SEWB+/CONSTRUCTION

アプリケーション開発ガイド

手引・文法書

3020-3-N84

マニュアルの購入方法

このマニュアル，および関連するマニュアルをご購入の際は，巻末の「ソフトウェアマニュアルのサービス ご案内」をご参照ください。

HITACHI

■対象製品

P-2451-1434 SEWB+ 基本開発環境セット 03-00（適用 OS：Windows 2000，Windows XP，Windows Server 2003，Windows Server 2003 x64，Windows Vista）

■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ならびに米国の輸出管理関連法規などの規制をご確認の上，必要な手続きをお取りください。
なお，ご不明な場合は，弊社担当営業にお問い合わせください。

■商標類

CORBA は，Object Management Group が提唱する分散処理環境アーキテクチャの名称です。
Microsoft は，米国およびその他の国における米国 Microsoft Corp. の登録商標です。
ODBC は，米国 Microsoft Corp. が提唱するデータベースアクセス機構です。
ORACLE は，米国 Oracle Corporation の登録商標です。
Oracle は，米国 Oracle Corporation 及びその子会社，関連会社の登録商標です。
Visual Basic は，米国およびその他の国における米国 Microsoft Corp. の登録商標です。
Visual C++ は，米国およびその他の国における米国 Microsoft Corp. の登録商標です。
Windows は，米国およびその他の国における米国 Microsoft Corp. の登録商標です。
Windows Server は，米国およびその他の国における米国 Microsoft Corp. の商標です。
Windows Vista は，米国 Microsoft Corporation の米国及びその他の国における登録商標または商標です。

■発行

2007 年 8 月（第 1 版）3020-3-N84

■著作権

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プログラム構築支援機の概要，操作方法および文法について説明したものです。
- P-2451-1434　SEWB+ 基本開発環境セット

SEWB+ 基本開発環境セットは，SEWB+ 基本開発環境と Groupmax ObjectServer から構成されます。
また，SEWB+ 基本開発環境は，次に示す機能から構成されます。
- リポジトリ管理機能
- リポジトリブラウザ機能
- プログラム構築支援機能
- レコード設計支援機能

なお，このマニュアルでは，SEWB+ 基本開発環境のプログラム構築支援機を SEWB+/CONSTRUCTION と表記しています。

■対象読者

このマニュアルは，SEWB+/CONSTRUCTION を使って，次の作業をする方を対象としています。
- クライアントサーバシステムやバッチシステムのプログラムを SEWB+/CONSTRUCTION で作るプログラマの方。
- システム開発で共用できるテンプレートを設計し，作成して提供する方。

ソースプログラムは，任意の言語で生成できますが，このマニュアルでは例に COBOL 言語を使っていますので，その点をあらかじめご了承ください。

■マニュアルの構成

このマニュアルは，次に示す編，章と付録から構成されています。

第 1 編　概要

第 1 章　SEWB+/CONSTRUCTION を使った AP 作成

SEWB+/CONSTRUCTION のテンプレート，データ定義，プログラム定義を使って，どのようにプログラムを開発するのか，その方法について説明しています。

第 2 編　操作編

第 2 章　データ定義

データ定義方法について説明しています。

第 3 章　プログラム定義

プログラム定義方法，およびソースプログラム生成について説明しています。

第 4 章　コンパイルと単体テスト

SEWB+/CONSTRUCTION で作成したソースプログラムのコンパイル，および単体テストについて説明しています。

第 3 編　テンプレート記述言語編

第 5 章　テンプレートとは

テンプレートとその作成方法について説明しています。

第 6 章　テンプレートを使った例題

例題を基に，テンプレートの作成方法について説明しています。

第 7 章　テンプレート記述言語

テンプレート記述言語の文法について説明しています。

付録 A　環境設定

AP 作成に必要な環境設定について説明しています。

付録 B　障害対策

障害発生時の，各ファイルの実行情報の取り方を説明しています。

付録 C　テンプレートおよび部品の紹介

クライアントサーバシステム用のプログラム作成およびバッチシステム用のプログラム作成を効率良く進めるために SEWB+ で用意されている，テンプレートおよび部品について説明しています。

付録 D　C/S システムサンプルプログラムの説明

SEWB+/CONSTRUCTION のサンプルプログラムについて説明しています。

付録 E　SEWB+/REPOSITORY で表示される名称

SEWB+/CONSTRUCTION の定義ファイルが SEWB+/REPOSITORY に格納された場合，表示される名称を示しています。

付録 F　SEWB+/CONSTRUCTION のインストールとアンインストール

SEWB+/CONSTRUCTION のインストールおよびアンインストールの方法について説明しています。

付録 G　前バージョンからの移行

前バージョンからの移行について説明しています。

付録 H　XML 形式ファイルのフォーマット

SEWB+/CONSTRUCTION の定義ファイルを XML 形式ファイルに出力する場合のファイル形式，および XML 形式ファイルの出力方法について説明しています。

付録 I　ファイルの互換性

SEWB+/CONSTRUCTION で作成したファイルのバージョン間の互換性について説明しています。

付録 J　用語解説

マニュアル中で使用している用語について説明しています。

■関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

SEWB+/REPOSITORY 運用ガイド（3020-3-N81）

SEWB+/REPOSITORY 辞書設計ガイド（3020-3-N82）

SEWB+ クライアントサーバシステム開発ガイド（3020-3-N83）

SEWB+/RECORD DEFINER ユーザーズガイド（3020-3-N85）

SEWB+　バッチシステム向けアプリケーションフレームワーク・部品　使用の手引（3020-3-711）

SEWB+/CS - DESIGN ユーザーズガイド（3020-3-770）

COBOL85 操作ガイド（3020-3-851）※ 1

COBOL85 操作ガイド（3020-3-873）※ 2

COBOL2002　操作ガイド（3020-3-D41）

WorkCoordinator Definer Version 3　ユーザーズガイド（3020-3-987）

EUR 導入（3020-7-053）※ 3

EUR 帳票設計（3020-7-054）※ 3

EUR 帳票出力（3020-7-055）※ 3

EUR 導入（3020-7-471）※ 4

EUR 帳票設計（3020-7-472）※ 4

EUR 帳票出力（3020-7-473）※ 4

Web Page Generator - Design ユーザーズガイド（3020-7-236）

Web Page Generator Enterprise ユーザーズガイド（3020-7-241）

画面・帳票サポートシステム XMAP3 プログラミングガイド 画面編（3020-7-583）

画面・帳票サポートシステム XMAP3 プログラミングガイド 帳票編（3020-7-584）

注※ 1

COBOL85 Version5.0 の場合にお読みください。

注※ 2

COBOL85 Version6.0，COBOL85 Version7.0 の場合にお読みください。

注※ 3

EUR Version 4 の場合にお読みください。

注※ 4

EUR Version 5 の場合にお読みください。

■マニュアル体系

SEWB+ シリーズのマニュアル体系を次に示します。

■読書手順

このマニュアルは，利用目的に合わせて次の個所をお読みいただくことをお勧めします。

マニュアルを読む目的	記述個所
SEWB+/CONSTRUCTION の概要について知りたい	1 章

マニュアルを読む目的		記述個所
データ定義方法，プログラム定義方法，およびコンパイルと単体テストについて知りたい	データ定義	2 章
	プログラム定義	3 章
	コンパイルと単体テスト	4 章
テンプレート記述言語について知りたい	テンプレートとは	5 章
	テンプレートを使った例題	6 章
	テンプレート記述言語	7 章
SEWB+/CONSTRUCTION の環境設定について知りたい		付録 A
障害発生時の，各ファイルの実行情報の取り方について知りたい		付録 B
プログラム作成を効率良く進めるために SEWB+ で用意されている，テンプレートおよび部品について知りたい		付録 C
SEWB+/CONSTRUCTION のサンプルプログラムについて知りたい		付録 D
SEWB+/CONSTRUCTION の定義ファイルが，SEWB+/REPOSITORY に格納された場合に表示される名称について知りたい		付録 E
SEWB+/CONSTRUCTION のインストールとアンインストールについて知りたい		付録 F
前バージョンからの移行について知りたい		付録 G
SEWB+/CONSTRUCTION の定義ファイルを XML 形式ファイルに出力する場合のファイル形式，および XML 形式ファイルの出力方法について知りたい		付録 H
SEWB+/CONSTRUCTION で作成したファイルのバージョン間の互換性について		付録 I
このマニュアルで使用する用語について知りたい		付録 J

■このマニュアルでの表記

このマニュアルでは，製品名および機能名を次のように表記しています。

正式名称	表記		
EUR Professional Edition	EUR		
End User Reporting			
Groupmax Object Server Version 6	Groupmax Object Server	Object Server	
Groupmax High-end Object Server Version 6	Groupmax High-end Object Server		
Microsoft(R) Visual Basic(R)	VB		
Microsoft(R) Visual C++(R)	VC++		
Microsoft(R) Windows(R) 2000 Professional Operating System 日本語版	Windows 2000 Professional	Windows 2000	Windows
Microsoft(R) Windows(R) 2000 Server Operating System 日本語版	Windows 2000 Server		

正式名称	表記
Microsoft(R) Windows(R) 2000 Advanced Server Operating System 日本語版	
Microsoft(R) Windows(R) 2000 Datacenter Server Operating System 日本語版	
Microsoft(R) Windows(R) XP Home Edition Operating System 日本語版	Windows XP
Microsoft(R) Windows(R) XP Professional Operating System 日本語版	
Microsoft(R) Windows Server(TM) 2003, Enterprise Edition x86 日本語版	Windows Server 2003
Microsoft(R) Windows Server(TM) 2003, Standard Edition x86 日本語版	
Microsoft(R) Windows Server(TM) 2003 R2, Enterprise Edition x86 日本語版	
Microsoft(R) Windows Server(TM) 2003 R2, Standard Edition x86 日本語版	
Microsoft(R) Windows Server(TM) 2003, Enterprise x64 Edition 日本語版	Windows Server 2003 x64
Microsoft(R) Windows Server(TM) 2003, Standard x64 Edition 日本語版	
Microsoft(R) Windows Server(TM) 2003 R2, Enterprise x64 Edition 日本語版	
Microsoft(R) Windows Server(TM) 2003 R2, Standard x64 Edition 日本語版	
Microsoft(R) Windows Vista(TM) Business 日本語版	Windows Vista

正式名称	表記	
Microsoft(R) Windows Vista(TM) Enterprise 日本語版		
Microsoft(R) Windows Vista(TM) Ultimate 日本語版		
SEWB+/CS - DESIGN	SEWB+/CS-DESIGN	
SEWB+ 基本開発環境のレコード設計支援機能	SEWB+/RECORD DEFINER	
SEWB+ 基本開発環境のリポジトリ管理機能	SEWB+/REPOSITORY	SEWB+/ REPOSITORY
SEWB+ 基本開発環境のリポジトリブラウザ機能	SEWB+/ REPOSITORY-BROWSER またはリポジトリブラウザ	

- このマニュアルでは，Microsoft Windows 2000 Datacenter Server Operating System を特定して示す場合は，Windows 2000 Datacenter Server と表記しています。
- このマニュアルでは「ディレクトリ」を「ドキュメントフォルダ」と表記しています。
- このマニュアルでは，TP モニタでの作業と CORBA での作業で共通する部分では，「AP（アプリケーションプログラム）」という表現を使用しています。CORBA で作業する方は，「AP」を「オブジェクト」に変えてお読みください。
- このマニュアルでは，パターンテンプレートをテンプレート，部品テンプレートを部品と表記しています。

■このマニュアルで使用する略語

このマニュアルで使用している略語を示します。

略語	正式名称
AP	Application Program
API	Application Program Interface
C/S システム	Client and Server System
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DB(RDB)	Data Base(Relational Data Base)
DBMS	Data Base Management System
DLL	Dynamic Linkage Library
DTD	Document Type Defnition
EUC	End User Computing
GUI	Graphical User Interface

略語	正式名称
ODBC	Open Database Connectivity
OLTP	Online Transaction Processing
ORB	Object Request Broker
OS	Operating System
PC	Personal Computer
RDBMS	Relational Data Base Management System
RPC	Remote Procedure Call
UOC	User Own Coding
XML	eXtensible Markup Language

■このマニュアルで使用している記号

このマニュアルで使用する記号を次のように定義します。

記号	意味
［ ］メニュー ［ ］ボタン など	メニュータイトル，メニュー項目，ボタンなどを示します。
［A］－［B］	－の前に示した［A］メニューから［B］を選択することを示します。

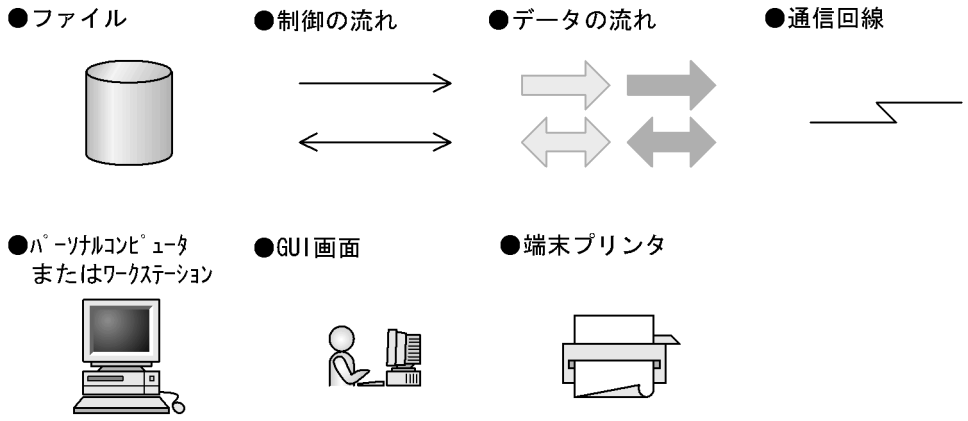
■構文の説明で使用する記号

文と関数の形式で使用している記号は表のとおりです。ただし，［ ］{ } はテンプレート中でも文字そのものとして記述でき，特別な意味を持ちます。例えば，［ ］は添字の配列を表すときに使います。また，{ } は一つのまとまりを示すときに使います。このように文字そのものを表しているときと，この表に記載された記号としての意味を持つときとを区別するために，このマニュアルでは，記号として使っている個所でその旨を記述しています。したがって，何も記述してないときは，文字そのものを表します。

記号	記 号 の 意 味
|	横に並べられた複数の項目に対する項目間の区切りを示し，「または」を意味します。
［ ］	［ ］内の項目は省略できることを表します。また，［ ］内に，複数の項目が並べてある場合は，すべてを省略するか，またはその中のどれか一つを選んで記述します。
{ }	{ } 内に，複数の項目が並べてある場合は，その中のどれか一つを選択することを示します。記号｜で区切られている場合は，そのうちの一つを選択します。

■図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。



■このマニュアルで使用する画面図と操作説明で使用する OS のメニュー項目について

このマニュアルでは，特に断りのないかぎり，次に示す OS の場合に表示される画面，メニュー名およびアイコンを使用して説明しています。これらの OS 以外を使用している場合，画面，メニュー名およびアイコンが異なることがあります。詳細は各 OS のマニュアルを参照してください。

機能名	OS
SEWB+/REPOSITORY	Windows Server 2003
SEWB+/REPOSITORY-BROWSER	Windows Vista
SEWB+/CONSTRUCTION	
SEWB+/RECORD DEFINER	

■ COBOL 言語について

このマニュアルの「4. コンパイルと単体テスト」は，プログラミング言語に日立 COBOL2002 を使用していることを前提に記載しています。日立 COBOL2002 以外で作成されたソースプログラムをコンパイル・テストする場合は，それぞれのコンパイラのマニュアルを参照してください。

■マニュアルとヘルプを効果的にお使いいただくために

SEWB+/CONSTRUCTION ではオンラインヘルプ（以降，ヘルプと略します）を提供しています。利用の目的に応じて，マニュアルとヘルプを使い分けることをお勧めします。

マニュアルは，SEWB+/CONSTRUCTION 全体の機能の使い方を理解したいときにお使いください。

ヘルプは，SEWB+/CONSTRUCTION を操作中に機能の詳細を知りたいときにお使いください。ダイアログの操作方法，メニューバーの利用方法の詳細，注意事項などを参照できます。

ヘルプでは次の内容を説明しています。
- 操作手順
- 用語およびダイアログやウィンドウの項目の説明
- データ定義ウィンドウの操作方法
- プログラム定義ウィンドウの操作方法
- テンプレート記述言語と文法

また，マニュアルの 6 章で使用している例題のテンプレートとソースプログラムは，SEWB+/CONSTRUCTION が提供しているサンプルプログラム中に「Manual Sample」として収められています。

■常用漢字以外の漢字の使用について

このマニュアルでは，常用漢字を使用することを基本としていますが，次に示す用語については，常用漢字以外の漢字を使用しています。
個所（かしょ） 貼り付け（はりつけ） 汎用（はんよう） 必須（ひっす）

■ 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。

目次

第 1 編　概要

第 2 編　操作編

図目次

表目次

1 SEWB+/CONSTRUCTION を使った AP 作成

SEWB+/CONSTRUCTION はテンプレート，データ定義およびプログラム定義を使って C/S システムやバッチシステムのプログラムを効率良く開発するための機能です。
この章では，SEWB+/CONSTRUCTION を使って AP を作成するための基本的な事柄について説明します。

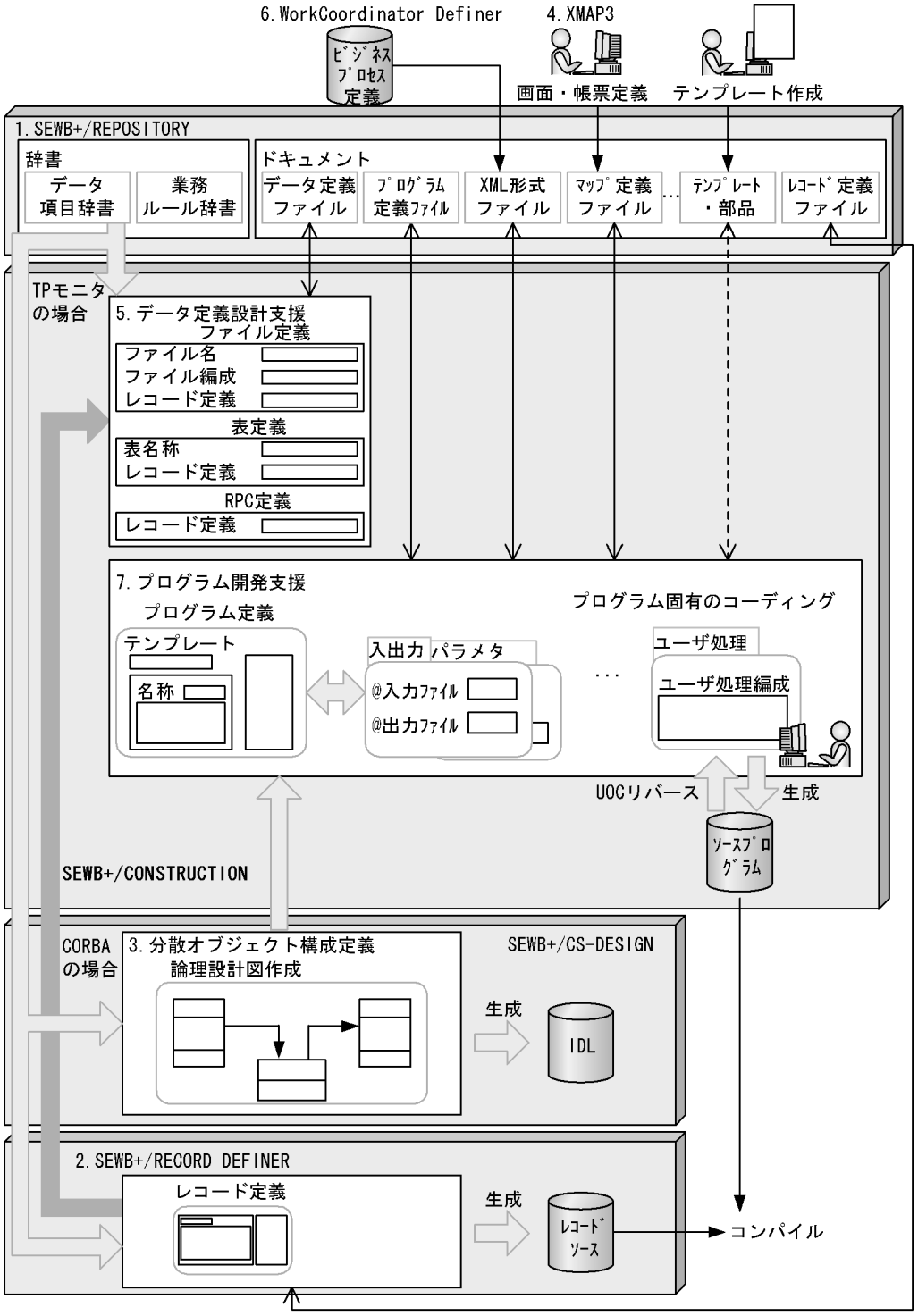
1.1 開発の流れ

SEWB+/CONSTRUCTION では，TP モニタや CORBA を使用する C/S システムのサーバ AP や，バッチシステムの AP を作成できます。

プログラムは，データ定義で定義したファイル・DB などを利用し，テンプレートの記述に従って定義します。

ここでは，SEWB+/CONSTRUCTION を使用してサーバ AP を作成するときの開発の流れおよびデータの流れについて説明します。

図 1-1　開発の流れ


6.WorkCoordinator Definer
4.XMAP3
ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ定義
画面・帳票定義
テンプレート作成
1.SEWB+/REPOSITORY
辞書
データ項目辞書
業務ルール辞書
ドキュメント
データ定義ファイル
ﾌﾟﾛｸﾞﾗﾑ定義ﾌｧｲﾙ
XML形式ファイル
ﾏｯﾌﾟ定義ファイル
ﾃﾝﾌﾟﾚｰﾄ・部品
ﾚｺｰﾄﾞ定義ファイル
TPモニタの場合
5.データ定義設計支援
ファイル定義
ファイル名
ファイル編成
レコード定義
表定義
表名称
レコード定義
RPC定義
レコード定義
7.プログラム開発支援
プログラム定義
テンプレート
名称
入出力パラメタ
@入力ﾌｧｲﾙ
@出力ﾌｧｲﾙ
プログラム固有のコーディング
ユーザ処理
ユーザ処理編成
UOCリバース
生成
ｿｰｽﾌﾟﾛｸﾞﾗﾑ
SEWB+/CONSTRUCTION
CORBAの場合
3.分散オブジェクト構成定義
論理設計図作成
SEWB+/CS-DESIGN
生成
IDL
2.SEWB+/RECORD DEFINER
レコード定義
生成
ﾚｺｰﾄﾞｿｰｽ
コンパイル

（凡例） <-> ：SEWB+/REPOSITORYに格納されている資源をローカルに持ってきたり（チェックアウト），作業が終わった資源を再びSEWB+/REPOSITORYに返したりする（チェックイン）ことを表す。
チェックイン・チェックアウトの詳細は，マニュアル「SEWB+/REPOSITORY 運用ガイド」を参照のこと。
<-> ：SEWB+/REPOSITORYに格納されている資源をローカルに持ってこないで，ネットワーク上で参照していることを表す。

1. SEWB+/REPOSITORY
   システム分析や DB 設計などの上流工程で抽出された，システム開発に必要な資源は，リポジトリで管理されています。
   開発に必要な資源のうち，SEWB+/CONSTRUCTION で利用する主なものを次に示します。
   - テンプレート・部品（ドキュメント）
   - 辞書（データ項目・業務ルール※）
   - 各種仕様書（ドキュメント）

   注※
   業務ルールとは，データ項目に着目して，データ項目特有の処理を部品化したものです。
2. SEWB+/RECORD DEFINER
   データ定義で利用するレコード定義を作成します。また，クライアント側のインタフェーステーブル用にレコードソースを生成することもできます。
3. 分散オブジェクト構成定義（CORBA でのシステム開発の場合）
   CORBA で動作する AP を作成する場合，SEWB+/CS-DESIGN を利用して，分散オブジェクトの論理設計図を作成し，IDL を生成します。
4. XMAP3（画面や帳票を使った AP を作成する場合）
   SEWB+/CONSTRUCTION では，XMAP3 で作成した画面や帳票の定義情報をマップ定義ファイルから取得し，ソースプログラムを生成することができます。
5. データ定義設計支援（データ定義）
   SEWB+/CONSTRUCTION での AP 作成に使用するファイルや DB を設計します。ファイルや DB の設計には，辞書のレコード定義（最上位結合項目）や SEWB+/RECORD DEFINER で作成したレコード定義を利用します。
6. WorkCoordinator Definer
   SEWB+/CONSTRUCTION では，XML ファイルをソースプログラム生成のパラメタとして利用できます。WorkCoordinator Definer で作成したビジネスプロセス定義ファイルなどの XML 形式ファイルが使用できます。WorkCoordinator Definer については，マニュアル「WorkCoordinator Definer Version 3　ユーザーズガイド」を参照してください。
7. プログラム開発支援（プログラム定義）
   プログラム定義でテンプレートを選び，必要な項目や業務ルールを定義します。また，必要に応じて，AP 固有の処理のコーディング（ユーザ追加処理）を追加してプログラムを定義します。定義終了後，ソースプログラムを生成します。

1.2 テンプレートと AP 作成

テンプレートはプログラムのひな型のようなものです。

テンプレートを利用することによって，AP 作成の標準化や効率化を進められます。

1.2.1 テンプレートの役割

テンプレートは，通常，業務システム全体を把握する専任者（テンプレート作成者）によって，テンプレート記述言語とプログラミング言語を使用して作成されます。テンプレートには，AP にどんな処理をさせるか，それにはどんなデータ定義が必要かなどのプログラムの処理および構成が書かれています。テンプレートの内容は特定の AP 用ではなく，類似した AP の作成に広く流用できるように，使用されているデータ項目やファイルの名前などは**可変記号**と呼ばれる仮の記号で書かれています。可変記号には，プログラム作成者が値を設定できます。これらは，プログラム定義で入力された情報を基に，ソースプログラム生成時に固有の情報に変更されます。

また，同じテンプレート記述言語とプログラミング言語で作成される部品には，RPC の処理や DB アクセスの処理など，複雑な処理や複数の AP で共用される処理を記述できます。

テンプレートや部品を使用すると，プログラム作成者が難しい処理をコーディングしたり，同じ処理を何度もコーディングしたりする手間が省けるので生産性と信頼性が向上し，プログラムの品質も均一にできます。標準的な業務処理の AP を作成する場合は，SEWB+/CS-FRAMEWORK や SEWB+/BATCH-FRAMEWORK を使用すると，効率良く作業を進められます。

なお，テンプレートの詳細は「第 3 編　5.　テンプレートとは」を参照してください。

1.3 データ定義と AP 作成

SEWB+/CONSTRUCTION では，サーバ AP のほかに，クライアント側のインタフェース用のテーブルを作成できます。

データ定義では，AP 中で使用するファイルや DB の定義およびクライアント側の AP と連携するためのインタフェースを定義します。

1.3.1 データ定義の役割

データ定義では，必要に応じて次の情報の入力をプログラム作成者に促すことができます。

- **ファイル情報**
  ファイルの入出力に必要なファイル名，ファイルの編成方法，およびレコード構造を定義します。
- **RDB 情報**
  RDB の入出力に必要な表名称，およびレコード構造を定義します。
- **DAM 情報**
  DAM（Direct Access Method）ファイルの入出力に必要な論理ファイル名，入出力ブロックサイズ，およびレコード構造を定義します。DAM ファイルは，分散トランザクション処理機能 OpenTP1 が提供する構造型ファイルサービスの一つです。ファイル内の相対位置を指定することによって，ブロック単位でダイレクトアクセスができる直接編成ファイルです。
- **TAM 情報**
  TAM（Table Access Method）テーブルは，OpenTP1 が提供する構造型ファイルサービスの一つです。主記憶に常駐してダイレクトアクセスができる直接編成ファイルです。この TAM テーブルの入出力に必要なテーブル名，TAM テーブルにアクセスするときのキー項目名，およびレコード構造を定義します。ただし，キー項目名は，COBOL ソースプログラムやレコードには，直接生成されません。指定されたキー項目名の長さは，TAM テーブルの入出力時に，キーの長さを宣言する項目の領域長に反映されます。
- **RPC 情報**
  リモートプロシジャコールで，SPP（Service Providing Program）へサービスを要求するときに必要な入力パラメタと，応答領域を定義します。SPP は，クライアントプログラムからのサービス要求を処理し，結果を返すサーバ側のサービス提供プログラムです。
- **メッセージ情報**
  PC や WS，端末，メインフレームなどとネットワークを介した他システムとのサービスの要求と提供を，メッセージを使って行う形態をメッセージ送受信形態といいます。メッセージはメッセージ処理プログラム MHP（Message Handling Program）が処理します。この MHP に対して送受信するメッセージを定義します。

- **ユーザジャーナル情報**
  ユーザジャーナルの出力に必要なユーザジャーナルコードおよびユーザジャーナルレコード構造を定義します。ユーザジャーナルは，障害発生時などのファイル回復や稼働統計情報を分析するために取得する情報です。
- **メッセージログ情報**
  メッセージログの出力に必要なメッセージログレコード構造を定義します。メッセージログは，オンラインシステムに障害が起きていないかどうかを監視するための情報です。
- **共通作業領域情報**
  プログラムの作業領域や，ほかのプログラムを呼び出すときの CALL 文の引数に使用する連絡領域のレコード構造を定義します。

データ定義ウィンドウのファイル情報の定義例を図 1-2 に示します。

図 1-2　データ定義ウィンドウの例

このデータ定義ウィンドウで定義されたレコード構造は，ソースプログラム生成時に自動的に生成されます。また，COBOL 言語で生成する場合は COPY メンバとして，C 言語で生成する場合はヘッダファイルとして生成される情報もあります。ただし，C 言語の場合にレコード定義を参照しているときは，レコード構造は生成されません。

なお，データ定義の詳細は「第 2 編　2.　データ定義」を参照してください。

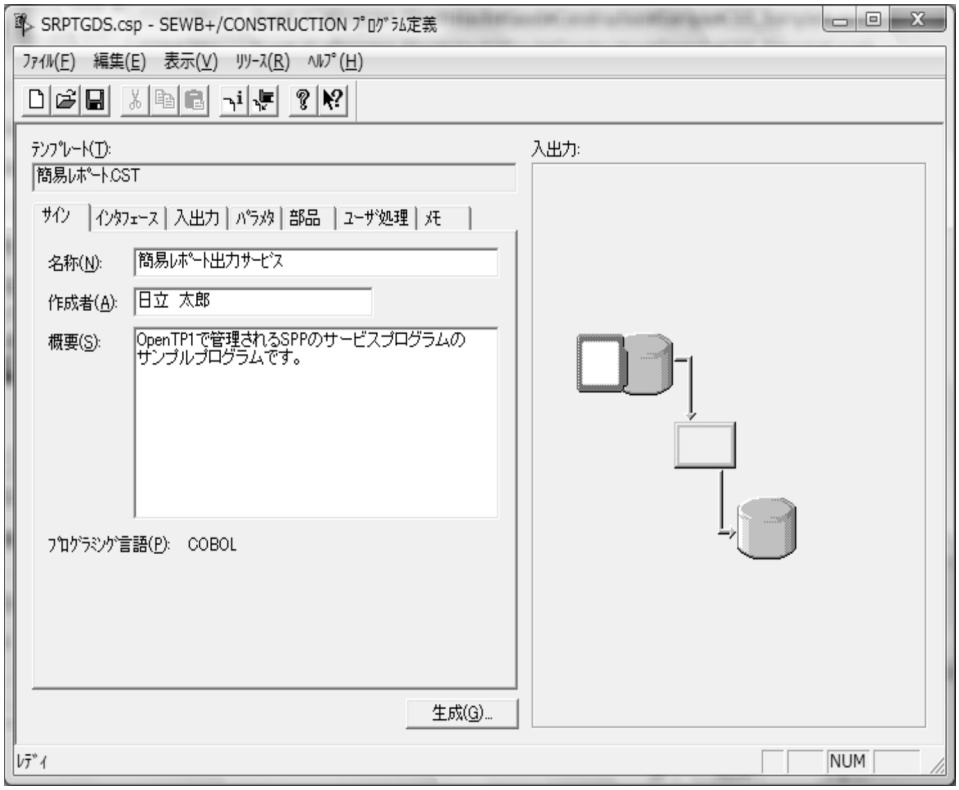
1.4　プログラム定義と AP 作成

プログラム定義では，テンプレートおよびデータ定義情報を取り込んで，ソースプログラム生成に必要な情報を設定します。

1.4.1　プログラム定義の役割

テンプレートは幾つものプログラムを作成するときに共用できるように作るため，プログラム固有の情報を定義する仕掛けが必要です。それがプログラム定義です。プログラム作成者は，定義ごとのタブを持つプログラム定義ウィンドウで入出力やパラメタの定義，UOC（ユーザ追加処理）の記述および業務ルール処理の設定などをし，テンプレートだけでは不足している固有情報を定義します。

図 1-3　プログラム定義ウィンドウの例



テンプレート作成者は，プログラム作成者に定義してもらいたい情報がプログラム定義ウィンドウに項目として表示されるように，テンプレート内にテンプレート記述言語を使って記述しておきます。プログラム作成者は，プログラム定義を終了したあとに，ソースプログラムを生成し，コンパイルします。

なお，プログラム定義の詳細は「第 2 編　3.　プログラム定義」を参照してください。

2 データ定義

SEWB+/CONSTRUCTION では，サーバ AP，およびクライアント側のインタフェース用のテーブルを作成できます。
データ定義では，AP で使用するファイルや DB の定義およびクライアント側の AP と連携するためのインタフェースを定義します。

2.1 システム開発で使用するデータの決定

作成する AP で使用するファイルや DB（データ定義の種別），およびレコードの内容を決めておきます。

C/S システムの AP を作成する場合は，データ定義種別を定義できます。それぞれのデータ定義種別とその内容，および TP モニタや CORBA との相性を表 2-1 に示します。

表 2-1　C/S システムの AP 作成時に定義できるデータ定義種別と情報

種　別	内　　容	TP モニタを利用する	TP モニタを利用しない	CORBA を利用する
ファイル	ファイルの入出力に必要な「ファイル名」,「ファイル編成」および「レコード構造※」を定義する。	○	○	○
RDB	RDB の入出力に必要な「表名称」および「レコード構造※」を定義する。	○	○	○
DAM	DAM ファイルの入出力に必要な「DAM ファイル名」,「入出力ブロックサイズ」および「レコード構造※」を定義する。	○	×	×
TAM	TAM テーブルの入出力に必要な「TAM テーブル名」および「レコード構造※」を定義する。	○	×	×
RPC 入力パラメタ	リモートプロシジャコールに必要な「入力パラメタ※」を定義する。	○	×	×
RPC 応答領域	リモートプロシジャコールに必要な「応答領域※」を定義する。	○	×	×
メッセージ	メッセージ送受信に必要な「メッセージ※」を定義する。	○	○	×
ユーザジャーナル	ユーザジャーナルの取得に必要な「ユーザジャーナルコード」「ユーザジャーナルレコード構造※」を定義する。	○	×	×
メッセージログ	メッセージログの出力に必要な「メッセージログレコード構造※」を定義する。	○	×	×
共通作業領域	AP 間で共通の作業領域や，AP 間の CALL の引数に使用する「レコード構造※」を定義する。	○	○	○

（凡例）
○：使用できる
×：使用できない

注※

レコード構造，RPC 入力パラメタ，RPC 応答領域，メッセージ，ユーザジャーナルレコード構造およびメッセージログレコード構造は，データ項目辞書の結合項目の構成機能，または SEWB+/RECORD DEFINER で定義します。データ定義では，あらかじめ定義された結合項目の構成（レコード構造）を選んで，各定義情報に反映させます。

2.2 データ定義

データ定義では，辞書のレコード定義（最上位結合項目）または SEWB+/RECORD DEFINER で作成したレコード定義を指定します。

作成する AP で使用するファイルや DB が決まったら，実際に AP 中で使用するファイル名，DB 名やレコード構造を定義します。これらの情報は，SEWB+/CONSTRUCTION のデータ定義ウィンドウ上で定義できます。

データ定義で定義した情報はデータ定義ファイルに格納され，インタフェーステーブル用のレコードソースとして生成されます。また，データ定義情報はソースプログラム生成時に展開されてソースプログラムに反映されたり，プログラム定義ウィンドウからの要求に応じて，テンプレート中の可変記号の値として設定されたりします。

データ定義情報は，プログラム定義ウィンドウからの要求に応じて，変更することができます。

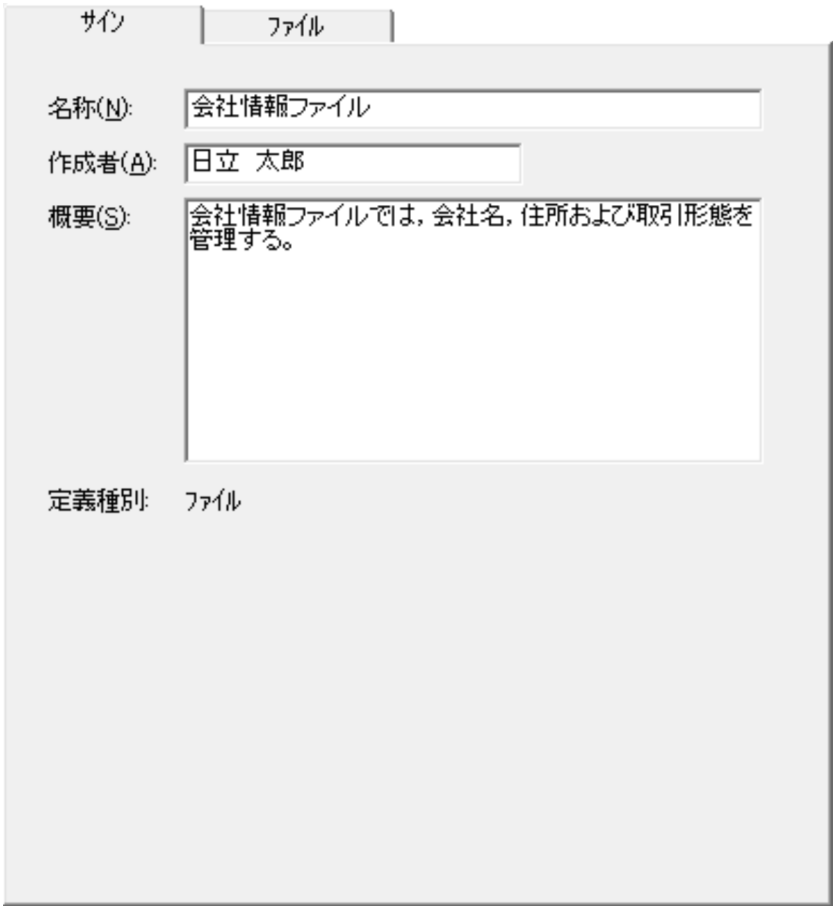
2.2.1 データ定義種別の選択

データ定義を新しく定義する場合は，まずデータ定義種別を選んで指定します。

データ定義種別を選んだら，それぞれに対する情報を定義します。情報の定義はデータ定義ウィンドウで行います。定義ウィンドウは，[サイン] タブとデータ定義タブから構成されています。

2.2.2 データ定義の概要情報の記述－[サイン] タブ

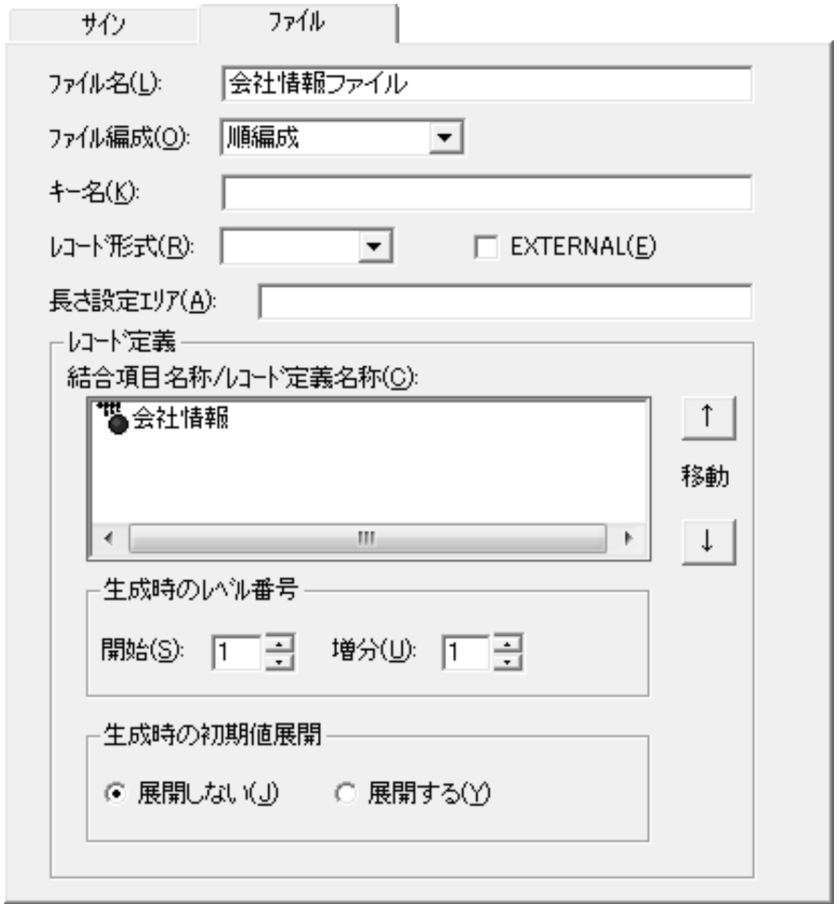
[サイン] タブを次に示します。



[サイン] タブには，作成するデータ定義の名前，作成者名，および概要を指定します。

2.2.3 各定義種別の定義情報の記述－各データ定義タブ

データ定義種別に対応したデータ定義をします。ここでは，例としてファイルを定義するときのタブを次に示します。



ここでは，データ定義種別ごとのデータ定義タブについて説明します。

なお，各データ定義タブにある「生成時のレベル番号」および「生成時の初期値展開」はデータ定義情報がレコード生成されるときのレベルおよび初期値展開を指定するものです※。すべての定義タブに共通しているので，これ以降での説明は省略しています。

注※

プログラミング言語に C を使用している場合は，レベル番号および初期値展開は無視されます。

●［ファイル］タブ

［ファイル］タブには，作成するファイルのファイル名，ファイル編成，キー名，レコード形式，EXTERNAL，長さ設定エリア，および使用するレコードを指定します。

● [RDB] タブ

[RDB] タブには，作成する DB の表名称，および使用するレコードを指定します。表名称は，直接入力するか，または [表選択] ボタンを押して表示される [表選択] ダイアログから，表名称を選んで指定します。

● [DAM] タブ

[DAM] タブには，作成する DAM ファイルの論理ファイル名，入出力ブロックサイズ，および使用するレコードを指定します。

● [TAM] タブ

[TAM] タブには，作成する TAM テーブルの TAM テーブル名，使用するレコードおよびキーを指定します。キーは，[ツール] ― [キー項目の選択] を選んで [データ項目選択] ダイアログを表示させ，その中からキーとなる項目を選んで指定します。

● [RPC 入力パラメタ] タブ

[RPC 入力パラメタ] タブには，作成する RPC インタフェースの入力パラメタに使用するレコードを指定します。

注

レコード定義に指定するレコードの構造に，入力パラメタ長格納領域は含まれません。

● [RPC 応答領域] タブ

[RPC 応答領域] タブには，作成する RPC インタフェースの応答領域に使用するレコードを指定します。

注

レコード定義に指定するレコードの構造に，応答領域長格納領域は含まれません。

● [メッセージ] タブ

[メッセージ] タブには，作成する MHP インタフェースのメッセージに使用するレコードを指定します。

注

メッセージ定義に指定するレコードの構造に，MCF（Message Control Facility）で使用するヘッダ領域は含まれません。

● [ユーザジャーナル] タブ

[ユーザジャーナル] タブには，作成するユーザジャーナルのコードおよび使用するレコードを指定します。

注

ユーザジャーナル定義に指定するレコードの構造に，ユーザジャーナル長およびユーザジャーナルコード格納領域は含まれません。

● [メッセージログ] タブ

[メッセージログ] タブには，作成するメッセージログに使用するレコードを指定します。

注

メッセージログ定義に指定するレコードの構造に，メッセージログ長格納領域は含まれません。

●[共通作業領域] タブ

[共通作業領域] タブには，複数の AP 間で共通に使用される作業領域のレコードを指定します。

2.2.4 レコードの選択

レコードとは，データ項目辞書のデータ項目の結合機能を利用して作成された，結合項目構成の最上位の結合項目，または SEWB+/RECORD DEFINER で作成されたレコード定義のことを指します。

- 結合項目の場合：[ツール] ― [最上位結合項目の検索] から表示される最上位結合項目を選んで指定します。
- レコード定義の場合：[ツール] ― [レコード定義の検索] から表示されるレコード定義を選んで指定します。

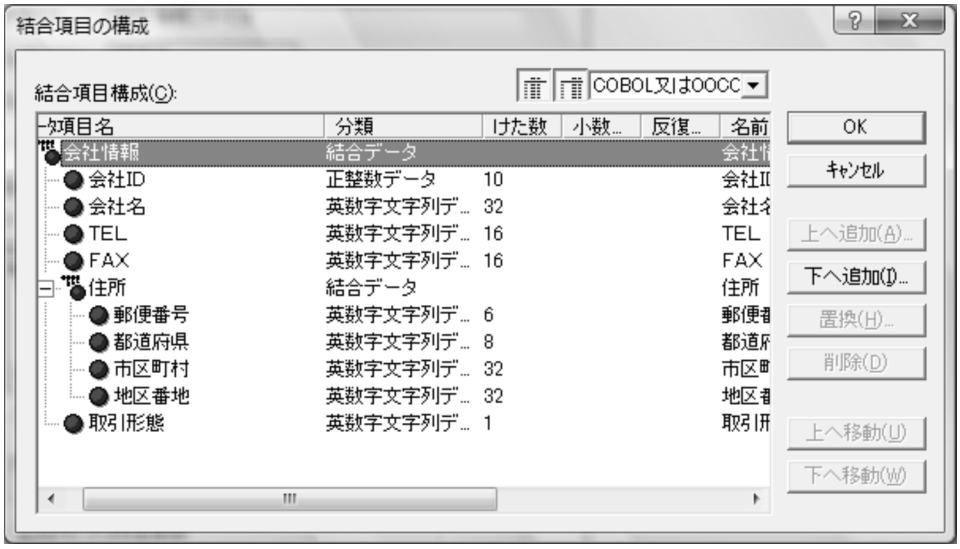
選択された最上位結合項目またはレコード定義は，結合項目名称 / レコード定義名称欄に反映されます。

2.3　レコード構成の定義

レコードとは，データ項目辞書のデータ項目の結合機能を利用して作成された，結合項目構成の最上位の結合項目，またはSEWB+/RECORD DEFINERで作成されたレコード定義のことを指します。

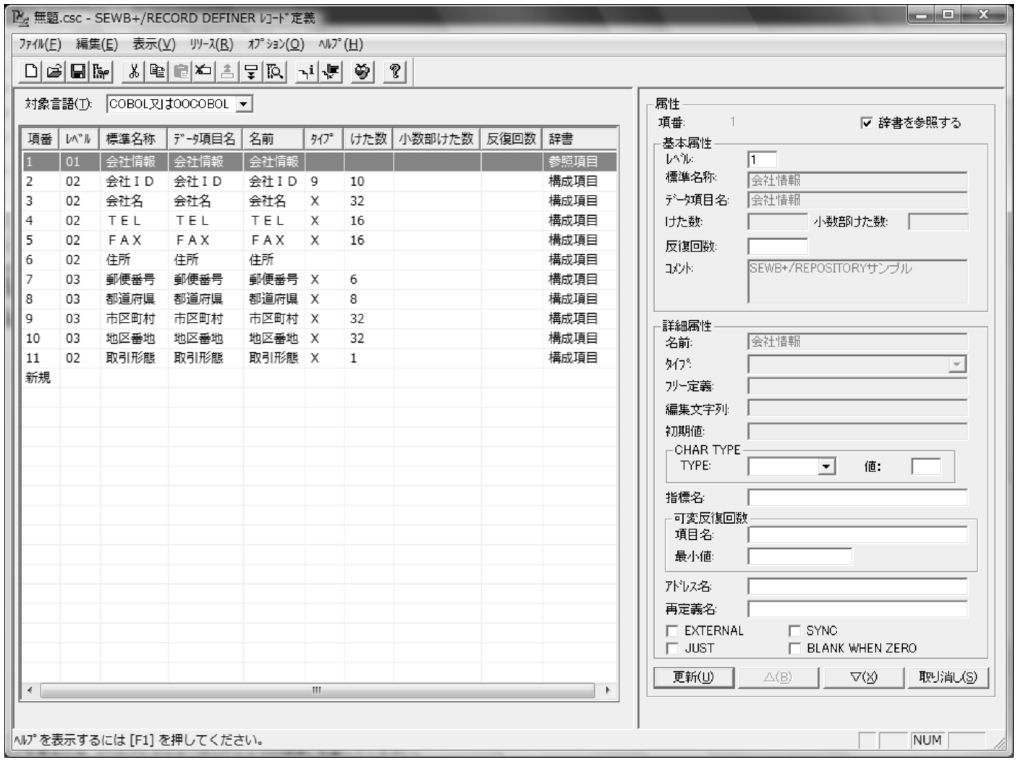
各データ定義の「結合項目名称/レコード定義名称」で結合項目を指定した場合は，[ツール]－[結合項目の構成]で表示されるダイアログで結合項目の構成を変更できます。図2-1に[結合項目の構成]ダイアログを示します。

図2-1　[結合項目の構成]ダイアログ



また，各データ定義の「結合項目名称/レコード定義名称」で，レコード定義を指定した場合は，SEWB+/RECORD DEFINERでレコードの構成を変更できます。図2-2にレコード定義を示します。レコード定義の詳細は，マニュアル「SEWB+/RECORD DEFINERユーザーズガイド」を参照してください。

図 2-2 レコード定義



2.3.1 SEWB+/CONSTRUCTION からレコードを更新するときの注意事項

AP で使用するデータ項目がレコード中にない場合，SEWB+/CONSTRUCTION からデータ項目辞書を呼び出して，データ項目間に新しく関連を付けてレコードの内容を更新できます。データ項目に新しく関連を付けたいときは，[ツール] ― [結合項目の構成] で表示されるダイアログを使用します。ただし，SEWB+/CONSTRUCTION のデータ定義では，結合項目および構成項目の構成は変更できますが，最上位結合項目を作成することはできません。最上位結合項目を作成する場合は，SEWB+/REPOSITORY-BROWSER を使用します。最上位結合項目の作成方法については，マニュアル「SEWB+/REPOSITORY 辞書設計ガイド」およびリポジトリブラウザのヘルプを参照してください。

2.4 クライアント側のインタフェーステーブル用レコードソースの生成

SEWB+/CONSTRUCTION でソースプログラムを生成する場合，レコード定義情報はソースプログラム中に展開します。また，クライアント側の AP，RPC およびメッセージのサーバ側にアクセスするためのインタフェーステーブルは，データ定義から生成することによって，インタフェースの不一致を防げます。インタフェーステーブル用レコードソースは，データ定義ウィンドウまたはコマンドで生成できます。ただし，直接生成できるのは，C または COBOL の場合です。それ以外の言語で生成する場合には，生成用のテンプレートを作成する必要があります。

2.4.1 データ定義ウィンドウからの生成

[ファイル] ― [生成] を選ぶと，[生成] ダイアログが表示されます。[生成] ダイアログの「生成先」および「生成言語」を指定し，[OK] ボタンを押すとレコードソースが生成されます。

2.4.2 コマンドでの生成（レコードソース）

レコードソースを生成するには，コマンド「CSDDGEN.EXE」を実行させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

形　式

```
CSDDGEN.EXE△/i△入力ファイル名
          △/o△出力ファイル名
          △/l△{COBOL｜C}
          △[/t△トレースファイル名]
          △[/r]
          △[/rp△レコード定義ファイルパス名]
          △[/d△インデンテーション長]
          △[/pi△PICTURE句などの生成位置]
          △[/cf△COBOLソース正書法]
          △[{/ds|/s}]
          △[/si△一連番号初期値]
          △[/sa△一連番号増分値]
```

（凡例）

△：1 文字以上の空白を示します。

注

［ ］は省略できることを示します。

{ } はどちらかを指定することを示します。

解 説

/i △入力ファイル名：

データ定義ファイル（.csd または .cse）のファイル名を指定します。

/o △出力ファイル名：

生成するレコードソースの出力先のファイル名を指定します。ファイル名を指定する場合は，拡張子を指定できます。

/l △生成言語種別：

生成するレコードソースの言語種別を「COBOL」または「C」で指定します。

表 2-2 生成言語種別の種類

生成言語種別	生成するソース
COBOL	COBOL 言語の COPY メンバを生成する
C	C 言語のヘッダファイルを生成する※1

/t △トレースファイル名：

トレース情報の出力先ファイル名を指定します。
トレース情報には，プログラムの実行などの情報が出力されています。

/r：

データ定義ファイルを最新状態にする場合に指定します。

/rp △レコード定義ファイルパス名※2：

レコード定義ファイル（.csc）のファイルパス名を指定します。

/d △インデンテーション長※2：

インデンテーション長を 0 ～ 8 の範囲で指定します。

/pi △ PICTURE 句などの生成位置※2：

PICTURE 句などの生成位置の範囲を指定します。

- COBOL ソースの形式が固定形式またはホスト向け固定形式の場合は 8 ～ 60。
- COBOL ソースの形式がフリー形式の場合は 1 ～ 60。

/cf △ COBOL ソース正書法※2：

COBOL ソースの形式を「fix」,「free」または「hostfix」で指定します。「fix」は COBOL ソースを固定形式で生成します。「free」は COBOL ソースをフリー形式で生成します。「hostfix」は COBOL ソースをホスト向け固定形式で生成します。

注

ホスト向け固定形式では，2 バイトコードがある場合，機能キャラクタ占有領域が考慮されます。また，項目名が 30 バイト（機能キャラクタ占有領域を含めて）を超えた場合，31 バイト以降は切り捨てられます。

/ds※2：

COBOL ソース正書法が固定形式またはホスト向け固定形式の場合に，一連番号を生成ソースに付加しないときに指定します。

/s※2：

COBOL ソース正書法が固定形式またはホスト向け固定形式の場合に，一連番号を生成ソースに付加する場合に指定します。

/si △一連番号初期値※2：

一連番号を生成ソースに付加する場合に，一連番号の初期値を 1 ～ 999999 の範囲で指定します。

/sa △一連番号増分値※2：

一連番号を生成ソースに付加する場合に，一連番号の増分値を 1 ～ 999999 の範囲で指定します。

注

大文字と小文字は区別されません。

注※ 1

レコード定義ファイルを使用するときは，C 言語のレコードソースは生成できません。C 言語を指定してソースの生成を行った場合，メッセージが出力され生成が中止されます。

注※ 2

何も指定しない場合は，環境設定で定義されている情報が有効になります。

2.5 データ定義のポイント

2.5.1 サーバ AP のレコードソース

● ソースレコードの展開先

SEWB+/CONSTRUCTION で用意されているサンプルプログラムでは，サーバ側のレコードソースは，生成ソースプログラム中に展開することを前提にしています。ソースを管理しやすくするため，コンパイル環境の負荷を軽くするために，レコードソースをソースプログラム中に展開することをお勧めします。

● レコードソースの変更

レコードソースがサーバ側の複数の AP で共用されている場合にレコードソースを変更しても，COBOL 開発マネージャのリビルドを利用すると，インタフェースの不一致を避けられます。リビルドでは，データ定義の変更時に，そのデータ定義を使用しているすべてのプログラム定義が自動的に再生成されます。

注

サーバ側の AP に COPY メンバを取り込むことができます。ただし，この場合は，テンプレート中に，COPY を展開させる位置を指示する記述が必要です。

2.5.2 キャラクタ（文字列）ベースでのインタフェーステーブル作成

インタフェーステーブルをキャラクタ（文字列）ベースで作成することで，次の場合に対処できます。

● クライアント側とサーバ側で使用している言語が異なる場合

SEWB+/CONSTRUCTION では，COBOL と C のインタフェーステーブルを生成できますが，数値が入っていると言語によるバウンダリ調整の違いで，データが不一致になる場合があります。そのため，キャラクタ（文字列）ベースでインタフェーステーブルを作成することをお勧めします。

● クライアント側とサーバ側の OS が異なる場合

インタフェーステーブルに数値が入っている場合，同じ言語でも，OS やマシン（ハードウェア）が違うと，テーブルの大きさが不一致になる場合があります。そのため，キャラクタ（文字列）ベースでインタフェーステーブルを作成することをお勧めします。

テンプレート中にあらかじめ，文字列と数値間の両方向への変換をする処理を組み込んでおくと便利です。

3 プログラム定義

プログラム定義では，テンプレートおよびデータ定義情報を取り込んで，ソースプログラム生成に必要な情報を設定します。また，ユーザ追加処理をプログラム定義ファイルに取り込むことができます。

3.1 プログラム定義の準備

プログラム定義では，作成したテンプレートやデータ定義情報を編集し，ソースプログラムを生成します。ソースプログラム生成に必要な情報のうち，次に示すものは，すでに定義されていなければなりません。

(テンプレートで定義されている情報)

入出力項目の宣言
インタフェースの宣言
指示項目の宣言
プログラムの処理
部品およびユーザ追加処理の挿入位置
業務ルール処理の挿入位置

(データ定義で定義されている情報)

ファイル情報
RDB 仕様情報
インタフェース情報
共通作業領域

(論理設計図で定義されている情報) ※1

オブジェクト情報
インタフェース情報

(マップ定義ファイルで定義されている情報) ※2

共通情報
オブジェクト情報

プログラム定義では，これらの情報以外でソースプログラム生成に必要な情報を定義します。プログラム定義で定義するものを次に示します。

(プログラム定義で定義する情報)

使用するテンプレート
作成するプログラムの概要情報
使用するインタフェース項目
入出力項目
指示項目
使用する部品
ユーザ追加処理
使用する業務ルール処理

注※ 1

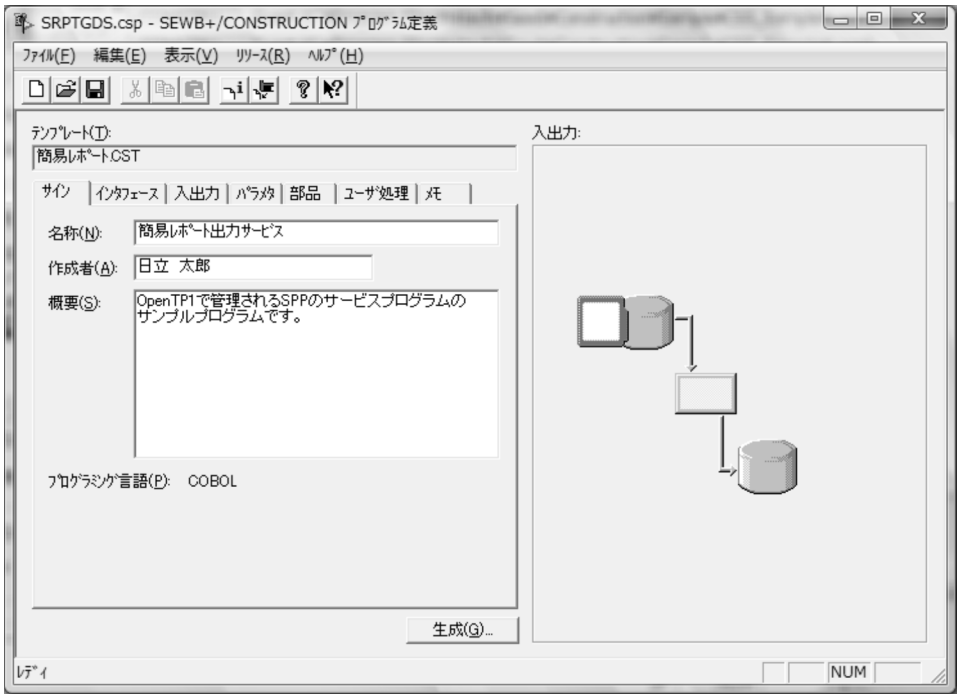
CORBA の C/S システム開発に必要です。

注※ 2

XMAP3 を利用した AP の開発に必要です。

3.1.1　プログラム定義の起動

プログラム定義を起動すると，プログラム定義のメインウィンドウが表示されます。新規に起動する場合は，テンプレート選択ファイルが表示されます。「3.1.2　使用するテンプレートの選択」を参照して，テンプレートを選んでください。テンプレートを選ぶと，テンプレートに関する情報がメインウィンドウの [サイン] タブに反映されます。以降のプログラム定義の作業の目安にしてください。また，既存のプログラム定義を変更する場合，入出力構成の情報がメインウィンドウに反映されます。メインウィンドウを次に示します。



(1) 次回作業時や，引継ぎに役立つ情報

● [サイン] タブ

[サイン] タブには，選んだテンプレートから作成したプログラムの情報を記述できます。プログラムの名称や，作成者名，およびプログラムの概要を書いておくと，次回に作業するときや，業務を引き継ぐときなどに参考になります。

● 入出力の欄

入出力の欄には，[入出力] タブでの設定を基にした入出力の構成図が表示されます。

(2) 起動時の注意事項

● プログラム定義ファイルがないまたは内容が不正な場合（既存起動時）

既存のプログラム定義ファイルを使用する場合は，プログラム定義ファイルを入力して起動します。

ただし，指定されたプログラム定義ファイルがない場合，またはファイルの内容が不正な場合は，エラーダイアログを表示して，プログラム定義をファイル未入力で起動します。

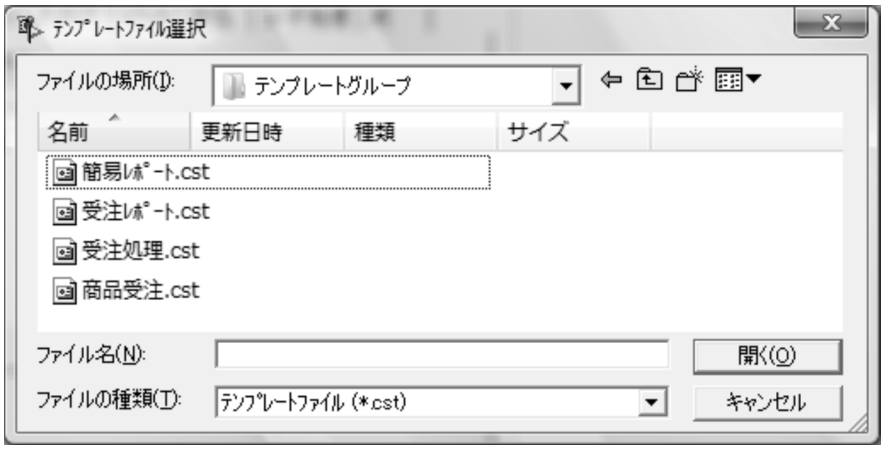
● テンプレートファイルまたはプログラム定義ファイルを入力できなかった場合（新規起動時または既存起動時）

新規起動または既存起動でテンプレートファイルまたはプログラム定義ファイルを入力できなかった場合は，ファイル未入力状態で起動します。この状態で起動した場合は，定義操作に関係のあるメニューおよびボタンは不活性状態になります。

3.1.2 使用するテンプレートの選択

プログラム定義ではまず，プログラムをどのテンプレートから作成するかを決めなければなりません。その後のプログラム定義は，すべてここで選んだテンプレートの情報が基になります。プログラム定義を新規に作成するときは，テンプレートを選ぶための [テンプレートファイル選択] ダイアログが表示されます。

[テンプレートファイル選択] ダイアログを次に示します。



(1) 効率良いテンプレートの検索

●「環境設定」で設定する

[テンプレートファイル選択] ダイアログに初期表示されるフォルダは，「環境設定」で変更できます。また，「環境設定」でパスを複数指定していると，[パス選択] ダイアログが表示されます。[パス選択] ダイアログについては，「(2) [パス選択] ダイアログ」を参照してください。

● 名前やプロパティ情報で判断する

テンプレートの概要は，名前から判断できます。また，テンプレートファイルのプロパティ情報を参照すると，名称，作成者および概要を見ることができます。

(2) [パス選択] ダイアログ

[パス選択] ダイアログには，「環境設定」で指定したパス，および前回 [テンプレートファイル選択] ダイアログで選択したファイルのパスが表示されます。ここでパスを選択すると，[テンプレートファイル選択] ダイアログの「ファイルの場所」に設定されます。

選択したパスがない場合，または [パス選択] ダイアログでパスを選択しなかった場合には，状況に応じて次の優先順位でパスが設定されます。

状況 1

「環境設定」でパスが複数指定されていて，[パス選択] ダイアログでパスを選択した場合

優先順位	パス
1	[パス選択] ダイアログで選択した検索パス
2	前回，[テンプレートファイル選択] ダイアログで選択したテンプレートファイルのパス
3	カレントパス

状況 2

「環境設定」でパスが複数指定されていて，[パス選択] ダイアログでパスを選択しなかった場合

優先順位	パス
1	前回，[テンプレートファイル選択] ダイアログで選択したテンプレートファイルのパス
2	「環境設定」で指定されている最初のパス
3	カレントパス

状況 3

「環境設定」でパスが複数指定されていない場合

優先順位	パス
1	「環境設定」で指定されているパス
2	カレントパス

3.1.3 プログラム定義を始める前に確認しておくこと

テンプレートを選ぶと，プログラム定義を開始できます。ただし，プログラム定義をする前に次のことを確認しておきます。

(1) 適切な情報を選んでいること

生成されるソースプログラムの内容は，使用されるテンプレートやデータ定義などで決まります。そのため，選んだテンプレートファイルや，データ定義ファイルなどが適切かどうかを確認してください。確認するときの目安として，テンプレートやデータ定義などに付けられた名前があります。また，[リソースステータス] ダイアログを表示させると，現在プログラム定義に取り込んでいるテンプレートファイルとデータ定義ファイルなどの名前と最新の更新日時も確認できます。[リソースステータス] ダイアログは，[リソース] ― [ステータス] で表示できます。

● 必要な情報を最新状態にしたい場合

選んだテンプレートファイルやデータ定義ファイルなどが更新・変更されている場合，プログラム定義で取り込んでいる情報を最新にする必要があります。取り込んだ情報を最新状態にする場合は，[リソース] ― [再入力] を選んで変更します。

● テンプレートを変更したい場合

テンプレートを変更したい場合，[リソース] ― [テンプレートの変更] を選んで変更します。テンプレートを変更することで，現在の定義情報が削除される場合は，その旨を知らせるメッセージダイアログが表示されます。

● 辞書の情報を利用したい場合

辞書に格納されている情報を利用して処理をするかどうかを指定します。[リソース] メニューの [辞書の利用] をチェックすると，辞書を利用します。チェックしないと，辞書を利用しないで，既存の設定情報を利用して処理をします。

● 効率良くデータ定義ファイルを検索したい場合

- 「環境設定」で設定する
  プログラム定義の [データ定義指示項目設定] ダイアログ中に初期表示されるドキュメントフォルダは，「環境設定」で変更できます。
- 名前やプロパティ情報で判断する
  データ定義の定義内容の概要は，名前から判断できます。また，データ定義ファイルのプロパティ情報を参照すると，ファイル名，作成者および概要を見ることができます。

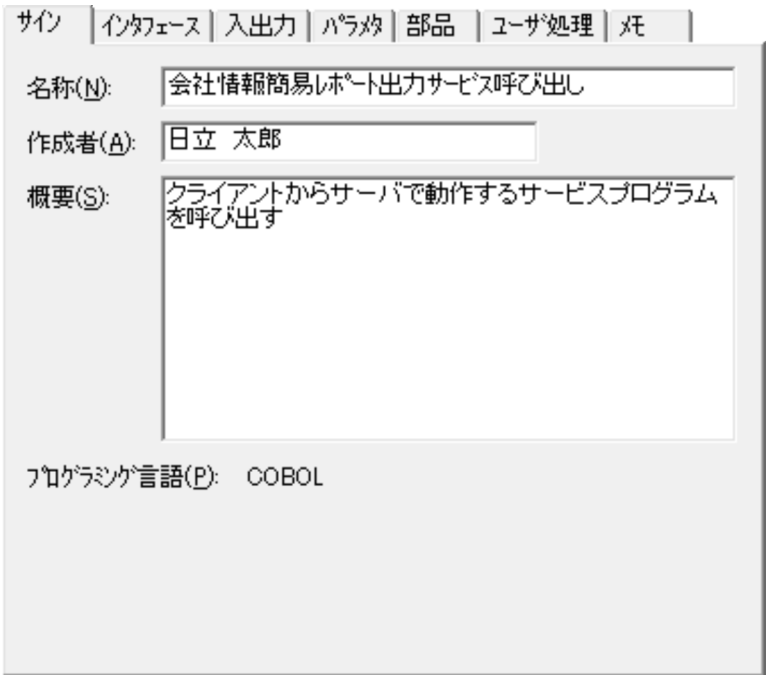
● ユーザ追加処理を編集するエディタを変更したい場合

ユーザ追加処理はメモ帳（notepad.exe）上で編集します。ユーザ追加処理を編集するエディタを変更したい場合は，「環境設定」で変更できます。

3.2　プログラムの定義

3.2.1　プログラムの概要情報の記述－[サインタブ]

[サイン] タブを次に示します。

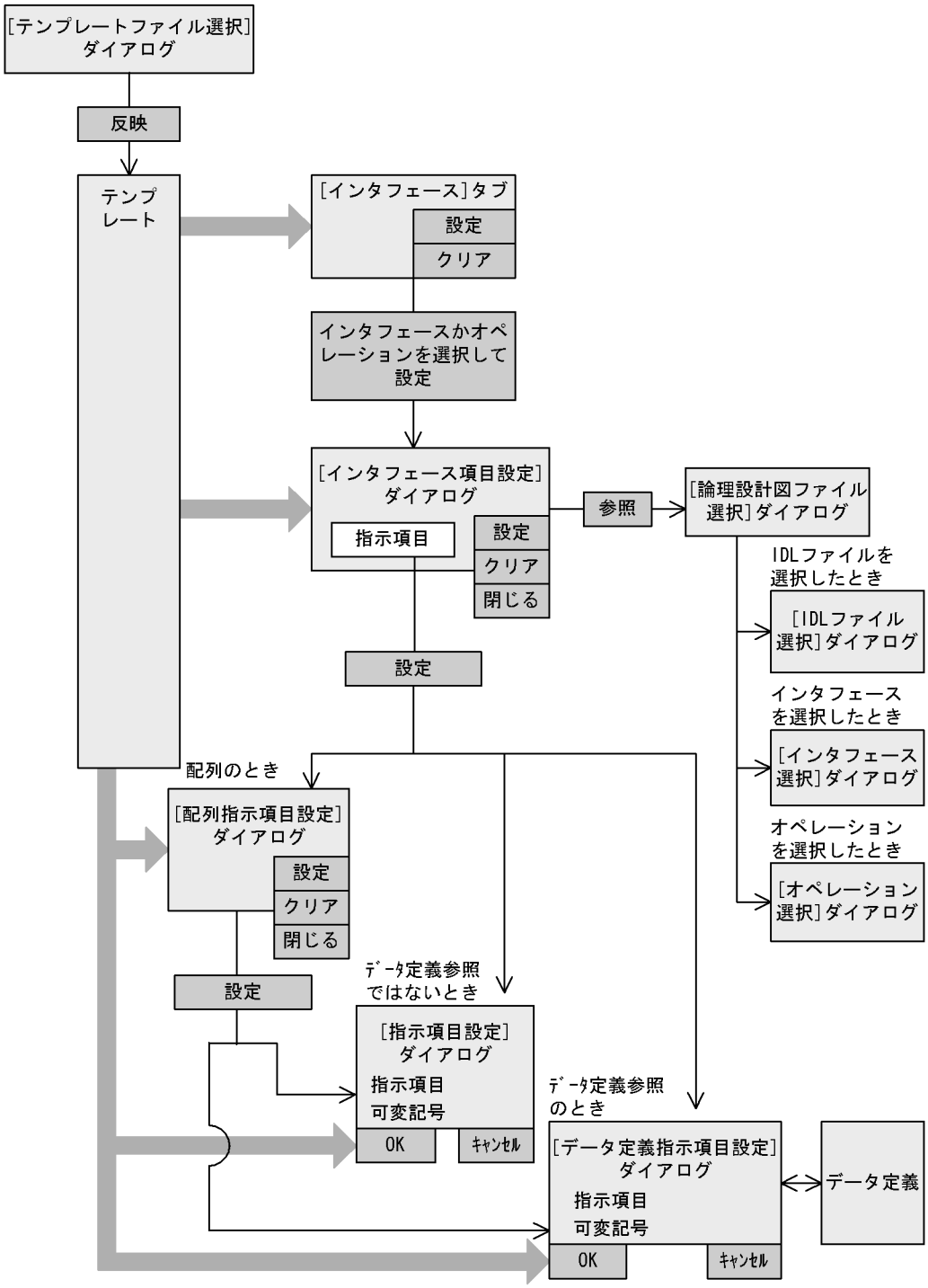


[サイン] タブには，作成する AP の名前，作成者名および概要を書きます。また，使用されているプログラミング言語が表示されます。

3.2.2　CORBA の実装プログラムで使用するインタフェース項目の定義－[インタフェース] タブ

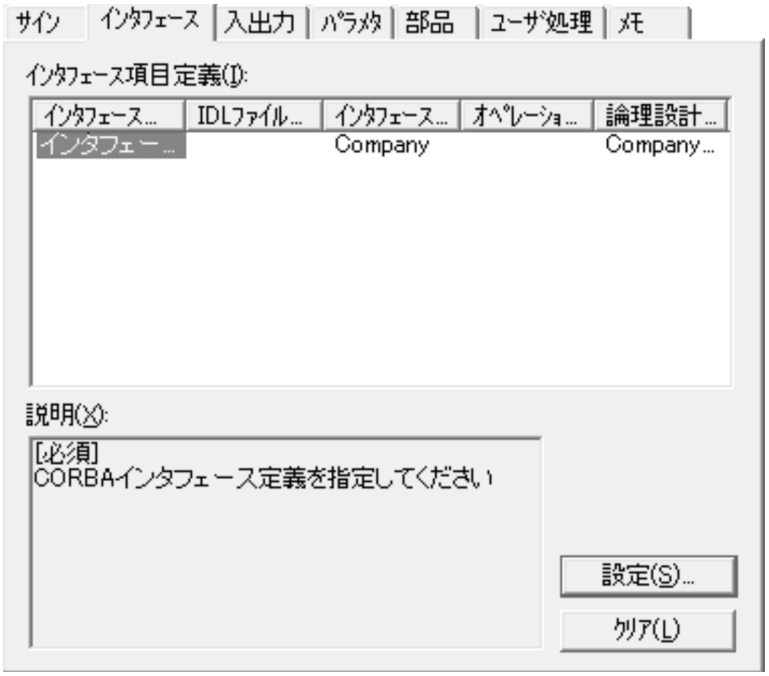
[インタフェース] タブでは，実装プログラムで使用するオブジェクトのインタフェース情報（IDL ファイル，インタフェース項目，オペレーション項目）を定義します。インタフェース項目やオペレーション項目を定義するときの [インタフェース] タブとダイアログの関係を図 3-1 に示します。

図 3-1 [インタフェース] タブとダイアログの関係



(1) [インタフェース] タブ

[インタフェース] タブを次に示します。



[インタフェース] タブには，CORBA のオブジェクトのインタフェースとして定義されたインタフェース項目に対応している可変記号が表示されます。使用するインタフェースを選びます。

(2) [インタフェース項目設定] ダイアログ

[インタフェース項目設定] ダイアログには，[インタフェース] タブで選ばれた IDL ファイル名，インタフェース項目名，およびインタフェース中のオペレーション項目名が表示されます。[インタフェース] 項目設定ダイアログを次に示します。

(3) [論理設計図ファイル選択] ダイアログ

[論理設計図ファイル選択] ダイアログには，論理設計図ファイルの一覧が表示されているので，その中から，実装プログラムで使用する IDL ファイル，インタフェース項目，またはオペレーション項目を含む論理設計図ファイルを選びます。

●[インタフェース] タブで IDL ファイルを選んだ場合

[IDL ファイルの選択] ダイアログが表示されます。実装プログラムで使用する IDL ファイルを選びます。

●[インタフェース] タブでインタフェース項目を選んだ場合

[インタフェース選択] ダイアログが表示されます。実装プログラムで使用するインタフェース項目を選びます。

●[インタフェース] タブでオペレーション項目を選んだ場合

[オペレーション選択] ダイアログが表示されます。実装プログラムで使用するオペ

レーション項目を選びます。

「環境設定」でパスが複数指定されている場合

「環境設定」で論理設計図のパスが複数指定されている場合，[論理設計図ファイル選択] ダイアログの「ファイルの場所」を選択する [パス選択] ダイアログが表示されます。[パス選択] ダイアログには，「環境設定」で指定したパスおよび前回 [論理設計図ファイル] 選択ダイアログで選択したパスが表示されます。[パス選択] ダイアログについては，「3.1.2(2) [パス選択] ダイアログ」を参照してください。なお，「テンプレート」は「論理設計図」に置き換えてお読みください。

(4) [指示項目設定] ダイアログ

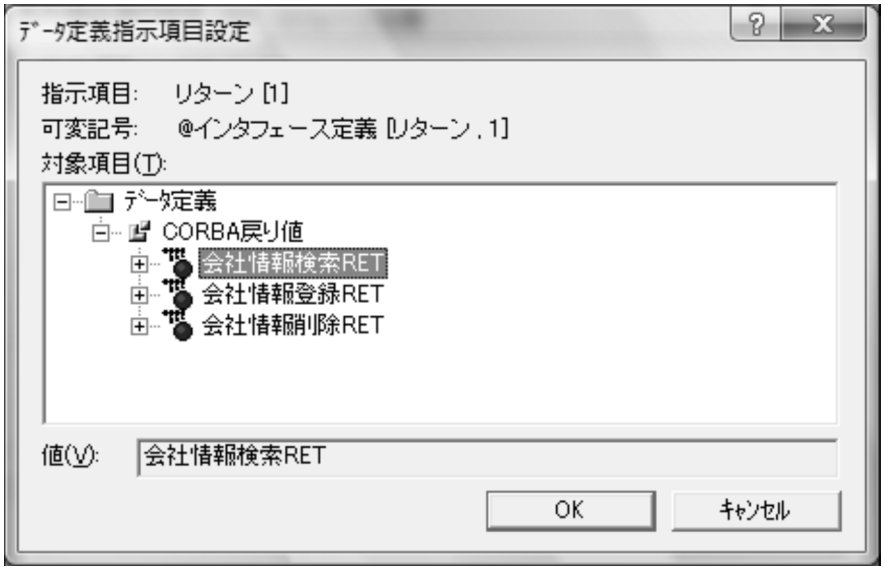
[指示項目設定] ダイアログを次に示します。

[論理設計図ファイル選択] ダイアログで選ばれた論理設計図ファイルを修飾する情報（接頭語，接尾語）を設定します。

なお，テンプレートで値が定義されていれば，値の欄にリスト形式で値が表示され，そこから選択できます。

(5) [データ定義指示項目設定] ダイアログ

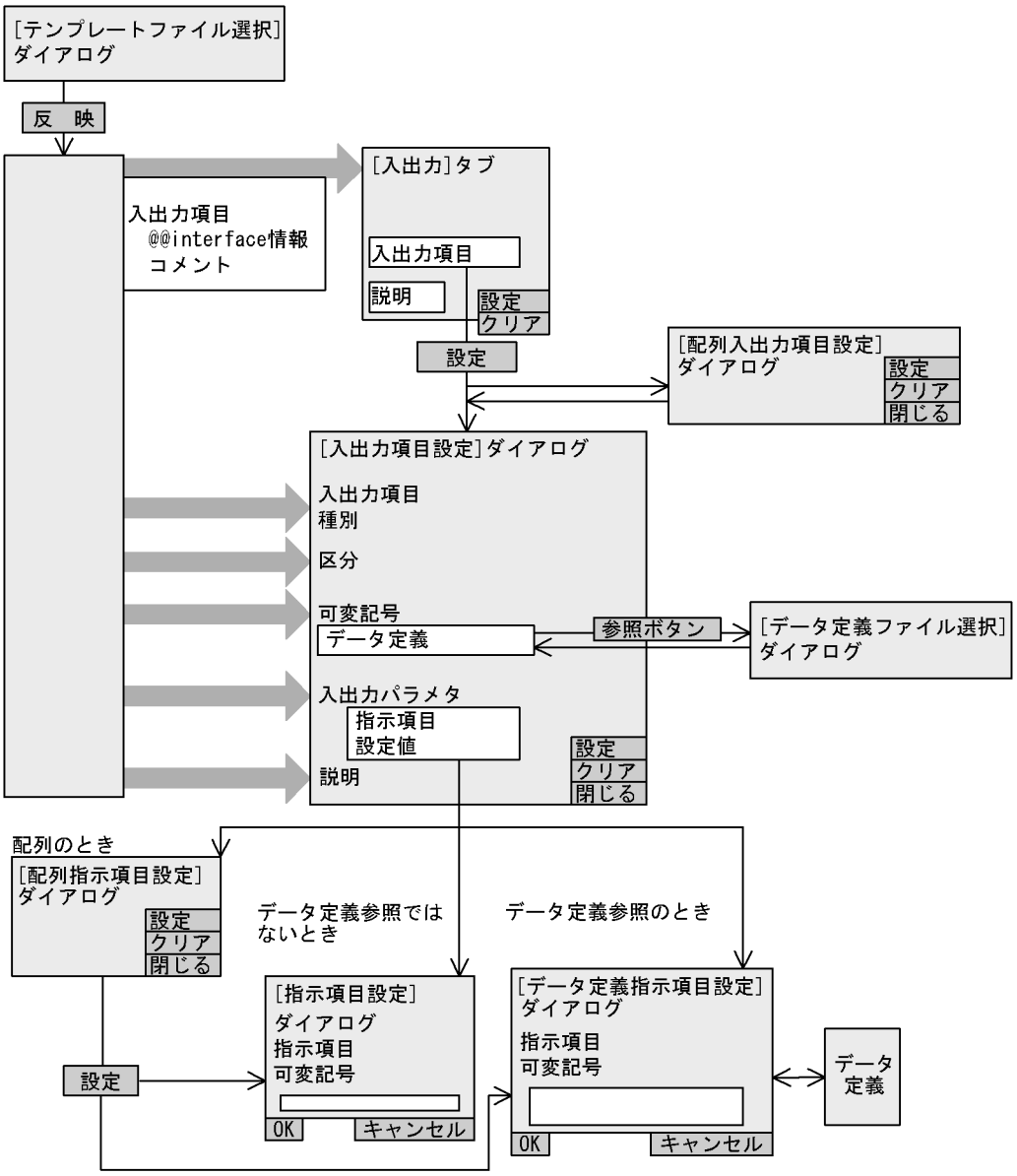
指示項目がデータ定義を参照するとテンプレートで宣言されているときは，[データ定義指示項目設定] ダイアログが表示されます。[データ定義指示項目設定] ダイアログを次に示します。



3.2.3 AP で使用する入出力項目の定義－ [入出力] タブ

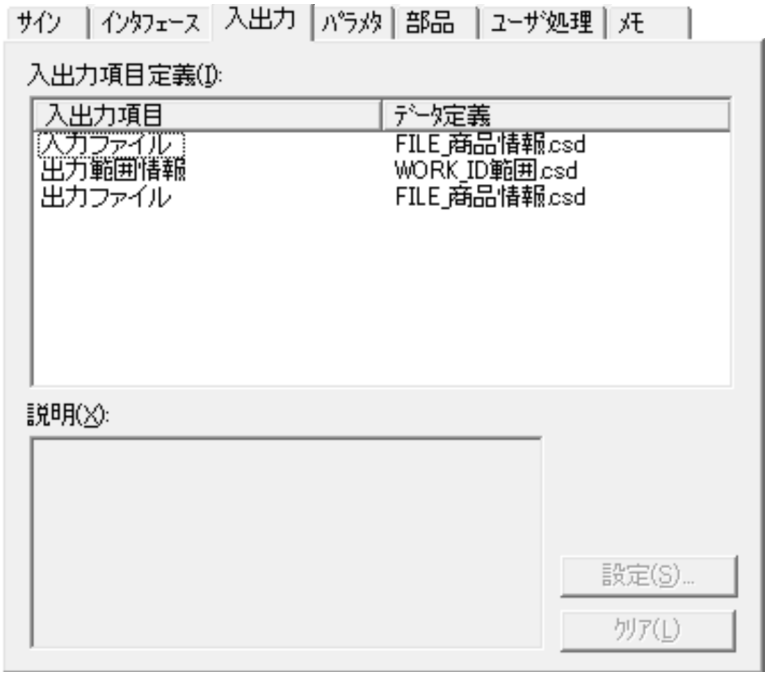
[入出力] タブでは AP で使用する入出力項目を定義します。入出力項目を定義するときの [入出力] タブとダイアログの関係を図 3-2 に示します。

図 3-2 [入出力] タブとダイアログの関係



(1) [入出力] タブ

[入出力] タブを次に示します。



[入出力] タブには，AP の入出力資源として使用されるデータ定義ファイルまたは XMAP3 に対応している可変記号が表示されるので，これらの可変記号に値を設定します。[入出力] タブでは，値を設定したい可変記号を選びます。

(2) [入出力項目設定] ダイアログ

[入出力項目設定] ダイアログを次に示します。

[入出力項目設定] ダイアログには，[入出力] タブで選ばれた入出力項目と，それに関する情報が表示されます。これらの情報を基に，AP 中で使用するデータ定義ファイルを選びます。また，選んだデータ定義ファイルを修飾する情報も指定できます。

● [データ定義ファイル選択] ダイアログ

[データ定義ファイル選択] ダイアログには，データ定義ファイルの一覧が表示されているので，その中から AP 中で使用するものを選びます。

(3) [指示項目設定] ダイアログ

[指示項目設定] ダイアログを次に示します。

[データ定義ファイル選択] ダイアログで選ばれたデータ定義ファイルを修飾する情報（外部装置名やアクセス種別など）を，設定します。[指示項目設定] ダイアログには，[入出力] タブで選ばれた指示項目が表示されます。この指示項目に対して，値を設定します。

なお，テンプレートで値が定義されていれば，値の欄にリスト形式で値が表示され，そこから選択できます。

「環境設定」でパスが複数指定されている場合

「環境設定」でデータ定義のパスが複数指定されている場合，[データ定義ファイル選択] ダイアログの「ファイルの場所」を選択する [パス選択] ダイアログが表示されます。[パス選択] ダイアログには，「環境設定」で指定したパスおよび前回 [データ定義ファイル選択] ダイアログで選択したパスが表示されます。[パス選択] ダイアログについては，「3.1.2(2) [パス選択] ダイアログ」を参照してください。なお，「テンプレート」は「データ定義」に置き換えてお読みください。

(4) [データ定義指示項目設定] ダイアログ

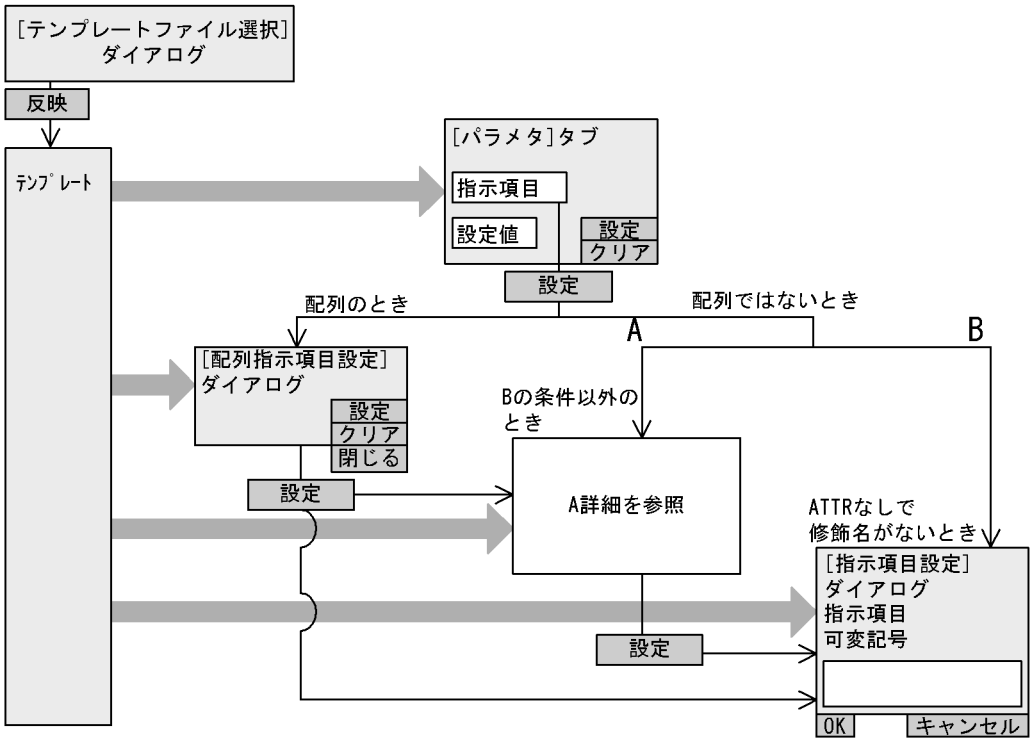
指示項目がデータ定義を参照するとテンプレートで宣言されているときは，[データ定義指示項目設定] ダイアログが表示されます。[データ定義指示項目設定] ダイアログを次に示します。

[データ定義指示項目設定] ダイアログでは，[入出力] タブで選ばれた指示項目が表示されます。指示項目に設定する値は，データ定義ファイルの項目から選びます。ただし，[入出力] タブで定義されていないデータ定義ファイルは選べません。

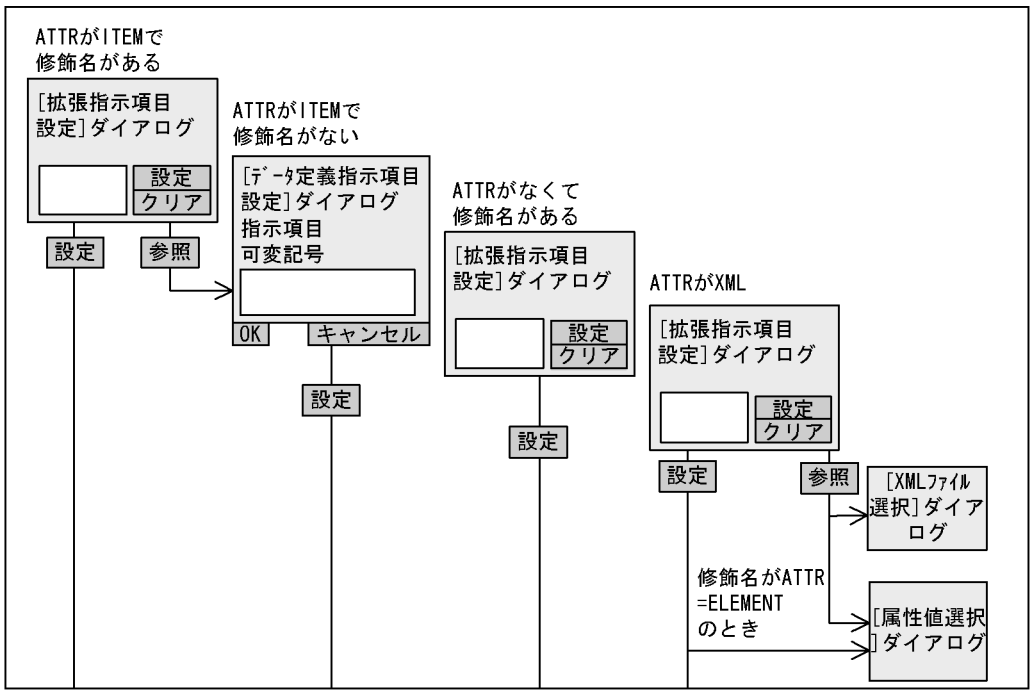
3.2.4 テンプレート中の指示項目の設定－ [パラメタ] タブ

[パラメタ] タブでは，テンプレート中の可変記号にプログラム生成時に使用するときの値を設定します。また，XML 文書を使用する場合も，[パラメタ] タブで設定します。値を設定するときの [パラメタ] タブとダイアログの関係を図 3-3 に示します。

図 3-3 [パラメタ] タブとダイアログの関係

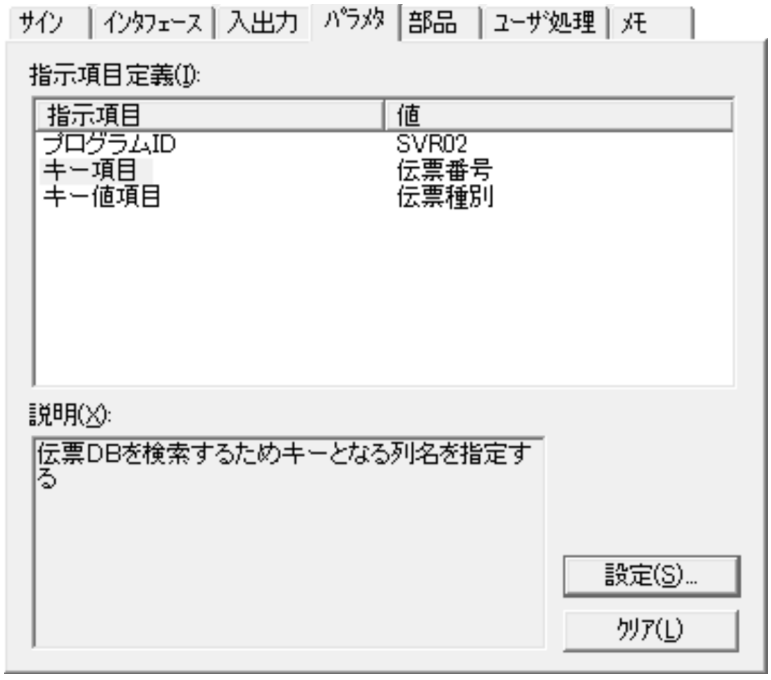


● A詳細



(1) [パラメタ] タブ

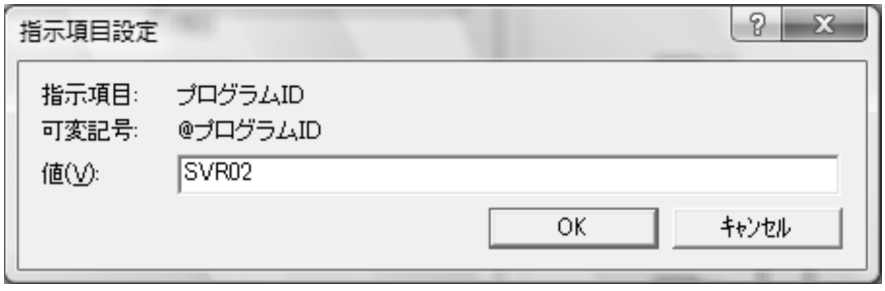
[パラメタ] タブを次に示します。



[パラメタ] タブには，AP 中でデータ項目として使用される可変記号が表示されるので，これらの可変記号に対して値を設定します。[パラメタ] タブでは，値を設定したい可変記号を選びます。

(2) [指示項目設定] ダイアログ

[指示項目設定] ダイアログを次に示します。



選ばれた指示項目に対する値を設定します。[指示項目設定] ダイアログには，[パラメタ] タブで選ばれた指示項目が表示されます。この指示項目に対して，値を設定します。

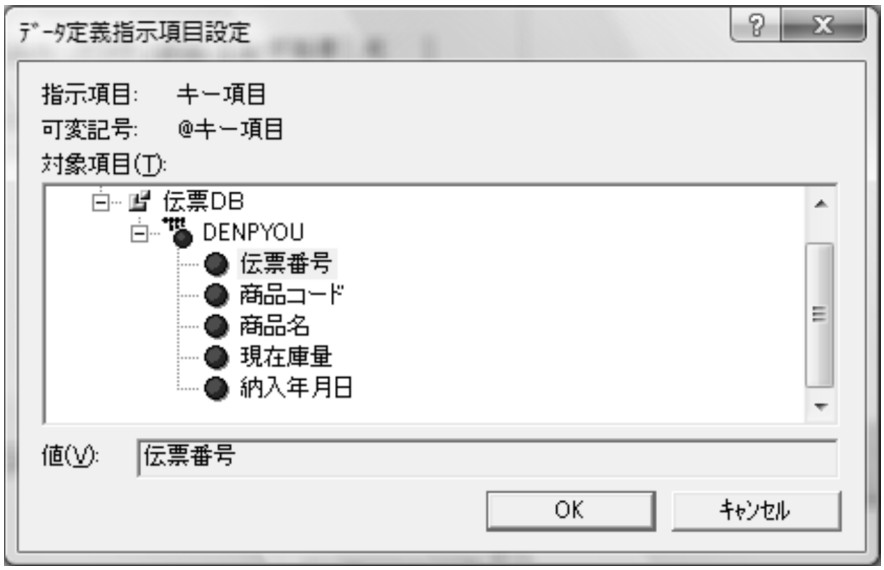
なお，テンプレートで値が定義されていれば，値の欄にリスト形式で値が表示され，そ

こから選択できます。

(3) [データ定義指示項目設定] ダイアログ

指示項目がデータ定義を参照するとテンプレートで宣言されているときは，[データ定義指示項目設定] ダイアログが表示されます。

[データ定義指示項目設定] ダイアログを次に示します。



[データ定義指示項目設定] ダイアログでは，[パラメタ] タブで選ばれた指示項目が表示されます。指示項目に設定する値は，データ定義ファイルの項目から選びます。ただし，[入出力] タブで定義されていないデータ定義ファイルは選べません。

「環境設定」でパスが複数指定されている場合

「環境設定」でデータ定義のパスが複数指定されている場合，[データ定義ファイル選択] ダイアログの「ファイルの場所」を選択する [パス選択] ダイアログが表示されます。[パス選択] ダイアログには，「環境設定」で指定したパスおよび前回 [データ定義ファイル選択] ダイアログで選択したパスが表示されます。[パス選択] ダイアログについては，「3.1.2(2) [パス選択] ダイアログ」を参照してください。なお，「テンプレート」は「データ定義」に置き換えてお読みください。

(4) 指示項目（拡張指示項目）設定ダイアログ

テンプレートで，ATTR に設定がある場合，または修飾名の ATTR に設定がある場合，次に示す [指示項目設定] ダイアログが表示されます。このとき表示される [指示項目設定] ダイアログを，[拡張指示項目設定] ダイアログと呼びます。

@@interface 文で属性を設定した場合，[参照] ボタンを押すと，属性によって次のダイアログが表示されます。

● **[データ定義指示項目] ダイアログ**

@@interface 文で ATTR=ITEM を指定している場合に表示されます。ここで，データ定義で定義された項目から対象となる項目を選択し，値を設定します。

● **[XML ファイル選択] ダイアログ**

@@interface 文で ATTR=XML を指定している場合に表示されます。ここで，使用する XML ファイルを選択します。

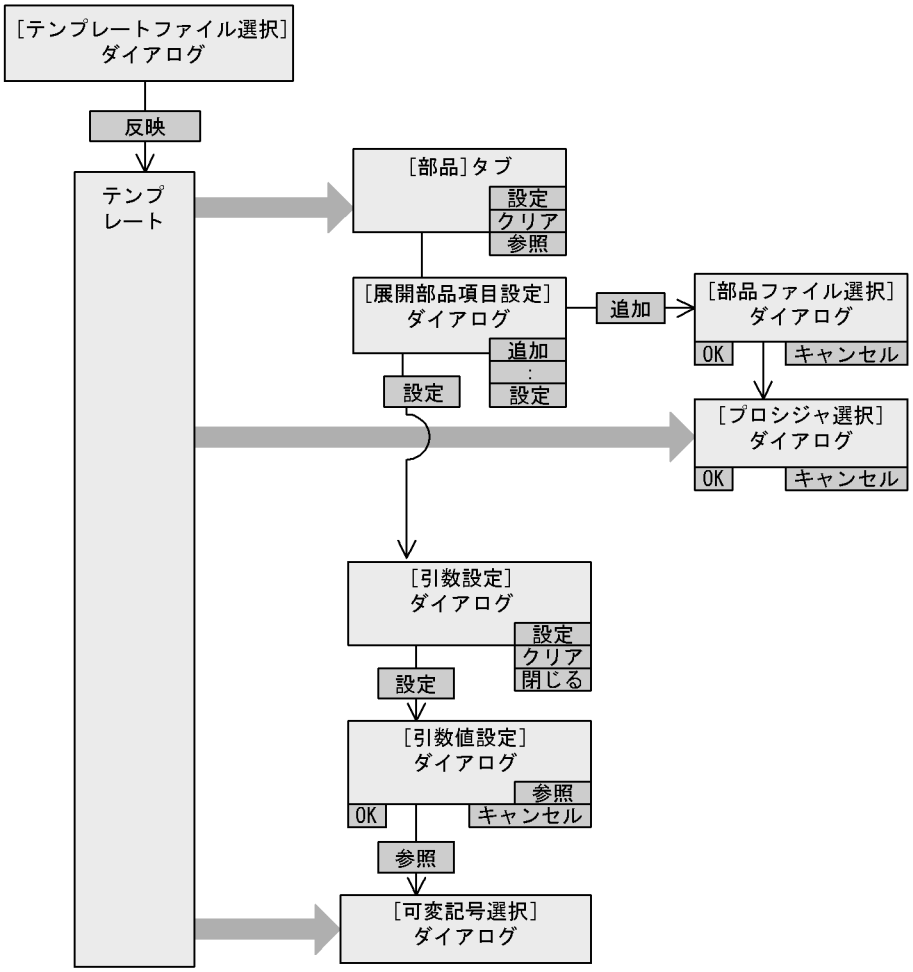
● **[属性値選択] ダイアログ**

@@interface 文で，修飾名に ATTR=ELEMENT を指定している場合に表示されます。ここで，各指示項目に対する属性値（属性ノード）を選択します。属性値（属性ノード）については，「7.8.2　XML 文書の構造」を参照してください。

3.2.5 AP で使用する部品の定義－ [部品] タブ

[部品] タブでは，AP で使用する部品項目を定義します。部品項目を定義するときの [部品] タブとダイアログの関係を図 3-4 に示します。

図 3-4 [部品] タブとダイアログの関係



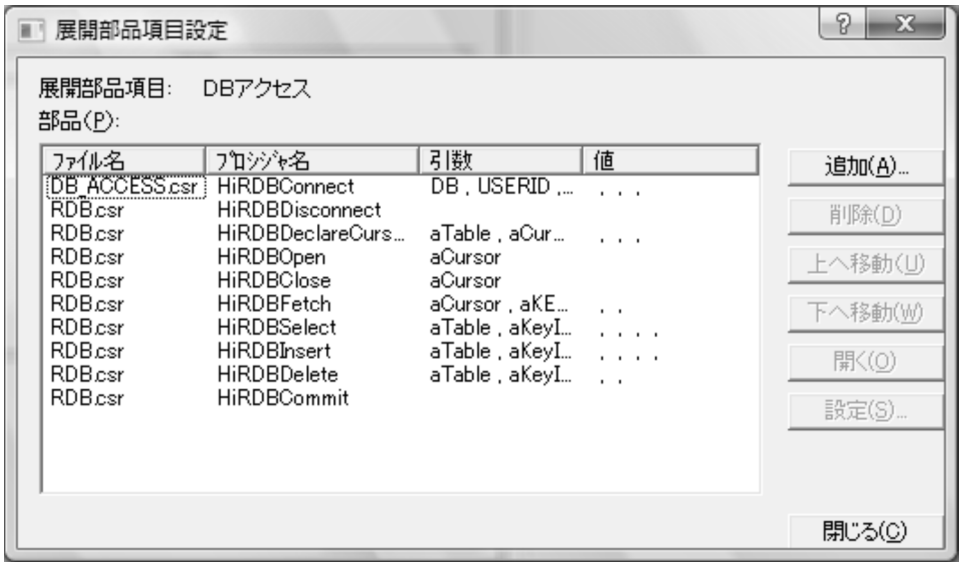
(1) [部品] タブ

[部品] タブを次に示します。

[部品] タブには，テンプレートで，AP の部品として使用されると宣言された部品項目に対応している可変記号が表示されます。AP で使用する部品を選びます。

(2) [展開部品項目設定] ダイアログ

[展開部品項目設定] ダイアログを次に示します。



[展開部品項目設定] ダイアログには，展開される部品が表示されます。

(3) [部品ファイル選択] ダイアログ

[部品ファイル選択] ダイアログを次に示します。

[部品ファイル選択] ダイアログには，部品ファイルの一覧が表示されます。この中から，すでに表示されている部品に追加する部品を選んだり，変更したりします。

「環境設定」でパスが複数指定されている場合

「環境設定」で部品のパスが複数指定されている場合，[部品ファイル選択] ダイアログの「ファイルの場所」を選択する [パス選択] ダイアログが表示されます。[パス選択] ダイアログには，「環境設定」で指定したパスおよび前回 [部品ファイル選択] ダイアログで選択したパスが表示されます。[パス選択] ダイアログについては，「3.1.2(2) [パス選択] ダイアログ」を参照してください。なお，「テンプレート」は「部品」に置き換えてお読みください。

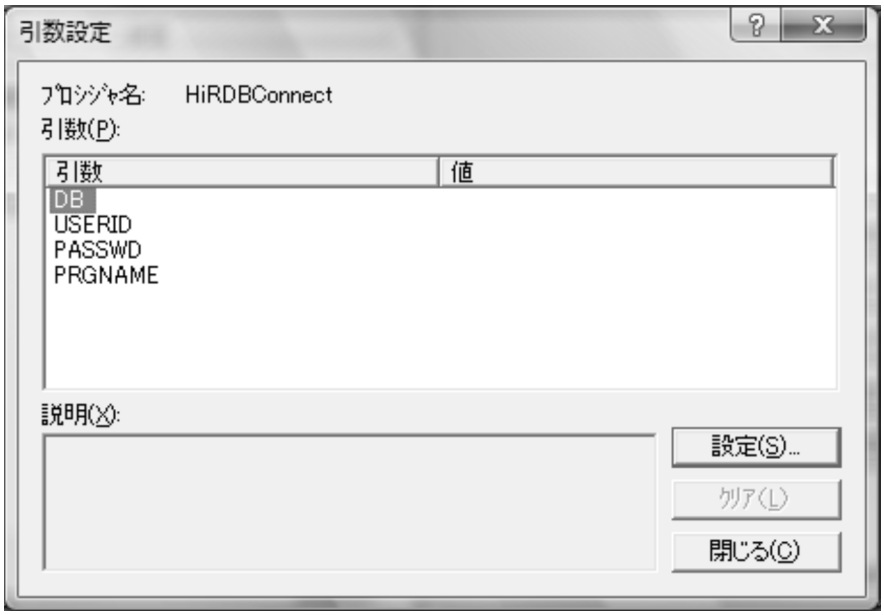
(4) [プロシジャ選択] ダイアログ

[プロシジャ選択] ダイアログを次に示します。

[部品ファイル選択] ダイアログで，プロシジャが記述されている部品を選択すると，[プロシジャ選択] ダイアログが表示されます。[プロシジャ選択] ダイアログでは，APで使用する部品に追加したいプロシジャを選びます。

(5) [引数設定] ダイアログ

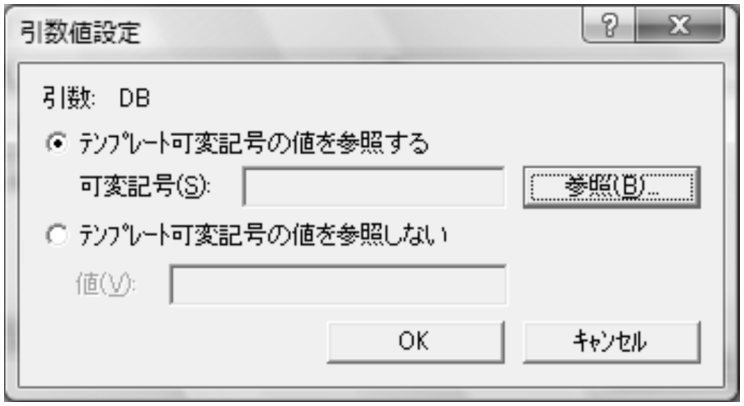
[引数設定] ダイアログを次に示します。



[引数設定] ダイアログには，[展開部品項目設定] ダイアログで選んだ部品のプロシジャの引数一覧が表示されます。引数に値を設定したい場合は，[引数値設定] ダイアログを表示します。

(6) [引数値設定] ダイアログ

[引数値設定] ダイアログを次に示します。



選択された引数の値を設定します。テンプレートに定義された可変記号の値を参照するときは，[可変記号選択] ダイアログを表示します。

(7) [可変記号選択] ダイアログ

[可変記号選択] ダイアログを次に示します。

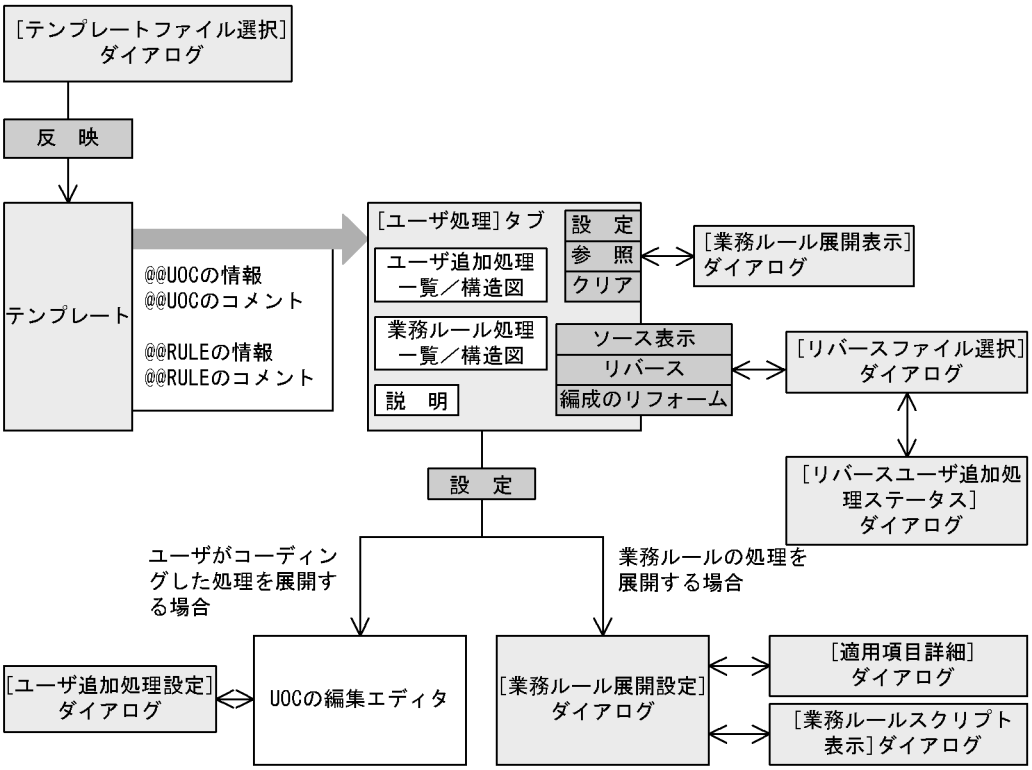
[可変記号選択] ダイアログには，可変記号の一覧が表示されています。プロシジャの引数として使用したい可変記号を選びます。

3.2.6 ユーザ処理の設定－ [ユーザ処理] タブ

[ユーザ処理] タブでは，ソースプログラム中で要求されているユーザ追加処理（UOC）の編集およびソースプログラム中で展開させる業務ルール処理の設定をします。また，ソースプログラム上で編集したユーザ追加処理をプログラム定義ファイルに取り込みます。

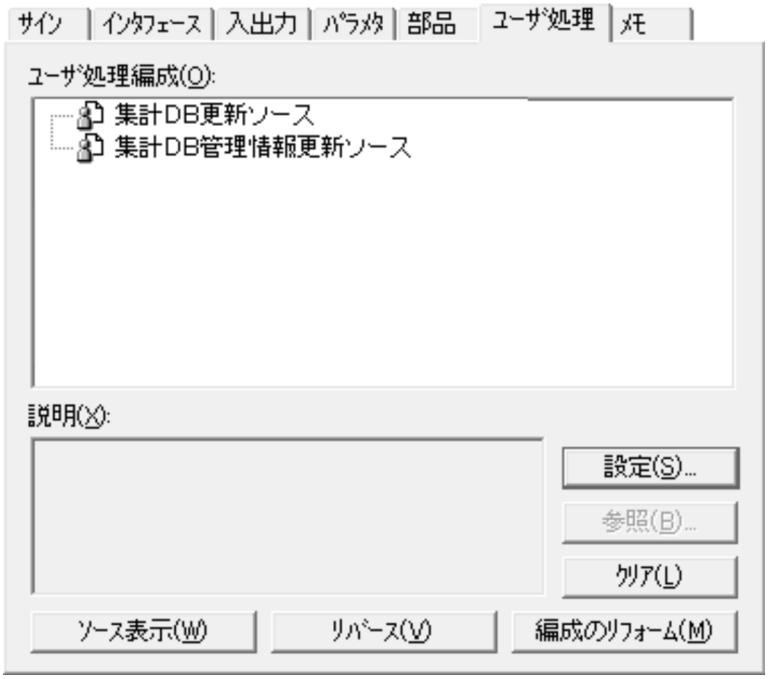
ユーザがソースプログラム中で要求されている処理をコーディングしたり（ユーザ追加処理の編集），ソースプログラム上で編集したユーザ追加処理をプログラム定義ファイルに取り込んだり（UOC リバース機能），自分でコーディングする代わりに，処理の書かれた業務ルールを設定したりする（業務ルール処理の設定）ときの [ユーザ処理] タブとダイアログの関係を図 3-5 に示します。

図 3-5 [ユーザ処理] タブとダイアログの関係



(1) [ユーザ処理] タブ

[ユーザ処理] タブを次に示します。



[ユーザ処理] タブには，ユーザ追加処理群および業務ルール処理群が表示されています。

(2) 要求されているユーザ処理を確認する

[ユーザ処理] タブには，ユーザ追加処理群および業務ルール処理群が表示されています。ソースプログラム中で要求されている処理をユーザがコーディングして展開させる場合は，表示されているユーザ追加処理群から，目的のユーザ追加処理を選んで作業を開始します。ソースプログラム中で要求されている処理を業務ルールを設定して展開させる場合は，表示されている業務ルール処理群から，目的の業務ルール処理を選んで作業を開始します。

● ユーザ処理の表示

ループなどが多い構造化されたテンプレートに含まれるユーザ追加処理や業務ルール処理は，階層化された一覧表で表示した方が理解しやすくなります。階層化された一覧表を表示させるには，テンプレートに @@diagram 文を書きます。@@diagram 文が書かれていない場合は，階層化されていない一覧表で表示されます。

注

ユーザ追加処理や業務ルール処理の内容，数および挿入位置は，[入出力] タブや [パラメタ] タブの内容で変わります。[入出力] タブまたは [パラメタ] タブを変

更した場合は，[編成のリフォーム] ボタンで最新の情報を反映した状態にしておきます。
このとき，[編成のリフォーム]」ボタンで反映されなかったユーザ追加処理や業務ルール処理を削除するかどうかを尋ねるダイアログが表示されます。反映されなかったユーザ追加処理や業務ルール処理を削除しない場合は [キャンセル] を選んでください。[編成のリフォーム] ボタンを押す前の状態に戻ります。

（3） ユーザ追加処理の編集と業務ルール処理の設定

[ユーザ処理] タブに表示されているユーザ追加処理や業務ルール処理から必要なものを選び，それぞれ編集したり設定したりします。また，ソースプログラム上で編集したユーザ追加処理をプログラム定義ファイルに取り込みます（UOC リバース機能）。

(a) ユーザ追加処理の編集

ソースプログラム上で要求されている処理を，ユーザが独自にコーディングし展開させる場合は，ユーザ追加処理を選びます。ユーザ追加処理を選んで [設定] ボタンを押すと，ユーザ追加処理を編集するためのエディタが表示されます。エディタ上で処理を編集してください。このとき，固定形式またはホスト向け固定形式でソースプログラムを生成する場合は，COBOL 言語の固定形式またはホスト向け固定形式の規則に従ってユーザ追加処理を編集してください。

●［ユーザ追加処理設定］ダイアログ

ユーザ追加処理を編集しているときに，［ユーザ追加処理設定］ダイアログが表示されます。このダイアログでは，ユーザ処理編成で選んだユーザ追加処理またはファイルを，エディタを起動させて表示させたり，編集したユーザ追加処理のソースプログラムをプログラム定義に取り込んだりできます。また，プレビューには，ユーザ処理編成で選んだユーザ追加処理の内容が表示されます。

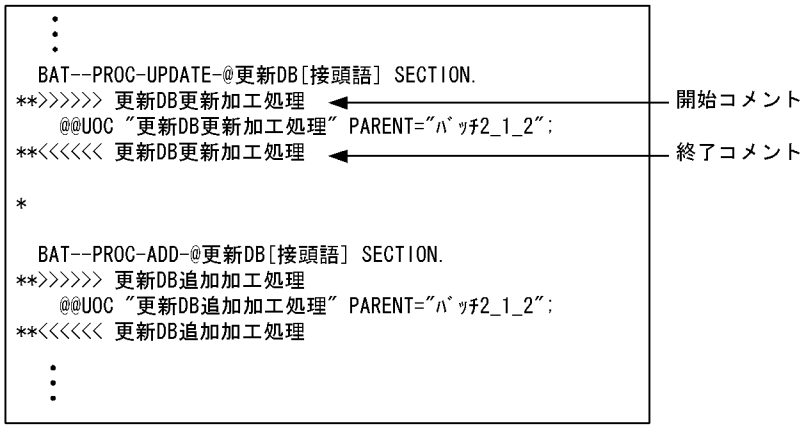
(b) ユーザ追加処理のプログラム定義ファイルへの取り込み

ソースファイル上で直接編集したユーザ追加処理を，プログラム定義ファイルのユーザ追加処理の中に取り込む（リバースする）ことができます。これを UOC リバース機能といいます。

UOC リバース機能を使うためには，ユーザ追加処理の開始と終了が判定できる UOC コメントが出力されている必要があります。UOC コメントを出力するには，次の二つの方法があります。

- テンプレートファイルに @@lang 文で指定する方法
  @@lang 文で UOC コメントの生成を指定する方法については，「7.9.32　@@lang 文」を参照してください。
- テンプレートファイルの @@uoc 文の前後に直接コメントを書き込む方法

例を示します。

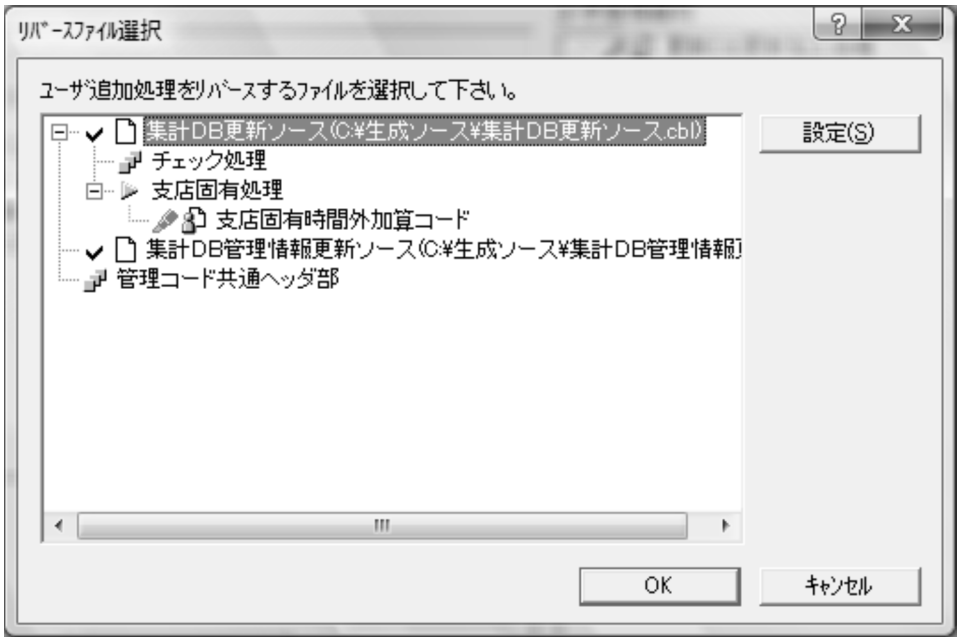


さらに，[SEWB+/CONSTRUCTION 環境設定] ダイアログの [ユーザ追加処理] タブで，コメントの開始と終了を示す文字列を設定しておきます。

UOC リバース機能を使う場合，UOC コメントをユーザ追加処理ごとにユニークにすることをお勧めします。なお，ユーザ追加処理の開始／終了文字列が @@lang 文と SEWB+/CONSTRUCTION 環境設定の両方で指定された場合には，@@lang 文で指定された文字列が優先されます。

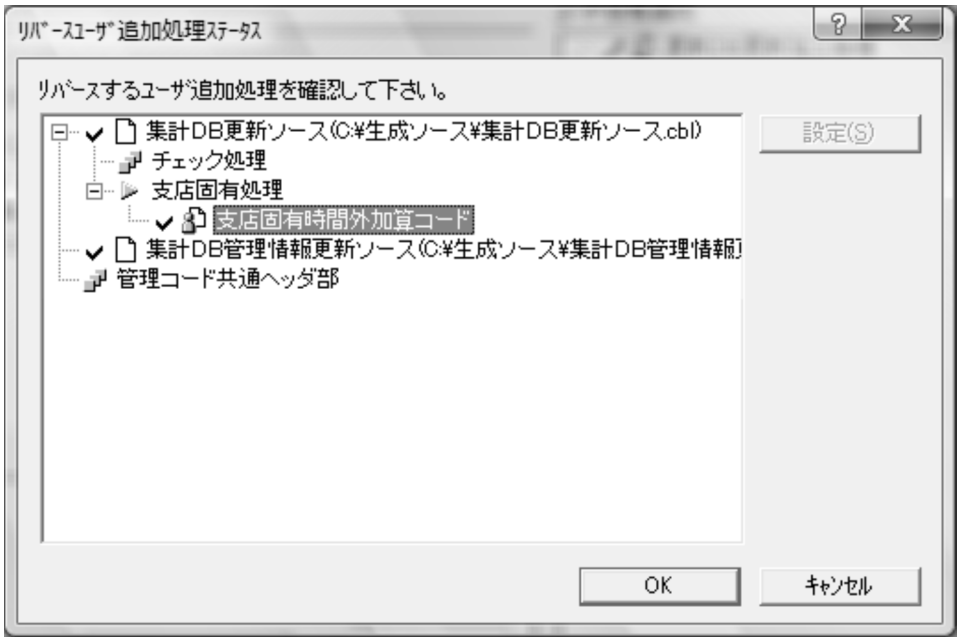
●[リバースファイル選択] ダイアログ

[ユーザ処理] タブの [リバース] ボタンを押すと，[リバースファイル選択] ダイアログが表示されます。ユーザ追加処理をリバースするファイルを選択します。デフォルトでは，プログラム定義が記憶している生成関連のローカルファイルがフルパスで表示されます。変更する場合は，[設定] ボタンを押して，[リバースファイル設定] ダイアログでファイルを選択します。



●[リバースユーザ追加処理] ステータスダイアログ

[リバースファイル選択] ダイアログでファイルを選んで [OK] ボタンを押すと，[リバースユーザ追加処理ステータス] ダイアログが表示されます。このダイアログには，ユーザ追加処理が一覧で表示され，変更されているかどうかなどのステータスが示されます。個々のユーザ追加処理について，プログラム定義ファイルに変更を取り込むかどうかを指定できます。[実行] ボタンを押すと，ソースファイルのユーザ追加処理の変更がプログラム定義ファイルのユーザ追加処理に取り込まれます。

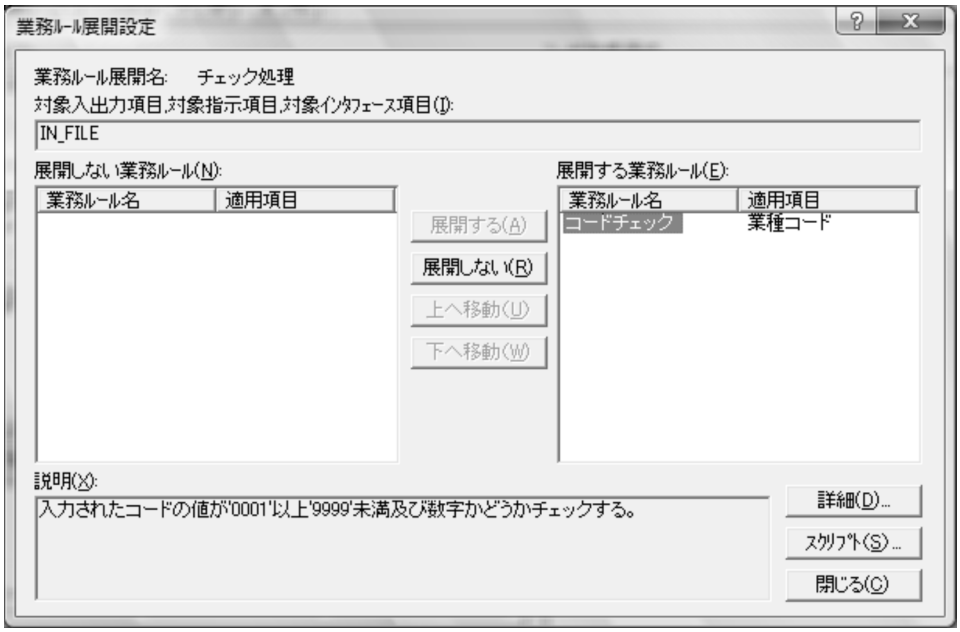


(c) 業務ルール処理の設定

ソースプログラム上で要求されている処理を，業務ルールを使用して展開させる場合は，業務ルール処理を選びます。業務ルール処理を選んで [設定] ボタンを押すと，実際にソースプログラム上に展開させる業務ルールを選ぶためのダイアログが表示されます。また，選ばれた業務ルールを確認する場合は，業務ルール処理を選んで [参照] ボタンを押し，[業務ルール展開表示] ダイアログを表示させます。

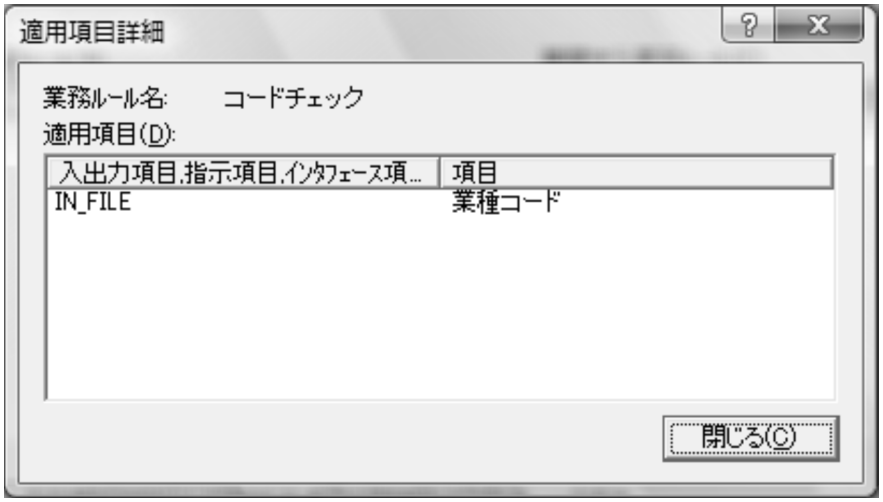
●[業務ルール展開設定] ダイアログ

ダイアログ中には，AP 中で展開される業務ルールと展開されない業務ルールの一覧が表示されます。展開する業務ルールと展開しない業務ルールは [展開する] ボタンおよび [展開しない] ボタンで編集できます。また，展開する業務ルールが複数表示されている場合，上に表示されている業務ルールから先に展開されます。展開順は [上へ移動] ボタンおよび [下へ移動] ボタンで編集できます。なお，それぞれの業務ルールの概要は「説明」に表示されているので参考にしてください。「説明」には，SEWB+/REPOSITORY の業務ルール辞書に書かれたコメントが表示されます。[業務ルール展開設定] ダイアログを次に示します。



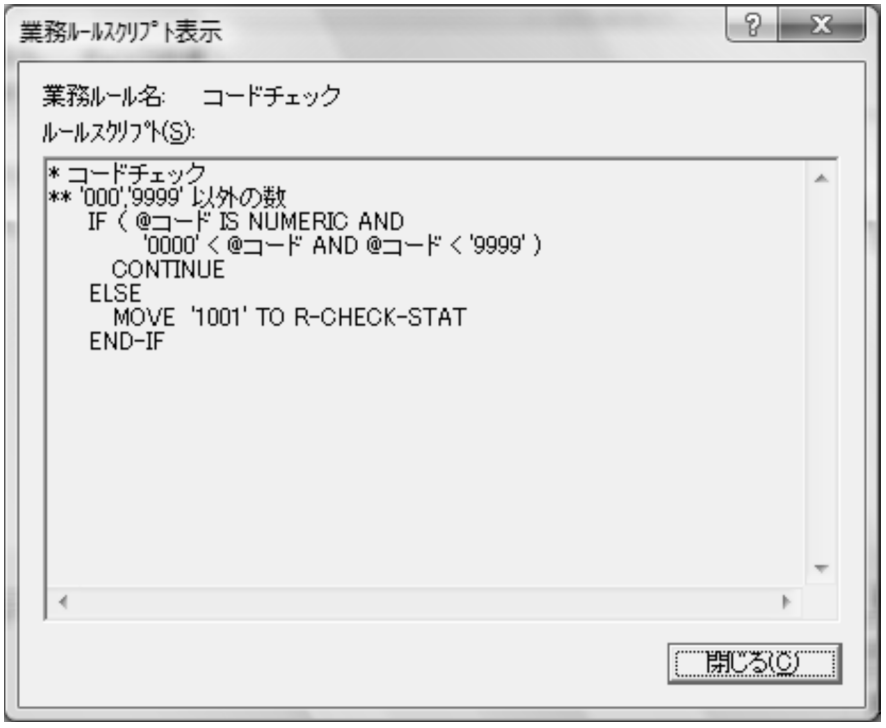
なお，業務ルールの詳細，およびスクリプトは，[適用項目詳細] ダイアログまたは [業務ルールスクリプト表示] ダイアログで見られます。

- [適用項目詳細] ダイアログ



選んだ業務ルールの適用項目の一覧を表示します。

- [業務ルールスクリプト表示] ダイアログ



選んだ業務ルールの処理内容を表示します。

注

[業務ルール展開参照] ダイアログは，[業務ルール展開設定] ダイアログと同じ形態です。ただし，[展開する] ボタン，[展開しない] ボタン，[上へ移動] ボタ

ンおよび [下へ移動] ボタンは押せません。

(4) コマンドによるプログラム定義ファイルへの取り込み

プログラム定義ファイルから生成されたソースファイルのユーザ追加処理をプログラム定義ファイルに反映させる場合，コマンド「CSPDREV.EXE」を実行させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

形 式

```
CSPDREV.EXE△リバースファイル名
            [△/sp△ソースファイル検索パス名]
            [△/l△ログファイル名]
            [△/help]
            [△/?]
```

（凡例）

△：1 文字以上の空白を示します。

解 説

リバースファイル名：

ユーザ追加処理をリバースするプログラム定義ファイルのファイル名を指定します。ファイル名にはワイルドカードが指定できます。

/sp △ソースファイル検索パス名：

ソースファイルを検索するパスの名称を指定します。“ 検索パス ”＋” プログラム定義ファイルの生成ソースファイルパスから切り出したファイル名 ” で，ソースファイルを検索します。

/sp の省略時は，プログラム定義ファイル生成ソースファイルパスで検索します。

/l △ログファイル名：

ログ情報の出力先ファイル名を指定します。

/help：

コマンドヘルプメッセージを表示します。

/? ：

コマンドヘルプメッセージを表示します。

3.3 ソースプログラムの生成

プログラム定義が終わったら，ソースプログラムを生成します。ソースプログラムは，プログラム定義のメインウィンドウから生成できます。

3.3.1 プログラム定義ウィンドウからの生成

[生成] ボタンを押すと，ソースプログラムの生成を開始します。

@@FILE が有効で生成するファイル名が確定している場合には，[生成先パス設定] ダイアログが表示されます。生成するソースプログラムの出力先パス名を指定してください。生成先のフォルダがない場合，フォルダを作成してファイルを生成します。

@@FILE がない場合，または @@FILE が有効でない場合には，[生成ファイルの保存] ダイアログが表示されます。生成するソースプログラムの出力先のファイル名を指定してください。

3.3.2 コマンドでの生成（ソースプログラム）

ソースプログラムを生成する場合は，コマンド「CSPDGEN.EXE」を実行させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

形　式

```
CSPDGEN.EXE△/i△入力ファイル名
          △[/op△出力先パス名]
          △[/o△出力ファイル名]
          △[/t△トレースファイル名]
          △[/u]
          △[/r]
          △[/tp△テンプレートファイルパス名]
          △[/dp△データ定義ファイルパス名]
          △[/lp△論理設計図ファイルパス名]
          △[/pp△部品テンプレートファイルパス名]
          △[/mp△XMLファイルパス名]
          △[/xp△マップ定義ファイルパス名]
          △[/rp△レコード定義ファイルパス名]
          △[/d△インデンテーション長]
          △[/pi△PICTURE句などの生成位置]
```

```
△[/cf△COBOLソース正書法]
△[{/f|/df}]
△[{/re|/dre}]
△[{/ud|/dud}]
△[{/us|/dus}]
△[{/s|/ds}]
△[/si△一連番号初期値]
△[/sa△一連番号増分値]
```

（凡例）

△：1 文字以上の空白を示します。

注

［ ］は省略できることを示します。

{ } はどちらかを指定することを示します。

解 説

/i △入力ファイル名：

プログラム定義ファイル（.csp または .csq）のパス名またはファイル名を指定します。

/op △出力先パス名※1：

生成するソースプログラムの出力先のパス名を指定します（@@FILE 文が有効で，生成するファイル名が確定しているとき使用されます）。

/o △出力ファイル名※1：

生成するソースプログラムの出力先のファイル名を指定します（@@FILE 文がないか，または @@FILE が有効でないとき使用されます）。

/t △トレースファイル名：

トレース情報※2 の出力先ファイル名を指定します。

/u：

テンプレートファイル，データ定義ファイル，レコード定義ファイル，論理設計図ファイルおよび辞書の更新によって，interface 文の可変記号，ユーザ追加処理，および業務ルールの設定が削除される場合に，生成を中止します。

/r※3：

プログラム定義ファイルを最新状態にする場合に指定します。

/tp △テンプレートファイルパス名※4：

テンプレートファイル（.cst）の検索パス名を指定します。

/dp △データ定義ファイルパス名※4：

データ定義ファイル（.csd または .cse）の検索パス名を指定します。

/lp △論理設計図ファイルパス名※4：

論理設計図ファイル（.dal）の検索パス名を指定します。

/pp △部品テンプレートファイルパス名※4：

部品テンプレートファイル（.csr）の検索パス名を指定します。

/mp △ XML ファイルパス名※4：

XML ファイルの検索パス名を指定します。

/xp △マップ定義ファイルパス名※4：

マップ定義ファイルの検索パス名を指定します。

/rp △レコード定義ファイルパス名※4：

レコード定義ファイルの検索パス名を指定します。

/d △インデンテーション長※4：

インデンテーション長を 0 ～ 8 の範囲で指定します。

/pi △ PICTURE 句などの生成位置※4：

PICTURE 句などの生成位置の範囲を指定します。

- COBOL ソースの形式が固定形式またはホスト向け固定形式の場合は 8 ～ 60。
- COBOL ソースの形式がフリー形式の場合は 1 ～ 60。

/cf △ COBOL ソース正書法※4：

COBOL ソースの形式を「fix/free/hostfix」で指定します。「fix」は COBOL ソースを固定形式またはホスト向け固定形式で生成します。「free」は COBOL ソースをフリー形式で生成します。「hostfix」は COBOL ソースをホスト向け固定形式で生成します。

/f※4：

プログラミング言語が COBOL であり，ソースプログラムを自動清書※5 させない場合に指定します。

/df※4：

プログラミング言語が COBOL であり，ソースプログラムを自動清書※5 させる場合に指定します。

/re※4：

@@rule 文によって抽出されたすべての業務ルールを，無条件に展開させて生成させる場合に指定します。

/dre※4：

@@rule 文によって抽出されたすべての業務ルールのうち，ユーザが選んだ業務ルールだけを展開させて，生成させる場合に指定します。

/ud※6：

データ項目辞書および業務ルール辞書を使用する場合に指定します。

/dud※6：

データ項目辞書および業務ルール辞書を使用しない場合に指定します。

/us※4：

プログラミング言語が COBOL であり，ユーザ追加処理をそのまま生成※7 するときに指定します。

/dus※4：

プログラミング言語が COBOL であり，ユーザ追加処理を展開文と同じように扱って生成するときに指定します。

/s※4：

固定形式またはホスト向け固定形式の COBOL ソースプログラムに，一連番号を付加するときに指定します。

/ds※4：

固定形式またはホスト向け固定形式の COBOL ソースプログラムに，一連番号を付加しないときに指定します。

/si △一連番号初期値※4：

固定形式またはホスト向け固定形式の COBOL ソースプログラムに，一連番号を付加するとき，一連番号の初期値を 1 ～ 999999 の範囲で指定します。

/sa △一連番号増分値※4：

固定形式またはホスト向け固定形式の COBOL ソースプログラムに，一連番号を付加するとき，一連番号の増分値を 1 ～ 999999 の範囲で指定します。

注※ 1

出力先は，生成結果により /op または /o のどちらかが使用されます。/op の指定がない場合には，カレントパスが出力先パスになります。/o の指定がない場合には，カレントパスが出力先パスになり，プログラム定義ファイル名の拡張子をプログラミング言語の拡張子に置き換えた名称がファイル名になります。

注※ 2

トレース情報には，関数トレースなどの情報が出力されています。

注※ 3

ファイルの生成が成功した場合に有効となります。

注※ 4

何も指定しない場合（「/f，/df」「/re，/dre」「/us，/dus」「/s，/ds」については，どちらも指定しない場合）は，環境設定で定義されている情報が有効になります。

注※ 5

自動清書とは，ソースプログラムのインデンテーションなどを自動的に整え，見やすい形式で生成する機能です。

注※ 6

何も指定しない場合は，/dud が指定されているとみなされます。

注※ 7

COBOL 正書法では，B 領域を超える部分はコメントとして扱われます。しかし，展開文に可変記号を使用する場合，生成したソースが B 領域を超えないようなテンプレートや部品を作成するのは困難です。このため，SEWB+/CONSTRUCTION では，生成ソースが B 領域を超える場合，複数行に分割して COBOL ソースを生成しています。
/us は，ユーザ追加処理をほかのテンプレートや部品から生成されるソースだけ B 領域を超えないよう分割し，ユーザ追加処理は指定されたまま生成する機能です。

戻り値

- 0：コマンドが正常に終了しました。
- -1：コマンドが異常終了しました。または，テンプレート・部品のパラメタや可変記号が解決できません。

4 コンパイルと単体テスト

SEWB+/CONSTRUCTION で作成したソースプログラムを，COBOL 開発マネージャ上でコンパイルします。また，COBOL2002 のテストデバッガを利用して単体テストをします。

4.1 COBOL 開発マネージャでのコンパイル

4.1 COBOL 開発マネージャでのコンパイル

SEWB+/CONSTRUCTION で作成したソースプログラムをコンパイルする場合，COBOL 開発マネージャを利用します。なお，ここでは，COBOL2002 の COBOL 開発マネージャを利用する場合について説明します。COBOL 開発マネージャの詳細は，マニュアル「COBOL2002　操作ガイド」を参照してください。

4.1.1 COBOL 開発マネージャ上でのコンパイルの手順

COBOL 開発マネージャでは，開発資源をプロジェクトおよびプロジェクトマスタ単位で管理します。また，ビルドおよびリビルド機能を利用する場合も，プロジェクトおよびプロジェクトマスタに登録されている開発資源の関連情報が必要です。

(1) プロジェクトとプロジェクトマスタを作成する

COBOL 開発マネージャでソースプログラムをコンパイルする場合は，ファイルなどの開発に必要な資源を一括して管理するためのプロジェクトと，複数のプロジェクトを一括して管理するためのプロジェクトマスタを作成します。

作成方法を，次に説明します。

プロジェクトマスタから新規に作成する場合

COBOL 開発マネージャの [プロジェクトマスタ] ― [新規作成] を選び，プロジェクトマスタとプロジェクトを作成します。

既存のプロジェクトマスタに追加する場合

COBOL 開発マネージャの [プロジェクト] ― [プロジェクトの作成] を選び，プロジェクトを作成します。

(2) ファイルの関連を登録する

AP 作成に使用するソースファイルおよびプログラム定義ファイルの関連を，次に説明する方法で登録します。

プロジェクトマスタから新規に作成する場合

1. COBOL 開発マネージャの [プロジェクトマスタ] ― [新規作成] を選びます。
   次に示すダイアログでは，メインファイル名を指定しないで [完了] ボタンを押してください。

注
Windows XP の場合に表示される画面を使用しています。

2. COBOL 開発マネージャの [プロジェクト] ― [ソースファイルの追加] を選び，プログラム定義ファイルを設定してください。プログラム定義ファイルとソースファイルの関連が登録されます。

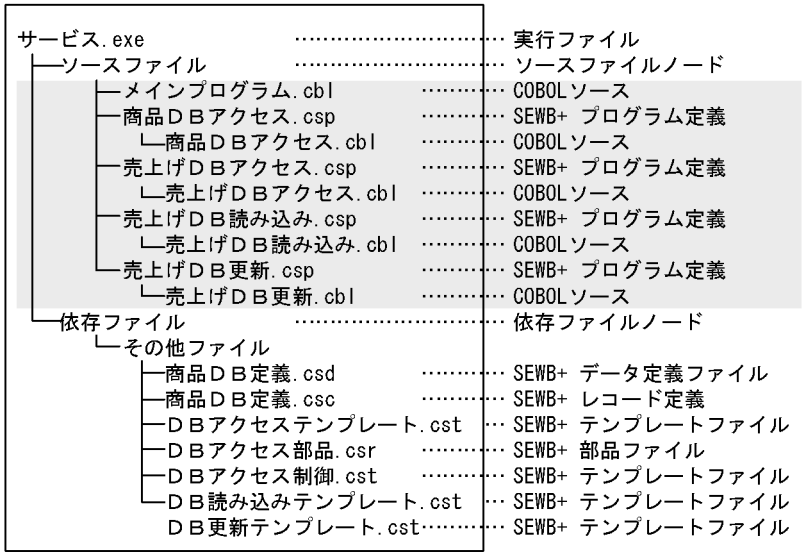
既存のプロジェクトマスタに追加する場合

COBOL 開発マネージャの [プロジェクト] ― [ソースファイルの追加] を選び，プログラム定義ファイルを設定することで，プログラム定義ファイルとソースファイルの関連が登録されます。

登録したファイルをビルドすると，それぞれのソースファイルおよびプログラム定義ファイルで使用された COPY メンバ，データ定義ファイル，テンプレート，および部品の関連が付けられます。

各ファイルの関連を図 4-1 に示します。

図 4-1 各ファイルの関連



（凡例） ：ユーザが登録する関連
注 この例題は，COBOLソース形式が固定形式の場合です。

(3) ビルドまたはリビルドを実行する

ビルドまたはリビルドを実行すると，プロジェクトに登録された資源から実行可能ファイル（.exe，.dll）が作成されます。

データ定義やプログラム定義の修正が起こった場合は，ビルドまたはリビルド機能で自動的にソースファイルを再生成させて，実行可能ファイルを作成することができます。

(4) 単体テストをする

COBOL 開発マネージャ上でテストをするには COBOL2002 のテストデバッガを使用します。COBOL 開発マネージャ上でのテストは単体テストなので，SEWB+/CONSTRUCTION で作成したサーバ側の AP をテストする場合は，COBOL2002 のシミュレーション機能を利用してください。クライアント側の AP を COBOL2002 言語で作成した場合は，クライアント側の AP もテストできます。

OpenTP1 を使用しているシステムの場合は，OpenTP1 のテスタ機能※を使用して，RPC およびメッセージ送受信のサーバ側の定義部分をテストできます。

注※

OpenTP1 をテストモードで動作させるオンラインテスタ（TP1/Online Tester），および OpenTP1 の API をテストするオフラインテスタ（TP1/Offline Tester）があります。

（5）SEWB+/REPOSITORY に登録する

単体テスト後，プロジェクトマスタファイル，実行ファイルおよびカバレージファイルなど作成されたファイルを SEWB+/REPOSITORY に登録します。

4.1.2 コンパイル時の注意事項

（1）プロジェクトマスタファイルをリポジトリに登録する場合

コンパイル・テスト後に作成されたファイルをリポジトリに登録する場合は，あらかじめ，COBOL 開発マネージャで使用するファイルをドキュメント種別としてリポジトリに登録しておきます。ドキュメント種別の登録の詳細は，マニュアル「SEWB+/REPOSITORY 運用ガイド」を参照してください。

（2）プリコンパイラを使用している場合（ORACLE を利用している場合など）

最初にプリコンパイラを実行させ，プリコンパイラが生成したソースプログラムを COBOL 開発マネージャに登録します。

（3）COBOL 言語以外のソースがある場合

TP モニタに OpenTP1 を使用している場合，スタブソースが生成されます。スタブソースは C 言語で作成されているため，あらかじめ C 言語のコンパイラでコンパイルしておきます。

このように COBOL 言語以外のソースがある場合は，あらかじめ専用のコンパイラでコンパイルし，オブジェクトファイルにしてから COBOL 開発マネージャに登録します。

（4）ソースファイルに修正が発生した場合

ソースファイルに修正が必要になった場合，ソースファイルの内容は修正しないでください。修正は，必ずソースファイルの生成元であるデータ定義およびプログラム定義に対して行います。データ定義やプログラム定義は，COBOL 開発マネージャから起動できます。また，UOC リバース機能を使えば，ソースプログラム上で編集したユーザ追加処理をプログラム定義ファイルに取り込むこともできます。

テンプレートに修正が必要な場合は，テンプレート作成者に依頼してください。

注

ビルドを利用すると，定義情報が変更されたものだけを自動的に再生成します。

（5）SEWB+/REPOSITORY で管理されているデータ定義ファイルを利用して，ビルドまたはリビルドを行う場合

ビルドまたはリビルドを行う前に，SEWB+/REPOSITORY でリポジトリ接続を起動させておく必要があります。

ただし，SEWB+/REPOSITORY-BROWSER が起動されている場合は，すでに SEWB+/REPOSITORY と接続されているため，特に作業する必要はありません。

(6) XMAP3 を使用している場合

COBOL85 Version5.0 以前の開発マネージャを使用している場合，ビルド時にマップファイル（論理マップ，物理マップ）が生成されない場合があります。

マップ定義ファイルを更新した場合，マップファイルも保存してください。XMAP3 ドローのメニューバーの [ファイル] から [上書き保存] または [ドローの終了] を選ぶと，マップファイルも保存されます。

運用の都合などでマップファイルを保存しない場合は，ビルド時にマップファイルが生成されないときがありますので，プロジェクトの編集で論理マップとマップ定義ファイルを関連づけてください。

(7) 辞書を参照してコンパイルする場合

COBOL 開発マネージャの [プロジェクト] ― [プロジェクトの設定] を選び，[プロジェクトの設定] ダイアログを起動します。[プロジェクトの設定] ダイアログのプロジェクト一覧でプログラム定義ファイルを選択したあと，コンパイラオプションに「/ud」を設定してください。「/ud」については，「3.3.2　コマンドでの生成（ソースプログラム）」を参照してください。

(8) プログラム定義ファイルを最新状態にする場合

COBOL 開発マネージャの [プロジェクト] ― [プロジェクトの設定] を選び，[プロジェクトの設定] ダイアログを起動します。[プロジェクトの設定] ダイアログのプロジェクト一覧でプログラム定義ファイルを選択したあと，コンパイラオプションに「/r」を設定してください。「/r」については，「3.3.2　コマンドでの生成（ソースプログラム）」を参照してください。

5 テンプレートとは

この章では，テンプレートを作成するために，まずは知っておかなければならない基本的な事柄について説明します。

5.1 テンプレートとは

C/S システムのサーバプログラムを作り，さらにクライアントプログラムに必要なサーバプログラムとのリンクのインタフェース（テーブル）を作り込むことは，複雑で面倒な仕事でした。その作業を軽減するためのツールが SEWB+/CONSTRUCTION です。

SEWB+/CONSTRUCTION はデータ定義という機能とテンプレート※を基にして，プログラム定義という機能を補助的に使うことによって，特に C/S システムのプログラムを効率良く作ることを目的にしています。

テンプレートとは，プログラムのひな型のようなものです。例えば，業務プログラムを作成するときに，検索や更新，帳票出力という処理ごとにテンプレートを用意しておけば，それだけ効率良くプログラムを作成できます。また，作成したプログラムは，テンプレートを使わない場合よりも標準化されます。このような考え方をベースに，業務の特徴や C/S システムの環境（オンライン処理なのか，リモート処理なのか，バッチ処理なのか，など）も考慮に入れてテンプレートを用意すれば，より効果的です。

さらに，SEWB+/CONSTRUCTION は，テンプレートでは決定できない（プログラムによって変わる）部分を，ウィンドウに表示させて入力を促す機能を持っています。それがプログラム定義のウィンドウです。データ定義やプログラム定義のウィンドウで必要なデータを入力し，ソースプログラムを生成するのは，プログラム作成者です。

テンプレート作成者は，生成したいソースプログラムの言語（COBOL または C）と，テンプレート記述言語を使ってテキストエディタでテンプレートを作成します。COBOL 言語のテンプレートの例（イメージ）を，図 5-1 に示します。

注※

バッチシステム向けおよび C/S システム向けのテンプレートと部品が用意されています。詳細は「付録 C　テンプレートおよび部品の紹介」を参照してください。

図 5-1　COBOL 言語のテンプレートの例（イメージ）

```
#NAME   = 'シーケンシャルファイルマスタ更新'
#OWNER    = '日立'
#OUTLINE =              '入力ファイルは順編成'
#       '出力は DB(HiRDB 用)'
@@lang  COBOL;
@@interface  @入力ファイル =
    {
       ATTR       = FILE,
       COMMENT    = '入力ファイル必須',
       IO         = IN,
       ファイル名   = {ATTR=FILE_NAME,    COMMENT='ファイル名'},
       編成         = {ATTR=ORGANIZATION,COMMENT='編成'},
       レコード名   = {ATTR=RECORD_NAME,  COMMENT='レコード名'},
       レコードキー = {COMMENT='キー'},
       外部装置名   = {COMMENT='外部装置名'},
       ブロック長   = {COMMENT='ブロック長'},
       アクセスモード={COMMENT='ファイルアクセス"S","R"または"D"を入力'}
    };
@@************************************************************
@@* シーケンシャルファイル入力処理
@@************************************************************
 IDENTIFICATION DIVISION.
 @@interface @PROG_ID ={COMMENT='プログラム ID 英数字で 8 文字以内'};
  PROGRAM-ID. @PROG_ID..
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
@@*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
       SELECT @入力ファイル[ファイル名]
                    ASSIGN TO @入力ファイル[外部装置名]
                    @@if (@入力ファイル[アクセスモード] eq 'S')
                    ACCESS MODE IS SEQUENTIAL
                    @@end;
                    FILE STATUS IS @入力ファイル[ファイル名].-STAT.
@@*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 DATA  DIVISION.
 FILE  SECTION.
@@*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
       FD  @入力ファイル[ファイル名]..
                    @@expand(@入力ファイル[レコード名]);
@@*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

5.1.1　C/S システムの AP 作成に必要なテンプレート

SEWB+/CONSTRUCTION で C/S システムの AP を作成するためには，どのような内容のテンプレートが必要になるのかを示します。

(1) サーバ側およびクライアント側の AP を作成するためのテンプレート

サーバ側およびクライアント側の AP で，SEWB+/CONSTRUCTION で作成することをお勧めしている AP と，それらの AP 作成に必要なテンプレート中の記述を表 5-1 に示します。

表 5-1 作成する AP と必要なテンプレート記述例

SEWB ＋ /CONSTRUCTION で作成することをお勧めする AP		テンプレート中に必要な記述
サーバ側の AP	処理 AP	DB アクセス，ファイルアクセスなどの制御処理※ 1，※ 2
	クライアント側からサーバ側にアクセスするためのインタフェーステーブル	メッセージ送受信の場合の制御処理
		SPP（サービス提供プログラム）としての処理※ 2
		DB やファイルの利用宣言※ 2，※ 3
		メッセージなどのインタフェースの利用宣言※ 2，※ 3
		CORBA オブジェクト，オペレーションの宣言
クライアント側の AP	ODBC インタフェース，または DB アクセス機能を利用して作成するクライアント側の AP	DB アクセス，ファイルアクセスなどの制御処理※ 1
	サーバ側にアクセスする AP（クライアント側のサーバアクセス用の AP をユーザが作成する）	サーバアクセスの処理
		RPC（RPC の場合）※ 2
		メッセージ送受信の場合は，SEND/RECEIVE の発行
		SUP（サービス利用プログラム），CUP（クライアントユーザプログラム）としての処理 DB やファイルの利用宣言※ 2，※ 3
		メッセージなどのインタフェースの利用宣言※ 2，※ 3
	GUI に XMAP3 を利用して作成するクライアント AP	画面入力データの取得処理

注※ 1

DB にアクセスする処理を部品として作成しておくと，複数の AP 間で共用できます。

注※2
サンプルプログラム中に書かれています。

注※3
テンプレートの宣言部に書きます。

5.1.2 テンプレートと各定義ファイルとの関係

テンプレートの内容が，SEWB+/CONSTRUCTION のそれぞれの定義に，どのように影響するかを示します。

(1) TP モニタを利用する C/S システムの AP を作成する場合

TP モニタを利用する C/S システムの AP を作成する場合に定義する情報，およびそれぞれの関係について説明します。

(a) 定義する情報

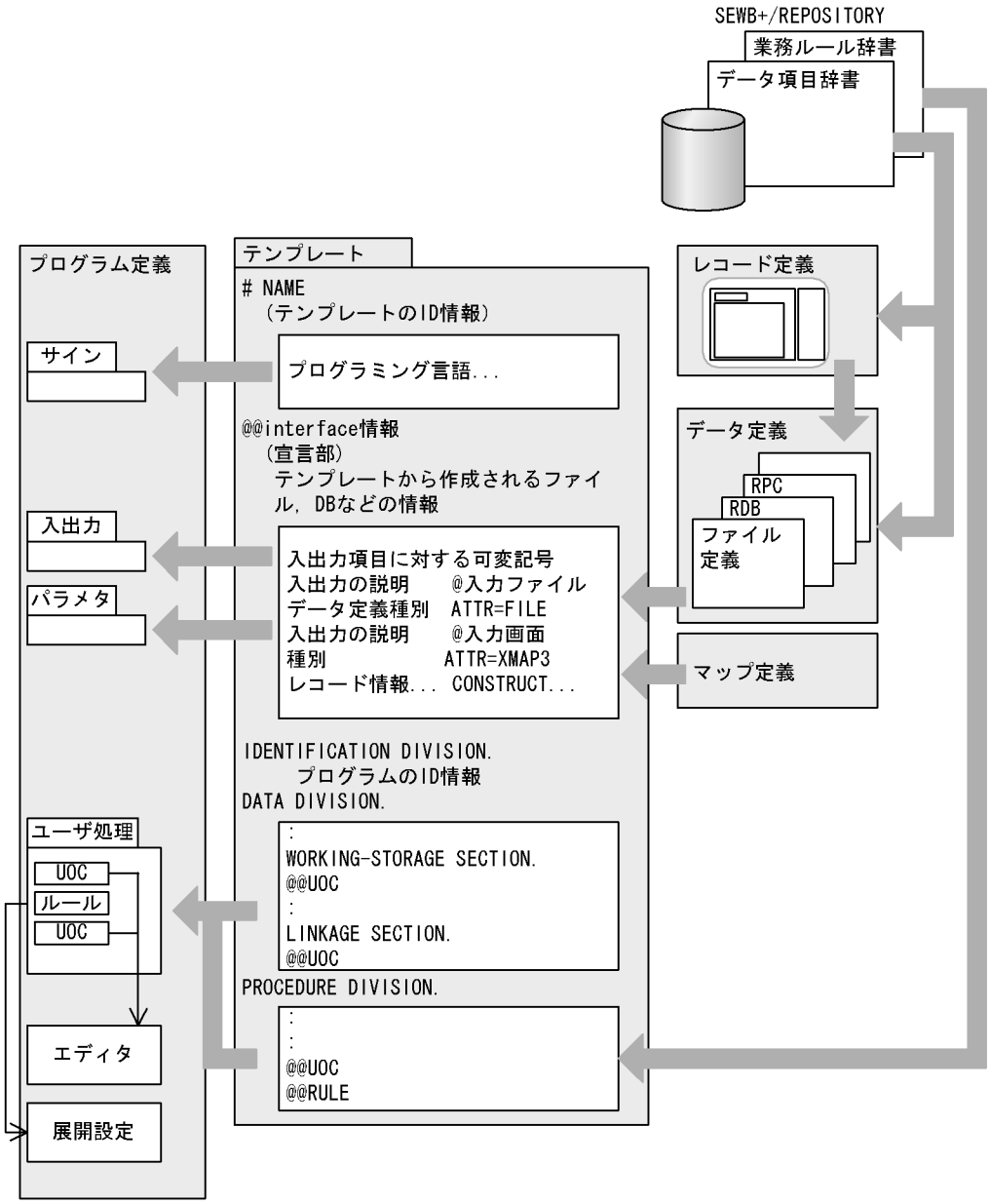
TP モニタを利用する C/S システムの AP を作成する場合には，次の定義が必要です。

- テンプレート作成（テンプレート作成者によってすでに作成されています）
- データ定義
- マップ定義
- プログラム定義

(b) 各定義の関係

テンプレートの内容が，データ定義，マップ定義およびプログラム定義にどのように影響するかを図 5-2 に示します。

図 5-2 テンプレートと，データ定義，マップ定義，プログラム定義の関係



(2) CORBA のオブジェクトを作成する場合

CORBA のオブジェクトを作成する場合に定義する情報，およびそれぞれの関係について説明します。

(a) 定義する情報

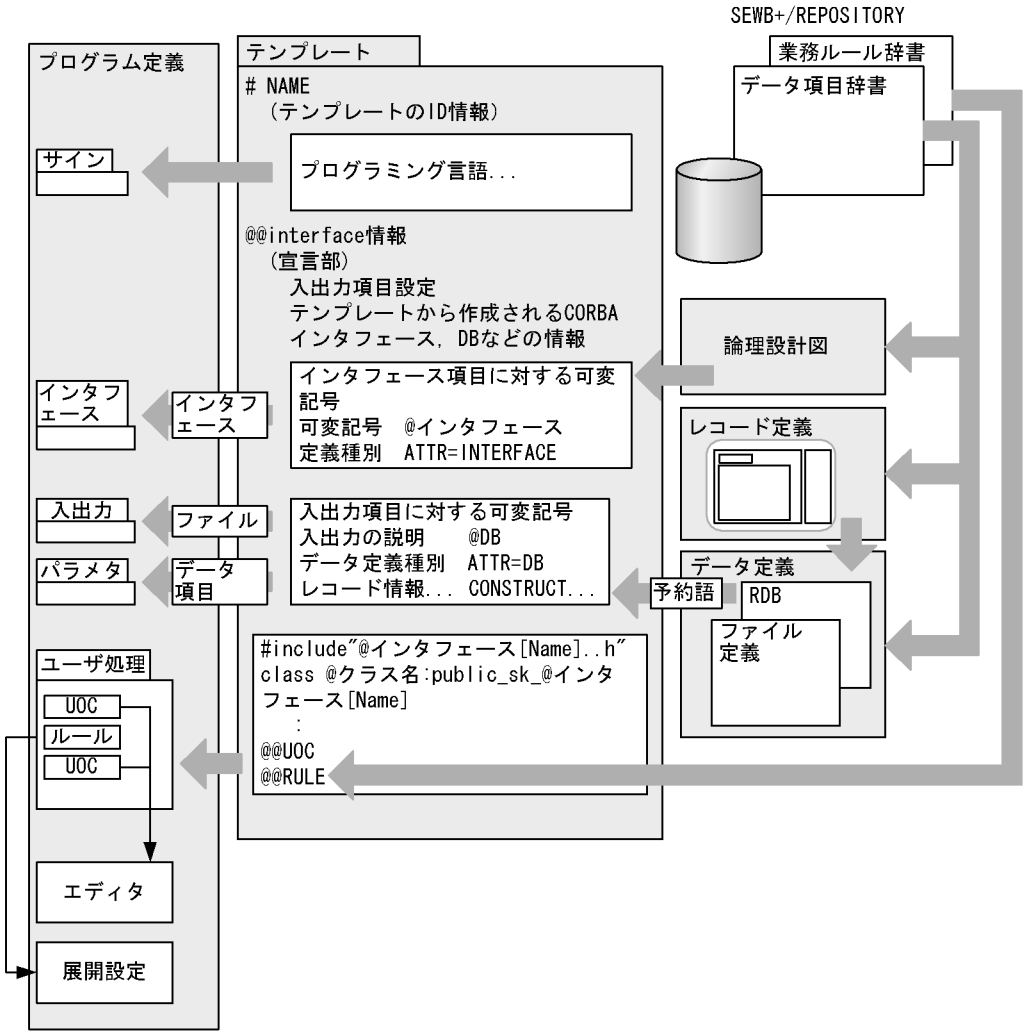
CORBA のオブジェクトを作成する場合には，次の定義が必要です。

- テンプレート作成（テンプレート作成者によってすでに作成されています）
- 論理設計図（SEWB+/CS-DESIGN ですでに作成されています）
- データ定義
- プログラム定義

(b) 各定義の関係

テンプレートの内容が，データ定義，プログラム定義および SEWB+/CS-DESIGN で定義した論理設計図にどのように影響するかを，図 5-3 に示します。

図 5-3　テンプレートとデータ定義，プログラム定義，論理設計図の関係



(3) テンプレートと定義内容の関係

(a) データ定義とテンプレート

データ定義では，AP で使用するファイルや DB，およびメッセージなどを定義します。データ定義でレコードを使用する場合は，SEWB+/REPOSITORY で定義された最上位結合項目，または SEWB+/RECORD DEFINER で定義されたレコード定義ファイルを参照して使用します。

また，TP モニタを利用する場合と利用しない場合とで，AP で使用できるデータ定義種別が決まります。詳細は，「2. データ定義」を参照してください。

データ定義での定義情報は，プログラム定義時にテンプレートからの要求に応じて，プログラム定義の入出力項目の定義またはパラメタの定義をする画面に表示されます。

テンプレートに @@rule 文がある場合は，データ定義で定義されたデータ項目と関連のある業務ルールおよび同一項目用業務ルールが，SEWB+/REPOSITORY から抽出されてプログラム定義のユーザ処理を設定する画面に表示されます。業務ルールの詳細および同一項目用業務ルールの詳細は，マニュアル「SEWB+/REPOSITORY　辞書設計ガイド」を参照してください。

(b) 論理設計図とテンプレート

SEWB+/CS-DESIGN で作成した論理設計図を使って，CORBA のオブジェクトの実装プログラムを作成する場合，テンプレートにはオブジェクト情報を取得するための @@interface 文を書きます。オブジェクト作成時には，@@interface 文が書かれているテンプレートを選びます。プログラム定義では，使用する論理設計図ファイルを選び，さらにその中からソース生成の対象となるオブジェクトを指定して，ソースプログラムを生成します。

(c) プログラム定義とテンプレート

プログラム定義ではテンプレートとデータ定義の情報を取り込み，プログラム生成に必要な情報を設定します。そのため，プログラム定義では次のような作業が必要になります。

- AP で使用する入出力項目の定義
- テンプレートに書かれている指示項目の定義
- AP 固有の処理であるユーザ追加処理の編集
- ソースプログラム中に展開させる業務ルール処理の設定
- オブジェクトで使用するインタフェースの定義
- AP やオブジェクトで使用する部品の定義

注

プログラム作成者は，用意されたテンプレートの中から目的の AP を作成するために最適なテンプレートを選んで使用します。該当するテンプレートがない場合や，定義項目についての説明が不十分でデータ定義やプログラム定義に支障が生じる場

合は，テンプレート作成者に修正を依頼してください。

(d) ビジネスプロセス定義ファイル（XML ファイル）とテンプレート

WorkCoordinator Definer で作成したビジネスプロセス定義ファイル（XML ファイル）を使用してソースプログラムを生成する場合，テンプレートには，入力パラメタとなるビジネスプロセス定義ファイル（XML ファイル）を取り込むための @@interface 文を記述します。プログラム定義では，ビジネスプロセス定義ファイル（XML ファイル）を基に，ソースプログラムに取り込む部分の項目を選択して，ソースプログラムを生成します。

(e) マップ定義ファイルとテンプレート

XMAP3 のドロー機能で作成した画面・帳票のレイアウト定義（マップ定義ファイル）を使用してソースプログラムを生成する場合，テンプレートには，定義情報を取得するための @@interface 文を記述します。プログラム定義では，使用するマップ定義ファイルを選び，ソースプログラムを生成します。

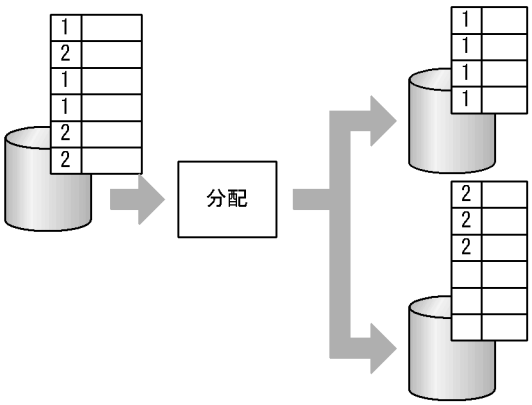
5.2 SEWB+/CONSTRUCTION で作れる C/S システムのプログラム

SEWB+/CONSTRUCTION で作れるプログラムの仕様は，生成するソースプログラムの言語（COBOL や C）に依存しています。ここでは，テンプレートを使って作成できる業務処理の代表的な例を紹介しますが，テンプレートの作り方によって，より複雑な業務処理を作成できます。

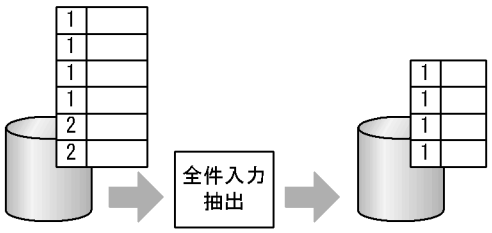
5.2.1 バッチ処理のプログラム

SEWB+/CONSTRUCTION で作れるファイルや DB，帳票を使ったバッチ処理の代表的な例を示します。ここでは，帳票は CSV（Comma Separated Values）ファイルに出力することを想定しています。

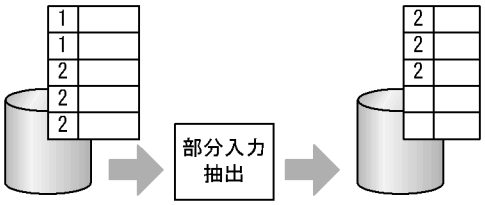
(1) 分配



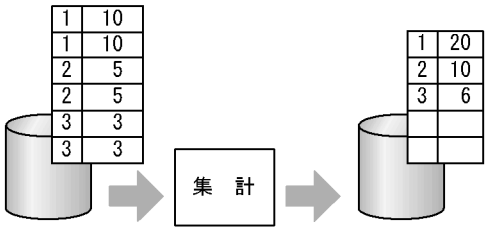
(2) 全件入力抽出



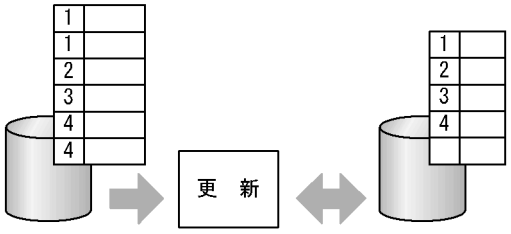
(3) 部分入力抽出



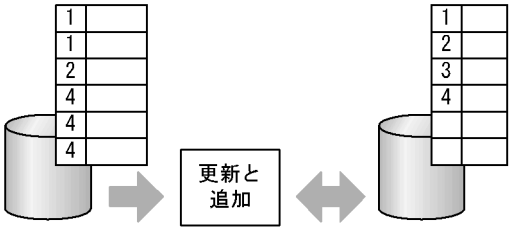
(4) 集計



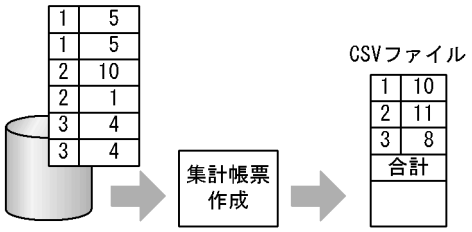
(5) 更新



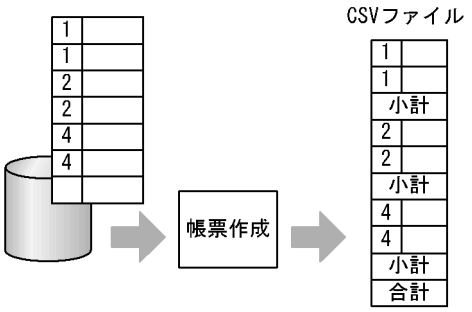
(6) 更新と追加



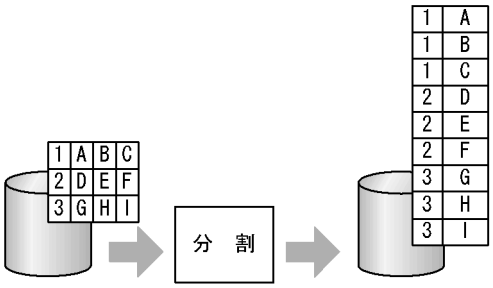
(7) 集計帳票作成



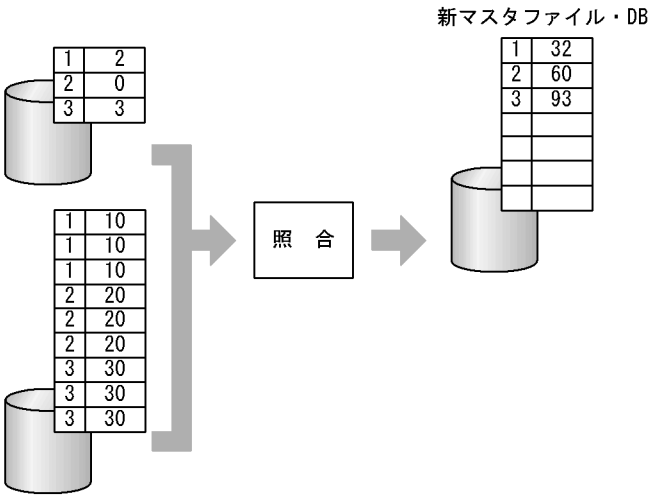
(8) 帳票作成



(9) 分割



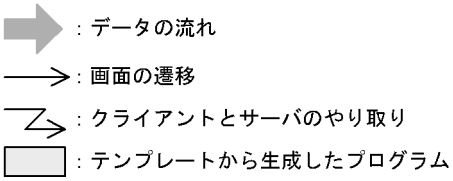
(10) 照合



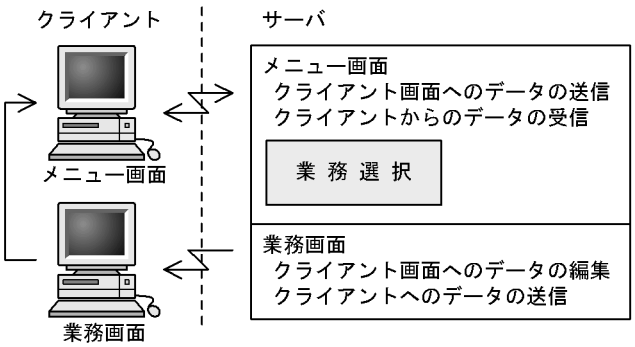
5.2.2　オンライン処理のプログラム

SEWB+/CONSTRUCTION で作れる，画面を使ったオンライン処理の代表的な例を示します。ここでは，クライアントプログラムとサーバプログラムとの通信に TP モニタを使っていることを想定しています。使う TP モニタによって，このほかの処理を作成することもできます。

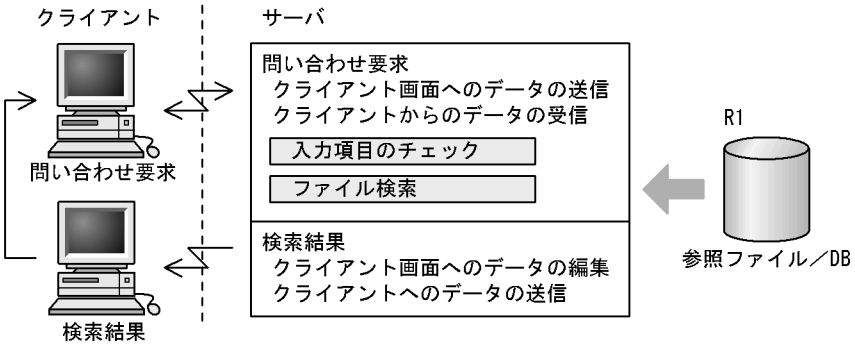
また，図中で使用している記号は次の意味を持っています。



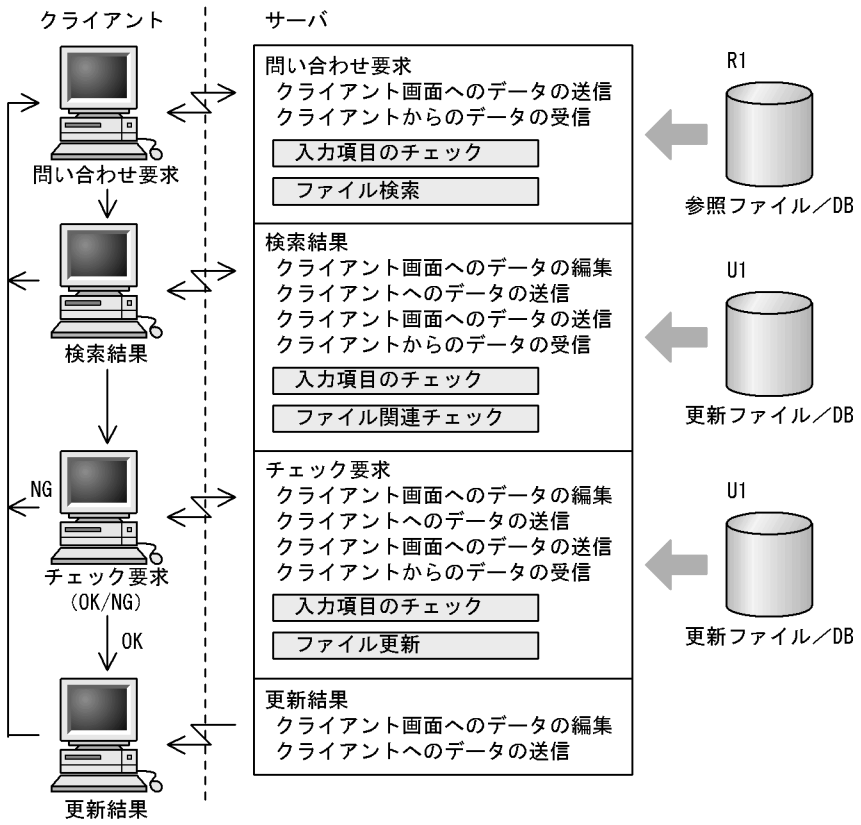
(1) 業務振り分け



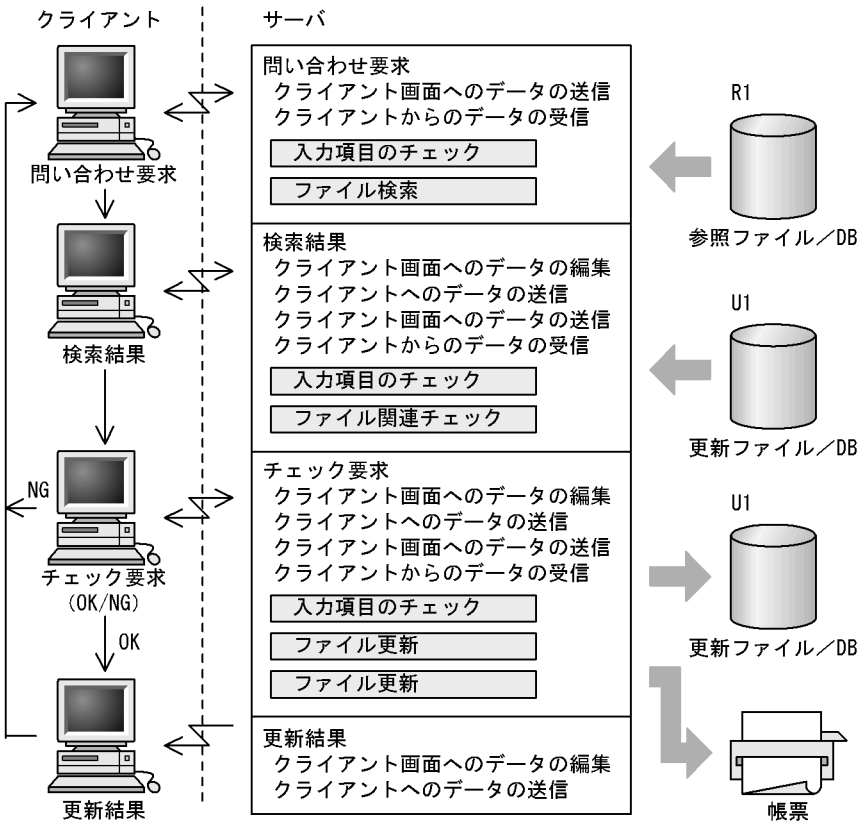
(2) 問い合わせ



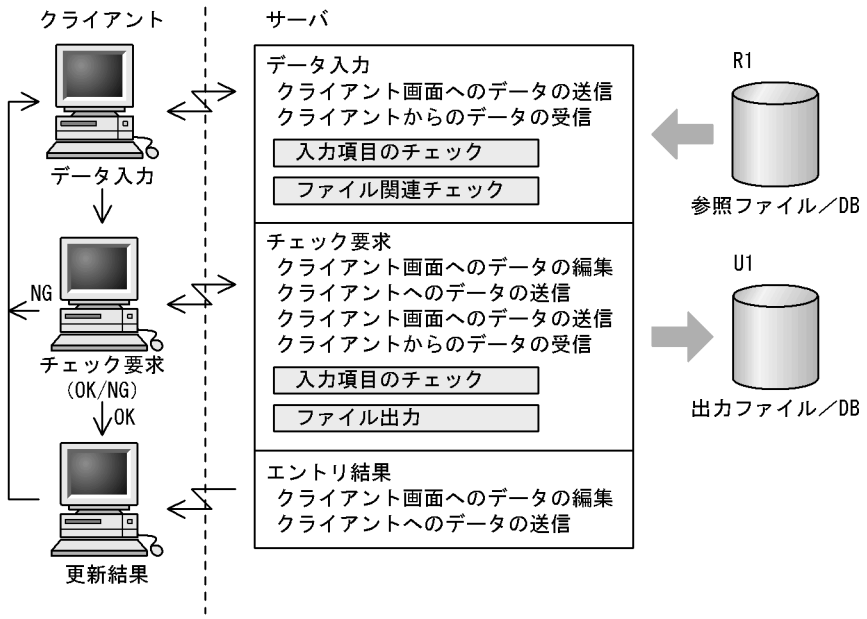
(3) 更新


クライアント
サーバ
問い合わせ要求
検索結果
NG
チェック要求
(OK/NG)
OK
更新結果
問い合わせ要求
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル検索
検索結果
クライアント画面へのデータの編集
クライアントへのデータの送信
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル関連チェック
チェック要求
クライアント画面へのデータの編集
クライアントへのデータの送信
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル更新
更新結果
クライアント画面へのデータの編集
クライアントへのデータの送信
R1
参照ファイル／DB
U1
更新ファイル／DB
U1
更新ファイル／DB

(4) 更新と帳票出力


クライアント
サーバ
問い合わせ要求
検索結果
NG
チェック要求
(OK/NG)
OK
更新結果
問い合わせ要求
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル検索
検索結果
クライアント画面へのデータの編集
クライアントへのデータの送信
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル関連チェック
チェック要求
クライアント画面へのデータの編集
クライアントへのデータの送信
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル更新
ファイル更新
更新結果
クライアント画面へのデータの編集
クライアントへのデータの送信
R1
参照ファイル／DB
U1
更新ファイル／DB
U1
更新ファイル／DB
帳票

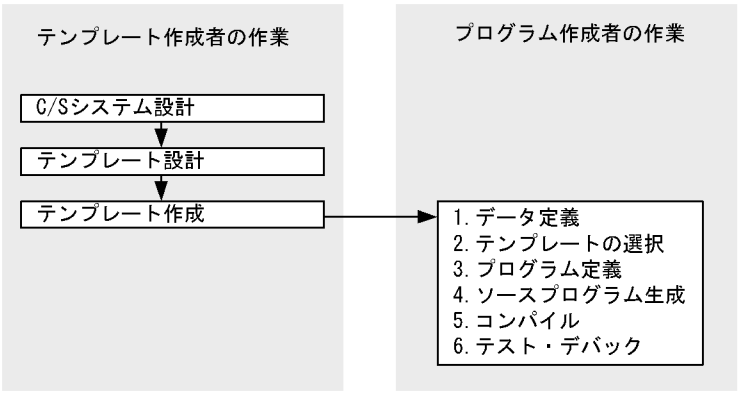
(5) データエントリ


クライアント
サーバ
データ入力
データ入力
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル関連チェック
R1
参照ファイル／DB
NG
チェック要求
(OK/NG)
OK
チェック要求
クライアント画面へのデータの編集
クライアントへのデータの送信
クライアント画面へのデータの送信
クライアントからのデータの受信
入力項目のチェック
ファイル出力
U1
出力ファイル／DB
更新結果
エントリ結果
クライアント画面へのデータの編集
クライアントへのデータの送信

5.3 SEWB+/CONSTRUCTION でソースプログラムを生成する手順

5.3.1 テンプレート作成者とプログラム作成者の作業

テンプレート作成者とプログラム作成者の作業は次のように分かれます。



テンプレート作成者は業務内容を理解し，それをどのような C/S システムとして構築するかを検討します。この検討結果を基に，テンプレートを設計し，作成します。作成したテンプレートはリポジトリに登録しておきます。

プログラム作成者は，作りたいプログラムに合うテンプレートをその中から選びます。そのあとは，データ定義，プログラム定義…と図のような手順で作業を進めます。

5.3.2 テンプレート作成の考え方

テンプレートを設計するとき，一つのプログラムに対して一つのテンプレートを作るのではテンプレートを利用するメリットがありません。一つのテンプレートで複数の業務に対応できるように，テンプレートを作ることが重要です。そのために，次の点をポイントにしてテンプレートを作成します。

1. **プログラム構造に影響していない処理で，固定化している処理を抽出し，部品として作成する。**
2. **プログラム構造がパターン化していて，ファイル名やデータ項目名が異なっている業務処理を集め，テンプレートと部品を作成する。**
   同じ処理を複数のプログラムで使用したり，複数個所で使用したりする場合に，その同じ処理をテンプレートや部品として作成します。SEWB+/CONSTRUCTION では，部品もテンプレートとして記述します。テンプレートとして記述することによって，ソースプログラムを生成するときにテンプレート中から部品を呼び出し，処理を生成することができます。

また，同じ処理でも使うファイルによって項目名だけが異なるとき，そのことをテンプレートや部品の中に定義します。SEWB+/CONSTRUCTION を使うと，プログラム生成時に使うファイルの項目名を換えて展開できます。

3. **条件によって一部の展開を変えるだけでパターン化できるような業務処理を集めてテンプレートと部品を作成する。**

SEWB+/CONSTRUCTION のテンプレートを使うと，与えられた条件（パラメタ）によって処理を選んで展開できます（生成するプログラムの処理内容そのものを変更できます)。

5.4 テンプレートの作り方

5.4.1 テンプレートに記述する内容

テンプレートには，次のような記述をします。

図 5-4 テンプレートに記述する内容

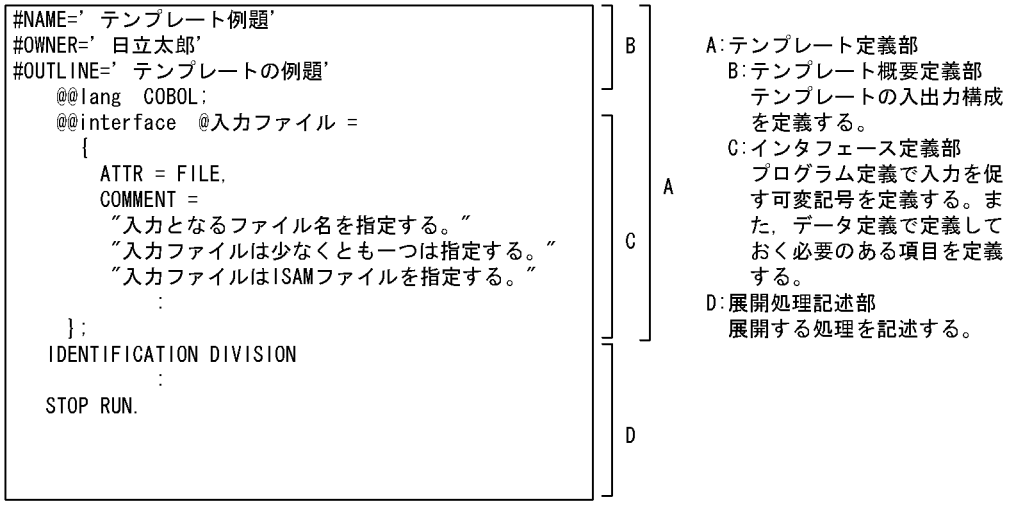


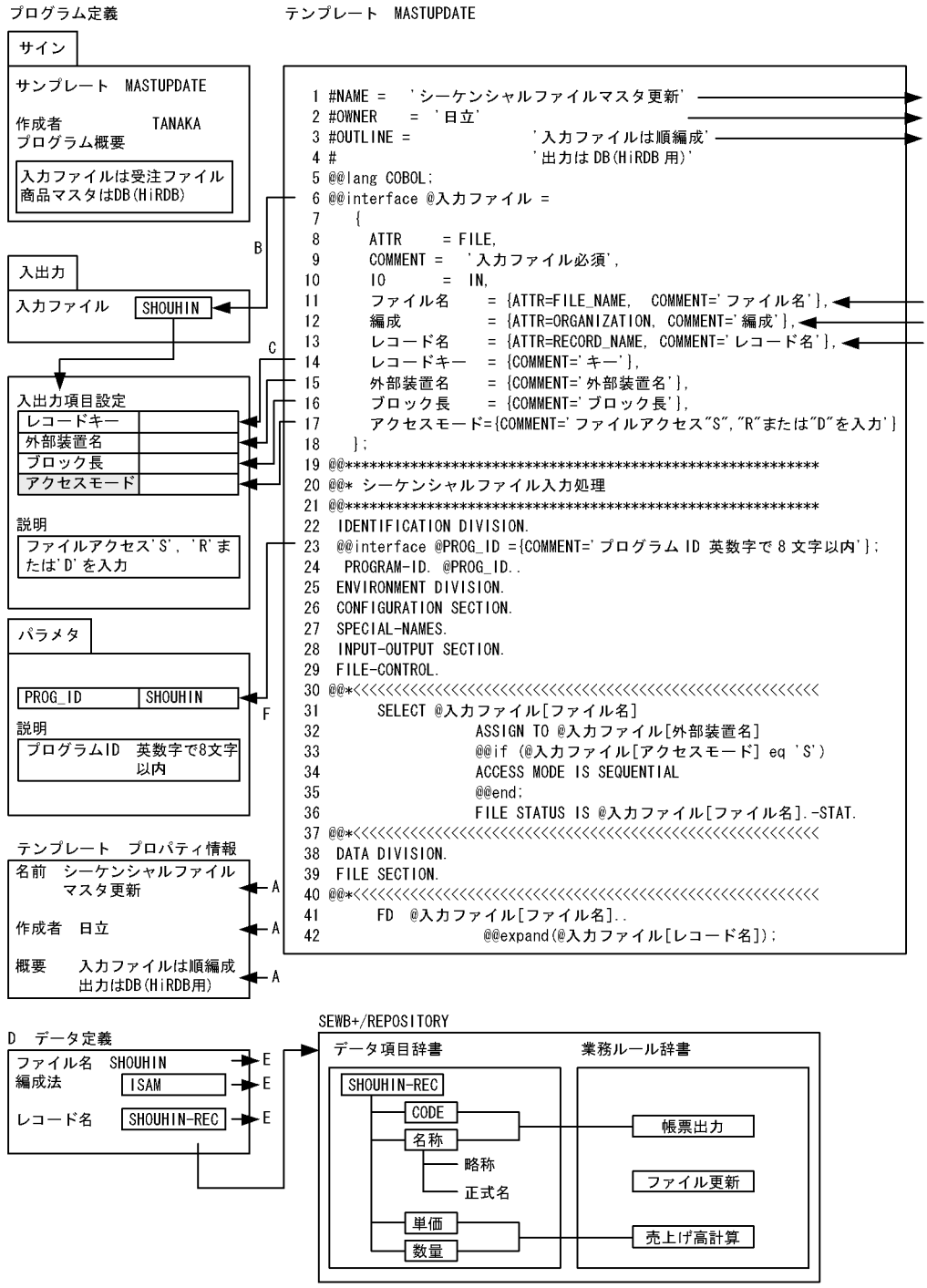
図 5-4 の A はテンプレート定義部です。まずテンプレートの概要を定義し（B），続いてデータ定義，プログラム定義とのインタフェースを定義します（C）。C で定義した内容が，プログラム定義のウィンドウで入力を促す項目として表示されます。また，ここで記述したコメントもウィンドウ上に表示されます。テンプレート定義部で定義した処理は，直接ソースプログラムには生成されません。

A（B，C）の記述の後に，そのままソースプログラムとして展開（生成）する処理を記述します（D）。生成したい言語を使い，その言語の記述規則に従って記述します。

5.4.2 テンプレートの定義

テンプレートとデータ定義，プログラム定義の関係について，例に沿って説明します（図 5-5 参照）。

図 5-5　テンプレートとデータ定義，プログラム定義との関係



(1) テンプレート概要定義部

1 ～ 4 行目はテンプレート概要定義部です。ここにはテンプレートの概要を記述します。

NAME，#OWNER，#OUTLINE はテンプレートの予約語です。ここに記述した内容は，テンプレートファイルのプロパティ情報として表示され，プログラム作成者はテンプレートを開かなくてもテンプレートの内容を知ることができます。

ここで特に重要なのは，#OUTLINE に記述する内容です。プログラム作成者に，テンプレートの目的，指定する内容，注意事項などを明確に伝える必要があります。テンプレート作成者が #OUTLINE に記述した内容は，テンプレートのプロパティ情報としてそのまま表示されます（図 5-5 の A)。プログラム作成者がテンプレートを選ぶ際に，疑問を持たないような，詳細な内容，わかりやすい表現を心掛けてください。

(2) インタフェース定義部

6 ～ 18 行目は，データ定義，プログラム定義とのインタフェース定義部です。

6 行目に @@interface @ 入力ファイルという記述があります。

@@interface @XXXXXX（この例の場合は「入力ファイル」）をテンプレートに記述することによって，プログラム定義ウィンドウに XXXXXX（この場合は「入力ファイル」）に対する指定を促す表示が出ます（図 5-5 の B)。そのため，ここをインタフェース定義部と呼びます。

@@ で始まる文は制御文であり，@ で始まる文は制御用の変数です。この変数は，プログラム作成者が値を設定できる可変記号です。

7 ～ 18 行目は，テンプレート作成者が，プログラム作成者に対してデータ定義とプログラム定義で定義してほしい，可変記号についての情報（属性など）を定義します。ここで定義した内容は，プログラム定義のウィンドウに入力を促す項目として表示されます（図 5-5 の C)。

8 行目の ATTR=FILE の ATTR は可変記号の種別を定義する ATTR であり，この例では可変記号「@ 入力ファイル」の種別がファイルであることを表しています。このほかにも，データ定義で定義できる，DB，DAM，TAM，RPC_INPARM（RPC 入力パラメタ)，RPC_REPLY（RPC 応答領域)，MSG（メッセージ)，UJ（ユーザジャーナル)，MSGLOG（メッセージログ)，WORK（共通作業領域）を記述できます。プログラム作成者は，プログラムの C/S システム形態（バッチ処理かオンライン処理か）や環境（使用する DB やデータ通信機能など）によって，データ定義で作成したファイル（図 5-5 の D）をプログラム定義で指定します。

また，11 ～ 13 行目の ATTR=FILE_NAME などで使われている ATTR はプログラム作成者がデータ定義で指定した内容をプログラムに取り込むための予約語の ATTR です（図 5-5 の E)。

14 ～ 17 行目の修飾名（レコードキー，外部装置名，…）は，プログラム定義の [入出力

項目設定] ダイアログで入力を促す項目として表示されます（図 5-5 の C）。このとき，修飾名に記述した COMMENT= の内容が，ウィンドウの説明欄に表示されます。例えば，アクセスモードが選択状態になっていれば，説明欄にはアクセスモードに記述した COMMENT= の内容が表示されるという具合に，表示内容が切り替わります。

（3） 展開処理記述部

19 行目からは，ソースプログラムとして生成する処理を記述する展開処理記述部です。

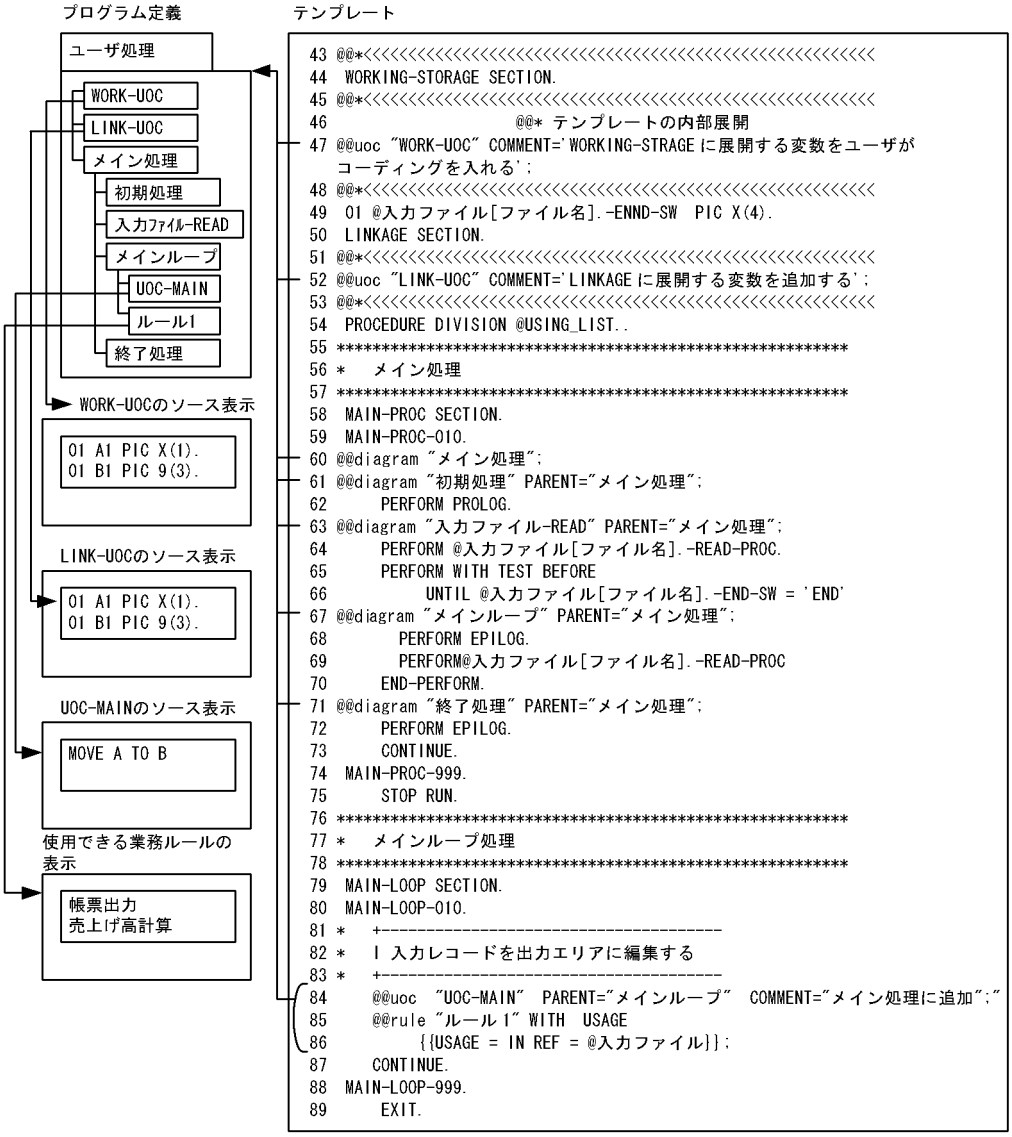
19 ～ 21 行目の @@ ＊は注釈です。

23 行目の @@interface @PROG_ID はプログラム定義の [パラメタ] タブに，入力を促す項目として表示されます（図 5-5 の F）。

@@interface の ATTR にデータ定義種別（ファイルや DB，DAM，TAM など）の指定がある場合は，[入出力設定] タブに表示され，それ以外は [パラメタ] タブに表示されます。

さらに，図 5-6 でテンプレートの続きを説明します。

図 5-6　テンプレートとプログラム定義（追加処理）の関係



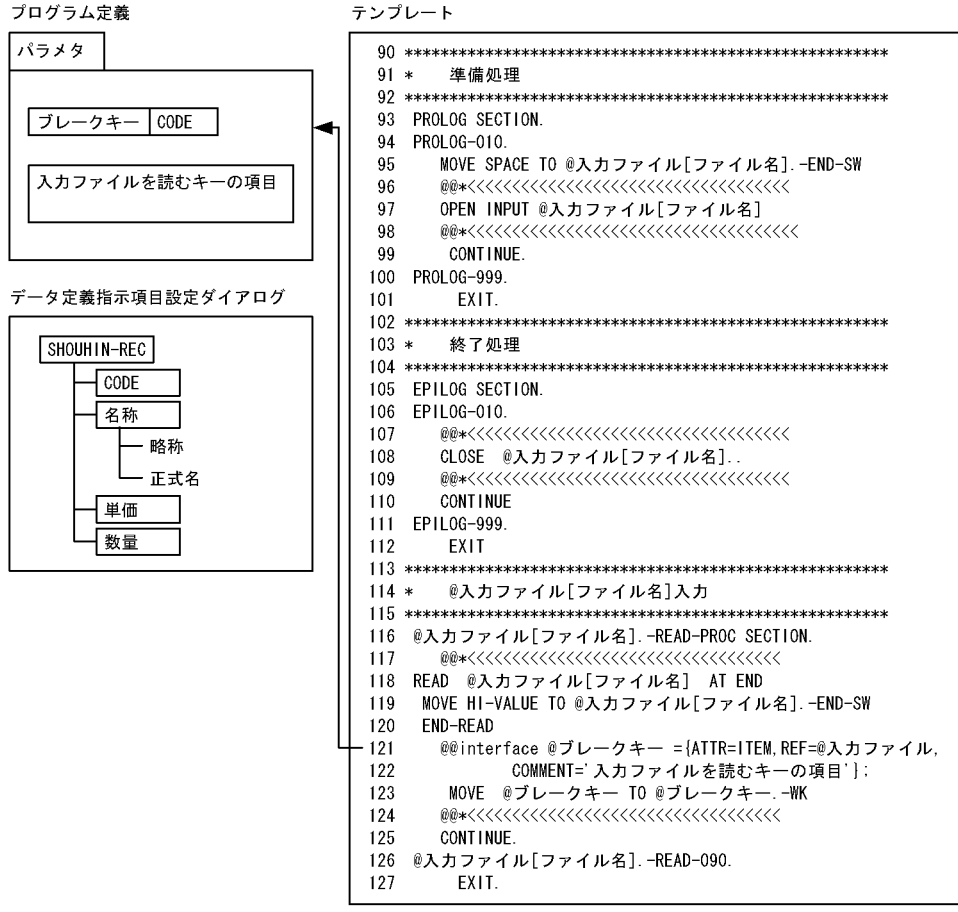
47 行目の @@UOC　"WORK-UOC" はユーザ追加処理に対する指定をプログラム定義のウィンドウで促すための制御文です。テンプレートに @@UOC　"UOC 名 " を記述することによって，プログラム定義の [ユーザ処理] タブで，指定したユーザ追加処理を表示することができます。なお，ユーザ追加処理の開始と終了コメントを設定する場合には，ソースファイル上で編集したユーザ追加処理をプログラム定義ファイルのユーザ追加処理の中に取り込む（UOC リバース）ときに，ユーザ追加処理の開始と終了を判定するための UOC コメントを設定します。UOC コメントを出力するには次の二つの方法があります。

- @@lang 文でユーザ追加処理（@@UOC）の前後に UOC コメントを生成する指定をする。
- @@uoc 文の前後に直接 UOC コメントを記述する。

また，85 ～ 86 行目の @@rule " ルール 1" WITH USAGE{{USAGE=IN REF=@ 入力ファイル }}; は，SEWB+/REPOSITORY の業務ルール辞書に定義されている業務ルールを利用するための制御文です。業務ルールとは，データ項目に着目して，データ項目特有の処理を部品化したものです。業務ルールは，SEWB+/REPOSITORY の業務ルール辞書に格納されています。テンプレートに @@rule 文 " 業務ルール展開名 " を記述すると，プログラム定義の [ユーザ処理] タブに，使用できる業務ルール処理を表示できます。

次に，図 5-7 で再度テンプレートの続きを説明します。

図 5-7　テンプレート（キー項目の指定）とプログラム定義の関係



121 行目の @@interface @ ブレークキーに ATTR=ITEM の予約語を指定すると，REF= に指定した可変記号のレコード定義の内容がプログラム定義の [データ定義指示項目設定] ダイアログに表示されます。この例の場合，@ 入力ファイルに指定された SHOUHIN

のレコードの内容が表示されます。プログラム作成者はブレークキーとして使用する項目をこの中から選びます。

このように，ブレークキーやマッチングキーなど，ファイルや DB の中から項目を選ばせたいときには，テンプレート中に前述のような指定をします。なお，REF= に指定する可変記号は，この指定より前に，データ定義種別が定義してあることが条件になります（定義していないと，レコードを表示できません）。この例では 6 ～ 8 行目（図 5-5）に @ 入力ファイルが定義してあります。

このほかにも，展開処理記述部には部品を呼び出したり，部品中に記述された処理を任意の位置に埋め込んだり，といった指定ができます。部品の利用方法については，「6.1.2 マスタ更新・追加出力の例題」を参照してください。

6 テンプレートを使った例題

一般的なバッチ処理とオンライン処理のプログラムを題材に，テンプレートの作り方を説明します。生成したソースプログラムの例も掲載していますが，対応しての見やすさを優先しているため，実際に生成されるものとは改行位置やインデンテーションなどに少し違いがあります。正確な生成ソースプログラムは，サンプルプログラム中の「Manual Sample」を参照してください。

6.1 バッチ処理の例題

「6.1.1 ファイル編集・帳票出力の例題」では基本的な作りのテンプレートの例題を，「6.1.2 マスタ更新・追加出力の例題」では少し工夫した作りの例題を紹介します。

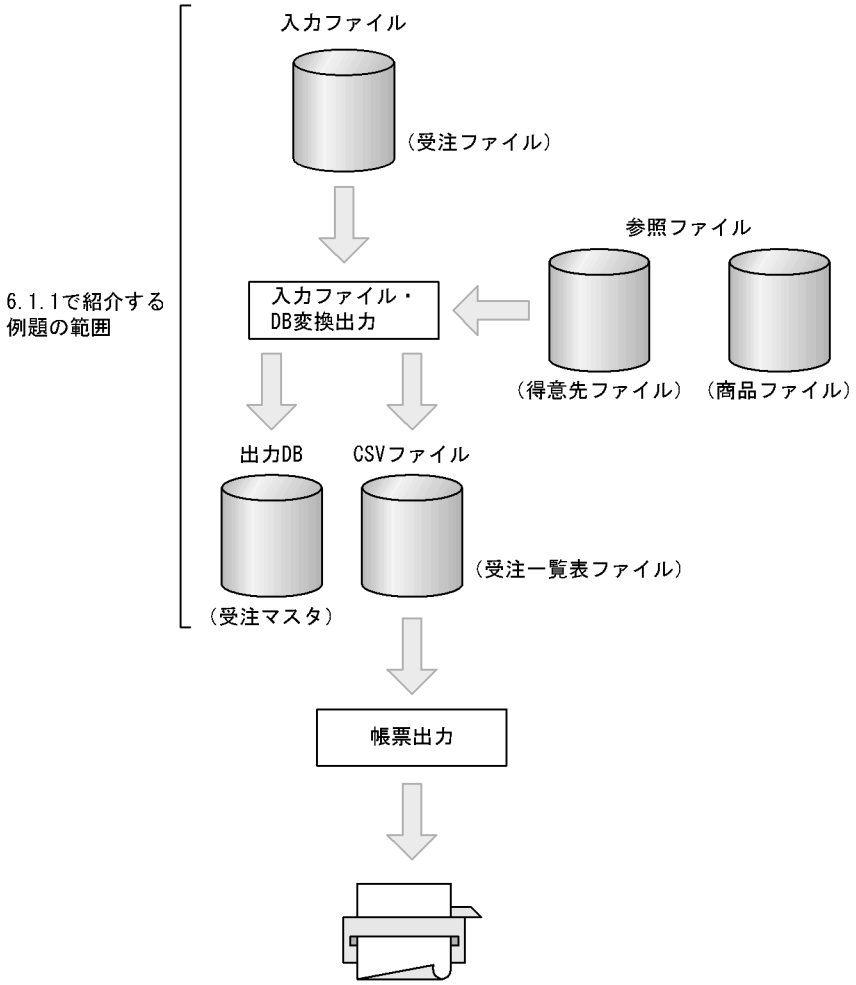
6.1.1 ファイル編集・帳票出力の例題

ここで紹介する例題は，テンプレートに可変記号を記述することでデータ定義やプログラム定義での入力を促すようにできる，SEWB+/CONSTRUCTION の基本的な機能だけを使ったテンプレートの例題です。部品なども使わないで，生成したい処理をそのままテンプレートに記述して作ります。

(1) 例題テンプレートで作れるプログラム

入力されたファイルのデータを，SQL を使って RDB に出力し，さらに，CSV ファイルの形式に変換して出力するプログラムが作成できます。CSV ファイルの形式に変換してあれば，EUR などの帳票出力ツールを使って印刷ができます。

(2) 例題テンプレートの入出力構成



(3) ファイル・DB のレコード形式

例題で使用するファイル・DB のレコード形式を次に示します。

- 入力ファイル（受注ファイル），および出力 DB（受注 DB）

伝票番号	得意先コード	受注日付	納期	納入年月日	商品コード

受注数量	受注単価

- 参照ファイル 1（得意先ファイル）

得意先コード	得意先名	得意先電話番号

- 参照ファイル２（商品ファイル）

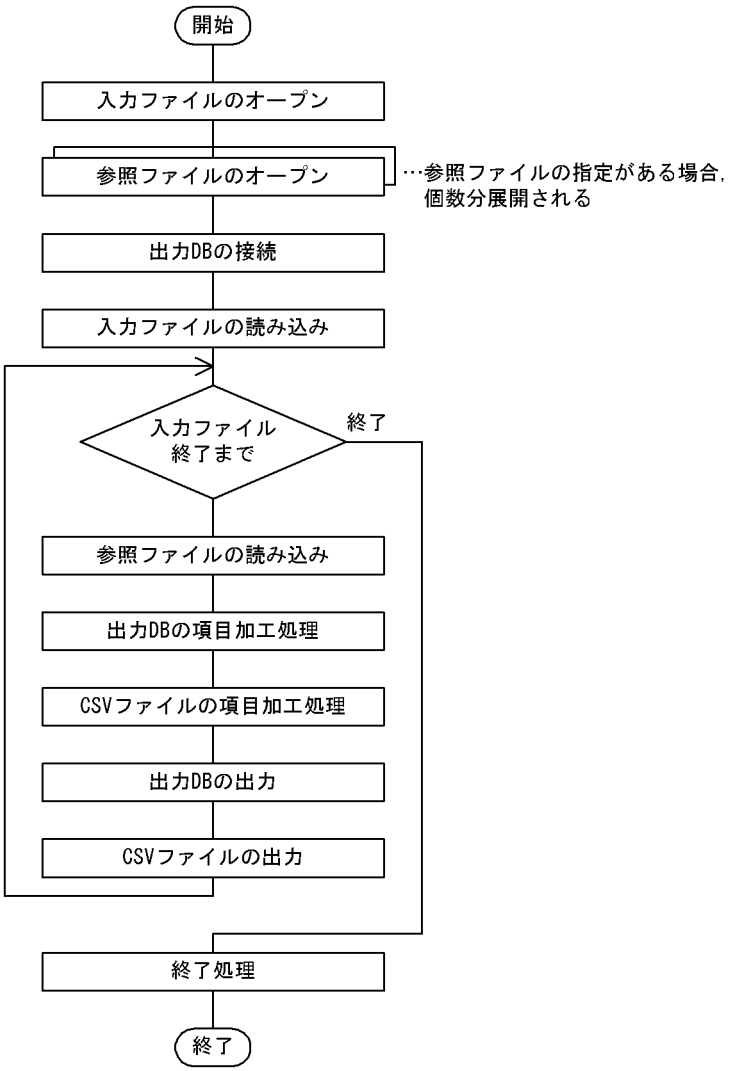
商品コード	商品名	色	標準卸単価	仕入先コード	仕入単価

- CSV ファイル（帳票 CSV）

伝票番号	得意先名	商品名	色	売上数量	売上単価	売上金額

（凡例）□ ：キーとなるデータ

(4) 例題テンプレートの処理概要



(5) 例題テンプレートの記述のポイント

1. @@interface 文を使って，入力ファイル，参照ファイル，出力 DB，CSV ファイルを使うことを宣言する。
2. プログラム作成者が，プログラム定義で入力ファイル，出力 DB，CSV ファイルの指定をしなかった場合に，エラーメッセージを出力し，ソースプログラムの生成を中止する処理を記述する（@@msg 文や @@errorexit 文を使用）。
3. 参照ファイルの個数をプログラム作成者に指定させるため，その指定された個数を取得する処理を記述する（@@count 関数を使用）。
4. @@expand 文を使って，プログラム作成者がデータ定義で定義したレコードを展開する処理を記述する。
5. プログラム作成者がプログラム定義の [入出力] タブで指定した項目（修飾名。この例では外部装置名）と同じ領域を持った項目を定義する（@@pic 関数を使用）。
6. プログラム作成者がデータ定義で定義した出力 DB のレコードの項目を，埋め込み変数として使用するために，個々の項目に分割する（@@itemlist 関数を使用）。

(6) 例題テンプレートと生成されたソースプログラム

(6) では，テンプレートを左ページに，生成されたソースプログラムを右ページに掲載しているので，ページの左と右でテンプレートと生成結果を対応させて見ることができます。

テンプレート（ファイル編集・帳票出力）1 ／ 10

```
000101 @@lang COBOL                                                          }
000102    UOC_BEGIN      = "**USER-S @uocname"                               } (1)
000103    UOC_END        = "**USER-E @uocname";                              }
000104 @@*定義部
000105  @@interface                                                          }
000106    @プログラムＩＤ =                                                  } (2)
000107     {COMMENT = "プログラムＩＤを英字８文字で入力" };                  }
000108  @@interface                                                          }
000109    @入力ファイル =                                                    }
000110     {ATTR           = FILE,                                           }
000111      COMMENT        = "入力必須、ファイルを指定する",                 }
000112      IO             = IN,                                             }
000113      ファイル名     = {ATTR = FILE_NAME},                             }
000114      レコード名     = {ATTR = RECORD_NAME},                           }
000115      編成           = {ATTR = ORGANIZATION},                          } (3)
000116      外部装置名     = {COMMENT = "外部装置名"},                       }
000117      キー           = {COMMENT = "アクセスレコードキー",              }
000118                        ATTR = ITEM,REF = @入力ファイル},              }
000119      アクセスモード = {COMMENT = "順アクセス：'S',キー順：'D'を指定"}, }
000120      接頭語         = {COMMENT = "ファイルのPREFIXを入れる"}           }
000121      };                                                               }
000122  @@*
000123  @@interface                                                          }
000124    @参照ファイル =                                                    }
000125     {ATTR           = FILE,                                           }
000126      COMMENT        = "最大６個までファイル形式を指定する",           }
000127      ARRAY_MAX      = 6,                                              }
000128      IO             = IN,                                             }
000129      ファイル名     = {ATTR = FILE_NAME   },                          }
000130      レコード名     = {ATTR = RECORD_NAME },                          }
000131      編成           = {ATTR = ORGANIZATION},                          } (4)
000132      外部装置名     = {COMMENT = "外部装置名"},                       }
000133      キー           = {COMMENT = "アクセスレコードキー",              }
000134                        ATTR = ITEM,REF=@参照ファイル },               }
000135      キー値項目     = {COMMENT = "参照ファイルを読み込むための項目を指定",}
000136                        ATTR = ITEM,REF=@入力ファイル },               }
000137      アクセスモード = {COMMENT = "順アクセス：'S',キー順：'D'を指定"  }, }
000138      接頭語         = {COMMENT = "ファイルのPREFIXを入れる"}           }
000139      };                                                               }
000140 @@*
000141  @@interface                                                          }
000142    @出力ＤＢ =                                                        }
000143     {ATTR           = DB,                                             }
000144      COMMENT        = "入力必須、ＲＤＢの表名を指定する",             }
000145      レコード名     = {ATTR = RECORD_NAME },                          }
000146      IO             = OUT,                                            } (5)
000147      表名           = {ATTR = TABLE_NAME},                            }
000148      ユーザＩＤ     = {},                                             }
000149      パスワード     = {},                                             }
000150      接頭語         = {COMMENT = "ファイルのPREFIXを入れる"}           }
000151      };                                                               }
```

(1) ユーザ追加処理のユニークな開始コメントおよび終了コメントを出力する。
(2) プログラムＩＤの宣言をする。
(3) 入力ファイルの宣言をする。
(4) 参照ファイルの宣言をする。
(5) 出力ＤＢの宣言をする。

テンプレート（ファイル編集・帳票出力）2 ／ 10

```
000152 @@*
000153  @@interface
000154    @帳票ファイル =
000155     {ATTR          = FILE,
000156      COMMENT       = "帳票出力のための CSV ファイルを指定する",
000157      IO            = OUT,
000158      ファイル名    = {ATTR = FILE_NAME },
000159      レコード名    = {ATTR = RECORD_NAME },
000160      編成          = {ATTR = ORGANIZATION },
000161      外部装置名    = {COMMENT = "外部装置名"},
000162      接頭語        = {COMMENT = "ファイルの PREFIX を入れる"}
000163      };                                                          (6)
000164 @@*
000165    @@if (@入力ファイル  ne  "")
000166      @@set  @入力ファイル指定フラグ = "Y";
000167    @@else
000168      @@msg  "入力ファイルの指定は必須です" ;                       (7)
000169      @@errorexit;
000170    @@end;
000171 @@*
000172    @@if (@出力 DB  ne  "")
000173      @@set  @出力 DB 指定フラグ = "Y";
000174    @@else
000175      @@msg  "出力 DB の指定は必須です" ;                           (8)
000176      @@errorexit;
000177    @@end;
000178 @@*
000179    @@if (@帳票ファイル  ne  "")
000180    @@set  @帳票ファイル指定フラグ = "Y";
000181    @@else
000182    @@msg  "帳票ファイルの指定は必須です" ;                         (9)
000183    @@errorexit;
000184    @@end;
000185 @@*
000186    @@set  @参照ファイルカウンタ = @@count(@参照ファイル);          (10)
000187    @@set  @I = 1 ;
000188    @@set  @参照ファイル指定フラグ = "N";
000189  @@while (@I <= @参照ファイルカウンタ)
000190    @@if (@参照ファイル[@I] ne "")
000191      @@set  @参照ファイル指定フラグ = "Y";
000192      @@break;                                                    (11)
000193    @@end;
000194    @@set  @I = @I + 1;
000195  @@end;
000196 @@*
```

(6)　帳票ファイル（ＣＳＶファイル）の宣言をする。
(7)　入力ﾌｧｲﾙの指定があるかをﾁｪｯｸする。指定がない場合，ﾒｯｾｰｼﾞを出力して生成を中止する。
(8)　出力ＤＢの指定があるかをﾁｪｯｸする。指定がない場合，ﾒｯｾｰｼﾞを出力して生成を中止する。
(9)　帳票ﾌｧｲﾙの指定があるかをﾁｪｯｸする。指定がない場合，生成を中止する。
(10) 参照ﾌｧｲﾙの指定された個数を取得する。
(11) 参照ﾌｧｲﾙ指定ﾌﾗｸﾞを設定する。

テンプレート（ファイル編集・帳票出力）3 ／ 10

```
000197  IDENTIFICATION DIVISION.
000198   PROGRAM-ID. @プログラム ID..                                          (12)
000199 *
000200  ENVIRONMENT DIVISION.
000201   INPUT-OUTPUT SECTION.
000202    FILE-CONTROL.
000203 *
000204      SELECT @入力ファイル[接頭語].-@入力ファイル[ファイル名]
000205             ASSIGN TO @入力ファイル[外部装置名]                          (13)
000206             @@if (@入力ファイル[アクセスモード] eq "S" )
000207               ACCESS MODE IS SEQUENTIAL
000208             @@else                                                       (14)
000209               ACCESS MODE IS DYNAMIC
000210               @@msg  "入力ファイルのアクセスモードは'S'を指定してください。";
000211             @@end;
000212             @@if (@入力ファイル[編成] == 4)
000213               ORGANIZATION IS INDEXED
000214             @@else
000215               @@msg  "入力ファイルが索引順編成ではありません";
000216               @@errorexit;
000217             @@end;
000218             RECORD KEY IS @入力ファイル[接頭語].-@入力ファイル[キー]..
000219             @@if (@参照ファイル指定フラグ eq "Y" )
000220               @@set  @I = 1 ;                                            (15)
000221               @@while (@I <= @参照ファイルカウンタ)
000222 *
000223        SELECT @参照ファイル[@I,接頭語].-@参照ファイル[@I,ファイル名]
000224               ASSIGN TO @参照ファイル[@I, 外部装置名]
000225               @@if (@参照ファイル[@I,アクセスモード] eq "S" )
000226                 ACCESS MODE IS SEQUENTIAL
000227               @@else
000228                 ACCESS MODE IS DYNAMIC
000229               @@end;
000230               @@if (@参照ファイル[@I,編成] == 4)
000231                 ORGANIZATION IS INDEXED                                  (16)
000232               @@else
000233                 @@msg  "参照ファイルが索引順編成ではありません";
000234                 @@errorexit;
000235               @@end;
000236               RECORD KEY IS @参照ファイル[@I,接頭語].-@参照ファイル[@I,キー]..
000237               @@set  @I = @I + 1 ;
000238               @@end;
000239             @@end;
000240 *
```

(12) プログラムＩＤを展開する。
(13) 入力ファイルを展開する。
(14) @@if文でアクセスモードの展開を切り換える。
(15) 参照ファイルを指定数分展開する。
(16) 参照ファイルを展開する。

ソースプログラム（ファイル編集・帳票出力）1／8

```
000101  IDENTIFICATION DIVISION.
000102   PROGRAM-ID. PROG01.                                          (12)
000103 *
000104  ENVIRONMENT DIVISION.
000105   INPUT-OUTPUT SECTION.
000106    FILE-CONTROL.
000107 *
000108      SELECT I1-JUCHUU                                          ┐
000109             ASSIGN TO I1                                       │
000110             ACCESS MODE IS SEQUENTIAL                          ├ (13),(14)
000111             ORGANIZATION IS INDEXED                            │
000112             RECORD KEY IS I1-伝票番号.                         ┘
000113 *
000114      SELECT R1-TSAKI                                           ┐
000115             ASSIGN TO R1                                       │
000116             ACCESS MODE IS DYNAMIC                             │
000117             ORGANIZATION IS INDEXED                            │
000118             RECORD KEY IS R1-得意先コード.                     │
000119 *                                                              ├ (15),(16)
000120      SELECT R2-SHOUHIN                                         │
000121             ASSIGN TO R2                                       │
000122             ACCESS MODE IS DYNAMIC                             │
000123             ORGANIZATION IS INDEXED                            │
000124             RECORD KEY IS R2-商品コード.                       ┘
000125 *
```

テンプレート（ファイル編集・帳票出力）4 ／ 10

```
000241      SELECT @帳票ファイル[接頭語].-@帳票ファイル[ファイル名]
000242             ASSIGN TO @帳票ファイル[外部装置名]
000243             ACCESS MODE IS SEQUENTIAL
000244             @@if (@帳票ファイル[編成] == 6)
000245               ORGANIZATION IS CSV.                                  (17)
000246             @@else
000247               @@msg  "帳票ファイルが CSV 形式ではありません";
000248               @@errorexit;
000249             @@end;
000250 *
000251  DATA DIVISION.
000252   FILE SECTION.
000253 *
000254    FD @入力ファイル[接頭語].-@入力ファイル[ファイル名]..                 (18)
000255       @@expand @入力ファイル[レコード名]
000256                PREFIX = "@入力ファイル[接頭語].-";
000257       @@if (@参照ファイル指定フラグ eq "Y" )
000258         @@set  @I = 1 ;
000259         @@while (@I <= @参照ファイルカウンタ)
000260 *
000261      FD @参照ファイル[@I,接頭語].-@参照ファイル[@I,ファイル名]..          (19)
000262         @@expand @参照ファイル[@I,レコード名]
000263                  PREFIX = "@参照ファイル[@I,接頭語].-";
000264         @@set  @I = @I + 1 ;
000265         @@end;
000266       @@end;
000267 *
000268    FD @帳票ファイル[接頭語].-@帳票ファイル[ファイル名]..                 (20)
000269       @@expand @帳票ファイル[レコード名]
000270                PREFIX = "@帳票ファイル[接頭語].-";
000271 *
```

(17) 帳票ファイル（ＣＳＶファイル）を展開する。
(18) 入力ファイルのレコードを展開する。
(19) 参照ファイルの指定がある場合，参照ファイルのレコードを展開する。展開時，項目に接頭語を付与する。
(20) 帳票ファイル（ＣＳＶファイル）のレコードを展開する。

ソースプログラム（ファイル編集・帳票出力）2／8

```
000126      SELECT L1-JUCHUU
000127             ASSIGN TO L1
000128             ACCESS MODE IS SEQUENTIAL                          (17)
000129             ORGANIZATION IS CSV.
000130 *
000131  DATA DIVISION.
000132   FILE SECTION.
000133 *
000134    FD I1-JUCHUU.
000135    01  I1-受注レコード.
000136     02  I1-伝票番号              PIC X(6).
000137     02  I1-得意先コード          PIC X(5).
000138     02  I1-受注日付              PIC X(6).
000139     02  I1-納期                  PIC X(6).                    (18)
000140     02  I1-納入年月日            PIC X(6).
000141     02  I1-商品コード            PIC X(4).
000142     02  I1-受注数量              PIC 9(3).
000143     02  I1-受注単価              PIC 9(4).
000144 *
000145    FD R1-TSAKI.
000146    01  R1-得意先レコード.
000147     02  R1-得意先コード          PIC X(5).
000148     02  R1-得意先名              PIC N(10).
000149     02  R1-得意先電話番号        PIC X(12).
000150 *
000151    FD R2-SHOUHIN.
000152    01  R2-商品レコード.                                         (19)
000153     02  R2-商品コード            PIC X(4).
000154     02  R2-商品名                PIC N(6).
000155     02  R2-色                    PIC N(1).
000156     02  R2-標準卸単価            PIC 9(4).
000157     02  R2-仕入先コード          PIC X(5).
000158     02  R2-仕入単価              PIC 9(4).
000159 *
000160    FD L1-JUCHUU.
000161    01  L1-受注伝票.
000162     02  L1-伝票番号              PIC X(6).
000163     02  L1-得意先名              PIC N(10).
000164     02  L1-商品名                PIC N(6).
000165     02  L1-色                    PIC N(1).                    (20)
000166     02  L1-売上数量              PIC 9(4).
000167     02  L1-売上単価              PIC 9(4).
000168     02  L1-売上金額              PIC 9(4).
000169 *
```

テンプレート（ファイル編集・帳票出力）5 ／ 10

```
000272   WORKING-STORAGE SECTION.
000273  *
000274  *     EXEC SQL INCLUDE SQLCA END-EXEC.
000275  *
000276   01  BAT1-OPN-@入力ファイル[接頭語]  PIC X(1)  VALUE ZERO.
000277   01  BAT1-INV-@入力ファイル[接頭語]  PIC X(1)  VALUE ZERO.
000278   01  BAT1-STATUS-@入力ファイル[接頭語]  PIC 9(2)  VALUE ZERO.        (21)
000279   01  BAT1-DUP-@入力ファイル[接頭語]  PIC X(1)  VALUE ZERO.
000280   01  BAT1-LOCK-@入力ファイル[接頭語]  PIC X(1)  VALUE ZERO.
000281       @@if (@参照ファイル指定フラグ eq "Y" )
000282         @@set  @I = 1 ;
000283         @@while (@I <= @参照ファイルカウンタ)
000284  *
000285   01  BAT1-OPN-@参照ファイル[@I,接頭語]  PIC X(1)  VALUE ZERO.
000286   01  BAT1-INV-@参照ファイル[@I,接頭語]  PIC X(1)  VALUE ZERO.        (22)
000287   01  BAT1-STATUS-@参照ファイル[@I,接頭語]  PIC 9(2)  VALUE ZERO.
000288   01  BAT1-DUP-@参照ファイル[@I,接頭語]  PIC X(1)  VALUE ZERO.
000289   01  BAT1-LOCK-@参照ファイル[@I,接頭語]  PIC X(1)  VALUE ZERO.
000290         @@set  @I = @I + 1 ;
000291         @@end;
000292       @@end;
000293   *
000294      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000295     77  ユーザ ID  PIC X(30).
000296     77  パスワード  PIC X(30).
000297       EXEC SQL END DECLARE SECTION END-EXEC.
000298   01  EOF-@入力ファイル[接頭語]  PIC 9(1).
000299  *
000300   @@if (@参照ファイル指定フラグ eq "Y" )
000301         @@set  @I = 1 ;
000302         @@while (@I <= @参照ファイルカウンタ)
000303   01  KEYVAL-@参照ファイル[@I,接頭語]..
000304           @@set @PIC1 =  @@pic(@参照ファイル[@I,キー]);
000305    02  KEYVAL-@参照ファイル[@I,接頭語].-1  @PIC1..
000306   01  KEY-@参照ファイル[@I,接頭語]..                                  (23)
000307            @@set @PIC2 =  @@pic(@参照ファイル[@I,キー値項目]);
000308   02  KEY-@参照ファイル[@I,接頭語].-1  @PIC2..
000309   01  REF-FAST-@参照ファイル[@I,接頭語]  PIC 9(1).
000310         @@set  @I = @I + 1 ;
000311         @@end;
000312   @@end;
000313   01  ERR-RTN-CODE  PIC S9(4) COMP  VALUE +16.
000314   01  RDB-ERR-MSG  PIC X(44)  VALUE
000315                'RDB アクセス時にエラーが発生しました。'
000316        .
000317  *
```

(21) 入力ファイルを処理するためのフラグを展開する。
(22) 参照ファイルの指定がある場合,参照ファイルを処理するためのフラグを展開する。
(23) 参照ファイルのキーワークエリアを展開する。キーとして指定された項目の属性と長さを @@pic 関数で取得し，ワークの領域として展開する。

ソースプログラム（ファイル編集・帳票出力）3／8

```
000170   WORKING-STORAGE SECTION.
000171 *
000172 *    EXEC SQL INCLUDE SQLCA END-EXEC.
000173 *
000174    01  BAT1-OPN-I1  PIC X(1)  VALUE ZERO.
000175    01  BAT1-INV-I1  PIC X(1)  VALUE ZERO.
000176    01  BAT1-STATUS-I1  PIC 9(2)  VALUE ZERO.                    (21)
000177    01  BAT1-DUP-I1  PIC X(1)  VALUE ZERO.
000178    01  BAT1-LOCK-I1  PIC X(1)  VALUE ZERO.
000179 *
000180    01  BAT1-OPN-R1  PIC X(1)  VALUE ZERO.
000181    01  BAT1-INV-R1  PIC X(1)  VALUE ZERO.
000182    01  BAT1-STATUS-R1  PIC 9(2)  VALUE ZERO.
000183    01  BAT1-DUP-R1  PIC X(1)  VALUE ZERO.
000184    01  BAT1-LOCK-R1  PIC X(1)  VALUE ZERO.
000185 *                                                              (22)
000186    01  BAT1-OPN-R2  PIC X(1)  VALUE ZERO.
000187    01  BAT1-INV-R2  PIC X(1)  VALUE ZERO.
000188    01  BAT1-STATUS-R2  PIC 9(2)  VALUE ZERO.
000189    01  BAT1-DUP-R2  PIC X(1)  VALUE ZERO.
000190    01  BAT1-LOCK-R2  PIC X(1)  VALUE ZERO.
000191 *
000192      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000193    77  ユーザ ID    PIC X(30).
000194    77  パスワード  PIC X(30).
000195      EXEC SQL END DECLARE SECTION END-EXEC.
000196    01  EOF-I1  PIC 9(1).
000197    01  KEYVAL-R1.
000198     02  KEYVAL-R1-1  PIC X(5).
000199    01  KEY-R1.                                                  (23)
000200     02  KEY-R1-1  PIC X(5).
000201    01  REF-FAST-R1  PIC 9(1).
000202    01  KEYVAL-R2.
000203     02  KEYVAL-R2-1  PIC X(4).
000204    01  KEY-R2.
000205     02  KEY-R2-1  PIC X(4).
000206    01  REF-FAST-R2  PIC 9(1).
000207    01  ERR-RTN-CODE  PIC S9(4) COMP  VALUE +16.
000208    01  RDB-ERR-MSG  PIC X(44)  VALUE
000209                        'RDB アクセス時にエラーが発生しました。'
000210        .
000211 *
```

テンプレート（ファイル編集・帳票出力）6 ／ 10

```
000318   EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000319  @@expand @出力DB[レコード名] PREFIX = "@出力DB[接頭語].-";          (24)
000320   EXEC SQL END DECLARE SECTION END-EXEC.
000321 *
000322  PROCEDURE DIVISION.
000323 *
000324   PROGRAM-CONTROL SECTION.
000325    PROGRAM-RUN.
000326      PERFORM PROLOGUE
000327      PERFORM MAIN
000328      PERFORM EPILOGUE
000329      EXIT PROGRAM
000330      .
000331    STOP-RUN.
000332      EXEC SQL
000333       WHENEVER SQLERROR
000334       CONTINUE
000335      END-EXEC
000336      EXEC SQL
000337       COMMIT RELEASE                                              (25)
000338      END-EXEC
000339      EXEC SQL
000340       WHENEVER SQLERROR
000341       GO TO ABEND-PROC
000342      END-EXEC
000343      STOP RUN
000344      .
000345 *
000346   PROLOGUE SECTION.
000347      OPEN INPUT @入力ファイル[接頭語].-@入力ファイル[ファイル名]          (26)
000348      @@if (@参照ファイル指定フラグ eq "Y" )
000349        @@set  @I = 1;
000350        @@while (@I <= @参照ファイルカウンタ)
000351      OPEN INPUT @参照ファイル[@I,接頭語].-@参照ファイル[@I,ファイル名]   (27)
000352        @@set  @I = @I + 1 ;
000353        @@end;
000354      @@end;
```

(24) 出力ＤＢの列に対する埋め込み変数を展開する。
(25) ＳＱＬのエラー時に処理をする。
(26) 入力ファイルをオープンする。
(27) 参照ファイルをオープンする。

ソースプログラム（ファイル編集・帳票出力）4／8

```
000212      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000213    01  01-受注レコード.                                          ┐
000214     02  01-伝票番号        PIC X(6).                            │
000215     02  01-得意先コード    PIC X(5).                            │
000216     02  01-受注日付        PIC X(6).                            │
000217     02  01-納期            PIC X(6).                            ├(24)
000218     02  01-納入年月日      PIC X(6).                            │
000219     02  01-商品コード      PIC X(4).                            │
000220     02  01-受注数量        PIC S9(3).                           │
000221     02  01-受注単価        PIC S9(4).                           ┘
000222      EXEC SQL END DECLARE SECTION END-EXEC.
000223 *
000224  PROCEDURE DIVISION.
000225 *
000226   PROGRAM-CONTROL SECTION.
000227      PERFORM PROLOGUE
000228      PERFORM MAIN
000229      PERFORM EPILOGUE
000230      EXIT PROGRAM
000231      .
000232    STOP-RUN.
000233      EXEC SQL                                                   ┐
000234       WHENEVER SQLERROR                                         │
000235       CONTINUE                                                  │
000236      END-EXEC                                                   │
000237      EXEC SQL                                                   │
000238       COMMIT RELEASE                                           ├(25)
000239      END-EXEC                                                   │
000240      EXEC SQL                                                   │
000241       WHENEVER SQLERROR                                         │
000242       GO TO ABEND-PROC                                          │
000243      END-EXEC                                                   ┘
000244      STOP RUN
000245      .
000246 *
000247   PROLOGUE SECTION.
000248      OPEN INPUT I1-JUCHUU                                       ┐(26)
000249      OPEN INPUT R1-TSAKI                                        ├(27)
000250      OPEN INPUT R2-SHOUHIN                                      ┘
```

テンプレート（ファイル編集・帳票出力）7 ／ 10

```
000355      OPEN OUTPUT @帳票ファイル[接頭語].-@帳票ファイル[ファイル名]                (28)
000356      EXEC SQL
000357       WHENEVER SQLERROR
000358       GO TO ABEND-PROC
000359      END-EXEC
000360      MOVE 0 TO EOF-@入力ファイル[接頭語]
000361      MOVE '0' TO BAT1-INV-@入力ファイル[接頭語]
000362      INITIALIZE @入力ファイル[接頭語].-@入力ファイル[レコード名]
000363      @@if (@参照ファイル指定フラグ eq "Y" )
000364        @@set  @I = 1;
000365        @@while (@I <= @参照ファイルカウンタ)
000366          MOVE HIGH-VALUE  TO KEYVAL-@参照ファイル[@I,接頭語]
000367          MOVE SPACE  TO @参照ファイル[@I,接頭語].-@参照ファイル[@I,キー]      (29)
000368          MOVE 0 TO REF-FAST-@参照ファイル[@I,接頭語]
000369          MOVE '1' TO BAT1-INV-@参照ファイル[@I,接頭語]
000370          INITIALIZE @参照ファイル[@I,接頭語].-@参照ファイル[@I,レコード名]
000371        @@set  @I = @I + 1 ;
000372        @@end;
000373      @@end;
000374      MOVE '@出力DB[ユーザID]' TO   ユーザID                                  (30)
000375      MOVE '@出力DB[パスワード]' TO   パスワード
000376      EXEC SQL
000377       CONNECT :ユーザID IDENTIFIED BY :パスワード
000378      END-EXEC
000379      EXEC SQL
000380       LOCK TABLE "@出力DB[表名]"                                             (31)
000381         IN ROW EXCLUSIVE MODE
000382      END-EXEC
000383       .
000384 *
```

(28) 帳票ファイル（ＣＳＶファイル）をオープンする。
(29) 参照ファイルを読み込むためのキーを初期設定する。
(30) 出力ＤＢのユーザＩＤとパスワードを展開する。
(31) 表名を展開する。

ソースプログラム（ファイル編集・帳票出力）5／8

```
000251      OPEN OUTPUT L1-JUCHUU                                          (28)
000252      EXEC SQL
000253       WHENEVER SQLERROR
000254       GO TO ABEND-PROC
000255      END-EXEC
000256      MOVE 0 TO EOF-I1
000257      MOVE '0' TO BAT1-INV-I1
000258      INITIALIZE I1-受注レコード
000259      MOVE HIGH-VALUE  TO KEYVAL-R1
000260      MOVE SPACE  TO R1-得意先コード
000261      MOVE 0 TO REF-FAST-R1
000262      MOVE '1' TO BAT1-INV-R1
000263      INITIALIZE R1-得意先レコード                                    (29)
000264      MOVE HIGH-VALUE  TO KEYVAL-R2
000265      MOVE SPACE  TO R2-商品コード
000266      MOVE 0 TO REF-FAST-R2
000267      MOVE '1' TO BAT1-INV-R2
000268      INITIALIZE R2-商品レコード
000269      MOVE 'WAKI' TO ユーザ ID
000270      MOVE 'WAKI' TO パスワード
000271      EXEC SQL                                                       (30)
000272       CONNECT :ユーザ ID IDENTIFIED BY :パスワード
000273      END-EXEC
000274      EXEC SQL
000275       LOCK TABLE "JYUTYUU"                                          (31)
000276         IN ROW EXCLUSIVE MODE
000277      END-EXEC
000278      .
000279 *
```

テンプレート（ファイル編集・帳票出力）8 ／ 10

```
000385   EPILOGUE SECTION.                                                    (32)
000386      CLOSE @入力ファイル[接頭語].-@入力ファイル[ファイル名]
000387      @@if (@参照ファイル指定フラグ eq "Y" )
000388        @@set  @I = 1;
000389        @@while (@I <= @参照ファイルカウンタ)                             (33)
000390      CLOSE @参照ファイル[@I,接頭語].-@参照ファイル[@I,ファイル名]
000391        @@set  @I = @I + 1 ;
000392        @@end;
000393      @@end;
000394      CLOSE @帳票ファイル[接頭語].-@帳票ファイル[ファイル名]                 (34)
000395      .
000396 *
000397   MAIN SECTION.
000398      PERFORM READ-FILE-@入力ファイル[接頭語]                            (35)
000399      PERFORM  WITH TEST BEFORE
000400       UNTIL EOF-@入力ファイル[接頭語] = 1
000401       PERFORM MAIN-LOOP
000402      END-PERFORM
000403      .
000404 *
000405   MAIN-LOOP SECTION.
000406     PERFORM READ-REF-FILE
000407     PERFORM PROC-@出力 DB[接頭語]
000408     PERFORM PROC-@帳票ファイル[接頭語]
000409     PERFORM WRITE-@出力 DB[接頭語]
000410     PERFORM WRITE-@帳票ファイル[接頭語]
000411     PERFORM READ-FILE-@入力ファイル[接頭語]
000412      .
000413 *
000414   READ-FILE-@入力ファイル[接頭語] SECTION.
000415      READ @入力ファイル[接頭語].-@入力ファイル[ファイル名]
000416       AT END
000417         MOVE 1 TO EOF-@入力ファイル[接頭語]                              (36)
000418         MOVE '1' TO BAT1-INV-@入力ファイル[接頭語]
000419       NOT AT END
000420         CONTINUE
000421      END-READ
000422      .
000423 *
```

(32) 入力ファイルをクローズする。
(33) 参照ファイルをクローズする。
(34) 帳票ファイル(CSV ファイル)をクローズする。
(35) 入力ファイルを入力するセクション名。
(36) 入力ファイルの入力処理。

ソースプログラム（ファイル編集・帳票出力）6／8

```
000280   EPILOGUE SECTION.
000281      CLOSE I1-JUCHUU                                          (32)
000282      CLOSE R1-TSAKI                                           (33)
000283      CLOSE R2-SHOUHIN
000284      CLOSE L1-JUCHUU                                          (34)
000285      .
000286 *
000287   MAIN SECTION.
000288      PERFORM READ-FILE-I1                                     (35)
000289      PERFORM  WITH TEST BEFORE
000290       UNTIL EOF-I1 = 1
000291       PERFORM MAIN-LOOP
000292      END-PERFORM
000293      .
000294 *
000295   MAIN-LOOP SECTION.
000296      PERFORM READ-REF-FILE
000297      PERFORM PROC-O1
000298      PERFORM PROC-L1
000299      PERFORM WRITE-O1
000300      PERFORM WRITE-L1
000301      PERFORM READ-FILE-I1
000302      .
000303 *
000304   READ-FILE-I1 SECTION.
000305      READ I1-JUCHUU
000306       AT END
000307        MOVE 1 TO EOF-I1                                       (36)
000308        MOVE '1' TO BAT1-INV-I1
000309       NOT AT END
000310        CONTINUE
000311      END-READ
000312      .
000313 *
```

テンプレート（ファイル編集・帳票出力）9／10

```
000424   READ-REF-FILE SECTION.
000425      @@if (@参照ファイル指定フラグ eq "Y" )
000426        @@set  @I = 1;
000427        @@while (@I <= @参照ファイルカウンタ)
000428      MOVE @入力ファイル[接頭語].-@参照ファイル[@I,キー値項目]
000429           TO KEY-@参照ファイル[@I,接頭語].-1
000430      IF (KEY-@参照ファイル[@I,接頭語]
000431          NOT = KEYVAL-@参照ファイル[@I,接頭語]
000432          OR REF-FAST-@参照ファイル[@I,接頭語] = 0)
000433       MOVE 1 TO REF-FAST-@参照ファイル[@I,接頭語]
000434       MOVE KEY-@参照ファイル[@I,接頭語]
000435            TO KEYVAL-@参照ファイル[@I,接頭語]
000436       MOVE @入力ファイル[接頭語].-@参照ファイル[@I,キー値項目]
000437            TO @参照ファイル[@I,接頭語].-@参照ファイル[@I,キー]
000438       READ @参照ファイル[@I,接頭語].-@参照ファイル[@I,ファイル名]
000439        INVALID KEY
000440         MOVE '1' TO BAT1-INV-@参照ファイル[@I,接頭語]
000441         INITIALIZE @参照ファイル[@I,接頭語].-@参照ファイル[@I,レコード名]
000442         MOVE @入力ファイル[接頭語].-@参照ファイル[@I,キー値項目]
000443              TO @参照ファイル[@I,接頭語].-@参照ファイル[@I,キー]
000444        NOT INVALID KEY
000445         MOVE '0' TO BAT1-INV-@参照ファイル[@I,接頭語]
000446       END-READ
000447      END-IF
000448        @@set  @I = @I + 1 ;
000449         @@end;
000450       @@end;
000451     .
000452 *
000453   ABEND-PROC SECTION.
000454      DISPLAY RDB-ERR-MSG
000455      CALL 'CBLABN' USING ERR-RTN-CODE
000456      STOP RUN
000457      .
```

(37)

(37) 参照ファイルの読み込みを展開する。
・プログラム定義で指定したキー値項目を基に読み込む。
・キー値と退避エリアとして生成した項目の値が同じかどうかをチェックする。

ソースプログラム（ファイル編集・帳票出力）7／8

```
000314   READ-REF-FILE SECTION.
000315      MOVE I1-得意先コード TO KEY-R1-1
000316      IF (KEY-R1
000317          NOT = KEYVAL-R1
000318          OR REF-FAST-R1 = 0)
000319       MOVE 1 TO REF-FAST-R1
000320       MOVE KEY-R1 TO KEYVAL-R1
000321       MOVE I1-得意先コード TO R1-得意先コード
000322       READ R1-TSAKI
000323        INVALID KEY
000324         MOVE '1' TO BAT1-INV-R1
000325         INITIALIZE R1-得意先レコード
000326         MOVE I1-得意先コード TO R1-得意先コード
000327        NOT INVALID KEY
000328         MOVE '0' TO BAT1-INV-R1
000329       END-READ
000330      END-IF
000331      MOVE I1-商品コード TO KEY-R2-1
000332      IF (KEY-R2
000333          NOT = KEYVAL-R2
000334              OR REF-FAST-R2 = 0)
000335       MOVE 1 TO REF-FAST-R2
000336       MOVE KEY-R2 TO KEYVAL-R2
000337       MOVE I1-商品コード TO R2-商品コード
000338       READ R2-SHOUHIN
000339        INVALID KEY
000340         MOVE '1' TO BAT1-INV-R2
000341         INITIALIZE R2-商品レコード
000342         MOVE I1-商品コード TO R2-商品コード
000343        NOT INVALID KEY
000344         MOVE '0' TO BAT1-INV-R2
000345       END-READ
000346      END-IF
000347      .
000348 *
000349   ABEND-PROC SECTION.
000350      DISPLAY RDB-ERR-MSG
000351      CALL 'CBLABN' USING ERR-RTN-CODE
000352      STOP RUN
000353      .
```

(37)

テンプレート（ファイル編集・帳票出力） 10 ／ 10

```
000458 *
000459   WRITE-@出力 DB[接頭語] SECTION.
000460      @@set @ITEM = @@ITEMLIST(@出力 DB);
000461      @@set @ITEMCOUNT = @@COUNT(@ITEM);                    (38)
000462      EXEC SQL
000463      INSERT INTO "@出力 DB[表名]"
000464      )
000465      VALUES ( : @出力 DB[接頭語].-@ITEM[1]
000466             @@set  @I = 2;
000467             @@while (@I <= @ITEMCOUNT)
000468             , : @出力 DB[接頭語].-@ITEM[@I]
000469             @@set @I = @I + 1;                             (39)
000470             @@end;
000471             )
000472      END-EXEC
000473      INITIALIZE @出力 DB[接頭語].-@出力 DB[レコード名]
000474      .
000475 *
000476   WRITE-@帳票ファイル[接頭語] SECTION.
000477      WRITE  @帳票ファイル[接頭語].-@帳票ファイル[レコード名]    (40)
000478      .
000479 *
000480   PROC-@出力 DB[接頭語] SECTION.
000481      @@uoc "出力 DB 加工処理" ;                             (41)
000482      .
000483 *
000484   PROC-@帳票ファイル[接頭語] SECTION.                       (42)
000485      @@uoc "帳票出力ファイル加工処理" ;
```

(38) @@ITEMLISTを使用し，出力ＤＢのレコード項目を@ITEMに配列として設定する。
　　@@COUNTを使用し，個数を取得する。
(39) 埋め込み変数の個数分項目名を展開する。
(40) 帳票ファイル（CSVファイル）へ出力処理をする。
(41) 出力ＤＢの項目への加工処理を記述する。
(42) 帳票ファイルの出力となる項目の加工処理を記述する。

ソースプログラム（ファイル編集・帳票出力）8／8

```
000354 *
000355   WRITE-O1 SECTION.
000356      EXEC SQL                                                    ┐
000357       INSERT INTO "JYUCHUU"                                      │
000358        VALUES (:O1-伝票番号                                      │
000359               ,:O1-得意先コード                                  │
000360               ,:O1-受注日付                                      │
000361               ,:O1-納期                                          ├(38),(39)
000362               ,:O1-納入年月日                                    │
000363               ,:O1-商品コード                                    │
000364               ,:O1-受注数量                                      │
000365               ,:O1-受注単価                                      │
000366               )                                                  ┘
000367      END-EXEC
000368      INITIALIZE O1-JYUCHUU .
000369 *
000370   WRITE-L1 SECTION.
000371      WRITE  L1-受注伝票                                          (40)
000372      .
000373 *
000374   PROC-O1 SECTION.
000375   **USER-S 出力ＤＢ加工処理                                      ┐
000376     MOVE I1-伝票番号 TO O1-伝票番号                              │
000377     MOVE I1-得意先コード TO O1-得意先コード                      │
000378     MOVE I1-受注日付 TO O1-受付日付                              │
000379     MOVE I1-納期 TO O1-納期                                      ├(41)
000380     MOVE I1-納入年月日 TO O1-納入年月日                          │
000381     MOVE I1-商品コード TO O1-商品コード                          │
000382     MOVE I1-受注数量 TO O1-受注数量                              │
000383     MOVE I1-受注単価 TO O1-受注単価                              ┘
000384   **USER-E 出力ＤＢ加工処理
000385     .
000386 *
000387   PROC-L1 SECTION.                                              ┐
000388   **USER-S 帳票出力ファイル加工処理                             │
000389     MOVE I1-伝票番号 TO L1-伝票番号                             │
000390     MOVE R1-得意先名 TO L1-得意先名                             │
000391     MOVE R2-商品名 TO L1-商品名                                 ├(42)
000392     MOVE R2-色 TO L1-色                                         │
000393     MOVE I1-売上数量 TO L1-売上数量                             │
000394     MOVE I1-売上単価 TO L1-売上単価                             │
000395     COMPUTE  L1-売上金額 = L1-売上数量 * L1-売上単価            │
000396   **USER-E 帳票出力ファイル加工処理                             ┘
000397     .
```

6.1.2 マスタ更新・追加出力の例題

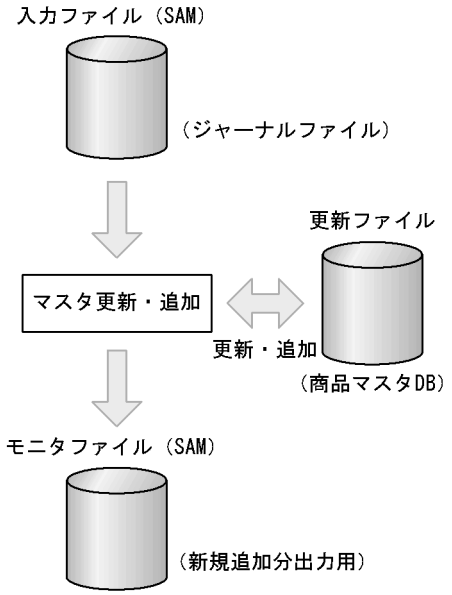
ここで紹介する例題は，RDB にアクセスする処理を部品化し，テンプレートから呼び出して使えるようにしたものです。RDB を部品化することによって，使う RDB の差異に，呼び出す部品を変えることで対応できるため，一つのテンプレートを，より多くのプロ

グラム作成に使うことができます。また，一部の処理を部品化することによって，その分，テンプレートでの記述が少なくて済み，見やすくなります。

(1) 例題テンプレートで作れるプログラム

ジャーナルファイル（SAM ファイル）のデータを基に，キーが存在すれば DB を更新し，存在しない場合は新規に追加します。さらに，新規に追加したものだけをモニタファイル（SAM ファイル）に出力します。

(2) 例題テンプレートの入出力構成



(3) ファイル・DB のレコード形式

例題で使用するファイル・DB のレコード形式を次に示します。

- 入力ファイル（商品ジャーナルファイル）

商品コード	商品名	現在庫量	納入年月日

- 更新ファイル（商品マスタ DB）

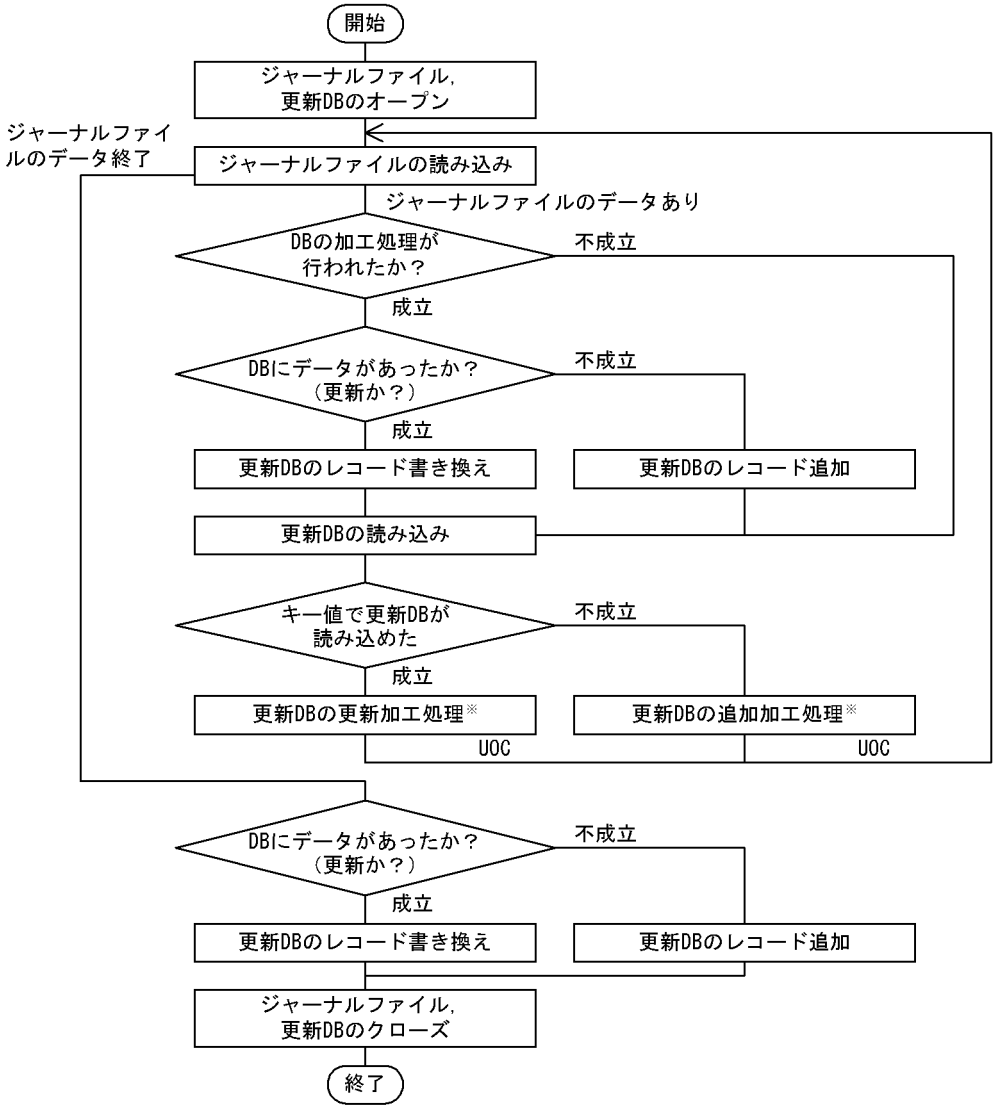
商品コード	商品名	現在庫量	納入年月日

- モニタファイル（商品モニタファイル）

商品コード	商品名	現在庫量	納入年月日

（凡例）□ ：キーとなるデータ

(4) 例題テンプレートの処理概要



注※　プログラム作成者がプログラム定義でUOC（ユーザ追加処理）として定義する。

(5) 部品の展開

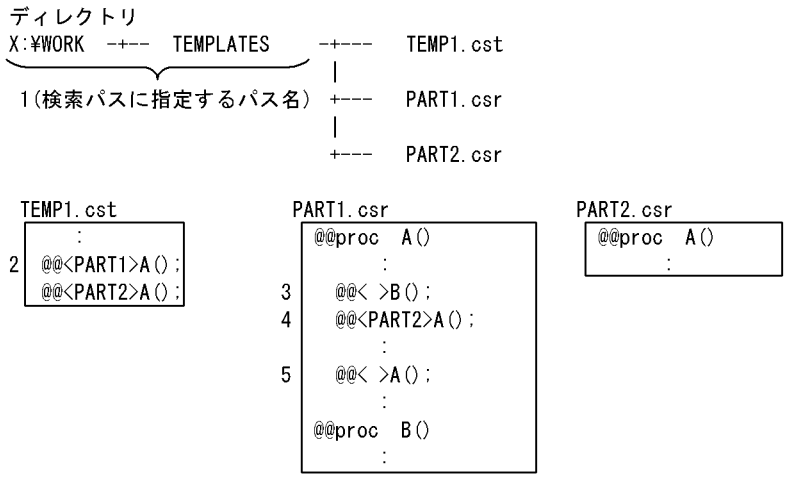
例題を理解するために，部品を呼び出して展開する方法と，部品中に記述された処理を

テンプレート中の任意の位置に埋め込む方法の 2 種類の部品の使い方を説明します。

(a) 部品の呼び出し

一つの部品にはプロシジャを複数指定できます。**@@proc プロシジャ名**を記述して，プロシジャ宣言します。部品は，テンプレート中から **@@ ＜ファイル名＞プロシジャ名**で呼び出します。

使用例



注 cst はテンプレートのサフィックスであり，csr は部品テンプレートのサフィックス。

1. ソースプログラムを生成する前に部品の検索パスを「環境設定」ウィンドウで指定する必要がある。
2. テンプレート TEMP1 からは別ファイルの部品 PART1.csr の @@proc A が呼び出せる。
3. 部品 PART1 の @@proc A は自分のファイルの @@proc B が呼び出せる。＜＞内に何も記述しない場合，自分のファイル名が仮定される。
4. 部品 PART1 からは別ファイルの部品 PART2.csr の @@proc A が呼び出せる（ネストして呼び出すこともできる）。
5. 部品 PART1 の中からもう一度自分自身（部品 PART1）を呼び出せる（リカーシブルに呼び出しができる）。

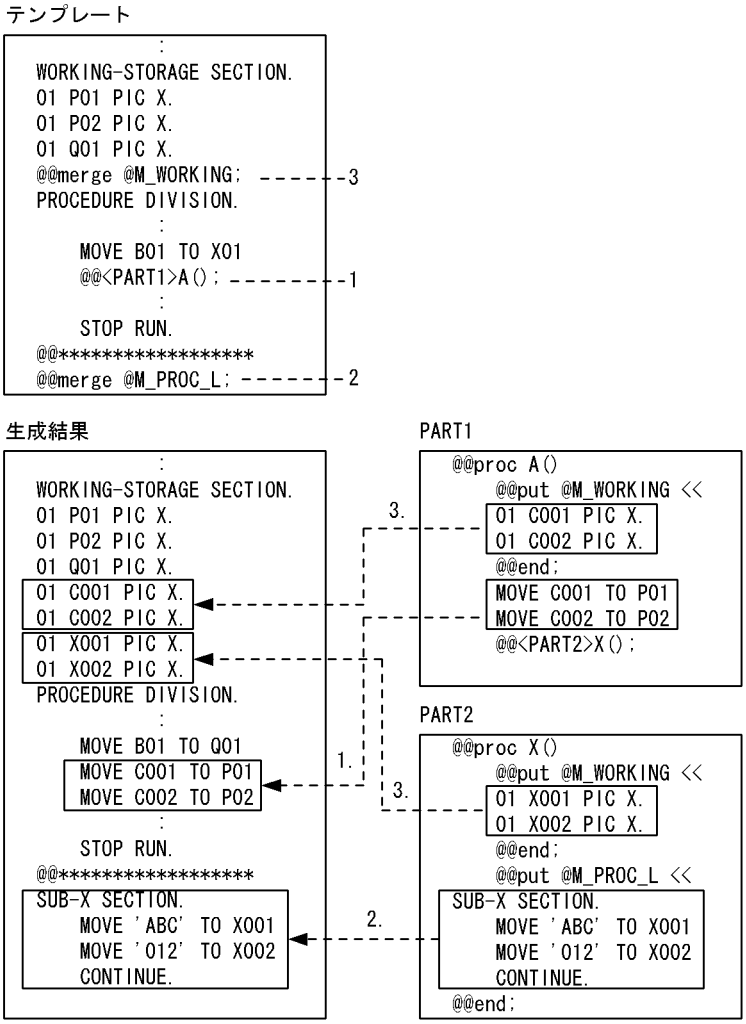
(b) 部品の処理の埋め込み

@@merge 文と @@put 文を使って，部品中に記述された処理をテンプレート中の任意の位置に埋め込むことができます。この機能を使えば，例えば部品中で COBOL の手続きを展開しながら，処理に必要なデータ定義を WORKING-STORAGE SECTION や LINKAGE SECTION の任意の位置に展開することができます。

部品には **@@put 可変記号名　＜＜**を指定し，続けて埋め込みテキストを記述します。テンプレートには，テキストを埋め込みたい個所を指示する **@@merge（可変記号名）**

を記述します。

使用例



1. テンプレートから呼ばれた部品中の処理は，呼ばれた位置に展開される。この例は，「(a) 部品の呼び出し」の例と同じである。
2. @@merge 文で @M_PROC_L の埋め込みを指定しているので，部品から @@put 文で同じ可変記号名を指定してある処理がテンプレートの @@merge 文の位置に展開される。
3. テンプレートから @@merge 文で埋め込み指示された可変記号名と同じ名称の処理が，@@put 文で複数実行された場合は，検索された順にテンプレートの @@merge 文の位置に展開される。

(6) 例題テンプレートの記述のポイント

1. RDB アクセスの部品を使うことによって，見やすいテンプレートにすることができる。さらに，部品を変えることによって DB の差異を反映したソースプログラムを生成できる。
2. @@put 文，@@merge 文を使って，部品の各セクションを呼び出し，展開する。
3. モニタファイルへの出力指示の有無を判断し，展開を制御する（@@if 文を使用)。

(7) 例題テンプレートと生成されたソースプログラム

(7）では，テンプレートを左ページに，生成されたソースプログラムを右ページに掲載し，ページの左と右で生成結果をテンプレートと対応させて見られるようになっています。

テンプレート（マスタ更新・追加出力）１／９

```
@@lang COBOL                                                                    (1)
    UOC_BEGIN       = "**USER-S @uocname"
    UOC_END         = "**USER-E @uocname";
@@interface                                                                     (2)
    @プログラムＩＤ =
     {COMMENT = "プログラムＩＤを英字８文字で入力"          };
  @@interface
    @ジャーナルファイル =                                                       (3)
     {ATTR           = FILE,
      COMMENT        = "データのジャーナルファイル（ＳＡＭ）を指定する",
      IO             = IN,
      ファイル名     = {ATTR = FILE_NAME                    },
      レコード名     = {ATTR = RECORD_NAME                  },
      編成           = {ATTR = ORGANIZATION                 },
      外部装置名     = {COMMENT = "外部装置名"              },
      キー           = {COMMENT = "アクセスレコードキー",
                          ATTR = ITEM,REF = @ジャーナルファイル        },
      アクセスモード = {COMMENT = "順アクセス：'S',キー順：'D'を指定"  },
      接頭語         = {COMMENT = "ファイルのPREFIXを入れる"          }
      };
  @@interface
    @モニタファイル =                                                           (4)
     {ATTR           = FILE,
      COMMENT        = "キーがなかった時モニタファイルとしてデータを出力する",
      IO             = OUT,
      ファイル名     = {ATTR = FILE_NAME                    },
      レコード名     = {ATTR = RECORD_NAME                  },
      編成           = {ATTR = ORGANIZATION                 },
      外部装置名     = {COMMENT = "外部装置名"              },
      キー           = {COMMENT = "アクセスレコードキー",
                          ATTR = ITEM,REF = @モニタファイル    },
      アクセスモード = {COMMENT = "順アクセス：'S',キー順：'D'を指定"  },
      接頭語         = {COMMENT = "ファイルのPREFIXを入れる"          }
      };
  @@interface
    @更新ＤＢ  =                                                                (5)
     {ATTR           = DB,
      COMMENT        = "入力必須、ＲＤＢの表名を指定する",
       IO             = OUT,
      表名           = {ATTR = TABLE_NAME                   },
      レコード名     = {ATTR = RECORD_NAME                  },
      ユーザＩＤ     = {},
      パスワード     = {},
      接頭語         = {COMMENT = "ファイルのPREFIXを入れる"          }
      };
```

(1) ユーザ追加処理のユニークな開始コメントおよび終了コメントを出力する。
(2) プログラムＩＤの宣言をする。
(3) ジャーナルファイルの宣言をする。
(4) モニタファイルの宣言をする。
(5) 更新ＤＢの宣言をする。

テンプレート（マスタ更新・追加出力）2／9

```
000146  @@interface
000147    @キー項目 =                                                          ┐
000148     {ATTR           = ITEM,                                           │
000149      COMMENT        = "更新ＤＢを検索するためキーとなる列名を指定する",  ├ (6)
000150      REF            = @更新ＤＢ                                       │
000151      };                                                               ┘
000152  @@interface
000153    @キー値項目 =                                                        ┐
000154     {ATTR           = ITEM,                                           │
000155      COMMENT        = "更新ＤＢを検索するキー値として対応する"        │
000156                       "ジャーナルファイルのレコード項目を指定",       ├ (7)
000157      REF            = @ジャーナルファイル                             │
000158      };                                                               │
000159  @@*                                                                  ┘
000160    @@if(@モニタファイル  eq  "")                                        ┐
000161      @@set  @モニタファイル指定フラグ  = "N";                           │
000162    @@else                                                             ├ (8)
000163      @@set  @モニタファイル指定フラグ  = "Y";                           │
000164    @@end;                                                             │
000165  @@*                                                                  ┘
000166    IDENTIFICATION DIVISION.
000167     PROGRAM-ID. @プログラムＩＤ..
000168  *
000169    ENVIRONMENT DIVISION.
000170     INPUT-OUTPUT SECTION.
000171      FILE-CONTROL.
000172  *
000173        SELECT @ジャーナルファイル[接頭語].-@ジャーナルファイル[ファイル名]   ┐
000174               ASSIGN TO @ジャーナルファイル[外部装置名]                    │
000175               @@if (@ジャーナルファイル[アクセスモード] eq "S" )           │
000176                 ACCESS MODE IS SEQUENTIAL                    ┐ (9)     ├ (10)
000177               @@else                                         ┘         │
000178                 ACCESS MODE IS DYNAMIC                                  │
000179               @@end;                                                    │
000180               @@if (@ジャーナルファイル[編成] == 4)                       │
000181                 ORGANIZATION IS INDEXED                                 │
000182               @@else                                                    │
000183                 @@msg  "ジャーナルファイルが索引順編成ではありません";   │
000184                 @@errorexit;                                            │
000185               @@end;                                                    │
000186               RECORD KEY IS @ジャーナルファイル[接頭語].-@ジャーナルファイル[キー]..
000187         @@if (@モニタファイル指定フラグ eq "Y" )                        ┘
000188  *
```

(6)　ＤＢの検索キーの定義，プログラム定義でレコード表示を指示する。
(7)　ＤＢの検索のキー値となる項目を定義する。
　　プログラム定義でレコードの表示を指示する。
(8)　モニタファイルの指定の有無をチェックする。
(9)　ジャーナルファイルを展開する。
(10) @@if文でアクセスモードの展開を切り換える。

ソースプログラム（マスタ更新・追加出力）1／8

```
000101  IDENTIFICATION DIVISION.
000102   PROGRAM-ID. BATTYU.
000103 *
000104  ENVIRONMENT DIVISION.                                    }
000105   INPUT-OUTPUT SECTION.                                   }  (9),(10)
000106    FILE-CONTROL.                                          }
000107 *
000108      SELECT I1-JYANAR
000109             ASSIGN TO I1
000110             ACCESS MODE IS SEQUENTIAL
000111             ORGANIZATION IS INDEXED
000112             RECORD KEY IS I1-商品コード.
000113 *
```

テンプレート（マスタ更新・追加出力）3／9

```
000189       SELECT @モニタファイル[接頭語].-@モニタファイル[ファイル名]                    (11)
000190              ASSIGN TO @モニタファイル[外部装置名]
000191              @@if (@モニタファイル[アクセスモード] eq "S" )
000192                ACCESS MODE IS SEQUENTIAL
000193              @@else                                                          (12)
000194                ACCESS MODE IS DYNAMIC
000195              @@end;
000196              @@if (@モニタファイル[編成] == 4)
000197                ORGANIZATION IS INDEXED
000198              @@else
000199                @@msg  "モニタファイルが索引順編成ではありません";
000200                @@errorexit;
000201              @@end;
000202              RECORD KEY IS @モニタファイル[接頭語].-@モニタファイル[キー]..
000203        @@end;
000204 *
000205  DATA DIVISION.
000206   FILE SECTION.
000207 *
000208    FD @ジャーナルファイル[接頭語].-@ジャーナルファイル[ファイル名]  .
000209         @@expand  @ジャーナルファイル[レコード名]                          (13)
000210                   PREFIX = "@ジャーナルファイル[接頭語].-";
000211    @@if(@モニタファイル指定フラグ  eq  "Y")
000212
000213    FD @モニタファイル[接頭語].-@モニタファイル[ファイル名]  .
000214         @@expand  @モニタファイル[レコード名]                              (14)
000215                   PREFIX = "@モニタファイル[接頭語].-";
000216    @@end;
000217 *
```

(11) モニタファイルを展開する。
(12) @@if文でアクセスモードの展開を切り換える。
(13) ジャーナルファイルのレコードを展開する。
(14) モニタファイルの指定がある場合，レコードを展開する。

ソースプログラム（マスタ更新・追加出力）2／8

```
000114      SELECT M1-MONITOR                                    ┐
000115             ASSIGN TO I1                                  │
000116             ACCESS MODE IS SEQUENTIAL                     ├ (11),(12)
000117             ORGANIZATION IS INDEXED                       │
000118             RECORD KEY IS M1-商品コード.                  ┘
000119 *
000120  DATA DIVISION.
000121   FILE SECTION.
000122 *                                                          ┐
000123    FD I1-JYANAR.                                           │
000124    01 I1-商品レコード.                                     │
000125     02 I1-商品コード     PIC X(4).                         ├ (13)
000126     02 I1-商品名         PIC X(6).                         │
000127     02 I1-現在庫量       PIC 9(3).                         │
000128     02 I1-納入年月日     PIC X(6).                         ┘
000129 *                                                          ┐
000130    FD M1-MONITOR.                                          │
000131    01 M1-商品レコード.                                     │
000132     02 M1-商品コード     PIC X(4).                         ├ (14)
000133     02 M1-商品名         PIC X(6).                         │
000134     02 M1-現在庫量       PIC 9(3).                         │
000135     02 M1-納入年月日     PIC X(6).                         ┘
000136 *
```

テンプレート（マスタ更新・追加出力）4／9

```
000218    WORKING-STORAGE SECTION.
000219 *
000220 *      EXEC SQL INCLUDE SQLCA END-EXEC.
000221 *
000222    01  BAT1-OPN-@ジャーナルファイル[接頭語]   PIC X(1)  VALUE ZERO.
000223    01  BAT1-INV-@ジャーナルファイル[接頭語]   PIC X(1)  VALUE ZERO.
000224    01  BAT1-STATUS-@ジャーナルファイル[接頭語]     PIC 9(2)  VALUE ZERO.
000225    01  BAT1-DUP-@ジャーナルファイル[接頭語]   PIC X(1)  VALUE ZERO.
000226    01  BAT1-LOCK-@ジャーナルファイル[接頭語]  PIC X(1)  VALUE ZERO.
000227    01  BAT1-INV-@更新ＤＢ[接頭語]            PIC X(1)  VALUE ZERO.
000228    @@if (@モニタファイル指定フラグ eq "Y" )
000229    01  BAT1-OPN-@モニタファイル[接頭語]       PIC X(1)  VALUE ZERO.
000230    01  BAT1-INV-@モニタファイル[接頭語]       PIC X(1)  VALUE ZERO.
000231    01  BAT1-STATUS-@モニタファイル[接頭語]    PIC 9(2)  VALUE ZERO.
000232    01  BAT1-DUP-@モニタファイル[接頭語]       PIC X(1)  VALUE ZERO.
000233    01  BAT1-LOCK-@モニタファイル[接頭語]      PIC X(1)  VALUE ZERO.
000234    01  BAT--@モニタファイル[接頭語]           PIC X(1)  VALUE ZERO.
000235    @@end;
000236 *
000237    @@merge @CONNECT;
000238    01  BAT--EOF-@ジャーナルファイル[接頭語]  PIC 9(1).
000239    01  BAT--ERR-RTN-CODE  PIC S9(4) COMP  VALUE +16.
000240    01  BAT--ERR-MSG  PIC X(44)  VALUE
000241                        'ＲＤＢアクセス時にエラーが発生しました。'.
000242
000243 *
000244     @@<RDBPARTS>DECLARE(@更新ＤＢ);
000245 *
000246    @@merge @RDBWORK;
```

(15) (16)

(17)

(18)

(19)

(15) 入力ファイルの処理フラグを展開する。
(16) モニタファイルの処理フラグを展開する。
(17) CONNECTの処理に必要な変数を展開する。
(18) ＲＤＢの列名に対する埋め込み変数の展開部品を呼び出す。
(19) ＲＤＢ部品中で使用する変数の宣言をする。

ソースプログラム（マスタ更新・追加出力）3／8

```
000137   WORKING-STORAGE SECTION.
000138 *
000139 *    EXEC SQL INCLUDE SQLCA END-EXEC.
000140 *
000141    01  BAT1-OPN-I1  PIC X(1)  VALUE ZERO.                    ┐
000142    01  BAT1-INV-I1  PIC X(1)  VALUE ZERO.                    │
000143    01  BAT1-STATUS-I1  PIC 9(2)  VALUE ZERO.                 ├ (15)
000144    01  BAT1-DUP-I1  PIC X(1)  VALUE ZERO.                    │
000145    01  BAT1-LOCK-I1  PIC X(1)  VALUE ZERO.                   ┘
000146    01  BAT1-INV-U1  PIC X(1)  VALUE ZERO .                   ┐
000147    01  BAT1-OPN-M1  PIC X(1)  VALUE ZERO.                    │
000148    01  BAT1-INV-M1  PIC X(1)  VALUE ZERO.                    │
000149    01  BAT1-STATUS-M1  PIC 9(2)  VALUE ZERO.                 ├ (16)
000150    01  BAT1-DUP-M1  PIC X(1)  VALUE ZERO.                    │
000151    01  BAT1-LOCK-M1  PIC X(1)  VALUE ZERO.                   │
000152    01  BAT--M1  PIC X(1).                                    ┘
000153 *
000154      EXEC SQL BEGIN DECLARE SECTION END-EXEC.               ┐
000155    77  ユーザＩＤ  PIC X(30).                                 ├ (17),(B-3)※1
000156    77  パスワード  PIC X(30).                                 ┘
000157      EXEC SQL END DECLARE SECTION END-EXEC.
000158    01  BAT--EOF-I1  PIC 9(1).
000159    01  BAT--ERR-RTN-CODE  PIC S9(4) COMP VALUE +16.
000160    01  RDB-ERR-MSG  PIC X(44)  VALUE
000161                        'ＲＤＢアクセス時にエラーが発生しました。'
000162        .
000163 *
000164      EXEC SQL BEGIN DECLARE SECTION END-EXEC.               ┐
000165    01  U1-SHOHIN.                                            │
000166     02  U1-商品コード  PIC X(4).                             ├ (18),(B-1)※2
000167     02  U1-商品名      PIC X(6).                             │
000168     02  U1-現在庫量    PIC X(3).                             │
000169     02  U1-納入年月日  PIC X(6).                             ┘
000170      EXEC SQL END DECLARE SECTION END-EXEC.
000171 *                                                           ┐
000172    01  SQLCODE            PIC S9(9) COMP.                    ├ (19)
000173    01  BAT--CURSOR-SW-U1  PIC X(1).                          │
000174    01  BAT--U1            PIC X(1).                          ┘
000175 *
```

次に示す個所はＤＢアクセス部品を参照のこと。
注※１　CONNECTのための変数宣言を定義する。
注※２　表の埋め込み変数を展開する。

テンプレート（マスタ更新・追加出力）5／9

```
000247   PROCEDURE DIVISION.
000248 *
000249     BAT--PROGRAM-CONTROL SECTION.                                    ┐
000250       PERFORM BAT-PROLOGUE                                           │
000251       PERFORM BAT--MAIN                                              ├(20)
000252       PERFORM BAT--EPILOGUE                                          │
000253       EXIT PROGRAM                                                   ┘
000254       .
000255     BAT--STOP-RUN.                                                   ┐
000256       EXEC SQL                                                       │
000257        WHENEVER SQLERROR                                             │
000258        CONTINUE                                                      │
000259       END-EXEC                                                       │
000260       EXEC SQL                                                       │
000261        COMMIT RELEASE                                                ├(21)
000262       END-EXEC                                                       │
000263       EXEC SQL                                                       │
000264        WHENEVER SQLERROR                                             │
000265        GO TO BAT--ABEND                                              │
000266       END-EXEC                                                       │
000267       STOP RUN                                                       │
000268       .                                                              ┘
000269 *
000270    BAT--PROLOGUE SECTION.
000271       OPEN INPUT @ジャーナルファイル[接頭語].-@ジャーナルファイル[ファイル名]  (22)
000272    @@if (@モニタファイル指定フラグ eq "Y" )
000273      OPEN OUTPUT @モニタファイル[接頭語].-@モニタファイル[ファイル名]        (23)
000274    @@end;
000275       EXEC SQL
000276        WHENEVER SQLERROR
000277        GO TO BAT--ABEND
000278       END-EXEC
000279       MOVE 0 TO BAT--EOF-@ジャーナルファイル[接頭語]
000280       MOVE '0' TO BAT1-INV-@ジャーナルファイル[接頭語]
000281       INITIALIZE @ジャーナルファイル[接頭語].-@ジャーナルファイル[レコード名]
000282       @@merge @RDBINIT;
000283       @@<RDBPARTS>CONNECT(@更新ＤＢ);                                 (24)
000284       @@<RDBPARTS>LOCKTABLE(@更新ＤＢ);                               (25)
000285       @@<RDBPARTS>DECLAREカーソル(@更新ＤＢ,@キー項目,@キー値項目);    (26)
000286      .
000287 *
```

(20) メインルーチン。
(21) ＳＱＬエラー時の処理の宣言をする。
(22) ジャーナルファイルをオープンする。
(23) モニタファイルをオープンする。
(24) 更新ＤＢのユーザＩＤとパスワードを展開する。
(25) ＤＢのLOCKTABLE部品を呼び出す。
(26) ＤＢのカーソル宣言部品を呼び出す。

ソースプログラム（マスタ更新・追加出力）4／8

```
000176  PROCEDURE DIVISION.
000177 *
000178    BAT--PROGRAM-CONTROL SECTION.                                    ┐
000179       PERFORM BAT--PROLOGUE                                         │
000180       PERFORM BAT--MAIN                                             ├ (20)
000181       PERFORM BAT--EPILOGUE                                         │
000182       EXIT PROGRAM                                                  ┘
000183       .
000184     BAT--STOP-RUN.                                                 ┐
000185       EXEC SQL                                                      │
000186        WHENEVER SQLERROR                                            │
000187        CONTINUE                                                     │
000188       END-EXEC                                                      │
000189       EXEC SQL                                                      │
000190        COMMIT RELEASE                                               ├ (21)
000191       END-EXEC                                                      │
000192       EXEC SQL                                                      │
000193        WHENEVER SQLERROR                                            │
000194        GO TO BAT--ABEND                                             │
000195       END-EXEC                                                      │
000196       STOP RUN                                                      ┘
000197       .
000198 *
000199    BAT--PROLOGUE SECTION.                                            (22)
000200       OPEN INPUT I1-JYANAR                                           (23)
000201       OPEN OUTPUT M1-MONITOR
000202       EXEC SQL
000203        WHENEVER SQLERROR
000204        GO TO BAT--ABEND
000205       END-EXEC
000206       MOVE 0 TO BAT--EOF-I1
000207       MOVE '0' TO BAT1-INV-I1
000208       INITIALIZE I1-商品レコード
000209       MOVE 0 TO BAT--U1
000210       MOVE 0 TO BAT--CURSOR-SW-U1
000211       MOVE 'HIRDB' TOユーザＩＤ                                      ┐
000212       MOVE 'HIRDB' TOパスワード                                      │
000213       EXEC SQL                                                      ├ (24),(B-2)※1
000214        CONNECT :ユーザＩＤ IDENTIFIED BY :パスワード                 │
000215       END-EXEC                                                      ┘
000216       EXEC SQL                                                      ┐
000217        LOCK TABLE "SHOUHIN"                                         ├ (25),(B-4)※2
000218          IN ROW EXCLUSIVE MODE                                      │
000219       END-EXEC                                                      ┘
000220       EXEC SQL                                                      ┐
000221        DECLARE CURSORU1 CURSOR FOR                                  │
000222        SELECT *                                                     ├ (26),(B-5)※3
000223        FROM "SHOUHIN"                                               │
000224        WHERE "商品コード" = :WK-商品コード                          │
000225       END-EXEC                                                      ┘
000226       .
000227 *
```

次に示す個所はＤＢアクセス部品を参照のこと。
注※１　部品で使用する変数を初期化する。
注※２　表の排他宣言をする。
注※３　ＤＢのカーソル宣言をする。

テンプレート（マスタ更新・追加出力）6／9

```
000288    BAT--EPILOGUE SECTION.
000289       CLOSE @ジャーナルファイル[接頭語].-@ジャーナルファイル[ファイル名]          (27)
000290       @@if (@モニタファイル指定フラグ eq "Y")
000291       CLOSE @モニタファイル[接頭語].-@モニタファイル[ファイル名]                  (28)
000292       @@end;
000293       .
000294 *
000295    BAT--MAIN SECTION.
000296       PERFORM BAT--READ-FILE-@ジャーナルファイル[接頭語]
000297       PERFORM  WITH TEST BEFORE
000298        UNTIL BAT--EOF-@ジャーナルファイル[接頭語] = 1
000299        PERFORM BAT--MAIN-LOOP
000300       END-PERFORM
000301       PERFORM BAT--OUTPUT-UPD-FILE
000302       .
000303 *
000304    BAT--MAIN-LOOP SECTION.
000305       PERFORM BAT--IN-OUT-UPD-FILE
000306       PERFORM BAT--PROCESS-UPD-MNT
000307       PERFORM BAT--WRITE-@モニタファイル[接頭語]
000308       PERFORM BAT--READ-FILE-@ジャーナルファイル[接頭語]
000309       .
000310 *
000311    BAT--READ-FILE-@ジャーナルファイル[接頭語] SECTION.
000312       READ @ジャーナルファイル[接頭語].-@ジャーナルファイル[ファイル名]
000313        AT END
000314         MOVE 1 TO BAT--EOF-@ジャーナルファイル[接頭語]                          (29)
000315         MOVE '1' TO BAT1-INV-@ジャーナルファイル[接頭語]
000316       NOT AT END
000317         CONTINUE
000318       END-READ
000319       .
000320 *
000321      @@<RDBPARTS>FETCH(@更新ＤＢ);                                          (30)
000322 *
000323      @@<RDBPARTS>OPEN(@更新ＤＢ);                                           (31)
000324 *
```

(27) ジャーナルファイルをクローズする。
(28) モニタファイルをクローズする。
(29) ジャーナルファイル入力処理をする。
(30) ＤＢのフェッチ(列読み込み)部品を呼び出す（フェッチセクションの展開）。
(31) カーソルオープン部品を呼び出す（オープンセクションの展開）。

ソースプログラム（マスタ更新・追加出力）5／8

```
000228   BAT--EPILOGUE SECTION.                                          } (27),(28)
000229      CLOSE I1-JYANAR                                              }
000230      CLOSE M1-MONITOR                                             }
000231      .
000232 *
000233   BAT--MAIN SECTION.
000234      PERFORM  BAT--READ-FILE-I1
000235      PERFORM  WITH TEST BEFORE
000236       UNTIL BAT--EOF-I1 = 1
000237       PERFORM BAT--MAIN-LOOP
000238      END-PERFORM
000239      PERFORM  BAT--OUTPUT-UPD-FILE
000240      .
000241 *
000242   BAT--MAIN-LOOP SECTION.
000243      PERFORM  BAT--IN-OUT-UPD-FILE
000244      PERFORM  BAT--PROCESS-UPD-MNT
000245      PERFORM  BAT--WRITE-M1
000246      PERFORM  BAT--READ-FILE-I1
000247      .
000248 *
000249   BAT--READ-FILE-I1 SECTION.                                       }
000250      READ I1-JYANAR                                                }
000251       AT END                                                       }
000252        MOVE 1 TO BAT--EOF-I1                                       } (29)
000253        MOVE '1' TO BAT1-INV-I1                                     }
000254       NOT AT END                                                   }
000255        CONTINUE                                                    }
000256      END-READ  .                                                   }
000257 *
000258   BAT--READ-U1 SECTION .                                           }
000259      EXEC SQL                                                      }
000260       FETCH CURSORU1                                               }
000261         INTO :U1-商品コード,:U1-商品名, :U1-現在庫量,               }
000262              :U1-納入年月日                                         } (30),(B-7)※1
000263      END-EXEC                                                      }
000264      IF SQLCODE = 1403                                             }
000265       PERFORM BAT--NOTFOUND-U1                                     }
000266      END-IF  .                                                     }
000267 *
000268   BAT--CURSOR-OPEN-U1 SECTION.                                     }
000269      IF BAT--CURSOR-SW-U1 = 0                                      }
000270       EXEC SQL                                                     }
000271         OPEN CURSORU1                                              } (31),(B-9)※2
000272       END-EXEC                                                     }
000273       MOVE 1 TO BAT--CURSOR-SW-U1                                  }
000274      END-IF  .                                                     }
000275 *
```

次に示す個所はＤＢアクセス部品を参照のこと。
注※１　列読み込みセクションの展開。
注※２　カーソルオープンセクションの展開。

テンプレート（マスタ更新・追加出力）7 ／ 9

```
000325      @@<RDBPARTS>CLOSE(@更新ＤＢ);                                    (32)
000326 *
000327     @@<RDBPARTS>NOTFOUND(@更新ＤＢ);                                 (33)
000328 *
000329      @@<RDBPARTS>ABEND();                                             (34)
000330 *
000331      @@<RDBPARTS>WRITE(@更新ＤＢ);                                    (35)
000332 *
000333    BAT--WRITE-@モニタファイル[接頭語]  SECTION.                       }
000334       WRITE  @モニタファイル[接頭語].-@モニタファイル[レコード名]      } (36)
000335       INITIALIZE  @モニタファイル[接頭語].-@モニタファイル[レコード名] }
000336       .
000337 *
```

(32) カーソルクローズ部品を呼び出す（クローズセクションの展開）。
(33) カーソルNOT FOUND部品を呼び出す（カーソルNOT FOUNDセクションの展開）。
(34) 異常終了部品を呼び出す（異常終了セクションの展開）。
(35) ＤＢの出力部品を呼び出す（出力セクションの展開）。
(36) モニタファイル出力処理をする。

ソースプログラム（マスタ更新・追加出力）6／8

```
000276   BAT--CURSOR-CLOSE-U1 SECTION.                                  ┐
000277      IF BAT--CURSOR-SW-U1 = 1                                    │
000278       EXEC SQL                                                   │
000279        WHENEVER SQLERROR                                         │
000280        CONTINUE                                                  │
000281       END-EXEC                                                   │
000282       EXEC SQL                                                   │
000283        CLOSE CURSORU1                                            ├ (32),(B-11)※1
000284       END-EXEC                                                   │
000285       EXEC SQL                                                   │
000286        WHENEVER SQLERROR                                         │
000287        GO TO BAT--ABEND                                          │
000288       END-EXEC                                                   │
000289       MOVE 0 TO BAT--CURSOR-SW-U1                               │
000290      END-IF  .                                                   ┘
000291 *
000292   BAT--NOTFOUND-U1 SECTION.                                      ┐
000293      MOVE '1' TO BAT1-INV-U1                                     ├ (33),(B-12)※2
000294      INITIALIZE U1-商品コード U1-商品名 U1-現在庫量               │
000295         U1-納入年月日  .                                         ┘
000296 *
000297   BAT--ABEND SECTION.                                            ┐
000298      DISPLAY BAT--ERR-MSG                                        ├ (34),(B-14)※3
000299      STOP RUN  .                                                 ┘
000300 *
000301   BAT--WRITE-U1 SECTION.                                         ┐
000302      EXEC SQL                                                    │
000303       INSERT INTO "SHOUHIN"                                      │
000304        VALUES (:U1-商品コード,:U1-商品名, :U1-現在庫量,          ├ (35),(B-15)※4,
000305           : U1-納入年月日)                                        │ (B-16)※5
000306      END-EXEC                                                    │
000307      INITIALIZE U1-商品コード U1-商品名 U1-現在庫量               │
000308         U1-納入年月日  .                                         ┘
000309 *
000310   BAT--WRITE-M1 SECTION.                                         ┐
000311      WRITE M1-商品レコード                                       ├ (36)
000312      INITIALIZE M1-商品レコード  .                               ┘
000313 *
```

次に示す個所はＤＢアクセス部品を参照のこと。
注※１　カーソルクローズセクションを展開する。
注※２　列が見つからなかった場合に処理する。
注※３　異常終了セクションを展開する。
注※４　ＤＢへの出力処理をする。
注※５　埋め込み変数の項目として展開する。

テンプレート（マスタ更新・追加出力）8／9

```
000338    @@<RDBPARTS>UPDATE(@更新ＤＢ);                                  (37)
000339 *
000340    BAT--IN-OUT-UPD-FILE SECTION.
000341       IF BAT--@更新ＤＢ[接頭語] = 1
000342        EVALUATE TRUE
000343         WHEN BAT1-INV-@更新ＤＢ[接頭語] = '0'
000344          @@merge @DB_REWRITE;                                      (38)
000345          @@merge @DB_CLOSE;                                        (39)
000346         WHEN BAT1-INV-@更新ＤＢ[接頭語] = '1'
000347          @@merge @DB_CLOSE;                                        (39)
000348          @@merge @DB_WRITE;                                        (40)
000349        END-EVALUATE
000350        MOVE 0 TO BAT--@更新ＤＢ[接頭語]
000351       END-IF
000352       @@merge @DB_CLOSE;
000353       MOVE '0' TO BAT1-INV-@更新ＤＢ[接頭語]
000354      @@merge @DB_OPEN;                                             (41)
000355      @@merge @DB_READ; .                                           (42)
000356 *
000357    BAT--OUTPUT-UPD-FILE SECTION.
000358       IF BAT--@更新ＤＢ[接頭語] = 1
000359        EVALUATE TRUE
000360         WHEN BAT1-INV-@更新ＤＢ[接頭語] = '0'
000361          @@merge @DB_REWRITE;                                      (43)
000362         WHEN BAT1-INV-@更新ＤＢ[接頭語] = '1'
000363          @@merge @DB_WRITE;                                        (44)
000364        END-EVALUATE
000365       END-IF  .
000366 *
```

(37) ＤＢの更新部品を呼び出す（ＤＢの更新セクションの展開）。
(38) ＤＢ更新セクションの呼び出し(PERFORM)を展開する。
(39) ＤＢのカーソルCLOSEセクションの呼び出し(PERFORM)を展開する。
(40) ＤＢへの出力セクションを呼び出す(PERFORM)文を生成する。
(41) ＤＢのカーソルOPENセクションの呼び出し(PERFORM)を展開する。
(42) ＤＢの読み込みセクションの呼び出し(PERFORM)を展開する。
(43) ＤＢ更新セクションを呼び出す。
(44) ＤＢ追加セクションを呼び出す。

ソースプログラム（マスタ更新・追加出力）7／8

```
000314   BAT--REWRITE-U1 SECTION.                                          ┐
000315      EXEC SQL                                                       │
000316       UPDATE "SHOUHIN" SET                                          │
000317        "納入年月日" = :U1-納入年月日,                                │
000318        "現在庫量" = :U1-現在庫量,                                    │
000319        "商品名" = :U1-商品名,                                        │
000320        "商品コード" = :U1-商品コード                                 │
000321        WHERE CURRENT OF CURSORU1                                     ├ (37),(B-17)※
000322      END-EXEC                                                       │
000323      MOVE '0' TO BAT1-INV-U1                                        │
000324      PERFORM BAT--READ-U1                                           │
000325      IF BAT1-INV-U1 = '0'                                           │
000326       PERFORM BAT--ABEND                                            │
000327      END-IF                                                         │
000328      INITIALIZE U1-商品コード U1-商品名 U1-現在庫量 U1-納入年月日  .  ┘
000329 *
000330   BAT--IN-OUT-UPD-FILE SECTION.
000331      IF BAT--U1 = 1
000332       EVALUATE TRUE
000333        WHEN BAT1-INV-U1 = '0'
000334         PERFORM BAT--REWRITE-U1                                     (38)
000335         PERFORM BAT--CURSOR-CLOSE-U1                                (39)
000336        WHEN BAT1-INV-U1 = '1'
000337         PERFORM BAT--CURSOR-CLOSE-U1                                (39)
000338         PERFORM BAT--WRITE-U1                                       (40)
000339       END-EVALUATE
000340       MOVE 0 TO BAT--U1
000341      END-IF
000342      PERFORM BAT--CURSOR-CLOSE-U1
000343      MOVE '0' TO BAT1-INV-U1
000344      PERFORM BAT--CURSOR-OPEN-U1                                    (41)
000345      PERFORM BAT--READ-U1  .                                        (42)
000346 *
000347   BAT--OUTPUT-UPD-FILE SECTION.
000348      IF BAT--U1 = 1
000349       EVALUATE TRUE
000350        WHEN BAT1-INV-U1 = '0'
000351         PERFORM BAT--REWRITE-U1                                     (43)
000352        WHEN BAT1-INV-U1 = '1'
000353         PERFORM BAT--WRITE-U1                                       (44)
000354       END-EVALUATE
000355      END-IF  .
000356 *
```

次に示す個所はＤＢアクセス部品を参照のこと。
注※　ＤＢの更新処理をする。

テンプレート（マスタ更新・追加出力）9／9

```
000367    BAT--PROCESS-UPD-MNT SECTION.
000368       IF BAT1-INV-@更新ＤＢ[接頭語] = '1'
000369        PERFORM BAT--PROC-ADD-@更新ＤＢ[接頭語]
000370        MOVE 1 TO BAT--@更新ＤＢ[接頭語]
000371        PERFORM BAT--PROC-UNMATCH-@モニタファイル[接頭語]
000372        MOVE 1 TO BAT--@モニタファイル[接頭語]
000373       ELSE
000374        PERFORM BAT--PROC-UPDATE-@更新ＤＢ[接頭語]
000375        MOVE 1 TO BAT--@更新ＤＢ[接頭語]
000376       END-IF  .
000377 *
000378    BAT--PROC-UPDATE-@更新ＤＢ[接頭語] SECTION.
000379       @@UOC "更新ＤＢ更新加工処理";                                    (45)
000380    .
000381 *
000382    BAT--PROC-ADD-@更新ＤＢ[接頭語] SECTION.
000383       @@UOC "更新ＤＢ追加加工処理";                                    (46)
000384    .
000385 *
000386    BAT--PROC-UNMATCH-@モニタファイル[接頭語] SECTION.
000387       @@UOC "モニタファイル加工処理";                                  (47)
000388    .
```

(45) ＤＢ更新時の加工処理をする。
(46) ＤＢ追加時の加工処理をする。
(47) モニタファイルの加工処理をする。

ソースプログラム（マスタ更新・追加出力）8／8

```
000357   BAT--PROCESS-UPD-MNT SECTION.
000358       IF BAT1-INV-U1 = '1'
000359        PERFORM BAT--PROC-ADD-U1
000360        MOVE 1 TO BAT--U1
000361        PERFORM BAT--PROC-UNMATCH-M1
000362        MOVE 1 TO BAT--M1
000363       ELSE
000364        PERFORM BAT--PROC-UPDATE-U1
000365        MOVE 1 TO BAT--U1
000366       END-IF  .
000367 *
000368   BAT--PROC-UPDATE-U1 SECTION.
000369 **USER-S 更新ＤＢ更新加工処理
000370       MOVE     I1-商品コード   TO   U1-商品コード
000371       MOVE     I1-商品名       TO   U1-商品名
000372       MOVE     I1-現在庫量     TO   U1-現在庫量
000373       MOVE     I1-納入年月日   TO   U1-納入年月日
000374 **USER-E 更新ＤＢ更新加工処理
000375       .
000376 *
000377  BAT--PROC-ADD-U1 SECTION.
000378 **USER-S 更新ＤＢ追加加工処理
000379       MOVE     I1-商品コード   TO   U1-商品コード
000380       MOVE     I1-商品名       TO   U1-商品名
000381       MOVE     I1-現在庫量     TO   U1-現在庫量
000382       MOVE     I1-納入年月日   TO   U1-納入年月日
000383 **USER-E 更新ＤＢ追加加工処理
000384       .
000385 *
000386  BAT--PROC-UNMATCH-M1 SECTION.
000387 **USER-S モニタファイル加工処理
000388       MOVE     I1-商品コード   TO   U1-商品コード
000389       MOVE     I1-商品名       TO   U1-商品名
000390       MOVE     I1-現在庫量     TO   U1-現在庫量
000391       MOVE     I1-納入年月日   TO   U1-納入年月日
000392 **USER-E モニタファイル加工処理
000393       .
```

(45)

(46)

(47)

DB アクセス部品（RDBPARTS）テンプレート（マスタ更新・追加出力）1 ／ 5

```
000101 @@**********************************
000102 @@*  ＤＢのＤＥＣＬＡＲＥ定義生成  *
000103 @@**********************************
000104   @@proc  DECLARE(@DB)                                              ┐
000105    EXEC SQL BEGIN DECLARE SECTION END-EXEC.                         │
000106    @@expand  @DB[レコード名] PREFIX = "@DB[接頭語].-";              ├ (B-1)
000107    EXEC SQL END DECLARE SECTION END-EXEC.                           │
000108    @@put  @RDBWORK <<                                               │
000109       01  SQLCODE         PIC     S9(9)  COMP.                     │
000110       01  BAT--CURSOR-SW-@DB[接頭語] PIC X(1).                     │
000111       01  BAT--@DB[接頭語]           PIC X(1).                     │
000112    @@end;                                                          ┘
000113    @@put  @RDBINIT <<                                               ┐
000114       MOVE  0  TO  BAT--@DB[接頭語]                                 ├ (B-2)
000115       MOVE  0  TO  BAT--CURSOR-SW-@DB[接頭語]                       │
000116    @@end;                                                          ┘
000117 @@**********************************
000118 @@*  ＤＢのＣＯＮＮＥＣＴ生成  *
000119 @@**********************************
000120   @@proc  CONNECT (@DB)
000121   @@put  @CONNECT <<
000122    EXEC SQL BEGIN DECLARE SECTION END-EXEC.                         ┐
000123     77  ユーザＩＤ  PIC  X(30).                                     ├ (B-3)
000124     77  パスワード  PIC  X(30).                                     │
000125    EXEC SQL END DECLARE SECTION END-EXEC.                           ┘
000126   @@end;
000127      MOVE  '@DB[ユーザＩＤ]' TO ユーザＩＤ
000128      MOVE  '@DB[パスワード]' TO パスワード
000129      EXEC SQL
000130       CONNECT :ユーザＩＤ IDENTIFIED BY :パスワード
000131      END-EXEC
000132 @@**************************************
000133 @@*  ＤＢのＬＯＣＫ  ＴＡＢＬＥ生成  *
000134 @@**************************************
000135   @@proc  LOCKTABLE(@DB)                                            ┐
000136    EXEC SQL                                                         ├ (B-4)
000137       LOCK TABLE "@DB[表名]"                                        │
000138         IN ROW EXCLUSIVE MODE                                       │
000139      END-EXEC                                                       ┘
```

(B-1) 表の埋め込み変数を展開する。
(B-2) 部品で使用する変数を初期化する。
(B-3) CONNECTのための変数宣言を定義する。
(B-4) 表の排他宣言をする。

DB アクセス部品（RDBPARTS）テンプレート（マスタ更新・追加出力）2 ／ 5

```
000140 @@******************
000141 @@*  カーソル宣言  *
000142 @@******************
000143   @@proc DECLAREカーソル(@DB,@キー項目,@キー値項目)                    ┐
000144      EXEC SQL                                                        │
000145      DECLARE CURSOR@DB[接頭語] CURSOR FOR                            │
000146      SELECT *                                                        ├ (B-5)
000147      FROM "@DB[表名]"                                                │
000148      WHERE "@キー項目" = :@キー値項目                                │
000149      END-EXEC                                                        ┘
000150 @@****************
000151 @@*  列読み込み  *
000152 @@****************
000153   @@proc FETCH(@DB)                                                  ┐
000154   @@put  @DB_READ <<                                                 ├ (B-6)
000155     PERFORM  BAT--READ-@DB[接頭語]                                   │
000156   @@end;                                                            ┘
000157   BAT--READ-@DB[接頭語] SECTION.                                     ┐
000158     EXEC SQL                                                         │
000159     FETCH                                                          │
000160     CURSOR@DB[接頭語]                                              │
000161     INTO                                                           │
000162     @@set @ITEM = @@itemlist(@DB) ;                                │
000163     @@set @ITEMCOUNT = @@count(@ITEM);                             │
000164       :@DB[接頭語].-@ITEM[1]                                       │
000165     @@set @I = 2;                                                  ├ (B-7)
000166     @@while (@I <= @ITEMCOUNT)                                     │
000167       :@DB[接頭語].-@ITEM[@I]                                      │
000168     @@set @I = @I + 1  ;                                           │
000169     @@end;                                                         │
000170     END-EXEC                                                       │
000171     IF SQLCODE = 1403                                              │
000172     PERFORM BAT--NOTFOUND-@DB[接頭語]                              │
000173     END-IF.                                                        ┘
000174 @@************************
000175 @@*  カーソルオープン  *
000176 @@************************
000177   @@proc OPEN(@DB)                                                   ┐
000178   @@put @DB_OPEN <<                                                  ├ (B-8)
000179     PERFORM  BAT--CURSOR-OPEN-@DB[接頭語]                            │
000180   @@end;                                                            ┘
000181   BAT--CURSOR-OPEN-@DB[接頭語] SECTION.                              ┐
000182      IF BAT--CURSOR-SW-@DB[接頭語] = 0                               │
000183       EXEC SQL                                                      │
000184        OPEN CURSOR@DB[接頭語]                                       ├ (B-9)
000185       END-EXEC                                                      │
000186       MOVE 1 TO BAT--CURSOR-SW-@DB[接頭語]                          │
000187      END-IF.                                                        ┘
```

(B-5) ＤＢのカーソル宣言をする。
(B-6) 列読み込みセクションの呼び出しを展開する。
(B-7) 列読み込みセクションを展開する。
(B-8) カーソルオープンの呼び出しを展開する。
(B-9) カーソルオープンセクションを展開する。

DB アクセス部品（RDBPARTS）テンプレート（マスタ更新・追加出力）3 ／ 5

```
000188 @@**********************
000189 @@*  カーソルクローズ  *
000190 @@**********************
000191   @@proc CLOSE(@DB)                                          ┐
000192   @@put  @DB_CLOSE <<                                        │
000193     PERFORM  BAT--CURSOR-CLOSE-@DB[接頭語]                   ├ (B-10)
000194   @@end;                                                     ┘
000195   BAT--CURSOR-CLOSE-@DB[接頭語] SECTION.                     ┐
000196      IF BAT--CURSOR-SW-@DB[接頭語] = 1                       │
000197       EXEC SQL                                              │
000198        WHENEVER SQLERROR                                    │
000199        CONTINUE                                             │
000200       END-EXEC                                              │
000201       EXEC SQL                                              │
000202        CLOSE CURSOR@DB[接頭語]                              ├ (B-11)
000203       END-EXEC                                              │
000204       EXEC SQL                                              │
000205        WHENEVER SQLERROR                                    │
000206        GO TO BAT--ABEND                                     │
000207       END-EXEC                                              │
000208       MOVE 0 TO BAT--CURSOR-SW-@DB[接頭語]                  │
000209      END-IF.                                                ┘
000210 @@**************************
000211 @@*  ＮＯＴ　ＦＯＵＮＤ  *
000212 @@**************************
000213   @@proc NOTFOUND(@DB)                                       ┐
000214   @@put @DB_NOTFOUND <<                                      │
000215     PERFORM BAT--NOTFOUND-DB[接頭語]                         │
000216   @@end;                                                     │
000217   BAT--NOTFOUND-@DB[接頭語] SECTION.                         ├ (B-12)
000218      MOVE '1' TO BAT1-INV-@DB[接頭語]                        │
000219      @@set  @ITEM = @@itemlist(@DB);                         │
000220      INITIALIZE                                              │
000221        @@foreach  @item (@ITEM)                              │
000222          @DB[接頭語].-@item                                  │
000223        @@end;                                                ┘
000224      .
000225 @@***************
000226 @@*  異常終了  *
000227 @@***************
000228   @@proc ABEND()                                             ┐
000229   @@put  @DB_ABEND <<  (B-13)                                ├ (B-13)
000230     PERFORM  BAT--ABEND                                      │
000231   @@end ;                                                    ┘
000232   BAT--ABEND SECTION.                                        ┐
000233      DISPLAY BAT--ERR-MSG                                    ├ (B-14)
000234      STOP RUN                                                │
000235      .                                                       ┘
```

(B-10) カーソルクローズの呼び出しを展開する。
(B-11) カーソルクローズセクションを展開する。
(B-12) 列が見つからなかった場合に処理する。
(B-13) 異常終了セクション呼び出しを展開する。
(B-14) 異常終了セクションを展開する。

DB アクセス部品（RDBPARTS）テンプレート（マスタ更新・追加出力）4 ／ 5

```
000236 @@**********
000237 @@*  出力  *
000238 @@**********
000239   @@proc WRITE(@DB)
000240     @@put  @DB_WRITE  <<
000241       PERFORM  BAT--WRITE-@DB[接頭語]
000242     @@end;
000243     BAT--WRITE-@DB[接頭語] SECTION.
000244     @@set @ITEM = @@itemlist(@DB);
000245     @@set @ITEMCOUNT = @@count(@ITEM);
000246     EXEC SQL (B-15)
000247       INSERT INTO "@DB[表名]"                                    (B-15)
000248        VALUES (
000249                @@set @ITEMCOUNT = @@count(@ITEM);
000250                  :@DB[接頭語].-@ITEM[1]
000251                @@set @I = 2;       (B-16)                     (B-16)
000252                @@while (@I <= @ITEMCOUNT)
000253                  :@DB[接頭語].-@ITEM[@I]
000254                @@set @I = @I + 1  ;
000255                @@end;
000256                )
000257     END-EXEC
000258     INITIALIZE
000259       @@foreach  @item(@ITEM)
000260         @DB[接頭語].-@item
000261       @@end;
000262   .
```

(B-15) ＤＢへ出力処理をする。
(B-16) 埋め込み変数の項目として展開する。

DB アクセス部品（RDBPARTS）テンプレート（マスタ更新・追加出力）5 ／ 5

```
000263 @@**********
000264 @@*  更新  *
000265 @@**********
000266   @@proc UPDATE(@DB)
000267     @@put  @DB_REWRITE  <<
000268       PERFORM  BAT--REWRITE-@DB[接頭語]
000269     @@end;
000270     BAT--REWRITE-@DB[接頭語] SECTION.
000271     @@set @ITEM = @@itemlist(@DB);
000272     @@set @COLUMITEM = @@columnlist(@DB);
000273     @@set @ITEMCOUNT = @@count(@ITEM);
000274     EXEC SQL
000275       UPDATE  "@DB[表名]" SET     (B-17)
000276               "@COLUMITEM[1]" = :@ITEM[1]
000277               @@set  @I = 2;
000278               @@while (@I <= @ITEMCOUNT)
000279               ,"@COLUMITEM[@I]" = :@ITEM[@I]
000280               @@set @I = @I + 1;
000281               @@end;
000282               WHERE CURRENT OF CURSOR@DB[接頭語]
000283     END-EXEC
000284     MOVE '0' TO BAT1-INV-@DB[接頭語]
000285     PERFORM BAT--READ-@DB[接頭語]
000286     IF BAT1-INV-@DB[接頭語] = '0'
000287       PERFORM BAT--ABEND
000288     END-IF
000289     INITIALIZE
000290       @@foreach  @item (@ITEM)
000291       @DB[接頭語].-@item
000292       @@end;
000293   .
```

(B-17)

(B-17) ＤＢの更新処理をする。

6.2 オンライン処理の例題

「6.2.1 データエントリの例題」では基本的な作りのテンプレートの例題を，「6.2.2 伝票入力・訂正の例題」では少し工夫した作りの例題を紹介します。

6.2.1 データエントリの例題

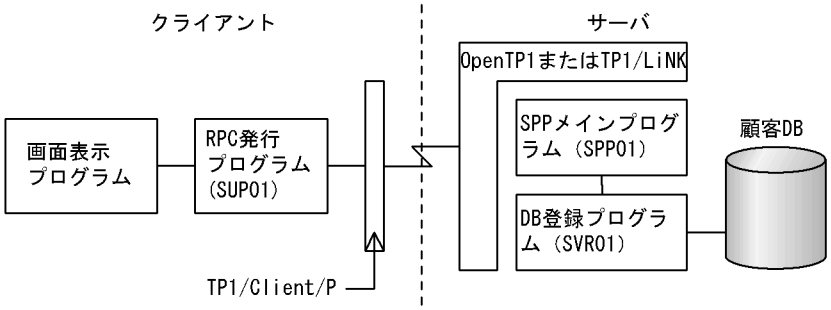
ここで紹介する例題は，OpenTP1 を使った C/S システムの標準的なプログラム構成である SPP と SUP（Service Using Program），そして DB 登録プログラム（SVR01）を作るためのものです。使っている SEWB+/CONSTRUCTION の機能は「6.1.1 ファイル編集・帳票出力の例題」と同様，基本的なものだけです。

(1) 例題テンプレートで作れるプログラム

クライアントのプログラム（SUP）から RPC を発行し，サーバ側のプログラム（SPP）にデータを渡します。SPP は渡されたデータを，DB 登録プログラム SVR01 を使って DB に登録します。SPP01，SUP01，SVR01 のそれぞれに対して，テンプレートを作成します。

なお，この例題では，画面に XMAP3 を使用しています。

(2) 例題テンプレートの入出力構成



注 6.2.1で紹介する例題はSUP01，SUPP01，SVR01。

(3) DB，RPC インタフェースレコードの形式

例題で使用する DB，RPC インタフェースレコードの形式を次に示します。

- 顧客 DB

顧客ID	氏名	役職	TEL	FAX

- RPC インタフェースレコード（SEND 用入力パラメタ）

顧客ID	氏名	役職	TEL	FAX

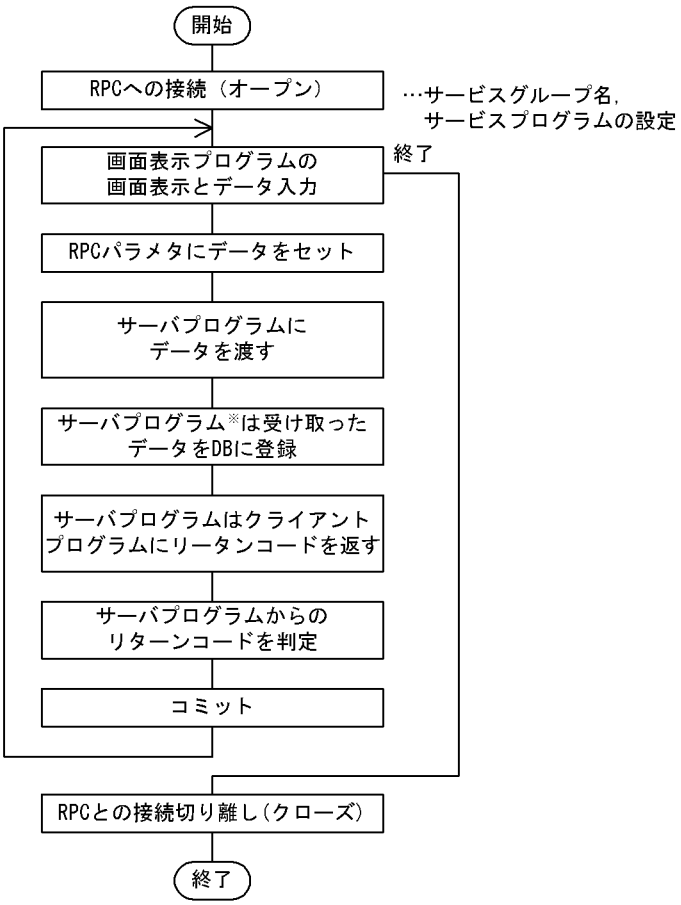
- RPC インタフェースレコード（RECEIVE 用応答領域）

顧客ID	氏名	役職	TEL	FAX

（凡例）□ ：キーとなるデータ

(4) 例題テンプレートの処理概要

サーバプログラムが行う処理は，フロー中で明記しています。それ以外はクライアントプログラムが行う処理です。



注※　サーバ側のSPPは事前に起動されていて，クライアントプログラムからの要求を待っている状態にある。

(5) 例題テンプレートの記述のポイント

1. RPC 発行プログラム用のテンプレートに，SPP へサービスを要求するときに必要な入力パラメタと応答領域（出力パラメタ）を，@@expand 文を使って展開する。
2. RPC 発行プログラム用のテンプレートに，@@reclen 関数を使って RPC のメッセージ長を記述する。
3. DB 登録プログラム用のテンプレートに，@@expand 文の prefix を使って DB のワーク領域を記述する。
4. DB 登録プログラム用のテンプレートに，RPC のメッセージ項目と同一のワーク項目を作成し，MOVE 文を自動生成する処理を記述する。このように，データ項目が同じものに着目し，それを生かすことによって，テンプレートにも項目の加工処理を記述できる。

(6) 例題テンプレートと生成されたソースプログラム

(6) では，テンプレートを左ページに，生成されたソースプログラムを右ページに掲載し，ページの左と右でテンプレートと生成結果を対応させて見られるようになっています。

テンプレート（データエントリ，SPP）1／2

```
000101   @@lang COBOL                                                          )
000102      UOC_BEGIN      = "**USER-S @uocname"                               ) (1)
000103      UOC_END        = "**USER-E @uocname";                              )
000104   @@interface                                                           )
000105     @サービスプログラム名 =                                             ) (2)
000106      {COMMENT = "SPP のサービスプログラム名を入力" };                   )
000107 *                                                                       )
000108 *  サービスメインプログラム @サービスプログラム名                       ) (3)
000109 *                                                                       )
000110   IDENTIFICATION DIVISION.
000111 *
000112    PROGRAM-ID. @サービスプログラム名..
000113 *
000114 * データ領域の設定
000115 *
000116   DATA DIVISION.
000117    WORKING-STORAGE SECTION.
000118 *
000119   01  RPC-ARG1.                                                         )
000120    02 REQ-CODE     PIC X(8) VALUE SPACE.                               )
000121    02 STATUS-CODE  PIC X(5) VALUE SPACE.                               ) (4)
000122    02 FILLER       PIC X(3).                                           )
000123    02 FLAGS        PIC S9(9) COMP.                                     )
000124 *
000125   PROCEDURE DIVISION.
000126 *
000127 * RPC-OPEN(UAP の開始)
000128 *
000129    MOVE 'OPEN' TO REQ-CODE OF RPC-ARG1
000130    MOVE ZERO   TO FLAGS   OF RPC-ARG1
000131    CALL 'CBLDCRPC' USING RPC-ARG1
000132      IF STATUS-CODE OF RPC-ARG1 NOT = '00000'                           )
000133      THEN                                                               )
000135        DISPLAY '@サービスプログラム名.:RPC-OPEN FAILED. CODE = '        ) (5)
000136        STATUS-CODE OF RPC-ARG1                                          )
000137        GO TO PROG-END                                                   )
000138      END-IF.
```

(1) ユーザ追加処理のユニークな開始コメントおよび終了コメントを出力する。
(2) サービスプログラム名の宣言をする。
(3) サービスプログラム名のを展開する。
(4) ＲＰＣのパラメタ。
(5) ＲＰＣのサーバプログラムをオープンする。

ソースプログラム（データエントリ，SPP）1／2

```
000101 *
000102 *  サービスメインプログラム SPP01
000103 *
000104  IDENTIFICATION DIVISION.
000105 *
000106   PROGRAM-ID. SPP01.
000107 *
000108 * データ領域の設定
000109 *
000110  DATA DIVISION.
000111   WORKING-STORAGE SECTION.
000112 *
000113   01  RPC-ARG1.                                                    ┐
000114     02 REQ-CODE     PIC X(8) VALUE SPACE.                         │
000115     02 STATUS-CODE  PIC X(5) VALUE SPACE.                         ├ (4)
000116     02 FILLER       PIC X(3).                                      │
000117     02 FLAGS        PIC S9(9) COMP.                               ┘
000118 *
000119  PROCEDURE DIVISION.
000120 *
000121 * RPC-OPEN(UAP の開始)
000122 *
000123   MOVE 'OPEN' TO REQ-CODE OF RPC-ARG1                             ┐
000124   MOVE ZERO   TO FLAGS   OF RPC-ARG1                              │
000125   CALL 'CBLDCRPC' USING RPC-ARG1                                  │
000126   IF STATUS-CODE OF RPC-ARG1 NOT = '00000'                        ├ (5)
000127   THEN                                                            │
000128     DISPLAY 'SPP01:RPC-OPEN FAILED. CODE = '                      │
000129     STATUS-CODE OF RPC-ARG1                                       │
000130     GO TO PROG-END                                                │
000131   END-IF                                                          ┘
```

テンプレート（データエントリ，SPP）2 ／ 2

```
000139 *
000140 * RPC-MAINLOOP(SPP のサービス開始)
000141 *
000142    DISPLAY '@サービスプログラム名.: MAINLOOP START.'
000143    MOVE 'MAINLOOP' TO REQ-CODE OF RPC-ARG1
000144    MOVE ZERO       TO FLAGS    OF RPC-ARG1
000145    CALL 'CBLDCRCV' USING RPC-ARG1
000146      IF STATUS-CODE OF RPC-ARG1 NOT = '00000'                    } (6)
000147      THEN
000148        DISPLAY '@サービスプログラム名.:RPC-MAINLOOP FAILED. CODE = '
000149        STATUS-CODE OF RPC-ARG1
000150      END-IF .
000151  PROG-END.
000152 *
000153 * RPC-CLOSE(UAP の終了)
000154 *
000155    MOVE 'CLOSE' TO REQ-CODE OF RPC-ARG1                          } (7)
000156    MOVE ZERO    TO FLAGS    OF RPC-ARG1
000157    CALL 'CBLDCRPC' USING RPC-ARG1
000158 *
000159 * 終了処理
000160 *
000161    DISPLAY '@サービスプログラム名.:Good-by!'
000162    STOP RUN.
```

(6) ＲＰＣサーバプログラムを起動する。
(7) ＲＰＣサーバプログラムを停止する。

ソースプログラム（データエントリ，SPP）2／2

```
000132 *
000133 * RPC-MAINLOOP(SPP のサービス開始)
000134 *
000135   DISPLAY 'SPP01: MAINLOOP START.'
000136   MOVE 'MAINLOOP' TO REQ-CODE OF RPC-ARG1
000137   MOVE ZERO       TO FLAGS    OF RPC-ARG1
000138   CALL 'CBLDCRCV' USING RPC-ARG1
000139   IF STATUS-CODE OF RPC-ARG1 NOT = '00000'                    } (6)
000140   THEN
000141     DISPLAY 'SPP01:RPC-MAINLOOP FAILED. CODE = '
000142     STATUS-CODE OF RPC-ARG1
000143   END-IF
000144   PROG-END.
000145 *
000146 * RPC-CLOSE(UAP の終了)
000147 *
000148   MOVE 'CLOSE' TO REQ-CODE OF RPC-ARG1
000149   MOVE ZERO    TO FLAGS    OF RPC-ARG1                        } (7)
000150   CALL 'CBLDCRPC' USING RPC-ARG1
000151 *
000152 * 終了処理
000153 *
000154   DISPLAY 'SPP01:Good-by!'
000155   STOP RUN.
000156
```

テンプレート（データエントリ，SUP）1 ／ 7

```
000101  @@lang COBOL
000102            UOC_BEGIN = "**USER-S @uocname"
000103            UOC_END   = "**USER-E @uocname";
000104  @@interface
000105    @ＲＰＣ受信メッセージ =
000106      {ATTR              = RPC_INPARM,
000107       COMMENT           = "ＲＰＣ受信メッセージ",
000108       IO                = IN,
000109       入力データ        = {ATTR = RECORD_NAME}
000110      };
000111  @@interface
000112    @ＲＰＣ発信メッセージ =
000113      {ATTR              = RPC_REPLY,
000114       COMMENT           = "ＲＰＣ発信メッセージ",
000115       IO                = OUT,
000116       出力データ        = {ATTR = RECORD_NAME }
000117      };
```

(1)
(2)
(3)

(1) ユーザ追加処理のユニークな開始コメントおよび終了コメントを出力する。
(2) RPC受信メッセージの宣言をする。
(3) RPC発信メッセージの宣言をする。

テンプレート（データエントリ，SUP）２／７

```
000118  @@interface
000119    @顧客情報登録画面          =
000120      {ATTR            = XMAP3,
000121       COMMENT         = 'データ入力するＸＭＡＰ画面を指定する。¥n （必須）',
000122       IO              = IN
000123      };
000124  @@interface
000125    @サーバグループ名   = {COMMENT = "サーバグループ名を指定して下さい" };
000126  @@interface
000127    @サーバサービス名   = {COMMENT = "サーバサービス名を指定して下さい" };
000128  @@interface
000129    @ＳＵＰ名           = {COMMENT = "プログラム名を入力して下さい"     };
000130  @@if( @顧客情報登録画面 ne "" )
000131    @@set @Xmap3Com = @@xmap3common(@顧客情報登録画面);
000132    @@if( @Xmap3Com[ 定義対象 ] ne "G1" && @Xmap3Com[ 定義対象 ] ne "CU" )
000133      @@msg "顧客情報登録画面には、GUI(一次)またはCUIを指定してください。";
000134      @@errorexit;
000135    @@end;
000136    @@else
000137      @@msg "顧客情報登録画面には、必ず、画面用マップを指定してください。";
000138      @@errorexit;
```

(4)　(5)　(6)　(7)　(8)

(4) 顧客情報登録画面の宣言をする。
(5) サーバグループ名の宣言をする。
(6) サーバサービス名の宣言をする。
(7) クライアントプログラム名の宣言をする。
(8) 顧客情報登録画面のチェックをする。

テンプレート（データエントリ，SUP）3／7

```
000139 *
000140 *  ＳＵＰクライアントプログラム @ＳＵＰ名
000141 *
000142   IDENTIFICATION DIVISION.
000143 *
000144    PROGRAM-ID. @ＳＵＰ名..
000145 *
000146 * データ領域の設定
000147 *
000148   DATA DIVISION.
000149    WORKING-STORAGE SECTION .
000150 *
000151    01 RPC-ARG1.                                              ┐
000152      02 REQUEST      PIC X(8) VALUE SPACE.                   │
000153      02 STATUS-CODE  PIC X(5) VALUE SPACE.                   │ (9)
000154      02 FILLER       PIC X(3).                               │
000155      02 FLAGS        PIC S9(9) COMP.                         ┘
000156 *
000157    01 RPC-ARG2.                                              ┐
000158      02 REQUEST      PIC X(8) VALUE SPACE.                   │
000159      02 STATUS-CODE  PIC X(5) VALUE SPACE.                   │
000160      02 FILLER       PIC X(3).                               │ (10)
000161      02 DESCRIPTOR   PIC S9(9) COMP VALUE ZERO.              │
000162      02 FLAGS        PIC S9(9) COMP VALUE ZERO.              │
000163      02 S-NAME       PIC X(32) VALUE SPACE.                  │
000164      02 G-NAME       PIC X(32) VALUE SPACE.                  ┘
```

(9) RPCパラメタ1の宣言をする。
(10)RPCパラメタ2の宣言をする。

ソースプログラム（データエントリ，SUP）1／5

```
000101   *
000102   *  ＳＵＰクライアントプログラム SUP01
000103   *
000104     IDENTIFICATION DIVISION.
000105   *
000106      PROGRAM-ID. SUP01.
000107   *
000108   * データ領域の設定
000109   *
000110     DATA DIVISION.
000111      WORKING-STORAGE SECTION .
000112   *
000113      01 RPC-ARG1.                                              )
000114        02 REQUEST      PIC X(8) VALUE SPACE.                   )
000115        02 STATUS-CODE  PIC X(5) VALUE SPACE.                   ) (9)
000116        02 FILLER       PIC X(3).                               )
000117        02 FLAGS        PIC S9(9) COMP.                         )
000118   *
000119      01 RPC-ARG2.                                              )
000120        02 REQUEST      PIC X(8) VALUE SPACE.                   )
000121        02 STATUS-CODE  PIC X(5) VALUE SPACE.                   )
000122        02 FILLER       PIC X(3).                               ) (10)
000123        02 DESCRIPTOR   PIC S9(9) COMP VALUE ZERO.              )
000124        02 FLAGS        PIC S9(9) COMP VALUE ZERO.              )
000125        02 S-NAME       PIC X(32) VALUE SPACE.                  )
000126        02 G-NAME       PIC X(32) VALUE SPACE.                  )
```

テンプレート（データエントリ，SUP）4 ／ 7

```
000165 *
000166    01 RPC-ARG3.                                                          ┐
000167      02 SEND-DATA-LENG  PIC S9(9) COMP.                                  ├(11)
000168       @@expand @ＲＰＣ発信メッセージ[出力データ]  PREFIX = "S-";          ┘
000169 *
000170    01 RPC-ARG4.                                                          ┐
000171      02 RECEIVE-DATA-LENG  PIC S9(9) COMP.                               ├(12)
000172       @@expand @ＲＰＣ受信メッセージ[入力データ]  PREFIX = "R-";         ┘
000173 *
000174    01 ADM-ARG1.                                                          ┐
000175      02 REQUEST      PIC X(8) VALUE SPACE.                              │
000176      02 STATUS-CODE  PIC X(5) VALUE SPACE.                              ├(13)
000177      02 FILLER       PIC X(3).                                          │
000178      02 FLAGS        PIC S9(9) COMP VALUE ZERO.                         │
000179      02 FILLER       PIC X(3).                                          ┘
000180 *
000181    01 TRN-ARG1.                                                          ┐
000182      02 REQUEST      PIC X(8) VALUE SPACE.                              ├(14)
000183      02 STATUS-CODE  PIC X(5) VALUE SPACE.                              ┘
000184 *
000185    01  SCREEN-FLAGS  PIC  S9(2).                                        ┘(15)
000186 *
000187 *                                                                        ┐
000188    COPY  @Xmap3Com[ 入力論理ﾏｯﾌﾟ名 ]..                                   ├(16)
000189    COPY  @Xmap3Com[ 出力論理ﾏｯﾌﾟ名 ]..                                   ┘
000190 *
000191 *
000192    COMMUNICATION SECTION.
000193    CD DSP                                                               ┐
000194      FOR I-O WS                                                         │
000195      MAP NAME            IS  DSP-MAPNAME                                │
000196      SYMBOLIC  TERMINAL  IS  DSP-TERM                                   ├(17)
000197      MAPPING MODE        IS  DSP-MODE                                   │
000198      STATUS  KEY         IS  DSP-STAT.                                  ┘
000199 *
```

(11) サーバ側への出力パラメタを展開する。
(12) サーバ側から入力パラメタを展開する。
(13) ADMパラメタを展開する。
(14) TRMパラメタを展開する。
(15) 画面表示フラグの宣言をする。
(16) 入出力論理マップの取り込み。
(17) ディスプレイに対する通信記述展開。

ソースプログラム（データエントリ，SUP）2／5

```
000127   *
000128      01 RPC-ARG3.                                                    ┐
000129        02 SEND-DATA-LENG  PIC S9(9) COMP.                            │
000130     02  S-顧客ＲＰＣ.                                                │
000131        03  S-顧客ＩＤ                 PIC X(10).                     │
000132        03  S-氏名                     PIC N(10).                     ├ (11)
000133        03  S-役職                     PIC X(10).                     │
000134        03  S-ＴＥＬ                   PIC X(10).                     │
000135        03  S-ＦＡＸ                   PIC X(10).                     ┘
000136   *
000137      01 RPC-ARG4.                                                    ┐
000138        02 RECEIVE-DATA-LENG  PIC S9(9) COMP.                          │
000139     02  R-顧客ＲＰＣ.                                                │
000140        03  R-顧客ＩＤ                 PIC X(10).                     │
000141        03  R-氏名                     PIC N(10).                     ├ (12)
000142        03  R-役職                     PIC X(10).                     │
000143        03  R-ＴＥＬ                   PIC X(10).                     │
000144        03  R-ＦＡＸ                   PIC X(10).                     ┘
000145   *
000146      01 ADM-ARG1.                                                    ┐
000147        02 REQUEST      PIC X(8)       VALUE SPACE.                   │
000148        02 STATUS-CODE  PIC X(5)       VALUE SPACE.                   │
000149        02 FILLER       PIC X(3).                                     ├ (13)
000150        02 FLAGS        PIC S9(9) COMP VALUE ZERO.                     │
000151        02 FILLER       PIC X(3).                                     ┘
000152   *
000153      01 TRN-ARG1.                                                    ┐
000154        02 REQUEST      PIC X(8)   VALUE SPACE.                       ├ (14)
000155        02 STATUS-CODE  PIC X(5)   VALUE SPACE.                       ┘
000156   *
000157      01  SCREEN-FLAGS  PIC  S9(2).                                   ┐ (15)
000158   *                                                                   ┘
000159        COPY  CUSTOMI.                                                ┐
000160        COPY  CUSTOMO.                                                ┘ (16)
000161   *
000162      COMMUNICATION SECTION.                                         ┐
000163     CD DSP                                                           │
000164      FOR I-O WS                                                      │
000165      MAP NAME            IS  DSP-MAPNAME                             ├ (17)
000166      SYMBOLIC  TERMINAL  IS  DSP-TERM                                │
000167      MAPPING MODE        IS  DSP-MODE                                │
000168      STATUS  KEY         IS  DSP-STAT.                               ┘
```

テンプレート（データエントリ，SUP）5／7

```
000200 *
000201   PROCEDURE DIVISION.
000202 *
000203 * RPC-OPEN(ＵＡＰの開始)
000204 *
000205     MOVE 'OPEN' TO REQUEST OF RPC-ARG1
000206     MOVE ZERO   TO FLAGS   OF RPC-ARG1
000207     CALL 'CBLDCRPC' USING RPC-ARG1
000208     IF STATUS-CODE OF RPC-ARG1 NOT = '00000'                    )
000209     THEN                                                        )
000210       DISPLAY '@サーバグループ名.:RPC-OPEN FAILED. CODE = '     ) (18)
000211       STATUS-CODE OF RPC-ARG1                                   )
000212       GO TO PROG-END                                            )
000213      END-IF                                                     )
000214 *
000215 * ADM-COMPLETE(ユーザサーバの開始処理完了の報告)
000216 *
000217     MOVE 'COMPLETE' TO REQUEST OF ADM-ARG1
000218 *   CALL 'CBLDCADM' USING ADM-ARG1
000219 *   IF STATUS-CODE OF ADM-ARG1 NOT = '00000'                  )
000220 *   THEN                                                      )
000221 *     DISPLAY '@サーバグループ名.:RPC-COMPLETE FAILED. CODE = ' ) (19)
000222 *     STATUS-CODE OF ADM-ARG1                                 )
000223 *     GO TO PROG-END                                          )
000224 *   END-IF                                                    )
000225 *
000226 * 画面からのデータ入力                                        )
000227 *                                                             )
000228 *     端末名の設定                                            )
000229       MOVE  'DSP001'  TO  DSP-TERM.                         )
000230 *     出力エリアのクリア                                      )
000231       MOVE  ALL X'1F'  TO   @Xmap3Com[ｼｰﾄ名].G.             )
000232 *     マッピングオプションの設定 (マージで表示)               ) (20)
000233       MOVE   SPACE    TO  DSP-MODE                          )
000234 *                                                           )
000235       MOVE      +0    TO  SCREEN-FLAGS.                     )
000236       PERFORM  UNTIL  SCREEN-FLAGS = 1                      )
000237         PERFORM SCREEN-INPUT                                )
000238       END-PERFORM                                           )
```

(18) RPCのオープン(UAP開始宣言)をする。
(19) ADMの宣言(ユーザサーバ開始処理完了報告)をする。
(20) XMAP3画面の表示をする。

ソースプログラム（データエントリ，SUP）3 ／ 5

```
000169   *
000170     PROCEDURE DIVISION.
000171   *
000172   * RPC-OPEN(ＵＡＰの開始)
000173   *
000174         MOVE 'OPEN' TO REQUEST OF RPC-ARG1
000175         MOVE ZERO   TO FLAGS   OF RPC-ARG1
000176       CALL 'CBLDCRPC' USING RPC-ARG1
000177       IF STATUS-CODE OF RPC-ARG1 NOT = '00000'
000178       THEN
000179         DISPLAY 'SUP01:RPC-OPEN FAILED. CODE = '
000180         STATUS-CODE OF RPC-ARG1
000181         GO TO PROG-END
000182       END-IF
000183   *
000184   * ADM-COMPLETE(ユーザサーバの開始処理完了の報告)
000185   *
000186         MOVE 'COMPLETE' TO REQUEST OF ADM-ARG1
000187       CALL 'CBLDCADM' USING ADM-ARG1
000188       IF STATUS-CODE OF ADM-ARG1 NOT = '00000'
000189       THEN
000190         DISPLAY 'SUP01:RPC-COMPLETE FAILED. CODE = '
000191         STATUS-CODE OF ADM-ARG1
000192        GO TO PROG-END
000193       END-IF
000194   *
000195   * 画面からのデータ入力
000196   *
000197   *    端末名の設定
000198         MOVE  'DSP001'  TO  DSP-TERM.
000199   *    出力エリアのクリア
000200         MOVE  ALL X'1F'  TO  CUSTOMG.
000201   *    マッピングオプションの設定（マージで表示）
000202         MOVE   SPACE    TO  DSP-MODE
000203   *
000204         MOVE      +0    TO  SCREEN-FLAGS.
000205         PERFORM  UNTIL  SCREEN-FLAGS = 1
000206           PERFORM SCREEN-INPUT
000207         END-PERFORM
```

(18)

(19)

(20)

テンプレート（データエントリ，SUP）6／7

```
000239 *
000240 * TRN-BEGIN(トランザクションの開始)
000241 *
000242    MOVE 'BEGIN' TO REQUEST OF TRN-ARG1
000243 *  CALL 'CBLDCTRN' USING TRN-ARG1
000244 *  IF STATUS-CODE OF TRN-ARG1 NOT = '00000'
000245 *  THEN
000246 *    DISPLAY '@サーバグループ名.:TRN-BEGIN FAILED. CODE = '
000247 *    STATUS-CODE OF TRN-ARG1
000248 *    GO TO TRAN-END
000249 *  END-IF .
000250 *
000251 * PRC-CALL(遠隔サービスの要求)
000252 *
000253    MOVE 'CALL'     TO REQUEST OF RPC-ARG2
000254    MOVE '@サーバグループ名'    TO G-NAME  OF RPC-ARG2
000255    MOVE '@サーバサービス名'    TO S-NAME  OF RPC-ARG2
000256    @@uoc  "ＲＰＣ出力データ作成";
000257    @@set  @LEN = @@reclen(@ＲＰＣ発信メッセージ[出力データ]);
000258    MOVE @LEN TO RECEIVE-DATA-LENG OF RPC-ARG4
000259 *  CALL 'CBLDCRPC' USING RPC-ARG2 RPC-ARG3 RPC-ARG4
000260    IF STATUS-CODE OF RPC-ARG2 NOT = '00000'
000261    THEN
000262      DISPLAY '@サーバグループ名.:RPC-CALL RETURN CODE = '
000263      STATUS-CODE OF RPC-ARG2
000264      GO TO TRAN-END
000265    END-IF.
000266    DISPLAY 'SERVICE FUNCTION RETURN = ' RPC-ARG2.
000267  TRAN-END.
000268 *
000269 * TRN-UNCHAINED-COMMIT(非連鎖モードのコミット)
000270 *
000271    MOVE 'U-COMMIT' TO REQUEST OF TRN-ARG1
000272 *  CALL 'CBLDCTRN' USING TRN-ARG1
000273    IF STATUS-CODE OF TRN-ARG1 NOT = '00000'
000274    THEN
000275      DISPLAY '@サーバグループ名.:TRN-UNCHAINED-COMMIT FAILED. CODE = '
000276      STATUS-CODE OF TRN-ARG1
000277    END-IF .
000278  PROG-END.
```

(21) クライアントのトランザクションを開始する。
(22) サーバ側へ出力データを設定する。
(23) サーバ側へRPCを発行する。
(24) トランザクションのコミット。

ソースプログラム（データエントリ，SUP）4 ／ 5

```
000208   *
000209   * TRN-BEGIN(トランザクションの開始)
000210   *
000211        MOVE 'BEGIN' TO REQUEST OF TRN-ARG1
000212      CALL 'CBLDCTRN' USING TRN-ARG1
000213      IF STATUS-CODE OF TRN-ARG1 NOT = '00000'                    ┐
000214      THEN                                                        │
000215        DISPLAY 'SUP01:TRN-BEGIN FAILED. CODE = '                 │
000216        STATUS-CODE OF TRN-ARG1                                   ├ (21)
000217        GO TO TRAN-END                                            │
000218      END-IF .                                                    ┘
000219   *
000220   * PRC-CALL(遠隔サービスの要求)
000221   *
000222        MOVE 'CALL'     TO REQUEST OF RPC-ARG2                    ┐
000223        MOVE 'SUP01'    TO G-NAME  OF RPC-ARG2                    ├ (22)
000224        MOVE 'SVR01'    TO S-NAME  OF RPC-ARG2                    ┘
000225   **USER-S ＲＰＣ出力データ作成
000226        MOVE CUSTOM-顧客ＩＤ-I  TO S-顧客ＩＤ
000227        MOVE CUSTOM-氏名-I      TO S-氏名
000228        MOVE CUSTOM-役職-I      TO S-役職
000229        MOVE CUSTOM-ＴＥＬ-I    TO S-ＴＥＬ
000230        MOVE CUSTOM-ＦＡＸ-I    TO S-ＦＡＸ
000231   **USER-E ＲＰＣ出力データ作成
000232        MOVE 60 TO RECEIVE-DATA-LENG OF RPC-ARG4
000233   *  CALL 'CBLDCRPC' USING RPC-ARG2 RPC-ARG3 RPC-ARG4
000234        IF STATUS-CODE OF RPC-ARG2 NOT = '00000'                  ┐
000235        THEN                                                      │
000236          DISPLAY 'SUP01:RPC-CALL RETURN CODE = '                 │
000237          STATUS-CODE OF RPC-ARG2                                 ├ (23)
000238          GO TO TRAN-END                                          │
000239        END-IF.                                                   │
000240        DISPLAY 'SERVICE FUNCTION RETURN = ' RPC-ARG2.            ┘
000241    TRAN-END.
000242   *
000243   * TRN-UNCHAINED-COMMIT(非連鎖モードのコミット)
000244   *
000245        MOVE 'U-COMMIT' TO REQUEST OF TRN-ARG1
000246   *  CALL 'CBLDCTRN' USING TRN-ARG1
000247        IF STATUS-CODE OF TRN-ARG1 NOT = '00000'                  ┐
000248        THEN                                                      │
000249          DISPLAY 'SUP01:TRN-UNCHAINED-COMMIT FAILED. CODE = '    ├ (24)
000250          STATUS-CODE OF TRN-ARG1                                 │
000251        END-IF .                                                  ┘
000252    PROG-END.
```

テンプレート（データエントリ，SUP）7 ／ 7

```
000279 *
000280 * 画面の繰り返しの終了
000281 *
000282 * RPC-CLOSE(ＵＡＰの終了)
000283 *
000284    MOVE 'CLOSE' TO REQUEST OF RPC-ARG1
000285    MOVE ZERO    TO FLAGS    OF RPC-ARG1
000286 *  CALL 'CBLDCRPC' USING RPC-ARG1
000287    DISPLAY '@サーバグループ名.:SUP PROCESS ENDED'
000288    STOP RUN .
000289 *
000290 * SCREEN-INPUT(画面からのデータ入力)
000291 *
000292  SCREEN-INPUT  SECTION.
000293  SCREEN-INPUT-START.
000294 *
000295 *    マップ名の設定
000296      MOVE  '@Xmap3Com[物理ﾏｯﾌﾟ名]'  TO  DSP-MAPNAME
000297 *    画面の表示と入力
000298      TRANSCEIVE DSP FROM  @Xmap3Com [出力論理ﾏｯﾌﾟ名] INTO @Xmap3Com[入力論理ﾏｯﾌﾟ名]
000299 *    SEND DSP FROM  @Xmap3Com[出力論理ﾏｯﾌﾟ名] WITH EMI
000300 *    RECEIVE DSP FIRST SEGMENT INTO @Xmap3Com[入力論理ﾏｯﾌﾟ名]
000301 *    結果確認
000302      IF  DSP-STAT  =  0
000303          CONTINUE
000304      ELSE
000305          DISABLE   DSP
000306          GO TO     PROG-END
000307      END-IF
000308 *    ボタン・ＰＦキーのチェックと該当処理の呼び出し
000309      EVALUATE @Xmap3Com[ｼｰﾄ名].-INCI
000310      WHEN  'PF02'
000311           MOVE 1 TO SCREEN-FLAGS
000312      WHEN  'PF10'
000313           MOVE 1 TO SCREEN-FLAGS
000314      WHEN  'BREK'
000315           CONTINUE
000316      WHEN  OTHER
000317           CONTINUE
000318      END-EVALUATE
000319 *
000320      CONTINUE.
000321  SCREEN-INPUT-END.
000322      EXIT.
```

(25) (26)

(25) クライアントのRPCをクローズする。
(26) 画面の表示とデータを入力する。

ソースプログラム（データエントリ，SUP）5 ／ 5

```
000253   *
000254   * 画面の繰り返しの終了
000255   *
000256   * RPC-CLOSE(ＵＡＰの終了)
000257   *
000258        MOVE 'CLOSE' TO REQUEST OF RPC-ARG1
000259        MOVE ZERO    TO FLAGS    OF RPC-ARG1
000260   *  CALL 'CBLDCRPC' USING RPC-ARG1
000261        DISPLAY 'SUP01:SUP PROCESS ENDED'
000262        STOP RUN .
000263   *
000264   * SCREEN-INPUT(画面からのデータ入力)
000265   *
000266    SCREEN-INPUT  SECTION.
000267    SCREEN-INPUT-START.
000268   *
000269   *    マップ名の設定
000270        MOVE  'CUSTOMND'  TO  DSP-MAPNAME
000271   *    画面の表示と入力
000272        TRANSCEIVE DSP FROM CUSTOMO INTO CUSTOMI
000273   *    SEND DSP FROM  CUSTOMO WITH EMI
000274   *    RECEIVE DSP FIRST SEGMENT INTO CUSTOMI
000275   *    結果確認
000276        IF  DSP-STAT  =  0
000277          CONTINUE
000278        ELSE
000279          DISABLE   DSP
000280          GO TO     PROG-END
000281        END-IF
000282   *    ボタン・ＰＦキーのチェックと該当処理の呼び出し
000283        EVALUATE  CUSTOM-INCI
000284          WHEN  'PF02'
000285            MOVE 1 TO SCREEN-FLAGS
000286          WHEN  'PF10'
000287            MOVE 1 TO SCREEN-FLAGS
000288          WHEN  'BREK'
000289            CONTINUE
000290          WHEN  OTHER
000291            CONTINUE
000292        END-EVALUATE
000293   *
000294        CONTINUE.
000295    SCREEN-INPUT-END.
000296        EXIT.
```

(25)

(26)

テンプレート（データエントリ，SVR01）1 ／ 4

```
000101 @@lang COBOL                                                          } (1)
000102  UOC_BEGIN      = "**USER-S @uocname"
000103  UOC_END        = "**USER-E @uocname";
000104  @@interface                                                          } (2)
000105    @プログラムＩＤ  =
000106      {COMMENT         = "プログラムＩＤを英字８文字で入力"              };
000107 @@*                                                                   }
000108  @@interface                                                          }
000109    @顧客ＤＢ =                                                        }
000110      {ATTR            = DB,                                           }
000111       COMMENT         = "入力必須、ＲＤＢの表名を指定する",           }
000112       IO              = OUT,                                          } (3)
000113       表名            = {ATTR = TABLE_NAME },                         }
000114       レコード名      = {ATTR = RECORD_NAME},                         }
000115       ユーザＩＤ      = {},                                           }
000116       パスワード      = {},                                           }
000117       接頭語          = {COMMENT = "ファイルのPREFIXを入れる" }       }
000118       };                                                              }
000119 @@*                                                                   }
000120  @@interface                                                          }
000121    @ＲＰＣ受信メッセージ =                                            }
000122      {ATTR            = RPC_INPARM,                                   }
000123       COMMENT         = "ＲＰＣ受信メッセージ",                       } (4)
000124       IO              = IN,                                           }
000125       入力データ      = {ATTR = RECORD_NAME}                          }
000126       };                                                              }
000127 @@*                                                                   }
000128  @@interface                                                          }
000129    @ＲＰＣ発信メッセージ =                                            }
000130      {ATTR            = RPC_REPLY,                                    }
000131       COMMENT         = "ＲＰＣ発信メッセージ",                       } (5)
000132       IO              = OUT,                                          }
000133       出力データ      = {ATTR = RECORD_NAME}                          }
000134       };                                                              }
000135 *                                                                     }
000136   IDENTIFICATION DIVISION.                                            } (6)
000137    PROGRAM-ID. @プログラムＩＤ..
000138 *
000139   DATA DIVISION.
000140 *
000141    WORKING-STORAGE SECTION.
000142 *
000143 *       EXEC SQL INCLUDE SQLCA END-EXEC.
000144 *
000145  EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000146   77  ユーザＩＤ  PIC X(30).
000147   77  パスワード  PIC X(30).
000148  EXEC SQL END DECLARE SECTION END-EXEC.
000149   01   RDB-ERR  PIC X(44)  VALUE
000150        'ＲＤＢアクセス時にエラーが発生しました。' .
000151 *
000152       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000153     @@expand @顧客ＤＢ[レコード名] PREFIX = "@顧客ＤＢ[接頭語].-";
000154       EXEC SQL END DECLARE SECTION END-EXEC.
```

(1) ユーザ追加処理のユニークな開始コメントおよび終了コメントを出力する。 (2) プログラムＩＤの宣言をする。 (3) 顧客ＤＢの宣言をする。 (4) ＰＣ受信メッセージの宣言をする。 (5) ＲＰＣ発信メッセージの宣言をする。 (6) プログラムＩＤを展開する。

ソースプログラム（データエントリ，SVR01）1／4

```
000101  IDENTIFICATION DIVISION.                                          }(6)
000102   PROGRAM-ID. SVR01.                                               }
000103 *
000104  DATA DIVISION.
000105 *
000106   WORKING-STORAGE SECTION.
000107 *
000108 *    EXEC SQL INCLUDE SQLCA END-EXEC.
000109 *
000110      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000111    77  ユーザＩＤ  PIC X(30).
000112    77  パスワード  PIC X(30).
000113      EXEC SQL END DECLARE SECTION END-EXEC.
000114  01   RDB-ERR  PIC X(44)  VALUE
000115                        'ＲＤＢアクセス時にエラーが発生しました。'
000116       .
000117 *
000118      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000119  01  01-顧客ＤＢ.
000120    02  01-顧客ＩＤ  PIC X(10).
000121    02  01-氏名      PIC X(10).
000122    02  01-役職      PIC X(10).
000123    02  01-ＴＥＬ    PIC X(10).
000124    02  01-ＦＡＸ    PIC X(10).
000125      EXEC SQL END DECLARE SECTION END-EXEC.
```

テンプレート（データエントリ，SVR01）2 ／ 4

```
000155 *
000156    LINKAGE SECTION.
000157 *
000158   01  INDATA.                                                          }(7)
000159       @@expand @ＲＰＣ受信メッセージ[入力データ]  PREFIX = "IN-";      }
000160   01  INDATA-LEN  PIC S9(9) COMP.
000161   01  OUTDATA.                                                         }(8)
000162       @@expand @ＲＰＣ発信メッセージ[出力データ]  PREFIX = "OUT-";     }
000163   01  OUTDATA-LEN  PIC S9(9) COMP.
000164 *
000165   PROCEDURE DIVISION
000166             USING INDATA INDATA-LEN OUTDATA OUTDATA-LEN.
000167 *
000168    PROGRAM-CONTROL SECTION.
000169     PROGRAM-RUN.
000170       PERFORM PROLOGUE
000171       PERFORM MAIN
000172       PERFORM EPILOGUE .
000173 *
000174     STOP-RUN-PROC.
000175       EXEC SQL
000176        WHENEVER SQLERROR
000177        CONTINUE
000178       END-EXEC
000179       EXEC SQL
000180        COMMIT RELEASE
000181       END-EXEC
000182       EXEC SQL
000183        WHENEVER SQLERROR
000184        GO TO ABEND-PROC
000185       END-EXEC
000186       STOP RUN .
```

(7) クライアントからの入力パラメタ領域を展開する。
(8) クライアントへの出力パラメタ領域を展開する。

ソースプログラム（データエントリ，SVR01）2／4

```
000126 *
000127   LINKAGE SECTION.
000128 *
000129  01  INDATA.                                                    ┐
000130    02  顧客ＲＰＣ.                                             │
000131      03  IN-顧客ＩＤ  PIC X(10).                               │
000132      03  IN-氏名      PIC X(10).                               ├ (7)
000133      03  IN-役職      PIC X(10).                               │
000134      03  IN-ＴＥＬ    PIC X(10).                               │
000135      03  IN-ＦＡＸ    PIC X(10).                               │
000136  01  INDATA-LEN  PIC S9(9) COMP.                              ┘
000137  01  OUTDATA.                                                   ┐
000138    02  顧客ＲＰＣ.                                             │
000139      03  OUT-顧客ＩＤ  PIC X(10).                              │
000140      03  OUT-氏名      PIC X(10).                               ├ (8)
000141      03  OUT-役職      PIC X(10).                               │
000142      03  OUT-ＴＥＬ    PIC X(10).                               │
000143      03  OUT-ＦＡＸ    PIC X(10).                               │
000144  01  OUTDATA-LEN  PIC S9(9) COMP.                             ┘
000145 *
000146  PROCEDURE DIVISION
000147                       USING INDATA IDATA-LEN OUTDATA OUTDATA-LEN.
000148 *
000149   PROGRAM-CONTROL SECTION.
000150   PROGRAM-RUN.
000151     PERFORM PROLOGUE
000152     PERFORM MAIN
000153     PERFORM EPILOGUE
000154     .
000155   STOP-RUN-PROC.
000156     EXEC SQL
000157       WHENEVER SQLERROR
000158       CONTINUE
000159     END-EXEC
000160     EXEC SQL
000161       COMMIT RELEASE
000162     END-EXEC
000163     EXEC SQL
000164       WHENEVER SQLERROR
000165       GO TO ABEND-PROC
000166     END-EXEC
000167     EXIT PROGRAM
000168     STOP RUN
000169     .
```

テンプレート（データエントリ，SVR01）3 ／ 4

```
000187 *
000188    PROLOGUE SECTION.
000189       EXEC SQL
000190        WHENEVER SQLERROR
000191        GO TO ABEND-PROC
000192       END-EXEC
000193       INITIALIZE ユーザＩＤ パスワード                              ┐
000194       MOVE '@顧客ＤＢ[ユーザＩＤ]' TO ユーザＩＤ                    │
000195       MOVE '@顧客ＤＢ[パスワード]' TO パスワード                    ├ (9)
000196       EXEC SQL                                                      │
000197        CONNECT :ユーザＩＤ IDENTIFIED BY :パスワード                ┘
000198       END-EXEC
000199       .
000200 *
000201    EPILOGUE SECTION.
000202       EXEC SQL
000203       WHENEVER SQLERROR
000204        CONTINUE
000205       END-EXEC
000206       EXEC SQL
000207        WHENEVER SQLERROR
000208        GO TO ABEND-PROC
000209       END-EXEC
000210       .
000211 *
000212    MAIN SECTION.
000213       PERFORM PROC-01
000214       PERFORM WRITE-01
000215       .
000216 *
000217    ABEND-PROC SECTION.
000218      DISPLAY RDB-ERR
000219       EXIT PROGRAM
000220       .
000221 *
```

(9) ユーザＩＤ，パスワードを展開する。

ソースプログラム（データエントリ，SVR01）3 ／ 4

```
000170 *
000171   PROLOGUE SECTION.
000172     EXEC SQL
000173       WHENEVER SQLERROR
000174       GO TO ABEND-PROC
000175     END-EXEC
000176     INITIALIZE ユーザＩＤ パスワード                                    ┐
000177     MOVE 'HIRDB' TO ユーザＩＤ                                         │
000178     MOVE 'HIRDB' TO パスワード                                         ├ (9)
000179     EXEC SQL                                                           │
000180       CONNECT :ユーザＩＤ IDENTIFIED BY :パスワード                    ┘
000181     END-EXEC
000182     .
000183 *
000184   EPILOGUE SECTION.
000185     EXEC SQL
000186       WHENEVER SQLERROR
000187       CONTINUE
000188     END-EXEC
000189     EXEC SQL
000190       WHENEVER SQLERROR
000191       GO TO ABEND-PROC
000192     END-EXEC
000193     .
000194 *
000195   MAIN SECTION.
000196     PERFORM PROC-01
000197     PERFORM WRITE-01
000198     .
000199 *
000200   ABEND-PROC SECTION.
000201     DISPLAY RDB-ERR
000202     EXIT-PROGRAM
000203     .
000204 *
```

テンプレート（データエントリ，SVR01 ）4 ／ 4

```
000222    WRITE-01 SECTION.
000223       @@set  @ITEM = @@itemlist(@顧客ＤＢ);
000224       @@set  @ITEM_CNT = @@count(@ITEM);
000225       EXEC SQL
000226         INSERT INTO @顧客ＤＢ[表名]
000227         VALUES
000228         ( :@顧客ＤＢ[接頭語]-@ITEM[1]
000229         @@set @I = 2;
000230         @@while ( @I <= @ITEM_CNT)
000231           ,:@顧客ＤＢ[接頭語].-@ITEM[@I]
000232           @@set @I = @I + 1;      (9)                    (10)
000233         @@END;
000234         )
000235       END-EXEC
000236       INITIALIZE
000237       @@set @I = 1;
000238       @@while ( @I <= @ITEM_CNT)
000239         @顧客ＤＢ[接頭語].-@ITEM[@I]
000240         @@set @I = @I + 1;
000241       @@END;
000242       .
000243 *
000244    PROC-01 SECTION.
000245       @@UOC "01-出力加工処理";                            (11)
000246       @@UOC "OUT-メッセージ編集";                         (12)
000247       .
000248 *
```

(10) 顧客ＤＢ へ出力する。
(11) 顧客ＤＢの加工処理を記述する。
(12) 送信メッセージの編集を記述する。

ソースプログラム（データエントリ，SVR01）4 ／ 4

```
000205   WRITE-01 SECTION.                                          ┐
000206     EXEC SQL                                                 │
000207       INSERT INTO "KOKYAKU"                                  │
000208       VALUES                                                 │
000209       (:01-顧客ＩＤ                                          │
000210         ,:01-氏名                                            │
000211         , :01-役職                                           │
000212         ,:01-ＴＥＬ                                          │
000213         ,:01-ＦＡＸ                                          │ (10)
000214       )                                                      │
000215     END-EXEC                                                 │
000216     INITIALIZE                                               │
000217       01-顧客ＩＤ                                            │
000218       01-氏名                                                │
000219       01-役職                                                │
000220       01-ＴＥＬ                                              │
000221       01-ＦＡＸ                                              │
000222     .                                                        ┘
000223 *
000224   PROC-01 SECTION.                                           ┐
000225 **USER-S 01ー出力加工処理                                     │
000226     MOVE L1-顧客ＩＤ    TO 01-顧客ＩＤ                      │
000227     MOVE L1-氏名        TO 01-氏名                          │
000228     MOVE L1-役職        TO 01-役職                          │ (11)
000229     MOVE L1-ＴＥＬ      TO 01-ＴＥＬ                        │
000230     MOVE L1-ＦＡＸ      TO 01-ＦＡＸ                        │
000231 **USER-E 01ー出力加工処理                                     ┘
000232 **USER-S OUTーメッセージ編集                                  ┐
000233     MOVE 01-顧客ＩＤ    TO OUT-顧客ＩＤ                     │
000234     MOVE 01-氏名        TO OUT-氏名                         │
000235     MOVE 01-役職        TO OUT-役職                         │ (12)
000236     MOVE 01-ＴＥＬ      TO OUT-ＴＥＬ                       │
000237     MOVE 01-ＦＡＸ      TO OUT-ＦＡＸ                       │
000238 **USER-E OUTーメッセージ編集      .                          ┘
000239     .
000240
```

6.2.2 伝票入力・訂正の例題

ここで紹介する例題は，OpenTP1 の TAM 機能や DB の処理などを部品化することによって，テンプレートでの記述を少なくしたものです。これによって C/S システムのプログラムを，部品の変数を指定するだけで容易に作成できます。

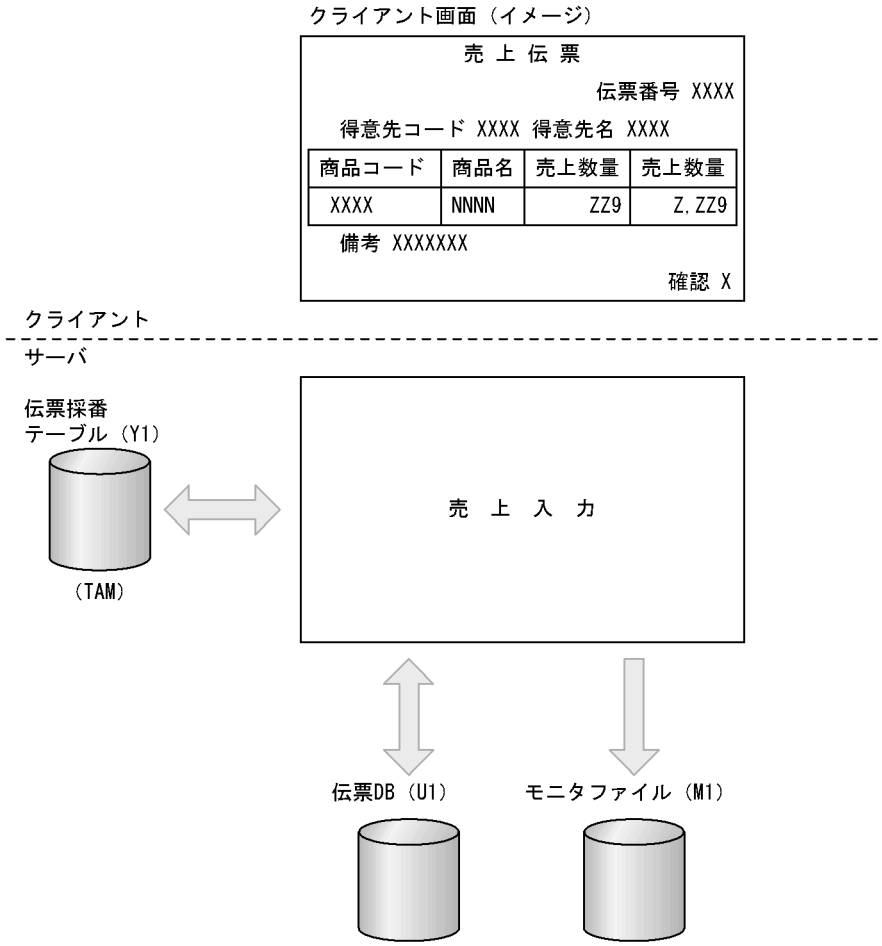
(1) 例題テンプレートで作れるプログラム

クライアントプログラム（SUP）からのデータを基に，伝票の処理区分が登録であれば伝票採番テーブル（TAM レコード）から新しい伝票番号を取ってきます。その伝票番号に 1 を足して伝票番号ファイルを書き換え，クライアントプログラムに伝票番号を付けたデータを送ります。

伝票の処理区分が更新であれば，伝票 DB の中から指定された伝票のデータを取ってきて更新し，更新後のデータをクライアントプログラムに送ります。

TAM は複数のプログラムで共通に，しかも高速にアクセスできるレコードとして使用できます。なお，TP モニタとして OpenTP1 を使い，TAM や DAM を使用する場合は，サーバは TP1/Server Base を使用する必要があります。

(2) 例題テンプレートの入出力構成



(3) ファイルレコードの形式

例題で使用するファイルレコードの形式を次に示します。

- 伝票採番テーブル（Y1）：伝票番号ファイル（TAM）

伝票種別	伝票番号

- 伝票 DB（U1）：伝票ファイル

伝票番号	商品コード	商品名	現在庫量	納入年月日

- モニタファイル（M1）：モニタファイル

伝票番号	商品コード	商品名	現在庫量	納入年月日

- 受信メッセージ

伝票種別	伝票番号	商品コード	商品名	受注量	納期

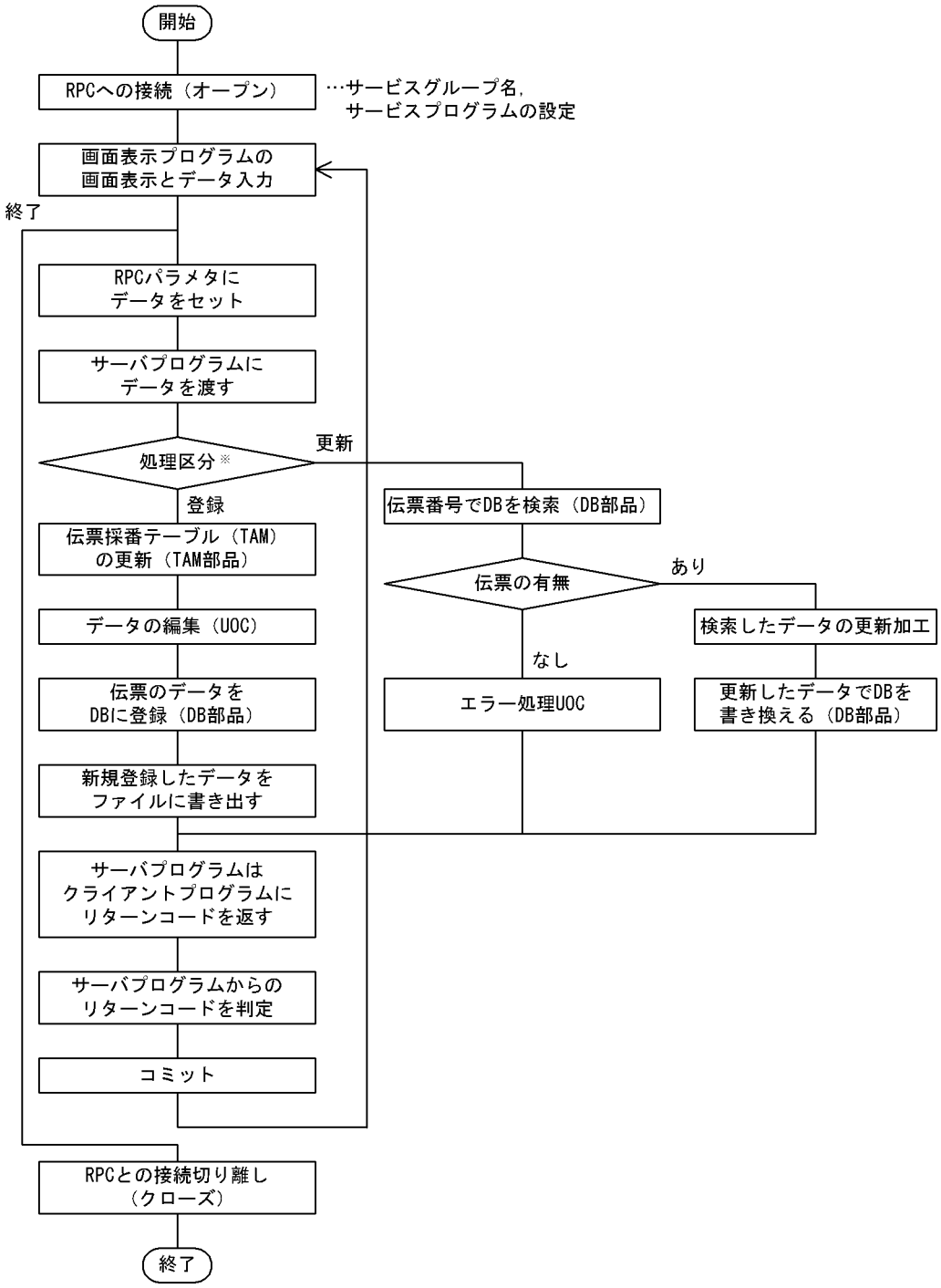
- 発信メッセージ

伝票番号	商品コード	商品名	受注量	納期

（凡例）□：キーとなるデータ

(4) 例題テンプレートの処理概要

サーバプログラムが行う処理は，フロー中で明記しています。それ以外はクライアントプログラムが行う処理です。



注※　サーバ側のSPPは事前に起動されていて，クライアントプログラムからの要求を待っている状態にある。

(5) 例題テンプレートの記述のポイント

1. TAM レコードにアクセスする処理を部品にする。
2. 伝票の登録処理を生成するか，または伝票の更新処理を生成するかを，プログラム作成者がプログラム定義ウィンドウで入力するパラメタで制御する（@@if 文を使用)。
3. ファイルや DB のアクセス処理，TAM アクセス処理の部品中のエラー処理をユーザ追加処理にする。エラー処理のプログラム固有の処理をプログラム作成者に記述させる（@@uoc 文を使用)。
4. モニタファイルへの出力指示の有無を判断し，展開を制御する（@@if 文を使用)。
5. @syscount を使用し，部品の呼び出しが 1 回目のときだけ TAM のパラメタを展開する。

(6) 例題テンプレートと生成されたソースプログラム

クライアント，およびサーバのメインプログラムのテンプレートとソースプログラムは「6.2.1　データエントリの例題」を参考にしてください。ここでは，サーバ側のプログラム（売上入力）のテンプレートとソースプログラムを掲載します。なお，DB アクセス部品は「6.1.2　マスタ更新・追加出力の例題」で使用している RDBPARTS と類似しているため，ここでは省略しています。

テンプレート（伝票入力・訂正）1 ／ 10

```
000101 @@lang COBOL                                                              (1)
000102   UOC_BEGIN      = "**USER-S @uocname"
000103   UOC_END        = "**USER-E @uocname";
000104   @@interface                                                             (2)
000105     @プログラムＩＤ  =
000106      {COMMENT        = "プログラムＩＤを英字８文字で入力"   };
000107 @@*
000108   @@interface
000109     @モニタファイル  =
000110      {ATTR           = FILE,
000111       COMMENT        = "キーがなかった時モニタファイルとしてデータを出力する",
000112       IO             = OUT,
000113       ファイル名     = {ATTR = FILE_NAME                    },
000114       レコード名     = {ATTR = RECORD_NAME                  },
000115       編成           = {ATTR = ORGANIZATION                 },                (3)
000116       外部装置名     = {COMMENT = "外部装置名"              },
000117       キー           = {COMMENT = "アクセスレコードキー",
000118                           ATTR = ITEM,REF = @モニタファイル   },
000119       アクセスモード = {COMMENT = "順アクセス：'S',キー順：'D'を指定"  },
000120       接頭語         = {COMMENT = "ファイルのPREFIXを入れる"          }
000121       };
000122 @@*
000123   @@interface
000124     @伝票ＤＢ  =
000125      {ATTR           = DB,
000126       COMMENT        = "入力必須、ＲＤＢの表名を指定する",
000127       IO             = OUT,
000128       表名           = {ATTR = TABLE_NAME                   },                (4)
000129       レコード名     = {ATTR = RECORD_NAME                  },
000130       ユーザＩＤ     = {},
000131       パスワード     = {},
000132       接頭語         = {COMMENT = "ファイルのPREFIXを入れる" }
000133       };
000134 @@*
000135   @@interface
000136     @キー項目 =
000137      {ATTR           = ITEM,
000138       COMMENT        = "伝票ＤＢを検索するためキーとなる列名を指定する",          (5)
000139       REF            = @伝票ＤＢ
000140       };
000141 @@*
```

(1) ユーザ追加処理のユニークな開始コメントおよび終了コメントを出力する。
(2) プログラムＩＤの宣言をする。
(3) モニタファイルの宣言をする。
(4) 伝票ＤＢの宣言をする。
(5) ＤＢの検索キーの定義，プログラム定義でレコード表示を指示する。

テンプレート（伝票入力・訂正）2 ／ 10

```
000142 @@*
000143   @@interface
000144     @ＲＰＣ受信メッセージ =                                                    ┐
000145      {ATTR           = RPC_INPARM,                                          │
000146       COMMENT        = "ＲＰＣ受信メッセージ",                              ├ (6)
000147       IO             = IN,                                                  │
000148       入力データ     = {ATTR = RECORD_NAME                  }               │
000149       };                                                                    ┘
000150 @@*
000151   @@interface
000152     @ＲＰＣ発信メッセージ =                                                    ┐
000153      {ATTR           = RPC_REPLY,                                           │
000154       COMMENT        = "ＲＰＣ発信メッセージ",                              ├ (7)
000155       IO             = OUT,                                                 │
000156       出力データ     = {ATTR = RECORD_NAME                  }               │
000157       };                                                                    ┘
000158   @@interface
000159     @キー値項目 =                                                            ┐
000160      {ATTR           = ITEM,                                                │
000161       COMMENT        = "伝票ＤＢを検索するキー値として対応する"             ├ (8)
000162                        "ＲＰＣの受信レコード項目を指定",                    │
000163       REF            = @ＲＰＣ受信メッセージ                                │
000164       };                                                                    ┘
000165 @@*
000166   @@interface
000167     @伝票採番テーブル =                                                      ┐
000168      {ATTR           = TAM,                                                 │
000169       COMMENT        = "伝票番号採番テーブル",                              │
000170       レコード名     = {ATTR = RECORD_NAME},                                │
000171       ＴＡＭキー     = {ATTR = ITEM,  REF = @伝票採番テーブル          },   ├ (9)
000172       ＴＡＭキー値   = {ATTR = ITEM,  REF = @ＲＰＣ受信メッセージ     },   │
000173       接頭語         = {}                                                   │
000174       };                                                                    ┘
000175 @@*
000176   @@if(@モニタファイル  eq  "")
000177     @@set  @モニタファイル指定フラグ  = "N";
000178   @@else
000179     @@set  @モニタファイル指定フラグ  = "Y";
000180   @@end;
000181 *
```

(6) ＲＰＣ受信メッセージの宣言をする。
(7) ＲＰＣ発信メッセージの宣言をする。
(8) ＤＢの検索キー値となる項目を定義する。プログラム定義でレコードの表示を指示する。
(9) 伝票番号を取得するＴＡＭファイルの宣言をする。

テンプレート（伝票入力・訂正）3 ／ 10

```
000182   IDENTIFICATION DIVISION.
000183    PROGRAM-ID. @プログラムＩＤ..
000184 *
000185   ENVIRONMENT DIVISION.
000186    INPUT-OUTPUT SECTION.
000187     FILE-CONTROL.
000188 *
000189      @@if (@モニタファイル指定フラグ  eq  "Y")
000190        SELECT @モニタファイル[接頭語].-@モニタファイル[ファイル名]                (10)
000191               ASSIGN TO @モニタファイル[外部装置名]
000192               @@if (@モニタファイル[アクセスモード] eq "S" )
000193                 ACCESS MODE IS SEQUENTIAL
000194               @@else                                                          (11)
000195                 ACCESS MODE IS DYNAMIC
000196               @@end;
000197               @@if (@モニタファイル[編成] == 4)
000198                 ORGANIZATION IS INDEXED
000199               @@else
000200                 @@msg  "モニタファイルが索引順編成ではありません";
000201                 @@errorexit;
000202               @@end;
000203               RECORD KEY IS @モニタファイル[接頭語].-@モニタファイル[キー]..
000204      @@end;
000205 *
000206   DATA DIVISION.
000207    FILE SECTION.
000208    @@if(@モニタファイル指定フラグ  eq  "Y")
000209      FD @モニタファイル[接頭語].-@モニタファイル[ファイル名]  .
000210      @@expand @モニタファイル[レコード名] PREFIX = "@モニタファイル[接頭語].-";
000211    @@end;
000212 *
000213   WORKING-STORAGE SECTION.
000214 *
000215 *      EXEC SQL INCLUDE SQLCA END-EXEC.
000216 *
000217    @@if (@モニタファイル指定フラグ eq "Y" )
000218    01  CSS1-OPN-@モニタファイル[接頭語]  PIC X(1)  VALUE ZERO.
000219    01  CSS1-INV-@モニタファイル[接頭語]  PIC X(1)  VALUE ZERO.
000220    01  CSS1-STATUS-@モニタファイル[接頭語]  PIC 9(2)  VALUE ZERO.
000221    01  CSS1-DUP-@モニタファイル[接頭語]  PIC X(1)  VALUE ZERO.
000222    01  CSS1-LOCK-@モニタファイル[接頭語]  PIC X(1)  VALUE ZERO.
000223    01  CSS--@モニタファイル[接頭語]  PIC X(1)  VALUE ZERO.
000224    @@end ;
000225    01  BAT1-INV-@伝票ＤＢ[接頭語]        PIC  X(1)  VALUE  ZERO .
000226    01  CSS--@伝票ＤＢ[接頭語]        PIC X(1)  VALUE ZERO .
000227 *
000228    @@merge  @CONNECT;
000229    @@merge  @RDBWORK;
000230    @@merge  @Tamwork;
000231    01  BAT--ERR-MSG  PIC X(44)  VALUE
000232                          'ＲＤＢアクセス時にエラーが発生しました。'.
```

(10) モニタファイルを展開する。
(11) @@if文でアクセスモードの展開の切り換えおよび編成法のチェックをする。

ソースプログラム（伝票入力・訂正）1／8

```
000101  IDENTIFICATION DIVISION.
000102   PROGRAM-ID. SVR02.
000103 *
000104  ENVIRONMENT DIVISION.
000105   INPUT-OUTPUT SECTION.
000106    FILE-CONTROL.
000107 *
000108      SELECT M1-MONITOR                                          ┐
000109             ASSIGN TO M1                                        │
000110             ACCESS MODE IS SEQUENTIAL                           ├ (10),(11)
000111             ORGANIZATION IS INDEXED                             │
000112             RECORD KEY IS M1-商品コード.                        ┘
000113 *
000114  DATA DIVISION.
000115   FILE SECTION.
000116 *
000117    FD M1-MONITOR.
000118    01 M1-商品レコード.
000119     02 M1-伝票番号  PIC 9(8).
000120     02 M1-商品コード  PIC X(4).
000121     02 M1-商品名  PIC X(6).
000122     02 M1-現在庫量  PIC 9(3).
000123     02 M1-納入年月日  PIC X(6).
000124 *
000125   WORKING-STORAGE SECTION.
000126 *
000127      EXEC SQL INCLUDE SQLCA END-EXEC.
000128 *
000129    01  CSS1-OPN-M1  PIC X(1)  VALUE ZERO.
000130    01  CSS1-INV-M1  PIC X(1)  VALUE ZERO.
000131    01  CSS1-STATUS-M1  PIC 9(2)  VALUE ZERO.
000132    01  CSS1-DUP-M1  PIC X(1)  VALUE ZERO.
000133    01  CSS1-LOCK-M1  PIC X(1)  VALUE ZERO.
000134    01  CSS--M1  PIC X(1)  VALUE ZERO.
000135    01  BAT1-INV-U1  PIC X(1)  VALUE ZERO.
000136    01  CSS--U1  PIC X(1)  VALUE ZERO.
000137 *
000138      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000139    77  ユーザＩＤ  PIC X(30).
000140    77  パスワード  PIC X(30).
000141      EXEC SQL END DECLARE SECTION END-EXEC.
000142    01  SQLCODE  PIC S9(9) COMP.
000143    01  BAT--CURSOR-SW-U1  PIC X(1).
000144    01  BAT--U1  PIC X(1).
000145    01  CSS--STATUS-Y1  PIC X(1)  VALUE ZERO.    .
000146    01  CSS--SYS-MSG  PIC X(80)  VALUE
000147                        'ＴＡＭアクセスでエラーが発生しました。'
000148    01  BAT--ERR-MSG  PIC X(44)  VALUE
000149                         'ＲＤＢアクセス時にエラーが発生しました。'
000150        .
```

テンプレート（伝票入力・訂正）4 ／ 10

```
000233 *
000234    @@merge  @Tamparam;                                                    (12)
000235    @@expand  @伝票採番テーブル[レコード名] PREFIX = "@伝票採番テーブル[接頭語].-";
000236    @@<RDBPARTS>DECLARE(@伝票ＤＢ);
000237 *
000238   LINKAGE SECTION.
000239 *
000240  01  INDATA.                                                             }
000241      @@expand @ＲＰＣ受信メッセージ[入力データ]  PREFIX = "IN-";          } (13)
000242  01  INDATA-LEN  PIC S9(9) COMP.                                         }
000243  01  OUTDATA.                                                            }
000244    @@expand @ＲＰＣ発信メッセージ[出力データ]  PREFIX = "OUT-";         } (14)
000245  01  OUTDATA-LEN  PIC S9(9) COMP.                                        }
000246 *
000247   PROCEDURE DIVISION
000248             USING INDATA INDATA-LEN OUTDATA OUTDATA-LEN.
000249 *
000250    CSS--PROGRAM-CONTROL SECTION.
000251      CSS--PROGRAM-RUN.
000252      PERFORM CSS--PROLOGUE
000253      PERFORM CSS--MAIN
000254      PERFORM CSS--EPILOGUE
000255      EXIT PROGRAM
000256      .
000257      CSS--STOP-RUN.
```

(12) ＴＡＭアクセスのパラメタ宣言をする。
(13) クライアントからの入力パラメタを展開する。
(14) クライアントへ出力パラメタを展開する。

ソースプログラム（伝票入力・訂正）2／8

```
000151 *
000152    01  CSS--TAM-ARG1.                                          ┐
000153     02  CSS--TAM-FC  PIC  X(4).                                │
000154     02  CSS--TAM-STAT  PIC  X(5).                              │
000155     02  CSS--TAM-FIL-1  PIC  X(3).                             │
000156     02  CSS--TAM-TBL-NAM  PIC  X(32).                          │
000157     02  CSS--TAM-FIL-2  PIC  X(68).                            │
000158     02  CSS--TAM-BUFL  PIC  S9(4) COMP.                        ├ (12)
000159     02  CSS--TAM-FIL-3  PIC  X(398).                           │
000160    01  CSS--TAM-REQ.                                           │
000161     02  CSS--TAM-REQ-COD  PIC  X(4).                           │
000162     02  CSS--TAM-LOCK-MOD  PIC  X(4).                          │
000163    01 Y1-DENREC                                               ┘
000164     02  Y1-伝票種別  PIC  X(2).
000165     02  Y1-伝票番号  PIC  9(8).
000166     02  Y1-予備  PIC  X(70).
000167   EXEC SQL BEGIN DECLARE SECTION END-EXEC.
000168    01  U1-DENPYOU.
000169     02  U1-伝票番号  PIC  9(8).
000170     02  U1-商品コード  PIC X(4).
000171     02  U1-商品名      PIC X(6).
000172     02  U1-現在庫量    PIC X(3).
000173     02  U1-納入年月日  PIC X(6).
000174   EXEC SQL END DECLARE SECTION END-EXEC.
000175 *
000176   LINKAGE SECTION.
000177 *
000178   01  INDATA.                                                 ┐
000179    02  IN-伝票レコード.                                        │
000180     03 IN-伝票種別  PIC X(2).                                 │
000181     03 IN-伝票番号  PIC X(4).                                 ├ (13)
000182     03 IN-商品コード  PIC X(4).                               │
000183     03 IN-商品名  PIC X(6).                                   │
000184     03 IN-受注量  PIC 9(3).                                   │
000185     03 IN-納期  PIC X(6).                                     ┘
000186   01  INDATA-LEN  S9(9) COMP.
000187   01  OUTDATA.                                                ┐
000188    02 OUT-伝票レコード.                                       │
000189     03 OUT-伝票番号  PIC X(4).                                │
000190     03 OUT-商品コード  PIC X(4).                              ├ (14)
000191     03 OUT-商品名  PIC X(6).                                  │
000192     03 OUT-受注量  PIC 9(3).                                  │
000193     03 OUT-納期  PIC X(6).                                    ┘
000194   01  OUTDATA-LEN  S9(9) COMP.
000195 *
000196  PROCEDURE DIVISION
000197                       USING INDATA INDATA-LEN OUTDATA OUTDATA-LEN.
000198 *
000199   CSS--PROGRAM-CONTROL SECTION.
000200    CSS--PROGRAM-RUN.
000201      PERFORM CSS--PROLOGUE
000202      PERFORM CSS--MAIN
000203      PERFORM CSS--EPILOGUE
000204      EXIT PROGRAM
000205      .
000206    CSS--STOP-RUN.
```

テンプレート（伝票入力・訂正）5 ／ 10

```
000258      EXEC SQL
000259        WHENEVER SQLERROR
000260        CONTINUE
000261      END-EXEC
000262      EXEC SQL
000263        COMMIT RELEASE
000264      END-EXEC                                                    (15)
000265      EXEC SQL
000266        WHENEVER SQLERROR
000267        GO TO CSS--ABEND
000268      END-EXEC
000269      STOP RUN
000270      .
000271 *
000272    CSS--PROLOGUE SECTION.
000273      @@if (@モニタファイル指定フラグ eq "Y")
000274        OPEN OUTPUT @モニタファイル[接頭語].-@モニタファイル[ファイル名]   (16)
000275      @@end;
000276      EXEC SQL
000277        WHENEVER SQLERROR
000278        GO TO CSS-ABEND
000279      END-EXEC
000280      @@merge @RDBINIT;
000281      @@if (@モニタファイル指定フラグ eq "Y")
000282        MOVE '1' TO CSS1-INV-@モニタファイル[接頭語]                 (17)
000283        INITIALIZE @モニタファイル[接頭語].-@モニタファイル[キー]
000284      @@end;
000285      @@<RDBPARTS>CONNECT(@伝票ＤＢ);                              (18)
000286      @@<RDBPARTS>LOCKTABLE(@伝票ＤＢ);                            (19)
000287      @@<RDBPARTS>DECLAREカーソル(@伝票ＤＢ,@キー項目,@キー値項目);  (20)
000288      .
000289 *
000290    CSS--EPILOGUE SECTION.
000291      @@if (@モニタファイル指定フラグ eq "Y")
000292        CLOSE @モニタファイル[接頭語].-@モニタファイル[ファイル名]    (21)
000293      @@end;
000294      .
```

(15) ＳＱＬ異常終了時に処理する。
(16) モニタファイルをオープンする。
(17) モニタファイルを読み込むためのキーを初期設定する。
(18) 伝票ＤＢのユーザＩＤとパスワードを展開する。
(19) ＤＢのLOCKTABLE部品を呼び出す。
(20) ＤＢのカーソル宣言部品を呼び出す。
(21) モニタファイルをクローズする。

ソースプログラム（伝票入力・訂正）3／8

```
000207      EXEC SQL                                                    ┐
000208       WHENEVER SQLERROR                                          │
000209       CONTINUE                                                   │
000210      END-EXEC                                                    │
000211      EXEC SQL                                                    │
000212       COMMIT RELEASE                                             ├ (15)
000213      END-EXEC                                                    │
000214      EXEC SQL                                                    │
000215       WHENEVER SQLERROR                                          │
000216       GO TO CSS--ABEND                                           │
000217      END-EXEC                                                    │
000218      STOP RUN                                                    ┘
000219      .
000220 *
000221   CSS--PROLOGUE SECTION.
000222      OPEN OUTPUT M1-MONITOR                                        (16)
000223      EXEC SQL
000224       WHENEVER SQLERROR
000225       GO TO CSS--ABEND
000226      END-EXEC
000227      MOVE 0 TO CSS--U1
000228      MOVE 0 TO CSS--CURSOR-SW-U1
000229      MOVE '1' TO CSS1-INV-M1                                       (17)
000230      INITIALIZE M1-伝票番号                                        ┐
000231      MOVE 'HIRDB' TOユーザＩＤ                                     ├ (18)
000232      MOVE 'HIRDB' TOパスワード                                     ┘
000233      EXEC SQL
000234       CONNECT :ユーザＩＤ IDENTIFIED BY :パスワード
000235      END-EXEC
000236      EXEC SQL                                                    ┐
000237       LOCK TABLE "DENPYOU"                                       ├ (19)
000238         IN ROW EXCLUSIVE MODE                                    │
000239      END-EXEC                                                    ┘
000240      EXEC SQL                                                    ┐
000241       DECLARE CURSORU1 CURSOR FOR                                │
000242         SELECT *                                                 │
000243           FROM "DENPYOU"                                         ├ (20)
000244           WHERE "伝票番号" = :U1-伝票番号                        │
000245      END-EXEC                                                    │
000246      .                                                           ┘
000247 *
000248   CSS--EPILOGUE SECTION.                                        ┐
000249      CLOSE M1-MONITOR                                           ├ (21)
000250      .                                                          ┘
```

テンプレート（伝票入力・訂正）6 ／ 10

```
000295 *
000296    CSS--MAIN SECTION.                                                    (22)
000297      @@merge @DB_OPEN;
000298      @@merge @DB_READ;
000299      PERFORM CSS--GET-DATA
000300      PERFORM CSS--PROCESS-UPD-MNT
000301      @@if  (@モニタファイル指定フラグ  eq  "Y")                          (23)
000302        PERFORM  CSS--WRITE-@モニタファイル[接頭語]
000303      @@end;
000304      PERFORM CSS--OUTPUT-UPD-FILE
000305      @@merge @DB_CLOSE;
000306      .
000307 *
000308    CSS--GET-DATA SECTION.
000309      IF IN-@伝票採番テーブル[ＴＡＭキー値] = '1'
000310        @@merge @READ;                                                    (24)
000311        MOVE IN-@伝票採番テーブル[ＴＡＭキー値]                           (25)
000312             TO @伝票採番テーブル[接頭語].-@伝票採番テーブル[ＴＡＭキー]
000313        @@uoc "伝票番号更新処理" ;
000314        @@merge @REWRITE;
000315      ELSE
000316        IF IN-@伝票採番テーブル[ＴＡＭキー値] = '2'
000317          IF  BAT1-INV-@伝票ＤＢ[接頭語] = '1'
000318            @@uoc "エラー処理" ;
000319            STOP RUN
000320          END-IF
000321        END-IF
000322      END-IF
000323      .
000324 *
000325      @@<RDBPARTS>FETCH(@伝票ＤＢ);                                       (26)
000326 *
000327      @@<RDBPARTS>OPEN(@伝票ＤＢ);                                        (27)
```

(22) メインルーチン。
(23) モニタファイルの指定がある場合，出力処理を展開する。
(24) ＴＡＭからデータを読み込む。
(25) 伝票番号をカウントアップする。
(26) ＤＢのフェッチ(列読み込み)部品を呼び出す（フェッチセクションの展開）。
(27) カーソルオープン部品を呼び出す（オープンセクションの展開）。

ソースプログラム（伝票入力・訂正）4 ／ 8

```
000251 *
000252   CSS--MAIN SECTION.
000253      PERFORM CSS--CURSOR-OPEN-U1
000254      PERFORM CSS--READ-U1
000255      PERFORM CSS--GET-DATA
000256      PERFORM CSS--PROCESS-UPD-MNT
000257      PERFORM CSS--WRITE-M1
000258      PERFORM CSS--OUTPUT-UPD-FILE
000259      PERFORM CSS--CURSOR-CLOSE-U1
000260      .
000261 *
000262   CSS--GET-DATA SECTION.
000263      IF IN-伝票種別 = '1'
000264        PERFORM  CSS-TAM-READ
000265        MOVE IN-伝票種別 TO Y1-伝票種別
000266 **USER-S 伝票番号更新処理
000267        COMPUTE  Y1-伝票番号  =  Y1-伝票番号 + 1
000268 **USER-E 伝票番号更新処理
000269        PERFORM  CSS--TAM-REWRITE
000270      ELSE
000271        IF IN-伝票種別 = '2'
000272          IF  BAT1-INV-U1 = '1'
000273 **USER-S エラー処理
000274            DISPLAY '更新エラー'
000275 **USER-E エラー処理
000276            STOP RUN
000277          END-IF
000278        END-IF
000279      END-IF
000280      .
000281 *
000282   BAT--READ-U1 SECTION.
000283      EXEC SQL
000284       FETCH CURSORU1
000285        INTO
000286             : U1-納入年月日
000287            ,: U1-現在庫量
000288            ,: U1-商品名
000289            ,: U1-商品コード
000290            ,: U1-伝票番号
000291      END-EXEC
000292      IF SQLCODE = 1403
000293       PERFORM BAT--NOTFOUND-U1
000294      END-IF
000295      .
000296 *
000297   BAT--CURSOR-OPEN-U1 SECTION.
000298      IF BAT--CURSOR-SW-U1 = 0
000299       EXEC SQL
000300        OPEN CURSORU1
000301       END-EXEC
000302       MOVE 1 TO BAT--CURSOR-SW-U1
000303      END-IF
000304      .
```

テンプレート（伝票入力・訂正）7 ／ 10

```
000328 *
000329       @@<RDBPARTS>CLOSE(@伝票ＤＢ);                              (28)
000330 *
000331       @@<RDBPARTS>NOTFOUND(@伝票ＤＢ);                           (29)
000332 *
000333       @@<RDBPARTS>ABEND();                                       (30)
000334 *
000335       @@<RDBPARTS>WRITE(@伝票ＤＢ);                              (31)
```

(28) カーソルクローズ部品を呼び出す（クローズセクションの展開）。
(29) カーソルNOT FOUND部品を呼び出す（カーソルNOT FOUNDセクションの展開）。
(30) 異常終了部品を呼び出す（異常終了セクションの展開）。
(31) ＤＢの追加部品を呼び出す（追加セクションの展開）。

ソースプログラム（伝票入力・訂正）5／8

```
000301 *
000302   BAT--CURSOR-CLOSE-U1 SECTION.
000303      IF BAT--CURSOR-SW-U1 = 1
000304       EXEC SQL
000305        WHENEVER SQLERROR
000306        CONTINUE
000307       END-EXEC
000308       EXEC SQL
000309        CLOSE CURSORU1
000310       END-EXEC
000311       EXEC SQL
000312        WHENEVER SQLERROR
000313        GO TO BAT--ABEND
000314       END-EXEC
000315       MOVE 0 TO BAT--CURSOR-SW-U1
000316      END-IF
000317      .
000318 *
000319   BAT--NOTFOUND-U1 SECTION.
000320      MOVE '1' TO BAT1-INV-U1
000321      INITIALIZE  U1-伝票番号
000322                  U1-商品コード
000323                  U1-商品名
000324                  U1-現在庫量
000325                  U1-納入年月日
000326      .
000327 *
000328   BAT--ABEND SECTION.
000329      DISPLAY BAT--ERR-MSG
000330      STOP RUN
000331      .
000332 *
000333   BAT--WRITE-U1 SECTION.
000334      EXEC SQL
000335       INSERT INTO "DENPYOU"
000336        VALUES (:U1-伝票番号
000337                ,:U1-商品コード
000338                ,:U1-商品名
000339                ,:U1-現在庫量
000340                ,:U1-納入年月日
000341                )
000342      END-EXEC
000343      INITIALIZE  U1-伝票番号
000344                  U1-商品コード
000345                  U1-商品名
000346                  U1-現在庫量
000347                  U1-納入年月日
000348      .
```

(28)

(29)

(30)

(31)

テンプレート（伝票入力・訂正）8 ／ 10

```
000336 *
000337      @@<RDBPARTS>UPDATE(@伝票ＤＢ);                                    (32)
000338 *
000339      @@if  (@モニタファイル指定フラグ  eq  "Y")
000340    CSS--WRITE-@モニタファイル[接頭語]  SECTION .
000341        WRITE  @モニタファイル[接頭語].-@モニタファイル[レコード名]
000342        INITIALIZE  @モニタファイル[接頭語].-@モニタファイル[レコード名]
000343        .
000344      @@end;
000345 *
000346    CSS--OUTPUT-UPD-FILE SECTION.
000347      IF CSS--@伝票ＤＢ[接頭語] = '1'
000348        EVALUATE TRUE
000349        WHEN BAT1-INV-@伝票ＤＢ[接頭語] = '0'
000350          @@merge @DB_REWRITE;                                          (33)
000351        WHEN BAT1-INV-@伝票ＤＢ[接頭語] = '1'
000352          @@merge @DB_WRITE;                                            (34)
000353        END-EVALUATE
000354      END-IF
000355      .
000356 *
```

(32) ＤＢの更新部品を呼び出す（更新セクションの展開）。
(33) ＤＢ更新セクションを呼び出す。
(34) ＤＢ追加セクションを呼び出す。

ソースプログラム（伝票入力・訂正）6／8

```
000349 *
000350   BAT--REWRITE-U1 SECTION.                                          ┐
000351      EXEC SQL                                                       │
000352       UPDATE "DENPYOU" SET                                          │
000353        "伝票番号" = :U1-伝票番号,                                   │
000354        "商品コード" = :U1-商品コード,                               │
000355        "商品名" = :U1-商品名,                                       │
000356        "現在庫量" = :U1-現在庫量,                                   │
000357        "納入年月日" = :U1-納入年月日                                │
000358        WHERE CURRENT OF CURSORU1                                     │
000359      END-EXEC                                                       │
000360      MOVE '0' TO BAT1-INV-U1                                        ├ (32)
000361      PERFORM BAT--READ-U1                                           │
000362      IF BAT1-INV-U1 = '0'                                           │
000363       PERFORM BAT--ABEND                                            │
000364      END-IF                                                         │
000365      INITIALIZE  U1-伝票番号                                        │
000366                  U1-商品コード                                      │
000367                  U1-商品名                                          │
000368                  U1-現在庫量                                        │
000369                  U1-納入年月日                                      ┘
000370      .
000371 *
000372   CSS--WRITE-M1 SECTION.
000373      WRITE M1-商品レコード
000374      INITIALIZE M1-商品レコード
000375      .
000376 *
000377   CSS--OUTPUT-UPD-FILE SECTION.
000378      IF CSS--U1 = 1
000379       EVALUATE TRUE
000380        WHEN CSS1-INV-U1 = '0'
000381         PERFORM CSS--REWRITE-U1                                     (33)
000382        WHEN CSS1-INV-U1 = '1'
000383         PERFORM CSS--WRITE-U1                                       (34)
000384       END-EVALUATE
000385      END-IF
000386      .
000387 *
```

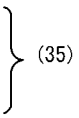
テンプレート（伝票入力・訂正）9 ／ 10

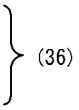
```
000357    CSS--PROCESS-UPD-MNT SECTION.
000358      IF BAT1-INV-@伝票ＤＢ[接頭語] = '1'
000359        PERFORM CSS--PROC-ADD-@伝票ＤＢ[接頭語]
000360        MOVE '1' TO CSS--@伝票ＤＢ[接頭語]
000361        PERFORM CSS--PROC-UNMATCH-@モニタファイル[接頭語]
000362        MOVE '1' TO CSS--@モニタファイル[接頭語]
000363        CONTINUE
000364      ELSE
000365        PERFORM CSS--PROC-UPDATE-@伝票ＤＢ[接頭語]
000366        MOVE '1' TO CSS--@伝票ＤＢ[接頭語]
000367      END-IF
000368      .
000369 *
000370    CSS--PROC-UPDATE-@伝票ＤＢ[接頭語] SECTION.
000371      @@UOC '伝票ＤＢ更新加工処理' ;                                    (35)
000372      .
000373 *
000374    CSS--PROC-ADD-@伝票ＤＢ[接頭語] SECTION.
000375      @@UOC '伝票ＤＢ追加加工処理' ;                                    (36)
000376      .
000377 *
000378    CSS--PROC-UNMATCH-@モニタファイル[接頭語] SECTION.
000379      @@UOC 'モニタファイル加工処理' ;                                  (37)
000380      .
000381 *
```

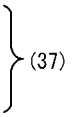
(35) ＤＢ更新加工処理を記述する。
(36) ＤＢ追加加工処理を記述する。
(37) モニタファイル加工処理を記述する。

ソースプログラム（伝票入力・訂正）7／8

```
000388   CSS--PROCESS-UPD-MNT SECTION.
000389       IF CSS1-INV-U1 = '1'
000390        PERFORM CSS--PROC-ADD-U1
000391        MOVE '1' TO CSS--U1
000392        PERFORM CSS--PROC-UNMATCH-M1
000393        MOVE '1' TO CSS--M1
000394        CONTINUE
000395       ELSE
000396        PERFORM CSS--PROC-UPDATE-U1
000397        MOVE '1' TO CSS--U1
000398       END-IF
000399       .
000400 *
000401   CSS--PROC-UPDATE-U1 SECTION.
000402 **USER-S 伝票ＤＢ更新加工処理
000403       MOVE Y1-伝票番号 TO U1-伝票番号
000404       MOVE IN-商品コード TO U1-商品コード
000405       MOVE IN-商品名 TO U1-商品名
000406       MOVE IN-受注量 TO U1-現在庫量
000407       MOVE IN-納期 TO U1-納入年月日
000408 **USER-E 伝票ＤＢ更新加工処理
000409       .
000410 *
000411   CSS--PROC-ADD-U1 SECTION.
000412 **USER-S 伝票ＤＢ追加加工処理
000413       MOVE Y1-伝票番号 TO U1-伝票番号
000414       MOVE IN-商品コード TO U1-商品コード
000415       MOVE IN-商品名 TO U1-商品名
000416       MOVE IN-受注量 TO U1-現在庫量
000417       MOVE IN-納期 TO U1-納入年月日
000418 **USER-E 伝票ＤＢ追加加工処理
000419       .
000420 *
000421   CSS--PROC-UNMATCH-M1 SECTION.
000422 **USER-S モニタファイル加工処理
000423       MOVE Y1-伝票番号 TO M1-伝票番号
000424       MOVE IN-商品コード TO M1-商品コード
000425       MOVE IN-商品名 TO M1-商品名
000426       MOVE IN-受注量 TO M1-現在庫量
000427       MOVE IN-納期 TO M1-納入年月日
000428 **USER-E モニタファイル加工処理
000429       .
000430 *
```







テンプレート（伝票入力・訂正） 10 ／ 10

```
000382      @@set @ＴＡＭアクセスキー
000383            = "@伝票採番テーブル[接頭語].-@伝票採番テーブル[ＴＡＭキー]";
000384      @@set @ＴＡＭアクセスキー値
000385            = "IN-@伝票採番テーブル[ＴＡＭキー値]";
000386      @@<TAMPARTS>TAMREAD(@伝票採番テーブル,                          }
000387                        @ＴＡＭアクセスキー,                          } (38)
000388                        @ＴＡＭアクセスキー値);                       }
000389      .
000390 *
000391      @@<TAMPARTS>TAMREWRITE(@伝票採番テーブル,                       }
000392                        @ＴＡＭアクセスキー,                          } (39)
000393                        @ＴＡＭアクセスキー値);                       }
000394      .
000395 *
000396      @@<TAMPARTS>TAMCHECK(@伝票採番テーブル,                         }
000397                        @ＴＡＭアクセスキー,                          } (40)
000398                        @ＴＡＭアクセスキー値);                       }
000399      .
```

(38) 伝票採番テーブル（ＴＡＭ）の読み込み部品を展開する。
(39) 伝票採番テーブル（ＴＡＭ）の更新部品を展開する。
(40) 伝票採番テーブルアクセスのチェックルーチンを展開する。

ソースプログラム（伝票入力・訂正）8／8

```
000425   CSS--READ-TAM-PROC SECTION.
000426      MOVE IN-伝票番号 TO Y1-伝票番号
000427      INITIALIZE CSS--TAM-ARG1 CSS--TAM-REQ
000428      MOVE 'FCHU' TO CSS--TAM-REQ-COD
000429      MOVE 'Y1-DENREC' TO CSS--TAM-TBL-NAM
000430      MOVE 80 TO CSS--TAM-BUFL     (37)
000431      MOVE 1 TO CSS-STATUS-Y1                                          (38)
000432      CALL 'CBLDCTAM' USING CSS--TAM-ARG1 CSS--TAM-REQ Y1-伝票番号
000433           Y1-DENPYOU
000434      PERFORM CSS--TAM-STATUS-CHECK
000435      .
000436 *
000437   CSS--TAM-REWRITE  SECTION.
000438      MOVE IN-伝票番号 TO Y1-伝票番号
000439      INITIALIZE CSS--TAM-ARG1 CSS--TAM-REQ
000440      MOVE 'FMY' TO CSS--TAM-REQ-COD
000441      MOVE 'Y1-DENREC' TO CSS--TAM-TBL-NAM
000442      MOVE 80 TO CSS--TAM-BUFL     (38)
000443      MOVE 1 TO CSS-STATUS-Y1
000444      CALL 'CBLDCTAM' USING CSS--TAM-ARG1 CSS--TAM-REQ Y1-伝票番号    (39)
000445           Y1- DENPYOU
000446      PERFORM CSS--TAM-STATUS-CHECK
000447      .
000448 *
000449   TAM-STATUS-CHECK  SECTION.
000450      EVALUATE CSS--TAM-STAT
000451      WHEN '00000'
000452        CONTINUE
000453      WHEN OTHER         (39)                                          (40)
000454        DISPLAY CSS--SYS-MSG
000455      END-EVALUATE.
000456      .
000457
```

部品（伝票入力・訂正）1 ／ 3

```
000101 @@*****************************************************************************
000102 @@* Name          : GenRec1
000103 @@* Abstract      : CBLDCTAM共通引数レコード生成（非公開プロシジャ）
000104 @@*                 初回呼び出し時に限り上記レコードを生成。
000105 @@*                 ２回目以降の呼び出しでは、何も行わない。
000106 @@* Argument      : なし
000107 @@*****************************************************************************
000108   @@proc TAMARG1()
000109     @@if ( @syscount == 1)
000110         @@put @Tamparam  <<
000111   01  CSS--TAM-ARG1.                                                    ┐
000112    02  CSS--TAM-FC  PIC  X(4).                                          │
000113    02  CSS--TAM-STAT  PIC  X(5).                                        │
000114    02  CSS--TAM-FIL-1  PIC  X(3).                                       │
000115    02  CSS--TAM-TBL-NAM  PIC  X(32).                                    ├ (B-1)
000116    02  CSS--TAM-FIL-2  PIC  X(68).                                      │
000117    02  CSS--TAM-BUFL  PIC  S9(4) COMP.                                  │
000118    02  CSS--TAM-FIL-3  PIC  X(398).                                     ┘
000119         @@end;
000120     @@end ;
000121 @@*****************************************************************************
000122 @@* Name          : GenRec2
000123 @@* Abstract      : CBLDCTAMリクエストコード用レコード生成（非公開プロシジャ）
000124 @@*                 初回呼び出し時に限り上記レコードを生成。
000125 @@*                 ２回目以降の呼び出しでは、何も行わない。
000126 @@* Argument      : なし
000127 @@*****************************************************************************
000128   @@proc TAMARG2()
000129     @@if ( @syscount == 1)                                             ┐
000130         @@put @Tamparam  <<                                            │
000131   01  CSS--TAM-REQ.                                                   ├(B-2)
000132    02  CSS--TAM-REQ-COD  PIC  X(4).                                   │
000133    02  CSS--TAM-LOCK-MOD  PIC  X(4).                                  │
000134         @@end;                                                        │
000135    @@end;                                                             ┘
000136 @@***********************
000137 @@*  ＴＡＭ　ＲＥＡＤ
000138 @@***********************
000139   @@proc  TAMREAD(@TAMtable, @KEYITEM, @KEYVAL)
000140 @@*********************************
000141 @@*  CBLDCTAM共通引数レコード
000142 @@*********************************
000143    @@<TAMPARTS>  TAMARG1();
```

(B-1) ＴＡＭアクセスの引数１。
(B-2) ＴＡＭアクセスの引数２。

部品（伝票入力・訂正）2／3

```
000144 @@*****************************
000145 @@*  CBLDCTAM共通引数レコード
000146 @@*****************************
000147   @@<TAMPARTS>  TAMARG2();
000148   @@put  @Tamwork <<
000149     01  CSS--STATUS-@TAMtable[接頭語]  PIC  X(1) VALUE  ZERO .
000150   @@end;
000151   @@put  @READ <<                                                    }(B-3)
000152     PERFORM  CSS-TAM-READ                                            }
000153   @@end;
000154   CSS-TAM-READ  SECTION.
000155     MOVE  @KEYVAL  TO  @KEYITEM
000156     INITIALIZE  CSS--TAM-ARG1  CSS--TAM-REQ
000157     MOVE 'FCHU' TO CSS--TAM-REQ-COD
000158     MOVE @TAMtable[接頭語].-@TAMtable[レコード名] TO CSS--TAM-TBL-NAM
000159      @@set @RECLEN = @@reclen(@TAMtable[レコード名]);
000160     MOVE   @RECLEN TO CSS--TAM-BUFL                                  }(B-4)
000161     MOVE 1 TO CSS--STATUS-@TAMtable[接頭語]
000162     CALL  'CBLDCTAM'  USING  CSS--TAM-ARG1  CSS--TAM-REQ  @KEYITEM
000163           @TAMtable[接頭語].-@TAMtable[レコード名]
000164     PERFORM  CSS--TAM-STATUS-CHECK  .
000165 @@*****************************
000166 @@*  ＴＡＭ　ＲＥＷＲＩＴＥ
000167 @@*****************************
000168   @@proc  TAMREWRITE(@TAMtable, @KEYITEM, @KEYVAL)                  }(B-5)
000169   @@put  @REWRITE <<                                               }
000170     PERFORM  CSS-TAM-REWRITE                                       }
000171   @@end;
000172 @@*******************************
000173 @@*  CBLDCTAM共通引数レコード
000174 @@*******************************
000175    @@<TAMPARTS>  TAMARG1();
000176 @@*******************************
000177 @@*  CBLDCTAM共通引数レコード
000178 @@*******************************
000179    @@<TAMPARTS>  TAMARG2();
000180   CSS-TAM-REWRITE  SECTION.                                          }
000181     MOVE  @KEYVAL  TO  @KEYITEM                                      }
000182     INITIALIZE  CSS--TAM-ARG1  CSS--TAM-REQ                          }
000183     MOVE 'MFY' TO CSS--TAM-REQ-COD                                   }
000184     MOVE @TAMtable[接頭語].-@TAMtable[レコード名] TO CSS--TAM-TBL-NAM  }(B-6)
000185     @@set @RECLEN  =  @@reclen(@TAMtable[レコード名]);               }
000186     MOVE @RECLEN TO CSS--TAM-BUFL                                    }
000187     MOVE 1 TO CSS--STATUS-@TAMtable[接頭語]                          }
000188     CALL  'CBLDCTAM'  USING  CSS--TAM-ARG1  CSS--TAM-REQ  @KEYITEM  }
000189           @TAMtable[接頭語].-@TAMtable[レコード名]                    }
000190     PERFORM  CSS--TAM-STATUS-CHECK .                                 }
```

(B-3) ＴＡＭ読み込みセクションを呼び出す。
(B-4) ＴＡＭ読み込みセクションを展開する。
(B-5) ＴＡＭ更新セクションを呼び出す。
(B-6) ＴＡＭ更新セクションを展開する。

部品（伝票入力・訂正）3／3

```
000191 @@************************
000192 @@*  ＴＡＭ　ＷＲＩＴＥ
000193 @@************************
000194   @@proc  TAMwrite(@TAMtable, @KEYITEM, @KEYVAL)                    }(B-7)
000195   @@put  @WRITE <<                                                 }
000196     PERFORM  TAMWRITE
000197   @@end;
000198 @@*******************************
000199 @@*  CBLDCTAM共通引数レコード
000200 @@*******************************
000201    @@<TAMPARTS>  TAMARG1();
000202 @@*******************************
000203 @@*  CBLDCTAM共通引数レコード
000204 @@*******************************
000205    @@<TAMPARTS>  TAMARG2();
000206   CSS-TAM-WRITE  SECTION.                                          )
000207     MOVE  @KEYVAL  TO  @KEYITEM                                    )
000208     INITIALIZE  CSS--TAM-ARG1  CSS--TAM-REQ                        )
000209     MOVE 'STR' TO CSS--TAM-REQ-COD                                 )
000210     MOVE @TAMtable[接頭語].-@TAMtable[レコード名] TO CSS--TAM-TBL-NAM )
000211     @@set  @RECL =  @@reclen(@TAMtable[レコード名]);                 }(B-8)
000212     MOVE @RECL TO CSS--TAM-BUFL                                    )
000213     MOVE 1 TO CSS--STATUS-@TAMtable[接頭語]                        )
000214     CALL  'CBLDCTAM'  USING  CSS--TAM-ARG1  CSS--TAM-REQ  @KEYITEM )
000215           @TAMtable[接頭語].-@TAMtable[レコード名]                   )
000216     PERFORM  CSS--TAM-STATUS-CHECK  .                              )
000217 @@*******************************
000218 @@*  ＴＡＭ　アクセスチェック
000219 @@*******************************
000220   @@PROC  TAMCHECK (@TAMtable, @KEYITEM, @KEYVAL)
000221   @@put @Tamwork <<
000222    01     CSS--SYS-MSG  PIC X(80)
000223             VALUE  'ＴＡＭアクセスでエラーが発生しました。'.
000224   @@end ;
000225   CSS--TAM-STATUS-CHECK  SECTION.                                  )
000226     EVALUATE CSS--TAM-STAT                                         )
000227     WHEN '00000'                                                   )
000228       CONTINUE                                                     }(B-9)
000229     WHEN OTHER                                                     )
000230       DISPLAY CSS--SYS-MSG                                         )
000231     END-EVALUATE.                                                  )
```

(B-7) ＴＡＭ追加セクションを呼び出す。
(B-8) ＴＡＭ追加セクションを展開する。
(B-9) ＴＡＭのアクセスチェックセクションを展開する。

7 テンプレート記述言語

テンプレートを作成するとき，そのままソースプログラムとして生成したい内容は，生成したい言語を使い，その文法規則に従って記述します。これに対して，生成を制御する文や，データ定義，プログラム定義で定義される情報を設定する変数は，SEWB+/CONSTRUCTION で用意されているテンプレート記述言語を使います。この章では，テンプレート記述言語の文法規則について説明します。

7.1 テンプレート記述言語の概要

7.1.1 使用できる文字セット

テンプレート記述言語で使用できる文字セットは次のとおりです。

- 英字
  英大文字：A ～ Z
  英小文字：a ～ z
  アンダーバー：_
- 数字：0 ～ 9
- 特殊文字：＋－＊／（ ）＝, ' "！＆｛ ｝［ ］ 空白 ｜（論理演算子）
- 漢字：2 バイトの漢字コード
- 仮名：1 バイトの仮名文字

空白とは，ブランク，水平のタブ，垂直のタブ，改行，および改ページのことです。

予約語に関しては大文字，小文字のどちらで記述しても予約語として解釈されますが，混在して指定することはできません。

展開文中で使用できる文字列は，展開する言語（COBOL や C など）の仕様に従います。

7.1.2 トークン

トークンとは分離記号で分離したあとの字句の列のことです。テンプレートは SEWB+/CONSTRUCTION によって字句が解析されたあと，次のトークンと文に分離されます。

● トークン
- 識別子（可変記号名，プロシジャ名，配列の添字）
- 分離記号
- 演算子
- 展開文字列
- キーワード

● 文
- 展開制御文：可変記号に対して演算や比較設定を行う文（@@set，@@if など）
- 展開文：ソースプログラムに展開される文（MOVE，include など）

7.1.3 定数

定数には数値定数と文字定数があります。

(1) 定数として解釈される場所

定数は展開制御文中（@@set 文の代入値や，@@if 文の条件式中など）に記述した場合だ

け定数として解釈されます。展開文中に記述しても，単なる展開文字列としか解釈されません。

（例）

```
@@if (@START_MSG == "Yes")
  DISPLAY '**** PROGRAM STARTED  *****'
@@end;
```

このとき，条件中の "Yes" は展開制御文中に記述されているため，文字定数として解釈されます。しかし，DISPLAY 文の '＊＊＊＊ PROGRAM STARTED ＊＊＊＊＊' は展開文中に記述されているため，文字定数ではなく，展開文の一部とみなされます。

（2） 数値定数（numeric literal）

数値定数は次の規則に従って数値を表現したものです。

- 整定数で記述する（小数は記述できない）。
- 数字の列で 9 けたまで指定できる。

（例）

```
123456        …10進数
```

（3） 文字定数（string literal）

文字定数は，文字の並びをアポストロフィ（'）かダブルクォーテーション（"）で囲んだものです。ただし，文字定数を囲む 1 組を，アポストロフィとダブルクオーテーションを混在して記述することはできません。

文字定数に可変記号を記述する場合は次の注意が必要です。

- 可変記号の内容を，文字定数として展開する場合にはダブルクォーテーションで囲む。可変記号を単なる文字列とする場合はアポストロフィで囲む。

（例）

```
@@set  @MSG = "****** START!! *****";
@@set  @START1 = "@MSG";      …ダブルクォーテーション
                                 で囲まれているため@MSG
                                 の内容"***** START!!
                                 *****"が設定される。
@@set  @START2 = '@MSG';      …アポストロフィで囲まれ
                                 ているため@MSGがそのま
                                 ま設定される。
```

したがって，そのあとの展開文での指定によって，展開される内容は次のようになります。

DISPLAY '@START1' なら DISPLAY '＊＊＊＊＊ START!! ＊＊＊＊＊' が展開される。

DISPLAY '@START2' なら '@MSG' が展開される。

文字定数を，行をわたって記述する場合は，各行を文字定数として記述します。

（例）

```
@COMMENT = "入力ファイル1個,"
           "出力ファイル2個,"
           "DB1はHiRDBです。"
```

この指定は次のように 1 行で記述したのと同じです。

@COMMENT = " 入力ファイル 1 個，出力ファイル 2 個，DB1 は HiRDB です。"

文字定数内で改行や復帰などの出力できない文字や，アポストロフィやダブルクォーテーションなどの特別な意味を持つ文字を表すときは，¥ 記号とそれに続く文字の組み合わせを記述することによって表現できます。

表現したい内容	記述する文字
展開制御開始文字	¥@
改行	¥n
復帰	¥r
タブ	¥t
円記号	¥¥
アポストロフィ	¥'
ダブルクォーテーション	¥"

7.1.4 可変記号

プログラム定義やデータ定義で定義された情報は @@interface 文を使って可変記号と呼ばれる変数に設定されます。また，生成する処理中で @@set 文を使って直接，可変記号に値を設定することもできます。

(1) 可変記号名の記述規則

可変記号名は次の規則に従って記述します。

- 先頭は展開制御開始文字 @ で始める。
- 名称の長さは 30 文字以下である。
- 英字の大文字と小文字は区別される。
- 特殊文字は使用できない。

（正しい例）

```
@i
@FILE_A
@001
```

（誤りの例）

```
@A=3    … 3番目の文字が特殊文字である。
@A-3    … 3番目の文字はハイフンとは判断されないで，演算子
          と判断される。
```

（展開例）

```
MOVE @XA TO B    … MOVE 8 TO B (@XAが8のとき)
X = @XA;         … X = 8;      (@XAが8のとき)
```

（2） @@set 文による値の設定

可変記号に値を設定するときは @@set 文を使います。

可変記号には文字列，または数値が設定できます。

（例）

```
@@set @i = 1;
@@set @PROG_NAME = "PROG01";
可変記号に設定された値は文脈によって文字列，または数値として解釈される。
1  @@set @z = "ABC";     …@Zは文字列「ABC」と解釈さ
                           れる。
2  @@set @k = "0000";    …@kは文字列「0000」と解釈さ
                           れる。
3  @@set @q = @k + 1;    …@kは数値「0」と解釈され@qは
                           「1」になる。
```

（3） 可変記号のグローバル宣言

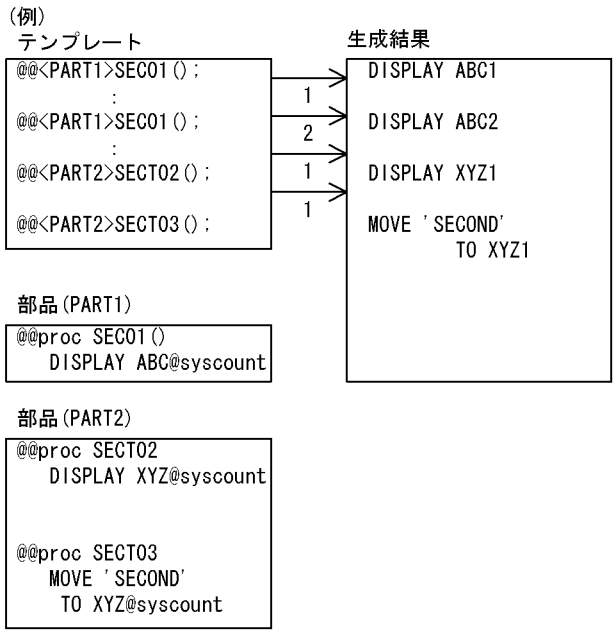
テンプレート中や部品中で共有して可変記号を参照したり更新したりする場合に @@global 文で指定します。詳細は「7.9.17　@@global 文」を参照してください。

（4） システム可変記号

システム可変記号とは，SEWB+/CONSTRUCTION が提供する可変記号です。

(a) @syscount

部品のプロシジャごとに呼び出した通算の回数を表す可変記号です。



(5) 可変記号を設定する文の組み合わせ

可変記号を設定する場合，組み合わせて使用できる文には次の制限があります。

後	先					
	@@merge	@@put	@@global	@@set	@@interface	@@foreach
@@merge	○	○	×	×	×	×
@@put	○	○	×	×	×	×
@@global	×	×	○	×	×	×
@@set	○	○	○	○	×	○
@@interface	×	×	○	×	×	×
@@foreach	○	○	○	○	○	○

(凡例)
○：指定できる。
×：指定できない。

7.1.5 配列の記述規則

可変記号を配列として参照するとき，次の書き方に従って記述します。

可変記号［添字［，添字］…］

配列は値の集合を格納するときに使う変数（可変記号）です。配列は連想配列方式

（associative array）です。配列の特定の値を参照するときに添字として，数値，文字列，または可変記号が使えます。

添字の指定はキーワード（文字列）で設定する方法（キーワード指定）と，数値で指定する方法（位置指定）の 2 通りがあります。

(1) キーワード指定

添字にキーワード（文字列）で指定します。

（例）

```
@@set  @item = {name="KEY001", pic="X(4)"};
```

とすると，@itemは次のように参照される。

```
@item[name]          …KEY001
@item[pic]           …X(4)
```

また，次のように，添字に可変記号を使用することもできる。

（例 1）1 から 3 の順に設定されていった場合

```
1  @@set @TEST[a]=1;     …添字「a」の値は「1」
2  @@set @TEST[b]=2;     …添字「b」の値は「2」
3  @@set @i="a";
```

このとき，@TEST[@i] は 1 となる。（@TEST[@i] は @TEST[a] のため）

（例 2）

```
@@set @WORKS={
   {level=1,name="ITM001"},
   {level=2,name="ITM002",pic="X(4)"},
   {level=2,name="ITM003",pic="9(4)"}
};
@@set @C = @WORKS[2];        … @Cには
                                {level=2,name="ITM002"
                                ,pic="X(4)"}が設定され
                                る。
@@set @D = @WORKS[3,name];   … @DにはITM003が設定さ
                                れる。
```

このとき，可変記号には次の値が設定されている。

```
@@count[@WORKS]     …3(@@countは配列の要素数を取り出
                       す関数)
@WORKS[1,name]      …ITM001
@WORKS[2,name]      …ITM002
@WORKS[2,pic]       …X(4)
@WORKS[3,name]      …ITM003
@WORKS[3,pic]       …9(4)
@C[name]            …ITM002
@C[pic]             …X(4)
@D[1]               …エラー(可変記号Dの内容が配列では
                       ないので構文エラーになる)
```

(2) 位置指定

{ } 内に値を「,」で区切って並べると，一度に配列の値が設定できます。{ } 内に値だけを並べた場合，その順番が配列の添字（1 からの数字）になります。

（例）

```
@@set @Ar1={"a01","a02","a03"};
このとき
@Ar1[1]       …a01
@Ar1[2]       …a02
@Ar1[3]       …a03
@Ar1[4]       …値が設定されていないためエラーになる
```

（3） 配列の規則

- 初期化は次のように指定する。
  文字列の場合，@@set @Arl[1] = "";
  数値の場合，@@set @Arl[1] = 0;
- 配列の添字の指定，および初期化の指定で位置指定とキーワード指定は混在できない。
- 一度位置指定で初期化した配列は，キーワード指定で再度設定できない。
- 一度キーワード指定で初期化した配列は，位置指定で再度設定できない。
- 展開文中の可変記号に配列を指定する場合，可変記号と配列指定の記号の間に空白を指定してはならない。空白がある場合，テキストとして解釈され展開される。

（誤りの例）

```
MOVE  @Ar1  [1]     …可変記号と配列指定の間に空白がある。
```

（4） 配列の添字の規則

配列の添字は次の三つのうちのどれかでなければなりません。

- 9 けた以内の数字。
- 可変記号。
- 名称の長さが 30 文字以内の文字列。このとき，英字の大文字と小文字は区別される。

7.1.6 部分参照の記述規則

可変記号の一部を参照したいときに次の書き方に従って指定します。

@@str（可変記号，先頭位置［，文字列長］）

部分参照の規則は次のとおりです。

- 先頭位置および文字列長には，数字定数，算術式，または可変記号を指定する。
- 文字列長（文字数）の指定は省略できる。その場合，先頭位置から可変記号の右端までを文字列長とする。
- データの文字列が 2 バイトコードの場合，データは 1 文字ずつ設定される。
- 指定された先頭位置にデータがない場合はエラーになる。
- 指定された文字列長よりもデータが少ない場合，データが入っているところまで設定される。

（例）

```
@@set @A001 = "ABCDE";
@@set @B001 = @@str(@A001,3,5);     …@B001の内容は
                                       'CDE'
```

7.1.7 予約語

展開制御開始文字を二つ連続した文字列「@@」で始まる文字列，および次の文字列は，予約語のため，別の用途で使用できません。

ARRAY_MAX

ASCEND

ATTR

ATTR_NAME

BASIC_ATTR

COMMENT

DESCEND

EQ

EXTENSION

FOR_REPOSITORY

GE

GT

IO

LE

LT

MODIFY_CONNECT

MODIFY_ORDER

NE

NOVALUE

OUTPUT_NAME

PARENT

PARSE_LEVEL

PREFIX

REF

START

START_POSITION

SUFFIX

SUPPRESS

TAG_NAME

TOP

TYPE

UOC_BEGIN

UOC_END

UP

USAGE

VALUE

WITH

7.1.8 展開文字列

展開文字列とは，展開文で可変記号以外の文字列を示します。空白や改行文字は展開文字列に含まれます。

（例）

7.1.9 分離記号

分離記号とは，文字列を区切るための記号です。テンプレートの展開制御文で使用します。

分離記号	名　　称	使　用　方　法
△	空白	演算子や予約語をほかの文字と区切る。文字列と文字列，文字列と分離記号および分離記号と分離記号の間に置いて，式や文を見やすくする。
,	コンマ	関数の引数や添字の並びに区切りを付ける。
( )	丸括弧	演算子の評価順序を示す。関数の引数となる文字列を囲む。
'	アポストロフィ	文字定数を囲む。
"	ダブルクォーテーション	文字定数を囲む。
== <= < >=> !=	比較演算子	比較演算子の左右の項を比較して，大小または等価を判定する。

分離記号	名　　称	使　用　方　法
+ − * /	算術演算子	算術演算子の左右の項目を，指定された演算子で計算する。
{}	波括弧	構造体の要素を示す。
[]	角括弧	配列の要素を示す。
なし	改行，改ページ，水平タブ，垂直タブ	改行，復帰，タブ（空白として扱う）
<<	テキスト代入子	可変記号にテキストを代入する。
&& ||	論理演算子	論理演算子の左右の項を論理演算する。

7.2 演算子

演算子には次の 5 種類があります。

- 算術演算子
- 比較演算子
- 論理演算子
- 代入演算子
- 文字列連結演算子

7.2.1 算術演算子

算術演算子には次の 4 種類があります。

算術演算子	意　味
＋	左項の値に右項の値を足す
－	左項の値から右項の値を引く
＊	左項の値に右項の値を掛ける
／	左項の値を右項の値で割る

7.2.2 比較演算子

比較演算子には数値の比較用と文字列の比較用に，それぞれ 6 種類ずつあります。比較の結果が真なら 1 を，偽なら 0 を返します。

数値の比較用	文字列の比較用	意　味
＝＝	eq	左項の値が右項の値に等しいとき真になる
＞	gt	左項の値が右項の値より大きいときに真になる
＞＝	ge	左項の値が右項の値以上のときに真になる
＜＝	le	左項の値が右項の値以下のとき真になる
＜	lt	左項の値が右項の値より小さいときに真になる
！＝	ne	左項の値が右項の値と異なるときに真になる

比較演算子は結合できません。したがって，次のような式は誤りです。

（誤りの例）

```
@a < @b < @c
```

（正しい例）

```
@a < @b &&  @b < @c
```

7.2.3 論理演算子

論理演算子には &&，||，！の 3 種類があります。

(a) 論理積「& &」

左辺の条件と右辺の条件の論理積を表します。

左辺と右辺の条件とその論理積は次のとおりです。

左辺	右辺	論理積
真	真	真
真	偽	偽
偽	真	偽
偽	偽	偽

(b) 論理和「｜｜」

左辺の条件と右辺の条件の論理和を表します。

左辺と右辺の条件とその論理和は次のとおりです。

左辺	右辺	論理和
真	真	真
真	偽	真
偽	真	真
偽	偽	偽

(c) 条件の否定「！」

条件の否定を表します。

条件とその否定は次のとおりです。

条　件	否　定
真	偽
偽	真

7.2.4 代入演算子

代入演算子は，「＝」です。右辺の式を左辺に代入します。詳細は，「7.9.45　@@set 文」を参照してください。

7.2.5 文字列連結演算子

文字列連結演算子は「.」です。連結演算子は可変記号と任意の文字列，または可変記号と可変記号を連結するときに使用します。

（例）

```
READ  @File_Name
 AT END
  SET @File_Name.EOF TO TRUE        …@File_Name
END-REA                               に設定された
                                      文字列に
                                      「EOF TO TRUE」
                                      が結合される。
```

可変記号のすぐ後ろにピリオドを生成したいときは，ピリオドを二つ続けて記述します。

（例）

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. @Prog_Name..          …@Prog_Nameに
                                     「PROG001」が設定さ
                                     れていると，
                                     PROGRAM-IDは
                                     「PROG001.」。
```

詳細は「7.3.4　連結式」を参照してください。

7.2.6 優先順位と結合規則

演算子はすべて左から右へと結合されます。なお，優先順位は次のとおりです。

優先順位	演　算　子
1	（　）内の演算子
2	＊／
3	＋－
4	＜　＜＝　＞　＞＝　lt le gt ge
5	＝＝　！＝　eq ne
6	&&
7	||

7.3 式

式には次の 5 種類があります。

- 算術式
- 条件式
- 代入式
- 連結式
- 記号生成式

7.3.1 算術式

形 式

正負記号　　項
項　算術演算子　項

注

「正負記号　　項」「項　算術演算子　項」のどちらかの形式で指定します。

機 能

四則演算をします。

規 則

- 項には，次の 4 種類が指定できる。
  算術式，可変記号，数字定数，数値関数
  （数値関数には @@count，@@reclen，@@length，@@lengthb の 4 種類がある）
- 可変記号の値は，数字でなければならない。
- 可変記号には部分参照は指定できない。
- 可変記号が配列を持っている場合，添字を指定する。
- 算術演算の精度は整数 9 けたである。
- 一つの式の中に複数の算術演算子があるときは，表に示す順番で演算する。ただし，一つの算術式の中に優先順位の同じ算術演算子が複数あるときは，左の演算子から順に演算する。

優先順位	算術演算子
1	（ ）内の算術式
2	算術演算子の「＊」および「／」
3	算術演算子の「＋」および「－」

使用例

```
+10
100 + 10
```

```
100 + -10
@A * @B
@B + 10
(@a+@B)  /  (@C+@D)
@A+@@reclen(@B)
```

算術式の評価中に可変記号の値が数値でないものがある場合は，エラーになる。

```
@@set @A ="A"
@@set @B =@A + 1    …@Aの値は0～9以外のためエラーに
                       なる。
```

7.3.2 条件式

条件式には，比較条件式と複合条件式があります。

(1) 比較条件式

形　式

論理関数

項　比較演算子　項

注

「論理関数」「項　比較演算子　項」のどちらかの形式で指定します。

機　能

比較条件式を挟む左側と右側の項の大小関係を比較し，成立または不成立を示します。

規　則

- 論理関数には @@defined を指定できる。
- 項には次の 3 種類を指定できる。
  算術式，可変記号，文字定数
- 比較演算子が文字列用のときは，文字コードや漢字コードの大小で比較される。
- 比較演算子が文字列用のとき，両側の文字数が異なるときは，左から順に 1 文字ずつ比較される。

使用例

HITAC と HITACHI を比較したとき
HITAC
HITACHI
左から H，I，T，A，C の順に比較されます。このとき，C の次の文字でヌル値と「H」が比較され，各文字コードの大小で比較条件が決まる。

(2) 複合条件式

形　式

[!]　条件式
条件式　｜｜　条件式
条件式　＆＆　条件式

注

「[!]　条件式」「条件式　｜｜　条件式」「条件式　＆＆　条件式」のどれかの形式で指定します。

機　能

処理の選択条件を示します。

規　則

- 項には次の 4 種類を指定できる。
  算術式，可変記号，数値定数，文字定数
- 条件式には，論理関数が指定できる。
- 一つの式の中に論理演算子が複数ある場合の判定の優先順位は次のとおりである。

判定の優先順位	論理演算子
1	括弧内の条件
2	!
3	＆＆
4	｜｜

- 条件が複数ある場合，条件を判定する順番は次の規則に従う。

括弧が多重の場合
　条件はいちばん内側の括弧の中から，外側の括弧に向かって順番に判定される。

比較条件がある場合
　条件は次の表の順番で判定される。

判定の優先順位	条　件
1	比較条件の中の算術式
2	比較演算子
3	論理演算子

使用例

```
(@A == 1) && (@B >= 100)
```

7.3.3 代入式

形　式

可変記号　＝　式

機　能

右辺の式を左辺の可変記号に代入します。

7.3.4 連結式

形　式

可変記号．文字列
可変記号．可変記号

注

「可変記号．文字列」「可変記号．可変記号」のどちらかの形式で指定します。

機　能

連結演算子（.）は，可変記号と任意の文字列，または可変記号と可変記号を連結するときに使用します。連結演算子は可変記号の直後にあるときだけ連結演算子として解釈されます。

使用例

@NAM が "A001"，@MOD が "（1）"，@SUB が 1 のとき

```
@NAM.1          …A0011
@NAM@MOD        …A001(1)
@NAM.@MOD       …A001(1)
@NAM..@MOD      …A001.(1)
@NAM(@SUB)      …A001(1)
```

7.3.5 記号生成式

形　式

@（可変記号）
@（連結式）
@（文字列［可変記号］）

注

「@（可変記号）」「@（連結式）」「@（文字列［可変記号］）」のどれかの形式で指定します。

機　能

幾つかの可変記号と文字列から可変記号を生成できます。

使用例

```
@@set @TAB[1]="AAA";
@@set @TAB[2]="BB";
@@set @TAB[3]="C";
@@set @TAB[4]="1";
@@set @TAB[5]="A*B";
```

このとき@（@TAB［@I］001）は次の可変記号を生成する。

@I が 1 のとき…

@AAA001

@I が 2 のとき…

@BB001

@I が 3 のとき…

@C001

@I が 4 のとき…

@1001

@I が 5 のとき…

@A ＊ B001 となり，英数字でない文字があるのでエラーとなる。

7.4 注釈

注釈の記述は「@@ ＊」で始めます。注釈は文字列「@@ ＊」が現れた行の末尾で終了します。注釈は生成されるソースプログラムには展開されません。

生成する言語の注釈（COBOL 言語の＊など）については，特別な解釈はされません。そのまま生成されます。

(例 1)

@@* ** 入力ファイル読み込み展開 **

とテンプレートに記述しても，ソースプログラムには何も生成されない。

(例 2)

* 入力ファイル　読み込み *

```
MOVE  @AA TO @BB
READ  @入力ファイル［ファイル名］
```

とテンプレートに記述すると，生成する言語の注釈のため次のように生成される。

* 入力ファイル　読み込み *

```
MOVE  商品コード TO I1-商品コード
READ  SHOUHIN
```

ソースプログラム中に文字列 @，または @@ を展開させたい場合は，「@@」を記述した直後に記述する。

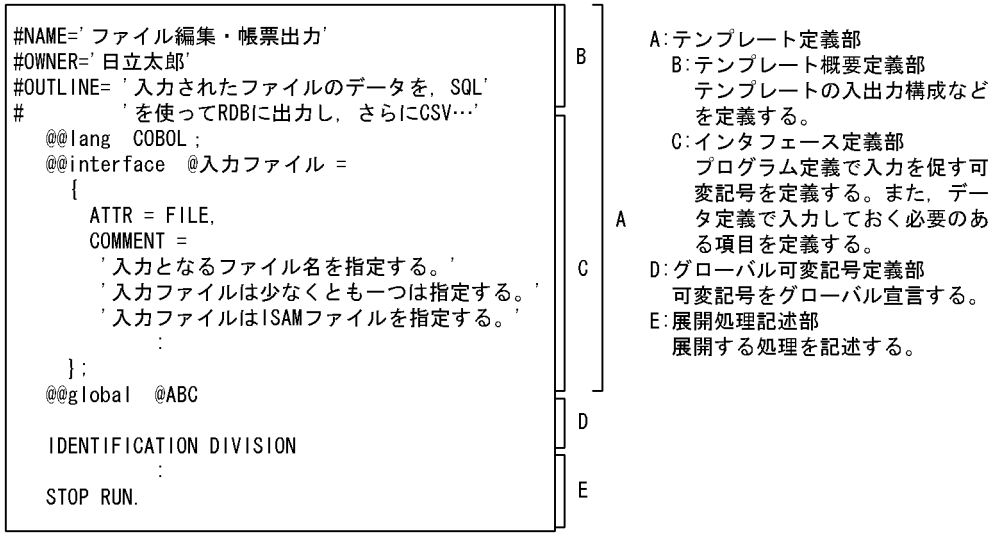
(例)

展開文の記述	→	展開結果
/* @@@@IF */		/* @@IF */
/* @@@PROG */		/* @PROG */

7.5　テンプレート定義部の宣言

7.5.1　テンプレート定義部の構成

テンプレート定義部は図のような構成で記述します。



7.5.2　テンプレート概要定義部

テンプレートに関する情報をテンプレート中に記述できます。記述できる情報はテンプレートの日本語名称，テンプレート作成者名，テンプレートの概要です。概要定義部はテンプレートの１行目の１カラム目から記述します。

記述した内容はプロパティ情報として見ることができます。したがって，テンプレートにこれらの定義がしてあれば，プログラム作成者はテンプレートを開いてそのつど処理の流れを読まなくても，プロパティ情報で簡単に内容を確認できます。

(1) テンプレートの日本語名称

形　式

#NAME = ' 文字列 '

規　則

- インタフェース定義部より前に記述する。
- １カラム目から記述しなければならない。

使用例

#NAME=' マスタファイル更新処理 '

(2) テンプレート作成者名

形　式

#OWNER = ' 文字列 '

規　則

- インタフェース定義部より前に記述する。
- 1 カラム目から記述しなければならない。

使用例

#OWNER=' 日立太郎 '

(3) テンプレートの概要

形　式

#OUTLINE = ' 文字列 ' [' 文字列 '…]

規　則

- 1 カラム目から記述しなければならない。
- 複数行記述する場合は，2 行目からは 1 カラム目に # を記述したあとで，アポストロフィー（'）で囲んで記述する。
- 文字定数に可変記号を指定してはならない。

使用例

#OUTLINE = ' マスタファイルを更新する '
#' 入力ファイルは SAM，または ISAM'
#' 更新 DB は HiRDB 用 '

7.5.3　インタフェース定義部

テンプレートとプログラム定義，データ定義，パラメタ，部品，および SEWB+/CS-DESIGN で作成した論理設計図との関係は，すべてテンプレートのインタフェース定義部で @@interface 文を使って定義します。

また，@@lang 文で，記述する言語の種別と生成規則を指定します。@@lang 文の詳細は「7.9.32　@@lang 文」を参照してください。

(1) @@interface 文の記述規則

形　式 1 データ定義を使用する場合（[入出力] タブ）

```
@@interface 可変記号 =
    {     ATTR = データ定義種別
        [,ARRAY_MAX = 数字定数 ]
        [,COMMENT = 文字定数 ]
        [,IO = { IN     } ]
               { OUT    }
               { IN_OUT }
        [,修飾名 =
               {  [ ATTR = { データ定義識別子 } ]
                           { ITEM           }
                  [,REF  = { 可変記号 } ]
                           { *        }
                  [,ARRAY_MAX = 数字定数 ]
                  [,VALUE =  {定数 [,定数]...} ]
                  [,COMMENT = 文字定数 ]
                  [,修飾名 =
                         { [ ARRAY_MAX = 数字定数         ]
                           [,VALUE = { 定数 [,定数 ]...}  ]
                           [,COMMENT = 文字定数           ]
                         } ] ...
               } ] ...
      } ;
```

形　式 2 [パラメタ] タブを使用する場合（[パラメタ] タブ）

```
@@interface 可変記号 =
    {   [ ATTR = ITEM       ]
        [,REF = { 可変記号 }  ]
                { *        }
        [,ARRAY_MAX = 数字定数 ]
        [,VALUE =  {定数 [,定数]...} ]
        [,COMMENT = 文字定数 ]
        [,修飾名 =
               { [ ARRAY_MAX = 数字定数         ]
                 [,VALUE = { 定数 [,定数 ]...}  ]
                 [,COMMENT = 文字定数           ]
               } ] ...
      } ;
```

形　式 3 XML を使用する場合（[パラメタ] タブ）

```
@@interface 可変記号 =
    {     ATTR = XML
        [,ARRAY_MAX = 数字定数 ]
        [,PARSE_LEVEL = { 1 } ]
                        { 2 }
        [,COMMENT = 文字定数 ]
        [,修飾名 =
                { [ ATTR = {ITEM   }    ]
                           {ELEMENT}
                  [,REF = {可変記号}
                           { *     }
                  [,ARRAY_MAX = 数字定数        ]
                  [,VALUE = { 定数 [,定数 ]...}  ]
                  [,TAG_NAME = {" タグ名" [," タグ名" ]...}]
                  [,ATTR_NAME = " 属性名"         ]
                  [,COMMENT = 文字定数           ]
                  [,修飾名 =
                          { [,ARRAY_MAX = 数字定数 ]
                            [,VALUE =  {定数 [,定数]...} ]
                            [,COMMENT = 文字定数 ]
                          } ] ...
                 } ] ...
      } ;
```

形　式 4 部品を使用する場合（[部品] タブ）

```
@@interface 可変記号 =
    {     ATTR = PARTS
        [,COMMENT = 文字定数 ]
     } ;
```

形　式 5 論理設計図を使用する場合（[インタフェース] タブ）

```
@@interface 可変記号 =
    {     ATTR = { IDL       }
                 { INTERFACE }
                 { OPERATION }
        [,ARRAY_MAX = 数字定数 ]
        [,COMMENT = 文字定数 ]
        [,修飾名 =
                {   [ ATTR = ITEM        ]
                    [,REF = {可変記号 }]
                            { *     }
                    [,ARRAY_MAX = 数字定数 ]
                    [,VALUE =  {定数 [,定数]...} ]
                    [,COMMENT = 文字定数 ]
                    [,修飾名 =
                            { [,ARRAY_MAX = 数字定数        ]
                              [,VALUE = { 定数 [,定数 ]...}  ]
                              [,COMMENT = 文字定数           ]
                             } ] ...
                } ] ...
      } ;
```

形　式6 マップ定義を使用する場合（[入出力] タブ）

```
@@interface 可変記号 =
    {     ATTR =   XMAP3
        [,ARRAY_MAX = 数字定数 ]
        [,COMMENT = 文字定数 ]
        [,IO = { IN     } ]
               { OUT    }
               { IN_OUT }
        [,修飾名 =
               {  [ ATTR =   ITEM ]
                  [,REF  = { 可変記号 } ]
                           { *        }
                   [,ARRAY_MAX = 数字定数 ]
                   [,VALUE =  {定数 [,定数]...} ]
                   [,COMMENT = 文字定数 ]
                    [,修飾名 =
                         { [ ARRAY_MAX = 数字定数          ]
                           [,VALUE = { 定数 [,定数 ]...} ]
                           [,COMMENT = 文字定数          ]
                          } ] ...
               } ] ...
     } ;
```

規　則

(a) ATTR

● 可変記号に指定する ATTR

@@interface 文で指定した可変記号に対し，プログラム作成者が定義できるデータ定義の種別や文書などを指定します。また，データ定義の中の項目を選んでブレークキーやマッチングキーなどに使う場合は，ITEM を指定します（ITEM を指定する場合に，データ定義からレコード定義を参照しているときは，参照しているレコード定義での辞書参照の有無に関係なく，データ項目が選択できます）。

- データ定義を使用する場合は，データ定義種別を次の表の中から指定する。

項番	データ定義種別	内　　容
1	FILE	ファイル情報
2	DB	DB 情報
3	TAM	TAM 情報
4	DAM	DAM 情報
5	RPC_INPARM	RPC 入力パラメタ情報
6	RPC_REPLY	RPC 応答領域情報
7	MSG	メッセージ情報
8	UJ	ユーザジャーナル情報
9	MSGLOG	メッセージログ情報
10	WORK	共通作業領域情報

- マップ定義を使用する場合は，XMAP3 を指定する。
- データ定義種別を指定した場合，プログラム定義の [入出力] タブの入出力項目欄に，可変記号から先頭の @ が除かれたものが表示される。プログラム作成者はここで，プログラムで使用するデータ定義が格納されたファイルを指定する。

（例）

```
@@interface @マスタDB={ATTR=DB,IO=IN_OUT,COMMENT="集計結果に従い…
"};
    :
@@interface @集計サービス={ATTR=RPC_INPARM,IO=IN,COMMENT="2集計元
ファイルを…"};
    :
@@interface @集計作業ファイル={ATTR=FILE,IO=IN_OUT,COMMENT="売上げ
を…"};
    :
@@interface @参照ファイル={ATTR=FILE,IO=IN,ARRAY_MAX=7,COMMENT="
マスタDBに…"};
```

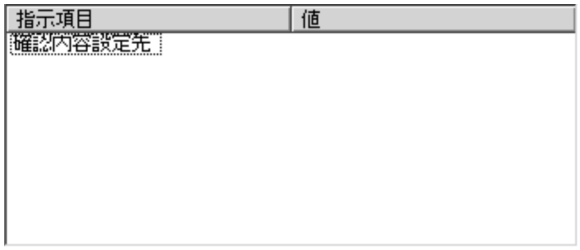
このとき，プログラム定義の [入出力] タブの入出力項目一覧には次のように表示される。

- ITEM は，指定された可変記号に，REF で指定されているデータ定義のレコードの項目のどれかを設定する場合に指定する。または，REF が省略されて @@interface で定義中のデータ定義の項目を指定する場合に使用する。

（例）

```
@@interface @入力ファイル={ATTR=DB,COMMENT="問い合わせ
                           データ…",IO=IN,表名
                           ={ATTR=TABLE_NAME}};
    :
@@interface @確認内容設定先={ATTR=ITEM,REF=@入力ファイ
                             ル,COMMENT="今回の確認内
                             容を…"};
```

このとき，プログラム定義の [パラメタ] タブの指示項目一覧には次のように表示される。



- AP で使用する部品をプログラム定義で指定する場合は，PARTS を指定する。
- PARTS を指定した場合，プログラム定義の [部品] タブの展開部品項目欄に，可変記号から先頭の @ が除かれたものが表示される。プログラム作成者は，ここで部品ファイルを選び，そのあとで展開するプロシジャおよび引数を選ぶ。

（例）

```
@@interface @追加部品={ATTR=PARTS,COMMENT="追加部品を選択する"};
```

- マップ定義を使用する場合は，XMAP3 を指定する。
- XMAP3 を指定した場合，プログラム定義の [入出力] タブの入出力項目欄に，可変記号から先頭の @ が除かれたものが表示される。プログラム作成者は，ここでマップ定義ファイルを選ぶ。
- XML で記述された XML 文書をプログラム生成のパラメタとして利用する場合は，XML を指定する。
- XML を指定した場合，プログラム定義の [パラメタ] タブの指示項目欄に，可変記号から先頭の @ が除かれたものが表示される。プログラム作成者は，ここで XML 文書を選ぶ。
- 論理設計図の IDL ファイルに含まれるインタフェース定義をまとめて取り出す場合は，IDL を指定する。
- 論理設計図にある個々のオブジェクトシンボルに定義されているオブジェクト定義を選ぶ場合は，INTERFACE を指定する。
- 論理設計図のオブジェクト定義の中のオペレーション定義を選ぶ場合は，OPERATION を指定する。
- IDL，INTERFACE または OPERATION を指定した場合，プログラム定義の [インタフェース] タブのインタフェース項目欄に，可変記号から先頭の @ が除かれたものが表示される。プログラム作成者は，ここで論理設計図ファイルを選び，そのあとで IDL ファイル，オブジェクト定義またはオペレーション定義を選ぶ。

（例）

```
@@interface @オペレーション={ATTR=INTERFACE,COMMENT="オペレーション
を選択する"};
```

- ATTR を指定していない場合，プログラム作成者は，プログラム定義の [パラメタ] タブで文字列を入力してデータ種別を定義しなければならない。

● 修飾名に指定する ATTR

プログラム作成者がデータ定義で定義しておく必要のある項目を修飾名として指定し，

ATTR= の後ろに，その項目をデータ定義識別子で記述します。指定できるデータ定義識別子は「(2) データ定義識別子と内容」を参照してください。

- データ定義識別子が指定されると，プログラム作成者がデータ定義で定義した値が直接取得され，ソースプログラムが生成される。したがって，修飾名は，プログラム定義の [入出力項目設定] ダイアログには表示されない。
- ATTR=ELEMENT が指定されると，XML 文書中のタグ名情報を取得する。
- 修飾名を指定して修飾名用 ATTR を指定しなかった場合，および修飾名に ATTR=ITEM を指定した場合は，プログラム定義の [入出力項目設定] ダイアログの指示項目欄に修飾名が表示される。ATTR にデータ定義識別子以外を設定しても構文エラーとはならない。プログラム生成時は，プログラム定義の [入出力項目設定] ダイアログで指定した文字列に変換される。

(b) ARRAY_MAX

- 可変記号および修飾名に反復がある場合，反復の最大数を指定する。
- ATTR に PARTS が指定されている場合，ARRAY_MAX は指定できない。

(c) COMMENT

- 可変記号および修飾名に対する注釈を記述する。
- ここで記述した注釈は，プログラム定義の定義ウィンドウやダイアログで「説明」欄に表示される。

(d) REF

- ATTR に ITEM が指定されている場合だけ有効である。
- テンプレート中でデータ定義種別が指定されている可変記号を指定する。
- プログラム定義で指定したすべてのデータ定義の項目から選択する場合，＊を指定する。

(e) IO

- ATTR にデータ定義種別，または XMAP3 が指定されている場合だけ有効である。
- データ定義，およびマップ定義ファイルの入出力の区分を指定する。
- ここで定義した入出力の区分は，プログラム定義のメインウィンドウの「入出力」に表示される。

(f) 修飾名

- 可変記号を修飾する名称を指定する。
- 可変記号名と同じ規則で記述する。
- テンプレートの展開処理記述部に記述する場合は，可変記号の添字として記述する（可変記号名［修飾名］）。

(g) VALUE

- ATTR の指定がない指示項目にだけ指定できる。
- プログラム定義の指示項目設定では，VALUE に指定した値の中から選択できる。

(h) PARSE_LEVEL

- XML 文書を解析するレベルを指定する。省略した場合は，PARSE_LEVEL=1 を仮定

する。

解析レベル	意味
1	文書インスタンスだけを解析し，XML 文書をテンプレートから参照できるようにする。
2	解析レベル 1 に加えて，DTD※を解析し，DTD で指定されたデフォルト値を挿入する。ただし，文書インスタンスと DTD の整合性は検証しない。

注※
DTD については，「7.8.1(2) DTD」を参照してください。

(i) TAG_NAME

- ATTR に ELEMENT が指定されている場合は必ず指定する。ATTR に ELEMENT が指定されている場合だけ有効である。
- XML 文書のタグ名を指定する。

(j) ATTR_NAME

- ATTR に ELEMENT が指定されている場合は必ず指定する。ATTR に ELEMENT が指定されている場合だけ有効である。
- XML 文書の属性名を指定する。

(2) データ定義識別子と内容

プログラム作成者が，データ定義で定義しておく必要のある項目を修飾名として指定し，ATTR= のあとにその項目をデータ定義識別子で記述します。

データ定義識別子の指定によって，データ定義で定義した内容をテンプレート上で使用できます。修飾名用 ATTR に指定できるデータ定義識別子の一覧を表 7-1 に示します。

表 7-1 データ定義識別子の一覧

データ定義種別	データ定義識別子	データ定義での定義個所	内　　容
FILE	FILE_NAME	ファイル名	SELECT 句，FD 句のファイル名。 データ定義ウィンドウの [ファイル] タブのファイル名フィールドの指定値。
	ORGANIZATION	ファイル編成	ORGANIZATION 句の編成。 データ定義ウィンドウの [ファイル] タブのファイル編成の指示番号。 ［1：順編成］［2：相対編成］ ［3：索引編成］［4：索引順編成］ ［5：テキスト編成］［6：CSV 形式］ ［7：直接編成］

データ定義種別	データ定義識別子	データ定義での定義個所	内　　容
	RECORD_NAME	レコード定義※1	FD 句のレコード記述項にある，最初の 01 レベルのデータ項目名。 データ定義ウィンドウの [ファイル] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※2。
	KEY_NAME	キー名	プログラムで使用するキー名。 データ定義ウィンドウの [ファイル] タブのキー名フィールドの指定値。
	RECORDING_MODE	レコード形式	レコードの形式。 データ定義ウィンドウの [ファイル] タブのレコード形式フィールドの指示番号。 ［0: 指定なし］［1: 固定長］［2: 可変長］
	FILE_EXTERNAL	EXTERNAL	外部属性の有無の指定。 データ定義ウィンドウの [ファイル] タブの EXTERNAL チェックボックスの指示番号。 ［0: チェック OFF］［1: チェック ON］
	VSIZE_ITEM	長さ設定エリア	レコードの長さを設定する項目の名称。 データ定義ウィンドウの [ファイル] タブの長さ設定エリア名フィールドの指定値。
DB	TABLE_NAME	表名称	表名称。 データ定義ウィンドウの [RDB] タブの表名称フィールドの指定値。
	RECORD_NAME	レコード定義	表に対応するテーブルにある，01 レベルのデータ項目名。 データ定義ウィンドウの [RDB] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※2。
DAM	FILE_NAME	ファイル定義	DAM ファイル名。 データ定義ウィンドウの [DAM] タブのファイル名フィールドの指定値。
	BLOCK_SIZE	入出力ブロックサイズ	アクセスするブロックの大きさ。 データ定義ウィンドウの [DAM] タブの入出力ブロックサイズの指定値。
	RECORD_NAME	レコード定義	DAM アクセス時の入出力領域にある，01 レベルのデータ項目名。 データ定義ウィンドウの [DAM] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※2。
TAM	TABLE_NAME	テーブル定義	TAM テーブル名。 データ定義ウィンドウの [TAM] タブのテーブル名称フィールドの指定値。

データ定義種別	データ定義識別子	データ定義での定義個所	内　　容
	KEY_SIZE	キー名称	キー項目のサイズ。 データ定義ウィンドウの [TAM] タブのキーフィールドに指定したデータ項目のサイズ。
	RECORD_NAME	レコード定義	TAM アクセス時の入出力領域にある，01 レベルのデータ項目名。 データ定義ウィンドウの [TAM] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※ 2。
RPC 入力パラメタ	RECORD_NAME	レコード定義	入力パラメタの 02 レベルのデータ項目名。 データ定義ウィンドウの [RPC] タブの入力パラメタ定義の結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※ 2。
RPC 応答領域	RECORD_NAME	レコード定義	応答領域の 02 レベルのデータ項目名。 データ定義ウィンドウの [RPC] タブの応答領域定義の結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※ 2。
メッセージ	RECORD_NAME	レコード定義	入力メッセージの 02 レベルのデータ項目名。 データ定義ウィンドウの [メッセージ] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※ 2。
ユーザジャーナル	USER_JOURNAL_CODE	コード	ユーザジャーナルのコード。 データ定義ウィンドウの [ユーザジャーナル] タブのコードフィールドの指定値。
	RECORD_NAME	レコード定義	ユーザジャーナルアクセス時の出力領域にある，02 レベルのデータ項目名。 データ定義ウィンドウの [ユーザジャーナル] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※ 2。
メッセージログ	RECORD_NAME	レコード定義	メッセージログアクセス時の出力領域にある 02 レベルのデータ項目名。 データ定義ウィンドウの [メッセージログ] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義の最初に定義さているデータ項目の名前※ 2。
共通作業領域	RECORD_NAME	レコード定義※ 1	共通作業領域の最初の 01 レベルのデータ項目名。 データ定義ウィンドウの [共通作業領域] タブの結合項目名称 / レコード定義名称フィールドの結合項目の指定値，またはレコード定義ファイルの最初に定義さているデータ項目の名前※ 2。

注※ 1

データ定義で結合項目またはレコード定義ファイルが複数指定されている場合，次の点を考慮してください。

- データ定義で選択されている結合項目またはレコード定義ファイルごとにレコードを生成する場合は，配列の要素に結合項目またはレコード定義ファイルの位置を指定します。なお，配列の要素は，修飾名の次に書きます。
- 結合項目またはレコード定義ファイルの位置には，データ定義の結合項目名称 / レコード定義名称のリストに表示される結合項目の順序を，数値で指定します。
  ただし，データ定義に表示されるすべての結合項目またはレコード定義ファイルのレコード名を生成する場合は，結合項目またはレコード定義ファイルの位置を省略します。

（例）@ 入力ファイルの 2 番目のレコード名を参照する。

```
@@interface  @ 入力ファイル ={ATTR = FILE,
                              レコード名 =
                              {ATTR=RECORD_NAME}};
        :
@ 入力ファイル [ レコード名 ,2]
```

注※ 2

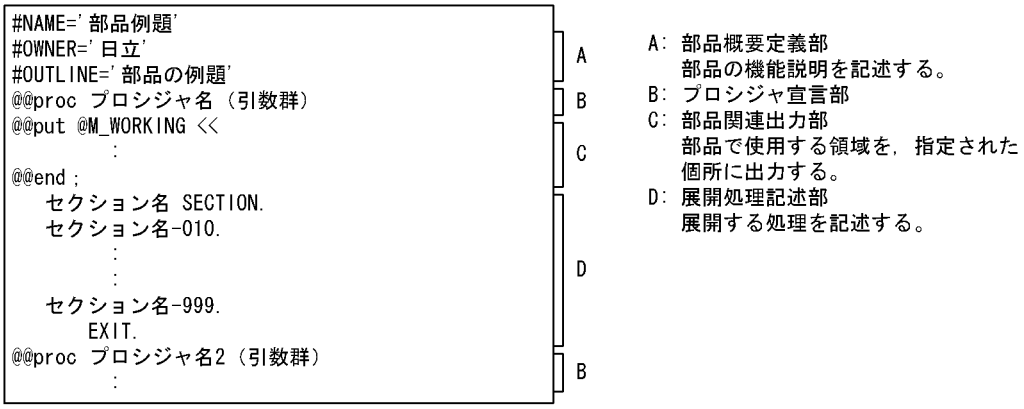
名前が指定されていない場合はデータ項目名になります。

7.6 部品定義と部品の呼び出し

7.6.1 部品定義

(1) 部品の構成

部品は図のような構成で記述します。



(2) 部品概要定義部

テンプレート概要定義部と同じように記述します（「7.5.2 テンプレート概要定義部」を参照）。

(3) プロシジャ宣言部

一つの部品の中にプロシジャは複数宣言できます。

(a) プロシジャ名

テンプレートに展開したいプロシジャの名称を指定します。プロシジャ名称は次の規則に従って記述します。

- 特殊文字は使用できない。
- 名称の長さは 30 文字以下であること。
- 英字の大文字と小文字は区別される。
- 数字だけの指定はできない。

(b) 引数

部品の展開内容を変更するために，テンプレートからプロシジャに引き渡す変数です。引数には可変記号だけが指定できます。

(4) 部品関連出力部

部品で特有の作業領域や，部品で展開されるセクションの呼び出し部分の展開などが必要な場合，@@merge 文，および @@put 文の記述規則に従って定義します。

(5) 展開処理記述部

展開する処理を記述します。

引数やグローバル宣言された可変記号を使用することで，各種の文および関数を使うことができます。

7.6.2 部品の使用方法

部品を使用する方法は，次の二つの場合で異なります。

- テンプレート作成時に使用する部品が確定している場合
- テンプレート作成時に使用する部品が確定していない場合

(1) テンプレート作成時に使用する部品が確定している場合

テンプレート作成時に AP 中で使用する部品が確定している場合は，テンプレート中に部品名と引数を指定して，部品を呼び出します。詳細は「7.6.3　部品の呼び出し」を参照してください。

(2) テンプレート作成時に使用する部品が確定していない場合

テンプレート作成時に AP 中で使用する部品が確定していない場合は，@@interface 文を使用して，プログラム定義の [部品] タブで，プログラム作成者に部品を選ばせます。プログラム定義の [部品] タブでは，プログラム作成者が AP 中で使用する部品ファイルや，プロシジャ，および引数を指定します。この方法でのテンプレート作成については，「7.5.3　インタフェース定義部」および「7.9.39　@@parts 文」を参照してください。

7.6.3 部品の呼び出し

(1) 部品の呼び出し方法

部品はテンプレート，業務ルールのルールスクリプトまたは部品中から呼び出せます。

形　式

@@ ＜部品ファイル名＞プロシジャ名（[引数 1 [，引数 2…]]）

規　則

- 部品ファイル名には，パスの指定もできる。
- プロシジャ名は，「7.6.1(3)(a) プロシジャ名」の規則に従う。
- 部品を使用する場合は，必ず環境設定で部品の検索パスを指定しておかなければならない。

- 引数には，可変記号を指定する。
- 引数の数は，部品の呼び出しと部品の定義で一致しなければならない。
- 引数の可変記号には配列を指定できる。部分参照は指定できない。
- 部品はネストして呼び出すことができる。また，リカーシブルに自分自身を呼び出すこともできる。
- 同じ部品ファィル中の部品（プロシジャ名）を呼び出す場合，部品ファイル名を省略できる。

(2) 部品ファイルの検索方法

テンプレート，業務ルールのルールスクリプトまたは部品中に指定された部品ファイルは「環境設定」で指定された検索パスに従って検索されます。

同一名称の部品（部品ファイル，プロシジャ名）が存在する場合は，検索パス中で先に見つかったものが呼び出されます。

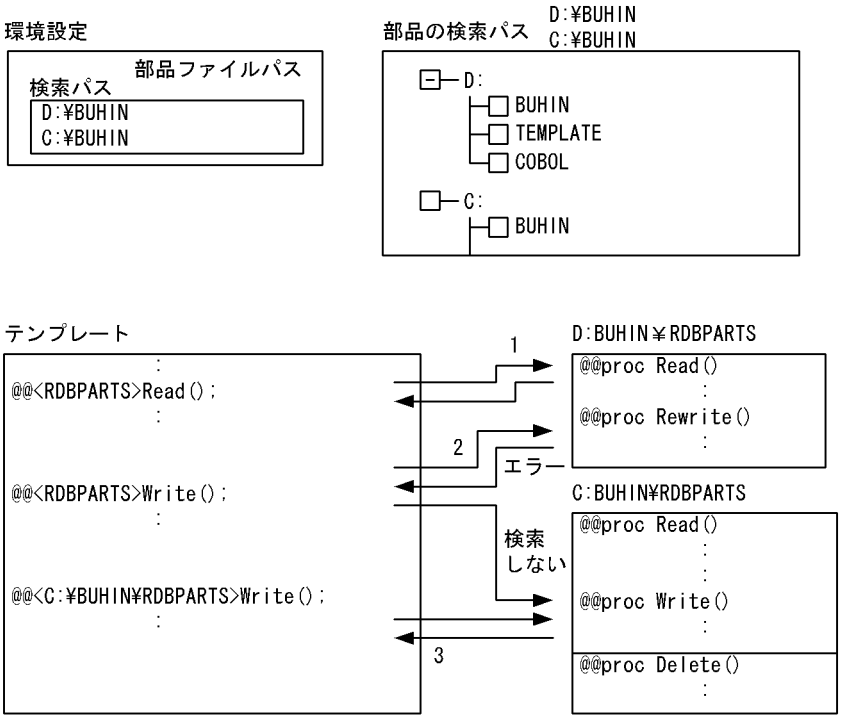
部品の検索をネットワークドライブのディレクトリパスや，ローカルファイルのディレクトリパスに設定することができます。

部品を検索するパスを指定する方法として以下の 3 種類があります。

1. SEWB+/REPOSITORY から常に参照する（部品の開発が完了している）場合
   リポジトリのネットワークドライブ名を指定します。
2. リポジトリまたはプログラム開発中のディレクトリから参照する（部品の開発が並行して進んでいる）場合
   リポジトリのネットワークドライブ名とカレント（プログラム定義やテンプレート）のディレクトリ名を指定します。
3. 開発中のディレクトリを参照する（開発の初期段階の）場合
   カレント（プログラム定義やテンプレート）のディレクトリ名だけを指定します。

注

パスについては，「付録 A　環境設定」を参照してください。



1. D:¥BUHINとC:¥BUHINの両方に，同じRDBPARTSでReadというプロシジャ名がある場合，先に検索された方が有効となる。
2. D:¥BUHINとC:¥BUHINで同じRDBPARTSという部品ファイルがあるが， D:¥BUHINにはWriteというプロシジャ名がないためエラーとなる。 C:¥BUHINのRDBPARTSの検索はしない。
3. 同一名称の部品ファイルがある場合，テンプレート中にはディレクトリを含めて部品ファイル名がユニークになるように指定する。

7.7　業務ルールの利用

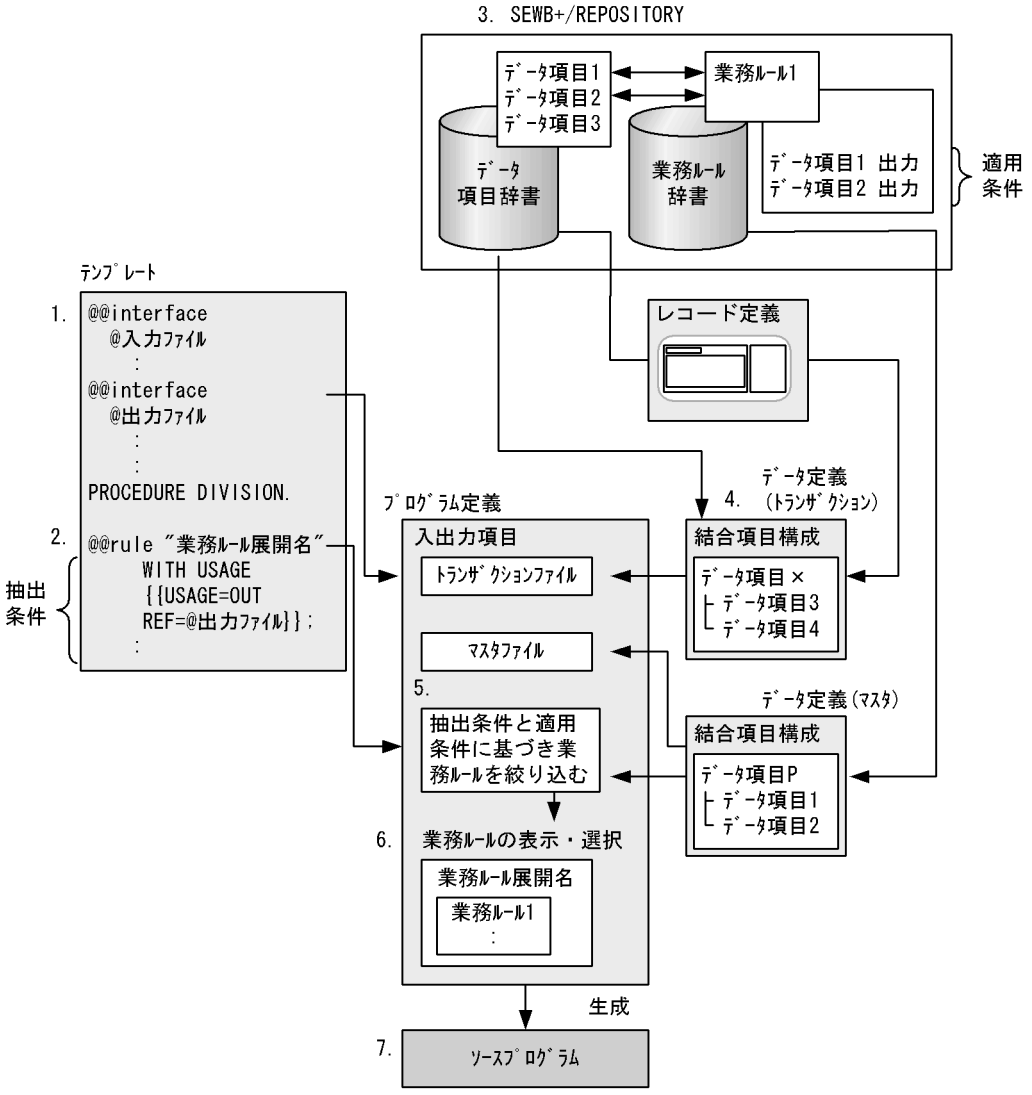
7.7.1　テンプレート記述と業務ルール

SEWB+/REPOSITORY で業務ルールと関連づけられているデータ項目を含むレコードを，SEWB+/CONSTRUCTION の @@interface 文で指定すると，業務ルールを利用できます。業務ルールとはデータ項目に着目して，チェック条件や編集処理など，データ項目特有の処理を汎用的なルールとして業務ルール辞書に格納したものです。そのため，業務ルールを利用すると，それぞれのデータ項目に対する処理をユーザが書く手間が省けます。業務ルールをソースプログラム上に展開させたい場合は，テンプレート中に @@rule 文を書きます。

なお，業務ルールの記述にテンプレート記述言語を使用できます。使用できるテンプレート記述言語の詳細は，「7.9.1　文と関数一覧」を参照してください。

テンプレート記述と業務ルールの関係を図 7-1 と図 7-2 に示します。

図 7-1 SEWB+/CONSTRUCTION での業務ルールの仕組み（データ定義を使用した場合）



1. データ定義種別を指定した可変記号の宣言
   作成するプログラムで使用するデータ定義種別の可変記号を，@@interface 文で宣言します。
2. 業務ルール使用の宣言
   テンプレートだけでは不足している情報は，ユーザが追加します（ユーザ処理）。
   ユーザ処理には，UOC を書く場合と，業務ルールを使用する場合とがあります。業務ルールを使用する場合は，テンプレートのユーザ処理の位置に @@rule 文を書きます。@@rule 文には「業務ルール展開名」と「抽出条件」を書きます。「抽出条件」には，業務ルールを抽出するための条件と，その条件の対象になるデータ項目が含まれ

ているデータ定義種別の可変記号を書きます。
UOC を書く場合については，「7.9.49　@@uoc 文」を参照してください。

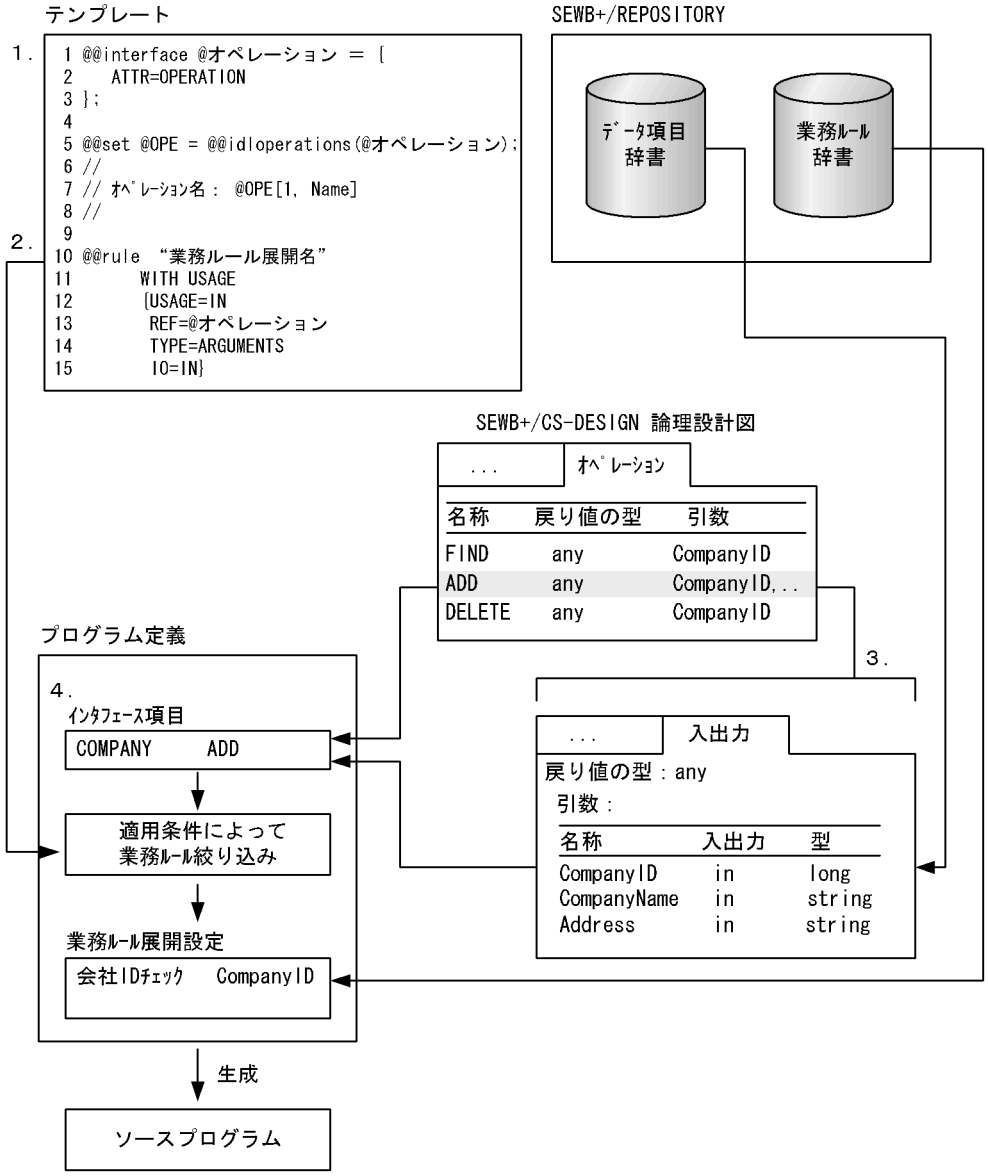
3. データ項目と業務ルールの定義（SEWB+/REPOSITORY）
   プログラムで使用するデータ項目と業務ルールが定義されています。また，これらの間の関連や，業務ルールの適用条件も定義されています。適用条件とは，ルールスクリプト※中のデータ項目がどのような条件（処理の入力，出力）であるときに，その業務ルールを適用させるかという情報です。

   注※
   　ルールスクリプトとは，業務ルールの処理内容のことです。
4. データ定義
   データ定義では，プログラムで使用するファイル，DB およびクライアントとサーバ間の通信インタフェースを定義します。データ定義でレコードを使用するときは，SEWB+/REPOSITORY で定義された最上位結合項目，または SEWB+/RECORD DEFINER で作成されたレコード定義を参照して定義されます。
5. 業務ルールの抽出と絞り込み
   業務ルールの抽出では，テンプレートの @@rule 文に書かれている抽出条件の「IN」「OUT」が，業務ルール辞書で定義されている適用条件「入力」「出力」と照会され，条件が一致する業務ルールが SEWB+/CONSTRUCTION のプログラム定義画面に表示されます。
   このような，適用条件と抽出条件による業務ルールの抽出を，「業務ルールの絞り込み」と言います。
   図 7-1 では，マスタファイルのレコードを構成するデータ項目（データ項目 1，2）は，業務ルール 1 と関連づけられています。また，@@rule 文の抽出条件と業務ルールの適用条件が一致しています（3. 参照）。したがって，@@rule 文にマスタファイル名を指定すると，抽出条件に従って，業務ルール 1 が抽出されます。
   適用条件の詳細は，マニュアル「SEWB+/REPOSITORY　辞書設計ガイド」を参照してください。
6. 業務ルールの表示と選択
   抽出された業務ルールは，プログラム定義の業務ルール展開設定画面に表示されます。プログラム作成者は，その中からソースプログラムに展開させたい業務ルールを選びます。
7. ソースプログラムの生成
   プログラム定義で選択された業務ルールが，ソースプログラム中に展開されます。

業務ルールを抽出する条件の対象に，論理設計図に定義されているデータ項目を指定することもできます。

テンプレート記述と業務ルール，論理設計図の関係を図 7-2 に示します。

図 7-2 SEWB+/CONSTRUCTION での業務ルールの仕組み（論理設計図を使用した場合）



1. 論理設計図を指定した可変記号の宣言
   作成するプログラムで使用する論理設計図の可変記号を，@@interface 文で宣言します。
2. 業務ルール使用の宣言
   @@rule 文には「業務ルール展開名」と「抽出条件」を書きます。「抽出条件」の REF 句には，可変記号を書きます。
   論理設計図のインタフェース定義情報，およびオペレーション定義情報で定義されて

いる特定のデータ項目を選択する場合には，TYPE 句を指定します。インタフェース定義情報では，「属性」のデータ項目，オペレーション定義情報では「引数」，「例外」，および「戻り値の型」のデータ項目を選択できます。

3. 論理設計図
   SEWB+/CS-DESIGN でオブジェクトのインタフェース情報や，オペレーション情報を定義します。業務ルールを使用する場合，SEWB+/REPOSITORY のデータ項目を使用します。
   論理設計図は，ユーザ定義型のようにデータ項目を使用しないで定義することもできますが，この場合は業務ルール検索の対象とはなりません。
   データ項目とユーザ定義型を混在して使用した場合は，データ項目だけが業務ルール検索の対象となります。
4. 業務ルールの抽出と絞り込み
   データ定義を指定した場合は，データ定義に含まれるデータ項目すべてが業務ルール検索の対象となりますが，論理設計図を指定した場合は「属性」，「引数」，「例外」，または「戻り値の型」のどれに定義されたデータ項目から業務ルールの抽出を行なうかを指定できます。
   この点を除いて，論理設計図を使用した場合も，業務ルール抽出と絞り込みの仕組みは図 7-1 のデータ定義を使用した場合と同じです。

7.7.2 業務ルールの展開と抽出条件

プログラム作成者が適切な業務ルールを効率良く確実に探せるように，テンプレート作成者は @@rule 文にコメントを書いたり，業務ルールを検索する条件を定義したりします。

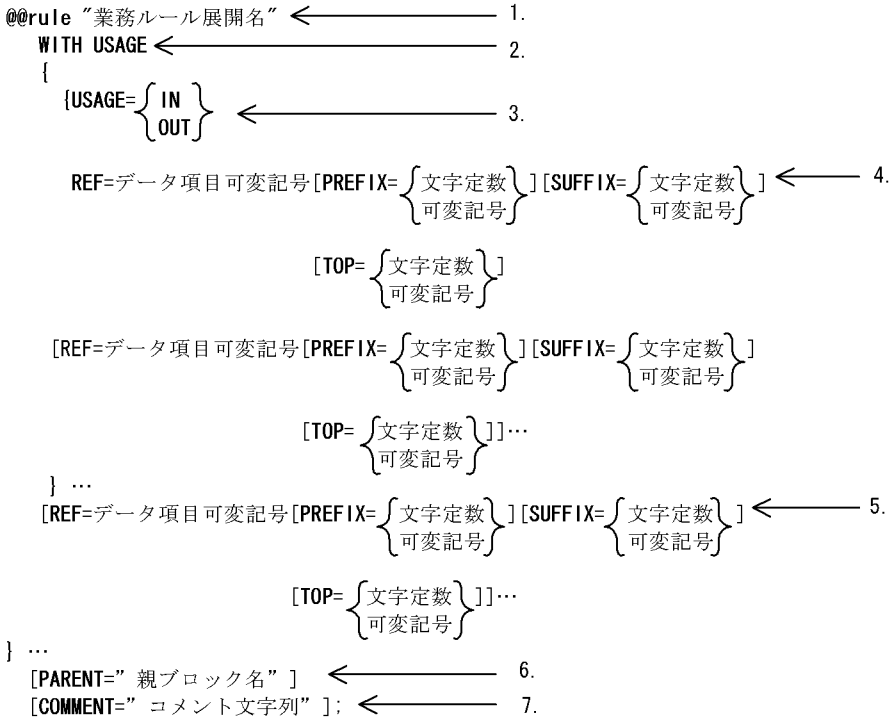
また，プログラム上の連続しない個所への定義が必要な場合は，業務ルールを分割して展開させることができます。

(1) @@rule 文の記述規則

ここでは，抽出条件を指定して，抽出される業務ルールを限定する @@rule 文の書き方，抽出条件を指定しないで広い範囲から業務ルールを抽出する @@rule 文の書き方，およびそれらの両方を合わせた @@rule 文の書き方を示します。

形　式１　－抽出条件を指定する書き方

抽出条件を指定する書き方です。業務ルール辞書で定義されている適用条件と合わせて業務ルールの絞り込みをするので，抽出される業務ルールは限定されます。

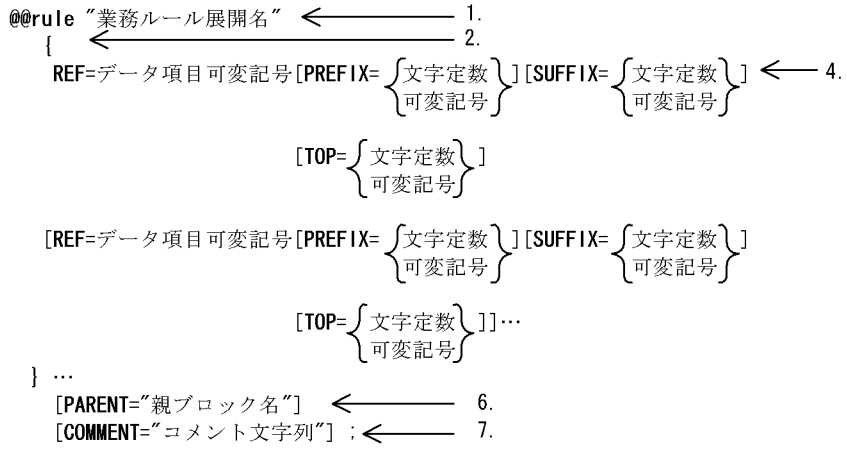


注

太字の文字は表記どおりに記述することを表します。{　} の中に，縦に複数の項目がある場合は，その中のどれか一つを選んで記述します。[] に囲まれた項目は省略できることを表します。形式中の数字は，「規則」の説明文と対応しています。

形　式2　－抽出条件を省略する書き方

抽出条件を省略する書き方です。抽出される業務ルールの範囲は，形式１の書き方より広くなります。

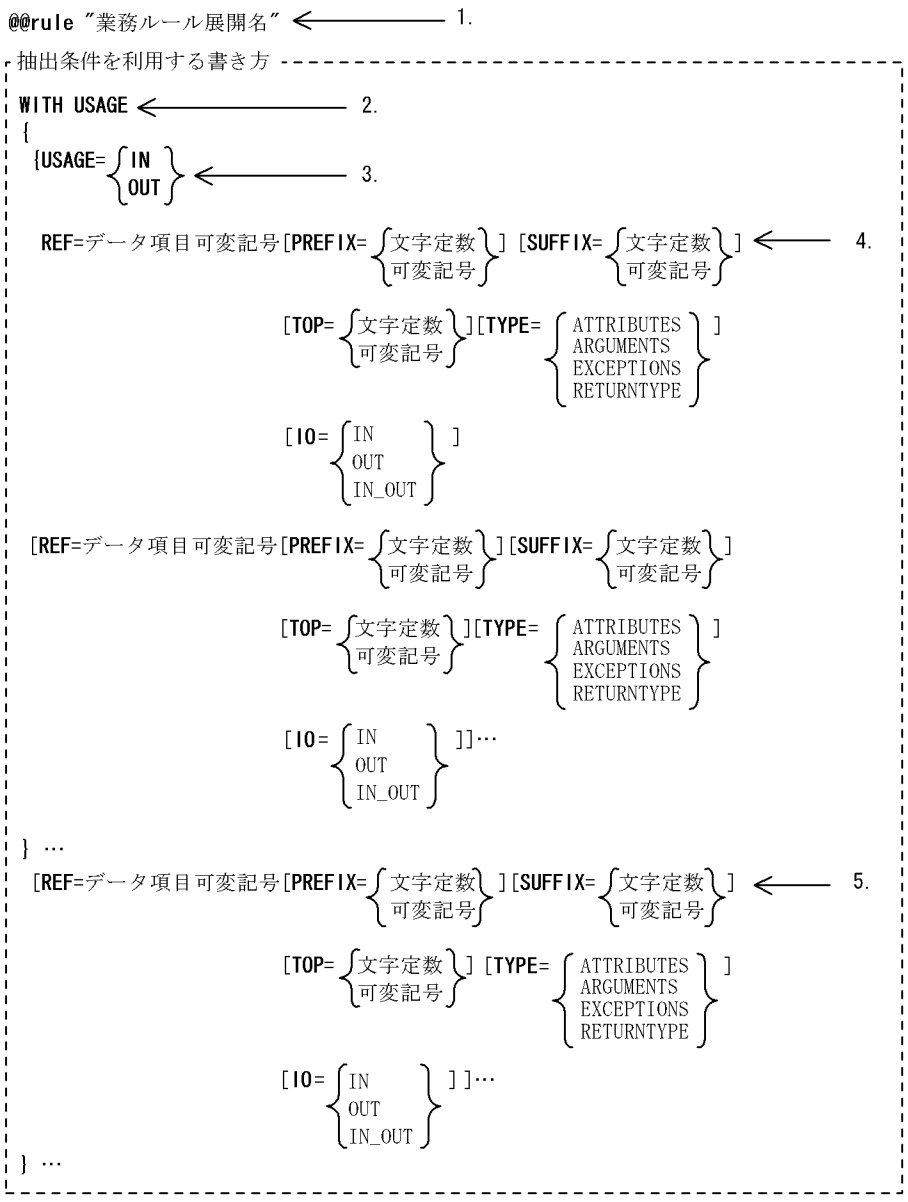


注

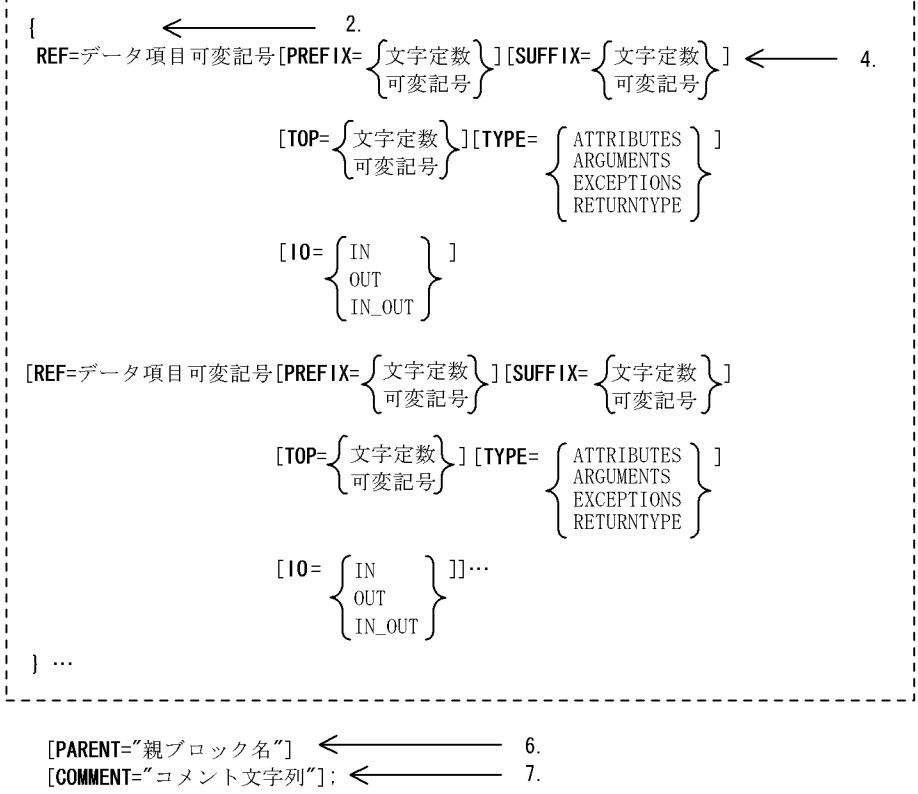
太字の文字は表記どおりに記述することを表します。{　} の中に，縦に複数の項目がある場合は，その中のどれか一つを選んで記述します。[ ] に囲まれた項目は省略できることを表します。形式中の数字は，「規則」の説明文と対応しています。

形　式 3　－限定された抽出と幅広い抽出を同時に行う書き方

形式１のように，抽出条件を利用して業務ルールを抽出する書き方と，形式２のように抽出条件を利用しないで業務ルールを抽出する書き方を混在させて書くことができます。



抽出条件を利用しない書き方



注

太字の文字は表記どおりに記述することを表します。{ } の中に，縦に複数の項目がある場合は，その中のどれか一つを選んで記述します。[ ] に囲まれた項目は省略できることを表します。形式中の数字は，「規則」の説明文と対応しています。

規 則

(a) 業務ルール展開名

1.@@rule " 業務ルール展開名 "

業務ルール展開名は，文字定数または可変記号で指定します。ここで指定した名前は，プログラム定義画面の [ユーザ処理] タブに表示されます。

(b) 抽出条件

2.WITH USAGE{ 抽出条件 }

- **抽出条件を指定する場合**

  業務ルールを抽出する条件を { } で囲んで記述します。この括弧は，形式の一部なので必ず書いてください。

- **抽出条件を省略する場合**

  抽出条件を省略する場合，WITH USAGE 句は書きません。抽出条件を省略した場合，業務ルールで使用されているデータ項目が，REF 句に書いたデータ定義に

含まれていることが，抽出の条件になります。

3. {USAGE=｛IN / OUT｝REF 句…}

業務ルール辞書で定義されている適用条件の中で，どの適用条件の業務ルールを抽出するのかを指定します。
「IN」を指定した場合，適用条件が「入力」である業務ルールが抽出されます。
「OUT」を指定した場合，適用条件が「出力」である業務ルールが抽出されます。
REF 句の指定までを{ }で囲んで記述します。この{ }は，形式の一部なので必ず書いてください。

4. REF=可変記号[PREFIX=｛文字定数 / 可変記号｝][SUFFIX=｛文字定数 / 可変記号｝]
[TOP=｛文字定数 / 可変記号｝]

データ定義種別，オブジェクト定義（ATTR=INTERFACE），またはオペレーション定義（ATTR=OPERATION）の可変記号を書きます。

- PREFIX 句

  ルールスクリプト中のキーワード※がデータ項目に置き換えられる際に付けられる接頭語を，文字定数または可変記号で指定します。
  接頭語はルールスクリプト中のキーワード（@MODIFY）で展開される修飾付きデータ項目にも付けられます。
  REF に繰り返しのある（ARRAY_MAX 指定のある）可変記号を指定する場合，PREFIX に指定する可変記号を配列にすると，繰り返しごとの接頭語を指定できます。

- SUFFIX 句

  ルールスクリプト中のキーワード※がデータ項目に置き換えられる際に付けられる接尾語を，文字定数または可変記号で指定します。
  接尾語はルールスクリプト中のキーワード（@MODIFY）で展開される修飾付きデータ項目にも付けられます。
  REF に繰り返しのある（ARRAY_MAX 指定のある）可変記号を指定する場合，SUFFIX に指定する可変記号を配列にすると，繰り返しごとの接尾語を指定できます。

- TOP 句

  ルールスクリプト中のキーワード（@MODIFY）で展開される修飾付きデータ項目のうち最も上位となるデータ項目名を別の文字列に変換する場合，その文字列を文字定数または可変記号で最上位項目名に指定します。

REF に繰り返しのある（ARRAY_MAX 指定のある）可変記号を指定する場合，TOP に指定する可変記号を配列にすると，繰り返しごとの最上位項目名を指定できます。

注※
業務ルールのキーワードの詳細は，マニュアル「SEWB+/REPOSITORY　辞書設計ガイド」を参照してください。

可変記号および修飾名に反復がある場合（@@interface 文の ARRAY_MAX に値が指定されている場合），PREFIX，SUFFIX または TOP には，@@set 文で作成した反復の可変記号を指定します。

（例）

- @@interface 文の内容

```
@@interface @FILE01 = {
            ATTR=FILE,
            ARRAY_MAX=3,
            COMMENT="ファイルを指定してください"
       };
```

- 反復の可変記号の作成

```
@@set @AFIX[1] = "I1-";
@@set @AFIX[2] = "I2-";
@@set @AFIX[3] = "I3-";
```

- @@rule 文での指定

```
@@rule "業務ルール展開名"{ REF=@FILE01 PREFIX=@AFIX };
```

- **TYPE 句**

REF 句にオブジェクト定義またはオペレーション定義を指定した場合に，指定できます。オブジェクト定義またはオペレーション定義から特定の項目を引き当てる場合に指定します。REF 句にオブジェクト定義を指定した場合は，属性（ATTRIBUTES）が指定できます。REF 句にオペレーション定義を指定した場合は，引数（ARGUMENTS），例外（EXCEPTIONS）または戻り値の型（RETURNTYPE）が指定できます。
指定しない場合は，REF 句に指定したオブジェクト定義またはオペレーション定義のすべての項目が対象となります。

- **IO 句**

REF 句にオペレーション定義を指定し，TYPE 句に引数（ARGUMENTS）を指定した場合に指定できます。オペレーション定義の引数の入出力区分を指定してデータ項目を引き当てる場合に指定します。
「IN」を指定した場合，入出力区分が「in」である引数が引き当てられます。
「OUT」を指定した場合，入出力区分が「out」である引数が引き当てられます。
「IN_OUT」を指定した場合，入出力区分が「inout」である引数が引き当てられます。
指定しない場合は，すべての引数が対象となります。

(c) 抽出補助項目

5.USAGE 句の{}外の REF 句

(b) の抽出条件での指定内容を補助するために指定します。USAGE 句の{}内に書かれた抽出条件に合っている業務ルールが抽出される場合，その業務ルールで使用されているデータ項目すべてが，USAGE 句の{ }内の REF 句に書かれたデータ定義のデータ項目であれば，その業務ルールは抽出されます。
しかし，業務ルールで使用されているデータ項目の中に，USAGE 句の{}内の REF 句に書かれたデータ定義には含まれていないデータ項目や，入出力の指定が違うものがある場合は，その業務ルールは抽出されません。
このようなとき，抽出補助項目を書いておけば，その REF 句に書かれた可変記号が示すデータ定義中のデータ項目を無条件に使用できるので，業務ルールを抽出できます。なお，{ }外の REF 句に書く可変記号は，データ定義種別の可変記号として @@interface 文で宣言されていなければなりません。

(例)

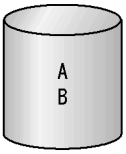
テンプレート中に，入れ替え処理の業務ルールを展開させたい場合，テンプレート作成者は次のような文を書きます。

```
@@rule "入れ替え処理"
 WITH  USAGE { {USAGE=IN  REF=@ファイル1}
               {USAGE=OUT  REF=@ファイル1}};
```

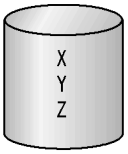
ファイルの内容と業務ルールおよび SEWB+/REPOSITORY の適用条件は次のようになっています。業務ルールのルールスクリプト中では，汎用性を持たせるため先頭に「@」が付けられた文字列（キーワード）が使用されます。

●ファイルの内容

ファイル1



ファイル2



●ルールスクリプトの内容

業務ルール01

```
      :
MOVE @A TO @X
MOVE @B TO @A
      :
```

業務ルール02

```
      :
MOVE @A TO @WA
MOVE @B TO @A
      :
```

●SEWB+/REPOSITORYでの適用条件定義の内容

業務ルール01

データ項目	キーワード	適用条件
A	@A	入力
A	@A	出力
B	@B	入力

業務ルール02

データ項目	キーワード	適用条件※
A	@A	-
B	@B	入力

注※ SEWB+/REPOSITORYで適用条件が指定されていない(ーの状態)データ項目は，入力項目として使用されていても，出力項目として使用されていてもかまいません。

このようなとき，テンプレートの @@rule 文の REF 句に指定されているデータ定義中のデータ項目を使用している業務ルールとして，業務ルール 01 および 02 が引き

当てられます。ここまでは，これらの業務ルールは @@rule 文の抽出条件に合っています。しかし，それぞれの業務ルールの中には，「X」と「WA」という，ファイル 1 には指定されていないデータ項目が含まれています。そこで，上の @@rule 文の最後に抽出補助項目「REF=@ ファイル 2」を追加してみます。

```
@@rule "入れ替え処理"
  WITH USAGE {{USAGE=IN REF=@ファイル1}{ USAGE=
              OUT REF=@ファイル1}  REF=@ファイル2};
```

このようにすると，業務ルール 01 中の「X」が抽出補助項目の REF 句に指定されたファイル 2 のデータ項目として指定されるので，業務ルール 01 で使用されているデータ項目が，すべて @@rule 文で指定されたことになります。したがって，この @@rule 文では業務ルール 01 が抽出されます。ファイル 2 のデータ項目に対して，どのように使用されているかという抽出条件は指定されていないので，データ項目の入出力は意識されません。

（d）抽出条件以外の指定

6.PARENT　" 親ブロック名 "

PARENT 句には，親ブロック名を指定します。親ブロック名について次に示します。

- 親ブロック名は，プログラム定義の [ユーザ処理] タブで業務ルールの一覧をツリー形式で表示させるときに，指定した業務ルール展開名の親になるブロック名を指定する。
- 親ブロック名は，可変記号で指定することができる。
- ループ内で @@rule 文を繰り返し実行する場合には，業務ルール名の値を変化させて，それぞれの業務ルール展開名を区別する。
- 親ブロック名を指定すると，ここで指定した業務ルール展開名は，この親ブロックの下に位置づけられる。
- 親ブロック名を省略すると，ツリーの最上位の階層に位置づけられる。ただし，@@rule 文が @@parts 文から展開されている場合，@@parts 文のブロックの下に位置づけられる。
- 親ブロック名が存在しなかった場合，ツリーの最上位の階層に位置づけられる。
- プログラム定義で，業務ルールが選択されていない場合，ソースプログラムへは展開されない。

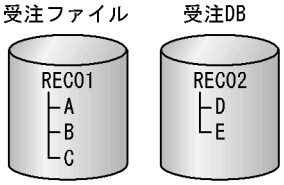
7.COMMENT　" コメント文字列 "

プログラム作成者が，適切な業務ルールを選択できるような注釈を書きます。ここに書いた注釈は，プログラム定義画面の [ユーザ処理] タブに表示されます。

使用例

SEWB+/REPOSITORY のデータ項目辞書の内容，プログラムで使用するファイル中のデータ項目の内容およびテンプレートの @@rule 文の内容を照らし合わせながら，業務ルールを抽出する仕組みを説明します。

プログラムで使用するファイルの内容



データ項目と業務ルールの関連

受注ファイル（@入力ファイル）

└ REC01

業務ルール / データ項目	S01	S02	S03	S04
A	i	i	i	o
B		o	o	i
C	-			

受注DB（@Xファイル）

└ REC02

業務ルール / データ項目	S01	S02	S03	S04
D			ー	
E				ー

（凡例）i　：適用条件が「入力」である。
o　：適用条件が「出力」である。
ー　：適用条件が定義されていない。
空欄：使用されていない。

業務ルールのルールスクリプト

業務ルールS01

```
     :
IF FLG_1 = 1
  THEN
   MOVE @A TO ERR_F
  ELSE
   PERFORM SUB_P
   UNTIL @C > 10
     :
```

業務ルールS02

```
     :
IF FLG_1 > 1
  THEN
    MOVE @A TO @B
  ELSE
    PERFORM ･･･
     :
```

業務ルールS03

```
     :
IF FLG_1 = 1
  THEN MOVE @A TO @B
  ELSE MOVE @D TO @B
     :
     :
```

業務ルールS04

```
     :
IF FLG_1 > 1
  THEN MOVE @B TO @A
  ELSE
    COMPUTE
      @E = @E - 1
     :
```

テンプレート中の @@rule 文の内容と抽出される業務ルールの関係

（例 1）

- **テンプレートの @@rule 文**

```
 @@rule "GYOMU1"
   WITH USAGE
  {{USAGE=IN
   REF=@入力ファイル}};
```

- **@@rule 文の見方**

```
@@rule  "GYOMU1"……このルールの名前。
 WITH USAGE……{  }の記述を抽出条件であると
                宣言する。
{{USAGE=IN ……適用条件が「入力」でデータ項
                目と関連づけられている業務ル
                ールを抽出する。
REF=@入力ファイル}};…業務ルール中で使用されてい
                るデータ項目は，@入力ファイ
                ルのデータ定義に含まれてい
                るものである。
```

- **例** 1 **の @@rule 文の抽出状況**

業務ルール	抽出状況	理　　由
S01	○	@ 入力ファイルのデータ項目「A」と適用条件「入力」で関連づけられている。また，適用条件が定義されていない「C」は @ 入力ファイルに格納されている。

業務ルール	抽出状況	理　由
S02	×	@入力ファイルのデータ項目「A」と適用条件「入力」で関連づけられているが，「B」と適用条件「出力」で関連づけられている。
S03	×	@入力ファイルのデータ項目「B」と適用条件「出力」で関連づけられている。また，S03 では@入力ファイルにないデータ項目「D」が使用されている。
S04	×	@入力ファイルのデータ項目「A」と適用条件「出力」で関連づけられている。また，@入力ファイルにはないデータ項目「E」が使用されている。

（凡例）
○：抽出される。
×：抽出されない。

（例 2）

- **テンプレートの @@rule 文**

```
@@rule "GYOMU2"
  WITH USAGE
 {{USAGE=IN REF=@入力ファイル}
 {USAGE=OUT REF=@入力ファイル}};
```

- **@@rule 文の見方**

```
@@rule "GYOMU2"……このルールの名前。
 WITH USAGE……{  }の記述を抽出条件であると
                宣言する。
{{USAGE=IN REF=@入力ファイル}
          ……適用条件が「入力」でデータ項
                目と関連づけられている業務ル
                ールを抽出する。データ項目は,
                @入力ファイルのデータ定義に
                含まれているものである。
{USAGE=OUT REF=@入力ファイル}};
          ……適用条件が「出力」でデータ項
                目と関連づけられている業務ル
                ールを抽出する。データ項目は,
                @入力ファイルのデータ定義に
                含まれているものである。
```

- **例 2 の @@rule 文の抽出状況**

業務ルール	抽出状況	理　由
S01	×	@入力ファイルのデータ項目「A」と適用条件「入力」で関連づけられているが，適用条件「出力」で関連づけられているデータ項目がない。
S02	○	@入力ファイルのデータ項目「A」「B」と適用条件「入力」および「出力」で関連づけられている。

業務ルール	抽出状況	理　　由
S03	×	@入力ファイルのデータ項目「A」「B」と適用条件「入力」および「出力」で関連づけられているが，S03 では@入力ファイルにないデータ項目「D」が使用されている。
S04	×	@入力ファイルのデータ項目「A」「B」と適用条件「入力」および「出力」で関連づけられているが，S04 では@入力ファイルにないデータ項目「E」が使用されている。

（凡例）
○：抽出される。
×：抽出されない。

（例 3）

- **テンプレートの @@rule 文**

```
@@rule "GYOMU3"
  WITH USAGE
 {{USAGE=IN REF=@入力ファイル}
  {USAGE=OUT REF=@入力ファイル}
  REF=@Xファイル} ;
```

- **@@rule 文の見方**

```
@@rule "GYOMU3"……このルールの名前。
 WITH USAGE……{  }の記述を抽出条件であると宣
               言する。
{{USAGE=IN REF=@入力ファイル}
        ……適用条件が「入力」でデータ項目
               と関連づけられている業務ルール
               を抽出する。データ項目は，@入力
               ファイルのデータ定義に含まれて
               いるものである。
{USAGE=OUT REF=@入力ファイル}
           ……適用条件が「出力」でデータ項目
               と関連づけられている業務ルール
               を抽出する。データ項目は，@入力
               ファイルのデータ定義に含まれてい
               るものである。
REF=@Xファイル};…{  }内のWITH USAGE句の定義を
               補助する。
```

- **例 3 の @@rule 文の抽出状況**

業務ルール	抽出状況	理　　由
S01	×	@入力ファイルのデータ項目「A」と適用条件「入力」で関連づけられているが，適用条件「出力」で関連づけられているデータ項目がない。
S02	○	@入力ファイルのデータ項目「A」「B」と適用条件「入力」および「出力」で関連づけられている。

業務ルール	抽出状況	理　　由
S03	○	@ 入力ファイルのデータ項目「A」「B」と適用条件「入力」および「出力」で関連づけられている。S03 では @ 入力ファイルにないデータ項目「D」も使用されているが，抽出補助項目として書かれている @X ファイル中に「D」が格納されている。
S04	○	@ 入力ファイルのデータ項目「A」「B」と適用条件「入力」および「出力」で関連づけられている。S04 では @ 入力ファイルにないデータ項目「E」も使用されているが，抽出補助項目として書かれている @X ファイル中に「E」が格納されている。

（凡例）
○：抽出される。
×：抽出されない。

（例 4）

- **テンプレートの @@rule 文**

```
@@rule "GYOMU4"
 { REF=@入力ファイル};
```

- **@@rule 文の見方**

```
@@rule "GYOMU4"……このルールの名前。
{ REF=@入力ファイル};…抽出条件は，業務ルールの中
                       で，@入力ファイルのデータ
                       項目が使用されているかどう
                       かという点だけ。
```

- **例 4 の @@rule 文の抽出状況**

業務ルール	抽出状況	理　　由
S01	○	適用条件を意識しないデータ項目「A」「C」が使用されている。
S02	○	適用条件を意識しないデータ項目「A」「B」が使用されている。
S03	×	適用条件を意識しないデータ項目「A」「B」が使用されているが，S03 では @ 入力ファイルにないデータ項目「D」が使用されている。※
S04	×	適用条件を意識しないデータ項目「A」「B」が使用されているが，S04 では @ 入力ファイルにないデータ項目「E」が使用されている。※

（凡例）
○：抽出される。
×：抽出されない。

注※

「D」および「E」を持つファイルを抽出補助項目として指定すると，すべての業務ルールが抽出されます。

7.7.3　業務ルールの分割展開

業務ルールは必要に応じて，プログラムのコード上，連続していない複数の個所へ展開できます。業務ルールを分割して展開させるには，@@section 文と @@merge 文を使用します。@@section 文は業務ルール中に，@@merge 文はテンプレート中に書きます。なお，@@section 文の詳細は，マニュアル「SEWB+/REPOSITORY　辞書設計ガイド」を参照してください。

（1）@@section 文と @@merge 文との関係

業務ルールの定義内容は，通常は，テンプレートで @@rule 文が書かれている個所に展開されます。ただし，業務ルールの中に @@section 文が書かれている場合は，その部分が，テンプレート中の @@merge 文が書かれている個所に展開されます。業務ルールでの記述とテンプレートでの記述，および展開先の場所の関係を表 7-2 に示します。

表 7-2　展開場所ごとの業務ルールの記述

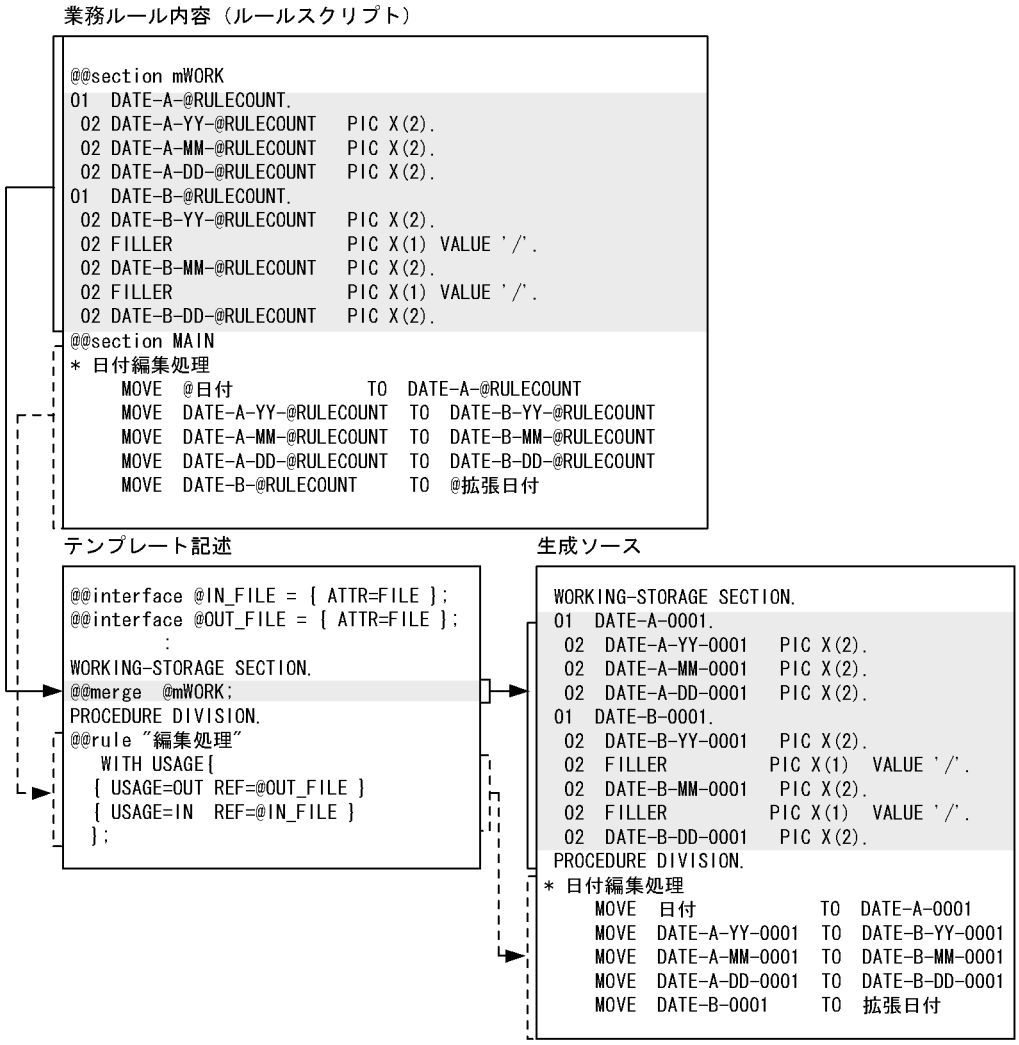
業務ルールの記述	テンプレートでの記述	展開場所
• @@section　MAIN • 業務ルールに @@section を書かない	@@rule" 業務ルール展開名 " が書かれている	@@rule 文を書いた場所
• @@section　コード名	@@merge　可変記号が書かれている 注 @@section 文のコード名と @@merge 文の可変記号が一致している※	@@merge 文を書いた場所

注※

可変記号は先頭に「@」が付けられるので，「@ コード名」となります。可変記号の先頭から「@」を削除した部分と，@@section 文のコード名が同一ならば，これらは一致しているとみなされます。

表 7-2 に示されているような展開の仕組みを利用して，一つの業務ルールを分割して展開させることができます。次に分割展開の例を示します。

（例）



（凡例）

実線：@@merge 文の位置に展開される場合を示しています。

@@section 文には，テンプレート中に書かれた @@merge 文に指定されている可変記号の，「@」以降の文字列と同じコード名を書きます。

可変記号については，「(2) 可変記号を指定するときの注意事項」を参照してください。

@@section 文が展開される範囲は，別の @@section 文が現れるまで有効です。

破線：@@rule 文の位置に展開される場合を示しています。

@@section MAIN を書くと，テンプレート中の @@rule 文が書かれた位置に，@@section 文の内容が展開されます。@@section 文が展開される範囲は，別の

@@section 文が現れるまで有効です。

(2) 可変記号を指定するときの注意事項

業務ルールを分割展開させるときには，業務ルール中の「@@section 文 コード名」のコード名と，テンプレート中の「@@merge 文 可変記号」の先頭の「@」を除いた部分が一致していなければなりません。このため，テンプレート作成者とデータ項目辞書作成者は，どこにどのような可変記号とコード名を使うかを，あらかじめ決めておく必要があります。

(3) @@section 文の制限事項

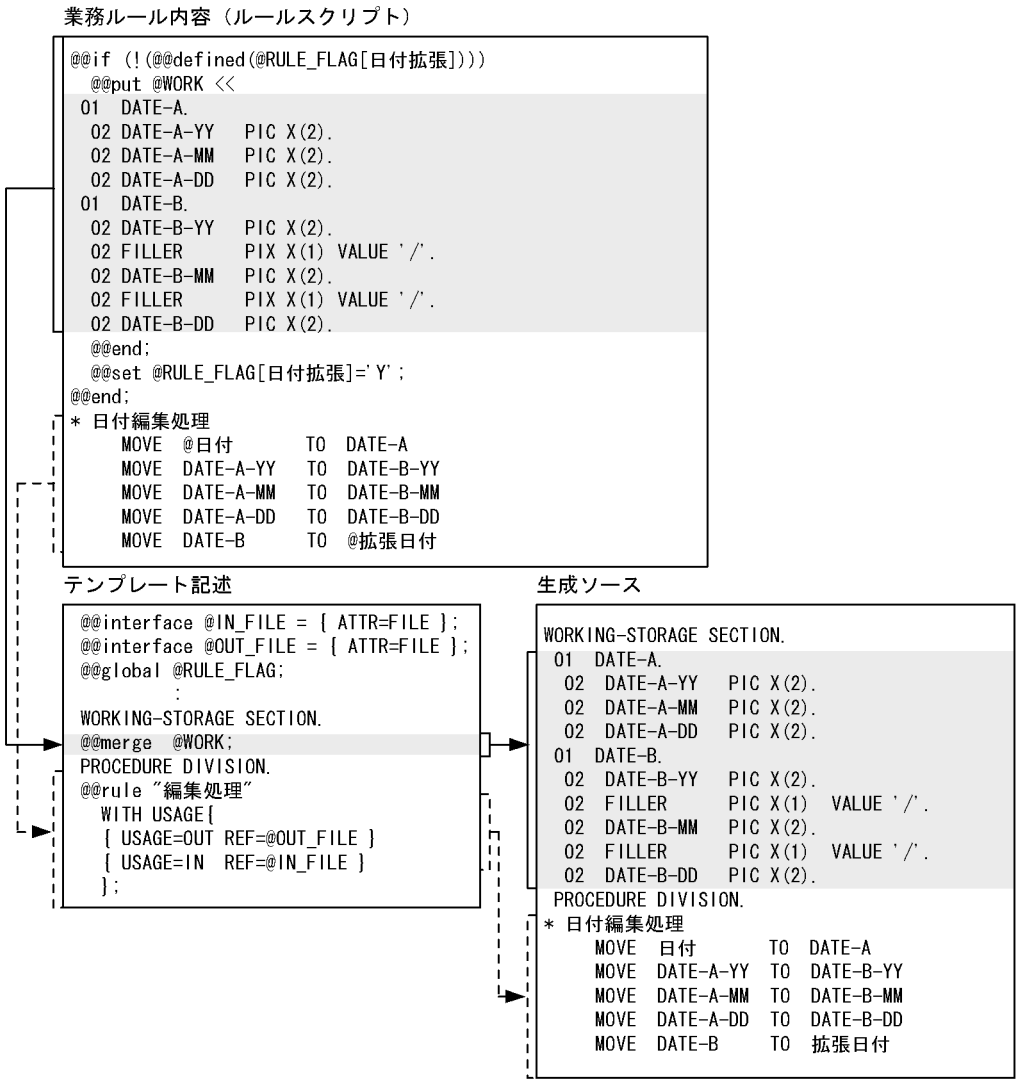
業務ルール中に @@section 文を使用した場合，@@section 文以降には展開制御文を使用できません。

7.7.4 展開制御文を使用した業務ルールの展開

業務ルールのルールスクリプトに展開制御文を使用できます。展開制御文を使用すると，さまざまな業務ルールを作成できます。

例えば，業務ルールを分割して展開させるには，@@section 文と @@merge 文を使用する方法以外に，展開制御文を使用する方法もあります。「7.7.3　業務ルールの分割展開」の例での業務ルール内容を，@@section 文を使用しないで @@put 文で記述した例を次に示します。

（例 1）



（凡例）

実線：@@merge 文の位置に展開される場合を示しています。

破線：@@rule 文の位置に展開される場合を示しています。

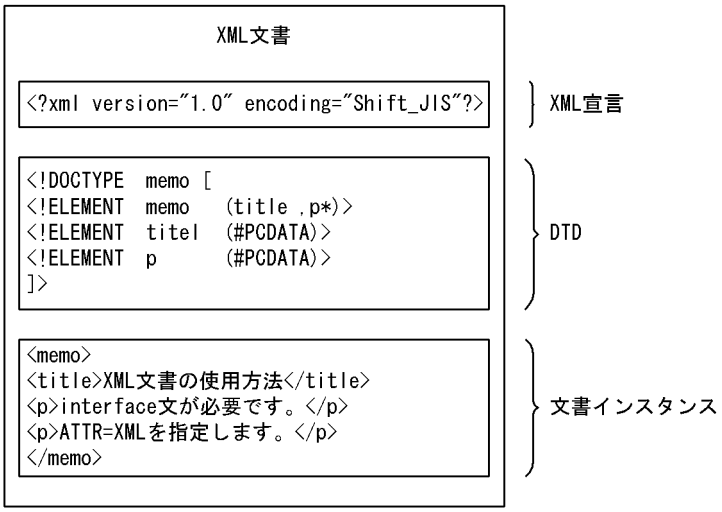
7.8 XML 文書の利用

XML には，タグを定義しデータ構造を階層化して表現できるという特徴があります。SEWB+/CONSTRUCTION では，WorkCoordinator Definer で作成したビジネスプロセス定義の XML で記述された文書を，プログラム生成のためのパラメタとして利用できます。WorkCoordinator Definer については，「WorkCoordinator Definer Version 3　ユーザーズガイド」を参照してください。

7.8.1 XML 文書とは

XML 文書は，XML 宣言，DTD，および文書インスタンスで構成されています。

テンプレートでは，文書インスタンスの内容を参照できます。XML 宣言や DTD の内容は参照できません。



(1) ＸＭＬ 宣言

XML 宣言では，XML のバージョンや文字コードを宣言します。XML のバージョンを宣言する場合，version="1.0" を指定します。XML 文書を SJIS で作成している場合，encoding="Shift_JIS" を指定する必要があります。なお，XML 宣言は省略できます。

(2) DTD

DTD では，要素の型や属性リストを宣言します。DTD の扱いは，@@interface 文で指定する解析レベル（PARSE_LEVEL）によって異なります。なお，DTD は省略できます。

解析レベル	意味
1	文書インスタンスだけを解析し，XML 文書をテンプレートから参照できるようにする。
2	解析レベル 1 に加えて，DTD を解析し，DTD で指定されたデフォルト値を挿入する。ただし，文書インスタンスと DTD の整合性は検証しない。

注 1

XML 文書の解析時間は，解析レベル 1 が短く，解析レベル 2 が長くなります。

注 2

@@interface 文で解析レベルを指定しない場合は，1 が仮定されます。

(3) 文書インスタンス

文書インスタンスは，タグ付きの文書で記述されます。この内容は，テンプレートに XML 関数を記述することで参照できます。また，解析レベル 2 を指定すると，XML 文書が持っている DTD のデフォルト値が入っている内容も参照できます。

7.8.2 XML 文書の構造

タグを付けたデータを階層化して表現する XML 文書は，テンプレートで参照する場合，木構造で表現されます。木構造は**ノード**で表現され，ノード同士が親子関係や包含関係を結んで形成します。情報を検索するときは，この木構造をたどります。テンプレートから参照する場合の XML 文書の階層構造の考え方を次に示します。

(1) ノードの種類

XML 文書のタグ，タグの属性およびタグを付けられたテキストをそれぞれノードとして扱います。ノードには次の 3 種類があります。

- 要素ノード
  XML 文書のタグに対応し，タグ名で表現されます。なお，ここでは要素ノードを次のような記号で表します。

- 属性ノード
  XML の属性リストに対応し，属性名および属性値で表現されます。なお，ここでは属性値を次のような記号で表します。

- テキストノード
  XML 文書の開始タグと終了タグで囲まれたテキストに対応します。なお，ここではテキストノードを次のような記号で表します。

：テキストノード

(2) ノード同士の関係

ノード同士の関係を次に示します。

(a) 親子関係（要素ノードと要素ノード，または要素ノードとテキストノード）

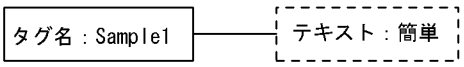
要素ノードと要素ノード，または要素ノードとテキストノードは，親と子供の関係で表現します。親子関係の例を次に示します。この場合，要素ノードが「親ノード」，テキストノードが「子供ノード」となります。

（例 1）

ＸＭＬ文書

```
<Sample1>簡単<Sample1/>
```

木構造での表現



（凡例） ――― ：親子関係

タグ名'Sample1'を持つ要素ノードが作成され，その子供ノードとして'簡単'という値を持つテキストノードが作成される。

(b) 包含関係（要素ノードと属性ノード）

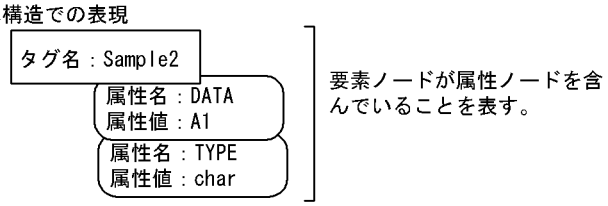
要素ノードと属性ノードは包含関係で表現します。包含関係の例を次に示します。

（例 2）

ＸＭＬ文書

```
<Sample2
   DATA="A1"
   TYPE="char"/>
```

木構造での表現



Sample2の要素ノードには，二つの属性ノードが作成される。一つは，属性名'DATA'と属性値'A1'を持ち，もう一つは，属性名'TYPE'と属性値'char'を持つ。要素ノードと属性ノードには親子関係はない。

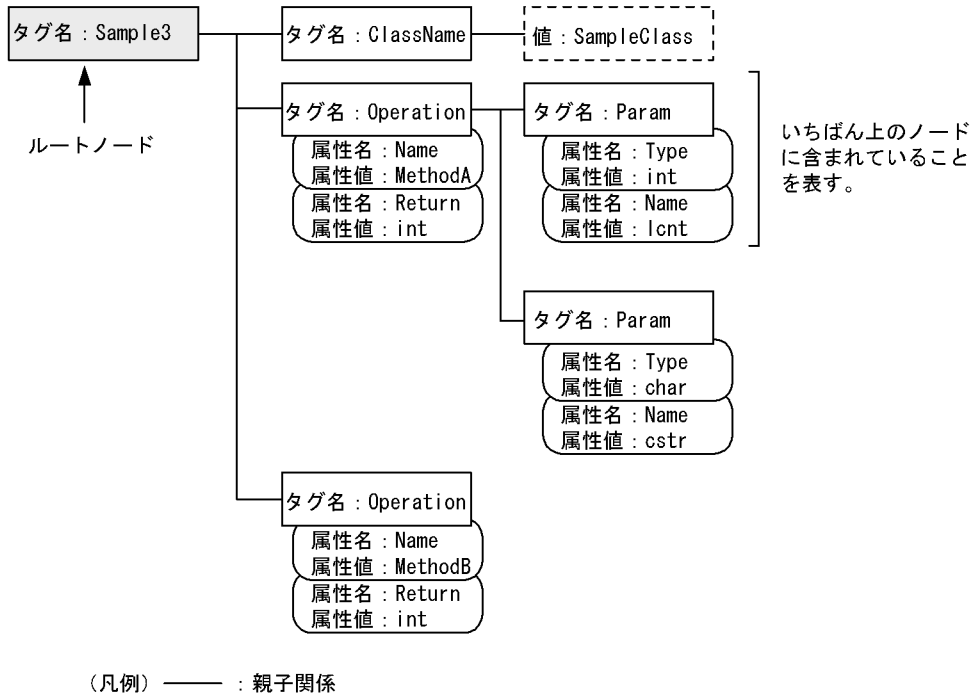
(c) ノード構造の例

ノードは，親子関係と包含関係で構成されています。ノード構造の例を次に示します。なお，構造中で最上位に位置するノードをルートノードと呼びます。テンプレートからXML 文書を参照する場合，必ずルートノードから参照を始めます。

（例 3）親子関係と包含関係の混在

```
ＸＭＬ文書
 <Sample3>
  <ClassName>SampleClass</ClassName>
  <Operatin  Name = "MethodA"
             Return = "int">
     <Param Type = "int"
            Name = "lcnt"/>
     <Param Type = "char"
            Name = "cstr"/>
  </Operation>
  <Operation Name = "MethodB"
             Return = "int">
  </Operation>
 </Sample3>
```

木構造での表現



Sample3要素ノードの下に，ClassName要素ノード，Operation要素ノードおよびParam要素ノードが作成され，タグの階層構造に従った木構造が作成される。

7.8.3　XML 文書の参照方法

テンプレートで XML 文書を参照する方法を説明します。

(1) ルートノードの設定

テンプレートで XML 文書を参照する場合，必ずルートノードから参照を始めることになります。テンプレートでは ATTR=XML を指定した @@interface 文の可変記号に，ルートノードが設定されます。

(例) 可変記号 @XmlDoc にルートノードを設定する。

```
@@interface @XmlDoc = {ATTR=XML};
```

(2) ノードの内容の参照

可変記号に設定されたノードの内容は，可変記号に添字を指定して参照します。ノードの種類によっては参照される情報が異なります。なお，ノードの種類はキーワードの Type で判断できます。

ノードの種類	添字に指定するキーワード		
	Type	Name	Value
要素ノード	2※1	タグ名※2	長さが 0 の文字列が設定される。
属性ノード	3※1	属性名※2	属性値※2
テキストノード	6※1	#text※1	テキスト※2

注※ 1
固有の値が参照されます。
注※ 2XML 文書で指定された値が参照されます。

(例) ルートタグ名をメッセージウィンドウに表示する。

```
@@interface @XmlDoc = {ATTR=XML};
@@msg "ルートのタグ名は @XmlDoc[NAME] です。";
```

(3) XML 文書の木構造をたどる

XML 関数を使用すると，木構造をたどって目的のノード情報を参照できます。

項番	関数名	機能	取得されるノード		
			要素	テキスト	属性
1	@@xmlparent（可変記号）	親ノード情報を取得する。	○	×	×
2	@@xmlchildnodes（可変記号）	すべての子供ノードの情報を取得する。この関数は複数のノードを配列で返す。	○	○	×
3	@@xmlfirstchild（可変記号）	先頭の子供のノード情報を取得する。	○	○	×

項番	関数名	機能	取得されるノード		
			要素	テキスト	属性
4	@@xmllastchild（可変記号）	最後の子供ノード情報を取得する。	○	○	×
5	@@xmlprevious（可変記号）	同じ親を持つ直前のノード情報を取得する。	○	○	×
6	@@xmlnext（可変記号）	同じ親を持つ直後のノード情報を取得する。	○	○	×
7	@@xmlelements（可変記号，" タグ名 "）	指定されたタグ名を持つ下位のすべての要素ノードの情報を取得する。この関数は，複数のノードを配列で返す。タグ名に '*' を指定した場合，すべての要素ノードを返す。	○	×	×
8	@@xmlattribute（可変記号，" 属性名 "）	指定された属性名を持つ属性ノード情報を取得する。属性名に '*' は指定できない。	×	×	○

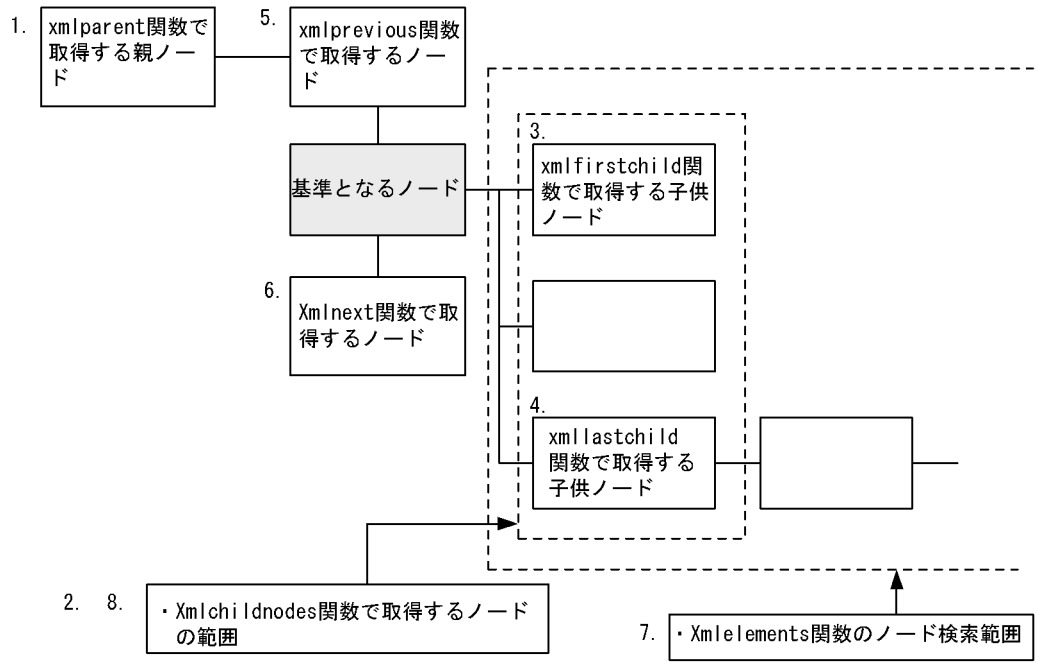
（凡例）

○：取得できる。

×：取得できない。

これらの関数を展開したとき，「基準となるノード」を起点にして，どのように木構造が検索されて，ノード情報が取得されるかを次の図に示します。

なお，図中の番号は「(3) XML 文書の木構造をたどる」の表の項番と対応しています。



7.8.4 SEWB+/CONSTRUCTION で取り込むことができる XML ファイル

SEWB+/CONSTRUCTION で使用できる XML ファイルの条件を次に示します。

(1) ファイル形式が確定し，公開されている

テンプレートを作成するためには，取り込む XML ファイルの形式（タグ名称，属性名など），構造を知っておく必要があります。タグ名称，属性名が不明または誤っていると，取り込み部分の選択が正しく表示されなかったり，基点となる属性ノードが定まらないで，生成するときに属性エラーになったりします。また，取り込み内容を展開するコーディングが XML ファイルの構造に一致していないと，生成するときに，構文エラーになります。

(2) 構文にエラーがない

SEWB+/CONSTRUCTION は，取り込む XML ファイルに対してファイルを解析するため，未完成の XML ファイルなど，構文にエラーのある XML ファイルを取り込むことができません。XML ファイルの選択で，構文にエラーのある XML ファイルを選択すると，エラーダイアログが表示されます（エラー詳細は，エラーファイルに出力されます）。

7.9 文と関数

7.9.1 文と関数一覧

テンプレート，部品および業務ルールの記述で使用できる文と関数の一覧を表 7-3 に示します。

表 7-3 文と関数の一覧

項番	文および関数	内 容	使用できる場所		
			テンプレート	部品	業務ルール
1	@@break；	ループ処理からの脱出	○	○	○
2	@@columnlist（可変記号）	列名を配列の値として取得	○	○	○
3	@@continue；	ループ処理ジャンプ	○	○	○
4	@@count（可変記号）	配列個数参照	○	○	○
5	@@defined（可変記号）	未定義値の判定	○	○	○
6	@@diagram "ブロック名 " [PARENT=" 親ブロック名 "] [COMMENT=" 説明文字列 "]；	ユーザ追加処理および業務ルール処理の一覧をツリー形式で表示	○	△	×
7	@@errorexit；	プログラム生成の中止	○	○	○
8	@@expand 可変記号 [修飾名 [, 位置]] [PREFIX= 文字定数] [SUFFIX= 文字定数] [START= 開始値] [UP= 増分値] [START_POSITION= 開始位置] [{VALUE ｜ NOVALUE}] ；	データ定義のレコードの展開	○	○	○
9	@@file 文 "ブロック名 " OUTPUT_NAME=" 出力ファイル名 " [PARENT=" 親ブロック名 "] [COMMENT=" 説明文字列 "]；	生成ファイルを指定	○	×	×
10	@@foreach 可変記号 1 （可変記号 2） 文 @@end；	リスト処理	○	○	○

項番	文および関数	内　容	使用できる場所		
			テンプレート	部品	業務ルール
11	@@getdata(可変記号 [{," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR}])	可変記号のソース生成属性を取得	○	○	○
12	@@getdate ()	日付と時刻を取得	○	○	○
13	@@getmemo ()	[メモ] タブの情報を取得	○	○	○
14	@@getrfile ()	リソースステータスの [ファイル] タブの情報を取得	○	○	○
15	@@getsign ()	[サイン] タブの情報を取得	○	○	○
16	@@global　可変記号 ;	グローバル宣言	○	○	○
17	@@idlargudata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])	SEWB+/ CS-DESIGN オブジェクト定義のオペレーション引数情報の引数名からソース生成属性を取得	○	○	○
18	@@idlarguments（可変記号）	SEWB+/ CS-DESIGN オブジェクト定義のオペレーション引数情報を取得	○	○	○
19	@@idlattrdata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])	SEWB+/ CS-DESIGN オブジェクト定義の属性情報の属性名からソース生成属性を取得	○	○	○
20	@@idlattributes（可変記号）	SEWB+/ CS-DESIGN オブジェクト定義の属性情報を取得	○	○	○

項番	文および関数	内　容	使用できる場所		
			テンプレート	部品	業務ルール
21	@@idlexcepdata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])	SEWB+/CS-DESIGN オブジェクト定義のオペレーション例外情報の例外名からソース生成属性を取得	○	○	○
22	@@idlexceptions（可変記号）	SEWB+/CS-DESIGN オブジェクト定義のオペレーション例外情報を取得	○	○	○
23	@@idlinterface（可変記号）	SEWB+/CS-DESIGN オブジェクト定義情報を取得	○	○	○
24	@@idlinterfaces（可変記号）	SEWB+/CS-DESIGN オブジェクト定義情報をまとめて取得	○	○	○
25	@@idlopedata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])	SEWB+/CS-DESIGN オブジェクト定義のオペレーション情報の戻り値からソース生成属性を取得	○	○	○
26	@@idloperations（可変記号）	SEWB+/CS-DESIGN オブジェクト定義のオペレーション情報を取得	○	○	○
27	@@idlreqcontext（可変記号）	SEWB+/CS-DESIGN オブジェクト定義のリクエスト・コンテキスト情報を取得	○	○	○

項番	文および関数	内　容	使用できる場所		
			テンプレート	部品	業務ルール
28	@@if （条件式1） 　文1 [[@@elseif （条件式2） 　文2],…] [@@else 　文3] @@end;	判定	○	○	○
29	@@interface 可変記号 {};	データ定義，プログラム定義とのインタフェースの指定	○	×	×
30	@@itemlist（可変記号）	列名の埋め込み変数を配列の値として取得	○	○	○
31	@@lang { COBOL | C | "言語種別" } [FOR_REPOSITORY = "リポジトリの言語区分"] [EXTENSION = "拡張子"] [MODIFY_CONNECT = "連結文字"] [MODIFY_ORDER = { DESCEND | ASCEND}] [UOC_BEGIN= "ユーザ追加処理の先頭に挿入する文字列"] [UOC_END = "ユーザ追加処理の末尾に挿入する文字列"];	生成言語種別の指定	○	×	×
32	@@leadbyte（可変記号）	2バイトコードのリードバイトの判定	○	○	○
33	@@length（可変記号）	文字列長を文字数で求める	○	○	○
34	@@lengthb（可変記号）	文字列長をバイト数で求める	○	○	○
35	@@merge 可変記号；	可変記号マージ	○	○	○
36	@@modify(可変記号 [,[接頭語] [,[接尾語][,[最上位項目名]]]])	修飾情報付きデータ項目名展開	○	○	○
37	@@msg 文字定数；	メッセージ出力	○	○	○
38	@@parts"ブロック名" 　REF=可変記号 　[PARENT="親ブロック名"] 　[COMMENT="説明文字列"]	プログラム定義で指定した部品の挿入位置	○	×	×

項番	文および関数	内　容	使用できる場所		
			テンプレート	部品	業務ルール
39	@@pic（可変記号）	PICTURE 文字列の取得	○	○	○
40	@@proc　プロシジャ名 （[引数 [COMMENT=" 引数の説明 "]]…） 　文	部品内プロシジャ宣言	×	○	×
41	@@put 可変記号 << 　文 @@end;	テキストの設定	○	○	○
42	@@reclen（可変記号［修飾名］）	レコード長の取得	○	○	○
43	@@rule " 業務ルール展開名 " WITH USAGE { {USAGE= {IN ｜ OUT} REF= データ項目可変記号 [PREFIX= { 文字定数 ｜ 可変記号 }] [SUFFIX= { 文字定数 ｜ 可変記号 }] [TOP= { 文字定数 ｜ 可変記号 }] [TYPE= {ATTRIBUTES ｜ ARGUMENTS ｜ EXCEPTIONS ｜ RETURNTYPE }] [IO= {IN ｜ OUT ｜ IN_OUT}]… }… }… [PARENT=" 親ブロック名 "] [COMMENT=" コメント文字列 "]; … }… [PARENT=" 親ブロック名 "] [COMMENT=" コメント文字列 "];	業務ルールの抽出および展開位置の指定	○	△	×
44	@@set 可変記号 = 代入式 ;	値設定	○	○	○
45	@@str（可変記号, 先頭位置［, 文字列長］）	文字単位での可変記号の部分参照	○	○	○
46	@@strb（可変記号）	バイト単位での可変記号の部分参照	○	○	○

項番	文および関数	内　容	使用できる場所		
			テンプレート	部品	業務ルール
47	@@switch（可変記号） @@case 値 1: 　文 1 [[@@case 値 2: 　[文 2]]…] [@@default: 　文 3] @@end;	選択	○	○	○
48	@@uoc "UOC 名 " [PARENT=" 親ブロック名 "] [COMMENT=" 説明文字列 "] [SUPPRESS];	ユーザ追加処理の位置指定	○	△	×
49	@@uocdefined （"UOC 名 "）	ユーザ追加処理がプログラム定義で入力されているか判定	○	○	○
50	@@while（条件式） 　文 @@end;	ループ	○	○	○
51	@@xmap3common（可変記号）	マップ定義の共通情報を取得	○	○	○
52	@@xmap3objects（可変記号）	マップ定義のオブジェクト情報を取得	○	○	○
53	@@xmlattribute（可変記号, " 属性名 "）	属性名をキーにした属性ノードを取得	○	○	○
54	@@xmlchildnodes（可変記号）	すべての子供ノードを取得	○	○	○
55	@@xmlelements（可変記号, " タグ名 "）	タグ名をキーにした要素ノードを取得	○	○	○
56	@@xmlfirstchild（可変記号）	最初の子供ノードを取得	○	○	○
57	@@xmllastchild（可変記号）	最後の子供ノードを取得	○	○	○
58	@@xmlnext（可変記号）	同じ親を持つ直後のノードを取得	○	○	○
59	@@xmlparent（可変記号）	親ノードを取得	○	○	○

項番	文および関数	内　容	使用できる場所		
			テンプレート	部品	業務ルール
60	@@xmlprevious（可変記号）	同じ親を持つ直前のノードを取得	○	○	○

（凡例）
○：使用できる。
△：使用できる。ただし，業務ルールから呼び出される場合は使用できない。
×：使用できない。

注
[ ] は省略できることを表します。
{ } は項目のどれか一つを記述することを表します。

7.9.2 @@break 文

形　式

@@break;

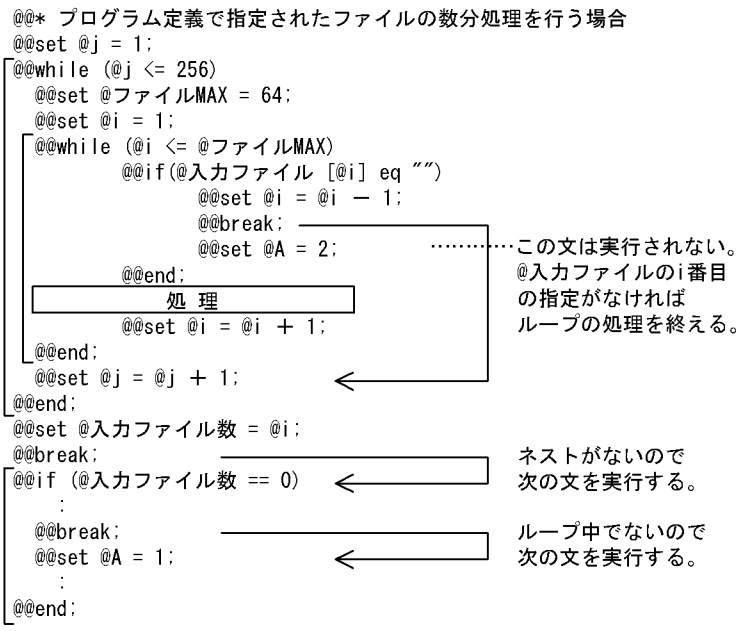
機　能

@@foreach 文，@@switch 文，または @@while 文のループの実行を終了し，ループ外の次の文を実行するときに使用します。
@@foreach 文，@@switch 文，@@while 文がネストしている場合，@@break 文が指定されているループの最も外側のネストとなる文までの実行を終了します。

規　則

ループ中でない場合は次の文を実行する。

使用例



7.9.3 @@columnlist 関数

形 式

@@columnlist（可変記号）

機 能

表・ビューを構成する列名を配列の値として取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- 可変記号は @@interface 文で ATTR=DB（RDB 定義）を指定した可変記号を指定する。
- 言語種別が SQL の場合，データ項目の名前※1 が取得できる。

  データ定義で結合項目を取り込んでいる場合，最上位結合項目の言語別の名前※1 が取得できる。

  データ定義でレコード定義ファイル※2 を取り込んでいる場合，2 番目のデータ項目※3 と同じレベル番号が指定されたデータ項目の名前※1 が取得できる。

注※ 1

SQL の名前が指定されていない場合は，データ項目名が取得されます。

注※ 2

SEWB+/RECORD DEFINER で定義したレコード定義ファイル（.csc）を指し

ます。

注※3

SEWB+/RECORD DEFINER のレコード定義では，項番 2 の項目に該当します。

使用例

```
@@set  @item     =  @@itemlist(@更新DB);
@@set  @itemcnt  =  @@count(@item);
@@set  @column     =  @@columnlist(@更新DB);
@@set  @columncnt  =  @@count(@column);

EXEC SQL
   UPDATE   @更新DB[表名]
         SET    @column[1]  =  :@item[1]
   @@set  @i =  2;
   @@while  (@i  <=  @itemcnt)
                 ,@column[@i]  =  :@item[@i]
     @@set  @i =  @i + 1;
   @@end;
         WHERE  CURRENT  OF  CRO1
END-EXEC
```

7.9.4 @@continue 文

形 式

@@continue;

機 能

@@continue 文は，@@foreach 文，または @@while 文中の @@continue 文が記述された文のループの実行を終了し，そのループの条件式を実行します。ネストしていない文で記述された場合は，直後の文を実行します。

規 則

- @@continue 文が実行されると，ループの条件式が実行する文に制御が移される。
- ネストしていない文中では，次の文を実行する。

使用例

```
@@* プログラム定義で指定されたファイルの数分処理を行う場合
@@set @ファイルMAX = 64;
@@set @i = 1;
@@while (@i <= @ファイルMAX)  <-------------+   @@if文の実行を終了して
  @@if (@入力ファイル [@i] eq "")           |   ループの先頭の条件判定
        @@set @i = @i + 1 ;                 |   を実行する。
        @@continue;  -----------------------+
        @@set @A = ' ABC' ;                     …この文は実行されない。
  @@else
  +-------------+
  | 処 理       |
  +-------------+
  @@end;
  @@set @i = @i + 1 ;
@@end;
@@continue;  <--------------+   ネストしていない文中では
@@set @i = @i + 1;          |   次の文を実行する。
```

7.9.5 @@count 関数

形 式

@@count（可変記号）

機 能

@@count 関数は 9 けたの正の整数を示す数値関数です。配列の要素数を取り出すときに使用します。

規 則

- 可変記号が配列の構造を持たないときは，値が 1 となる。
- 可変記号の添字がキーワードでない場合，要素の数となる。
- 配列の途中を飛ばして値が設定されている場合，値が設定されているところまでの配列数を返す。

使用例

（例 1）

入出力項目設定

入出力名　参照ファイル [　]
可変記号　@参照ファイル

入出力項目	データ定義
参照ファイル [1]	商品マスタファイル
参照ファイル [2]	得意先ファイル
参照ファイル [3]	
参照ファイル [4]	
参照ファイル [5]	
参照ファイル [6]	

```
@@* 指定されたファイルの数を求める
@@set @CNT = @@count(@参照ファイル);  …参照ファイルに
                                        指定された値が
```

```
                                                    [2]までのため，
                                                    @CNTには2が設定
                                                    される。
```

（例 2）

```
    @@set @A[1] = 1;
    @@set @A[100] = 10 ;
    @@set @CNT = @@count(@A) ;  …@CNTには100が入る。
```

（例 3）

```
    @@set @B[a] = 'A' ;
    @@set @B[abc] = 'B' ;
    @@set @CNT = @@count(@B) ; …@Bの添字の要素が2
                                  個のため@CNTには2
                                  が入る。
```

7.9.6 @@defined 関数

形 式

@@defined（可変記号）

機 能

@@defined 関数は，可変記号が未定義値かどうかを判定するときに使用する論理関数です。可変記号が未定義とは，文字列値も数値も設定されていない状態です。

規 則

- @@merge 文，@@put 文，@@global 文に指定された可変記号には真を返す。
- @@interface 文で指定された可変記号に修飾名を指定しない場合，真を返す。
- @@interface 文で指定された可変記号に修飾名を指定し，プログラム定義でデータ定義ファイルを選択したときは真を返し，データ定義ファイルを選択しないときは偽を返す。
- @@set 文で可変記号に値が設定されているときには真を返し，@@set 文で値が設定されていないときは，偽を返す。
- @@defined 関数は条件式中でしか使用できない。

使用例

```
@@set @FLAG = 'N';
@@if (@FLAG eq 'Y')
  @@set @DB = 'HiRDB' ;
  @@set @DC = 'OpenTP1' ;
@@else
  @@set @DB = 'SAM' ;
@@end;
@@if (@@defined (@DC))
  @@continue;
@@else
  @@msg 'DC区分が設定されていません';
  @@set @DC = 'TSS';
@@end;
```

@DCが設定されていないためメッセージを出力し，設定する処理を行う。

7.9.7 @@diagram 文

形 式

@@diagram " ブロック名 "[PARENT = " 親ブロック名 "]
[COMMENT = " 説明文字列 "];

注

［ ］は省略できることを表します。

機 能

プログラム定義の [ユーザ処理] タブでのユーザ追加処理（UOC）および業務ルール処理の一覧を，ツリー形式で表示させるときに使用します。@@diagram 文を使って生成するソースプログラムの構造を表現し，さらに @@uoc 文や @@rule 文を使って，その中でユーザ追加処理や業務ルール処理を使う位置に配置します。これによって，プログラム作成者がユーザ追加処理や業務ルール処理を開いて見なくてもおおよその内容がわかるようにできます。

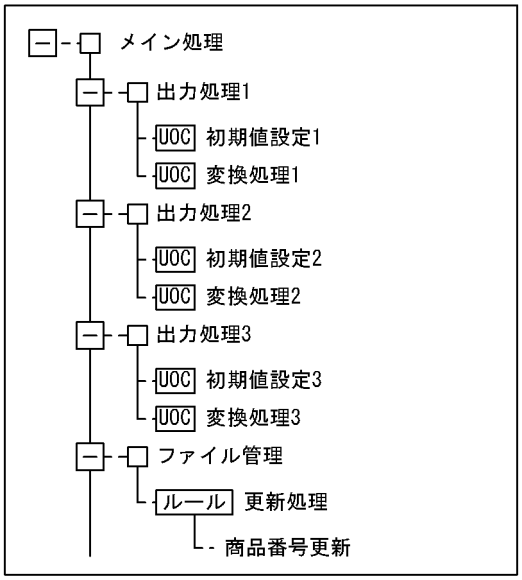
規 則

- ブロック名には，処理のブロック（処理の単位）を識別するための文字列を指定する。
- ブロック名，親ブロック名の文字列には可変記号を指定する。
- ループの中で @@diagram 文を繰り返し実行する場合には，ブロック名に可変記号を指定してユニークにし，それぞれのブロックを区別する。
- 親ブロック名を指定した場合，ブロック名は親ブロックの子ブロックとして位置づけられる。
- 親ブロック名を省略すると，最上位の階層に位置づけられる。ただし，@@diagram 文が @@parts 文から展開されている場合，@@parts 文のブロックの下に展開される。
- 説明文字列は [ユーザ処理] タブの説明欄に表示される。
- 親ブロック名が存在しない場合は，最上位の階層に位置づけられる。

使用例

```
@@diagram  "メイン処理"  COMMENT="ファイルマージメイン処理";
MAIN SECTION.
     :
@@set @I = 1;
@@while(@I<=3)
  @@diagram  "出力処理@I"  PARENT="メイン処理";
  OUT-PUT-@I SECTION.
     :
  @@uoc  "初期値設定@I"  PARENT="入力処理@I";
  COMMENT="出力レコードのデフォルト値を設ける場合に記述してください。";
     :
  @@uoc  "変換処理@I"  PARENT="入力処理@I";
  COMMENT="レコードの内容を出力するときの変換処理を記述してください。";
     :
@@set @I = @I +1;
@@end;
@@rule  "更新処理" WITH USAGE {{USAGE=IN REF=@商品ファイル}
         REF=@更新ファイル}
         PARENT= "ファイル管理"
         COMMENT="ファイルデータの更新処理を選択してください。";
```

このとき，[ユーザ処理]タブには，次のようにユーザ追加処理（UOC）および業務ルール処理の一覧が表示されます。



7.9.8 @@errorexit 文

形　式

@@errorexit;

機　能

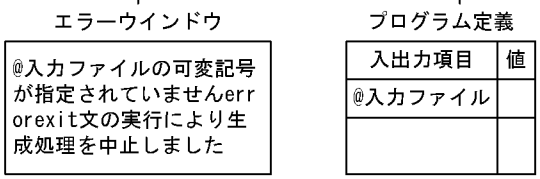
プログラム生成を中止するときに使用します。処理を中止し，プログラム生成中にエラーがあったという生成中止のメッセージをエラーウィンドウに表示します。

規　則

「errorexit 文の実行により生成処理を中止しました。」というメッセージが出力される。

使用例

```
@@interface
  @入力ファイル =
   {ATTR = DB, IO = IN_OUT, COMMENT = "入力必須"};
    :
@@if (@入力ファイル eq "")
  @@msg '@入力ファイルの可変記号が指定されていません';
  @@errorexit;
@@end;
    :
```



7.9.9 @@expand 文

形　式

@@expand 可変記号 [修飾名 [, 位置]]
[PREFIX= 文字定数] [SUFFIX= 文字定数]
[START= 開始値] [UP= 増分値]
[START_POSITION= 開始位置]
[{VALUE | NOVALUE}] ;

注

[ ] は省略できることを表します。
{ } は項目のどちらか一つを記述することを表します。

機　能

データ定義のレコードの生成を指示するときに使用します。

規　則

- 言語種別が C，または COBOL の場合に指定する。
- 言語種別が COBOL の場合，レコード定義を参照するデータ定義が指定できる。
- 可変記号名は @@interface 文の ATTR でデータ定義種別を指定した可変記号を指定する。
- 修飾名は，データ定義識別子が RECORD_NAME でなければならない。

- データ定義で結合項目またはレコード定義ファイルが複数指定されている場合，生成したい結合項目またはレコード定義ファイルの位置（データ定義で指定した結合項目の順序）を指定する。すべての結合項目およびレコード定義ファイルをレコード生成の対象にする場合，結合項目の位置を省略する。

（例）

データ定義で指定した 2 番目の結合項目のレコード生成を行う。

@@expand @ 入力ファイル [レコード名，2] ；

- PREFIX には，生成するデータ項目名の先頭に付ける文字列を文字定数で指定する。
- SUFFIX には，生成するデータ項目名の末尾に付ける文字列を文字定数で指定する。
- PREFIX および SUFFIX の指定は，言語種別が COBOL でデータ項目名が FILLER の場合，無視される。
- レベル番号の開始値または増分値の指定がない場合，データ定義で定義された開始値または増分値で展開される。
- START は，言語種別が COBOL の結合項目のレコード生成のときだけ有効である。ほかの場合は無視する。なお，COBOL のレコード定義ファイルのレコード生成の場合は，レコード定義ファイルで指定されたレベル番号で生成される。
- START には，COBOL 言語のレベル番号の開始値を数値定数または可変記号で指定する。開始値は 1 ～ 49 の範囲で指定する。
- UP は，言語種別が COBOL のときだけ有効である。多言語の場合は無視する。
- UP には，COBOL 言語のレベル番号の増分値を数値定数または可変記号で指定する。増分値は 1 ～ 48 の範囲で指定する。
- START および UP は，結合項目のレコード生成の場合だけ有効である。レコード定義ファイルのレコード生成の場合は無視され，レコード定義ファイルで指定されたレベル番号で生成される。
- レコード定義ファイルにコピー文（タイプ @）の指定がある場合，COPY 文で展開する。
- COPY 文で展開する場合，PREFIX および SUFFIX の指定値を，COPY 文の PREFIX 句および SUFFIX 句に生成する。
- START_POSITION には，レコード生成を開始するカラム位置を数字定数または可変記号で指定する。カラム位置は 1 ～ 99 の範囲で指定する。省略した場合は 2 が仮定される。
- 言語種別が COBOL で，環境変数の COBOL ソース正書法が固定形式またはホスト向け固定形式の場合，START_POSITION の値に 6 を加えたカラム位置から生成する。
- レベル番号が 49 を超えた場合はエラーメッセージを出力し，レコードの展開を中止する。
- 展開時に，データ項目の重複チェックおよびデータ項目名長のチェックは行われない。
- VALUE，および NOVALUE は，言語種別が COBOL のときだけ有効である。他

言語の場合は無視する。

- VALUE はデータ項目の初期値を生成するときに指定する。NOVALUE のときにはデータ項目の初期値を生成しない。VALUE，および NOVALUE の指定がない場合には NOVALUE を仮定する。
- レコード生成をするデータ項目のタイプには，リポジトリに最初から登録されている C または COBOL 用のタイプだけ使用できる。

生成規則

レコードの生成規則を表 7-4 ～ 7-6 に示します。

表 7-4　レコード生成時に有効に使用されるデータ項目の定義情報

項番	区分	データ項目定義情報	COBOL	C
1	名称	データ項目名	○※1	○※1
2		標準名称	×	×
3		振り仮名	×	×
4	属性	分類	×	○
5		けた数	○	○
6		小数部けた数	○	○
7		反復回数	○	○
8	付加情報	フィールド 1 ～ 20	×	×
9		コメント	×	×
10	言語別	名前	○	○
11	詳細情報	タイプ	○	○
12		数字編集文字列	○	×
13		初期値	○※2	×
14		言語別フィールド	×	×
15		取りうる値	○	○

（凡例）

○：使用する。

×：使用しない。

注※ 1

言語別詳細情報の名前が指定されていない場合，使用します。

注※ 2

@@expand 文に VALUE の指定がある場合，使用します。

表 7-5 COBOL 言語用レコード生成キーワード表

項番	辞書属性（タイプ）	レコード生成キーワード
1	X（英数字項目）	データ項目名 PIC X（けた数） OCCURS 反復回数 VALUE 初期値
2	Z（数字編集項目）	データ項目名 PIC 数字編集文字列 OCCURS 反復回数 VALUE 初期値
3	N（漢字項目）	データ項目名 PIC N(けた数) OCCURS 反復回数 VALUE 初期値
4	9（符号なし外部 10 進項目）	データ項目名 PIC 9（けた数－小数部けた数） V9（小数部けた数） OCCURS 反復回数 VALUE 初期値
5	S（符号付き外部 10 進項目）	データ項目名 PIC S9（けた数－小数部けた数） V9（小数部けた数） OCCURS 反復回数 VALUE 初期値
6	U（符号なし内部 10 進項目）	データ項目名 PIC 9（けた数－小数部けた数） V9（小数部けた数） USAGE PACKED-DECIMAL OCCURS 反復回数 VALUE 初期値
7	P（符号付き内部 10 進項目）	データ項目名 PIC S9（けた数－小数部けた数） V9（小数部けた数）USAGE PACKED-DECIMAL OCCURS 反復回数 VALUE 初期値
8	BU（符号なし 2 進項目）	データ項目名 PIC 9（けた数－小数部けた数） V9(小数部けた数） USAGE BINARY OCCURS 反復回数 VALUE 初期値
9	B（符号付き 2 進項目）	データ項目名 PIC S9（けた数－小数部けた数） V9(小数部けた数） USAGE BINARY OCCURS 反復回数 VALUE 初期値
10	E（外部浮動小数点項目）	データ項目名 PIC 数字編集文字列 OCCURS 反復回数
11	D（内部浮動小数点項目）	データ項目名 COMP-1 または COMP-2 OCCURS 反復回数 VALUE 初期値
12	1 （内部ブール項目）	データ項目名 PIC 1（けた数）USAGE BIT OCCURS 反復回数 VALUE 初期値
13	8（外部ブール項目）	データ項目名 PIC 1（けた数）USAGE DISPLAY OCCURS 反復回数 VALUE 初期値
14	T（アドレスデータ項目）	データ項目名 USAGE ADDRESS OCCURS 反復回数 VALUE 初期値
15	フリー定義	データ項目名 フリー定義

注 1

日立 COBOL2002 または日立 COBOL85 以外の言語種別を使用している場合は，レコード生成

キーワードのカスタマイズが必要になる場合があります。レコード生成キーワードのカスタマイズについては，マニュアル「SEWB+/REPOSITORY　辞書設計ガイド」の環境構築ユティリティについての記述を参照してください。

注2

「取りうる値」が定義されている場合，表中の生成に続けて，次に示す形式でレコード生成されます。ただし，値が指定されていない場合，その値を含む「88 記号　VALUE 値」は生成されません。

```
88  記号  VALUE値.
[88  記号  VALUE値.]...
```

表 7-6　C 言語用生成規則

項番	辞書属性（タイプ）	生成キーワード
1	char（文字型）	char　データ項目名　［反復回数］［けた数］※
2	char*（文字型）	char*　データ項目名　［反復回数］
3	short（符号付き短整数型）	short　データ項目名　［反復回数］
4	unsigned　short（符号なし短整数型）	unsigned　short　データ項目名　［反復回数］
5	int（符号付き整数型）	int　データ項目名　［反復回数］
6	unsigned　int（符号なし整数型）	unsigned　int　データ項目名［反復回数］
7	long（符号付き長整数型）	long　データ項目名　［反復回数］
8	unsigned　long（符号なし長整数型）	unsigned　long　データ項目名　［反復回数］
9	float（単精度浮動小数点型）	float　データ項目名　［反復回数］
10	double（倍精度浮動小数点型）	double　データ項目名　［反復回数］
11	long　double（拡張精度浮動小数点型）	long　double　データ項目名　［反復回数］
12	フリー定義	フリー定義 データ項目名

注

「取りうる値」が定義されている場合，表中の生成に続けて，次に示す形式でレコード生成されます。ただし，値が指定されていない場合，その値を含む「＝値」は生成されません。

```
enum項目名 {
            記号 [=値],
            [記号 [=値]]...
           }
```

注※

けた数は，辞書で指定したけた数に文字列の末尾を示す 1 文字分を追加した値で生成されます。また，データ項目の分類が漢字項目の場合，けた数は 2 倍して生成されます。

使用例

（例 1）言語種別が COBOL のとき

SEWB+/REPOSITORY

結合項目

```
REC
 ├X1
 │ ├── X11
 │ ├── X12
 │ └── X13
 ├Z1
 ├N1
 │ ├── N11
 │ ├── N12
 │ └── N13
 ├91
 ├S1
 ├U1
 └P1
```

データ項目

データ項目	タイプ	けた数	少数部けた数	反復回復	数学編集	初期値
X11	X	5	−	−	−	−
X12	X	10	−	−	−	SPACE
X13	X	15	−	10	−	−
Z1	Z	−	−	−	ZZ9	−
N11	N	1	−	−	−	NC'あ'
N12	N	10	−	−	−	−
N13	N	15	−	10	−	−
91	9	5	−	−	−	−
S1	S	10	5	−	−	−
U1	U	5	−	−	−	−
P1	P	5	−	−	−	−

@@expand @入力ファイル[レコード名];の生成結果

```
       01  REC.
         02  X1.
           03  X11  PIC  X(5).
           03  X12  PIC  X(10).
           03  X13  PIC  X(15)
                         OCCURS  10.
         02  Z1           PIC  ZZ9.
         02  N1.
           03  N11  PIC  N(1).
           03  N12  PIC  N(10).
           03  N13  PIC  N(15)
                         OCCURS  10.
         02  91  PIC  9(5).
         02  S1  PIC  S9(5)V9(5).
         02  U1  PIC  9(5) USAGE
                PACKED-DECIMAL.
         02  P1  PIC  S9(5) USAGE
                PACKED-DECIMAL.
```

```
@@expand @入力ファイル[レコード名] PREFIX="PRE-" SUFFIX="-SUF" VALUE;の生成結果
     01  PRE-REC-SUF.
       02  PRE-X1-SUF.
         03  PRE-X11-SUF  PIC  X(5).
         03  PRE-X12-SUF  PIC  X(10)
                          VALUE SPACE.
         03 PRE-X13-SUF   PIC  X(15)
                          OCCURS  10.
       02  PRE-Z1-SUF     PIC ZZ9.
       02  PRE-N1-SUF.
         03  PRE-N11-SUF  PIC  N(1)
                          VALUE NC'あ'.
         03  PRE-N12-SUF  PIC  N(10).
         03  PRE-N13-SUF  PIC  N(15)
                          OCCURS  10.
       02  PRE-91-SUF     PIC  9(5).
       02  PRE-S1-SUF     PIC  S9(5)V9(5).
       02  PRE-U1-SUF     PIC  9(5)  USAGE
                          PACKED-DECIMAL.
       02  PRE-P1-SUF     PIC  S9(5) USAGE
                          PACKED-DECIMAL.
```

（例 2）言語種別が C 言語のとき

SEWB+/REPOSITORY

結合項目

```
rec
 ├── char_x1
 │    ├── char_11
 │    ├── char_x12
 │    ├── char_x13
 │    ├── char_n11
 │    ├── char_n12
 │    └── char_n13
 ├── char_x2
 ├── short_1
 ├── ushort_1
 ├── int_1
 ├── uint_1
 └── long_1
```

データ項目

データ項目	分　類	タイプ	けた数	反復回数
char_x11	英数字	char	1	-
char_x12	英数字	char	5	-
char_x13	英数字	char	10	5
char_n11	日本語	char	1	-
char_n12	日本語	char	5	-
char_n13	日本語	char	10	5
char_x2	英数字	char*	10	-
short_1	整数	short	-	-
ushort_1	整数	unsigned　short	-	-
int_1	整数	int	-	-
uint_1	整数	unsigned　int	-	-
long_1	整数	long	-	-

```
@@expand @入力ファイル[レコード名];の生成結果
     typedef struct rec {
      struct char_x1 {
       char char_x11[2];
       char char_x12[6];
       char char_x13[5][11];
       char char_n11[3];
       char char_n12[11];
       char char_n13[5][21];
      } char_x1;
      char* char_x2;
      short short_1;
      unsigned short ushort_1;
      int int_1;
      unsigned int uint_1;
      long long_1;
     }rec;
```

@@expand @入力ファイル[レコード名] PREFIX="pre_" SUFFIX="_suf";の生成結果

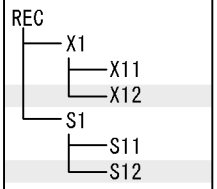
```
typedef struct pre_rec_suf{
 struct pre_char_x1_suf{
  char pre_char_x11_suf[2];
  char pre_char_x12_suf[6];
  char pre_char_x13_suf[5][11];
  char pre_char_n11_suf[3];
  char pre_char_n12_suf[11];
  char pre_char_n13_suf[5][21];
 }pre_char_x1_suf;
 char* pre_char_x2_suf;
 short pre_short_1_suf;
 unsigned short pre_ushort_1_suf;
 int pre_int_1_suf;
 unsigned int pre_uint_1_suf;
 long pre_long_1_suf;
}pre_rec_suf;
```

（例 3）取りうる値の例

- 言語種別が COBOL 言語のとき

SEWB+/REPOSITORY

結合項目



データ項目

データ項目	タイプ	けた数	少数部けた数	反復回復	数学編集	初期値
X11	X	5	−	−	−	−
X12	X	10	−	−	−	−
S11	S	5	5	−	−	−
S12	S	5	−	−	−	−

項目X12の取りうる値

記号	値
X-A	'A '
X-B	'B '

項目S12の取りうる値

記号	値
S-A	1
S-B	5
S-C	

COPYメンバ

```
    01 REC.
     02 X1.
       03 X11   PIC X(5).
       03 X12   PIC X(10)
         88 X-A VALUE 'A'.
         88 X-B VALUE 'B'.
     02 S1.
       03 S11   PIC S9(5)V9(5).
       03 S12   PIC S9(5).
         88 S-A VALUE 1.
         88 S-B VALUE 5.
```

（凡例）　　：取りうる値。

注　「項目S-C」は値の指定がないので，生成されません。

- 言語種別が C 言語のとき

SEWB+/REPOSITORY

結合項目

rec
- char_1
- char_2
- short_1
- short_2

データ項目

データ項目	分類	タイプ	けた数	反復 回復
char_1	英数字	char	5	-
char_2	英数字	char	10	-
short_1	整数	short	5	-
short_2	整数	short	5	-

項目char_2の取りうる値リスト

記号	値
c_a	0
c-b	

項目short_2の取りうる値リスト

記号	値
s_a	1
s_b	5

レコード生成キーワード

タイプ	レコード生成キーワード
char	@DATAITEM PIC X@LEN @OCCURS @VALUE
short	@DATAITEM PIC S9@ILEN@DLEN @OCCURS @VALUE

ヘッダファイル

```
typedef struct rec {
    char char_1[5];
    enum char_2 {
        c_a = 0,
        c_b
    }char_2;
    short short_1;
    enum short_2 {
        s_a = 1,
        s_b = 5
    }short_2;
}rec;
```

（凡例）　　：取りうる値。

注　enumを生成する場合，生成キーワードでの項目の展開はしません。

7.9.10　@@file 文

形　式

@@file " ブロック名 " OUTPUT_NAME=" 出力ファイル名 "

[PARENT=" 親ブロック名 "]
[COMMENT=" 説明文字列 "];

注
［ ］は省略できることを表します。

機 能

1 テンプレート中に複数の出力先を明記することで，1 回の生成操作で複数のファイルを生成できます。ただし，この機能を使用すると COBOL 開発支援との連携はできません。

規 則

- @@file 文から次の @@file 文まで（または最後まで）を，指定した出力ファイル名で生成する。
- ブロック名には，処理のブロック（処理の単位）を識別するための文字列を指定する。
- 出力ファイル名には，ドライブ名は指定できない。
- 出力ファイル名に存在しないディレクトリを指定した場合，ディレクトリを作成する。
- OUTPUT_NAME 句は，ブロック名の直後に指定する。
- ブロック名，親ブロック名，出力ファイル名の文字列には可変記号を指定できる。
- ブロック名は [ユーザ処理] タブに表示される。
- ブロック名は @@diagram 文，@@uoc 文，@@rule 文の PARENT に指定できる。
- 親ブロック名は，プログラム定義の [ユーザ処理] タブでファイルの一覧をツリー形式で表示させるときに，指定したブロック名の親になるブロック名を指定する。
- 親ブロック名を指定した場合，ブロック名は親ブロックの子ブロックとして位置づけられる。
- 親ブロック名を省略すると，最上位の階層に位置づけられる。
- 親ブロック名が存在しない場合は，最上位の階層に位置づけられる。
- 説明文字列は [ユーザ処理] タブの説明欄に表示される。
- 最初の @@file 文以前に以下の文は記述できない。
  - 展開文
  - @@uoc 文
  - @@rule 文
  - @@merge 文
  - @@put 文に記述されていない @@expand 文

使用例

```
@@lang C;
@@interface @入力ファイル = {
  ATTR = FILE,
  IO = IN,
  レコード名 = {ATTR=RECORD_NAME} };
@@interface @出力ファイル = {
  ATTR = FILE,
  IO = OUT,
  レコード名 = {ATTR=RECORD_NAME} };
@@file "ヘッダ定義" OUTPUT_NAME = "fileio.h";          1.
  @@expand(@入力ファイル[レコード名]);
  @@expand(@出力ファイル[レコード名]);
@@file "エラー処理" OUTPUT_NAME = "fileio.cpp";        2.
#include "stdafx.h"
#include "fileio.h"
void file_io_main()
{
    int rtn_code = 0;
    :
    rtn_code = file_io_err();
    :
    return(true);
}
```

1. @@file ヘッダを"fileio.h"に生成する。
2. @@file ソースファイルを"fileio.cpp"に生成する。

7.9.11 @@foreach 文

形 式

@@foreach 可変記号 1 (可変記号 2)
　　文
@@end;

機 能

可変記号 2 の配列中の要素を可変記号 1 に一つずつ取り出して，処理を繰り返して実行するときに使用します。

規 則

- 可変記号 1 には可変記号 2 の配列の値が順に設定されて，配列の要素がなくなるまで文が繰り返し実行される。
- 可変記号 1 は @@foreach 文だけで使用され，@@foreach 文のループの外側に同一の可変記号名があった場合，別の可変記号として扱われる。
- 可変記号 2 が配列の要素を持たない場合，1 個の要素として処理される。
- 可変記号 2 の添字が位置指定（数字）の場合，1 から要素の最後までループする。
- 可変記号の添字がキーワード指定（文字）の場合，設定された順序で最後までループする。

使用例

```
(例1)                                          展開結果
@@set @ITEM01 ={'a','b','c','d'};              01  WORKa  PIC X(10).
@@foreach @WORK (@ITEM01)          →           01  WORKb  PIC X(10).
  01  WORK@WORK  PIC  X(10).                   01  WORKc  PIC X(10).
@@end ;                                        01  WORKd  PIC X(10).
```

```
(例2)
@@set  @ITEM02[a] = 1;                         展開結果
@@set  @ITEM02[c] = 3;                         01  WORK1  PIC X(10).
@@set  @ITEM02[b] = 2;                         01  WORK3  PIC X(10).
@@foreach @WORK (@ITEM02)          →           01  WORK2  PIC X(10).
  01  WORK@WORK  PIC  X(10).
@@end ;
@@set  @WORKwork = 1;          ……@@foreachの@WORKとは
                                               別のものである。
```

```
(例3)
@Items[1]の値:   {'01-ITEM1', 'I1-ITEM1'}
@Items[2]の値:   {'01-ITEM2', 'I1-A01'}
@Items[3]の値:   {'01-ITEM3', 'I1-B01'}
のとき,
@@foreach @pair (@Items)
    MOVE @pair[2]  TO @pair[1]
@@end;
とすると，次のように展開されます。
MOVE I1-ITEM1 TO 01-ITEM1
MOVE I1-A01 TO 01-ITEM2
MOVE I1-B01 TO 01-ITEM3
```

7.9.12 @@getdata 関数

形 式

@@getdata(可変記号 [{,"リポジトリの言語区分 "[,BASIC_ATTR]
|,,BASIC_ATTR}])

注

[ ] は省略できることを表します。

{ } は項目のどちらか一つを記述することを表します。

機 能

@@interface 文で指定した可変記号からデータ項目の定義情報を取得するときに使用します。データ定義ファイルからレコード定義を参照している場合，レコード定義で定義されたデータ項目の情報も取得できます。

規 則

- @@set 文の右辺に指定する。
- 可変記号には @@interface 文で ATTR=ITEM を指定した修飾名，または ATTR でデータ定義種別を指定した可変記号を指定できる。
- リポジトリの言語区分によって，言語別詳細情報の対象言語を選択できる。

- リポジトリの言語区分は SEWB+/REPOSITORY に登録されている言語区分を文字定数で指定する。省略した場合，@@lang 文の FOR_REPOSITORY 句の指定に従う。
- BASIC_ATTR を指定すると，データ項目の基本属性タブ情報の取得ができる。
- データ項目の定義情報は，@@set 文の左辺の可変記号に二次元の配列で設定される。
- 一次元目の配列はデータ項目ごとに確保され，その要素数は下位のデータ項目を含めた数と同じ。一次元の配列の添字には，数値を指定する。
- 二次元目の配列は，データ項目の定義情報ごとに確保される。二次元の配列の添字には，キーワードを指定する。
- 基本属性タブ情報の標準名称，振り仮名，コメント，およびフィールドは，辞書での指定順に三次元の配列として設定される。
- 三次元の配列の要素には，各定義情報の定義順を数値で指定する。
- レコード定義ファイルにコピー文（タイプ @）の指定がある場合，下位のレコード定義情報は参照できない。

取得できるデータ項目の定義情報は，データ定義に結合項目が指定されている場合と，レコード定義が指定されている場合で異なります。データ項目の定義情報を以下に示します。

表 7-7　データ項目の定義情報（データ定義に結合項目が指定されている場合）

データ項目の定義情報	添字に指定するキーワード	
名称	データ項目名	DataItemName
属性	分類	Type ※1
	けた数	Digit ※2 ※3
	小数部けた数	FloatDigit ※2 ※3
	反復回数	RepeatLevel ※2
言語別詳細情報	名前	SourceName
	タイプ／フリー定義	SourceType ※4
	タイプ修飾情報	TypeModifyInfo
	初期値	InitValue
	フィールド	Field
	取りうる値の定義数	AvailableCount ※5
	取りうる値の記号	AvailableName ※6
	取りうる値の値	AvailableValue ※6
データ項目の結合関係を表す情報	レベル	Level ※7
	上位の項目の位置	ParentRow ※8
基本属性タブ情報	国語別情報の定義数	LangTypeCount ※9

データ項目の定義情報	添字に指定するキーワード	
	標準名称	StdName ※10
	振り仮名	Furigana ※10
	コメント	Comment ※10
	フィールドの定義数	FieldCount ※11
	フィールド	BasicField ※12

表 7-8 データ項目の定義情報（データ定義にレコード定義ファイルが指定されている場合）

データ項目の定義情報	添字に指定するキーワード		辞書参照なし	辞書参照あり
名称	データ項目名	DataItemName	○	○
属性	分類	Type ※1	×	○
	けた数	Digit ※2※3	○	○
	小数部けた数	FloatDigit ※2※3	○	○
	反復回数	RepeatLevel ※2	○※19	○
言語別詳細情報	名前	SourceName	○	○
	タイプ／フリー定義	SourceType ※4	○	○
	タイプ修飾情報	TypeModifyInfo	○※19	○
	初期値	InitValue	○※19	○
	フィールド	Field	×	○
	取りうる値の定義数	AvailableCount ※5	×	○
	取りうる値の記号	AvailableName ※6	×	○
	取りうる値の値	AvailableValue ※6	×	○
データ項目の結合関係を表す情報	レベル	Level ※7	○	○
	上位の項目の位置	ParentRow ※8	○	○
基本属性タブ情報	国語別情報の定義数	LangTypeCount ※9	○	○
	標準名称	StdName ※10	○	○
	振り仮名	Furigana ※10	×	○
	コメント	Comment ※10	○	○
	フィールドの定義数	FieldCount ※11	×	○
	フィールド	BasicField ※12	×	○

データ項目の定義情報	添字に指定するキーワード		辞書参照なし	辞書参照あり
レコード定義固有の情報	CHARCTER TYPE	CharcterType ※13	○※19	○
	CHARCTER TYPE 値	CharcterTypeValue	○※19	○
	指標名	Index	○※19	○
	可変反復回数項目名	RepeatLevelName	○※19	○
	可変反復回数最小値	RepeatMini	○※19	○
	アドレス名	AddressName	○※19	○
	再定義名	RedefineName	○※19	○
	EXTERNAL	External ※14	○※19	○
	SYNC	Sync ※15	○※19	○
	JUST	Just ※16	○※19	○
	BLANK ZERO	BlankZero ※17	○※19	○
	辞書参照	Dict ※18	○※19	○

（凡例）

○：定義情報を取得できる

×：常に 0 文字の文字列が設定される

注

SourceType，Level，ParentRow 以外は SEWB+/REPOSITORY で定義した値になります。

注※ 1

Type の値は，SEWB+/REPOSITORY の属性の分類で選択した値が設定されます。設定値を以下に示します。

- 結合データ
- 英数字文字列データ
- 日本語文字列データ
- 整数データ
- 正整数データ
- 実数データ
- 日付データ
- 時刻データ
- 分類なし

注※ 2

数値を指定する項目の定義情報が未指定の場合，0 文字の文字列が設定されます。

注※ 3

結合データの場合，設定される値は不定です。

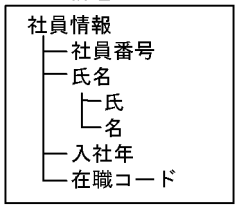
注※ 4

SourceType の値は，ルールスクリプト中のキーワード（@TYPE）の値と同じになります。ルールスクリプト中のキーワードについてはマニュアル「SEWB+/REPOSITORY 辞書設計ガイド」を参照してください。

データ項目のタイプが集団項目の場合，SourceType の値は 0 文字の文字列が設定され，タイプ「@コピー文」が指定された場合，SourceType の値は ”@” が設定されます。
注※ 5
AvailableCount の値は，AvailableName と AvailableValue の三次元目の配列要素の数が設定されます。
注※ 6
AvailableName と AvailableValue の値は，データ項目辞書で指定された順に三次元目の配列要素として設定されます。
注※ 7
Level の値は，可変記号に指定されたデータ項目を 1 として，結合関係の階層の深さが設定されます。
注※ 8
ParentRow の値は，上位のデータ項目の配列の位置が設定されます。上位のデータ項目が最上位結合項目の場合は，0 が設定されます。
注※ 9
LangTypeCount の値は，StdName，Furigana，および Comment の三次元の配列要素の数が設定されます。要素数は，常に 4（日本語，英語，未設定 1，未設定 2）です。
注※ 10
StdName，Furigana，および Comment の値は，国語別で指定された順に三次元の配列要素として設定されます。
注※ 11
FieldCount の値は，BasicField の三次元の配列要素の数が設定されます。要素数は，常に 20 です。
注※ 12
BasicField の値は，データ項目辞書で指定された順（フィールド 1 〜フィールド 20）に三次元の配列要素として設定されます。
注※ 13
レコード定義で選択した CHARCTER TYPE が設定されます。
注※ 14
レコード定義で EXTERNAL が選択されている場合，” EXTERNAL ” が設定されます。選択されていない場合，0 文字の文字列が設定されます。
注※ 15
レコード定義で SYNC が選択されている場合，” SYNC ” が設定されます。選択されていない場合，0 文字の文字列が設定されます。
注※ 16
レコード定義で JIST が選択されている場合，” JIST ” が設定されます。選択されていない場合，0 文字の文字列が設定されます。
注※ 17
レコード定義で BLANK WHEN ZERO が選択されている場合，” BLANK WHEN ZERO ” が設定されます。選択されていない場合，0 文字の文字列が設定されます。
注※ 18
辞書参照の場合，「参照」または「構成」が設定されます。辞書を参照していない場合，0 文字の文字列が設定されます。
注※ 19
言語種別が SQL の場合は，常に 0 文字の文字列が設定されます。

使用例 1（BASIC_ATTR **オペランド指定なし**）

データ構造



@@getdata 関数で取得されるデータ項目の定義情報（言語区分は "COBOL **又は** OOCOBOL"**）** 1 ／ 2

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	社員情報	結合データ	※	※		0
2	2	社員番号	英数字文字列データ	15			1
3	2	氏名	結合データ	※	※		1
4	3	氏	英数字文字列データ	30			3
5	3	名	英数字文字列データ	30			3
6	2	入社年	整数データ	4			1
7	2	在職コード	英数字文字列データ	1			1

定義情報の続き 2 ／ 2

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
1	社員情報_CBL							
2	社員番号_CBL	X						
3	氏名_CBL							
4	氏_CBL	X						
5	名_CBL	X						

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
6	入社年_CBL	9						
7	在職コード_CBL	X				2	在職	SPACE
							退職	'A'

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

注※

結合データのけた数（Digit）小数けた数（FloatDigit）の値は不定です。

テンプレート

```
  :
@@interface  @FILE1 = {
  ATTR=FILE,
  KEY={ATTR=ITEM, REF=@FILE1}
  };
  :
@@set  @REC=@@getdata(@FILE1, "COBOL又はOOCOBOL");    1.
@@set  @REC_cnt = @@count(@REC);
@@set  @cnt = 1;
@@while ( @cnt <= @REC_cnt)            2.
 @@switch (@REC[@cnt ,SourceType] )    3.
     @@case  "":
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  @REC[@cnt,External].          } 4.
       @@break;
     @@case  "X":
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  PIC  X(@REC[@cnt,Digit]).     } 5.
       @@break;
     @@case  "9":
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  PIC  9(@REC[@cnt,Digit]).     } 6.
       @@break;
      :
     @@default:
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  @REC[@cnt ,SourceType].       } 7.
 @@end;
 @@if (@REC[@cnt, AvailableCount] ne "" )
   @@set  @cnt2 = 1;                                                          )
     @@while(@cnt2 <= @REC[@cnt, AvailableCount])                             |
       88  @REC[@cnt, AvailableName,@cnt2] VALUE @REC[@cnt,                   |
 AvailableValue,@cnt2]..                                                      |
       @@set  @cnt2  =  @cnt2  +  1;                                          } 8.
     @@end;                                                                   |
  @@end;                                                                      |
    @@set  @cnt = @cnt + 1;                                                  |
 @@end;                                                                      )

 @@set  @cnt = 1;
 @@while ( @cnt <= @REC_cnt)     9.
    @@if(@REC[@cnt,Type]  ne  "結合データ")
       @@set @parent_pos = @REC[@cnt,ParentRow] ;    10.
       @@set @string  =  @REC[@cnt,SourceName] ;
       @@while (@parent_pos > 0)                     11.
           @@set  @string = "@string  OF  @REC[@parent_pos,SourceName]" ;    12.
           @@set @parent_pos = @REC[@parent_pos,ParentRow];                  13.
       @@end;
       @@switch(@REC[@cnt,SourceType])                                      )
           @@case  "X":                                                     |
              MOVE  SPACE  TO  @string.                                     |
              @@break;                                                      |
           @@case  "9":                                                     } 14.
              MOVE  ZERO  TO  @string.                                      |
              @@break;                                                      |
            :                                                               |
       @@end;                                                               )
    @@end;
    @@set  @cnt = @cnt + 1;
 @@end;
```

1. データ項目の定義情報を取得し，@RECに設定。
2. 取得したデータ項目定義情報の項目数分繰り返す。
3. SourceTypeの値で生成を分ける。
4. 結合項目の生成。
   「レベル 名前.」
5. SourceTypeが"X"の生成。
   「レベル　名前　PIC　X（けた数）.」
6. SourceTypeが"9"の生成。
   「レベル　名前　PIC　9（けた数）.」
7. フリー定義の生成。
   「レベル 名前 フリー定義文字.」
8. 条件名の生成。
9. 取得したデータ項目定義情報の項目数分繰り返す。
10. 上位の項目の位置を設定する。
11. 上位の項目がなくなるまで繰り返す。
12. 可変記号に設定されている文字列の末尾に「OF　上位の項目の名前」を追加する。
13. 上位の項目の位置を設定する。
14. 単一項目の初期化。

● 可変記号に最上位結合項目を指定した場合

@@set 文

```
@@set @REC = @@getdata(@FILE1,"COBOL又はOOCOBOL");
```

@REC の内容

@REC の値は SEWB+/REPOSITORY での定義内容が展開されます。

生成ソース

```
1 社員情報_CBL.
2 社員番号_CBL PIC X(15).
2 氏名_CBL.
3 氏_CBL PIC X(30).
3 名_CBL PIC X(30).
2 入社年_CBL PIC 9(4).
2 在職コード_CBL PIC X(1).
 88 在職 VALUE SPACE.
 88 退職 VALUE 'A'.
:
MOVE SPACE TO 社員番号_CBL OF 社員情報_CBL.
MOVE SPACE TO 氏_CBL OF 氏名_CBL OF 社員情報_CBL.
MOVE SPACE TO 名_CBL OF 氏名_CBL OF 社員情報_CBL.
MOVE ZERO TO 入社年_CBL OF 社員情報_CBL.
```

● 可変記号に結合項目（氏名）を指定した場合

@@set 文

```
@@set @REC = @@getdata(@FILE1[KEY],"COBOL又はOOCOBOL");
```

@REC の内容 1 ／ 2

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	氏名	結合データ	※	※		0
2	2	氏	英数字文字列データ	30			1
3	2	名	英数字文字列データ	30			1

@REC の内容の続き 2 ／ 2

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field
1	氏名 _CBL				
2	氏 _CBL	X			
3	名 _CBL	X			

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

注※

結合データのけた数（Digit）小数けた数（FloatDigit）の値は不定です。

生成ソース

```
1 氏名_CBL.
2 氏_CBL  PIC X(30).
2 名_CBL  PIC X(30).
:
MOVE SPACE TO 氏_CBL OF 氏名_CBL.
MOVE SPACE TO 名_CBL OF 氏名_CBL.
```

● 可変記号に単項目（氏）を指定した場合

@@set 文

```
@@set @REC = @@getdata(@FILE1[KEY],"COBOL又はOOCOBOL");
```

@REC の内容 1 ／ 2

Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	氏	英数字文字列データ	30			0

@REC の内容の続き 2 ／ 2

SourceName	SourceType	TypeModifyInfo	InitValue	Field
氏 _CBL	X			

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

生成ソース

```
1 氏_CBL PIC X(30).
:
MOVE SPACE TO 氏_CBL.
```

● 可変記号にレコード定義ファイルを指定した場合

@@set 文

```
@@set @REC = @@getdata(@FILE1,"COBOL又はOOCOBOL");
```

@REC の内容

@REC の値は SEWB+/RECORD DEFINER で定義したレコード定義ファイルの内容が展開されます。

レコード定義ファイル

項番	レベル	データ項目名	名前	タイプ	けた数	少数けた数	EXTERNAL	辞書参照
1	1	社員情報	社員情報 _CBL				オン	
2	2	社員番号	社員番号 _CBL	X	15		オフ	参照
3	2	氏名	氏名 _CBL				オフ	参照
4	3	氏	氏 _CBL	X	30		オフ	構成
5	3	名	名 _CBL	X	30		オフ	構成

注

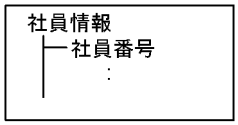
表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

生成ソース

```
1  社員情報_CBL  EXTERNAL.
2  社員番号_CBL  PIC  X(15) .
2  氏名_CBL.
3  氏_CBL       PIC  X(30).
3  名_CBL       PIC  X(30).
     :
MOVE  SPACE  TO  社員番号_CBL  OF  社員情報_CBL.
MOVE  SPACE  TO  氏_CBL  OF  氏名_CBL  OF  社員情報_CBL.
MOVE  SPACE  TO  名_CBL  OF  氏名_CBL  OF  社員情報_CBL.
```

使用例 2（BASIC_ATTR オペランド指定あり）

データ構造



@@getdata 関数で取得されるデータ項目の定義情報（言語区分は "COBOL 又は OOCOBOL"） 1 ／ 3

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	社員情報	結合データ	※1	※1		0
2	2	社員番号	英数字文字列データ	15			1
3	:	:	:				

@@getdata 関数で取得されるデータ項目の定義情報の続き 2 ／ 3

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
1	社員情報_CBL							
2	社員番号_CBL	X						
3	:							

@@getdata 関数で取得されるデータ項目の定義情報の続き 3 ／ 3

項番	LangType Count	StdName	Furigana	Comment	FieldCount	BasicField
1	4	社員情報	シャインジョウホウ※2	社員情報ファイル	20	フィールド01
		clerk_data		clerk_data_file		フィールド02
						フィールド03
						： ：
						フィールド20
2	4	社員番号	シャインバンゴウ※2		20	
3	：				：	

注

表中のキーワードは説明上，付けているものです。

注※1

結合データのけた数（Digit）小数けた数（FloatDigit）の値は不定です。

注※2

半角で表示されます。

テンプレート

```
 :
@@interface  @FILE1 = {
  ATTR=FILE,
  KEY={ATTR=ITEM, REF=@FILE1}
  };
  :
@@set  @REC=@@getdata(@FILE1, "COBOL又はOOCOBOL",BASIC_ATTR);    1.
@@set  @REC_cnt = @@count(@REC);
@@set  @cnt = 1;
@@while ( @cnt <= @REC_cnt)           2.
 @@if (@REC[@cnt, Comment,1] ne "" )                                  }
   **  @REC[@cnt, Comment,1]    @REC[@cnt, BasicField,1]              } 3.
 @@end;
 @@switch (@REC[@cnt ,SourceType])   4.
     @@case  "":                                                      }
       @REC[@cnt,Level]  @REC[@cnt,SourceName]..                      } 5.
       @@break;                                                       }
     @@case  "X":                                                     }
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  PIC  X(@REC[@cnt,Digit]).  } 6.
       @@break;                                                       }
     @@case  "9":                                                     }
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  PIC  9(@REC[@cnt,Digit]).  } 7.
       @@break;                                                       }
       :                                                              }
     @@default:                                                       }
       @REC[@cnt,Level]  @REC[@cnt,SourceName]  @REC[@cnt ,SourceType].  } 8.
 @@end;
 @@if (@REC[@cnt, AvailableCount] ne "" )                                    }
    @@set  @cnt2 = 1;                                                        }
    @@while(@cnt2 <= @REC[@cnt, AvailableCount])                             } 9.
      88  @REC[@cnt, AvailableName,@cnt2] VALUE @REC[@cnt, AvailableValue,@cnt2]..  }
      @@set  @cnt2  =  @cnt2  +  1;                                          }
    @@end;                                                                   }
 @@end;
   @@set  @cnt = @cnt + 1;
@@end;
@@set  @cnt = 1;
@@while ( @cnt <= @REC_cnt)           10.
   @@if(@REC[@cnt,Type]  ne  "結合データ")
      @@set @parent_pos = @REC[@cnt,ParentRow] ;    11.
      @@set @string  =  @REC[@cnt,SourceName] ;
      @@while (@parent_pos > 0)          12.
          @@set  @string = "@string  OF  @REC[@parent_pos,SourceName]" ;    13.
          @@set @parent_pos = @REC[@parent_pos,ParentRow];                 14.
      @@end;
      @@switch(@REC[@cnt,SourceType])                                     }
          @@case  "X":                                                    }
             MOVE  SPACE  TO  @string.                                    }
             @@break;                                                     }
          @@case  "9":                                                    } 15.
             MOVE  ZERO  TO  @string.                                     }
             @@break;                                                     }
           :                                                              }
      @@end;                                                              }
   @@end;
   @@set  @cnt = @cnt + 1;
@@end;
```

1.データ項目の定義情報を取得し@RECに設定。
2.取得したデータ項目定義情報の項目数分繰り返す。
3.データ項目のコメント文の生成。
4.SourceTypeの値で生成を分ける。
5.結合項目の生成。
「レベル 名前. 」
6.SourceTypeが"X"の生成。
「レベル 名前 PIC X(けた数). 」
7.SourceTypeが"9"の生成。
「レベル データ項目名 PIC 9(けた数). 」
8.フリー定義の生成。
「レベル 名前 フリー定義文字. 」
9.条件名の生成。
10.取得したデータ項目定義情報の項目数分繰り返す。
11.上位の項目の位置を設定。
12.上位の項目がなくなるまで繰り返す。
13.可変記号に設定されている文字列の末尾に,「OF 上位の項目の名前」を追加する。
14.上位の項目の位置を設定。
15.単一項目の初期化。

● 可変記号に最上位結合項目またはレコード定義ファイルを指定した場合

@@set 文

```
@@set  @REC  =  @@getdata(@FILE1, "COBOLまたはOOCOBOL",BASIC_ATTR);
```

@REC の内容

@REC の値は，SEWB+/REPOSITORY およびレコード定義での定義内容が展開されます。

生成ソース

```
**  社員情報ファイル      フィールドＯ１
  1  社員情報_CBL.
  2  社員番号_CBL  PIC  X(15).
  2  氏名_CBL.
  3  氏_CBL       PIC  X(30).
  3  名_CBL       PIC  X(30).
  2  入社年_CBL    PIC  9(4).
  2  在職コード_CBL  PIC  X(1).
   88  在職  VALUE  SPACE.
   88  退職  VALUE  'A'.
   :
  MOVE  SPACE  TO  社員番号_CBL  OF  社員情報_CBL.
  MOVE  SPACE  TO  氏_CBL  OF  氏名_CBL  OF  社員情報_CBL.
  MOVE  SPACE  TO  名_CBL  OF  氏名_CBL  OF  社員情報_CBL.
  MOVE  ZERO  TO  入社年_CBL  OF    社員情報_CBL.
```

7.9.13 @@getdate 関数

形 式

@@getdate（）

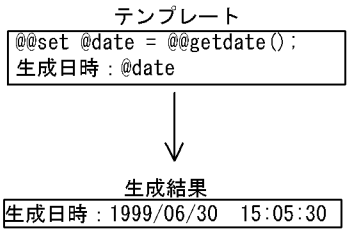
機 能

@@getdate が実行された日付を取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- 日時は，@@set 文の左辺に指定した可変記号に，次の形式で格納される。
  YYYY/MM/DD　HH：MM：SS

使用例



7.9.14 @@getmemo 関数

形 式

@@getmemo（）

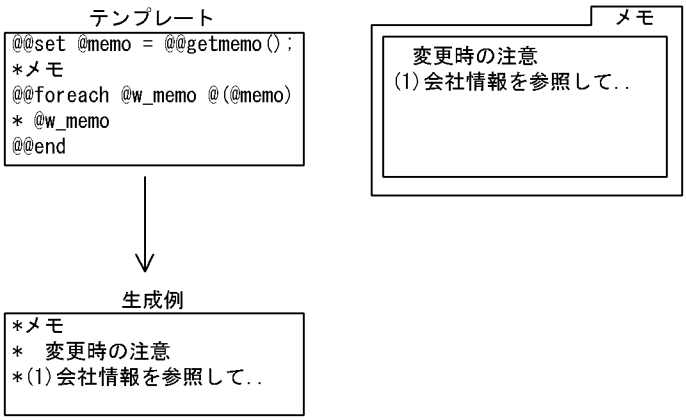
機 能

プログラム定義の [メモ] タブで定義したメモの内容を取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- メモの内容は @@set 文の左辺の可変記号に，一次元の配列で設定される。
- [メモ] タブの改行ごとに別の配列が確保される。

使用例



7.9.15 @@getrfile 関数

形 式

@@getrfile（）

機 能

プログラム定義の [リソースステータス] ダイアログの [ファイル] タブ情報を取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- [ファイル] タブの情報は，@@set 文の左辺の可変記号に，二次元の配列で設定される。
- 一次元目は位置指定の配列で，ファイル単位で確保される。
  配列の設定順序を次に示します。利用するファイル種別（利用種別）ごとに設定されます。

1. テンプレートファイル
2. データ定義ファイル
3. 部品ファイル
4. 論理設計図ファイル
5. マップ定義ファイル
6. XML 文書ファイル

- 二次元目はキーワード指定の配列で，ファイル情報ごとに確保される。

内容	添字に指定するキーワード
ファイル名（フルパス）	FullPath
ファイル名 （ディレクトリ名を含まない）	FileName

内容	添字に指定するキーワード
利用種別	Kind
更新日時	Date

また，利用種別には，次の値が設定されます。

利用種別	設定値
テンプレートファイル	CST
データ定義ファイル	CSD
部品ファイル	CSR
論理設計図ファイル	DAL
マップ定義ファイル	IMP
XML 文書ファイル	XML

- 更新日時は次の形式で設定される。
  YYYY/MM/DD　HH：MM：SS

使用例

テンプレート

```
*リソースファイル
@@set @FILE = @@getrfile();
*種別　更新日時                ファイル名
@@foreach @WFILE(@FILE)
* @WFILE[Kind]  @WFILE[Date]  @WFILE[FullPath]
@@end;
```


生成結果

```
*リソースファイル
*種別　更新日時                  ファイル名
*CST   1999/06/30  13:05:16   N:¥Template¥バッチ2_1_1.cst
*CSD   1999/06/29  09:35:02   N:¥Data_Def¥File¥受注ファイル.csd
```

7.9.16　@@getsign 関数

形　式

@@getsign（）

機　能

プログラム定義の［サイン］タブで定義した名称，作成者および概要を取得するときに使用します。

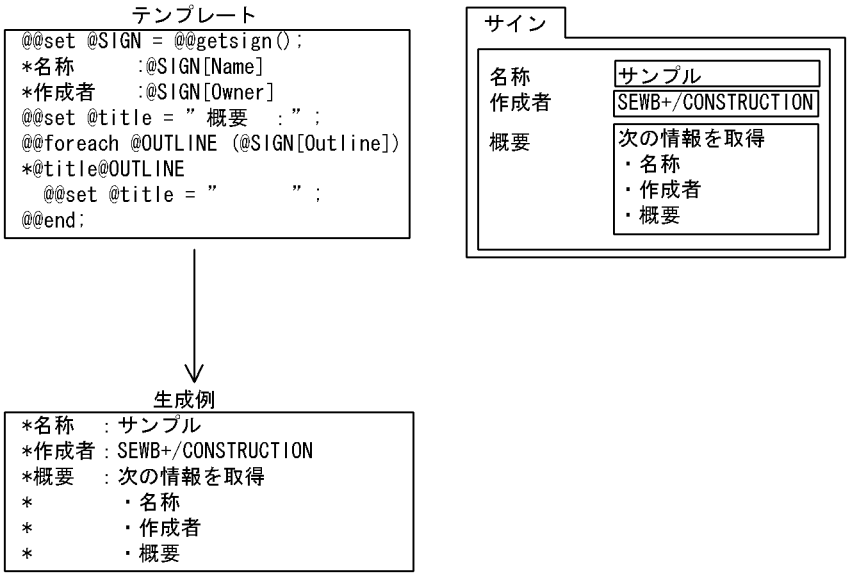
規　則

- @@set 文の右辺に指定する。
- [サイン] タブの情報は，@@set 文の左辺の可変記号に設定される。
- 可変記号の添字には次のキーワードを指定する。

内容	添字に指定するキーワード
名称	Name
作成者	Owner
概要	Outline

- 概要は二次元の配列で，改行ごとに別の配列が確保される。

使用例



7.9.17　@@global 文

形　式

@@global 可変記号；

機　能

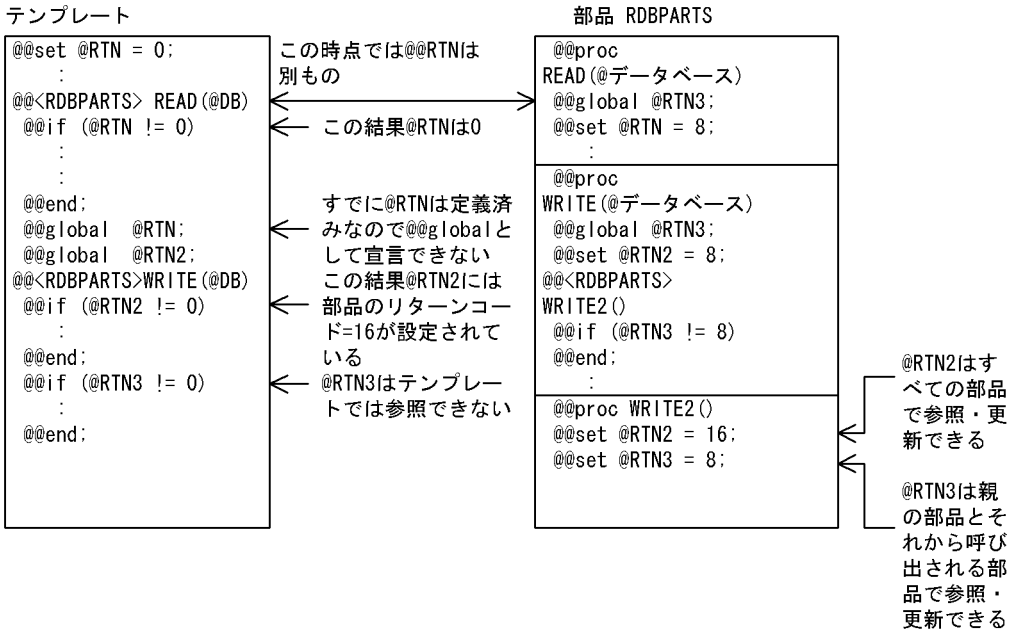
可変記号をテンプレートやテンプレートから呼び出される部品のプロシジャまたは部品のプロシジャ間で共通に使用できる（参照や更新が相互にできる）ように宣言するときに使用します。

部品内で @@global 宣言した可変記号はその部品から呼び出されたほかの部品のプロシジャで参照・更新ができます。テンプレートやその部品が呼び出される以前の部品では参照・更新できません。

規　則

- 可変記号に配列や生成式は指定できない。
- 可変記号は @@global 文の宣言が実行されてから有効となる。
- テンプレート中ですでに定義済みの（使用されている）可変記号を @@global 文で宣言することはできない。したがって，@@interface 文や @@set 文で可変記号を使用する以前に @@global 文で宣言しなければならない。
- @@interface 文や @@set 文で使用する可変記号に配列を指定する場合，@@global 文では配列を指定しない可変記号を指定する。

使用例



7.9.18　@@idlargudata 関数

形 式

@@idlargudata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])

注

[　] は省略できることを表します。

{　} はどちらか一つを記述することを表します。

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義から，オペレーション引数情報のデータ項目の定義情報を取得するときに指定します。

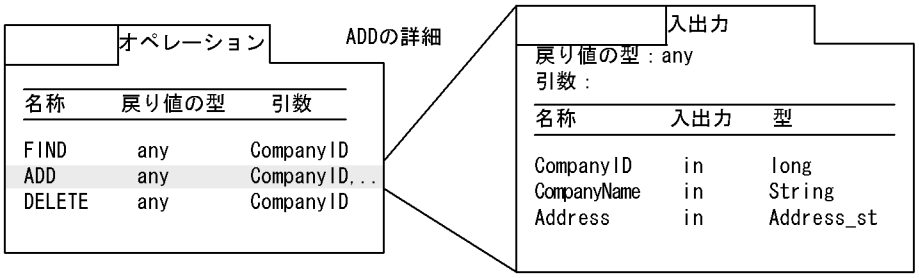
規 則

- @@set 文の右辺に指定する。
- オペレーション情報は，@@set 文の左辺の可変記号に二次元の配列で設定される。
- 可変記号は，@@idloperations 関数で取り出したオブジェクト定義情報を指定した可変記号に添字を付けて指定する。または，@@interface 文で ATTR=OPERATION を指定した可変記号を指定する。
- リポジトリの言語区分によって，言語別詳細情報の対象言語を選択できる。
- リポジトリの言語区分は，SEWB+/REPOSITORY に登録されている言語区分を文字定数で指定する。省略した場合，@@lang 文の FOR_REPOSITORY 句の指定に従う。
- BASIC_ATTR を指定すると，データ項目の基本属性タブ情報を取得できる。取り出せる情報は，@@getdata 関数と同じです。詳細は，「7.9.12　@@getdata 関数」を参照してください。
- オペレーション引数情報のデータ項目が結合項目の場合，Level は 1 から昇順に設定される。また，単項目の場合，Level は 1 が設定される。
- リポジトリから入力したデータ項目以外の引数情報は，DataItemName 以降が NULL データで設定され，Level は 0 となる。

使用例 1 （BASIC_ATTR オペランド指定なし）

BASIC_ATTR オペランド指定がない場合に，オペレーション引数情報のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN **での定義情報**



データ項目情報 1 ／ 2

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	CompanyID	整数データ	8			0
2	0						
3	1	Address	結合データ	37			0
4	2	ZipCode	英数字文字列データ	7			3

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
5	2	AddressName	英数字文字列データ	30			3

データ項目情報の続き 2 ／ 2

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
1	CompanyID	long						
2								
3	Address							
4	ZipCode	char						
5	AddressName	char						

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = {ATTR=OPERATION};
@@set @ARGDATAS = @@idlargudata(@オペレーション);
    :
@@*=====================================================
@@* CORBA オペレーション引数のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
@@foreach @dt (@ARGDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]
@@end;
```

生成ソース

プログラム定義で，SEWB+/CS-DESIGN のオブジェクト定義情報の ADD オペレーションが選択されたときの展開例を次に示します。

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
//  1  CompanyID  long  8
//  0
//  1  Address
//  2  ZipCode  char  7
//  2 AddressName  char  30
```

使用例 2（BASIC_ATTR オペランド指定あり）

BASIC_ATTR オペランド指定がある場合に，オペレーション引数情報のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報

使用例 1 と同じ情報を使用します。

データ項目情報

使用例 1 と同じ情報を使用します。また，基本属性の指定内容を次に示します。

DataItemName	LangType Count	StdName	Furigana	Comment	FieldCount	BasicField
CoumpanyID	4	会社 ID	カイシャ ID※	社員 ID	20	
Address	4	住所情報	ジュウショジョウホウ※	住所情報	20	
ZipCode	4	郵便番号	ユウビンバンゴウ※	郵便番号	20	
AddressName	4	住所	ジュウショ※	住所	20	

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

注※

半角で表示されます。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = {ATTR=OPERATION};
@@set @ARGDATAS = @@idlargudata(@オペレーション,,BASIC_ATTR);
    :
@@*=====================================================
@@* CORBA オペレーション引数のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ   コメント
@@foreach @dt (@ARGDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]    //@dt[Comment,1]
@@end;
```

生成ソース

プログラム定義で，SEWB+/CS-DESIGN のオブジェクト情報の ADD オペレーションが選択されたときの展開例を次に示します。

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ    コメント
//  1  CompanyID  long  8    //社員ＩＤ
//  0
//  1  Address          //住所情報
//  2  ZipCode  char  7   //郵便番号
//  2  AddressName  char  30   //住所
```

7.9.19 @@idlarguments 関数

形 式

@@idlarguments（可変記号）

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義のオペレーション引数情報を取得するときに使用します。

規 則

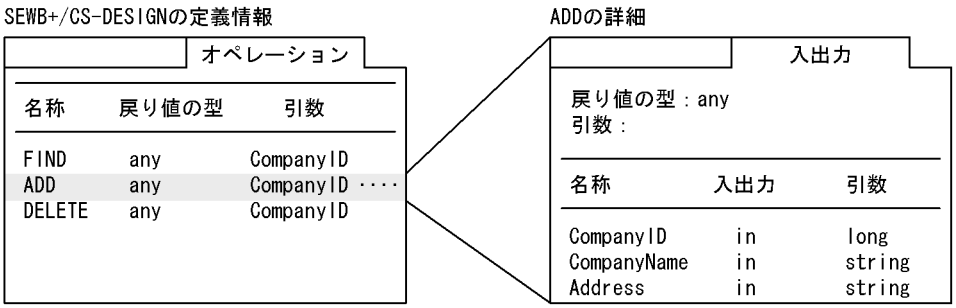
- @@set 文の右辺に指定する。
- オペレーション情報は @@set 文の左辺の可変記号に，二次元の配列で設定される。
- 可変記号は，@@idloperations 関数で取り出したオブジェクト定義情報を指定した可変記号に添字を付けて指定する。または，@@interface 文で ATTR=OPERATION を指定した可変記号を指定する。
  取得できるオブジェクト定義のオペレーション引数情報を次に示します。

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定するキーワード	備　　考
引数名	Name	
モード	Mode	次に示す引数の属性のどれか。"in"（入力），"out"（出力），"inout"（入出力）
型	TypeString	型が設定される。ユーザ定義型の場合は，スコープ解決子は付けられない。2
ユーザ定義型	UserDefType	SEWB+/CS-DESIGN で SEWB+/REPOSITORY の結合項目を参照したときに IDL に取り込まれるユーザ定義型名。CORBA 基本型のときは" "（0 文字の文字列）が設定される。
スコープ解決子付き名称	ScopedName	対象の IDL の引数を最上位の IDL モジュールから修飾した包括的な名前。

使用例

オペレーション引数情報を取得するときの，SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報



テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = { ATTR= OPERATION };
@@set @ARGS = @@idlarguments(@オペレーション);
    :
@@*=====================================================
@@* CORBA オペレーション引数展開
@@*=====================================================
@@set @cnt = @@count(@ARGS);
@@if (@cnt > 0)
    @@set @Mod = ",";
    @@set @j = 1;
    @@foreach @arg (@ARGS)
      @@set @ItemName = @arg[Name];
      @@if (@j == @cnt)
          @@set @Mod = "";
      @@end;
      @@*=======================================
      @@* IDLマッピング規則に従い，引数の属性を決定する
      @@*=======================================
      @@switch (@arg[TypeString])
       @@case "string":
        const char* @ItemName.@Mod
          @@break;
       @@case "long":
        CORBA::Long @ItemName.@Mod
          @@break;
       @@default:
          @@set @TypeString6 = @@str(@arg[TypeString],1,6);
          @@if (@TypeString6 eq "string")
        const char* @ItemName.@Mod
          @@end;
          @@break;
      @@end;
      @@set @j = @j + 1;
    @@end;
  @@end;
@@*
```

生成ソース

プログラム定義で SEWB+/CS-DESIGN のオブジェクト定義情報の ADD オペレー

ションが選択されたときの展開例を次に示します。

```
CORBA::Long CompanyID,
const char* CompanyName,
const char* Address
```

7.9.20 @@idlattrdata 関数

形 式

@@idlattrdata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])

注

［ ］は省略できることを表します。
{ } は項目のどちらか一つを記述することを表します。

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義情報から，属性情報のデータ項目の定義情報を取得するときに使用します。

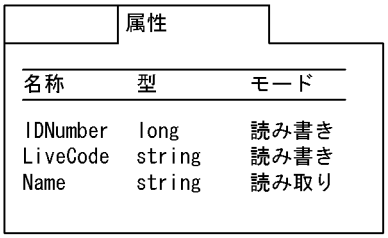
規 則

- @@set 文の右辺に指定する。
- @@interface 文で ATTR=INTERFACE，または ATTR=OPERATION を指定した可変記号を指定する。または，@@idlinterfaces 関数で取り出した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に，添字を付けて指定する。
- リポジトリの言語区分によって，言語別詳細情報の対象言語を選択できる。
- リポジトリの言語区分は，SEWB+/REPOSITORY に登録されている言語区分を文字定数で指定する。省略した場合，@@lang 文の FOR_REPOSITORY 句の指定に従う。
- BASIC_ATTR を指定すると，データ項目の基本属性タブ情報を取得できる。取り出せる情報は，@@getdata 関数と同じです。詳細は，「7.9.12 @@getdata 関数」を参照してください。
- 属性情報のデータ項目が結合項目の場合，Level は 1 から昇順に設定される。また，単項目の場合，Level は 1 が設定される。
- リポジトリから入力したデータ項目以外の属性情報は，DataItemName 以降が NULL データで設定され，Level は 0 となる。

使用例 1 （BASIC_ATTR オペランド指定なし）

BASIC_ATTR オペランド指定がない場合に，属性情報のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報，テンプレート，および生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報



データ項目情報 1 ／ 2

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	IDNumber	整数データ	8			0
2	0						
3	1	Name	結合データ				0
4	2	FamilyName	英数字文字列データ	15			3
5	2	FirstName	英数字文字列データ	15			3

データ項目情報の続き 2 ／ 2

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
1	IDNumber	long						
2								
3	Name							
4	FamilyName	char						
5	FirstName	char						

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@set @ATTRDATAS = @@idlattrdata(@インタフェース);
    :
@@*=====================================================
@@* CORBA 属性情報のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
@@foreach @dt (@ATTRDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]
@@end;
```

生成ソース

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
//  1  IDNumber  long  8
//  0
//  1  Name
//  2  FamilyName  char  15
//  2  FirstName  char  15
```

使用例 2（BASIC_ATTR オペランド指定あり）

BASIC_ATTR オペランド指定がある場合に，属性情報のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレート，および生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報

使用例 1 と同じ情報を使用します。

データ項目情報

使用例 1 と同じ情報を使用します。また，基本属性の指定内容を次に示します。

DataItemName	LangType Count	StdName	Furigana	Comment	FieldCount	BasicField
IDNumber	4	ID 番号	ID バンゴウ※	ID 番号	20	
Name	4	氏名	シメイ※	氏名	20	
FamilyName	4	名字	ミョウジ※	名字	20	
FirstName	4	名前	ナマエ※	名前	20	

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

注※
　半角で表示されます。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@set @ATTRDATAS = @@idlattrdata(@インタフェース,,BASIC_ATTR);
    :
@@*=====================================================
@@* CORBA 属性情報のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ    コメント
@@foreach @dt (@ATTRDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]    //@dt[Comment,1]
@@end;
```

生成ソース

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ     コメント
//  1  IDNumber  long  8    // ＩＤ番号
//  0
//  1  Name          //氏名
//  2  FamilyName  char  15    //名字
//  2  FirstName  char  15    //名前
```

7.9.21 @@idlattributes 関数

形　式

@@idlattributes（可変記号）

機　能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義の属性定義情報を取得するときに使用します。

規　則

- @@set 文の右辺に指定する。
- オペレーション情報は @@set 文の左辺の可変記号に，二次元の配列で設定される。
- @@interface 文で ATTR=INTERFACE，または ATTR=OPERATION を指定した可変記号を指定する。または，@@idlinterfaces 関数で取り出した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に，添字を付けて指定する。

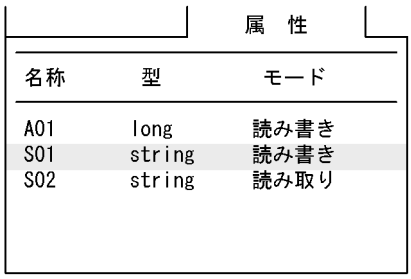
  取得できるオブジェクト定義の属性定義情報を次に示します。

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定するキーワード	備　　考
属性名	Name	
readonly 属性	Mode	読み取り専用のとき "readonly" が設定される。それ以外は "　"（0 文字の文字列）が設定される。
型文字列	TypeString	型文字列が設定される。ユーザ定義型の場合は，スコープ解決子は付けられない。
ユーザ定義型	UserDefType	SEWB+/CS-DESIGN で SEWB+/REPOSITORY の結合項目を参照したときに IDL に取り込まれるユーザ定義型名。CORBA 基本型のときは "　"（0 文字の文字列）が設定される。
スコープ解決子付き名称	ScopedName	対象の IDL の属性を最上位の IDL モジュールから修飾した，包括的な名前。

使用例

属性定義情報を取得するときの，SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN **での定義情報**



属　性

名称	型	モード
A01	long	読み書き
S01	string	読み書き
S02	string	読み取り

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@set @ATTRS = @@idlattributes(@ｲﾝﾀﾌｪｰｽ);
    :
@@*=====================================================
@@* CORBA 属性操作宣言
@@* 読み書きモードのときは属性の参照・設定オペレーション宣言を展開
@@* 読み込みモードのときは属性の参照オペレーション宣言を展開
@@*=====================================================
@@foreach @atr (@ATTRS)
      @@*===================================
      @@* IDLマッピング規則に従い,
      @@* 引数の属性・戻り値の型を決定し展開する
      @@*===================================
      @@switch (@atr[TypeString])
      @@case "string":
          @@set @Ret = "char*";
          @@set @Arg = "const char*";
          @@break;
      @@case "long":
          @@set @Ret = "CORBA::Long";
          @@set @Arg = "CORBA::Long";
          @@break;
      @@end;
      @@*
      @@* 値参照用オペレーション宣言 ----------------------
      @Ret @atr[Name](); // @atr[Name] 値参照
      @@if (@atr[Mode] eq "")
      @@*
      @@* 値設定用オペレーション宣言 ----------------------
      void @Ret @atr[Name](@Arg val); // @atr[Name] 値設定
      @@end;
@@end;
@@*
```

生成ソース

読み書き属性の A01，S01 に対して参照・設定のオペレーション宣言が展開されます。読み込み属性の S02 に対しては参照用のオペレーション宣言が展開されます。

```
:
        CORBA::Long A01(); // A01 値参照
        void CORBA::Long A01(CORBA::Long val); // A01 値設定
```

```
char* S01(); // S01 値参照
void char* S01(const char* val); // S01 値設定
char* S02(); // S02 値参照
```

7.9.22 @@idlexcepdata 関数

形 式

@@idlexcepdata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])

注

[ ] は省略できることを表します。
{ } は項目のどちらか一つを記述することを表します。

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義から，オペレーション例外情報のデータ項目の定義情報を取得するときに使用します。

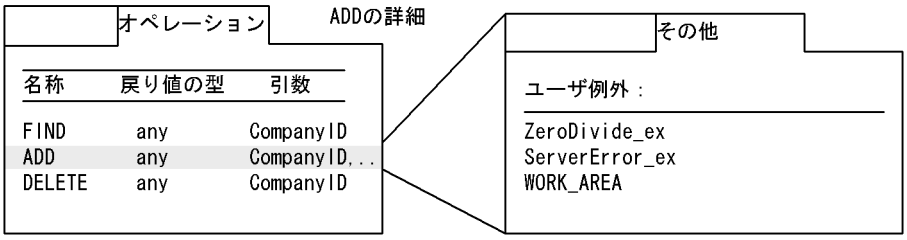
規 則

- @@set 文の右辺に指定する。
- 可変記号は @@idloperations 関数で取り出した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に添字を付けて指定する。または，@@interface 文で ATTR=OPERATION を指定した可変記号を指定する。
- オペレーション情報は，@@set 文の左辺の可変記号に二次元の配列で設定される。
- リポジトリの言語区分によって，言語別詳細情報の対象言語を選択できる。
- リポジトリの言語区分は，SEWB+/REPOSITORY に登録されている言語区分を文字定数で指定する。省略した場合は，@@lang 文の FOR_REPOSITORY 句の指定に従う。
- BASIC_ATTR を指定すると，データ項目の基本属性タブ情報を取得できる。
  取り出せる情報は，@@getdata 関数と同じです。詳細は，「7.9.12 @@getdata 関数」を参照してください。
- オペレーション例外情報のデータ項目が結合項目の場合，Level は 1 から昇順に設定される。また，単項目の場合，Level は 1 が設定される。
- リポジトリから入力したデータ項目以外のユーザ例外は，DataItemName 以降が NULL データで設定され，Level は 0 となる。

使用例 1 （BASIC_ATTR オペランド指定なし）

BASIC_ATTR オペランド指定がない場合に，オペレーション例外情報のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレート，および生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報



データ項目情報 1 ／ 2

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	ZeroDivide	結合データ	8			0
2	2	Text1	英数字文字列データ	256			1
3	2	Text2	英数字文字列データ	256			1
4	1	ServerError	結合データ				0
5	2	ErrorID	英数字文字列データ	10			4
6	2	ErrorMessage	英数字文字列データ	256			4
7	0						

データ項目情報の続き 2 ／ 2

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
1	ZeroDivide							
2	Text1	char						
3	Text2	char						
4	ServerError							
5	ErrorID	char						
6	ErrorMessage	char						
7								

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = {ATTR=OPERATION};
@@set @EXCDATAS = @@idlexcepdata(@オペレーション);
    :
@@*=====================================================
@@* CORBA ユーザ例外情報のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
@@foreach @dt (@EXCDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]
@@end;
```

生成ソース

プログラム定義で SEWB+/CS-DESIGN のオブジェクト定義情報の ADD オペレーションが選択されたときの展開例を次に示します。

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
//  1  ZeroDivide
//  2  Text1  char  256
//  2  Text2  char  256
//  1  ServerError
//  2  ErrorID  char  10
//  2  ErrorMessage  char  256
//  0
```

使用例 2（BASIC_ATTR オペランド指定あり）

BASIC_ATTR オペランド指定がある場合に，オペレーション例外情報のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレート，および生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報

使用例 1 と同じ情報を使用します。

データ項目情報

使用例 1 と同じ情報を使用します。また，基本属性の指定内容を次に示します。

DataItemName	LangType Count	StdName	Furigana	Comment	FieldCount	BasicField
ZeroDivide	4	ゼロ除算エラー	ゼロジョサン※	ゼロ除算エラー	20	
Text1	4	エラー情報 1		エラー情報 1	20	

DataItemName	LangType Count	StdName	Furigana	Comment	FieldCount	BasicField
Text2	4	エラー情報２	シメイ※	エラー情報２	20	
ServerError	4	サーバエラー情報	サーバエラー※	サーバエラー情報	20	
ErrorID	4	エラー ID	シメイ※	エラー ID	20	
ErrorMessage	4	エラーメッセージ	エラーメッセージ※	エラーメッセージ	20	

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

注※

半角で表示されます。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = {ATTR=OPERATION};
@@set @EXCDATAS = @@idlexcepdata(@オペレーション,,BASIC_ATTR);
    :
@@*=====================================================
@@* CORBA ユーザ例外情報のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ    コメント
@@foreach @dt (@EXCDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]    //@dt[Comment,1]
@@end;
```

生成ソース

プログラム定義で SEWB+/CS-DESIGN のオブジェクト定義情報の ADD オペレーションが選択されたときの展開例を次に示します。

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ    コメント
//  1  ZeroDivide          //ゼロ除算エラー
//  2  Text1  char  256    //エラー情報１
//  2  Text2  char  256    //エラー情報２
//  1  ServerError          //サーバエラー情報
//  2  ErrorID  char  10    //エラーＩＤ
//  2  ErrorMessage  char  256    //エラーメッセージ
//  0
```

7.9.23 @@idlexceptions 関数

形 式

@@idlexceptions（可変記号）

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義のオペレーション例外情報を取得するときに使用します。

規 則

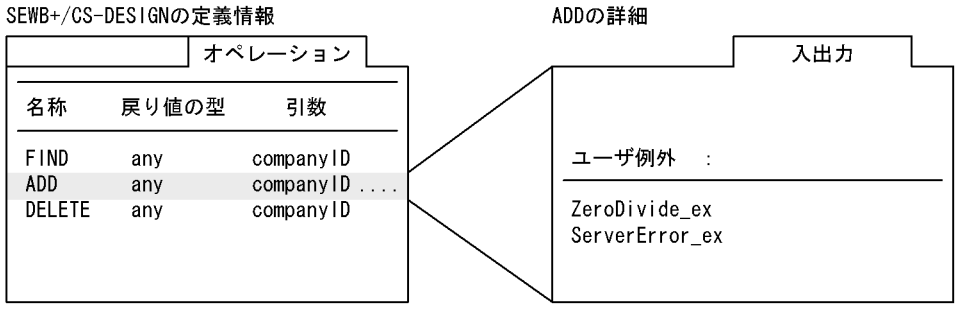
- @@set 文の右辺に指定する。
- オペレーション情報は @@set 文の左辺の可変記号に，二次元の配列で設定される。
- 可変記号は，@@idloperations 関数で取り出したオブジェクト定義情報を指定した可変記号に添字を付けて指定する。または，@@interface 文で ATTR=OPERATION を指定した可変記号を指定する。
  取得できるオブジェクト定義のオペレーション例外情報を次に示します。

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定するキーワード	備　　考
例外名	Name	
スコープ解決子付き名称	ScopedName	対象の IDL の属性を最上位の IDL モジュールから修飾した，包括的な名前。

使用例

オペレーション例外情報を取得するときの，SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報



テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = {ATTR=OPERATION};
@@set @EXCEPTS = @@idlexceptions(@オペレーション);
    :
@@*=====================================================
@@* CORBA ユーザ例外処理展開
@@*=====================================================
@@set @cnt = 0;
@@set @cnt = @@count(@EXCEPTS);
@@if (@cnt > 0)
  // 以下の例外処理をthrowします
  @@foreach @ex (@EXCEPTS)
  //   @ex[ScopedName]
  @@end;
@@end;
@@*
```

生成ソース

プログラム定義で SEWB+/CS-DESIGN のオブジェクト定義情報の ADD オペレーションが選択されたときの展開例を次に示します。

```
    :
// 以下の例外処理をthrowします
//    ::comm::ZeroDivide__ex
//    ::comm::ServerError__ex
```

7.9.24 @@idlinterface 関数

形　式

@@idlinterface（可変記号）

機　能

SEWB+/CS-DESIGN で定義した論理設計図のオブジェクト定義情報を取得するときに使用します。

規　則

- @@set 文の右辺に指定する。
- オブジェクト定義情報は @@set 文の左辺の可変記号に，一次元の配列で設定される。
- @@interface 文で ATTR=INTERFACE，または ATTR=OPERATION を指定した可変記号を指定する。または，@@idlinterfaces 関数で取り出した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に，添字を付けて指定する。

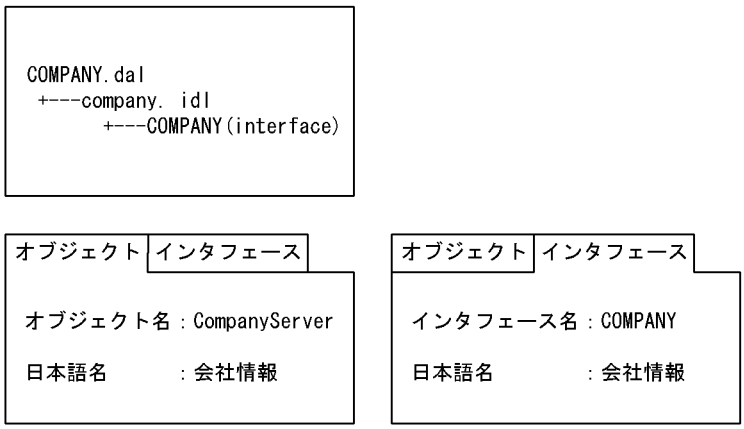
取得できるオブジェクト定義情報を次に示します。

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定するキーワード	備　　考
インタフェース名	Name	
日本語名	Alias	
コメント	Comment	
オブジェクト名	ObjectName	
オブジェクト日本語名	ObjectAliasName	
スコープ解決子付き名称	ScopedName	対象の IDL インタフェース（interface）を最上位の IDL モジュールから修飾した，包括的な名前。
IDL ファイル名	IDLFileName	対象の IDL インタフェース（interface）が属する IDL ファイル名。
モジュール名	ModuleName	対象の IDL インタフェース（interface）が属する，直上の IDL モジュール名。
モジュール日本語名	ModuleAliasName	

使用例

SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN **での定義情報**



テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@interface @実装クラス = {COMMENT="実装クラス名を入力してください"};
@@set @IFC = @@idlinterface(@インタフェース);
     :
    // オブジェクトの生成
    @IFC[Name]._ptr impl = new @実装クラス("@IFC[ObjectName]");
     :
```

生成ソース

プログラム定義で「@ インタフェース」に SEWB+/CS-DESIGN の COMPANY オブジェクト，「@ 実装クラス」に [パラメタ] タブから「CCompany」を入力したときの生成例を次に示します。

```
  :
// オブジェクトの生成
COMPANY_ptr impl = new CCompany("CompanyServer");
 :
```

7.9.25 @@idlinterfaces 関数

形　式

@@idlinterfaces（可変記号）

機　能

SEWB+/CS-DESIGN で定義した論理設計図のオブジェクト定義情報を取得するときに使用します。

規　則

- @@set 文の右辺に指定する。
- オブジェクト定義情報は @@set 文の左辺の可変記号に，二次元の配列で設定される。
- @@interface 文で ATTR=IDL を指定した可変記号を指定する。
  取得できるオブジェクト定義情報を次に示します。

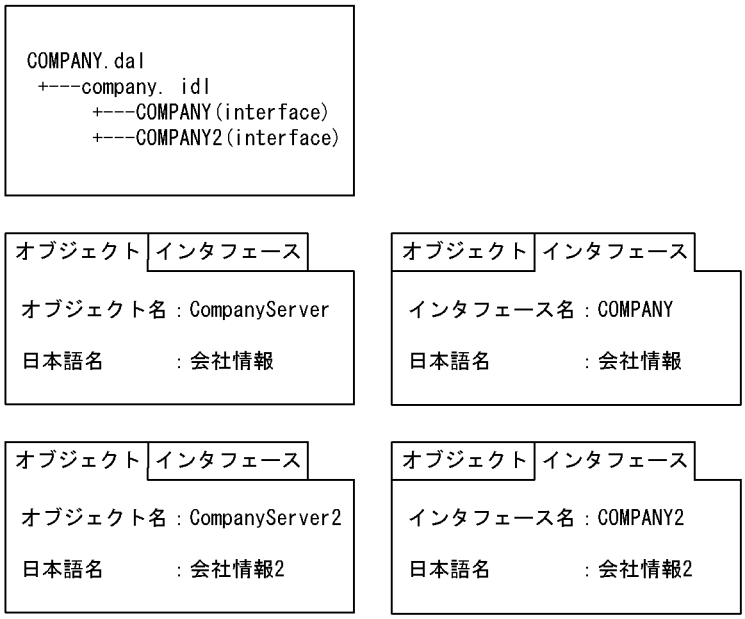
SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定する キーワード	備　　考
インタフェース名	Name	
日本語名	Alias	
コメント	Comment	
オブジェクト名	ObjectName	
オブジェクト日本語名	ObjectAliasName	

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定する キーワード	備　　考
スコープ解決子付き名称	ScopedName	対象の IDL インタフェース（interface）を最上位の IDL モジュールから修飾した，包括的な名前。
IDL ファイル名	IDLFileName	対象の IDL インタフェース（interface）が属する IDL ファイル名。
モジュール名	ModuleName	対象の IDL インタフェース（interface）が属する，直上の IDL モジュール名。
モジュール日本語名	ModuleAliasName	

使用例

SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN **での定義情報**



テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @ＩＤＬ情報 = {ATTR=IDL};
@@interface @実装クラス = {COMMENT="実装クラス名を入力してください"};
@@set @IDL = @@idlinterfaces(@ＩＤＬ情報);
     :
@@foreach @IFC (@IDL)
    // オブジェクトの生成
    @IFC[Name]._ptr impl = new @実装クラス("@IFC[ObjectName]");
@@end
     :
```

生成ソース

プログラム定義で「@ インタフェース」に SEWB+/CS-DESIGN の COMPANY オブジェクト，「@ 実装クラス」に [パラメタ] タブから「CCompany」を入力したときの生成例を次に示します。

```
 :
// オブジェクトの生成
COMPANY_ptr  impl = new CCompany("CompanyServer");
COMPANY2_ptr impl = new CCompany("CompanyServer2");
 :
```

7.9.26 @@idlopedata 関数

形 式

@@idlopedata(可変記号 [{ ," リポジトリの言語区分 " [,BASIC_ATTR] | ,,BASIC_ATTR }])

注

[ ] は省略できることを表します。

{ } は項目のどちらか一つを記述することを表します。

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義情報から，オペレーション情報の戻り値のデータ項目の定義情報を取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- @@interface 文で ATTR=INTERFACE，または ATTR=OPERATION を指定した可変記号を指定する。または，@@idlinterfaces 関数で取り出した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に，添字を付けて指定する。
- オペレーション情報は，@@set 文の左辺の可変記号に二次元の配列で指定される。
- リポジトリの言語区分によって，言語別詳細情報の対象言語を選択できる。省略した場合は，@@lang 文の FOR_REPOSITORY 句の指定に従う。
- リポジトリの言語区分は，SEWB+/REPOSITORY に登録されている言語区分を

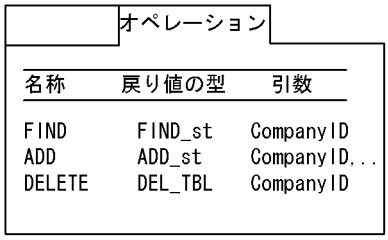
文字定数で指定する。省略した場合，@@lang 文の FOR_REPOSITORY 句の指定に従う。

- BASIC_ATTR を指定すると，データ項目の基本属性タブ情報を取得できる。
  取り出せる情報は，@@getdata 関数と同じです。詳細は，「7.9.12　@@getdata 関数」を参照してください。
- 戻り値のデータ項目が結合項目の場合，Level は 1 から昇順に設定される。また，単項目の場合，Level は 1 が設定される。
- リポジトリから入力したデータ項目以外の戻り値情報は，DataItemName 以降が NULL データで設定され，Level は 0 となる。

使用例 1（BASIC_ATTR オペランド指定なし）

BASIC_ATTR オペランド指定がない場合に，オペレーション情報の戻り値のデータ項目定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN **での定義情報**



データ項目情報 1 ／ 2

項番	Level	DataItemName	Type	Digit	FloatDigit	RepeatLevel	ParentRow
1	1	FIND	結合データ				0
2	2	RTN_CODE	整数データ	8			1
3	1	ADD	結合データ				0
4	2	FirstName	整数データ	8			3
5	0						

データ項目情報の続き 2 ／ 2

項番	SourceName	SourceType	TypeModifyInfo	InitValue	Field	AvailableCount	AvailableName	AvailableValue
1	FindRtnTbl							
2	ReturnCode	long						
3	AddRtnTbl							
4	ReturnCode	long						
5								

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@set @OPEDATAS = @@idlopedata(@インタフェース);
   :
@@*=================================================
@@* CORBAオペレーション情報のデータ項目展開
@@*=================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
@@foreach @dt (@OPEDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]
@@end;
```

生成ソース

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ
//  1  FindRtnTbl
//  2  ReturnCode long 8
//  1  AddRtnTbl
//  2  ReturnCode long 8
//  0
```

使用例 2（BASIC_ATTR オペランド指定あり）

BASIC_ATTR オペランド指定がある場合に，オペレーション情報の戻り値のデータ項目の定義情報を取得するときの SEWB+/CS-DESIGN での定義情報，データ項目情報と，テンプレート，および生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報

使用例 1 と同じ情報を使用します。

データ項目情報

使用例１と同じ情報を使用します。また，基本属性の指定内容を次に示します。

DataItemName	LangType Count	StdName	Furigana	Comment	FieldCount	BasicField
FIND	4	検索結果	ケンサクケッカ※	検索結果情報テーブル	20	
RTN_CODE		リターンコード	リターンコード※	リターンコード		
ADD	4	追加結果	ツイカケッカ※	追加結果情報テーブル	20	
FirstName	4	リターンコード	リターンコード※	リターンコード	20	

注

表中のキーワードは説明上，付けているものです。また，空欄は，画面上で値が設定されていないことを示します。

注※

半角で表示されます。

テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@set @OPEDATAS = @@ idlopedata(@インタフェース,,BASIC_ATTR);
    :
@@*=====================================================
@@* CORBAオペレーション情報のデータ項目展開
@@*=====================================================
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ    コメント
@@foreach @dt (@OPEDATAS)
//  @dt[Level]  @dt[SourceName]  @dt[SourceType]  @dt[Digit]    //@dt[Comment,1]
@@end;
```

生成ソース

```
//  データ項目情報一覧
//  レベル  データ項目名  タイプ  長さ    コメント
//  1  FindRtnTbl          // 検索結果情報テーブル
//  2  ReturnCode long 8    // リターンコード
//  1  AddRtnTbl          //氏名
// ReturnCode long 8    // リターンコード
//  0
```

7.9.27 @@idloperations 関数

形 式

@@idloperations（可変記号）

機 能

SEWB+/CS-DESIGN で定義した論理設計図のオブジェクト定義のオペレーション定義情報を取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- オペレーション定義情報は @@set 文の左辺の可変記号に，二次元の配列で設定される。
- @@interface 文で ATTR=INTERFACE，または ATTR=OPERATION を指定した可変記号を指定する。または，@@idlinterfaces 関数で取り出した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に，添字を付けて指定する。

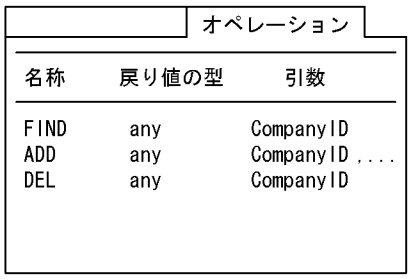
  取得できるオペレーション定義情報を次に示します。

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定するキーワード	備　考
オペレーション名	Name	
オペレーション日本語名	Alias	
コメント	Comment	
Oneway 属性	Oneway	オペレーションの処理結果をクライアントに返さないとき "oneway" が設定される。それ以外のときは " "（0 文字の文字列）が設定される。
戻り値の型	ReturnType	戻り値の型が設定される。ユーザ定義型の場合は，スコープ解決子は付けられない。
ユーザ定義型	UserDefType	SEWB+/CS-DESIGN で REPOSITORY の結合項目を参照したときに IDL 上に取り込まれるユーザ定義型名。CORBA 基本型のときは " "（0 文字の文字列）が設定される。
スコープ解決子付き名称	ScopedName	対象の IDL オペレーションを最上位の IDL モジュールから修飾した包括的な名前。

使用例

オペレーション定義情報を取得するときの，SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を示します。

SEWB+/CS-DESIGN での定義情報



テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @インタフェース = {ATTR=INTERFACE};
@@set @OPES = @@idloperations(@インタフェース);
class C001 {
public:
@@foreach @ope (@OPES)
      @@*=======================================
      @@* IDLマッピング規則に従い、戻り値の型を決定する
      @@*=======================================
      @@switch (@ope[ReturnType])
      @@case "long":
          @@set @Ret = "CORBA::Long";
          @@break;
      @@case "any":
          @@set @Ret = "CORBA::Any";
          @@break;
      @@end;
  @Ret @ope[Name](
    @@* 引数の展開
          :
  );
@@end;
};
```

生成ソース

```
class C001 {
public:
  CORBA::Any FIND(
            :
  );
  CORBA::Any ADD(
            :
  );
  CORBA::Any DEL(
            :
  );
```

引数情報の展開例については「7.9.19　@@idlarguments 関数」を参照してください。

7.9.28 @@idlreqcontext 関数

形 式

@@idlreqcontext（可変記号）

機 能

SEWB+/CS-DESIGN で作成した論理設計図のオブジェクト定義からリクエスト・コンテキスト情報を取得するときに使用します。

規 則

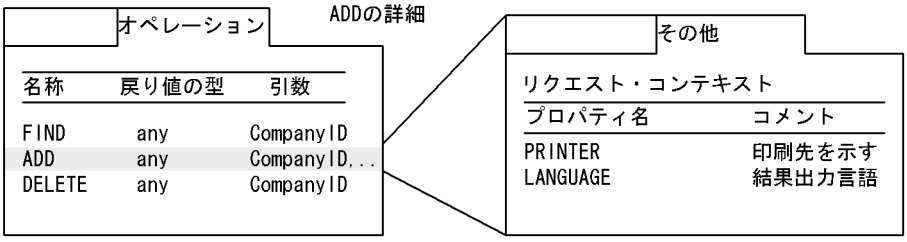
- @@set 文の右辺に指定する。
- 可変記号は，@@idloperations 関数で取得した SEWB+/CS-DESIGN のオブジェクト定義情報を指定した可変記号に添字を付けて指定する。または，ATTR=OPERATION を指定した @@interface 文の可変記号を指定する。
- オペレーション情報は，@@set 文の左辺の可変記号に二次元の配列で設定される。取得できるオブジェクト定義のリクエスト・コンテキスト情報を次に示します。

SEWB+/CS-DESIGN オブジェクト定義情報	添字に指定するキーワード	備考
プロパティ名	Name	
コメント	Comment	

使用例

リクエスト・コンテキスト情報を取得するときの，SEWB+/CS-DESIGN での定義情報と，テンプレートおよび生成ソースの関係を次に示します。

SEWB+/CS-DESIGN での定義情報



テンプレート

```
@@lang C EXTENSION = ".cpp";
@@interface @オペレーション = {ATTR=OPERATION};
@@set @CONTEXTS = @@idlreqcontext(@オペレーション);
    :
@@*=====================================================
@@* CORBAリクエスト・コンテキスト情報展開
@@*=====================================================
@@set @cnt = 0;
@@set @cnt = @@count(@CONTEXTS);
@@if (@cnt > 1)
  // リクエスト情報一覧
  @@foreach @CONTEXT (@CONTEXTS)
  //   @CONTEXT[Name] : @CONTEXT[Comment]
  @@end;
@@end;
@@*
```

生成ソース

プログラム定義で SEWB+/CS-DESIGN のオブジェクト定義情報の ADD オペレーションが選択されたときの展開例を次に示します。

```
// リクエスト情報一覧
//PRINTER：印刷先を示す
// LANGUAGE：結果出力言語
```

7.9.29 @@if 文

形 式

@@if（条件式 1)
　文 1
[[@@elseif（条件式 2)
　文 2],…]
[@@else
　文 3]
@@end;

注

［ ］は省略できることを表します。

機 能

条件式の真偽により，処理の実行を制御するときに使用します。

規 則

- @@if 文の条件式 1 が真のときに，文 1 が実行される。
- @@if 文の条件式 1 が偽のときは，@@elseif の条件式 2 が実行される。
- @@if 文の条件式 1，および @@elseif の条件式 2 がすべて偽のときは，@@else の文 3 が実行される。

使用例

```
@@if (@MESSAGE eq "AC")  @@* メッセージ出力オプションありのとき
  ACCEPT START-DATE FROM DATE
  DISPLAY ' @PROG_ID: STARTED ' START-DATE
@@elseif (@MESSAGE eq "96")
  MOVE  '96-03-18' TO START-DATE
@@else
  MOVE  '95-02-27' TO START-DATE
@@end;
```

7.9.30 @@interface 文

「7.5.3 インタフェース定義部」を参照してください。

7.9.31 @@itemlist 関数

形 式

@@itemlist（可変記号）

機 能

表・ビューを構成する列名の埋め込み変数を配列の値として取得するときに使用します。

データ定義で最上位項目を取り込んでいる場合，最上位項目を直接構成しているデータ項目の言語別の名前が取得できます。このとき，言語別の名前が指定されていない場合は，データ項目名が取得されます。

また，データ定義でレコード定義ファイルを取り込んでいる場合，2 番目のデータ項目（項番 2 の項目が該当します）と同じレベル番号が指定されたデータ項目の言語別の名前が取得できます。言語別の名前が指定されていない場合は，データ項目名が取得されます。

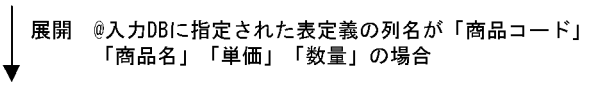
規 則

- 言語種別が C，または COBOL の場合に指定する。
- @@set 文の右辺に指定する。
- 可変記号は @@interface 文で ATTR=DB（RDB 定義）を指定した可変記号を指定する。
- 言語種別が C の場合，レコード定義を参照するデータ定義は指定できない。

使用例

```
@@set @item = @@itemlist(@入力DB);
@@set @itemcnt = @@count(@item);

EXEC SQL
  FETCH  CR01  INTO
  :I1-@item[1]
  @@set  @i = 2;
  @@while(@i <= @itemcnt)
                ,:I1-@item[@i]
  @@set @i = @i + 1;
  @@end;
END-EXEC
```



生成結果

```
EXEC SQL
FETCH  CR01  INTO
  :I1-商品コード
 ,:I1-商品名
 ,:I1-単価
 ,:I1-数量
END- EXEC
```

7.9.32　@@lang 文

形　式

@@lang { COBOL | C | " 言語種別 " }
[FOR_REPOSITORY = " リポジトリの言語区分 "]
[EXTENSION = " 拡張子 "]
[MODIFY_CONNECT = " 連結文字 "]
[MODIFY_ORDER = { DESCEND | ASCEND }]
[UOC_BEGIN = " ユーザ追加処理の先頭に挿入する文字列 "]
[UOC_END = " ユーザ追加処理の末尾に挿入する文字列 "];

注
{　} は項目のどれか一つを記述することを表します。
[　] は省略できることを表します。

機　能

言語種別と言語種別に関連する生成規則を指定します。

規　則

- @@lang 文中では可変記号は使用できない。
- テンプレート概要定義部の直後に記述しなければならない。

- テンプレートに 1 回だけ指定できる。
- 部品のテンプレートには指定できない。
- @@lang 文がテンプレート概要定義部の直後に指定されないでインタフェース定義部の記述が始まっている場合，COBOL を仮定する。
- 言語種別は，COBOL，C 以外の任意の生成言語種別を指定する。
- リポジトリの言語区分は，@@getdata 関数，@@modify 関数，@@rule 文での SEWB+/REPOSITORY アクセス時の言語区分を指定する。
- FOR_REPOSITORY 句を省略した場合，言語種別から仮定する。

言語種別	FOR_REPOSITORY の言語区分	
	指定あり	指定なし
COBOL	指定値でアクセス	COBOL，または OOCOBOL でアクセス
C	指定値でアクセス	C，または C++ でアクセス
" 言語種別 "	指定値でアクセス	REPOSITORY アクセスできない
@@lang 文なし（COBOL）	該当しない	COBOL，または OOCOBOL でアクセス

- 拡張子は，生成ソース出力時の拡張子を "." 付きで指定する。
- EXTENSION 句を省略した場合，言語種別，および環境設定に沿った拡張子で出力する。

言語種別	EXTENSION の拡張子	
	指定あり	指定なし
COBOL	指定値で出力	ソース正書法に従う（.CBL，または .CBF で出力）
C	指定値で出力	.C で出力
" 言語種別 "	指定値で出力	拡張子なしで出力

- 連結文字は，@@modify 関数，およびルールスクリプト中の @MODIFY キーワードを生成するときの，修飾情報（データ項目名）の連結文字を指定する。ルールスクリプト中のキーワードについてはマニュアル「SEWB+/REPOSITORY 辞書設計ガイド」を参照してください。
- MODIFY_CONNECT 句を省略した場合は言語種別から連結文字を仮定する。

言語種別	MODIFY_CONNECT の連結文字	
	指定あり	指定なし
COBOL	指定値で連結	「OF」で連結
C	指定値で連結	「.」で連結
" 言語種別 "	指定値で連結	連結文字を仮定しない

- MODIFY_ORDER 句は，@@modify 関数，およびルールスクリプト中の @MODIFY キーワードを生成するときの，修飾情報（データ項目名）の修飾順序を指定する。ルールスクリプト中のキーワードについてはマニュアル「SEWB+/REPOSITORY 辞書設計ガイド」を参照してください。
- MODIFY_ORDER 句を省略した場合は言語種別から修飾順序を仮定する。

言語種別	MODIFY_ORDER の指定	
	指定あり	指定なし
COBOL	指定値の順序	ASCEND（下位から上位）を仮定
C	指定値の順序	DESCEND（上位から下位）を仮定
" 言語種別 "	指定値の順序	DESCEND（上位から下位）を仮定

- UOC_BEGIN には，ユーザ追加処理の先頭に挿入する文字列を指定する。
- UOC_END には，ユーザ追加処理の末尾に挿入する文字列を指定する。
- ユーザ追加処理の前後に挿入する文字列には，ユーザ追加処理のブロック名が展開する可変記号「@uocname」を指定できる。
  可変記号「@uocname」はユーザ追加処理中，つまり UOC_BEGIN と UOC_END に指定する文字列の中でだけ有効な可変記号です。
- 言語種別が COBOL で，かつ環境設定の COBOL ソース正書法が固定形式またはホスト向け固定形式の場合，UOC_BEGIN と UOC_END で指定した文字列の先頭に 6 文字の空白が追加される。
- UOC リバース機能を使う場合は，UOC_BEGIN と UOC_END に次の形式の文字列を指定する必要がある。

```
UOC_BEGIN="UOCの開始を意味する文字列[@可変記号][任意の文字列]"
UOC_END   ="UOCの終了を意味する文字列[@可変記号][任意の文字列]"
```

なお，次のように指定すると，UOC コメントを UOC 単位でユニークにできる。この指定を推奨する。

```
UOC_BEGIN="UOCの開始を意味する文字列 @uocname [任意の文字列]"
UOC_END   ="UOCの終了を意味する文字列 @uocname [任意の文字列]"
```

- @uoc 文に SUPPRESS が指定してある場合は，UOC_BEGIN と UOC_END で指定した文字列は生成されない。したがって，UOC リバース機能を使う場合，@uoc 文に SUPPRESS を指定してはならない。

使用例

テンプレート1

```
#NAME = 'ファイル作成・更新'
#OWNER = 'HITACHI
#OUTLINE = 'マスタファイル更新プログラム'
#
   @@lang  COBOL;
           :
   @@lang  C;
           :
   @@lang  COBOL;
           :
```

テンプレート概要定義部

2回目以降は
エラーとなる。

テンプレート2

```
#  NAME = 'ファイル作成・更新'
#
#
@@*
@@*
   @@interface
     :
   @@set @A = 1;
   @@lang = C;
```

インタフェース定義部の記述
が始まっているので@@lang
COBOLが仮定される。

COBOLが仮定されているので
エラーとなる。

テンプレート３

```
#  NAME = 'ファイル作成・更新'
#
#
@@*
@@*
  @@lang  COBOL
   FOR_REPOSITORY = "COBOL又はOOCOBOL"
   EXTENSION = ".CBL"
     MODIFY_CONNECT = " OF "
   MODIFY_ORDER = ASCEND
     UOC_BEGIN = "*uocname *"
     UOC_END  = "*uocname *";
```

言語種別
リポジトリの言語区分
拡張子
連結文字
修飾順序

ユーザ追加処理の前後
にコメント行の挿入を
指定。

7.9.33　@@leadbyte 関数

形　式

@@leadbyte（可変記号，位置）

機　能

可変記号に設定されている文字列について，位置で指定されたコードが２バイトコードのリードバイト（先頭の１バイト）かどうかを判定するときに使用します。

規　則

- リードバイトの場合は真を返す。リードバイト以外の場合は偽を返す。
- 可変記号の内容が配列のときは，添字を指定する。

- 判定するコードの先頭からのバイト位置を位置に指定する。
- 可変記号に数値を指定している場合は，その数値のけた数を返す。

使用例

可変記号に設定された長い文字列を，80 バイト改行して生成します。80 バイト目がリードバイトの場合は，79 バイトで区切ります。

```
@@set @string = "長い文字列...";                    ··· @string:分割対象文字列
@@set @len = @@lengthb(@string);                    ··· @len: 分割対象文字列のバイト数
@@set @cnt = 0;                                     ··· @cnt:分割処理済みバイト数
@@while (@cnt < @len)
  @@if (@@leadbyte(@string,@cnt + 80))              ···80バイト目がリードバイトの場合
    @@set @gen = @@strb(@string,@cnt + 1,79);       ···79バイト切り出す
    @@set @cnt = @cnt + 79;                         ··· 分割処理済みバイト数に79加算
  @@else
    @@set @gen = @@strb(@string,@cnt + 1,80);       ···80バイト切り出す
    @@set @cnt = @cnt + 80;                         ··· 分割処理済みバイト数に80加算
  @@end;
@gen                                                ··· 切り出した文字列を展開
@@end;
```

7.9.34 @@length 関数

形 式

@@length（可変記号）

機 能

可変記号に文字列または数値が設定されている場合，設定されている文字列の長さまたはけた数を返す数値関数として使用します。この関数は，文字列の長さを文字単位で返します。

規 則

- 可変記号の内容が配列のときは，添字を指定する。
- 可変記号に文字列を設定している場合は，その文字列長（文字数）を返す。
- 可変記号に数値を指定している場合は，その数値のけた数を返す。

使用例

```
@@set @A = "ABC";
@@set @B = 1234 ;
@@set @C = 01234 ;
@@set @D = "いずみ中央";
@@set @A_LEN = @@length(@A);     @A_LENには，「3」が設
                                 定される。
@@set @B_LEN = @@length(@B);     @B_LENには，「4」が設
                                 定される。
@@set @C_LEN = @@length(@C);     @C_LENには，「5」が設
                                 定される。
                                 けた数なので「01234」
                                 の5けたとなる。
@@set @D_LEN = @@length(@D);     @D_LENには，「5」が設
```

```
                                        定される。
                                        2バイトコードで5文字
                                        のため5けたとなる。
```

7.9.35 @@lengthb 関数

形 式

@@lengthb（可変記号）

機 能

可変記号に文字列または数値が設定されている場合，設定されている文字列の長さまたはけた数を返す数値関数として使用します。この関数は，文字列の長さをバイト単位で返します。

規 則

- 可変記号の内容が配列のときは，添字を指定する。
- 可変記号に文字列を指定している場合は，その文字列長（バイト数）を返す。
- 可変記号に数値を指定している場合は，その数値のけた数を返す。

使用例

```
@@set @A = "ABC";
@@set @B = 1234 ;
@@set @C = 01234 ;
@@set @D = "いずみ中央";
@@set @A_LEN = @@lengthb(@A);     @A_LENには，「3」が設
                                  定される。
@@set @B_LEN = @@lengthb(@B);     @B_LENには，「4」が設
                                  定される。
@@set @C_LEN = @@lengthb(@C);     @C_LENには，「5」が設
                                  定される。
                                  けた数なので「01234」
                                  の5けたとなる。
@@set @D_LEN = @@lengthb(@D);     @D_LENには，「10」が設
                                  定される。
                                  2バイトコードで5文字
                                  のため10バイトとなる。
```

7.9.36 @@merge 文

形 式

@@merge　可変記号；

機 能

@@merge 文は，@@put 文に定義されている文字列を生成ソースに展開するときに使用します。また，テンプレートで抽出した業務ルールに書かれている，@@section 文に定義されている文字列を生成ソースに展開させるのにも使用します。

@@put 文に関しては「7.9.42　@@put 文」を参照してください。

@@section 文に関しては，マニュアル「SEWB+/REPOSITORY　辞書設計ガイド」を参照してください。

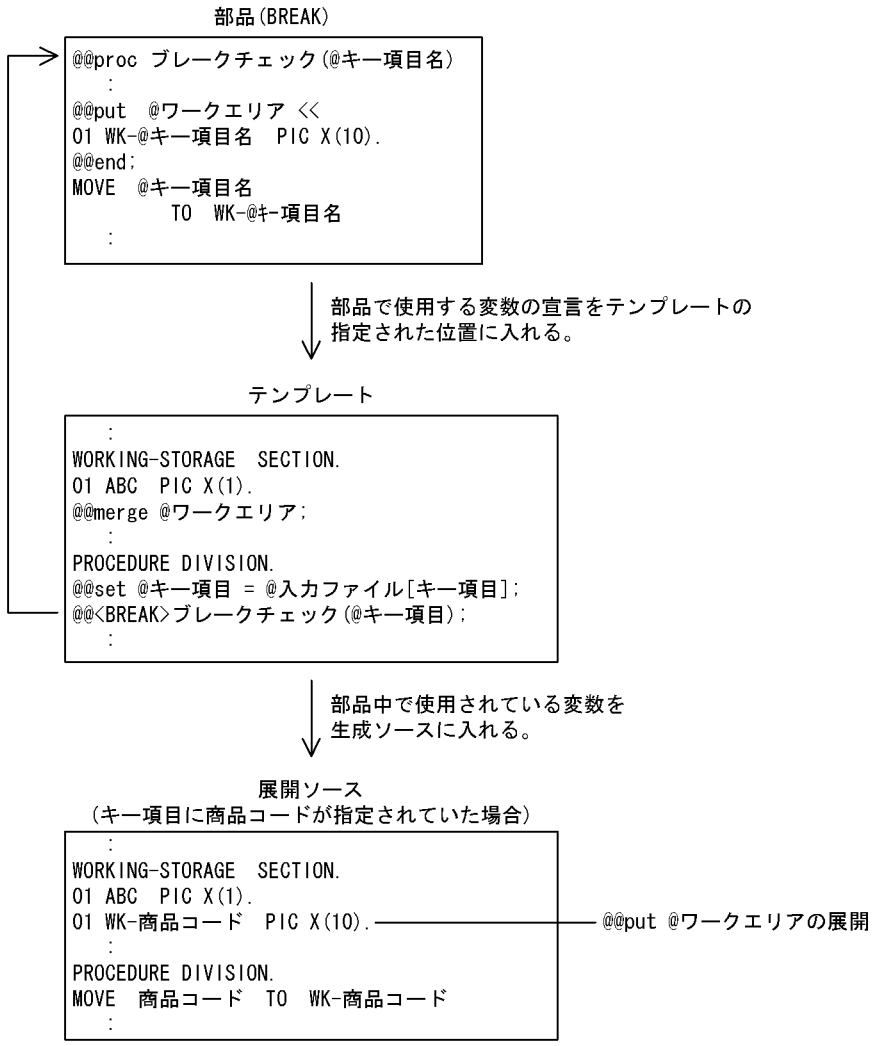
規　則

- @@merge 文の可変記号には，@@put 文に書かれた可変記号，または @@section 文に書かれたコード名の先頭に「@」を付けたものを指定する。
- @@merge 文に書かれた可変記号は，グローバル宣言されたものとして処理される。
- @@put 文で可変記号に値が設定されていない場合，何も展開されない。
- @@section 文に書かれたコード名と，@@merge 文の可変記号から「@」を削除した部分が異なる場合，何も展開されない。
- @@section 文に「MAIN」が設定されている場合，@@section 文に定義されている文字列は，@@rule 文の位置に展開される。
- 可変記号には配列が指定されている可変記号は指定できない。

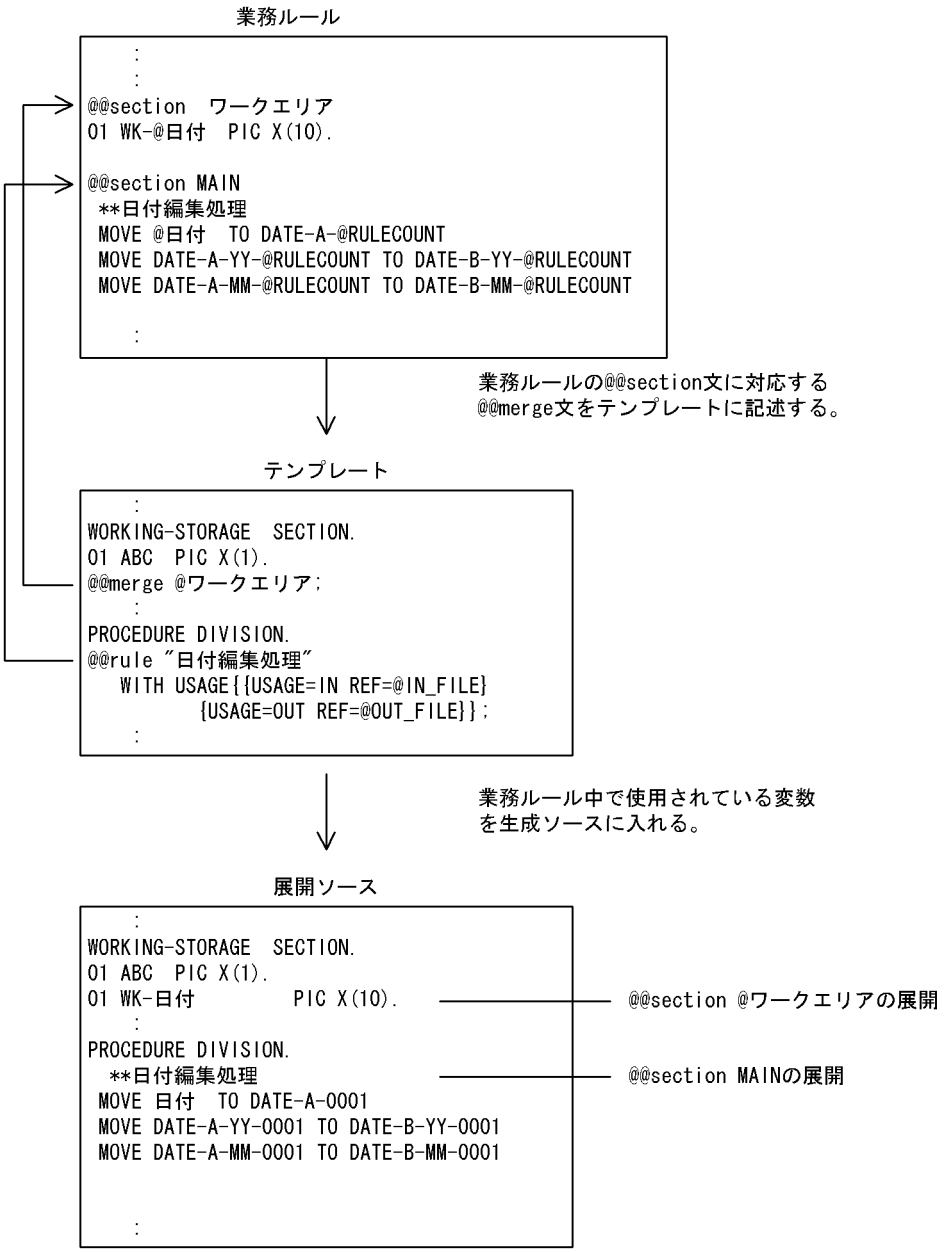
使用例

部品中の @@put 文に書かれた可変記号をテンプレートで呼び出し，@@put 文に定義されている文字列を生成ソースに展開させます。
また，業務ルール中の @@section 文に指定されたコード名をテンプレートで呼び出し，@@section 文に定義されている文字列を生成ソースに展開させます。プログラムのコード上，連続していない個所に，業務ルールを分割して展開させるときに使用します。
なお，@@put 文を使用しても同様に，業務ルールを分割して展開できます。

@@put 文と @@merge 文の使用



@@section 文と @@merge 文の使用



注意事項

@@merge 文に書かれた可変記号と同じものが，@@put 文にも @@section 文にもある場合※，@@put 文と @@section 文がそれぞれ展開されます。このように指定内容が重複していると，不要な文字列が展開され，プログラムの処理の内容を変えてしまうことがあるので注意してください。特に，@@section 文を呼び出すためのコー

ド名は，データ項目辞書作成者とテンプレート作成者の間で統一してください。

注※

@@section 文の場合は，@@merge 文に書かれた可変記号の先頭から「@」を削除した部分と，@@section 文のコード名が同一ならば，これらは一致しているとみなされます。

7.9.37 @@modify 関数

形 式

@@modify（可変記号 [，[接頭語][，[接尾語][，[最上位項目名]]]]）

注

［］は省略できることを示します。

機 能

@@interface 文で指定されたデータ項目の修飾（全パスの項目名）付き項目名を取得する場合に使用します。

規 則

- @@set 文の右辺に指定する。
- 可変記号は @@interface 文で ATTR=ITEM を指定した可変記号を指定する。
- @@interface 文で指定されたデータ項目またはその上位の項目に繰り返しがある場合はエラーになる。
- 接頭語，接尾語，および最上位項目名はそれぞれ省略できるが，コンマだけの指定はできない。指定した場合はエラーになる。
- 接頭語は，修飾付きデータ項目名の先頭に付ける文字列を文字定数で指定する。
- 接尾語は，修飾付きデータ項目名の末尾に付ける文字列を文字定数で指定する。
- 最上位項目名は，修飾名付きデータ項目名のうち最も上位となるデータ項目名を別の名称に置き換える場合，その名称を文字定数で指定する。
- 言語種別が COBOL 以外の場合，可変記号にレコード定義のデータ項目を指定できない。
- 取得する修飾情報（項目名）の順序は @@lang 文の MODIFY_ORDER 句の指定※に従う。
- 取得する修飾情報（項目名）の言語区分は，@@lang 文の FOR_REPOSITORY 句の指定に従う。指定された言語区分用のデータ項目名が定義されていない場合には，言語区分に依存しない共通のデータ項目名となる。指定された言語区分がリポジトリに登録されているものと一致しない場合は，生成時にエラーになる。
- 修飾情報の連結文字は，@@lang 文の MODIFY_CONNECT 句の指定に従う。言語種別が C，COBOL 以外の場合には，修飾情報（データ項目名）がそのまま連結される。

注※

MODIFY_ORDER 句が DESCEND なら，上位レベル項目から下位レベル項目

の順序で修飾されます。MODIFY_ORDER 句が ASCEND なら，下位レベル項目から上位レベル項目の順序で修飾されます。

使用例

- **データ構造**

```
1 XXX              1 YYY
2  UPYEAR          2  DATE
2  UPDATE          3      YEAR
3      UPMONT      3      MONT
3      UPDAY       3      DAY
```

- **テンプレート**（COBOL）

```
@@lang COBOL                                  ┐
  FOR_REPOSITORY                              │
    = "COBOL 又は OOCOBOL"                    ├ 1.
  MODIFY_ORDER = ASCEND                       │
  MODIFY_CONNECT = " OF ";                    ┘
@@interface @IN = {                           ┐
  ATTR=FILE,                                  │
  IO=IN,                                      ├ 2.
  IN_KEY={ATTR=ITEM,REF=@IN},                 │
  REC_IN={ATTR=RECORD_NAME}};                 ┘
@@interface @OUT = {                          ┐
  ATTR=FILE,                                  │
  IO=OUT,                                     ├ 3.
  OUT_KEY={ATTR=ITEM,REF=@OUT},               │
  REC_OUT={ATTR=RECORD_NAME}};                ┘
  :
@@expand @IN[REC_IN]        4.
  PREFIX="IN-";
@@expand @OUT[REC_OUT]      5.
  PREFIX="OUT-";
  :
@@set @IN_ITEM =                              ┐
  @@modify(@IN[IN_KEY],"IN-");                ├ 6.
@@set @OUT_ITEM =                             │
  @@modify(@OUT[OUT_KEY],"OUT-");             ┘
MOVE @IN_ITEM TO @OUT_ITEM.       7.
```

1. 項目修飾は，下位項目から上位項目の順に，OFを使用し修飾する。
2. プログラム定義時に，@INに最上位項目YYYを選択したデータ定義，@IN[IN_KEY]にデータ項目YEARを指定する。
3. プログラム定義時に，@OUTに最上位項目XXXを選択したデータ定義，@OUT[OUT_KEY]にデータ項目UPYEARを指定する。
4. 5.レコード展開。
6. 修飾した項目名を可変記号に設定する。
7. 代入文を生成する。

- **生成ソース**（COBOL）

```
  :
01  IN-YYY.                                  ┐
  02  IN-DATE.                               │
    03  IN-YEAR  PIC  X(4).                  ├ 4.
    03  IN-MONT  PIC  X(2).                  │
    03  IN-DAY   PIC  X(2).                  ┘
01  OUT-XXX.                                 ┐
  02  OUT-UPYEAR  PIC  X(4).                 │
  02  OUT-UPDATE.                            ├ 5.
    03  OUT-UPMONT  PIC  X(2).               │
    03  OUT-UPDAY   PIC  X(2).               ┘
  :
MOVE                                        ┐
  IN-YEAR OF IN-DATE OF IN-YYY              ├ 7.
    TO                                      │
  OUT-UPYEAR OF OUT-XXX.                    ┘
```

- **テンプレート**（C）

```
@@lang C                                    ┐
  FOR_REPOSITORY  =  "C  又は  C++"         │
  MODIFY_ORDER  =  DESCEND                  ├ 1.
  MODIFY_CONNECT  =  ".";                   ┘
@@interface  @IN = {                        ┐
  ATTR=FILE,                                │
  IO=IN,                                    ├ 2.
  IN_KEY={ATTR=ITEM,REF=@IN},               │
  REC_IN={ATTR=RECORD_NAME}};               ┘
@@interface  @OUT = {                       ┐
  ATTR=FILE,                                │
  IO=OUT,                                   ├ 3.
  OUT_KEY={ATTR=ITEM,REF=@OUT},             │
  REC_OUT={ATTR=RECORD_NAME}};              ┘
  :
@@expand  @IN[REC_IN];       4.
@@expand  @OUT[REC_OUT];     5.
  :
@IN[REC_IN]     wk_in;       6.
@OUT[REC_OUT]   wk_out;      7.
  :
@@set @OUT_ITEM =                           ┐
  @@modify(@OUT[OUT_KEY],,,"wk_out");       ├ 8.
@@set @IN_ITEM =                            │
  @@modify(@IN[IN_KEY],,,"wk_in");          ┘
@OUT_ITEM = @IN_ITEM;        9.
```

1. 項目修飾は，上位項目から下位項目の順に，ピリオドを使用し修飾する。
2. プログラム定義時に，@INに最上位項目YYYを選択したデータ定義，@IN[IN_KEY]にデータ項目YEARを指定する。
3. プログラム定義時に，@OUTに最上位項目XXXを選択したデータ定義，@OUT[OUT_KEY]にデータ項目UPYEARを指定する。
4. 5.レコード展開。
6. 7.構造体識別子の宣言。
8. 修飾した項目名を可変記号に設定する。
9. 代入文を生成する。

- **生成ソース（C）**

```
  :
typedef  struct  YYY{                    ┐
  struct  DATE{                          │
    int  YEAR;                           │
    int  MONT;                           ├ 4.
    int  DAY;                            │
  }DATE;                                 │
}YYY;                                    ┘
typedef  struct  XXX{                    ┐
  int  UPYEAR;                           │
  struct  UPDATE{                        │
    int  UPMONT;                         ├ 5.
    int  UPDAY;                          │
  }UPDATE;                               │
}XXX;                                    ┘
  :
YYY wk_in;   6.
XXX wk_out;  7.
  :
wk_out.UPYEAR                              9.
    = wk_in.DATE.YEAR;
```

7.9.38 @@msg 文

形　式

@@msg　文字定数；

機　能

ソースの展開中に，指定された文字列をメッセージファイル（メッセージウィンドウ）に出力するときに使用します。

展開ログ情報，展開部品の使用上の注意事項などを出力します。

規　則

- 文字定数は継続して指定できる。
- メッセージを出力しても生成の終了コードは変化しない。

使用例

```
@@if (@入力ファイル[アクセス方法]ne "S" )
    @@msg   "アクセス方法の指定に誤りがあります。";
@@end;
```

↓ 生成結果

メッセージウィンドウ

```
アクセス方法の指定に誤りがあります。
```

7.9.39 @@parts 文

形 式

@@parts " 部品ブロック名 "
　　REF= 可変記号名
　　[PARENT=" 親ブロック名 "]
　　[COMMENT=" 説明文字列 "];

注
　　[ ] は省略できることを表します。

機 能

プログラム定義で選んだ部品を展開するときに使用します。

規 則

- 可変記号名には，@@interface 文で ATTR=PARTS を指定した可変記号を指定する。
- 部品ブロック名には，部品展開位置の名前を，可変記号または文字定数で指定する。
- @@parts 文の位置には，プログラム定義の [部品] タブで選んだ部品が展開される。
- 親ブロック名には，ユーザ追加処理の一覧をツリー形式で表示させるときに，指定した部品ブロック名の親になるブロック名を指定する。
- 部品ブロック名，親ブロック名には可変記号を指定できる。
- 親ブロック名を指定すると，@@parts 文の直後に書いた部品ブロック名は，指定した親ブロックの下の該当する部品ブロック名に位置づけられる。
- 親ブロック名を省略すると，ツリーの最上位の階層に位置づけられる。
- 親ブロックが存在しなかった場合，ツリーの最上位の階層に位置づけられる。
- 説明文字列は，プログラム定義の [追加処理] タブの説明欄に表示される。

使用例

```
@@interface @展開部品 = {
      ATTR = PARTS,
      COMMENT = 'チェック部品を選択します。'};
            :
@@parts "チェック部品展開"
        REF=@展開部品
        PARENT="MAIN"
        COMMNET="展開部品に指定した部品が"
                "展開されます。"
```

7.9.40 @@pic 関数

形 式

@@pic（可変記号）

機　能

@@interface 文で指定されたデータ定義のデータ項目と同じタイプと長さを持つデータ項目を定義する場合に使用します。データ定義から参照しているレコード定義ファイルのデータ項目にも使用できます。

規　則

- 言語種別が COBOL の場合に指定できる。
- @@set 文の右辺に指定する。
- 可変記号は @@interface 文で ATTR=ITEM を指定した可変記号を指定する。
- 属性と長さの生成規則は「7.9.9 @@expand 文」を参照のこと。
- データ項目のタイプが集団項目または @ コピー文の場合，0 文字の文字列が展開される。
- 定義できるデータ項目のタイプは，SEWB+/REPOSITORY が標準で提供している COBOL 用のタイプだけ使用できる。

使用例

テンプレート

```
@@interface
 @参照ファイル =
 {ATTR = file ,
  キー項目 = {ATTR = ITEM, REF = @参照ファイル}
 };
   :
WORKING-STORAGE  SECTION.
@@set  @編集文字列 = @@pic (@参照ファイル[キー項目] );
 01 WK-@参照ファイル[キー項目]  @編集文字列  .
   :
PROCEDURE DIVISION.
 MOVE @参照ファイル[キー項目] TO WK-@参照ファイル[キー項目]
   :
```

↓ @参照ファイル[キー項目]の名称が商品コードで
タイプが「X」で長さが10バイトの場合

生成結果

```
   :
WORKING-STORAGE  SECTION.
 01  WK-商品コード  PIC  X(10).
   :
PROCEDURE DIVISION.
 MOVE  商品コード  TO  WK-商品コード
   :
```

7.9.41　@@proc 文

形　式

@@proc プロシジャ名

([引数 [COMMENT=" 引数の説明 "]]…)
　文

注
　[　] は省略できることを表します。

機　能

部品内のプロシジャを定義するときに使用します。

規　則

- @@proc 文から次の @@proc 文またはファイルの終了までの間に部品の処理を記述する。
- プロシジャ名は次の規則に従って記述する。
  名称の長さが 30 文字以内の文字列。ただし，数字だけで記述することはできない。
  英字の大文字と小文字は区別される。
  引数には可変記号だけが指定できる。
- COMMENT に指定した引数の説明は，プログラム定義の [部品] タブから部品のパラメタを指定するとき，コメントとして表示される。

使用例

一つの RDBPARTS という部品ファイルに，DECLARE と CONNECT のプロシジャを持っている例です。

部品RDBPARTS

```
@@* DBのDECLARE定義生成
@@proc  DECLARE(@DB COMMENT=必須:データ定義(RDB)の可変機能を指定する。)
EXEC SQL BEGIN DECLARE SECTION END-EXEC.
@@expand  @DB;
EXEC SQL END DECLARE SECTION END-EXEC.

@@* DBのCONNECT生成
@@proc  CONNECT (@ユーザID COMMENT ="任意:実行時のユーザIDを使用するときは¥n"
                                    "指定しない。",
                 @パスワードCOMMNET="ユーザIDを指定した場合，必ず指定する。 ")
EXEC SQL
  @@if (@ユーザIDeq "")
  CONNECT
  @@else
  CONNECT :@ユーザID IDENTIFIED BY :@パスワード
  @@end;
END-EXEC
```

7.9.42　@@put 文

形　式

@@put 可変記号 <<
文

@@end;

機　能

ソースの生成時に，@@merge 文で記述されている同じ可変記号の個所に，@@put 文で代入された文の内容が展開されます。

規　則

- 文には，@@expand 文以外の制御文は記述できない。
- 可変記号にすでにテキストが設定されているとき，文は設定済みの文の後ろに追加設定される。
- @@put 文に指定した可変記号は，グローバル宣言されたものとして処理される。
- 埋め込みテキスト（＜＜文）中の可変記号は，@@put 文が実行された時の値で展開される。
- @@merge 文と対で使用される。

使用例

「7.9.36　@@merge 文」の使用例を参照してください。

7.9.43　@@reclen 関数

形　式

@@reclen（可変記号［修飾名］）

注

［　］は省略できることを表します。

機　能

レコード長を取得する数値関数です。

規　則

- 言語種別が COBOL の場合に指定できる。
- 可変記号名は @@interface 文の ATTR でデータ定義種別を指定した可変記号を指定する。
- 修飾名は，データ定義識別子が RECORD_NAME でなければならない。
- レコード中の項目の属性（タイプ）が Z（数字編集項目），E（外部浮動小数点項目），D（内部浮動小数点項目），またはフリー定義がある場合は指定できない。
- @@set 文の右辺に指定する。
- データ項目のタイプには，SEWB+/REPOSITORY に最初から登録されている COBOL 用のタイプだけ使用できる。
- レコード定義ファイルで定義されたレコードは指定できない。

使用例

@@set @ レコード長 = @@reclen（@ 入力ファイル［レコード名］）；
MOVE @ レコード長 TO メッセージ長 .

@入力ファイル：JUCHUUのレコード定義

項目名	タイプ	長さ
コード	X	8
名前	X	20
備考	X	32

生成結果

@ 入力ファイルに指定したレコード長が 60 なので，生成結果は以下のようになる。

```
MOVE  60  TO  メッセージ長
```

7.9.44 @@rule 文

「7.7 業務ルールの利用」を参照してください。

7.9.45 @@set 文

形 式

@@set 可変記号 = 式；

機 能

可変記号に値を設定するときに使用します。

規 則

- 式には，算術式，文字定数，連結式が指定できる。
- 右辺をコンマ（,）で区切って{}内に並べると，配列のすべての要素に値が設定できる。
- 配列の要素は{}内の順番で設定される。添字は「1」から始まる。また，値を「キーワード＝値」で指定したときは，キーワードが添字になる。
- 値の指定で，位置指定とキーワード指定は混在できない。
- 一度，位置指定で値を設定した配列は，キーワード指定で再度値の設定はできない。
- 一度，キーワード指定で値を設定した配列は，位置指定で再度値の設定はできない。

使用例

```
（例 1） 位置指定
@@set @Ar1= {001,002,003} ;
  このとき
@Ar1[1]      … 001
@Ar1[2]      … 002
@Ar1[3]      … 003

（例 2） キーワード指定
@@set @Ar1 = {time=10,date=0105,year='H08'};
  このとき
@Ar1[time]     …10
@Ar1[date]     …0105
@Ar1[year]     …H08


(例 3)
@@set @Ar1={'a01','a02','a03'};
@@set @Ar2={lvl=1, name="ITM001", pic="9(4) BINARY"};
 このとき
@Ar1[1]          … a01
@Ar1[2]          … a02
@Ar1[3]          … a03
@Ar2[lvl]        … 1
@Ar2[name]       … ITM001
@Ar2[pic]        … 9(4) BINARY

(例 4)  エラーケース
@@set @A = {1,name='ITEM001'};<== 位置指定とキーワード指定が混在している。
@@set @B[1] = 10 ;              ┐ 一度位置指定で設定されているものに
@@set @B[name] = 'waki' ;       <┘ 対してキーワード指定で再設定しよう
@@set @C = 'ABC' ;                 としている。
@@set @C = @C + 1               <== 式の値が文字なのでエラーになる。
```

7.9.46 @@str 関数

形 式

@@str（可変記号，先頭位置［，文字列長］）

注

［ ］は省略できることを表します。

機 能

可変記号の一部を参照したいときに使用します。この関数は，文字列を文字単位で操作します。

規 則

- 先頭位置，および文字列長には，数値定数，算術式，または可変記号を指定する。
- 文字列長（文字数）の指定は省略できる。その場合，先頭位置から可変記号の右端までを文字列長とする。

- データの文字列が2バイトコードの場合，データは1文字ずつ設定される。
- 指定された先頭位置にデータがない場合はエラーになる。
- 指定された文字列長よりもデータが少ない場合，データが入っているところまで設定される。

使用例

```
@@set @A001 = 'ABCDE';
@@set @B001 = @@str(@A001,2,3);   <――@B001 の内容は'BCD'が設定される。
@@set @C001 = 12345;
@@set @D001 = @@str(@C001,3,3) ;  <――@D001 の内容は'345'が設定される。
@@set @E001 = @@str(@C001,6,1) ;  <――@C001 は 5 文字しかないのでエラーに
                                      なる。
@@set @F001 = @@str(@C001,3,10) ; <――@001 の 3 番目からは 3 文字しかないの
                                      で@F001 は'345'が設定される。
```

7.9.47 @@strb 関数

形 式

@@strb（可変記号，先頭位置［，文字列長］）

注

［ ］は省略できることを表します。

機 能

可変記号の一部を参照したいときに使用します。この関数は，文字列をバイト単位で操作します。

規 則

- 先頭位置，および文字列長には，数値定数，算術式，または可変記号を指定する。
- 文字列長（バイト数）の指定は省略できる。その場合，先頭位置から可変記号の右端までを文字列長とする。
- 2バイトコードの途中で切る場合はエラーとなる。2バイトコードの判定については「7.9.33 @@leadbyte 関数」を参照のこと。
- 指定された先頭位置にデータがない場合はエラーになる。
- 指定された文字列長よりもデータが少ない場合，データが入っているところまで設定される。

使用例

```
@@set @A001 = 'AB漢字CD';
@@set @B001 = @@str(@A001,2,3); <―――― @B001の内容は'B漢'が設定される。
@@set @C001 = '漢字データ';
@@set @D001 = @@str(@C001,2,4) ;<―――― 2バイトコードの途中から始まるのでエラー
@@set @E001 = @@str(@C001,1,5) ;<―――― 2バイトコードの途中で終わるのでエラー
@@set @F001 = @@str(@C001,5,10) ; <―――― @C001の5バイト目からは6バイト(3文字)しかないので
                                      @F001は'データ'が設定される。
```

7.9.48 @@switch 文

形 式

@@switch (可変記号)
@@case 値 1:
　文 1
[[@@case 値 2:
　[文 2]]…]
[@@default:
　文 3]
@@end;

注
　［ ］は省略できることを表します。

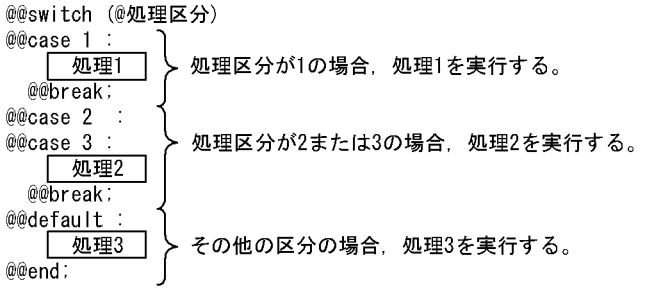
機 能

変数の値によって複数の異なる処理の中から該当する処理を選ぶときに使用します。

規 則

- @@case を少なくとも一つは記述しなければならない。
- 値式に指定できるのは，可変記号，文字定数または数値定数である。
- @@case の値は上から順に評価され，可変記号と一致するときに @@case の次の文が処理される。
- @@switch 文を抜けるには，@@break 文を使用する。@@break 文がないときは，次の @@case に制御が移る。
- 変数に一致する @@case の値がないときは，@@default の文 3 が実行される。予測できない値が可変記号に代入されたときに備えて，@@default は記述しておいた方がよい。

使用例



7.9.49 @@uoc 文

形 式

@@uoc "UOC 名 "
[PARENT=" 親ブロック名 "]
[COMMENT=" 説明文字列 "]
[SUPPRESS];

注
［ ］は省略できることを表します。

機 能

ソース生成中に展開させるユーザ追加処理の位置を指定するときに使用します。

規 則

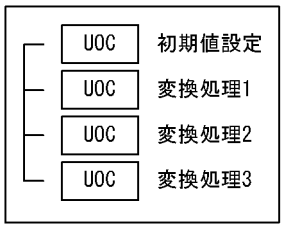
- UOC 名は，ユーザ追加処理の名前を文字定数で指定する。
- @@uoc 文の位置には，プログラム定義の [ユーザ処理] タブで選んだユーザ追加処理の内容が取り込まれる。
- 親ブロック名は，プログラム定義の [ユーザ処理] タブでユーザ追加処理の一覧をツリー形式で表示させるときに，指定した UOC 名の親になるブロック名を指定する。
- UOC 名，親ブロック名の文字列には可変記号を指定することができる。ループの中で @@uoc 文を繰り返し実行する場合には，UOC 名の値を変化させて，それぞれの UOC 名を区別する。
- 親ブロック名を指定すると，ここで指定した UOC 名は，この親ブロックの下に位置づけられる。
- 親ブロック名を省略すると，ツリーの最上位の階層に位置づけられる。ただし，@@uoc 文が @@parts 文から展開されている場合，@@parts 文のブロックの下に位置づけられる。
- 親ブロック名が存在しなかった場合，最上位の階層に位置づけられる。
- 説明文字列は [ユーザ処理] タブの説明欄に表示される。
- SUPPRESS を指定すると，@@lang 文で指定したユーザ追加処理の前後に挿入する文字列は生成されない。したがって，UOC リバース機能を使用する場合は SUPPRESS を指定してはならない。
- プログラム定義でユーザ追加処理の内容が指定されていない場合，ソースへは生成されない。ただし，@@lang 文でユーザ追加処理の前後に文字列を挿入する指定がある場合，挿入する文字列は生成される。

使用例

```
@@uoc "初期値設定"  COMMENT="出力レコードのデフォルト値を設ける場合に記述してください。";
@@set  @i = 1;
@@while  (@i <= 3)
@@uoc "変換処理@i"  COMMENT="レコードの内容を出力するときの変換処理を記述してください。";
```

```
@@set  @i = @i + 1;
@@end;
```

このとき，[ユーザ処理] タブには，次のようにユーザ追加処理の一覧が表示されます。



ツリー形式で表示するときの指定は「7.9.7　@@diagram 文」を参照してください。

7.9.50　@@uocdefined 関数

形　式

@@uocdefined（"UOC 名 "）

機　能

ユーザ追加処理がプログラム定義で作成されているか判定するときに使用します。

規　則

- UOC 名は，@@uoc 文で指定したユーザ追加処理の名前を可変記号または文字定数で指定する。
- 指定された UOC 名を持つユーザ追加処理が，プログラム定義で作成されている場合は，真を返す。それ以外の場合は偽を返す。

使用例

```
@@uoc"初期値設定"  COMMENT="必ず指定します。";
@@if (!@uocdefined("初期値設定"))
  @@msg "ユーザ追加処¥'初期値設定¥'は必須です。"
@@end;
```

7.9.51　@@while 文

形　式

@@while（条件式）

文

@@end;

機　能

指定する条件式が成立する間，文を繰り返して展開するときに使用します。

規　則

条件式が成立するとき，文が処理される。条件式の比較は文の処理前に行われる。

使用例

```
(例1)
  @@*  ファイルMAXの数だけ処理1を実行する。
@@set @ファイルMAX = 256;
@@set @i = 1;
@@while (@i <= @ファイルMAX)
    処理1
  @@set @i = @i + 1;
@@end;

(例2)
  @Names の値が {KNO, KNAME, HYEAR, SAL} のとき,
@@* Namesの要素をH_Namesにすべて設定する
@@set @n = @@count(@Names);
@@set @H_Names = ":@Names[1]";
@@if (n >= 2)
  @@set @i = 2;
  @@while (@i <= @n)
    @@set @H_Names = "@H_Names, :@Names[@i]"
    @@set @i = @i + 1;
  @@end;
@@end;
  このとき, @H_Namesの値は ":KNO, :KNAME, :HYEAR, :SAL"となります。
```

7.9.52 @@xmap3common 関数

形 式

@@xmap3common（可変記号）

機 能

@@interface 文で XMAP3 を指定した可変記号からマップ定義の共通情報を取得するときに使用します。なお，XMAP3 の詳細はマニュアル「画面・帳票サポートシステム XMAP3 プログラミングガイド 画面編」，および「画面・帳票サポートシステム XMAP3 プログラミングガイド 帳票編」を参照してください。

規 則

- @@set 文の右辺に指定する。
- 可変記号には，@@interface 文で ATTR=XMAP3 を指定した可変記号を指定する。
- マップ定義の共通情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列は，情報の種類ごとに確保される。配列の添字には，キーワードを指定する。取得できるマップ定義の共通情報を次に示します。

マップ定義の情報	添字に指定するキーワード
マップ名	シート名
定義対象	定義対象※
物理マップの名称	物理マップ名

マップ定義の情報	添字に指定するキーワード
出力論理マップ名称	出力論理マップ名
入力論理マップ名称	入力論理マップ名

注

キーワードの仮名文字は，半角仮名文字を使用します。

注※

定義対象には，次の表の値が設定されます。

定義対象の設定値	意味
G1	GUI 画面（1 次）
G2	GUI 画面（2 次）
CU	CUI 画面
SR	けい線帳票
PR	プレプリント帳票
LP	網掛け帳票
GR	グラフィック帳票

注

定義対象は，半角英数字で設定されます。

使用例

論理マップの展開と画面入出力処理の例です。

テンプレート

```
@@set @共通情報 = @@xmap3common(@画面);
*         論理マップのCOPY
          COPY @共通情報［入力論理ﾏｯﾌﾟ名］..
          COPY @共通情報［出力論理ﾏｯﾌﾟ名］..
                :
*         物理マップ名称の設定
          MOVE  '@共通情報［物理ﾏｯﾌﾟ名］'  TO  画面マップ名
*         画面の表示と入力
          TRANSCEIVE DSP FROM @共通情報［出力論理ﾏｯﾌﾟ名］INTO  @共通情報［入力論理ﾏｯﾌﾟ名］
```

↓

生成結果

```
*         論理マップのCOPY
          COPY  MAP001CI.
          COPY  MAP001CO.
                :
*         物理マップ名称の設定
          MOVE  'MAP001NC'  TO  画面マップ名
*         画面の表示と入力
          TRANSCEIVE  DSP  FROM  MAP001CO  INTO  MAP001CI
```

7.9.53 @@xmap3objects 関数

形 式

@@xmap3objects（可変記号）

機 能

@@interface 文で XMAP3 を指定した可変記号からマップ定義のオブジェクト情報を取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- 可変記号は @@interface 文で ATTR=XMAP3 を指定した可変記号を指定する。
- データ項目の定義情報は，@@set 文の左辺の可変記号に二次元の配列で設定される。
- 一次元の配列は，マップ定義のオブジェクト情報ごとに確保される。一次元の配列の添字には，数値を指定する。
- 二次元の配列は，マップ定義のオブジェクト情報の種類ごとに確保される。二次元の配列の添字には，キーワードを指定する。
  取得できるデータ項目の定義情報を次に示します。

マップ定義の情報	添字に指定するキーワード
オブジェクトの位置（行）	行
オブジェクトの位置（列）	列
オブジェクトの種類（大分類）	種別※1
オブジェクトの区分（小分類）	区分※2
オブジェクトの使用目的	使用目的※3
データ名（接頭語・接尾語なし）	データ名
動的変更	動的変更※4
出力用のデータ長	データ長出
出力用のデータ型	データ型出※5
入力用のデータ長	データ長入
入力用のデータ型	データ型入※5
反復回数（縦）	反復縦
反復回数（横）	反復横
反復間隔（縦）	間隔縦
反復間隔（横）	間隔横
数字編集文字列	数字編集文字
コメント	コメント
固定テキストのテキスト / 初期値 / トグル種別※6	テキスト

注
キーワードの仮名文字と英数字は，半角文字を指定します。

注※ 1
「種別」に設定される値については「表 7-9 種別に設定される値」を参照してください。

注※ 2
「区分」に設定される値については「表 7-10 区分に設定される値」を参照してください。

注※ 3
「使用目的」に設定される値については「表 7-11 使用目的に設定される値」を参照してください。

注※ 4
「動的変更」に設定される値については「表 7-12 動的変更に設定される値」を参照してください。

注※ 5
「データ型出」，および「データ型入」に設定される値については「表 7-13 データ型出，およびデータ型入に設定される値」を参照してください。

注※ 6
「トグル種別」に設定される値については「表 7-14 トグル種別に設定される値」を参照してください。

表 7-9　種別に設定される値

種類の設定値	XMAP3 のオブジェクトの種類（大分類）
隠し	隠しフィールド
固定	固定テキスト / フィールド
出力	出力テキスト / 出力日付テキスト / 出力時刻テキスト / 出力フィールド / 出力日付フィールド / 出力時刻フィールド
出力従	出力テキスト / 出力日付テキスト / 出力時刻テキスト / 出力フィールド / 出力日付フィールド / 出力時刻フィールド / スピンボックス / 出力バーコード / 出力 OCR の下位項目，または入出力テキスト / 入出力日付テキスト / 入出力時刻テキスト / 入出力フィールド / 入出力日付フィールド / 入出力時刻フィールドの出力項目の下位項目
入力	入出力テキスト / 入出力日付テキスト / 入出力時刻テキスト / 入出力フィールド / 入出力日付フィールド / 入出力時刻フィールド
入力従	入出力テキスト / 入出力日付テキスト / 入出力時刻テキスト / 入出力フィールド / 入出力日付フィールド / 入出力時刻フィールド / スピンボックスの入力項目の下位項目
スピン	スピンボックス
固 POP	固定ポップアップ / フィールド
可 POP	可変ポップアップ / フィールド

種類の設定値	XMAP3 のオブジェクトの種類（大分類）
固コンボ	固定コンボボックス
可コンボ	可変コンボボックス
単リスト	単一選択リストボックス
複リスト	複数選択リストボックス
プッシュ	プッシュボタン
固ラジオ	固定ラジオボタン
可ラジオ	可変ラジオボタン
固チェック	固定チェックボタン
可チェック	可変チェックボタン
フィールド	フィールドボックス
固グラフ	固定グラフィック
出グラフ	出力グラフィック
アイコン	アイコン
出 BCD	出力バーコード
出 OCR	出力 OCR
トグル	トグルフィールド
フレーム	フレーム
グループ	グループボックス
予約	予約テキスト / 予約フィールド

注

設定値の仮名文字と英数字は，半角文字で設定されます。

表 7-10　区分に設定される値

区分の設定置	XMAP3 のオブジェクトの区分（小分類）
BOX	ボックス
メニュー	メニュー
ボタン	ボタン
スタート	フレームのスタート
エンド	フレームのエンド

注

設定値の仮名文字と英数字は，半角文字で設定されます。

表 7-11 使用目的に設定される値

使用目的の設定値	対応する XMAP3 の使用目的
文字	日本語
数字	数字
英数	英数
カナ	カナ
数値	数値
パス	パスワード
MCR	MCR
金額	金額
日付	日付
時刻	時刻
手数	POP（手動－数字）
手英	POP（手動－英数）
手カナ	POP（手動－カナ）
手日	POP（手動－日本語）
自英	POP（自動－英数）
自日	POP（自動－日本語）
ファイル	メニューデータをファイル指定する可変ポップアップ
LAB	ラベル
COD	通知コード
KEY	アクセスキー
TXT	選択ラベル
FLD	可変チェックボタンボックス / 複数選択リストボックスの選択データ
CD39	CODE39（バーコード）
JN13	JAN（13）（バーコード）
JAN8	JAN（8）（バーコード）
ITF	ITF（バーコード）
NW7	NW7（バーコード）
カスタマ	カスタマ（バーコード）

注

設定値の仮名文字と英数字は，半角文字で設定されます。

表 7-12 動的変更に設定される値

使用目的の設定値	XMAP3 での動的変更の指定
Y	動的変更あり
N	動的変更なし

注

設定値の仮名文字と英数字は，半角文字で設定されます。

表 7-13 データ型出，およびデータ型入に設定される値

データ型の設定値	XMAP3 でのデータ型の指定
X	文字
N	漢字
9	数字または数字編集

注

設定値の仮名文字と英数字は，半角文字で設定されます。

表 7-14 トグル種別に設定される値

トグル種別の設定値	XMAP3 でのトグル種別の指定
チェック	チェック（トグル）
O/X	O/X（トグル）
Y/N	Y/N（トグル）

注

設定値の仮名文字と英数字は，半角文字で設定されます。

使用例

XMAP3 で定義しているオブジェクトの種類，使用目的，データ名および動的変更の有無を展開します。

テンプレート

```
@@interface @画面 = {ATTR=XMAP3} ;
@@set @画面データ  = @@xmap3objects（@画面）;
  最初のデータ名=@画面データ[1,ﾃﾞｰﾀ名]
                    :
@@foreach @データ  （@画面データ）
  @@if(@データ[種別] ne ″固定″)   @@*固定テキストは除外
@データ[種別],@データ[使用目的],@データ[ﾃﾞｰﾀ名],@データ[動的変更]
  @@end;
@@end;
```

生成結果

```
  最初のデータ名=商品コード
        :
入出力,英数,商品コード,Y
出力,英数,商品名,N
```

7.9.54 @@xmlattribute 関数

形　式

@@xmlattribute（可変記号，" 属性名 "）

機　能

可変記号に設定された要素ノードから，" 属性名 " と一致する属性名を持つ属性ノードを取得するときに使用します。

規　則

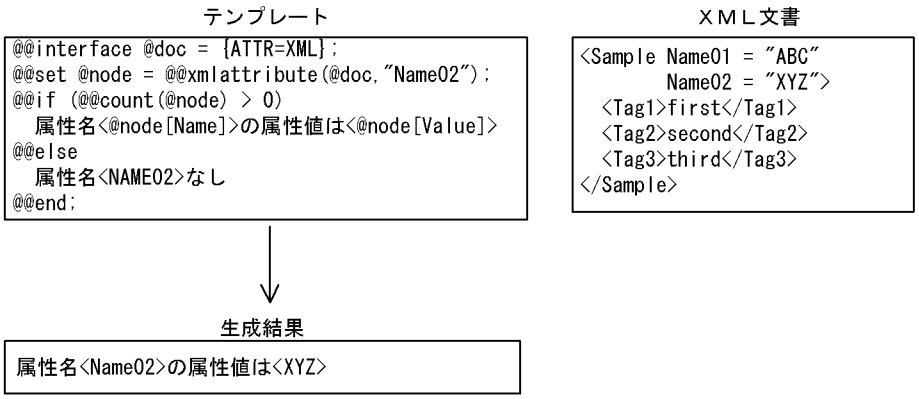
- @@set 文の右辺に指定する。
- 可変記号には，要素ノードが設定された可変記号を指定する。
- 属性ノードの定義情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列は属性ノードの情報の種類ごとに確保される。配列の添字には，キーワードを指定する。

添字に指定するキーワード	属性ノードの情報
Type	タイプ（固定値：3）
Name	属性名
Value	属性値

- 属性ノードが取得できたかどうかは，@@count 関数を使用して判定する。
  @@count 関数は，取得できない場合には 0 を返し，取得できた場合には 1 以上を返す。

使用例

@@xmlattribute 関数を使用して，属性名が Name2 の属性ノードを取得します。



7.9.55 @@xmlchildnodes 関数

形 式

@@xmlchildnodes（可変記号）

機 能

可変記号に設定されているノードの，すべての子供ノードを取得するときに使用します。要素ノードおよびテキストノードが取得の対象となります。

規 則

- @@set 文の右辺に指定する。
- 可変記号はノードを示す可変記号を指定する。
- 一次元の配列は子供ノードごとに確保される。配列の添字には数値を指定する。
- 二次元の配列は，要素ノードの情報の種類ごとに確保される。配列の添字には，キーワードを指定する。
- 要素ノードの場合，次の情報が取得できる。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	タグ名

- テキストノードの場合，次の情報が取得できる。

添字に指定するキーワード	テキストノードの情報
Type	タイプ（固定値：6）
Name	名前 （固定値：#text）
Value	タグで囲まれた文字列

- 子供ノードが取得できたかどうかは，@@count 関数を使用して判定する。
  @@count 関数は，取得できない場合には 0 を返し，取得できた場合には子供ノードの数を返す。

使用例

@@xmlchildnodes 関数を使用して，Sample ノードのすべての子供ノードを取得します。

テンプレート

```
@@interface @doc = {ATTR=XML};
@@set @nodes = @@xmlchildnodes(@doc);
<@doc[Name]>の子供ノード一覧
@@foreach @node (@nodes)
  @@switch (@node[Type])
    @@case 2:
・要素ノード タグ名=@node[Name]
      @@break;
    @@case 6:
・テキストノード 値=@node[Value]
      @@break;
    @@default:
・その他 Type=@node[Type]
  @@end;
@@end;
```

ＸＭＬ文書

```
<Sample Name01 = "ABC"
        Name02 = "XYZ">
  <Tag1>first</Tag1>
  <Tag2>second</Tag2>
  <Tag3>third</Tag3>
</Sample>
```

生成結果

```
<Sample>の子供ノード一覧
・要素ノード タグ名=Tag1
・要素ノード タグ名=Tag2
・要素ノード タグ名=Tag3
```

7.9.56 @@xmlelements 関数

形　式

@@xmlelements（可変記号，" タグ名 "）

機　能

" タグ名 " と一致するタグ名を持つ要素ノードを取得するときに使用します。可変記号に設定されている要素ノードの下位にあるすべての要素ノードが，検索の対象となります。

規　則

- @@set 文の右辺に指定する。
- 可変記号には，要素ノードを示す可変記号を指定する。
- タグ名に '*' を指定した場合，下位のすべてのノードが取得される。

- データ項目の定義情報は，@@set 文の左辺の可変記号に二次元の配列で指定される。
- 一次元の配列は，指定されたタグ名に該当する要素ノードごとに確保される。一次元の配列の添字には，数値を指定する。
- 二次元の配列は，要素ノードの情報の種類ごとに確保される。二次元の配列の添字には，キーワードを指定する。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	タグ名

- 要素ノードが取得できたかどうかは，@@count 関数を使用して判定する。
  @@count 関数は，取得できない場合には 0 を返し，取得できた場合には要素ノードの数を返す。

使用例

@@xmlelements 関数を使用して，タグ名が Tag2 の要素ノードを取得します。

テンプレート

```
@@interface @doc = {ATTR=XML};
@@set @tags = @@xmlelements(@doc,"Tag2");
@@set @tag2 = @tags[1];
@@set @text = @@xmlfirstchild(@tag2);
<@tags[1,Name]>のテキストは<@text[Value]>です。
```

ＸＭＬ文書

```
<Sample Name01 = "ABC"
        Name02 = "XYZ">
  <Tag1>first</Tag1>
  <Tag2>second</Tag2>
  <Tag3>third</Tag3>
</Sample>
```

生成結果

```
<Tag2>のテキストは<second>です。
```

7.9.57 @@xmlfirstchild 関数

形　式

@@xmlfirstchild（可変記号）

機　能

可変記号に設定されているノードの最初の子供ノードを取得するときに使用します。

規　則

- @@set 文の右辺に指定する。
- 可変記号にはノードを示す可変記号を指定する。
- 取得したノードの情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列は，ノードの情報の種類ごとに確保される。配列の添字には，キーワードを

指定する。

- 取得できたノードの種類は，ノード情報のタイプで判断する。
- 要素ノードの場合，次の情報が取得できる。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	タグ名

- テキストノードの場合，次の情報が取得できる。

添字に指定するキーワード	テキストノードの情報
Type	タイプ（固定値：6）
Name	名前 （固定値：#text）
Value	タグで囲まれた文字列

- ノードが取得できたかどうかは，@@count 関数を使用して判定する。@@count 関数は，取得できない場合には 0 を返し，取得できた場合には 1 以上を返す。

使用例

@@xmlfirstchild 関数を使用して，Sample ノードから最初の子供ノードを取得します。

テンプレート

```
@@interface @doc = {ATTR=XML};
@@set @node = @@xmlfirstchild(@doc);
<@doc[Name]>の最初の子供ノードは<@node[Name]>です。
```

XML文書

```
<Sample Name01 = "ABC"
        Name02 = "XYZ">
  <Tag1>first</Tag1>
  <Tag2>second</Tag2>
  <Tag3>third</Tag3>
</Sample>
```

生成結果

```
<Sample>の最初の子供ノードは<Tag1>です。
```

7.9.58 @@xmllastchild 関数

形　式

@@xmllastchild（可変記号）

機　能

可変記号に設定されているノードの最後の子供ノードを取得するときに使用します。

規　則

- @@set 文の右辺に指定できる。

- 可変記号にはノードを示す可変記号を指定する。
- 取得したノードの情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列はノードの情報の種類ごとに確保される。配列の添字には，キーワードを指定する。
- 取得したノードの種類はノード情報のタイプで判断する。
- 要素ノードの場合，次の情報が取得できる。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	タグ名

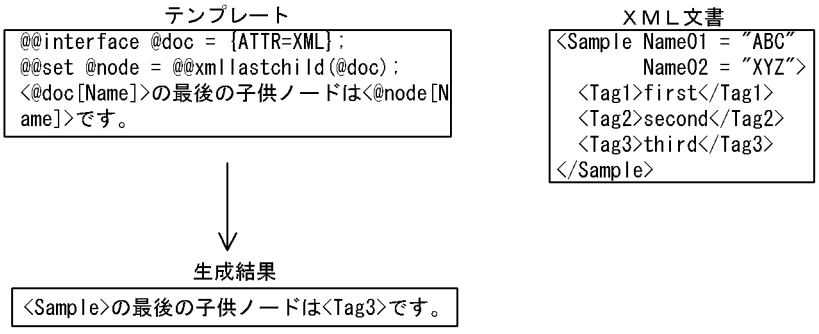
- テキストノードの場合，次の情報が取得できる。

添字に指定するキーワード	テキストノードの情報
Type	タイプ（固定値：6）
Name	名前 （固定値：#text）
Value	タグで囲まれた文字列

- ノードが取得できたかどうかは，@@count 関数を使用して判定する。@@count 関数は，取得できない場合には 0 を返し，取得できた場合には 1 以上を返す。

使用例

@@xmllastchild 関数を使用して，Sample ノードから最後の子供ノードを取得します。



7.9.59 @@xmlnext 関数

形　式

@@xmlnext（可変記号）

機　能

可変記号に設定されているノードと同じ親ノードを持つ直後のノードを取得するときに使用します。

規　則

- @@set 文の右辺に指定する。
- 可変記号にはノードを示す可変記号を指定する。
- 取得したノードの情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列は，ノードの情報の種類ごとに確保される。配列の添字には，キーワードを指定する。
- 取得できたノードの種類はノード情報のタイプで判断する。
- 要素ノードの場合，次の情報が取得できる。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	タグ名

- テキストノードの場合，次の情報が取得できる。

添字に指定するキーワード	テキストノードの情報
Type	タイプ（固定値：6）
Name	名前　（固定値：#text）
Value	タグで囲まれた文字列

- ノードが取得できたかどうかは，@@count 関数を使用して判定する。@@count 関数は，取得できない場合には 0 を返し，取得できた場合には 1 以上を返す。

使用例

@@xmlnext 関数を使用して，Tag2 ノードから直後のノードを取得します。

テンプレート

```
@@interface @doc = {ATTR=XML};
@@set @tags = @@xmlelements(@doc,"Tag2");
@@set @tag2 = @tags[1];
@@set @node = @@xmlnext(@tag2);
<@tags[1,Name]>の直後のノードは<@node[Name]>です。
```

ＸＭＬ文書

```
<Sample Name01 = "ABC"
        Name02 = "XYZ">
  <Tag1>first</Tag1>
  <Tag2>second</Tag2>
  <Tag3>third</Tag3>
</Sample>
```

生成結果

```
<Tag2>の直後のノードは<Tag3>です。
```

7.9.60 @@xmlparent 関数

形 式

@@xmlparent（可変記号）

機 能

可変記号に設定されているノードの親ノードを取得するときに使用します。

規 則

- @@set 文の右辺に指定する。
- 可変記号にはノードを示す可変記号を指定する。
- 取得したノードの情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列は，ノードの情報の種類ごとに確保される。配列の添字には，キーワードを指定する。
- 次の情報が取得できる。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	名前

- ノードが取得できたかどうかは，@@count 関数を使用して判定する。@@count 関数は，取得できない場合には 0 を返し，取得できた場合には 1 以上を返す。

使用例

@@xmlparent 関数を使用し，Tag2 ノードから親ノードを取得します。

テンプレート

```
@@interface @doc = {ATTR=XML};
@@set @tags = @@xmlelements(@doc,"Tag2");
@@set @tag2 = @tags[1];
@@set @node = @@xmlparent(@tag2);
<@tags[1,Name]>の親ノードは<@node[Name]>です。
```

ＸＭＬ文書

```
<Sample Name01 = "ABC"
        Name02 = "XYZ">
  <Tag1>first</Tag1>
  <Tag2>second</Tag2>
  <Tag3>third</Tag3>
</Sample>
```

生成結果

```
<Tag2>の親ノードは<Sample>です。
```

7.9.61 @@xmlprevious 関数

形 式

@@xmlprevious（可変記号）

機　能

可変記号に設定されているノードと同じ親を持つ直前のノードを取得するときに使用します。

規　則

- @@set 文の右辺に指定する。
- 可変記号にはノードを示す可変記号を指定する。
- 取得したノードの情報は，@@set 文の左辺の可変記号に一次元の配列で設定される。
- 配列は，ノードの情報の種類ごとに確保される。配列の添字には，キーワードを指定する。
- 取得できたノードの種類はノード情報のタイプで判断する。
- 要素ノードの場合，次の情報が取得できる。

添字に指定するキーワード	要素ノードの情報
Type	タイプ（固定値：2）
Name	タグ名

- テキストノードの場合，次の情報が取得できる。

添字に指定するキーワード	テキストノードの情報
Type	タイプ（固定値：6）
Name	名前 （固定値：#text）
Value	タグで囲まれた文字列

- ノードが取得できたかどうかは，@@count 関数を使用して判定する。@@count 関数は，取得できない場合には 0 を返し，取得できた場合には 1 以上を返す。

使用例

@@xmlprevious 関数を使用し，Tag2 ノードから直前のノードを取得します。

テンプレート

```
@@interface @doc = {ATTR=XML};
@@set @tags = @@xmlelements(@doc,"Tag2");
@@set @tag2 = @tags[1];
@@set @node = @@xmlprevious(@tag2);
<@tags[1,Name]>の直前のノードは<@node[Name]>です。
```

ＸＭＬ文書

```
<Sample NameO1 = "ABC"
        NameO2 = "XYZ">
  <Tag1>first</Tag1>
  <Tag2>second</Tag2>
  <Tag3>third</Tag3>
</Sample>
```

生成結果

```
<Tag2>の直前のノードは<Tag1>です。
```

付録

付録 A　環境設定

付録 A.1　SEWB+/CONSTRUCTION の環境設定

SEWB+/CONSTRUCTION の環境設定を，次に示します。環境情報を設定したり更新したりする場合は，環境設定ダイアログを利用してください。

表 A-1　SEWB+/CONSTRUCTION の環境設定

ダイアログ中での表示名	設定内容	該当個所	省略時の値
エディタの指定	ユーザ処理での UOC 編集時や編成のリフォーム使用時，または生成時に，ソースプログラムを表示させるエディタ。	プログラム定義	notepad.exe
インデントサイズ	ソースプログラム生成時のインデンテーションの長さ。0 ～ 8 の範囲で指定する。 （COBOL だけ有効）	ソースプログラム生成 レコードソース生成	2
COBOL ソース正書法※1	ソースプログラム生成およびレコードソース生成時のソース形式。 （固定形式／フリー形式／ホスト向け固定形式） （COBOL だけ有効）	ソースプログラム生成 レコードソース生成	固定形式
データ定義およびレコード定義で辞書の自動再入力をしない	レコード定義がチェックインされている場合に，レコード定義ファイルを参照するデータ定義やプログラム定義が，レコード定義ファイルに辞書の変更内容が反映されたように動作するかどうかの設定。 また，データ定義がチェックインされている場合に，データ定義ファイルを参照するプログラム定義が，データ定義ファイルに辞書変更内容が反映されたように動作するかどうかの設定。	データ定義 レコード定義	データ定義およびレコード定義で辞書の自動再入力をする
テンプレートファイルパス	プログラム定義で利用するテンプレートファイルを検索するパス。複数のパスを指定できる。	プログラム定義	カレントパス
データ定義ファイルパス	プログラム定義で利用するデータ定義ファイルを検索するパス。複数のパスを指定できる。	プログラム定義	カレントパス
論理設計図ファイルパス	プログラム定義で利用する論理設計図ファイルを検索するパス。複数のパスを指定できる。	プログラム定義	カレントパス
部品ファイルパス	ソースプログラム生成時に，部品を検索するパス。複数のパスを指定できる。	ソースプログラム生成	カレントパス
XML ファイルパス	プログラム定義で利用する XML ファイルを検索するパス。複数のパスを指定できる。	プログラム定義	カレントパス

ダイアログ中での表示名	設定内容	該当個所	省略時の値
マップ定義ファイルパス	プログラム定義で利用するマップ定義ファイルの検索パス。複数のパスを指定できる。	プログラム定義	カレントパス
レコード定義ファイルパス	データ定義で利用するレコード定義ファイルの検索パス。複数のパスを指定できる。 SEWB+/RECORD DEFINER のインストールの有無に関係なく指定できる。	データ定義	カレントパス
PIC 句生成開始位置※2	PICTURE 句，OCCURS 句および VALUE 句の開始位置。COBOL ソース正書法がフリー形式の場合は 1 ～ 60，固定形式またはホスト向け固定形式の場合は 8 ～ 60 の範囲で指定できる。	ソースプログラム生成 レコードソース生成	41
すべての業務ルールを無条件に展開	業務ルール展開設定を無条件に展開しない（ユーザが選択する）か，無条件に展開する（全選択にする）かの設定。	プログラム定義	無条件に展開しない
ソースプログラムを自動清書しない※3	プログラム定義からソースプログラムを生成するときに清書するか，清書しないかの設定。 （COBOL だけ有効）	ソースプログラム生成 レコードソース生成	自動清書する
ユーザ追加処理をそのまま生成する※4	ソースプログラム生成時にユーザ追加処理を指定されたとおりにそのまま生成するか，展開文と同じように扱って生成するかの設定。 （COBOL だけ有効）	ソースプログラム生成	ユーザ追加処理を展開文と同じように扱って生成する。
一連番号の付加	ソースプログラム生成およびレコードソース生成時に，一連番号を付加するかどうかの設定。 （COBOL だけ有効）	ソースプログラム生成 レコードソース生成	付加しない
初期値	一連番号を付加する場合の初期値。1 ～ 999999 の範囲で指定する。「一連番号の付加」をチェックしていない場合は，不活性になる。 （COBOL だけ有効）	ソースプログラム生成	100
増分値	一連番号を付加する場合の増分値。1 ～ 999999 の範囲で指定する。「一連番号の付加」をチェックしていない場合は，不活性になる。 （COBOL だけ有効）	ソースプログラム生成	100
開始コメント※5	ユーザ追加処理のリバースで，ソースプログラムからユーザ追加処理を検索するときに使用するユーザ追加処理の開始コメントを指定する。	ユーザ追加処理のリバース	なし
終了コメント※5	ユーザ追加処理のリバースで，ソースプログラムからユーザ追加処理を検索するときに使用するユーザ追加処理の終了コメントを指定する。	ユーザ追加処理のリバース	なし

ダイアログ中での表示名	設定内容	該当個所	省略時の値
プログラム定義ファイルの新規格納形式	プログラム定義でファイルを保存する場合に，バイナリ形式（*.csp）にするかXML 形式（*.csq）にするかを指定する。	プログラム定義	バイナリ形式（*.csp）
データ定義ファイルの新規格納形式	データ定義でファイルを保存する場合に，バイナリ形式（*.csd）にするかXML 形式（*.cse）にするかを指定する。	データ定義	バイナリ形式（*.csd）

注※ 1

固定形式またはホスト向け固定形式でソースプログラムを生成する場合は，COBOL 言語の固定形式またはホスト向け固定形式の規則に従ってユーザ追加処理を編集してください。
ホスト向け固定形式では，2 バイトコードがある場合，機能キャラクタ占有領域が考慮されます。また，項目名が 30 バイト（機能キャラクタ占有領域を含めて）を超えた場合，31 バイト以降は切り捨てられます。

注※ 2

データ項目名が，指定された PICTURE 句の生成開始位置を超えている場合，PICTURE 句生成開始位置の指定は無視されます。PICTURE 句は，データ項目名から 1 文字分のスペースを開けて生成されます。
また，データ項目名と PICTURE 句が同行に入らない場合は，PICTURE 句は改行されて生成されます。そのときの生成位置は，次の行のレベル番号と同じ位置です。なお，OCCURS 句，VALUE 句の場合も同様です。

注※ 3

「ソースプログラムを自動清書しない」の設定に関係なく，「PIC 句生成開始位置」および「インデントサイズ」はデータ定義のレコード生成時（プログラム定義の@@ expand 文の展開を含む）に有効です。

注※ 4

COBOL 正書法では，B 領域を超える部分はコメントとして扱われます。しかし，展開文に可変記号を使用する場合，生成したソースが B 領域を超えないようなテンプレートや部品を作成するのは困難です。このため，SEWB ＋ /CONSTRUCTION では，生成ソースが B 領域を超える場合，複数行に分割して COBOL ソースを生成しています。
「ユーザ追加処理をそのまま生成する」オプションは，ユーザ追加処理をほかのテンプレートや部品から生成されるソースだけ B 領域を超えないよう分割し，ユーザ追加処理は指定されたまま生成する機能です。

注※ 5

@@lang 文に UOC_BEGIN ／ UOC_END が指定されている場合は無効になります。

付録 A.2　プログラム定義で選択したファイル名の保存方法

プログラム定義で選択したファイルの格納場所を保存する場合，そのファイルを相対パスで検索したか絶対パスで検索したかによって保存方法が異なります。

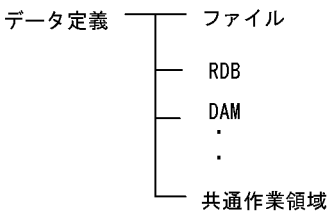
(1) 相対パスと絶対パス

● 相対パスでの保存
環境設定でパスを指定した場合，フォルダを除いた形（相対パス）でファイル名を保存します。相対パスでの保存をお勧めします。

● 絶対パスでの保存
環境設定でパスを指定しない場合，ドライブ名およびコンピュータ名を含むファイル名（絶対パス）で保存します。

注
絶対パスから相対パスに変更する場合は，コマンド「CSPDCNVP.EXE」を実行する必要があります。コマンド「CSPDCNVP.EXE」については，「(2) 絶対パスから相対パスへの変換」を参照してください。

環境設定で指定するパスの下にフォルダを置くこともできます。データ定義は，定義種別ごとにサブフォルダを分けた方が便利な場合もあります。



(2) 絶対パスから相対パスへの変換

プログラム定義ファイルに格納されているテンプレートファイル，データ定義ファイルなどのファイルパスを絶対パスから相対パスに変換する場合，コマンド「CSPDCNVP.EXE」を実行させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

形 式

CSPDCNVP.EXE△変換元ファイル名
{[△/o△変換先ファイル名] | [△/op△変換先ファイル出力先パス名]}

```
[△/t△トレースファイル名]
[△/help]
[△/?]
```

（凡例）

△：1 文字以上の空白を示します。

注

{　} はどちらかを指定することを示します。

解 説

変換元ファイル名：

パスを変換するプログラム定義ファイルのファイル名※ 1 を指定します。

/o △変換先ファイル名※ 2：

パスが変換されたプログラム定義ファイルを保存するパスの名称を指定します。

/op △変換先ファイル出力先パス名※ 2：

パスが変換されたプログラム定義ファイルを保存するパスの名称を指定します。ファイル名は，変換元ファイル名となります。

/t △トレースファイル名：

トレース情報の出力先ファイル名を指定します。

/help：

コマンドヘルプメッセージを表示します。

/? ：

コマンドヘルプメッセージを表示します。

注※ 1

ワイルドカードの指定ができます。ファイル名にワイルドカードを指定した場合，/o は指定できません。

注※ 2

/o または /op の指定がない場合，パス変換されたプログラム定義ファイルは，変換元ファイルに上書きされます。

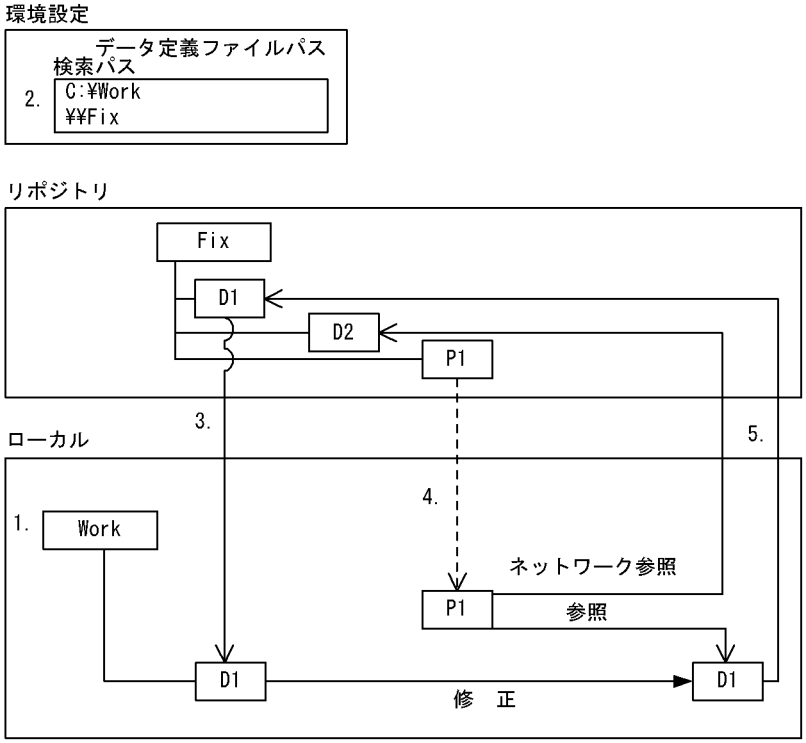
(3) 複数パスの指定

環境設定では，検索パスとして複数のパスを指定できます。複数のパスを指定しておくと，プロジェクトで共用しているテンプレートファイルやデータ定義を修正して確認する場合に便利です。

例えば，データ定義ファイルを修正して確認するために，修正前または修正中のファイルと修正後のファイルを別々のパスで保管します。それぞれのパスを検索したい順番に従って環境設定に設定※しておくと，設定した順番にプログラム定義で使用されます。次に複数パスの設定を利用した作業手順を示します。

注※

SEWB+/CONSTRUCTION 環境設定ダイアログの検索パスの欄で，検索したい順番に従って上から順に設定してください。



1. **データ定義を修正するために，修正作業用のフォルダ「Work」をローカルに作る。**
   このとき，リポジトリには次のドキュメントが格納されていると仮定します。これらはリポジトリのフォルダ「Fix」に格納されています。
   - D1：修正するデータ定義ファイル
   - D2：修正しないデータ定義ファイル
   - P1：D1，D2 を参照するプログラム
2. **環境設定で「Work」「Fix」の順番で複数パスを設定しておく。**
   「Work」はローカルパスで，「Fix」はネットワークパスでそれぞれ設定します。修正作業用のフォルダを必ず最初に設定してください。
3. **リポジトリから修正したいデータ定義を「Work」にチェックアウトして，修正する。**
4. **修正が終わったら，修正したデータ定義の妥当性をテストするために，プログラム定義を起動させる。**
   複数パスの設定で先に「Work」が設定されているため，必ず「Work」のデータ定義が参照されます。「P1」が参照する「D1」は修正された「Work」フォルダのものが参照され，「D2」はリポジトリの「Fix」フォルダのものがネットワーク参照されます。

5. **テストでデータ定義の妥当性が確認されたら，「Work」のデータ定義を「Fix」にチェックインします。**
   以降，プログラム定義を起動すると，複数パスの最初に設定された「Work」には該当するデータ定義がないため，2 番目に設定された「Fix」のデータ定義を参照するようになります。

（4） 複数パス設定の注意事項

環境設定で複数のパスを設定した場合，先に設定されたパスが優先されます。例えば，「C:¥ 修正中データ定義」「¥¥NET¥ データ定義」の順番に設定されている場合，「¥¥NET¥ データ定義」のファイルを参照したい場合でも，同じ名前のファイルが「C:¥ 修正中データ定義」にあれば，そちらのファイルが参照されます。

付録 B　障害対策

SEWB+/CONSTRUCTION での作業中に障害が起きた場合はトレース出力ファイルに実行情報を出力し，保守員に連絡してください。

付録 B.1　データ定義時に障害が起きた場合

データ定義時に障害が起きた場合は，コマンド「CSDDDEF.EXE」または「CSDDGEN.EXE」でデータ定義ファイルを実行させ，実行情報をトレースファイルに出力させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

(1) データ定義の起動

形　式

CSDDDEF.EXE△[入力ファイル名]
　　△/t△トレースファイル名

（凡例）
　△：1 文字以上の空白を示します。

注
　［　］は省略できることを示します。

解　説

入力ファイル名：
　データ定義ファイル（.csd または .cse）のファイル名を指定します。

/t △トレースファイル名：
　トレース情報※の出力先ファイル名を指定します。

注※
　トレース情報には，データ定義ファイルの実行情報などの情報が出力されます。

(2) レコードソース生成

コマンド「CSDDGEN.EXE」については，「2.4.2　コマンドでの生成（レコードソー

ス)」を参照してください。

付録 B.2　プログラム定義時に障害が起きた場合

プログラム定義時に障害が起きた場合は，コマンド「CSPDDEF.EXE」または「CSPDGEN.EXE」でプログラム定義ファイルを実行させ，実行情報をトレースファイルに出力させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

(1) プログラム定義の起動

形　式

CSPDDEF.EXE△[入力ファイル名]
　　△/t△トレースファイル名

（凡例）
△：1 文字以上の空白を示します。

注
［ ］は省略できることを示します。

解　説

入力ファイル名：
プログラム定義ファイル（.csp または .csq）のファイル名を指定します。

/t △トレースファイル名：
トレース情報※の出力先ファイル名を指定します。

注※
トレース情報には，プログラムの実行情報などの情報が出力されます。

(2) ソースプログラムの生成

コマンド「CSPDGEN.EXE」については，「3.3.2　コマンドでの生成（ソースプログラム)」を参照してください。

付録 C　テンプレートおよび部品の紹介

SEWB+ では，信頼性の高いプログラムを効率良く作成するために，標準的なプログラム作成用のテンプレートと部品の組み合わせを用意しています。

バッチシステムのプログラムを作成するためのバッチシステム向けテンプレート・部品，および C/S システムのプログラムを作成するための C/S システム向けテンプレート・部品があります。

付録 C.1　バッチシステム向けテンプレート・部品

バッチシステムのプログラム作成に利用できるテンプレート・部品を，次に示します。

(1) テンプレート（バッチシステム）

バッチシステム向けテンプレートを次に示します。これらのテンプレートの詳細は，マニュアル「SEWB+　バッチシステム向けアプリケーションフレームワーク・部品　使用の手引」を参照してください。

- ファイルの変換
- ファイルの分配
- レコードの抽出
- レコードの集計
- ファイルのマージ
- シーケンシャルファイルの照合
- シーケンシャルファイルの更新（エラー）
- シーケンシャルファイルの更新（追加）
- ランダムファイルの更新（エラー）
- ランダムファイルの更新（追加）
- 項目チェック
- 突合チェック

(2) 部品（バッチシステム）

バッチシステム向け部品を次に示します。これらの部品の詳細は，マニュアル「SEWB+　バッチシステム向けアプリケーションフレームワーク・部品　使用の手引」を参照してください。

- プログラム情報部品
- ファイル入出力部品

付録 C.2　C/S システム向けテンプレート・部品

C/S システムのプログラム作成に利用できるテンプレート・部品を，次に示します。

（1） テンプレート（C/S システム）

C/S システム向けテンプレートを次に示します。

- **サービス提供プログラム（SPP）用テンプレート**
  SPP メインプログラム
  アプリケーション起動 SPP（EXECAP 処理 SPP）
  メッセージ送受信 SPP（SEND-RECV 処理 SPP）
  デバイスアクセス SPP
- **メッセージ処理プログラム（MHP）用テンプレート**
  MHP メインプログラム
  デバイスアクセス MHP

（2） 部品（C/S システム）

C/S システム向け部品を次に示します。

- OpenTP1 **部品**
  リモートプロシジャコール部品（RPC 部品）
  トランザクション制御部品
  メッセージ制御部品（MCF 部品）
  OpenTP1 専用ユーザファイルの操作部品（DAM，TAM 操作部品）
- RDB **操作部品**
  RDB 操作部品
- SECTION **生成部品**
  デバイスアクセス SECTION 生成部品

付録 D　C/S システムサンプルプログラムの説明

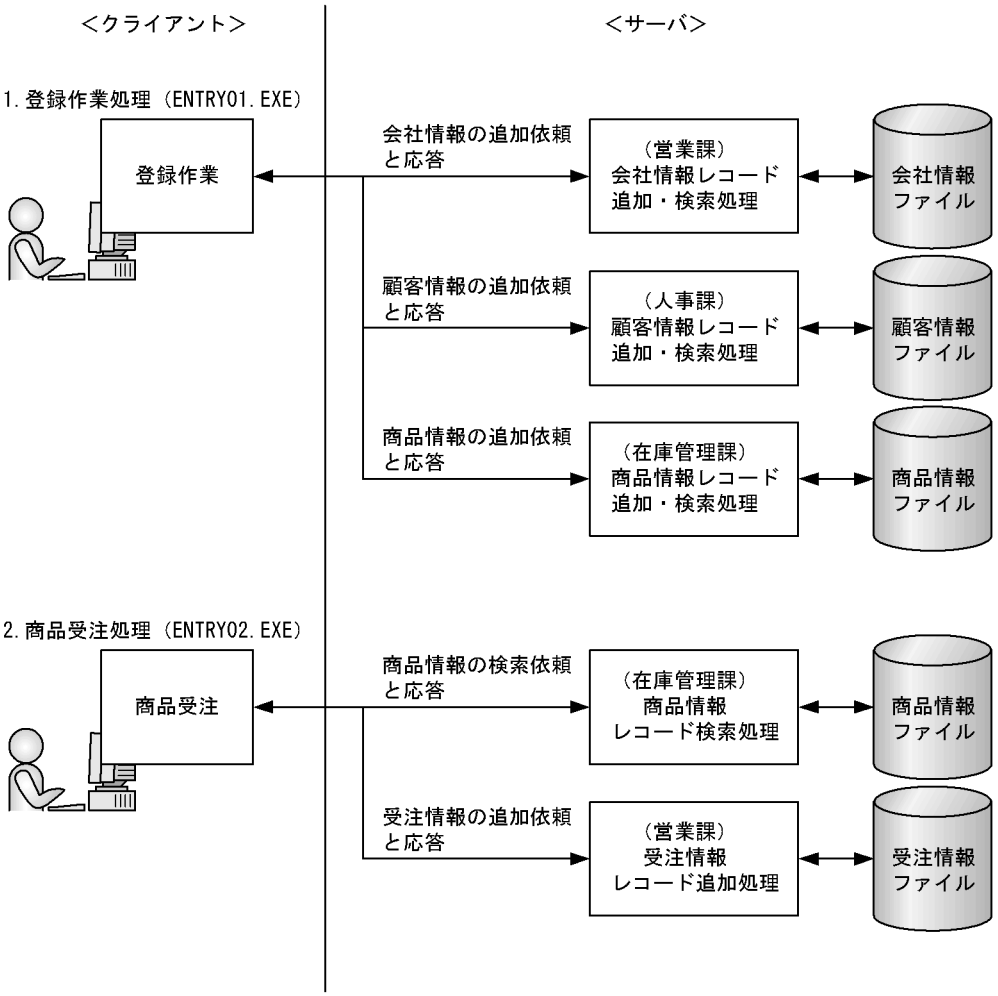
SEWB+/CONSTRUCTION では，実際の業務処理を想定した C/S システムサンプルプログラムが用意されています。SEWB+/CONSTRUCTION を使用してプログラム開発をする場合の，テンプレート作成や，プログラム定義およびデータ定義を，例題を通して見ることができます。また，このサンプルプログラムを，実際にテンプレートを作成したりプログラムの設計をする場合のガイドとして利用することもできます。

なお，C/S システムサンプルを使用したり実行させたりする場合に必要な環境設定については，Readme.TXT を参照してください。

付録 D.1　C/S システムサンプルプログラムの全体の概要

C/S システムサンプルプログラムの業務処理システムは，クライアント側に作成した GUI 画面からデータを入力し，サーバ側の処理 AP を起動させて処理を実行し，その結果を GUI 画面に出力したり，帳票として出力したりするものです。C/S システムサンプルプログラムの全体の概要を次に示します。

図 D-1　C/S システムサンプルプログラムの全体の概要



付録 D.2　C/S システムサンプルプログラムの詳細と構成

C/S システムサンプルプログラムの二つの処理 1. 登録作業処理，2. 商品受注処理について，それぞれの処理内容とシステムの詳細を示します。

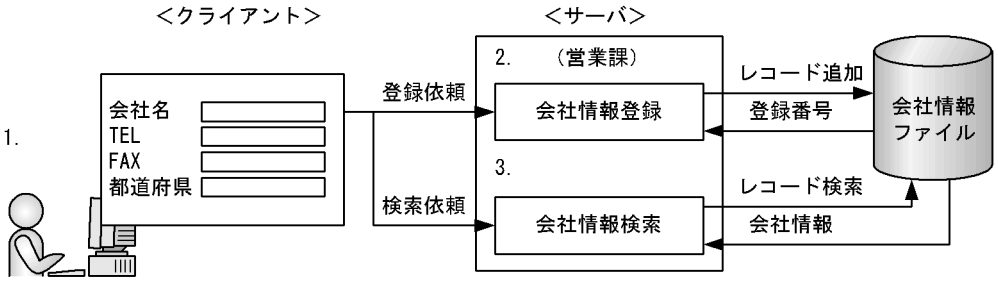
(1) 登録作業処理

登録作業処理には，会社情報を登録する場合，顧客情報を登録する場合および商品情報を登録する場合があり，それぞれアクセスするサーバが異なります。また，この処理では，それぞれの情報を検索する機能も利用できますが，この場合も検索する情報によってアクセスするサーバが異なります。

(a) 登録作業処理の内容

ここでは登録作業処理の内容について示します。なお，図に示されているのは会社情報の登録作業です。顧客情報登録および商品情報登録の場合も，アクセスするサーバファイルや設定する情報が異なるだけで，処理内容は同じです。

図 D-2　登録作業処理の内容



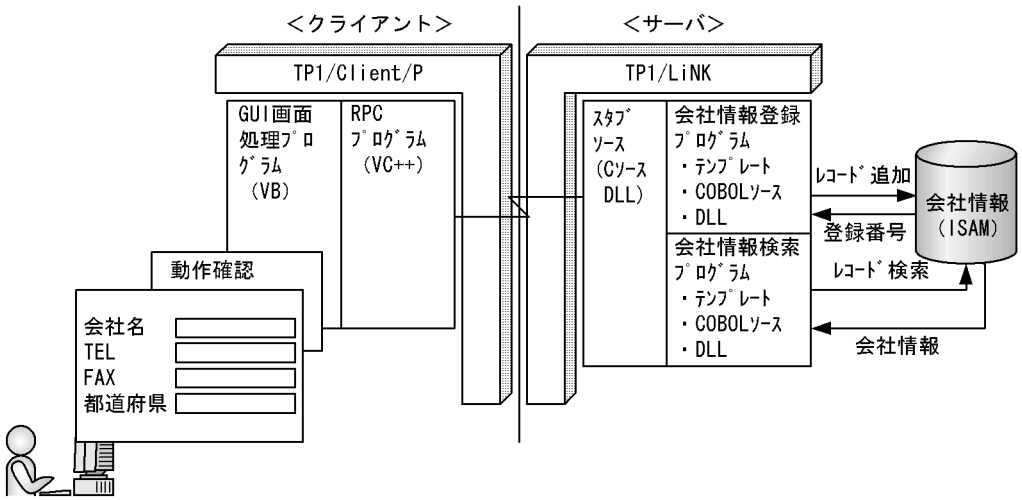
登録作業処理（会社情報の場合）

1. クライアント側の GUI 画面には，会社情報登録画面と顧客情報登録画面および商品登録画面があります。それぞれ営業課，人事課，および在庫管理課のサーバにアクセスします。どのサーバにアクセスするかは，動作環境の設定ダイアログで設定します。
2. 会社情報登録画面から登録依頼をすると，サーバプログラムが起動され，会社情報登録処理が実行されます。会社情報ファイルのレコードが追加され，登録番号が GUI 画面に返されます。
3. 会社情報登録画面から検索依頼をすると，サーバプログラムが起動され，会社情報検索処理が実行されます。会社情報ファイルが検索され，会社情報が GUI 画面に返されます。

(b) 登録処理システムの詳細

ここでは，会社情報登録処理の詳細を示します。顧客情報登録処理や，商品情報登録処理のシステムも同じ構造です。

図 D-3　会社情報登録処理システムの詳細

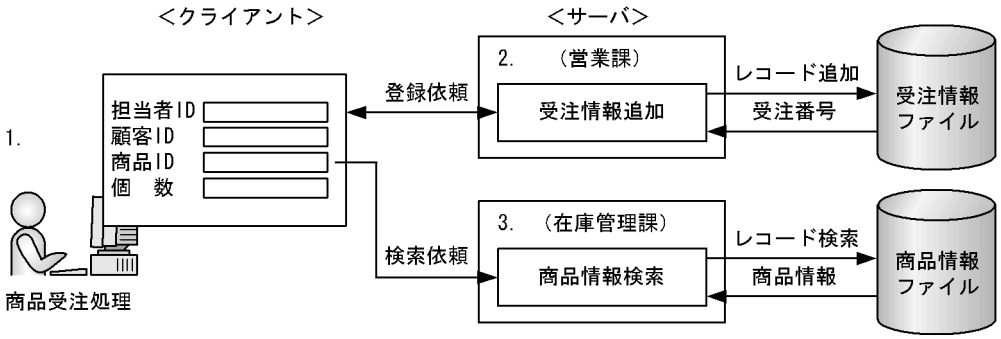


(2) 商品受注処理

商品受注処理では，受注情報を登録します。また商品受注では，画面から入力された商品 ID をキーに商品情報ファイルを検索して，商品名や単価を表示します。

(a) 商品受注処理の内容

図 D-4　商品受注処理の内容



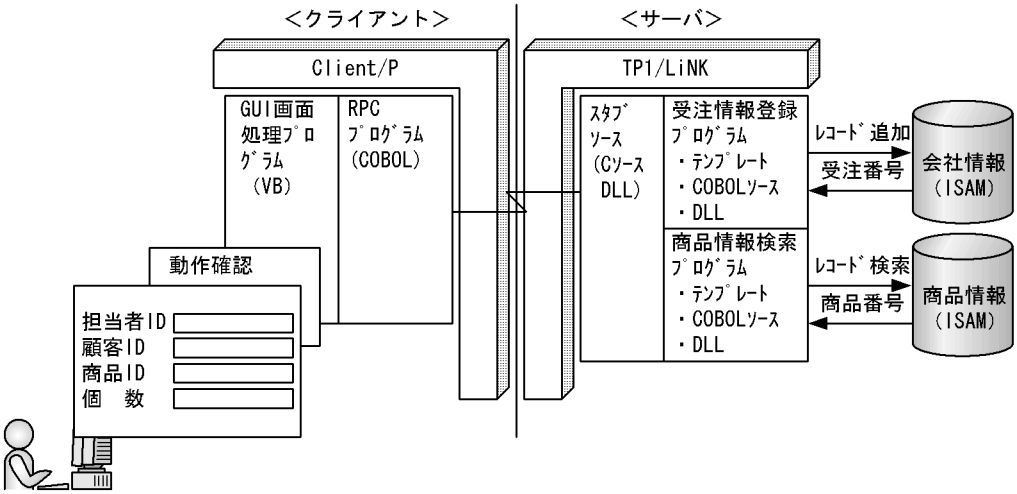
1. クライアント側の GUI 画面には，商品受注画面があり，営業課のサーバにアクセスします。
2. 商品受注画面から受注情報の追加をクリックすると，サーバプログラムが起動され，商品情報検索処理が実行されます。検索された商品情報の商品名称や単価，および合計などが GUI 画面に表示されます。
3. 商品受注画面から受注をクリックすると，サーバプログラムが起動され，受注情報追加処理が実行されます。受注情報ファイルが更新され，受注番号が GUI 画面に返さ

れます。

(b) 商品受注処理システムの詳細

ここでは，商品受注処理の詳細を示します。

図 D-5　商品受注処理システムの詳細



(3) C/S システムサンプルプログラムの構成

サンプルフォルダの構成を示します。

```
CSS_Sample ┬ Readme.txt   : 説明ファイル
           ├ Client       : C/Sシステム-クライアント環境
           ├ Data_def     : データ定義情報
           ├ Etc          : OpenTP1,EURなどの各種情報
           ├ Gui          : 入力画面用(C/Sシステム，ローカル共通)
           ├ Local        : ローカル動作用環境
           ├ Server       : C/Sシステム-サーバ環境
           └ Template     : テンプレートファイル
```

付録 D.3　多様な開発環境に対応する C/S システムサンプルプログラムの詳細と構成

付録 D.2 で説明しているサンプルプログラムでは，プログラミング言語に COBOL，データ管理に ISAM を使用して，OpenTP1 環境下で動作するプログラムを開発します。

SEWB+/CONSTRUCTION では，C 言語や RDB を使用するプログラム開発を対象にしたサンプルプログラムも用意しています。

多様な開発環境に対応する C/S システムサンプルプログラムを，次に示します。

表 D-1　多様な開発環境に対応する C/S システムサンプルプログラム

サンプル名称	通信管理	言語	データ管理	GUI	備考
C/S システムサンプル	Open/TP1	COBOL	順編成ファイル	VB	
C 言語 -HiRDB　サンプル	Open/TP1	C	HiRDB	VB	※
COBOL-HiRDB　サンプル	Open/TP1	COBOL	HiRDB	VB	※
TPBroker-C 言語　サンプル	TPBroker	C	HiRDB	Web Page Generator	※
XMAP3-COBOL 言語サンプル	Open/TP1	COBOL	順編成ファイル	XMAP3	※

注※

C/S システムサンプルの会社情報レコード追加・検索処理だけに提供されています。

これらのサンプルは，次に示すフォルダに格納されています。

SEWB+ 基本開発環境の組み込み先のパス名 ¥Construction¥Sample

SEWB+ 基本開発環境の組み込み先のパス名は，デフォルトでは次のように設定されています。

システムドライブとは，各 OS が組み込まれているドライブです。

<Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合 >

システムドライブ :¥Program Files¥HITACHI¥Sewb

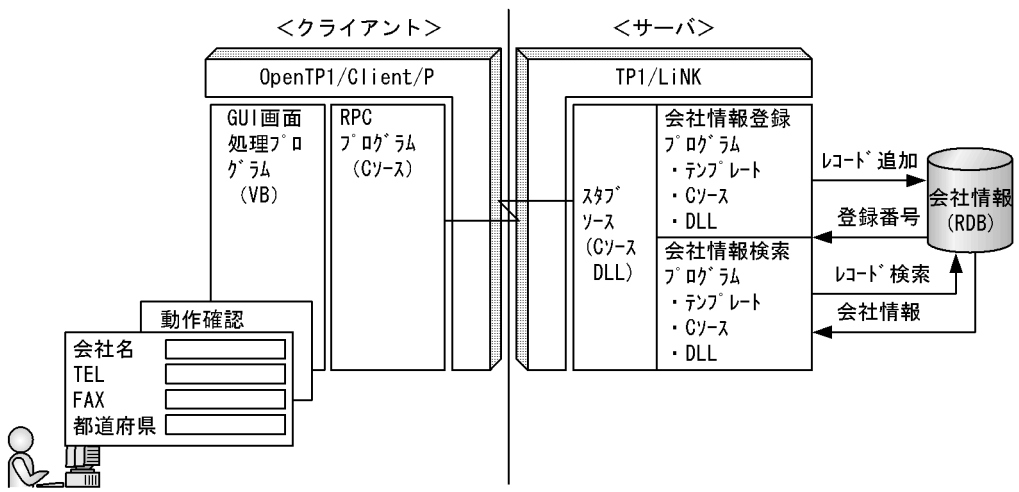
<Windows Server 2003 x64 の場合 >

システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

サンプルフォルダの詳細は，「(5) 多様な開発環境に対応する C/S システムサンプルプログラムの構成」を参照してください。

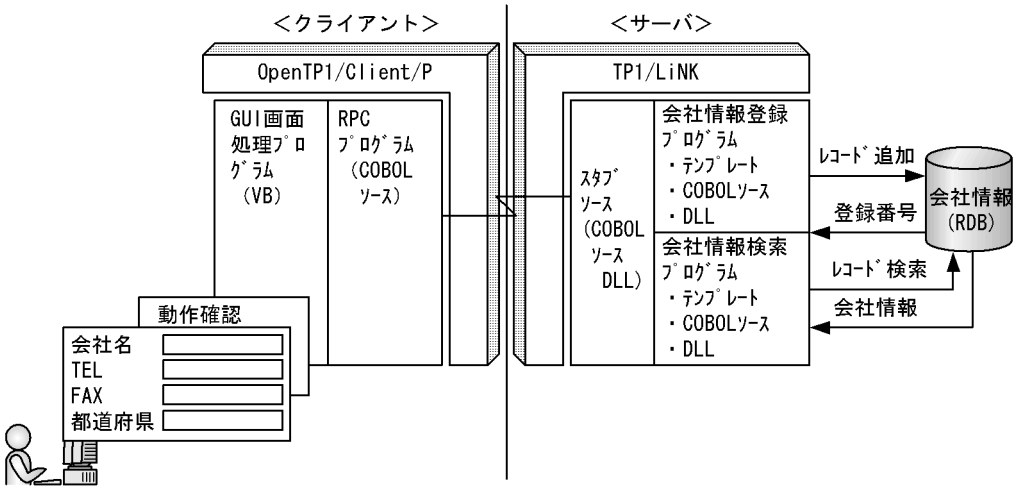
(1) C 言語 -HiRDB サンプルの登録作業処理

C 言語 -HiRDB サンプルの会社情報登録処理を示します。



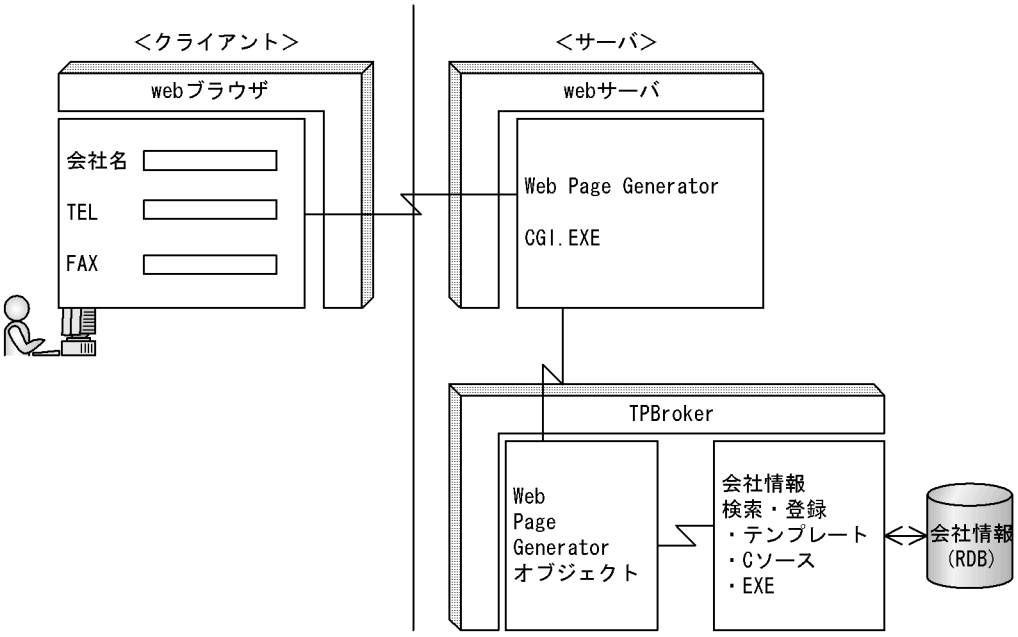
（2） COBOL-HiRDB サンプルの登録作業処理

COBOL-HiRDB サンプルの会社情報登録処理を示します。



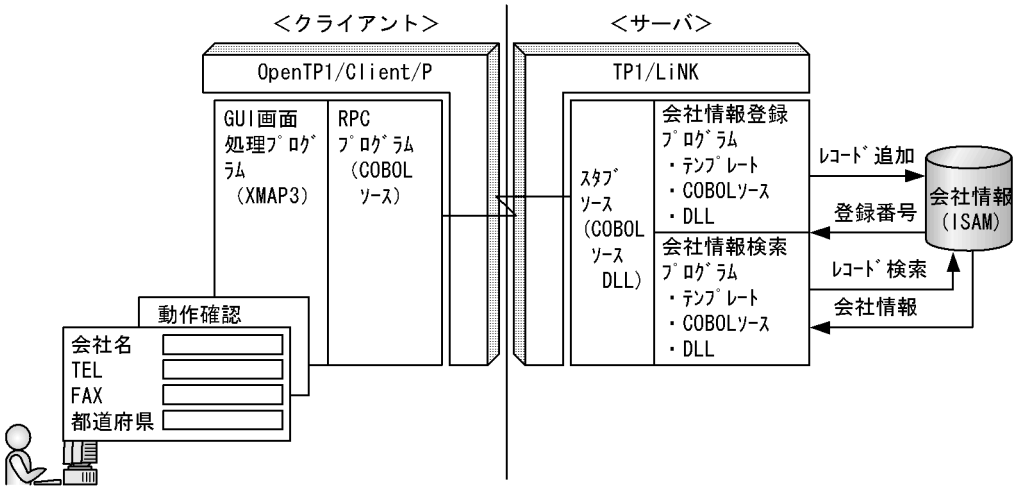
（3） TPBroker-C 言語サンプルの登録作業処理

TPBroker-C 言語サンプルの会社情報登録処理を示します。



(4) XMAP3-COBOL サンプルの登録作業処理

XMAP3-COBOL サンプルの会社情報登録処理を示します。



(5) 多様な開発環境に対応する C/S システムサンプルプログラムの構成

サンプルプログラムの構成を示します。

```
Sample ┬ CSS_Sample           : C/Sシステムサンプル
       ├ HiRDB_Sample_C       : C言語-HiRDB サンプル
       ├ HiRDB_Sample_COBOL   : COBOL-HiRDB サンプル
       ├ TpbSample_C          : TPBroker-C言語 サンプル
       └ XMAP3_Sample         : XMAP3-COBOL サンプル
```

付録 E　SEWB+/REPOSITORY で表示される名称

SEWB+/CONSTRUCTION の各定義ファイルが，SEWB+/REPOSITORY に格納された場合，表示される名称を次に示します。

表 E-1　SEWB+/REPOSITORY で表示される名称

表示される名称※	ドキュメント種別名	拡張子
SEWB+ プログラム定義	CONSTRUCTION_PROGRAMDOC	.csp
SEWB+ テンプレート	CONSTRUCTION_TEMPLATEDOC	.cst
SEWB+ ファイル定義	CONSTRUCTION_FILEDOC	.csd
SEWB+ RDB 定義	CONSTRUCTION_RDBDOC	
SEWB+ DAM 定義	CONSTRUCTION_DAMDOC	
SEWB+ TAM 定義	CONSTRUCTION_TAMDOC	
SEWB+ RPC 入力パラメタ定義	CONSTRUCTION_RPCINPUTPARMDOC	
SEWB+ RPC 応答領域定義	CONSTRUCTION_RPCREPLYPARMDOC	
SEWB+ メッセージ定義	CONSTRUCTION_MESSAGEDOC	
SEWB+ ユーザジャーナル定義	CONSTRUCTION_USERJOURNALDOC	
SEWB+ メッセージログ定義	CONSTRUCTION_MESSAGELOGDOC	
SEWB+ 共通作業領域定義	CONSTRUCTION_COMMONWORKAREADOC	
SEWB+ プログラム定義（XML 形式）	CONSTRUCTION_XML_PROGRAMDOC	.csq
SEWB+ ファイル定義（XML 形式）	CONSTRUCTION_XML_FILEDOC	.cse
SEWB+ RDB 定義（XML 形式）	CONSTRUCTION_XML_RDBDOC	
SEWB+ DAM 定義（XML 形式）	CONSTRUCTION_XML_DAMDOC	
SEWB+ TAM 定義（XML 形式）	CONSTRUCTION_XML_TAMDOC	
SEWB+ RPC 入力パラメタ定義（XML 形式）	CONSTRUCTION_XML_RPCINDOC	
SEWB+ RPC 応答領域定義（XML 形式）	CONSTRUCTION_XML_RPCREPLYDOC	
SEWB+ メッセージ定義（XML 形式）	CONSTRUCTION_XML_MESSAGEDOC	
SEWB+ ユーザジャーナル定義（XML 形式）	CONSTRUCTION_XML_USERJOURNALDOC	
SEWB+ メッセージログ定義（XML 形式）	CONSTRUCTION_XML_MESSAGELOGDOC	
SEWB+ 共通作業領域定義（XML 形式）	CONSTRUCTION_XML_COMMONWORKDOC	

表示される名称※	ドキュメント種別名	拡張子
SEWB+ 部品	CONSTRUCTION_PARTSDOC	.csr
SEWB+ レコード定義	CONSTRUCTION_RECORDDOC	.csc

注※

SEWB+/REPOSITORY-BROWSER のウィンドウに表示されます。

付録 F　SEWB+/CONSTRUCTION のインストールとアンインストール

SEWB+/CONSTRUCTION のインストールとアンインストールの方法について説明します。

（1）SEWB+/CONSTRUCTION のインストール

SEWB+/CONSTRUCTION は，SEWB+ 基本開発環境のインストールを実行することで，インストールされます。SEWB+ 基本開発環境のインストール手順などの詳細は，マニュアル「SEWB+/REPOSITORY 運用ガイド」を参照してください。

インストール後の環境設定については，「付録 A　環境設定」を参照してください。

（2）SEWB+/CONSTRUCTION のアンインストール

SEWB+/CONSTRUCTION は，SEWB+ 基本開発環境のアンインストールを実行することで，アンインストールされます。SEWB+ 基本開発環境のアンインストール手順などの詳細は，マニュアル「SEWB+/REPOSITORY 運用ガイド」を参照してください。

（3）SEWB+/CONSTRUCTION の環境設定情報の保持

SEWB+/CONSTRUCTION を再インストールする場合，環境設定情報を保持できます。

付録 G　前バージョンからの移行

前バージョンの SEWB+/CONSTRUCTION※から SEWB+ 基本開発環境の SEWB+/CONSTRUCTION にバージョンアップした場合，前バージョンの SEWB+/CONSTRUCTION 使用時にユーザが設定していた環境設定情報は，引き継がれます。

前バージョンの SEWB+/CONSTRUCTION から SEWB+ 基本開発環境の SEWB+/CONSTRUCTION にバージョンアップする手順などの詳細は，マニュアル「SEWB+/REPOSITORY 運用ガイド」を参照してください。

注※

前バージョンの SEWB+/CONSTRUCTION とは次に示す製品です。

- P-2651-8324　SEWB+/CONSTRUCTION　02-nn（nn：数字）
- P-2451-8F24　SEWB+/CONSTRUCTION Server　02-nn（nn：数字）

付録 H　XML 形式ファイルのフォーマット

SEWB+/CONSTRUCTION では，データ定義およびプログラム定義の情報を XML 形式ファイルに出力することができます。

出力されるファイルのファイル形式，および出力方法について説明します。

付録 H.1　XML 形式ファイルの出力方法

(1) データ定義情報の出力方法

(a) データ定義で出力する場合

1. **環境設定の「データ定義ファイルの格納形式」で，格納形式を「XML 形式」に設定する**
2. **XML 形式ファイルを出力するデータ定義ファイルを開く**
3. **データ定義ファイルを保存する**

(b) コマンドで出力する場合

データ定義ファイルのフォーマットを XML 形式のフォーマットに変更する場合，コマンド「CSDDCNVX.EXE」を実行させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されています。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

形 式

```
CSDDCNVX.EXE△変換元ファイル名
             { [△/o△変換先ファイル名] | [△/op△変換先ファイル出力先パス名] }
             [△/t△トレースファイル名]
             [△/help]
             [△/?]
```

（凡例）

△：1 文字以上の空白を示します。

注

［ ］は省略できることを示します。

{　} はどちらかを指定することを示します。

解 説

変換元ファイル名：

フォーマットを変換するプログラム定義ファイルのファイル名※1 を指定します。

/o △変換先ファイル名※2：

フォーマットが変換されたファイルを保存するパスの名称を指定します。

/op △変換先ファイル出力先パス名※2：

フォーマットが変換されたファイルを保存するパスの名称を指定します。ファイル名は，”変換元ファイル名（拡張子なし）”＋”.cse ”となります。

/t △トレースファイル名：

トレース情報の出力先ファイル名を指定します。

/help：

コマンドヘルプメッセージを表示します。

/? ：

コマンドヘルプメッセージを表示します。

注※ 1

ワイルドカードの指定ができます。ファイル名にワイルドカードを指定した場合，/o は指定できません。

注※ 2

/o または /op の指定がない場合，変換元ファイルと同一フォルダに ”変換元ファイル名（拡張子なし）”＋”.cse ”で作成されます。

(2) プログラム情報の出力方法

(a) プログラム定義で出力する場合

1. **環境設定の「プログラム定義ファイルの格納形式」で，格納形式を「XML 形式」に設定する**
2. **XML 形式ファイルを出力するプログラム定義ファイルを開く**
3. **プログラム定義を保存する**

(b) コマンドで出力する場合

プログラム定義ファイルのフォーマットを XML 形式のフォーマットに変更する場合，コマンド「CSPDCNVX.EXE」を実行させます。

コマンドは，次のディレクトリに格納されています。

SEWB+ 基本開発環境の組み込み先パス名¥Construction

SEWB+ 基本開発環境の組み込み先パス名は，デフォルトでは次のように設定されてい

ます。システムドライブとは，Windows が組み込まれているドライブです。

- Windows 2000，Windows XP，Windows Server 2003 または Windows Vista の場合
  システムドライブ :¥Program Files¥HITACHI¥Sewb
- Windows Server 2003 x64 の場合
  システムドライブ :¥Program Files(x86)¥HITACHI¥Sewb

形 式

```
CSPDCNVX.EXE△変換元ファイル名
          { [△/o△変換先ファイル名] | [△/op△ 変換先ファイル出力先パス名] }
          [△/t△トレースファイル名]
          [△/help]
          [△/?]
```

（凡例）

△：1 文字以上の空白を示します。

注

［ ］は省略できることを示します。
{ } はどちらかを指定することを示します。

解 説

変換元ファイル名：

フォーマットを変換するプログラム定義ファイルのファイル名※ 1 を指定します。

/o △変換先ファイル名※ 2：

フォーマットが変換されたファイルを保存するパスの名称を指定します。

/op △変換先ファイル出力先パス名※ 2：

フォーマットが変換されたファイルを保存するパスの名称を指定します。ファイル名は，” 変換元ファイル名（拡張子なし）” + ” .csq ” となります。

/t △トレースファイル名：

トレース情報の出力先ファイル名を指定します。

/help：

コマンドヘルプメッセージを表示します。

/? ：

コマンドヘルプメッセージを表示します。

注※ 1

ワイルドカードの指定ができます。ファイル名にワイルドカードを指定した場合，/o は指定できません。

注※ 2

/o または /op の指定がない場合，変換元ファイルと同一フォルダに ” 変換元ファイル名（拡張子なし）” + ” .csq ” で作成されます。

付録 H.2　データ定義情報の XML 形式ファイルへの出力

(1) ファイルフォーマット

データ定義を XML 形式ファイルに出力するときのファイルフォーマットを，次に示します。

表 H-1　データ定義ファイル（.cse）のファイルフォーマット

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
データ定義パッケージ：Construction					
データ定義ファイル情報	*File*	1	―		
データ定義情報	*Data*	1	―		
データ定義ファイル情報：File					
ID	ID	1	"P-2651-8324"	PP 型名	
種別	Type	1	"DATADEFINITION"	データ定義ファイルのファイル種別	
バージョン	Version	1	"01"	ファイルバージョン	
ステータス	Status	1	文字列	ファイルステータス	
データ定義情報：Data					
データ定義名称	Name	0 または 1	文字列	名称（[サイン] タブ）の内容	
作成者	Author	0 または 1	文字列	作成者（[サイン] タブ）の内容	
概要	Outline	0 または 1	文字列（改行を含む）	概要（[サイン] タブ）の内容	

タグ種					
含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
種別	Type	1	"FILE"：ファイル "RDB"：RDB "DAM"：DAM "TAM"：TAM "RPCIN"：RPC 入力パラメタ "RPCREPLY"：RPC 応答領域 "MESSAGE"：メッセージ "USERJOURNAL"：ユーザジャナル "MESSAGELOG"：メッセージログ "WORK"：ワーク	定義種別（定義種別選択ダイアログで選択）	
結合項目数	ItemNum	1	文字列	結合項目（レコード定義）の数	
レベル開始値	InitialLevel	1	文字列	生成時のレベル番号開始の値	
レベル増分値	IncrementLevel	1	文字列	生成時のレベル番号増分の値	
初期値生成	InitValue	1	"TRUE"：する "FALSE"：しない	生成時の初期値展開の値	
COBOL ソースパス	CobolSourcePath	1	文字列	生成先パス（生成ダイアログ オプションが COBOL）の値	
C ソースパス	CSourcePath	1	文字列	生成先パス（生成ダイアログ オプションが C）の値	
ファイル名	FileName	1	文字列	ファイル名（[ファイル] タブ）の内容	ファイル情報

タグ種					
含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
ファイル種別	FileType	1	"SEQUENTIAL"：順編成 "RELATIVE"：相対編成 "INDEX"：索引編成 "INDEXEDSEQUENTIAL"：索引順編成 "TEXT"：テキスト編成 "CSV"：CSV 編成	ファイル編成（[ファイル] タブ）の内容	ファイル情報
レコード形成	FlRecForm	1	"FIXED"：固定長 "VARIABLE"：可変長	レコード形式（[ファイル] タブ）の内容	
キー名	FlKeyName	1	文字列	キー（[ファイル] タブ）の内容	
キー名 OID	FlKeyObjectId	1	文字列	キー（[ファイル] タブ）の OID	
キー名日付	FlKeyUpdateTime	1	文字列	キー（[ファイル] タブ）の日付	
長さ設定エリア	AreaName	1	文字列		
EXTERNAL	External	1	文字列		
テーブル名	TableName	1	文字列	表名称（[RDB] タブ）の内容	RDB 情報
DAM ファイル名	DamFailName	1	文字列	ファイル名（[DAM] タブ）の内容	DAM 情報
プロックサイズ	BlockSize	1	文字列	プロックサイズ（[DAM] タブ）の内容	DAM 情報
TAM テーブル名	TamTableName	1	文字列	テーブル名称（[TAM] タブ）の内容	TAM 情報
キー項目名	KeyName	1	文字列	キー（[TAM] タブ）の内容	TAM 情報
キー項目 OID	KeyObjectId	1	文字列	キー（データ項目）の OID	TAM 情報
キー項目更新日付	KeyUpdateTime	1	文字列	テーブル名称（[TAM] タブ）の内容	TAM 情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
ユーザジャーナルコード	UserJournalCode	1	文字列	コード（[ユーザジャーナル] タブ）の内容	ユーザジャーナル情報
定義種別	DefinitionType	1	"0"：結合項目名称 "1"：レコード定義名称	結合項目名称，またはレコード定義名称の区分	
最上位結合項目名称	HighOrderKeyName	1	文字列	最上位結合項目（レコード定義）の名称	
最上位結合項目 OID	HighOrderKeyObjectId	1	文字列	最上位結合項目の OID	
最上位結合項目更新日時	HighOrderKeyUpdateTime	1	文字列	最上位結合項目（レコード定義）の更新日時	
パス使用有無	UsePath	1	"USE"：利用する "NOUSE"：利用しない		
レコードファイル名フルパス	RecFullName	1	文字列		
キー名使用有無	KeyUse	1	"1"：利用する "0"：利用しない		
生成キーワード情報	*GenerateKeyword*	1	―		
結合項目下位情報	*Record*	1	―	辞書（レコード定義）の情報	
生成キーワード情報：GenerateKeyword					
英数字項目	CobolX	1	文字列		辞書情報
数字編集項目	CobolZ	1	文字列		辞書情報
漢字項目	CobolN	1	文字列		辞書情報
符号なし外部 10 進項目	Cobol9	1	文字列		辞書情報
符号付き外部 10 進項目	CobolS	1	文字列		辞書情報
符号なし内部 10 進項目	CobolU	1	文字列		辞書情報

タグ種					
含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
符号付き内部 10 進項目	CobolP	1	文字列		辞書情報
符号なし 2 進項目	CobolBU	1	文字列		辞書情報
符号付き 2 進項目	CobolB	1	文字列		辞書情報
外部浮動小数点項目	CobolE	1	文字列		辞書情報
内部浮動小数点項目	CobolD	1	文字列		辞書情報
内部ブール項目	Cobol1	1	文字列		辞書情報
外部ブール項目	Cobol8	1	文字列		辞書情報
時刻項目	CobolXT	1	文字列		辞書情報
日付項目	CobolXD	1	文字列		辞書情報
アドレスデータ項目	CobolT	1	文字列		辞書情報
フリー定義	CobolFree	1	文字列		辞書情報
文字型	Cchar	1	文字列		辞書情報
文字型 (ポインタ)	Cpchar	1	文字列		辞書情報
符号付き短整数型	Cshort	1	文字列		辞書情報
符号なし短整数型	Cushort	1	文字列		辞書情報
符号付き整数型	Cint	1	文字列		辞書情報
符号なし整数型	Cuint	1	文字列		辞書情報
符号付き長整数型	Clong	1	文字列		辞書情報
符号なし長整数型	Culong	1	文字列		辞書情報
単精度浮動小数点型	Cfloat	1	文字列		辞書情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
倍精度浮動小数点型	Cdouble	1	文字列		辞書情報
拡張精度浮動小数点型	ClongDouble	1	文字列		辞書情報
時刻型	Cchart	1	文字列		辞書情報
日付型	Cchard	1	文字列		辞書情報
フリー定義	Cfree	1	文字列		辞書情報
ユーザ定義型	UserCustomType	0 以上	文字列		辞書情報
結合項目下位情報：Record ※					
結合項目 OID	ObjectId	1	文字列		辞書情報
データ項目数	DataItemNum	1	文字列		辞書情報
データ項目情報	*DataItem*	1	―		
データ項目情報：DataItem ※					
データ項目名	Nname	1	文字列		辞書情報
データ項目 OID	ObjectId	1	文字列		辞書情報
データ項目種別	Kind	1	文字列		辞書情報
データ項目レベル番号	Level	1	文字列		辞書情報
標準名称	StandardName	4	文字列	標準名称	辞書情報
フリガナ	Kana	4	文字列		辞書情報
コメント	Comment	4	文字列		辞書情報
フィールド	Field	20	文字列		辞書情報
分類	TypeGroup	1	文字列		辞書情報
分類文字数	TypeGroupString	1	文字列		辞書情報

タグ種					
含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
けた数	Total	1	文字列		辞書情報
小数部けた数	DecimalPart	1	文字列		辞書情報
反復回数	Repeat	1	文字列		辞書情報
辞書参照状態	DictStatus	1	文字列		
データ項目更新日付	Date	1	文字列		
辞書情報	*Dict*	12	―	辞書（レコード定義）の情報	
生成キーワード	*Keyword*	2	―		
レコード属性	*RecordAttribute*	2	―		
辞書情報：Dict ※					
言語区分	Lang	0 または 1	文字列		辞書情報
データ項目名	Name	0 または 1	文字列		辞書情報
タイプ	Type	0 または 1	文字列		辞書情報
タイプ文字列	TypeString	0 または 1	文字列		辞書情報
フリー定義文字列	FreeString	0 または 1	文字列		辞書情報
編集文字列	EditString	0 または 1	文字列		辞書情報
初期値	InitValue	0 または 1	文字列		辞書情報
フィールド	Field	0 または 1	文字列		辞書情報
生成キーワード：Keyword ※					

タグ種					
含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
@DATAITEM	DataItem	0 または 1	文字列		辞書情報
@LEN	Len	0 または 1	文字列		辞書情報
@PLEN	Plen	0 または 1	文字列		辞書情報
@ILEN	Ilen	0 または 1	文字列		辞書情報
@DLEN	Dlen	0 または 1	文字列		辞書情報
@VALUE	Value	0 または 1	文字列		辞書情報
@OCCURS	Occurs	0 または 1	文字列		辞書情報
@EDITCAHR	Editchar	0 または 1	文字列		辞書情報
@COMP	Comp	0 または 1	文字列		辞書情報
@FREE	Free	0 または 1	文字列		辞書情報
レコード属性：RecordAttribute					
CHARCTER TYPE	CharType	0 または 1	文字列		
CHARCTER TYPE 値	CharTypeVal	0 または 1	文字列		
指標名	Index	0 または 1	文字列		

タグ種					
含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
可変反復回数項目名	ValOccursName	0 または 1	文字列		
可変反復回数最小値	ValOccursMini	0 または 1	文字列		
アドレス名	Address	0 または 1	文字列		
再定義名	Redefine	0 または 1	文字列		
EXTERNAL 句	External	0 または 1	文字列		
SYNC 句	Sync	0 または 1	文字列		
JUST 句	Just	0 または 1	文字列		
BRANK ZERO 句	BrankZero	0 または 1	文字列		

注 1
斜体で表記している要素型名の設定可能値については，対応するタグ種を参照してください。
注 2
属性値に含まれる「<」「&」「"」，および内容（文字データ）に含まれる「<」「&」は，それぞれ次のように置換します。

- <：<
- &：&
- "："

注※
備考に「辞書情報」と記載しているタグ情報でも，レコード定義ファイルを使用している場合には，レコード定義の内容によってレコード定義情報が設定されます。

（2）DTD（Document Type Definiton）

データ定義ファイルの DTD を次に示します。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE XMI [
```

```
<!ELEMENT XMI.content (Construction.File,Construction.Data)>
<!ELEMENT Construction.File
(Construction.File.ID,Construction.File.Type,Construction.File.Ver
sion,Construction.File.Status)>
<!ELEMENT Construction.File.ID (#PCDATA)>
<!ELEMENT Construction.File.Type (#PCDATA)>
<!ELEMENT Construction.File.Version (#PCDATA)>
<!ELEMENT Construction.File.Status (#PCDATA)>
<!ELEMENT Construction.Data
(Construction.Data.Name?,Construction.Data.Author?,

Construction.Data.Outline?,Construction.Data.Type,

Construction.Data.ItemNum,Construction.Data.InitialLevel,

Construction.Data.IncrementLevel,Construction.Data.InitValue,

Construction.Data.CobolSourcePath,Construction.Data.CSourcePath,

Construction.Data.FileName,Construction.Data.FileType,

Construction.Data.FlKeyName,Construction.Data.FlKeyObjectId,

Construction.Data.FlKeyUpdateTime,Construction.Data.AreaName,

Construction.Data.External,Construction.Data.TableName,

Construction.Data.DamFileName,Construction.Data.BlockSize,

Construction.Data.TamTableName,Construction.Data.KeyName,

Construction.Data.KeyObjectID,Construction.Data.KeyUpdateTime,

Construction.Data.UserJournalCode,Construction.Data.HighOrderKeyNa
me*,

Construction.Data.HighOrderKeyObjectID*,Construction.Data.HighOrde
rKeyUpdateTime*,

Construction.Data.generatekeyword,Construction.Data.record*)>
<!ELEMENT Construction.Data.Name (#PCDATA)>
<!ELEMENT Construction.Data.Author (#PCDATA)>
<!ELEMENT Construction.Data.Outline (#PCDATA)>
<!ELEMENT Construction.Data.Type (#PCDATA)>
<!ELEMENT Construction.Data.ItemNum (#PCDATA)>
<!ELEMENT Construction.Data.InitialLevel (#PCDATA)>
<!ELEMENT Construction.Data.IncrementLevel (#PCDATA)>
<!ELEMENT Construction.Data.InitValue (#PCDATA)>
<!ELEMENT Construction.Data.CobolSourcePath (#PCDATA)>
<!ELEMENT Construction.Data.CSourcePath (#PCDATA)>
<!ELEMENT Construction.Data.FileName (#PCDATA)>
<!ELEMENT Construction.Data.FileType (#PCDATA)>
<!ELEMENT Construction.Data.TableName (#PCDATA)>
<!ELEMENT Construction.Data.DamFileName (#PCDATA)>
<!ELEMENT Construction.Data.BlockSize (#PCDATA)>
<!ELEMENT Construction.Data.TamTableName (#PCDATA)>
<!ELEMENT Construction.Data.KeyName (#PCDATA)>
<!ELEMENT Construction.Data.KeyObjectID (#PCDATA)>
<!ELEMENT Construction.Data.KeyUpdateTime (#PCDATA)>
<!ELEMENT Construction.Data.UserJournalCode (#PCDATA)>
<!ELEMENT Construction.Data.DefinitionType (#PCDATA)>
<!ELEMENT Construction.Data.HighOrderKeyName (#PCDATA)>
<!ELEMENT Construction.Data.HighOrderKeyObjectID (#PCDATA)>
<!ELEMENT Construction.Data.HighOrderKeyUpdateTime (#PCDATA)>
<!ELEMENT Construction.Data.UsePath (#PCDATA)>
```

```
<!ELEMENT Construction.Data.RecFlName (#PCDATA)>
<!ELEMENT Construction.Data.KeyUse (#PCDATA)>
<!ELEMENT Construction.Data.generatekeyword
(Construction.GenerateKeyword)>
<!ELEMENT Construction.GenerateKeyword
(Construction.GenerateKeyword.CobolX,Construction.GenerateKeyword.
CobolZ,

Construction.GenerateKeyword.CobolN,Construction.GenerateKeyword.C
obol9,

Construction.GenerateKeyword.CobolS,Construction.GenerateKeyword.C
obolU,

Construction.GenerateKeyword.CobolP,Construction.GenerateKeyword.C
obolBU,

Construction.GenerateKeyword.CobolB,Construction.GenerateKeyword.C
obolE,

Construction.GenerateKeyword.CobolD,Construction.GenerateKeyword.C
obol1,

Construction.GenerateKeyword.Cobol8,Construction.GenerateKeyword.C
obolXT,

Construction.GenerateKeyword.CobolXD,Construction.GenerateKeyword.
CobolT,

Construction.GenerateKeyword.CobolFree,Construction.GenerateKeywor
d.CChar,

Construction.GenerateKeyword.CPChar,Construction.GenerateKeyword.C
Short,

Construction.GenerateKeyword.CUShort,Construction.GenerateKeyword.
CInt,

Construction.GenerateKeyword.CUInt,Construction.GenerateKeyword.CL
ong,

Construction.GenerateKeyword.CULong,Construction.GenerateKeyword.C
Float,

Construction.GenerateKeyword.CDouble,Construction.GenerateKeyword.
CLongDouble,

Construction.GenerateKeyword.CCharT,Construction.GenerateKeyword.C
CharD,

Construction.GenerateKeyword.CFree,Construction.GenerateKeyword.Us
erCustomType*)>
<!ELEMENT Construction.GenerateKeyword.CobolX (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolZ (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolN (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.Cobol9 (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolS (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolU (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolP (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolBU (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolB (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolE (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolD (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.Cobol1 (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.Cobol8 (#PCDATA)>
```

```
<!ELEMENT Construction.GenerateKeyword.CobolXT (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolXD (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolT (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CobolFree (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CChar (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CPChar (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CShort (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CUShort (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CInt (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CUInt (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CLong (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CULong (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CFloat (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CDouble (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CLongDouble (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CCharT (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CCharD (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.CFree (#PCDATA)>
<!ELEMENT Construction.GenerateKeyword.UserCustomType (#PCDATA)>
<!ELEMENT Construction.Data.record (Construction.Record)>
<!ELEMENT Construction.Record
(Construction.Record.ObjectID,Construction.Record.DataItemNum,Cons
truction.Record.dataitem)>
<!ELEMENT Construction.Record.ObjectID (#PCDATA)>
<!ELEMENT Construction.Record.DataItemNum (#PCDATA)>
<!ELEMENT Construction.Record.dataitem (Construction.DataItem+)>
<!ELEMENT Construction.DataItem
(Construction.DataItem.Name,Construction.DataItem.ObjectID,

Construction.DataItem.Kind,Construction.DataItem.Level,

Construction.DataItem.StandardName+,Construction.DataItem.Kana+,

Construction.DataItem.Comment+,Construction.DataItem.Field+,

Construction.DataItem.TypeGroup,Construction.DataItem.TypeGroupStr
ing,

Construction.DataItem.Total,Construction.DataItem.DecimalPart,

Construction.DataItem.Repeat,Construction.DataItem.RepeatSet,

Construction.DataItem.dictstate,Construction.DataItem.date,

Construction.DataItem.dict,Construction.DataItem.keyword,

Construction.DataItem.recordattribute)>
<!ELEMENT Construction.DataItem.Name (#PCDATA)>
<!ELEMENT Construction.DataItem.ObjectID (#PCDATA)>
<!ELEMENT Construction.DataItem.Kind (#PCDATA)>
<!ELEMENT Construction.DataItem.Level (#PCDATA)>
<!ELEMENT Construction.DataItem.StandardName (#PCDATA)>
<!ELEMENT Construction.DataItem.Kana (#PCDATA)>
<!ELEMENT Construction.DataItem.Comment (#PCDATA)>
<!ELEMENT Construction.DataItem.Field (#PCDATA)>
<!ELEMENT Construction.DataItem.TypeGroup (#PCDATA)>
<!ELEMENT Construction.DataItem.TypeGroupString (#PCDATA)>
<!ELEMENT Construction.DataItem.Total (#PCDATA)>
<!ELEMENT Construction.DataItem.DecimalPart (#PCDATA)>
<!ELEMENT Construction.DataItem.Repeat (#PCDATA)>
<!ELEMENT Construction.DataItem.RepeatSet (#PCDATA)>
<!ELEMENT Construction.DataItem.DictState (#PCDATA)>
<!ELEMENT Construction.DataItem.Date (#PCDATA)>
<!ELEMENT Construction.DataItem.dict (Construction.Dict)>
<!ELEMENT Construction.Dict
```

```
(Construction.Dict.Lang,Construction.Dict.Name?,

Construction.Dict.Type?,Construction.Dict.TypeString?,

Construction.Dict.FreeString?,Construction.Dict.EditString?,

Construction.Dict.InitValue?,Construction.Dict.Field?)*>
<!ELEMENT Construction.Dict.Lang (#PCDATA)>
<!ELEMENT Construction.Dict.Name (#PCDATA)>
<!ELEMENT Construction.Dict.Type (#PCDATA)>
<!ELEMENT Construction.Dict.TypeString (#PCDATA)>
<!ELEMENT Construction.Dict.FreeString (#PCDATA)>
<!ELEMENT Construction.Dict.EditString (#PCDATA)>
<!ELEMENT Construction.Dict.InitValue (#PCDATA)>
<!ELEMENT Construction.Dict.Field (#PCDATA)>
<!ELEMENT Construction.DataItem.keyword (Construction.Keyword)>
<!ELEMENT Construction.Keyword (Construction.Keyword.DataItem,
                                Construction.Keyword.Len?,
                                Construction.Keyword.PLen?,
                                Construction.Keyword.ILen?,
                                Construction.Keyword.DLen?,
                                Construction.Keyword.Value?,
                                Construction.Keyword.Occurs?,
                                Construction.Keyword.Editchar?,
                                Construction.Keyword.Comp?,
                                Construction.Keyword.Free?)*>
<!ELEMENT Construction.DataItem.recordattribute
(Construction.RecordAttribute)>
<!ELEMENT Construction.RecordAttribute
(Construction.RecordAttribute.CharType?,
                        Construction.RecordAttribute.CharTypeVal?,
                              Construction.RecordAttribute.Index?,

Construction.RecordAttribute.ValOccursName?,

Construction.RecordAttribute.ValOccursMini?,
                             Construction.RecordAttribute.Address?,
                            Construction.RecordAttribute.Redefine?,
                             Construction.RecordAttribute.External,
                                 Construction.RecordAttribute.Sync,
                                 Construction.RecordAttribute.Just,
                           Construction.RecordAttribute.BrankZero)*>
<!ELEMENT Construction.Keyword.DataItem (#PCDATA)>
<!ELEMENT Construction.Keyword.Len (#PCDATA)>
<!ELEMENT Construction.Keyword.PLen (#PCDATA)>
<!ELEMENT Construction.Keyword.ILen (#PCDATA)>
<!ELEMENT Construction.Keyword.DLen (#PCDATA)>
<!ELEMENT Construction.Keyword.Value (#PCDATA)>
<!ELEMENT Construction.Keyword.Occurs (#PCDATA)>
<!ELEMENT Construction.Keyword.Editchar (#PCDATA)>
<!ELEMENT Construction.Keyword.Comp (#PCDATA)>
<!ELEMENT Construction.Keyword.Free (#PCDATA)>
]>
```

付録 H.3　プログラム定義情報の XML 形式ファイルへの出力

(1) ファイルフォーマット

プログラム定義を XML 形式ファイルに出力するときのファイルフォーマットを，次に示します。

表 H-2　プログラム定義ファイル（.csq）のファイルフォーマット

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
プログラム定義パッケージ：Construction					
プログラム定義ファイル情報	*File*	1	―		
プログラム定義情報	*Program*	1	―		
共通可変記号情報	*CommonValiableSymbol*	1	―		
共通ユーザ追加処理情報	*CommonUOC*	1	―		
プログラム定義ファイル情報：File					
ID	ID	1	"P-2651-8324"	PP 型名	
種別	Type	1	"PROGRAMDEFINITION"	プログラム定義ファイルのファイル種別	
バージョン	Version	1	"02"	ファイルバージョン	
ステータス	Status	1	文字列	ファイルステータス	
プログラム定義情報：Program					
名称	Name	1	文字列	名称（[サイン] タブ）の内容	
作成者	Author	1	文字列	作成者（[サイン] タブ）の内容	
概要	Outline	1	文字列（改行を含む）	概要（[サイン] タブ）の内容	
プログラミング言語	ProgramLang	1	文字列	言語種別（@@lang 文）の値	テンプレート情報
メモ	Memo	1	文字列（改行を含む）	メモ（[メモ] タブ）の内容	
テンプレートファイル情報	*TemplateFile*	1	―	テンプレートファイル情報	
リソースファイル情報	*ResourceFile*	0 以上	―	リソースファイル情報	
業務ルール情報	*Rule*	0 以上	―	展開業務ルール情報	
テンプレートファイル情報：TemplateFile					

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
ファイル名	Name	1	文字列	テンプレートファイルのファイル名	
更新日付	UpdateTime	1	文字列："YYYY/MM/DD" 形式	テンプレートファイルの更新日付	
パス参照フラグ	PathUse	1	"USE"：参照あり "NOUSE"：参照なし	環境設定パス（テンプレートパス）の参照有無	
リソースファイル情報：ResourceFile					
ファイル名	Name	1	文字列	リソースファイルのファイル名	
属性	Kind	1	"DATADEFINITION"：データ定義 "PARTS"：部品 "USERPARTS"：追加部品 "LOGICDESIGN"：論理設計図 "XMAP3"：マップ定義 "XML"：XML ファイル "GENERATE"：生成ファイル	リソースファイルのファイル種別	
ファイルパス	Path	1	文字列	リソースファイルのファイルパス	
更新日付	UpdateTime	1	文字列："YYYY/MM/DD" 形式	リソースファイルの更新日付	
パス参照フラグ	PathUse	1	"USE"：参照あり "NOUSE"：参照なし	環境設定パス（リソースパス）の参照有無	
更新カウンタ	ReferenceCount	1	文字列	リソースファイルの参照数	
業務ルール情報：Rule					
業務ルール名	Name	1	文字列	業務ルール名	
OID	ObjectID	1	文字列	業務ルール OID	

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
更新日付	UpdateTime	1	文字列："YYYY/MM/DD" 形式	業務ルール更新日付	
更新カウンタ	ReferenceCount	1	文字列	業務ルールの参照数	
共通可変記号情報：CommonValiableSymbol					
プログラミング言語	ProgramLang	1	文字列	言語種別（@@lang 文）の値	テンプレート情報
プログラミング言語の文字列区分	LangStringType	1	"SYSTEM"：システム "USER"：ユーザ	言語種別（@@lang 文）の区分	テンプレート情報
言語種別	LangType	1	"ABSOLUTENESS"：絶対 "NAME"：名称	言語種別（@@lang 文）の種別	テンプレート情報
リポジトリの言語区分	RepositoryLang	1	文字列	FOR_REPOSITORY（@@lang 文）の値	テンプレート情報
拡張子	Extension	1	文字列	EXTENSION（@@lang 文）の値	テンプレート情報
連結文字	ModifyConnect	1	文字列	MODIFY_CONNECT（@@lang 文）の値	テンプレート情報
修飾順序	ModifyOrder	1	"ASCEND"：昇順 "DESCEND"：降順	MODIFY_ORDER（@@lang 文）の値	テンプレート情報
可変記号先頭文字	TopChar	1	文字	可変記号先頭文字（ExpandChar）の値	レジストリ情報
可変記号情報	*ValiableSymbol*	0 以上			
可変記号情報：ValiableSymbol					
可変記号名	Name	1	文字列	可変記号名（@@interface 文）の値	テンプレート情報

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
属性	Kind	1	"STANDARD"：標準 "DATA"：データ定義 "DATAITEM"：データ項目 "PARTS"：部品 "IDL"：IDL "INTERFACE"：インタフェース "OPERATION"：オペレーション "XMAP3"：マップ定義 "XML"：XML "XMLELEMENT"：XMLELEMENT	可変記号の属性 ATTR（@@interface 文）の値	テンプレート情報
説明	Comment	1	文字列（改行を含む）	可変記号の説明 COMMENT（@@interface 文）の値	テンプレート情報
配列最大数	ArrayMax	1	文字列	可変記号の配列最大数 ARRAY_MAX（@@interface 文）の値	テンプレート情報
標準可変記号情報	*StandardValiableSymbol*	0 または 1	－	標準可変記号の場合	
標準可変記号補足情報	*StandardValiableSymbolAuxiliary*	0 または 1	－	標準可変記号の場合	
データ定義可変記号情報	*DataValiableSymbol*	0 または 1	－	データ定義可変記号（ATTR= データ定義種別）の場合	
データ定義可変記号補足情報	*DataValiableSymbolAuxiliary*	0 または 1	－	データ定義可変記号（ATTR= データ定義種別）の場合	
データ項目可変記号情報	*DataItemValiableSymbol*	0 または 1	－	データ項目可変記号（ATTR=ITEM）の場合	

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
データ項目可変記号補足情報	*DataItemValiableSymbolAuxiliary*	0 または 1	―	データ項目可変記号（ATTR=ITEM）の場合	
部品可変記号情報	*PartsValiableSymbol*	0 または 1	―	部品可変記号（ATTR=PARTS）の場合	
インタフェース可変記号情報	*InterfaceValiableSymbol*	0 または 1	―	インタフェース可変記号（ATTR=IDL/INTERFACE/OPERATION）の場合	
インタフェース可変記号補足情報	*InterfaceValiableSymbolAuxiliary*	0 または 1	―	インタフェース可変記号（ATTR=IDL/INTERFACE/OPERATION）の場合	
マップ定義可変記号情報	*XMAP3ValiableSymbol*	0 または 1	―	マップ定義可変記号（ATTR=XMAP3）の場合	
マップ定義可変記号補足情報	*XMAP3ValiableSymbolAuxiliary*	0 または 1	―	マップ定義可変記号（ATTR=XMAP3）の場合	
XML 可変記号情報	*XMLValiableSymbol*	0 または 1	―	XML 可変記号（ATTR=XML）の場合	
XML 可変記号補足情報	*XMLValiableSymbolAuxiliary*	0 または 1	―	XML 可変記号（ATTR=XML）の場合	
XMLElement 可変記号情報	*XMLElementValiableSymbol*	0 または 1	―	XMLElement 可変記号（ATTR=ELEMENT）の場合	
XMLElement 可変記号補足情報	*XMLElementValiableSymbolAuxiliary*	0 または 1	―	XMLElement 可変記号（ATTR=ELEMENT）の場合	
標準可変記号情報：StandardValiableSymbol					
値	Value	1	文字列	可変記号の値	

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
子可変記号情報	*ValiableSymbol*	1			可変記号情報
標準可変記号補足情報：StandardValiableSymbolAuxiliary					
値	Value	1	文字列	VALUE（@@interface 文）の値	テンプレート情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
データ定義可変記号情報：DataValiableSymbol					
値	Value	1	文字列	可変記号の値（データ定義ファイル名）	
データ定義ファイル情報	*ResourceFile*	1	―		リソースファイル情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
データ定義可変記号補足情報：DataValiableSymbolAuxiliary					
種別	Kind	1	"FILE"：ファイル "DB"：DB "TAM"：TAM "DAM"：DAM "RPCINPUTPARM"：RPC 入力パラメタ "RPCREPLYPARM"：RPC 応答領域 "MESSAGE"：メッセージ "USERJOURNAL"：ユーザジャーナル "MESSAGELOG"：メッセージログ "WORK"：共通作業領域	データ定義種別 ATTR（@@interface 文）の値	テンプレート情報

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
入出力種別	IOKind	1	"NONE"：なし "IN"：入力 "OUT"：出力 "INOUT"：入出力	データ定義の入出力の区分 IO（@@interface 文）の値	テンプレート情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
データ項目可変記号情報：DataItemValiableSymbol ※					
値	Value	1	文字列	可変記号の値（データ項目名）	
標準名称	StandardName	4	文字列	標準名称	辞書情報
上位データ項目	UpperDataItem	1	文字列	上位データ項目名	
データ定義可変記号名	ValiableSymbolName	1	文字列	データ項目を選択したデータ定義可変記号	
データ定義可変記号添字	Contarray	1	文字列	データ項目を選択したデータ定義可変記号の配列添字	
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
データ項目可変記号補足情報：DataItemValiableSymbolAuxiliary					
参照区分	ReferenceKind	1	"OWN"：自データ定義可変記号 "VALIABLESYMBOL"：指定データ定義可変記号 "ALL"：全データ定義可変記号	データ定義可変記号の参照区分 REF（@@interface 文）の値	テンプレート情報
参照先可変記号	ReferenceValiableSymbol	1	文字列	参照データ定義可変記号 参照区分が "ALL" の場合	
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
部品可変記号情報：PartsValiableSymbol					

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
部品ファイル情報	*ResourceFile*	1	―		リソースファイル情報
プロシジャ情報	*Procedure*	1	―		
プロシジャ情報：Procedure					
プロシジャ名	Name	1	文字列	プロシジャの名称（@@proc 文）の値	テンプレート情報
コメント	Comment	1	文字列	プロシジャの説明（@@proc 文）の値	テンプレート情報
引数情報	*Parameter*	1	―		
引数情報：Parameter					
引数名	Name	1	文字列	引数の名称（@@proc 文）の値	テンプレート情報
コメント	Comment	1	文字列	引数の説明（@@proc 文）の値	テンプレート情報
タイプ	Type	1	"DIRECT"：直接 "REFERENCE"：参照	引数の値の参照区分（引数値設定ダイアログ）の値	
値	Value	1	文字列	引数の値 タイプが "DIRECT" の場合	
参照可変記号名	ValiableSymbolName	1	文字列	引数の値を参照する可変記号名 タイプが "REFERENCE" の場合	
参照可変記号情報	*ValiableSymbol*	1	―	引数の値を参照する可変記号の情報 タイプが "REFERENCE" の場合	可変記号情報
参照可変記号添字	Contarray	1	文字列	引数の値を参照する可変記号名の配列添字 タイプが "REFERENCE" の場合	
インタフェース可変記号情報：InterfaceValiableSymbol					
値	Value	1	文字列	可変記号の値（論理設計図ファイル名）	

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
論理設計図ファイル情報	*ResourceFile*	1	―		リソースファイル情報
IDL ファイル名	IDLFileName	1	文字列	選択 IDL ファイルの名称（インタフェース項目設定ダイアログ）	
インタフェース名	InterFaceName	1	文字列	選択インタフェースの名称（インタフェース項目設定ダイアログ）	
オペレーション名	OperationName	1	文字列	選択オペレーションの名称（インタフェース項目設定ダイアログ）	
IDL ファイルオブジェクト情報	IDLFileObject	1	―	選択 IDL ファイルの情報	
インタフェース定義オブジェクト情報	*InterfaceObject*	1	―	選択インタフェースの情報	
オペレーション情報	*Operation*	1	―	選択オペレーションの情報	
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
IDL ファイルオブジェクト情報：IDLFileObject					
インタフェースオブジェクト情報	InterfaceObject	0 以上	―	選択インタフェースの情報	
インタフェース定義オブジェクト情報：InterfaceObject					
IDL ファイル名	IDLFileName	1	文字列	IDL ファイルのファイル名	論理設計図ファイル情報
モジュール情報	*Module*	1	―		
インタフェース情報	*Interface*	1	―		
属性情報	*Attribute*	1	―		
オペレーション情報	*Operation*	1	―		
モジュール情報：Module					
モジュール名	Name	1	文字列	モジュールの名称	論理設計図ファイル情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
日本語名	AliasName	1	文字列	モジュールの日本語名	論理設計図ファイル情報
コメント	Comment	1	文字列	モジュールのコメント	論理設計図ファイル情報
修飾名	ScopeName	1	文字列	モジュールの修飾名	論理設計図ファイル情報
インタフェース情報：Interface					
インタフェース名	Name	1	文字列	インタフェースの名称	論理設計図ファイル情報
日本語名	AliasName	1	文字列	インタフェースの日本語名	論理設計図ファイル情報
コメント	Comment	1	文字列	インタフェースのコメント	論理設計図ファイル情報
オブジェクト名	ObjectName	1	文字列	オブジェクトの名称	論理設計図ファイル情報
オブジェクト日本語名	ObjectAliasName	1	文字列	オブジェクトの日本語名	論理設計図ファイル情報
修飾名	ScopeName	1	文字列	オブジェクトの修飾名	論理設計図ファイル情報
ベースインタフェース名	BaseInterfaceName	1	文字列	ベースインタフェースの名称	論理設計図ファイル情報
属性情報：Attribute					
属性名	Name	1	文字列	属性の名称	論理設計図ファイル情報
readonly 属性	Readonly	1	"READONLY" "NOTREADONLY"	readonly 属性	論理設計図ファイル情報
型名	TypeName	1	文字列	属性の型名	論理設計図ファイル情報

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
ユーザ定義型	UserType	1	文字列	属性の型名（ユーザ定義型の場合）	論理設計図ファイル情報
修飾名	ScopeName	1	文字列	属性の修飾名	論理設計図ファイル情報
データ項目 OID	ObjectID	1	文字列	属性の型のデータ項目 OID（ユーザ定義型の場合）	論理設計図ファイル情報
インタフェースデータ項目情報	*InterfaceDataItem*				
インタフェースデータ項目情報：InterfaceDataItem					
データ項目 OID	ObjectID	1	文字列		
更新日付	UpdateTime	1	文字列		
種別			Type SIMPLE COMPOSITE		
レコード情報	*Record*	0 以上	文字列		
データ項目情報	*DataItem*	0 以上	文字列		
レコード情報：Record					
結合項目 OID	ObjectID	1	文字列		辞書情報
データ項目数	DataItemNum	1	文字列		辞書情報
データ項目情報	*DataItem*	1	―		
データ項目情報：DataItem					
データ項目名	Nname	1	文字列		辞書情報
データ項目 OID	ObjectID	1	文字列		辞書情報
データ項目種別	Kind	1	文字列		辞書情報
データ項目レベル番号	Level	1	文字列		辞書情報
標準名称	StandardName	4	文字列		辞書情報
フリガナ	Kana	4	文字列		辞書情報
コメント	Comment	4	文字列		辞書情報
フィールド	Field	20	文字列		辞書情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
分類	TypeGroup	1	文字列		辞書情報
分類文字数	TypeGroupString	1	文字列		辞書情報
けた数	Total	1	文字列		辞書情報
小数部けた数	DecimalPart	1	文字列		辞書情報
反復回数	Repeat	1	文字列		辞書情報
辞書情報	*Dict*	1	−		
生成キーワード	*Keyword*	1	−		
辞書情報：Dict					
言語区分	Lang	1	文字列		辞書情報
データ項目名	Name	1	文字列		辞書情報
タイプ	Type	1	文字列		辞書情報
タイプ文字列	TypeString	1	文字列		辞書情報
フリー定義文字列	FreeString	1	文字列		辞書情報
編集文字列	EditString	1	文字列		辞書情報
初期値	InitValue	1	文字列		辞書情報
フィールド	Field	1	文字列		辞書情報
生成キーワード：Keyword					
@DATAITEM	DataItem	1	文字列		辞書情報
@LEN	Len	1	文字列		辞書情報
@PLEN	Plen	1	文字列		辞書情報
@ILEN	Ilen	1	文字列		辞書情報
@DLEN	Dlen	1	文字列		辞書情報
@VALUE	Value	1	文字列		辞書情報
@OCCURS	Occurs	1	文字列		辞書情報
@EDITCHAR	Editchar	1	文字列		辞書情報
@COMP	Comp	1	文字列		辞書情報
@FREE	Free	1	文字列		辞書情報
オペレーション情報：Operation					
オペレーション名	Name	1	文字列	オペレーションの名称	論理設計図ファイル情報

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
日本語名	AliasName	1	文字列	オペレーションの日本語名	論理設計図ファイル情報
コメント	Comment	1	文字列	オペレーションのコメント	論理設計図ファイル情報
oneway 属性	Oneway	1	"ONEWAY" "NOTONEWAY"	オペレーションの oneway 属性	論理設計図ファイル情報
戻り値型名	ReturnTypeName	1	文字列	戻り値の型名	論理設計図ファイル情報
戻り値ユーザ定義型	ReturnUserType	1	文字列	戻り値の型名（ユーザ定義型の場合）	論理設計図ファイル情報
修飾名	ScopeName	1	文字列	戻り値の修飾名	論理設計図ファイル情報
データ項目 OID	ObjectID	1	文字列	戻り値の型のデータ項目 OID	論理設計図ファイル情報
引数情報	*Argument*	1	―		
例外情報	*Exception*	1	―		
リクエストコンテキスト情報	*RequestContext*	1	―		
引数情報：Argument					
引数名	Name	1	文字列	引数の名称	論理設計図ファイル情報
モード	Mode	1	"IN"：入力 "OUT"：出力 "INOUT"：入出力	引数の入出力区分	論理設計図ファイル情報
型名	TypeName	1	文字列	引数の型名	論理設計図ファイル情報
ユーザ定義型	UserType	1	文字列	引数の型名（ユーザ定義型の場合）	論理設計図ファイル情報
修飾名	ScopeName	1	文字列	引数の修飾名	論理設計図ファイル情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
データ項目 OID	ObjectID	1	文字列	引数の型のデータ項目 OID	論理設計図ファイル情報
インタフェースデータ項目情報	*InterfaceDataItem*	0 以上			
例外情報：Exception					
例外名	Name	1	文字列	例外の名称	論理設計図ファイル情報
ユーザ定義型	UserType	1	文字列	例外の型名（ユーザ定義型の場合）	論理設計図ファイル情報
修飾名	ScopeName	1	文字列	例外の修飾名	論理設計図ファイル情報
データ項目 OID	ObjectID	1	文字列	例外の型のデータ項目 OID	論理設計図ファイル情報
インタフェースデータ項目情報	*InterfaceDataItem*	0 以上			
リクエストコンテキスト情報：RequestContext					
プロパティ名	Name	1	文字列	プロパティの名称	論理設計図ファイル情報
コメント	Comment	1	文字列	リクエストコンテキストのコメント	論理設計図ファイル情報
インタフェース可変記号補足情報：InterfaceValiableSymbolAuxiliary					
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
マップ定義可変記号情報：XMAP3ValiableSymbol					
値	Value	1	文字列	可変記号の値（マップ定義ファイル名）	
マップ定義ファイル情報	*ResourceFile*	1	―		リソースファイル情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
マップ定義可変記号補足情報：XMAP3ValiableSymbolAuxiliary					

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
入出力種別	IOKind	1	"NONE"：なし "IN"：入力 "OUT"：出力 "INOUT"：入出力	マップ定義ファイルの入出力の区分（@@interface 文）の値	テンプレート情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
XML **可変記号情報：**XMLValiableSymbol					
値	Value	1	文字列	可変記号の値（XML ファイル名）	
XML ファイル情報	*ResourceFile*	1	―		リソースファイル情報
パースレベル	ParseLevel	1	文字列	パースレベル PARSE_LEVEL（@@interface 文）の値	テンプレートファイル情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
XML **可変記号補足情報：**XMLValiableSymbolAuxiliary					
パースレベル	ParseLevel	1	文字列	パースレベル PARSE_LEVEL（@@interface 文）の値	テンプレートファイル情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
XMLElement **可変記号情報：**XMLElementValiableSymbol					
値	Value	1 以上	文字列	可変記号の値（ELEMENT）	テンプレートファイル情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
XMLElement **可変記号補足情報：**XMLElementValiableSymbolAuxiliary					
タグ名	TagName	1 以上	文字列	タグ名 TAG_NAME（@@interface 文）の値	テンプレートファイル情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
属性名	AttributeName	1	文字列	属性名 ATTR_NAME（@@interface 文）の値	テンプレートファイル情報
子可変記号情報	*ValiableSymbol*	1	―		可変記号情報
共通ユーザ追加処理情報：CommonUoc					
言語区分番号	ProgramNO	1	文字列	リポジトリアクセスの言語区分番号	
ユーザ追加処理情報	*UocBlock*	1	―		
ユーザ追加処理情報：UocBlock					
ブロック ID	ID	1	文字列	UOC 名（@@uoc 文）/ ブロック名（@@diagram 文）/ 業務ルール展開名（@@rule 文）/ ファイル名（@@file 文）/ 部品ブロック名（@@parts 文）	テンプレートファイル情報
親ブロック ID	ParentID	1	文字列	親ブロック名 PARENT（@@uoc 文 /@@diagram 文 /@@rule 文 /@@file 文 /@@parts 文）の値	テンプレートファイル情報
種別	Kind	1	"UOC"：ユーザ追加処理 "BLOCK"：ブロック "RULE"：業務ルール "FILE"：ファイル "PARTS"：部品	ブロックの種別	
コメント	Comment	1	文字列	ブロックの説明 COMMENT（@@uoc 文 / @@diagram 文 / @@rule 文 /@@file 文 /@@parts 文）の値	テンプレートファイル情報

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
ユーザ追加処理テキスト	Text	1	文字列	ユーザ追加処理のテキスト（種別が "UOC" の場合）	
対応ソースファイル情報	*MutiSrc*	1	―	対応ソースファイル情報	
業務ルール文情報	*RuleSentence*	1	―	業務ルール文情報（種別が "RULE" の場合）	
業務ルール展開情報	*RuleExpand*	1	―	業務ルール展開情報（種別が "RULE" の場合）	
対応ソースファイル情報：MultiSrc					
対応ソースファイル名	FileName	1	文字列	ソースファイル名	テンプレートファイル情報
業務ルール文情報：RuleSentence					
適用条件	Usage	1	"USE"：利用する "NOUSE"：利用しない	適用条件有無 WITH USAGE（@@rule 文）の値	
業務ルール文詳細情報	*RuleSentenceFull*	1	―		
業務ルール文詳細情報：RuleSentenceFull					
種別	Kind	1	"IN"：入力 "OUT"：出力 "NONE"：なし	適用条件の種別 USAGE（@@rule 文）の値	テンプレートファイル情報
業務ルール可変記号情報	*RuleValiableSymbol*	1	―		
業務ルール可変記号情報：RuleValiableSymbol					
可変記号	Name	1	文字列	REF（@@rule 文）の可変記号名	テンプレートファイル情報
可変記号配列	Array	1	文字列	REF（@@rule 文）の可変記号の添字	テンプレートファイル情報
子可変記号	ChildName	1	文字列	REF（@@rule 文）の可変記号の子可変記号名（ATTR=ITEM の場合）	テンプレートファイル情報

タグ種

含まれるタグ	要素型名	タグ出現回数	設定可能値	説明	備考
子可変記号配列	ChildArray	1	文字列	REF（@@rule 文）の可変記号の子可変記号の添字	テンプレートファイル情報
データ項目 OID リスト	ObjectID	1 以上	文字列	REF（@@rule 文）の可変記号のデータ項目ＯＩＤ	テンプレートファイル情報
接頭語	Prefix	1	文字列	PREFIX 句（@@rule 文）の値	テンプレートファイル情報
接尾語	Suffix	1	文字列	SUFFIX 句（@@rule 文）の値	テンプレートファイル情報
TOP 句	TopSpecified	1	"NOSPECIFIED"：TOP 句なし "SPECIFIED"：TOP 句あり	TOP 句の有無	
最上位データ項目名	TopItemName	1	文字列	TOP 句（@@rule 文）の値	テンプレートファイル情報
業務ルール展開情報：RuleExpand					
業務ルール名	Name	1	文字列	業務ルールの名称	辞書情報
業務ルール OID	ObjectID	1	文字列	業務ルールの OID	辞書情報
更新日付	UpdateTime	1	文字列："YYYY/MM/DD" 形式	データ項目定義ファイルの更新日付	
業務ルール更新日付	RuleUpdateTime	1	文字列："YYYY/MM/DD" 形式	業務ルールの更新日付辞書情報	辞書情報
展開順序	Order	1	文字列	業務ルールの展開順序（業務ルール展開設定ダイアログ）	
コメント	Comment	1	文字列	業務ルールのコメント	辞書情報
業務ルールデータ項目情報	*RuleDataItem*	1	―	業務ルールの関連データ項目情報	辞書情報
業務ルールスクリプト	Script	1	文字列	業務ルールのスクリプト	辞書情報
業務ルールデータ項目情報：RuleDataItem					
データ項目名	Name	1	文字列	データ項目名	辞書情報

タグ種					
含まれるタグ	**要素型名**	**タグ出現回数**	**設定可能値**	**説明**	**備考**
データ項目 OID	ObjectID	1	文字列	データ項目の OID	辞書情報
言語別名称	LangName	1	文字列	言語別名称	辞書情報
標準名称	StandardName	4	文字列	標準名称	辞書情報
キーワード	Keyword	1	文字列	キーワード	辞書情報
適用条件	ApplyCondition	1	"IN"：入力 "OUT"：出力 "NONE"：なし	適用条件	辞書情報
業務ルール可変記号情報	*RuleValiableSymbol*	1	―	関連業務ルール可変記号情報	
業務ルールデータ項目階層情報	*RuleDataItemStructure*	1	―	データ項目階層情報	辞書情報
業務ルールデータ項目階層情報：RuleDataItemStructure					
データ項目名	Name	1	文字列	データ項目名	辞書情報
データ項目 OID	ObjectID	1	文字列	データ項目の OID	辞書情報
言語別名称	LangName	1	文字列	言語別名称	辞書情報

注 1
斜体で表記している要素型名の設定可能値については，対応するタグ種を参照してください。
注 2
属性値に含まれる「<」「&」「"」，および内容（文字データ）に含まれる「<」「&」は，それぞれ次のように置換します。

- <：<
- &：&
- "："

注※
備考に「辞書情報」と記載しているタグ情報でも，レコード定義ファイルを使用している場合には，レコード定義の内容によってレコード定義情報が設定されます。

（2）DTD（Document Type Definiton）

プログラム定義ファイルの DTD を次に示します。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE XMI [
<!ELEMENT XMI.content (Construction.File,
                       Construction.Program,
                       Construction.CommonValiableSymbol,
                       Construction.CommonUoc)>
<!ELEMENT Construction.File (Construction.File.ID,
                             Construction.File.Type,
```

```
                              Construction.File.Version,
                              Construction.File.Status?)>
<!ELEMENT Construction.File.ID (#PCDATA)>
<!ELEMENT Construction.File.Type (#PCDATA)>
<!ELEMENT Construction.File.Version (#PCDATA)>
<!ELEMENT Construction.File.Status (#PCDATA)>
<!ELEMENT Construction.Program (Construction.Program.Name?,
                                Construction.Program.Author?,
                                Construction.Program.Outline?,
                                Construction.Program.ProgramLang,
                                Construction.Program.Memo?,
                                Construction.Program.templatefile,
                               Construction.Program.resourcefile*,
                                Construction.Program.rule*)>
<!ELEMENT Construction.Program.Name (#PCDATA)>
<!ELEMENT Construction.Program.Author (#PCDATA)>
<!ELEMENT Construction.Program.Outline (#PCDATA)>
<!ELEMENT Construction.Program.ProgramLang (#PCDATA)>
<!ELEMENT Construction.Program.Memo (#PCDATA)>
<!ELEMENT Construction.Program.templatefile
(Construction.TemplateFile)>
<!ELEMENT Construction.TemplateFile
(Construction.TemplateFile.Name,
                             Construction.TemplateFile.UpdateTime,
                               Construction.TemplateFile.PathUse)>
<!ELEMENT Construction.TemplateFile.Name (#PCDATA)>
<!ELEMENT Construction.TemplateFile.UpdateTime (#PCDATA)>
<!ELEMENT Construction.TemplateFile.PathUse (#PCDATA)>
<!ELEMENT Construction.Program.resourcefile
(Construction.ResourceFile*)>
<!ELEMENT Construction.ResourceFile
(Construction.ResourceFile.Name,
                                   Construction.ResourceFile.Kind,
                                  Construction.ResourceFile.Path?,
                             Construction.ResourceFile.UpdateTime,
                                Construction.ResourceFile.PathUse,

Construction.ResourceFile.ReferenceCount?)>
<!ELEMENT Construction.ResourceFile.Name (#PCDATA)>
<!ELEMENT Construction.ResourceFile.Kind (#PCDATA)>
<!ELEMENT Construction.ResourceFile.Path (#PCDATA)>
<!ELEMENT Construction.ResourceFile.UpdateTime (#PCDATA)>
<!ELEMENT Construction.ResourceFile.PathUse (#PCDATA)>
<!ELEMENT Construction.ResourceFile.ReferenceCount (#PCDATA)>
<!ELEMENT Construction.Program.rule (Construction.Rule*)>
<!ELEMENT Construction.Rule (Construction.Rule.Name,
                             Construction.Rule.ObjectID,
                             Construction.Rule.UpdateTime,
                             Construction.Rule.ReferenceCount)>
<!ELEMENT Construction.Rule.Name (#PCDATA)>
<!ELEMENT Construction.Rule.ObjectID (#PCDATA)>
<!ELEMENT Construction.Rule.UpdateTime (#PCDATA)>
<!ELEMENT Construction.Rule.ReferenceCount (#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol
(Construction.CommonValiableSymbol.ProgramLang,

Construction.CommonValiableSymbol.LangStringType,

Construction.CommonValiableSymbol.LangType,

Construction.CommonValiableSymbol.RepositoryLang?,

Construction.CommonValiableSymbol.Extension,

Construction.CommonValiableSymbol.ModifyConnect,
```

```
Construction.CommonValiableSymbol.ModifyOrder,

Construction.CommonValiableSymbol.TopChar,

Construction.CommonValiableSymbol.valiablesymbol*)>
<!ELEMENT Construction.CommonValiableSymbol.ProgramLang (#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.LangStringType
(#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.LangType (#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.RepositoryLang
(#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.Extension (#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.ModifyConnect
(#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.ModifyOrder (#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.TopChar (#PCDATA)>
<!ELEMENT Construction.CommonValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol
(Construction.ValiableSymbol.Name,
                                Construction.ValiableSymbol.Kind,
                              Construction.ValiableSymbol.Comment?,
                              Construction.ValiableSymbol.ArrayMax,

Construction.ValiableSymbol.standardvaliablesymbol*,

Construction.ValiableSymbol.standardvaliablesymbolauxiliary*,

Construction.ValiableSymbol.datavaliablesymbol*,

Construction.ValiableSymbol.datavaliablesymbolauxiliary*,

Construction.ValiableSymbol.dataitemvaliablesymbol*,

Construction.ValiableSymbol.dataitemvaliablesymbolauxiliary*,

Construction.ValiableSymbol.partsvaliablesymbol*,

Construction.ValiableSymbol.interfacevaliablesymbol*,

Construction.ValiableSymbol.interfacevaliablesymbolauxiliary*,

Construction.ValiableSymbol.xmap3valiablesymbol*,

Construction.ValiableSymbol.xmap3valiablesymbolauxiliary*,

Construction.ValiableSymbol.xmlvaliablesymbol*,

Construction.ValiableSymbol.xmlvaliablesymbolauxiliary*,

Construction.ValiableSymbol.xmlelementvaliablesymbol*,

Construction.ValiableSymbol.xmlelementvaliablesymbolauxiliary*
                                        )>
<!ELEMENT Construction.ValiableSymbol.Name (#PCDATA)>
<!ELEMENT Construction.ValiableSymbol.Kind (#PCDATA)>
<!ELEMENT Construction.ValiableSymbol.Comment (#PCDATA)>
<!ELEMENT Construction.ValiableSymbol.ArrayMax (#PCDATA)>
<!ELEMENT Construction.ValiableSymbol.standardvaliablesymbol
(Construction.StandardValiableSymbol*)>
<!ELEMENT Construction.StandardValiableSymbol
(Construction.StandardValiableSymbol.Value,

Construction.StandardValiableSymbol.valiablesymbol*)>
```

```
<!ELEMENT Construction.StandardValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.StandardValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT
Construction.ValiableSymbol.standardvaliablesymbolauxiliary
(Construction.StandardValiableSymbolAuxiliary)>
<!ELEMENT Construction.StandardValiableSymbolAuxiliary
(Construction.StandardValiableSymbolAuxiliary.Value,

Construction.StandardValiableSymbolAuxiliary.valiablesymbol)>
<!ELEMENT Construction.StandardValiableSymbolAuxiliary.Value
(#PCDATA)>
<!ELEMENT
Construction.StandardValiableSymbolAuxiliary.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.datavaliablesymbol
(Construction.DataValiableSymbol*)>
<!ELEMENT Construction.DataValiableSymbol
(Construction.DataValiableSymbol.Value,

Construction.DataValiableSymbol.resourcefile,

Construction.DataValiableSymbol.valiablesymbol)>
<!ELEMENT Construction.DataValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.DataValiableSymbol.resourcefile
(Construction.ResourceFile*)>
<!ELEMENT Construction.DataValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.datavaliablesymbolauxiliary
(Construction.DataValiableSymbolAuxiliary*)>
<!ELEMENT Construction.DataValiableSymbolAuxiliary
(Construction.DataValiableSymbolAuxiliary.Kind,

Construction.DataValiableSymbolAuxiliary.IOKind,

Construction.DataValiableSymbolAuxiliary.valiablesymbol)>
<!ELEMENT Construction.DataValiableSymbolAuxiliary.Kind (#PCDATA)>
<!ELEMENT Construction.DataValiableSymbolAuxiliary.IOKind
(#PCDATA)>
<!ELEMENT Construction.DataValiableSymbolAuxiliary.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.dataitemvaliablesymbol
(Construction.DataItemValiableSymbol*)>
<!ELEMENT Construction.DataItemValiableSymbol
(Construction.DataItemValiableSymbol.Value,

Construction.DataItemValiableSymbol.StandardName,

Construction.DataItemValiableSymbol.UpperDataItem,

Construction.DataItemValiableSymbol.ValiableSymbolName,

Construction.DataItemValiableSymbol.Contarray,

Construction.DataItemValiableSymbol.valiablesymbol)>
<!ELEMENT Construction.DataItemValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.DataItemValiableSymbol.StandardName
(#PCDATA)>
<!ELEMENT Construction.DataItemValiableSymbol.UpperDataItem
(#PCDATA)>
<!ELEMENT Construction.DataItemValiableSymbol.ValiableSymbolName
(#PCDATA)>
<!ELEMENT Construction.DataItemValiableSymbol.Contarray (#PCDATA)>
<!ELEMENT Construction.DataItemValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
```

```
<!ELEMENT
Construction.ValiableSymbol.dataitemvaliablesymbolauxiliary
(Construction.DataItemValiableSymbolAuxiliary)>
<!ELEMENT Construction.DataItemValiableSymbolAuxiliary
(Construction.DataItemValiableSymbolAuxiliary.ReferenceKind,

Construction.DataItemValiableSymbolAuxiliary.ReferenceValiableSymb
ol?,

Construction.DataItemValiableSymbolAuxiliary.valiablesymbol)>
<!ELEMENT
Construction.DataItemValiableSymbolAuxiliary.ReferenceKind
(#PCDATA)>
<!ELEMENT
Construction.DataItemValiableSymbolAuxiliary.ReferenceValiableSymb
ol (#PCDATA)>
<!ELEMENT
Construction.DataItemValiableSymbolAuxiliary.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.partsvaliablesymbol
(Construction.PartsValiableSymbol*)>
<!ELEMENT Construction.partsvaliablesymbol
(Construction.PartsValiableSymbol.resourcefile,

Construction.PartsValiableSymbol.procedure)>
<!ELEMENT Construction.PartsValiableSymbol.resourcefile
(Construction.ResourceFile*)>
<!ELEMENT Construction.PartsValiableSymbol.procedure
(Construction.Procedure*)>
<!ELEMENT Construction.Procedure (Construction.Procedure.Name,
                                  Construction.Procedure.Comment?,
                                 Construction.Procedure.parameter)>
<!ELEMENT Construction.Procedure.Name (#PCDATA)>
<!ELEMENT Construction.Procedure.Comment (#PCDATA)>
<!ELEMENT Construction.Procedure.parameter
(Construction.Parameter*)>
<!ELEMENT Construction.Parameter (Construction.Parameter.Name,
                                  Construction.Parameter.Comment?,
                                  Construction.Parameter.Type,
                                  Construction.Parameter.Value?,

Construction.Parameter.ValiableSymbolName?,
                                 Construction.Parameter.Contarray)>
<!ELEMENT Construction.Parameter.Name (#PCDATA)>
<!ELEMENT Construction.Parameter.Comment (#PCDATA)>
<!ELEMENT Construction.Parameter.Type (#PCDATA)>
<!ELEMENT Construction.Parameter.Value (#PCDATA)>
<!ELEMENT Construction.Parameter.ValiableSymbolName (#PCDATA)>
<!ELEMENT Construction.Parameter.Contarray (#PCDATA)>
<!ELEMENT Construction.ValiableSymbol.interfacevaliablesymbol
(Construction.InterfaceValiableSymbol*)>
<!ELEMENT Construction.InterfaceValiableSymbol
(Construction.InterfaceValiableSymbol.Value,

Construction.InterfaceValiableSymbol.resourcefile,

Construction.InterfaceValiableSymbol.IDLFileName,

Construction.InterfaceValiableSymbol.InterfaceName,

Construction.InterfaceValiableSymbol.OperationName?,

Construction.InterfaceValiableSymbol.IDLFileObject,

Construction.InterfaceValiableSymbol.interfaceobject,
```

```
Construction.InterfaceValiableSymbol.operation,

Construction.InterfaceValiableSymbol.valiablesymbol)>
<!ELEMENT Construction.InterfaceValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.InterfaceValiableSymbol.resourcefile
(Construction.ResourceFile*)>
<!ELEMENT Construction.InterfaceValiableSymbol.IDLFileName
(#PCDATA)>
<!ELEMENT Construction.InterfaceValiableSymbol.InterfaceName
(#PCDATA)>
<!ELEMENT Construction.InterfaceValiableSymbol.OperationName
(#PCDATA)>
<!ELEMENT Construction.InterfaceValiableSymbol.IDLFileObject
(Construction.IDLFileObject*)>
<!ELEMENT Construction.IDLFileObject
(Construction.IDLFileObject.interfaceobject?)>
<!ELEMENT Construction.IDLFileObject.interfaceobject
(Construction.InterfaceObject*)>
<!ELEMENT Construction.InterfaceValiableSymbol.interfaceobject
(Construction.InterfaceObject*)>
<!ELEMENT Construction.InterfaceObject
(Construction.InterfaceObject.IDLFileName,
                              Construction.InterfaceObject.module,

Construction.InterfaceObject.interface,

Construction.InterfaceObject.attribute,

Construction.InterfaceObject.operation)>
<!ELEMENT Construction.InterfaceObject.IDLFileName (#PCDATA)>
<!ELEMENT Construction.InterfaceObject.module
(Construction.Module*)>
<!ELEMENT Construction.Module (Construction.Module.Name?,
                               Construction.Module.AliasName?,
                               Construction.Module.Comment?,
                               Construction.Module.ScopeName?)>
<!ELEMENT Construction.Module.Name (#PCDATA)>
<!ELEMENT Construction.Module.AliasName (#PCDATA)>
<!ELEMENT Construction.Module.Comment (#PCDATA)>
<!ELEMENT Construction.Module.ScopeName (#PCDATA)>
<!ELEMENT Construction.InterfaceObject.interface
(Construction.Interface*)>
<!ELEMENT Construction.Interface (Construction.Interface.Name,
                                  Construction.Interface.AliasName,
                                   Construction.Interface.Comment?,
                                 Construction.Interface.ObjectName,
                           Construction.Interface.ObjectAliaseName,
                                  Construction.Interface.ScopeName,

Construction.Interface.BaseInterfaceName?)>
<!ELEMENT Construction.Interface.Name (#PCDATA)>
<!ELEMENT Construction.Interface.AliasName (#PCDATA)>
<!ELEMENT Construction.Interface.Comment (#PCDATA)>
<!ELEMENT Construction.Interface.ObjectName (#PCDATA)>
<!ELEMENT Construction.Interface.ObjectAliaseName (#PCDATA)>
<!ELEMENT Construction.Interface.ScopeName (#PCDATA)>
<!ELEMENT Construction.Interface.BaseInterfaceName (#PCDATA)>
<!ELEMENT Construction.InterfaceObject.attribute
(Construction.Attribute*)>
<!ELEMENT Construction.Attribute (Construction.Attribute.Name,
                                  Construction.Attribute.Readonly,
                                   Construction.Attribute.TypeName,
                                   Construction.Attribute.UserType,
                                  Construction.Attribute.ScopeName,
```

```
                                        Construction.Attribute.ObjectID,

Construction.Attribute.interfacedataitem)>
<!ELEMENT Construction.Attribute.Name (#PCDATA)>
<!ELEMENT Construction.Attribute.Readonly (#PCDATA)>
<!ELEMENT Construction.Attribute.TypeName (#PCDATA)>
<!ELEMENT Construction.Attribute.UserType (#PCDATA)>
<!ELEMENT Construction.Attribute.ScopeName (#PCDATA)>
<!ELEMENT Construction.Attribute.ObjectID (#PCDATA)>
<!ELEMENT Construction.Attribute.interfacedataitem
(Construction.InterfaceDataItem*)>
<!ELEMENT Construction.InterfaceDataItem
(Construction.InterfaceDataItem.ObjectID,

Construction.InterfaceDataItem.UpdateTime,

Construction.InterfaceDataItem.Type,

Construction.InterfaceDataItem.record*,

Construction.InterfaceDataItem.dataitem*)>
<!ELEMENT Construction.InterfaceDataItem.ObjectID (#PCDATA)>
<!ELEMENT Construction.InterfaceDataItem.UpdateTime (#PCDATA)>
<!ELEMENT Construction.InterfaceDataItem.Type (#PCDATA)>
<!ELEMENT Construction.InterfaceDataItem.record
(Construction.Record)>
<!ELEMENT Construction.Record (Construction.Record.ObjectID,
                               Construction.Record.DataItemNum,
                               Construction.Record.dataitem)>
<!ELEMENT Construction.Record.ObjectID (#PCDATA)>
<!ELEMENT Construction.Record.DataItemNum (#PCDATA)>
<!ELEMENT Construction.Record.dataitem (Construction.DataItem+)>
<!ELEMENT Construction.DataItem
(Construction.DataItem.Name,Construction.DataItem.ObjectID,

Construction.DataItem.Kind,Construction.DataItem.Level,

Construction.DataItem.StandardName+,Construction.DataItem.Kana+,

Construction.DataItem.Comment*,Construction.DataItem.Field+,

Construction.DataItem.TypeGroup,Construction.DataItem.TypeGroupStr
ing,

Construction.DataItem.Total,Construction.DataItem.DecimalPart,

Construction.DataItem.Repeat,Construction.DataItem.dict,
                                    Construction.DataItem.keyword)>
<!ELEMENT Construction.DataItem.Name (#PCDATA)>
<!ELEMENT Construction.DataItem.ObjectID (#PCDATA)>
<!ELEMENT Construction.DataItem.Kind (#PCDATA)>
<!ELEMENT Construction.DataItem.Level (#PCDATA)>
<!ELEMENT Construction.DataItem.StandardName (#PCDATA)>
<!ELEMENT Construction.DataItem.Kana (#PCDATA)>
<!ELEMENT Construction.DataItem.Comment (#PCDATA)>
<!ELEMENT Construction.DataItem.Field (#PCDATA)>
<!ELEMENT Construction.DataItem.TypeGroup (#PCDATA)>
<!ELEMENT Construction.DataItem.TypeGroupString (#PCDATA)>
<!ELEMENT Construction.DataItem.Total (#PCDATA)>
<!ELEMENT Construction.DataItem.DecimalPart (#PCDATA)>
<!ELEMENT Construction.DataItem.Repeat (#PCDATA)>
<!ELEMENT Construction.DataItem.dict (Construction.Dict)>
<!ELEMENT Construction.Dict
(Construction.Dict.Lang,Construction.Dict.Name?,
```

```
Construction.Dict.Type?,Construction.Dict.TypeString?,

Construction.Dict.FreeString?,Construction.Dict.EditString?,

Construction.Dict.InitValue?,Construction.Dict.Field?)*>
<!ELEMENT Construction.Dict.Lang (#PCDATA)>
<!ELEMENT Construction.Dict.Name (#PCDATA)>
<!ELEMENT Construction.Dict.Type (#PCDATA)>
<!ELEMENT Construction.Dict.TypeString (#PCDATA)>
<!ELEMENT Construction.Dict.FreeString (#PCDATA)>
<!ELEMENT Construction.Dict.EditString (#PCDATA)>
<!ELEMENT Construction.Dict.InitValue (#PCDATA)>
<!ELEMENT Construction.Dict.Field (#PCDATA)>
<!ELEMENT Construction.DataItem.keyword (Construction.Keyword)>
<!ELEMENT Construction.Keyword (Construction.Keyword.DataItem,
                                Construction.Keyword.Len?,
                                Construction.Keyword.PLen?,
                                Construction.Keyword.ILen?,
                                Construction.Keyword.DLen?,
                                Construction.Keyword.Value?,
                                Construction.Keyword.Occurs?,
                                Construction.Keyword.Editchar?,
                                Construction.Keyword.Comp?,
                                Construction.Keyword.Free?)*>
<!ELEMENT Construction.Keyword.DataItem (#PCDATA)>
<!ELEMENT Construction.Keyword.Len (#PCDATA)>
<!ELEMENT Construction.Keyword.PLen (#PCDATA)>
<!ELEMENT Construction.Keyword.ILen (#PCDATA)>
<!ELEMENT Construction.Keyword.DLen (#PCDATA)>
<!ELEMENT Construction.Keyword.Value (#PCDATA)>
<!ELEMENT Construction.Keyword.Occurs (#PCDATA)>
<!ELEMENT Construction.Keyword.Editchar (#PCDATA)>
<!ELEMENT Construction.Keyword.Comp (#PCDATA)>
<!ELEMENT Construction.Keyword.Free (#PCDATA)>
<!ELEMENT Construction.InterfaceDataItem.dataitem
(Construction.DataItem+)>
<!ELEMENT Construction.InterfaceObject.operation
(Construction.Operation*)>
<!ELEMENT Construction.Operation (Construction.Operation.Name,
                                  Construction.Operation.AliasName,
                                   Construction.Operation.Comment?,
                                   Construction.Operation.Oneway,
                               Construction.Operation.ReturnTypeName,
                              Construction.Operation.ReturnUserType?,
                                   Construction.Operation.ScopeName?,
                                    Construction.Operation.ObjectID?,

Construction.Operation.interfacedataitem,
                                    Construction.Operation.argument,
                                   Construction.Operation.exception,
                         Construction.Operation.requestcontext)>
<!ELEMENT Construction.Operation.Name (#PCDATA)>
<!ELEMENT Construction.Operation.AliasName (#PCDATA)>
<!ELEMENT Construction.Operation.Comment (#PCDATA)>
<!ELEMENT Construction.Operation.Oneway (#PCDATA)>
<!ELEMENT Construction.Operation.ReturnTypeName (#PCDATA)>
<!ELEMENT Construction.Operation.ReturnUserType (#PCDATA)>
<!ELEMENT Construction.Operation.ScopeName (#PCDATA)>
<!ELEMENT Construction.Operation.ObjectID (#PCDATA)>
<!ELEMENT Construction.Operation.interfacedataitem
(Construction.InterfaceDataItem*)>
<!ELEMENT Construction.Operation.argument
(Construction.Argument*)>
<!ELEMENT Construction.Argument (Construction.Argument.Name,
                                 Construction.Argument.Mode,
```

```
                                    Construction.Argument.TypeName,
                                    Construction.Argument.UserType?,
                                    Construction.Argument.ScopeName?,
                                    Construction.Argument.ObjectID,
                                Construction.Argument.interfacedataitem)>
<!ELEMENT Construction.Argument.Name (#PCDATA)>
<!ELEMENT Construction.Argument.Mode (#PCDATA)>
<!ELEMENT Construction.Argument.TypeName (#PCDATA)>
<!ELEMENT Construction.Argument.UserType (#PCDATA)>
<!ELEMENT Construction.Argument.ScopeName (#PCDATA)>
<!ELEMENT Construction.Argument.ObjectID (#PCDATA)>
<!ELEMENT Construction.Argument.interfacedataitem
(Construction.InterfaceDataItem*)>
<!ELEMENT Construction.Operation.exception
(Construction.Exception*)>
<!ELEMENT Construction.Exception (Construction.Exception.Name,
                                  Construction.Exception.UserType,
                                  Construction.Exception.ScopeName,
                                  Construction.Exception.ObjectID,

Construction.Exception.interfacedataitem)>
<!ELEMENT Construction.Exception.Name (#PCDATA)>
<!ELEMENT Construction.Exception.UserType (#PCDATA)>
<!ELEMENT Construction.Exception.ScopeName (#PCDATA)>
<!ELEMENT Construction.Exception.ObjectID (#PCDATA)>
<!ELEMENT Construction.Exception.interfacedataitem
(Construction.InterfaceDataItem*)>
<!ELEMENT Construction.Operation.requestcontext
(Construction.RequestContext*)>
<!ELEMENT Construction.RequestContext
(Construction.RequestContext.Name,

Construction.RequestContext.Comment?)>
<!ELEMENT Construction.RequestContext.Name (#PCDATA)>
<!ELEMENT Construction.RequestContext.Comment (#PCDATA)>
<!ELEMENT
Construction.ValiableSymbol.interfacevaliablesymbolauxiliary
(Construction.InterfaceValiableSymbolAuxiliary*)>
<!ELEMENT Construction.InterfaceValiableSymbolAuxiliary
(Construction.InterfaceValiableSymbolAuxiliary.valiablesymbol*)>
<!ELEMENT
Construction.InterfaceValiableSymbolAuxiliary.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.xmap3valiablesymbol
(Construction.Xmap3ValiableSymbol*)>
<!ELEMENT Construction.Xmap3ValiableSymbol
(Construction.Xmap3ValiableSymbol.Value,

Construction.Xmap3ValiableSymbol.resourcefile,

Construction.Xmap3ValiableSymbol.valiablesymbol)>
<!ELEMENT Construction.Xmap3ValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.Xmap3ValiableSymbol.resourcefile
(Construction.ResourceFile*)>
<!ELEMENT Construction.Xmap3ValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.xmap3valiablesymbolauxiliary
(Construction.Xmap3ValiableSymbolAuxiliary*)>
<!ELEMENT Construction.Xmap3ValiableSymbolAuxiliary
(Construction.Xmap3ValiableSymbolAuxiliary.IOKind,

Construction.Xmap3ValiableSymbolAuxiliary.valiablesymbol)>
<!ELEMENT Construction.Xmap3ValiableSymbolAuxiliary.IOKind
(#PCDATA)>
<!ELEMENT Construction.Xmap3ValiableSymbolAuxiliary.valiablesymbol
```

```
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.xmlvaliablesymbol
(Construction.XMLValiableSymbol)*>
<!ELEMENT Construction.XMLValiableSymbol
(Construction.XMLValiableSymbol.Value,

Construction.XMLValiableSymbol.resource,

Construction.XMLValiableSymbol.ParseLevel,

Construction.XMLValiableSymbol.valiablesymbol)>
<!ELEMENT Construction.XMLValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.XMLValiableSymbol.resourcefile
(Construction.ResourceFile*)>
<!ELEMENT Construction.XMLValiableSymbol.ParseLevel (#PCDATA)>
<!ELEMENT Construction.XMLValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.xmlvaliablesymbolauxiliary
(Construction.XMLValiableSymbolAuxiliary)>
<!ELEMENT Construction.XMLValiableSymbolAuxiliary
(Construction.XMLValiableSymbolAuxiliary.ParseLevel,

Construction.XMLValiableSymbolAuxiliary.valiablesymbol)>
<!ELEMENT Construction.XMLValiableSymbolAuxiliary.ParseLevel
(#PCDATA)>
<!ELEMENT Construction.XMLValiableSymbolAuxiliary.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.ValiableSymbol.xmlelementvaliablesymbol
(Construction.XMLElementValiableSymbol*)>
<!ELEMENT Construction.XMLElementValiableSymbol
(Construction.XMLElementValiableSymbol.Value,

Construction.XMLElementValiableSymbol.valiablesymbol)>
<!ELEMENT Construction.XMLElementValiableSymbol.Value (#PCDATA)>
<!ELEMENT Construction.XMLElementValiableSymbol.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT
Construction.ValiableSymbol.xmlelementvaliablesymbolauxiliary
(Construction.XMLElementValiableSymbolAuxiliary)*>
<!ELEMENT Construction.XMLElementValiableSymbolAuxiliary
(Construction.XMLElementValiableSymbolAuxiliary.TagName,

Construction.XMLElementValiableSymbolAuxiliary.AttributeName,

Construction.XMLElementValiableSymbolAuxiliary.valiablesymbol)>
<!ELEMENT Construction.XMLElementValiableSymbolAuxiliary.TagName
(#PCDATA)>
<!ELEMENT
Construction.XMLElementValiableSymbolAuxiliary.AttributeName
(#PCDATA)>
<!ELEMENT
Construction.XMLElementValiableSymbolAuxiliary.valiablesymbol
(Construction.ValiableSymbol*)>
<!ELEMENT Construction.CommonUoc
(Construction.CommonUoc.ProgramNo,
                                  Construction.CommonUoc.uocblock)>
<!ELEMENT Construction.CommonUoc.ProgramNo (#PCDATA)>
<!ELEMENT Construction.CommonUoc.uocblock
(Construction.UocBlock*)>
<!ELEMENT Construction.UocBlock (Construction.UocBlock.ID,
                                 Construction.UocBlock.ParentID?,
                                 Construction.UocBlock.Kind,
                                 Construction.UocBlock.Comment?,
                                 Construction.UocBlock.Text?,
                                 Construction.UocBlock.Thinkaid?,
```

```
                                    Construction.UocBlock.multisrc,
                                   Construction.UocBlock.rulesentence,
                                    Construction.UocBlock.ruleexpand)>
<!ELEMENT Construction.UocBlock.ID (#PCDATA)>
<!ELEMENT Construction.UocBlock.ParentID (#PCDATA)>
<!ELEMENT Construction.UocBlock.Kind (#PCDATA)>
<!ELEMENT Construction.UocBlock.Comment (#PCDATA)>
<!ELEMENT Construction.UocBlock.Text (#PCDATA)>
<!ELEMENT Construction.UocBlock.Thinkaid (#PCDATA)>
<!ELEMENT Construction.UocBlock.multisrc (Construction.MultiSrc*)>
<!ELEMENT Construction.MultiSrc (Construction.MultiSrc.FileName)>
<!ELEMENT Construction.MultiSrc.FileName (#PCDATA)>
<!ELEMENT Construction.UocBlock.rulesentence
(Construction.RuleSentence*)>
<!ELEMENT Construction.RuleSentence
(Construction.RuleSentence.Usage,

Construction.RuleSentence.rulesentencefull)>
<!ELEMENT Construction.RuleSentence.Usage (#PCDATA)>
<!ELEMENT Construction.RuleSentence.rulesentencefull
(Construction.RuleSentenceFull*)>
<!ELEMENT Construction.RuleSentenceFull
(Construction.RuleSentenceFull.Kind,

Construction.RuleSentenceFull.rulevaliablesymbol)>
<!ELEMENT Construction.RuleSentenceFull.Kind (#PCDATA)>
<!ELEMENT Construction.RuleSentenceFull.rulevaliablesymbol
(Construction.RuleValiableSymbol*)>
<!ELEMENT Construction.RuleValiableSymbol
(Construction.RuleValiableSymbol.Name,

Construction.RuleValiableSymbol.Array,

Construction.RuleValiableSymbol.ChildName?,

Construction.RuleValiableSymbol.ChildArray,

Construction.RuleValiableSymbol.ObjectID+,

Construction.RuleValiableSymbol.Prefix?,

Construction.RuleValiableSymbol.Suffix?,

Construction.RuleValiableSymbol.TopSpecified,

Construction.RuleValiableSymbol.TopItemName?)>
<!ELEMENT Construction.RuleValiableSymbol.Name (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.Array (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.ChildName (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.ChildArray (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.ObjectID (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.Prefix (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.Suffix (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.TopSpecified (#PCDATA)>
<!ELEMENT Construction.RuleValiableSymbol.TopItemName (#PCDATA)>
<!ELEMENT Construction.UocBlock.ruleexpand
(Construction.RuleExpand*)>
<!ELEMENT Construction.RuleExpand (Construction.RuleExpand.Name,
                                   Construction.RuleExpand.ObjectID,
                                  Construction.RuleExpand.UpdateTime,
                               Construction.RuleExpand.RuleUpdateTime,
                                     Construction.RuleExpand.Order,
                                  Construction.RuleExpand.Comment?,
                                Construction.RuleExpand.ruledataitem,
                                     Construction.RuleExpand.Script)>
```

```
<!ELEMENT Construction.RuleExpand.Name (#PCDATA)>
<!ELEMENT Construction.RuleExpand.ObjectID (#PCDATA)>
<!ELEMENT Construction.RuleExpand.UpdateTime (#PCDATA)>
<!ELEMENT Construction.RuleExpand.RuleUpdateTime (#PCDATA)>
<!ELEMENT Construction.RuleExpand.Order (#PCDATA)>
<!ELEMENT Construction.RuleExpand.Comment (#PCDATA)>
<!ELEMENT Construction.RuleExpand.ruledataitem
(Construction.RuleDataItem*)>
<!ELEMENT Construction.RuleDataItem
(Construction.RuleDataItem.Name,
                              Construction.RuleDataItem.ObjectID,
                              Construction.RuleDataItem.LangName,

Construction.RuleDataItem.StandardName+,
                               Construction.RuleDataItem.Keyword,

Construction.RuleDataItem.Applycondition,

Construction.RuleDataItem.rulevaliablesymbol,

Construction.RuleDataItem.ruledataitemstructure)>
<!ELEMENT Construction.RuleDataItem.Name (#PCDATA)>
<!ELEMENT Construction.RuleDataItem.ObjectID (#PCDATA)>
<!ELEMENT Construction.RuleDataItem.LangName (#PCDATA)>
<!ELEMENT Construction.RuleDataItem.StandardName (#PCDATA)>
<!ELEMENT Construction.RuleDataItem.Keyword (#PCDATA)>
<!ELEMENT Construction.RuleDataItem.Applycondition (#PCDATA)>

<!ELEMENT Construction.RuleDataItem.rulevaliablesymbol
(Construction.RuleValiableSymbol*)>
<!ELEMENT Construction.RuleDataItem.ruledataitemstructure
(Construction.RuleDataItemStructure*)>
<!ELEMENT Construction.RuleDataItemStructure
(Construction.RuleDataItemStructure.Name,

Construction.RuleDataItemStructure.ObjectID,

Construction.RuleDataItemStructure.LangName)>
<!ELEMENT Construction.RuleDataItemStructure.Name (#PCDATA)>
<!ELEMENT Construction.RuleDataItemStructure.ObjectID (#PCDATA)>
<!ELEMENT Construction.RuleDataItemStructure.LangName (#PCDATA)>
<!ELEMENT Construction.RuleExpand.Script (#PCDATA)>
]>
```

付録 I　ファイルの互換性

SEWB+/CONSTRUCTION で作成したファイルのバージョン間の互換性について説明します。

(1) SEWB+/CONSTRUCTION のファイルの互換性

各ファイルの互換性について説明します。なお，新しいバージョンの SEWB+/CONSTRUCTION で作成したファイルは，旧バージョンの SEWB+/CONSTRUCTION で使用することはできません。

- テンプレート，部品，およびプログラム定義ファイル
  SEWB+/CONSTRUCTION の新しいバージョンを利用する場合，旧バージョンの SEWB+/CONSTRUCTION で作成したテンプレート，部品，およびプログラム定義ファイルは，そのまま使用できます。
- データ定義ファイル
  データ定義ファイルについては，ファイルを作成した SEWB+/CONSTRUCTION のバージョンよりあとにサポートされた機能を利用する際，結合項目を再入力して更新する必要があります。対象となるバージョンと該当する機能については，「(2) 旧バージョンの SEWB+/CONSTRUCTION で作成したデータ定義ファイルの扱い」を参照してください。

(2) 旧バージョンの SEWB+/CONSTRUCTION で作成したデータ定義ファイルの扱い

旧バージョンの SEWB+/CONSTRUCTION で作成したデータ定義ファイルを使用する際，留意が必要となるバージョンと機能について説明します。結合項目を再入力して更新する方法については，「データ定義ファイルへの結合項目の再入力」を参照してください。

- バージョン 02-00 以前で作成したデータ定義ファイル
  SEWB+/CONSTRUCTION 02-01 でサポートされた機能のうち，次に示す機能を使用する際には，SEWB+/CONSTRUCTION 02-00 以前で作成したデータ定義ファイルの更新が必要な場合があります。
  - テンプレート記述言語の @@getdata 関数
  - テンプレート記述言語の @@modify 関数
  - テンプレート記述言語の FOR_REPOSITORY 句を指定した @@lang 文
  - 業務ルールスクリプトの @modify キーワード
- バージョン 02-03 以前で作成したデータ定義ファイル
  SEWB+/CONSTRUCTION 02-08 でサポートされた機能のうち，次に示す機能を使用する際には，SEWB+/CONSTRUCTION 02-03 以前で作成したデータ定義ファイルの更新が必要な場合があります。

- データ項目名称―標準名称表示切替機能

● バージョン 02-04 以前で作成したデータ定義ファイル

SEWB+/CONSTRUCTION 02-05 でサポートされた機能のうち，次に示す機能を使用する際には，SEWB+/CONSTRUCTION 02-04 以前で作成したデータ定義ファイルの更新が必要な場合があります。

- タイプ追加レコードを生成する機能

● バージョン 02-05 以前で作成したデータ定義ファイル

SEWB+/CONSTRUCTION 02-06 でサポートされた機能のうち，次に示す機能を使用する際には，SEWB+/CONSTRUCTION 02-05 以前で作成したデータ定義ファイルの更新が必要な場合があります。

- レコード生成キーワードに @PLEN 可変記号を指定できる機能

データ定義ファイルへの結合項目の再入力

データ定義ファイルの使用状態に応じて，次に説明する操作をしてください。

- データ定義ファイルをチェックインして使用している場合は，そのまま使用できます。
- データ定義ファイルをチェックアウトして使用している場合は，次のうち，どれかの操作が必要です。
  ・データ定義ファイルをチェックインして使用する
  ・データ定義ファイルをチェックインし，再度チェックアウトする
  ・データ定義ファイルを開いて結合項目を再入力する
- データ定義ファイルをリポジトリで管理しないで使用している場合は，次の操作が必要です。
  ・データ定義ファイルを開いて結合項目を再入力する

（3）旧バージョンの SEWB+/CONSTRUCTION で作成したプログラム定義のファイルの扱い

● バージョン 02-07 以前で作成したプログラム定義ファイル

SEWB+/CONSTRUCTION 02-08 でサポートされた機能のうち，次に示す機能を使用する際には，SEWB+/CONSTRUCTION 02-07 以前で作成したプログラム定義ファイルを開いて，データ定義ファイルの再入力をする必要があります。

- データ項目名称―標準名称表示切替機能

付録Ｊ　用語解説

C/S システム

クライアントサーバシステムの略称です。サービスを提供するサーバと，サービスを要求するクライアントから構成されます。一つの処理をネットワーク上のクライアントとサーバに分散させる業務形態です。

COBOL 開発マネージャ

ソースプログラムや画面定義ファイルなど，関連する一連のファイルをプロジェクトという単位で管理し，効率の良い AP 開発を支援するツールです。

CORBA（Common Object Request Broker Architecture）

OMG（Object Management Group）が標準化を進めている，ORB の標準仕様です。

CSV ファイル（Comma Separated Values ファイル）

リレーショナルデータベースで扱えるテキストデータを格納しているファイルです。データの区切りをコンマ（,），レコードの区切りを改行で表します。レコードは可変長形式になります。

DAM ファイル（Direct Access Method ファイル）

OpenTP1 が提供するファイルサービスの一つで，障害発生時にジャーナル情報からトランザクションの回復ができます。ファイル内の相対位置を指定し，ブロック単位でダイレクトアクセスができる直接編成ファイルです。OpenTP1 専用のユーザファイルとして使用できます。DAM ファイルにアクセスする場合，オフライン上では物理ファイルに，オンライン上では論理ファイルにアクセスします。

OpenTP1

分散トランザクション処理システムを実現するために，日立が提供している製品です。アプリケーションをネットワーク上に分散し，DB を管理するためのサーバを設ける分散システムでは，DBMS の機能だけを利用していたのでは，実現できるクライアントサーバシステムにも限界があります。そこで，複数の WS や PC（クライアント）とサーバを連携し，信頼性とパフォーマンスに優れたシステムを実現するために，オンライントランザクション処理技術が必要になってきます。OpenTP1 は，RPC 機能のほかにも，複数マシンにわたるデータ更新などの一連の処理の整合性を確保できるトランザクショナル RPC 機能，クライアントからのサービス要求を処理するスケジューリング機能，障害発生時にジャーナル情報からトランザクションの回復ができるファイルサービスなどを提供します。

RPC（Remote Procedure Call）

RPC は，同一 LAN 内や，また WAN でつながれた LAN といった水平分散上のクライアントとサーバ間で，サブプログラムの呼び出しと同じような簡単な手続きで通信ができる機能です。クライアントのプログラムがサービス提供プログラム（SPP）に対してサービス要求するときは，サービス要求に必要な RPC の項目を指定します。

SEWB+/RECORD DEFINER

SEWB+ 基本開発環境のレコード設計支援機能のことです。
SEWB+/REPOSITORY の辞書情報を使用して，レコード情報を定義できます。

SEWB+/REPORT MANAGER

SEWB+ で定義した情報の印刷を支援するツールです。SEWB+/CONSTRUCTION と連携してプログラム処理概要図を，SEWB+/REPOSITORY と連携してインパクトレポートやデータ項目または業務ルールの定義内容などを，SEWB+/RECORD DEFINER と連携してレコード定義を印刷できます。これらのドキュメントを定義内容の確認や保守作業に役立てられます。

SEWB+/REPOSITORY

SEWB+ 基本開発環境のリポジトリ管理機能のことです。
SEWB+/REPOSITORY は，リポジトリに格納された資源（ドキュメントと辞書），およびその資源間の関連をサーバ上で統合管理します。また，SEWB+/REPOSITORY と SEWB+/REPOSITORY-BROWSER を合わせて SEWB+/REPOSITORY と呼ぶこともあります。

SEWB+/REPOSITORY-BROWSER

SEWB+ 基本開発環境のリポジトリブラウザ機能のことです。
SEWB+/REPOSITORY-BROWSER を使うと，リポジトリのシステム開発資源を，クライアントでビジュアルに操作できます。SEWB+/REPOSITORY-BROWSER では，リポジトリへ辞書やドキュメントを登録したり，目的の資源を検索したり，資源間の関連を手がかりにブラウジングしたりできます。

SEWB+ 基本開発環境

次に示す機能を一つに統合したツールです。

- リポジトリ管理機能（SEWB+/REPOSITORY）
- リポジトリブラウザ機能（SEWB+/REPOSITORY-BROWSER）
- プログラム構築支援機能（SEWB+/CONSTRUCTION）
- レコード設計支援機能（SEWB+/RECORD DEFINER）

SEWB+ 基本開発環境セット

SEWB+ 基本開発環境と Groupmax ObjectServer から構成されるプログラムプロダクトです。

SEWB+ ツール

PC を使った分散開発環境で，システム開発を支援する SEWB+ シリーズのツールの総称です。

SPP（Service Providing Program）

クライアントプログラム（サービス利用プログラム :SUP）やメッセージ処理プログラム（MHP），またはほかの SPP からの RPC によって呼び出され，要求されたサービスを処理し，結果を返すサービス提供プログラムです。

SUP（Service Using Program の略称です。）

サービス提供プログラム（SPP）に対して RPC を発行し，サービスを要求するクライアントプログラム（サービス利用プログラム）です。

TAM テーブル（Table Access Method テーブル）

OpenTP1 が提供するファイルサービスの一つで，障害発生時にジャーナル情報からトランザクションの回復ができます。OpenTP1 専用のユーザファイルとして使用でき，直接編成ファイルへ高速にアクセスできます。ファイルを構成するレコードがメモリにロードされているため，実際のファイルではなくメモリへキーアクセスします。そのためアプリケーションからの高速アクセスを実現で

きます。

TP モニタ

トランザクション処理の監視，および制御をするソフトウェアのことです。オンラインシステムを構築するための基盤になる機能を提供しています。主な機能として，通信機能，スケジュール機能および障害発生時の回復機能があります。

UOC（User Own Coding）

詳細は，「ユーザ追加処理」を参照してください。

UOC リバース機能

ソースファイルで直接編集された UOC（ユーザ追加処理）を，プログラム定義ファイルの UOC に取り込む機能です。

WorkCoordinator Definer

ビジネスプロセスを定義，管理および運用するプログラムプロダクトのことです。
WorkCoordinator Definer は，ビジネスプロセス管理および案件運用操作で構成されます。

XMAP3

基幹業務で利用する画面や帳票を作成するためのツールです。XMAP3 では，画面や帳票の作成から，表示・印刷までを一貫して支援しています。なお，XMAP3 で開発した表示・印刷の AP は，PC や WS のスタンドアロン環境のほかに，C/S システム環境やホストと連携した環境でも運用できます。

XML（eXtensible Markup Language の略称です）

XML は HTML と同様の言語で，タグとテキストで構成されたデータ構造を階層化して表現できるという特徴があります。また，XML では，作成者自身がタグを定義できます。SEWB+/CONSTRUCTION では，このような特徴を持つ XML で記述された文書を，プログラム生成のためのパラメタとして利用することができます。

可変記号

テンプレート中で使用される仮の名称です。可変記号には，ソースプログラム生成時にプログラム定義ウィンドウで定義した値が設定されます。可変記号にどのような値を設定するか，プログラム作成者は定義ウィンドウ上に表示される説明文を参照します。テンプレート作成者は，定義ウィンドウ上に表示する説明文を作成します。

関連

ソースファイルとテンプレート，データ定義とデータ項目といったように，資源間に付けられた関係をいいます。関連には，SEWB+/REPOSITORY-BROWSER を使用して任意に設定できるユーザ関連と，SEWB+/CONSTRUCTION が資源間に設定する関連とがあります。SEWB+/REPOSITORY-BROWSER の関連ブラウザ機能を使用すると，関連づけられた資源をブラウジングでき，変更波及の解析などに利用できます。

業務ルール

データ項目辞書に格納されている複数のデータ項目に共通する処理を，汎用的なルールとして業務ルール辞書に格納し，データ項目と関連を持たせたものです。SEWB+/CONSTRUCTION で業務ルールを使用するときは，テンプレートに「@@rule 文」または「@@merge 文」を書いて必要な業務ルールを取り込みます。

業務ルール辞書

業務ルールを格納している辞書です。業務ルール辞書はデータ項目辞書と一緒に，辞書フォルダに格納されています。

結合項目

複数のデータ項目が連結して定義されるデータ項目です。COBOL の集団項目，C 言語の構造体の考え方に相当します。

サイン

データ定義やプログラム定義の名称，および定義の概要といった情報をまとめてサインといいます。各定義ウィンドウの [サイン] タブで記述します。

サンプルテンプレート

SEWB+/CONSTRUCTION が用意しているテンプレートです。サンプルのドキュメントフォルダの OpenTP1 用またはバッチ用の下にあります。サンプルテンプレートは例題として用意されていますが，カスタマイズするとユーザ独自の環境にも流用できるようになります。

指示項目

テンプレートの「@@interface」で宣言された可変記号を修飾する，任意の文字列のことです。テンプレート中で入出力項目と宣言されている可変記号に対する指示項目は，レコードの構成要素やキー項目のサイズおよび表名称などに当たります。テンプレート中で入出力項目以外と宣言されている可変記号に対する指示項目は，単独の項目です。

データ項目辞書

システム中の複数の AP 間で共用するデータ項目を格納する辞書です。データ中心アプローチに基づいてデータ分析し，標準化したデータ項目を蓄積します。SEWB+/CONSTRUCTION で作成する AP で使用するデータ項目は，データ項目辞書から取り込みます。

データ定義

作成する AP 中で使用されるファイルや DB などを定義します。データ定義情報からレコードを生成できます。これらの定義に必要なデータ項目は，すべてデータ項目辞書から取り込みます。データ定義で定義したレコード構造を，COPY メンバファイルやヘッダファイルとして生成する機能を「レコード生成機能」といいます。

テンプレート

テンプレートは，生成時に AP の枠組みになるプログラム構成を記述したものです。部品や UOC の挿入場所，可変項目への値の設定の指示なども記述されています。テンプレートはテンプレート記述言語と使用するプログラミング言語で記述します。テンプレート，および部品を使用すると，形式が統一されたプログラムを効率良く作成できます。だれが作成しても形式が同じなので，管理や保守がしやすくなります。

テンプレート記述言語

テンプレートを作成するための言語です。テンプレート記述言語でテンプレートを作成すると，どの言語種別にも対応できる汎用性の高いテンプレートを作成できます。

入出力項目

テンプレートの「@@interface」で，入出力資源（ファイル，DB などの形式情報）であると宣言された可変記号です。

バッチシステム

一連の処理をシーケンシャルにバッチ処理するシステムです。

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に変更があった場合，変更のあったファイルおよびそのファイルに関連のある資源だけを使用して実行ファイルを生成する機能です。

部品

複数の AP 間で利用される処理です。テンプレート記述言語と使用するプログラミング言語で記述します。部品は，ソースプログラム生成時に，テンプレートに取り込まれて展開されます。

プログラム構造図

AP の構成をツリー形式で表示したものです。テンプレート中で，ユーザ追加処理（UOC）を挿入する指示（@@UOC）や業務ルール処理を挿入する指示（@@rule）とプログラム構造図を出力する指示（@@diagram）を書くと，UOC の挿入位置を含んだプログラム構造図がプログラム定義ウィンドウ上に表示されます。複雑な階層を持つテンプレート中の，UOC 挿入位置などを確認するのに便利です。

プログラム定義

作成されたテンプレートやデータ定義情報を基にして，ソースプログラム生成に必要な情報や定義を洗い出し，それらを定義します。このとき UOC も編集できます。

マップ定義ファイル

XMAP3 で作成した画面や帳票の定義情報を格納したファイルのことです。SEWB+/CONSTRUCTION では，マップ定義ファイルを利用したクライアント AP の作成もできます。

メッセージログレコード構造

OpenTP1 を利用すると，プログラムから関数を使ってメッセージログの出力を要求し，メッセージログファイルに出力できます。また，メッセージログをリアルタイムにコンソールに出力することもできます。プログラムには，メッセージログの出力を要求したプログラムの ID やメッセージ ID，メッセージログテキストのレコード構造を指定する必要があります。

ユーザジャーナル

OpenTP1 を使用する場合，OpenTP1 のシステムジャーナルファイルに AP から出力できるユーザ任意の情報です。SEWB+/CONSTRUCTION では，ユーザジャーナルの単位になるユーザジャーナルレコード（UJ）およびその構造を定義します。

ユーザジャーナルレコード

OpenTP1 を利用すると，プログラム中のトランザクションから関数を使って，ユーザの任意の情報をジャーナルとしてシステムジャーナルファイルに出力できます。この関数 1 回の呼び出しで取得できるユーザジャーナルをユーザジャーナルレコード（UJ）と呼びます。ジャーナルを取得することによって，障害発生時などのファイル回復や稼働統計情報の分析に役立てることができます。

ユーザ処理

ユーザ追加処理の編集と，業務ルール処理の設定のことです。プログラム定義の [ユーザ処理] タブでは，テンプレートに書かれたユーザ追加処理挿入位置に展開させる，AP 特有の処理のコーディングをエディタで編集します。また，テンプレートに書かれた業務ルール処理挿入位置に展開させる

業務ルールを設定します。

ユーザ追加処理

ユーザが作成するコーディングで，主にテンプレートや部品では対応できない AP 独自仕様の部分です。ユーザ追加処理はプログラム定義ウィンドウで編集します。テンプレート中にユーザ追加処理の挿入位置を記述しておくと，プログラム定義ウィンドウ上に反映され，表示されます。ユーザ追加処理は UOC（User Own Coding）と表記されることもあります。

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に変更があった場合，プロジェクトに登録されているすべての資源に対して実行ファイルを生成する機能です。

ルールスクリプト

業務ルールの処理内容です。データ項目辞書のデータ項目に共通の処理を，可変記号などを使用して汎用的なルールとして作成し，業務ルール辞書に格納します。

レコード

データ項目辞書の「データ項目の結合」機能を利用して作成される結合項目編成の，最上位の結合項目です。

レコード生成機能

データ定義で定義したレコード構造を COPY メンバファイルまたはヘッダファイルとして生成する機能です。

レコードソース

COBOL の COPY 登録集や C のヘッダファイルのことを指します。

論理設計図

SEWB+/CS-DESIGN でオブジェクトの論理的な関係を表す図です。論理設計図では，3 層モデルでのオブジェクトの配置や，オブジェクトのオペレーションの呼び出し関係を表現できます。

索引

記号

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W

X

あ

い

え

お

か

き

け

こ

さ

し

す

せ

そ

た

ち

て

と

な

に

の

は

ひ

ふ

ま

め

も

ゆ

よ

り

る

れ

ろ

ソフトウェアマニュアルのサービス ご案内

ソフトウェアマニュアルについて，３種類のサービスをご案内します。ご活用ください。

１．マニュアル情報ホームページ

ソフトウェアマニュアルの情報をインターネットで公開しております。

URL　http://www.hitachi.co.jp/soft/manual/

ホームページのメニューは次のとおりです。

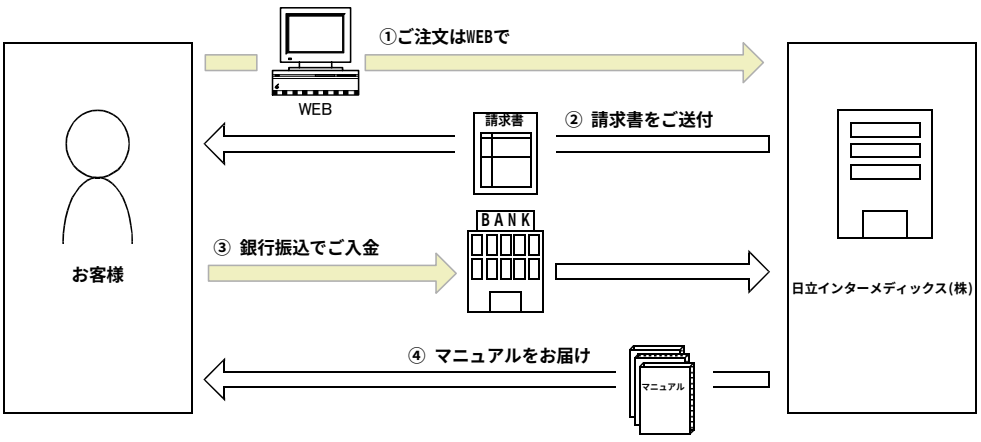
■マニュアル一覧	日立コンピュータ製品マニュアルを製品カテゴリ，マニュアル名称，資料番号のいずれかから検索できます。
■CD-ROMマニュアル情報	複数マニュアルを格納したCD-ROMマニュアルを提供しています。どの製品に対応したCD-ROMマニュアルがあるか，を参照できます。
■マニュアルのご購入	日立インターメディックス(株)の「日立コンピュータ製品マニュアルサイト」からお申し込みできます。 (詳細は「３．マニュアルのご注文」を参照してください。)
■Web提供マニュアル一覧	インターネットで参照できるマニュアルの一覧を提供しています。 (詳細は「２．インターネットからのマニュアル参照」を参照してください。)
■ご意見・お問い合わせ	マニュアルに関するご意見，ご要望をお寄せください。

２．インターネットからのマニュアル参照(ソフトウェアサポートサービス)

ソフトウェアサポートサービスの契約をしていただくと，インターネットでマニュアルを参照できます。
本サービスの対象となる契約の種別，及び参照できるマニュアルは，マニュアル情報ホームページでご確認ください。なお，ソフトウェアサポートサービスは，マニュアル参照だけでなく，対象製品に対するご質問への回答，問題解決支援，バージョン更新版の提供など，お客様のシステムの安定的な稼働のためのサービスをご提供しています。まだご契約いただいていない場合は，ぜひご契約いただくことをお勧めします。

３．マニュアルのご注文

日立インターメディックス(株)の「日立コンピュータ製品マニュアルサイト」からご注文ください。



① 下記 URL にアクセスして必要事項を入力してください。

URL　http://www2.himdx.net/manual/privacy.asp?purchase_flag=1

② ご注文いただいたマニュアルについて，請求書をお送りします。

③ 請求書の金額を指定銀行へ振り込んでください。なお，送料は弊社で負担します。

④ 入金確認後，７日以内にお届けします。在庫切れの場合は，納期を別途ご案内いたします。