の得点を入力し，３教科の平均点を表示するプログラムです。

```
000010 IDENTIFICATION  DIVISION.
000020 PROGRAM-ID.     AVERAGE.
000030
000040 ENVIRONMENT     DIVISION.
000050 INPUT-OUTPUT    SECTION.
000060 FILE-CONTROL.
000070      SELECT     IN-FILE  ASSIGN TO SYS001
000080        ORGANIZATION IS LINE SEQUENTIAL.
000090      SELECT     OUT-FILE ASSIGN TO SYS002.
000100 DATA            DIVISION.
000110 FILE            SECTION.
000120 FD   IN-FILE.
000130 01   IN-REC.
000140  02   IN-POINT  PIC 9(9).
000150  02   FILLER    PIC X.
000160 FD   OUT-FILE   LABEL RECORD OMITTED.
000170 01   OUT-REC    PIC X(80).
000180 WORKING-STORAGE SECTION.
000190 01  PR-REC.
000200  02  FILLER     PIC X(13) VALUE SPACE.
000210  02  E-POINT    PIC Z(3)9.
000220  02  FILLER     PIC X(6)  VALUE SPACE.
000230  02  K-POINT    PIC Z(3)9.
000240  02  FILLER     PIC X(8)  VALUE SPACE.
000250  02  S-POINT    PIC Z(3)9.
000260  02  FILLER     PIC X(5)  VALUE SPACE.
000270  02  A-POINT    PIC Z(3)9.
000280  02  FILLER     PIC X(31) VALUE SPACE.
000290 01  TITLE.
000300  02  FILLER     PIC X(10) VALUE SPACE.
000310  02  T-REC      PIC X(40)
000320              VALUE 'English  Japanese  Mathematics  Average'.
000330  02  FILLER     PIC X(30) VALUE SPACE.
000340 01  ERR-MSG.
000350  02  FILLER     PIC X(10)  VALUE SPACE.
000360  02  ERMSG      PIC X(34)
000370              VALUE '** Error:  Input data is mistaken.'.
000380  02  ERMSG2     PIC X(18) VALUE ' ==> error data:'.
000390  02  ER-REC.
000400   03  ER-DATA   PIC 9(9).
000410  02  FILLER     PIC X(9)  VALUE SPACE.
000420 01  NO-MSG.
000430  02  FILLER     PIC X(10) VALUE SPACE.
```

```
000440  02  NMSG       PIC X(40)
000450              VALUE '** Information:  There is no input data.'.
000460  02  FILLER     PIC X(30) VALUE SPACE.
000470 77  ENDM       PIC X(3).
000480 88  END-OF-FIL  VALUE 'EOF'.
000490 01  WORK-TBL.
000500  02  W-TBL      OCCURS 20.
000510   03  W-REC.
000520    04  WE-P     PIC 9(3).
000530    04  WK-P     PIC 9(3).
000540    04  WS-P     PIC 9(3).
000550   03  WA-P      PIC 9(4).
000560   03  R-CODE    PIC 9.
000570 77  I           PIC 9(3).
000580 77  J           PIC 9(3).
000590*
000600 PROCEDURE       DIVISION.
000610 MAIN-PROCESS    SECTION.
000620 OPEN-FILE.
000630        OPEN INPUT  IN-FILE
000640             OUTPUT OUT-FILE.
000650 MAKING-WTBL.
000660        READ IN-FILE
000670             AT END MOVE 'EOF' TO ENDM.
000680        PERFORM VARYING I FROM 1 BY 1 UNTIL END-OF-FIL
000690          MOVE IN-POINT TO W-REC(I)
000700          READ IN-FILE AT END MOVE 'EOF' TO ENDM
000710          END-READ
000720         END-PERFORM.
000730 ZERO-CHECK.
000740        COMPUTE I = I - 1.
000750        IF I = 0
000760          WRITE OUT-REC FROM NO-MSG AFTER 1
000770          GO TO CLOSE-FILE
000780        ELSE
000790          CONTINUE
000800        END-IF.
000810 CALCULATE.
000820        CALL 'SUBAV' USING WORK-TBL I.
000830 PRINTING-DATA.
000840        PERFORM  DATA-PUT.
000850 CLOSE-FILE.
000860       CLOSE IN-FILE OUT-FILE.
000870       STOP RUN.
000880*
000890 DATA-PUT        SECTION.
000900 PRINTING-TITLE.
000910        WRITE OUT-REC FROM TITLE AFTER 1.
000920 PRINTING-AVERAGE.
000930        PERFORM VARYING J FROM 1 BY 1 UNTIL J > I
000940          IF R-CODE(J) = 8
000950            MOVE W-REC(J) TO ER-REC
000960            WRITE OUT-REC FROM ERR-MSG AFTER 1
000970          ELSE
000980            MOVE WE-P(J) TO E-POINT
000990            MOVE WK-P(J) TO K-POINT
001000            MOVE WS-P(J) TO S-POINT
001010            MOVE WA-P(J) TO A-POINT
```

```
001020            WRITE OUT-REC FROM PR-REC AFTER 1
001030          END-IF
001040        END-PERFORM.
001050 END PROGRAM AVERAGE.
```

(2) 副プログラム（SUBAV）

平均点を計算するプログラムです。

```
000010 IDENTIFICATION  DIVISION.
000020 PROGRAM-ID.     SUBAV.
000030 ENVIRONMENT     DIVISION.
000040 DATA            DIVISION.
000050 WORKING-STORAGE SECTION.
000060 77  I            PIC S9(8) COMP.
000070 77  J            PIC S9(8) COMP.
000080 77  K            PIC S9(8) COMP.
000090 77  L            PIC S9(8) COMP.
000100 77  M            PIC S9(8) COMP.
000110 77  MM           PIC S9(8) COMP.
000120 77  CHGSW        PIC X     VALUE LOW-VALUE.
000130 88  NO-CHANGE    VALUE HIGH-VALUE.
000140 88  ALL-CHANGE   VALUE LOW-VALUE.
000150 01  SORT-TBL.
000160  02  S-TBL       OCCURS 20 INDEXED BY DATA-INDEX
000170                                 COMP-INDEX1
000180                                 COMP-INDEX2.
000190   03  S-P.
000200    04  FILLER    PIC 9(9).
000210   03  SA-P        PIC 9(4).
000220   03  FILLER      PIC 9.
000230 01  SW-TBL.
000240  02  SORT-ITEM   INDEX  OCCURS 20 INDEXED BY SORT-INDEX.
000250 LINKAGE         SECTION.
000260 01  WORK-TBL.
000270  02  W-TBL       OCCURS 20.
000280   03  W-REC.
000290    04  WE-P      PIC 9(3).
000300    04  WK-P      PIC 9(3).
000310    04  WS-P      PIC 9(3).
000320   03  WA-P        PIC 9(4).
000330   03  R-CODE      PIC 9.
000340 77  KOSU          PIC 9(3).
000350*
000360 PROCEDURE       DIVISION USING WORK-TBL KOSU.
000370 MAIN-PROCESS    SECTION.
000380 DATA-CHEK-AND-COMPUTE.
000390        PERFORM VARYING I FROM 1 BY 1 UNTIL I > KOSU
000400          MOVE ZERO TO WA-P(I)
000410          IF WE-P(I) > 100 OR
000420             WK-P(I) > 100 OR
000430             WS-P(I) > 100
000440            MOVE 8 TO R-CODE(I)
000450          ELSE
000460            MOVE 0 TO R-CODE(I)
000470            COMPUTE WA-P(I) ROUNDED
```

```
000480             = (WE-P(I) + WK-P(I) + WS-P(I)) / 3
000490           END-IF
000500           MOVE W-TBL(I) TO S-TBL(I).
000510         END-PERFORM.
000520         PERFORM DATA-SORT.
000530         EXIT PROGRAM.
000540*
000550 DATA-SORT       SECTION.
000560         IF KOSU = 1 THEN
000570           GO TO DATA-SORT-EXIT
000580         ELSE
000590           CONTINUE
000600         END-IF.
000610         PERFORM VARYING I FROM 1 BY 1 UNTIL I > 20
000620           SET DATA-INDEX TO I
000630           SET SORT-ITEM(I) TO DATA-INDEX
000640         END-PERFORM.
000650         MOVE 2 TO M.
000660         PERFORM UNTIL M > KOSU
000670           COMPUTE M = M * 2
000680         END-PERFORM.
000690         COMPUTE M = M / 2 - 1.
000700         PERFORM UNTIL M < 1
000710           PERFORM VARYING I FROM 1 BY 1 UNTIL I > M
000720             PERFORM VARYING J FROM I BY M UNTIL J + M > KOSU
000730               COMPUTE K = J + M
000740               SET COMP-INDEX1 TO SORT-ITEM(J)
000750               SET COMP-INDEX2 TO SORT-ITEM(K)
000760               IF SA-P(COMP-INDEX1) < SA-P(COMP-INDEX2) THEN
000770                 SET SORT-ITEM(K) TO COMP-INDEX1
000780                 MOVE LOW-VALUE TO CHGSW
000790                 COMPUTE MM = - M
000800                 PERFORM VARYING K FROM J BY MM
000810                         UNTIL K - M < I OR NO-CHANGE
000820                   COMPUTE L = K - M
000830                   SET COMP-INDEX1 TO SORT-ITEM(L)
000840                   IF SA-P(COMP-INDEX1) < SA-P(COMP-INDEX2) THEN
000850                     SET SORT-ITEM(K) TO COMP-INDEX1
000860                   ELSE
000870                     SET SORT-ITEM(K) TO COMP-INDEX2
000880                     MOVE HIGH-VALUE TO CHGSW
000890                   END-IF
000900                 END-PERFORM
000910                 IF ALL-CHANGE THEN
000920                   IF J = I THEN
000930                     SET SORT-ITEM(J) TO COMP-INDEX2
000940                   ELSE
000950                     SET SORT-ITEM(L) TO COMP-INDEX2
000960                   END-IF
000970                 ELSE
000980                   CONTINUE
000990                 END-IF
001000               ELSE
001010                 CONTINUE
001020               END-IF
001030             END-PERFORM
001040           END-PERFORM
001050           COMPUTE M = M / 2
```

```
001060         END-PERFORM.
001070         PERFORM VARYING J FROM 1 BY 1 UNTIL J > KOSU
001080           SET DATA-INDEX TO SORT-ITEM(J)
001090           MOVE S-TBL(DATA-INDEX) TO W-TBL(J)
001100         END-PERFORM.
001110 DATA-SORT-EXIT.
001120         EXIT.
001130 END PROGRAM SUBAV.
```

(3) 実行結果

```
 English  Japanese  Mathematics  Average
     100       100          100      100
      78        89           99       89
      90        90           80       87
      74        85           96       85
      75        48           86       70
     100         0          100       67
      32        65           98       65
      30        70           80       60
      45        56           78       60
      68        58           27       51
      46        85           21       51
      40        50           60       50
      12        45           78       45
      14        25           36       25
      11        22           33       22
      10        20           30       20
** Error:  Input data is mistaken. ==> error data:  200400300
** Error:  Input data is mistaken. ==> error data:  110450900
** Error:  Input data is mistaken. ==> error data:  111222333
       0         0            0        0
```

付録 D.2 TD コマンドとその結果

(1) TD コマンドの例

入力ファイル（IN-FILE）と副プログラム（SUBAV）が未完成な場合，擬似実行によって主プログラムをテストします。

図 D-1 テストケース 1 （TD コマンドの例）

```
#INCLUDE INFILE(INCAV1)
*>************************************************************************
*>      Input File 及び SUBAV が未作成                                    *
*>************************************************************************
*>------------------------------------------------------------------------*
*>      入力データがない場合     テスト NO  --> 001                       *
*>------------------------------------------------------------------------*
SET QUALIFICATION(#PROGRAM(AVERAGE/AVERAGE))
SIMULATE FILE(IN-FILE) OPENMODE(INPUT)          *> READ文の擬似実行手続き
    GO END                                                                  ※
ENDSIMULATE
SIMULATE FILE(OUT-FILE) OPENMODE(OUTPUT)        *> WRITE文の擬似実行手続き
ENDSIMULATE
GO COVERAGE
KCCC1200T-I ユーザプログラムの実行を開始しました。
KCCC1205T-I COBOLプログラムの実行を開始しました。
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）660 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）760 <AVERAGE/AVERAGE>
KCCC1206T-I COBOLプログラムの実行を終了しました。
KCCC1201T-I ユーザプログラムの実行を終了しました。
KCCC3405T-I カバレージ情報の蓄積を行いました。翻訳単位名(AVERAGE) ファイル（D:¥2002¥TP¥AVERAGE.cbp）
```

注※　テストケース１の TD コマンド群

図 D-2　テストケース２（TD コマンドの例）

```
#INCLUDE INFILE(INCAV2)
*>*************************************************************************
*>      Input File 及び SUBAV が未作成                                     *
*>*************************************************************************
*>-------------------------------------------------------------------------*
*>      入力データが全て正しい場合     テスト NO  --> 002                  *
*>-------------------------------------------------------------------------*
SET QUALIFICATION(#PROGRAM(AVERAGE/AVERAGE))
SIMULATE FILE(IN-FILE) OPENMODE(INPUT)          *> READ文の擬似実行手続き
    REPEAT
        ASSIGN DATA(IN-REC) VALUE(080100090)
        DISPLAY DATA(IN-REC)
    ENDREPEAT
    REPEAT
        ASSIGN DATA(IN-REC) VALUE(050060070)
        DISPLAY DATA(IN-REC)
    ENDREPEAT
    REPEAT
        ASSIGN DATA(IN-REC) VALUE(010050060)
        DISPLAY DATA(IN-REC)
    ENDREPEAT
    REPEAT
        GO END
    ENDREPEAT
ENDSIMULATE
SIMULATE FILE(OUT-FILE) OPENMODE(OUTPUT)        *> WRITE文の擬似実行手続き
ENDSIMULATE
SIMULATE SUB(#PROGRAM(SUBAV))                   *> 副プログラムの擬似実行手続き
    ASSIGN DATA(WA-P(1)) VALUE(90)
    ASSIGN DATA(R-CODE(1)) VALUE(0)
    ASSIGN DATA(WA-P(2)) VALUE(60)
    ASSIGN DATA(R-CODE(2)) VALUE(0)
    ASSIGN DATA(WA-P(3)) VALUE(40)
    ASSIGN DATA(R-CODE(3)) VALUE(0)
    DISPLAY DATA(WA-P(1..3))
    DISPLAY DATA(R-CODE(1..3))
ENDSIMULATE
GO COVERAGE
KCCC1200T-I ユーザプログラムの実行を開始しました。
KCCC1205T-I COBOLプログラムの実行を開始しました。
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）660 <AVERAGE/AVERAGE>
名　称      IN-REC
  値        080100090
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）700 <AVERAGE/AVERAGE>
名　称      IN-REC
  値        050060070
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）700 <AVERAGE/AVERAGE>
名　称      IN-REC
  値        010050060
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）700 <AVERAGE/AVERAGE>
KCCC1250T-I 副プログラムシミュレーションを開始します。 プログラム名（SUBAV）820 <AVERAGE/AVERAGE>
名　称      WA-P(1..3)
  値        0090    0060    0040
名　称      R-CODE(1..3)
  値        0    0    0
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）910 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）1020 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）1020 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）1020 <AVERAGE/AVERAGE>
KCCC1206T-I COBOLプログラムの実行を終了しました。
KCCC1201T-I ユーザプログラムの実行を終了しました。
KCCC3405T-I カバレージ情報の蓄積を行いました。翻訳単位名(AVERAGE) ファイル（D:¥2002¥TP¥AVERAGE.cbp）
```

※

注※　テストケース２の TD コマンド群

図 D-3　テストケース３（TD コマンドの例）

```
#INCLUDE INFILE(INCAV3)
*>*************************************************************************
*>      Input File 及び SUBAV が未作成                                     *
*>*************************************************************************
*>-------------------------------------------------------------------------*
*>      入力データにエラーがある場合     テスト NO  --> 003                 *
*>-------------------------------------------------------------------------*
SET QUALIFICATION(#PROGRAM(AVERAGE/AVERAGE))
SIMULATE FILE(IN-FILE) OPENMODE(INPUT)          *> READ文の擬似実行手続き
    REPEAT
        ASSIGN DATA(IN-REC) VALUE(080060100)
        DISPLAY DATA(IN-REC)
    ENDREPEAT
    REPEAT
        ASSIGN DATA(IN-REC) VALUE(110020300)
        DISPLAY DATA(IN-REC)
    ENDREPEAT
    REPEAT
        ASSIGN DATA(IN-REC) VALUE(060080040)
        DISPLAY DATA(IN-REC)
    ENDREPEAT
    REPEAT
        GO END
    ENDREPEAT
ENDSIMULATE
SIMULATE FILE(OUT-FILE) OPENMODE(OUTPUT)        *> WRITE文の擬似実行手続き
ENDSIMULATE
SIMULATE SUB(#PROGRAM(SUBAV))                   *> 副プログラムの擬似実行手続き
    ASSIGN DATA(WA-P(1)) VALUE(80)
    ASSIGN DATA(R-CODE(1)) VALUE(0)
    ASSIGN DATA(WA-P(2)) VALUE(0)
    ASSIGN DATA(R-CODE(2)) VALUE(8)
    ASSIGN DATA(WA-P(3)) VALUE(60)
    ASSIGN DATA(R-CODE(3)) VALUE(0)
    DISPLAY DATA(WA-P(1..3))
    DISPLAY DATA(R-CODE(1..3))
ENDSIMULATE
GO COVERAGE
KCCC1200T-I ユーザプログラムの実行を開始しました。
KCCC1205T-I COBOLプログラムの実行を開始しました。
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）660 <AVERAGE/AVERAGE>
名　称      IN-REC
  値        080060100
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）700 <AVERAGE/AVERAGE>
名　称      IN-REC
  値        110020300
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）700 <AVERAGE/AVERAGE>
名　称      IN-REC
  値        060080040
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（IN-FILE）700<AVERAGE/AVERAGE>
KCCC1250T-I 副プログラムシミュレーションを開始します。 プログラム名（SUBAV）820 <AVERAGE/AVERAGE>
名　称      WA-P(1..3)
  値        0080    0000    0060
名　称      R-CODE(1..3)
  値        0    8    0
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）910 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）1020 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）960 <AVERAGE/AVERAGE>
KCCC1232T-I ファイルシミュレーションを開始します。ファイル名（OUT-FILE）1020 <AVERAGE/AVERAGE>
KCCC1206T-I COBOLプログラムの実行を終了しました。
KCCC1201T-I ユーザプログラムの実行を終了しました。
KCCC3405T-I カバレージ情報の蓄積を行いました。翻訳単位名(AVERAGE) ファイル（D:¥2002¥TP¥AVERAGE.cbp）
```

※

注※　テストケース３の TD コマンド群

(2)　カバレージ情報

付録 D.2（１）に示した TD コマンドの例（テストケース１～３）で，主プログラム（AVERAGE）の単体テストをした結果のカバレージ情報を次に示します。

図 D-4　テストケース１（カバレージ情報）

```
                                ******************************
                                *       カバレージ情報       *
     COBOL2002 (c)  VV-RR       ******************************        YYYY-MM-DD HH:MM:SS
-------------------------------------------------------------------------------------------
  プログラム名   : AVERAGE
  コンパイル日時: YYYY-MM-DD HH:MM:SS                                  変更回数    :     0
  テスト日時     : YYYY-MM-DD HH:MM:SS                                  テスト回数  :     1

-------------------------------------<全ソース表示>-------------------------------------
プログラム名 : AVERAGE
変更<C0 > <C1 >-----------------------------------------------------------------------------
                0000600 PROCEDURE       DIVISION.
                0000610 MAIN-PROCESS    SECTION.
                0000620 OPEN-FILE.
    *           0000630         OPEN INPUT  IN-FILE
                0000640              OUTPUT OUT-FILE.
                0000650 MAKING-WTBL.
    *           0000660         READ IN-FILE
    *     @     0000670              AT END MOVE 'EOF' TO ENDM.
          .
    *           0000680         PERFORM VARYING I FROM 1 BY 1 UNTIL END-OF-FIL
    .     .     0000690           MOVE IN-POINT TO W-REC(I)
    ..    .     0000700           READ IN-FILE AT END MOVE 'EOF' TO ENDM
                0000710           END-READ
                0000720          END-PERFORM.
          .
          @
                0000730 ZERO-CHECK.
    *           0000740         COMPUTE I = I - 1.
    *           0000750         IF I = 0
    *     @     0000760           WRITE OUT-REC FROM NO-MSG AFTER 1
    *           0000770           GO TO CLOSE-FILE
          .     0000780         ELSE
    .           0000790           CONTINUE
                0000800         END-IF.
                0000810 CALCULATE.
    .           0000820         CALL 'SUBAV' USING WORK-TBL I.
                0000830 PRINTING-DATA.
    .           0000840         PERFORM  DATA-PUT.
                0000850 CLOSE-FILE.
    *           0000860        CLOSE IN-FILE OUT-FILE.
    *           0000870        STOP RUN.
                0000890 DATA-PUT        SECTION.
                0000900 PRINTING-TITLE.
    .           0000910         WRITE OUT-REC FROM TITLE AFTER 1.
                0000920 PRINTING-AVERAGE.
    .           0000930         PERFORM VARYING J FROM 1 BY 1 UNTIL J > I
          .
    .           0000940           IF R-CODE(J) = 8
    .     .     0000950             MOVE W-REC(J) TO ER-REC
    .           0000960             WRITE OUT-REC FROM ERR-MSG AFTER 1
          .     0000970           ELSE
    .           0000980             MOVE WE-P(J) TO E-POINT
    .           0000990             MOVE WK-P(J) TO K-POINT
    .           0001000             MOVE WS-P(J) TO S-POINT
    .           0001010             MOVE WA-P(J) TO A-POINT
    .           0001020             WRITE OUT-REC FROM PR-REC AFTER 1
                0001030           END-IF
                0001040         END-PERFORM.
                0001050 END PROGRAM AVERAGE.
          .

*******************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         26            0         12            0          1            0      *
*     実行済数         10            0          3            0          0            0      *
*     未実行数         16            0          9            0          1            0      *
*     カバレージ率   38.4%         0.0%       25.0%         0.0%       0.0%         0.0%      *
*******************************************************************************************
```

図 D-5　テストケース 2（カバレージ情報）

```
                                ******************************
                                *      カバレージ情報        *
     COBOL2002 (c)  VV-RR       ******************************        YYYY-MM-DD HH:MM:SS
-------------------------------------------------------------------------------------------
  プログラム名   : AVERAGE
  コンパイル日時: YYYY-MM-DD HH:MM:SS                                  変更回数    :     0
  テスト日時     : YYYY-MM-DD HH:MM:SS                                  テスト回数  :     2

-------------------------------------<全ソース表示>-------------------------------------
プログラム名 : AVERAGE
変更<C0 > <C1 >-----------------------------------------------------------------------------
                0000600 PROCEDURE       DIVISION.
                0000610 MAIN-PROCESS    SECTION.
                0000620 OPEN-FILE.
    *           0000630         OPEN INPUT  IN-FILE
                0000640              OUTPUT OUT-FILE.
                0000650 MAKING-WTBL.
    *           0000660         READ IN-FILE
    *       @   0000670              AT END MOVE 'EOF' TO ENDM.
            @
    *           0000680         PERFORM VARYING I FROM 1 BY 1 UNTIL END-OF-FIL
    *       @   0000690           MOVE IN-POINT TO W-REC(I)
    **      @   0000700           READ IN-FILE AT END MOVE 'EOF' TO ENDM
                0000710           END-READ
                0000720          END-PERFORM.
            @
            @
                0000730 ZERO-CHECK.
    *           0000740         COMPUTE I = I - 1.
    *           0000750         IF I = 0
    *       @   0000760           WRITE OUT-REC FROM NO-MSG AFTER 1
    *           0000770           GO TO CLOSE-FILE
            @   0000780         ELSE
    *           0000790           CONTINUE
                0000800         END-IF.
                0000810 CALCULATE.
    *           0000820         CALL 'SUBAV' USING WORK-TBL I.
                0000830 PRINTING-DATA.
    *           0000840         PERFORM  DATA-PUT.
                0000850 CLOSE-FILE.
    *           0000860        CLOSE IN-FILE OUT-FILE.
    *           0000870        STOP RUN.
                0000890 DATA-PUT        SECTION.
                0000900 PRINTING-TITLE.
    *           0000910         WRITE OUT-REC FROM TITLE AFTER 1.
                0000920 PRINTING-AVERAGE.
    *           0000930         PERFORM VARYING J FROM 1 BY 1 UNTIL J > I
            @
    *           0000940           IF R-CODE(J) = 8
    .       .   0000950             MOVE W-REC(J) TO ER-REC
    .           0000960             WRITE OUT-REC FROM ERR-MSG AFTER 1
            @   0000970           ELSE
    *           0000980             MOVE WE-P(J) TO E-POINT
    *           0000990             MOVE WK-P(J) TO K-POINT
    *           0001000             MOVE WS-P(J) TO S-POINT
    *           0001010             MOVE WA-P(J) TO A-POINT
    *           0001020             WRITE OUT-REC FROM PR-REC AFTER 1
                0001030           END-IF
                0001040         END-PERFORM.
                0001050 END PROGRAM AVERAGE.
            @

*******************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         26            0         12            0          1            0      *
*     実行済数         24            0         11            0          1            0      *
*     未実行数          2            0          1            0          0            0      *
*     カバレージ率   92.3%         0.0%       91.6%         0.0%     100.0%         0.0%      *
*******************************************************************************************
```

図 D-6　テストケース３（カバレージ情報）

```
                          ******************************
                          *      カバレージ情報        *
     COBOL2002 (c)  VV-RR ******************************          YYYY-MM-DD HH:MM:SS
--------------------------------------------------------------------------------------------
  プログラム名   : AVERAGE
  コンパイル日時: YYYY-MM-DD HH:MM:SS                              変更回数     :      0
  テスト日時     : YYYY-MM-DD HH:MM:SS                              テスト回数   :      3

----------------------------------<全ソース表示>----------------------------------------
プログラム名 : AVERAGE
変更<C0 > <C1 >-----------------------------------------------------------------------------
              0000600 PROCEDURE       DIVISION.
              0000610 MAIN-PROCESS    SECTION.
              0000620 OPEN-FILE.
    *         0000630         OPEN INPUT  IN-FILE
              0000640                 OUTPUT OUT-FILE.
              0000650 MAKING-WTBL.
    *         0000660         READ IN-FILE
    *     @   0000670             AT END MOVE 'EOF' TO ENDM.
          @
    *         0000680         PERFORM VARYING I FROM 1 BY 1 UNTIL END-OF-FIL
    *     @   0000690           MOVE IN-POINT TO W-REC(I)
    **    @   0000700           READ IN-FILE AT END MOVE 'EOF' TO ENDM
              0000710           END-READ
              0000720         END-PERFORM.
          @
          @
              0000730 ZERO-CHECK.
    *         0000740         COMPUTE I = I - 1.
    *         0000750         IF I = 0
    *     @   0000760           WRITE OUT-REC FROM NO-MSG AFTER 1
    *         0000770           GO TO CLOSE-FILE
          @   0000780         ELSE
    *         0000790           CONTINUE
              0000800         END-IF.
              0000810 CALCULATE.
    *         0000820         CALL 'SUBAV' USING WORK-TBL I.
              0000830 PRINTING-DATA.
    *         0000840         PERFORM  DATA-PUT.
              0000850 CLOSE-FILE.
    *         0000860        CLOSE IN-FILE OUT-FILE.
    *         0000870        STOP RUN.
              0000890 DATA-PUT        SECTION.
              0000900 PRINTING-TITLE.
    *         0000910         WRITE OUT-REC FROM TITLE AFTER 1.
              0000920 PRINTING-AVERAGE.
    *         0000930         PERFORM VARYING J FROM 1 BY 1 UNTIL J > I
          @
    *         0000940           IF R-CODE(J) = 8
    *     @   0000950             MOVE W-REC(J) TO ER-REC
    *         0000960             WRITE OUT-REC FROM ERR-MSG AFTER 1
          @   0000970           ELSE
    *         0000980             MOVE WE-P(J) TO E-POINT
    *         0000990             MOVE WK-P(J) TO K-POINT
    *         0001000             MOVE WS-P(J) TO S-POINT
    *         0001010             MOVE WA-P(J) TO A-POINT
    *         0001020             WRITE OUT-REC FROM PR-REC AFTER 1
              0001030           END-IF
              0001040         END-PERFORM.
              0001050 END PROGRAM AVERAGE.
          @

********************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>     *
*     対象総数         26            0         12            0          1            0     *
*     実行済数         26            0         12            0          1            0     *
*     未実行数          0            0          0            0          0            0     *
*     カバレージ率   100.0%         0.0%      100.0%         0.0%     100.0%         0.0%     *
********************************************************************************************
```

付録 D.3　カバレージ情報の操作

(1)　プログラム情報ファイルのマージ

次の例では，プログラム情報ファイルをマスタ用ファイルとテスト用ファイルに分けて使用します。カバレージ情報はテスト用ファイルに蓄積し，テスト後にマスタ用ファイルにマージします。このようにプログラム情報ファイルを使用すると，ファイルを保護したり，同一のテストプログラムを複数のオペレータでテストしたりできます。

(使用例)

プログラム情報ファイル c:¥users¥test¥master¥testa.cbp をマスタ用ファイルとして使用する。

テスト方法

```
copy c:¥users¥test¥master¥testa.cbp c:¥users¥temp¥testa.cbp (1.)
set CBLPIDIR=c:¥users¥temp (2.)
cbltd2k -input c:¥users¥batch¥testa01.tdi -execute c:¥users¥test¥testa.exe (3.)
```

1. マスタ用のプログラム情報ファイルをコピーする。
2. 採取したカバレージ情報を格納するフォルダを環境変数 CBLPIDIR で設定する。
3. cbltd2k コマンドでプログラムのテストを実行し，カバレージ情報を採取する。TD コマンド格納ファイル中の GO TD コマンドには，COVERAGE オペランドを指定しておく。

マージ方法

cblca2k -merge c:¥users¥test¥master¥testa.cbp -input c:¥users¥tesmp¥testa.cbp

付録 E　カバレージ情報の表示例

付録 E.1　カバレージ情報の表示例（テキスト形式）

(1)　翻訳単位の一覧表示

```
                                        ************************************************
                                        *  カ バ レ ー ジ 情 報 一 覧          *
    COBOL2002 (c)    VV-RR              ************************************************        2003-01-01 09:00:00

                  *-----------C0-----------*  *----------差分C0----------*  *-----------C1-----------*  *----------差分C1----------*
番号  名  称   種別 対象総数  実行済数   C0   対象総数  実行済数  差分C0  対象総数  実行済数   C1   対象総数  実行済数  差分C1
   1 MAKELINE     C       16        14  87.5%         3         1  33.3%         4         3  75.0%         2         1  50.0%
   2 OUTLINE      C       25        10  40.0%         0         0   0.0%         6         3  50.0%         0         0   0.0%
   3  *           C       10         2  20.0%         0         0   0.0%         0         0   0.0%         0         0   0.0%
------------------------------------------------------------------------------------------------------------------------------------

    <名称>
    *  3   FORMAT_OUTLINE

************************************************************************************************************************************
*      合計                51        26  50.9%         3         1  33.3%        10         6  60.0%         2         1  50.0%*
************************************************************************************************************************************
```

名称欄には，プログラム名，またはクラス名が最大 8 文字で表示されます。名称が 9 文字以上の場合，名称欄には記号（＊）が表示され，正しい名称は〈名称〉の欄に表示されます。

種別欄には，プログラムの場合は「P」が，クラスの場合は「C」，利用者定義関数の場合は「T」が表示されます。

(2)　まとめ情報の表示

```
                                ****************************
                                *     カバレージ情報       *
    COBOL2002 (c)  VV-RR        ****************************       2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :       1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数  :       2

*************************************************************************************************
*                    <C0>    <差分C0>    <C1>    <差分C1>    <S1>    <差分S1>     *
*      対象総数        16           3          4          2          5          0     *
*      実行済数        14           1          3          1          5          0     *
*      未実行数         2           2          1          1          0          0     *
*      カバレージ率   87.5%       33.3%      75.0%      50.0%     100.0%      0.0%     *
*************************************************************************************************
```

- 変更回数：最初のコンパイル時からの COBOL ソースファイルの変更回数を示します。
- テスト回数：カバレージ情報を蓄積した回数を示します。
- テスト日時：カバレージ情報を蓄積した最新の日時を表示します。

(3)　全ソース表示

```
                              ******************************
                              *     カバレージ情報         *
     COBOL2002 (c)  VV-RR     ******************************      2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数     :      1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数   :      2

-----------------------------------<全ソース表示>----------------------------------------
メソッド名    :INIT_MAKELINE_F
変更<C0 > <C1 >-------------------------------------------------------------------------------
                0000021 PROCEDURE  DIVISION.
    *           0000022     INVOKE SUPER  'INIT-COLORS-F'.
    *           0000023     CALL 'CBLEXEC' USING EXEC-NAME-LEN EXEC-NAME EXEC-PARM.
    *           0000024     EXIT  METHOD.
                0000025 END  METHOD  INIT-MAKELINE-F.
                0000026
                0000027 END FACTORY.

メソッド名    :INIT_MAKELINE_O
変更<C0 > <C1 >-------------------------------------------------------------------------------
                0000042 PROCEDURE  DIVISION.
    *           0000043     INVOKE  SUPER  'INIT-COLORS-O'.
    *           0000044     COMPUTE  MSGCOUNT  =  0.
    *           0000045     EXIT  METHOD.
                0000046 END  METHOD  INIT-MAKELINE-O.

メソッド名    :DRAWLINE
変更<C0 > <C1 >-------------------------------------------------------------------------------
                0000056 PROCEDURE  DIVISION  USING  I-COLOR.
    *           0000057     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE.
    *           0000058     EVALUATE  MINE
          @     0000059       WHEN  I-COLOR
    *           0000060         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR
                0000061                                  RETURNING  IRO
                0000062         END-INVOKE
    *           0000063           DISPLAY  IRO  'の線を書きました'
          @     0000064       WHEN  OTHER
    *           0000067         COMPUTE  MSGCOUNT  =  MSGCOUNT  +  1
    *           0000068         DISPLAY  '同色の絵の具がパレットにありません'
                0000069     END-EVALUATE.
 Y  *     .     0000070     IF  MSGCOUNT  >  20  THEN
 #  .           0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .           0000072       COMPUTE  MSGCOUNT  =  0
                0000073     END-IF.
 #        @
    *           0000074     EXIT  METHOD.
                0000075 END METHOD  DRAWLINE.
                0000077 END OBJECT.
                0000079 END CLASS MAKELINE.

*************************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         16            3          4           2          5           0      *
*     実行済数         14            1          3           1          5           0      *
*     未実行数          2            2          1           1          0           0      *
*     カバレージ率    87.5%        33.3%       75.0%       50.0%      100.0%       0.0%      *
*************************************************************************************************
```

表 E-1　全ソース表示の説明

項目	内容
翻訳単位名の種別	プログラム名・クラス名・利用者定義関数名
翻訳単位名	翻訳単位の名称
コンパイル日時	翻訳単位を含む原始プログラムをコンパイルした日時
テスト日時	翻訳単位を含む原始プログラムのカバレージ情報を最後に蓄積した日時

項目	内容
変更回数	翻訳単位を含む原始プログラムを変更した回数
テスト回数	翻訳単位を含む原始プログラムを実行した回数
ソース要素名の種別	プログラム名・メソッド名・利用者定義関数名
ソース要素名	ソース要素の名称
変更	Y：原始プログラムの変更された行
	#：原始プログラムの変更によって影響がある行
<C0 >	*：実行が済んだ文
	.：実行していない文
<C1 >	@：実行が済んだ分岐
	.：実行していない分岐
行番号	コンパイラが振り直した行番号
ソーステキスト	原始プログラムの行

(4) 差分ソース表示

```
                              ****************************
                              *     カバレージ情報       *
    COBOL2002 (c)  VV-RR      ****************************        2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :     1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数  :     2

-------------------------------------<差分表示>-------------------------------------------
メソッド名    : DRAWLINE
変更<C0 > <C1 >-------------------------------------------------------------------------------
 Y  *     .     0000070     IF  MSGCOUNT  >  20  THEN
 #  .           0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .           0000072       COMPUTE  MSGCOUNT  =  0
                0000073     END-IF.
 #        @

*************************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>     *
*     対象総数         16            3          4           2          5           0     *
*     実行済数         14            1          3           1          5           0     *
*     未実行数          2            2          1           1          0           0     *
*     カバレージ率    87.5%        33.3%       75.0%       50.0%      100.0%       0.0%     *
*************************************************************************************************
```

(5)　未実行表示

```
                              ******************************
                              *       カバレージ情報       *
     COBOL2002 (c)  VV-RR     ******************************        2003-01-01 09:30:00
-------------------------------------------------------------------------------------------
  クラス名        : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :     1
  テスト日時      : 2003-01-01 09:15:00                                テスト回数  :     2

-----------------------------------<未実行表示>------------------------------------------
メソッド名    : DRAWLINE
変更<C0 > <C1 >---------------------------------------------------------------------------
 Y  *     .     0000070      IF  MSGCOUNT  >  20  THEN
 #  .           0000071        DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .           0000072        COMPUTE  MSGCOUNT  =  0

*******************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         16            3          4            2          5            0     *
*     実行済数         14            1          3            1          5            0     *
*     未実行数          2            2          1            1          0            0     *
*     カバレージ率    87.5%        33.3%      75.0%        50.0%      100.0%       0.0%     *
*******************************************************************************************
```

(6)　差分未実行ソース表示

```
                              ******************************
                              *       カバレージ情報       *
     COBOL2002 (c)  VV-RR     ******************************        2003-01-01 09:30:00
-------------------------------------------------------------------------------------------
  クラス名        : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数    :     1
  テスト日時      : 2003-01-01 09:15:00                                テスト回数  :     2

-----------------------------------<差分未実行表示>--------------------------------------
メソッド名    : DRAWLINE
変更<C0 > <C1 >---------------------------------------------------------------------------
 Y  *     .     0000070      IF  MSGCOUNT  >  20  THEN
 #  .           0000071        DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
 #  .           0000072        COMPUTE  MSGCOUNT  =  0

*******************************************************************************************
*                    <C0>     <差分C0>     <C1>     <差分C1>     <S1>     <差分S1>      *
*     対象総数         16            3          4            2          5            0     *
*     実行済数         14            1          3            1          5            0     *
*     未実行数          2            2          1            1          0            0     *
*     カバレージ率    87.5%        33.3%      75.0%        50.0%      100.0%       0.0%     *
*******************************************************************************************
```

(7)　呼び出し文ソース表示

```
                              ******************************
                              *     カバレージ情報         *
     COBOL2002 (c)  VV-RR     ******************************      2003-01-01 09:30:00
-------------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00                                  変更回数     :     1
  テスト日時     : 2003-01-01 09:15:00                                  テスト回数   :     2

-----------------------------------<呼び出し文表示>-------------------------------------
メソッド名    : INIT_MAKELINE_F
変更<C0 > <C1 >-------------------------------------------------------------------------------
     *          0000022    INVOKE  SUPER  'INIT-COLORS-F'.
     *          0000023    CALL 'CBLEXEC' USING EXEC-NAME-LEN EXEC-NAME EXEC-PARM.

メソッド名    : INIT_MAKELINE_O
変更<C0 > <C1 >-------------------------------------------------------------------------------
    *          0000043     INVOKE  SUPER  'INIT-COLORS-O'.

メソッド名    : DRAWLINE
変更<C0 > <C1 >-------------------------------------------------------------------------------
    *          0000057     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE.
    *     @    0000060         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR


*************************************************************************************************
*                    <C0>    <差分C0>    <C1>    <差分C1>    <S1>    <差分S1>     *
*     対象総数         16           3          4          2          5          0     *
*     実行済数         14           1          3          1          5          0     *
*     未実行数          2           2          1          1          0          0     *
*     カバレージ率    87.5%       33.3%      75.0%      50.0%     100.0%       0.0%     *
*************************************************************************************************
```

付録 E.2　カバレージ情報の表示例（CSV 形式）

(1)　カバレージ統計情報の出力

```
”バージョン”,”出力日時”
”COBOL2002 (X)  04-10”,”2019-04-01 11:00:00”

”プログラム情報ファイル格納先”,”プログラム情報ファイル名”,”ソースカバレージ情報CSV”,”翻訳単位名”,”コンパイル日時”,”テスト日時”,”変更回数”,”テスト回数”,”種別”,”C0対象総数”,”C0実行済数”,”C0カバレージ率”,”差分C0対象総数”,”差分C0実行済数”,”差分C0カバレージ率”,”C1対象総数”,”C1実行済数”,”C1カバレージ率”,”差分C1対象総数”,”差分C1実行済数”,”差分C1カバレージ率”,”S1対象総数”,”S1実行済数”,”S1カバレージ率”,”差分S1対象総数”,”差分S1実行済数”,”差分S1カバレージ率”
”C:¥COBOL¥CBP¥”,”foemat_outline.cbp”,”C:¥COBOL¥CV_CSV¥foemat_outline_source.csv”,”FORMAT_OUTLINE”,”2019-04-01 09:00:00”,”2019-04-01 10:00:00”,”0”,”1”,”C”,”10”,”2”,”20.0”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”
”C:¥COBOL¥CBP¥”,”foemat_outline.cbp”,”C:¥COBOL¥CV_CSV¥foemat_outline_source.csv”,”-”,”2019-04-01 09:00:00”,”-”,”-”,”-”,”-”,”10”,”2”,”20.0”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”
”C:¥COBOL¥CBP¥”,”makeline.cbp”,”C:¥COBOL¥CV_CSV¥makeline_source.csv”,”MAKELINE”,”2019-04-01 09:00:00”,”2019-04-01 10:00:00”,”1”,”2”,”C”,”16”,”12”,”75.0”,”3”,”1”,”33.3”,”4”,”2”,”50.0”,”2”,”1”,”50.0”,”5”,”5”,”100.0”,”0”,”0”,”-”
”C:¥COBOL¥CBP¥”,”makeline.cbp”,”C:¥COBOL¥CV_CSV¥makeline_source.csv”,”-”,”2019-04-01 09:00:00”,”-”,”-”,”-”,”-”,”16”,”12”,”75.0”,”3”,”1”,”33.3”,”4”,”2”,”50.0”,”2”,”1”,”50.0”,”5”,”5”,”100.0”,”0”,”0”,”-”
”C:¥COBOL¥CBP¥”,”outline.cbp”,”C:¥COBOL¥CV_CSV¥outline_source.csv”,”OUTLINE”,”2019-04-01 09:00:00”,”2019-04-01 10:00:00”,”1”,”2”,”C”,”25”,”10”,”40.0”,”1”,”1”,”100.0”,”6”,”3”,”50.0”,”0”,”0”,”-”,”0”,”0”,”-”,”0”,”0”,”-”
”C:¥COBOL¥CBP¥”,”outline.cbp”,”C:¥COBOL¥CV_CSV¥outline_source.csv”,”-”,”2019-04-01 09:00:00”
```

```
,"-","-","-","-","25","10","40.0","1","1","100.0","6","3","50.0","0","0","-","0","0","-","0"
,"0","-"
```

出力内容については，「13.2.6　カバレージ情報の表示（CSV 形式）」の「(1)　カバレージ統計情報の出力」を参照してください。

(2)　ソースカバレージ情報の出力

```
"バージョン","出力日時","コンパイル日時","プログラム情報ファイル格納先","プログラム情報ファイル名"
"COBOL2002 (X)  04-10","2019-04-01 11:00:00","2019-04-01 09:00:00","C:¥COBOL¥CBP¥","makeline.cbp"

"翻訳単位名","ソース要素種別","ソース要素名","未実行ソース","差分ソース","差分未実行ソース","呼び出し文ソース","コメント行","変更","C0","C1","行番号","ソーステキスト"
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,,,,,,"0000021"," PROCEDURE  DIVISION."
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,"ACT",,,"*",,"0000022","     INVOKE SUPER  'INIT-COLORS-F'."
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,"ACT",,,"*",,"0000023","     CALL 'CBLEXEC' USING EXEC-NAME-LEN EXEC-NAME EXEC-PARM."
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,,,,"*",,"0000024"," EXIT  METHOD."
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,,,,,,"0000025"," END  METHOD  INIT-MAKELINE-F."
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,,,,,,"0000026",
"MAKELINE","METHOD","INIT_MAKELINE_F",,,,,,,,,"0000027"," END FACTORY."
"MAKELINE","METHOD","INIT_MAKELINE_O",,,,,,,,,"0000042"," PROCEDURE  DIVISION."
"MAKELINE","METHOD","INIT_MAKELINE_O",,,,"ACT",,,"*",,"0000043","     INVOKE  SUPER  'INIT-COLORS-O'."
"MAKELINE","METHOD","INIT_MAKELINE_O",,,,,,,"*",,"0000044","     COMPUTE  MSGCOUNT  =  0."
"MAKELINE","METHOD","INIT_MAKELINE_O",,,,,,,"*",,"0000045","     EXIT  METHOD."
"MAKELINE","METHOD","INIT_MAKELINE_O",,,,,,,,,"0000046"," END  METHOD  INIT-MAKELINE-O."
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"0000056"," PROCEDURE  DIVISION  USING  I-COLOR."
"MAKELINE","METHOD","DRAWLINE",,,,"ACT",,,"*",,"0000057","     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE."
"MAKELINE","METHOD","DRAWLINE",,,,,,,"*",,"0000058","     EVALUATE  MINE"
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"@","0000059","       WHEN  I-COLOR"
"MAKELINE","METHOD","DRAWLINE",,,,"ACT",,,"*","@","0000060","         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR"
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"0000061","                                 RETURNING  IRO"
"MAKELINE","METHOD","DRAWLINE",,,,,,,"*",,"0000063","           DISPLAY  IRO  'の線を書きました'"
"MAKELINE","METHOD","DRAWLINE","UNEX",,,,,,".","0000064","       WHEN  OTHER"
"MAKELINE","METHOD","DRAWLINE","UNEX",,,,,,".",,"0000067","         COMPUTE  MSGCOUNT  =  MSGCOUNT  +  1"
"MAKELINE","METHOD","DRAWLINE","UNEX",,,,,,".",,"0000068","         DISPLAY  '同色の絵の具がパレットにありません'"
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"0000069","     END-EVALUATE."
"MAKELINE","METHOD","DRAWLINE","UNEX","DIFF","UNEXDIFF",,,"Y","*",".","0000070","     IF  MSGCOUNT  >  20  THEN"
"MAKELINE","METHOD","DRAWLINE","UNEX","DIFF","UNEXDIFF",,,"#",".",,"0000071","       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'"
"MAKELINE","METHOD","DRAWLINE","UNEX","DIFF","UNEXDIFF",,,"#",".",,"0000072","       COMPUTE  MSGCOUNT  =  0"
"MAKELINE","METHOD","DRAWLINE",,"DIFF",,,,,,,"0000073","     END-IF."
"MAKELINE","METHOD","DRAWLINE",,"DIFF",,,,,"#",,"@",,
"MAKELINE","METHOD","DRAWLINE",,,,,,,"*",,"0000074","     EXIT  METHOD."
```

```
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"0000075"," END METHOD  DRAWLINE."
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"0000077"," END OBJECT."
"MAKELINE","METHOD","DRAWLINE",,,,,,,,,"0000079"," END CLASS MAKELINE."
```

出力内容については，「13.2.6　カバレージ情報の表示（CSV 形式）」の「(2)　ソースカバレージ情報の出力」を参照してください。

付録 E.3　カウント情報の表示例

```
                          ****************************
                          *       カウント情報       *
    COBOL2002 (c)  VV-RR  ****************************          2003-01-01 09:30:00
------------------------------------------------------------------------------------------
  クラス名       : MAKELINE
  コンパイル日時: 2003-01-01 09:00:00
  実行日時       : 2003-01-01 09:15:00

------------------------------------------------------------------------------------------
メソッド名    : INIT_MAKELINE_F
  実行回数    ------------------------------------------------------------------------------
              0000021 PROCEDURE  DIVISION.
         1    0000022     INVOKE  SUPER  'INIT-COLORS-F'.
         1    0000023     EXIT  METHOD.
              0000024 END  METHOD  INIT-MAKELINE-F.
              0000025
              0000026 END  FACTORY.

メソッド名    : INIT_MAKELINE_O
  実行回数    ------------------------------------------------------------------------------
              0000042 PROCEDURE  DIVISION.
         2    0000043     INVOKE  SUPER  'INIT-COLORS-O'.
         2    0000044     COMPUTE  MSGCOUNT  =  0.
              0000045     EXIT  METHOD.
              0000046 END  METHOD  INIT-MAKELINE-O.

メソッド名    : DRAWLINE
  実行回数    ------------------------------------------------------------------------------
              0000056 PROCEDURE  DIVISION  USING  I-COLOR.
         6    0000057     INVOKE  SELF  'CHECK-MY-PALETTE'  RETURNING  MINE.
         6    0000058     EVALUATE  MINE
              0000059       WHEN  I-COLOR
         4    0000060         INVOKE  SUPER  'WHATCOLOR'  USING  BY  CONTENT  I-COLOR
              0000061                                  RETURNING IRO
         4    0000063         DISPLAY  IRO  'の線を書きました'
              0000064         WHEN  OTHER
         2    0000067           COMPUTE  MSGCOUNT  =  MSGCOUNT  +  1
              0000068           DISPLAY  '同色の絵の具がパレットにありません'
              0000069     END-EVALUATE.
         6    0000070     IF  MSGCOUNT  >  20  THEN
         0    0000071       DISPLAY  'ﾍﾙﾌﾟを参照して使用方法を確認してください'
              0000072       COMPUTE  MSGCOUNT  =  0
              0000073     END-IF.
         6    0000074     EXIT  METHOD.
              0000075 END  METHOD  DRAWLINE.
              0000077 END  OBJECT.
              0000079 END  CLASS  MAKELINE.
```

付録 F　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 F.1　関連マニュアル

このマニュアルは次のマニュアルと関連があります。必要に応じてお読みください。

- COBOL2002　ユーザーズガイド　（4010-1J-800）
- COBOL2002　言語　標準仕様編　（4010-1J-804）
- COBOL2002　言語　拡張仕様編　（4010-1J-805）
- COBOL2002　Cosminexus 連携機能ガイド　（4010-1J-806）
- COBOL2002　Java プログラム呼び出し機能ガイド　（4010-1J-807）
- COBOL2002　XML 連携機能ガイド　（4010-1J-808）
- COBOL2002　メッセージ　（4010-1J-809）
- COBOL2002 Professional 製品 導入ガイド　（4010-1J-815）
- COBOL2002 Professional Tool Kit COBOL ソース解析ガイド　（4010-1J-816）
- COBOL2002 Professional Tool Kit データ影響波及分析ガイド　（4010-1J-817）
- COBOL2002 Professional Tool Kit 単体テスト支援ガイド　（4010-1J-818）
- ソートマージ　（3020-3-N73）
- Hitachi Code Converter（Windows(R)編）（3020-7-359）
- XMAP3 Version 5 画面・帳票サポートシステム XMAP3 概説　（3020-7-511）
- XMAP3 Version 5 画面・帳票サポートシステム XMAP3 開発ガイド　（3020-7-512）
- XMAP3 Version 5 画面・帳票サポートシステム XMAP3 プログラミングガイド　（3020-7-513）
- XMAP3 Version 5 画面・帳票サポートシステム XMAP3 実行ガイド　（3020-7-514）
- HiRDB Version 10 解説（3020-6-551）
- HiRDB Version 10 コマンドリファレンス（Windows(R)用）（3020-6-559）
- HiRDB Version 10 UAP 開発ガイド（3020-6-560）
- HiRDB Version 10 SQL リファレンス（3020-6-561）
- HiRDB Version 10 メッセージ（3020-6-562）

付録 F.2　このマニュアルでの表記

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

マニュアルでの表記		製品名
SQL Server		Microsoft SQL Server
MS-DOS		Microsoft MS-DOS
MSDN		Microsoft MSDN
Visual Basic		Microsoft Visual Basic
Visual C++		Microsoft Visual C++
Visual Studio		Microsoft Visual Studio
Windows	Windows 10(x64)	Windows 10 Pro 日本語版(64 ビット版)
		Windows 10 Enterprise 日本語版(64 ビット版)
	Windows 11	Windows 11 Pro 日本語版
		Windows 11 Enterprise 日本語版
	Windows Server 2019	Windows Server 2019 Standard 日本語版
		Windows Server 2019 Datacenter 日本語版
	Windows Server 2022	Windows Server 2022 Standard 日本語版
		Windows Server 2022 Datacenter 日本語版
	Windows Server 2025	Windows Server 2025 Standard 日本語版
		Windows Server 2025 Datacenter 日本語版

このマニュアル中では，「このシステム」と表現している場合，「COBOL2002」を示しています。

また，このマニュアルは，製品種別によって相違点があります。本文中での製品種別ごとの表記を次に示します。

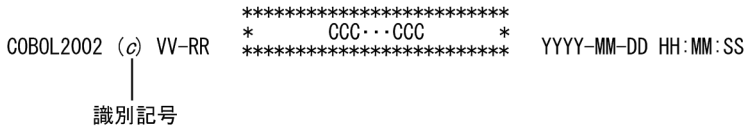
マニュアルでの表記			該当する製品の形名
COBOL2002	Windows COBOL2002	Windows(x86) COBOL2002	P-2636-2354 P-2436-5354 P-2636-3354 P-2436-6354 P-2636-4354 P-2636-7354
		Windows(x64) COBOL2002	P-2936-2354 P-2936-5354 P-2936-6354 P-2936-7354

- COBOL2002 Developer Professional と COBOL2002 Developer Professional(64)を総称して，COBOL2002 Developer Professional と表記しています。

また，このマニュアルでは，各製品を次のように表記しています。

マニュアルでの表記	製品名
Pervasive.SQL	Pervasive.SQL
	Pervasive.SQL 2000
XMAP3	XMAP3 Server
	XMAP3 Developer Version 5
	XMAP3 Server Runtime Version 5
	XMAP3 Client Runtime Version 5
Windows SDK	Windows SDK for Windows 11

- XMAP3 の製品を区別する必要がある場合は，それぞれの製品名称を表記しています。
- 日立 COBOL2002 のことを日立 COBOL2002，または単に COBOL2002 と表記しています。
- 日立 COBOL85 のことを日立 COBOL85，または単に COBOL85 と表記しています。また，プラットフォームを明確にする必要がある場合は，「PC COBOL85」「VOS3 COBOL85」のように表記しています。
- Enterprise JavaBeans を EJB と表記しています。
- 特に断り書きがない場合，プログラム定義，関数定義，およびメソッド定義を総称して「プログラム」と表記しています。
- リストのヘッダ部に表示される COBOL2002 の識別記号を次のように表記しています。識別記号以外については，「13.2.5　カバレージ情報の表示（テキスト形式）」，「13.3.3　カウント情報の表示」を参照してください。



識別記号（c）	内容
X と表示された場合	Windows(x86) COBOL2002 であることを示します。
E と表示された場合	Windows(x64) COBOL2002 であることを示します。

- CSV ファイルのヘッダ部の「バージョン」に表示される COBOL2002 の識別記号を（c）と表記しています。識別記号以外については，「13.2.6　カバレージ情報の表示（CSV 形式）」を参照してください。

識別記号（c）	内容
X と表示された場合	Windows(x86) COBOL2002 であることを示します。

識別記号（c）	内容
E と表示された場合	Windows(x64) COBOL2002 であることを示します。

- コンパイラオプションの説明では，次の表記を使用します。

  「XXX オプション」，または単に「XXX」とオプション名が表記されている場合

  XXX オプションについて，サブオプションの組み合わせを含む，すべての場合を意味します。

  「XXX,YYY オプション」，または単に「XXX,YYY」とサブオプションを含めたオプション名が表記されている場合

  XXX,YYY オプションだけの場合を意味します。

  「XXX コンパイラオプション」と表記されている場合

  リンカオプションなど，ほかのオプションと明確に区別する必要がある場合を意味します。

  （例 1）

  「-Compile オプション」または「-Compile」と記載している場合，-Compile オプションのサブオプションの組み合わせすべて（-Compile,CheckOnly／-Compile,NoLink）を意味します。

  （例 2）

  「-Compile,CheckOnly オプション」または「-Compile,CheckOnly」と記載している場合，-Compile,CheckOnly だけを意味します。

付録 F.3　KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

付録 G　用語解説

(英字)

BOM（バイトオーダマーク）

ファイルの先頭に付加された，Unicode の表現形式を表す情報。

COBOL2002 では，テキスト編成ファイルに対してこの情報を付加する。本文中では，Unicode シグニチャと表記する。

C0 メジャー情報

全体の実行文の数に対する実行済み文の数の割合を示す情報。

C1 メジャー情報

全体の分岐の数に対する実行済みの分岐の数の割合を示す情報。

COBOL2002 Developer Professional

COBOL2002 Net Developer と COBOL2002 Professional Tool Kit を一つにしたプログラムプロダクトのこと。Windows の稼働環境に応じて，次の 2 製品がある。

- COBOL2002 Developer Professional（32bit 版）
- COBOL2002 Developer Professional(64)（64bit 版）

このマニュアルでは，上記の製品を総称して，COBOL2002 Developer Professional と表記する。

COBOL エディタ

COBOL2002 に内蔵されているテキストエディタ。COBOL 構文によるテキストの色分け，プログラムテンプレートなど，COBOL 編集に適した機能を持っている。

COBOL サービスルーチンファイル

COBOL エディタでサービスルーチン名として扱う語を登録したファイル。

CSV 編成ファイル

CSV（Comma Separated Values）形式でデータが記述されたファイルのこと。表計算プログラムファイルともいう。

CSV 形式とは，表計算プログラムやリレーショナルデータベースでデータを扱えるテキストデータの形式をいう。データの区切りをコンマ（,），レコードの区切りを改行で表す。レコードは可変長形式になる。コンマで区切られた個々のデータは，セルと呼ばれる。

HTML（Hyper Text Markup Language）

インターネット経由で閲覧できる Web ページを作成するための言語。テキストファイル中にタグと呼ばれる制御文字を埋め込むと，文字書式の設定や画像の埋め込み，ほかのページへのリンクなどを表現できる。

IVS（Ideographic Variation Sequence/Selector）

漢字を表す Unicode の直後に Variation Selector と呼ばれるコードを付加し，漢字の「異体字」を表現する方法のこと。

COBOL2002 では，用途が DISPLAY および NATIONAL の項目で使用する。

OLE オブジェクト

OLE2 オートメーション機能を使用するときの操作の対象。OLE オブジェクトは，OLE プロパティと OLE メソッドを持つ。

OLE オブジェクト参照データ項目

USAGE 句で用途は OLE オブジェクト参照（OBJECT REFERENCE OLE）と指定されたデータ項目。インスタンス化された OLE オブジェクトを指すポインタを格納する。

OLE プロパティ

OLE オブジェクトの状態または属性。

OLE メソッド

OLE オブジェクトに対する操作。

S1 メジャー情報

全体の呼び出し文（CALL 文，OLE メソッド操作を除く INVOKE 文）の数に対する実行済みの呼び出し文の数の割合を示す情報。

SELF

メッセージの受け手のオブジェクトを参照するための名前のこと。SELF が参照するオブジェクトを SELF オブジェクトという。メソッド中で使用される SELF は，そのメソッドの呼び起こし対象のオブジェクト自身のことを表す。

TD コマンド

テストデバッグで使用するコマンド。

UAC（User Account Control）

管理者ユーザとしてサインインしても，既定ではアプリケーションの実行権限は標準権限に下げられる。管理者権限を必要とするアプリケーションを起動しようとすると，システムは昇格ダイアログボックスを表示して，ユーザの確認を求める。ユーザが実行を許可した場合だけ，アプリケーションは管理者権限で実行されることになる。これによって，ユーザが知らない間

に悪意のあるソフトウェアをインストールされたり，システム設定を変更されたりすることを防止できる。

UCS-2（Universal multi-octet Character Set 2）

符号化文字集合の一つの形式。1 文字を 2 バイトで表現する。

COBOL2002 では，用途が DISPLAY および NATIONAL の項目でサポートする。

UCS-4（Universal multi-octet Character Set 4）

符号化文字集合の一つの形式。1 文字を 4 バイトで表現する。

COBOL2002 では，用途が DISPLAY および NATIONAL の項目で UCS-4 の範囲をサポートする。

Unicode

1 バイトでは表現できない文字セットを含めあらゆる文字セットをサポートした文字コード。代表的な符号化文字集合として UCS-2，UCS-4 がある。代表的なエンコーディングスキーマとして UTF-8，UTF-16 がある。

COBOL2002 では，UCS-4 の範囲（UCS-2 の範囲を含む）をサポートする。

本文中では，符号化文字集合，およびエンコーディングスキーマを文字コードと表記する。

UTF-16(16-bit UCS Transformation Format)

エンコーディングスキーマの一つの形式。1 つのコード単位を 2 バイトとし，1 文字を 1 コード単位（2 バイト），または 2 コード単位（4 バイト）で表現する。UTF-16 では 2 バイトのコード単位の 1 バイト目を先に書くビッグエンディアン形式（UTF-16BE）と 1 バイト目を後に書くリトルエンディアン形式（UTF-16LE）がある。

COBOL2002 では，用途が NATIONAL の項目で UCS-4 の範囲（UCS-2 の範囲を含む）をサポートする。

UTF-8（8-bit UCS Transformation Format）

エンコーディングスキーマの一つの形式。ASCII 文字を 1 バイト，日本語文字を 3~8 バイト，半角かたかなを 3 バイトで表現する。

COBOL2002 では，用途が DISPLAY の項目で使用する。

WRP（Windows Resource Protection）

Windows システムの安定性，および信頼性を向上させるための機能。特定の OS ファイル，フォルダ，レジストリキーなど，Windows の読み取り専用リソースを保護する。

（ア行）

アニメーション

プログラムの実行状況を画面上に追跡表示すること。

入れ子のプログラム

原始プログラムが入れ子構造になっている場合の，内側のプログラムのこと。内側のプログラム，または内部プログラムともいう。

インスタンスオブジェクト（instance object）

あるクラスに属するオブジェクトのこと。「BASE クラス」のファクトリオブジェクトのメソッド「NEW クラス」の呼び起こしによって生成され，実行単位の終了か，ガーベジコレクションによって消滅させられる。

インスタンスメソッド

オブジェクトデータを操作するためのメソッド。

インタフェース（interface）

メッセージの送り手から見えるオブジェクトの部分のこと。

例えば，現金支払い機を一つのオブジェクトとすると，「引き出し」「預け入れ」などのメニューがインタフェースに当たり，そのサービスを実現するための内部的な仕組みが実装に当たる。インタフェースは，メソッド原型の集合である。つまり，どのようなメソッド原型を持つかによって，オブジェクトが持つインタフェースの型が決まる。インタフェースは，クラスとは別に定義できる。

内側のプログラム

原始プログラムが入れ子構造になっている場合の，内側のプログラムのこと。入れ子のプログラム，または内部プログラムともいう。

オブジェクト（object）

データとそのデータを操作するメソッド（手続き）を一体化させたモジュール。オブジェクトは，ある特定の仕事を受け持ち，所定のメッセージに対してメソッドを動作させることで問題を処理する。オブジェクト指向のプログラムでは，オブジェクト同士がメッセージのやり取りすることによって，問題が処理される。

オブジェクト参照データ項目

オブジェクトを参照するために使用される基本データ項目。

オブジェクトデータ

オブジェクトが独自に持つデータ。インスタンスメソッドによって操作される。

オブジェクトプロパティ

PROPERTY 句を指定することで，別のクラスからでも直接に操作できると宣言されたデータ項目。

オプション

コンパイラやコマンドへの動作を指示するためのもの。

（カ行）

開発マネージャ

プロジェクトの定義やツールの起動などを支援する，COBOL2002 の開発環境を統合する機能。

COBOL2002 Professional Tool Kit の機能である，COBOL ソース解析，データ影響波及分析用データベースの作成，COBOL ソースファイルを指定して単体テスト支援のテストプロジェクトの作成ができる。

外部スイッチ

ハードウェアまたはソフトウェアの装置であって，作成者によって定義，命令され，二者択一の状態のどちらかであることを示すために使用される。

外部プログラム

原始プログラムが入れ子構造になっている場合の，最も外側のプログラムのこと。このマニュアルでは，外部プログラムのことを最外側のプログラムと呼ぶ。

カウント情報

プログラム中の文の実行回数をカウントして数値で表したもの。

型（type）

ある集合の中の個々を，何によって識別するかを明示するもの。例えば，プログラムを構成する要素をデータごとに識別する場合は，データ型となる。

活性化されるプログラム

CALL 文の対象となるプログラムで，実行時に呼び出し元プログラムと組み合わされて 1 個の実行単位となる。このマニュアルでは，活性化されるプログラムのことを呼び出し先のプログラムと呼ぶ。

カバレージ情報

テストの進捗状況を数値で表したもので，C0 メジャー情報，C1 メジャー情報，S1 メジャー情報，およびこれらの差分情報がある。

環境変数

コンパイラ環境へのオプションを変数として設定や実行可能ファイルおよびその実行環境へのオプションを変数として設定しておくもの。

管理者権限

管理者ユーザおよびビルトイン管理者（Administrator アカウント）に与えられる権限。

管理者ユーザ

Windows の Administrators グループに属するアカウント。ビルトイン管理者（Administrator アカウント）とは区別される。

クラス（class）

ある共通の性質を持つオブジェクトを一つのグループにまとめて定義したもの。クラスでは，それに属するオブジェクトのデータフォーマットおよびメソッドを定義することで，そのクラスに属するオブジェクトの型を規定する。

継承(inheritance)

スーパクラスから，その性質（メソッド）をサブクラスが受け継ぐ仕組みのこと。

構成管理ツール

プログラム開発チーム内で各種ファイルのバージョン管理のために使用するツールのこと。開発マネージャのメニューから開くダイアログボックスで，構成管理ツールの動作を設定できる。

構成管理ツールの設定は，COBOL2002 Developer Professional でだけ使用できる。

コンパイル

原始プログラムを翻訳すること。

COBOL2002 のコンパイルには，次の四つの方法がある。

- ビルド
- リビルド
- クイックビルド
- ファイル単位のコンパイル

（サ行）

最外側のプログラム

原始プログラムが入れ子構造になっている場合の，最も外側のプログラムのこと。外部プログラムともいう。

サブクラス（subclass）

クラスの階層関係の中で，継承する側のクラスのこと。

差分カバレージ情報

原始プログラムの修正によって影響を受けた部分に対するカバレージ情報。

サロゲート

UTF-16 の拡張で，2 つのコード単位（4 バイト）で 1 文字を表す機能。この 2 つのコード単位の組み合わせのことをサロゲートペアと呼ぶ。

COBOL2002 では，用途が NATIONAL の項目で使用する。

実行可能プログラム

呼び出し元のプログラムと呼び出し先のプログラムをコンパイル，リンクし，一つの実行できるプログラムにしたもの。

実行時要素

プログラム定義，メソッド定義，および関数定義の総称。このマニュアルでは，実行時要素のことをプログラムと呼ぶ。

昇格ダイアログボックス

Windows の UAC で表示されるダイアログボックス。アプリケーションの実行権限を管理者権限に昇格しようとする場合，ユーザに権限昇格の確認を求めるダイアログボックス。

スーパクラス（superclass）

クラスの階層関係の中で，継承される側のクラスのこと。

選択マージン

COBOL エディタのソースウィンドウ中の行の左余白。この領域を選んで行を選択できる。

（タ行）

単体テスト

プログラム（コンパイル）単位のテスト。

定義ファイル

画面定義やファイル／レコード定義などの定義情報を格納するファイル。

適合（conformance）

メッセージの送り先のオブジェクトが適切かどうかを調べるための概念。適合は，オブジェクトが持つインタフェースに基づいてチェックされる。あるインタフェース B を持つオブジェク

トに対するメッセージが，別のインタフェース A を持つオブジェクトでも対応できる場合，インタフェース A はインタフェース B に適合しているという。インタフェース A がインタフェース B に適合する場合，インタフェース B に適合するインタフェース型のオブジェクトが使用できれば，インタフェース A に適合するインタフェース型のオブジェクトも使用できる。適合は，コンパイル時または実行時にチェックされる。これによって，オブジェクトの誤用が避けられるだけでなく，同じメッセージを異なるクラスのオブジェクトに送れるようになる。

動的アクセス

大記憶ファイルで，同一 OPEN 文の範囲で順アクセスと乱アクセスの切り替えができるアクセス法 。

登録集原文

COBOL プログラム中でよく利用される標準化した手続き，ファイル記述，レコード記述，または完全な一つのプログラムなどを，コンパイルするプログラムとは別のファイルに登録したもの。

トップレベルプロジェクト

プロジェクトの依存関係で，親プロジェクト（上位のプロジェクト）を持たないプロジェクトのこと。

開発マネージャでは，トップレベルプロジェクトに対してビルドを実行することによって，必要な下位のプロジェクトをまとめてビルドできる。

トラッキングバー

COBOL エディタのソースウィンドウのルーラー上に表示され，現在のマウスカーソルの位置を示す。

トレース

実行した命令を一つ一つ追跡し，実行履歴を確認すること。

（ナ行）

内部プログラム

原始プログラムが入れ子構造になっている場合の，内側のプログラムのこと。このマニュアルでは，内部プログラムのことを内側のプログラム，または入れ子のプログラムと呼ぶ。

（ハ行）

バイトオーダ

2 バイト以上のデータの記録を行う順序のこと。例えば，0x1234 のデータを 0x1234 のように最上位のバイトから順番に記録する方式をビッグエンディアン，0x3412 のように最下位の

バイトから順番に記録する方式をリトルエンディアンという。2 バイトの UTF-16 は，バイトオーダを意識する。

バリアントデータ項目

USAGE 句で用途はバリアント（VARIANT）と指定されたデータ項目。INVOKE 文や SET 文で使用する VARIANT 値を指すポインタを格納するためのデータ領域。

標準権限

標準ユーザに与えられる権限。故意に，または誤ってシステムに変更を加える許可を持たない。

標準ユーザ

Windows の Users グループに属するアカウント。

表要素

表中の繰り返す項目の組に属する 1 個のデータ項目。

ビルド

プロジェクトに登録されたソースファイルや定義ファイルから最終生成ファイル（実行可能ファイル，DLL，標準ライブラリなど）を生成する方法。ビルドの対象となるファイルは，変更のあったファイルだけである。

ファイル単位のコンパイル

ソースファイルをコンパイルしてオブジェクトファイルを生成する方法。また，定義ファイルからはソース（登録集原文）を生成する。

ファクトリオブジェクト

一つのクラス中のすべてのオブジェクトが共有するデータ（ファクトリデータ）とそれを操作するためのメソッド（ファクトリメソッド）から成るオブジェクト。ファクトリオブジェクトは，実行単位の開始によって生成され，実行単位が終了すると消滅させられる。

ファクトリデータ

一つのクラス中のすべてのオブジェクトが共有するデータのこと。ファクトリメソッドによって操作される。

ファクトリメソッド

ファクトリデータを操作するためのメソッド。

ブックマーク

COBOL エディタのソースウィンドウ中に表示される目印。

プログラム

プログラム定義，メソッド定義，および関数定義の総称。実行時要素ともいう。

プロジェクト

実行可能ファイルやダイナミックリンクライブラリ（DLL）を作成するための資源を一括して管理するための概念。一つのプロジェクトは，一つの実行可能ファイル，または DLL に対応している。

プロジェクト情報ファイル

プロジェクトマスタに含まれるプロジェクトの情報を出力したファイル。プロジェクト情報ファイルを利用すると，プロジェクトマスタ間でプロジェクトの情報を流用できる。

プロジェクトマスタ

複数のプロジェクトを管理するための概念。プロジェクトマスタにはプロジェクト間の親子関係や，すべてのプロジェクトに共通する設定を定義する。

プロジェクトマスタ情報

プロジェクトマスタに関する情報のこと。開発マネージャで設定する。複数の開発マネージャで設定内容を使用できるように，プロジェクトマスタファイルに出力できる。

プロジェクトマスタ情報ファイル

プロジェクトマスタの情報を編集するために使用する CSV 形式のファイル。

プロジェクトマスタファイル

プロジェクトマスタの情報を保管するファイル。

（マ行）

マーカ

COBOL エディタのソースウィンドウに表示される縦の直線。

マルチスレッド

プログラム内の仕事を，スレッドという単位に分けて実行する方式。複数のプログラムを並列に実行できる。

見た目幅

Unicode 機能の組み込み関数で使用する，文字の見た目の幅。半角文字の幅は 1，全角文字の幅は 2 として扱う。

Unicode 機能の組み込み関数で，文字を見た目幅で数える場合に使用する。

メインフレーム

大規模な業務システムなどに用いられる汎用大型コンピュータ。

メソッド（method）

オブジェクトが提供するサービス（メソッド原型という）とそのサービスの実装を定義した手続き。オブジェクト間のやり取りは，すべてメソッドを通して行われる。メソッドには，ファクトリメソッドおよびインスタンスメソッドの 2 種類がある。

メソッド原型(method prototype)

メソッドを定義する際，メソッドが提供するインタフェースを定義した部分。メソッド原型は，メソッド名と，パラメタおよび戻り値から成る。

メソッドの呼び起こし（method invocation）

オブジェクトに対してメソッドを呼び起こすこと。COBOL2002 では，INVOKE 文によってメソッドを呼び起こす。

（ヤ行）

ユーザキーワードファイル

COBOL エディタでユーザキーワードとして扱う語を登録したファイル。

呼び出し先プログラム(called program，subprogram)

CALL 文の対象となるプログラムで，実行時に呼び出し元プログラムと組み合わされて 1 個の実行単位となる。

呼び出し元プログラム(calling program)

CALL 文を実行してほかのプログラムを呼ぶプログラム。

（ラ行）

リビルド

プロジェクトに登録されたソースファイルや定義ファイルから最終生成ファイル（実行可能ファイル，DLL，標準ライブラリなど）を生成する方法。リビルドの対象となるファイルは，プロジェクトに登録されたすべてのファイルである。

索引

記号

.bkf 28
.cbf 29
.cbl 28
.cbp 188
.cbr 282
.cbs 194
.cbw 28, 195
.cob 29
.cxc 28
.cxd 28
.def 197
.dll 28
.dtd 28
.exe 28, 282
.flf 28
.hmf 28
.hmp 28, 66
.ico 28
.lib 28
.obj 28
.ocb 29
.ocf 29
.pkg 28
.rc 198
.rdf 28
.rep 28
.utsproj 28
.wdf 28
.xml 28
#INCLUDE 690
#INCLUDE コマンド 766
#OPTION 691
-? 248
-BigEndian 227
-Bin1Byte 236
-BinExtend 240
-Cblctr 213
-CBLVALUE 244
-CheckUninitData 253
-CmAster 214
-CmDol 215
-Comp5 215
-Compati85 209
-CompatiM7 207
-CompatiV3 207
-Compile 196
-CVInf 189, 699
-DebugCompati 186
-DebugData 187
-DebugInf 185
-DebugLine 185
-DebugRange 190
-DefFile 197
-Define 245
-Details 246
-DigitsTrunc 213
-Dll 175
-DllInit 182
-DoubleQuote 226
-DynamicLink 200
-EquivRule 239
-ErrSup 236
-FunctionECSup,CodeConvErr 253
-H8Switch 212
-Help 248
-IconFile 198
-IgnoreAPPLY,FILESHARE オプション 233
-IgnoreLCC 214
-IntResult,DecFloat40 250
-IsamExtend 179
-JPN 237
-Lib 196
-Link 200

-LiteralExtend 251
-LowerAsUpper 243
-Main 174
-MainNotCBL 182
-ManifestFileExt 201
-MaxDigits38 250
-MinusZero 240
-MultiThread 181
-NumAccept 180
-NumCsv 181
-OldForm 247
-OpenTP1 180
-Optimize 183
-OutputFile 199
-PortabilityCheck 232
-RDBTran 178
-Repository 244
-RepositoryCheck 245
-ResrcFile 198
-ScreenSpeed 184
-SimIdent 193, 490
-SimMain 191, 490
-SimSub 192, 490
-SpaceAsZero 252
-SQL 177
-SQLDisp 178
-SrcList 234
-Std2002 204
-Std85 203
-StdCall 195
-StdCallFile 195
-StdMIA 202
-StdVersion 205
-Switch 218
-TDInf 188, 490
-TestCmd 190
-TruncCheck 241
-UniEndian 249
-UniObjGen 248
-UscoreStart 239
-V3ConvtName 217
-V3Rec 223
-V3RecEased オプション 233
-V3RecFCSpace 224
-V3Spec 216
-VOSCBL 229
-XMAP 179

数字

16 進数表示 504
1 バイトの 2 進項目 237
64bit アプリケーションの開発 807
80 バイトを超える行の検索 348
80 バイトを超える行を検索
　COBOL エディタウィンドウのメニュー 317
80 バイトを超える行を知らせる 367

A

ALLOCATE AREA 648
APPLY FORMS-OVERLAY 句 179
ASCII 219
ASSIGN ADDRESS（アドレスの取得） 649
ASSIGN CASECODE 650
ASSIGN DATA 651
AUTO 405

B

BACKGROUND-COLOR 404
Back Space
　文字列の編集 340
BELL 405
BLANK 405
BLANK WHEN ZERO 405, 444
BOM 882
Btrieve（Pervasive.SQL）による索引編成ファイル 179

C

C0 メジャー 702
C0 メジャー情報 882
C1 メジャー 702
C1 メジャー情報 882
CALL 定数 239, 244
CASE 651
CBL_RDBSYS 260
CBL_UNINITDATA_BREAKOFF 260
cbl2kinfoextra コマンド 785
cblca2k コマンド 754
cblcc2k コマンド 750
cblcl2k コマンド 747
cblcn2k コマンド 757
CBLCOPT 261
CBLCOPT2002 261
cblcv2k コマンド 742
CBLERRMAX 262
CBLFIX 262
CBLFIXEDFORMLINE 262
CBLFREE 263
CBLINITVALUE 263
CBLLIB 264
CBLLINKER 264
CBLLINKINTERVAL 265
CBLMANIFESTTOOL 266
CBLPIDIR 266
CBLPIDIR(GUI モードのカバレージ) 729
CBLPIDIR(GUI モードのテストデバッグ) 553
CBLPIDIR(TD コマンド格納ファイルの出力) 764
CBLPIDIR(バッチモードのカウント情報の表示) 759
CBLPIDIR(バッチモードのカバレージ情報の操作) 756
CBLPIDIR(バッチモードのカバレージ情報の蓄積) 744
CBLPIDIR(バッチモードのカバレージ情報の表示(CSV 形式)) 753
CBLPIDIR(バッチモードのカバレージ情報の表示(テキスト形式)) 749
CBLPIDIR(バッチモードのテストデバッグ) 624
CBLREP 267
CBLRESOURCECOMPILER 267
CBLSYSREP 268
CBLTAB 268
cbltd2k コマンド 625
CBLTDEXEC 745, 760
CBLUNINITDATA_EXITPROGRAM 268
CBLUNINITDATA_OUTRESULTLIST 269
CBLUNINITDATA_OUTRESULTLISTDIR 270
CBLV3UNICODE 271
CBLVALUE 244, 270
CBL-CTR 特殊レジスタ 213
ccbl2002 コマンドに指定するオプション列 261
ccbl2002 コマンドのヘルプ 248
ccbl コマンドに指定するオプション列 261
CHARACTER TYPE 句 179
CHAR TYPE 444
COBOL2002 Developer Professional 26, 882
COBOL2002 Net Developer 26
COBOL2002 仕様 204
COBOL2002 情報抽出ツール 782
COBOL2002 情報抽出ツールで使用するファイルと出力リスト 783
COBOL2002 情報抽出ツールの使用手順 785
COBOL2002 情報抽出ツールの入出力構成 783
COBOL2002 情報抽出ツールの入力ファイルと抽出できる情報の対応 782
COBOL2002 情報抽出ツールのメッセージ 804
COBOL2002 のレコード定義 382
COBOL エディタ 309, 882
COBOL エディタウィンドウ 315
COBOL エディタウィンドウのメニュー
  インポート 317
  エクスポート 317
COBOL エディタウィンドウのメニュー項目一覧 844
COBOL エディタが提供しているプログラムテンプレート 331
COBOL エディタの起動 313
COBOL エディタの終了 314
  COBOL エディタウィンドウのメニュー 316

COBOL エディタのプロセス　313
COBOL サービスルーチンファイル　369, 882
COBOL サービスルーチンファイル，ユーザキーワードファイルの作成　371
COBOL サービスルーチンファイルの形式　371
COBOL 生成　426, 458
　画面定義ウィンドウのメニュー　398
　ファイル／レコード定義ウィンドウのメニュー　437
COBOL 生成許可　428, 465
　画面定義ウィンドウのメニュー　398
　ファイル／レコード定義ウィンドウのメニュー　437
COBOL ソース　372
COBOL ソースの形式　310
COBOL ソース表示形式の切り替え
　COBOL エディタウィンドウのメニュー　318
COBOL ソースファイル　28
COBOL ソースファイルの生成　426
COBOL ソースを構文で色分け表示する　362
COBOL 属性　406
COLOR　404
COMMIT　178
COMP-5　215
COPY 原文名　441
CSV 形式で出力　466
　ファイル／レコード定義ウィンドウのメニュー　437
CSV ファイル　466
CSV ファイルに指定できるファイルの種別　128
CSV ファイルに指定できるプロジェクトの種類　128
CSV ファイル入出力時のエラーチェック　130
CSV ファイルのアクセス権に関するチェック　130
CSV ファイルのエントリ行　113
CSV ファイルの使用方法　109
CSV ファイルのセクション　112
CSV ファイルのセクションの形式　113
CSV ファイルの内容に関するチェック　130
CSV ファイル編集時の規則　110
CSV 編成ファイル　181, 882
CUI モードのユーザプログラム　196

D

DC シミュレーション　518
DC シミュレーションの設定　686
Delete
　画面定義の編集　410
　文字列の編集　340
DISPLAY BREAK（中断点の表示）　652
DISPLAY COMMENT　653
DISPLAY DATA　653
DISPLAY FACTORY　660
DISPLAY FLOW　658
DISPLAY OBJECT　660
DISPLAY POINT（現在位置の表示）　666
Dll の属性を cdecl にする　176
Dll の属性を fastcall にする　176
Dll の属性を stdcall にする　176
DLL ファイルだけの連動実行について　613

E

EBCDIC　219
EBCDIK　219
ENTER-CHECK　405
EnvironmentDataInf セクション　126
ERASE　405
ESC／P モード印刷機能　288
EXTERNAL　440, 445

F

F10 キーの設定　296
FileInf セクション　121
FOREGROUND-COLOR　404
FREE AREA　649
FROM　406
FULL　406

G

GDI モード印刷機能　287
GETPROPERTY　667
GLOBAL　440, 445

GO 669
GO END 670
GO EOP 670
GO ERROR 670
GO INVALID 670
GRID 404
GUI の操作のための機能 526
GUI モードによるカバレージ 728
GUI モードによるテストデバッグ 549
GUI モードによるテストデバッグの概要 550
GUI モードのユーザプログラム 196

H

HELP 406
HELP 属性 406
HIGHLIGHT 404
HIGH-INTENSITY 404
HTML 883
HTML ファイル 77

I

IF 671
ISAM 情報 453
ISAM 情報の設定ダイアログボックス 453
ISAM ファイル 453
ISAM ファイル生成 457
　ファイル／レコード定義ウィンドウのメニュー 437
IVS 883

J

JIS 仕様 203
JUST 405, 444

L

LEFTLINE 404
LETTER（入力チェック用） 405
LETTER（表示用） 404
LOWLIGHT 404

M

MIA 仕様 202
MIA バージョン 1.3 仕様の範囲外チェック 202
MIA バージョン 1.4 仕様の範囲外チェック 203
MIOS7 COBOL85 互換機能 808

O

ODBC インタフェース機能 177
ODBC 定義
　開発マネージャのウィンドウのメニュー 34
ODBC レコード定義生成 381
OLE2 オートメーションのテストデバッグ 540
OLE オブジェクト 883
OLE オブジェクト参照データ項目 883
OLE プロパティ 883
OLE プロパティ指定 644
OLE プロパティの値の表示 541
OLE メソッド 883
OpenTP1 180
OptionDataInf セクション 124
OVERLINE 404

P

PCA 829
PICTURE 406
ProfileInf セクション 115
ProjectInf セクション 117
ProjectMasterInf セクション 113
ProjectRelationInf セクション 114
PROMPT 405

Q

QUIT 672

R

RegisterInf セクション 116
REPEAT 672
REQUIRED 405
RESET 405

RESET BREAK 676
RESET FLOW 677
RESET LOG 678
RESET PRINT 679
RESET QUALIFICATION 680
RESET TRACE 682
RESET WATCH 683
REVERSE 404
REVERSE-VIDEO 404
ROLLBACK 178
RULE IS OVER 404
RULE IS SQUARE 404
RULE IS UNDER 404
RULE IS VERTICAL 404

S

S1 メジャー 702
S1 メジャー情報 883
SECRET 404
SECURE 404
SELECT ACTION 673
SELF 883
SET BREAK 673
SET BREAK コマンド（節名） 766
SET BREAK コマンド（プログラム名） 767
SET BREAK コマンド（文） 766
SET FLOW 677
SET LOG 678
SET PRINT 679
SET QUALIFICATION 680
SET QUALIFICATION コマンド 765
SET TRACE 680
SET WATCH 682
SIGN 404, 444
SIMULATE DC 686
SIMULATE FILE 687
SIMULATE FILE コマンド（INPUT／IO モード） 770
SIMULATE FILE コマンド（OUTPUT／EXTEND モード） 771
SIMULATE MAIN 688
SIMULATE MAIN コマンド 768
SIMULATE SUB 688
SIMULATE SUB コマンド 769
SOURCE 406
SQL データ型 384
SQL のデータ型から生成される COBOL2002 のレコード定義 385
stdcall 呼び出し規約 195
stdcall 呼び出し指示ファイル 28, 76, 195
STEP IN 684
STEP OVER 684
STEP TO 685
STOP 686
SYNC 445
System 指定メインプログラム 54, 174

T

TD コマンド 627, 883
TD コマンド格納ファイル 190, 763
TD コマンド格納ファイルの出力先 764
TD コマンド格納ファイルの取り込み 690
TD コマンド格納ファイルの入力 580
　テストデバッグウィンドウのメニュー 559
TD コマンド格納ファイルの読み込み 525
TD コマンド群 639
TD コマンド生成機能 762
TD コマンド生成機能とは 475
TD コマンド入力ウィンドウ 572
　テストデバッグウィンドウのメニュー 561
TD コマンド入力ウィンドウの設定 576
TD コマンドの一覧 645
TD コマンドの行 628
TD コマンドの形式 628
TD コマンドの詳細 648
TD コマンドの生成例 774
TD コマンド名 628
TD 利用者定義語 635
TEST 689

TO 406

U

UAC 883
UCS-2 884
UCS-4 884
UNDERLINE 404
Unicode 534, 884
Unicode 機能 476
Unicode 機能に必要な環境変数 477
Unicode 機能に必要なコンパイラオプション 477
UNIX 形式の改行コードで保存する 366
USING 405
UTF-16 884
UTF-8 884

V

V3 指定メインプログラム 54, 174
VALUE 406
VALUE 句のないデータ項目の初期値 271

W

Windows(x64) COBOL2002 開発マネージャの機能 813
Windows(x64) COBOL2002 だけで使用できる
コンパイラオプション 815
Windows(x64) COBOL2002 で使用できない
移行のコンパイラオプション 810
機能 808
コンパイラオプション 816
最終生成物の種類のコンパイラオプション 809
製品連携のコンパイラオプション 809
ファイルの実行時環境変数 810
リンクのコンパイラオプション 810
Windows(x64) COBOL2002 について 808
Windows(x64) COBOL2002 の実行支援の機能821
Windows のプログラム一覧にある［COBOL2002］メニューからの開始 556
Windows リソース保護 828
WRP 828, 884

X

XMAP3 179
XML データ定義ファイル 28
XML 文書型定義ファイル 28

Z

ZERO-FILL 405

あ

アイコン 198
アイコンファイル 28
新しいウィンドウ
COBOL エディタウィンドウのメニュー 318
開発マネージャのウィンドウのメニュー 36
新しいファイルの種類の新規登録ダイアログボックス 103
新しいファイルの種類の登録ダイアログボックス 98
あとに指定した方のオプションが有効となる場合 157
アドレス名 444
アニメーション 599, 885
アニメーション表示 502
アニメーション抑止の解除 600
アニメーション抑止の設定 599
アプリケーションの主プログラム 174

い

異常終了時要約情報リスト 186
依存 81
依存関係更新
開発マネージャのウィンドウのメニュー 34
依存関係の設定 73
依存ファイルの仮登録 72
開発マネージャのウィンドウのメニュー 33
依存ファイルの設定
開発マネージャのウィンドウのメニュー 33
依存ファイルの登録と除外 72
位置 443
位置付け
ファイル／レコード定義ウィンドウのメニュー 437

一連番号　311, 631
一連番号の検索　347
一連番号の指定
　COBOL エディタウィンドウのメニュー　318
一連番号を検索
　COBOL エディタウィンドウのメニュー　317
一般規則　156
移動
　画面定義の編集　410
　文字列の編集　340
入口，出口指定　638
入口名　632
入口・出口・文　547
入れ子のプログラム　885
色とフォントの初期化　576
色の設定　361
印刷　378, 460
　COBOL エディタウィンドウのメニュー　316
　ファイル／レコード定義ウィンドウのメニュー　437
印刷行数　303
印刷書式　300
　実行支援ウィンドウのメニュー　286
印刷書式のデフォルト　305
印刷書式を無効にする　305
印刷制御付き　179
印刷フォント　379
印刷プレビュー　379, 460
　COBOL エディタウィンドウのメニュー　316
　ファイル／レコード定義ウィンドウのメニュー　437
印刷ページの設定　459
インスタンスオブジェクト　885
インスタンスメソッド　885
インタフェース　885
インデント　341
インポートライブラリ　28

う

ウィンドウサイズを最大にする　360
ウィンドウ内の文字列検索　608
ウィンドウの設定　359
ウィンドウ用ツールバー　38, 562
内側のプログラム　885
埋め込み SQL 文　177
上書き保存
　COBOL エディタウィンドウのメニュー　316
　COBOL エディタのファイルの保存　328
　画面定義ウィンドウのメニュー　398
　画面定義ファイルの操作　430
　実行環境ファイルの保存　307
　実行支援ウィンドウのメニュー　285
　ファイル／レコード定義ウィンドウのメニュー　437

え

英大文字と英小文字，英数字文字と拡張文字を区別しての文字列の指定方法　629
英数字定数の定数長拡張機能　546
英数字定数の分離符　226
エディタの設定ダイアログボックス　91
エディタの登録　92
エディットカーソル　400
エディットポップアップメニュー　323
エラージャンプ時の起動オプション　93, 101
エントリ行　113

お

大文字と小文字を区別　343–345
大文字に変換　341
　COBOL エディタウィンドウのメニュー　316
同じオプションを重複して指定した場合の規則　161
オブジェクト　534, 885
オブジェクト参照データ項目　885
オブジェクトデータ　885
オブジェクトデータの監視　537
オブジェクトデータの表示　535
オブジェクトのデータ値の表示　660
オブジェクトファイル　28
オブジェクトファイル，標準ライブラリからの抽出情報　799

オブジェクトファイルとライブラリファイル　811
オブジェクトプロパティ　886
オプション　330, 886
　テストデバッグウィンドウのメニュー　560
オプション指定項　155
オプション設定のダイアログボックス　617
オプションの打ち消し指定　162
オプションの変更　691
オプションを指定することによって，仮定されるオプション　159
オプションを指定すると，ほかのオプションが無効となる場合　157
オペランドの補助語　635
親プロジェクト　56
折り返し　379

か

カーソル　324
カーソル移動を記憶する　358
カーソルの移動　337
改行　360
開発マネージャ　27, 886
開発マネージャウィンドウのメニュー項目一覧　833
開発マネージャからの開始　553
開発マネージャで使用するファイル名，およびフォルダ名の規則　29
開発マネージャのウィンドウ　31
開発マネージャのウィンドウのメニュー
　COBOL ソース解析　35, 836
　単体テスト支援　36, 837
　データ影響波及分析　35, 836
開発マネージャのカスタマイズ　86
開発マネージャの起動　30
開発マネージャの終了　30
　開発マネージャのウィンドウのメニュー　32
開発マネージャの使用上の注意事項　830
開発マネージャの制限値，限界値　855
開発マネージャのテストデバッグ対象プロジェクト　555
開発マネージャの動作オプション
　開発マネージャのウィンドウのメニュー　34
開発マネージャのユーザカスタマイズ
　開発マネージャのウィンドウのメニュー　34
開発マネージャを使用した場合の登録集原文の検索順序　264
開発マネージャを標準権限で起動した場合の注意事項　831
外部スイッチ　886
外部ツールの設定ダイアログボックス　143
外部ツールの操作　141
外部ツールを操作するメニュー
　開発マネージャのウィンドウのメニュー　32
外部プログラム　886
外部プログラム名　631
カウンタ変数　641
カウント情報　473, 725, 737, 886
カウント情報の表示　700, 725, 757
カウント情報の表示の手順　757, 759
カウント情報の表示例　877
拡張 16 進定数　634
拡張コード文字　239
拡張子　332
カスタマイズ　357
　COBOL エディタウィンドウのメニュー　317
　テストデバッグウィンドウのメニュー　560
カスタマイズ情報
　インポート方法　377
　エクスポート方法　376
　保存できる情報　376
カスタマイズ情報の外部ファイル保存機能　376
カスタマイズダイアログボックス　357, 574
型　886
カタログファイル　28
活性化されるプログラム　886
カバレージ
　開発マネージャのウィンドウのメニュー　33
カバレージウィンドウのメニュー　732
カバレージウィンドウのメニュー項目一覧　841
カバレージ機能で使用するファイル　696

カバレージ機能の概要　693
カバレージ機能の入出力構成　694
カバレージ情報　189, 473, 886
カバレージ情報の０％化　723
カバレージ情報の操作　723, 754
カバレージ情報の操作の方法　754
カバレージ情報の蓄積　708, 742
カバレージ情報の蓄積の手順　742
カバレージ情報の表示（CSV 形式）　717, 750
カバレージ情報の表示（テキスト形式）　708, 747
カバレージ情報の表示の手順（CSV 形式）　750
カバレージ情報の表示の手順（テキスト形式）　747
カバレージ情報の表示の方法（CSV 形式）　750
カバレージ情報の表示例　871
カバレージ情報のマージ　724, 736
カバレージ情報表示の手順　706
カバレージで動作させるプログラムについての注意事項　701
カバレージ統計情報の出力　717
カバレージとは　473
カバレージのウィンドウ　732
カバレージの機能一覧　474
カバレージの終了
　カバレージウィンドウのメニュー　732
カバレージの手順　729
画面環境　296
　実行支援ウィンドウのメニュー　286
画面環境の設定　296
画面環境を無効にする　300
画面サイズ　297
画面定義　394
画面定義ウィンドウ　397
画面定義ウィンドウのメニュー項目一覧　841
画面定義環境設定　427
　画面定義ウィンドウのメニュー　398
画面定義の開始　396
画面定義の終了　396
　画面定義ウィンドウのメニュー　398
画面定義の制限値，限界値　856
画面定義の手順例　407
画面定義の流れ　394
画面定義ファイル　28
画面の配色の設定　298
画面表示の設定　574
環境変数　290, 887
環境変数 CBLUNIENDIAN の値と仮定するバイトオーダ　484
環境変数の登録　286
監視条件の解除　594
関数指定　636
関数名　631
管理者権限　887
管理者権限についての注意事項　823
管理者ユーザ　887
関連　82

き

キー　453, 456
キー位置　454, 456
キー長　454, 456
キーによるカーソルの移動　337
キーの削除　455, 456
キーの追加　454, 456
キー判定属性　454, 456
キーボードマクロ　311
キーボードマクロ実行
　COBOL エディタウィンドウのメニュー　317
キーボードマクロ登録開始
　COBOL エディタウィンドウのメニュー　317
キーボードマクロ登録終了
　COBOL エディタウィンドウのメニュー　317
キーワード補完　311, 354, 355
　COBOL エディタウィンドウのメニュー　317
キー割り当て　296
キー割り当ての設定　299
記号名　513, 642
記号名構造指定　692
擬似プログラム用プログラム情報ファイル　194

既定義オブジェクト名　635
起動時のウィンドウサイズ　297
機能キャラクタ　311
機能キャラクタの設定　360
旧形式のオプション　247
旧バージョンで作成したプロジェクトマスタファイルを移行した場合の注意事項　831
行　406, 407
行，列　403
行間　407
行間隔　303
行指定の背景色分け表示の設定　311, 374
行選択　338
共通実行環境の設定　295
共通実行環境ファイル　282
共通例外を発生させるプログラムのテストデバッグ　543
行の範囲選択　338
行番号　359, 379, 631
行番号指定時の起動オプション　94
行末から次行に移動する　358
行末空白削除　379
業務名称　441
切り取り
　COBOL エディタウィンドウのメニュー　316
　画面定義ウィンドウのメニュー　398
　画面定義の編集　410
　テストデバッグウィンドウのメニュー　559
　ファイル／レコード定義ウィンドウのメニュー　437
　文字列の編集　340

く

クイックビルド　149
　開発マネージャのウィンドウのメニュー　33
空白　360
空白と括弧を含む文字列の指定方法　628
クラス　887
クラス指定　636
クラス名　631
グラフィック　360
クリア
　テストデバッグウィンドウのメニュー　559
繰り返し回数　406
繰り返し指定　672
グループ名　402

け

継承　887
罫線　421
罫線作成モード　421
　画面定義ウィンドウのメニュー　398
罫線の移動　422
罫線の削除　422
罫線の作成　421
罫線の選択　421
罫線の複写　422
罫線表示　402, 407
ケースコードによる条件判定　521
ケースコードの設定　650
ケース識別子の指定　520
結果出力先の指定　525
限界値　848
現在位置の表示　606
　テストデバッグウィンドウのメニュー　559
現在の内容で実行　306
　実行支援ウィンドウのメニュー　286
検索　343
　ファイル／レコード定義ウィンドウのメニュー　437
検索する方向　343
検索する文字列　334
検索場所　346
検索文字列にブックマークを設定　343
検索用ツールバー　562

こ

構成管理ツール　887
構成管理ツールとの連携　133
構成管理ツールの仕様　134

構成管理ツールの設定ダイアログボックス　137
構成管理ツールの操作　134
合成キーの追加　455
合成キーの追加ダイアログボックス　455
合成キーの変更　455
合成キー名　456
項番　443
構文　333
構文規則　155
構文チェック　311, 352
　COBOL エディタウィンドウのメニュー　317
構文チェック オプション　352
構文チェック　オプション
　COBOL エディタウィンドウのメニュー　317
構文チェック結果をウィンドウに合せて折り返す 354
構文チェックするウィンドウを自動的に保存する 354
構文チェックの環境設定　352
構文チェックの中止
　COBOL エディタウィンドウのメニュー　317
構文ツリー欄　334
構文テンプレート　310, 333
　COBOL エディタウィンドウのメニュー　317
構文テンプレートダイアログボックス　333
構文テンプレートファイル　334
語選択　338
固定／自由形式　365
固定形式正書法　262
固定形式ファイル　325
固定設定　595
古典的要素　205
コピー
　COBOL エディタウィンドウのメニュー　316
　画面定義ウィンドウのメニュー　398
　画面定義の編集　410
　テストデバッグウィンドウのメニュー　559
　ファイル／レコード定義ウィンドウのメニュー 437
　文字列の編集　340
子プロジェクト　56
コマンドファイルインタフェースで起動する　101
コマンドを入力してカウント情報を取得　757
コメント　83
コメントの設定　40
コメントの表示　42
小文字に変換　341
　COBOL エディタウィンドウのメニュー　316
コンパイラオプション　155
コンパイラオプションの移行　819
コンパイラオプションの一覧　163
コンパイラオプションの一般規則　156
コンパイラオプションの詳細情報　247
コンパイラオプションの設定　65, 151
コンパイラオプションの表記　155
コンパイラオプションの優先順位　156
コンパイラ環境変数　260
コンパイラ環境変数の一覧　259
コンパイラ環境変数の設定　66
コンパイラの制限値，限界値　848
コンパイル　148, 149, 887
　開発マネージャのウィンドウのメニュー　33
コンパイルリストの出力形式を指定する　234

さ

再帰によるソース要素の実行　542
最終生成物の種類（プロジェクトの種類）の設定 174
最小レコード長　440
最新のファイル
　開発マネージャのウィンドウのメニュー　32
最新を取得　133
　開発マネージャのウィンドウのメニュー　32
サイズ　304, 403, 407, 443
最外側のプログラム　887
再定義名　444
最適化のレベル　183
作業中プロジェクト　58
作業中プロジェクト設定ダイアログボックス　59
作業中プロジェクトの設定　58
　開発マネージャのウィンドウのメニュー　33
作業フォルダ　51

作業フォルダ名　60
削除　450
　COBOL エディタウィンドウのメニュー　316
　開発マネージャのウィンドウのメニュー　32
　画面定義ウィンドウのメニュー　398
　画面定義の編集　411
　ファイル／レコード定義ウィンドウのメニュー　437
　文字列の編集　340
作成　411
作成できるプログラムの相違　814
サブオプション　156
サブクラス　888
差分 C0 メジャー　702
差分 C1 メジャー　702
差分 S1 メジャー　702
差分カバレージ情報　888
差分カバレージ情報の０％化　724
差分カバレージ情報のクリア　724
差分ソース表示　715
差分未実行ソース表示　716
サロゲート　888
参照元　455, 457

し

字間隔　304
システム属性　330
システムの種類　330
実行　281
　開発マネージャのウィンドウのメニュー　34
実行可能ファイル　28, 282
実行可能ファイル，DLL ファイルからの抽出情報　797
実行可能ファイルと DLL ファイル　811
実行可能ファイルの実行権限についての注意事項　829
実行可能プログラム　888
実行環境の設定　286
実行環境ファイルのオープン　307
実行環境ファイルの作成　307
実行環境ファイルの保存　307
実行形式プログラム　52
実行形式プログラムで選択できるプロジェクトの種類　54
実行結果出力ファイルにメッセージを出力できないときの処理　746, 761
実行時エラーが発生した場合の中断　497
実行支援　283
　開発マネージャのウィンドウのメニュー　34
実行支援ウィンドウ　285
実行支援ウィンドウのメニュー項目一覧　837
実行支援で使用できない機能　821
実行支援の制限値，限界値　856
実行時環境変数の優先順位　295
実行時環境変数の一覧　290
実行時の注意事項　829
実行時要素　888
実行時要素の切り替え　602
実行の開始　283
実行の開始／再開　669
実行の開始と開始時にできる指定　499
実行の強制終了　499
実行の終了　284
実行の中止　597
実行の割り込み中断　598
実行方法の種類　282
指定行の表示　526, 584
　テストデバッグウィンドウのメニュー　559
自動保存　366
指標名　444
シフト JIS コード　534
シミュレーション情報　191, 768
ジャンプ　346, 498, 597
　COBOL エディタウィンドウのメニュー　317
ジャンプ実行　499, 597
主／副プログラムシミュレーションで生成する擬似プログラム用プログラム情報ファイル　194
自由形式正書法　263
自由形式ファイル　325
集団項目　411
集団項目作成　411

画面定義ウィンドウのメニュー　398
集団項目の移動　412
集団項目の削除　412
集団項目の作成　411
集団項目の選択　411
集団項目の複写　412
終了　284
実行支援ウィンドウのメニュー　285
出力　62, 81
出力先ファイル　79
出力ファイル種別　100
出力ファイルの変更
開発マネージャのウィンドウのメニュー　32
出力ファイルの編集
開発マネージャのウィンドウのメニュー　32
出力ファイル名の設定　66
主プログラムシミュレーション　506
主プログラムシミュレーション機能　191
主プログラムシミュレーションの設定　688
主レコードキー　453
主レコードキーとして設定　454
順序　454, 456
順編成ファイル（可変長），相対編成ファイルからの抽出情報　800
昇格ダイアログボックス　888
照合順序　218
常時データ表示ウィンドウ　568
テストデバッグウィンドウのメニュー　560
常時データ表示に設定
テストデバッグウィンドウのメニュー　560
使用するエディタ名　93
使用するコード変換ライブラリ　476
仕様チェックオプションを複数指定した場合　159
使用できない環境変数　810
使用できない機能　540
使用できないコンパイラオプション　809
情報抽出ツール出力結果リスト　784
オブジェクトファイル，標準ライブラリからの抽出情報　799
実行可能ファイル，DLL ファイルからの抽出情報　797
順編成ファイル（可変長），相対編成ファイルからの抽出情報　800
プロセスダンプからの抽出情報　788
情報抽出ツール出力結果リストの出力形式　787
ショートカットキー　88
ショートカットキーの削除　89
ショートカットキーの初期化　89
ショートカットキーの登録　88
初期化
テストデバッグウィンドウのメニュー　559
初期化できるテストデバッガの機能　524
初期値　444
初期値へ戻す　379
書式オーバレイ　179
書体の設定　304
字類条件　218
字類条件の判断基準と文字の大小関係　219
新規作成
COBOL エディタウィンドウのメニュー　316
COBOL エディタのファイルの作成　326
開発マネージャのウィンドウのメニュー（ファイル）　32
開発マネージャのウィンドウのメニュー(プロジェクトマスタ)　32
画面定義ウィンドウのメニュー　397
画面定義ファイルの操作　429
実行環境ファイルの作成　307
実行支援ウィンドウのメニュー　285
ファイル／レコード定義ウィンドウのメニュー　436
プログラムテンプレートの作成　331
新規作成時の起動オプション　93, 101
新規作成ダイアログボックス　331
新規作成メニューの相違　814

す

数字項目のけた拡張機能　543
スーパクラス　888
スタイル　304

ステップイン　497, 596
　テストデバッグウィンドウのメニュー　559
ステップイン実行の開始／再開　684
ステップオーバー　498, 596
　テストデバッグウィンドウのメニュー　560
ステップオーバー実行の開始／再開　684
ステップツー実行の再開　685
スパースキーを指定　454
すべて選択
　テストデバッグウィンドウのメニュー　559
すべてのソーステキストウィンドウを閉じる　584
すべての中断点の解除
　テストデバッグウィンドウのメニュー　560
すべてのプロジェクトに対する操作
　開発マネージャのウィンドウのメニュー　34
すべての余白を 0 にする　303
すべて保存
　COBOL エディタウィンドウのメニュー　316
　COBOL エディタのファイルの保存　328
すべてを選択　339
　COBOL エディタウィンドウのメニュー　316
スレッド ID　548

せ

制限値　848
生成ツール　96
セクション　112
セクション名の一覧　112
接続　329
設定値を登録　332
全ソース表示　712
選択　411, 448
選択行の左シフト
　COBOL エディタウィンドウのメニュー　318
選択行の右シフト
　COBOL エディタウィンドウのメニュー　318
選択マージン　359, 888
前提条件　382

そ

操作　735
　カバレージウィンドウのメニュー　732
操作できるファイルの扱いの相違　813
挿入
　ファイル／レコード定義ウィンドウのメニュー　437
添字　632
添字に指定　632
ソースカバレージ情報の出力　721
ソーステキストウィンドウ　563
　テストデバッグウィンドウのメニュー　560
ソーステキストウィンドウからの表示　582
ソーステキストの表示　581
ソーステキストの編集　526
ソースファイルの追加
　開発マネージャのウィンドウのメニュー　33
ソースファイルの編集　607
　テストデバッグウィンドウのメニュー　559
ソース要素指定　637
ソース要素の入口　638
ソース要素の出口　638
ソース要素の表示形式　547
ソース要素の呼び出しの順序を表示　501
ソース要素・中断点・通過点の表示形式　547
属性エディタウィンドウ　401, 422
属性エディタを開く
　画面定義ウィンドウのメニュー　398
属性項目の編集　425
属性設定ウィンドウ　443
属性設定を開く
　ファイル／レコード定義ウィンドウのメニュー　437
属性の設定方法　422
その他の注意事項（カバレージ）　701
その他の注意事項（テストデバッガ）　492

た

第 1 次規格　205
第 2 次規格　205
第 3 次規格　205

第 4 次規格　205
ダイナミックライブラリ　28
ダイナミックリンクライブラリ　52
ダイナミックリンクライブラリの属性の選択　53
タグジャンプ　311, 348
縦スクロールバー　360
他の実行可能ファイル　306
　実行支援ウィンドウのメニュー　286
タブ　360
タブコード　268
タブを空白文字に置換
　COBOL エディタウィンドウのメニュー　316
段組　379
単語で検索　343–345
単体テスト　646, 888

ち

チェックアウト　133
　開発マネージャのウィンドウのメニュー　32
チェックアウト状態の変更
　開発マネージャのウィンドウのメニュー　32
チェックイン　133
　開発マネージャのウィンドウのメニュー　32
置換　345
　COBOL エディタウィンドウのメニュー　316
　ファイル／レコード定義ウィンドウのメニュー 437
置換ダイアログボックス　345
置換文字列　94, 102
蓄積　733
　カバレージウィンドウのメニュー　732
中間ファイル　51
注釈　445, 629
注釈行（シミュレーション情報の開始）　768
注釈行（シミュレーション情報の終了）　771
注釈行（主プログラム）　768
注釈行（中断点情報の開始）　765
注釈行（中断点情報の終了）　767
注釈行（ファイル）　770
注釈行（副プログラム）　769
注釈の表示　525, 653
中断時のスレッド　540
中断時のソーステキストの表示　583
中断点一覧の表示　587
中断点情報　191, 765
中断点情報とシミュレーション情報　765
中断点の一覧
　テストデバッグウィンドウのメニュー　560
中断点の解除　586, 589, 676
中断点の設定　585, 673
中断点の設定/解除
　テストデバッグウィンドウのメニュー　560
中断点の設定と解除　585
中断点の設定による中断　494
中断点の全解除　589
中断点の無効化　589
重複キーの順序を保証　454
重複キーを許可する　454

つ

通常使うプリンタに出力　302
通信状態　324
ツールバー　36, 286, 319, 399, 438
　画面定義ウィンドウのメニュー　398
　ファイル／レコード定義ウィンドウのメニュー 437
ツールバーの設定　577
ツールバーの表示
　COBOL エディタウィンドウのメニュー　317
ツール名の絶対パス　100, 101
ツール用ツールバー　38
次のブックマーク　346
　COBOL エディタウィンドウのメニュー　317
次を検索　343
ツリービューウィンドウ　38
ツリービューウィンドウのアイコン　41

て

定義ツール　98
定義ファイル　888

定義ファイルのオープン　430, 447
定義ファイルの作成　429, 446
定義ファイルの保存　430, 447
定義名一覧　99
定数　633
定数一覧　633
データ値の代入　480
データ監視条件一覧ダイアログボックス　593
データ監視条件一覧ダイアログボックスで設定する内容　591, 593
データ監視条件一覧の表示と解除　593
データ監視条件の一覧
　テストデバッグウィンドウのメニュー　560
データ監視条件の解除　683
データ監視条件の設定　590, 682
　テストデバッグウィンドウのメニュー　560
データ監視条件の設定と解除　590
データ監視条件の設定による中断　495
データ項目　448, 455, 456
データ項目種別　403, 407
データ項目の位置付け　451
データ項目の移動　449
データ項目の検索　450
データ項目の削除　450
データ項目の選択　448
データ項目の挿入　448
データ項目の属性の編集　448
データ項目の置換　450
データ項目の複写　449
データ設定値　771
データソース　382
データソースの選択　387
データ属性表示　504
データ値の代入　605
　テストデバッグウィンドウのメニュー　560
データ値の表示　478, 603
　テストデバッグウィンドウのメニュー　560
データの値代入　651
データの値比較　671
データの値表示　653
データの型と VARIANTARG 構造体の vt の値の対応　541
データの設定解除　570
データの操作　540, 645
データの代入　504
データの代入規則　527
データの比較　505
データの比較規則　530
データの部分を参照　633
データ名　443, 632
データ名指定　639
データ名の設定　569
データ名・ファイル名が参照できる範囲　533
データ例外の検出　187
適合　888
テキスト　360
テキストエディタの設定　91
テキストの色分け　310
テキストファイルをユーザキーワードで色分け表示する　362
テスト機能　470
テストケース　519
テストケースの指定　651
テストケースの選択　689
テスト結果蓄積先の解除　679
テスト結果蓄積先の設定　679
テスト結果の出力　525
テストしたいプログラムの実行　491, 699
テスト時の入出力構成　487
テストで使用するファイル　488
テストデバッガで動作させるプログラムについての注意事項　492
テストデバッガとは　469
テストデバッガの概要と規則　486
テストデバッガの起動　553
テストデバッガの機能一覧　471
テストデバッガの終了　556
　テストデバッグウィンドウのメニュー　559

テストデバッガの初期化　523
テストデバッガの制限値，限界値　857
テストデバッグ一覧ウィンドウ　566
　テストデバッグウィンドウのメニュー　560
テストデバッグ一覧ウィンドウからの表示　581
テストデバッグウィンドウ　558
テストデバッグウィンドウのメニュー項目一覧　838
テストデバッグ機能の初期化　579
テストデバッグ対象とする実行可能ファイルと DLL ファイルの変更　523, 578
テストデバッグ対象の設定
　テストデバッグウィンドウのメニュー　559
テストデバッグで使用するウィンドウ　558
テストデバッグで使用するウィンドウのカスタマイズ　573
テストデバッグのオプション設定　617
テストデバッグの概要　494
テストデバッグの環境設定　647
テストデバッグの終了　672
テストデバッグのための設定　523
テストデバッグの手順　550
テストデバッグ用ツールバー　561
テストプロジェクトファイル　28
手続き名　632
手続き名指定　638
デバッガ
　開発マネージャのウィンドウのメニュー　34
デバッグ機能　469
デバッグ行　185
デバッグ情報　188
デバッグ手順　478
デフォルトオプションの設定
　開発マネージャのウィンドウのメニュー　32
デフォルトオプションの設定方法　151
デフォルトで機能キャラクタを表示する　361
デフォルトの環境変数の設定方法　255
テンプレート　331
テンプレートフォルダ　370

と

等価規則　524, 629
動的アクセス　889
登録集環境変数　273
登録集原文　264, 273, 889
登録集原文の表示／非表示　582
登録集原文を登録する　74
登録集名　273
特殊文字を含んだファイルを開いた場合　326
閉じる
　COBOL エディタウィンドウのメニュー　316
　COBOL エディタのファイルのクローズ　328
　開発マネージャのウィンドウのメニュー　32
　画面定義ウィンドウのメニュー　397
　ファイル／レコード定義ウィンドウのメニュー　436
　プロジェクトマスタファイルを閉じる　46
トップレベルプロジェクト　57, 889
トラッキングバー　323, 359, 889
ドラッグ＆ドロップで開く　46
取消　452
　ファイル／レコード定義ウィンドウのメニュー　437
トレース　889
トレースバック情報　186
トレース表示　503
トレース表示の開始　680
トレース表示の中止　682

な

内部プログラム　889
内部プログラム名　631
名前を付けて保存　333
　COBOL エディタウィンドウのメニュー　316
　COBOL エディタのファイルの保存　328
　画面定義ウィンドウのメニュー　398
　画面定義ファイルの操作　430
　実行環境ファイルの保存　308
　実行支援ウィンドウのメニュー　285
　ファイル／レコード定義ウィンドウのメニュー　437

に

日本語入力のオン／オフを自動的に切り替える　297
入出力条件シミュレーション　670
入出力条件のシミュレーション　512
入出力ファイル　811
入出力文選択指定　673
入力　61
入力する文字の扱い　524
入力チェック　405
入力編集　405
入力モード　324
任意オペランド　628

は

配色　296
配色の設定　90
配色の設定方法　575
配置　304
バイトオーダ　484, 889
バイトオーダマーク　882
背反関係にあるサブオプション同士を指定した場合　161
廃要素　203, 204
バックアップの作成　366
バックアップファイル　28
バックトレース　600
　テストデバッグウィンドウのメニュー　559
バックトレース表示　503
バックトレース用ツールバー　563
パッケージ情報ファイル　28
パッケージ情報ファイル（.pkg）の移行　830
パッケージ情報ファイル（.pkg）の出力先　104
バッチによるテスト　519, 647
バッチモードでのテストの方法　624
バッチモードによるカバレージ　740
バッチモードによるテストデバッグ　622
バリアントデータ項目　890
バリアントデータ項目の値の表示　540
貼り付け
　COBOL エディタウィンドウのメニュー　316
　画面定義ウィンドウのメニュー　398
　画面定義の編集　410
　テストデバッグウィンドウのメニュー　559
　ファイル／レコード定義ウィンドウのメニュー　437
　文字列の編集　340
範囲外チェック　202–204
範囲選択　338
反復　444

ひ

比較条件式　643
比較条件式の設定　483
ビッグエンディアン形式　227
必須オペランド　628
ビューカーソル　400
表意定数の扱い　485
表項目　400
表項目の繰り返し数の変更　417
表作成　399
　画面定義ウィンドウのメニュー　398
表示　734
　カバレージウィンドウのメニュー(カウント情報)732
　カバレージウィンドウのメニュー(カバレージ情報)　732
表示／制御　296
標識領域　311
標識領域の指定　340
　COBOL エディタウィンドウのメニュー　318
表示属性　404
表示属性の設定　571
表示速度を重視する　184
表示の設定　358
表示の相違　813
表示前動作　405
標準権限　890
標準コード文字　239
標準配置　573
標準ユーザ　890

標準ライブラリ　53, 176
表の移動　418
表のサイズ変更　417
表の削除　419
表の作成　415
表の参照　632
表の選択　388, 416
表の複写　418
表名　406
表要素　890
表要素長の変更　417
表要素の移動　420
表要素の切り取り　420
　画面定義ウィンドウのメニュー　398
表要素のコピー　420
　画面定義ウィンドウのメニュー　398
表要素の削除　420
　画面定義ウィンドウのメニュー　398
表要素の追加　419
　画面定義ウィンドウのメニュー　398
表要素の貼り付け　420
　画面定義ウィンドウのメニュー　398
表要素の複写　420
開いているファイルを記憶する　367
開く
　COBOL エディタウィンドウのメニュー　316
　COBOL エディタのファイルのオープン　326
　開発マネージャのウィンドウのメニュー（ファイル）　32
　開発マネージャのウィンドウのメニュー(プロジェクトマスタ)　32
　画面定義ウィンドウのメニュー　397
　画面定義ファイルの操作　430
　実行環境ファイルのオープン　307
　実行支援ウィンドウのメニュー　285
　ツリービューウィンドウに表示されているファイルを開く　78
　ファイル／レコード定義ウィンドウのメニュー　436
　プロジェクトマスタを開く　45
開くメニューの相違　814
ビルド　149, 890
　開発マネージャのウィンドウのメニュー　33
ビルド処理用ツールバー　37
ビルドの中止
　開発マネージャのウィンドウのメニュー　34

ふ

ファイル／レコード定義　431, 432
ファイル／レコード定義ウィンドウ　436
ファイル／レコード定義ウィンドウのメニュー項目一覧　842
ファイル／レコード定義環境設定　465
　ファイル／レコード定義ウィンドウのメニュー　437
ファイル／レコード定義の印刷　459
ファイル／レコード定義の開始　434
ファイル／レコード定義の終了　434
　ファイル／レコード定義ウィンドウのメニュー　437
ファイル／レコード定義の制限値，限界値　857
ファイル／レコード定義の流れ　432
ファイル一覧から開く　46
ファイル格納先についての注意事項　828
ファイルから検索　344
　COBOL エディタウィンドウのメニュー　316
ファイルから検索を中止
　COBOL エディタウィンドウのメニュー　316
ファイル区分　441
ファイルシミュレーション　509, 538, 540
ファイルシミュレーションの設定　687
ファイル仕様定義ウィンドウ　438
ファイル仕様定義の印刷規則　461
ファイル情報　428
　画面定義ウィンドウのメニュー　398
　ファイル／レコード定義ウィンドウのメニュー　437
ファイル情報の参照　328
ファイル情報の表示　465
ファイル単位のコンパイル　150, 890
ファイル定義ファイル　28, 432
ファイル定義名　99

ファイル入出力文のシミュレーション　509
ファイルの移行　819
ファイルの書き込み時の制約（カバレージ）　701
ファイルの書き込み時の制約（テストデバッガ）　492
ファイルの拡張子　100
ファイルの種類　329, 332, 365
ファイルの種類の削除　104
ファイルの種類の新規登録　103
ファイルの種類の登録情報の編集　103
ファイルの種類の登録方法　103
ファイルの種類の保存　104
ファイルの情報の確認　79
ファイルの新規作成　70
ファイルの設定　83
　開発マネージャのウィンドウのメニュー　32
ファイルの挿入　342
　COBOL エディタウィンドウのメニュー　316
ファイルの登録と管理　70
ファイルの登録と除外　70
ファイルの登録と除外の規則　76
ファイルの場所　329
ファイル表示名　99
ファイル編成　440, 808
ファイル名　329, 440
ファイル名指定　639
ファイル名指定時の起動オプション　93, 101
ファイル名の前に付加する起動オプション　100
ファイルメニューの相違　814
ファイルを開く　77
ファクトリオブジェクト　890
ファクトリオブジェクトとインスタンスオブジェクト　534
ファクトリデータ　890
ファクトリメソッド　890
フィールド構成ウィンドウ　399
フィールド項目　400
フィールド作成モード
　画面定義ウィンドウのメニュー　398
フィールド種別　403, 407
フィールドの移動　414
フィールドのサイズ変更　414
フィールドの削除　415
フィールドの作成　412
フィールドの選択　413
フィールドの複写　414
フィールド名　403
フォームエディタウィンドウ　400
フォームエディタを開く
　画面定義ウィンドウのメニュー　398
フォルダ指定によるファイルの追加　84
フォント　297, 304, 364, 460
フォントの設定　90
フォントの設定方法　575
フォント名　304
複写
　画面定義の編集　410
　文字列の編集　340
副プログラムシミュレーション　507
副プログラムシミュレーション機能　192
副プログラムシミュレーションの設定　688
ブックマーク　346, 890
ブックマークによる検索　346
ブックマークの設定／解除　346
　COBOL エディタウィンドウのメニュー　316
ブックマークの設定と解除　346
ブックマークをすべて解除　346
　COBOL エディタウィンドウのメニュー　317
フッタ　378, 459
浮動継続指示子　177
不当な一連番号の検索　347
不当な一連番号を検索
　COBOL エディタウィンドウのメニュー　317
負の符号を持つゼロ　240
部分参照　633
プリンタの設定　302, 379, 460
フロー情報の蓄積開始　677
フロー情報の蓄積中止　677
フロー情報の表示　503, 658

プログラム　891
プログラムからの連動実行　542, 609
プログラムからの連動実行についての注意事項　493
プログラムからの連動実行による方法　744, 759
プログラム互換性アシスタント　829
プログラム指定　636
プログラム情報ファイル　188, 189, 266, 812
プログラム情報ファイルのエラーについて（カバレージ）　701
プログラムテンプレート　310, 331, 332
プログラムテンプレートの格納場所　331
プログラムの強制終了　686
プログラムのコンパイル　490, 699, 707
プログラムの実行　539
プログラムの実行環境の文字コード　534
プログラムの実行制御　595
プログラムの実行追跡　599
プログラムの実行の制御と追跡　645
プログラムの実行を終了
　テストデバッグウィンドウのメニュー　560
プログラムの単体テスト　506
プログラムの中断　494
プログラムの中断位置のソーステキストを表示　502
プログラム別実行環境ファイル　282
プログラムを実行する場合　283
プロジェクト　26, 891
プロジェクト間の関係設定　56
プロジェクト間の関係設定での規則　58
プロジェクト管理用ツールバー　37
プロジェクト作成ウィザード　52
プロジェクト情報の出力　66
プロジェクト情報の取り込み　67
プロジェクト情報ファイル　28, 66, 891
プロジェクト設定ダイアログボックス　64
プロジェクト設定の相違　815
プロジェクトの移行　817
プロジェクトのインポート　67
　開発マネージャのウィンドウのメニュー　33
プロジェクトのエクスポート　66
　開発マネージャのウィンドウのメニュー　33
プロジェクトの関係設定　57
　開発マネージャのウィンドウのメニュー　33
プロジェクトの管理　27
プロジェクトの作業フォルダの設定　66
プロジェクトの削除　63
　開発マネージャのウィンドウのメニュー　33
プロジェクトの作成　50
　開発マネージャのウィンドウのメニュー　33
プロジェクトの作成ダイアログボックス　50
プロジェクトの種類　174
プロジェクトの設定　64
　開発マネージャのウィンドウのメニュー　33
プロジェクトの相違　815
プロジェクトの定義　50
プロジェクトの定義内容の確認　59
プロジェクトのプロパティダイアログボックス　60
プロジェクトフォルダ　51
プロジェクトフォルダと作業フォルダを同一にする 52
プロジェクトフォルダ名　60
プロジェクトマスタ　26, 891
プロジェクトマスタ情報　891
プロジェクトマスタ情報の CSV ファイル入出力　108
プロジェクトマスタ情報の CSV ファイルの形式　112
プロジェクトマスタ情報の入出力
　開発マネージャのウィンドウのメニュー　35
プロジェクトマスタ情報ファイル　891
プロジェクトマスタの作成ダイアログボックス　44
プロジェクトマスタの新規作成　44
プロジェクトマスタの定義内容の確認　47
プロジェクトマスタの複写　48
プロジェクトマスタのプロパティダイアログボックス　47
プロジェクトマスタの保存　47
プロジェクトマスタファイル　28, 44, 891
プロジェクトマスタファイルの移行　817
プロジェクトマスタファイルの相違　813
プロジェクトマスタファイルのバックアップファイル　46

プロジェクトマスタフォルダ名　45
プロジェクトマスタ名　45
プロジェクトマスタを閉じる　46
プロジェクトマスタを開く　45
プロジェクト名　51
プロジェクト名の変更
　開発マネージャのウィンドウのメニュー　33
プロセスダンプからの抽出情報　788
ブロック選択　339
ブロック長　441
プロパティ
　COBOL エディタウィンドウのメニュー　316
　COBOL エディタのファイル情報の参照　328
　開発マネージャのウィンドウのメニュー(ファイル)　33
　開発マネージャのウィンドウのメニュー(プロジェクト)　33
　開発マネージャのウィンドウのメニュー(プロジェクトマスタ)　32
プロパティの値表示　667
文番号　631
文番号指定　638

へ

ページの設定　459
　ファイル／レコード定義ウィンドウのメニュー　437
ページレイアウトの設定　378
　COBOL エディタウィンドウのメニュー　316
ヘッダ　378, 459
編集動作の設定　357
編集文字の設定　360
編集用 PICTURE　444

ほ

報告書作成機能　213
補助語　635
保存　47, 366
　開発マネージャのウィンドウのメニュー　32
ポップアップメニュー　321
翻訳単位指定　636
翻訳単位の一覧表示　709
翻訳単位・ソース要素の指定　680
翻訳単位・ソース要素の指定の解除　680
翻訳変数名　245

ま

マーカ　310, 323, 891
マージ
　カバレージウィンドウのメニュー　732
毎回設定　595
マウスポインタ　323
前のブックマーク　346
　COBOL エディタウィンドウのメニュー　317
まとめ情報の表示　710
マルチスレッド　181, 891
マルチスレッド対応 COBOL プログラムのテストデバッグ　539

み

未実行表示　715
見出し領域　311
見出し領域の置換　341
　COBOL エディタウィンドウのメニュー　318
見た目幅　891

め

メインファイル名　56
メインファイルを指定する　183
メインフレーム　207, 892
メインプログラムなし　54, 175
メソッド　631, 892
メソッド原型　892
メソッドの呼び起こし　892
メッセージウィンドウ
　開発マネージャのウィンドウのメニュー　36
メッセージウィンドウの表示の設定　89
メニューから開く　45
メニュー項目一覧　833
メニューとウィンドウ　526

メニューバーのアイコンとファイル種別　325

も

文字コードについての注意事項　828
文字の大小関係　219
モジュール定義ファイル　197
文字列の検索　343, 526
　COBOL エディタウィンドウのメニュー　316
　テストデバッグウィンドウのメニュー　559
文字列の検索ダイアログボックス　343
文字列のシフト　311
文字列の選択　338
文字列の複写　449
元に戻す　340
元に戻す
　COBOL エディタウィンドウのメニュー　316
モニタウィンドウ　571
　テストデバッグウィンドウのメニュー　561
モニタウィンドウ表示内容のファイル出力の解除 678
モニタウィンドウ表示内容のファイル出力の設定 678

や

やり直し　340
　COBOL エディタウィンドウのメニュー　316

ゆ

ユーザインタフェースの設定　87
ユーザキー　296
ユーザキーワードファイル　369, 892
ユーザキーワードファイルの形式　371
ユーザキーワード補完　355
　COBOL エディタウィンドウのメニュー　317
ユーザ構文テンプレート　334
ユーザ資産の移行　816

よ

用語解説　882
要素名　407
横スクロールバー　360
余白　303, 379, 459
呼び出し一覧ウィンドウ　567
　テストデバッグウィンドウのメニュー　560
呼び出し一覧ウィンドウからの表示　582
呼び出し先のプログラム　892
呼び出し順序番号　501
呼び出し文ソース表　716
呼び出し元プログラム　892
予約語　367

ら

ライブラリの指定　196
ライブラリファイル　28

り

リソース定義ファイル　198
リトルエンディアン形式　227
リビルド　149, 892
　開発マネージャのウィンドウのメニュー　34
リポジトリファイル　28, 244, 267, 268, 812
領域の解放　649
領域の確保　648
領域の操作　505
リレーショナルデータベース（XDM/RD）操作シミュレーション機能　177
リンカ　200
リンカオプションの設定　66
リンカオプションの設定方法　274
リンケージを行わない　196

る

ルーラー　322, 359
ルーラーの表示　323
ルーラーポップアップメニュー　322

れ

レコード形式　440
レコード仕様定義ウィンドウ　441
レコード仕様定義の印刷規則　461

レコード長　440
レコード長設定エリア名　441
レコード定義ウィンドウ　442
レコード定義生成の流れ　382
レコード定義の確認　389
レコード定義の生成　388
レコード定義の生成例　391
レコード定義ファイル　28, 432, 454, 456
レコード定義ファイル名　441
列　406, 407
列の情報　384
レベル　443
レベル番号　402, 403, 407, 692
連携できる外部ツールの仕様　141
連続実行　497, 596
　テストデバッグウィンドウのメニュー　559
連動実行　542
連動実行(DLL ファイルだけ)　613
連動実行(GUI モード)　609
連動実行(カウント情報の表示)　759
連動実行(カバレージ情報の蓄積)　744
連動実行のエラーについて　615
連動実行の待機
　テストデバッグウィンドウのメニュー　560

ろ

論理誤りチェックと入出力状態の値　512

わ

割り込み中断
　テストデバッグウィンドウのメニュー　560
割り込みによる中断　496