ASCET V5.2

ユーザーズガイド

著作権について

Document EC010001 R5.2.2 JP

目次

1 はじめに

本書『ASCET V5.2 ユーザーズガイド』には、ASCET の操作方法についてのすべての情報が含まれています。ASCET を初めて使われる方は、本書の前に『ASCET V5.2 入門ガイド』の「ASCET を理解する」、および「チュートリアル」の章を先にお読みいただき、ASCET の概要をご理解いただくことをお勧めします。本書においては、ASCET を使って実際にコンポーネントやプロジェクトを作成して実験を行う、という一連の作業における各工程の操作が、各章ごとに順に説明されています。

各章の構成はどれも共通しています。章のタイトルに続けて短い前書きがあり、次にユーザーインターフェースやメニューコマンドについての解説がまとめられ、最後に一般的なワークフローに沿った操作手順が説明されています。

1.1 典型的なワークフロー

ASCET で開発を行うには、ASCET 独自のワークフローに基いて作業を進める必要があります。たとえば、ある過程においては必ず規定のシーケンスでダイアログボックスが開く個所などもあり、このような規定された手順に沿って作業することによって効率的なワークフローが実現されます。実際には、一般のような順で作業を進めます。

- ASCET をセットアップする
- データベースとそのアイテムを作成する
- 必要なファンクションを追加する
- コンポーネントを記述してシミュレーションを行う
- プロジェクトを構築してシミュレーションを行う
- データセットを作成して編集する
- インプリメンテーション（コード実装情報）を作成して編集する
- シグナルとアイコンを作成する
- シミュレーション実験を行う
- 結果のドキュメントを作成する

1.2 ASCET の起動

ASCET を起動する：

- ASCET を PC にインストールすると、デフォルト状態においてはデスクトップ上に ASCET のアイコンが作成されます。このアイコンをダブルクリックしてプログラムを起動してください。
- また、Windows の **スタート** メニューからも ASCET を起動できます。

ASCET が起動すると、ASCET のメインウィンドウである「コンポーネントマネージャ」が開きます。

注記

ASCET のユーザーインターフェースについて、各種ファイル用のパス、表示色やフォント、インポート／エクスポートオプションなどを独自に設定することができます。詳しくは、2.2「コンポーネントマネージャ - ASCET をセットアップする」の項を参照してください。

2 コンポーネントマネージャ

この章では、ASCET のメインウィンドウである「コンポーネントマネージャ」の操作方法について説明します。このウィンドウは、ASCET の従来のバージョンの「スタートウィンドウ」と「データベースブラウザ」の両方の機能を併せ持つものです。

ここにはすべてのメニューコマンドについて説明も含まれています。ASCET を初めて使われる方は、本書の前に『ASCET 入門ガイド』の「ASCET を理解する」、および「チュートリアル」の章を先にお読みいただき、ASCET の概要をご理解いただくことをお勧めします。

2.1 コンポーネントマネージャ - ユーザーインターフェースの概要

2.1.1 概要

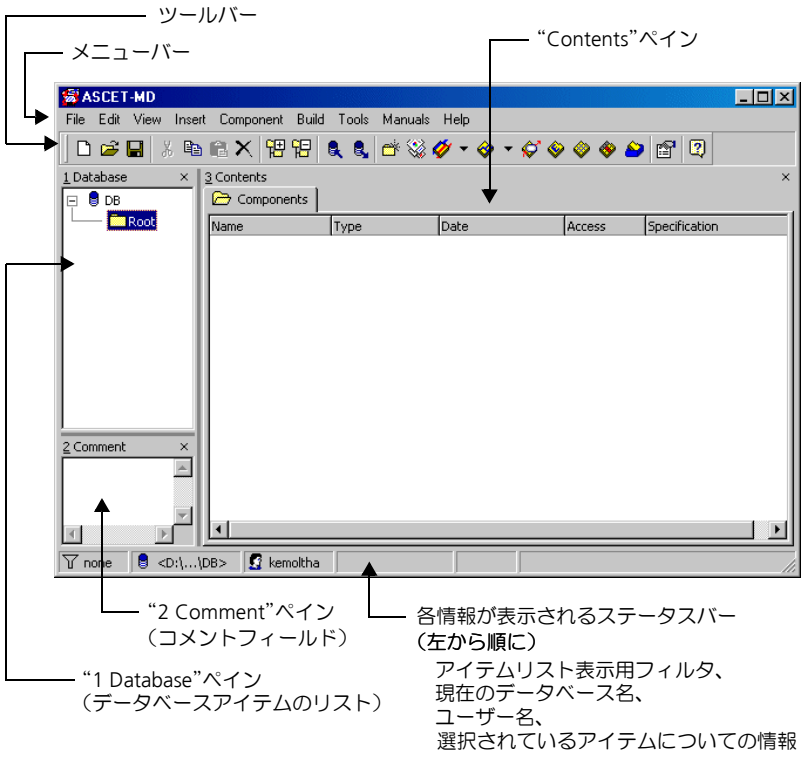
ASCET コンポーネントマネージャは、ASCET を起動すると自動的に開き、このウィンドウから各種エディタを開いて作業を行います（詳しくは第 4 章 「コンポーネントとプロジェクトの定義」を参照してください）。またこのウィンドウでは、ユーザープロファイルを管理したり、オプション設定用ダイアログボックス各種オプション（ファイルを格納するディレクトリ、画面表示、ソフトウェアの起動条件など）を自由に設定することができます（37 ページの「コンポーネントマネージャ - ASCET をセットアップする」を参照してください）。

そしてコンポーネントマネージャの最も重要な機能として、ASCET で作成されたデータをデータベースとして構造的に管理します（74 ページの「コンポーネントマネージャ - データの管理」を参照してください）。

本章ではコンポーネントマネージャウィンドウで直接行う操作に関して説明されていて、個別の操作ウィンドウ（各種エディタ、実験環境など）やその他の追加機能などについては、それぞれ別の章に説明されています。

2.1.2 コンポーネントマネージャのユーザーインターフェース

ASCET のデータベースに含まれる各種アイテム（「データベースアイテム」とも呼ばれ、74 ページの「データベースアイテム」の項に詳しく説明されています）は、すべてコンポーネントマネージャで管理されます。ここで、データベース内にフォルダやサブフォルダを作成したり、各種アイテムの作成、移動、コピー、インポート、エクスポートなどを行います。また、旧バージョンにおいては ASCET のプロジェクトやコンポーネントの編集はすべてそれぞれのエディタでしか行えませんでしたが、現バージョンではそれらをコンポーネントマネージャから直接編集することも可能です。



“1 Database”リストには、現在のデータベースに含まれるフォルダとアイテムの一覧がツリー構造で表示されます。データベース名はツリー構造の最上階に表示されます。各データベースには 1 つまたは複数のフォルダが含まれ、各フォルダにはさらにサブフォルダやデータベースアイテムが含まれます。これらすべてが“1 Database”リストに表示され、ここで管理されます。

“2 Comment”ペインはテキストフィールドで、そこには現在選択されているフォルダまたはアイテムについてのメモや内部コメントが表示されます。一般に、このフィールドは開発者のコメントや内部のバージョン情報に用いられます。こ

こに含まれる情報は、自動作成されるドキュメントに出力される注釈（“Notes Editor”で編集します）とは異なります（675 ページの 7.4「注釈」の章を参照してください）。

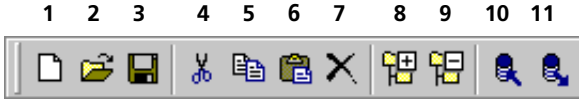
注記

エクスポート処理（90 ページの 2.3.3 項を参照してください）の際、“2 Comment”フィールドのコメントがエクスポートされるのはトップレベルのフォルダとアイテムのみです。**サブフォルダ内のコンポーネントのコメントはエクスポートされません。**

“3 Contents”ペインの各タブ上には、選択されているフォルダまたはデータベースアイテムについての情報が表示されます。表示される内容は、現在選択されているオブジェクト（フォルダまたは各種データベースアイテム）によって異なります。詳しくは 2.3.2 項の「各種ビューモードを利用した作業」を参照してください。

各ツールバーに含まれる操作ボタンの機能説明

デフォルトツールバー：



1. New - 新規作成（新しいデータベースの作成）
2. Open - 開く
3. Save - 保存
4. Cut - 切り取り
5. Copy - コピー
6. Paste - 貼り付け
7. Delete - 削除
8. Expand all - すべて展開
9. Collapse all - すべて省略
10. Import - インポート
11. Export - エクスポート

“Insert”ツールバー：

1. Insert Folder - フォルダの作成

2. Insert Project - プロジェクトの作成
3. Insert Module - <Type> - モジュールの作成 < タイプ >
4. Insert Class - <Type> - クラスの作成 < タイプ >
   矢印ボタン（▼）3a および 4a でオブジェクトタイプ（BDM、C コード、ESDL）を選択します。
5. Insert State Machine - ステートマシンの作成
6. Insert Enumeration - 列挙型の作成
7. Insert Boolean Table - 論理テーブルの作成
8. Insert Conditional Table - 条件テーブルの作成
9. Insert Container - コンテナの作成

“Tools”ツールバー：

1 2

1. Options - オプション
2. ? - “About ASCET”ウィンドウ を開く
   インストールされている ASCET 製品ファミリについての情報が表示されます。

データベースアイテム用シンボル

1 2 3 4 5 6 7 8 9 10 11 12 13

1. データベース
2. プロジェクト
3. モジュール
4. クラス
5. CT ブロック
6. ステートマシン
7. 列挙型
8. 論理テーブル
9. 条件テーブル
10. アイコン
11. シグナル
12. コンテナ
13. ASAM-MCD-2MC プロジェクト

- **File**
  - *New Database* （**<Ctrl> + <N>**）
    新しいデータベースを作成します。
  - *Open Database* （**<Ctrl> + <O>**）
    データベースを開きます。
  - *Close Database*
    データベースを閉じます。
  - *Save Database*
    データベースの変更内容を上書き保存します。
  - *Save Database as...*
    データベースを、別名で任意のパスに保存します。
  - *Import* （**<Ctrl> + <M>**）
    エクスポートファイルから“1 Database”リスト内にデータベースアイテムをインポートします。
  - *Import directory...*
    指定のディレクトリの内容をすべてインポートします。
  - *Export* （**<Ctrl> + <E>**）

    注記

    この機能は、ASCET-MD がインストールされている場合にのみ使用可能です。その際、同じコマンドを“1 Database”リスト内のショートカットメニューから実行することもできます。

    “1 Database”リスト内の任意のアイテムまたはフォルダを、1 つまたは複数のエクスポートファイル（`*.exp` フォーマット）にエクスポートします。
  - *1 D:¥...¥DB* （この形式でデータベース名とそのパスが表示されます）
    最近 ASCET で使用されたデータベースの一覧です。
  - *Exit* （**<Alt> + <F4>**）
    ASCET を終了します。

- **Edit**

> **注記**
>
> 以下の **Cut**、**Copy**、**Paste**、**Delete**、**Rename** の各コマンドは、“1 Database”リスト内のショートカットメニューから実行することもできます。また **Find** と **Query** は、ASCET-MD がインストールされている場合のみ使用できます。

  — *Cut* （**<Ctrl> + <X>**）

  選択されているデータベースアイテムを切り取ってクリップボードに移動します。

  — *Copy* （**<Ctrl> + <C>**）

  選択されているデータベースアイテムをクリップボードにコピーします

  — *Paste* （**<Ctrl> + <V>**）

  選択されているフォルダにクリップボードの内容を貼り付けます。

  — *Delete* （**<Del>**）

  選択されているデータベースアイテムを完全に削除します。フォルダを削除すると、その下に含まれるフォルダとアイテムもすべて削除されます。

  > **注記**
  >
  > **Edit → Delete** を実行すると、取り消しができません。ご注意ください。

  — *Rename*（**<F2>**）

  選択されているデータベースアイテムの名前を変更します。

  — *Find* （**<Ctrl> + <F>**）

  C コードコンポーネントまたは ESDL コンポーネントに含まれる文字列を検索します。

  — *Replace* （**<Ctrl> + <H>**）

  C コードコンポーネントまたは ESDL コンポーネントに含まれる文字列を置換します。

  — *Query* （**<Ctrl> + <Q>**）

  データベース内の各種アイテムを検索します（132 ページの「データベースのブラウズ」を参照してください）。

- **View**

— *Expand all*

アイテムリストの階層を展開し、データベースの全階層のフォルダとアイテムをすべて表示します。

— *Collapse all*

アイテムリストの階層を省略し、最上位の階層の内容のみを表示します。

— *Filter →< コンポーネントタイプ>*

“1 Database”リストに表示されるデータベースアイテムをフィルタリングします。

— *Show/Hide*

コンポーネントマネージャウィンドウ内の各ペインの表示／非表示を切り替えます。

→ *Database List* — “1 Database”リスト
→ *Comment* — “2 Comment”ペイン
→ *Contents* — “3 Contents”ペイン
→ *Monitor* — モニタウィンドウ

— *Update* （**<F5>**）

コンポーネントマネージャウィンドウの表示内容を更新します。

- **Insert**

注記

Insert メニューのコマンドは、“1 Database”リスト内のショートカットメューからからでも実行可能です。またこのメニューは ASCET-MD がインストールされている場合にのみ有効です。

— *Folder* （**<Ins>**）

新しいフォルダを追加します。

— *Project*

新しいプロジェクトを追加します。

— *Module*

新しいモジュールを追加します。

→ *Block Diagram* — ブロックダイアグラム
→ *ESDL* — ESDL で記述されるモジュール
→ *C Code* — C コードで記述されるモジュール

— *Class →< モジュールタイプ>*

新しいクラスを追加します。サブメニューの内容は *Module* と同様です。

- *Continuous Time Block →< モジュールタイプ>*

  新しい CT ブロックを追加します。サブメニューの内容は *Module* と同様です。

- *State Machine*

  新しいステートマシンを追加します。

- *Enumeration*

  新しい列挙型（列挙型の宣言テーブル）を追加します。

- *Boolean Table*

  新しい論理テーブルを追加します。

- *Conditional Table*

  新しい条件テーブル（4.7 項を参照してください）を追加します。

- *Icon*

  新しいアイコンを追加します。

- *Signal*

  新しいシグナルを追加します。

- *Container*

  新しいコンテナを追加します。

- **Component**

注記

Component メニューの **Edit Item** コマンドは、“1 Database”リスト内のショートカットメニューからも実行できます。
また **Edit Layout**、**Replace References**、**Become another Item**、**Reproduce As** コマンドは、ASCET-MD がインストールされている場合にのみ有効です。

- *Edit Item*（**<Enter>**）

  選択されているデータベースアイテム用のエディタを開きます。

- *Edit Layout*

  コンポーネントのレイアウトを編集します。

- *Enumeration*

  “1 Database”リストで列挙型が選択されていて、かつ“3 Contents”ペインがアクティブになっている場合にのみ有効です。詳しくは、28 ページを参照してください。

- *Disallow Import*

  インポート処理によるアイテムの上書きを禁止します。

— *Access Rights*

フォルダまたはアイテムのアクセス権の設定を変更します。

— *Password*

フォルダのパスワード保護のオン／オフを切り替えます。

— *Show References*

アイテムへの参照情報を表示します。

— *Replace References*

アイテムへの参照情報を置き換えます。

— *Become another Item*

データベースアイテムを置き換えます。

— *Reproduce As*

コンポーネントタイプを置き換えたコピーを作成します。

→ *Block Diagram* — ブロックダイアグラム
→ *ESDL* — ESDL で記述されるモジュール
→ *C Code* — C コードで記述されるモジュール

— *Notes*

データベースアイテムの注釈を編集します。

— *Reset Operator Implementation*

演算子インプリメンテーションをリセットします。
→ *Flat* — 編集対象のコンポーネントのみ
→ *Recursive* — 被参照コンポーネントも含める

- **Build**

  — *Touch All*

  すべてのデータベースアイテムに内部的な「変更済み」マークを付け、プロジェクト全体が強制的にコンパイルされるようにします。

  — *Clean All*

  データベース内に格納されている生成済みコードを消去します。

- **Tools**

  — *Documentation → Contents*

  “Documentation Contents”ダイアログボックスが開きます。ここでドキュメントに出力されるアイテムを選択します。

  — *Documentation → Options*

  ドキュメント生成オプションを設定します。

— *Database → Performance Utilities*

“Database Info for: <Database>”ダイアログボックスが開きます。ここで、以下の 4 通りのデータベース管理機能を実行できます。各オプションは組み合わせて実行することも可能です。

Optimize は、頻繁な編集によってデータベース内に生じた断片の結合（デフラグメンテーション）を行います。

Convert は、データベース構成の変更によって不要になったデータレコード間の内部参照情報（つまり、「X が Y を参照する」という情報）をクリーンアップして、アクセス速度を向上させます。

Repair は、**Optimize** および **Convert** 機能を利用してデータベースをすべて再構成（「Rebuild」）します。

Check は、データベース内の構成と参照関係をチェックし、その結果をモニタウィンドウに出力します。

— *Database → Convert*

データベースを変換します。

→ *Modify components to force impl/data update* ーすべてのコンポーネントを「modified」ステートにします。これにより、次に各コンポーネントがアクセスされる際、そのインプリメンテーションがチェックされ、必要に応じて調整されます。

→ *All Names to ANSI-C* ー データベース内のすべての名前を ANSI-C 準拠の名前に変換します。

→ *System Constants to Constants* ー システム定数を定数に変換します（『ASCET リファレンスガイド』の「エレメントの種類」の項を参照してください）

→ *Variable to Volatile* ー データベース内のすべての変数に *Volatile* 属性（112 ページと 443 ページを参照してください）を与えます。

→ *Variable to Nonvolatile* ー データベース内のすべての変数に *Non-Volatile* 属性を与えます。

→ *Components/Elements to default memory location* ー データベース内のすべてのエレメントについて、インプリメンテーションのメモリロケーションを Default に設定します。

→ *Operator Implementations to Impl. Casts* ー 存在する演算子インプリメンテーションをインプリメンテーションキャスト（497 ページ参照）に変換します。

→ *Reset Operator Implementations* ー データベース内のすべての演算子インプリメンテーションを削除します。

— *Database → List Operator Implementation*

演算子インプリメンテーションの一覧を ASCET モニタウィンドウに表示します。

— *Database → Compare Database*

現在のデータベースの内容を、別のデータベースを比較します。

— *Arithmetic Service Editor*

ターゲットに応じた算術演算サービス用エディタを開きます（4.14 項を参照してください）。

→ *>PC<* — PC ターゲット
ASCET-RP や ASCET-SE がインストールされていると、他のターゲットも追加されます。

— *Network Settings*

ETAS ネットワークマネージャの“ETAS ハードウェア用のネットワーク設定”ウィンドウを開きます。ASCET-RP 以外では使用しません。

— *Views*

ビューを管理するための“Views”ダイアログボックスを開きます（7.3 項を参照してください）。

— *Options*

各種オプション設定を行うための“Options”ウィンドウを開きます（37 ページの 2.2 項を参照してください）。

- **Manuals**

注記

ASCET インストール時にマニュアルをインストールしなかった場合、このメニューにはメニューコマンドが表示されません。

注記

日本語版マニュアルについてのメニューコマンドは用意されていません。

— *English user's guide*

英語版のユーザーズガイドを Acrobat Reader で開きます。

— *English reference*

英語版のリファレンスマニュアルを Acrobat Reader で開きます。

— *German user's guide*

ドイツ語版のユーザーズガイドを Acrobat Reader で開きます。

— *German reference*

ドイツ語版のリファレンスマニュアルを Acrobat Reader で開きます。

— *Open manuals folder*

`ETAS¥ETASManuals` ディレクトリを開きます。

ASCET-RP または ASCET-SE がインストールされていると、それぞれのマニュアル用メニューコマンドが表示されます。

- **Help**
  - — *Contents*

    ASCET オンラインヘルプのコンテンツを開きます。
  - — *Index*

    ASCET オンラインヘルプのインデックスを開きます。
  - — *Loaded Packages*

    インストールされている ASCET モジュールの一覧をモニタウィンドウに表示します（『ASCET 入門ガイド』の「ASCET ファミリ」の項を参照してください）。
  - — *Loaded Targets...*

    インストールされているターゲット別 ASCET-SE の一覧をモニタウィンドウに表示します。
  - — *Product Disclaimer*

    免責条項のウィンドウが開き、ASCET を使用する際の安全に関する注意事項が表示されます。
  - — *Problem Report*

    障害レポート作成／送信ツールを起動します（2.1.4 項を参照してください）。
  - — *Hotkey Assignment*

    キーボードショートカット（キーボードコマンド）の一覧を表示します。
  - — *About*

    インストールされている ASCET 製品ファミリについての概要を表示します。
  - — *Support*

    ASCET ホットラインの窓口の一覧を表示します。
  - — *License Info*

    ライセンス情報を表示します（『ASCET 入門ガイド』の「ライセンスについて」の章を参照してください）。

“3 Contents” ペインの表示内容に依存するメニューコマンドの概要

“3 Contents” ペインがアクティブになっている場合、**Edit** メニューと **Component** メニューに含まれる各コマンドの機能は、現在 “3 Contents” ペインに表示されている内容によって異なります。この項では、“3 Contents” ペイン内

の各タブがアクティブになっている時、および“3 Contents”ペインに列挙型のラベルリストが表示されている時の各コマンドの機能を説明します。なおここに説明されている各コマンドは、“3 Contents”ペインのショートカットメニューからでも実行できます。

注記

これら 2 つのメニューのコマンドのうち以下に記載されていないものは、常に同じ機能を持ちます。

“Elements”タブがアクティブになっている場合：

- **Edit**
  - *Copy*（**<Ctrl> + <C>**）
    選択されているアイテムをクリップボードにコピーします
  - *Paste*（**<Ctrl> + <V>**）
    クリップボードにコピーされているエレメントをコンポーネントに貼り付けます。
  - *Delete*（**<Del>**）
    選択されているエレメントをコンポーネントから削除します。
  - *Rename*（**<F2>**）
    選択されているエレメントの名前を変更します。
- **Component**
  - *Edit Item*（**<Enter>**）
    選択されているエレメントをエレメントエディタで開きます。

“Data”タブがアクティブになっている場合：

- **Edit**
  - *Copy*（**<Ctrl> + <C>**）
    選択されているエレメントのデータ（値）をクリップボードにコピーします
  - *Paste*（**<Ctrl> + <V>**）
    クリップボードにコピーされているデータを選択されているエレメントに貼り付けます。
  - *Delete*（**<Del>**）
    選択されているエレメントをコンポーネントから削除します。
  - *Rename*（**<F2>**）
    選択されているエレメントの名前を変更します。
- **Component**

— *Edit Item*（**<Enter>**）
選択されているエレメントをデータエディタで開きます。

“Implementation”タブがアクティブになっている場合：

- **Edit**
  - — *Copy*（**<Ctrl> + <C>**）
    選択されているエレメントのインプリメンテーションをクリップボードにコピーします
  - — *Paste*（**<Ctrl> + <V>**）
    クリップボードにコピーされているインプリメンテーションを選択されているエレメントに貼り付けます。
  - — *Delete*（**<Del>**）
    選択されているエレメントをコンポーネントから削除します。
  - — *Rename*（**<F2>**）
    選択されているエレメントの名前を変更します。
- **Component**
  - — *Edit Item*（**<Enter>**）
    選択されているエレメントをインプリメンテーションエディタで開きます。

“1 Database”ペインで列挙型が選択されていて、かつ“3 Contents”ペインがアクティブになっている場合：

- **Edit**
  - — *Delete*（**<Del>**）
    選択されている列挙型をコンポーネントから削除します。
  - — *Rename*（**<F2>**）
    選択されている列挙型の名前を変更します。
- **Component**
  - — *Enumeration → Add*（**<Insert>**）
    列挙型のラベルを追加します。
  - — *Enumeration → Shift Up*（**<Ctrl> + <U>**）
    選択されているラベルをラベルテーブル内で上に移動します。
  - — *Enumeration → Shift Down*（**<Ctrl> + <D>**）
    選択されているラベルをラベルテーブル内で下に移動します。

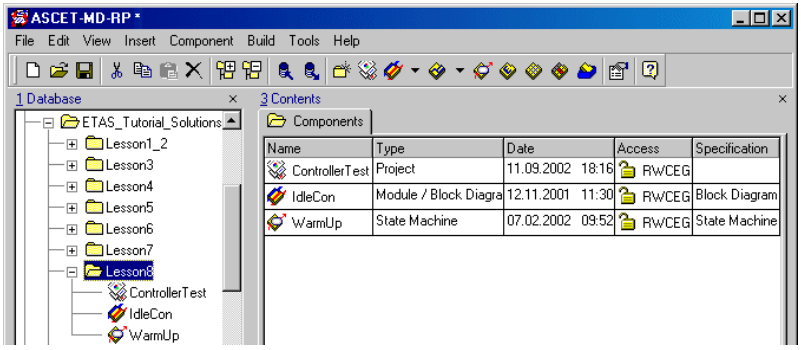
2.1.3 コンポーネントマネージャのビューモード

ASCET には、データベースアイテムを表示するための強力な表示機能が用意されていて、フォルダ、コンポーネント、プロジェクト、コンテナ、列挙型といったアイテムのタイプ（74 ページの「データベースアイテム」を参照してください）に応じた内容が“3 Contents”ペインに表示されます。コンポーネントとプロジェクトの場合は、“3 Contents”の各タブにエレメント、データ、インプリメンテーションが表示され、コンポーネントの場合はさらにレイアウトが表示されます。またフォルダの場合は、フォルダに含まれるアイテムとそれらに関する情報が表示され、コンテナの場合もフォルダとほぼ同じ内容が表示されます。これらの各表示モードは、表示される内容により、「エレメントビュー」、「データビュー」、「インプリメンテーションビュー」、「コンテナビュー」、「レイアウトビュー」などと呼ばれます。

各タブのいつくかの項目は、**Edit** および **Component** メニュー、または“3 Contents”ペイン内のショートカットメニューを利用して編集するこができます。

フォルダビューを選択する：

- “1 Database”ペインのフォルダを強調表示します。



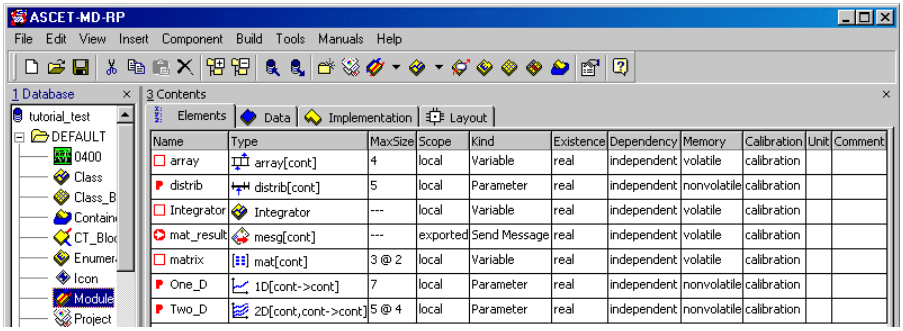
そのフォルダの内容、名前、タイプ、作成日時、アクセス権、およびコンポーネントの記述タイプが表示されます。

このタブ上での作業方法は、82 ページの「コンポーネントマネージャでのコンポーネントとプロジェクトの編集」の項に詳しく説明されています。

コンポーネントのエレメントビューを選択する：

- “1 Database”ペインのコンポーネントまたはプロジェクトを強調表示します。

- “3 Contents”ペインの“Elements”タブをクリックして、エレメントビューに切り替えます。



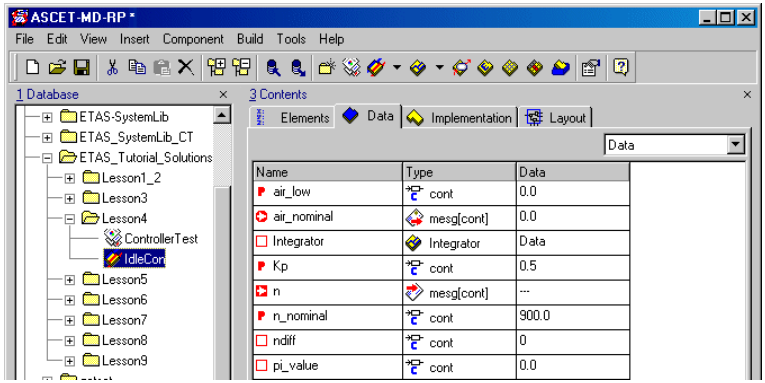
コンポーネントまたはプロジェクトに含まれるすべてのエレメントと、それらの名前、タイプ、スコープ、種類、単位などが表示されます。エレメントのコメント（443 ページ参照）が編集されている場合はその内容も表示されます。
配列、マトリックス、カーブ／マップの場合は、それらの最大許容サイズも表示されます。

各エレメントの編集方法は、82 ページの「コンポーネントマネージャでのコンポーネントとプロジェクトの編集」の項に詳しく説明されています。

データビューを選択する：

- “1 Database”ペインのコンポーネントまたはプロジェクトを強調表示します。

- “3 Contents”ペインの“Data”タブをクリックして、データビューに切り替えます。



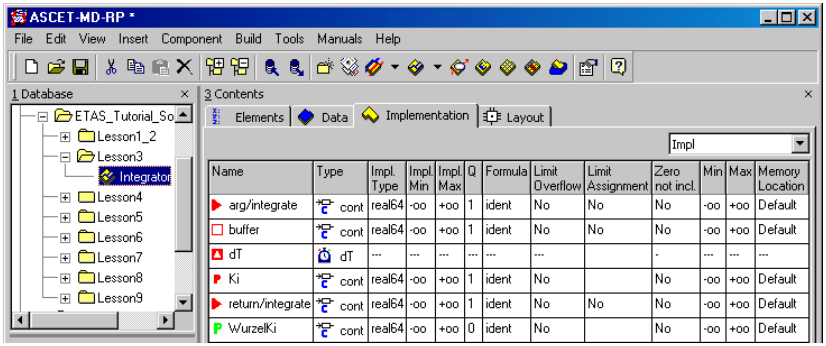
エレメントの名前、タイプ、およびデータが表示されます。コンボボックスで、他のデータセットに切り替えることができます。

各データの編集方法は、82 ページの「コンポーネントマネージャでのコンポーネントとプロジェクトの編集」の項に詳しく説明されています。

インプリメンテーションビューを選択する：

- “1 Database”ペインのコンポーネントまたはプロジェクトを強調表示します。

- “3 Contents”ペインの“Implementation”タブをクリックして、インプリメンテーションビューに切り替えます。



現在のインプリメンテーションについての全情報が表示されます。コンボボックスで、別のインプリメンテーションに切り替えることができます。

各データの編集方法は、82 ページの「コンポーネントマネージャでのコンポーネントとプロジェクトの編集」の項に詳しく説明されています。

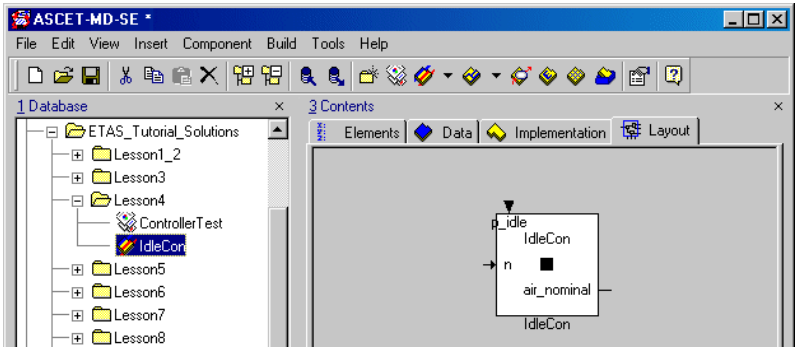
コンポーネントのレイアウトビューを選択する：

- “1 Database”ペインのコンポーネントを強調表示します。

注記

プロジェクトにはレイアウトは含まれません。

- “3 Contents”ペインの“Layout”タブをクリックして、レイアウトビューに切り替えます。

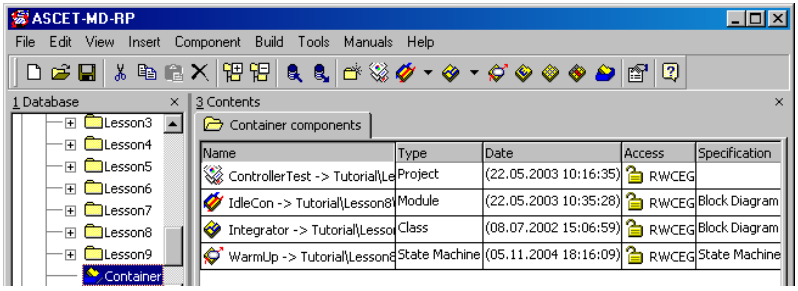


コンポーネントのレイアウトが表示されます。

レイアウトの編集方法は、82 ページの「コンポーネントマネージャでのコンポーネントとプロジェクトの編集」の項に詳しく説明されています。

コンテナビューを選択する：

- “1 Database”ペインのコンテナを強調表示します。

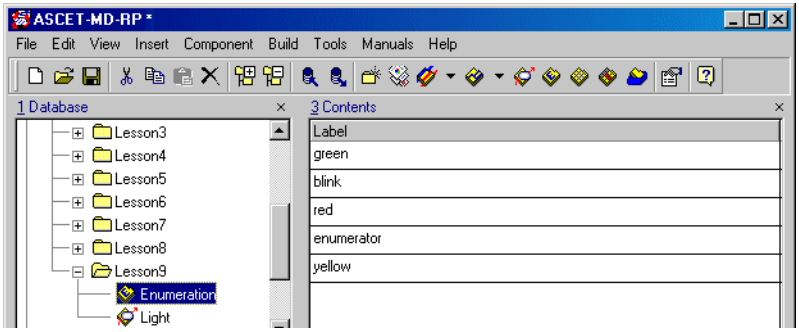


コンテナに含まれるアイテムの名前、タイプ、作成日時、アクセス権、およびコンポーネントの記述タイプが表示されます。

このタブ上での作業方法は、435 ページの「コンテナの扱い」の項に詳しく説明されています。

列挙型ビューを選択する：

- “1 Database”ペインの列挙型を強調表示します。
  “3 Contents”ペインに列挙型のラベルリストが表示されます。



列挙型の編集方法は、77 ページの「列挙型を作成する：」の項に詳しく説明されています。

2.1.4 ETAS のユーザーサポート機能 - 障害レポートの作成／送信

ASCET には、ASCET での作業中に発生した問題点に関するレポートを ETAS に送るためのユーザーサポート機能が用意されています。この機能を実行すると、ASCET はログディレクトリに含まれるすべての `*.log` ファイルと注釈文を圧縮し、`...¥ETAS¥LogFiles¥` ディレクトリに `EtasLogFiles00.zip` というアーカイブファイルとして保存します。この機能を繰り返し実行すると、このファイル名に含まれる番号が `00` から `19` まで順位にインクリメントされるので、既存のファイルはすぐには上書きされません。

注記

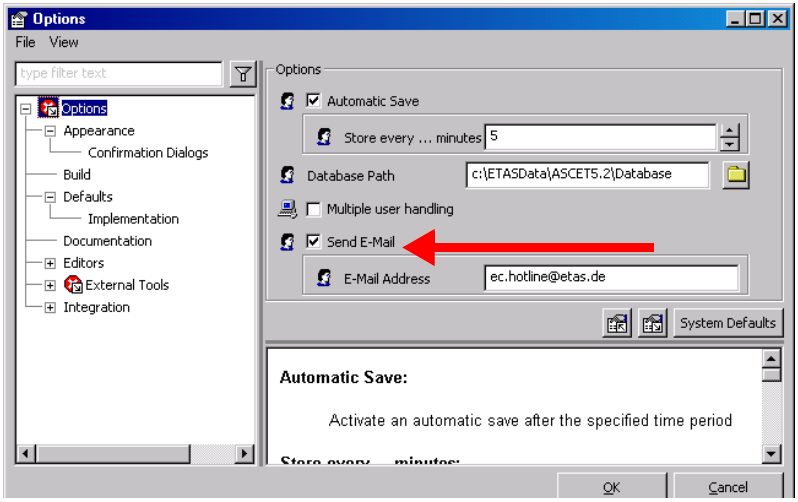
20 個以上のアーカイブファイルが作成されると、ファイル名の番号は再び 00 に戻るため、既存の同名のファイルが上書きされます。

ASCET のオプション設定で、このアーカイブファイルが ETAS ホットライン（ETAS のユーザーサポート窓口）に自動的に送信されるようにすることができます。このためには、MAPI に準拠した E メールアプリケーション（MS Exchange など）が必要です。その他のアプリケーションをお使いの場合は、このアーカイブファイルをマニュアル操作で E メールに添付してお送りください。

障害レポート機能をセットアップする：

- コンポーネントマネージャで、**Tools → Options** メニューコマンドを選択します。
  “Options”ウィンドウが開きます。

- “Options”ノードを開き、**Send E-Mail** オプションをオンにします。



ETAS ホットラインへの E メールの自動送信機能が有効になります。
使用しているメールアプリケーションが MAPI 準拠でない場合には、このオプションは無効です。

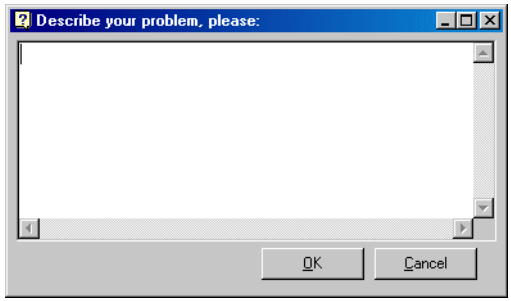
- ホットラインの E メールアドレスを修正する必要がある場合には、“E-Mail Address”フィールドをクリックします。イータス株式会社の組み込みシステム開発ツールのホットラインの E メールアドレスは以下のとおりです。

  `ec.hotline@etas.co.jp`

トラブルレポートを送信する：

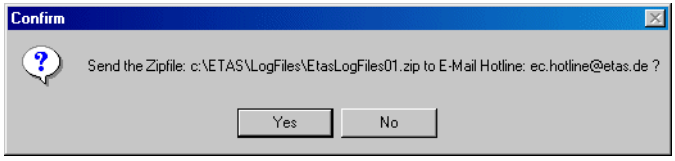
- ETAS にレポートを送信するには、コンポーネントマネージャから **Help → Problem Report** を選択します。

  トラブルについて説明するテキストを入力するためのダイアログボックスが開きます。



- 説明文を入力してから **OK** をクリックします。

  製品に登録されているライセンスがチェックされます。登録されているライセンスが見つかると、テキストと共に必要なログファイルがすべて圧縮され、アーカイブファイルが作成されます。自動送信機能が有効になっていれば、アーカイブファイルは自動的に送信されます。そうでない場合には、このアーカイブファイルを送信するかどうかが尋ねられます。



- ファイルを直ちに送信する場合は **Yes** をクリックし、そうでない場合、または MAPI 準拠のメールアプリケーションを使用していない場合は **No** をクリックします。

2.2 コンポーネントマネージャ - ASCET をセットアップする

本項では、ASCET のコンフィギュレーションオプションの内容と、ユーザープロファイルの扱いについて説明します。

注記

本マニュアルには、ASCET-MD のオプションに関してのみ記述されています。他のアドオン製品（ASCET-RP など）をインストールすると、各アドオン用の専用オプションが追加されますが、それらのオプションについては各アドオンのマニュアルを参照してください。

ASCET にはさまざまなオプションが用意されていて、各オプションは、全ユーザーに適用される「共通オプション」と、各ユーザーに固有な「ユーザーオプション」に分類されます。どちらのオプションも“Options”ウィンドウで管理され、XML ファイルに保存されます。この XML ファイルは、ASCET のインストール後に初めて起動した時点においてはまだ存在しません。このファイルは ASCET によって自動生成され、システムのデフォルト設定が保存されます。旧バージョンがインストールされている状態で ASCET をインストールした場合は、旧バージョンの設定がそのまま保存されます。

「共通オプション」（  アイコン）は、1 台のワークステーション（PC）にインストールされた ASCET について共通して使用されるオプションです。コード生成などに使用される各種デフォルトディレクトリやプレビューの設定、また ASCET 実行時のユーザーモード（シングルユーザーまたはマルチユーザー）の選択などが含まれます。

共通オプションは、`stationSettings.xml` というファイルに保存されます。

「ユーザーオプション」（  アイコン）は、ダイアグラム、エクスポート／インポートファイルのパスやその他の一般的なオプションの設定など、さまざまなオプションで構成されています。これらのオプションによって、ユーザーごとに異なる作業環境をカスタマイズすることができます。ASCET 起動時にマルチユーザーモードになっていると、ユーザーオプションの設定内容はユーザーごとに格納されます。新しいユーザーの場合は各オプションともデフォルト設定が使用されます。

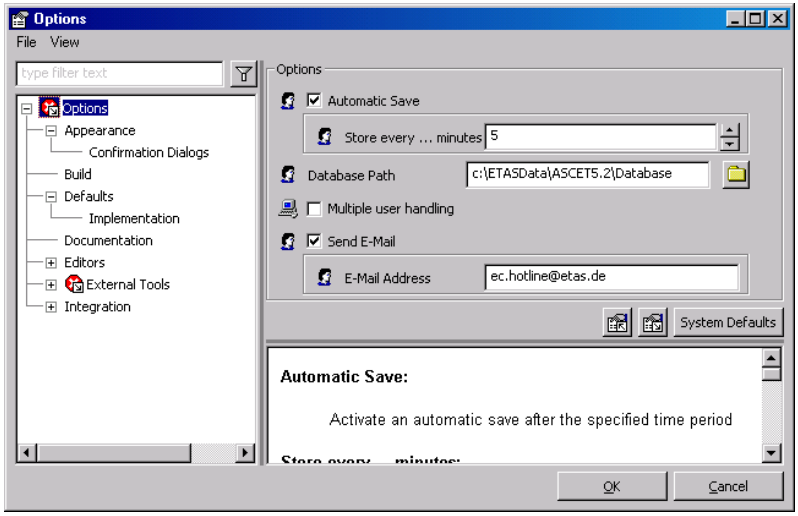
ユーザーオプションは、各ユーザーのディレクトリ（`ETASData¥ASCET5.2¥User¥<username>`）内の `userSettings.xml` というファイルに保存されます。

注記

ユーザーオプションがユーザーごとに格納されるようにするためには、あらかじめ、ASCET をマルチユーザーモードに設定しておく必要があります。

無効な値が設定されているオプションには、赤い×印のついたアイコン（  ）が表示されます。

ASCET オプションウィンドウ： ASCET のオプションは、以下の“Options”ウィンドウに表示されます。



このダイアログボックスの表示内容は以下のとおりです。

- 左側（上部）：オプションアイテムのフィルタを指定できます。
- 左側：オプションのツリービューが表示されます。各オプショングループがそれぞれツリーの「ノード」として表示されます。

  ノードにシンボル（ または ）が表示されている場合、そのノードまたはそのサブノードに無効なオプションアイテムが含まれていることを示しています。
- 右側上部：選択されているノード（オプショングループ）に含まれるオプションが表示されます。
- 右側下部：上部に表示されている各オプションについての簡単な説明が表示されます。
- ボタン：**Import Options for selected Node from XML File**、**Export Options of selected Node into XML File**、**System Defaults**
- **File** メニュー：以下のコマンドが含まれます。
  - **Import**

    選択されているノードのオプションの値を XML ファイルからインポートします。
  - **Export**

    選択されているノードのオプションの値を XML ファイルにエクスポートします。
- **View** メニュー：以下のコマンドが含まれます。

— **Expand All**

ツリービューをすべて展開表示します。

— **Collapse All**

ツリービューをすべて省略表示します。展開されているノードがすべて折りたたまれます。

— **Show/Hide Descrptiohn**

オプションについての説明文の表示を、オン／オフします。

ASCET のオプションを設定する：

- コンポーネントマネージャから、**Tools** → **Options** を選択します。

または

- **Options** ボタンをクリックします。

  ASCET のオプションウィンドウ（“Options”ウィンドウ）が開きます。

  各オプションは関連した項目ごとに複数のグループに分類され、各グループは、オプションウィンドウ左側のツリービューのノードとして表示されます。

- ツリービューから、設定したいノードを選択します。

  各オプションごとに、入力形式（ラジオボタン、入力フィールド、コンボボックスなど）が異なります。また、左のようなボタンで別のダイアログボックスを開いて設定するオプションもあります。

- 各オプションを設定します。

  変更されたオプションには * マークが付き、またデフォルト値と異なる内容が設定されたオプションは**太字**で表示されます。

  複数のノードのオプションを連続して設定することができます。

- **System Defaults** ボタンをクリックすると、現在選択されているノードの設定がすべてデフォルト設定に戻ります。

- 必要な項目をすべて設定したら、**OK** をクリックします。

  設定内容が適用され、“Options”ウィンドウが閉じます。

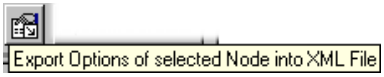
注記

いくつかのオプション項目は、変更内容がすぐに反映されないものもあります。たとえば ASCET ユーザーインターフェース用フォントを変更した場合、新しいフォントは、すでに開いているウィンドウには適用されません。新しく設定したフォントを適用するには、開いているウィンドウを一旦閉じてから再度開く必要があります。

ASCET のオプション設定は、ノードごとに XML ファイルにエクスポートすることができます。

オプションをエクスポートする：

- “Options”ウィンドウを開き、設定をエクスポートしたいノードを選択します。
- 必要に応じてオプション設定を変更します。
- **Export Options of selected Node into XML File** ボタンをクリックします。



または

- **File → Export** を選択します。

  Windows のファイル選択ダイアログボックスが開きます。ファイルの種類として `*.xml` が選択されています。
- エクスポートファイルのパスと名前を指定します。
- **Save** をクリックします。

  選択されているノードのオプション設定が、指定された XML ファイルに書き込まれます。

ファイルにエクスポートされたオプション設定は、再度インポートすることができます。これにより、ユーザー間で同じオプション設定を共有することができます。

オプションをインポートする：

- “Options”ウィンドウを開き、設定をインポートしたいノードを選択します。

- **Import Options of selected Node from XML File** ボタンをクリックします。

または

- **File → Import** を選択します。

  設定の上書きを確認する“Import Options”ダイアログボックスが開きます。

- 今後、同じタイプの質問に毎回同じ回答を行いたい場合は、**Remember my Decision** オプションをオンにします（46 ページ参照）。

  上記の設定を行うと、以後“Import Options”ダイアログボックスは開かなくなります。この設定は、ASCET“Options”ウィンドウの“Confirmation Dialogs”ノードで確認／変更できます（46 ページの「確認ダイアログボックスに関するオプション」参照）。

- 現在の設定が上書きされても問題ない場合、**OK** をクリックします。

  Windows のファイル選択ダイアログボックスが開きます。ファイルの種類として `*.xml` が選択されています。

- インポートしたい設定が保存されているXMLファイルを指定します。
- **Open** をクリックします。

  選択されているノードに、XML ファイルに保存されているオプション設定が読み込まれます。

フィルタ機能を利用して、必要なオプションを容易に検索することができます。

オプションをフィルタリングする：

- “Options”ウィンドウ左上のフィルタフィールドに、検索したい文字列を入力します。
- 複数の文字列を入力することもできます。
- **Apply Filter String** ボタンをクリックします。

  ダイアログボックス右側のフィールドに、オプション名、値、またはその説明文に、指定された文字列が含まれるがオプションアイテムのみが表示されます。

  また、左側のツリービューにはフィルタにマッチするノードだけが表示されます。ただし、マッチしないノードであっても、階層を維持するために必要なものは表示されます。

2.2.1 一般オプション

オプションウィンドウの最上位ノードである“Options”ノードで、一般的なオプションを設定します。

オプション	値	説明
Automatic Save	オン / オフ	自動保存機能の有効化／無効化
Store every ... minutes	数値	データベースを自動保存する間隔
Database Path	パス	ASCET データベースのメインディレクトリ
Multiple user handling	オン / オフ	ユーザー選択機能（マルチユーザーモード）の有効化／無効化
Send E-Mail	オン / オフ	サービスレポートの自動送信
E-Mail Address	アドレス	ETAS ホットラインの E-Mail アドレス

Automatic Save オプションをオンにすると自動保存機能が有効になり、指定の時間ごとにデータが自動保存されます。

パフォーマンス上の理由から、データベースへの入力を必要とする一部のユーザー操作は、キャッシュメモリに対してしか行われませんが、この自動保存機能を利用すれば、データベースキャッシュの内容を定期的に保存することができます。

自動保存機能を有効にする：

- **Automatic Save** オプションをオンにします。
  自動保存機能が有効になります。
- “Store every ... minutes”フィールドに、保存する間隔を数字で入力します。

または

- 矢印キーで間隔を変更します。
  データベースキャッシュの内容が、指定の時間間隔で自動的に保存されます。

Multiple user handling オプションでは、ASCET でユーザープロファイルの管理を行うかどうかを設定します。ユーザープロファイルについての詳細は、2.2.11 項 「ユーザープロファイル」を参照してください。

Send E-Mail オプションがオンになっていると、障害発生時に作成されたサービスレポートが自動的に ETAS ホットラインに送信されます。

E メールでのレポート自動送信機能を有効にする：

- **Send E-Mail** オプションをオンにします。

- ホットラインサービスのアドレスを変更する必要がある場合には、“E-Mail Address”フィールドのアドレスを編集します。（イータス株式会社のホットラインの E メールアドレスは ec.hotline@etas.co.jp です。）
- **OK** をクリックします。

  以後、コンポーネントマネージャで **Help → Problem Report** を選択すると、障害に関するレポートが作成され、自動的に E メールで ETAS ホットラインに送信されます。この機能についての詳細は 2.1.4 項を参照してください。

2.2.2 表示オプション

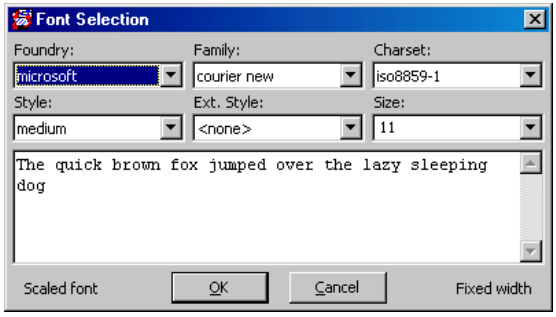
“Appearance”ノードで、ASCET や、その各種エディタの表示に関する設定を行えます。各オプションは下位のサブノードに分類されます。

オプション	値	説明
Extended Infos in Database Browser	オン / オフ	コンポーネントマネージャ上の各アイテムに、アクセス権限を表示
Show Disclaimer	オン / オフ	ASCET 起動時に免責条項を表示
List Font	フォント	リストエントリ用フォント
Text Font	フォント	ユーザーインターフェース用フォント
Edit Font Mapping	記述子	Windows フォントから Postscript フォントへのマッピング

フォントを設定する：

- 変更したいフォントの隣のボタンをクリックします。

  “Font Selection”ダイアログボックスが開きます。



- “Foundry”コンボボックスで、フォントの供給元を選択します。

- “Family”コンボボックスでフォントを選択します。
- “Charset”コンボボックスで文字コードセットを選択します。
- “Style”コンボボックスでスタイルを選択します。
- “Size”コンボボックスでフォントサイズを選択します。

  使用できるサイズは、フォントによって異なります。

  下の部分に、現在選択されているフォント設定の表示例が示されます。“Ext. Style”コンボボックスや **OK** ／**キャンセル** ボタンの左右にその他の情報が表示されます。
- **OK** をクリックして“Font Selection”ダイアログボックスを閉じます。

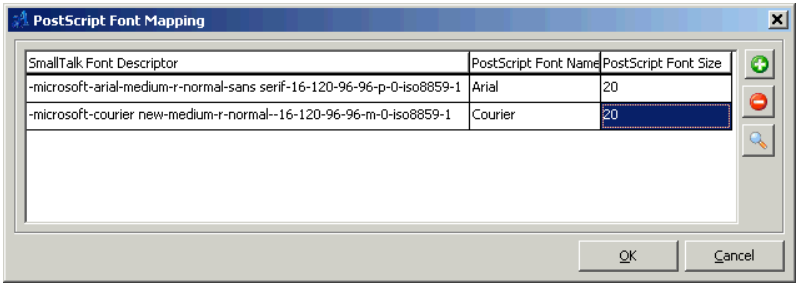
  ここで設定された内容は、すでに開いているウィンドウには反映されません。

“Edit Font Mapping”オプションは、Windows フォントを PostScript フォントにマッピングするためのものです。ブロックダイアグラムを PostScript ファイルに保存すると、各 Windows フォントの代わりに、ここで定義された PostScript フォントが使用されます。

フォントのマッピングを定義する：

- **Edit Font Mapping** オプションのボタンをクリックします。

  “PostScript Font Mapping”ダイアログボックスが開き、既存のマッピングエントリが表示されます。



列	意味
SmallTalk Font Descriptor	Windows フォント（SmallTalk）
PostScript Font Name	PostScript フォント
PostScript Font Size	PostScript フォントのサイズ

1. マッピングエントリの追加と削除

- 新しいマッピングエントリを作成するには、“PostScript Font Mapping”ダイアログボックスの **Add new Mapping Entry** ボタンをクリックします。

注記

1 つの Windows フォントを複数の PostScript フォントにマップすることができます。ただし PostScript ファイル作成時には、そのフォントの 1 番目のマッピングエントリのみが使用されます。

- マッピングエントリを削除するには、“PostScript Font Mapping”ダイアログボックスの **Remove selected Mapping Entry** ボタンをクリックします。

2. マッピングエントリの編集

- “PostScript Font Mapping”ダイアログボックスで、“SmallTalk Font Descriptor”列のセルをダブルクリックします。

  “Font Selection”ダイアログボックスが開きます（43 ページ参照）。
- “Font Selection”ダイアログボックスから Windows フォントを選択して **OK** をクリックします。
- “PostScript Font Mapping”ダイアログボックスの“PostScript Font Name”列のセルをダブルクリックし、Windows フォントをどの PostScript フォントに置き換えるかを選択します。
- “PostScript Font Mapping”ダイアログボックスの“PostScript Font Size”列のセルをダブルクリックし、PostScript フォントのサイズを設定します。

- “PostScript Font Mapping”ダイアログボックスの **Preview selected Mapping Entry** ボタンをクリックして、設定されたフォントのマッピングを確認します。

3. 設定の確定とキャンセル

- **OK** ボタンをクリックすると、設定内容が確定され、“PostScript Font Mapping”ダイアログボックスが閉じます。
- **Cancel** ボタンをクリックすると、設定内容が破棄され、“PostScript Font Mapping”ダイアログボックスが閉じます。

確認ダイアログボックスに関するオプション

ASCET で作業を行うと、所定の箇所でさまざまな「確認ダイアログボックス」が開きます。このダイアログボックスは、任意に「非表示」にすることができます。現在非表示に設定されているダイアログボックスは、“Options”ウィンドウの“Confirmation Dialogs”ノードで確認できます。非表示になっているダイアログボックスは  シンボルで示され、ここで任意のダイアログボックスを非表示に設定することもできます。

ただし、通常このノードは、ASCET での実際の作業中に非表示に設定したものを確認する目的でのみ使用するようにしてください。まだ実際の作業で開いたことのないダイアログボックスをこのノードで非表示に設定することは、お勧めできません。

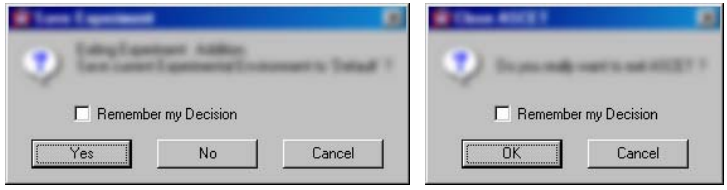
“Dialog”列	“Decision”列	説明
Change DB Path	`Show / Yes / No`	デフォルトディレクトリ以外の場所に保存されているデータベースを開く際に、デフォルトのデータベースパスを変更するかどうかを確認する
Close ASCET	`Show / OK`	ASCET 終了時の確認
Close Boolean Table	`Show / OK`	不正なエントリを含む論理テーブルを閉じる際の確認
Close existing Editor	`Show / OK`	すでにエディタで開いているコンポーネントを再度重複して開こうとした時の確認
Deassign Application Mode <inactive>	`Show / OK`	OS エディタにおいて、アプリケーションモードが `inactive` である init タスクのモードが変更された際
Edit complex Element	`Show / Hide`	新しい複合エレメントを作成する際にエレメントエディタが自動的に開くようにするかどうかを指定します（`Show`：開く）。
Edit CT Element	`Show / Hide`	新しい CT ブロックを作成する際にエレメントエディタが自動的に開くようにするかどうかを指定します（`Show`：開く）。
Edit Implementation Cast	`Show / Hide`	新しいインプリメンテーションキャストを作成する際に、エレメントエディタを開くかどうかを指定します（`Show`：開く）。

“Dialog” 列	“Decision” 列	説明
Edit primitive Element	`Show / Hide`	新しいエレメント（変数、パラメータ、特性カーブ／マップ）を作成する際に、エレメントエディタが開くようにするかどうかを指定します。
Export Components	`Show / OK`	**One File for Each Item** オプションがオンになっている場合の、エクスポート時の確認
Export Message Configuration	`Show / OK`	メッセージコンフィギュレーションをエクスポートする際の確認（ 2.4.2 項参照 ）
Import Options	`Show / OK`	オプションをインポートする際の確認
Import Views	`Show / OK`	ビューをインポートする際の確認
Overwrite Files/Folders	`Show / OK`	コンポーネントを読み込んで既存のコンポーネントに上書きする際の確認
Resize Drawing Area	`Show / Yes / No`	ブロックダイアグラムを開く際の、描画領域サイズの変更確認
Save Experiment	`Show / Yes / No`	実験を終了する際に実験を保存するかどうかを確認するダイアログ
Set Layout to Default	`Show / OK`	新しいコンポーネントのレイアウトをデフォルトレイアウトに設定する際の確認（ 231 ページ参照）
Set Options to Default	`Show / OK`	オプションをデフォルト設定に戻す際の確認
Set Target Settings to Default	`Show / OK`	未定義なターゲット／コンパイラ用のプロジェクトを ASCET で開く際の確認
Start Offline Simulation	`Show / OK`	オフライン実験を開始してイベントジェネレータにイベントが存在しない場合の確認
Store Changes in Graphic	`Show / Yes / No`	変更内容を保存しないでブロックダイアグラムを閉じようとした際の確認

各オプションで選択できる値は、ダイアログボックスの内容に応じて異なります。

- `Show`： どのオプションでも選択できます。確認ダイアログボックスが表示されます。
- `Yes` または `No`： 確認ダイアログボックスに **Yes** ボタンと **Yes** ボタンが含まれている場合、これらの値を選択できます。

- OK： 確認ダイアログボックスに **OK** ボタンが含まれている場合、この値を選択できます。

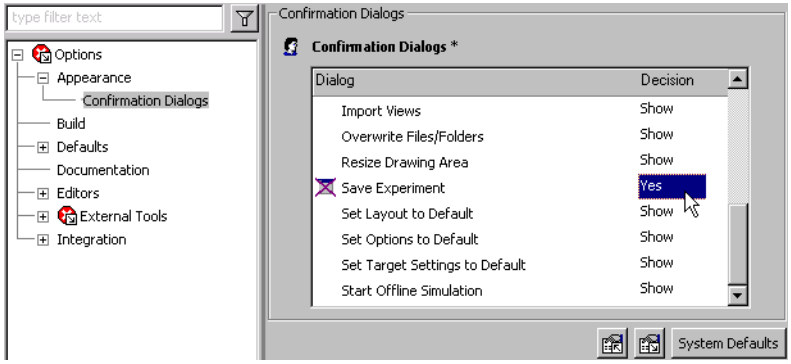


Yes、No、OK のいずれかが選択されている確認ダイアログボックスは、表示されず、選択されている値がユーザーからのレスポンスとして使用されます。

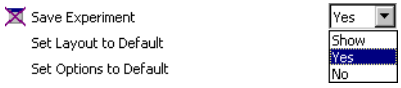
- Hide： 新しいエレメントを作成する際にエレメントエディタが自動的開くようにするかどうかを指定するオプションの場合、この値を選択できます。Hide に設定すると、エレメントエディタが自動で開くことはありません。

確認ダイアログウィンドウの表示／非表示を切り替える：

- “Options”ウィンドウの”Confirmation Dialogs”ノードを開きます。
- “Confirmation Dialogs”フィールドで、設定を変更したいアイテムの“Decision”列をクリックします。



- コンボボックスからエントリを選択します。



選択内容に応じて確認ダイアログボックスの表示／非表示が切り替わります。

2.2.3 ビルドオプション

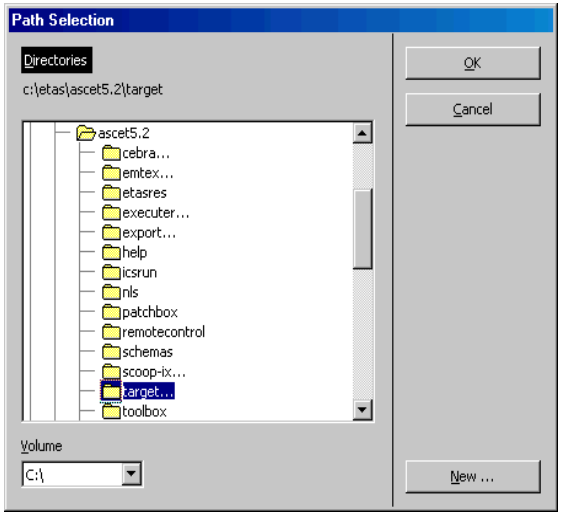
ASCET でのコード生成に影響するオプションは、“Build” ノードに含まれます。関連するオプションごとに、各サブノードにまとめられています。

オプション	値	説明
Code Preview Path	パス	生成された C コードファイルまたは HTML/XML ファイルの保存先ディレクトリ
Code Generation Path	パス	生成されたコードが保存されるディレクトリ （デフォルト : `ETAS¥Ascet5.2¥cgen`）
Target Root Path	パス	各ターゲットのメインディレクトリ。この下に各ターゲット用サブディレクトリが作成されます。
Specific Path	パス	プロジェクトファイル用に任意に設定できるパス
Write Project Files on Build	オン / オフ	コード生成時にプロジェクトファイルをディスクに書き込むかどうかを指定します。
Keep files in Generation Directory on shutdown	オン / オフ	ASCET を終了する際にコード生成ディレクトリ（**Code Generation Path** オプションで設定）の内容を削除せずにそのまま保存するかどうかを指定します。

パスを設定する：

- パスを変更するには、パス名の隣にあるボタンをクリックします。

  “Path Selection”ダイアログボックスが開きます。



- 必要に応じて、“Volume”コンボボックスでボリュームを選択します。
- “Directories”リストから、既存のディレクトリを選択します。

または

- 新しいディレクトリを作成するには、**New** ボタンをクリックします。
- **OK** をクリックします。

  指定されたディレクトリが“Options”ウィンドウに表示されます。

2.2.4 デフォルトオプション

“Default”ノードでは、新しく作成されるプロジェクト（デフォルトプロジェクトを含みます）とインプリメンテーション（コード実装情報）のデフォルト設定を指定します。インプリメンテーションのデフォルト設定は、“Implementation”サブノードに含まれています。

オプション	値	説明
Use Arithmetic Service	オン / オフ	新しいプロジェクトで算術演算サービスを使用するかどうかを指定します。
Use first available Service set	オン / オフ	`services.ini` ファイル内に保存されている最初の算術演算サービスのセットを、新しいプロジェクトに割り当てます。
Use Project Options Template	オン / オフ	新しいプロジェクトのプロジェクト設定のテンプレートとして、XML ファイルを使用します。
Project Options Template	ファイル	プロジェクト設定のテンプレートとして使用する XML ファイルのパスとファイル名（**Use Project Options Template** がオンの場合のみ有効）

新しく作成されるプロジェクトについて算術演算サービスが有効になるようにするには、以下のように設定します。

算術演算サービスのデフォルト設定を指定する：

- “Defaults”ノードの **Use Arithmetic Service** オプションをオンにします。

  これにより、次回から新しく作成されるプロジェクトについて、算術演算サービスが有効になります。
- **Use first available Service set** オプションをオンにします。

  新しく作成されるプロジェクトで、`services.ini` ファイルに保存されている算術演算サービスのうち、最初のものが使用されます。
- **OK** をクリックして設定内容を確定します。

算術演算サービスについての詳細は、4.14 項を参照してください。

オプション	値	説明
Limit to maximum bit length	オン / オフ	演算においてオーバーフローが発生した場合、演算結果の範囲を制限するかどうかを指定します。
Implementation Master	`Model / Implementation`	インプリメンテーションマスタ
Resolution Handling	`Automatic / Reduce Resolution / Keep Resolution`	オーバーフロー発生時の精度の扱い
Minimum cont Data Type	`uint8/ int8/ uint16/ sint16/ uint32/ sint32`	`cont` 型変数の最小の実装データ型
Minimum log Data Type	`uint8/ int8/ uint16/ sint16/ uint32/ sint32`	`log` 型変数の最小の実装データ型
Minimum sdisc Data Type	`uint8/ sint8/ uint16/ sint16/ uint32/ sint32`	`sdisc` 型変数の最小の実装データ型
Minimum udisc Data Type	`uint8/ sint8/ uint16/ sint16/ uint32/ sint32`	`udisc` 型変数の最小の実装データ型
Default cont Data Type	`real64/ real32/ uint8/ sint8/ uint16/ sint16/ uint32/ sint32`	`cont` 型変数のデフォルトの実装データ型
Default log Data Type	`uint8/ sint8/ uint16/ sint16/ uint32/ sint32`	`log` 型変数のデフォルトの実装データ型
Default sdisc Data Type	`uint8/ sint8/ uint16/ sint16/ uint32/ sint32`	`sdisc` 型変数のデフォルトの実装データ型
Default udisc Data Type	`uint8/ sint8/ uint16/ sint16/ uint32/ sint32`	`udisc` 型変数のデフォルトの実装データ型

インプリメンテーションのデフォルトオプションを設定する：

- オーバーフローを防ぐため演算結果の値の範囲を限定するには、**Limit to maximum bit length** オプションをオンにします。

- “Implementation Master”コンボボックスでは、新しく作成されるインプリメンテーションにおいて、モデルデータ側と実装データ側のどちらを「インプリメンテーションマスタ」とするかを指定します。

  新しいインプリメンテーションを作成すると、インプリメンテーションエディタにおいて、ここで選択された側の設定が自動的にインプリメンテーションマスタとして使用されます。
- “Minimum <*type*> Data Type”コンボボックスでは、<*type*> 型のモデル変数に使用できる最小の実装データ型を指定します。
- “Default <*type*> Data Type”コンボボックスでは、<*type*> 型のモデル変数に割り当てられるデフォルトの実装データ型を設定します。

  <*type*> には `cont`、`sdisc`、`udisc`、`log` があります。

  各タイプの変数のインプリメンテーションを編集する際、ここで選択した型がデフォルトの実装型として割り当てられます。

  旧バージョンの ASCET で作成されたデータベースを開くと、`cont` 型変数には `real64` 、`sdisc` 型変数には `int8` 、`udisc` 型変数には `uint8`、`log` 型変数には `int8` が適用されます。
- **OK** をクリックして設定を確定します。

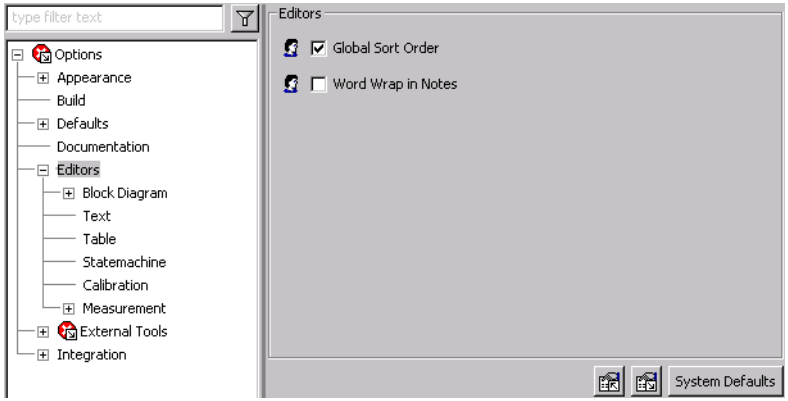
2.2.5 ドキュメント自動生成用オプション

“Documentation" ノードには、ドキュメント自動生成機能（第 7 章を参照してください）に関する一般的なオプションが含まれます。

オプション	値	説明
Documentor Working Path	パス	生成されるドキュメントが保存されるパス
Scale EPS Graphic	オン / オフ	**View → Save as Postscript** コマンドで作成される EPS グラフィックが、印刷サイズ（57 ページ参照）に合わせて調整されるようにするかどうかを選択する

2.2.6 エディタ用オプション

“Editors”ノードでは、各種エディタや測定／適合ウィンドウについて設定できます。各エディタごとに個別のサブノードにまとめられています。



オプション	値	説明
Global Sort Order	オン / オフ	全エディタのエレメントリストについて、共通のソーティングルールを適用するかどうかを指定
Word Wrap in Notes	オン / オフ	注釈エディタの自動ラインブレイクを有効にするかどうかを指定

ブロックダイアグラムエディタのオプション

“Block Diagram”ノードにはブロックダイアグラムについてのオプションが含まれます。

オプション	値	説明
Activate flexible layout	オン / オフ	内容されるコンポーネントのレイアウトをブロックダイアグラムエディタ内で変更できるようにするかどうかを指定する
Size of Undo Biffer	数値	ブロックダイアグラムエディタとステートダイアグラムエディタでの Undo 用バッファのサイズ（取り消すことのできる操作の数）
Drawing Area Size (px)	`2000@2000 / 5000@5000 / 10000@10000`	ブロックダイアグラムの描画領域サイズ（単位：ピクセル）
Selection Marker Size (Pixel)	数値	ブロックダイアグラムの選択マーカのサイズ

オプション	値	説明
Show Watermark	オン / オフ	選択されているビューの名前を、ブロックダイアグラム内に「ウォーターマーク」（透かし文字）で表示するかどうかを指定
Watermark Size (px)	数値	ウォーターマークのフォントサイズ
Show Page Number	オン / オフ	ブロックダイアグラムにページ番号を表示するかどうかを指定
BDE Graphical Comments	フォント	ブロックダイアグラム内のユーザーコメント用フォント
BDE Graphical Names	フォント	ブロックダイアグラム内のグラフィックエレメント名用フォント

フレキシブルレイアウトを有効にする（グローバル設定）：

- “Block Diagram”ノードで **Activation flexible layout** オプションをオンにします。

  ブロックダイアグラムエディタ、またはプロジェクトエディタの“Graphics”タブで、内包されるコンポーネントのレイアウトを変更できるようになります。

フォントの選択方法は、43 ページを参照してください。

“Colors”ノード： ブロックダイアグラムの表示色は、“Colors”ノードで設定します。

オプション	値	説明
Background Color	色	ブロックダイアグラムエディタ、およびプロジェクトエディタの“Graphic”タブの背景色
Grid Color	色	ブロックダイアグラムのグリッドの色
Comment Line Color	色	ブロックダイアグラムとプロジェクトのコメントラインの色
Selection Marker Color	色	ブロックダイアグラム内のハンドル（選択されたアイテムのマーカ）の色
Watermark Color	色	ウォーターマーク（透かし文字で表示される、現在のビューの名前）の色

ブロックダイアグラムの表示色や、入力領域（論理テーブル - 4.6 項参照 - の入力領域と条件テーブル - 4.7 項参照 - の条件領域）の表示色、および出力領域（論理テーブルの出力領域と条件テーブルのアクション領域）の表示色は、任意に設定することができます。

色を選択する：

- 変更したいオプションの隣のコンボボックスを選択します。

  色の選択ウインドウが開きます。現在選択されている色は四角で囲んで示されます。
- 希望する背景色を選択します。

  指定した背景色を実際に適用するには、現在開いているウィンドウを一度閉じてから再度開いてください。

“Sequencing”ノード： このノードにはブロックダイアグラム内のシーケンスコール（4.1.6 項参照）に関するオプションが含まれます。

オプション	値	説明
Sequence Shift Offset	数値	シーケンスコール番号をシフトする際のオフセット
Sequence Step Size	数値	シーケンスコール番号の自動割り当てを行う際のステップサイズ
Use gaps	オン / オフ	シーケンスコール番号の自動割り当てを行う際、現在の各シーケンスコール間のギャップが埋められます。

シーケンスコールのオプションを設定する：

- “Sequencing Step Size”フィールドで、シーケンスコールの自動割り当てを行う際の番号の間隔を指定します。
- “Sequencing Shift Offset”フィールドで、シーケンスコール番号の自動シフトを行う際のオフセットを指定します。
- **Use gaps** オプションがオンになっていると、シーケンスコール番号の自動割り当て時に、既存のシーケンスコールの番号の間隔が埋められます。

“Paper Size”ノード： このノードにはブロックダイアグラムの印刷に関するオプションが含まれます。

オプション	値	説明

Paper Size	`A5 (148x210)/`
	`A4 (210x297)/`
	`Letter (216x279)/`
	`Legal (216x355)/`
	`Userdefined`
		印刷領域のサイズ（`Userdefined`を選択すると下の 2 つのオプションで任意のサイズを指定可能）

User Size Width (mm)	数値	印刷領域の高さ
User Size Height (mm)	数値	印刷領域の幅
Paper Orientation	`Landscape / Portrait`	印刷領域の方向

テキストエディタのオプション

“Text”ノードには、C コードエディタと ESDL エディタに関するオプションのうち、ステートやトランジションに関連しないものが含まれます。

オプション	値	説明
Tabulator Size	数値	テキストエディタのタブの文字数
Code Font	フォント	ESDL および C コードエディタ内のテキスト用フォント

フォントの選択方法は、43 ページを参照してください。

テーブルエディタのオプション

“Tables”ノードには論理テーブルと条件テーブルの表示色を設定するオプションアイテムが含まれます。

オプション	値	説明
Input Color	色	テーブルの入力領域の背景色
Output Color	色	テーブルの出力領域の背景色

色の選択方法は、56 ページを参照してください。

ステートマシンエディタのオプション

“Statemachine”ノードにはステートマシンエディタ関するオプションが含まれます。

オプション	値	説明
Use ESDL as default for state machine	オン / オフ	ステートとトランジション内のアクションのデフォルトエディタを ESDL エディタに設定
Animated States Color	色	実験中にアニメートされるステートの色

ステートマシン用のデフォルトオプションを設定する：

- **Use ESDL as default for state machine** をオンにします。
- **OK** をクリックして設定内容を確定します。

  ステートマシン内に新しいステートまたはトランジションを作成すると、ステートまたはトランジション用のデフォルトエディタとして、ESDL エディタが使用されます。

色の選択方法は、56 ページを参照してください。

適合ウィンドウ用オプション

“Calibration”ノードでは、適合ウィンドウのデフォルト設定を定義します。ここでの設定は、個々のウィンドウにおいて任意に変更することができます。

各オプションアイテムの内容は、“Options”ウィンドウに表示される説明を参照してください。

測定ウィンドウ用オプション

“Measurement”ノードでは、測定ウィンドウのデフォルト設定を定義します。ここでの設定は、個々のウィンドウにおいて任意に変更することができます。

各オプションアイテムの内容は、“Options”ウィンドウに表示される説明を参照してください。

“Datalogger”ノード： このノードにはデータロガー（6.1.9 項参照）用のオプションが含まれます。

オプション	値	説明
Auto Increment DataLogger File Name	オン / オフ	データロガーのログファイル名に、自動的にインクリメントされる番号が付加されるようにするかどうかを指定します。

2.2.7 実験オプション

“Experiment”ノードには、オフライン／オンラインについてのオプションが含まれます。

オプション	値	説明
Initialize variables at OS start	オン / オフ	シミュレーション（オフライン実験の場合）またはオペレーティングシステム（オンライン実験の場合）が開始されるたびに変数の値が初期化されるようにするかどうかを指定します。
Initialize parameters at OS start	オン / オフ	シミュレーション（オフライン実験の場合）またはオペレーティングシステム（オンライン実験の場合）が開始されるたびにパラメータ（適合変数）の値が初期化されるようにするかどうかを指定します。

2.2.8 外部ツールオプション

“External Tools”ノードには外部ツール用オプションが含まれます。ESDL／C コードエディタや ASCET モニタウィンドウから呼び出されるテキストエディタや、コンパイラなどについての設定は、個別のサブノードで行い、その他の設定は“External Tools”ノードで行います。

オプション	値	説明
Perl compiler	ファイル	Perl コンパイラ
External Make Utility	ファイル	外部 Make ファイル
Output Redirector Utility	ファイル	出力のリダイレクト
C Preprocessor	ファイル	C コードプリプロセッサ
C Code Beautifier	ファイル	生成コードのフォーマッティングユーティリティ

外部ツールのパスを設定する：

- “External Tools”ノードで、変更したいオプション項目の隣のボタンをクリックします。

  Windows のファイル選択ダイアログボックスが“開きます。
- ツールが保存されているディレクトリを選択します。
- ファイルを選択して **Open** をクリックします。

  ボタンの左側のフィールドに、パスとファイル名が表示されます。

ASCII エディタオプション

オプション	値	説明
Use external ASCII Editor	オン / オフ	外部テキストエディタを使用するかどうかを指定
ASCII Editor	ファイル	外部テキストエディタのパスと名前
Commandline Arguments	文字列	外部テキストエディタのコマンドライン引数
Begin RegionMarker	文字列	エディタがリージョンの先頭を認識するためのマーカ
End RegionMarker	文字列	エディタがリージョンの最後尾を認識するためのマーカ

コマンドライン引数とリージョンマーカについては、いくつかのテキストエディタで使用される文字列を選択することもできます。

コンパイラオプション

“Compiler”ノードには、各コンパイラに対応するサブノードが含まれます。デフォルト状態において表示されるコンパイラは、Borland-C V4.5、Borland-C V5.5、Microsoft Visual C++ です。ASCET-RP や ASCET-SE がインストールされていると、各マイクロコントローラターゲット用のコンパイラも表示されます。

以下のオプション（最後の 1 つのオプションアイテムを除く）は、すべてのコンパイラで使用されます。

オプション	値	説明
Tool Root Path	パス	コンパイラのインストールディレクトリ
Private directive	命令文	プライベート関数用コンパイラ命令
Purblic directive	命令文	パブリック関数用コンパイラ命令
Unique file extension	文字列	コンパイラのユニークな識別子
Object file extension	ファイル拡張子	オブジェクトファイルのファイル識別子
Result file extension	ファイル拡張子	実行ファイルのファイル識別子
Supports long filenames	オン / オフ	コンパイラがファイルのロングネームをサポートしているかどうかを指定

オプション	値	説明
Inline directive	命令文	インライン関数用のコンパイラ命令
Change CYGWIN Registry Settings		Cygwin のレジストリ設定を変更して GNU コンパイラが正常に動作するようにするためのオプション（GNU-C V2.95.3 専用のオプション）
Supports precompiled header	オン / オフ	コンパイラがプリコンパイルされたヘッダをサポートしているかどうかを示すオプション（編集不可）

2.2.9 統合オプション

“Integration”ノードでは、インポート／エクスポート、データ交換などのオプションを設定します。関連するオプションごとに個別のサブノードにまとめられています。

オプション	値	説明
Tool Server Activation	オン / オフ	ASCET 内部で用いられるオプション（常にオンにしておいてください）
Use .NET based Tool API	オン / オフ	最新の .NET ベースの API と、従来（旧式）の MS J++ ツール API を切り替えます。

エクスポートオプション

“Expot”ノードでは、選択されたデータをどの範囲でどのような方法でエクスポートするかを設定することができます。つまり、各データベースアイテムごとに個別のファイルに保存されるように指定したり、被参照アイテムも含めてエクスポートするように指定することができます。

また、データベースアイテムのエクスポート時にディスクリプションファイルが生成されるようにするかどうかを指定することができ、このファイルのフォーマットや内容も設定できます。

オプション	値	説明
Default Export Path	パス	デフォルトのエクスポートパス
Include Referenced Items	オン / オフ	リカーシブエクスポート（被参照アイテムを含んだエクスポート）を行う
One File for each Item	オン / オフ	各アイテムごとに個別のファイルにエクスポートする
Write ASCII file on Binary Export	オン / オフ	ASCII フォーマットのディスクリプションファイルを生成
Write XML file on Binary Export	オン / オフ	XML フォーマットのディスクリプションファイルを生成
Create HTML File	オン / オフ	XML ディスクリプションファイル用の HTML ファイルを生成

オプション	値	説明
Write Default Project Info	オン / オフ	XML ディスクリプションファイルにデフォルトプロジェクトに関する情報を追加
Write Version Info	オン / オフ	XML ディスクリプションファイルにバージョン情報を追加
Write Graphic Specification	オン / オフ	XML ディスクリプションファイルにグラフィック情報を追加
Sort XML Tags	パス	XML ディスクリプションファイル内のタグをアルファベット順にソート
Create complete Hierarchy	オン / オフ	AMD エクスポートの際、ASCET データベースのフォルダ構成を反映しエクスポートファイルを作成
Export Experiment Environment	オン / オフ	コンポーネントとともに実験環境をエクスポート
Encryption Key	キー	AMD エクスポート時のファイル暗号化用キー

エクスポートオプションを設定する：

- “Default Export Path”フィールドで、エクスポート先のデフォルトディレクトリを選択します。
- データベースアイテムをリカーシブにエクスポートする（つまり被参照アイテムもすべて含めてエクスポートする）には、**Include Referenced Items** オプションをオンにします。
- アイテムごとに個別のファイルにエクスポートするには、**One File for each Item** オプションをオンにします。
- **Write ASCII file for Binary Export** オプションをオンにすると、ASCII フォーマットのディスクリプションファイルが生成されます。
- **Write XML file for Binary Export** オプションをオンにすると、XML フォーマットのディスクリプションファイルが生成されます。

  Write XML file for Binary Export の下のフィールドのオプションが有効になります。

  – XML ディスクリプションファイルの内容を指定するための各オプションのオン／オフを切り替えます。
- “AMD Format”フィールドの各オプションのオン／オフを切り替え、AMD ディスクリプションファイルの内容を指定します。

データベースアイテムのエクスポートについての詳細は、90 ページの「フォルダやデータベースアイテムのエクスポート」を参照してください。

インポートオプション

“Import”ノードには、データベースアイテムをインポートする際のオプションが含まれます。さらに“Autofixes”サブノードには、AMD インポートを行う際の自動修復に関するオプションが含まれます。

オプション	値	説明
Default Import Path	パス	デフォルトのインポートパス
Discard Existing Implementation	オン / オフ	既存のインプリメンテーションを、インポートされたインプリメンテーションで上書きする
Keep Folder Path	オン / オフ	既存のインプリメンテーションをインポートする際、データベース内のパスをそのまま保持する
Remove Version Information	オン / オフ	バージョン管理ツールからの情報を削除する
Keep Hierarchy	オン / オフ	AMD インポートの際、エクスポートファイルの階層を ASCET データベースに反映させる
Import Referenced Items	オン / オフ	リカーシブな AMD エクスポート（被参照アイテムを含んだインポート）を行う
Remap OIDs	オン / オフ	インポートされたすべてのアイテムに新しいオブジェクト ID を割り当てる（これにより、既存のアイテムを再度インポートすると、アイテムのコピーが作成されます）
Encryption Key	キー	AMD インポート時のファイル暗号化用キー

インポートオプションを設定する：

- “Default Import Path”フィールドで、インポートのデフォルトディレクトリを選択します。
- 既存のインプリメンテーションをすべてインポートされたインプリメンテーションに置き換えるには、**Discard Existing Implementations** オプションをオンにします。

  この場合、既存のインプリメンテーションはすべて失われ、インポートされたアイテムのインプリメンテーションが有効になります。

  このオプションは、既存のアイテムをインポートする場合にだけ意味を持ちます。

- データベース内のフォルダ構造を保持するには、**Keep Folder Path** オプションをオンにします。

  このオプションがオフになっていると、既存のアイテムをインポートする際、そのアイテムは、エクスポートファイル内に保存されたパスにインポートされます。

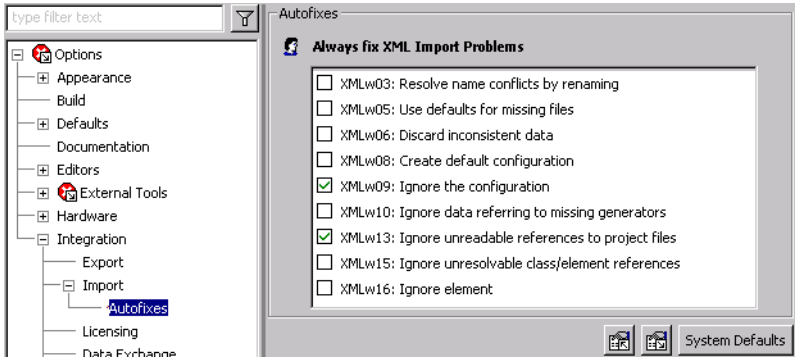
- バージョン情報を削除するには、**Remove Version Information** オプションをオンにします。

  このオプションをオフの状態のままにして、バージョン情報が付加されたアイテムをインポートすると、オリジナルのアーカイブにアクセスできない場合、バージョン管理用データベースにそのアイテムを保存することができません。

- AMD フォーマットのインポートについては、“AMD Format” フィールドで設定します。

データベースアイテムのインポートについての詳細は、94 ページの「データベースアイテムのインポート」を参照してください。

“Autofixes” ノード： このノードには、AMD インポート時に発生する可能性のある問題に関する対処法の一覧が表示されます。チェックマークを付けた問題は自動的に解決され、通知も行われません。



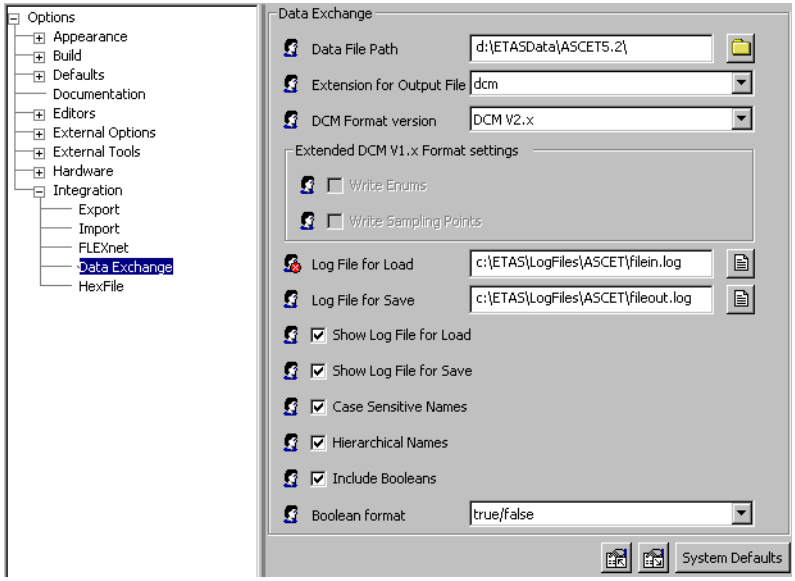
ライセンスオプション

“Licensing”ノードでは、サーバーライセンスの借用期限を設定できます。

オプション	値	説明
Borrow license for days	数値	ライセンスを借用できる期間（『ASCET 入門ガイド』の「ライセンスについて」の章を参照してください）
Activate idle time shutdown	オン / オフ	アイドル時間経過後に ASCET をシャットダウンする
Idle period [min]	数値	アイドル時間（単位：分）

データ交換オプション

“Data Exchange”ノードではデータ交換に関するオプションを設定できます。サポートされている交換フォーマットは DCM V1.x および V2.x で、INCA と ASCET（V4 以降）とのデータ交換が可能です。



オプション	値	説明
Data File Path	パス	データ交換ファイルのパス
Extension for Output File	`dcf` / `dcm` / `kon`	データ交換ファイルのファイル拡張子
DCM Format version	`DCM V1.x` / `DCM V2.x`	DCM フォーマットのバージョン
Write Enum	オン / オフ	列挙型データをテキストフォーマットで書き出す（`DCM V1.x` の場合のみ）
Write Sampling Points	オン / オフ	サンプリングポイントを DCM V2.x の構文で書き出す（`DCM V1.x` の場合のみ）
Case Sensitive Names	オン / オフ	大文字と小文字を区別するかどうかを指定（このオプションがオフになっていると、たとえば `CONT` という名前のパラメータは、`Cont` という名前でインポートされます）
Hierarchical Names	オン / オフ	グローバルエレメントの名前を、階層名を含む完全な名前にする（オン）か、またはアイテム名のみにする（オフ）かを指定
Include Booleans	オン / オフ	Boolean 変数をデータ交換処理に含めるかどうかを指定
Boolean format	`true/false` / `Integer`	Boolean 変数のデータ交換フォーマット（DCM V2.x の場合のみ）
Show Log File for Load / Show Log File for Save	オン / オフ	書き込み／読み込み時に作成されるログファイルを画面に表示するかどうかを指定
Log File for Load / Log File for Save	ファイル名	書き込み／読み込み時に作成されるログファイルのパスと名前

実行ファイルオプション

“HexFile”ノードには、実行ファイルに関するオプションが含まれます。

オプション	値	説明
HexFormat	`IntelHex` / `MotorolaSRecord`	エクスポートされる HEX ファイルのデフォルトフォーマット
IntelHexRecord Size	`16/32/64`	インテル HEX フォーマットのフィールドあたりの最大バイト数
SRecordCount	オン / オフ	各ブロックの最後に（それに続くターミネーションレコードがある場合はその前に）データフィールド数を付加

オプション	値	説明
SRecordFormat	`16/24/32`	アドレス幅（ビット数）
SRecordSize	`16/24/32`	フィールドあたりの最大バイト数
SRecord Termination	オン / オフ	各ブロックの最後にターミネーションレコードを付加

SRecord* オプションは、**HexFormat** オプションで `MotorolaSRecord` が選択されている場合にのみ有効です。

2.2.10 外部オプション

ASCET のオプションに、ユーザー定義のオプションを追加することができます。ユーザー定義オプションは XML ファイルに定義します。このファイルは ASCET のインストールディレクトリに保存し、拡張子は **`*.aod.xml`** を使用します。

1 つのオプションを定義するフォーマットは、以下のとおりです（*斜体* で書かれた部分は、実際のオプションで使用する値に置き換えてください）。

```
<OptionDeclaration
   xmlCategory="path"
   optionCategory="value"
   optionClass="type"
   attributeName="option name"
   optionFile="filename.xml">
   <Group>path</Group>
   <Label>text</Label>
   <Description>text</Description>
   <Tooltip>text</Tooltip>
   <InitialValue>value</InitialValue>
   <DefaultValue>value</DefaultValue>
</OptionDeclaration>
```

各属性とそのアイテムは、表 2-1 のとおりです。

オプションのタイプによってはさらに別のアイテムが使用されます（表 2-2 を参照してください）。

属性／アイテム	意味
xmlCategory[a]	オプションが定義された XML ファイル（optionFile で指定されます）の保存先のパス（例： Sample¥Option）
optionCategory	オプションの保存方法（FILE： ファイルに保存、FIXED： 保存しない）
optionClass	オプションタイプ（表 2-2 参照）
attributeName	XML ファイル（optionFile で指定されます）に定義されているオプションの名前。ファイル内でユニークである必要があります。
optionFilea	オプションの値が定義された XML ファイル
<Group>	ASCET オプションウィンドウにおいて、オプションを表示するパス（Node¥Subnode¥...）。新規のパスまたは既存のパスを使用できます。
<Label>	ASCET オプションウィンドウに表示するオプションの名前
<Description>	オプションについての簡単な説明文
<Tooltip>	ASCET オプションウィンドウで使用するツールチップ用テキスト
<InitialValue>	初期値：optionFile で指定された XML ファイルに値が保存されていない場合に使用される値
<DefaultValue>	デフォルト値： System Defaults として、および <InitialValue> が設定されていない場合は初期値として使用される値

a. 複数のオプションの値を 1 つの XML ファイルに保存したり、複数の XML ファイルを同じパスに保存することができます。

表 2-1　属性とアイテムの意味

ユーザー定義できるオプションのタイプ（optionClass）には以下のものがあります。

optionClass	意味
EtasBooleanOption	オプションボックス
EtasButtonOption	コマンド実行用ボタン
	<InitialValue> と <DefaultValue> には ignored という値が設定されます。 追加するアイテム： <ButtonOption> <Action>*path/executable file*</Action> </ButtonOption>
EtasEnumerationOption	文字列選択用コンボボックス
	追加するアイテム： <EnumerationOption> <StringValues> <StringValue>*string1*</StringValue> ... <StringValue>*stringN*</StringValue> </StringValues> <Values> <Value>*value1*</Value> ... <Value>*valueN*</Value> </Values> </EnumerationOption>

optionClass	意味
EtasFileOption	ファイルのパスと名前に使用するテキストフィールド、およびファイル選択ダイアログボックスを開くためのボタン

	追加するアイテム：
	`<FileOption>`
	`<DialogTitle>`*`text`*
	`</DialogTitle>`
	`<SearchPath>`*`path`*`</SearchPath>`
	`<SearchMask>`*`mask`*`</SearchMask>`
	`<InvalidCharacters>`*`characters`*
	`</InvalidCharacters>`
	`<FilterTypes>`
	`<FilterType` *`filter`*`>`
	`<Description>`*`text`*
	`</Description>`
	`</FilterType>`
	`...`
	`</FilterTypes>`
	`</FileOption>`

EtasFloatOption	矢印ボタン付き入力フィールド（浮動小数点値用）

	追加するアイテム：:
	`<FloatOption>`
	`<MinValue>`*`value`*`</MinValue>`
	`<MaxValue>`*`value`*`</MaxValue>`
	`</FloatOption>`

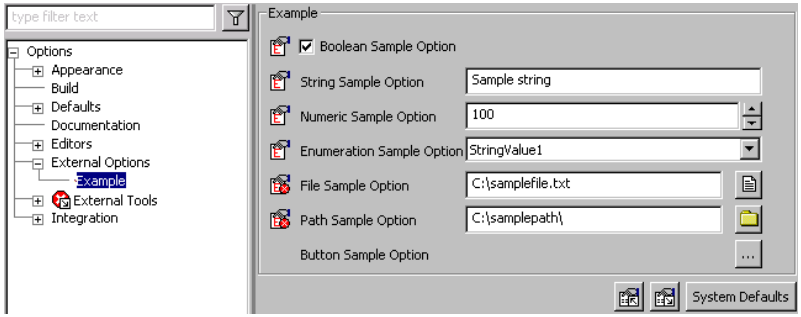
EtasNumericOption	矢印ボタン付き入力フィールド（整数値用）

	追加するアイテム：
	`<NumericOption>`
	`<MinValue>`*`value`*`</MinValue>`
	`<MaxValue>`*`value`*`</MaxValue>`
	`</NumericOption>`

EtasPathOption	“Path Selection”ダイアログボックスのディレクトリパスボタン用テキストフィールド
EtasStringOption	文字列入力用テキストフィールド

表 2-2　ユーザー定義オプションのタイプ（コードセクション内で*斜体*で書かれた部分は、実際のオプションで使用する値に置き換えてください）

ASCET のインストール CD には、`externalOptionsExample.aod.xml` というサンプルファイルが収められています。ここには以下のようなオプションが定義されています。



2.2.11 ユーザープロファイル

ASCET では、複数の「ユーザープロファイル」、つまりユーザーごとの作業環境を作成して管理することができます。この機能を使うことによって、たとえば複数のユーザーで 1 台のコンピュータを共用する場合、各ユーザーが独自の作業条件に合わせて ASCET のユーザーオプション（スクリーンフォント、フォントサイズ、各種エディタ、各種エレメントのデフォルト設定など）を設定し、それをユーザープロファイルとして保存しておくことができます。

注記

コンピュータを 1 ユーザーのみで使用する場合でも、複数のユーザープロファイルを使用できます。これによって、たとえばオフィス用と出張用に異なる作業環境を作成しておくこともできます。

ユーザープロファイルに設定された環境は、具体的な作業内容とは無関係に、ASCET での作業全般に渡って適用されます。

プログラム実行中にオプション設定を変更すると、現在 ASCET を使用しているユーザーのプロファイルの内容が変更されます。他のユーザープロファイルに切り替えるには、ASCET を再起動する必要がありますが、これを行うには、あらかじめオプションウィンドウ（**Tools → Options**、”Options”ノード）でマルチユーザーモードを有効にしておく必要があります。ユーザープロファイルの編集（ユーザーの追加／変更／削除）は、マルチユーザーモードにおいてのみ可能です。

ユーザー選択機能を有効にする

マルチユーザーモードが無効になっていると、システム（Windows）にログオンしたユーザーのユーザープロファイルが自動的に適用されます。

このモードを有効にすると、次回から ASCET を起動する時にユーザーを選択するダイアログボックスが開くので、ここで既存のユーザープロファイルを選択するか、または新しいユーザープロファイルを追加してログオンすることができます。この手順を以下に詳しく説明します。

ユーザー選択機能を有効にする：

- **Tools → Options** を選択します。

または

- **Options** ボタンをクリックします。

  “Options”ウィンドウが開きます。

- “Options”ノードで **Multiple user handling** オプションをオンにします。

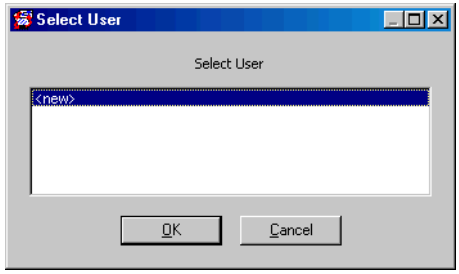
  これでマルチユーザーモードが有効になり、この後、ASCET を起動するたびに、既存のユーザーを選択するかまたは新しいユーザーとしてログオンするかが尋ねられます。

新しいユーザープロファイルを追加する：

- “Options”ウィンドウでマルチユーザーモードを有効にしてから ASCET を再起動します。

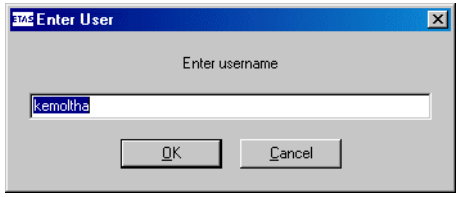
  “Select User”ダイアログボックスが開き、現在登録されているユーザーの一覧が表示されます。

  初めてマルチユーザーモードに切り替えた後は、このダイアログボックスには <new> というエントリのみが表示されます。



- 一覧から <new> を選択してから **OK** をクリックします。

  “Enter User”ダイアログボックスが開き、新しいユーザープロファイルの名前を入力するよう求められます。



- 名前を入力してから **OK** をクリックします。

  ユーザープロファイル名は 8 文字以内です。

  これで、デフォルトのユーザープロファイル、つまりデフォルトのユーザーオプション設定のコピーを使用して、コンポーネントマネージャが開きます。ウィンドウ最下のステータスバーには現在のユーザープロファイル名が表示されます。

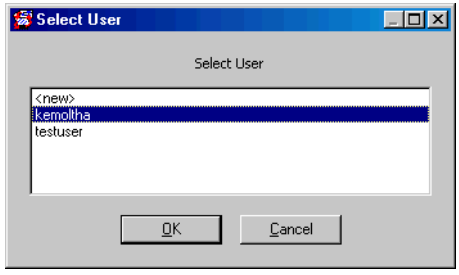
ASCET でのセッションを行っている時に変更されたユーザーオプションは、プログラム起動時に選択したユーザープロファイルに格納され、次の起動時に同じユーザーを選択すると、前回の設定内容が再び適用されます。

さまざまな条件に合わせて作業環境をカスタマイズする方法については、2.2.1 項 ～ 2.2.10 項を参照してください。

既存のユーザープロファイルを選択する：

- “Options”ウィンドウでマルチユーザーモードを有効にしてから ASCET を再起動します。

  “Select User”ダイアログボックスが開きます。



- 一覧の中からユーザープロファイルを選択して **OK** をクリックします。

  コンポーネントマネージャが開き、選択されたプロファイルに格納されたユーザーオプション設定が有効になります。

注記

プログラム実行中に他のユーザープロファイルに切り替えることはできません。現在のセッションを終了してから ASCET を再起動して、別のユーザーを選択してください。

2.3 コンポーネントマネージャ - データの管理

ここまでに説明されているように、コンポーネントマネージャの主な機能は、ASCET での作業中に作成されたすべてのデータ（クラス、モジュール、プロジェクトなど）をデータベースに保存して管理することです。コンポーネントマネージャのユーザーインターフェースにはさまざまな機能が用意されていて、データベースアイテムを効率的に扱うことができます。Windows エクスプローラに近いコンセプトでフォルダやサブフォルダ、またはアイテムを作成し、それらを個々にコピーしたり移動したり、さらにインポート／エクスポートすることができます。また、新しいデータベースを作成することも可能です。このようにして、一般的なファイルシステムと同じ要領でデータを管理することができます。

また、こういったデータ管理以外に、ASCET で作成されるソフトウェアシステムを構成する個々のコンポーネントやプロジェクトの編集も行えます。

データベースアイテム

ASCET のデータベースには、さまざまな種類のデータベースアイテムが格納されます。本項では、ASCET で使用できるデータベースアイテムの種類について概説します。

コンポーネント： ASCET で作成される組み込みソフトウェアシステムは、複数のコンポーネントで構成されます。コンポーネントは、アルゴリズムとデータを内包する機能単位です。1 つのコンポーネント内に定義されているアルゴリズムは、それぞれ個別に実行することができます。これらのコンポーネントを組み合わせて、組み込み制御システムのプロジェクトを構築します。

コンポーネントは物理レベルで定義（記述）され、その手法として、グラフィックを用いたブロックダイアグラムやステートマシン、または ESDL コードや C コードを使用することもできます。グラフィックや ESDL によって定義されたコンポーネントの内容は実装されるプラットフォーム、つまり実際にソフトウェアが実行されるハードウェア環境に依存しないので、これをさまざまなプラットフォーム用のコード生成に使用できます。一方、C コードで定義されたコンポーネントはプラットフォームに依存して記述され、ターゲット固有の処理がカプセル化されている必要があります。

ASCET で使用できるコンポーネントの種類とその用途については、『ASCET リファレンスマニュアル』の「コンポーネント」という章で説明されています。コンポーネントの扱いについては、コンポーネントの種類に応じたエディタについての説明個所を参照してください。

プロジェクト： プロジェクトは、組み込み制御システム全体の機能が定義されたものです。そこにはすべてのコンポーネントとそれに必要な「プロトコル」（コンポーネント間のインターフェース）、またオペレーティングシステムやコード生成に関する設定などが含まれ、プロジェクトにおいて、コンポーネント間の通信やアルゴリズムの実行順序が決定されます。

プロジェクトの概念については、『ASCET リファレンスガイド』の「プロジェクト」という章で、またプロジェクトの扱いについては 363 ページの「プロジェクトエディタ」という項で説明されています。

アイコン： ：コンポーネントの中に別のコンポーネントが内包される場合、内側のコンポーネントは、外側のコンポーネントのダイアグラム内の 1 ブロックとして表示されます。内容されているコンポーネントにアイコンを割り当てることにより、複雑なダイアグラムを見やすくすることができます。

アイコンは、あらかじめ ASCET で用意されているものを選択して利用することができます。またユーザー独自のアイコンを定義して編集することも可能です。アイコンの作成やインポート、および編集を行う方法については、544 ページの「アイコンエディタ」という項を参照してください。

シグナル： ASCET のモデルを現実的な条件下でテストするために、実際の測定データを ASCET にインポートして、外部からモデルへ供給されるシグナル（「スティミュラスシグナル」と呼ばれます）として使用することができます。このデータは「シグナル」というアイテムとしてデータベースに格納されます。

シグナルの扱いについての詳細は、541 ページの「シグナルビューア」を参照してください。

コンテナ： 「コンテナ」は、プロジェクト、クラス、モジュールを任意に格納しておく格納庫で、これは、モデルやデータベースを構造化し、異なるデータベース内のアイテムを共通のバージョン管理下に置くために使用されます。詳細は 435 ページの「コンテナ」の項を参照してください。

列挙型： 一連の番号に対応する複数の列挙子（「Enumerator」）が定義された、特殊なデータ型です。

ASAM-MCD-2MC プロジェクト： ASAM-MCD-2MC ファイルには、標準の ASAM-MCD フォーマットで、ASCET-MD と他のプログラム（INCA、ASCET-RP など）とのインターフェースが定義されています。

2.3.1 データベースアイテムの管理

ASCET では、データベースアイテムはフォルダ内に分類されて格納されます。データベースには任意の数のトップレベルフォルダを含めることができ、さらにトップレベルフォルダの中に他のフォルダを含めることができます。データベースアイテムは必ずフォルダ内に格納されている必要があるので、データベースアイテムをデータベースのルートレベルに作成することはできません。各データ

ベースアイテムは、それぞれ「オブジェクト ID」によって識別されます。1 つのフォルダ内の 2 つのオブジェクトが同じ名前を持つことはできませんが、互いにフォルダが異なれば、同じ名前を持つことができます。

本項では、データベースアイテムを管理するための一般的な操作方法について説明します。

フォルダを作成する：

- コンポーネントマネージャの“1 Database”リストから、データベース名、または新しく作成するフォルダを格納したいフォルダを選択します。

  フォルダは、データベース名より下の階層にしか格納できません。

- **Insert** → **Folder** を選択します。

または

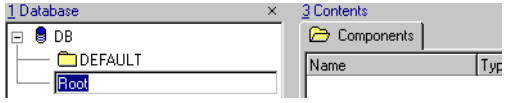
- **Insert Folder** ボタンをクリックします。

または

- **<Insert>** キーを押します。

  “1 Database”リストに新しいフォルダが追加されます。トップレベルにフォルダを作成した場合は、`Root` というデフォルト名が付き、それ以外の場合は `Folder` というデフォルト名が付きます。

- フォルダ名を編集します。強調表示されている名前の上から新しい名前を入力して **<Enter>** を押してください。



注記

フォルダ名とコンポーネント名では、大文字と小文字は区別されません。同じ文字で大文字／小文字のみが値が異なる複数の名前を使用することができません。

1 つのデータベースには必ず 1 つのトップレベルフォルダが必要なので、新しいデータベースを作成すると、必ず自動的にトップレベルフォルダが 1 つ作成されます。

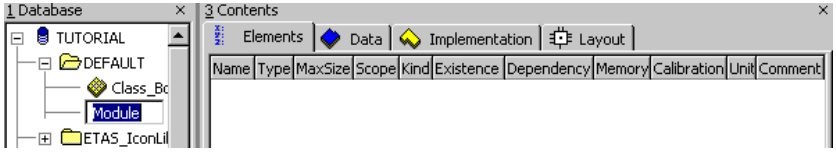
データベースアイテムを作成する：

- “1 Database”リストから、新しいアイテムを追加したいフォルダを選択します。
- **Insert → *<component type>*** またはアイテムのタイプによっては **Insert → *<component type>* → *<item type>*** を選択します。

または

- “Insert”ツールバー内のアイテムタイプに対応するボタンをクリックします。

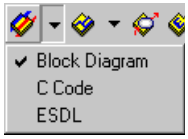
新しいアイテムが、デフォルト名で作成されます。またこのアイテムにはデフォルトのレイアウトが割り当てられています。



- アイテム名を編集してから **<Enter>** を押します。

`Module`、`Class`、または `Continuous Time Block` タイプのコンポーネントを作成する場合は、そのアイテムの記述形式（ブロックダイアグラム、ESDL、C コード）、つまりコンポーネントの定義を行う際に使用するエディタを選択します。エディタは、***<item type>*** サブメニューで指定するか、またはボタンを使用してコンポーネントを作成する場合は、これらのコンポーネントタイプのボタンの右側にある下向きの▼をクリックしてデフォルトエディタを選択しておきます。

クラス／モジュール用デフォルトエディタを設定する：



- **Insert Module - *<item type>*** または **Inser Class - *<item type>*** ボタンの隣の ▼ をクリックします。

ドロップダウンリストが開きます。現在のデフォルトエディタにチェックマークが付いています。

- クラスまたはモジュールのデフォルトエディタを選択します。

この後、ボタンを使用してコンポーネントを作成すると、ここで指定されたデフォルトエディタが割り当てられ、コンポーネントのデフォルト名にはエディタの種類を表わすテキストが含まれます。

列挙型を作成する：

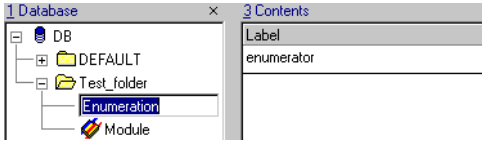
- “1 Database”リストから、列挙型アイテムを作成したいフォルダを選択します。

- **Insert → Enumeration** を選択します。

または

- **Insert Enumeration** ボタンをクリックします。

  新しい列挙型がデフォルト名で作成されます。また 1 番目のデフォルトの列挙子が“3 Contents”ペインに表示されます。



- “3 Contents”ペインの中の任意の場所をクリックします。

  コンポーネントマネージャのメニューが、28 ページに示された内容に変わります。

- 列挙子を追加するには、**Component → Enumeration → Add** を選択します。

または

- **<Insert>** キーを押します。

または

- “3 Contents”フィールドを右クリックし、ショートカットメニューから **Add** を選択してサブメニューを開いてコマンドを選択します。

  各コマンドに応じた位置に新しい列挙子が追加されます。

as first	1 行目
as last	最終行
before selection	選択された列挙子の前
after selection	選択された列挙子の後

- 列挙子の名前を変更するには、列挙子を選択して **Edit → Rename** を選択します。

または

- **<F2>** キーを押しします。
- 選択された列挙子を削除するには、**Edit → Delete** を選択します。

または

- **<Del>** キーを押します。

- 列挙子を上に移動するには、**Component → Enumeration → Shift Up** を選択します。
- 列挙子を下に移動するには、**Component → Enumeration → Shift Down** を選択します。

これらのメニューコマンドは、すべて“3 Contents”ペイン内のショートカットメニューにも含まれています。

データベースアイテムを編集する：

- “1 Database”リストから、編集したいアイテムを選択します。
- アイテムをダブルクリックします。

または

- **Component → Edit Item** を選択します。

または

- **<Enter>** キーを押します。

アイテム用に所定のエディタが開きます。

フォルダやデータベースアイテムの名前を変更する：

- “1 Database”リストから、名前を変更したいフォルダまたはアイテムを選択します。
- **Edit → Rename** を選択します。

または

- **<F2>** キーを押します。

現在の名前が強調表示されて入力モードになります。

- 名前を編集してして **<Enter>** キーを押します。

フォルダやデータベースアイテムを削除する：

注記

フォルダおよびコンポーネントを削除すると、それらはデータベースから完全に削除されてしまい、復元することはできません。また、フォルダを削除するとそのフォルダ以内のアイテムもすべて削除されます。“Delete”操作を行う際は、細心の注意が必要です。

- “1 Database”リストから、削除したいフォルダまたはアイテムを選択します。
- **Edit → Delete** を選択します。

または

- **Delete** ボタンをクリックします。

または

- **<Del>** キーを押します。

  中に他のフォルダやアイテムが入っているフォルダを削除しようとすると、確認ダイアログボックスが開きます。空のフォルダの削除は、確認ダイアログは表示されずに無条件に行われます。

- **OK** をクリックして、削除を実行します。

フォルダ（内容を含む）やアイテムは、格納位置を移動したり、コピーを作成することができます。コピーまたは移動先のフォルダに同名のアイテムがすでに存在している場合、コピーまたは移動されたアイテムのデフォルト名は、元のアイテム名の後ろに _1 が付いたものになります。

コピーと移動は、一般のファイルシステムの場合と同様に、クリップボードを介して行われます。

アイテムまたはフォルダをクリップボードコピーする：

- コピーしたいアイテムまたはフォルダを選択します。
- **Edit → Copy** を選択します。

または

- **Copy** ボタンをクリックします。

または

- **<Ctrl> + <C>** キーを押します。

  アイテムがクリップボードにコピーされます。

アイテムまたはフォルダをクリップボードに移動する：

- 移動したいアイテムまたはフォルダを選択します。
- **Edit → Cut** を選択します。

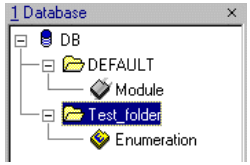
または

- **Cut** ボタンをクリックします。

または

- **<Ctrl> + <X>** キーを押します。

  アイテムがクリップボードに移動します。移動したアイテムは、アイテムリスト内では以下のようにモノクロ表示に変わります。



注記

クリップボード内にアイテムやフォルダが移動した状態でデータベースを閉じると、クリップボードは空になりますが、元のアイテムやフォルダは削除されずにデータベース内に残ります。

アイテムまたはフォルダをアイテムリスト内に貼り付ける：

- コンポーネントマネージャの“1 Database”リスト内で、クリップボードにコピー（または移動）したアイテムまたはフォルダを貼り付けたいフォルダを選択します。
- **Edit** → **Paste** を選択します。

または

- **Paste** ボタンをクリックします。

または

- **<Ctrl> + <V>** キーを押します。

  選択されたフォルダに、アイテムまたはフォルダが挿入されます。アイテムの移動を行った場合は、元のアイテムが削除されます。

アイテムまたはフォルダを同じデータベース内で移動（またはコピー）した場合、通常は移動先のフォルダ内でも移動前のアイテム名やフォルダ名がそのまま維持されますが、移動先のフォルダ内に同じ名前のアイテムまたはフォルダがすでに存在する場合は、移動（またはコピー）されたアイテムの名前は自動的に変更され、名前の衝突が回避されます。

既存のブロックダイアグラムコンポーネントを C コードまたは ESDL 形式に変更したい場合、またはその逆を行いたい場合は、オリジナルのコピーを行う際にコンポーネントのインターフェースをコピーして、再記述の手間を少なくすることができます。

コンポーネントの構成をコピーする：

- コンポーネントマネージャで、他の構成をコピーしたいコンポーネントを選択します。

- **Component → Reproduce As → *<item type>*** を選択して、新しく作成するコンポーネントのエディタを指定します。

  新しいコンポーネントが元のコンポーネントと同じフォルダに作成され、元の名前に "_1" という番号を付加したデフォルト名が割り当てられます。

  新しいコンポーネントには元のコンポーネントと同じインターフェースが定義されていますが、機能はまったく定義されていません。
- 新しいコンポーネントの名前を任意に編集します。
- 新しいアイテムをダブルクリックして所定のコンポーネントエディタを開き、機能を定義します。

ASCET データベースに対して何らかの変更を行うと、その変更内容は、RAM のキャッシュ内に保存されます。この変更内容を恒久的なものにするには、データベースをハードディスクに保存する必要があります。

データベースを保存する：

- **File → Save Database** を選択します。

または

- **Save** ボタンをクリックします。

または

- **<Ctrl> + <S>** を押します。

  キャッシュの内容がハードディスクに書き込まれます。

データベースは、指定の間隔で自動的に保存することができます。詳しくは 42 ページの「一般オプション」を参照してください。

2.3.2 各種ビューモードを利用した作業

コンポーネントマネージャでは、“3 Contents”ペインに表示されるさまざまなビューモードを利用してコンポーネントの編集などを行うことができます。これに関連するメニューコマンドは、19 ページの「メニューコマンドの概要」の項にまとめられています。

コンポーネントマネージャでのコンポーネントとプロジェクトの編集

各コンポーネントやプロジェクトのプロパティ（属性）は、コンポーネントマネージャで直接変更することができます。“3 Contents”ペインでエレメント（コンポーネントまたはプロジェクトを構成する部品）を選択し、プロパティ、データセット、インプリメンテーション、レイアウトなどを編集できます。

ただし、メソッドの引数と戻り値は、コンポーネントエディタ上でしか変更できません。

データベースアイテムを編集する：

- “1 Database”ペインでフォルダを選択します。

  そのフォルダのフォルダビュー、つまりフォルダに含まれるアイテムの情報が“3 Contents”ペインに表示されます。
- “Components”タブ上のアイテムをクリックします。
- **Components → Edit Item** を選択します。

または

- **<Enter>** を押します。

または

- 選択されて強調表示されているアイテムをダブルクリックします。

  所定のエディタが開きます。
- 選択されたアイテムの名前を変更するには、ショートカットメニューから **Rename** を選択します。

または

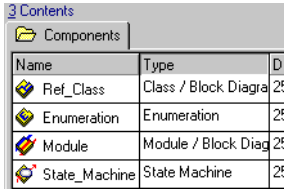
- **<F2>** キーを押します。
- すべてのアイテムを選択するには、ショートカットメニューから **Select All** を選択します。
- 選択されたアイテムを削除するには、ショートカットメニューから **Delete** を選択します。

または

- **<Del>** キーを押します。
- 任意のカラムを基準にアイテムをソートするには、ショートカットメニューから **Sort by → *<column>*** を選択します。

または

- ソートの基準としたいカラムのタイトルフィールドをクリックします。



昇順ソートの場合は数字、アルファベット（大文字）、アルファベット（小文字）の順で並び、降順ソートの場合はその逆です。

コンポーネントのエレメントを編集する：

- “1 Database”ペインでコンポーネントまたはプロジェクトを選択します。
- “3 Contents”ペインで、“Elements”タブをクリックします。

  「エレメントビュー」が表示されます。
- カラムタイトルをクリックすると、そのカラムを基準にエレメントがソートされます。
- “Elements”タブでエレメントを選択します。
- **Components** → **Edit Item** を選択します。

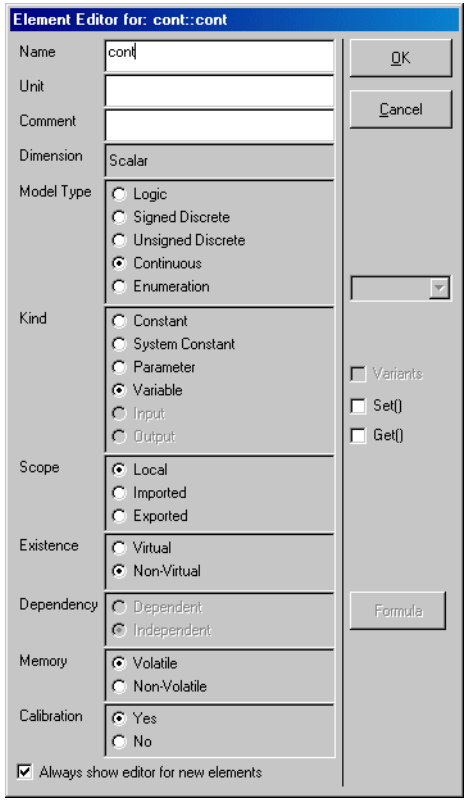
または

- **<Enter>** を押します。

または

- 選択されて強調表示されているアイテムをダブルクリックします。

  エレメントエディタが開きます。



エレメントのプロパティの編集については、4.10「エレメントプロパティの編集」に詳しく説明されています。

- 選択されたエレメントの名前を変更するには、**Edit** → **Rename** を選択します。

または

- **<F2>** を押します。
- すべてのエレメントを選択するには、ショートカットメニューから **Select All** を選択します。
- 選択されたエレメントをクリップボードにコピーするには、**Edit** → **Copy** を選択します。

または

- **<Ctrl> + <C>** を押します。
- クリップボードにコピーされているエレメントをコンポーネントに貼り付けるには、**Edit → Paste** を選択します。

または

- **<Ctrl> + <V>** を押します。

  同じ名前のエレメントがすでに存在している場合は、貼り付けられたエレメントの名前に番号 "_n" が付加されます。
- 選択されたエレメントを削除するには、**Edit → Delete** を選択します。

または

- **<Del>** キーを押します。

データを編集する：

- “1 Database”ペインで、編集したいコンポーネントまたはプロジェクトを選択します。
- “3 Contents”ペインの“Data”タブをクリックします。

  「データビュー」が表示されます。
- カラムタイトルをクリックすると、そのカラムを基準にエレメントがソートされます。
- 編集したいエレメントを選択します。
- **Component → Edit Item** を選択します。

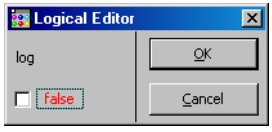
または

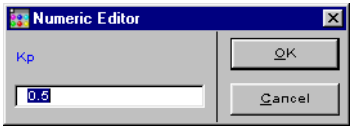
- **<Enter>** を押します。

または

- 選択したエレメントをダブルクリックします。

  エレメントのデータ型に対応するエディタが開きます。たとえば論理エレメントの場合は“Logical Editor”が開き、スカラエレメントの場合は“Numeric Editor”が開きます。





データとデータセットの編集については、4.11 項に詳しく説明されています。

- 選択されたエレメントの名前を変更するには、**Edit → Rename** を選択します。

または

- **<F2>** を押します。
- すべてのエレメントを選択するには、ショートカットメニューから **Select All** を選択します。
- 選択されたエレメントをクリップボードにコピーするには、**Edit → Copy** を選択します。

または

- **<Ctrl> + <C>** を押します。
- クリップボードにコピーされているエレメントをコンポーネントに貼り付けるには、**Edit → Paste** を選択します。

または

- **<Ctrl> + <V>** を押します。
- 選択されたエレメントを削除するには、**Edit → Delete** を選択します。

または

- **<Del>** キーを押します。

インプリメンテーションを編集する：

- “1 Database”ペインで編集したいコンポーネントまたはプロジェクトを選択します。
- “3 Contents””ペインの“Implementation”タブをクリックします。

  「インプリメンテーションビュー」が表示されます。
- カラムタイトルをクリックすると、そのカラムを基準にエレメントがソートされます。
- 編集したいエレメントを選択します。

  ここでは基本エレメントまたは適合カーブなどの複合エレメントを選択できます。
- **Component → Edit Item** を選択します。

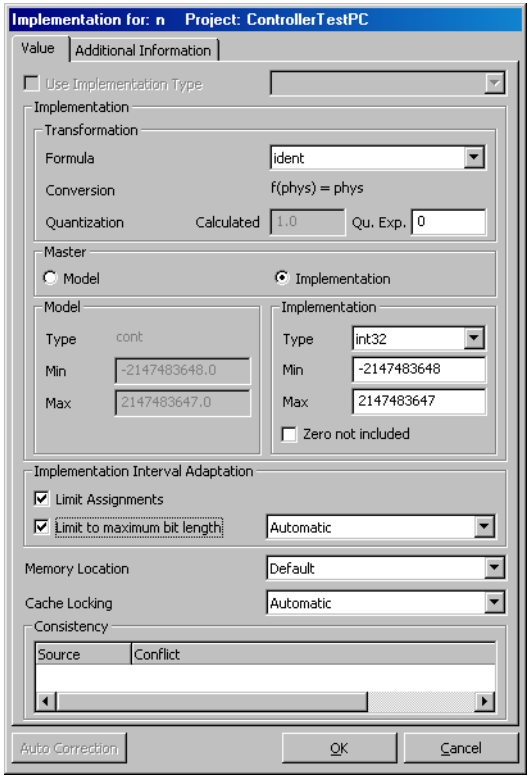
または

- **<Enter>** を押します。

または

- 選択したエレメントをダブルクリックします。

  エレメントの型に応じたインプリメンテーションエディタが開きます。次の図は、基本エレメントを選択した場合に開くインプリメンテーションエディタの例です。



インプリメンテーションの編集については、4.12 項に詳しく説明されています。

- 選択されたエレメントの名前を変更するには、**Edit → Rename** を選択します。

または

- **<F2>** を押します。
- すべてのエレメントを選択するには、ショートカットメニューから **Select All** を選択します。

- 選択されたエレメントのインプリメンテーションをクリップボードにコピーするには、**Edit → Copy** を選択します。

または

- **<Ctrl> + <C>** を押します。
- クリップボードにコピーされているインプリメンテーションを選択されたエレメントに貼り付けるには、**Edit → Paste** を選択します。

または

- **<Ctrl> + <V>** を押します。

注記

インプリメンテーションのコピーは、同じ型のエレメント間でのみ行えます。つまり、たとえば特性カーブエレメントのインプリメンテーションをスカラエレメントや配列エレメントにコピーすることはできません。

- 選択されたエレメントを削除するには、**Edit → Delete** を選択します。

または

- **<Del>** キーを押します。

レイアウトを編集する：

- “1 Database”ペインで編集したいコンポーネントを選択します。

注記

“Layout”タブは、プロジェクトを選択した場合は表示されません。

- “3 Contents”ペインの“Layout”タブをクリックします。

  「レイアウトビュー」が表示されます。
- **Component → Edit Layout** を選択します。

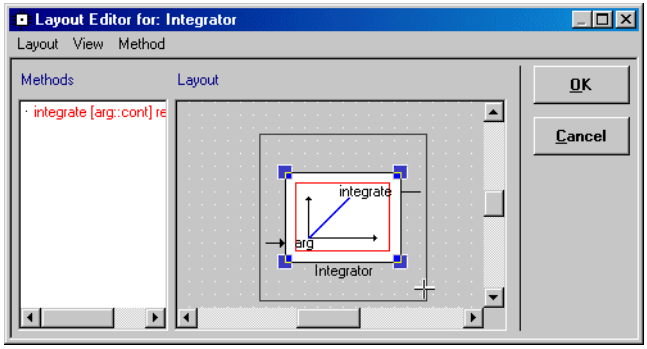
または

- **<Enter>** を押します。

または

- 表示されているレイアウトをダブルクリックします。

  レイアウトエディタが開きます。



コンポーネントレイアウトの編集については、4.13 項に詳しく説明されています。

別のデータセットやインプリメンテーションセットを選択する

コンポーネント内に複数のデータセットやインプリメンテーションセットが作成されている場合は、どのセットを使用するかをコンボボックスで選択することができます。このコンボボックスは、データビューとインプリメンテーションビューにのみ表示されます。

別のデータセットやインプリメンテーションセットを選択する：

- “Data”または“Implementation”タブで、左の図のようなコンボボックスをクリックします。
- データセットまたはインプリメンテーションセットを選択します。

  選択された内容に応じて、各エレメントの値やインプリメンテーションが変わります。

2.3.3 フォルダやデータベースアイテムのエクスポート

ASCET では、データベース間でフォルダやデータベースアイテムを交換することができます。エクスポートの範囲は、フォルダ全体、個々のデータベースアイテム、または任意に選択されたアイテムを指定できます。また選択されたアイテム

のみをエクスポートする方法（「フラットエクスポート」）と、選択されたアイテムによって参照されるアイテムも共にエクスポートする方法（「リカーシブエクスポート」）とがあります。

注記

エクスポート処理（49 ページの 2.2.3 項を参照してください）の際、“2 Comment”フィールドのコメントがエクスポートされるのはトップレベルのフォルダとアイテムのみです。**サブフォルダ内のコンポーネントのコメントはエクスポートされません。**

バイナリエクスポート

エクスポートファイル `*.exp` はコンパクトなバイナリフォーマットを使用しているため、エクスポート／インポートを迅速に行うことができます。ただし、このファイルを生成する際には非常に大きなメモリスペースが必要になる可能性があるため、大きなフォルダを交換するような場合は、その内容を複数のファイルに分散させることによってエクスポート／インポート処理を高速化することができます。このような場合、自動的に各アイテムごとに 1 つのファイルを生成させるように指定することができます。

オプションウィンドウで **Write XML file on Binary Export** または **Write ASCII file on Binary Export** オプション（61 ページ「エクスポートオプション」参照）がオンになっていると、それぞれのフォーマットでディスクリプションファイルが生成されます。これらのファイルはインターネットエクスプローラまたはテキストエディタで表示できます。

注記

XML フォーマットのディスクリプションファイルを生成するには、Microsoft Internet Explorer V5.5（MSXML Parser V3.0 を含む）または V6 が必要です。

AMD エクスポート

ASCET 5.2 では XML ベースのエクスポートフォーマット `*.amd` がサポートされています。このフォーマットを選択すると、エクスポートされるアイテムごとに `*.amd` ファイルのセットが作成され、これらのファイルにモデル記述が保存されます。ファイル名は以下のようになります。

```
<component name>.<information type>.amd
```

表 2-3 に、information type の種類と、そのファイルに保存される内容をまとめます。

Information type	ファイルの内容
main	プロジェクト／コンポーネント内のエレメントの名前とプロパティ（439 ページ参照）
data	コンポーネント／プロジェクト内のエレメントのデータ（449 ページ参照）
experiments	コンポーネント／プロジェクトに定義された実験環境（547 ページ参照）
implementation	コンポーネント／プロジェクト内のエレメントのインプリメンテーション（466 ページ参照）
specification	モデル記述の詳細（インターフェース情報、ブロックダイアグラムの内容、ESDL または C コンポーネントのコードなど）
project[a]	プロジェクト固有のデータ：オペレーティングシステムの設定（383 ページ参照）、プロジェクト設定（400 ページ参照）など
project.formulasa	プロジェクト内に定義された変換式（416 ページ参照）
project.implementationTypesa	プロジェクト内に定義されたインプリメンテーション型（422 ページ参照）

a. プロジェクトをエクスポートした場合のみ

表 2-3　AMD エクスポートファイルの種類

各 AMD ファイルのインポート時には、ファイルに書き込まれている「シグネチャ」が参照され、ファイルの内容がエクスポートされた後に変更されているかどうかがチェックされます。

複数の AMD ファイルをエクスポートする場合、それらを 1 つの ZIP ファイルに圧縮することができます。ZIP ファイルの拡張子は *.axl です。

その他のエクスポートフォーマット

上記の 2 つのフォーマットに加え、XML フォーマット（*.xml）も使用できます。これはバイナリエクスポートを行う際に、「XML ディスクリプション」として補助的に作成されるものです。この XML フォーマットは AMD フォーマットとは全く異なるもので、ASCET にインポートすることはできません。

ASAM-MCD-2MC プロジェクトのエクスポートには、*.a2l フォーマットが使用されます。

注記

ASAM-MCD-2MC プロジェクト以外のものを *.a2l フォーマットでエクスポートしようとすると、エラーが発生します。

エクスポートを行う

フォルダまたはデータベースアイテムをエクスポートする前に、エクスポートモード（被参照アイテムを含めるかどうか）、複数ファイルへの分散などを、ユーザーオプションで選択してください（61 ページの「エクスポートオプション」を参照してください）。

フォルダやデータベースアイテムをエクスポートする：

- コンポーネントマネージャの“1 Database”リストから、エクスポートしたいフォルダまたはアイテムを選択します。
- **File** → **Export** を選択します。

または

- ショートカットメニューから **Export** を選択します。

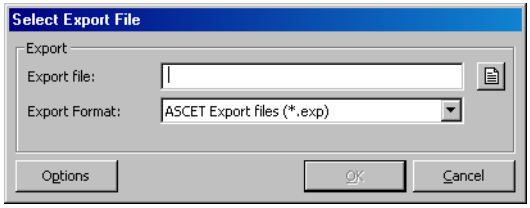
または

- **Export** ボタンをクリックします。

または

- **<Ctrl> + <E>** を押します。

“Select Export File”ダイアログボックスが開きます。



- “Export Format”フィールドで、エクスポートフォーマットを選択します。

`ASAP2 files (*.a2l)`	ASAM-MCD-2MC プロジェクトのみ
`ASCET Model Data files (*.amd)`	AMD エクスポート（91 ページ参照）
`ASCET compressed Model Data files (*.axl)`	AMD エクスポート（圧縮）
`ASCET Export files (*.exp)`	バイナリエクスポート（91 ページ参照）
`ASCET XML files old format (*.xml)`	XML エクスポート（ASCET にインポート不可能）

- “Export File”フィールドにインポートファイルのパスと名前を入力します。マニュアル入力、または  ボタンを使用します。

注記

エクスポートオプション **One File for each Item** をオンにすると、このフィールドのタイトルは“Export folder”に変わり、ボタンも  に変わります。

OK ボタンが有効になります。

- エクスポートオプションを変更する場合は、**Options** をクリックします。

  “Options”ウィンドウが開き、“Export”ノードが表示されます（「エクスポートオプション」の項を参照してください）。

  — エクスポートオプションを設定します。

  — **OK** をクリックして“Options”ウィンドウを閉じます。

- “Select Export File”ダイアログボックスの **OK** ボタンをクリックすると、エクスポート処理が開始されます。

  指定されたオブジェクトが、指定のファイルにエクスポートされます。

  AMD エクスポートの場合、各オブジェクト一連の `*.amd` ファイルが生成され、実行中は、モニタウィンドウに進行状況が表示されます

注記

各アイテムごとに 1 つのファイルにエクスポートしたり（**One File for each Item** オプションがオンの場合）、AMD エクスポートを行う際、被参照アイテムを含めたエクスポートを行う場合は、空のディレクトリをエクスポート先に指定するようにしてください。エクスポート先のデフォルトパスは、“Default Export Path”オプションタブで指定します。

2.3.4 データベースアイテムのインポート

ASCET データベース内の各オブジェクトにはそれぞれユニークな識別タグ（「オブジェクト ID」と呼ばれます）が付けられていて、ASCET は各オブジェクトを、コンポーネントマネージャに表示される名前ではなくこの ID によって識別します。

下記のフォーマットのエクスポートファイルをデータベースにインポートでき、複数のファイルからのインポートも可能です。

バイナリエクスポートファイル（91 ページ参照）	`ASCET Export files (*.exp;*.prj)`
AMD エクスポートファイル（91 ページ参照）	`ASCET Model Data files (*.main.amd; *.main.xml)`
圧縮された AMD エクスポートファイル	`ASCET compressed Model Data files (*.axl;*.zip)`
ASAM-MCD-2MC プロジェクト	`ASAP2 files (*.a2l)`

XML エクスポートファイル（92 ページ）はインポートできません。

インポートの際に必要となるフォルダやアイテムは自動的に生成されます。インポートによって既存のフォルダやアイテムが上書きされる場合は、その前にユーザーに対して確認が求められます。

エクスポートファイルには、アイテム自体のデータ以外に、データベース内のアイテムのパスも保存されます。インポート先のデータベースに同じパス（フォルダ）が存在する場合は、アイテムはそのフォルダにインポートされます。存在しない場合はフォルダが自動的に作成され、元のデータベースが複数の階層フォルダで構成されていた場合、同じ階層フォルダが作成されます。

インポートされるアイテムと同じアイテムがすでにターゲットデータベースに存在する場合、そのアイテムは上書きされます。この時、アイテム間の対応付けは表示されている名前ではなく「オブジェクト ID」により行われるため、アイテム名が変更されていても上書きは行われてしまいます。AMD インポートの場合は、**Remap OID** オプションを利用して、既存のアイテムが上書きされないようにすることができます。この際、新しいオブジェクト ID を持つアイテムのコピーが作成されます。

インポートされるアイテムによって既存のアイテムが上書きされる場合、それが実際にどの位置にインポートされるかは、インポートオプション（63 ページの「インポートオプション」を参照してください）で指定されます。**Keep Folder Path in Database** オプションがオフになっていると、インポートされたアイテムはエクスポートされた時のデータベースパスに格納されます。現在開いているデータベースのパスに格納したい場合は、このオプションをオンにしておいてください。

必要に応じて、個々のアイテムについてインポートによる上書きを禁止することもできます。

インポート時のデータの上書きを禁止する：

- コンポーネントマネージャで、保護したいアイテムやフォルダを選択します。

- **Component → Disallow Import** を選択します。

  選択されたアイテムやフォルダが、インポート時に上書きされなくなります。保護されてたアイテムには、“1 Database”リストのアイテム名に `<Disallow Import>` というマークが付きます。

フォルダをインポート禁止にした場合、上書き禁止になるのはそのフォルダ自体のみです。

インポートを行う

フォルダやアイテムをインポートする前に、ユーザーオプションのインポートオプションを設定してください（63 ページの「インポートオプション」を参照してください）。

1 つのエクスポートファイルからのインポートを行う：

- コンポーネントマネージャで、インポート先となるデータベースを開きます。
- AMD インポート、または ASAM-MCD-2MC ファイルのインポートを行う場合は、インポート先のフォルダを選択します。
- **File → Import** を選択します。

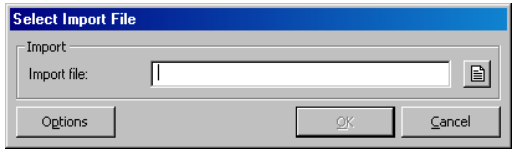
または

- **Import** ボタンをクリックします。

または

- **<Ctrl> + <M>** キーを押します。

  “Select Import File”ダイアログボックスが開きます。



- “Import File”フィールドにインポートファイルのパスと名前を入力します。マニュアル入力、または  ボタンを使用します。

注記

ファイルの複数選択も可能です。

ファイルフォーマットは、ファイル拡張子から自動的に識別されるので、明示的に指定する必要はありません。

OK ボタンが有効になります。

- インポートオプションを変更する場合は、**Options** をクリックします。

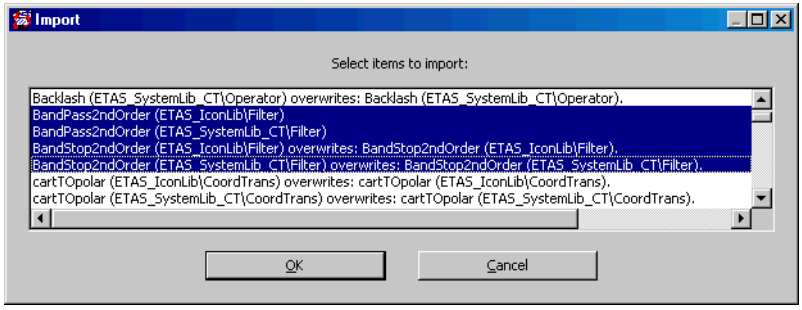
  “Options”ウィンドウが開き、“Import”ノードが表示されます（63 ページの「インポートオプション」の項を参照してください）。

  — インポートオプションを設定します。

  — **OK** をクリックして“Options”ウィンドウを閉じます。

- “Select Import File”ダイアログボックスの **OK** ボタンをクリックすると、インポート処理が開始されます。

  “Import”または“Items available for import”ダイアログボックスが開き、指定されたエクスポートファイルの内容が一覧表示されます。その中に現在データベース上にすでに存在しているもの（つまり同じオブジェクト ID を持つアイテム）がある場合、そのアイテムの行に“`... overwrites: ...`”というテキストで示されます。



- インポートするアイテムをすべて選択し、**OK** をクリックします。

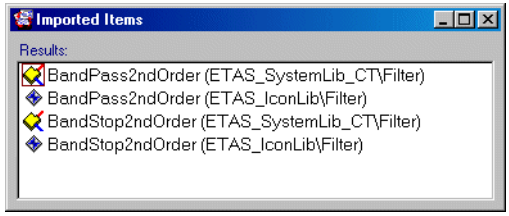
  データベース上にすでに存在しているアイテムが選択されていた場合、上書きしてよいかを確認するダイアログボックスが開きます。

注記

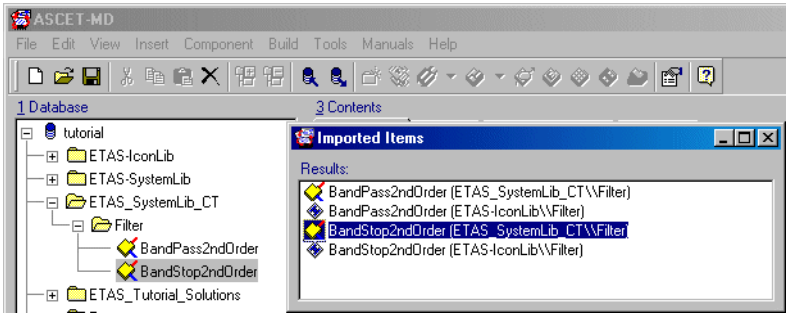
AMD フォーマット用のインポートオプションである **Remap OID** がオンになっている場合、現在のデータベース内に存在するアイテムは上書きされず、新しいオブジェクト ID を持つコピーが作成されます。

- メッセージを確認し、**OK** でダイアログボックスを閉じます。

  “Import” ダイアログボックスで選択されたアイテムがインポートされ、すべて終了すると、“Imported Items” ダイアログボックスが開いて、実際にインポートされたアイテムの一覧が表示されます。



- “Imported Items” ダイアログ内のアイテムをクリックすると、そのアイテムがコンポーネントマネージャ上で強調表示されます。



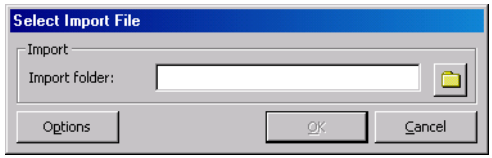
ディレクトリ全体のインポート

あるディレクトリに含まれるすべてのエクスポートファイルをインポートするには、**File → Import directory** を使用します。

ディレクトリ全体をインポートする：

- コンポーネントマネージャで、**File → Import directory** を選択します。

  “Select Import File”ダイアログボックスが開きます。



- “Import Folder”フィールドにインポートファイルのパスと名前を入力します。マニュアル入力、または  ボタンを使用します。

  OK ボタンが有効になります。

- インポートオプションを変更する場合は、**Options** をクリックします。

  “Options”ウィンドウが開き、“Import”ノードが表示されます（63 ページの「インポートオプション」の項を参照してください）。

  — インポートオプションを設定します。

  — **OK** をクリックして“Options”ウィンドウを閉じます。

  最初のインポートファイルについて、“Import”または“Items available for import”ダイアログボックスが開き、指定されたエクスポートファイルの内容が一覧表示されます。その中に現在データベース上にすでに存在しているもの（つまり同じオブジェクト ID を持つアイテム）がある場合、“`... overwrites: ...`”というテキストがそのアイテムの行に示されます。

- インポートするアイテムをすべて選択し、**OK** をクリックします。

  データベース上にすでに存在しているアイテムが選択されていた場合、上書きしてよいかを確認するダイアログボックスが開きます。

- メッセージを確認し、**OK** でダイアログボックスを閉じます。

  ダイアロ選択されたアイテムがインポートされ、終了すると、次のインポートファイルについて“Import”または“Items available for import”ダイアログボックスが開きます。

  すべてのインポートが終了すると、インポートされたアイテムの一覧が表示されます。

AMD インポートの特殊機能

AMD ファイルをインポートすると、インポート処理中に以下のようなテストが行われます。

正当性チェック（“Integrity check”）： 署名を使用して、ファイルが第三者によって変更されているかどうかをチェックします。

整合性チェック（“Consistency check”）： 以下のエラーがないかをチェックします。

- 「無効なファイル」の有無

  AMD ファイルの中に 1 つでも無効なファイル（読み込めない、XML に準拠していない、所定の XML スキーマに準拠していない、など）があった場合、以下のように処理されます。

  — コンポーネントのインポートは行われません。

  — “Import Problems”ダイアログボックスが開き、問題のあるファイルとその内容が表示されます。

  — 自動的にこれを解決する処理は行われません。

- 「不足しているファイル」の有無

  ASCET の 1 つのコンポーネントは、複数のファイルに分割して保存されます（「AMD エクスポート」を参照してください）。これらのうちで足りないものがあると、以下のように処理されます。

  — コンポーネントのインポートは行われません。

  — “Import Problems”ダイアログボックスが開き、足りないファイルが表示されます。

  — ユーザーの指示により、足りないファイルに代わる「デフォルトファイル」が作成されます。

- 「不足しているコンポーネント」の有無

  あるコンポーネントによって参照されているコンポーネントが AMD ファイル内に保存されていないか、または使用できない場合、“Import Problems”ダイアログボックスが開いて、足りないファイルが表示されます。

- ファイル内で「不足している情報」の有無

  AMD ファイル内で不足している情報（あるエレメントが定義されていて、そのデータが保存されていない、など）があると、もしファイル自体が有効（読み込み可能で、XML フォーマットと所定のスキーマに準拠していること）であれば、以下のように処理されます。

  — コンポーネントのインポートは行われません。

  — “Import Problems”ダイアログボックスが開き、足りないファイルが表示されます。

  — ユーザーの指示により、足りないファイルに代わる「デフォルト値」が作成されます。

- ファイル内の「誤った情報」の有無

  AMD ファイル内に誤った情報（あるコンポーネントのデータが、他の AMD ファイルで定義されている情報に一致しない、など）があると、以下のように処理されます。

  — コンポーネントのインポートは行われません。

  — “Import Problems”ダイアログボックスが開き、誤った情報が表示されます。

  — 自動的にこれを解決する処理は行われません。

- ファイル内の「不要な情報」の有無

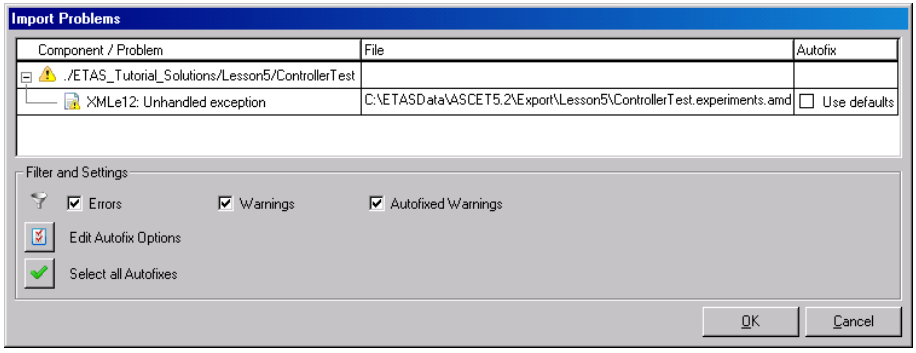
  AMD ファイル内に必要でない情報（実際に存在しないエレメントについてのデータコンフィギュレーションなど）があると、以下のように処理されます。

  — コンポーネントのインポートは行われません。

  — “Import Problems”ダイアログボックスが開き、不要な情報が表示されます。

  — ユーザーの指示により、不要な情報が無視されて処理が実行されます。

上記のような問題が検出されると、“Import Problems”ウィンドウに以下のような情報が表示されます。



- “Component / Problem”列

  影響を受けるコンポーネントと、問題のタイプ（例：`unhandled exception`、`Invalid AMD file (missing)`、`Data for Element <name> missing`、`Obsolete data for undefined element <name>`）

  発生した問題は、ワーニング（シンボル  ／  ）またはエラー（シンボル  ／  ）に分類されます。
- “File”列

  問題の発生したファイルのパスと名前
- “Autofix”列

  自動修復処理（ 実行するアクションが選択されている必要があります）

  実行できる自動修復処理がない問題については、この列には何も表示されません。
- テーブル内のショートカットメニューに含まれるコマンド
  - **Expand all Nodes**（**<Alt> + <*>**）

    “Component / Problem”列の全ノードを展開します。
  - **Select All Autofixes**（**<Ctrl> + <A>**）

    “Autofix”列のすべての自動修復処理をオン（有効）にします。
  - **Select Autofixes of this Type**（**<Alt> + <A>**）

    このコマンドは、“Autofix”列のいずれかのエントリが選択されている場合にのみ有効です。選択されている自動修復処理と同じタイプの処理をすべてオン（有効）にします。
  - **Always Autofix Problems of this Type**

    このコマンドは、“Autofix”列のいずれかのエントリが選択されている場合にのみ有効です。選択されている自動修復処理と同じタイプの処理が、常に自動的に実行されるようにします。このコマンドを実行すると、それらの処理は“Import Problems”ウィンドウ上でオン（有効）になっていなくても、常に実行されます。
  - **Save Issue List as XML**（**<Ctrl> + <S>**）

    ウィンドウの内容を XML ファイルに保存します。エラーのタイプによっては、所定の情報が追加されます。

    XML ファイルに保存しておくと、問題点を自動的に分析する際に役立ちます。
- フィルタ機能
  - **Errors**

    「エラー」タイプの問題のみを表示します。

- **Warnings**

  「ワーニング」タイプの問題のみを表示します。

- **Autofixed Warnings**

  自動修復された「ワーニング」タイプの問題のみを表示します。

- **Edit Autofix Options** ボタン（  ）

  “Options”ウィンドウが開き、問題の自動修復に関する設定を行えます。

- **Select all** ボタン（  ）

  “Autofix”列に表示されているすべての自動修復処理を有効にします。

- **OK** ボタン

  ウィンドウを閉じ、有効になっている自動修復処理を実行します。

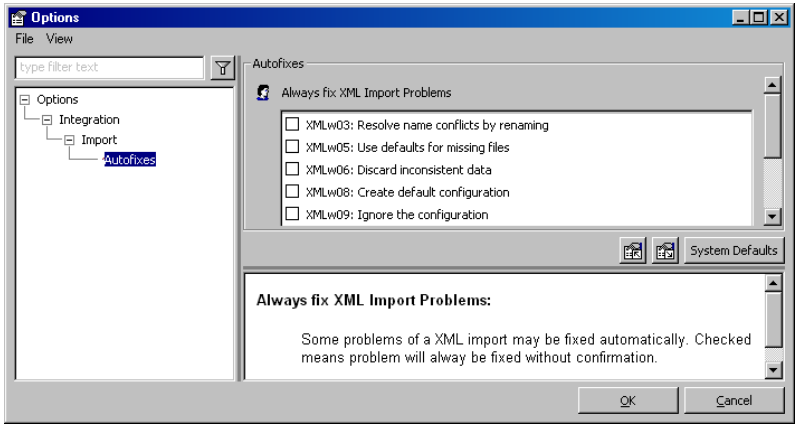
- **Cancel** ボタン

  ウィンドウを閉じます。自動修復処理は実行されません。

自動修復処理のオプションを設定する：

- “Import Problems”ウィンドウで、**Options** をクリックします。

  “Options”ウィンドウが開き、“Import”ノードのみが表示されます。



  “Always fix AMD Import Problems”フィールドに、AMD インポート時に実行できる自動修復処理の一覧が表示されます。

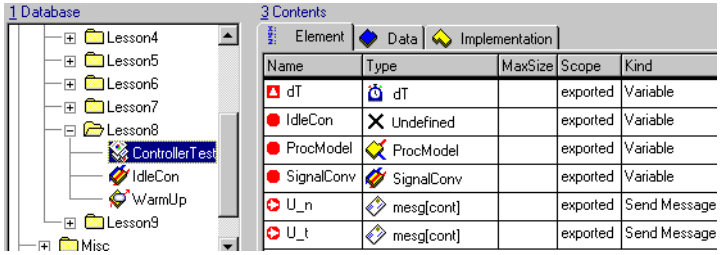
- ユーザーの確認を行わずに（“Import Problems”ウィンドウを開かずに）実行したい処理をすべてオンにします。

- **OK** をクリックして“Options”ウィンドウを閉じます。

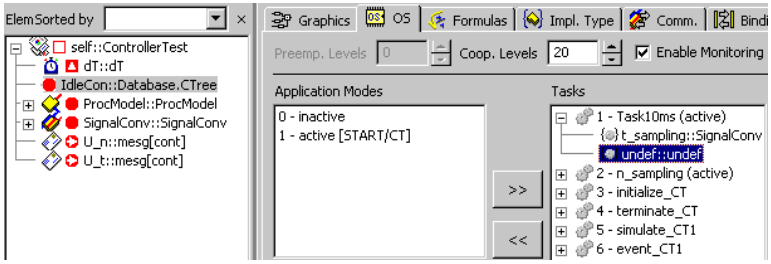
  これ以降、AMD インポートでエラーでのエラー発生時には、有効になっている修正処理は自動的に実行されるので、エラー発生時に“Import Problems”ウィンドウでオプションを個別に選択する必要はなくなります。

プロジェクトのインポート

プロジェクトも、他のデータベースアイテムと同様にインポートすることができます。インポートされたプロジェクトに属するアイテムのうち、データベース内に存在しないものがあった場合、その旨がコンポーネントマネージャのコンポーネントビュー内に表示されます。



このようなプロジェクトをプロジェクトエディタ（363 ページの 4.8「プロジェクトエディタ」の章を参照してください）で開くと、足りないエレメントは“Elements”リストに表示されます。また“OS”タブ上のオペレーティングシステムエディタ（383 ページの「OS エディタでのスケジューリングの定義」を参照してください）では、失われたプロセスは“Process”フィールドから削除されますが、タスクリスト内には未定義な参照（`undef::undef`）として表示されます。



足りないモジュールは、プロジェクトが開いた状態で、後からインポートすることができます。インポートが正常に行われれば、その時点でプロジェクト内の参照は解決され、プロセスは自動的に元のタスクに組み込まれます。

旧バージョンのASCET で編集されたアイテムのインポート

ASCET V5.2 のデータベースには ASCET V4.0 より古いバージョンのデータベースフォーマットとの互換性がないため、これらのバージョンで作成されたエクスポートファイルを直接 ASCET V5.2 にインポートすることはできません。このようなエクスポートファイルを 96 ページの方法でインポートしようとすると、以下のようなエラーメッセージが表示されます。

```
This file is compatible with ASCET-SD V3.0 or earlier. It
cannot be imported directly. Convert it to a database
using your old version of ASCET-SD, then open this
database.
```

現在 ASCET-SD V4.x をお持ちの場合、以下のような方法で、V4.0 より古いバージョンのデータベースを使用することができます。

V4.0 より古いバージョンの ASCET-SD からのインポートを行う：

- ASCET-SD V4.x を起動します。

  これらのバージョンは、それより前のバージョンのデータベースフォーマットをサポートしています。
- 古いバージョンのエクスポートファイルをインポートします。

  インポートされたデータベースアイテムは、現行のフォーマットに変換されます。
- インポートしたアイテムをエクスポートします。
- ASCET V4.x を閉じます。
- ASCET V5.2 を起動します。
- V4.xでエクスポートしたファイルを96ページの方法でインポートします。

2.3.5 データベースアイテムの扱い

アイテムへの参照

アイテムへの「参照」は、あるアイテムが別のアイテムに使用されたり（たとえばあるモジュールがあるプロジェクトから参照される場合）、アイテムが他のアイテムに内包されるたびに、必ず生成されます。フォルダ間でアイテムを移動したりフォルダ内の既存のアイテムの名前を変更すると、そのアイテムへの「参照」は自動的に更新されます。

データベース内で参照が変更された場合、その内容はコンポーネントマネージャ上では自動的には更新されません。コンポーネントマネージャに表示される内容をデータベースの内容と一致させるには、明示的に表示を更新する必要があります。

コンポーネントマネージャ上の参照を更新する：

- コンポーネントマネージャで、**View → Update** を選択します

または

- **<F5>** キーを押します。

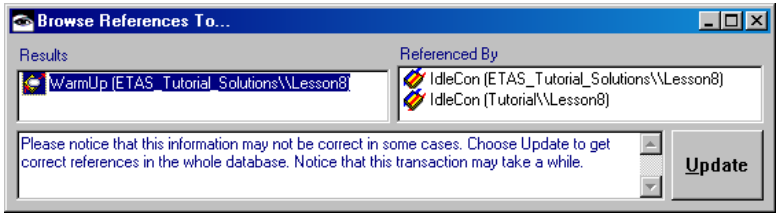
  コンポーネントマネージャの表示内容がデータベースと比較され、異なる点があればデータベースの内容に合わせて再表示されます。

アイテムまたはフォルダをデータベースから削除する際は、データベース内の整合性を確保して安全に削除を行えるように、参照の一覧を表示して確認することができます。

データベースアイテムへの参照を表示する：

- データベースアイテムを選択します。
- **Component → Show References** を選択します。

  “Browse References To”ダイアログボックスが開きます。



選択されたアイテムを参照するアイテムの名前がすべて表示されます。

「再帰的」な参照、つまり、あるアイテムを、そのアイテムが参照するアイテムから参照することも可能なため、常に被参照アイテムが失われないように注意する必要があります。他のアイテムに参照されているアイテムを削除しようとすると、ASCET は警告および参照の一覧を表示します。

他のアイテムに参照されるデータベースアイテムを削除すると、参照側のアイテムが破損してしまう可能性があります。このような事態を防ぐため、削除したい被参照アイテムを他のアイテムに置き換えることができます。

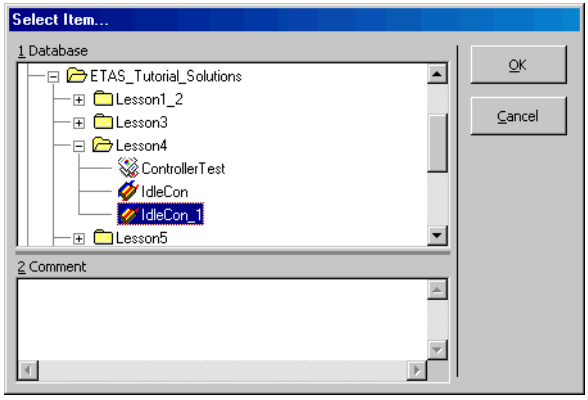
被参照アイテムを別のアイテムに置き換えると、元のアイテムを参照しているすべてのデータベースアイテムが、置き換えられたアイテムを参照するようになり、元のアイテムはどのアイテムからも参照されなくなります。

データベースアイテムへの参照を置換する：

- コンポーネントマネージャの“1 Database”リストから、置換したいアイテムを選択します。

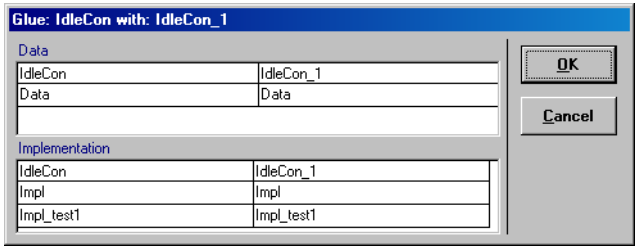
- **Component → Replace References** を選択します。

  “Select Item”ダイアログボックスが開きます。



- 最初に選択したアイテムと置換したいアイテムを選択して、**OK** をクリックします。
- “Confirm”ダイアログボックスで **OK** をクリックして確定します。

  “Glue: <item> with: <item>”ダイアログボックスが開き、実行されるすべての置換内容が一覧表示されます。



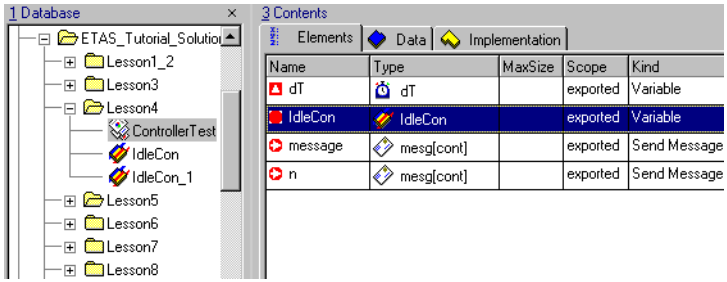
- **OK** をクリックします。

  最初に選択したアイテムへの参照が、すべて新しいアイテムへの参照に置き換えられます。

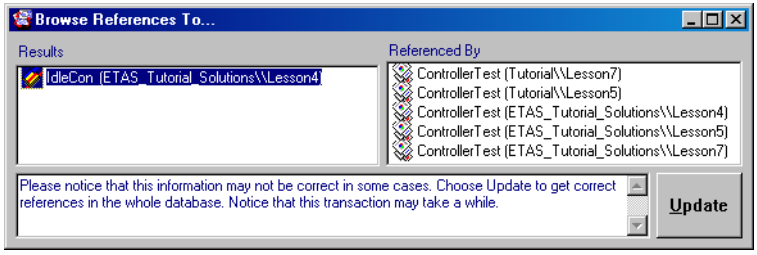
この処理は、あるアイテムへの参照を別のアイテムへの参照に置き換えますが、データベースアイテムそのものに対しては影響を与えません。つまり、処理に関連したアイテムが削除されたり別の位置に移動するようなことはありません。

ここで、**Component → Replace Reference** の実際の処理内容について、例を用いてわかりやすく説明します。

下図では、“tutorial”データベースの Lesson4 フォルダ内にある ControllerTest プロジェクトの内容が、コンポーネントマネージャのエレメントビューで表示されています。このプロジェクトは IdleCon というコンポーネントを参照しています。



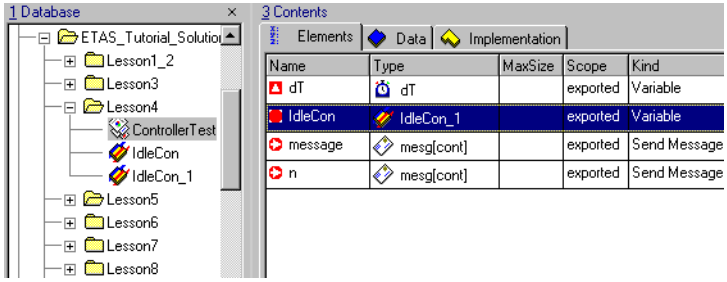
“1 Database”リストから IdleCon を選択して **Component → Show References** を選択すると、その参照関係が表示されます。



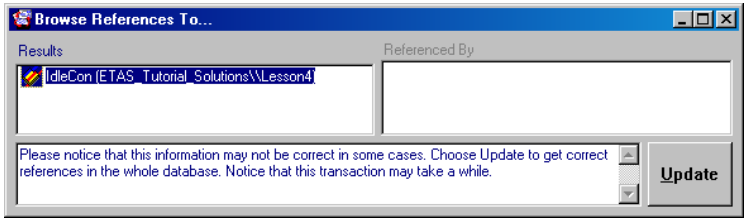
この表示内容から、IdleCon コンポーネントはデータベース内の 5 つの同名のプロジェクト（ControllerTest）から参照されていることがわかります。

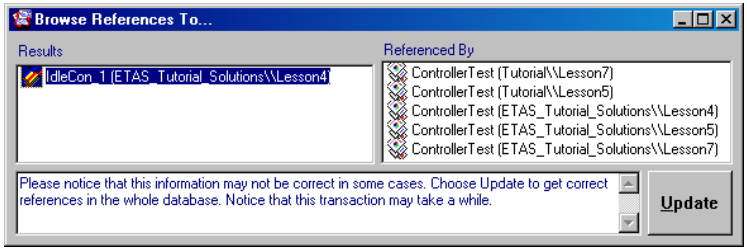
ここで前述の **Item → Replace References** を実行して、IdleCon への参照を IdleCon_1 への参照に置き換えます。このコマンドが実行されると、次の図に示されるように、“1 Database”ペインの内容には変化は見られませんが、エレメ

ントビューでは ControllerTest コンポーネントが IdleCon という名前を用いて IdleCon_1 を参照するようになります。ただしデータベース内にはこれらの積分器は両方とも以前と同様に存在しています。



Component → Show References を用いてこれら 2 つの積分器について調べると、以下のような結果になります。





IdleCon への参照はなくなり、ControllerTest は IdleCon_1 を参照するようになりました。

また ASCET 5.2 では、あるデータベースアイテムそのものを完全に他のアイテムに置き換え、同時にそのアイテムへの他のコンポーネントやプロジェクトからの参照をすべての置き換える、ということができます。この機能は、たとえばコンフィギュレーション管理ツールが使用されていて、複数の開発ストリームが 1 つのシステムにまとめられているような場合に、特に有効です。

データベースでは各データベースアイテムをそのオブジェクト ID によってのみ認識するので、あるアイテムに別のアイテムのオブジェクト ID を与えるだけで、別のアイテムの代わりとして扱うことができます。たとえば、データベース内に A1 と A2 という 2 つのオブジェクトがあり、A1 を A2 に置き換えると、オブジェクト A2 は元の機能と名前を持ったまま A1 の識別子と参照を持つようになります。

データベースアイテムを置換する：

- コンポーネントマネージャの“1 Database”リストから、元のアイテムの代わりにするアイテムを選択します。
- メニューコマンド **Component → Become another item** を選択します。

  “Select Item”ダイアログボックスが開きます。
- 最初に選んだ第 1 のアイテムに置き換えたいアイテムを選択します。
- **OK** をクリックして、第 2 のアイテムへの参照と第 2 のアイテム自体の両方を置き換えます。
- “Confirm”ダイアログボックスで **OK** をクリックして確定します。

  “Glue *<item>* with: *<item>*”ダイアログボックスが開きます。すべての置き換えが一覧表示されます。
- **OK** をクリックして実行します。

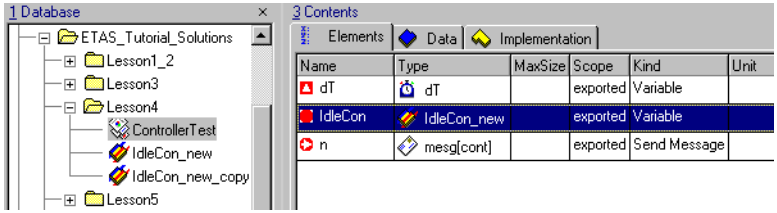
  第 2 のアイテムに置き換わる第 1 のアイテムの元のインスタンスがその ID を保持し、*< 第 1 のアイテム名 >*_copy という名前に変わります。

このコマンドの機能を具体的に示す例を紹介します。前出の例と似た操作を行いますが、ここでは `IdleCon` コンポーネントに対する参照だけでなく `IdleCon` コンポーネント自体も完全に置き換えられます。

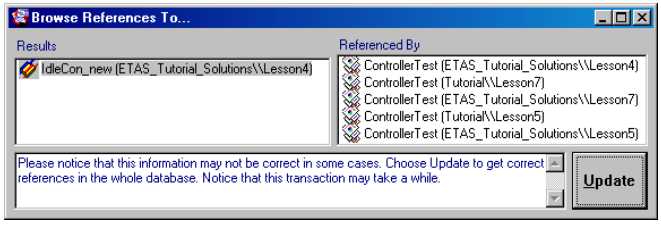
まず、**Component → Become another Item** コマンドが実行される前は、`ControllerTest` モジュールが `IdleCon` コンポーネントを参照していて、別のプロジェクト（`Project`）が `IdleCon_new` というコンポーネントを参照しているものとします。

ここで **Component → Become another Item** コマンドを実行すると、`ControllerTest` は `IdleCon` コンポーネントの代わりに `IdleCon_new` コンポーネントを用いるようになります。このコマンドを実行すると、`IdleCon` コン

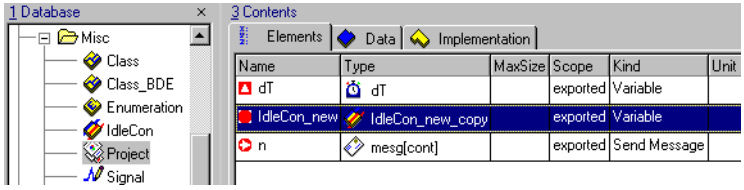
ポーネントはコンポーネントマネージャ上から削除され、“1 Database”リストには IdleCon_new および IdleCon_new_copy というコンポーネントが表示されます。

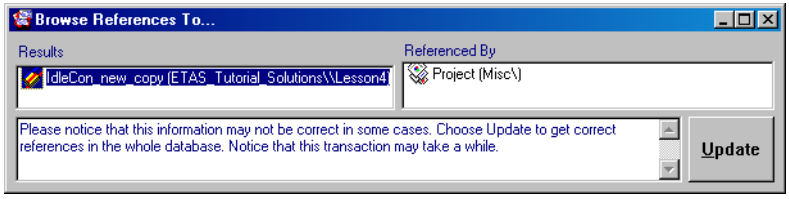


エレメントビューを見ると、ControllerTest が IdleCon という名前で IdleCon_new を参照するようになったことがわかります。これは、このコンポーネントに対して置き換えられた IdleCon コンポーネントの識別子が与えられたためです。



IdleCon_new_copy コンポーネントには、IdleCon_new コンポーネントの元のインスタンスが入っているので、このコンポーネントがプロジェクト Project によって参照されます。





編集

コンポーネントのレイアウト編集は、コンポーネント用のブロックダイアグラムエディタを開かなくてもコンポーネントマネージャから行えます。コンポーネントのパブリックインターフェースの宣言は、レイアウトエディタで行います。

コンポーネントのレイアウトを編集する：

- コンポーネントを選択します。
- **Component → Edit Layout** を選択します。

  選択されたコンポーネント用のレイアウトエディタが開きます。このエディタについては、500 ページの「コンポーネントのレイアウトの編集」で説明されています。

データベースアイテムには、コメントテキストと注釈（“notes”）という 2 種類のテキストを加えることができます。アイテムのコメントは、コンポーネントマネージャの“2 Comment”ペインに入力することによって自動的に格納されます。またアイテムの注釈は、専用のエディタウィンドウから入力します。自動生成されたドキュメントに出力されるのは、この注釈のみです。

データベースアイテムの注釈を編集する：

- 注釈を編集したいデータベースアイテムを選択します。
- **Components → Notes** を選択します。

  選択したデータベースアイテム用の注釈エディタが開きます。詳細は 7.4「注釈」の項を参照してください。

コード生成時の変数の扱いは、“Memory”属性（440 ページの「エレメントの設定」を参照してください）によって決まります。初期値の代入は Volatile エレメントに対してのみ自動的に行われ、Non-Volatile エレメントについては行われません。

以下のようにすると、データベース内のすべての変数に Volatile 属性を与えることができます。

すべての変数に Volatile 属性を割り当てる：

- コンポーネントマネージャで、**Tools → Database → Convert → Variables to Volatile** を選択します。

  すべての変数が Volatile 属性となり、自動的に初期化が行われるようになります。

また、データベース内のすべてのパラメータに Non-Volatile 属性を与えるには、以下のように操作します。

すべての変数に Non-Volatile 属性を割り当てる：

- コンポーネントマネージャで、**Tools → Database → Convert → Parameters to Non-volatile** を選択します。

  すべてのパラメータが Non-Volatile 属性となり、初期化時に値が書き込まれなくなります。

C コード／ESDL コンポーネント内の検索と置換

コンポーネントマネージャにおいて、データベース内のすべての C コードおよび ESDL コンポーネントに含まれる文字列を検索したり置換することができます。文字列の置換は、個別に、またはコンポーネント内やデータベース内のすべての「オカレンス」（コード内で実際に使用されているエレメント名）について一括して行うことが可能です。

この機能は、メッセージなどのグローバルエレメントをより扱いやすくするためのものです。同じメッセージはその名前で結び付けられるため（『ASCET リファレンスガイド』の「プロセス間通信」の項を参照してください）、メッセージの名前を 1 個所で変更した時は同じメッセージのすべてのオカレンスの名前を変更する必要がありますが、この機能によってデータベース全体を検索して一度に置換することができるので、以前のように手作業で 1 つずつ変更する必要はありません。

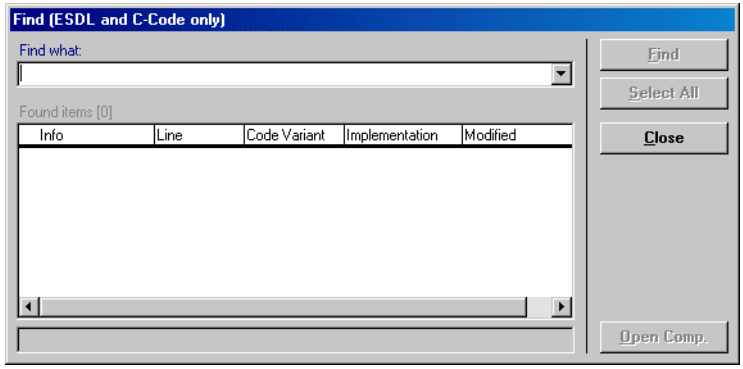
文字列を検索する：

- コンポーネントマネージャで **Edit → Find** を選択します。

または

- **<Ctrl> + <F>** を押します。

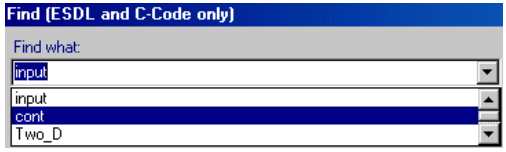
  “Find (ESDL and C-Code only)” ダイアログボックスが開きます。



この時点では、アクティブなボタンは **Close** のみです。

- “Find what”フィールドに検索したい文字列を入力します。

  前に検索したことのある文字列を再度検索するには、同フィールドのコンボボックスからその文字列を選択できます。



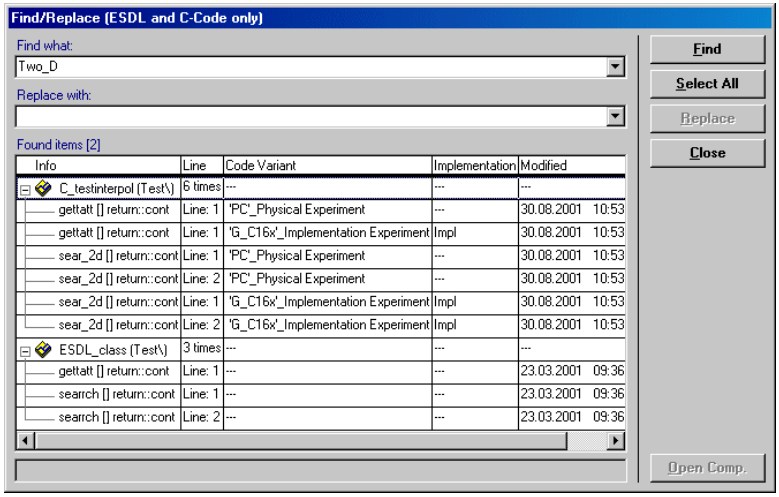
  Find ボタンがアクティブになります。

- **Find** ボタンをクリックすると検索が行われます。

  データベースを最初に検索する時は、検索に数分かかる場合があります。

この検索においては、大文字と小文字は区別されません。たとえば、`cont` と入力して検索を行うと、`Cont` や `CONT` といった文字列も検索されてリストアップされます。また長い文字列の一部分に検索文字列が含まれていた場合も、その文字列もリストアップされます（たとえば `cont` で検索すると `Continuous` もリストアップされます）。

文字列が見つかると、結果が“Found Items”リストに表示され、**Select All** ボタンがアクティブになります。“Find what”フィールドに入力された文字列を含むコンポーネントの数が、リストタイトル“Found Items”の右側に表示されます。



“Found Items”リストには、以下の 2 つのカラムが表示されます。

- “Infos”カラムには、指定の文字列を含むすべてのコンポーネントと個々のメソッド／プロセスがリストアップされます。
- “Line”カラムには、コンポーネントの場合は見つかった文字列の数が、またコンポーネント内のメソッド／プロセスについては文字列が含まれる行の番号が表示されます。
- “Code Variant”カラムには、C コードコンポーネントのコードがどのターゲットおよび実験用に記述されているかが表示されます。
- “Implementation”カラムには、使用されているインプリメンテーションが表示されます。
- “Modified”カラムには、そのコンポーネントが最後に変更された時の日付と時間が表示されます。

検索結果を表示する：

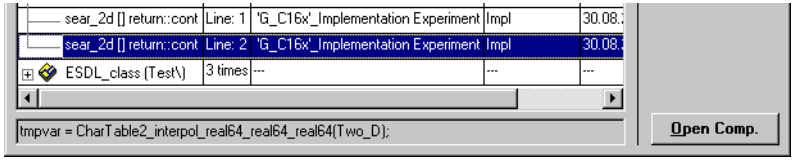
- “Found items”リストに表示されているコンポーネントの左側の“+”シンボルをクリックします。

または

- コンポーネントをダブルクリックします。

  メソッド／コンポーネントのリストが表示されます。
- 表示されたいずれかの行をクリックします。

  ダイアログボックス下部のステータスバーに、実際のコードが表示されます。



以下のようにして、検索ウィンドウ内から直接コンポーネントを開くことができます。

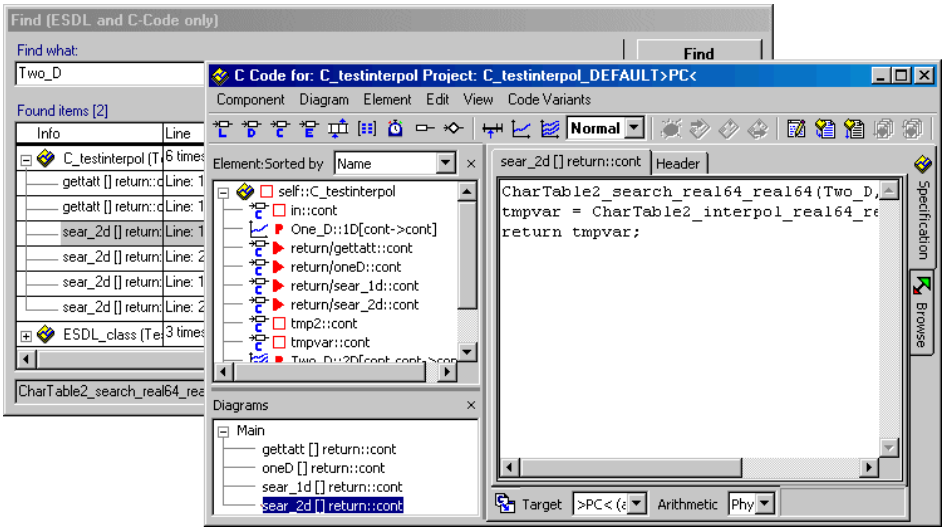
検索ダイアログボックスからコンポーネントを開く：

- **Open Comp.** ボタンをクリックします。

  コンポーネントタイプに応じたエディタが開いてメソッド／プロセスが表示され、検索文字列を含む行が強調表示されます。

  C コードコンポーネントの場合は、“Code Variant”および“Implementation”カラムに示されたバリエーションのコードが表示され、このバ

リエーションはエディタ内のコンボボックスの内容でも確認できます（4.3 章「C コードエディタ」を参照してください）。



任意の文字列を置換する：

注記

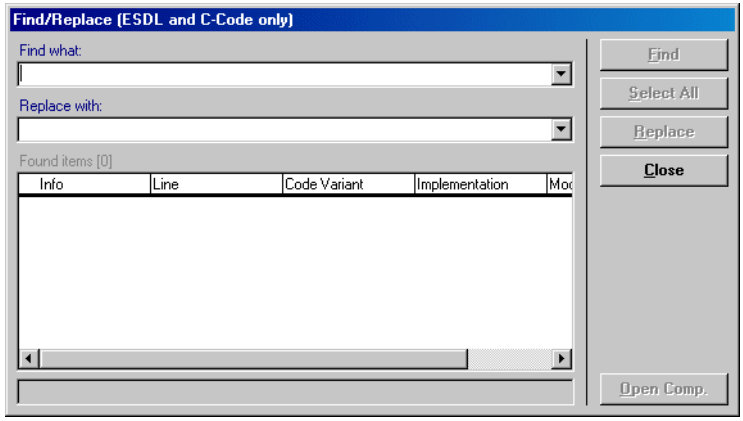
置換操作には“Undo”機能がサポートされていません。充分に注意して行ってください。

- コンポーネントマネージャメニューの **Edit → Replace** を選択します。

または

- **<Ctrl> + <H>** を押します。

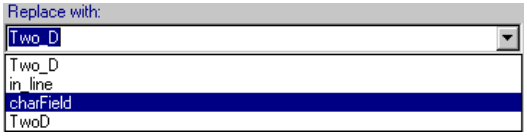
- “Find/Repalce (ESDL and C-Code only)”ダイアログボックスが開きます。



このダイアログは、“Replace with”フィールドと **Replace** ボタンがある以外は検索ダイアログ（113 ページ参照）と同じです。

- 検索して置換したい文字列を“Find what”フィールドに入力します。
- 置換後の新しい文字列を“Replace with”フィールドに入力します。

  前に置換したことのある文字列で再度置換するには、同フィールドのコンボボックスからその文字列を選択できます。



- **Find** をクリックして置換対象の文字列を検索します。

注記

上記の 2 つのステップの順番は、逆でもかまいません。

ここで表示されたコンポーネントは、115 ページの方法で開くことができます。

Replace ボタンはこの時点ではまだアクティブになりません。

- 検索結果のリストの中から置換したい文字列を含むコンポーネントを探し、展開してメソッド／プロセスリストを表示します。
- 置換したいすべての対象を選択（強調表示）します。

  これで **Replace** ボタンがアクティブになります。
- Replace をクリックします。

  選択された個所の文字列が新しいテキストに置換されます。置換された個所は、“Found Items”リストから削除されます。

選択されたコンポーネント内の同じ文字列をすべて置換するには、以下のように操作します。

コンポーネント内の同じ文字列をすべて置換する：

- 置換したい文字列を 113 ページの方法で検索します。
- “Find/Repalce (ESDL and C-Code only)”ダイアログボックスの“Replace with”フィールドに置換後の新しい文字列を入力します。
- 検索結果のリストの中から、置換したい文字列を含むコンポーネントを選択（強調表示）します。

  これによって、このコンポーネントに含まれるすべての文字列が選択されたことになります。
- Replace をクリックします。

  選択されたコンポーネント内の文字列（C コードの場合はすべてのインプリメンテーションとターゲットのバリエーションを含む）が新しいテキストに置換されます。

データベース内の同じ文字列をすべて置換するには、以下のように操作します。

データベース内の同じ文字列をすべて置換する：

- 置換したい文字列を 113 ページの方法で検索します。
- “Find/Repalce (ESDL and C-Code only)”ダイアログボックスの“Replace with”フィールドに置換後の新しい文字列を入力します。
- **Select All** をクリックしてすべてのコンポーネントに含まれる文字列を選択します。
- Replace をクリックすると、データベース全体に含まれるすべての文字列が置換されます。

2.3.6 データベースの扱い

新しいプロジェクトを作成する際には、通常、データベースを新たに作成することから始めます。これは、PC での一般的な作業に例えれば、まず最初にハードディスク上に新しいディレクトリを作成し、そこに必要なサブディレクトリを作成して階層構造を構築して既存のファイルをコピーしてきたり新しいファイルを作成するのと同じ要領です。

注記

現在開いているデータベースの名前は、コンポーネントマネージャ最下部のステータスバーに表示されます。

一般的な作業手順

新しいデータベースを作成する：

注記

ASCET のパフォーマンスを最適な状態に維持するためには、1 つのデータベースのサイズが大きくなり過ぎないように、大量のデータは、意味のある単位で複数のデータベースに分割しておくようにしてください。

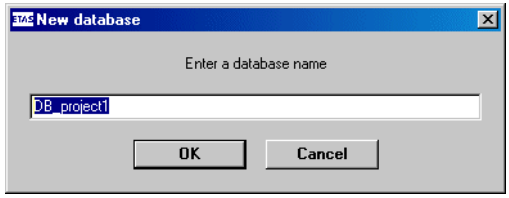
- **File** → **New Database** を選択します。

または

- **New** ボタンをクリックします。

または

- **<Ctrl> + <N>** キーを押します。
  “New database”ダイアログボックスが開きます。



- データベース名を入力します。
- **OK** をクリックします。
  新しいデータベースが開きます。このデータベースは、データベース名と、DEFAULT という名前のデフォルトフォルダのみが含まれる空のデータベースです。
  これと同時に、**Database Path** オプションに設定されているディレクトリ（42 ページ参照）に、データベースと同名のサブディレクトリが作成されます。
- 75 ページの「データベースアイテムの管理」に説明されている方法で、各作業を進めます。

データベースを開く：

注記

旧バージョンのデータベースを開くこともできます。詳しくは 127 ページの「ASCET-SD V4.x のデータベースを変換する：」を参照してください。

- **File → Open Database** を選択します。

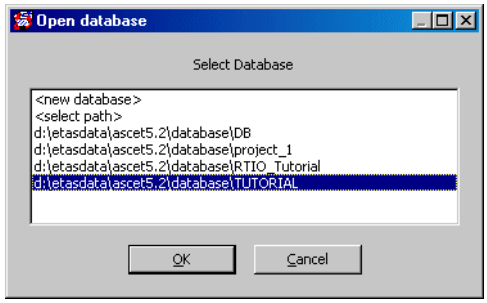
または

- **Open** ボタンをクリックします。

または

- **<Ctrl> + <O>** キーを押します。

  “Open database”ダイアログボックスが開きます。



  データベースのデフォルトディレクトリに保存されているすべてのデータベースが一覧表示されます。

- データベースを選択して **OK** をクリックします。
- **OK** をクリックします。

  デフォルトディレクトリ以外の場所に格納されているデータベースを使用するには、`<select path>` というエントリを選択すると、パスを任意に指定してデータベースを開くことができます。

- **OK** をクリックして確定します。

  データベースがロードされます。

注記

このダイアログボックスから新しいデータベースを作成することもできます。この場合は `<new database>` を選択してください。

データベースを保存する：

- **File → Save Database** を選択します。

または

- **Save** ボタンをクリックします。

または

- **<Ctrl> + <S>** キーを押します。

  データベースの変更内容がすべてハードディスクに保存されます。作業中は定期的にこのコマンドを実行するようにしてください。

注記

ユーザーオプション（**Tools → Options**、“Options”ノード）を使用して、任意の間隔でデータベースが自動保存されるようにすることができます。また、コンポーネントマネージャを閉じる時、つまり ASCET のセッションを終了する時にもデータベースは自動的に保存されます。

データベースを別名で保存する：

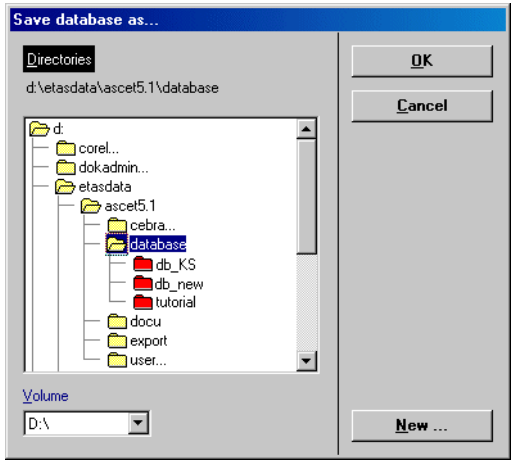
現在開いているデータベースを別名で保存することにより、データベース全体のバックアップを作成することが可能です。この際、現在のデータベースを構成するすべてのファイルがバックアップされます。

- コンポーネントマネージャで、**File → Save Database As** を選択します。

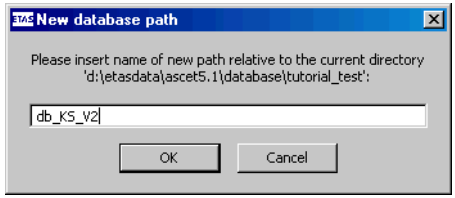
  パス選択ダイアログボックスが開きます。

- データベースの保存先とするディレクトリを選択します。

  この際、既存のデータベースのディレクトリ（赤いフォルダシンボル）を選択しないように注意してください。

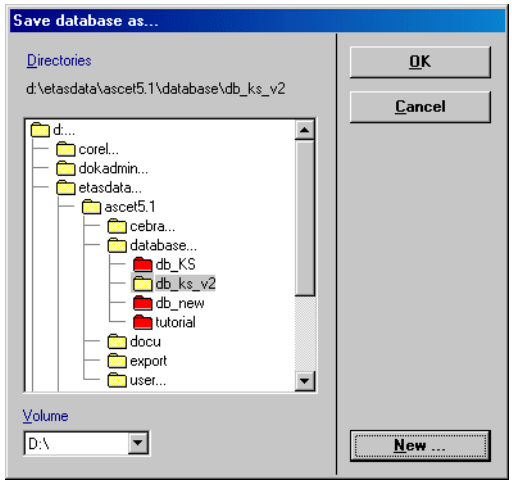


- **New** ボタンをクリックして新しい空のディレクトリを作成します。



- **OK** をクリックします。

  新しいディレクトリが作成され、パス選択ダイアログボックスのフォルダツリー内に、ターゲットディレクトリとして選択された状態で表示されます。フォルダシンボルの色は、ファイルのコピーが終了した後に自動的に赤に変わります。



- パス選択ダイアログボックスの **OK** ボタンをクリックします。

注記

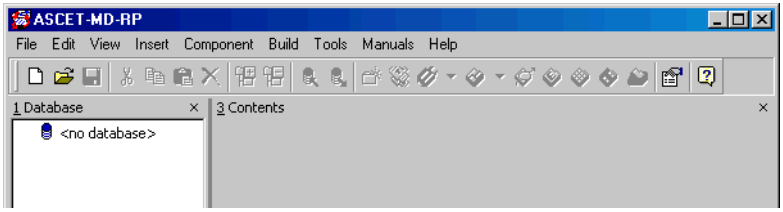
ここでもし新しいディレクトリを作成せずに既存のディレクトリが選択されていた場合、この操作を続行すると既存のディレクトリの現在の内容がすべて失われてしまう旨を示す警告メッセージが表示されるので、この時点で操作を中止することができます。

現在開いているデータベースの内容がすべてコピーされた新しいデータベースがコンポーネントマネージャにロードされます。これは、ウィンドウ最下部のステータスバーでも確認できます。

データベースを閉じる：

- **File → Close Database** を選択します。

  データベースが閉じます。コンポーネントマネージャは開いたままですが、ほとんどのボタンは無効になります。



データベースの削除は、ASCET 上からは行えません。

データベースを削除する：

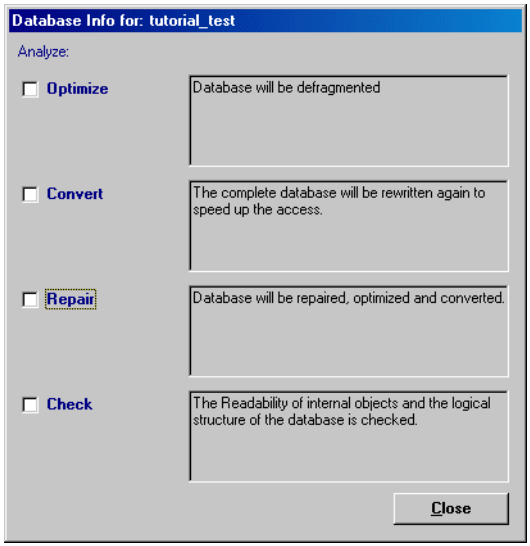
- 削除したいデータベースが ASCET で開いている場合、そのデータベースを閉じてください。

  ASCET で開いているデータベースを削除しようとすると、エラーメッセージが表示されます。
- Windows エクスプローラで、削除したいデータベースが格納されているディレクトリ（例：`..¥ETASData¥Ascet5.2`）を選択します。
- Windows エクスプローラで、そのデータベースが含まれるサブディレクトリを削除します。

データベースを最適化する：

- コンポーネントマネージャで **Tools** → **Database** → **Performace Utilities** を選択します。

  “Database Info”ダイアログボックスが開きます。



- **Optimize** オプションをオンにします。

  このコマンドは、頻繁な編集によってデータベース内に生じた断片の結合（デフラグメンテーション）を行います。

- **Convert** オプションをオンにします。

  このコマンドは、データベース構成の変更によって不要になったデータレコード間の内部参照情報をクリーンアップして、アクセス速度を向上させます。

Optimize コマンドを実行しておけば、通常のパフォーマンスは保たれます。パフォーマンスが著しく低下して作業に重大な影響を与えるようになった場合にのみ **Convert** オプションを使用してください。

注記

大規模なデータベースの変換には時間が非常に長くかかる可能性があるので、ASCET を使用する必要のない夜間等に実行させておくとよいでしょう。

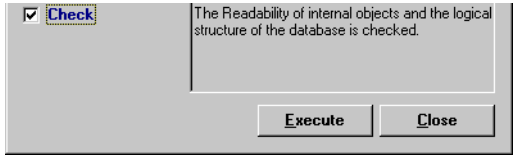
- **Repair** オプションをオンにします。

  このコマンドは、データベースをすべて再構成し、破損したデータ（参照情報など）の修復を行います。このコマンドは“Optimize”と“Convert”の機能を使用するので、これら 2 つのオプションは任意に選択することができません。

- **Check** オプションをオンにします。

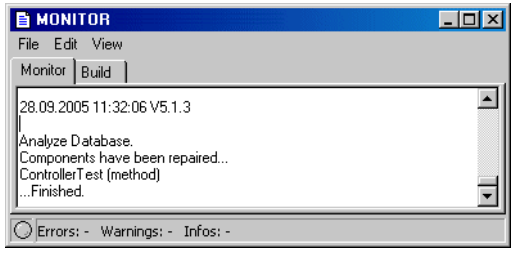
  このコマンドは、データベース内の構成と参照関係をチェックし、その結果をモニタウィンドウに出力します。

  4 つのオプションのうちいずれかをオンにすると、**Execute** ボタンが有効になります。



- **Execute** ボタンをクリックします。

  選択されたデータベース管理プログラムが起動します。結果は ASCET モニタウィンドウに出力されます。



ASCET では、2 つのデータベースの内容全体を比較することができます。この機能は、1 つのデータベースの複数のコピーを使用したり、開発ストリームごとに異なるデータベースを用いたり、また開発チームで作業する場合などに便利です。

データベースを比較する：

- コンポーネントマネージャから **Tools** → **Database** → **Compare Database** を選択します。

  “Compare database”ダイアログボックスが開きます。

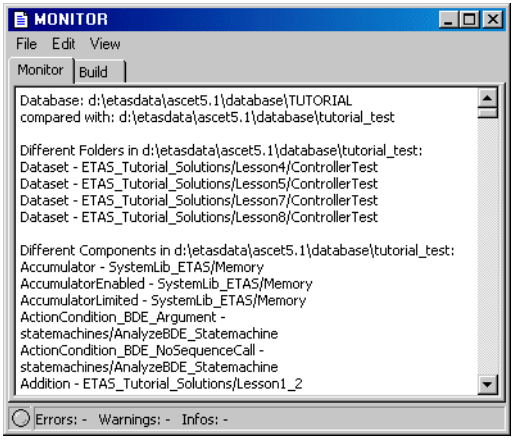
- 現在開いているデータベースと比較したいデータベースを選択します。

- **OK** をクリックして、比較を開始します。

注記

大規模なデータベースの比較には、非常に長い時間がかかる場合があります。

2 つのデータベースが比較されます。両者の間で差異の見られたアイテムやプロジェクトの一覧が、モニタウィンドウに表示されます。



旧バージョンの ASCET データベースの使用方法

旧バージョンの ASCET-SD で作成されたデータベースを ASCET 5.2 で使用することは可能ですが、ASCET 5.2 のデータベースフォーマットには以前のバージョンで作成されたデータベースフォーマットとの互換性がないため、そのまま使用することはできません。

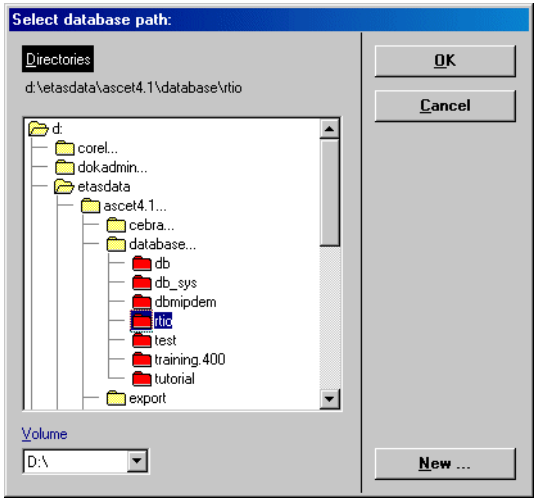
ASCET-SD V4.x で作成されたデータベースを最初に開くと、自動的に現行のバージョンに変換されます。

ASCET-SD V4.x のデータベースを変換する：

- 120 ページの「データベースを開く：」に説明されている通常の方法でデータベースを開きます。

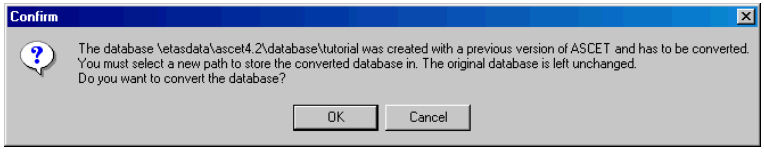
- “Open Database”ダイアログボックスで <select path> というエントリを選択して **OK** をクリックします。

  “Select database path”ダイアログボックスが開きます。



- 旧バージョンのデータベースが格納されたディレクトリを選択します。
- **OK** をクリックして確定します。

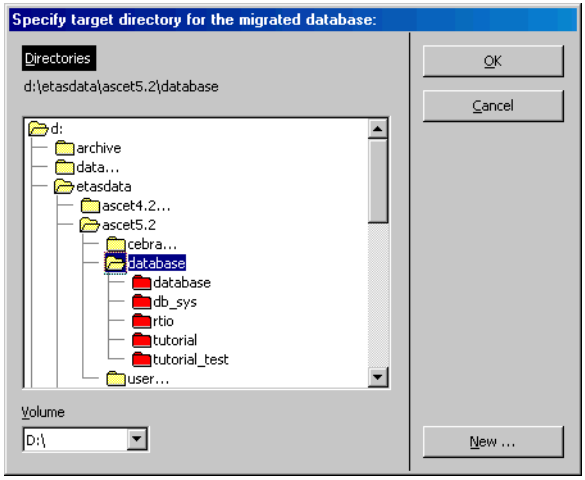
  データベースの変換を実行することについての確認メッセージが表示されます。



  ここで **Cancel** をクリックすると変換は中止されます。

- **OK** をクリックすると変換作業が続行されます。

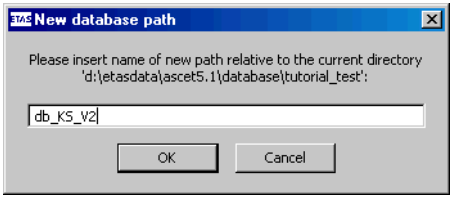
  “Specify target directory for the migrated database”ダイアログボックスが開きます。これは“Select database path”ダイアログボックスと同じ内容です。



注記

ここで、空でないディレクトリを選択すると、ワーニングが発生します。その際は、ワーニングを確認し、新しい空のディレクトリを作成してください。

- 新しい空のディレクトリを作成するには、**New** ボタンをクリックします。



- ディレクトリ名を入力し、**OK** で確定します。

  新しいディレクトリが作成され、変換後のデータベースが保存されるターゲットディレクトリとなります。

- パス選択ダイアログボックスで **OK** をクリックして変換処理を開始します。

  データベースが ASCET 5.2 フォーマットに変換され、新しいディレクトリに保存されます。変換に要する時間はデータベース内のデータ量に依存します。

  古いデータベースはそのまま保持されます。

ASCET 5.2 のデータベースには ASCET V4.0 より古いバージョンのデータベースフォーマットとの互換性がないため、これらのバージョンで作成されたデータベースを直接 ASCET 5.2 で開くことはできません。このような古いデータベースを 127 ページの方法で開こうとすると、以下のようなエラーメッセージが表示されます。

```
This file is compatible with ASCET-SD V3.0 or earlier. It
cannot be imported directly. Convert it to a database
using your old version of ASCET-SD, then open this
database.
```

現在 ASCET-SD V4.x をお持ちの場合、以下のような方法で、V4.0 より古いバージョンのデータベースを使用することができます。

V4.0 より古いバージョンの ASCET-SD からのインポートを行う：

- ASCET-SD V4.x を起動します。

  これらのバージョンは、それより前のバージョンのデータベースフォーマットをサポートしています。
- 古いバージョンのエクスポートファイルをインポートします。

  インポートされたデータベースアイテムは、現行のフォーマットに変換されます。
- インポートしたアイテムをエクスポートします。
- ASCET V4.x を閉じます。
- ASCET V5.2 を起動します。
- V4.xでエクスポートしたファイルを96ページの方法でインポートします。

ANSI C への変換

ASCET のバージョン 2.x 以降のデータベースアイテム名は、すべて ANSI C に準拠している必要があります。新しいアイテムを追加するときや既存のアイテム名を変更するときには、必ず ANSI C に準拠した名前を使用してください。

それよりも前のバージョンのデータベースを変換した後には、既存のフォルダ名やアイテム名の変換を明示的に行う必要があります。フォルダ名とアイテム名はすべて機械的に変換され、特殊文字は表 2-4 に従って置換されます。

特殊文字	変換後の文字
Ä, ä, Ö, ö, Ü, ü, ß	Ae, ae, Oe, oe, Ue, ue, ss
<space>, <underscore>, <dash>, <dot>	すべて <underscore>

表 2-4　ANSI-C への対応

データベースを ANSI-C に変換する：

- 名前の変換を行いたいデータベースを開きます。
- **Tools → Database → Convert → All Names to ANSI C** を選択します。

  すべてのフォルダ、アイテム、メソッド、メッセージ、変数などの名前について、すべての特殊文字やスペースが表 7-1 に従って ANSI-C 準拠の文字に変換されます。

注記

大規模なデータベースを変換する際には、変換後にデータベースを最適化してデータベースアクセスを高速化することをお勧めします。詳しくは 125 ページの「データベースを最適化する：」という項を参照してください。

名前の衝突を解決する

既存のバージョン 2.x データベース内には、句読点やスペース以外はまったく同じ名前のアイテムが複数存在する場合があります。このような場合、デフォルト条件で ANSI-C に変換すると名前の衝突が起こる可能性があります。

たとえば、“Integrator I21”および“Integrator-I21”というアイテム名をデフォルト変換すると、どちらも“Integrator_I21”になります。アイテム名はユニークでなければならないので、この場合は、変換アルゴリズムにより 2 個目のアイテムは“Integrator_211”という名前になります。

名前の衝突を避けるために、ANSI-C への変換時の上記のルールをそのまま使用するか、あるいはマップファイルを使用することもできます。ASCET ディレクトリに `mapping.ini` というファイルがあるので、これを任意のテキストエディタで編集します。

注記

スペースについての変換規則を変更することはできません。スペースは必ずアンダースコアに変換されます。

データベースのメンテナンス

ASCET には、これまでに説明されたデータベースの最適化や修復、またはバックアップや比較といった各種ツールのほかに、データベース内のブラウズ、生成されたコードの廃棄、データベース全体の build 処理の強制実行、といった機能も用意されています。以降に、コンポーネントマネージャから利用できる、これらのデータベースユーティリティについて説明します。

旧バージョンのデータベースの変換については、127 ページの「ASCET-SD V4.x のデータベースを変換する：」を参照してください。

生成されたコードの削除

実験中に生成されたすべてのコードをデータベースから削除することにより、データベースのサイズが小さくなり、コードをすべて再生成することができます。

データベース内に保存された生成済みコードを削除する：

- **Build → Clean All** を選択します。

  確認メッセージが表示されます。
- OK で確定します。

  自動生成されてデータベース内に格納されているコードがすべて削除されます。

コード生成の際は Make 処理が実行され、通常は前回の Make 処理以降に変更されたアイテムだけについてコードが生成されコンパイルが行われますが、場合によっては完全な Build 処理、つまり全アイテムのコード生成とコンパイルを行うことが必要になる場合があります。この場合、すでに生成されているコードは削除されません。

コード生成時に完全な Build 処理が行われるようにする：

- **Build → Touch All** を選択します。

  データベース内のすべてのコンポーネントに「変更済み」マークが付き、続いて Build 処理を行うと、強制的にすべてのコードが新しく生成され、コンパイルされます。

注記

大規模なデータベースについて完全な Build 処理を強制実行すると、非常に長い時間を要する場合があります。

データベースのブラウズ

データベースをブラウズすることにより、アイテム間の参照関係や演算子インプリメンテーションなどについての情報を一覧表示することができ、たとえば、他のアイテムから参照されているすべてのアイテムを把握することも可能です。データベースのブラウザを行う際は、さまざまな検索基準を組み合わせて条件に合ったデータベースアイテムを表示し、これらの関係を確認できます。

ASCET のデータベースをブラウズする際、現在選択されているフォルダやアイテムとは無関係に、必ずデータベース全体を対象として検索が行われます。検索基準にはワイルドカードを使用することができ、たとえば検索基準を *acc* と定義すると、名前の中に文字列 acc が含まれているすべてのアイテムが検索されます。検索の際、大文字と小文字は区別されません。

演算子インプリメンテーションの検索（**Tools → Database → List Operator Implementations**）については、493 ページの「演算子のインプリメンテーションを検索する：」を参照してください。

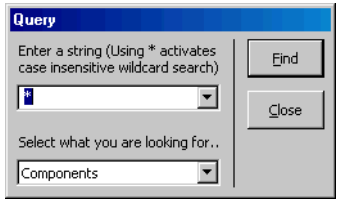
データベースをブラウズする（”Query”ダイアログボックスを使用）：

- コンポーネントマネージャで、ブラウズを行いたいデータベースを開きます。
- **Edit → Query** を選択します。

または

- **<Ctrl> + <Q>** キーを押します。

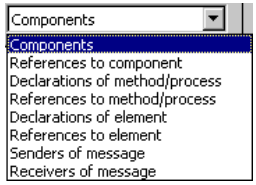
“Query”ダイアログボックスが開きます。



- “Enter a string”フィールドに検索基準とする文字列を入力します。

  検索文字列として使用できるのは、データベース内のアイテム（メソッド、プロセス、またはエレメント）の名前のみです。

  コンボボックスから以前に検索したことのある文字列を選択することもできます。
- “Select what you are looking for”コンボボックスで、検索する情報の種類を選択します。



- **Find** をクリックして検索を開始します。

  見つかった情報がダイアログボックスに表示されます。

コンポーネントマネージャのツールバーでもブラウズを行えます。

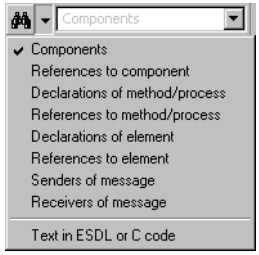
データベースをブラウズする（ツールバーを使用）：

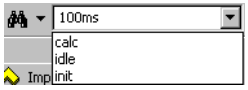
- コンポーネントマネージャで、**Search - *<information type>*** ボタンの隣の矢印ボタンをクリックします。

  検索対象（135 ページを参照してください）として選択できる「情報タイプ」のリストが開きます。現在選択されている情報タイプにチェックマークが付いています。

- 情報タイプを選択します。

  コンボボックス内に、選択された対象が薄い文字で表示されます。





- コンボボックスに検索したいアイテム名を入力します。

または

- 検索履歴からアイテムを選択します。

- **Search - *<information type>*** をクリックして検索を開始します。

  検索が開始されると、“Search *<objects>*”ダイアログボックスが開きます。このダイアログボックス内の **Cancel** ボタンで検索を中断できます。

  検索が終了すると、結果がダイアログボックスに表示されます。

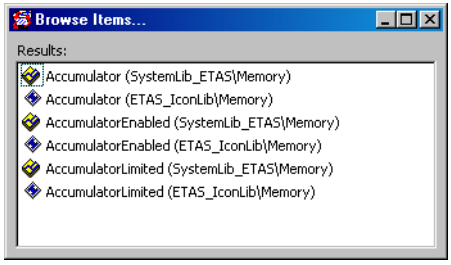
注記

情報タイプとして `Text in ESDL` または `C code` が選択されていると、**Search Text in ESDL** または **C** を実行しても、その結果を表示するダイアログボックスは開きません。その代わりに“Find (ESDL and C-Code only)”ダイアログボックスが開きます。詳しくは 113 ページの「文字列を検索する：」を参照してください。

検索対象として選択できる情報は、以下のとおりです。

Components:

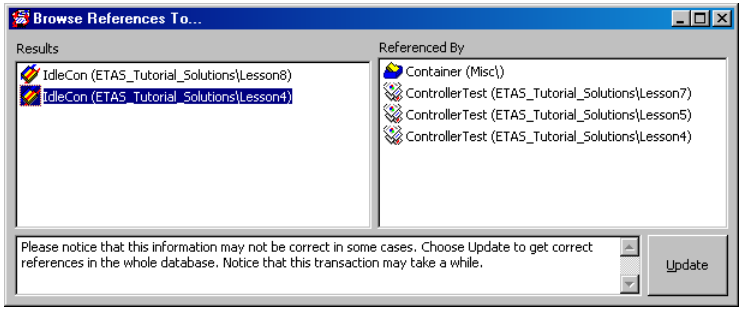
データベース内から、検索文字列と一致する名前のデータベースアイテムが検索され、結果が以下のようなダイアログボックスに表示されます。



“Browse Items”ダイアログボックスには、検索条件に合ったアイテムのフルネームが、そのデータベースパスと共に表示されます。任意のコンポーネント名をクリックすると、そのコンポーネントがコンポーネントマネージャに表示され、強調表示されます。

References to component:

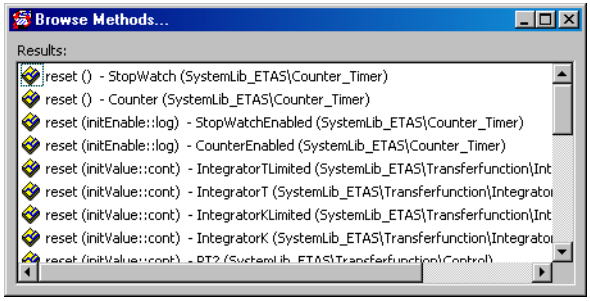
データベース内から、検索文字列と一致する名前のデータベースアイテムへの参照が検索され、結果が以下のようなダイアログボックスに表示されます。



上の図に示されるように、検索条件と一致したアイテムはダイアログボックスの“Results”ペインに表示され、ここに表示されているアイテムをクリックすると、そのアイテムを参照しているすべてのアイテムが“Referenced By”ペインに表示されます。どちらのペインのアイテムをクリックしても、クリックされたコンポーネントがコンポーネントマネージャに表示されて強調表示されます。

Declaration of method/process:

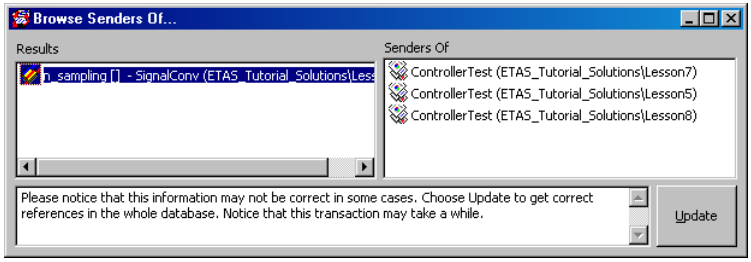
データベース内から、検索文字列と一致する名前のメソッドまたはプロセスが検索され、結果が以下のようなダイアログボックスに表示されます。



この“Browse Methods”ダイアログボックスには、展開されたメソッドセレクタと、それが定義されているコンポーネントのデータベースパスが表示されます。任意のメソッドをクリックすると、それが定義されたコンポーネントがコンポーネントマネージャに表示され、強調表示されます。

References to method/process:

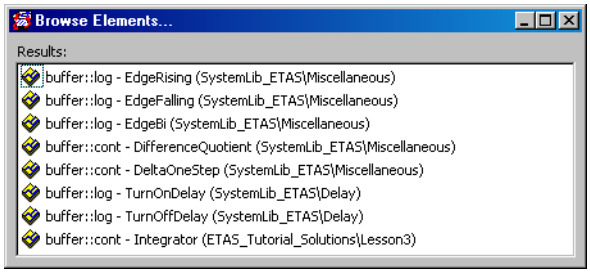
データベース内から、検索文字列と一致する名前のメソッドまたはプロセスへの参照が検索され、結果が以下のようなダイアログボックスに表示されます。



この“Browse Senders Of”ダイアログボックスには、検索文字列と一致するアイテムが“Results”ペインに表示されます。ここには、メソッド／プロセスセレクタがそれを定義しているコンポーネントと共に一覧表示され、任意のメソッド／プロセスをクリックすると、そのメソッド／プロセスを呼び出しているコンポーネントの一覧が、ダイアログボックス右側の“Senders Of”ペインに表示されます。どちらのペインのアイテムをクリックしても、それに関係するコンポーネントがコンポーネントマネージャに表示され、強調表示されます。

Declaration of element:

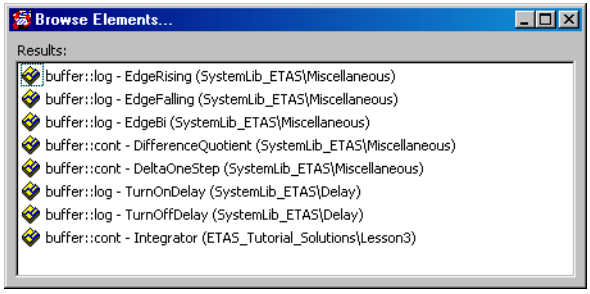
検索文字列と一致するエレメント（変数、引数など）が宣言されているコンポーネントが検索され、結果が以下のようなダイアログボックスに表示されます。



“Browse Elements”ダイアログボックスには、検索されたエレメントが、それを宣言しているコンポーネントとそのデータベースパスと共に表示されます。任意のエレメントをクリックすると、それに関係するコンポーネントがコンポーネントマネージャに表示され、強調表示されます。

References to element:

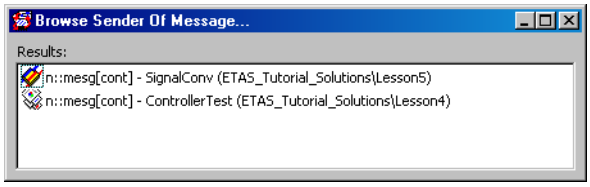
検索文字列と一致するすべてのエレメント（例、変数または引数）への参照を含むコンポーネントが検索され、結果が以下のようなダイアログボックスに表示されます。



この“Browse Elements”ダイアログボックスには、検索されたエレメントが、それを参照しているコンポーネントとそのデータベースパスと共に表示されます。任意のエレメントをクリックすると、そのコンポーネントがコンポーネントマネージャに表示され、強調表示されます。

Sender of message:

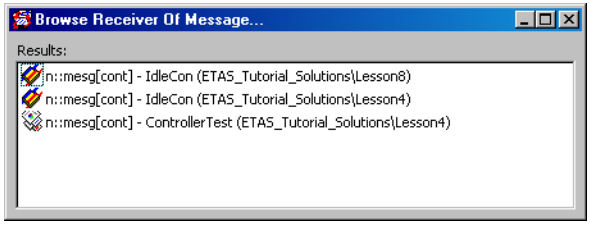
検索文字列と一致する送信メッセージを送信するモジュールまたはプロジェクトが検索され、結果が以下のようなダイアログボックスに表示されます。



各メッセージは、それを定義しているモジュール／プロジェクト名と共に表示されます。任意のメッセージをクリックすると、そのモジュール／プロジェクトがコンポーネントマネージャに表示され、強調表示されます。

Receiver of Message:

Sender of message コマンドと同様の検索が、受信メッセージについて行われます。



2.3.7 データベースアクセス

個々のデータベースアイテムまたはフォルダ全体についてアクセス権を指定することができ、これによって重要なアイテムを不用意に削除したり書き換えてしまったりすることを防ぐことができます。またアクセス権の設定をパスワードで保護することにより、パスワードを知っているユーザーだけがアクセス権を変更できるようになり、ASCET のデータ交換時の機密保護が確保されます。

たとえば、あるコンポーネントのアクセス権を“execute only”（実行のみ）に設定しておくと、そのコンポーネントを 1 つの機能単位として他のコンポーネント内で使用することはできますが、そのコンポーネントの内容、つまりコンポーネントのアルゴリズムは見ることはできません。

各アイテムには以下のアクセス権を設定することができます。

Read:

このアクセス権が有効になっていると、データベースアイテムは所定のエディタで内容を見ることができます。

Write:

このアクセス権が有効になっているアイテムは、エディタで内容を編集したり、データベースから削除することができます。

Calibration:

このアクセス権が有効になっているアイテムは、実験で値を適合することが可能です。

Execute:

このアクセス権が有効になっているアイテムは、実験を行うことができます。これは被参照アイテムの場合も同様で、あるアイテムについて実行権を持たないユーザーは、そのアイテムを別のアイテム内で使用したり実験を行うことができません。

Code Generation:

このアクセス権が有効になっているアイテムは、書き込み権が有効になっていなくてもコードを生成することができます。

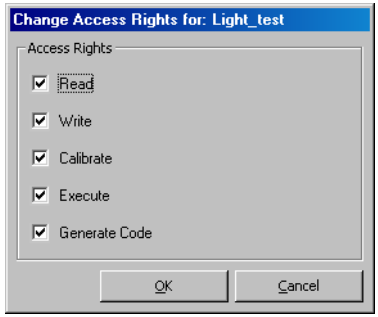
これらのアクセス権は、最新の設定内容が常に優先して適用されます。たとえば、1 つのコンポーネントのアクセス権を変更した後に、そのコンポーネントが含まれるフォルダ全体のアクセス権を変更すると、フォルダ全体についての設定内容が有効になり、先に行われたコンポーネントについての変更内容は無効になります。

選択したアイテムに対するユーザーのアクセス権は、コンポーネントマネージャのステータスバーに表示されます。

フォルダまたはアイテムに対するすべてのアクセス権を上書きして変更する：

- コンポーネントマネージャの“1 Database”リストから、アクセス権を変更したいフォルダまたはアイテムを選択します。
- **Component → Access Rights** を選択します。

  “Overwrite Access Rights for”ダイアログボックスが開きます。そこには、選択されたアイテムの現在のアクセス権が表示されます。.



- アクセス権を選択します。

  チェックマークのついたアクセス権が有効となります。
- **OK** をクリックして、変更を確定します。

どのデータベースアイテムについてもアクセス権を設定することができます。ただし、パスワード保護が有効になっている場合は行える設定範囲は限定されます。アクセス権変更のパスワード保護は、データベース内の各ルートフォルダごとに個別に設定できます。パスワード保護が有効になっている場合、アクセス権の変更はパスワードを入力しないと行えません。

注記

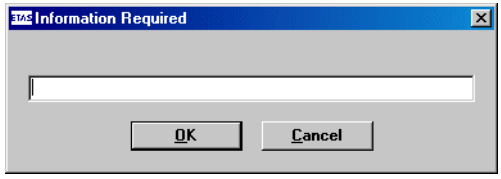
パスワード保護が有効になっている場合、アクセス権を変更しようとするたびにパスワードを入力する必要があります。これでは、多くのアイテムのアクセス権を修正するような場合には非常に面倒なため、先にアクセス権をすべて設定してからパスワード保護を有効にする方が効率的です。

パスワード保護を有効にする：

- コンポーネントマネージャの“1 Database”リストで、最上位レベルのフォルダを選択します。
- **Component → Password** を選択します。

  パスワード保護を有効にすることについての確認のダイアログボックスが開きます。
- この“Confirm”ダイアログボックスで **Yes** をクリックして操作を続けます。

  “Information Required”ダイアログボックスが開きます。



- 6 文字以上のパスワードを入力してから、**OK** をクリックします。

  パスワードを確認するダイアログボックスが開きます。

- 同じパスワードをもう 1 度入力してから **OK** をクリックすると、パスワード保護が有効になります。

  以降、このフォルダのアクセス権を修正するにはパスワードの入力が必要になります。

パスワード保護を無効にするには、パスワード入力が必要です。

パスワード保護を無効にする：

- コンポーネントマネージャの“1 Database”リストで、パスワード保護を無効にしたい最上位レベルのフォルダを選択します。
- **Component → Password** を選択します。

  “Information Required”ダイアログボックスが開きます。
- パスワードを入力して、**OK** をクリックします。

  パスワード保護を無効にすることについての確認メッセージが表示されます。
- **Yes** をクリックして確定します。

  以後、パスワード保護が行われなくなります。

2.4 モニタウィンドウ

ASCET モニタウィンドウには 2 つのタブがあり、ASCET によって実行されたさまざまな処理の結果がこれらのタブに表示されます。

“Monitor”タブには以下のような処理の結果が出力されます。

- コード生成、実験（4.1.10 項、4.2.5 項、4.5.4 項、4.8.10 項参照）
- ダイアグラムの分析（256 ページ、327 ページ参照）
- データベースの最適化（125 ページ参照）
- エクスポート／インポート処理（2.3.3 項、2.3.4 項参照）
- 演算子インプリメンテーションの検索（493 ページ参照）

“Build”タブには以下のような処理の結果が出力されます。

- コード生成、実験
- ダイアグラムの検証

メニューコマンドの概要：

- **File**

  — *Open*（**<Ctrl> + <O>**）

  ハードディスクに保存されているログファイルを開きます。このコマンドを実行すると、“Build”タブが表示されていても必ず“Monitor”タブが表示され、それまで“Monitor”タブ上に出力されていた内容は消去されます。

  — *Save*（**<Ctrl> + <S>**）

  タブ上に出力された情報をログファイル（“Monitor”タブの内容は`ASCET_Monitor.log`、“`Build`”タブの内容は`CodeGeneration.log`）に保存します。

  — *Save As*

  タブ上に出力された情報を任意のファイルに保存します。

  — *Show in editor*

  テキストエディタで、現在のタブのログファイルを開きます。ASCETオプションウィンドウの“External Tools”ノード（54 ページ参照）で任意のテキストエディタを選択できます。

  — *Reposition*

  モニタウィンドウを、元のサイズと位置に戻します。

  — *Exit*（**<Alt> + <F4>**）

  ASCET モニタウィンドウを閉じます。

- **Edit**

  以下のメニューコマンドは、“Monitor”タブのショートカットメニューからも実行できます。

  注記

  以下のコマンドは“Monitor”タブでしか使用できません。

  - *Cut*（**<Ctrl> + <X>**）
    選択されているテキストを切り取ってクリップボードに貼り付けます
  - *Copy*（**<Ctrl> + <C>**）
    選択されているテキストをクリップボードにコピーします
  - *Paste*（**<Ctrl> + <V>**）
    タブ上のカーソルポイントにクリップボード上のテキストをコピーします。
    この処理は、**<Ctrl> + <Z>** で取り消すことができます。
  - *Select All*（**<Ctrl> + <A>**）
    タブ上のテキスト全体を選択します。
  - *Find/Replace*（**<Ctrl> + <F>**）
    タブ上のテキストの検索／置換を行います。
  - *Clear*（**<Ctrl> + <R>**）
    タブ上の情報をすべて消去します。

- **View**

  注記

  以下のコマンドは、”Build”タブが表示されている場合にのみ有効です。

  - *Collapse all*
    タブ上の情報が省略表示されます。
  - *Expand all*
    タブ上の情報が展開表示されます。

ショートカットメニューコマンドの概要：

- “Monitor”タブ
  Edit メニューと同じコマンドが含まれます。
- “Build”タブ
  - *Open*
    メッセージが発行されたコンポーネントを、エディタで開きます。

— *File Out*

タブに表示された内容を、任意のファイルに保存します。

— *Hide*

選択されたタイプのメッセージをすべて「非表示」にします。

— *Promote to warning*

「情報」をのステータスレベルを上げて「ワーニング」にプロモートします。

— *Promote to error*

「情報」または「ワーニング」のステータスレベルを上げて「エラー」にプロモートします。

— *Revoke Promotion*

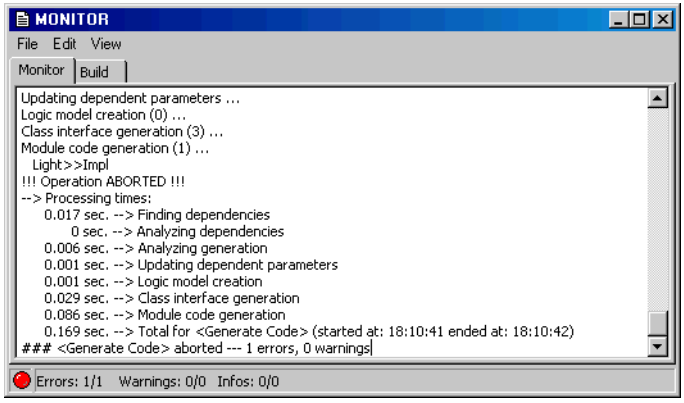
プロモートしていた「情報」または「ワーニング」を、元のレベルに戻します。

— *Settings*

“CodeGen Message Configuration”ダイアログボックスを開きます（153 ページ参照）。

2.4.1 “Monitor”タブ

“Monitor”タブには、さまざまな処理の結果が出力されます。以下の画面の例の場合、ステートマシンのコード生成とコンパイルを行った結果が出力されています。続けて別の処理を行うと、それらの処理結果は現在までに出力されている情報に続けて出力されます。



“Monitor”タブの内容は、以下のようにして保存できます。

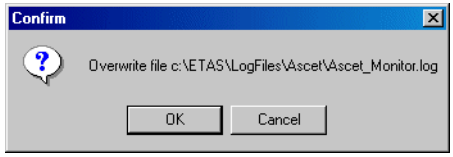
“Monitor”タブの内容をデフォルトファイルに保存する：

- テキストを`ETAS¥LogFiles¥Ascet`ディレクトリの`Ascet_Monitor.log`ファイルに保存するには、**File → Save** を選択します。

または

- **<Ctrl> + <S>** キーを押します。

  同じディレクトリ内に同名のファイルが存在する場合は、ファイルを上書きしてよいかどうかを確認するメッセージが表示されます。



- **OK** をクリックします。

  “Monitor”タブの内容が保存されます。

“Monitor”タブの内容を任意のファイルに保存する：

- ログ情報を任意のファイルに保存するには、**File → Save As** を選択します。

  ファイル選択ダイアログボックスが開きます。

- パスとファイル名を選択して Save をクリックします。

  “Monitor”タブの内容が指定のファイルに保存されます。

モニタウィンドウ内の情報は、一般のテキストエディタのように削除やコピーを行ったり、任意のテキストを追加することもできます。**Edit** メニューには、**Cut** ／ **Copy** ／ **Paste** コマンドや、検索／置換を行うための **Find/Replace** コマンドが含まれています。

注記

ここでの編集作業には”Undo”機能はサポートされていません。

“Monitor”タブ上のテキストの検索／置換を行う：

- **File → Find/Replace** を選択します。

または

- <Ctrl> + <F> キーを押します。
  “Find/Replace”ダイアログボックスが開きます。

- 検索する文字列を“Find”フィールドに入力します。
- 必要に応じて、置き換える文字列を“Replace With”フィールドに入力します。
- 検索する方向を“Direction”フィールドで指定します。
- 大文字と小文字を区別して検索するには、**Case Sensitive** オプションをオンにします。
- タブの最後（または先頭）まで検索した後に続けて先頭（または最後）から検索を行うには、**Wrap Search** オプションをオンにします。
- **Find Next** をクリックすると、次の文字列が検索されます。
- **Replace Selection** をクリックすると、検索された文字列が“Replace With”フィールドに入力された文字列に置き換わります。
- **Replace/Find** をクリックすると、検索された文字列が“Replace With”に入力された文字列に置き換わります。
- **Replace All** をクリックすると、検索条件に合う文字列がすべて置換されます。
- **Close** をクリックすると“Find/Replace”ダイアログボックスが閉じます。

結果を保存したい作業を開始する前などにタブ上の情報をすべて削除しておくと、情報が見やすくなります。

タブ上の情報をすべて消去する：

- **Edit → Clear** を選択します。

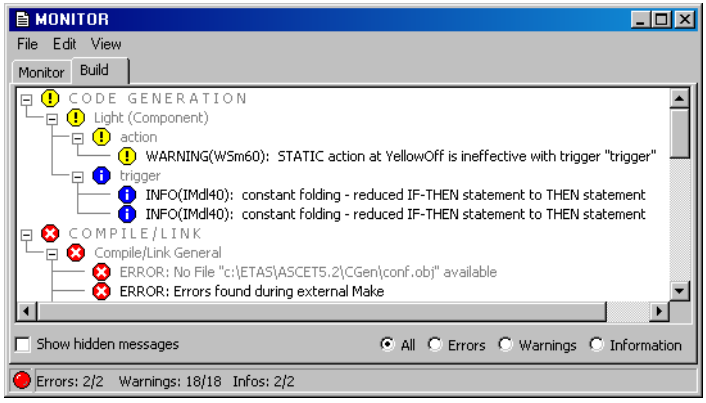
または

- <Ctrl> + <R> キーを押します。

  “Monitor”タブ上に出力された内容がすべて消去されます。

2.4.2 “Build”タブ

“Build”タブには、コード生成やコンパイルの結果やダイアグラムの検証結果が出力されます。以下の画面の例には、前出の“Monitor”タブ上の例と同じコード生成とコンパイルの結果が出力されています。“Monitor”タブの場合とは異なり、前回行われた処理の結果は消去されます。



実行された処理から出力された 3 つのレベルのメッセージ（エラー：赤丸のシンボル、ワーニング：黄色い丸のシンボル、情報：青い丸のシンボル）が表示され、メッセージタイプは略語（`WSm60`、`Idml40` など）で示されます。各処理から出力されたメッセージはコンポーネントやメソッド／プロセスの構造に対応するツリー構造で表示されます。

問題の発生した個所を容易に見つけることができるように、各メッセージはそれぞれの発生個所にリンクしています。

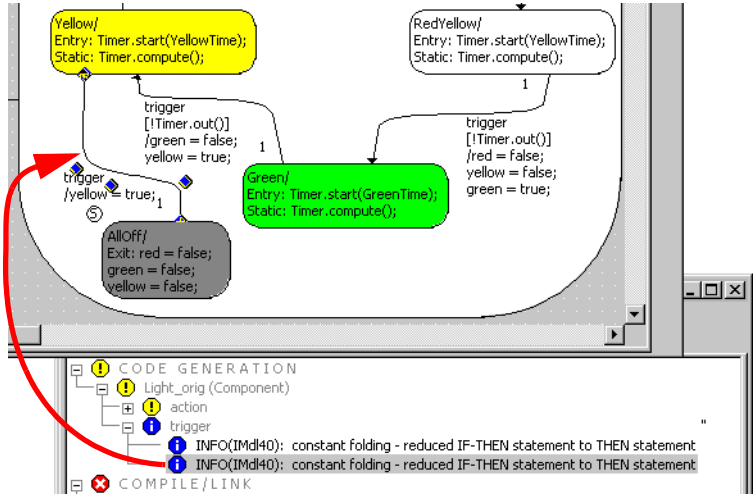
メッセージの発生個所を表示する：

- “Build”タブ上のメッセージをダブルクリックします。

または

- メッセージを右クリックし、ショートカットメニューから **Open** を選択します。

  関連するコンポーネントが所定のエディタで開き、問題が発生した個所が強調表示されます。



上図の例では、ステータス `AllOff` から `Yellow` へのトランジション上でワーニングが発生したことが示されています。

コンパイラエラーやリンカエラーが発生した場合は、別のテキストウィンドウが開いて生成されたファイルが表示されます。使用するテキストエディタは、ASCET オプションウィンドウの“External Tools”ノードで選択できます（54 ページを参照してください）。

“Build”タブの内容は、以下のようにして保存できます。

“Build”タブの内容を保存する：

- ログテキストを `ETAS¥LogFiles¥Ascet` ディレクトリの `CodeGeneration.log` ファイルに保存するには、**File → Save** を選択します。

または

- **<Ctrl> + <S>** キーを押します。

  “Monitor”タブの場合とは異なり、同じディレクトリ内に同名のファイルが存在する場合、そのファイルは警告なしに上書きされます。

  “Build”タブに出力されている情報が、ファイルに書き込まれます。

注記

この際、**非表示**になっているメッセージ（150 ページ参照）は、**保存されません。**

“Build”タブの内容を任意のファイルに保存する：

- ログ情報を任意のファイルに保存するには、ショートカットメニューから **File Out** を選択します。

または

- **File → Save As** を選択します。

または

- タブ上でショートカットメニューを開き、**File Out** コマンドを選択します。

  ファイル選択ダイアログボックスが開きます。

- パスとファイル名を選択して Save をクリックします。

  “Build”タブの内容が指定のファイルに保存されます。

“Build”タブ上に表示されるメッセージのコンフィギュレーション

“Build”タブに表示されるメッセージのコンフィギュレーションは、いくつかの方法で設定できます。また、153 ページの「“CodeGen Message Configuration”ダイアログボックスでのメッセージ設定」にも、メッセージの設定方法が説明されています。設定されたコンフィギュレーションは、現在のプロジェクト（コンポーネントについて作業している場合はデフォルトプロジェクト）のオプションとして保存されます（407 ページを参照してください）。

メッセージの「表示／非表示」の切り替え： ある特定のタイプのメッセージを非表示にしてウィンドウを見やすくしたり、特定のタイプのメッセージのみを表示して詳しく分析することができます。

注記

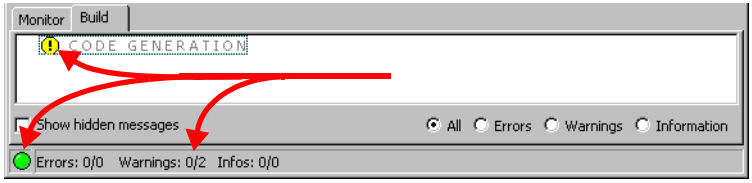
エラーメッセージ（情報またはワーニングから「プロモート」されたものを含みます）を非表示にすることはできません。

メッセージを非表示にする：

- “Build”タブ上で非表示にしたいタイプのメッセージを右クリックします。
- ショートカットメニューから **Hide** を選択します。

  選択されたメッセージと同じタイプのメッセージがすべて表示されなくなり、ウィンドウ最下部のステータスバーに表示されたメッセージ数も変わります。

  メッセージをすべて非表示にすると、ステータスバーの左隅に緑色の丸いシンボルが表示され、非表示になっているメッセージがあることが示されます。また非表示になっているメッセージの数も表示されます。



非表示にしたメッセージは、以下のようにして再度表示することができます。

非表示になっていたメッセージを表示する：

- “Build”タブ上で **Show hidden messages** オプションをオンにします。

  非表示になっていたメッセージがすべて表示されます。

注記

特定のタイプのメッセージのみを「非表示」から「表示」に切り替えることはできません。これが行えるのは“CodeGen Message Configuration”ダイアログボックス（153 ページ参照）のみです。

メッセージをプロモートする（ステータスレベルを上げる）： コード生成処理中に情報またはワーニングメッセージが発行された場合、コード生成処理はそのまま続行されます。しかしプロジェクトによっては、この条件を厳しくすることが必要となる場合もあるため、特定のタイプのメッセージを「プロモート（“promote”）」する、つまりメッセージのステータスレベルを上げることができます。以下に、プロモートできるステータスを示します。

↓右記のレベルに上げる→	情報	ワーニング	エラー
情報		可能	可能
ワーニング	---		可能
エラーメッセージ	---	---	

「非表示」になっているメッセージをプロモートすると、そのメッセージは、再度表示されるようになります。

情報とワーニングをプロモートする：

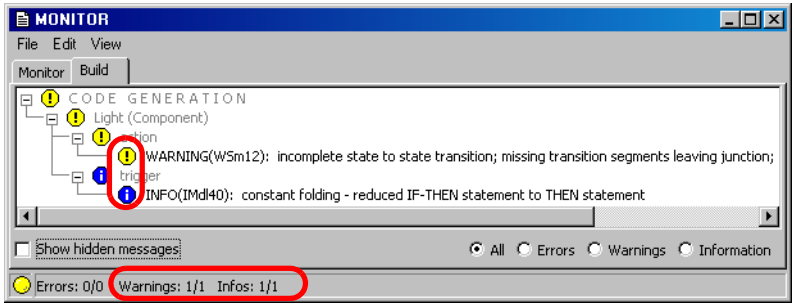
- “Build”タブで、プロモートしたいタイプのメッセージを右クリックします。
- **Promote to warning** を選択します。

または

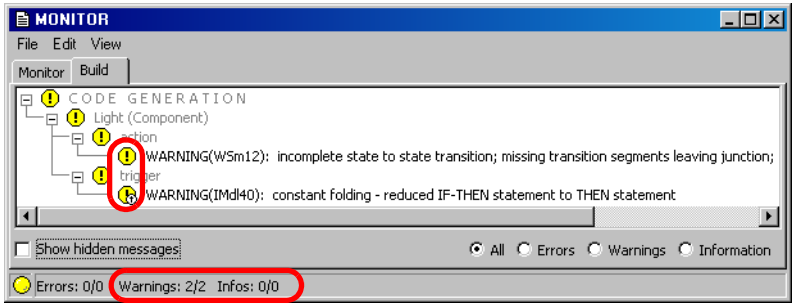
- **Promote to error** を選択します。

次回のコード生成時から、選択されたタイプのメッセージがプロモートされ、ワーニングまたはエラーとして表示されます。

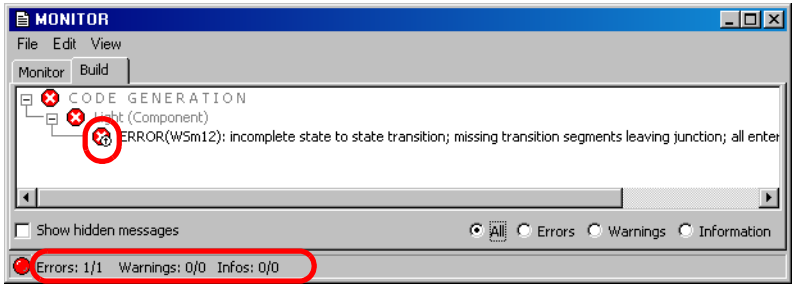
以下に、プロモーションの仕組みを具体的に説明します。次の図では、コード生成によってワーニングメッセージと情報メッセージが 1 つずつ発行されています。



そして次の図では、IMdl40 というタイプの情報が、ワーニングにプロモートされています。タイプは IMdl40 のままですが、プロモートされたものであることを示すマークが付加されたワーニングアイコンが表示されています。



さらに次の図では、WSm12 というタイプのワーニングがエラーにプロモートされています。タイプは WSm12 のままですが、プロモートされたものであることを示すマークが付加されたエラーアイコンが表示されています。この「プロモートされたエラー」が発生した時点でコード生成が中断されたため、次のステップは実行されていません。そのため、“Build" タブには 1 つのメッセージしか表示されていません。



“Build” タブ内でプロモートされているメッセージは、次のようにして元に戻すことができます。

プロモーションを取り消す：

- “Build” タブで、プロモートされているメッセージを右クリックします。
- **Revoke Promotion** を選択します。

  次回のコード生成時から、選択されたメッセージはもとのレベルに戻ります。

“CodeGen Message Configuration” ダイアログボックスでのメッセージ設定

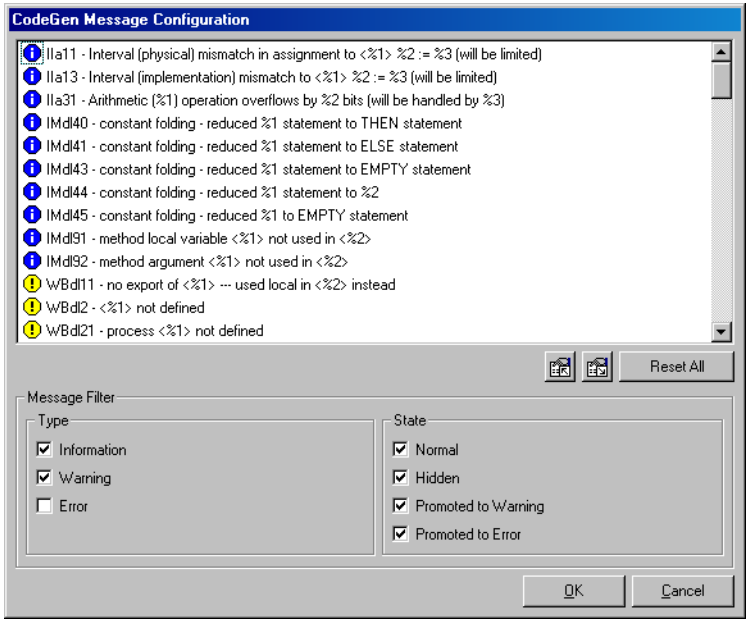
“CodeGen Message Configuration”ダイアログボックスでは、その他のコンフィギュレーション設定を行えます。ここでは、現在表示されているもの以外のタイプのメッセージについても、プロモートや表示／非表示の切り替えを設定することができます。

“Build”タブに 1 回以上メッセージが表示されると、その後、モニタウィンドからこのダイアログボックスを開くことができます。プロジェクトオプションから“CodeGen Message Configuration”ダイアログボックスを開く方法については、405 ページの「ビルドオプション（“Build”ノード）」を参照してください。

“CodeGen Message Configuration”ダイアログボックスを開く：

- “Build”タブを右クリックします。
- ショートカットメニューから **Settings** を選択します。

  “CodeGen Message Configuration”ダイアログボックスが開きます。



このダイアログボックスには以下の情報が含まれます。

- リストフィールド

  すべてのタイプの情報、ワーニング、エラーが、アルファベット順に表示されます。“Message Filter“フィールドで、表示されるタイプを指定することができます。

- **mport Message Configuration from XML File and Export Message Configuration to XML File** ボタン

  メッセージコンフィギュレーションのインポート／エクスポートを行います。

- **Reset All** ボタン

  すべてのメッセージを、システムで定義されたレベルに戻します。

- “Message Filter”フィールド

  リストフィールドに表示するメッセージを指定します。

  — “Type”フィールド（**Information** / **Warning** / **Error**）

  表示するメッセージレベルを指定します。さらに以下のオプションで表示するタイプを詳しく指定できます。

  — **Normal**

  オリジナルの状態（非表示になっておらず、プロモートもされていないもの）のメッセージの表示／非表示を指定します。

  — **Hidden**

  非表示になっているメッセージの表示／非表示を指定します。

  — **Promoted to Warning** / **Promoted to Error**

  ワーニングまたはエラーにプロモートされているメッセージの表示／非表示を指定します。

表示を設定する：

- “CodeGen Message Configuration”ダイアログボックスのリストフィールドに表示したいメッセージレベルを設定します。

  Information、**Warning**、**Error** についてそれぞれ設定できます。

- **Normal**、**Hidden**、**Promoted to Warning/Error**）をそれぞれ設定します。

  これによって、任意のタイプのメッセージのみをリスト表示することができます。たとえば以下のように設定すると、非表示になっているメッセージのみが表示されます。

モニタウィンドウ内でメッセージを非表示にする：

注記

エラーメッセージ（情報またはワーニングから「プロモート」されたものを含みます）を非表示にすることはできません。

- “CodeGen Message Configuration”ダイアログボックスのリストフィールドから、非表示したいメッセージを選択します（複数選択可能）。
- ショートカットメニューから **Hide** を選択します。

  プロモートされているメッセージには、リスト内で薄い色のアイコン  で示されます。
- **OK** をクリックして“CodeGen Message Configuration”ダイアログボックスを閉じます。

  選択されたタイプのメッセージが、モニタウィンドウの“Build”タブにおいてすべて非表示となります。

  モニタウィンドウのステータスバーに表示されるメッセージ数が調節されます。すべてのワーニングとエラーが非表示になっていると、ステータスバーの左隅に緑色の丸いアイコンが表示されます。また非表示になっているメッセージの数も表示されます。

非表示になっているメッセージを元に戻す：

- “CodeGen Message Configuration”ダイアログボックスのリストフィールドから、非表示になているメッセージを選択します（複数選択可能）。

  Normal と **Promoted to *** をオフにすると、非表示になっているメッセージのみが表示され、目的のメッセージを見つけやすくなります。
- ショートカットメニューから **Show** を選択します。
- **OK** をクリックして“CodeGen Message Configuration”ダイアログボックスを閉じます。

  選択されたタイプのメッセージが、モニタウィンドウの“Build”タブにおいてすべて表示されるようになります。

情報とワーニングをプロモートする：

- “CodeGen Message Configuration”ダイアログボックスのリストフィールドから、プロモートしたいメッセージを選択します（複数選択可能）。
- ショートカットメニューから **Promote to warning** を選択します。

または

- ショートカットメニューから **Promote to error** を選択します。
- **OK** をクリックして“CodeGen Message Configuration”ダイアログボックスを閉じます。

選択されたタイプのメッセージは、次回のコード生成時から、上位レベルのメッセージとしてモニタウィンドウの“Build”タブに表示されます。

プロモートされたメッセージは、リスト内で以下のアイコンで示されます。


「ワーニング」にプロモートされた「情報」


「エラー」にプロモートされた「情報」


「エラー」にプロモートされた「ワーニング」

プロモートされた情報とワーニングを元に戻す：

- “CodeGen Message Configuration”ダイアログボックスのリストフィールドから、元のレベルに戻したいメッセージを選択します（複数選択可能）。
- ショートカットメニューから **Revoke Promotion** を選択します。
- **OK** をクリックして“CodeGen Message Configuration”ダイアログボックスを閉じます。

選択されたタイプのメッセージは、次回のコード生成時から、元のレベルのメッセージとしてモニタウィンドウの“Build”タブに表示されます。

ここで設定した“Build”タブのコンフィギュレーションは、XML ファイルにエクスポートでき、後にその情報をインポートすることができます。

設定をエクスポートする：

- “CodeGen Message Configuration”ダイアログボックスを開きます。

- **Export Option of selected Node into XML File** ボタンをクリックします。

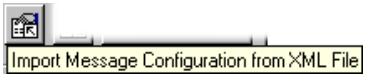
  確認のダイアログボックスが開きます。
- 今後、この確認ダイアログが表示されないようにするには、**Show next time** オプションをオフにします（46 ページ参照）。
- **OK** をクリックして処理を続行します。

  Windows のファイル選択ダイアログボックスが開きます。ファイルフォーマットには `*.xml` が選択されています。
- エクスポートファイルのパスとファイル名を指定します。
- **Save** をクリックします。

  非表示とプロモーションの設定が XML ファイルに書き込まれます。

設定をインポートする：

- “CodeGen Message Configuration”ダイアログボックスを開きます。



- **Import Message Configuration from XML File** ボタンをクリックします。

  Windows のファイル選択ダイアログボックスが開きます。ファイルフォーマットには `*.xml` が選択されています。
- インポートしたい設定が保存されているXMLファイルを選択します。
- **Open** をクリックします。

  非表示とプロモーションの設定が XML ファイルから読み込まれ、“CodeGen Message Configuration”ダイアログボックスとモニタウィンドウの表示に反映されます。

3 ユーザー定義機能の追加

ASCET のアドオン製品（バージョン管理ツール等）や他の外部プログラムを使用する場合、以下のようなユーザー独自の機能を定義して容易に操作を行うことができます。

- 任意の ASCET ウィンドウに独自のメニューコマンドを追加
- ASCET 起動時に実行されるオートスタートアクション
- ASCET 終了時に実行されるシャットダウンアクション

これらの情報は、ASCET のインストールディレクトリのサブディレクトリである `ETAS¥Ascet5.2¥Executer`、またはデータディレクトリのサブディレクトリである `ETASData¥Ascet5.2¥Executer` 内の複数の `*.ini` ファイルに格納します。このサブディレクトリ `Executer` には、さらにサブディレクトリを含めることもできます。

注記

ASCET のバージョンアップや再インストールを行う際に `Executer` サブディレクトリの内容がそのまま保たれるようにするためには、このディレクトリを `ETASData¥Ascet5.2` 以下に配置しておいてください。

`*.ini` ファイルは ASCET の起動時にのみ読み込まれ、その後に変更された内容は、ASCET の次回の起動時に初めて有効になります。このファイルの記述形式は、Windows の一般的な `*.ini` ファイルと同じです。

各 `*.ini` ファイルには 1 つまたは複数のセクションを含めることができ、各セクションの名前は [ ] で囲みます。以下のようなセクションが使用可能です。

- `[<NAME>]` ー 新しいメニューアイテムの定義

  任意の名前を使用できますが、1 つの `*.ini` ファイルに同じ名前のメニューを定義することはできません。3.1「メニューアイテムの定義」を参照してください。
- `[AUTOSTART]` ー オートスタートアクションの定義

  3.2「オートスタートアクションの定義」を参照してください。
- `[SHUTDOWN]` ー シャットダウンアクションの定義

  3.3「シャットダウンアクションの定義」を参照してください。

3.1 メニューアイテムの定義

メニューアイテムは、`*.ini` ファイル内の以下のようなセクションで定義します。

```
[<Name>]
WINDOW=<window name>
MENU=~<menu name>
```

```
ITEM=~<menu item name>
DESCRIPTION="<text>"
SEPARATOR=<separator>
FILE=<filename>.txt
```

注記

等号の左側の語句は、必ず大文字にしてください。

各変数名の意味 (機能) は以下のとおりです。

- <Name> はセクション名です。1 つの *.ini ファイル内には同じセクション名は存在できません。大文字と小文字は区別されないため、たとえば [Input] と [INPUT] は同じものとして扱われます。

  値：任意、ただしファイル内でユニークであること。

- <window name> はメニューが追加されるウィンドウ名です。

  値：161 ページの表 3-1 を参照してください。

- <menu name> は追加されるメニューの名前です。

  値：任意

- <menu item name> は <menu name> メニューに追加されるメニューアイテムの名前です。

  値：任意

- <text> はメニューアイテムの説明で、ドキュメンテーションの目的にのみ使用されます。

  値：任意

- <separator> は、メニューアイテムの前後にセパレータを表示するかどうかを指定します。セパレータが必要ない場合（以下の例では Nothing)、この行は省略できます。

  値：Before / After / Nothing

- <filename> ― メニュアイテムが選択されると、<filename>.txt というスクリプトファイルが実行されます。このファイルの構成は、3.4「スクリプトファイルの構成」を参照してください。

  値：任意

<window name>	ASCET ウィンドウ
mainwindow または databasebrowser	コンポーネントマネージャ
projecteditor	プロジェクトエディタ
blockdiagrameditor	ブロックダイアグラムエディタ
smeditor	ステートマシンエディタ
esdleditor	ESDL エディタ
ccodeeditor	C コードエディタ
conttimeblockdiagrameditor	ブロックダイアグラムエディタ（CT ブロック）
conttimeesdleditor	ESDL エディタ（CT ブロック）
conttimeccodeditor	C コードエディタ（CT ブロック）
booleantableeditor	論理テーブルエディタ
conditionaltableeditor	条件テーブルエディタ
datadialog	“Data for ...” ダイアログ
impldialog	インプリメンテーションエディタ
onlinesimulation	オンライン実験環境
offlinesimulation	オフライン実験環境
intecriobackanimation	INTECRIO でのバックアニメーション用実験環境
incabackanimation	INCA でのバックアニメーション用実験環境

表 3-1　メニューアイテムを追加できるウィンドウの名前

複数の *.ini ファイルが存在する場合、それらのファイルはアルファベット順に評価されます。同じメニューに複数のメニューアイテムが存在する場合、asd_menu.ini という名前のファイル内で定義されたアイテムが、my_menu.ini 内で定義されたものより上に表示されます。また同じ *.ini ファイル内では、定義された順に表示されます。

メニューアイテムを定義する：

- 新しい *.ini ファイルを作成します。

または

- 既存の *.ini ファイルを開きます。
- セクション名を入力します。

  例： [Input]

  同じファイル内に同じセクション名が 2 つ以上存在しないようにしてください。

- メニューを追加するウィンドウ名を入力します。

  例：`WINDOW=databasebrowser`
- メニュー名を入力します。

  例：`MENU=~My Menu`
- メニューに含まれるメニューアイテム名を入力します。

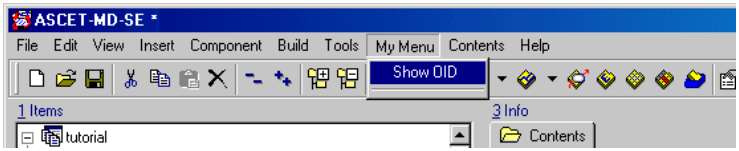
  例： `ITEM=~Show OID`
- 注釈を入力します。

  例： `DESCRIPTION=show object IDs`
- セパレータを表示する位置を入力します。

  例： `SEPARATOR=After`
- 実行するスクリプトファイル名を入力します。

  例： `FILE=show.txt`

上記の例のように設定した後に ASCET を起動すると、以下のようにコンポーネントマネージャに **My Menu** というメニューが追加されます。



同じウィンドウまたは異なるウィンドウに複数のメニューを追加するには、個々のメニューアイテムごとにこのようなセクションを定義する必要があります。以下のように記述しても、2 番目のアイテムは無視されます。

```
[EXPORT]
WINDOW=databasebrowser
MENU=~My Menu
ITEM=~Show OID
DESCRIPTION=show object IDs
FILE=show.txt
ITEM=~open editor
DESCRIPTION=opens external text editor
FILE=openeditor.txt
```

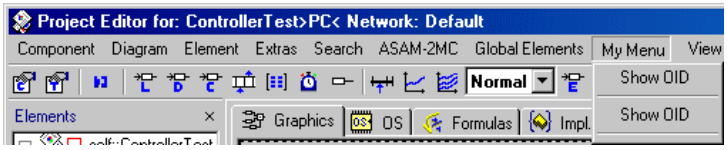
複数のセクションを、1 つまたは複数の `*.ini` ファイルに定義することができますが、メニュー内の複数のアイテムはファイル名の順番（アルファベット順）に従って表示されるため、別のファイルに分けた方が各アイテムの表示位置の指定が容易になります。

1 つの *.ini ファイルのみを使用した場合、同じセクション名が 2 個所以上存在しないようにしてください。もしも同じ名前が 2 個所存在した場合、2 個所目の定義内容に関わらず、1 個所目の内容が 2 回実行されてしまいます。

例：以下の 2 つのセクションは、1 つはプロジェクトエディタに **My Menu → Show OID** というメニューアイテムを追加するもので、もう 1 つはコンポーネントマネージャに **My Menu → Open Editor** というメニューアイテムを追加するためのものです。

```
[Input]
WINDOW=projecteditor
MENU=~My Menu
ITEM=~Show OID
DESCRIPTION=show object IDs
SEPARATOR=After
FILE=show.txt

[INPUT]
WINDOW=databasebrowser
MENU=~My Menu
ITEM=~Open Editor
DESCRIPTION=open external text editor
SEPARATOR=After
FILE=openeditor.txt
```

大文字と小文字は区別されないため、上記の 2 つのセクション名は同じであると判断され、**My Menu** というメニューの下に **Show OID** というメニューアイテムが 2 回生成されてしまいます。



3.2 オートスタートアクションの定義

オートスタートアクションは、*.ini ファイルの [AUTOSTART] セクションに定義します。セクションは *.ini ファイル内のどの位置にでも定義できますが、複数の *.ini ファイルが存在する場合でも、1 つのファイルの 1 個所にしか定義できません。このセクションには、FILE=<filename> という 1 行のみが含まれます。

例：

```
[AUTOSTART]
FILE=start.txt
```

この例では、ASCET 起動時に `start.txt` というスクリプトファイルが実行されます。

オートスタート機能は、ASCET 起動時にファイルオペレーションに関する処理（例：コード生成用のテンプレート設定、ディレクトリの削除）を実行するものです。これによって、毎回同じ条件で ASCET を起動することが可能となります。またエディタやその他のツールを呼び出して、セッションの準備に必要な作業を行うこともできます。設定された内容をメッセージ（167 ページ参照）で表示することもできます。

スクリプトは、たとえば以下のように記述されます。

```
EXECUTE c:¥Programme¥TextPad 4¥TextPad.exe
        c:¥ETAS¥Ascet5.2¥target¥c16x¥codegen.ini
OBJECT message: "codegen.ini is updated"
```

この時点でオートメーションサーバはまだ起動していないため、オートスタートアクション内で COM-API にアクセスすることはできません。

3.3 シャットダウンアクションの定義

シャットダウンアクションは、`*.ini` ファイルの `[SHUTDOWN]` セクションに定義されます。セクションは `*.ini` ファイル内のどの位置にでも定義でき、このセクションには、`FILE=<filename>` という 1 行のみが含まれます。

例：

```
[SHUTDOWN]
FILE=shutdown.txt
```

この例では、ASCET 終了時に `shutdown.txt` というスクリプトファイルが実行されます。

この時点でオートメーションサーバはすでに終了しているため、シャットダウンアクション内で COM-API にアクセスすることはできません。

3.4 スクリプトファイルの構成

スクリプトファイル（`*.ini` ファイル内の `File =...` という行で指定されるファイル）は、テキストファイルとして作成します。以下の 9 種類のキーワードで各アクションを記述します。

```
EXECUTE, NOWAIT, OBJECT, SELECTOR, MENUITEM, FILE, FORK,
SEND, Post
```

以下に、各キーワードについて説明します。

スクリプトファイルの内容が変更されると、*.ini ファイルの場合とは異なり、そのファイルを使用するメニューが次に選択された時、すぐにその内容が有効となります。

3.4.1 EXECUTE

EXECUTE というキーワードによって、外部プログラムやバッチファイルが実行されます。外部ファイルが実行されている間は、スクリプトファイルの実行は保留されます。

例：

- `EXECUTE C:¥ETAS¥Ascet5.2¥import.bat` は、`C:¥ETAS¥Ascet5.2` というディレクトリの `import.bat` ファイルを呼び出します。
- `EXECUTE C:¥PVCS¥nt¥PVCSvmnt.exe` は、バージョン管理プログラム・PVCS を起動します。

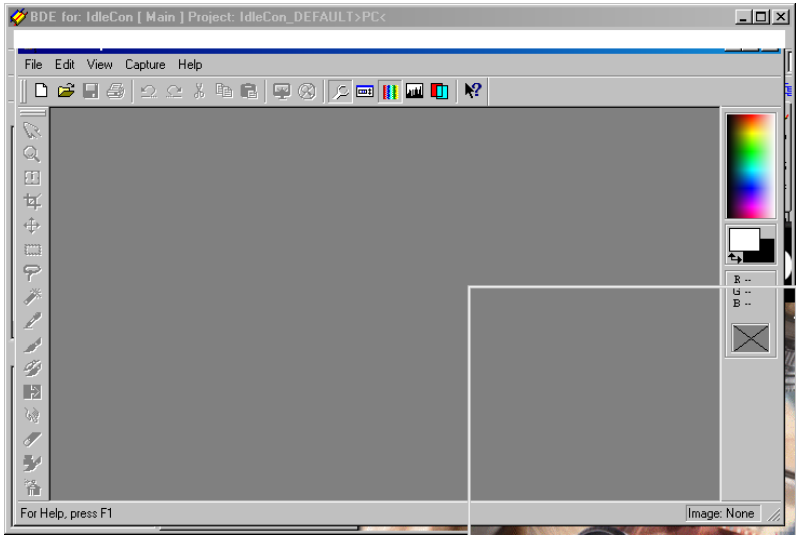
スクリプトファイルの実行が保留されると、予想外の状態が生じる場合があります。たとえば、ASCET の画面コピーを容易に作成することができるように、**My Menu → open PSP** というメニューアイテムを追加したとします。すると、以下のような不具合が発生します。

- メニューアイテムで以下の内容のスクリプトファイルが呼び出されます。

```
OBJECT popWindow: BDE
EXECUTE c:¥Programme¥Paint Shop Pro 5¥psp.exe
```

- 1 行目でブロックダイアグラムエディタが開き、2 行目でイメージ処理プログラムが起動されます。

  しかしこの時スクリプトファイルの実行が保留されるため、ブロックダイアグラムエディタは更新されず、以下のような状態になってしまいます。



2 つの行の位置を入れ替えてもこの問題は解決しません。入れ替えによって、ブロックダイアグラムはイメージ処理プログラムが終了してから表示されるようになってしまうためです。

3.4.2 NOWAIT

キーワード `NOWAIT` は、`EXECUTE` と同じように使用されますが、スクリプトファイルの実行は保留されません。

例：

下記の例は `EXECUTE` の例と同じですが、`EXECUTE` の代わりに `NOWAIT` が使用されています。

```
OBJECT popWindow: BDE

NOWAIT c:¥Programme¥Paint Shop Pro 5¥psp.exe
```

前の例と同様に 1 行目でブロックダイアグラムエディタが開き、2 行目でイメージ処理プログラムが起動されますが、ここではスクリプトファイルの実行は保留されないため、ブロックダイアグラムエディタはすぐに更新され、画面コピーを正しく取得できます。

3.4.3 OBJECT

OBJECT というキーワードは、何種類かのコマンドを表す識別子と共に使用されます。

- `help: <editor>`

  指定のエディタで exe_hlp.txt というヘルプファイルを表示します。

  例：`OBJECT help: notepad` と定義されていると、ノートパッドエディタ（メモ帳）でヘルプが表示されます。

注記

どの識別子にも、:（コロン）のあとにスペースが必要です。コロンが抜けていると、コマンドは実行されません。

- `log: <editor>`

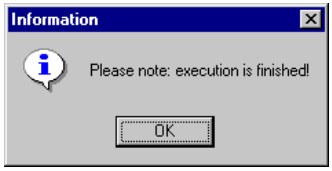
  指定のエディタで exe_log.txt というログファイルを表示します。

  例：`OBJECT log: notepad`

- `message: <text>`

  ユーザー向けのメッセージを表示するウィンドウを開きます。

  例：`OBJECT message: “Please not: execution is finished!”` と定義されていると、以下のようなウィンドウが開きます。



- `popWindow: <title>`

  指定の ASCET ウィンドウがポップアップ表示されます。<title> の部分には、タイトルバーに表示されるウィンドウ名、またはその一部を指定します。

  例：`OBJECT popWindow: Database` と定義されていると、コンポーネントマネージャがポップアップ表示されます。

- `generatePopWindowCommandFile: <filename>`

  指定の名前のファイルが作成され、現在アクティブな ASCET ウィンドウから実行された `popWindow` コマンドが格納されます。ファイルは必ず `<filename>` で指定し、パス名は任意に付加します。

  例：`OBJECT generatePopWindowCommandFile: pop.txt` と定義されていると、コンポーネントマネージャから呼び出された時に、`pop.txt` という名前のファイルが以下の内容で作成されます。

```
OBJECT popWindow: "Database Browser :<tutorial>"
```

- `wait: <n>`

  スクリプトファイルの実行を、n 秒間停止させます。

  例：

  ```
  OBJECT wait: 5
  EXECUTE C:¥PVCS¥nt¥PVCSvmnt.exe
  ```

  という 2 行が定義されていると、PVCS が起動されるまで 5 秒間のディレイが確保されます。

- `windows: <editor>`

  指定のエディタで、現在開いているすべての ASCET ウィンドウの Smalltalk 名のリスト（`exe_win.txt`）を表示します。

  例：`OBJECT windows: notepad` と定義されていると、ノートパッドで表示されます。

3.4.4 SELECTOR

`SELECTOR` というキーワードは、コンポーネントマネージャや各種機能記述エディタ内で、さまざまな定義済みコマンドを呼び出すために使用されます。

例： • 選択されたコンポーネントのオブジェクト `ID` の出力

`SELECTOR executerFileOutSelectedElementsTo: "<filename>"` と定義されていると、オブジェクト ID を含んだファイルが生成されます。ファイル名 `<filename>` で指定し、パス名は任意に付加します。

注記

`<filename>` 内にパス名を含める場合は、そこに含まれるすべてのサブディレクトリが正しく存在しているかどうかを確認する必要があります。存在しない場合、ファイルは生成されません。

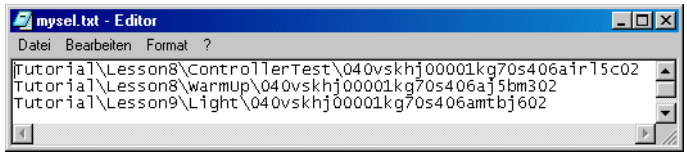
コンポーネントマネージャに、以下のスクリプトファイルを呼び出すメニューアイテムが追加されます。

```
SELECTOR executerFileOutSelectedElementsTo:
     “.¥Executer¥mysel.txt”

EXECUTE Notepad .¥Executer¥mysel.txt

MENUITEM View>Update
```

`ControllerTest`、`WarmUp`、および `Light` というコンポーネントが選択され、メニューアイテムが呼び出されると、以下の処理が行われます。

- 1 行目（`SELECTOR = ...`）で、`Executer` というサブディレクトリに `mysel.txt` が生成されます。

- 2 行目（EXECUTE ...）で、生成されたファイルがメモ帳で開きます。



- 3 行目（MENUITEM ...）により、メモ帳が閉じた時にコンポーネントマネージャが更新されます。

使用できるコマンド：

- executerCollapseAll

  機能：“1 Database”ペインのエレメントリストをすべて省略表示します。

  使用できるウィンドウ：コンポーネントマネージャ

  文型：

  ```
  SELECTOR executerCollapseAll
  ```

- executerExpandCollapseSelectedElement

  機能：“1 Database”ペイン内で選択されたフォルダを展開または省略表示します。（サブフォルダは省略表示されますが、展開は行われません。）

  使用できるウィンドウ：コンポーネントマネージャ

  文型：

  ```
  SELECTOR executerExpandCollapseSelectedElement
  ```

- executerDeselectElements

  機能：エレメントやコンポーネントの選択を取り消します。

  使用できるウィンドウ：コンポーネントマネージャ、および機能定義エディタ

  文型：

  ```
  SELECTOR executerDeselectElements
  ```

- `executerFileOutSelectedDiagrams`

  機能：選択されたコンポーネントのすべてのダイアグラムと階層、およびその中に含まれるコンポーネントを、ASCET のインストールディレクトリにイメージファイルとして書き込みます。
  このコマンドは、ESDL および C コードコンポーネントには影響を与えません。

  使用できるウィンドウ：コンポーネントマネージャ

  文型：

  ```
  SELECTOR executerDeselectElements
  ```

- `executerFileOutSelectedElementsTo:`

  機能：選択されたエレメントのオブジェクト ID を、指定のファイルに書き込みます。

  使用できるウィンドウ：コンポーネントマネージャ、および機能記述エディタ

  文型[1]：

  ```
  SELECTOR executerFileOutSelectedElementsTo:
          "<filename>"
  ```

- `executerSelectElementFromString:`

  機能：“1 Database”リスト（コンポーネントマネージャの場合）または“Elements”リスト（機能記述エディタの場合）で、<name> という検索文字列で始まる名前のエレメントが最初に表示されるようにします。
  省略表示されたフォルダ内のエレメントやコンポーネントは、除外されます。

  使用できるウィンドウ：コンポーネントマネージャ、および機能記述エディタ

  文型[2]：

  ```
  SELECTOR executerSelectElementFromString: "<name>"
  ```

- `executerFileOutDiagrams`

  機能：編集対象のコンポーネントのすべてのダイアグラムと階層を、ASCET のインストールディレクトリにイメージファイルとして書き込みます。
  このコマンドは、ESDL および C コードコンポーネントには影響を与えません。

  使用できるウィンドウ：機能記述エディタ

  文型：

---

1. `<filename>` にはファイル名が必要で、パス指定はオプションです。
2. 文字列の検索時、大文字と小文字は区別されません。

```
SELECTOR executerFileOutDiagrams
```

- `executerFileOutSelectedDiagramTo:`

  機能：編集対象のコンポーネントの選択されたダイアグラムのパス名を、指定のファイルに書き込みます。ブロックダイアグラムの場合、選択されたダイアグラムとそれに含まれる階層についても、そのイメージが ASCET インストールディレクトリに生成されます。

  使用できるウィンドウ：機能記述エディタ

  文型：

  ```
  SELECTOR executerFileOutSelectedElementsTo:
          "<filename>"
  ```

- `executerSelectDiagramFromString:`

  機能：検索文字列 <name> を含むダイアグラムを選択し、ロードは行いません。

  使用できるウィンドウ：機能記述エディタ

  文型[1]：

  ```
  SELECTOR executerSelectElementFromString: "<name>"
  ```

- `executerSelectPage:`

  機能：指定のプロジェクトエディタタブを開きます。

  使用できるウィンドウ：プロジェクトエディタ

  文型：

  ```
  SELECTOR executerSelectPage: "<tab name>"
  ```

  （タブ名は全体を正確に記述する必要があります。）

3.4.5 MENUITEM

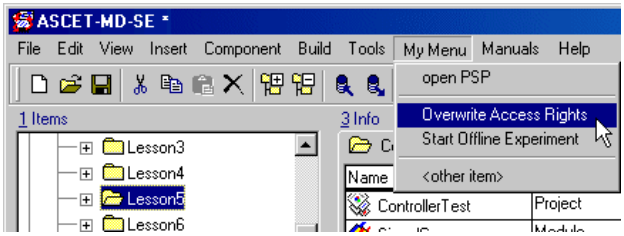
`MENUITEM` キーワードは、アクティブウィンドウから既存のメニューアイテムを呼び出す際に使用します。

例：

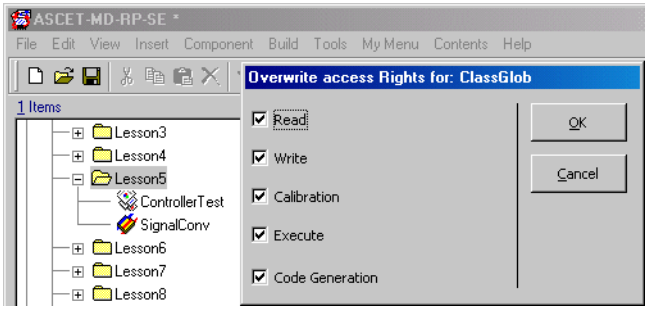
コンポーネントマネージャに **My Menu** というメニューが追加されているので、その中の **Overwrite Access Rights** というメニューアイテムが、以下の内容を含む `set_access.txt` というファイルを呼び出すようにします。

```
MENUITEM Item>Change Access Rights>Overwrite
```

1. 文字列の検索時、大文字と小文字は区別されません。



このメニューコマンドが実行されると、選択されているデータベースアイテムに対して“Overwrite Access Rights for ...”というダイアログウィンドウが開きます。



データベースアイテムのアクセス権についての詳細は、138 ページの 2.3.7「データベースアクセス」の章を参照してください。

3.4.6 FILE

`FILE` キーワードは、スクリプトファイルからスクリプトファイルを呼び出すために使用します。呼び出し元のスクリプトファイルの実行は、呼び出されたスクリプトファイルが実行されている間、保留されます。

例：

- `File set_access.txt` と定義されていると、現在のスクリプトファイルから `set_access.txt` というスクリプトファイルが呼び出されます。

3.4.7 FORK

`FORK` キーワードは、`FILE` キーワードと同様に使用されますが、`FORK` を使用すると呼び出し元のスクリプトファイルの実行は保留されません。

3.4.8 SEND

`SEND` キーワードは、Windows メッセージを他の実行中のアプリケーションに送るするために使用されます。ASCET は送信先のアプリケーションがメッセージの処理を終了するまで待ちます。以下のようなメッセージ送信を行えます。

```
SEND <class> <message>
```

```
SEND <class> <window> <message>
SEND <class> <window> <message> <lparam>
SEND <class> <window> <message> <lparam> <wparam>
```

<class> はアプリケーションのクラス名です。

<window> はアプリケーションのウィンドウタイトルです。

<message> は送信するメッセージの名前と識別子です。

<lparam> は 1 番目のオプション引数です。（0 ... 4294967295）

<wparam> は 2 番目のオプション引数です。（0 ... 4294967295）.

メッセージ送信についての詳細は、Microsoft Windows Win32 インターフェースの解説書に含まれる SendMessage キーワードについての説明を参照してください。

3.4.9 POST

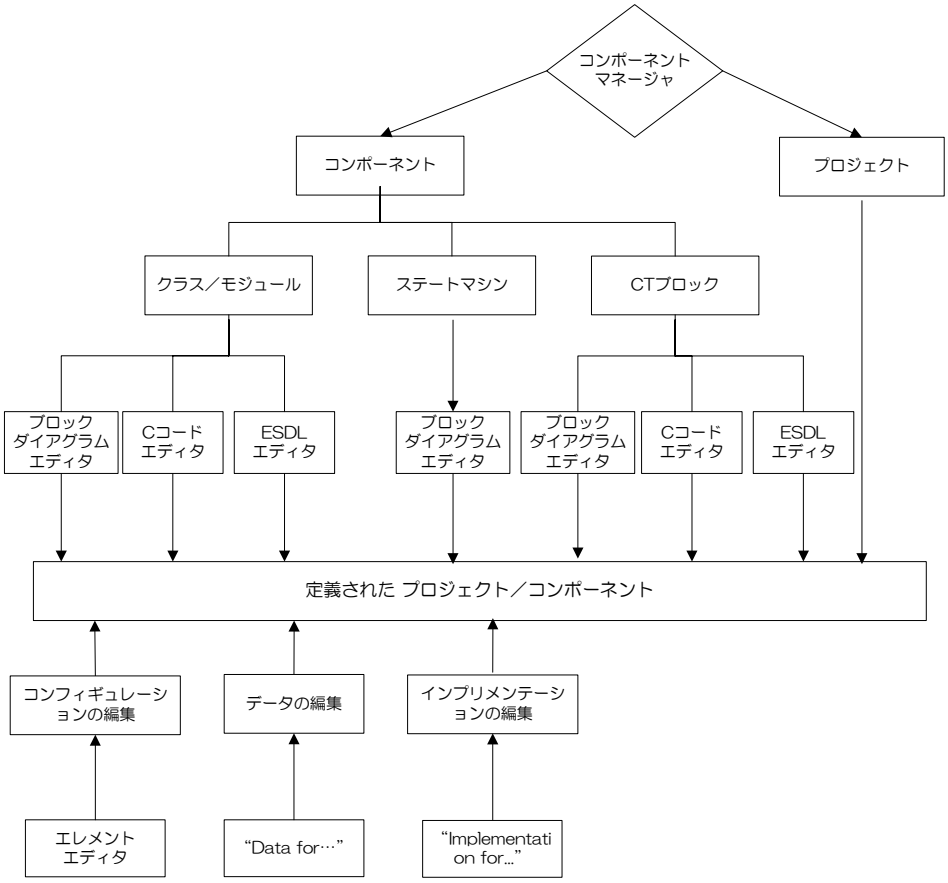
POST キーワードは、SEND と同様に Windows メッセージを他の実行中のアプリケーションに送るためのものですが、SEND とは異なり、ASCET は送信先アプリケーションによるメッセージ処理の終了を待ちません。

メッセージ送信についての詳細は、Microsoft Windows Win32 インターフェースの解説書に含まれる PostMessage キーワードについての説明を参照してください。

4 コンポーネントとプロジェクトの定義

組み込み制御システムは 1 つの「プロジェクト」として定義され、そのプロジェクトはさまざまな「コンポーネント」で構成されます。コンポーネントの種類と詳しい内容については、『ASCET リファレンスガイド』の「モデリング言語」の章を参照してください。

コンポーネントには、データ（変数の宣言、パラメータ）、インターフェース（メソッド、プロセス）、およびアルゴリズムが定義されます。



コンポーネントは、グラフィック、ESDL、C コードのいずれかの形式で記述することができます。

コンポーネント内に定義される各エレメントのプロパティ、インプリメンテーション、およびデータ（値）にはそれぞれデフォルト値が割り当てられ、エディタを用いてそれらの設定を変更することができます。

本章では、さまざまなコンポーネントの記述用エディタの使用方法や、コンポーネントに含まれるエレメントのプロパティの編集方法などについて詳しく説明します。

4.1 ブロックダイアグラムエディタ

本項では、ブロックダイアグラムの使用法について詳しく説明します。ここでは、特別な機能を開発することではなく、このエディタの一般的な使用方法に重点が置かれています。ASCET で使用するグラフィック言語については、『ASCET リファレンスガイド』の「モデリング言語」の章で詳しく説明されています。

ブロックダイアグラムエディタは、コンポーネントのグラフィック定義を行うためのエディタです。このエディタで、コンポーネントの処理内容を表わすブロックダイアグラムを作成します。

コンポーネント内には、基本エレメントや複合エレメント、演算子、処理フロー制御文、および他のコンポーネントへの参照を含めることができます。コンポーネントの作成方法については 77 ページの「データベースアイテムを作成する：」を参照してください。

ブロックダイアグラムエディタを開く：

- コンポーネントマネージャで、編集したいアイテムを強調表示します。
- アイテムをダブルクリックします。

または

- **Component → Edit Item** を選択します。

または

- ショートカットメニューから、**Edit Item** を選択します。

または

- **<Enter>** キーを押します。

選択されたアイテムがブロックダイアグラムエディタで開きます。

開こうとするコンポーネントのサイズが、現在の描画領域のサイズより大きい場合、メッセージダイアログボックスが開き、現在の描画領域のサイズとコンポーネントのサイズが表示されます。またサイズ変更のオプションも表示されます（179 ページの「描画領域のサイズを設定する：」を参照してください）。

ブロックダイアグラムエディタを閉じる：

- ブロックダイアグラムエディタで、**Component → Exit** を選択します。

または

- エディタウィンドウのタイトルバーの右上端の✕印をクリックします。

  ダイアグラムで変更された内容がまだ保存されていない場合、ここで変更内容を保存するかどうかを尋ねるメッセージが表示されます。

- 同じタイプの質問に対して毎回同じ回答を行うには、**Remember my Decision** オプションをオンにします。

  このようにすると、以後“Save Changes in Graphic”ダイアログボックスは開かなくなります。この設定は、ASCET オプションウィンドウの“Confirmation Dialogs”ノード（46 ページ参照）で確認／変更することができます。

- 保存するには **Yes** をクリックします。

  変更内容がキャッシュメモリに保存され、ダイアグラムエディタが閉じます。

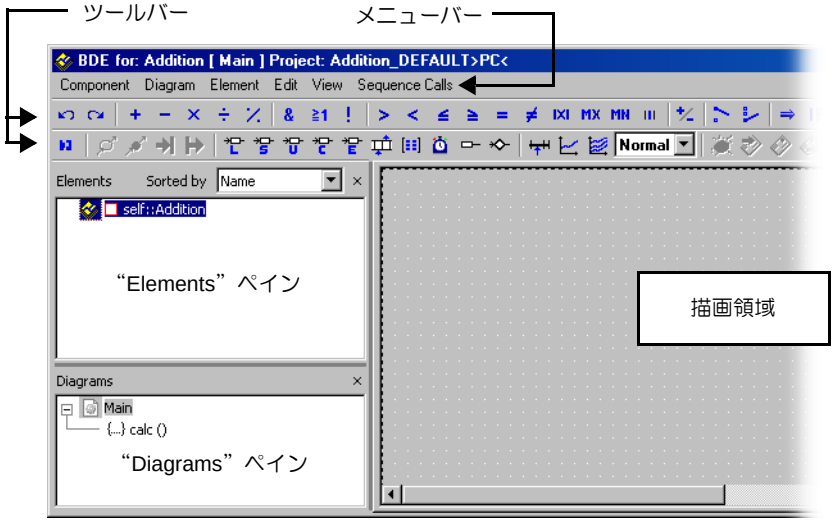
- 保存しない場合は **No** をクリックします。

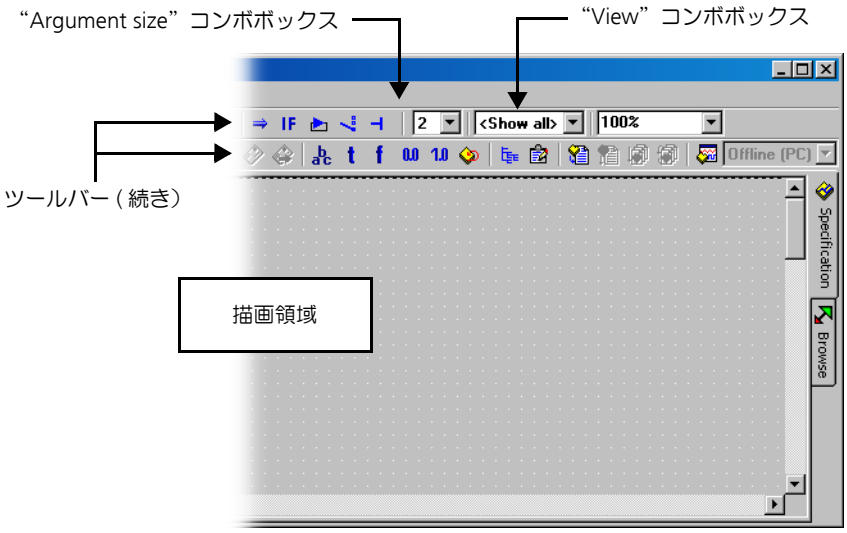
  変更内容は破棄され、ダイアグラムエディタが閉じます。

- ブロックダイアグラムを閉じる操作を中止するには **Cancel** をクリックします。

4.1.1 ブロックダイアグラムエディタのユーザーインターフェース

ブロックダイアグラムエディタには 2 種類の表示モードを切り替えるための 2 枚のタブがあり、ウィンドウ右側のタブをクリックするとモードが切り替わります。





上の図はブロックダイアグラムエディタで「定義ビュー」（”Specification”タブ）を開いた状態を表わし、重要な部分にはラベルを付けています。

ブロックダイアグラムエディタの「タイトルバー」には、現在開いているコンポーネント、ダイアグラム、プロジェクト、およびターゲットの名前が表示されます。

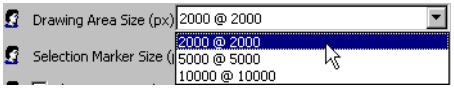
「描画領域」は、実際にブロックダイアグラムを記述する領域です。コンポーネントのエレメント（入力、出力、変数など）は“Elements”ペインにツリー構造で表示され、ツリーのルート（最上位の階層）にコンポーネント名が表示されます。ダイアグラム名、およびメソッド名またはプロセス名は、“Diagrams”ペインに表示されます。“Elements”および“Diagrams”ペインは、表示されないようにすることもできます。

描画領域のサイズは、ASCET オプションウィンドウで設定できますが、次のことに注意してください。旧バージョンの ASCET（V5.2.0 以前）の固定サイズ（2000 × 2000 ピクセル）より大きいサイズに設定し、モデリングにそれよりも大きいサイズのエリアを使用してモデリングを行った場合、旧バージョンでそのモデルを表示すると、コンポーネントの一部が表示されなくなってしまいます。この場合、描画領域の範囲の外側にあるエレメントは、コンポーネントから削除できなくなります。これは、描画領域からそのオカレンスを削除できなくなるためです。

サイズを変更すると、そのサイズがすべてのグラフィカルエディタ、つまり、ブロックダイアグラムエディタ（CT ブロックを含みます）、ステートマシンエディタ、プロジェクトエディタ（“Graphics”タブ）に適用されます。しかし、すでに開いているウィンドウに対しては適用されません。

描画領域のサイズを設定する：

- コンポーネントマネージャから **Tools → Options** を選択して ASCET オプションウィンドウを開きます。
- “Block Diagram”ノードを開きます（54 ページ参照）。
- “Drawing Area Size (px)”コンボボックスからサイズを選択します。



`2000 @ 2000` より大きいサイズを選択すると、以下のメッセージが表示されます。

```
If a size > 2000 @ 2000 px is
selected, components may be
displayed incompletely in ASCET
versions < V5.2.1!
```

- **OK** をクリックしてサイズを確定します。
- **OK** をクリックしてオプションウィンドウを閉じます。

以降、ブロックダイアグラムエディタ（またはその他のグラフィカルエディタ）を開くと、選択されたサイズのエディタが開きます。

現在選択されている描画領域サイズより大きいサイズのコンポーネントを開くと、メッセージダイアログボックスが開き、描画領域のサイズとコンポーネントのサイズが表示され、さらにサイズ変更のオプションも表示されます。

描画領域のサイズより大きいコンポーネントを開く：

- 同じタイプの質問に対して毎回同じ回答を行うには、**Remember my Decision** オプションをオンにします。

  このようにすると、以後“Save Changes in Graphic”ダイアログボックスは開かなくなります。この設定は、ASCET オプションウィンドウの“Confirmation Dialogs”ノード（46 ページ参照）で確認／変更することができます。
- 描画領域のサイズを調整するには、**Yes** をクリックします。

  コンポーネント全体が表示されるサイズが選択されます。
- 描画領域のサイズを変更しない場合は、**No** をクリックします。

  コンポーネントの中で、描画領域外に配置された部分は表示されません。またエリア外のエレメントは削除できません。

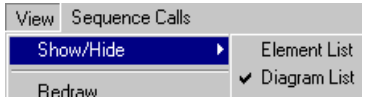
リストの表示／非表示を切り替える：

- 非表示にしたいリストの右上端の ×印をクリックします。

または

- **View → Show/Hide → Element List**（または **Diagram List**）を選択します。

  指定のリストが表示されなくなります。メニュー内の対応する項目のチェックマークが削除されます。

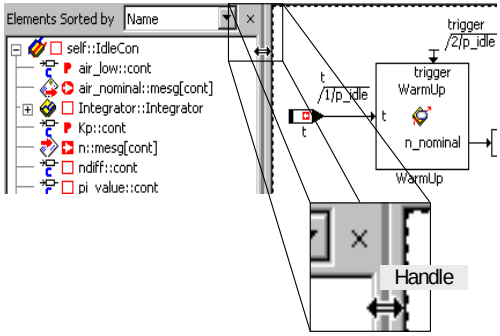


- リストを再表示するには、再度 **View → Show/Hide → Element List**（または **Diagram List**）を選択します。

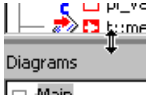
リストのサイズを調節する：

- マウスカーソルを“Elements”リストと描画領域の間のスプリットバーの上に置きます。

  マウスカーソルがリサイズ用ハンドルに変わります。



- スプリットバーを左右に動かしてリストの幅を調整します。



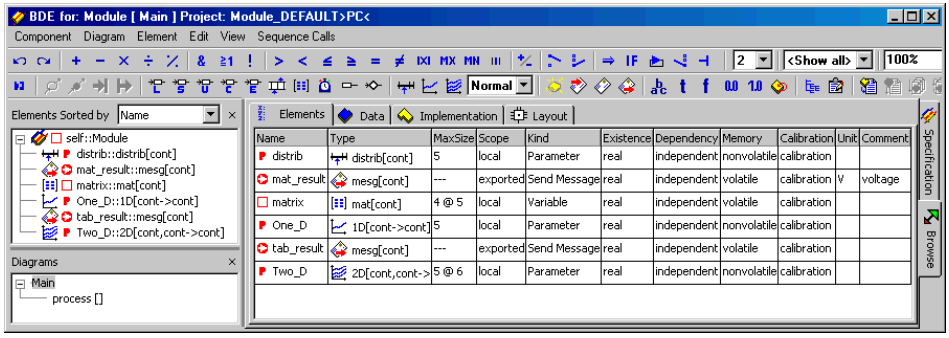
- “Elements”ペインと“Diagrams”ペインの間のスプリットバーを上下に動かして、リストの高さを調整します。

ブロックダイアグラムエディタを含むすべてのエディタでは、ブラウズモードに切り替えると、コンポーネントに含まれる各エレメントについての詳しい情報が表示されます。

表示モードを切り替える：

- **Browse** タブをクリックします。

  以下のような「ブラウザビュー」が表示されます。



このブラウザビューの構成と機能は、コンポーネントマネージャの各ビューモードと同じです（2.1.3 項と 2.3.2 項を参照してください）。ただしここでは、一部の編集用メニューコマンドはショートカットメニューからしか実行できません。

- **Specification** タブをクリックすると、「定義ビュー」に戻ります。

ビューについての詳細は、7.3 項 「ビュー」を参照してください。

ダイアグラムのビューを選択する：

- ツールバーの右上部にある“View”コンボボックスでビューを選択します。

ビューの設定に従ってダイアグラムアイテムが表示されます。非表示に設定されているアイテムは、すべて表示されなくなります。

メニューコマンドの概要

- **Component**
  - — *Clean code generation directory*

    コード生成ディレクトリ内のファイルをすべて削除します。

— *Touch*

次回のコード生成時に強制的にコードが再生成されるようにします。
→ *Flat* — 編集対象のコンポーネントのみ
→ *Recursive* — 被参照コンポーネントも含める

— *Generate Code*

コンポーネントのコードを生成します。

— *Compile*

生成されたコードをコンパイルします。

— *Open Experiment*

実験を開始します。

— *Default Project*

コンポーネントの実験に使用されるデフォルトプロジェクトの編集を行います。

→ *Edit* — デフォルトプロジェクト用のエディタを開きます。
→ *Resolve Globals* — コンポーネント内のインポートエレメントのうち、それに対応するエクスポートエレメントがないものに対して、グローバルエレメントが生成されます。
→ *Delete Unused Globals* — 実際には使用されていないグローバルエレメントを削除します。

— *Edit Layout*

レイアウトエディタを開きます。

— *Edit Data*

コンポーネント用のデータエディタを開きます。コンポーネントのデータの検索が可能です。

— *Edit Implementation*

インプリメンテーションエディタを開きます。コンポーネントのインプリメンテーションの検索が可能です。

— *Edit Notes*

注釈エディタを開きます。このエディタでは、コンポーネントについての注釈を入力することができます。

— *Check Dependency*

仮パラメータのモデルパラメータへの割り当てが正しいかどうかを調べます。

— *Export Data*

コンポーネントのデータセットをエクスポートします。

— *Delete Unused Elements*

ダイアグラムに使用されていないエレメントを“Elements”ペインから削除します（ステートマシンの場合、この機能は無効です）。

— *Show Path*

内包されるコンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *File out Generated Code*

生成されたコードをファイルに保存します。
→ *Flat* — 編集対象コンポーネントのみ
→ *Recursive* — 被参照コンポーネントも含める

— *View Generated Code*

コンポーネント用のコードを生成し、それをテキストエディタで表示します。
テキストエディタは、ASCET オプションウィンドウの“ASCET Editor”ノード（60 ページ参照）

— *File Out Generic Code For External Make*

後で外部ツールを使用して行う Make やビルド処理などを行うために、生成されるコードを任意のディレクトリに保存します。

— *Exit*

ブロックダイアグラムエディタを終了します。

- **Diagram**
（“Diagrams”ペインのショートカットメニューからでも実行可能）

— *Store to Cache*

コンポーネントの定義内容をキャッシュメモリに保存します。（データベースに永久的に保存するわけではありません。）

— *Analyze Diagram*

現在開いているダイアグラムを解析します。

— *Load Diagram*

ダイアグラムをロードします。

— *Add Diagram*

新しいダイアグラムを作成します。

→ *Public* — パブリックメソッドのみ
→ *Private* — プライベートメソッドのみ

- *Rename Diagram*

  ダイアグラムの名前を変更します。
- *Delete Diagram*

  ダイアグラムをコンポーネントから削除します。
- *Add Method*

  メソッドを作成します。このコマンドはクラスとモジュールについてのみ有効です。
- *Add Process*

  プロセスを作成します。
- *Add Trigger / Action / Condition*

  ステートマシンエディタでのみ有効です（267 ページの「特殊なメニューコマンド」を参照してください）
- *Edit*

  メソッド／プロセスのインターフェースを編集します。
- *Copy*

  選択されたメソッド／プロセスのコピーを作成します。
- *Rename*

  メソッド／プロセスの名前を変更します。
- *Delete*

  メソッド／プロセスを削除します。
- *Move Up*

  ダイアグラムまたはメソッド／プロセスを上方に移動します。
- *Move Down*

  ダイアグラムまたはメソッド／プロセスを下方に移動します。
- *Move*

  メソッド／プロセスを他のダイアグラムに移動します。
- *Edit Implementation*

  メソッド／プロセスのインプリメンテーションを定義します。

- **Element**
  （“Elements”ペインのショートカットメニューからでも実行可能）

注記

以下のコマンドのうち、**Add Item**、**Rename**、**Edit** コマンドは、ASCET-MD がインストールされている場合にのみ使用できます。

— *View → Collapse all*

"Elements" ペイン内のツリー構造をすべて格納し、コンポーネントのみが表示されるようにします。

— *View → Expand all*

ツリー構造を展開し、各コンポーネントの内容すべてが表示されるようにします。

— *Add Item*

コンポーネントを複合エレメントとして組み込みます。

— *Rename*（**<F2>**）

選択されているダイアグラムエレメントの名前を変更します。

— *Delete*（**<Del>**）

選択されているダイアグラムエレメントを削除します。

— *Show Occurrences*

エレメントのすべての「オカレンス」を表示します。「オカレンス」とは、ダイアグラム上に配置されたダイアグラムアイテムを指します。（ブロックダイアグラムエディタでのみ有効）

— *Edit*

選択されているエレメントのプロパティを編集します。

— *Edit Data*

選択されているエレメントのデータを編集します。

— *Edit Implementation*

選択されているエレメントのインプリメンテーションを編集します。

— *Set Cache Locking*

このコマンドは ASCET-RP 用のものです。詳しくは ASCET-RP のユーザーズガイドを参照してください。

— *Edit Component*

選択されている内部コンポーネント（現在編集中のコンポーネントに内包されるコンポーネント）のコンポーネントエディタを開きます。

— *Replace Component*

コンポーネントを別のコンポーネントで置き換えます。コンポーネント名は古いコンポーネントのものがそのまま使われます。

— *Show Path*

選択されている内部コンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル

内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* ― データベースパス
→ *Database asd:// link* ― ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

― *Notes*

選択されている内部コンポーネントの注釈エディタを開きます。

― *Edit Distribution*

選択されているディストリビューション用のエディタを開きます。（グループカーブ／マップの場合のみ有効）

― *Edit Max Size*

配列、マトリックス、特性カーブ／マップの最大サイズを定義するためのエディタを開きます。

― *Copy Elements*（**<Ctrl> + <C>**）

“Elements”ペインから、選択されている 1 つまたは複数のエレメントをコピーします。

― *Paste Elements*（**<Ctrl> + <V>**）

コピーされた 1 つまたは複数のエレメントを“Elements”ペインにペーストします。

― *File Out Data*

配列、マトリックス、特性カーブ／マップからファイルにデータを書き出します。

― *File In Data*

配列、マトリックス、特性カーブ／マップのデータをファイルから読み込みます。

― *Export Data*

内部コンポーネントからデータセットをエクスポートします。

- **Edit**

― *Cut*（**<Ctrl> + <X>**）

ダイアグラムアイテムをカット（削除）します。

― *Copy*（**<Ctrl> + <C>**）

ダイアグラムアイテムをコピーします。

― *Paste*（**<Ctrl> + <V>**）

ダイアグラムアイテムをペーストします。

― *Delete*（**<Del>**）

接続線またはエレメントを削除します。

— *File In Buffer*

ダイアグラムをファイルから読み込みます。

— *File Out Buffer*

ダイアグラムの一部をファイルに格納します。

— *Views*

“Views”ダイアログボックスを開きます。ダイアグラムアイテムの表示モードを、各ビューごとに選択できます。

- **View**

— *Show/Hide*

エディタまたはコンポーネントの一部の表示／非表示を切り替えます。

→ *Elements List* — “Elements”ペイン
→ *Diagram List* — “Diagrams”ペイン
→ *Connected Elements* — 選択されたダイアグラムエレメントに接続されているエレメントのブラウザビュー（219 ページ参照）
→ *Impl. Casts in Element List* — “Element”ペイン内のインプリメンテーションキャスト（他に表示指定されているものがない場合は、インプリメンテーションキャストのみが描画領域に表示されます）

— *Redraw*

ダイアグラム全体を再描画します。

— *Rebuild Connections*

演算子インプリメンテーションの自動変換により煩雑になった接続線を、スムーズな線に再描画します。

— *Undo*（**<Ctrl> + <Z>**）

最後の操作を取り消します。

— *Redo*（**<Ctrl> + <Y>**）

Undo コマンドを取り消します。

— *Show Hierarchy Path*

コンポーネントの完全な階層パスを表示します。

— *Parent Component*

編集中のコンポーネントの 1 階層上の親コンポーネント用のエディタを開きます。（このオプションは、内包されたコンポーネントの編集中に限り使用できます。）

— *Parent Hierarchy Level*

階層パスを表示します。

— *Page frame Portrait*

ダイアグラムを縦長のフォーマットで表示します。

— *Page frame Landscape*

ダイアグラムを横長のフォーマットで表示します。

— *Grid*

描画領域のグリッドを変更します。

— *Print Diagram*

コンポーネントを印刷します。

— *Save as Postscript/Bitmap/RTF/GIF*

指定されたフォーマットでダイアグラムを保存します。

- **Sequence Calls**

— *Sequencing - Ignore Info*

シーケンスコールの自動割り当てを行います。

— *Sequencing Starting With*

指定された番号から始まるシーケンスコールの自動割り当てを行います。

— *Sequencing Appending*

既存シーケンスの後ろに続くシーケンスコールを追加します。

— *Scale To Step Size*

シーケンスコールの間隔を設定されているスケールに変更します。

→ *For Diagrams* — ダイアグラム全体
→ *For Mithod/Process* — 1 つのメソッド／プロセスのみ

— *Reset*

シーケンスコールをリセットします。

→ *For Diagrams* — ダイアグラム全体
→ *For Method/Process* — 1 つのメソッド／プロセスのみ
→ *For Selected Blocks* — 選択されたブロックのみ

— *Show*

一連のシーケンスコールが表示されるようにします。

→ *For Diagrams* — ダイアグラム全体
→ *For Method/Process* — 1 つのメソッド／プロセスのみ
→ *For Selected Blocks* — 選択されたブロックのみ
→ *Unused* — 未使用のシーケンスコールのみ

— *Hide*

一連のシーケンスコールを非表示にします。
（条件は **Sequence Call → Show** と同じ）

— *Next Seq. Call*（**<Ctrl> + < → >**）

次のシーケンスコールを選択します。

— *Previous Seq. Call*（**<Ctrl> + < ← >**）

前のシーケンスコールを選択します。

4.1.2 コンポーネントのインターフェースの定義

コンポーネントを定義する場合、最初にそのコンポーネントのインターフェースを定義する必要があります。コンポーネントの「インターフェース」とは、コンポーネントが他のコンポーネントとどのような相互関係を持つか、つまり、どのようなデータをどのようなアドレッシングで受け渡しするかを規定するものです。ブロックダイアグラムエディタでコンポーネントを定義する際、コンポーネントのタイプによって最も大きく異なるのがこの「インターフェース」です。以下に、「クラス」の場合と「モジュール」の場合について別々に説明します。

クラス

クラスのインターフェースは、**パブリックメソッド**、および**引数**と**戻り値**です。引数がクラスの入力パラメータで、戻り値が出力パラメータとなります。メソッドに外部から入力を与えることにより、コンポーネント内の処理を開始させることができます。

メソッドを作成する：

- **Diagram → Add Method** を選択します。

  新しいメソッドが、“Diagrams”ペインに作成されます。
- そのメソッドの名前を入力して **<Enter>** を押します。

クラスには 1 つ以上のメソッドが必要なため、新しく作成されたクラスにはデフォルトメソッド `calc` が自動的に作成されます。1 つのメソッドには任意の数の引数と 1 つの戻り値を定義することができ、引数と戻り値の内容は、インターフェースエディタで変更します。

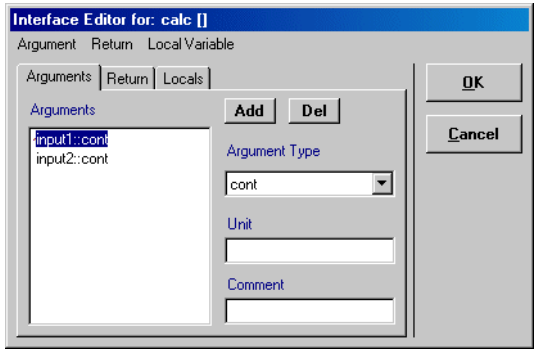
メソッドのインターフェースを編集する：

- “Diagrams”ペインから、編集したいメソッドの名前を選択します。
- **Diagram → Edit** を選択します。

または

- メソッドをダブルクリックします。

  選択されたメソッド用のインターフェースエディタが開きます。このダイアログボックスで、メソッドの引数と戻り値を割り当てたり、ローカル変数を定義することができます。



- 以下に説明されている手順に従い、メソッドの引数、戻り値、およびローカル変数を定義します。
- **OK** をクリックしてインターフェースエディタを閉じます。

メソッドに引数を追加する：

- インターフェースエディタの“Arguments”タブを選択します。
- **Add** ボタンをクリックします。

または

- **Argument → Add** を選択します。

  新しい引数が追加され、その名前が“Arguments”リスト上で強調表示されて編集モードになります。
- 引数名を入力して **<Enter>** を押します。
- “Argument Type”コンボボックスで引数の型を指定します。

  デフォルトの型は `cont` です。
- 引数の型として配列（`array[...]`）またはマトリックス（`mat[...]`）を選択した場合は、メニューから **Argument → Edit Max Size** を選択します。

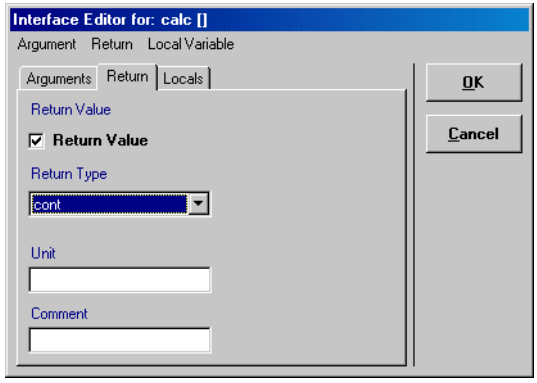
- “Max Size for: <argument>”ダイアログボックスで、配列またはマトリックスの最大サイズを入力します。
- 引数の単位を“Unit”ボックスに入力します。

  単位は純粋に表記上のもので、クラスの機能には影響しません。
- 引数についてのコメントを“Comment”ボックスに入力します。

  すべての引数や戻り値にコメントを付けることができます。このコメントは、コンポーネントについて作成されるドキュメントに出力されます。

メソッドに戻り値を追加する：

- インターフェースエディタの“Return”タブを選択します。



- メソッドに戻り値を定義する場合には、**Return Value** ボックスにチェックマークを付けます。
- 戻り値の型を指定します。
- 配列（`array[...]`）またはマトリックス（`mat[...]`）を選択した場合は、メニューから **Return → Edit Max Size** を選択します。
- “Max Size for: <return>”ダイアログボックスで、配列またはマトリックスの最大サイズを入力します。
- コメントと単位を入力します。

メソッドにローカル変数を追加する：

- インターフェースエディタの“Locals”タブを選択します。

  “Locals”タブに表示される情報は、“Arguments”タブと同じです。
- **Add** ボタンをクリックします。

または

- **Local Variable → Add** を選択します。

  新しいローカル変数が“Local Variables”リストに追加されます。
- ローカル変数の名前を入力します。
- ローカル変数の型を指定します。
- 配列（`array[...]`）またはマトリックス（`mat[...]`）を選択した場合は、メニューから **Local Variable → Edit Max Size** を選択します。
- “Max Size for: <local>”ダイアログボックスで、配列またはマトリックスの最大サイズを入力します。
- ローカル変数のコメントと単位を入力します。

引数とローカル変数を編集する：

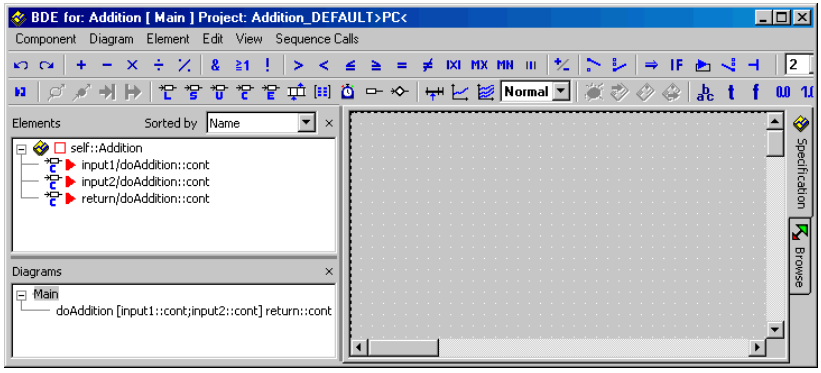
- 引数（またはローカル変数）の名前を変更するには **Argument → Rename** を選択します。
- リストに表示された引数の位置を上下に移動するには、**Argument → Move Up** または **Move Down** を選択します。
- 引数を削除するには、**Argument → Delete** を選択します。

または

- **Del** ボタンをクリックします。

ローカル変数の編集は、“Locals”タブの **Local Variable** メニューを使用して、引数やローカル変数と同じ方法で行います。

引数、ローカル変数、および戻り値は、ブロックダイアグラムエディタの“Elements”リストに表示され、引数と戻り値には右向きの赤い三角のマークが付きます。



メソッドの名前を変更する／メソッドを削除する：

- “Diagrams”ペインからメソッドを選択します。
- メソッド名を変更するには、**Diagram** → **Rename** を選択します。

または

- **<F2>** キーを押します。
- メソッドを削除するには、**Diagram** → **Delete** を選択します。

または

- **<Del>** キーを押します。

注記

“Diagrams”ペインの 1 番上に表示されているダイアグラム名だけは削除できません。

デフォルトでは、ダイアグラムとメソッドは、作成された順に“Diagrams”リストに表示され、新たに作成されたダイアグラムまたはメソッドはこのリストの最後に追加されます。この順序は、任意に変更することができます。

ダイアグラムやメソッドを移動する：

- “Diagrams”リストから、表示位置を移動したいアイテムを選択します。
- アイテムを上に移動するには、**Diagram** → **Move Up** を選択します。

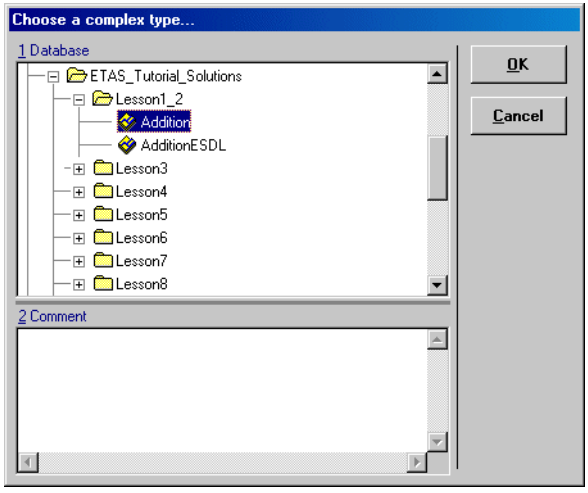
- アイテムを下に移動するには、**Diagram → Move Down** を選択します。

コンポーネントのインターフェースエレメントとして、他のコンポーネントを使用することができます。この方法について、ここでは「ユーザー定義型」の引数の場合を例にして説明します。複合型の戻り値も、同じ方法で定義できます。

コンポーネントを引数として割り当てる：

- コンポーネントを追加したいメソッドのインターフェースエディタを開きます。
- “Arguments”リストから、型を指定したいアイテムを選択（または、新しいアイテムを追加）します。
- “Argument Type”リストから、`<user defined>` を選択します。

  データベースに含まれるコンポーネントを選択するためのダイアログボックスが開きます。



- “1 Database”リストからユーザー定義型として使用するコンポーネントを選択し、**OK** をクリックしてダイアログボックスを閉じます。

  選択されたコンポーネントが、インターフェースエディタの“Argument Type”ペインに表示されます。
- **OK** をクリックして変更内容を確定します。

モジュール

モジュールのインターフェースには**プロセス**が使用されます。新たに作成されるモジュールにはデフォルトプロセスが 1 つ自動的に作成されます。プロセスはモジュールに含まれる各機能の実行順序を規定するもので、これには入出力は定義されません。モジュールはメッセージやグローバルエレメント（インポート／エクスポートエレメント）を用いて通信や相互作用を行います。

プロセスを作成する：

- **Diagram → Add Process** を選択します。

  新しいプロセスが作成され、その名前が“Diagrams”ペインに強調表示され、直接編集できる状態になります。
- プロセス名を入力してから **<Enter>** を押します。

  プロセスの場合も、モジュールと同様にリスト内の位置を移動（194 ページ参照）したり削除（194 ページ参照）することができます。

プロセスは引数を持たないので、インターフェースエディタには“Locals”タブと **Local Variable** メニュー（193 ページ参照）しか表示されません。

4.1.3 複合型のインターフェースエレメント

この項では、複合型およびユーザー定義型のエレメントをインターフェースエレメントとして使用する方法を、例に沿って説明します。

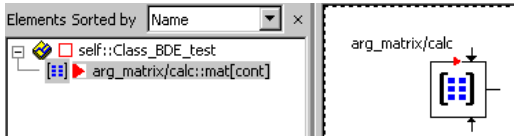
配列とマトリックス

複合型またはユーザー定義型のエレメント（配列、マトリックス、クラス）を使用する際は、通常のエレメントピン経由でアクセスできますが、それ以外に Get ポートと Set ポート経由でそのエレメント内のデータ構造体に直接アクセスすることが可能です。以下に、マトリックスを例にとってこの手順を説明します。

マトリックスの引数を作成する：

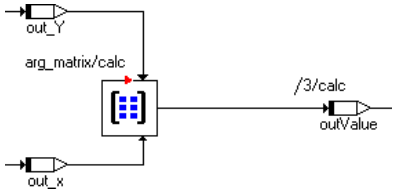
- メソッド内に `mat(cont)` 型の引数を作成します（191 ページ参照）。

- その引数を描画領域に配置します。



この時点では、エレメントには読み取り用ピンしか表示されていないため（『ASCET リファレンスガイド』の「配列とマトリックス」の項を参照してください）、マトリックスに書き込みを行うことはできません。

- 以下の図のようにピンを接続して出力を戻り値に接続します。

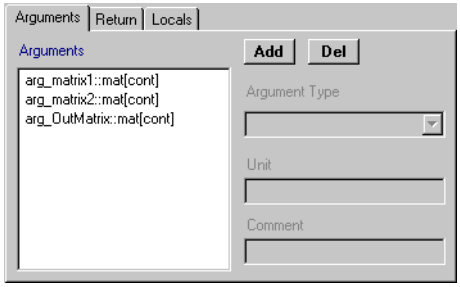


配列やマトリックスを戻り値として使用する方法はこれより複雑です。以下に、2つのマトリックスの加算処理を例にあげて説明します。

演算クラスを作成する：

- マトリックスの演算を行うメソッドを含むクラスを作成します。
- メソッド内に、加算される 2 つのマトリックス用に `mat(cont)` 型の引数を 2 つ作成します。

- 演算結果を格納するマトリックスとして、同じ型の引数（例、`arg_OutMatrix`）をさらに加えます。



引数にデータを書き込むことはできないので、このマトリックスが必要になります。

- **Matrix** ボタンで `cont` 型のマトリックスを作成します。
- “Max Size for”ダイアログボックスでプリセット値を確定します。

  3 番目の引数は、Get ポート経由で補助用マトリックスに代入されます。そのためサイズは自動的に決定されるので、明示的に指定する必要はありません。

注記

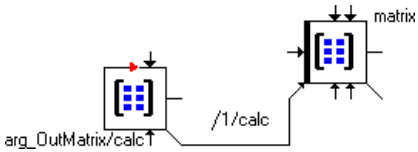
この引数には特定のサイズを指定していないので、必ずプリセット値をそのまま使用してください。これを変更すると、コード生成時にエラーメッセージが出力されます。

配列の加算処理を記述する：

- 補助用マトリックスと引数`arg_OutMatrix`を描画領域内に配置します。
- 各エレメントを右クリックし、ショートカットメニューから **Get/Set Ports** を選択します。

  引数の Get ポートと補助用マトリックスの Get ／ Set ポートが、ダイアグラム上に表示されます。

- 引数 `arg_OutMatrix` を、Set ポート経由で補助用マトリックスに代入します。

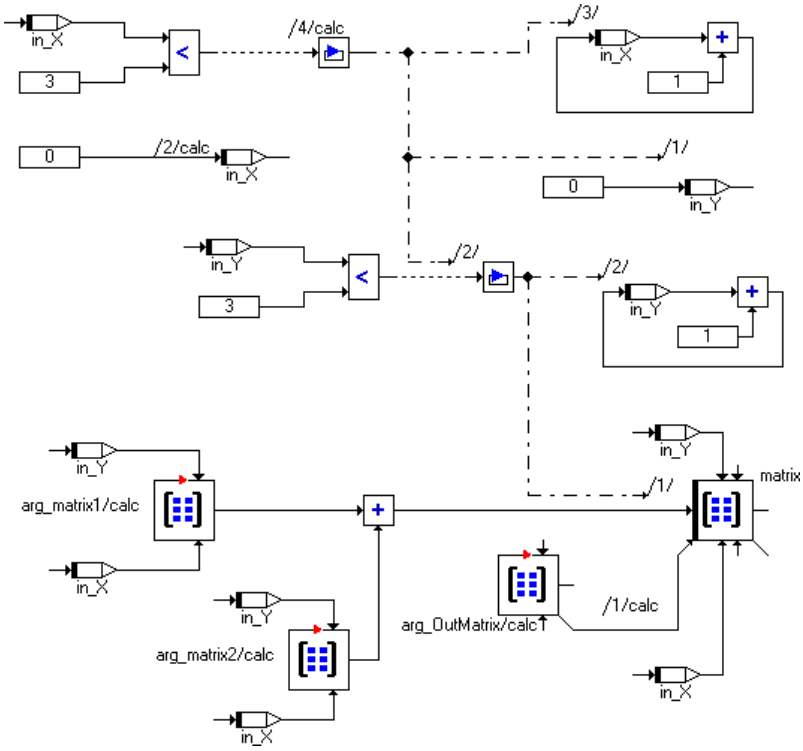


- シーケンスコールのシーケンス番号を 1 に設定し、メソッドの最初のステップでこの代入が行われるようにします。

  （シーケンスコールについての詳細は、232 ページの「シーケンスコール」を参照してください。）

  補助用マトリックスは、`arg_OutMatrix` のメモリ領域に直接アクセスするため、演算結果はメソッド外部でも利用可能となります。

- 必要なインデックスを作成し、マトリックスの加算処理を記述します。



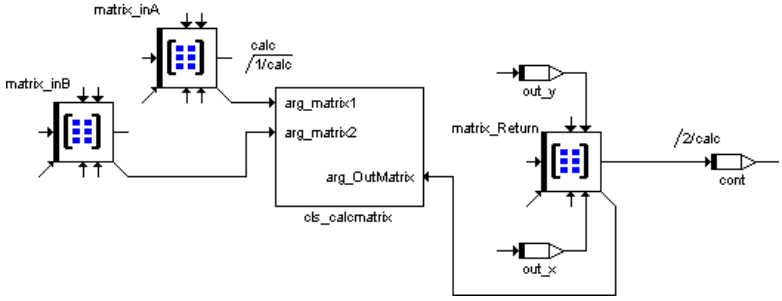
この例では、3 × 3 のマトリックスを使用しています。実際の条件に合わせてループ回数を設定してください。ただし、引数と補助用マトリックスのサイズは変更しないでください。

実際に演算を行うためには、演算クラスの引数に値を代入するもうひとつのクラスまたはメソッドが必要です。

演算を実行する：

- クラスまたはモジュールを 1 つ作成します。
- **Elements → Add** で、演算クラスを複合エレメント（215 ページ参照）として組み込みます。
- cont 型のマトリックスを 2 つ作成し、入力データを割り当てます。
- 演算結果用として、同じ型で同じサイズのマトリックスをもう 1 つ作成します。

- 描画領域に各エレメントを配置し、以下のように接続します。



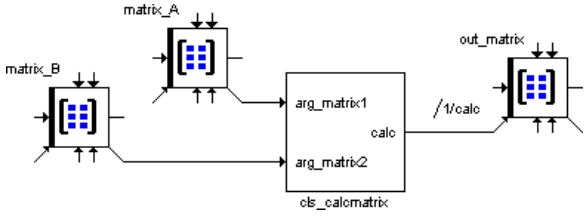
- 演算結果のマトリックスの内容を読み取るには、出力ピンを使用します。

注記

“Max size”ダイアログボックスにおいて、プリセットされた値以外のサイズをマトリックスに割り当てると、コード生成時に以下のようなワーニングメッセージが出力されます。

```
type mismatch in array max size: <1@1> and <x@y> -
accepted since larger matches smaller
```

ここまでの例でマトリックスや配列を戻り値として戻すために行ったアプローチは、メソッドに適した型の戻り値を追加して Get ／ Set ポート経由でその戻り値にアクセスする、という方法でした。以下の図を見ると、`calc_matrix` クラスは、2 つのマトリックス引数と 1 つのマトリックス戻り値を持つ calc メソッドを含んでいます。



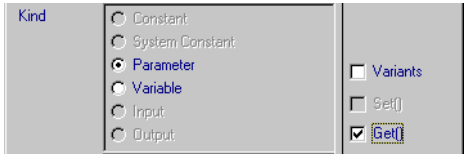
しかしこれは正しく機能しません。上の例でポートが転送するものは、メソッド内の構造体へのポインタですが、この構造体はメソッドの処理中においてのみ有効なので、`out_matrix` への代入を行う時点では、すでにデータは無効となっています。

特性カーブやマップを引数として直接渡すことはできません。これを行うためには、クラスに特性カーブ／マップを持たせ、そのクラスを引数として使用します。以下の手順は、特性カーブの例です。

特性カーブの準備：

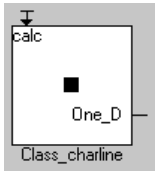
- クラス（例、`Class_charline`）を作成します。
- 
  One D Table Parameterボタンで特性カーブを作成してセットアップします（208 ページ参照）。

  この際、必ず特性カーブがクラスの外側からアクセスできるようにしてください。これは、以下の手順で行います。
- 特性カーブをエレメントエディタで開きます（440 ページの「エレメントの設定」も参考にしてください）。
- **Get()** オプションをオンにして、このカーブに出力を追加します。



- エレメントエディタを閉じます。

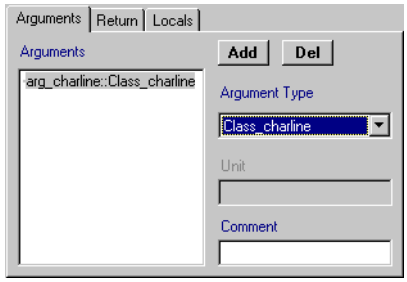
  クラスレイアウト上に、特性カーブの出力が追加されました。



特性カーブをメソッド引数として渡す：

- メソッド引数として、特性カーブを持つクラスを作成します。

- 195 ページに説明されている方法で、メソッドの 1 つに、<user-defined> 型の引数としてクラス Class_charline を追加します。



- **OK** をクリックしてインターフェースエディタを閉じます。

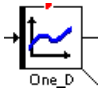
特性カーブを実際にメソッド引数として使用するには、以下のようにします。

特性カーブをメソッド引数として使用する：

- 複合引数を描画領域に配置します。

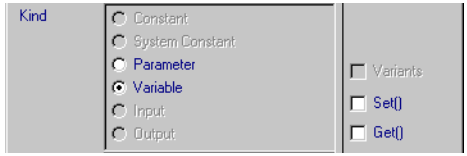
  マトリックスや配列の引数と異なり、特性カーブのピンにはアクセスできません。このため、ローカルの特性カーブに引き渡す必要があります。

- 特性カーブを作成してセットアップ（208 ページ参照）し、描画領域に配置します。
- 特性カーブを右クリックし、ショートカットメニューから **Get/Set Ports** を選択します。



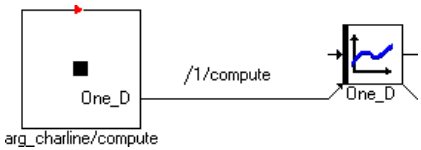
  特性カーブは「パラメータ」（適合値）として作成されるので、プログラム内部からは書き込みできず、Get ポートのみが使用可能です。

- 特性カーブをエレメントエディタで開き、種類（“Kind”）を **Variable** に設定してください（440 ページの「エレメントの設定」も参考にしてください）。



これで Set ポートも使用できるようになり、複合型の引数を特性カーブに代入することが可能となります。

- 下図のように、引数を特性カーブの Set ポートに接続します。



- シーケンスコールのシーケンス番号を 1 にして、メソッドの最初のステップでこの代入が行われるようにします。

注記

メソッドの最初のステップでこの代入が行われないと、データの矛盾が発生します。

これで、特性カーブを読み込んで処理できるようになりました。データは、引数として渡される特性カーブが位置しているメモリ領域から読み込まれます。

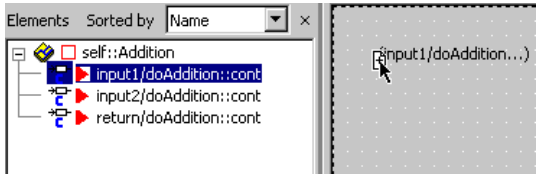
4.1.4 ブロックダイアグラムの作成

コンポーネントのインターフェースを定義すると、引数と戻り値、あるいは入力と出力がブロックダイアグラムエディタの“Elements”ペインに表示されます。これらを描画領域に配置して他の変数や演算子などのグラフィックエレメントと接続します。

エレメント／演算子／接続線

インターフェースエレメントを配置する：

- “Elements”ペインから、任意のエレメントを選択して描画領域内の任意の位置にドラッグします。



エレメントが描画領域に配置されます

- 描画領域内のエレメントを選択してドラッグし、別の位置に移動することもできます。

基本エレメントを作成する：

- 作成したいエレメントの型のボタンをクリックし、マウスカーソルにその型のエレメントをロードします。

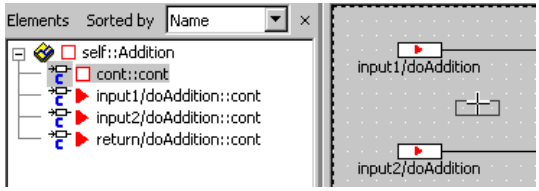
  エレメントのボタンは、ツールバーの 2 行目にあります。

  自動的にエレメントエディタが開くので、そこでエレメントのプロパティを設定できます。

注記

エレメント作成時にエレメントエディタが自動的に開かないようにするには、エディタの下部にある **Always show Edit Dialog for new elements** オプションをオフにするか、または ASCET オプションウィンドウの“Confirmation Dialogs”ノード（46 ページ参照）で **Edit primitive Element** をオフにしてください。

- 描画領域内をクリックして、新しいアイテムを配置します。



  クリックした位置にエレメントが配置されます。これをドラッグして別の位置に移動することができます。

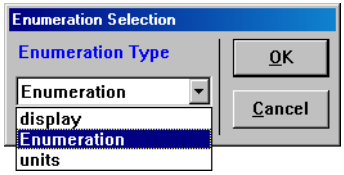
  “Elements”リストに新しいエレメントが追加されます。

エレメントを右クリックしてショートカットメニューを開くといくつかのコマンドが表示され、それらを使用してエレメントのプロパティを変更することができます。これらのコマンドについては、439 ページの「エレメントプロパティの編集」という項で説明します。

列挙型データを作成する：

- **Enumeration** ボタンをクリックします。

  “Enumeration Selection”ダイアログボックスが開きます。



  “Enumeration Type”コンボボックスに、データベースに定義されているすべての列挙型が表示されます。

- コンボボックスから、使用する列挙型を選択します。
- **OK** をクリックしてダイアログボックスを閉じます。

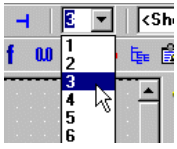
  マウスカーソルに列挙型データがロードされ、エレメントエディタが開きます。

- 必要に応じてエレメントのプロパティを変更し、**OK** をクリックします。
- 描画領域内をクリックします。

  列挙型データが配置され、“Elements”リストにもそのデータがエレメントとして追加されます。

- エレメント名を入力して **<Enter>** を押します。

演算子を配置する：



- “Argument Size”コンボボックスから、演算子の入力の数を選択します。

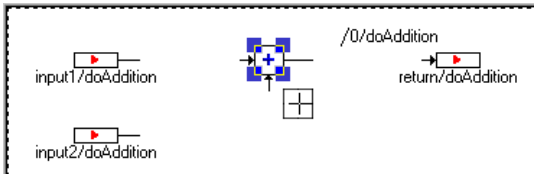
注記

演算子の種類によっては入力の数は変更することができないので（除算、減算、比較等）、数を選択しても実際には無視されます。

- ツールバーから、作成したい演算子を選択してクリックし、マウスカーソルにその演算子をロードします。

- 描画領域内をクリックして、演算子を配置します。

  演算子がダイアグラムに追加されます。これを別の位置にドラッグすることもできます。



ダイアグラム内のデータの流れは、描画領域内のアイテムを接続することによって定義されます。

ダイアグラム上のアイテムを接続する：

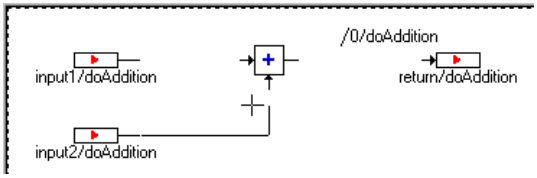
- 描画領域内の、エレメントが表示されていない位置を右クリックします。

または

- **Connect** ボタンをクリックします。

  これで、接続モードが開始されます。接続モードにおいては、カーソルが十字カーソルになります。
- 接続したい最初のダイアグラムエレメントのピンをクリックして、接続を開始します。
- カーソルを接続の終了点まで移動します。

  カーソルの軌跡に従って線が描かれます。描画中にエレメントが表示されていない位置をクリックすると、そこが接続線の折点として固定され、右クリックすると、最後に作成された固定点が削除されます。



- 最後に接続先のダイアグラムアイテムのピンをクリックして、接続を完成させます。

  接続が確立され、接続線が描かれます。

描かれる接続線は、接続されるエレメントの種類によって異なります。

線の色／スタイル	接続されるエレメント
黒／実線	2 つの数値ピン間の接続
黒／破線	2 つの論理ピン間の接続
黒／ 1 点破線	処理フローの接続（212 ページ参照）
指定色 （デフォルト：緑）	コメントライン、つまりまだシーケンシングが行われていない制御文や演算子についての接続（4.1.6 項参照）。 コメントラインの色は、ASCET ユーザーオプションで選択できます（55 ページ「“Colors” ノード」を参照してください）。
赤	無効な接続（例：数値ピンと論理ピン間の接続）

エレメントをドラッグすると、接続線も一緒に移動し、場合によってはダイアグラムが煩雑になってしまう場合もあります。そのような場合には、接続線をドラッグして経路を任意に変更することができます。

接続モードを終了する：

- 接続モードを終了するには、描画領域内の、エレメントが何も表示されていない位置をもう一度右クリックします。

または

- **Connect** ボタンをクリックします。

  接続モードが終了します。

配列／マトリックス／特性カーブ／特性マップ

ここでは配列、マトリックス、特性カーブ／マップの作成方法について説明します。これらのエレメントの値の編集方法については、449 ページの「データの編集」の項を参照してください。

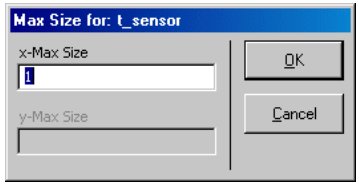
配列／マトリックスを作成する：

- **Array** または **Matrix** ボタンをクリックします。

  エレメントエディタが開きます。配列とマトリックスは変数として作成されるため、**Volatile** 属性が割り当てられます。

- 必要に応じてエレメントのプロパティを変更し、**OK** で確定します。

  “Max size”ダイアログボックスが開きます。



- "x-Max Size" および "y-Max Size" フィールドで、各座標ポイントの最大数を指定します。

  "y-Max Size" フィールドは、マトリックスの場合のみ入力可能です。

- **OK** をクリックします。

  入力した値が最大サイズを超えていると、以下のワーニングメッセージが表示されます。

  `*-Max size exceeded. Reduce to limit <limit>.`

  — **OK** をクリックして、提示されたサイズに変更します。

  または

  — **Cancel** をクリックして“Max size”ダイアログボックスに戻ります。

- 作成したエレメントを描画領域に配置します。

通常の特性カーブ／マップまたは固定カーブ／マップを作成する：

- ツールバーのコンボボックスで、カーブ／マップタイプとして `Normal` または `Fixed` を選択します。
- **One D Table Parameter** または **Two D Table Parameter** ボタンをクリックします。

  エレメントエディタが開きます。カーブとマップはパラメータとして作成されるため、**Non-volatile** 属性が割り当てられ、“Memory”フィールドは無効となります。

- 必要に応じてエレメントのプロパティを変更し、**OK** で確定します。

  “Max size”ダイアログボックスが開きます。

- "x-Max Size" および "y-Max Size" フィールドで、各座標ポイントの最大数を指定します。

注記

固定カーブ／マップには、各軸について 2 つ以上の座標ポイントが必要です。1 つしかない場合は、値の連続性チェック（monotony check）においてエラーが発生します。

"y-Max Size" フィールドは、特性マップの場合のみ入力可能です。

- **OK** をクリックします。

入力した値が最大サイズを超えていると、以下のワーニングメッセージが表示されます。

`*-Max size exceeded. Reduce to limit <limit>.`

— **OK** をクリックして、提示されたサイズに変更します。

または

— **Cancel** をクリックして“Max size”ダイアログボックスに戻ります。

- 作成したエレメントを描画領域に配置します。

ディストリビューションを作成する：

グループカーブ／マップ（456 ページ参照）を使用するには、それに対応するディストリビューションを作成する必要があります。

- **Distribution** ボタンをクリックします。

エレメントエディタが開きます。ディストリビューションはパラメータとして作成されるため、**Non-volatile** 属性が割り当てられ、“Memory”フィールドは無効となります。

- 必要に応じてエレメントのプロパティを変更します。
- **OK** をクリックします。

“Max size”ダイアログボックスが開きます。

- “x-Max Size”フィールドで、座標ポイントの最大数を指定します。

“y-Max Size”フィールドは必要ありません。

"y-Max Size" フィールドは、特性マップの場合のみ入力可能です。

- **OK** をクリックします。

  入力した値が最大サイズを超えていると、以下のワーニングメッセージが表示されます。

  `x-Max size exceeded. Reduce to limit 2048.`

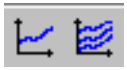
  — **OK** をクリックして、提示された最大数に変更します。

  または

  — **Cancel** をクリックして“Max size”ダイアログボックスに戻ります。
- 作成したエレメントを描画領域に配置します。

グループカーブ／マップを作成する：

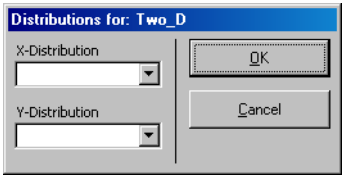
- ツールバーのコンボボックスで、カーブ／マップタイプとして `Group` を選択します。



- **One D Table Parameter** または **Two D Table Parameter** ボタンをクリックします。

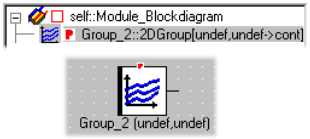
  エレメントエディタが開きます。カーブとマップはパラメータとして作成されるため、**Non-volatile** 属性が割り当てられ、“Memory”フィールドは無効となります。
- 必要に応じてエレメントのプロパティを変更し、**OK** で確定します。

  “Distribution for”ダイアログボックスが開きます。



- “x-Distribution”および“y-Distribution”コンボボックスで、使用するディストリビューションを選択します。

  “y-Distribution”コンボボックスは、グループマップの場合のみ入力可能です。



- **OK** をクリックします。

Cancel をクリックすると、ディストリビューションは割り当てられずにマップが作成され、マップの名前に含まれるディストリビューション名は undef となります。

このマップを実際に使用するには、**Element → Edit Distribution** でディストリビューションを選択する必要があります。

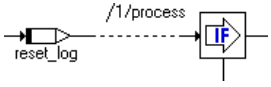
- 作成したエレメントを描画領域に配置します。

フロー制御文

If 文を使用する：

- **If then** または **If then else** ボタンをクリックして、マウスカーソルにいずれかのブロックをロードします。
- そのブロックを描画領域に配置します。

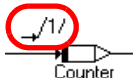
If...Then ブロックと If...Then...Else ブロックは似ていますが、後者には、枝分かれする処理フローを表わす 2 本の枝が付きます。



- ブロックの入力に論理エレメントを接続します。

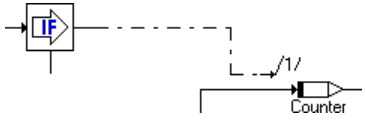
以下のようにして各枝のアクションを定義します。

- いずれかの枝に接続したいシーケンスコールを右クリックして、ショートカットメニューから **Connector** を選択します。



選択したシーケンスコールがコネクタに変わり、シーケンシングが未解決であることを表わす接続線（オプション設定された色で表示されます）が、黒に変わります。

- 枝をそのコネクタに接続します。



1 つの枝を複数のアクションに接続することができます。その際は、コネクタにユニークな番号をシーケンスコールとして割り当てる必要があります（233 ページを参照してください）。

- If...Then...Else ブロックの場合は 2 本目の枝に対しても同様の手順でアクションを指定します。

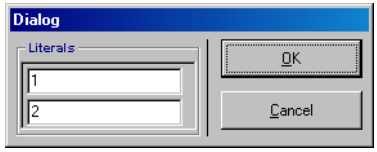
Switch 文を使用する：

- “Argument Size” コンボボックスでスイッチの枝の数を指定します。

  枝の数は 2 本以上を指定してください。default の枝はこの数には含まれません。

- **Switch** ボタンをクリックして、マウスカーソルに Switch ブロックをロードします。
- そのブロックを描画領域に配置します。
- 枝の数を設定するには、ブロックを右クリックしてショートカットメニューを開き、**Edit Literals** を選択します。

  エディタダイアログボックスが開きます。それぞれの枝に対応する入力フィールドが含まれます。



  - “Literals” フィールドに、各枝の値を入力します。
  - **OK** をクリックして確定します。
- `sdisc` または `udisc` 型のエレメントを、Switch ブロックの上部にある入力に接続します。

注記

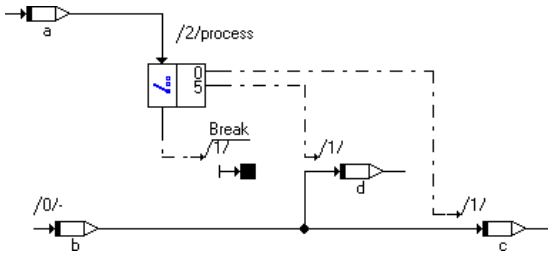
`cont` ピンを Switch の入力に接続すると、コード生成時にエラーが発生します。

- 各枝のアクションを定義します。
  - 枝に接続したいシーケンスコールを右クリックして、ショートカットメニューから **Connector** を選択します。

    選択したシーケンスコールがコネクタに変わります。
  - 枝をそのコネクタに接続します。

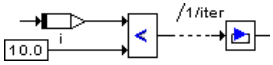
    1 つの枝を複数のアクションに接続することができます。その際は、コネクタにユニークな番号をシーケンスコールとして割り当てる必要があります（233 ページを参照してください）。

- 他の枝に対しても同様の手順でアクションを指定します。



ブロックダイアグラム内で使用できるループ制御は、`While` ループのみです。

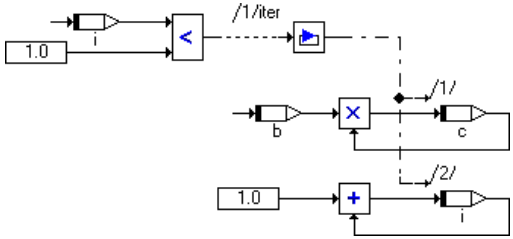
While ループを使用する：



- **While** ボタンをクリックして、マウスカーソルに While ブロックをロードします。
- そのブロックを描画領域に配置します。
- 以下のようにしてループ条件を定義します。
  - ループの入力に条件を接続します。
- 以下のようにしてループアクションを定義します。
  - いずれかの枝に接続したいシーケンスコールを右クリックして、ショートカットメニューから **Connector** を選択します。

    選択したシーケンスコールがコネクタに変わります。

– ループの出力をそのコネクタに接続します。



出力は複数のアクションに接続することができます。その際は、コネクタにユニークな番号をシーケンスコールとして割り当てる必要があります（233 ページを参照してください）。

注記

無限ループや、リアルタイムアプリケーションに適さないようなループは絶対に使用しないでください。

複合エレメントとしてのコンポーネント

コンポーネントは、「複合エレメント」として他のコンポーネント内に含めることができます。「基本エレメント」はそのエレメントが内包されるコンポーネント内で定義されたものですが、「複合エレメント」は参照により内包されます。そのため、複合エレメントとして内包されたコンポーネント自体に変更が加わると、その変更内容は、コンポーネントを内包するコンポーネント内においても有効になります。内包されるコンポーネントに対しては、他のダイアグラムアイテムをその入力や出力に接続することができます。

コンポーネントは以下のようにして組み込みます。

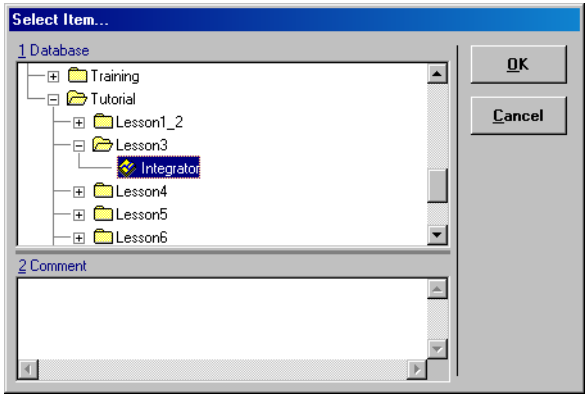
コンポーネントを複合エレメントとして他のコンポーネントに組み込む（内包させる）：

- コンポーネントを“Elements”リストに追加するために、**Element → Add Item** を選択します。

または

- “Elements”リストのショートカットメニューから **Add Item** を選択します。

  “Select Item”ダイアログボックスが開き、現在のデータベースに含まれるアイテムが表示されます。



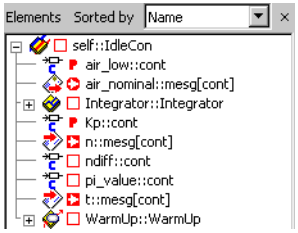
- “1 Database”リストから、使用したいコンポーネントを選択します。
- **OK** をクリックして、コンポーネントを追加します。

  コンポーネントが“Elements”リストに追加され、そのインスタンス名が強調表示されて直接編集できる状態になります。
- インスタンス名を編集してから **<Enter>** を押します。
- コンポーネントを、描画領域にドラッグしてダイアグラムに追加します。
- コンポーネントのピンを、他のダイアグラムアイテムのピンと同じ要領で接続します。

注記

データベースアイテムの追加は、ここで説明した方法以外に、アイテムをコンポーネントマネージャからブロックダイアグラムエディタに直接ドラッグ & ドロップすることもできます。

コンポーネントを組み込むと、そのインスタンス名が“Elements”ペインに表示されます。インスタンス名の先頭には“＋”シンボル（⊞）が付き、さらにツリーを展開できることを示しています。



同じコンポーネントをさらにもう 1 つ組み込むと、そのコンポーネントの新しいインスタンスが作成され、ユニークなインスタンス名が割り当てられます。ダイアグラム内では、インスタンス名はグラフィックブロックの下に表示されます。“Elements”リスト内には、エレメント名としてインスタンス名とクラス名の両方が `<instanceName>::<className>` の形式で表示されます。名前全体が見えない場合は、リストの幅を調節してください。

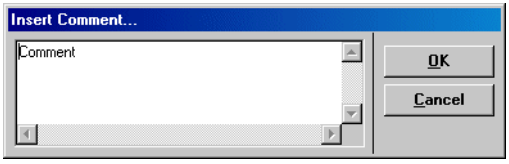
コメントと注釈

コメントと注釈は、コンポーネントの機能には影響しません。ソフトウェアモデルを文書化する際に説明文として使用されます。

コメントを追加する：

- **Comment** ボタンをクリックします。

  テキストボックスが開きます。



- コメントのテキストを入力します。
- **OK** をクリックします。

  マウスカーソルにコメントがロードされます。
- 描画領域内をクリックして、コメントを配置します。

配置されたコメントは、任意の位置にドラッグすることができます。編集を行う際は、コメントを右クリックしてショートカットメニューを開き、**Edit Commen** を選択します。

また、コンポーネント全体、または内包されるコンポーネントに注釈を付け加えることができます。注釈は、専用のエディタで入力します。この注釈は、自動生成されるドキュメントに出力されます。

コンポーネント用の注釈を編集する：

- 現在開いているコンポーネント全体の注釈を編集するには、**Component → Notes** を選択します。

または

- エレメントとして使用されているコンポーネントの注釈を編集するには、“Elements” ペインまたは描画領域でコンポーネントを選択し、そのコンポーネントのショートカットメニューから **Notes** を選択します。

  注釈エディタが開きます。このエディタについての詳しい説明は、7.4「注釈」の項を参照してください。

4.1.5 ブロックダイアグラムの編集

本項では、描画領域における一般的な編集機能と、ダイアグラムアイテムの表示形式を変更する機能について説明します。

ダイアグラム上で実行された操作は、ASCET 内部のクリップボードにステップ単位で記録されているため、直前に行われた操作を調べたり、前の状態に戻って作業し直すことができます。ただし前の状態に戻ることができるのは 1 つのエディタが開いている間だけです。エディタを終了すると、その時点の状態でダイアグラムが保存され、クリップボードに記録されていたそれ以前の状態はすべて削除されます。

最後の操作を取り消す：

- **View → Undo** を選択します。

または

- **Undo** ボタンをクリックします。

または

- **<Ctrl> + <Z>** キーを押します

  最後に行った操作が取り消され、その操作を行う前の状態に戻ります。

注記

前に戻った状態で作業を続行すると、その後に行われていた操作はすべて削除され、代わりに新しく実行される操作が記録されます。

Undo コマンドを取り消す：

- **View → Redo** を選択します。

または

- **Redo** ボタンをクリックします。

または

- **<Ctrl> + <Y>** キーを押します

  最後に取り消された操作が再度実行されます。

エレメントビュー

ブロックダイアグラムエディタには、個々のエレメントについての情報が一目で確認できる便利な機能が用意されています。

エレメントのすべてのオカレンスを表示する：

- 描画領域または“Elements”ペインでエレメントを選択します。
- **Element → Show Occurrences** を選択します。

または

- ショートカットメニューから **Show Occurences** を選択します。

  現在のプログラム内の、このエレメントのすべての「オカレンス」（ダイアグラム上に配置されたダイアグラムアイテム）が強調表示されます。

ブロックダイアグラムを表示したまま、つまりブラウザビューに切り替えることなく、個々のグラフィックオブジェクト（エレメント、演算子、接続線）に接続されているエレメントを一覧表示して、インプリメンテーションを直接編集することができます。この機能は、特に演算子の入出力のインプリメンテーションのテストを行う際に役立ちます。以下にその手順を詳しく説明します。

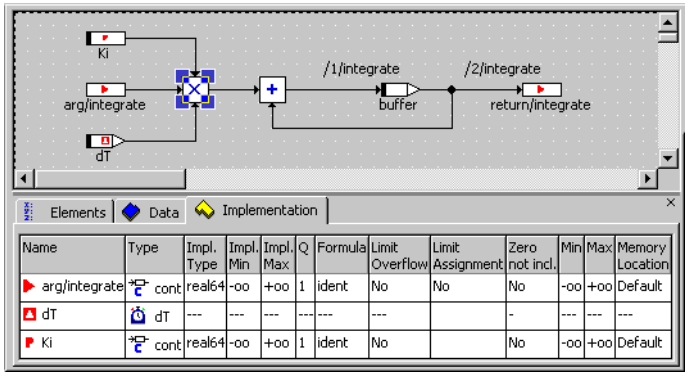
1 つのエレメントに接続されているすべてのエレメントを一覧表示する：

- テストを行いたいダイアグラムアイテムを右クリックし、ショートカットメニューから **Browse Connected Elements** を選択します。

  次の図のように、ブロックダイアグラムの下に、“Elements”、“Data”、“Implementation”という 3 つのタブが含まれる [エレメントビュー」

フィールドが開き、このダイアグラムアイテムに直接接続されているすべてのエレメントが一覧表示されます。

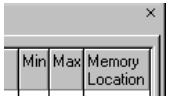
選択されたダイアグラムアイテムがエレメントであった場合は、そのエレメント自身もこのリスト内に表示されます。演算子はリストには含まれません。



各タブの内容は、コンポーネントマネージャのエレメントビューと同じです（2.1.3 項を参照してください）。デフォルトでは“Implementation”タブが表示されます。

- 各タブの使用方法は、コンポーネントマネージャの場合と同じです（2.3.2 項を参照してください）。

エレメントの行をダブルクリックすると、所定のエディタが開きます。



- このフィールドを閉じるには、フィールドの右上端の✕シンボルをクリックします。

または

- **View → Show/Hide → Connected Elements** を選択します。
- もう 1 度 **View → Show/Hide → Connected Elements** を選択すると、このフィールドが再度開きますが、このメニューコマンドで開いた場合は、オブジェクトのショートカットメニューで開いた場合と異なり、現在どのオブジェクトが選択されているかが反映されず、前回フィールドを閉じた時の内容がそのまま表示されます。

エレメントの名前を変更する／エレメントを削除する：

- エレメント名を変更するには、**Element → Rename** を選択します。
- エレメントを削除するには、**Element → Delete** を選択します。

  エレメントを削除する際は、まずそのオカレンスをダイアグラムから削除しておく必要があります。
  Show Occurrences コマンドを実行すると、ダイアグラム内にあるエレメントのオカレンスがすべて選択されるので、それらを同時に削除することができます。
- **Component → Delete Unused Elements** を選択すると、“Elements”ペインに表示されていてもダイアグラムには使用されていないすべてのエレメントが削除されます。

ダイアグラムアイテムを置き換える：

- 演算子を置き換える場合、ツールバーから演算子を選択して、マウスカーソルに新しい演算子をロードします。
- ダイアグラム内の、新しい演算子に置き換えたい既存の演算子をクリックします。

  確認のダイアログボックスが開きます。
- **Yes** をクリックすると、演算子が置き換わります。

上記の演算子の場合と同様に、エレメントを置き換えることもできます。エレメントの場合は、“Elements”リスト内のエレメントを置き換えたい既存のダイアグラムアイテム上にドラッグすることもできます。また、ダイアグラム内の演算子をエレメントに置き換えたり、エレメントを演算子に置き換えることもできます。

ダイアグラムアイテムのカット／コピー／ペーストを行う：

- カットまたはコピーしたいダイアグラムアイテムをクリックします。

  選択されたアイテムにハンドル（アイテムの四隅に表示される青い四角形のシンボル）が付きます。
- 選択されたダイアグラムアイテムを切り取ってクリップボードに移動するには、**Edit → Cut** を選択します。

または

- **<Ctrl> + <X>** キーを押します。
- 選択されたダイアグラムアイテムをクリップボードにコピーするには、**Edit → Copy** を選択します。

または

- **<Ctrl> + <C>** キーを押します。
- クリップボード上のアイテムをダイアグラム上にコピーするには、**Edit → Paste** を選択します。

または

- **<Ctrl> + <V>** キーを押します。

  クリップボード上のアイテムが、ダイアグラム上の元の位置付近に貼り付けられるので、それを任意の位置にドラッグします。

ASCET ではグラフィック情報用に独自の内部クリップボードを使用するので、上記の操作を行っても Windows のクリップボードの内容は変わりません。また、あるコンポーネントのダイアグラムアイテムを他のコンポーネント内にコピーすることもできますが、その場合にコピーされるのはグラフィック情報だけで、インターフェースエレメントはコピーされません。

ダイアグラムアイテムを削除する：

- 描画領域から、接続線またはエレメントのオカレンスを選択します。
- **Edit → Delete** を選択します。

または

- **<Del>** キーを押します。

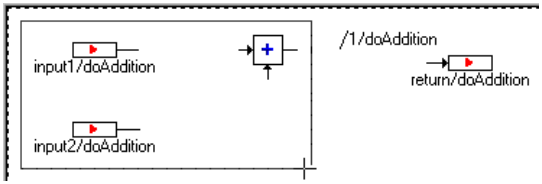
  選択されたアイテムがダイアグラムから削除されます。

複数のダイアグラムアイテムを選択する：

- **<Ctrl>** キーを押し下げた状態で、選択したいすべてのダイアグラムアイテムをクリックします。

または

- 描画領域内で、選択したいアイテムのまわりをマウスでドラッグして長方形で囲みます。



長方形で囲まれたダイアグラムアイテムが選択されて、各アイテムにハンドルが付きます。

- 1 つのアイテムについて行う場合と同様に、このグループをカット、コピー、削除、または移動することができます。

ダイアグラムアイテムの表示形式

ダイアグラムアイテムの表示形式を変更する：

ここで紹介されているコマンドは、描画領域内に配置されたコンポーネント（内包されたコンポーネント）のショートカットメニューからしか実行できず、その機能は、選択されているオカレンスにのみ影響します。

注記

下記の設定内容は、「内包されるコンポーネントのレイアウト」の項に説明されている「デフォルトレイアウト」には含めることができません。

- **Show/Hide Name** を選択すると、ダイアグラムアイテムのインスタンス名の表示／非表示が切り替わります。
- **Ports → Get/Set** を選択するとダイアグラムアイテムの Get/Set ピンの表示／非表示が切り替わります。

  このコマンドは、配列、マトリックス、特性カーブ／マップにも使用できます。

個々のダイアグラムアイテム（基本エレメント、内包されたコンポーネント、演算子、接続線など）について、各ビュー（7.3 項を参照してください）における表示モードを以下のようにして設定することができます。

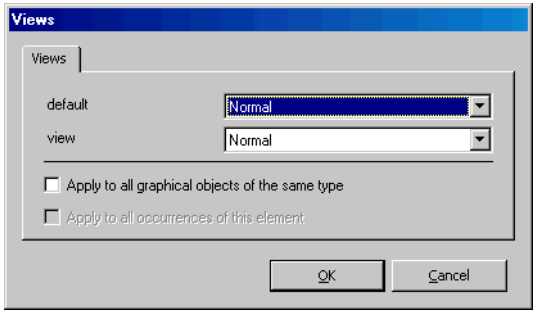
ダイアグラムアイテムの各ビューでの表示モードを編集する：

- ダイアグラム上で、ビューを編集したいダイアグラムアイテムを選択します。
- **Edit → Views** を選択します。

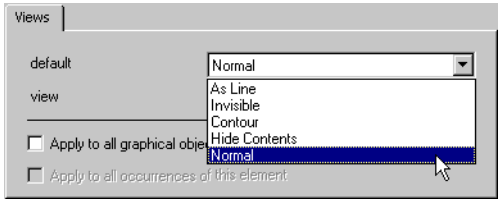
または

- エレメントを右クリックし、ショートカットメニューから **Views** を選択します。

  “Views”ダイアログボックスが開き、現在のデータベースに定義されているすべてのビューが表示されます。

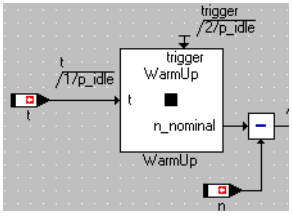


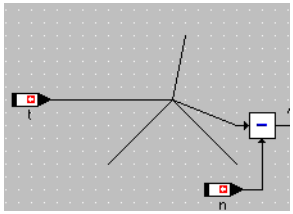
- 表示モードを変更したいビューのコンボボックスを開きます。



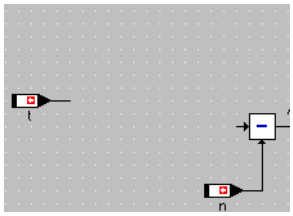
- アイテムに割り当てたい表示モードをコンボボックスから選択します。

  — `Normal`: アイテムは、すべてのグラフィックエレメント（シンボル、名前、シーケンスコールなど）を含むデフォルト形式で表示されます。

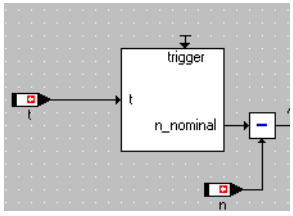




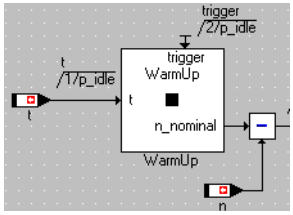
— `As Line`: アイテムの代わりに、アイテムの中心とピンを結ぶ 1 本または複数の線で表示されます。



— `Invisible`: アイテムはダイアグラム上に表示されず、マウスで選択することもできません。



— `Contour`: アイテムの本体とピン名のみが表示され、シーケンスコール、名前、シンボルなどは表示されません。



— `Hide Content`: `Normal` 設定と同じ形式で表示されますが、コンポーネントや階層を描画領域から開くことはできません。このように設定された階層はドキュメントには反映されませんが、コンポーネントの内容は出力されます。

- 選択した表示モードを、同じタイプのダイアグラムエレメントすべてに適用するには、**Apply to all graphical objects of the same type** オプションをオンにします。
- 選択した表示モードを、同じエレメントのすべてのオカレンスに適用するには、**Apply to all occurrences of this element** オプションをオンにします。

- **OK** をクリックして、選択を有効にします。

  ビューを切り替えると、そのアイテムは現在のビュー用に指定された表示モードで表示されます。

  このコマンドは、すべてのダイアグラムアイテムについて使用できます。ビューについては、7.3「ビュー」の項で説明されています。

As Line モードになっているアイテムは、そこにアイテムが存在していることがわかりづらくなる場合があります。そのため、気付かずに As Line モードになっているアイテムやその接続線をマウスで選択すると、以下のような警告メッセージが表示されます。

```
CAUTION: Element shown in "As Line" view could be
affected!
```

内包されるコンポーネントのレイアウト

コンポーネントの内部に別のコンポーネントを追加して（215 ページ参照）それを描画領域に配置すると、そのコンポーネントは、レイアウトエディタで編集されたデフォルトレイアウトで表示されます。このデフォルトレイアウトが現在のグラフィック画面に適していない場合、2 つの方法で対処することができます。

1 つめの方法は、レイアウトエディタ（500 ページ参照）でデフォルトレイアウトを変更する方法です。ただしこの場合、レイアウトエディタで編集された内容はすでにダイアグラム上に配置されている個々のオカレンスには影響しないため、新しいレイアウトを有効にするには、そのオカレンスをマニュアル操作で置き換え、あたらいレイアウトをロードする必要があります。

もう 1 つは、ブロックダイアグラム上で個々のオカレンスのレイアウトを変更する方法です。これを行うには、ASCET の“Options”ウィンドウで **Activate flexible layout** オプションをオンにします（54 ページの「ブロックダイアグラムエディタのオプション」を参照してください）。この方法では、編集されたオカレンスのレイアウトのみが変更され、その他のオカレンスやデフォルトレイアウトは変わりません。このため、同じコンポーネントが複数のオカレンスを持っている場合、個々のオカレンスを異なるレイアウトにすることが可能です。

ブロックについては、そのサイズを変更したり、移動や、ポートの表示／非表示の切り替えなどが行えます。ただし以下の点を注意してください。

- 1 つのオカレンスの最小サイズは、2 つの入力を持つ加算演算子の大きさ以上である必要があります。
- 2 つのポートを同じ位置に表示することはできないため、オカレンスの最小サイズは、現在表示されているポートの数によっても変わります。
- デフォルトレイアウトにアイコンが含まれている場合、オカレンスのサイズがアイコンのサイズよりも小さい場合、アイコンは削除されます。

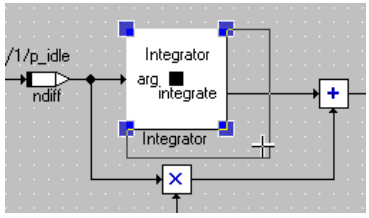
この方法で個別に編集されたオカレンスをコピーまたはカットして、別の位置に貼り付けた場合（221 ページ参照）、レイアウトも共にコピー（または移動）されます。

オカレンスのサイズは、マニュアル操作で変更したり、自動的に最小サイズにすることができます。

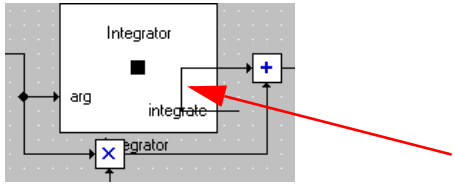
オカレンスのサイズを変更する：

- 描画領域で、編集したいオカレンスを選択します。

  オカレンスの四隅にハンドルが付きます。
- ハンドルをドラッグしてサイズを変更します。

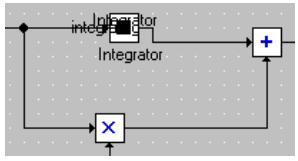


この際、他のアイテムは影響を受けません。既存の接続線の接続は保たれますが、サイズ変更によって見づらくなる場合もあります。



- 必要に応じて他のダイアグラムアイテムのサイズや位置を変更し、レイアウトを調整します。
- オカレンスを右クリックして、ショートカットメニューから **Minimal Size** を選択します。

  オカレンスのサイズは、ポートの数に応じて最小化されます。この際、ポート名とオカレンス名の表示位置は考慮されません。

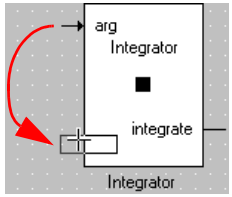


- 必要に応じて、オカレンスのショートカットメニューから **Show/Hide Name** を選択し、コンポーネント名の表示／非表示を切り替えます。

以下のようにして、オカレンスのポートを移動したり、表示／非表示を切り替えることができます。

ポートを編集する：

- ポートの位置を移動するには、マウスでポートをドラッグします。



他のポートが配置されていない場所であれば、どこにでも移動できます。

- ポートを選択してショートカットメニューから **Pin Names *<name>*** を選択すると、ポート名の表示／非表示が切り替わります。

  ポート名が表示されているときは、**Pin Names *<name>*** メニューアイテムにチェックマークが付きます。
- オカレンスのショートカットメニューから **Show Pin Names** を選択すると、そのオカレンスのすべてのポート名が表示されます。
- 同じく **Hide Pin Names** を選択すると、そのオカレンスのすべてのポート名が表示されなくなります。
- **Unconnected Ports** を選択すると、オカレンスの、現在接続されていないすべてのポートの表示／非表示が切り替わります。1 つのメソッドに使用されるポートは、すべて同時にしか表示／非表示を切り替えられません。

注記

ポートは上記のように非表示にすることはできますが、削除はできません。**Minimal Size** などのコマンドは、ポートが表示されているものとみなして処理を行います。

コンポーネントの表示形式は、ショートカットメニュー から **Port** コマンドを選択して変更します。コンポーネントのポートは、以下のように、メソッドごとに追加または削除します。

注記

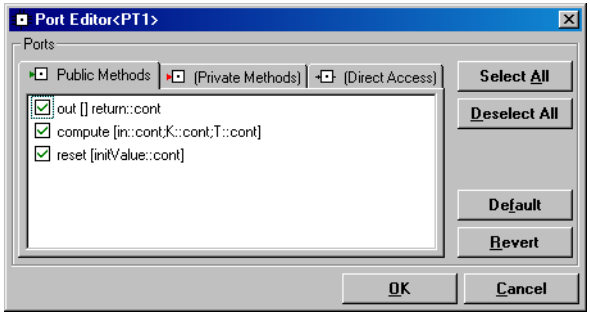
1 つのメソッドに関しては、すべてのポートが同時に追加／削除されます。

ポートの表示／非表示を切り替える：

1. パブリックメソッドの場合
   - オカレンスのショートカットメニューから **Ports → Methods** を選択します。

     ポートエディタが開いて、“Public Methods”タブが表示されます。追加されるコンポーネントのパブリックメソッドがすべて一覧表示されます。

     選択されたメソッドにはチェックマークが付きます。

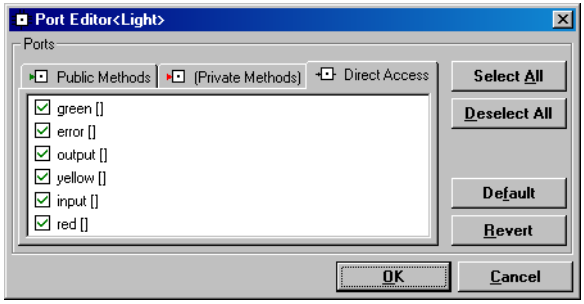


2. プライベートメソッドの場合
   - オカレンスのショートカットメニューから **Ports → Methods** を選択します。
   - “Private Methods”タブを選択します。
3. ダイレクトアクセスの場合
   - オカレンスのショートカットメニューから **Ports → Direct Access** を選択します。

     ポートエディタが開いて、“Direct Access”タブが表示されます。追加されるコンポーネントのダイレクトアクセスメソッド（ステートマシンの入出力など）がすべて一覧表示されます。

   または
   - オカレンスのショートカットメニューから **Ports → Methods** を選択します。

- “Direct Access”タブを選択します。



4. 編集の方法

- メソッドを選択するには、チェックマークが付いていないアイテムをダブルクリックします。
- すべてのメソッドを選択するには、**Select All** をクリックします。
- 選択を取り消すには、チェックマークのついたメソッドをダブルクリックします。
- すべてのメソッドを無効にするには、**Deselect All** をクリックします。
- 現在までの変更内容を取り消すには、**Revert** をクリックします。
- レイアウトエディタで定義されたレイアウトに戻すには、**Default** をクリックします。
- 各タブについて同様の設定を行います。
- **Cancel** をクリックすると、変更内容が取り消されてダイアログボックスが閉じます。
- **OK** をクリックすると、変更内容が確定されてダイアログボックスが閉じます。

チェックマークの付いたメソッド／プロセスのポートのみが表示されます。

他のエレメントに接続されているポートを削除すると、その接続線も削除されます。

新しく追加されたポートの配置は自動的に決定され、入力ポートは左側、出力ポートは右側に配置されます。スペースの足りないエレメントにポートを追加すると、そのエレメントは自動的に拡大されます。

オカレンスについて変更したレイアウトを新しいデフォルトレイアウトとして使用するには、以下のように操作します。

変更したレイアウトをデフォルトレイアウトとして使用する；

- オカレンスのショートカットメニューから **Use as Default Layout** を選択します。

  以下の確認メッセージが表示されます。

  ```
  Do you want to use the current
  layout as the default layout for all
  future occurrences of this class?
  ```

- 処理を中止するには **Cancel** をクリックします。
- 処理を実行するには **OK** をクリックします。

  変更されたレイアウトが新しいデフォルトレイアウトになります。

  同じコンポーネントのオカレンスを新たに作成すると（つまりコンポーネントを新たにダイアグラム内に配置すると）、この新しいデフォルトレイアウトが使用されます。しかし、ダイアグラム内にすでに存在している他のオカレンスのレイアウトは更新されません。

Use as Default Layout コマンドを実行する前であれば、変更したレイアウトを元のデフォルトレイアウトに戻すことができます。

デフォルトレイアウトに戻す：

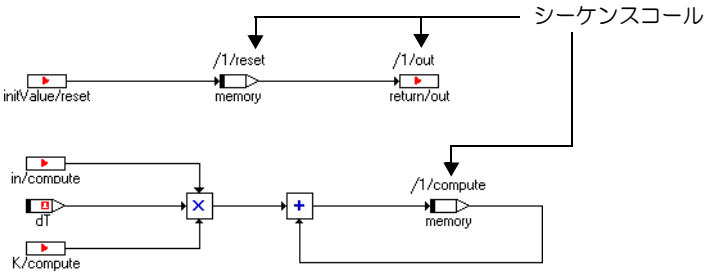
- オカレンスのショートカットメニューから **Default Layout** を選択します。

  コンポーネントのレイアウトエディタで定義されたデフォルトレイアウトに戻ります。

  この操作を行う過程で、接続されたポートが削除されると、その接続線も削除されます。また接続されたポートが移動されると、接続線も一緒に移動し、接続は保たれます。

4.1.6 シーケンスコール

シーケンスコールは、すべての代入処理や、内包されているコンポーネントのメソッドコールに割り当てられ、1 つのシーケンスコールが ASCET ダイアグラム内の 1 つの命令文に対応します。このシーケンスコールにメソッドと番号を割り当てることによって、各メソッドで実行される命令文とその実行順序が定義され、ダイアグラムの処理フローが決まります。

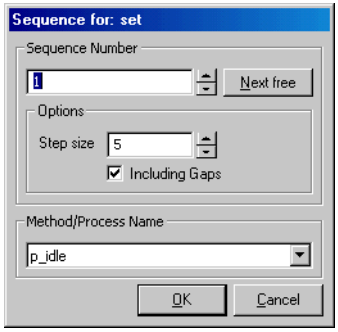


ブロックダイアグラムエディタ上でエレメントや演算子を結ぶ接続線は、エレメントに関連する命令文や演算のシーケンシングが未解決のうちは、ASCET ユーザーオプションで指定された色（デフォルト：緑）で表示されます。つまりこの緑の接続線は、シーケンシングを解決する必要があることを示します。シーケンスコールが定義されると、線は黒色に変わります。

シーケンスコールの編集は、個別に、または単位で行え、またマニュアル操作で行うことも、自動的に一括編集することもできます。

個々のシーケンスコールの編集

個々のシーケンスコールの編集は、シーケンスコールエディタで行います。



このエディタには以下のフィールドが含まれます。

- “Sequence Number”フィールド

  シーケンスコールの番号を入力します。まだ番号が割り当てられていないシーケンスコールの場合は、デフォルト値として 1 が表示されます。

- **Next free** ボタン

  設定されている規則に従い（234 ページを参照してください）、次の空き番号を使用します。

- “Step size”ボックス

  Next free ボタンで次の空き番号を探す際のステップサイズを指定します。

  ASCET ユーザーオプションの **Sequence Step Size** オプション（56 ページの「“Sequencing”ノード」を参照してください）に、ステップサイズのデフォルト値が設定されています。シーケンスエディタでこの値を変更すると、オプションダイアログボックスの設定内容も変更されます。

- **Including Gaps** オプション

  Next free ボタンで次の空き番号を探す際に、既存のシーケンス番号間のギャップ（割り振られていない番号）を検索対象とするかどうかを指定します。

  ASCET ユーザーオプション（56 ページの「“Sequencing”ノード」を参照してください）に、このオプションのデフォルト値が設定されています。シーケンスエディタでこの設定を変更すると、オプションダイアログボックスの設定内容も変更されます。

- “Method/Process Name”コンボボックス

  シーケンスコールを実行するメソッドまたはプロセスを選択します。まだメソッド／プロセスに割り当てられていないシーケンスコールを編集する場合は、現在ブロックダイアグラムエディタの“Diagrams”ペイン内で選択されているメソッド／プロセスが、デフォルトとして選択されています。シーケンスエディタ内で他のメソッド／プロセスを選択すると、“Diagram”ペイン内の選択項目も変わります。

- **OK** ボタン／ **Cancel** ボタン

  設定内容を確定または破棄し、シーケンスエディタを閉じます。

シーケンスエディタを使用してシーケンスコールを編集する：

- 描画領域で、編集したいシーケンスコールを右クリックし、ショートカットメニューを開きます。
- ショートカットメニューから **Edit** を選択し、シーケンスエディタ（“Sequence for”ダイアログボックス）を開きます。
- “Method/Process Name”コンボボックスから、シーケンスコールを割り当てるメソッド／プロセスを選択します。

- “Step size”フィールドに、シーケンス番号の自動割り当てを行う際のステップサイズを入力します。
- シーケンス番号の自動割り当てを行う際に、使用可能な番号を、既存の番号間のギャップも含めて検索するには、**Including Gaps** オプションをオンにします。
- “Sequence Number”ボックスに、シーケンスコールの番号を入力します。

または

- **Next free** をクリックして、選択されているメソッド／プロセス内の空き番号を自動検索します。
- **OK** をクリックします。

  指定された番号がメソッド／プロセス内ですでに使用されているかどうかがチェックされ、まだ使用されていなければ、指定の番号がシーケンスコールに割り当てられ、ブロックダイアグラム上に表示されます。

  “Step size”と“Including Gaps”の設定内容が、ASCET ユーザーオプションにもデフォルト値として設定されます。

Next free を使用して空き番号を決定する場合、以下の規則が適用されます。

1. “Step size”フィールド（ASCET オプションウィンドウの“Sequencing”ノードに含まれる“Sequence Step Size”オプションに相当します）の設定値の整数倍の番号のみが使用されます。たとえばこの値が 5 に設定されていると、5、10、15、20、..... という番号のみが検索対象となります。

2. **Including Gaps** オプションがオンになっていると、既存のシーケンス番号間のギャップ（使用されていない番号）も検索対象となります。この際、上記の 1 番目の条件も有効となり、指定のステップサイズの整数倍でない番号は、検索対象となりません。

   例：

   現在 1 ～ 3、5 ～ 9、および 11 がすでに使用されている場合、“Sequence Step Size”に 5 を設定すると、割り当てられる番号は 4 ではなく 10 となります。

   注記

   ASCET は最後に割り当てられたシーケンス番号を記憶していて、この番号以下のギャップ、つまりこの番号より小さい番号は検索対象となりません。
   エディタを閉じるか、1 つ（237 ページ参照）または複数の（241 ページ参照）シーケンスコールをリセットすると、この記憶されていた番号がリセットされ、自動割り当ては 1 から行われるようになります。

3. **Including Gaps** オプションがオフになっていると、既存のシーケンス番号のうちの最大の番号以降で、かつ“Sequence Step Size”の値の整数倍となる番号が検索対象となります。

   上の例（1 ～ 3、5 ～ 9、および 11 がすでに使用されている）の場合、15 が割り当てられます。

個々のシーケンスコールの編集は、以下のような方法で簡単に行うことができます。

個々のシーケンスコールを自動的に割り当てる：

- “Diagrams”ペインで、シーケンスコールを割り当てたいメソッド／プロセスを選択します。
- 描画領域で、編集したシーケンスコールをダブルクリックします。

または

- シーケンスコールを右クリックし、ショートカットメニューから **Next Number** を選択します。

  どちらの場合も、まだ割り当てられていないシーケンスコールとすでに編集されているシーケンスコールのいずれも選択できます。

注記

ここでメソッド／プロセスが選択されていないと、シーケンスエディタが開き、自動割り当ては行われません。

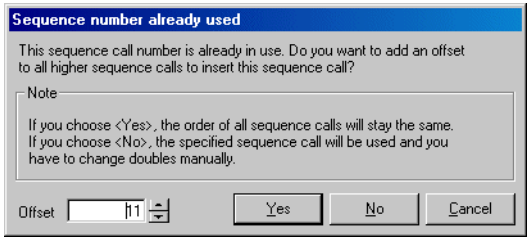
シーケンスコールに、メソッド／プロセスおよび次の空き番号が割り当てられます。

番号は、234 ページに示された規則に基づいて決定されます。

個々のシーケンスコールの番号を増減する：

- 番号を変更したいシーケンスコールを右クリックしてショートカットメニューを開きます。
- **Change → Increment** を選択すると、シーケンスコールの番号が 1 だけ大きくなります。
- **Change → Decrement** を選択すると、シーケンスコールの番号が 1 だけ小さくなります。

同じメソッド／プロセス内ですでに割り当てられている番号を別のシーケンスコールに割り当てようとした場合、以下の警告メッセージが表示されます。



ここで、以下のようにして既存の番号を指定のオフセットでシフトすることができます。この際、シフトされたシーケンスコールの順番は、保持されます。

- 上記のダイアログボックスの“Offset”フィールドに、シーケンスコールのオフセットを入力します。

  デフォルト値として、ASCET ユーザーオプションに含まれる“Sequence Shift Offset”の値（56 ページ参照）が表示されます。

- **Yes** をクリックします。

  編集されたシーケンスコールは、シーケンスエディタで、またはインクリメント／デクリメント操作によって指定された番号が割り当てられ、番号が重複したシーケンスコール、および同じメソッド／プロセス内の以降のシーケンスコールについては、その番号に“Offset”の値が加算されます。

または

- **No** をクリックして、無条件に指定の番号を割り当てます。

  最初の割り当てが間違っていたような場合には、このようにして強制的に番号を割り当て直します。

注記

No を選択した場合は、ダイアグラム上で一方のシーケンスコールをダブルクリックして重複を解消してください。
これを行わないと、コード生成時にエラーが発生します。

または

- **Cancel** をクリックして、シーケンスエディタに戻り、重複しない設定に変更します。

個々のシーケンスコールの「リセット」、つまりシーケンスコールに設定されていた情報の消去は、以下のように行います。

個々のシーケンスコールをリセットする：

- シーケンスコールを右クリックします。
- ショートカットメニューから、**Change → Rreset** を選択します。

  シーケンスコールの番号がリセットされ、接続線が ASCET ユーザーオプションで指定された色（デフォルト：緑）に戻ります。

  同時に、内部で記憶されていた最後に割り当てられた番号がリセットされ、番号の自動割り当てを行う際に、現在空いている最も小さい値から空き番号が検索されるようになります。

前後のシーケンスコールを選択する：

- 描画領域内のシーケンスコールを選択します。

- **Sequence Call → Next Seq. Call** を選択すると、次の番号のシーケンスコールが選択されます。
- **Sequence Call → Last Seq. Call** を選択すると、前の番号のシーケンスコールが選択されます。

個々のシーケンスコールの表示モードを切り替える：

- シーケンスコールが割り当てられているエレメントの入力ピンを右クリックし、ショートカットメニューの **Sequence Call** コマンドのチェックマークをはずします。

または

- シーケンスコールを右クリックし、ショートカットメニューから **Hide** を選択します。

  シーケンスコールが表示されなくなります。
- シーケンスコールが割り当てられているエレメントの入力ピンを右クリックし、ショートカットメニューの **Sequence Call** コマンドにチェックマークを付けます。

  シーケンスコールが表示されます。
- **Select Complete Port** を選択して、シーケンスコールを関連付けるポートをマークします。

  そのダイアグラムアイテムの入力ピンが青色で表示されます。この機能は、複雑なダイアグラムでシーケンスコールを追跡するような場合に便利です。

保護されたシーケンスを作成する：

- **Atomic → Start** を選択して、保護されたシーケンスコールの指定を開始します。

  リアルタイム実行環境において、保護された一連のシーケンスコール（「アトミックシーケンス」とも呼ばれます）は中断されません。
- **Atomic → Stop** を選択して、保護されたシーケンスコールの指定を終了します。

複数のシーケンスコールの編集

複雑なダイアグラムにおいては、非常に多くの数のシーケンスコールが必要となる可能性がありますが、このような場合は、複数のシーケンスコールを同時に編集する機能を利用すると効率的に作業を行えます。選択されたブロック内、または個々のメソッド／プロセス、さらにダイアグラム全体のシーケンスコールをすべて同時に編集することができます。

注記

これらの操作を行う場合は、必ずシーケンスコールのみを選択するようにしてください。もしコネクタ（242 ページ参照）も一緒に選択されていると、自動割り当て機能は動作しません。
コネクタも選択されている場合に使用できるコマンドは、**Sequence Calls → Sequencing - Ignore Info** のみです。

複数のシーケンスコールを自動的に割り当てる：

- “Diagrams”ペインで、シーケンスコールを割り当てるメソッド／プロセスを選択します。
- シーケンスコールを指定するエレメントをすべて選択します。

  シーケンスコールの自動割り当ては、同時には 1 つのメソッドまたはプロセスにしか行えないので、ここでは、同じメソッド／プロセスに割り当てるエレメントのみを選択してください。

注記

複数のメソッド／プロセスが存在する場合、エレメントを 1 つも選択しないで自動割り当てを行うと、不完全、または不正な割り当てが行われてしまいます。

- **Sequence Calls → Sequencing - Ignore Info** を選択します。

  このコマンドは、ダイアグラムを解析し、ASCET に組み込まれたシーケンシングアルゴリズムに従ってシーケンスコールを割り当てます。

自動シーケンシングアルゴリズムではデータフローに基づいて処理を行います。つまり、データが流れる順に従ってエレメントのシーケンシングが行われます。異なる順序で処理を行う必要がある場合は、シーケンスコールをマニュアル操作で指定する必要があります。

指定の番号から始まるシーケンスコールの自動割り当てを行う：

- “Diagrams”ペインで、エレメントを割り当てるメソッド／プロセスを選択します。

- シーケンスコールを指定するエレメントをすべて選択します。
- **Sequence Calls → Sequencing Starting With** を選択します。
- ダイアログボックスの入力フィールドに番号を入力します。

  選択されたエレメントが、指定の番号から始まるシーケンスコールで指定のメソッド／プロセスに組み込まれます。

すでに定義されているシーケンスの次、つまり、すでにメソッド／プロセスに割り当てられている一連のシーケンスコールの後ろに新たなシーケンスコールを追加することができます。その場合、新しく追加される最初のシーケンスコールの番号は、すでに定義されているシーケンス内の最後のシーケンスコールの番号より 1 だけ大きい番号になります。

既存のシーケンスの後ろにシーケンスコールを追加する：

- “Diagrams”ペインで、エレメントを割り当てるメソッド／プロセスを選択します。
- すでに定義されているシーケンスの後ろに追加したいエレメントをすべて選択します。
- **Sequence Calls → Sequencing Appending** を選択します。

  選択されたエレメントのシーケンスコールが、選択されたメソッド／プロセスに組み込まれているシーケンスの後ろに追加されます。

連続する複数のシーケンスコールの番号を一括してシフトするには、以下のように操作します。

一連のシーケンスコールを一括してシフトする：

- シフトしたい一連のシーケンスコールのうち、最も小さい番号のシーケンスコールを選択します。
- ショートカットメニューから **Change → Shift by offset** を選択します。

  選択されたシーケンスコール、および同じメソッド／プロセスに割り当てられたそれよりも大きい番号のシーケンスコールの番号が、ASCET ユーザーオプションに含まれる“Sequence Shift Offset”（56 ページの「“Sequencing”ノード」を参照してください）の値だけ大きくなります。

1 つのメソッド／プロセス、またはダイアグラム全体について、シーケンス番号の間隔を変更することができます。

シーケンスコールの間隔を変更する：

- “Diagrams”ペインから、シーケンスコールの間隔を調整したいメソッド／プロセスを選択します。
- **Sequence Calls → Scale To Step Size → For Method/Process** を選択します。

  選択されたメソッド／プロセスに割り当てられたシーケンスコールの番号が、ASCET ユーザーオプションに含まれる“Sequence Step Size”（56 ページ参照）の値になります。

または

- **Sequence Calls → Scale To Step Size → For Diagram** を選択します。

  ダイアグラム内のすべてのシーケンスコールの番号の間隔が変更されます。

複数のシーケンスコールをリセットする：

- **Sequence Calls → Reset → For Diagram** を選択すると、現在のダイアグラム内のすべてのシーケンスコールがリセットされます。
- **Sequence Calls → Reset → For Method/Process** を選択すると、“Diagrams”ペインで現在選択されているメソッド／プロセスに割り当てられているすべてのシーケンスコールがリセットされます。
- **Sequence Calls → Reset → For Selected Blocks** を選択すると、描画領域で現在選択されている複数のエレメントに割り当てられているシーケンスコールがすべてリセットされます。

  上記のどのコマンドの場合も、最初にユーザーに対して処理の実行についての確認が求められます。ここで実行を了承すると、該当するシーケンス番号がすべて 0 になり、シーケンス名も削除されます。接続線は ASCET ユーザーオプションで指定された色（デフォルト：緑）に変わります。

シーケンスコールの表示モードを切り替える：

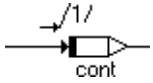
- **Sequence Calls → Hide → For Diagram** を選択すると、ダイアグラム内のすべてのシーケンスコールが非表示になります。

- **Sequence Calls → Hide → For Method/Process** を選択すると、現在“Diagrams”ペインで選択されているメソッド／プロセスに割り当てられているすべてのシーケンスコールが非表示になります。
- **Sequence Calls → Hide → For Selected Blocks** を選択すると、ブロック選択された複数のエレメントに割り当てられているすべてのシーケンスコールが非表示になります。
- **Sequence Calls → Hide → Unused** を選択すると、現在使用されていないすべてのシーケンスコールが非表示になります。
- 上記の **Hide** コマンドと逆の処理を行う **Show** コマンドにも、同じ 4 種類のオプションがあります。

コネクタ

コネクタは、`If...Then` または `If...Then...Else` のような処理フロー制御文に代入処理を接続するために使用します。

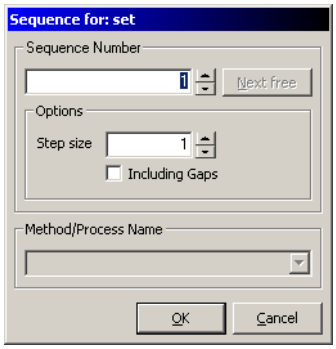
コネクタを作成する：



- コネクタとして使用したいシーケンスコールを右クリックします。
- ショートカットメニューから **Connector** を選択します。

  シーケンスコールがコネクタに変わります。コネクタは、処理フロー制御文に接続することができます。

コネクタの編集は、シーケンスコールと同様にシーケンスエディタ（233 ページ「シーケンスエディタを使用してシーケンスコールを編集する：」の項を参照してください）で行います。ただしコネクタの場合は、**Next free** ボタンと“Method/Process Name”フィールドは無効です。エディタを開くには、コネクタをダブルクリックするか、またはショートカットメニューから **Next free** を選択します。



コネクタの場合、シーケンスコールで使用される自動番号割り当て機能は、ほとんど利用できません。

コネクタを自動的に割り当てる：

注記

コネクタについて利用できる自動割り当て機能は、この機能のみです。

- “Diagrams”ペインで、シーケンスを割り当てるメソッド／プロセスを選択します。
- 選択したメソッド／プロセスで使用するコネクタをすべて選択します。

注記

ここでコネクタ以外のアイテムを選択すると、以下の操作は機能しません。

- **Sequence Calls → Sequencing - Ignore Info** を選択します。

  このコマンドは、ダイアグラムを解析し、ASCET に組み込まれたアルゴリズムに従ってコネクタの番号を割り当てます。

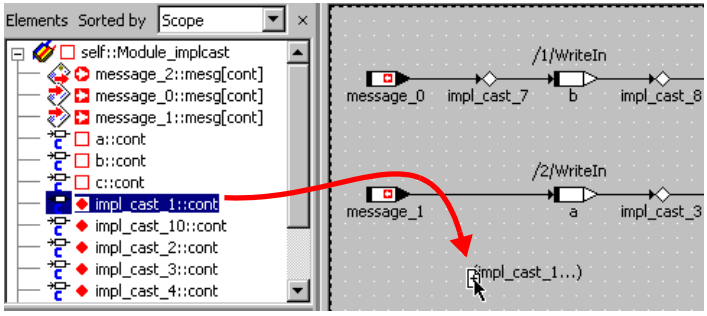
4.1.7 ブロックダイアグラム内のインプリメントキャスト

ブロックダイアグラムエディタでは、他のグラフィックエレメントと同様に、ツールバーのボタン（**mplementation cast** -  ）を使用してインプリメンテーションキャスト（『ASCET リファレンスガイド』の「インプリメンテーションキャスト」の項を参照してください）を追加し、編集することができます。作成されたインプリメンテーションキャストはエレメントリストに追加され、それをドラッグ＆ドロップで描画領域に配置し、接続します。

注記

インプリメンテーションキャストは、論理エレメントには適用できません。論理エレメントに接続しようとすると、接続線の色はエラーを表わす赤になります。

インプリメンテーションキャストにはシーケンスコールは割り当てられません。コード生成時に、コンテキストに基づいた正しい処理順序が決定されます。



またブロックダイアグラムエディタには、既存の演算子にインプリメンテーションキャストを自動的に追加する便利な機能も用意されています。演算子（＋、－、×、÷、abs、neg）のショートカットメニューから **Add Implementation Casts** を選択すると、演算子のすべての入力と出力にインプリメンテーションキャストが付きます。

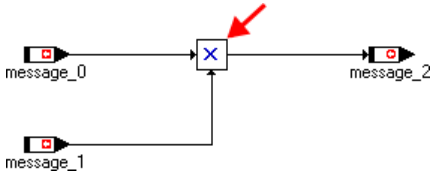
演算子にインプリメンテーションキャストを自動的に追加する：

注記

いずれかの入出力がすでにインプリメンテーションキャストに直接接続されている演算子についてはこの機能は実行できません。実行しようとすると、以下のエラーメッセージが表示されます。

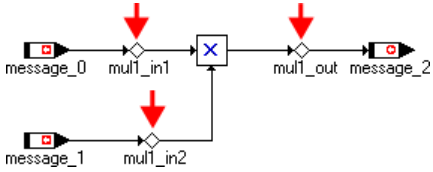
```
This operator is already connected to at least one
implementation cast. Please specify further
implementation casts individually.
```

- インプリメンテーションキャストを追加したい演算子を選択します。



- 演算子を右クリックしてショートカットメニューを開き、**Add Implementation Casts** を選択します。

  ダイアグラム上に充分な空きスペースがある場合は、演算子のすべての入力と出力の接続線上にインプリメンテーションキャストが挿入されます。



  充分な空きスペースがない場合は、以下のようなメッセージが表示されます。

```
The layout is too tight to place an implementation cast
automatically. Please rearrange the diagram.
```

- 演算子の接続線が長くなるように配置を調整し、再度上記の操作を行ってください。

インプリメンテーションキャストの名前は、以下の規則に従って自動的に割り当てられます。

```
<operator><m>_<pin type><n>
```

- <operator>：選択された演算子に応じて、add、sub、mul、div、abs、neg のいずれか
- <m>： 演算子番号

  インプリメンテーションキャストが追加された同じタイプの演算子の番号です。最初の演算子には 1 が割り当てられ、順に 1 ずつ大きくなります。
- <pin type>: インプリメンテーションキャストに接続された演算子のピンについての説明（例： in = 入力、out = 出力）
- <n>： インプリメンテーションキャストの番号
  - 演算子の 1 番目の入力に接続されたインプリメンテーションキャストに番号 1 が割り当てられ、それ以降、順に番号が割り当てられます。演算子に 1 つしか入力がない場合、番号は省略されます。
  - 演算子の出力が複数のエレメントに接続されている場合、インプリメンテーションキャストには 1 から順に番号が割り当てられます。演算子の出力が 1 つのエレメントにのみ接続されている場合は、番号は省略されます。

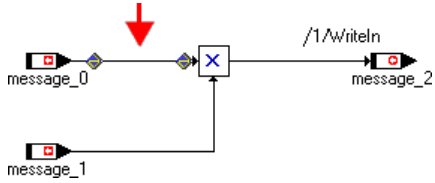
接続線上にインプリメンテーションキャストを追加する：

数値が接続されたすべての接続線上（つまりデータの経路上）に、マニュアル操作でインプリメンテーションキャストを追加することができます。

注記

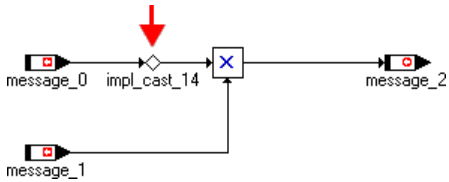
この操作は、+, -, *, /, abs, neg, max, min, mux 以外の演算子、論理エレメントへの接続線、および処理フロー制御文からの接続線上では行えません。

- インプリメンテーションキャストを追加したい接続線を選択します。



- 接続線を右クリックしてショートカットメニューを開き、**Add Implementation Casts** を選択します。

  接続線上にインプリメンテーションキャストが挿入されます。



この方法で作成されたインプリメンテーションキャストの名前は、以下の規則に従って自動的に割り当てられます。

```
impl_cast_<n>
```

上記の <n> はインプリメンテーションキャストの番号です。1 つの接続線上に最初に作成されたインプリメンテーションキャストにはこの番号は付きません。2 番めのものに番号 1 が付き、順に 1 ずつ大きくなります。

この命名規則は、演算子用に作成されたもの（245 ページ参照）以外のすべてのインプリメンテーションキャストに適用されます。

デフォルト状態においては、インプリメンテーションキャストは他のすべてのエレメントと同様に“Elements“リストに表示されますが、これを以下のようにして非表示にすることができます。

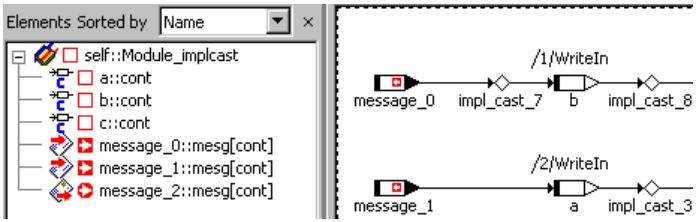
インプリメンテーションキャストの表示／非表示を切り替える：

- **View → Show/Hide** を選択してそのサブメニューを開きます。

  現在インプリメンテーションキャストが表示されている場合は、**Impl. Casts in Elements List** というメニュー項目にチェックマークが付いています。

- **View → Show/Hide → Impl. Casts in Elements List** を選択します。

  “Elements”リスト内にインプリメンテーションキャストが表示されなくなり、上記のメニュー項目のチェックマークが消えます。

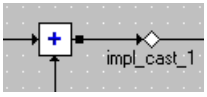


  また描画領域のインプリメンテーションキャストはそのまま表示されますが、ショートカットメニューのアイテム数は少なくなります。

- 再び表示状態に戻すには、もう 1 度 **View → Show/Hide → Impl. Casts in Elements List** を選択します。

以下のようにして、インプリメンテーションキャストとテンポラリ変数を共用することができます。

- テンポラリ変数の後方にインプリメンテーションキャストが付加されている場合、そのテンポラリ変数はインプリメンテーションキャストの影響は受けません。



- インプリメンテーションキャストの直後にテンポラリ変数を使用することができます。

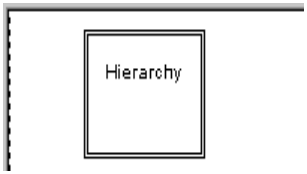
4.1.8 グラフィック階層

複雑なダイアグラムをわかりやすく整理するために、ダイアグラムの各部分を階層ブロック内にまとめ、ダイアグラム上でシンボルとして表示されるようにすることができます。この階層化はグラフィックを見やすくするためだけのものなので、どのアイテムを階層ブロック内に入れても、ダイアグラム全体の機能にはまったく影響しません。階層はネストさせることが可能なので、階層ブロックの中にさらに下位の階層ブロックを含めることができます。

新しい階層を追加する：

- ブロックダイアグラムエディタのツールバーにある **Hierarchy** ボタンをクリックして、マウスカーソルに階層ブロックをロードします。
- 描画領域内の、階層ブロックを配置したい位置をクリックします。

  階層ブロックがダイアグラムに追加されます。



ダイアグラムの一部のアイテムを階層ブロックにする：

- 階層ブロックにしたいダイアグラムアイテムのグループを選択します。
- **Hierarchy** ボタンをクリックします。

  選択されていたアイテムをすべて含む階層ブロックが、ダイアグラム上に配置されます。

選択されたダイアグラムアイテムは、新しく作成された階層ブロック内に配置されます。また、ブロックのフレームの外側のアイテムと内側のアイテムを接続する接続線ごとにピンが作成され、デフォルト名が割り当てられます。ダイアグラムアイテムを階層ブロック内にドラッグ＆ドロップで移動することはできません。以下の方法で、クリップボードを介して移動またはコピーしてください。

ダイアグラムアイテムを階層フレーム内／外に移動する：

- **Cut** または **Copy** コマンドで、階層ブロックの外側のアイテムをクリップボードに移動またはコピーします。
- 階層ブロックをダブルクリックします。

  階層ブロックの内容が表示されます。
- **Paste** コマンドでアイテムを貼り付けます。
- 描画領域をダブルクリックして、階層ブロックの上位階層に戻ります。

階層ブロックを展開／削除する：

- 階層ブロックを右クリックして、ショートカットメニューから **Resolve** を選択します。

  階層ブロックは削除され、中に含まれていたすべてのアイテムは、1 つ上の階層に表示されます。

または

- 階層ブロックを選択します。
- **Edit → Delete** を選択します。

  階層ブロックおよびその中に含まれるすべてのアイテムは削除されます。

階層ブロックの表示モードを変更する：

- 階層ブロック名を変更するには、ブロック内を右クリックして、ショートカットメニューから **Rename Hierarchy** を選択します。

  階層ブロック名を入力するよう求められます。
- 名前を入力し、**OK** をクリックします。
- 階層ブロック名の表示／非表示を切り替えるには、ブロック内を右クリックし、ショートカットメニューから **Show/Hide Name** を選択します。

  階層ブロック名が表示されなくなります。このコマンドをもう一度選択すると、階層ブロック名が再び表示されます。
- 階層ブロックにアイコンファイルを割り当てるには、ブロック内を右クリックし、**Change Icon** を選択します。

  ファイル選択ダイアログボックスが開き、そこでは TIFF、PCX、または BMP フォーマットのビットマップイメージファイルを選択できます。
- イメージファイルを選択してから **OK** をクリックします。

  イメージファイルが、ダイアグラムと ASCET データベースに格納されます。
- 階層ブロックのサイズを変更するには、ブロックを選択し、ハンドルをドラッグします。
- 階層ブロックをデフォルトサイズに戻すには、ショートカットメニューから **Set To Default Size** を選択します。

階層レベル間を行き来する方法は、ダイアグラム内に組み込まれているコンポーネント間を行き来する方法と同じです。

階層レベル間をナビゲートする：

- 階層ブロックをダブルクリックします。

または

- ブロック内を右クリックし、ショートカットメニューから **Next Level** を選択します。

  階層ブロックが開き、その内容が描画領域に表示されます。

- 階層ブロックの描画領域内で、アイテムが表示されていない部分をダブルクリックします。

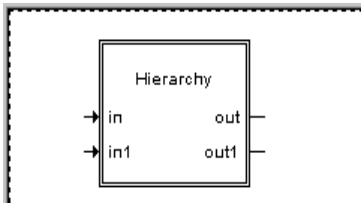
  階層ブロックが閉じて上位階層の内容が表示されます。階層ブロックは、選択された状態でブロックとして表示されます。

異なる階層レベル間にまたがるデータフローは、入力ポートと出力ポートで表わされます。これらは、単にレベルの境界にまたがる接続線を表わしたものです。

階層ブロックに入力ピンと出力ピンを追加する：

- 階層ブロックを右クリックします。
- ショートカットメニューから **Add Inpin** または **Add Outpin** を選択します。

  階層ブロックのフレーム上には、任意の数の入力ピンと出力ピンを追加することができます。入力ピンと出力ピンは矢印で表され、ピン名は階層ブロックの内側に表示されます。



入力ピンと出力ピンの表示モードを変更する：

- 入力ピンまたは出力ピンを右クリックします。
- ピン名を変更するには、ショートカットメニューから **Rename Pin** を選択します。
- 名前を入力してから **OK** をクリックします。
- ピン名を非表示にするには、**Pin Name** を選択します。

- もう一度このコマンドを選択すると、ピン名が再び表示されます。
- 入力ピンまたは出力ピンを削除するには、**Remove Pin** を選択します。
- 階層ブロックを右クリックし、**Show Pin Names** または **Hide Pin Names** を選択すると、その階層ブロックのすべてのピン名の表示／非表示が同時に切り替わります。

ダイアグラムアイテムをピンに接続してそのピンを階層ブロック内の別のアイテムに接続するのは、この 2 つのアイテムを直接接続するのと同じ意味を持ちます。

4.1.9 ダイアグラム間のナビゲート

複数のダイアグラムで定義されるコンポーネント

1 つのコンポーネント（クラスまたはモジュール）を、複数のダイアグラムに分けて構造化し、複雑な機能をわかりやすく記述することができます。ダイアグラムには、パブリックメソッドだけを含むパブリックダイアグラムと、プライベートメソッドだけを含むプライベートダイアグラムがあります。

パブリックメソッドには他のコンポーネントからもアクセスできますが、プライベートメソッドにはコンポーネント内からしかアクセスできません。またプライベートダイアグラムには、ステートマシンの場合に限り、アクションやコンディションを定義できます。

ブロックダイアグラムで定義された各コンポーネントには、少なくとも 1 つのパブリックダイアグラムが含まれ、これには `Main` という名前が割り当てられます。クラスには任意の数のパブリックまたはプライベートダイアグラムを使用できますが、モジュールにはパブリックダイアグラムしか使用できません。

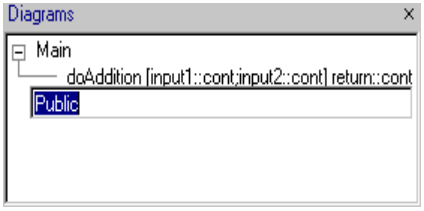
新しいダイアグラムを作成する：

- **Diagram → Add Diagram → Public**（または **Private**）を選択します。

または

- “Diagrams”ペインのショートカットメニューから、**Diagram → Add Diagram → Public**（または **Private**）を選択します。。

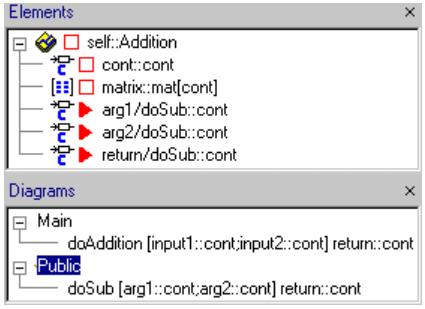
  “Diagrams”ペインに新しいパブリック（またはプライベート）ダイアグラムが追加され、その名前が選択された状態となります。。



- 名前を入力して **<Enter>** をクリックします。

  後にダイアグラムの名前を編集するには、**Diagram → Rename Diagram** を選択します。
- ダイアグラムにメソッドまたはプロセスを追加し、機能を定義します。

“Elements”リストには、現在開いているダイアグラムのインターフェースアイテム（引数、ローカル変数、戻り値）のみが表示されます。その他のアイテム（変数やパラメータ）は、複数のダイアグラムにおいて使用できるので、常に“Elements”リストに表示されます。



ブロックダイアグラムエディタにおいては、1 度に 1 つのダイアグラムしか編集できません。1 つのコンポーネントに複数のダイアグラムが含まれる場合は、各ダイアグラムを 1 つずつ描画領域にロードして編集します。

ダイアグラムをロードする：

- “Elements”リストからダイアグラムを選択します。

- **Diagram → Load Diagram** を選択します。

または

- ショートカットメニューから **Load Diagram** を選択します。

  ダイアグラムが描画領域にロードされます。

  それまで開いていたダイアグラムに対する変更内容がまだ保存されていなかった場合は、新しいダイアグラムがロードされる前に変更内容を保存するプロンプトが開きます。

コンポーネントからダイアグラムを削除する：

- “Diagrams”リストから、削除したいダイアグラムを選択します。
- **Diagram → Delete Diagram** を選択します。

または

- ショートカットメニューから **Delete Diagram** を選択します。

  ダイアグラムと、そこに定義されているすべてのメソッド／プロセスが削除されます。

注記

“Diagrams”ペイン内の 1 番目のダイアグラムは削除できません。

メソッドを他のダイアグラムに移動する：

- “Diagrams”ペインから、移動したいメソッドを選択します。

- **Diagram → Move** を選択します。

  “Specification”ダイアログボックスが開き、そこに現在のコンポーネント内のすべてのダイアグラムが一覧表示されます。

- メソッドの移動先のダイアグラムを選択して、**OK** をクリックします。

  選択されたメソッドがターゲットダイアグラムに移動します。元のダイアグラム内にあった、このメソッドのシーケンスコールはすべて削除されます。

メソッドを移動できるのは、そのメソッドのインターフェースエレメントのオカレンスがダイアグラム内にない場合だけです。インターフェースエレメントのオカレンスがある場合には、それを先に削除してからメソッドを移動してください。

コンポーネント間をナビゲートする

他のコンポーネントを内包するコンポーネントをブロックダイアグラムエディタで編集している場合、そこから直接サブコンポーネントを開いて編集することができます。

ブロックダイアグラム内で異なるレベル間を行き来する：

- 編集したいコンポーネントを、描画領域または“Elements”ペインから選択します。
- **Element → Edit Component** を選択します。

または

- ショートカットメニューから **Edit Component** を選択します。

または

- 選択したコンポーネントをダブルクリックします。

  選択されたコンポーネントが開きます。

  それまで開いていた上位コンポーネントについて変更内容が保存されていない場合は、サブコンポーネント用に新しいエディタウィンドウが開きます。そうでない場合には、元のエディタウィンドウにサブコンポーネントがロードされます。
- 内包されたコンポーネントから上位のコンポーネントに戻るには、描画領域内の、アイテムが表示されていない位置をダブルクリックします。

  すると、1 つ上のレベル上に戻ります。サブコンポーネントに変更を加えていた場合は、上位のコンポーネント用に新しいエディタウィンドウが開き、サブコンポーネント用のウィンドウはそのまま保持されます。

同じコンポーネント内のグラフィック階層のレベル間を行き来する際も、上記の方法で行います。

4.1.10 コンポーネントの検証

作成したブロックダイアグラムを検証するためには、実験を行って意図したとおりにコンポーネントが機能するかどうかを確認します。本項では、コンポーネントの実験の準備方法、および実験環境の起動方法について説明します。

注記

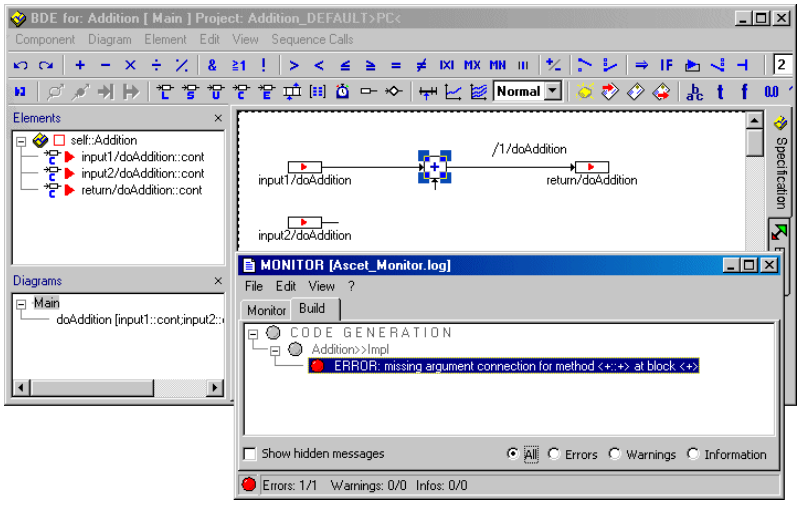
ASCET の個々のモジュールに関して、プロジェクトなしで単体でコード生成とシミュレーションを行えるのは、コードジェネレータとして `Physical Experiment`（405 ページ参照）が選択されている場合のみです。他のコードジェネレータを使用して実験を行う場合、モジュールは必ずプロジェクトに組み込まれている必要があります。このため、各クラスやモジュールには、デフォルトプロジェクトと呼ばれるダミーのプロジェクトが定義され、これを利用しないとインプリメンテーション（実装情報）にアクセスできません。プロジェクトが存在しないと、インポートされるエレメントの変換式やインプリメンテーションは失われてしまいます。

ダイアグラムを解析する：

- **Diagram → Analyze Diagram** を選択して、現在のダイアグラムを解析します。

  ダイアグラムに構文エラーがないかどうかがチェックされます。割り当てられていないシーケンスコール、接続の欠如、ループなどがあると、

モニタウィンドウの“Build”タブにその情報が出力されます（2.4「モニタウィンドウ」の項を参照してください）。



- モニタウィンドウ内のいずれかのエラーメッセージをクリックすると、ブロックダイアグラムエディタ内でそのエラーが発生した箇所が強調表示されます。

コンポーネントのコードを生成する：

- **Component → Generate Code** を選択します。

または

- **Generate Code** ボタンをクリックします。

  現在開いているコンポーネントの C コードが生成されます。

  コードジェネレータが出力するエラーメッセージが表示され、ここでも、エラーメッセージをクリックすることにより、各エラーに関連するダイアグラムアイテムを見つけることができます。

コンポーネントのコードが正常に生成されると、そのコンポーネント用の実験環境を開くことができます。

ASCET で生成されたコードはデータベースに格納され、これをユーザーが見ることはできません。しかし、任意のテキストエディタで見ることができるように、ファイルにコードを書き込むことができます。

生成されたコードを表示する：

- 前述の方法で、現在開いているコンポーネントのコードを生成します。
- **Component → File Out Generated Code → Flat**（または **Recursive**）を選択すると、生成されたコードがファイルに格納されます。

  作成されたファイルは指定のディレクトリに書き込まれます。コードをリカーシブに格納すると、各コンポーネントごとに 1 つのファイルが作成されます。生成されるファイルの名前は、ASCET モニタウィンドウに一覧表示されます。
- 作成されたファイルをテキストエディタで開くと、その内容が表示されます。

Flat を選択すると、現在選択されているコンポーネントのコードだけがファイルに書き込まれます。**Recursive** を選択すると、現在のコンポーネントから参照しているコンポーネントのコードもファイルに書き込まれます。

生成されたコードの内容を見るには、外部エディタ（Notepad、Codewright など）やインターネットブラウザを使用することもできます。外部エディタを使用するには、Windows で“*.c”や“*.h”というファイル拡張子をそのエディタに関連付けておく必要があります。

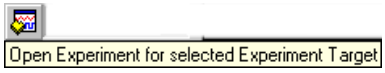
- **Component → View Generated Code** を選択します。

  C コードが生成され、外部エディタで表示されます。
  この際に使用されるテキストエディタは、ASCET オプションウィンドウの“ASCII Editor”ノード（60 ページ参照）で選択できます。

オフラインを開始する：

- **Component → Open Experiment**を選択します。

または



- **Open Experiment for selected Experiment Target** ボタンをクリックします。

  コードが生成され、さらに現在のターゲット用に指定されているコンパイラでコンパイルされて、コンポーネント用の実験環境が開きます。生成されたファイルは、cgen ディレクトリに保存されます。

実験を行う方法は、547 ページの「実験」の章で詳しく説明します。

デフォルトプロジェクトにおいては、グローバルエレメント（380 ページの「グローバル通信の定義」を参照してください）が正しく更新されない場合があります。この場合、ASCET モニタウィンドウに以下のエラーメッセージが表示されます。

```
Error: need export for imported element <name> with type
<type>
```

このエラーを解決するためには、以下のようにしてください。

デフォルトプロジェクト内のグローバルエレメントを定義する：

- ブロックダイアグラムエディタで、**Component → Default Project → Resolve Globals** を選択してグローバルエレメントを解決します。
- **Component → Default Project → Delete Unused Globals** を選択して、使用されていないグローバルエレメントを削除します。

  これで、再び実験を開始することができます。

4.1.11 データ交換

データセットは、必ずそのデータセットが定義されたコンポーネントに関連付けられます。コンポーネントマネージャの **File → Import** コマンドでインポートされたデータセットは、そのデータセットが定義されたコンポーネントに書き込まれます。データセットのエクスポートを行う際には、データベースアイテムのエクスポートと同じ規則が適用されます。詳しくは、90 ページの「フォルダやデータベースアイテムのエクスポート」の項を参照してください。

コンポーネントのデータセットをエクスポートする：

- **Component → Export Data** を選択します。

  ファイル選択ダイアログボックスが開きます。
- ファイル名とパス名を選択します。
- **OK** をクリックします。

  選択されたファイルにデータセットが書き込まれます。

ダイアグラムの一部分をファイルに格納し、それを別のコンポーネントにインポートすることができます。インターフェースエレメント以外のすべてのダイアグラムアイテムをファイルに格納できます。

ダイアグラムの一部分をファイルに格納する：

- ダイアグラムの、ファイルに書き込みたい部分を選択します。
- **Edit → Copy** を選択します。

- **Edit → File Out Buffer** を選択します。

  ファイル選択ダイアログボックスが開きます。
- パスを選択し、ファイル名と拡張子 `.asc` を指定します。
- **OK** をクリックします。

  クリップボード上のダイアグラムアイテムが、指定されたファイルに格納されます。

ダイアグラムの一部をファイルから読み取る：

- ファイルに書き込まれたダイアグラムの一部分を現在開いているコンポーネントにロードするには、**Edit → File In Buffer** を選択します。

  ファイル選択ダイアログボックスが開きます。
- インポートしたいダイアグラムエレメントが入っているファイルを選択します。
- **OK** をクリックします。

  ファイルに格納されているダイアグラムの一部分が ASCET 内部のクリップボードにコピーされます。
- **Edit → Paste** を選択します。

  クリップボードの内容が、現在開いているコンポーネントのダイアグラムに書き込まれます。

特性カーブ／マップや配列のデータをファイルに書き込み、それを再度読み取ることができます。データはタブで区切られた ASCII フォーマットで書き込まれるので、任意のスプレッドシートやワードプロセッサで読んだり編集することが可能です。

配列や特性カーブ／マップのデータをファイルに書き込む：

- “Elements” ペインから、特性カーブまたはマップ、または配列を選択します。
- **Element → File Out Data** を選択します。

  ファイル選択ダイアログボックスが開きます。
- パスと、拡張子 `.dat` の付いたファイル名を指定します。
- **OK** をクリックします。

  選択されたエレメントのデータがファイルに書き込まれます。

配列や特性カーブ／マップのデータをファイルから読み込む：

- “Elements”ペインから、特性カーブまたはマップ、または配列を選択します。
- **Element → File In Data** を選択します。

  ファイル選択ダイアログボックスが開きます。
- 読み込みたいデータが格納されているファイルを選択します。
- **Open** をクリックします。

  ファイルから読み込まれたデータが、選択されたエレメントに書き込まれます。

4.1.12 ブロックダイアグラムエディタの使用法

ダイアグラムの保存

定義されたコンポーネントの保存は、以下のように行います。

コンポーネントの定義を保存する：

- **Diagram → Store to Cache** を選択します。

または

- “Diagrams”ペインで、ショートカットメニューから **Store to Cache** を選択します。

  コンポーネントがキャッシュメモリに格納されますが、データベース内には保存されることはありません。
- 変更内容を恒久的にデータベースに保存するには、**File → Save Database** を選択します。

または

- **Save** ボタンをクリックします。

  Save コマンドを実行すると、すべての変更内容がデータベースに保存されます。

ブロックダイアグラムの表示内容の更新や表示スケールの変更を行う：

- ダイアグラムを再描画するには、**View** → **Redraw** を選択します。

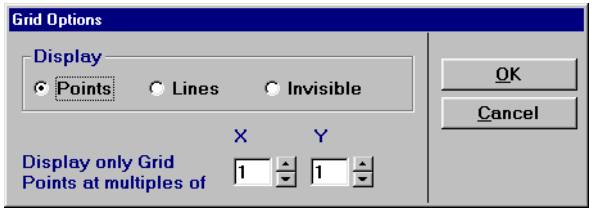
  エレメントの位置を何度も変更していると、ダイアグラムが正しく表示されなくなってしまう場合があります。このような場合にこのコマンドを選択すると、ダイアグラムが再描画されて正しく表示されます。
- “Zoom”コンボボックスでダイアグラムの表示スケールを選択します。またここに任意の倍率を直接入力することもできます。
- ダイアグラム全体を見るには、“Zoom”コンボボックスで `Page Layout` を選択します。

  ダイアグラムがズームアウトされ、1 ページ目の全体が表示されます。
- “Zoom”コンボボックスで `100%` を選択すると、デフォルトの表示スケールに戻ります。

描画領域のグリッドを変更する：

- **View** → **Grid** を選択します。

  “Grid Options”ダイアログボックスが開きます。



- **Points** ラジオボタンをクリックすると、描画領域にグリッドポイントが表示されます。
- **Lines** ラジオボタンをクリックすると、グリッドラインが表示されます。
- **Invisible** ラジオボタンをクリックすると、グリッドが表示されなくなります。
- つまりグリッドポイントまたはラインを表示する間隔を決定する倍数を指定します。

  たとえば、**Points** オプションを選択して倍数を 2 に設定すると、グリッドポイントの間隔はデフォルトの 2 倍になります。

- **OK** をクリックすると、グリッドオプションが設定されます。

印刷領域を設定する：

- ASCET オプションウィンドウの“Paper Size”ノード（57 ページ参照）で用紙サイズを選択します。

  選択されたサイズは、次回ブロックダイアグラムを開く時に有効となります。

- **View → Page frame Portrait** を選択すると、ダイアグラムが縦長のフォーマットで表示されます。

  この設定で、ダイアグラムを印刷する際の向きが決まります。使用できる描画領域は実際に印刷されるページよりも大きいので、ダイアグラム中の印刷される部分が破線で示されます。この線により定義されるフレームの形は、選択されている向きによって変わります。

- **View → Page frame Landscape** を選択すると、横長のフォーマットに変わります。

  用紙の方向のデフォルトは、ASCET オプションウィンドウの“Paper Size”ノード（57 ページ参照）の“Paper Orientation”オプションで設定します。

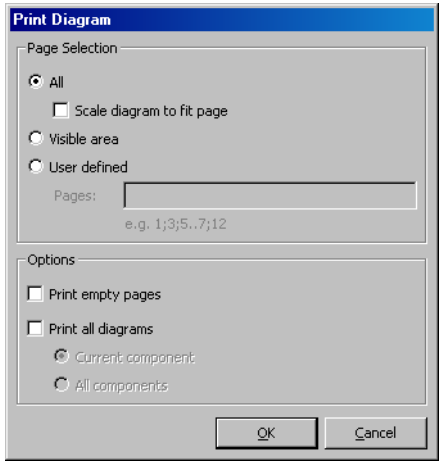
描画領域が選択されたページフォーマットよりも大きい場合、描画領域内に、印刷されるページ範囲が点線で示されます。さらに、ASCET オプションウィンドウの“Block Diagram”ノード（54 ページ参照）に含まれる **Show Page Number** オプションで、ブロックダイアルグラムエディタにページ番号を表示するかどうかを指定することができます。ページ境界の点線とページ番号は、ダイアグラムエレメントによって隠れる場合もあります。

ページ境界線の色とページ番号の色は、ASCET オプションウィンドウの“Colors”ノード（55 ページ参照）に含まれる **Watermark Color** オプションで変更できます。

ブロックダイアグラムを印刷する：

- 描画領域を印刷するには、**View → Print Diagram** を選択します。

  “Print Diagram”ダイアログボックスが開きます。



- 印刷用オプションを設定します。
- **OK** をクリックします。

  “Printer Selection”ダイアログボックスが開きます。
- “Printer”フィールドでプリンタを選択します

  プリンタの設定を変更する場合は **Setup** をクリックします。
- **OK** をクリックして設定を確定します。

  設定に従って、コンポーネントの内容が印刷されます。

“Print Diagrams”ダイアログボックスで設定できるオプションは、以下のとおりです。

- **All**

  描画領域全体を印刷します。
- **Scale diagram to fit page**

  ダイアグラムが 1 ページに印刷されるようにスケーリングを調整します。**All** オプションがオンの場合にのみ有効です。
- **Visible area**

  描画領域のうち、現在表示されている部分のみを印刷します。

- **User defined**

  描画領域のうち、ユーザー定義部分のみを印刷します。

  > **注記**
  >
  > **User defined** がオンになっていると、**Print empty pages** と **Print all diagrams** の設定は無効です。

- “Pages:”

  印刷するページを入力するフィールドです。個々のページはセミコロン ";" で区切って入力し、連続するページ範囲を指定するには、先頭ページ番号と最終ページ番号を 2 つのピリオド ".." で接続します。このフィールドは **User defined** がオンの場合にのみ有効です。

- **Print empty pages**

  空のページも印刷します。

- **Print all diagrams**

  コンポーネントの全ダイアグラムを印刷します。

- **Current components**

  現在表示されているコンポーネントのダイアグラムのみ印刷します。このオプションは **Print all diagrams** がオンの場合にのみ有効です。

- **All components**

  内包されるコンポーネントのダイアグラムも印刷します。このオプションは **Print all diagrams** がオンの場合にのみ有効です。

被参照コンポーネントの使用

被参照コンポーネント、つまり、現在のコンポーネントに内包されているコンポーネントを処理対象とするコマンドがいくつかあります。これらのコマンドの機能は **Component** メニューのコマンドと同じですが、処理対象が被参照コンポーネントである点のみが異なります。

内包されているコンポーネントを編集する：

- “Elements” ペインで、内包されているコンポーネントを選択します。
- **Element → Edit Component** を選択して、選択されているコンポーネントのブロックダイアグラムを開きます。

または

- 選択されたコンポーネントをダブルクリックします。

- インポートされたコンポーネントの注釈を見るには、**Elements → Notes** を選択します。

  このコマンドを選択すると、コンポーネントに関連付けられている注釈を読み取り専用で表示することができます。（編集はできません）。ビューについては、665 ページの「ビュー」という項を参照してください。

- 被参照コンポーネントのデータセットをエクスポートするには、**Element → Export Data** を選択します。

  このコマンドの機能は **Component → Export Data** と同じですが、“Elements”リスト内で選択されている被参照コンポーネントだけが処理対象となります。

4.2 ステートマシンエディタ

ステートマシンは、「ステートダイアグラム」と、それを補ういくつかのダイアグラムで構成されます。ステートダイアグラムはブロックダイアグラムの一種で、ここでは「ステート」（状態）が楕円で表現され、「トランジション」（遷移）が方向性のある曲線で表現されます。各ステートを「階層ステート」、つまり別のステートダイアグラムを内包するステートにすることもできます。ステートマシンの「アクション」と「コンディション」は、個別のブロックダイアグラムまたはESDL コードで定義することができます。またパブリックメソッドを個別のダイアグラムとして定義することもできます。なおステートマシンについての詳細とその挙動については、『ASCET リファレンスガイド』の「ステートマシン」の項に詳しく説明されています。

ステートマシンの記述方法は、多くの点でブロックダイアグラムと似ていて、ステートマシンの一部分にブロックダイアグラムを使用することもできます。ブロックダイアグラムエディタに含まれるほとんど機能がステートマシンエディタにも用意されています。この項ではそのような共通部分については詳しく説明されていないので、本項をお読みになる前に、ブロックダイアグラムエディタでブロックダイアグラムを定義する方法をよくご理解いただいておくことをお勧めします。ブロックダイアグラムエディタについては、176 ページの 4.1 項で詳しく説明されています。

ステートマシンの定義は、以下のステップで行います。以降に、各ステップについて詳しく説明します。

- ステートダイアグラムを作成する（ステートやトランジションの編集）
- コンディションとアクションを定義する
- コンディションとアクションをステートマシンのステートとトランジションに割り当てる
- 独立したダイアグラムにパブリックメソッドを定義する

特殊なメニューコマンド

ステートマシン用のメニューコマンドは、ダイアグラムエディタの場合とほぼ同じです（182 ページの「メニューコマンドの概要」を参照してください）が、**Diagram** メニューにはステートマシン専用のコマンドが含まれます。

- Diagram
  - *Add Diagram*

    新しいダイアグラムを作成します。

    → *Public* — パブリックメソッドのみ
    → *Private* — プライベートメソッドのみ
    → *Actions/Conditions BDE* — アクションとコンディションのブロックダイアグラム
    → *Actions/Conditions ESDL* — アクションとコンディションの ESDL コード
  - *Add Trigger*

    トリガを作成します。
  - *Add Action*

    アクションを作成します。
  - *Add Condition*

    コンディションを作成します。
  - *Create State Code Comments*

    ステートとトランジションの ESDL コードの第 1 行目にコメントを挿入します（295 ページ参照）。

4.2.1 ステートダイアグラムの描画

新しいステートマシンを作成する：

- コンポーネントマネージャで、新しいステートマシンを作成するフォルダを選択します。
- **Insert → State Machine** を選択します。

または

- **Insert State Machine** ボタンをクリックします。

  新しいステートマシンが作成されます。
- ステートマシンの名前を入力します。
- メニューバーから **Component → Edit Item** を選択します。

または

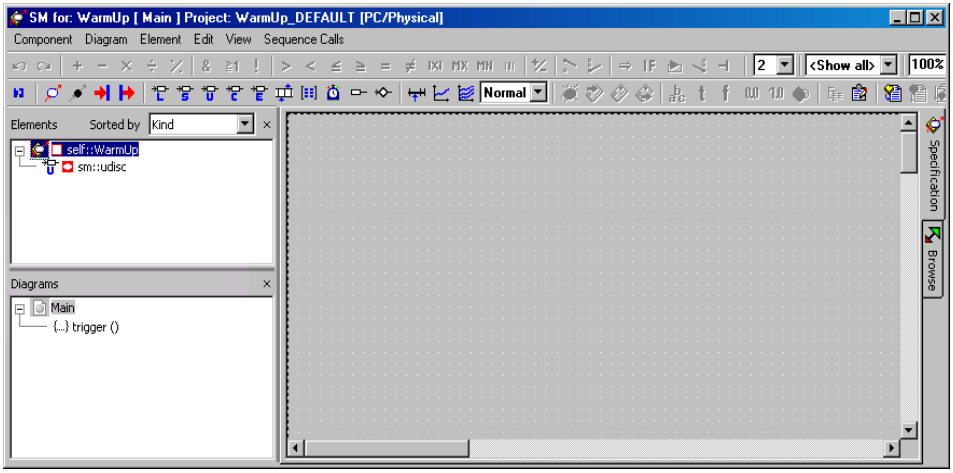
- **<Enter>** キーを押します。

または

- “Elements”リスト内のステートマシン名をダブルクリックします。

  ステートマシンエディタが開きます。

  ステートダイアグラムが開きます。ステートダイアグラム内では演算子は使用できないので、ツールバーの演算子ボタンは無効になっています。



ダイアグラムのステートを配置する：

- **State** ボタンをクリックし、マウスカーソルに新しいステートをロードします。
- 描画領域の、ステートを配置したい位置をクリックします。

  クリックした位置にステートシンボルが配置されます。
- 必要なすべてのステートについて、以上の処理を繰り返します。

1 つのステートマシンには必ず 1 つの「開始ステート」が必要です。クラスが最初に起動される時、ステートマシンは「開始ステート」の状態となります。

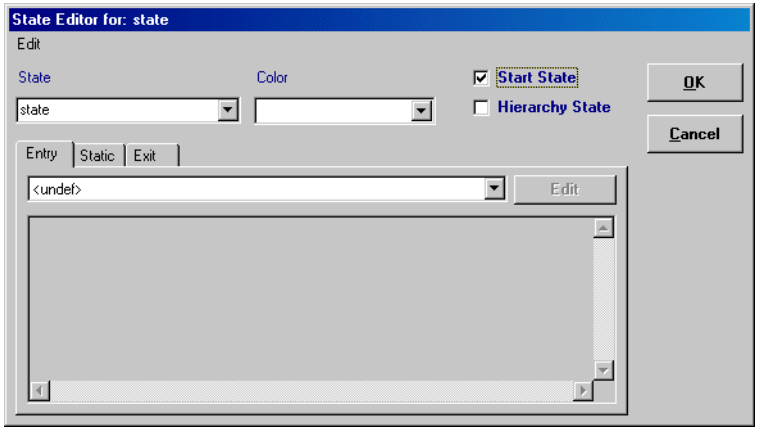
開始ステートを定義する：

- 開始ステートにしたいステートを右クリックして、ショートカットメニューから **Edit State** を選択します。

または

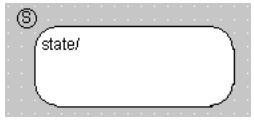
- ステートシンボルをダブルクリックします。

  ステートエディタ（“State Editor for”ダイアログボックス）が開きます。



- **Start State** オプションにチェックマークを付けます。
- **OK** をクリックします。

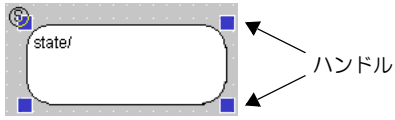
  選択されたステートの左上に、開始ステートであることを示す丸囲みの‘S’が表示されます。



ステートを編集する：

- ステートシンボルを移動するには、そのシンボルを描画領域内の任意の位置にドラッグします。

  ステートを移動させると、そのシンボルに付加されたトランジション（接続線）も自動的に移動し、必要に応じてルーティングも変わります。
- ステートシンボルのサイズを変更するには、そのシンボルをクリックします。



- ハンドルをドラッグして、シンボルのサイズを調整します。
- 他のダイアグラムエレメントと同様に、クリップボードを介してステートシンボルのカット、コピー、ペーストを行うことができます（221 ページ参照）。

最初に作成されたステートのデフォルト名は `state` で、次のステートは `state_1` となります。この名前は、ANSI C のネーミング規則の範囲で変更することが可能です。

ステートの名前を変更する：

- 描画領域内で、名前を変更したいステートを選択します。
- ステートエディタを開きます（268 ページ参照）。
- "State" フィールドに新しい名前を入力します。
- **OK** をクリックします。

  入力された名前が ANSI C 規格に準じていない場合は、メッセージが表示されます。

- **OK** をクリックしてダイアログボックスを閉じた後、有効な名前を再入力します。

  すでに存在している名前が入力された場合、_n というシンボルが自動的に付加されます。ここで付加される n は、入力された名前にまだ付加されていない番号の中で最小のものとなります。

  新しい名前がステートシンボル内に表示されます。

新しく作成されたステートは白で表示されます。この色を変更するには、以下のように操作します。

ステートの表示色を変更する：

- 編集したいステートをステートエディタで開きます。
- “Color”コンボボックスで表示色を選択します。

  ステートシンボルの色が選択された色に変わります。

以下のようにして、ステートのレイアウト（形と色）を他のステートにコピーすることができます。

ステートのレイアウトをコピーする：

- レイアウトのコピー元とするステートを右クリックします。
- ショートカットメニューから **State Layout → Copy Layout** を選択します。

  ステートの形と色がクリップボードにコピーされます。
- レイアウトのコピー先とするステートを右クリックします。
- ショートカットメニューから **State Layout → Paste Layout** を選択します。

  クリップボードにコピーされていたレイアウトが割り当てられ、最初のステートと同じ色と形になります。

ステート以外にも、ジャンクションダイアグラム（『ASCET リファレンスガイド』の「ジャンクション」の項を参照してください）でも色の変更が行えます。

ジャンクションを作成する：

- **Junction** ボタンをクリックしてマウスカーソルに新しいジャンクションをロードします。
- 描画領域内で、ジャンクションを配置したい場所をクリックします。

  ジャンクションのシンボル（名前の付いていない丸印）がクリックした場所に描画されます。
- 複数のジャンクションを作成する場合は、上記のステップを繰り返してください。

ステートの場合とは異なり、ジャンクションにアクションを割り当てることはできません。ジャンクションの編集は、移動とサイズの変更のみが可能です。

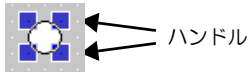
ジャンクションを編集する：

- ジャンクションの位置を変えるには、描画領域内でシンボルをマウスでドラッグします。

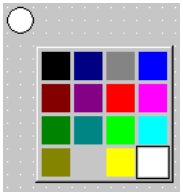
  ジャンクションを移動させると、それに接続されたトランジションも移動し、必要に応じてルーティングの変更も行われます。

- ジャンクションとステート、または 2 つのジャンクションが重なると、両方の境界線が赤で警告表示されます。その場合は、一方を移動して重ならないようにしてください。

- ジャンクションの大きさを変更するには、そのシンボルをクリックします。



- 表示されたバンドルをドラッグして、シンボルの大きさを調節します。
- ジャンクションは、他のダイアグラムエレメントと同様に、クリップボードを介したカット／コピー／ペースト（221 ページ参照）を行うことができます。



- ジャンクションの表示色を変更するには、ジャンクションを右クリックし、ショートカットメニューから **Edit Color** を選択します。

  使用可能な色を示す選択ボックスが開きます。
- 色を選択します。

  ジャンクションの色が、選択された色に変わります。

トランジションを作成する：

- **Connect** ボタンをクリックします。

または

- 描画領域の、エレメントが表示されていない部分を右クリックします。

  接続モードがアクティブになり、カーソルが十字カーソルに変わります。

- 接続を開始したいステートまたはジャンクションのシンボルの内側をクリックしてから、マウスカーソルを接続先のステートまたはジャンクションまで移動します。

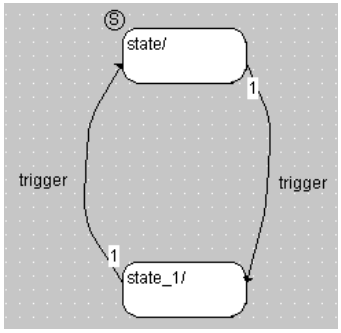
  カーソルの軌跡に線が描かれます。この線はクリックしたシンボルの縁から始まります。

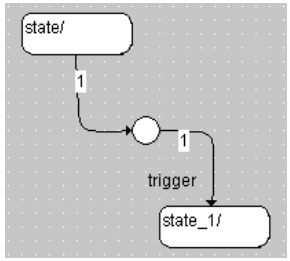
注記

ジャンクションからジャンクションへのトランジションは作成できません。

- 接続先にしたいシンボルの内側をクリックします。

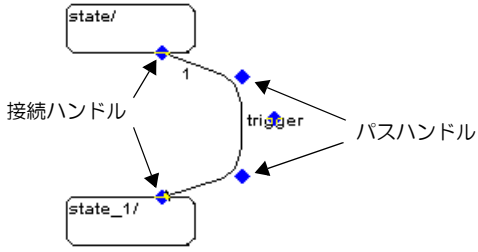
  1 本の曲線で表されるトランジションが 2 つのシンボルの間に描かれます。この曲線には、第 1 のステート／ジャンクションシンボルから第 2 のステート／ジャンクションシンボルに向かう矢印が付きます。





トランジションのルーティングを変更する：

- トランジションをクリックして、ダイアグラムにパスハンドルと接続ハンドルを表示します。



- パスハンドルをドラッグして、トランジションの道筋を変更します。
- 接続ハンドルをドラッグして、ステートシンボル上の接続位置を変更します。

  既存の接続を、接続ハンドルをドラッグするだけで別のソースステートまたはターゲットステートに割り当てることができます。

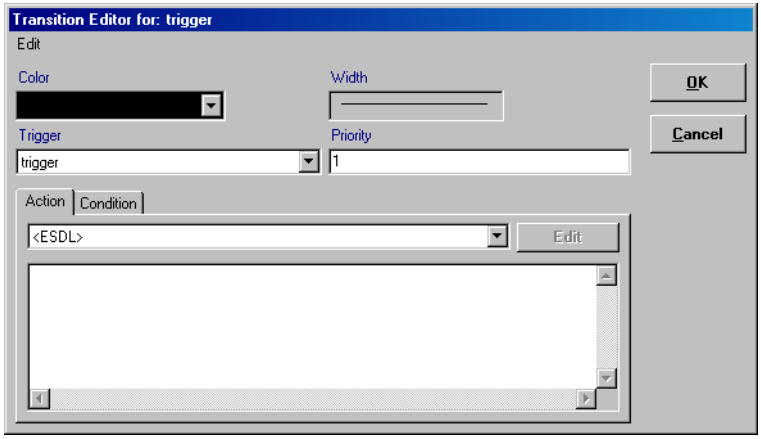
トランジションの外観を変更する：

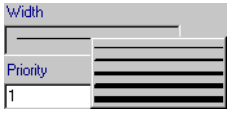
- トランジションを右クリックして、ショートカットメニューから **Edit Transition** を選択します。

または

- トランジションをダブルクリックします。

  トランジションエディタ（“Transition Editor”ダイアログボックス）が開きます。





- “Color”コンボボックスから、色を選択します。
- “Width”コンボボックスから、線の幅を選択します。
- **OK** をクリックして、設定内容を確定します。

  トランジションが、選択された色と幅で表示されます。

1 つの新しいステートに対して、デフォルトで 1 つのトリガが生成されます。これに別のトリガを追加することは可能ですが、1 つのステートマシン内に複数のトリガが存在すると、冗長なプログラムコードが生成されてしまいます（『ASCET リファレンスガイド』の「コードサイズの最適化」の項を参照してください）。

トリガを追加する：

- **Diagram → Add Trigger** を選択して、新しいトリガを作成します。

  トリガが“Diagrams”ペインに追加されます。トリガの名前は強調表示された状態になっていて、直接変更することができます。
- トリガの名前を入力してから **<Enter>** を押します。

4.2.2 階層ステート

ステートダイアグラムは、階層構造（『ASCET リファレンスガイド』の「階層ステート」についての記述を参照してください）にすることができます。つまり、あるステートが別のステートダイアグラムを内包することが可能です。ステートマシンにおいて階層ステートがアクティブになった時、実際にはその階層ステート内のサブダイアグラムの開始ステートがアクティブとなります。それに対して、階層ステートがヒストリ（『ASCET リファレンスガイド』の「ヒストリ」についての記述を参照してください）を持っている場合は、前回その階層ステートが終了したときにアクティブであったサブダイアグラムのステートが、再びアクティブとなります。

また、階層ステートには「閉階層ステート」と「開階層ステート」とがあります。閉階層ステートではサブダイアグラムが新しい描画レベルに作成されるのに対し、開階層ステートのサブダイアグラムは同じ描画レベルに作成されます。しかし、どちらのタイプも機能は同じです。つまり、開階層ステートを使用している場合には、描画レベルを切り替える必要がありません。

1 つのステートダイアグラム内に、開階層ステートと閉階層ステートを共存させることはできますが、1 つのステートにはどちらか一方の階層タイプしか定義できません。描画レベル内でのステートのオーバーラップにより不適切な構成になっている場合には、関連するステートシンボルの表示色が変わります。

階層ステートマシンのコードは、フラット（すべてのステートを 1 つの `switch` 文で記述）または階層的（階層に従って `witch` 文をネストさせる）に生成することができます。後者の場合、コードサイズが飛躍的に小さくなります（『ASCET リファレンスガイド』の「階層コードの生成」を参照してください）。

以下のようにして、階層コードを生成するかどうかを切り替えます。

階層コード生成のオン／オフを切り替える：

- ステートマシンを含むプロジェクトを開きます。

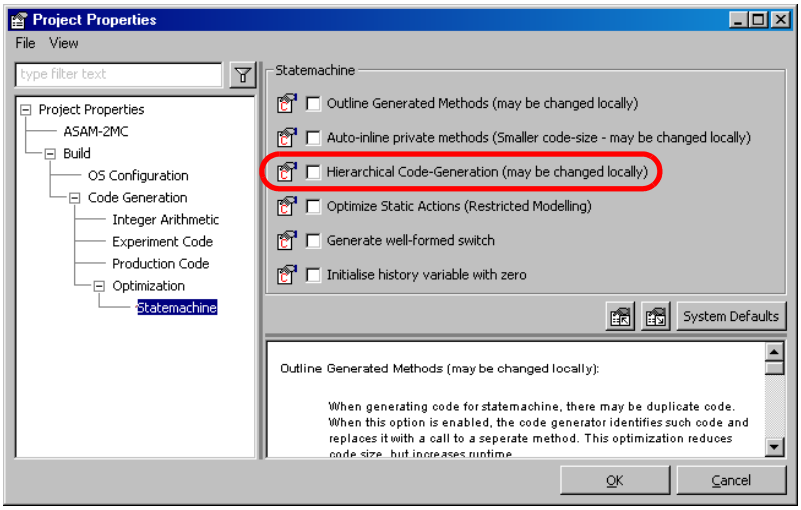
  デフォルトプロジェクトも使用できます。

- プロジェクトエディタで、“Project Properties”ダイアログボックスを開きます（401 ページ参照）

1. オフにする

- “Statemachine”ノード（414 ページ参照）で、**Hierarchical Code-Generation (may be changed locally)** オプションをオフにします。

  プロジェクト内のすべてのステートマシンについて、階層コードの生成が行われなくなります。



この場合、各ステートマシンごとのオン／オフ設定は無効となります。

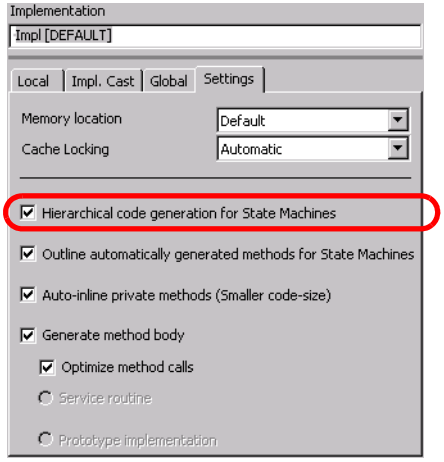
2. オンにする

- “Statemachine”ノード（414 ページ参照）で、**Hierarchical Code-Generation (may be changed locally)** オプションをオンにします。
- “Project Properties”ダイアログボックスを閉じます。

  階層コードの生成を行うには、さらに、各ステートマシンごとに以下の設定を行う必要があります。

- プロジェクト内のステートマシンを開き、**Component → Edit Implementation** を選択してステートマシンのインプリメンテーションエディタを開きます。

- “Settings”タブで、**Hierarchical code generation for State Machines** オプションをオンにします。



- インプリメンテーションエディタを閉じます。

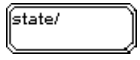
  上記の 2 つのオプションがオンになっている場合にのみ、階層コードが生成されます。

閉階層ステート

閉階層ステートを追加する：

- 階層ステートにするステートを、親ダイアグラムに追加します。
- そのステートを右クリックし、ショートカットメニューから **Edit State** を選択します。
- ステートエディタの、**Hierarchy State** オプションにチェックマークを付けます。
- **OK** をクリックします。

  ステートが階層ステートになり、そのステートシンボルが 2 重線で示されます。.



別の階層レベルに移動する：

- 階層ステートをダブルクリックします。

または

- 階層ステートを右クリックして、ショートカットメニューから **Next Level** を選択します。

  1 レベル下の階層が開きます。つまり、階層ステートに内包されているステートダイアグラムが表示され、階層ステートの編集が可能となります。

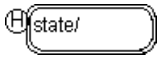
- 階層ステート内において、描画領域内の、ダイアグラムアイテムが表示されていない位置をダブルクリックします。

  1 レベル上に戻ります。

階層ステートにヒストリを持たせる：

- 階層ステートを右クリックし、ショートカットメニューから **History** を選択します。

  その階層ステートはヒストリを持つようになるので、この階層ステートに入った時、開始ステートではなく、前回この階層ステートが終了したときのステートとなります。



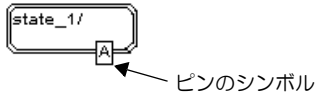
階層ステートのシンボル上のピンを用いて、互いに異なる階層にあるステートを接続することができます。ピンを通じてステートを接続しても、機能上は直接接続したのと同じことになります。

階層ステートにピンを追加する：

- 階層ステートを右クリックします。
- ショートカットメニューから **Add Pin** を選択します。

  デフォルト名のピンがステートシンボル上に作成され、対応するシンボルが階層の内側に描かれます。

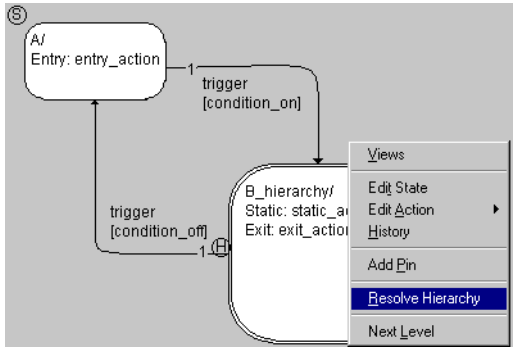
  これで、ステートシンボル上のピンにトランジションを接続してから、そのピンを階層内のステートに接続できるようになりました。



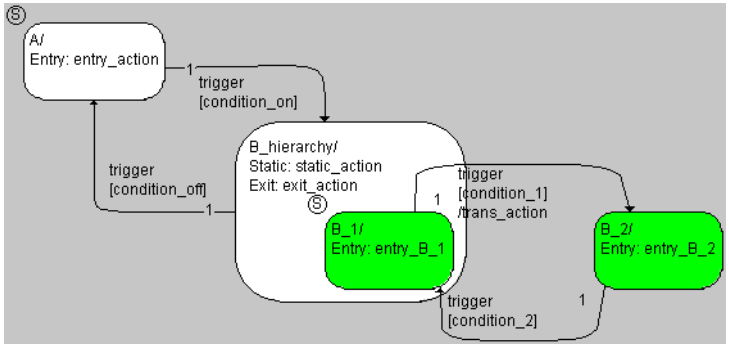
階層ステートを展開する：

- 階層ステートを右クリックします。

- ショートカットメニューから、**Resolve Hierarchy** を選択します。



すべてのエレメントとその接続線が、階層ステートからその上の描画レベルの同じ位置にコピーされます。これで、開始ステートを持つ第 2 の独立したステートダイアグラムが作成されました。また閉じていた階層ステートを囲んでいた二重線は消え、このステートにヒストリがあった場合、そのヒストリも削除されています。



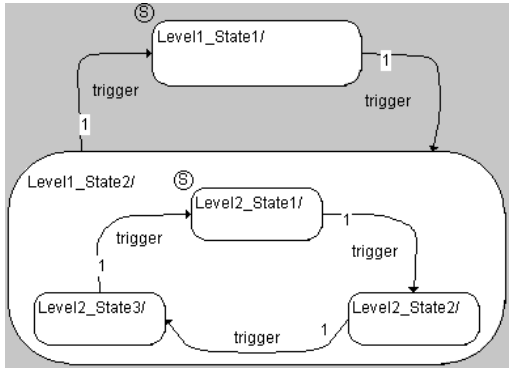
階層を展開した状態では、ステートマシンの機能は不明確になり、挙動が変わってしまうことがよくあります。たとえば上の例において、ステート B_1 と B_2 は、元は閉階層ステートである B_hierarcy に含まれていたものですが、展開によって、現在は B_1 のみが B_hierarcy に含まれるようになりました。この状態で B_1 から B_2 へのトランジションが発生すると、それに付随して他のアクションも実行されてしまいます（『ASCET リファレンスガイド』の「階層ステートマシン」の項を参照してください）。

明確なダイアグラムを作成して正しい機能を実現するには、展開を行った際は、適宜修正を行う必要があります。

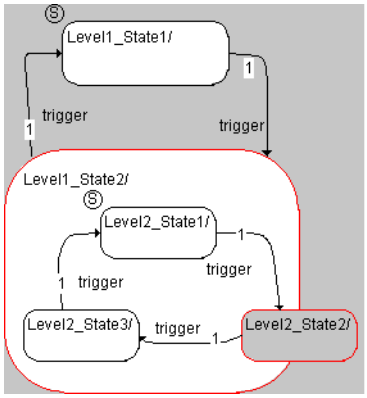
階層ステートを追加する：

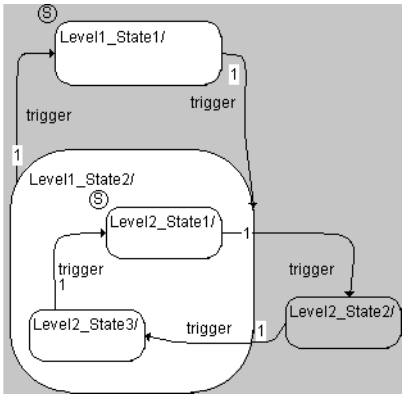
- ハンドルをドラッグして、階層ステートを追加するステートの領域を広げます。
- サブダイアグラム内の必要なすべてのステートとトランジションを、選択した階層ステート内のステートシンボルに追加します。
- サブダイアグラムの開始ステートとするステートを選択します。

  ステートシンボル内にステートダイアグラムを作成する方法は、描画領域内にステートダイアグラムを作成する方法と同じです。



サブステートから、そのサブステートが属する階層の外のステートに遷移することも可能です。ただし、異なる階層レベル間のトランジションの際に実行されるアクションは、同レベル間のトランジションの場合とは異なる点に注意してください（『ASCET リファレンスガイド』の「階層ステートマシン」の例 12 と例 15 とを参照してください）。以下の図の `Level2_State2` のように、ステートの全体または一部が階層ステートの外に配置されている場合、そのステートはその階層の外側にある、とみなされます。





閉階層ステートの場合と同様に、開階層ステートにもヒストリを持たせることができます。

4.2.3 コンディションとアクションの定義

各ステートには Entry、Static および Exit アクションが定義され、各トランジションにはトリガ、コンディション、およびトランジションアクションが定義されます。「コンディション」(= 遷移条件）と「アクション」(= 処理）はメソッドに似ていて、クラスのメソッドと同じ方法で定義されます。(「コンディション」と「アクション」についての詳細は『ASCET リファレンスガイド』を参照してください。)

コンディションに記述された遷移条件は、そのコンディションが定義されているトランジションを持つステートがアクティブである時は、常に評価されます。コンディションが true なら、トランジション（= 遷移）が発生します。コンディションの戻り値は必ず `true` か `false` です。Entry、Exit、トランジションアクションが定義されている場合、それらはトランジションが発生するたびに実行され、Static アクションはトランジションが発生しないときに実行されます。ステートマシンの実際の動作は、『ASCET リファレンスガイド』の「ステートマシンの基本動作」および「階層ステートマシン」の項に詳しく説明されています。

コンディションとアクションは、それぞれ個別のダイアグラム（`ActionCondition` ダイアグラム）として、ブロックダイアグラムまたは ESDL コードで定義することができます。この場合、ステートマシンには少なくとも 2 つのダイアグラムが含まれることになります。一方、ステートエディタやトランジションエディタにおいてコンディションとアクションを ESDL コードで直接定義する場合は、個別のダイアグラムは使用されません。

階層ステートの Static アクションについては、コードサイズの最適化を行うことができます。詳しくは『ASCET リファレンスガイド』の「階層ステートの Static アクション」の項を参照してください。

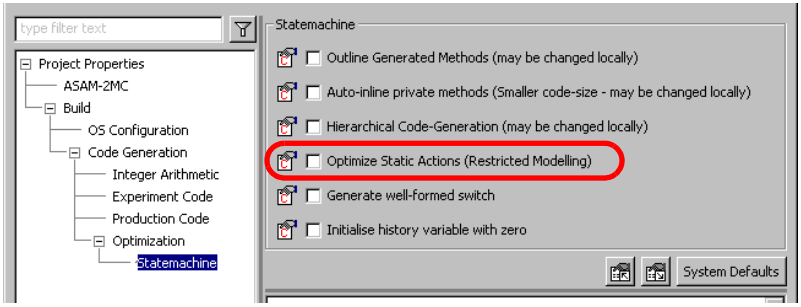
Static アクションの最適化を行うかどうかは、以下のようにして切り替えます。

Static アクション最適化のオン／オフを切り替える：

- ステートマシンを含むプロジェクトを開きます。
  デフォルトプロジェクトも使用できます。
- プロジェクトエディタで、“Project Properties”ダイアログボックスを開きます（401 ページ参照）

1. オフにする

- “Statemachine”ノード（414 ページ参照）で、**Optimize Static Actions (Restricted Modelling)** オプションをオフにします。
  プロジェクト内のすべてのステートマシンについて、Static アクションの最適化が行われなくなります。



2. オンにする

注記

最適化をオンにすると、モデリングに際して以下の制約が発生します。ステートマシンのサブステートに、階層ステートの外に遷移するトランジションが含まれる場合、このトランジションには、そのサブステートからのすべてのトランジション内で最上位の優先度を割り当てる必要があります。

- “Statemachine”ノード（414 ページ参照）で、**Optimize Static Actions (Restricted Modelling)** オプションをオンにします。
  Static アクションの最適化が有効になります。生成されるコードが変わるため、ステートマシンの挙動が変化する場合があります。既存のステートマシンについてこのオプションをオンに切り替える際は、挙動の変化に注意してください。

コンディションとアクションを個別のダイアグラムで定義する

コンディションとアクションは、複数の `ActionCondition` ダイアグラムの中に定義することができます。1 つのステートマシンには任意の数の `ActionCondition` ダイアグラムを含めることができ、各ダイアグラムにはブロックダイアグラムまたは ESDL コードが含まれます。

このようにコンディションとアクションを定義するには、まず必要なダイアグラムを作成し、そこに通常のブロックダイアグラムまたは ESDL コードでアクションとコンディションを記述します。ブロックダイアグラムの記述方法は 204 ページの「ブロックダイアグラムの作成」、ESDL コードの記述方法は 323 ページの「ESDL エディタ」を参照してください。

アクション／コンディションのダイアグラムを作成する：

1. ブロックダイアグラム

   - **Diagram → Add Diagram → Actions/ Conditions BDE** を選択します。

   または

   - “Diagrams” ペインを右クリックして、ショートカットメニューから **Add Diagram → Actions/ Conditions BDE** を選択します。

     `ActionCondition_BDE` ダイアグラムが作成されます。

   - “Diagrams” ペインで、ダイアグラムに名前を付けます。

2. ESDL

   - **Diagram → Add Diagram → Actions/ Conditions ESDL** を選択します。

   または

   - “Diagrams” ペインを右クリックして、ショートカットメニューから **Add Diagram → Actions/ Conditions ESDL** を選択します。

     `ActionCondition_ESDL` ダイアグラムが作成されます。

   - “Diagrams” ペインで、ダイアグラムに名前を付けます。

コンディションを追加する：

- “Diagram” ペインで、コンディションを追加したいダイアグラムの名前をダブルクリックします。
- **Diagram → Add Condition** を選択します。

または

- “Diagrams”ペインのショートカットメニューから **Add Condition** を選択します。

  新しいコンディションが作成され、その名前が“Diagrams”ペインに表示されます。

  コンディションは自動的に作成される logical 型の戻り値を持ちます。コンディションが定義されたダイアグラムを開くと、“Elements”ペインに戻り値が表示されます。
- コンディションの名前を入力してから **<Enter>** を押します。

アクションには通常、引数も戻り値もありません。そのため、アクション用のインターフェースエレメントは作成されませんが、それ以外の点では、コンディションと同じ方法でアクションを追加できます。**Diagram → Add Condition** またはショートカットメニュー **Add → Action** を使用します。

アクション／コンディションのダイアグラムを開く：

- “Diagrams”ペインで、開きたい `ActionCondition` ダイアグラムを選択します。
- ダイアグラムをダブルクリックします。

または

- **Diagram → Load Diagram** を選択します。

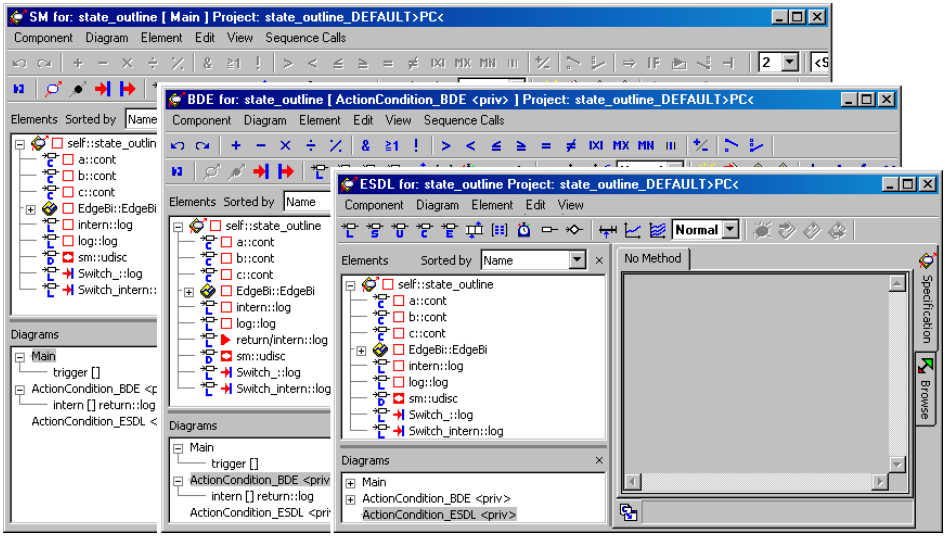
  現在開いているダイアグラムの変更内容がまだ保存されていない場合、変更内容を保存するかどうかを尋ねるメッセージが表示されます。

  — **Yes** をクリックして変更内容を保存します。

  または

— **No** をクリックして変更内容を破棄します。

エディタウィンドウに `ActionCondition` ダイアグラムが開きます。



コンディションとアクションの定義は、一般的なブロックダイアグラムや ESDL コンポーネントの場合と同じです。定義の方法は、4.1 項「ブロックダイアグラムエディタ」と 4.4 項「ESDL エディタ」を参照してください。ただし、コード生成用のメニューコマンドとボタンは、ステートマシンエディタでのみ有効です。

定義されたアクション／コンディションは、独立した関数としてコード生成するか、または必要な箇所にインラインコードとして挿入する（「自動インライニング」：『ASCET リファレンスガイド』の「ステートマシンの最適化」を参照してください）ことができます。

以下のようにして、自動インライニングを行うかどうかを切り替えます。

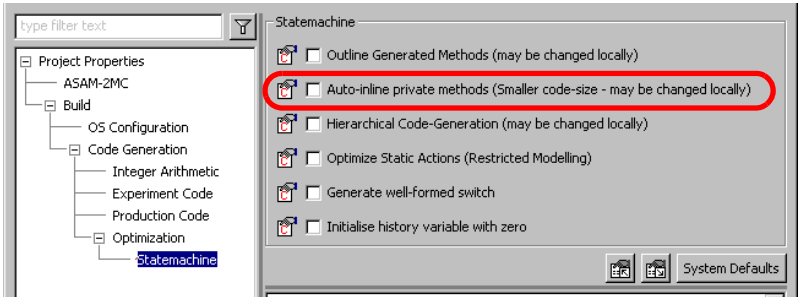
自動インライニングのオン／オフを切り替える：

- ステートマシンを含むプロジェクトを開きます。デフォルトプロジェクトも使用できます。
- プロジェクトエディタで、“Project Properties”ダイアログボックスを開きます（401 ページ参照）

1. オフにする

- “Statemachine”ノード（414 ページ参照）で、**Auto-inline private mathods (Smaller code size - may be changed locally)** オプションをオフにします。

  プロジェクト内のすべてのステートマシンについて、自動インライニングが行われなくなります。



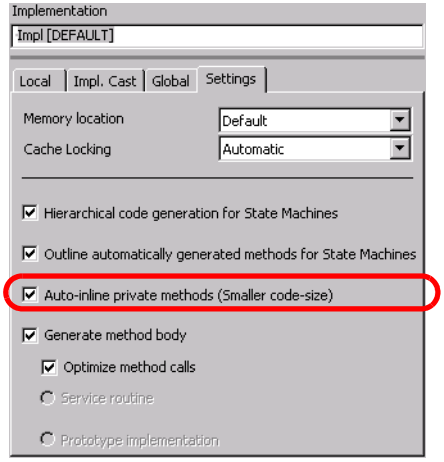
  この場合、各ステートマシンごとのオン／オフ設定は無効となります。

2. オンにする

- “Statemachine”ノード（414 ページ参照）で、**Auto-inline private mathods (Smaller code size - may be changed locally)** オプションをオンにします。
- “Project Properties”ダイアログボックスを閉じます。

  自動インライニングを行うには、さらに、各ステートマシンごとに以下の設定を行う必要があります。
- プロジェクト内のステートマシンを開き、**Component → Edit Implementation** を選択してステートマシンのインプリメンテーションエディタを開きます。

- “Settings”タブで、**Auto-inline private mathods (Smaller code size)** オプションをオンにします。



- インプリメンテーションエディタを閉じます。

  上記の 2 つのオプションがオンになっている場合にのみ、インラインコードが生成されます。

  後者のオプションにより、個々のステートマシンについて自動インライニングが行われないようにすることができます。

コンディションとアクションを使用する

`ActionCondition` ダイアグラム内のアクションとコンディションは、それらが使用されるトランジションやステートに明確に割り当てる必要があります。

トランジションにトリガ／優先度／コンディション／アクションを割り当てる：

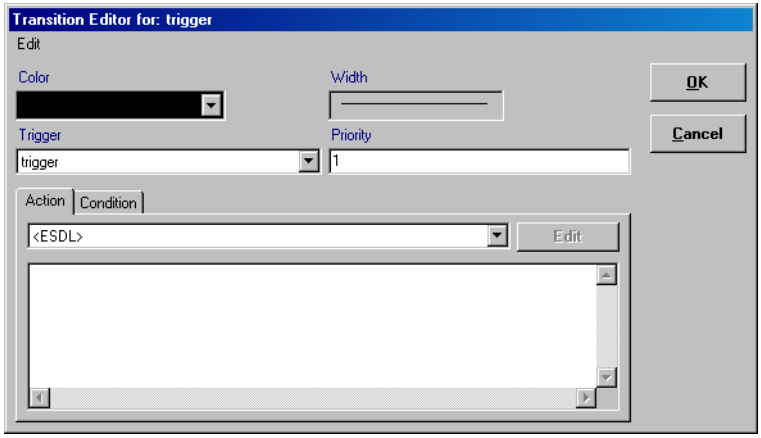
トランジションには、トリガと優先度は必ず定義しなければなりませんが、コンディションとトランジションアクションは必ずしも必要ではありません。

- トランジションまたはトランジションセグメントをダブルクリックします。

または

- トランジション／セグメントを右クリックし、ショートカットメニューから **Edit Transition** を選択します。

  トランジションエディタが開きます。



- “Priority”フィールドに数値を入力して、トランジションの優先度を指定します。

注記

優先度の数値が大きいほど、そのトランジションの優先度が高くなります。

1 つのステートまたはジャンクションからの同じトリガに複数のトランジション／トランジションセグメントが定義づけられている場合、それらのトランジションコンディションは優先度の順に調べられるので、1 つのステート／ジャンクションからの同じトリガによるトランジションには、それぞれ異なる優先度を割り当てておく必要があります。

- “Trigger”コンボボックスからトリガを選択します。

注記

トランジションがジャンクションによって 2 つのセグメントに分かれている場合、そのうちのいずれかのセグメントにのみトリガを割り当てることができます。

ステートダイアグラムに定義されているすべてのトリガイベントが、このボックスに一覧表示されます。

- “Action”タブのコンボボックスから、トランジションアクションを選択します。

注記

ジャンクションに向かうトランジションの場合、アクションを割り当てることはできないため、このコンボボックスは非アクティブ状態になっています。

このコンボボックスには、`ActionCondition` ダイアグラムで定義されたすべてのアクションが一覧表示され、任意のものをトランジションアクションとして指定することができます。

- “Condition”タブのコンボボックスからコンディションを選択します。

  このコンボボックスには `ActionCondition` ダイアグラムで定義されているすべてのコンディションが一覧表示されます。

注記

コンディションが定義されていないと、コンディションは常に true と見なされ、トランジションは毎回発生するようになります。

- **OK** をクリックします。

  トランジション／セグメントのトリガイベント名、コンディションの優先度と名前、およびトランジションアクションが表示されます。

必要に応じて、各ステートの Entry アクション、Static アクションおよび Exit アクションを指定することができます。これらの設定はすべて任意です。

ステートにアクションを割り当てる：

- ステートをダブルクリックします。

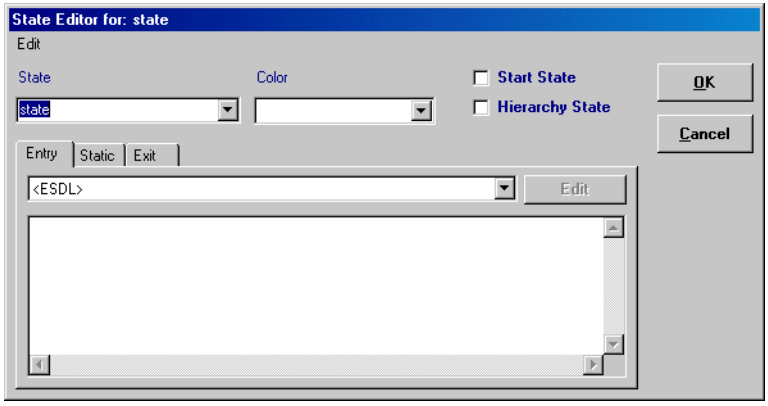
または

- ステートを右クリックし、ショートカットメニューから **Edit State** を選択します。

または

- ショートカットメニューから **Edit Action → *<option>*** でアクションのタイプを選択します。

  ステートエディタが開きます。

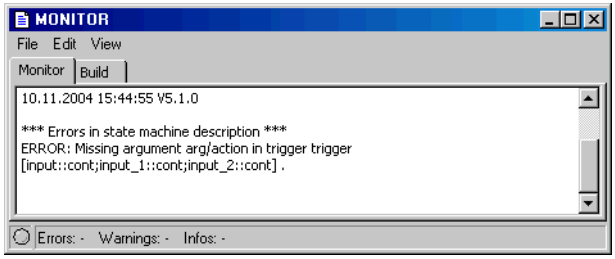


- “Entry”タブで、コンボボックスから Entry アクションを選択します。
- “Static”および“Exit”タブで、Static アクションと Exit アクションを選択します。

  ここでは、すべてのアクションを使用することができます。
- **OK** をクリックします。

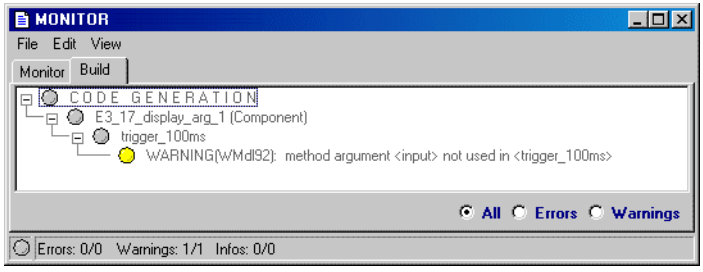
  アクションの名前は、ステートシンボル内に表示されます。

アクションやコンディションに引数を割り当てると、ステートエディタまたはトランジションエディタを閉じる時にすべてのトリガがチェックされ、同じ名前と型を持つ引数があるかどうかが自動的に調べられます。ない場合は、ASCET モニタウィンドウにエラーメッセージが表示されます。

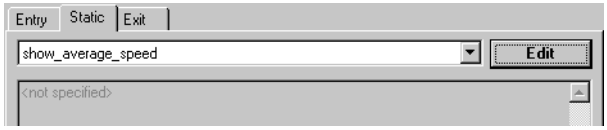


その場合は、エラーメッセージに示されたトリガ引数を 298 ページの方法で作成してください。

反対に、作成されたトリガ引数がいずれのアクションやコンディションで使用されていなくても、ステートマシンの機能には影響しません。コード生成時に、ASCET モニタウィンドウの“Build”タブにワーニングメッセージが表示されるのみです。



`ActionCondition` ダイアグラムでアクションやコンディションを定義すると、各タブ上の **Edit** ボタンがアクティブになります。このボタンで、それぞれのタブ上のアクションやコンディションを編集することができます。



アクション／コンディションを編集する：

- ステートエディタまたはトランジションエディタを開きます。
- いずれかのタブ上で、アクションまたはコンディションを指定します。

- 指定したアクション／コンディションを編集するために、**Edit** をクリックします。
- “Confirm”ダイアログボックスで、入力内容をエディタ上で確定します。

  ステート／トランジションエディタが閉じで、設定内容が確定されます。

  もう 1 度“Confirm”ダイアログボックスが開きます。
- 今度は、入力内容をステートダイアグラムに保存するかどうかを確認します。

  ステートダイアグラムが保存されます。

  指定されたアクションまたはコンディションを含むダイアグラムが、別のエディタウィンドウ上に開きます。

コンディションとアクションをステートダイアグラムで定義する

ステートダイアグラム内のコンディションやアクションを定義する ESDL コードを、ステートエディタまたはトランジションエディタに直接入力することもできます。この場合、個別のダイアグラムを作成する必要はなく、また **Edit** ボタンは無効になります。

変数とパラメータを宣言するには、ステートマシンエディタの各種エレメントボタンをクリックして“Elements”ペインに名前を入力します。これらの変数は、ESDL コード内でその名前を直接キー入力して使用することができます。同じように、トリガ引数や、インポートされたすべてのコンポーネント、およびその中で定義されたメソッドやコンポーネントを、名前をキー入力して参照することができます。これにより、ブロックダイアグラムを追加せずに、数行のコードのみで複雑なステートマシンの動作を記述することが可能となります。ESDL 言語については、『ASCET リファレンスガイド』の「ESDL によるボディ記述」に説明されています。

アクションをステートエディタで記述する：

- アクションを指定するステートを右クリックします。
- Entryアクションを編集するにはショートカットメニューから **Edit Action → Entry** を選択します。
- Static アクションを編集するにはショートカットメニューから **Edit Action → Static** を選択します。
- Exit アクションを編集するにはショートカットメニューから **Edit Action → Exit** を選択します。

  ステートエディタが開きます。

- コンボボックスから <ESDL> を選択します。

  コンボボックスの下の入力フィールドが有効になります。

- ESDL コードを入力フィールドに入力します。

  入力の際、**<Ctrl> + <Z>** で最後の入力内容を取り消すことができます。
- **OK** をクリックします。

ステートエディタの **Edit** メニューには、ESDL でのアクション記述時に使用できるいくつかの編集機能が用意されています。

ESDL コードのカット、コピー、ペースト

- 編集したいコードの範囲を選択します。

または

- **Edit → Select All を使用して、現在編集中のアクション内のコード全体を選択します。**
- **Edit → Cut** で、選択された範囲のコードが削除され、クリップボードに保存されます。
- **Edit → Copy** で、選択された範囲のコードがクリップボードにコピーされます。
- **Edit → Paste** で、クリップボードからコードが貼り付けられます。

コードの検索や置換も可能です。316 ページの「C コードテキストの検索と置換を行う：」を参照してください。

また、外部ソースのコードをアクションに挿入したり、記述したコードを外部ファイルに保存することも可能です。これについては 322 ページの「外部のソースファイルを統合する：」を参照してください。

コードの印刷も可能です。317 ページの「C コードを印刷する：」を参照してください。

コンディションやトランジションアクションの記述に使用するトランジションエディタには、ステートエディタと同様の機能が備えられています。

コンディションとトランジションアクションをトランジションエディタで記述する：

- トランジションを右クリックします。
- ショートカットメニューから **Edit Transition** を選択します。

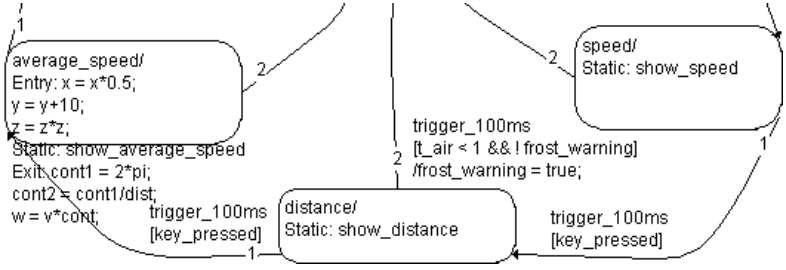
または

- トランジションをダブルクリックします。

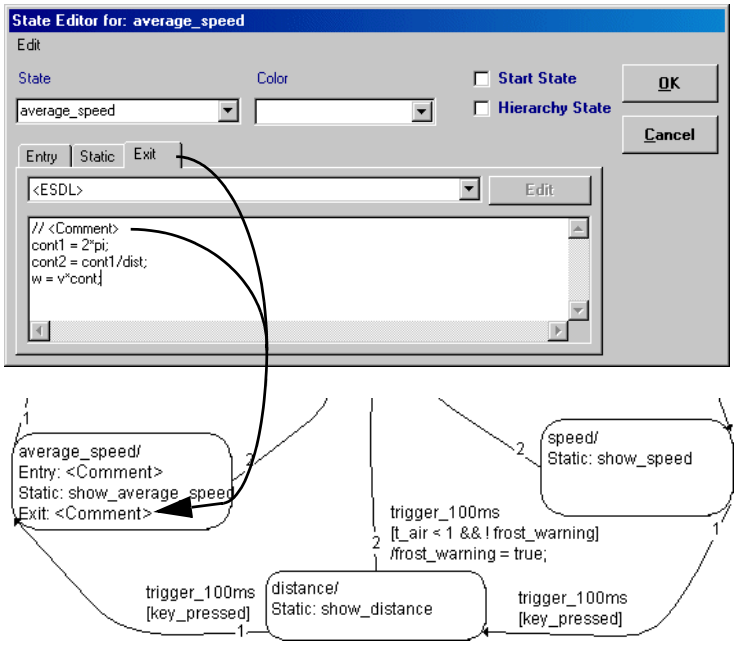
  トランジションエディタが開きます。
- “Condition”タブまたは“Action”タブ上で、コンボボックスから **<ESDL>** を選択します。

  コンボボックスの下の入力フィールドが有効になります。
- ESDL コードを入力フィールドに入力します。
- **OK** をクリックします。

ステートダイアグラム上には、ステートまたはトランジションに定義されたすべてのコンディションとアクションの ESDL コードが表示されます。そのため、複雑なステートマシンの場合は画面が煩雑になってしまいます。



このような場合、コンディションやアクションの 1 行目にコメントを入力することによって、ダイアグラムを簡潔にすることができます。1 行のみのコメントは行頭に // を付け、複数行の場合は /* <Comment> */ のように記述します。コメントを付けると、ダイアグラムにはコメントのみが表示されます。



コメント行は、後から自動的に追加することもできます。

自動的にコメント行を追加する：

- ステートマシンエディタでステートマシンを開きます。
- **Diagram → Create State Code Comments** を選択します。

  ステートダイアグラム内のすべてのコンディションとアクションについて、その 1 行目のコードと同じ内容のコメント行が、1 行目の前に挿入されます。

ステートマシンのコードが生成される際、ステートとトランジションに定義されたアクションとコンディションのコードは、独立したメソッド、つまりアウトラインコードとして生成されます。この「アウトライニング機能」については、『ASCET リファレンスガイド』の「ステートマシンの最適化」という項に説明されています。このアウトライニング機能は無効にすることもできます。

以下のようにして、アウトライニングを行うかどうかを切り替えます。

アクション／コンディションのアウトライニングのオン／オフを切り替える：

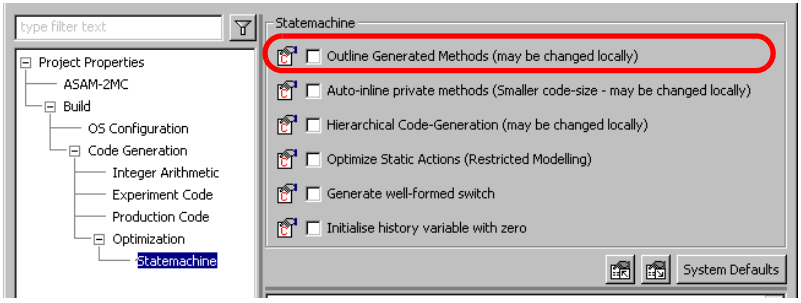
- ステートマシンを含むプロジェクトを開きます。

  デフォルトプロジェクトも使用できます。
- プロジェクトエディタで、“Project Properties”ダイアログボックスを開きます（401 ページ参照）

1. オフにする

- “Statemachine”ノード（414 ページ参照）で、**Outline Generated Methods (may be changed locally)** オプションをオフにします。

  プロジェクト内のすべてのステートマシンについて、アウトライニングが行われなくなります。



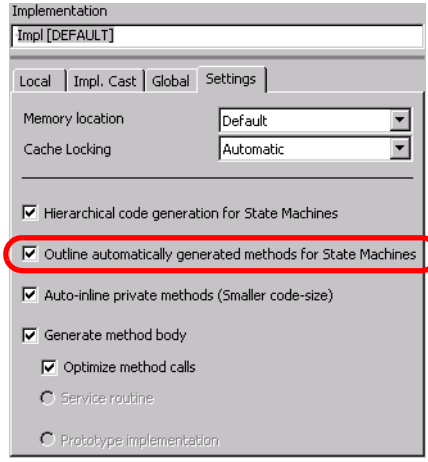
この場合、各ステートマシンごとのオン／オフ設定は無効となります。

2. オンにする

- “Statemachine”ノード（414 ページ参照）で、**Outline Generated Methods (may be changed locally)** オプションをオンにします。
- “Project Properties”ダイアログボックスを閉じます。

  アウトライニングを行うには、さらに、各ステートマシンごとに以下の設定を行う必要があります。
- プロジェクト内のステートマシンを開き、**Component → Edit Implementation** を選択してステートマシンのインプリメンテーションエディタを開きます。

- “Settings”タブで、**Outline Generated Methods (may be changed locally)** オプションをオンにします。



- インプリメンテーションエディタを閉じます。

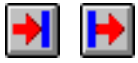
  上記の 2 つのオプションがオンになっている場合にのみ、アウトラインコードが生成されます。

  後者のオプションにより、個々のステートマシンについてアウトライニングが行われないようにすることができます。

他のコンポーネントとの通信

ステートマシンは、他の ASCET コンポーネントとの通信を行うことができます。このためには、入力および出力、トリガ引数、パブリックメソッド、といったオプションを利用します（『ASCET リファレンスガイド』の「クラスとしてのステートマシン」を参照してください）。

ステートマシンに入力や出力を追加する：



- **Input** または **Output** ボタンをクリックします。
- `ActionCondition` ダイアグラムがロードされた状態で、描画領域内の、入力または出力を配置したい位置をクリックします。

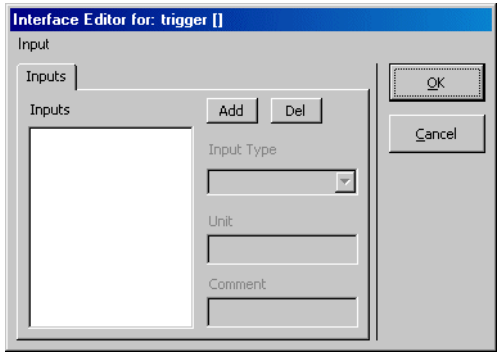
  入力または出力が作成されます。これとの接続を、引数や変数との接続と同じ方法で作成し、任意のシーケンスコールを割り当てることができます。

トリガ引数を使用して外部との通信を行うには、一連の手順が必要です。

まずは、1 つまたは複数のトリガ引数が必要です。トリガ引数をどのような場合にどこに定義すればよいかは、『ASCET リファレンスガイド』の「クラスとしてのステートマシン」の項に説明されています。

トリガ引数を追加する：

- 引数を追加したいトリガを、インターフェースエディタ（190 ページ参照）で開きます。



通常のメソッドのインターフェースとは異なり、タブは“Input”のみで、メニューも **Input** のみです。

- “Input”タブに、必要な引数を追加します（191 ページ参照）。

トリガ引数は、ステートダイアグラムのパブリックメソッドに属します。そのため、コンディションやアクションを ESDL コードで記述する際（292 ページの「コンディションとアクションをステートダイアグラムで定義する」を参照してください）、変数やパラメータと同じように使用することができます。

トリガ引数をブロックダイアグラムのアクションやコンディション内で使用する場合は、そのアクションやコンディションに、使用するトリガ引数と同じ型と名前を持つ引数を追加する必要があります。トリガ、およびアクション／コンディション内の引数は、名前と型によって結び付けられます。

コンディション／アクションに引数を追加する：

- “Diagram”ペインで、アクションまたはコンディションを選択します。
- インターフェースエディタを開きます。

  アクション／コンディションのインターフェースエディタは、通常のメソッドのものと同じです。

- “Arguments”タブで、引数を追加します。この際、名前と型がトリガ引数と一致するように注意してください。

  このようにして追加した引数により、パラメータと同様に接続します。
- “Return”タブでは、必要に応じてコンディションの戻り値の注釈を入力することもできます。

  アクションに戻り値を定義することはできますが、その値は、アクション実行時には無視されます。

4.2.4 パブリックメソッド

パブリックメソッドは、専用のダイアグラム内に定義することができます。これらのメソッドは、他のコンポーネントと同様、ステートマシン内からコールすることができます。

パブリックダイアグラムを作成する：

- **Diagram → Add Diagram → Public** を選択します。

または

- “Diagrams”ペインを右クリックして、ショートカットメニューから **Add Diagram → Public** を選択します。

  パブリックダイアグラムが作成されます。
- “Diagrams”ペインで、ダイアグラムの名前を入力します。
- “Diagrams”ペインで、ダイアグラムの名前をダブルクリックします。

  ダイアグラムが、新しいエディタウィンドウ上に開きます。

パブリックメソッドを追加する：

- “Diagrams”ペインで、パブリックダイアグラムを選択します。
- **Diagram → Add Method** を選択します。

または

- “Diagrams”ペインを右クリックして、ショートカットメニューから **Add Method** を選択します。

  新しいメソッドが作成されて、その名前が“Diagrams”ペインに表示されます。
- メソッドの名前を入力して <**Enter**> を押します。

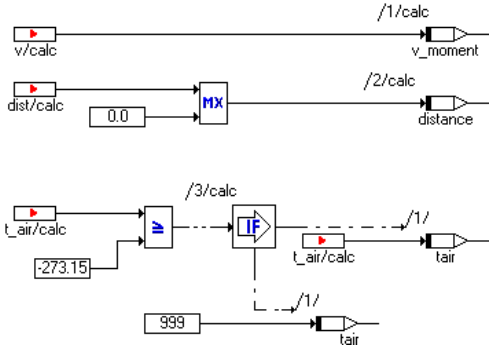
メソッドの定義方法は、4.1 章「ブロックダイアグラムエディタ」に説明されています。以下に具体例をあげますが、一部省略されている部分もありますのでご注意ください。

外部との通信にパブリックダイアグラムを使用する：

- パブリックダイアグラム内にパブリックメソッドを作成します。
- 191 ページに書かれている方法で、必要な引数を追加します。

  引数は、メソッド内のみで読み取り可能です。ステートマシン内で利用できるようにするためには、これらの値を変数に代入する必要があります。
- 必要な数の変数を追加します。
- 各引数を変数で接続します。

  変数は、ステートマシン内のどこでも使用できます。以下の図に、メソッド内の入力値を処理したりチェックする例を示します。

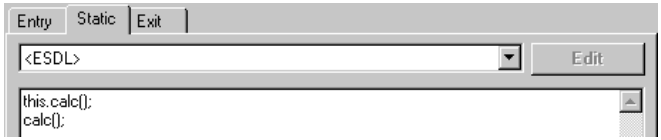


戻り値を持たないパブリックメソッドは、ステート／トランジションエディタの“Action”タブ上のコンボボックスに表示されます。また、戻り値を持つパブリックメソッドは、トランジションエディタの“Condition”タブ上のコンボボックスに表示されます。

パブリックメソッドをアクションおよびコンディションに使用する：

- 290 ページを参考にして、アクションにパブリックメソッドを割り当てます。
- 287 ページを参考にして、コンディションにパブリックメソッドを割り当てます。

アクションやコンディションを ESDL で記述する場合、インポートされたクラスのメソッドとまったく同じ方法で、ステートマシンのパブリックメソッドをコールすることができます。クラス名は、`this` または `self` を使用するか、または省略します。



パブリックメソッド `count` を上図で示されたいずれの方法でコールしても、同じ結果が得られます。

4.2.5 ステートマシンの検証

ステートマシンの検証も、他のコンポーネントと同様の方法で行うことができます。ステートマシン用に実験環境を起動するには、256 ページの「コンポーネントの検証」という項で説明されている方法と同様の手順を実行します。

ステートマシンに含まれるステートダイアグラム、または他のダイアグラムの解析は、以下のように行います。

ダイアグラムを解析する：

- 解析したいダイアグラムを描画領域にロードします。
- **Diagram → Analyze Diagram** を選択してダイアグラムを解析します。

  ダイアグラムに構文エラーがないかどうかがチェックされます。エラーが見つかると、モニタウィンドウの“Build”タブにその情報が出力されます（2.4「モニタウィンドウ」の項を参照してください）。

注記

ここでは、コンディションやアクションの記述に使用された ESDL は解析されません。

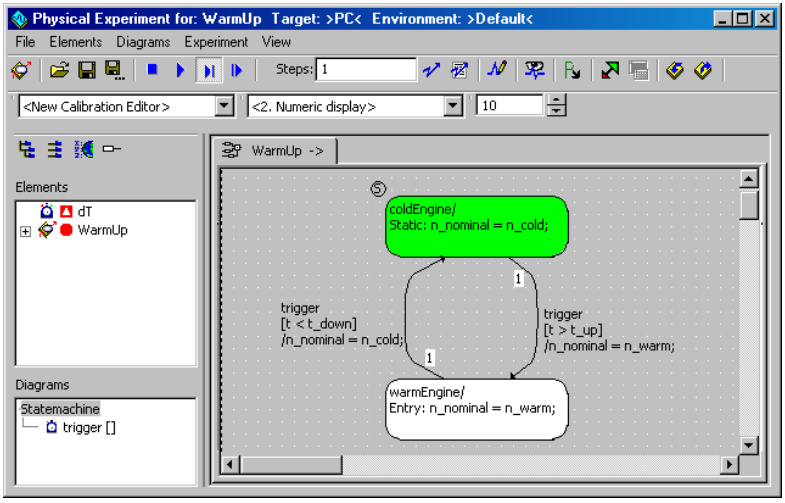
- “Build”タブに出力されたいずれかのエラーメッセージをクリックすると、ブロックダイアグラムエディタ内でそのエラーが発生した箇所が強調表示されます。

実験のセットアップも、すべての種類のコンポーネントと同じ方法で行うことができ、詳しくは 547 ページの「実験」で説明します。ステートマシンエディタには、他のエディタと同様の機能に加えて、ステートマシンのアニメーション機能があります。

ステートマシンのアニメーション機能を使用する：

- ステートマシンのオフライン実験を開始します。
- 必要に応じて実験をセットアップします。
- ステートダイアグラムに切り替えます。
- 任意のステートを右クリックします。
- ショートカットメニューから **Animate States** を選択します。
- シミュレーションを開始します。

  シミュレーションが実行され、ステートダイアグラム内において、現在のステートがアニメーションカラーで強調表示されます。



アニメーションカラーを変更するには、以下のように操作します。

アニメーションカラーを変更する：

- コンポーネントマネージャで、**Tools → Oprions を選択します。**
- “Options”ウィンドウで、“Fonts & Graphics”タブを選択します。

- “Color of Animated States”コンボボックスで、アニメーションカラーを選択します。
- OK をクリックします。

  選択された色で、ステートのアニメーション表示が行われます。アニメーションカラーの変更は、実験中でも行えます。

実験中に、以下のようにして個々のステートやトランジションの定義内容を確認することができます。

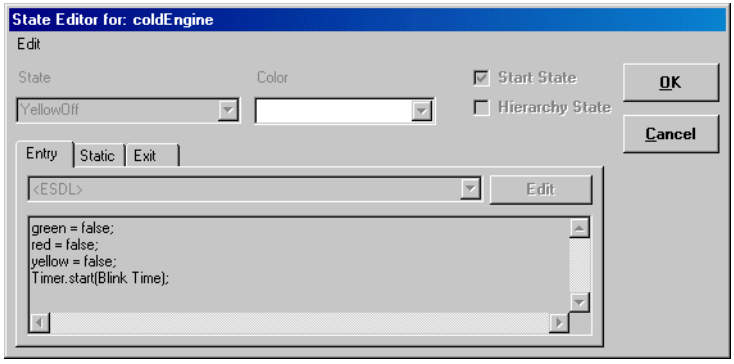
ステートやトランジションについての情報を表示する：

- ステートを右クリックし、ショートカットメニューから **State Info を選択します。**

または

- トランジションを右クリックし、ショートカットメニューから **Transition Info を選択します。**

  ステートエディタまたはトランジションエディタが開きます。この時、すべての編集機能は使用不可となっています。



  各タブ上で割り当てられたアクションやコンディション、および ESDL の記述内容を見ることができます。

- OK または Cancel をクリックしてウィンドウを閉じます。

また、実験中にステートマシンを初期ステートにリセットすることが可能です。

ステートマシンをリセットする：

- いずれかのステートを右クリックします。

- ショートカットメニューから **Reset Statemachine** を選択します。

  ステートマシンは開始ステートにリセットされます。

  これは、ステートマシン内の各変数の値には影響しません。実験を終了しない限り、開始ステートから通常の処理が継続されます。

4.3 C コードエディタ

C コードエディタを使用すれば、コンポーネント（クラス、モジュール、または連続系ブロック）を C 言語で定義することができます。その際は、クラスのコードを直接入力するか、または既存のプログラムを組み込んでそれを必要に応じて修正します。

本項では、C コードエディタでクラスやモジュールを定義する方法について説明します。C コードエディタは、ブロックダイアグラムエディタ（176 ページの「ブロックダイアグラムエディタ」を参照してください）と同じ原理に基づいた機能を持っています。

C コードコンポーネントを作成する：

- コンポーネントマネージャで、新しいコンポーネントを作成するフォルダを選択します。
- **Insert → Class → C Code**（または **Module → C Code**）を選択します。

または

- 以下のようにしてツールバーのボタンを使用します。
  - **Insert Class - <Type>** または **Insert Module - <Type>** ボタンの右側にある下向きの▼ボタンをクリックして、コンポーネントのデフォルトの記述形式を **C Code** にします。
  - **Insert Class - C Code** または **Insert Module - C Code** ボタンをクリックします。

  新しいコンポーネントが作成されます。
- コンポーネントの名前を入力して、**<Enter>** を押します。
- メニューバーから、**Component → Edit Item** を選択します。

または

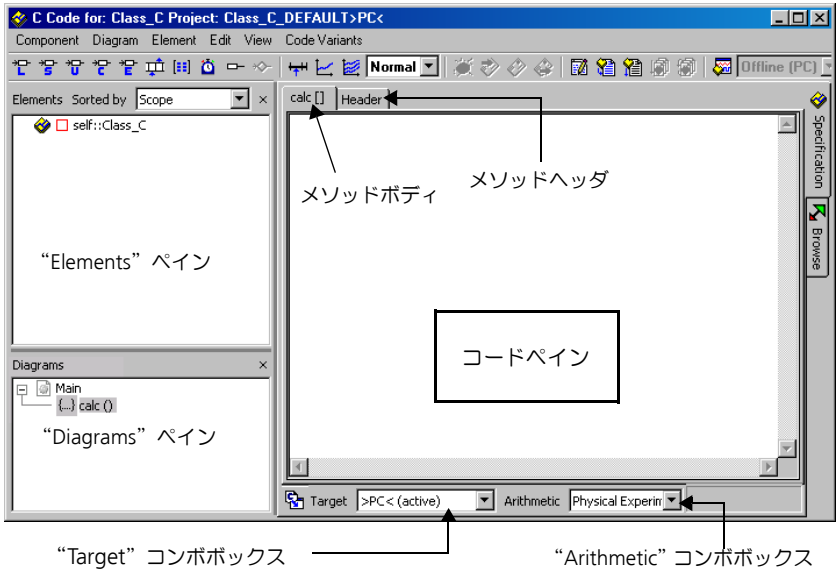
- **<Enter>** を押します。

または

- “1 Database”リスト内のコンポーネント名をダブルクリックします。

  新しいコンポーネント用の C コードエディタが開きます。

C コードエディタを下図に示します。重要な部分には注釈が付けられています。



“Elements”および“Diagrams”ペインの機能は、ブロックダイアグラムエディタの場合と同じです。ダイアグラム機能も使用できますが、コンポーネントのメソッドをパブリックメソッドとプライベートメソッドのグループにまとめるためだけの機能です。“Diagrams”ペインで C コードコンポーネントのメソッドまたはプロセスを選択すると、C コードペインに、そのメソッドまたはプロセスのボディとなるコードが表示されます。タイトルバーには、コンポーネントの名前、現在のプロジェクトおよびターゲットが表示されます。

“Target”コンボボックスには、現在のターゲットが表示されます。C コードコンポーネントには複数のターゲット用のコードを含めることができますが、コードは各ターゲット用に個別に定義されている必要があります。C コードはターゲットに依存して記述されるものであるため、あるターゲット用に開発された C コードをそのまま別のターゲットで実行することはできないからです。

1 つの各ターゲットについてもいくつかの異なる算術演算機構（例：浮動小数点と固定小数点のいずれを使用するか）を用いて実験を行う可能性があり、その場合も各算術演算機構用に個別の C コードを作成する必要が生じます。このような場合、1 度作成したコードを異なるターゲット／算術演算機構用にコピーし、それを適宜修正して使用することも可能です。

メソッドまたはプロセスのボディをコードペインに入力します。メソッドは C の関数に相当するので、複数の引数を入力し、1 つの戻り値を出力することができますが、プロセスには引数も戻り値も使用できません。関数の宣言文は ASCET が生成するので、ユーザーは関数のボディだけを定義します。

```
calc (a,b)
  {
     < ユーザーが C コードで処理を記述します >
  }
```

関数の引数と戻り値は、ブロックダイアグラムエディタの場合と同様に定義します。

システムライブラリを使用したい場合には、クラスまたはモジュールのヘッダに include 文を記述します。コードペインの“Header”タブをクリックすると、ヘッダが表示されます。メソッドのローカル変数も、ここに通常の C の文で宣言します。変数のスコープ（有効範囲）は、その変数が宣言されているメソッドまたはプロセスなので、1 つのメソッドまたはプロセスで宣言された変数を別のメソッドまたはプロセスから参照することはできません。

基本エレメントは、ブロックダイアグラムエディタの場合と同様に、エレメントボタンの 1 つをクリックして作成します。エレメントの機能は、ブロックダイアグラムで定義されたコンポーネント内のエレメントと同様で、そのコンポーネントのすべてのメソッドまたはプロセス内で使用することができ、それをグローバルエレメントにすることもできます。作成したエレメントの名前をコードペインに入力すれば、そのエレメントを参照することができます。

また、クラスの引数と戻り値、およびモジュールのメッセージも、ブロックダイアグラムコンポーネントの場合と同様に定義することができます。しかし、他のコンポーネントを参照するコンポーネントの場合には構文が異なるので、コンポーネントが他のコンポーネントを参照するかどうかを早い段階に決めておく必要があります。

メニューコマンドの概要

- **Component**
  - — *Clean code generation directory*

    コード生成ディレクトリ内のファイルをすべて削除します。
  - — *Touch*

    次回のコード生成時に強制的にコード生成が行なわれるようにします。
    → *Flat* — 編集対象コンポーネントのみ、
    → *Recursive* — 被参照コンポーネントも含める
  - — *Generate Code*

    コンポーネントのコードを生成します。
  - — *Open Experiment*

    実験を開始します。

— *Default Project*

コンポーネントの実験に用いるデフォルトプロジェクトの編集を行います。
→ *Edit* — デフォルトプロジェクト用のエディタを開きます
→ *Resolve Globals* — コンポーネント内のインポートエレメントのうち、それに対応するエクスポートエレメントがないものに対して、グローバルエレメントが生成されます。
→ *Delete Unused Globals* — 実際には使用されていないグローバルエレメントを削除します。

— *Edit Layout*

レイアウトエディタを開きます。

— *Edit Data*

コンポーネント用のデータエディタを開きます。コンポーネントデータを検索することができます。

— *Edit Implementation*

インプリメンテーションエディタを開きます。コンポーネントインプリメンテーションを検索することができます。

— *Edit Notes*

注釈エディタを開きます。このエディタでは、コンポーネントについての注釈を入力することができます。

— *Check Dependency*

仮パラメータのモデルパラメータへの割り当てが正しいかどうかを調べます。

— *Export Data*

コンポーネントデータセットをエクスポートします。

— *Show Path*

選択されている内部コンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *File out Generated Code*

生成されたコードをファイルに保存します。
→ *Flat* — 編集対象コンポーネントのみ、
→ *Recursive* — 被参照コンポーネントも含める

— *View Generated Code*

コンポーネント用のコードを生成し、それをテキストエディタで表示します。
テキストエディタは、ASCET オプションウィンドウの“ASCET Editor”ノード（60 ページ参照）

— *File out Generic Code For External Make*

生成されたコードを、外部ツール（Make/Buide 処理など）で処理するために、任意のディレクトリに保存します。

— *Exit*

C コードエディタを終了します。

- **Diagram**

注記

このメニューに含まれるコマンドは、“Diagrams”ペインのショートカットメニューからも実行できます。

— *Analyze Diagram*

現在アクティブになっているダイアグラムを分析します（C コードでは使用できません）。

— *Add Diagram*

新しいダイアグラムを作成します。
→ *Public* — パブリックメソッドのみを含むダイアグラム
→ *Private* — プライベートメソッドのみを含むダイアグラム

— *Rename Diagram*

ダイアグラムの名前を変更します。

— *Delete Diagram*

ダイアグラムをコンポーネントから削除します。

— *Add Method*

新しいメソッドを作成します。このコマンドはクラスとモジュールについてのみ有効です。

— *Add Process*

新しいプロセスを作成します。

— *Add Trigger / Action / Condition*

（ステートマシンエディタでのみ有効です）

— *Edit*

メソッド／プロセスのインターフェースを編集します。

— *Rename*

メソッド／プロセスの名前を変更します。

— *Delete*

メソッド／プロセスを削除します。

— *Move Up*

ダイアグラムまたはメソッド／プロセスを上方に移動します。

— *Move Down*

ダイアグラムまたはメソッド／プロセスを下方に移動します。

— *Move*

メソッド／プロセスを他のダイアグラムに移動します。

— *Edit Implementation*

メソッド／プロセスのインプリメンテーションを定義します。

- **Element**

> **注記**
>
> 以下のコマンドのうち、***Add Item***、***Rename***、***Edit*** コマンドは、ASCET-MD がインストールされている場合にのみ使用できます。このメニューに含まれるコマンドは、“Diagrams”ペインのショートカットメニューからも実行できます。

— *View → Collapse all*

“Elements”ペイン内のツリー構造をすべて格納し、コンポーネントのみが表示されるようにします。

— *View → Expand all*

ツリー構造を展開し、各コンポーネントの内容すべてが表示されるようにします。

— *Add Item*

コンポーネントを複合エレメントとして組み込みます。

— *Rename*（**<F2>**）

選択されているダイアグラムエレメントの名前を変更します。

— *Delete*（**<Del>**）

選択されているダイアグラムエレメントを削除します。

— *Show Occurrences*

エレメントのすべての「オカレンス」を表示します。「オカレンス」とは、ダイアグラム上に配置されたダイアグラムアイテムを指します。（ブロックダイアグラムエディタでのみ有効）

— *Edit*

選択されているエレメントのプロパティを編集します。

— *Edit Data*

選択されているエレメントのデータを編集します。

— *Edit Implementation*

選択されているエレメントのインプリメンテーションを編集します。

— *Set Cache Locking*

このコマンドは ASCET-RP 用のものです。詳しくは ASCET-RP のユーザーズガイドを参照してください。

— *Edit Component*

選択されている内部コンポーネント（現在編集中のコンポーネントに内包されるコンポーネント）のコンポーネントエディタを開きます。

— *Replace Component*

コンポーネントを別のコンポーネントで置き換えます。コンポーネント名は古いコンポーネントのものがそのまま使われます。

— *Show Path*

選択されている内部コンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *Notes*

選択されている内部コンポーネントの注釈エディタを開きます。

— *Edit Distribution*

選択されているディストリビューション用のエディタを開きます。（グループカーブ／マップの場合のみ有効）

— *Edit Max Size*

配列、マトリックス、特性カーブ／マップの最大サイズを定義するためのエディタを開きます。

— *Copy Elements*（**<Ctrl> + <C>**）

“Elements”ペインから、選択されている 1 つまたは複数のエレメントをコピーします。

— *Paste Elements*（**<Ctrl> + <V>**）

コピーされた 1 つまたは複数のエレメントを“Elements”ペインにペーストします。

— *File Out Data*

配列、マトリックス、特性カーブ／マップからファイルにデータを書き出します。

— *File In Data*

配列、マトリックス、特性カーブ／マップのデータをファイルから読み込みます。

— *Export Data*

内部コンポーネントからデータセットをエクスポートします。

- **Edit**

— *Element / Data / Implementation*

それぞれ、ショートカットメニューの **Edit** / **Edit Data** / **Edit Implementation** コマンドに対応します。

— *Cut*（**<Ctrl> + <X>**）

ダイアグラムアイテムをカット（削除）します。

— *Copy*（**<Ctrl> + <C>**）

ダイアグラムアイテムをコピーします。

— *Paste*（**<Ctrl> + <V>**）

ダイアグラムアイテムをペーストします。

— *Find/Replace*（**<Ctrl> + <F>**）

タブ上のテキストの検索／置換を行います。

— *Select All*（**<Ctrl> + <A>**）

タブ上のテキスト全体を選択します。

— *Load from file*

ファイルからインポート／挿入します。

— *Store to file*

コードを外部ファイルに書き込みます。

— *Print*

現在のコードを印刷します。

— *Printer Setup*

プリンタの選択とセットアップを行う“Printer Selection”ダイアログボックスを開きます。

— *Save* （**<Ctrl> + <S>**）

メソッドまたはプロセスを保存します。

- **View**

  — *Show/Hide*

  エディタまたはコンポーネントの一部の表示／非表示を切り替えます。

  →*Elementt List* —“Elements”ペイン
  →*Diagram List* —“Diagrams”ペイン

- **Code Variants**

  — *Copy C-Code to*

  選択されたターゲット／算術演算機構の組み合わせに対して、コードをコピーします。この際、コードの内容は変更されません。

  — *Copy C-Code from*

  ターゲット／算術演算機構の組み合わせを現在の環境用にコピーします。コードの内容は変更されません。

4.3.1 コンポーネントを C コードで定義する

C コードで記述されるコンポーネントには、`calc` という名前のデフォルトのメソッドが 1 つ作成されます。メソッドおよびプロセスの作成やそれらのインターフェースの定義は、ブロックダイアグラムエディタの場合と同じように行います（190 ページの「コンポーネントのインターフェースの定義」を参照してください）また基本エレメント（205 ページ参照）や列挙型データ（206 ページ参照）の作成、コンポーネントを複合エレメントとして組み込む（215 ページ参照）方法、および注釈（218 ページ参照）の編集も、ブロックダイアグラムの場合と同様に行えます。

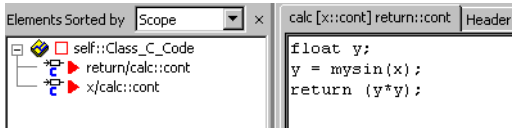
メソッド／プロセスを定義する：

- “Diagrams”ペインから、定義したいメソッドまたはプロセスを選択します。

  メソッドの名前がコードペインの上に表示されます。

- “Target”コンボボックスからターゲットを選択します。

  ここでは、ASCET システムにインストールされているすべてのターゲットから選択することができます。
- 入力するコードに使用する算術演算機構を“Arithmetic”コンボボックスから選択します。
- “Header”タブをクリックします。
- クラスまたはメソッドのヘッダ情報（ローカル変数、include 文、マクロなど）を入力します。
- “<method name>”タブをクリックします。
- メソッドのコードを入力します。



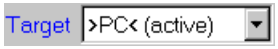
  “Elements”ペインに定義されている基本エレメントは、その名前を入力するだけで参照することができます。ただし、そのコンポーネント内で被参照コンポーネントが使われていない場合に限ります。
- すべてのメソッドまたはプロセスについて、以上を繰り返します。

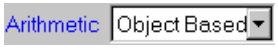
C コードは実行環境に依存するため、各ターゲットの各算術演算機構ごとに 1 つの C コードコンポーネントを定義する必要がありますが、1 つのターゲット／算術演算機構用に開発されたコードをコピーして、それを別のターゲット／算術演算機構用に修正すれば、大幅に工数を削減できます。

1 つのクラス／モジュールの C コードをコピーする：

ある 1 つのクラス／モジュール用の C コードを、他のターゲット、または異なる実験タイプやインプリメンテーションにコピーするには、以下のいずれかの方法で行います。

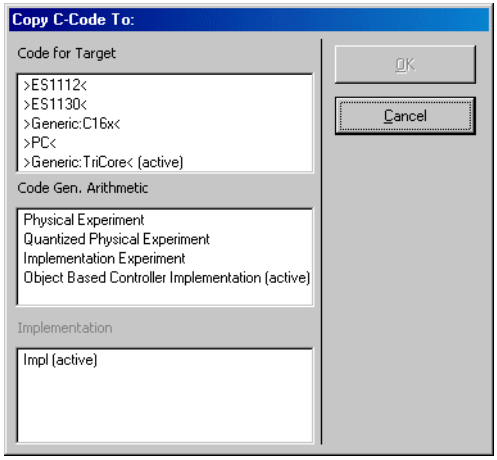
1. メニューコマンド **Code Variants → Copy C-Code To** を使用する：
   - C コードエディタでクラスまたはモジュールを開きます。
   - “Target”コンボボックスで、C コードが記述されたときのターゲットを選択します。
   - “Arithmetic”コンボボックスで、C コードが記述されたときの実験タイプを選択します。





- “Implementation”コンボボックスで、C コードが記述されたときのインプリメンテーションを選択します。
- **Code Variants → Copy C-Code To**を選択します。

  “Copy C-Code to”ダイアログボックスが開きます。



- “Code for Target”ペインからターゲットを選択します。
- “Code Gen. Arithmetic”ペインから実験タイプを選択します。

  すべての実験にインプリメンテーションを選択できます。定義されているすべてのインプリメンテーションが、“Implementation”ペインに表示されます。
- インプリメンテーションを選択します。

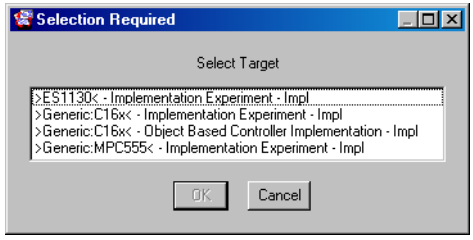
  すべての選択を終了すると、**OK** ボタンが有効になります。
- OK をクリックします。

  C コードが、選択された環境用にコピーされます。

2. メニューコマンド **Code Variants → Copy C-Code From** を使用する：
   - C コードエディタの“Target”、“Arithmetic”、“Implementation”コンボボックスで、C コードを使用する条件を設定します。

- **Code Variants → Copy C-Code From** を選択します。

  “Selection Required”ダイアログボックスが開きます。



- ターゲット、実験タイプ、インプリメンテーションを選択して **OK** をクリックします。

  C コードが、現在使用している環境用にコピーされます。

メソッド／プロセス用のコードを編集する：

- 編集したいコード部分を強調表示します。

または

- **Edit → Select All** を選択すると、現在のメソッドまたはプロセスのコード全体が強調表示されます。
- **Edit → Cut** を選択すると、強調表示されているコードが削除され、クリップボードに格納されます。
- **Edit → Copy** を選択すると、強調表示されているコードがクリップボードにコピーされます。
- **Edit → Paste** を選択すると、クリップボードのコードがペーストされます。

カットまたはコピーされたテキストは Windows のクリップボードに格納されるので、他のアプリケーション（C 言語システム開発ツールなど）と交換することができます。また、メソッドやプロセスの間で、またはコンポーネント間でコードを交換することもできます。

C コードテキストの検索と置換を行う：

- **Edit → Find/Replace** を選択します。

  “Find/Replace”ダイアログボックスが開きます。

- 検索したいテキストを、“Find”フィールドに入力します。

  ここではワイルドカードを使用できます。# は任意の 1 文字、* は任意の文字列を表します。
- 検索テキストを置き換えたいテキストを、“Replace With”フィールドに入力します。
- **Find Next** ボタンをクリックして、検索テキストに一致するテキストを検索します。
- 検索テキストに一致するテキストが強調表示されます。一致するテキストが見つからない場合には、ダイアログボックスの一番下にあるステータス行に `String Not Found` というメッセージが表示されます。
- **Replace Selection** ボタンをクリックすると、強調表示されているテキストが“Replace With”フィールドのテキストに変わります。

  このボタンは、一致するテキストが見つかった場合に限り使用できます。
- **Replace/Find** ボタンをクリックすると、強調表示されているテキストが置き換えられ、検索テキストに一致する、次のテキストが見つけだされます。
- **Replace All** ボタンをクリックすると、検索テキストと一致するテキストがすべて一括して置き換えられます。

- **Close** ボタンをクリックすると、“Find/Replace” ダイアログボックスが閉じます。

検索条件をカスタマイズする：

- **Forward** オプションをオンにすると、カーソルポイントからテキストの最後部に向かって検索が行われます。

または

- **Backward** オプションをオンにすると、カーソルポイントからテキストの先頭方向に検索が行われます。
- **Case Sensitive** をオンにすると、検索時に大文字と小文字が区別されます。
- **Wrap Search** をオンにすると、テキスト全体が検索されます。

  このオプションがオンになっていると、検索は現在のカーソル位置から指定方向に進められ、テキストの終わり（または始め）まで行われた後、始め（または終わり）からカーソル位置まで行われます。

メソッド／プロセスを保存する：

- **Edit → Save** を選択します。

または

- **<Ctrl> + <S>** を押します。

  現在のメソッドまたはプロセスが保存されます。

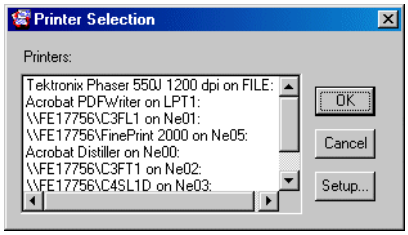
  このコマンドは、現在のメソッドまたはプロセスが前回の保存以降に変更された場合に限り使用できます。変更を加えたメソッドを保存しないで他のメソッドに切り替えようとすると、現在のメソッドを保存するかどうかを尋ねるメッセージが表示されます。

C コードを印刷する：

- **Edit → Print** を選択すると、コードをデフォルトプリンタに印刷することができます。

- プリンタの選択を行うには、**Edit → Print Setup** を使用します。

  “Printer Selection”ダイアログボックスが開きます。



- “Printers”フィールド内のプリンタを選択します。
- **Setup** をクリックしてプリンタ設定ダイアログボックスを開きます。
- プリンタ設定を調整します。
- **OK** をクリックして選択を確定します。

  “Printer Font Selection”ダイアログボックスが開きます。

- 使用するフォントを設定します。
- **OK** をクリックして選択を確定します。

4.3.2 外部エディタ

C コードエディタの機能は限定されていて、単純な操作しか行えないため、任意の外部エディタ（例、Notepad、Codewright など）を使用してコードを編集することができます。外部エディタを使用するには、Windows で“*.c”や“*.h”というファイル拡張子をそのエディタに関連付けておく必要があります。エディタが関連付けられていないと ASCET から外部エディタを開くことはできないため、エラーメッセージが表示されます。

外部エディタ起動用のボタンをクリックすると、C コードエディタの表示が上下の部分に分かれ、上の部分にはコンポーネントで使用可能なすべてのメソッドまたはプロセスが一覧表示され、下の部分には C コードが表示されます。

外部エディタが開かれると、記述されるメソッドまたはプロセスのコードは 1 つまたは複数のファイルに書き込まれてテンポラリディレクトリに保存されます。編集が終わったら、その内容を ASCET 環境に戻すために、まずファイルを外部エディタで保存する必要があります。ASCET から外部エディタを閉じると、一時ファイルが読み取られて削除され、保存されていたコードの内容が外部エディタから ASCET 環境に書き込まれます。その後は、外部エディタ環境でコードに変更を加えても、それを ASCET に戻すことはできません。コードを再度編集する必要がある場合は、もう一度外部エディタを ASCET 環境から起動する必要があります。

注記

外部エディタでコードに加えた変更を ASCET 環境に戻すには、まずコードを外部エディタで保存する必要があります。

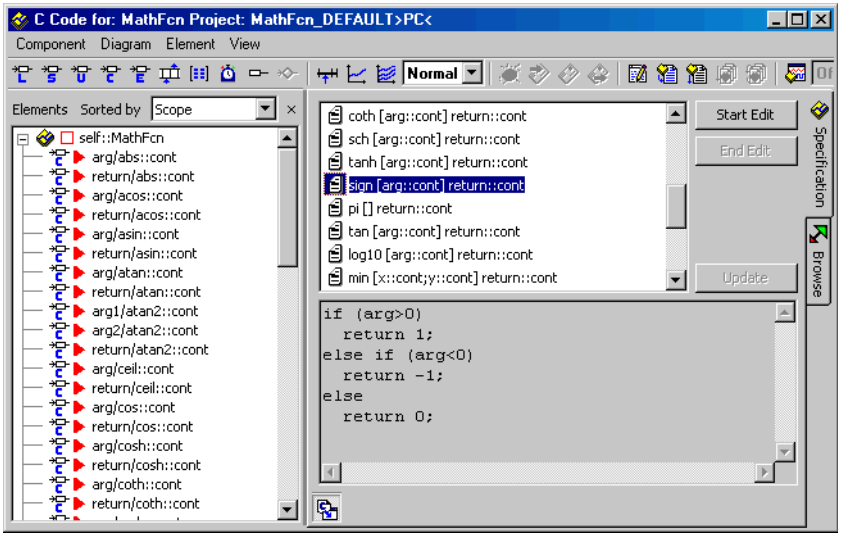
外部エディタを開く：

- エディタウィンドウ最下部の **Activate External Editor** ボタンをクリックします。

  C コードエディタの表示が変わります。“Diagrams”ペインが表示されなくなり、コードペインの変わりに選択フィールドとテキストフィールドが表示されます。定義されているすべてのメソッドまたはプロセスの一覧が、上側の選択フィールドに表示されます。

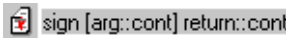
- 選択フィールドから、コードを変更したいメソッドまたはプロセスを選択します。

  選択されたメソッドまたはプロセスのコードが下側のテキストフィールドに表示されるので、内容を確認できます。



- **Start Edit** ボタンをクリックすると、外部エディタが開き、強調表示されているメソッドまたはプロセスのコードが表示されます。

  外部エディタが開くと、上側の選択フィールドではそのメソッドまたはプロセスのシンボルに赤い印が付きます。



- 外部エディタでコードを編集します。

外部エディタを閉じる：

- 編集したコードを、外部ディタで保存します。
- C コードエディタで **Update** ボタンをクリックします。

  外部エディタで編集したコードが ASCET 環境に転送され、テキストフィールドに表示されます。

  この操作を行っても、外部エディタへの接続は終了しません。つまり、この後も外部エディタでの作業を続け、その内容を ASCET 環境に転送することができます。

- 外部エディタでの作業を終了するには、**End Edit** ボタンをクリックします。

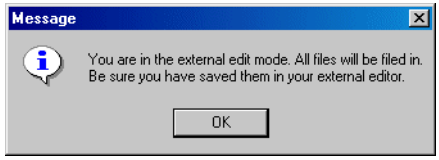
  外部エディタへの接続が終了し、メソッドまたはプロセスのシンボルに付いていた赤い印が消えます。

  この操作を行っても外部エディタは閉じませんが、これより後に外部エディタで編集された内容を ASCET に取り込むことはできません。
- 外部エディタを閉じます。

外部エディタモードを終了する：

- **Activate External Editor** ボタンをクリックします。

  この前に **End Edit** 操作を行っていないと、以下の警告メッセージが表示されます。外部エディタでの編集内容が ASCET 環境に読み込まれることをユーザーに知らせるためのものです。



  — 必要に応じて、外部エディタで編集したファイルを保存し、最新の内容が ASCET に反映されるようにします。

  — **OK** をクリックしてダイアログボックスを閉じます。

  外部エディタモードが終了し、C コードエディタが通常の表示内容に戻ります。

4.3.3 外部ソースエディタの使用法

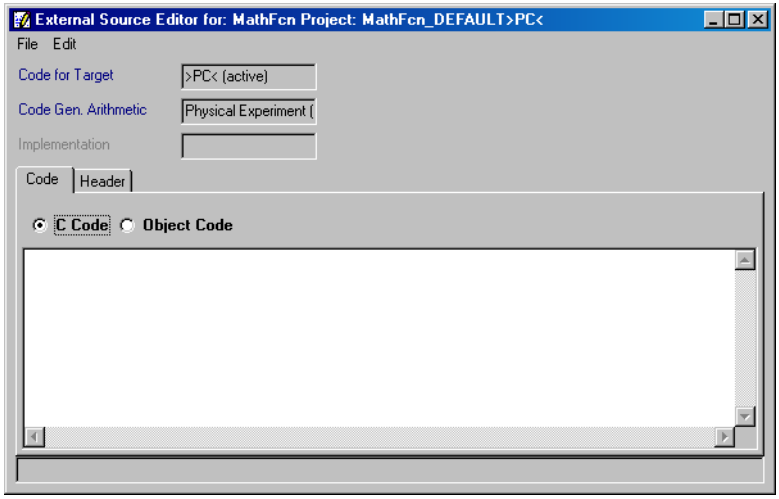
外部ソースエディタを使えば、既存の C プログラムを ASCET の C コードコンポーネントの一部として利用することができます。C のソースコードとオブジェクトコードのどちらを組み込むこともできます。このためには、外部ソースファイル（ソースモジュールとオブジェクトモジュール）に入っているすべての関数を、コンポーネントの一部であるヘッダファイルで宣言します。

外部モジュールも、各ターゲットおよび各算術演算機構ごとに別々に記述する必要があります。ターゲットおよび算術演算機構の選択は、前述の方法で行ってください。

外部ソースエディタを開く：

- **External Source Editor**ボタンをクリックします。

  外部ソースエディタが開きます。



外部のソースファイルを統合する：

- “Code”タブをクリックします。

  1 つのコンポーネントに統合できる外部ソースファイルは 1 つだけです。外部ソースファイルは C コードモジュールでもオブジェクトコードモジュールでもかまいません。

- **C Code**または**Object Code**オプションをクリックします。
- 外部モジュールの関数の内容を入力します。（**C Code** オプションの場合のみ）

または

- **Edit → Load from File** を選択します。

  “File Selection”ダイアログボックスが開きます。

- 外部 C コードまたはオブジェクトモジュールを選択します。
- **Open** をクリックします。

  選択されたファイルの内容がテキストペインに書き込まれます。元のファイルの内容は影響を受けません。

- **Edit → Store to File** をクリックします。

  “File Selection”ダイアログボックスが開きます。テキストペインの内容が選択されたファイルに書き込まれます。
- **Edit → Save** を選択すると、外部モジュールが保存されます。

外部ソースエディタでは、C コードエディタのものと同じ編集コマンドを使用できます。

外部クラスに外部ソースモジュールが含まれている場合、ソースモジュール内の関数を宣言するヘッダファイルも必要です。

ソースモジュールにヘッダファイルを追加する：

- “Header”タブをクリックします。

  ヘッダファイルは、ソースモジュールの前後どちらに追加してもかまいません。
- ヘッダファイルのテキストを入力するか、前述の方法で外部ソースからロードします。
- **Use header globally** オプションをオンにすると、同じヘッダファイルがすべての C コードファイルに使用されます。

外部モジュールをテストする：

- **File → Compile Code** を選択します。

  このコマンドにより、現在選択されているコンパイラが起動され、外部モジュール内の `.c` および `.h` ファイルがコンパイルされます。これはソースコードの構文が正しいかどうかを確かめるためだけに行われる処理なので、実行形式のコードは生成されません。構文エラーが見つかった場合は、コンパイラからエラーメッセージが出力されます。

4.4 ESDL エディタ

コンポーネントを ESDL コードで定義するには、ESDL エディタを使用します。ESDL コンポーネントはクラス、モジュール、または連続系ブロックのいずれにも使用できます。ESDL コードのクラスは、ブロックダイアグラムで定義されたクラスと同様に機能し、また ESDL のクラスが別のコンポーネントにより使用される場合も、ブロックダイアグラムのクラスと同様に扱われます。

本項では、クラスやモジュールを ESDL コードで作成する方法について説明します。本項を読む際は、176 ページの「ブロックダイアグラムエディタ」という項で説明されているブロックダイアグラムでのクラスの作成方法について理解していることが必要です。また ESDL 言語については、『ASCET リファレンスガイド』の「ESDL によるボディ記述」という項で説明されています。

クラスを ESDL コードで作成する：

- コンポーネントマネージャで、新しいコンポーネントを作成するフォルダを選択します。
- **Insert → Class → ESDL**（または **Module → ESDL**）を選択します。

または

- 以下のようにしてツールバーのボタンを使用します。
  - **Insert Class - <Type>** または **Insert Module - <Type>** ボタンの右側にある下向きの▼ボタンをクリックして、コンポーネントのデフォルトの記述形式を **ESDL** にします。
  - **Insert Class - ESDL** または **Insert Module - ESDL** ボタンをクリックします。

  新しいコンポーネントが作成されます。
- コンポーネントの名前を入力して、**<Enter>** を押します。
- メニューバーから、**Component → Edit Item** を選択します。

または

- **<Enter>** を押します。

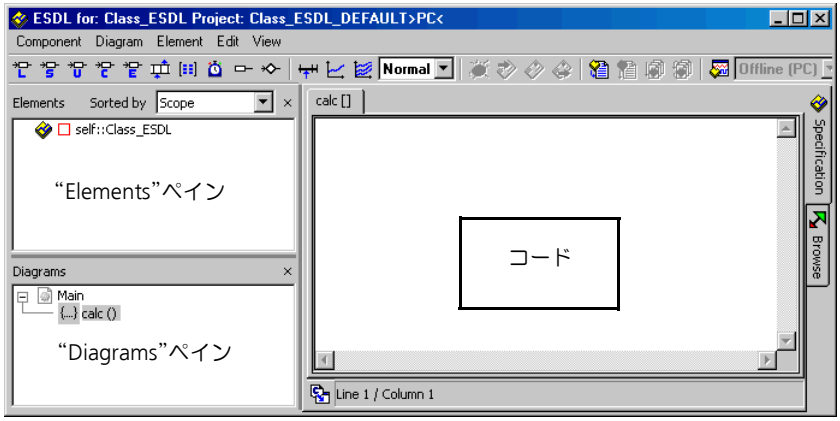
または

- “1 Database”リスト内のコンポーネント名をダブルクリックします。

  新しいコンポーネント用の ESDL エディタが開きます。

ブロックダイアグラムエディタの場合と同様に、ESDL コードで定義されたコンポーネントにもメソッドやプロセスが含まれ、その中にさまざまなエレメントを定義することができます。ブロックダイアグラムと同じ種類のエレメントを使用でき、インターフェースやメソッドを定義する方法も同じです。「ダイアグラム」

の機能も含まれますが、これはコンポーネントのメソッドをパブリックメソッドやプライベートメソッドのグループにまとめる目的にのみ使用されます。クラスのESDL コードは、ESDL エディタのコードペインに、1 メソッドずつ表示されます。



ESDL エディタのメニューバーには、C コードエディタ（306 ページ参照）と同じメニューコマンドと、さらに **Diagram → Analyze Diagram** および **View → Show/Hide → Impl. Casts in Element List** が含まれています。

4.4.1 ESDL でのクラスの定義

ESDL コードを作成する：

- 190 ページの「コンポーネントのインターフェースの定義」に説明されている方法で、必要なメソッドを追加します。
- “Diagrams” ペインからメソッドを選択します。
- エレメントボタンをクリックしてエレメントの名前を入力し、希望のエレメントを作成します。
- メソッドのコードをコードペインに入力します。

  C コードエディタと同様、外部エディタを使用することもできます（4.3.2 項参照）。
- 現在のメソッドのコードを保存するには、**Edit → Save** を選択します。

または

- **<Ctrl> + <S>** を押します。

  別のメソッドに切り替えようとするときにも、保存するかどうかが必ず尋ねられます。

テキストで機能記述を行った場合、命令文の処理順序はシーケンスコールではなくソースコード内での記述順序により決まります。エレメントやその他のメソッドは、コード内ではその名前により参照されます。

ESDL コード内でインプリメンテーションキャストを使用するには、以下のようにボタンを使用してインプリメンテーションキャストを“Elements”リストに追加し、その名前をコード内で用います。

ESDL 内でインプリメンテーションキャストを使用する：

- **Implementation cast** ボタンで、必要な数のインプリメンテーションキャストを追加します。
- 『ASCET リファレンスガイド』の「ESDL で使用されるインプリメンテーションキャスト」の項に説明されている規則に基づいて、インプリメンテーションキャストを ESDL コード内に追加します。規則の概要は以下のとおりです。
  - インプリメンテーションキャストは、その名前を使用して参照されます。
  - インプリメンテーションキャストは括弧で囲みます。
  - インプリメンテーションキャストは、それが参照するエレメントの直前に記述します。
  - インプリメンテーションキャストが演算の結果を参照する場合、この演算を括弧で囲みます。その演算は、他の演算の一部であってもかまいません。

注記

論理変数または論理式にインプリメンテーションキャストを使用すると、コード生成時にエラーが発生します。

“Elements”リスト内のインプリメンテーションキャストは、ダイアグラムエディタの場合と同様、**View → Show/Hide → Impl. Casts in Elements List** コマンドで表示／非表示を切り替えることができます（247 ページ参照）。

以下のようにして、メソッド間、またはメソッド内で ESDL コードをカット、コピー、およびペーストすることができます。

ESDL コードを編集する：

- 編集したいコードを選択します。

または

- **Edit → Select All** を選択すると、現在のメソッドのコード全体が選択されます。

- **Edit** → **Cut** を選択すると、強調表示されているテキストがカットされ、また **Edit** → **Copy** を選択すると、強調表示されているテキストがクリップボードにコピーされます。
- テキストを追加したい位置をクリックします。
- **Edit** → **Paste** を選択すると、クリップボード上のテキストがペーストされます。

ESDL コードの検索と置換を行う：

ESDL エディタでは、C コードエディタと同様の高度な検索／置換機能を利用できます。詳細は、316 ページの「C コードテキストの検索と置換を行う：」という項を参照してください。

また、C コードと同様に ESDL コードも印刷することができます（317 ページの「C コードを印刷する：」を参照してください）。

4.4.2 ESDL でのモジュールの定義

モジュールを ESDL で定義する方法は、クラスを定義する方法と同じですが、異なる点は、メソッドではなくプロセスを定義する点です。また、モジュールにはメッセージを定義することができます。ESDL エディタの **message** ボタンは、クラスを定義するときにはグレイアウトされて無効になり、モジュールを定義するときにのみ使用可能になります。

4.4.3 ESDL コンポーネントの検証

ESDL コンポーネントを作成した後は、ブロックダイアグラムの場合と同じように実験を行って動作を検証することができます。詳しい方法は、256 ページの「コンポーネントの検証」を参照してください。

注記

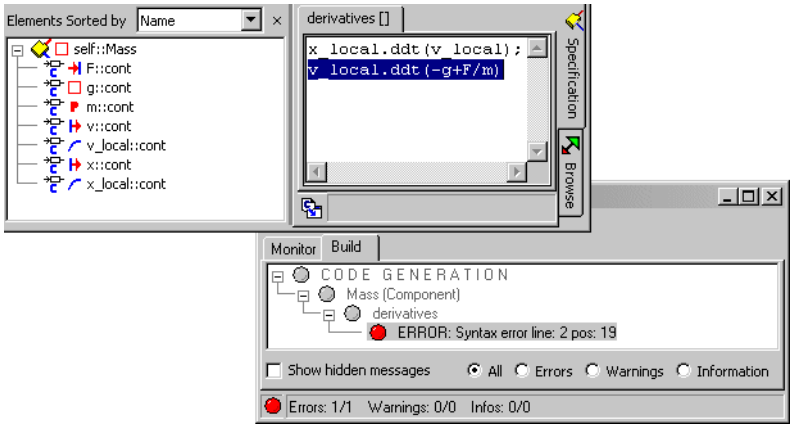
ASCET の個々のモジュールに関して、プロジェクトなしで単体でコード生成とシミュレーションを行えるのは、物理実験の場合のみです。他の条件での実験を行う場合、モジュールは必ずプロジェクトに組み込まれている必要があります。このため、各クラスやモジュールには、デフォルトプロジェクトと呼ばれるダミーのプロジェクトが定義され、これを利用しないとインプリメンテーション（実装情報）にアクセスできません。プロジェクトが存在しないと、インポートされるエレメントの変換式やインプリメンテーションは失われてしまいます。

ESDL コンポーネントを解析する：

- **Diagram** → **Analyze Diagram** で、現在アクティブな ESDL コンポーネントが解析されます。

  ESDL のシンタックスエラーが、エラーがあった場合は、ASCET モニタウィンドウに表示されます。

- エラーメッセージをクリックすると、ESDL コードの該当個所が表示されます。



4.5 連続系ブロックの定義

「連続系ブロック」（以下、「CT ブロック」とも記します）は、ASCET の組み込みソフトウェアにより制御される対象となる物理プロセスモデルを定義するために使用されます。これを利用すれば、ASCET 内で制御ループ全体をモデリングしてテストすることができます。連続系ブロックについては、『ASCET リファレンスガイド』の「連続系」の項に詳述されています。

連続系ブロックは、他のコンポーネントと同様に、ブロックダイアグラム、C コード、または ESDL コードで定義することができます。しかし、連続系ブロックをブロックダイアグラムで定義した場合とコードで定義した場合とでは、機能が異なります。

たとえばホイール、ブレーキ、油圧管、といった基本的な物理コンポーネントは、「連続系基本ブロック」としてコードで定義します。一般的に微分方程式を使用して定義されるこのようなコンポーネントモデルは、ブロックダイアグラムよりもコードの方が簡単に記述できるためです。そしてそれらのコンポーネントが、ブロックダイアグラムで記述された連続系構造ブロックに組み込まれ、1 つの部品となります。

連続系ブロックから標準の ASCET クラスを参照することはできますが、これは、それらのクラスがレコードつまり、複合変数として使用される場合に限り有効です。標準クラスとして定義されているアルゴリズムを連続系ブロックで使用することはできません。

本項では、ブロックダイアグラムやコードで連続系ブロックを定義する方法について説明します。使用するエディタは標準の ASCET クラス用のエディタと同じですので、ここでは標準クラスの場合と異なる点だけについて説明します。そのた

め、標準クラスの定義の方法について理解されていることが前提となります（176 ページの「ブロックダイアグラムエディタ」、304 ページの「C コードエディタ」、および 323 ページの「ESDL エディタ」を参照してください）。

4.5.1 ブロックダイアグラムによる連続系ブロックの定義

CT ブロックを定義する：

- コンポーネントマネージャの“1 Database”リストから、新しいブロックを作成するフォルダを選択します。
- **Insert → Continuous Time Block → C Code** を選択します。
- コンポーネントの名前を入力して **<Enter>** を押します。
- メニューバーから **Component → Edit Item** を選択します。

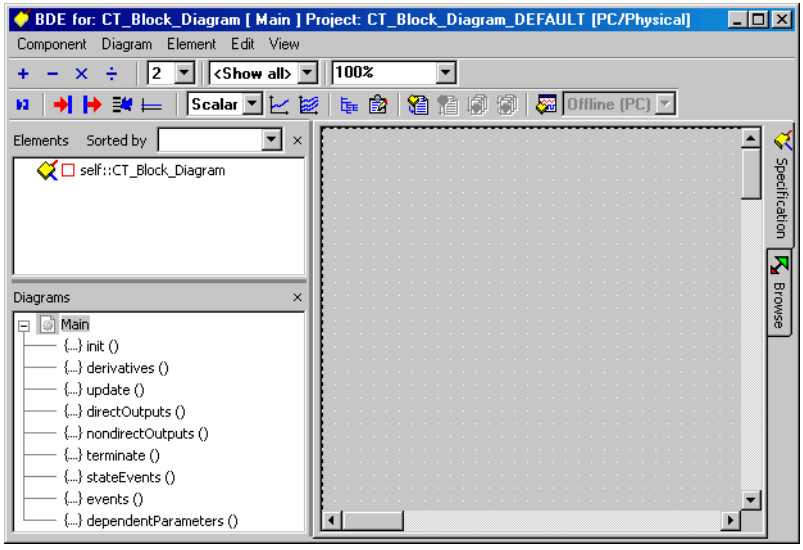
または

- **<Enter>** を押します。

または

- コンポーネント名をダブルクリックします。

CT ブロック用のブロックダイアグラムエディタが開きます。



前述のように、ブロックダイアグラム上では、複数の基本ブロックなどが接続されて CT ブロックとして定義されます。ダイアグラムによる定義のしかたは通常のクラスを定義する場合とよく似ていますが、以下のような重要な相違があります。

- 使用できる演算子は、4 種類（加算、除算、乗算、除算）のみです。より複雑な非線形演算子が必要な場合には、別のブロックで記述してください。
- CT ブロック内で定義できる基本エレメントは、特性カーブとフィールドだけはクラスの場合と同じですが、それ以外は異なります。CT ブロックで使用できる基本エレメントについては、『ASCET リファレンスガイド』の「ブロックインターフェース」の項を参照してください。
- 1 つの CT ブロック内にはダイアグラムを 1 つしか持たせることができないので、複数のダイアグラムを含めることはできません。ただし、ダイアグラムを階層構造にすることはできます。
- インターフェースを定義する必要はありません。CT ブロックが作成された時点で、一連のメソッドが自動的に定義されます。これらのメソッドに変更を加えたり、追加のメソッドを定義することはできません。
- CT ブロックから参照できるのは、クラスと他の CT ブロックだけです。クラスはレコードの定義を行う際にのみ使用でき、機能の記述には使用できません。クラスが参照される場合、システムはそのクラスを入力と出力のどちらとして使用するかをユーザーに尋ねます。
- CT ブロック内にはシーケンスコールはありません。評価順序は自動的に決まります。

以上の点を除き、クラスを定義する場合に使用されるブロックダイアグラムエディタの機能をすべて使用できます。

4.5.2 C コードによる連続系ブロックの定義

CT ブロックを C コードで定義する場合も、ブロックダイアグラムで定義する場合と同じ条件が当てはまります。つまり、前述されたいくつかの相違点を除き、すべてクラスを定義する場合と同じように行えます。CT ブロックをブロックダイアグラムで定義する場合と C コードで定義する場合とでは、使用できる基本エレメントが異なります。C コードで CT ブロックを定義する方法についての詳細は、『ASCET リファレンスガイド』の「C コードでモデリングを行う」の項を参照してください。

C コードで定義される CT ブロックは、主としてプロセスモデルの基本コンポーネントです。基本コンポーネントのモデリングに使用される微分方程式は、ブロックダイアグラムよりもコードによる方が早く簡潔に記述できます。

CT ブロックを作成する：

- コンポーネントマネージャで、新しいブロックを作成するフォルダを選択します。
- **Insert → Continuous Time Block → C Code** を選択します。

- コンポーネントの名前を入力し、**<Enter>** を押します。
- **Component → Edit Item** を選択します。

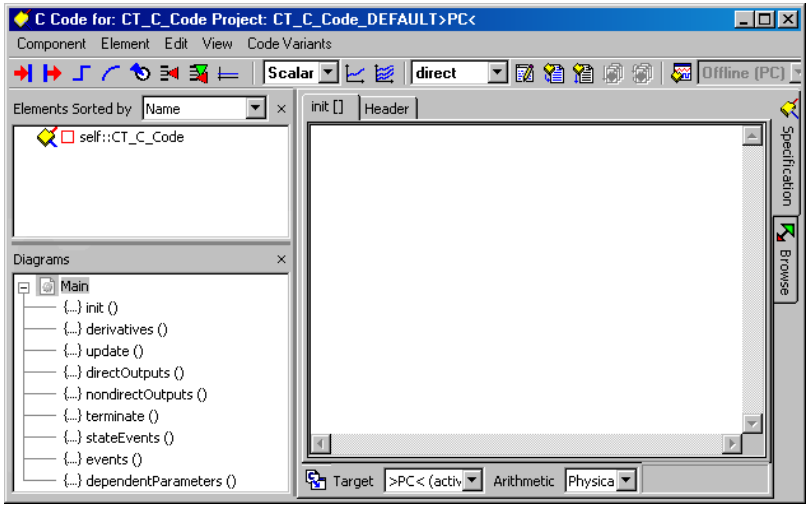
または

- **<Enter>** を押します。

または

- “1 Database”リスト内のコンポーネント名をダブルクリックします。

  CT ブロック用の C コードエディタを開きます。



- ツールバーの右側のコンボボックスから、`direct` または `nondirect` を選択します。
- コードを入力するか、必要に応じて外部モジュールをインポートします。

  C コードエディタの使用法については、304 ページの「C コードエディタ」で詳述されています。

C コードで定義される CT ブロックは直接出力と間接出力のどちらにも対応できますが、一度に両方を指定することはできません。ツールバーの右側のコンボボックスで `direct` が選択されていると `directOutputs[]` メソッドだけが使用でき、`nondirect` が選択されていると `nondirectOutputs[]` メソッドだけが使用できます。直接出力と間接出力については、『ASCET リファレンスガイド』の「代数ループ」の項を参照してください。

4.5.3 ESDL コードによる連続系ブロックの定義

CT ブロックの基本コンポーネントは、C コード以外に ESDL で定義することもできます。前述の相違点以外はクラスを定義する方法と非常によく似ていて、C コードで定義する場合と同じ基本エレメントを使用することができます。

CT ブロックを作成する：

- コンポーネントマネージャで、新しいブロックを作成するフォルダを選択します。
- **Insert → Continuous Time Block → ESDL**を選択します。
- コンポーネントの名前を入力し、**<Enter>** を押します。
- **Component → Edit Item** を選択します。

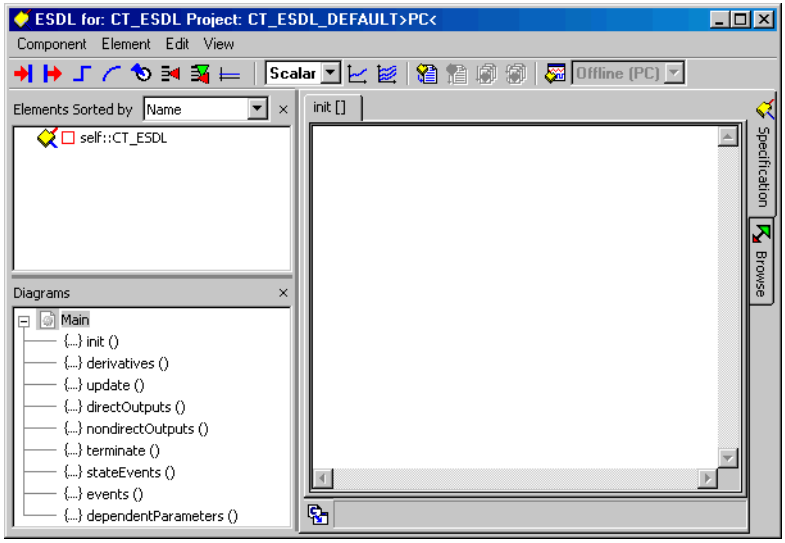
または

- **<Enter>** を押します。

または

- コンポーネント名をダブルクリックします。

CT ブロック用の ESDL エディタを開きます。



- ブロックのコードを記述します。

ESDL エディタの使用方法についての詳細は、323 ページの「ESDL エディタ」で詳述しています。

4.5.4 連続系ブロックの検証

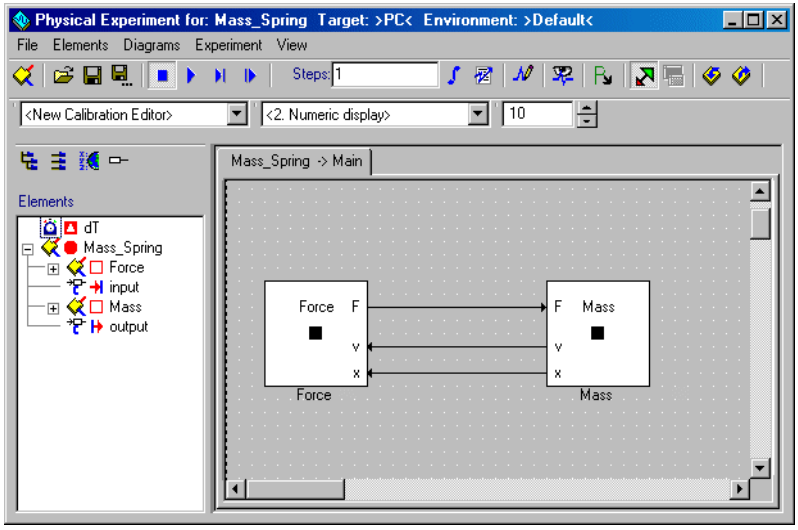
CT ブロックは個々のブロックごとにオフライン実験を行うことができ、複合ブロック、つまり他のブロックを参照しているブロックでもブロック単位の実験は可能ですが、この場合、ハイブリッド実験は行えません。ハイブリッド実験は、クラスと CT ブロックの両方を使用してコントローラとプロセスモデルからなる制御ループのシミュレーションを行う実験です。このようなループの実験はプロジェクト単位で行います。

個々の CT ブロックのオフライン実験の方法はクラスの実験（256 ページの「コンポーネントの検証」を参照してください）の場合と同様ですが、イベントジェネレータは使いません。代わりに「ソルバ」を定義する必要があります。

CT ブロックを使用した実験を行う：

- コンポーネントマネージャで、実験を行いたい CT ブロックを選択します。
- **Build → Experiment → Offline** を選択して実験環境を開きます。

  実験環境は、エディタからでも起動することができます。



- データジェネレータを設定し、さらに測定ウィンドウと適合ウィンドウを設定して実験を行います。

CT ブロックのオフライン実験の操作方法は、ソルバの設定以外は通常の実験と同じです。詳しい操作方法については、547 ページの「実験」を参照してください。

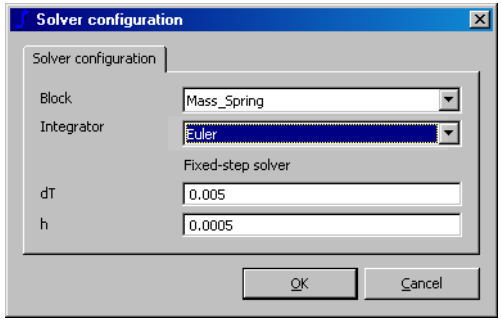
ソルバを設定する：

- **Experiment → Open CT Solver** を選択します。

または

- **Open CT Solver** ボタンをクリックします。

  “Solver Configuration”ダイアログボックスが開きます。



- “Integrator”コンボボックスからソルバを選択します。

  選択されたメソッドに応じて、表示されるパラメータフィールドが異なります。

- パラメータ dT および h の値を設定します。

注記

オンライン実験（429 ページの「オンライン実験とオフライン実験」を参照してください）においては、dT はプロジェクトエディタの“OS”上で設定します。また測定中は“Solver Configuration”の入力フィールドは無効になり、変更できません。

- **OK** をクリックします。

選択されたソルバは、モデル全体の積分用にグローバルに使用されますが、プロジェクト単位で実験を行っている場合は、各ブロック別にソルバを定義することができます（「マルチレート」）。ソルバは実験中に変更することができます。使用できるソルバは以下のとおりです。

- Adams-Moulton 2
- Euler
- Mulstep 2
- Heun

- Runge-Kutta 4
- Variable-step Dormand/Prince RK5
- Variable-step Calvo 6(5)
- Variable-step Dormand/Prince RK8
- Variable-step implicit RK2
- Variable-step implicit RK4
- Variable-step implicit Gear 1
- Variable-step implicit Gear 2

上の 1 ～ 5 番目のソルバについては、*dT* および *h* パラメータだけを設定できます。Gear 4 ソルバには、この 2 つのパラメータ以外にいくつかのパラメータがあります。ソルバについての詳しい説明は、『ASCET リファレンスガイド』の「微分方程式の解法 — 積分のアルゴリズム」の項を参照してください。各パラメータの意味は以下のとおりです。

- *dT*：通信タイムフレーム、つまりブロックが外部と通信を行うインターバル（周期）です。
- *h*：積分のステップサイズ、つまり内部計算のインターバルです。
- *Initial h*: モデルのダイナミクスに応じて h パラメータが算出されるソルバにおいて、h の初期値として使用されます。
- *Minimum h* および *Maximum h* ：算出される h の下限値と上限値です。`Maximum h` の値は `dT` 以下でなければなりません。
- *Relative error* ：h の計算で許容される相対誤差です。
- *Absolute error* ：h の計算で許容される絶対誤差です。
- *Max. iterations*：h の計算を行う最大ステップ数です。指定された数のステップが実行された時点での h の値が、それ以降の積分で使用されます。

サイクルタイムの監視

プロジェクトエディタの“OS”タブ（393 ページの「モニタリングオプション」を参照してください）には、タイマタスクのサイクルタイム（実行周期）を測定する機能が用意されています。

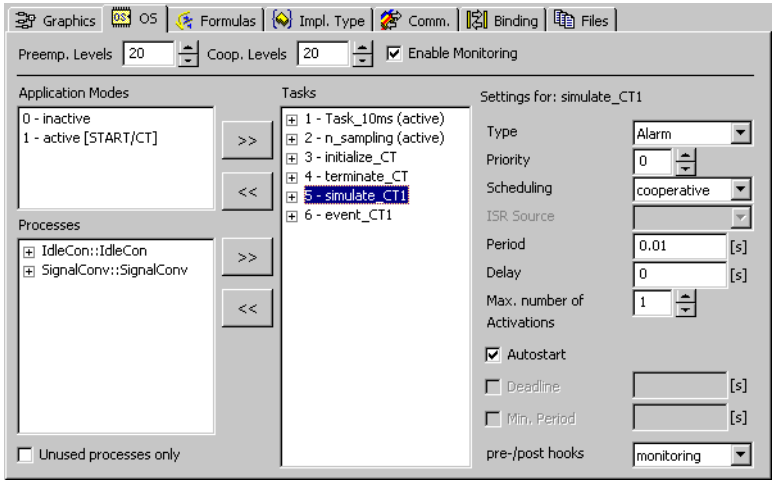
サイクルタイム監視を有効にする：

- CT ブロックを開きます。
- **Component → Default Project→ Edit** を選択し、CT ブロック用のデフォルトプロジェクトを開きます。

または

- ハイブリッドプロジェクトを開きます（『ASCET リファレンスガイド』の「プロジェクトとハイブリッドプロジェクト」および「ハイブリッドプロジェクト」の項を参照してください）。
- プロジェクトエディタの“OS”タブを開きます。

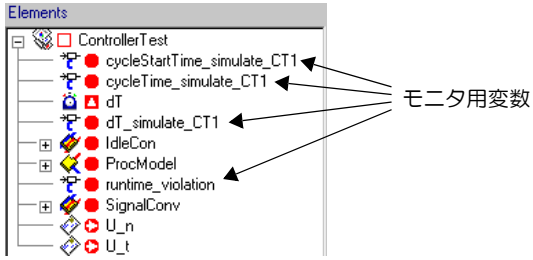
  ”Tasks”リスト内に、CT ブロック用として自動的に定義されたタスクが表示されます（『ASCET リファレンスガイド』の「連続系ブロックとモジュールを結合する」の項を参照してください）。



- `simulate_CT1` タスクを選択します。
- “Pre-/post hooks”コンボボックスで、`Monitoring` を選択します。

  次に実験用のコードを生成する時に、各タスク用のモニタ用変数が生成されます。これらの変数は、測定ウィンドウに表示したり、またオフライ

ン実験の場合はデータロガーに書き込むことができます（582 ページの「データロガー」を参照してください）。



CT ブロックの実験を行う際は、以下の点について注意してください。CT ブロックを含むプロジェクトのシミュレーションにおいては、選択されたソルバ用に 2 つのパラメータを設定できます。

- 外部通信インターバル dT（『ASCET リファレンスガイド』の「外部通信インターバル」の項を参照してください）
- 積分ステップサイズ h（『ASCET リファレンスガイド』の「積分ステップサイズ」の項を参照してください）

シミュレーションタスク simulate_CTn のサイクルタイムは、1 回の dT ステップ内で実行されるすべての積分ステップを考慮したものです。そのため、dT/h の比率が大きくなるほどサイクルタイムは長くなります。補正用ファクタは、モデルのサイズや型といった他のファクタと同様に、任意に指定することはできません。

一方、Simulink で記述された CT ブロックは、dT と h を区別しません。このような CT ブロックを ASCET にインポートした場合は、実験を行う時にソルバや h を変更することはできません。ASCET または Simulink で記述された CT ブロックのサイクルタイムを得るためには、SCET ブロック内の dT および h に、Simulink モデルの積分ステップサイズを設定しておく必要があります。

4.6 論理テーブルエディタ

「論理テーブル」は、モデルの論理的な応答を定義するものです。論理テーブルの各列には論理型の入力と出力を割り当て、「コンビネーション」と呼ばれる各行に、入力と出力の値の組合せを定義します。論理テーブルは、他のコンポーネントにおいて論理接続を行う際に使用されます。また論理テーブルには他のコンポーネントと同様に注釈を付けることができます（218 ページ参照）。

論理テーブルは、論理テーブルエディタを使用して定義します。このエディタを閉じると、指定された出力値を生成する ESDL コードが自動生成されます。入力は論理変数として生成され、各出力ごとにメソッドが作成されます。

論理テーブルを作成する：

- コンポーネントマネージャで、新しいテーブルを作成するフォルダを選択します。
- **Insert → Boolean Table** を選択します。

または

- **Insert Boolean Table** ボタンをクリックします。
- コンポーネントの名前を入力してから **<Enter>** を押します。

  新しい論理テーブルが作成されます。
- **Component → Edit Item** を選択します。

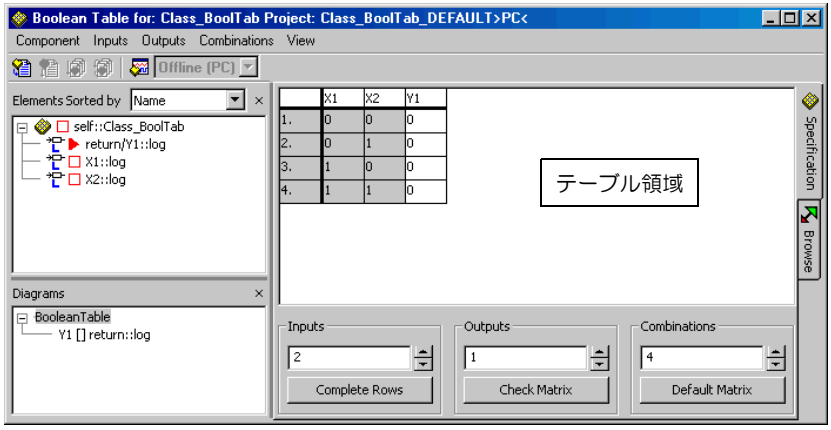
または

- **<Enter>** を押します。

または

- 選択されたテーブルをダブルクリックします。

  論理テーブルエディタが開きます。



このエディタでは、インターフェースを編集することはできません。また、“Elements”ペインで入力や出力を編集することもできません。このため、他のエディタで使用されるほとんどのメニューアイテムは、このエディタでは省略されています。また“Browse”タブではレイアウトの表示／編集しか行えません。

テーブルの表示色は、ASCET オプションで指定できます（57 ページの「テーブルエディタのオプション」を参照してください）。

- **Component**
  - — *Clean code generation directory*

    コード生成ディレクトリ内のファイルをすべて削除します。
  - — *Touch*

    次回のコード生成時に強制的にコードが再生成されるようにします。
    → *Flat* — 編集対象のコンポーネントのみ
    → *Recursive* — 被参照コンポーネントも含める
  - — *Generate Code*

    コンポーネントのコードを生成します。
  - — *Compile*

    生成されたコードをコンパイルします。
  - — *Open Experiment*

    実験を開始します。
  - — *Default Project*

    コンポーネントの実験に使用されるデフォルトプロジェクトの編集を行います。

    →*Edit* — デフォルトプロジェクト用のエディタを開きます。
    → *Resolve Globals* — コンポーネント内のインポートエレメントのうち、それに対応するエクスポートエレメントがないものに対して、グローバルエレメントが生成されます。
    → *Delete Unused Globals* — 実際には使用されていないグローバルエレメントを削除します。
  - — *Edit Layout*

    レイアウトエディタを開きます。
  - — *Edit Data*

    コンポーネント用のデータエディタを開きます。コンポーネントデータの検索が可能です。
  - — *Edit Implementation*

    インプリメンテーションエディタを開きます。コンポーネントインプリメンテーションの検索が可能です。
  - — *Edit Notes*

    注釈エディタを開きます。このエディタでは、コンポーネントについての注釈を入力することができます。
  - — *Check Dependency*

    仮パラメータのモデルパラメータへの割り当てが正しいかどうかを調べます。

— *Export Data*

コンポーネントデータセットをエクスポートします。

— *Show Path*

選択されている内部コンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *File out Generated Code*

生成されたコードをファイルに保存します。
→ *Flat* — 編集対象コンポーネントのみ
→ *Recursive* — 被参照コンポーネントも含める

— *View Generated Code*

コンポーネント用のコードを生成し、それをテキストエディタで表示します。
テキストエディタは、ASCET オプションウィンドウの“ASCET Editor”ノード（60 ページ参照）

— *File Out Generic Code For External Make*

後で外部ツールを使用して行う Make やビルド処理などを行うために、生成されるコードを任意のディレクトリに保存します。

— *Exit*

ブロックダイアグラムエディタを終了します。

- **Input**

— *Add*

既存の最後の入力の右側に、新しい入力を追加します。

— *Insert*

選択されている入力の左側に、新しい入力を追加します。

— *Delete*

選択されている入力を削除します。

— *Rename*

選択されている入力の名前を変更します。

— *Move Left*

選択されている入力を左へ移動します。

— *Move Right*

選択されている入力を右へ移動します。

- **Output**

  Output メニューは、出力の編集用に使用されます。各コマンドの機能は、**Input** メニューと同じです。

- **Combination**

  — *Add*

  テーブルの最後の行の下に、新しいコンビネーションを追加します。

  — *Insert*

  選択されているコンビネーションの上に、新しいコンビネーションを追加します。

  — *Delete*

  選択されているコンビネーションを削除します。

  — *Move Up*

  選択されているコンビネーションを上に移動します。

  — *Move Down*

  選択されているコンビネーションを下へ移動します。

- **View**

  — *Show/Hide*

  エディタまたはコンポーネントの一部の表示／非表示を切り替えます。

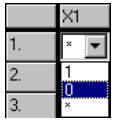
  → *Elements List* — “Elements” ペイン
  → *Diagram List* — “Diagrams” ペイン

論理テーブルの定義

論理テーブルを新しく作成すると、デフォルトで 2 つの入力 X1、X2 と 1 つの出力 Y1、入力値の組合せにそれぞれ対応する 4 つのコンビネーション（1 ～ 4）が作成されます。出力値はどのコンビネーションについても 0 になっています。

各コンビネーションの入出力値は任意に編集することができます。

入力／出力の値を指定する：



- テーブル領域の、変更したい入力値または出力値を右クリックします。

  コンボボックスが開きます。

- 割り当てたい値をコンボボックスから選択します（`0 - false`、`1 - true`）。

注記

`0` と１以外に、`*`（未定義）という値を設定することもできます。

入力または出力の追加は、以下のように行います。

入力／出力を追加する：

- **Inputs** → **Add** または **Outputs** → **Add** を選択します。

または

- テーブル領域を右クリックし、ショートカットメニューから **Inputs** → **Add** または **Outputs** → **Add** を選択します。

  入力または出力が 1 つ追加されます。

または

- ウィンドウ下部の“Inputs”または“Outputs”フィールドの ▲／▼ ボタンを使用して、入力または出力の数を調整します。

  指定された数の入力または出力が作成されます。

  新しく作成された入力の値はデフォルトで `*`（未定義）となり、出力の値は、`0`（`false`）となります。

コンビネーションを追加するには、**Combinations** → **Add** または“Combinations”フィールドを使用します。

追加した入力／出力には、前述の方法で任意の値を設定できます。また、値を 1 つずつ個別に編集する変わりに、マトリックス全体の値を自動的にデフォルト設定することもできます。

デフォルトのマトリックスを生成する：

- 必要な数の入力を作成します。
- **Default Matrix** ボタンをクリックします。

  入力の数に対応して、起こり得るすべての入力値の組合せに対応できる数のコンビネーションが作成されます。出力の数は変更されず、出力値はすべて `0`（`false`）となります。たとえば、3 つの

入力と 1 つの出力を作成して **Default Matrix** ボタンをクリックすると、下のようなテーブルが作成されます。

	X1	X2	X3	Y1
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	0
4.	0	1	1	0
5.	1	0	0	0
6.	1	0	1	0
7.	1	1	0	0
8.	1	1	1	0

入力、出力、およびコンビネーションの削除は、以下のようにして行います。

入力／出力／コンビネーションを削除する：

- テーブル領域で、削除したい入力、出力、またはコンビネーションを選択します。
- **Inputs** → **Delete**、**Outputs** → **Delete** または **Combinations** → **Delete** を選択します。

または

- テーブル領域を右クリックし、ショートカットメニューから **Inputs** → **Delete**、**Outputs** → **Delete** または **Combinations** → **Delete** を選択します。

  1 つの入力、出力またはコンビネーションが削除されます。

または

- “Inputs”、“Outputs” または “Combinations” フィールドの ▲／▼ ボタンを使用して、入力、出力またはコンビネーションの数を調整します。

  指定された内容に従って、入力、出力、またはコンビネーションが削除されます。

入力や出力の名前は変更することができます。（コンビネーションの名前は変更できません。）

入力／出力の名前を変更する：

- テーブル領域の、名前を変更したい入力または出力を強調表示します。
- **Inputs** → **Rename** または **Outputs** → **Rename** を選択します。

または

- 強調表示されている入力または出力を右クリックし、ショートカットメニューから **Inputs → Rename** または **Outputs → Rename** を選択します。

  ダイアログウィンドウが開きます。
- 新しい名前を入力し、**OK** で確定します。

入力値や出力値が表示されている列や、コンビネーションの行をシフトすることができます。

テーブルの列や行をシフトする：

- テーブル領域の、シフトしたい入力を強調表示します。
- **Inputs → Move Left** または **Inputs → Move Right** を選択します。

または

- 強調表示されている入力を右クリックし、ショートカットメニューから **Inputs → Move Left** または **Inputs → Move Right** を選択します。

  列の内容が、それぞれ左または右にシフトされます。列の名前は、元の位置のままです。
- 出力のシフトは、**Outputs → Move Left** または **Outputs → Move Right** で行います。

注記

入力は入力とだけ、また出力は出力とだけ交換できます。入力の最終列と出力の先頭列とを交換することはできません。

- 組み合わせのシフトは、**Combinations → Move Up** または **Combinations → Move Down** で行います。

入力や出力の値をマニュアル操作で編集したり、入力や出力を削除した際には、以下のようにしてテーブルの整合性を確認してください。

テーブルの内容を確認する：

- テーブルを任意に編集した場合には、**Check Matrix** ボタンをクリックします。

  入力値の組合せが同じで出力値が異なる複数のコンビネーションがある場合には、それらの行の境界線が赤色で表示されます。

	X1	X2	X3	Y1
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	1
4.	0	1	1	0
5.	1	0	0	0
6.	1	0	1	0
7.	0	1	0	0
8.	1	1	0	0
9.	1	1	1	0

- 不要なコンビネーションは **Combinations → Delete** で削除してください。

論理テーブルの実験は、他のコンポーネントの場合と同様に **Open Experiment for selected Experiment Target** ボタンで行います。

4.7 条件テーブルエディタ

条件テーブルは、論理テーブルと同様に、モデルのさまざまな論理的応答をテーブルによって定義するものです。複数の列と行によって、異なる条件下での挙動を定義します。

条件テーブル内には基本エレメントを使用できます。また挿入されたクラスのパブリックメソッドにアクセスすることも可能です。条件テーブルの機能は `If…Then…Else` および `If…Then` 文と同じで、図 4-1 の例のように、各行がそれぞれ `If` や `Elseif` の部分に相当します。図 4-1 では、最初の 3 列（“cont”、“log”、“enum_1”）に、AND で結び付けられる条件が定義されていて、残りの列（“out”、“disc”、“methods”）に、その条件が満たされた時の処理内容が定義されています。

	cont	log	enum_1	out	disc	methods
1.	> -1.0 < 0.0	== true	== Speed	= 100.0	= 5	ClassXYZ.doThis(cont,out)
2.	== 0.0	== true	== Speed	= 0.0	= 10	ClassABC.doThat()
3.	> 0.0 < 1.0	== false	== Distance	= -100.0	= ClassX.GetA()	ClassABC.doThat() ClassXYZ.doThis(out,cont)
default	*	*	*	= 3.14159	= 0	*

図 4-1 条件テーブルの例

図 4-1 に定義された内容は、以下のコードに相当します。

```
if ((cont > -1.0) && (cont < 0.0) && (log == true)
                  && (enum_1 == Speed)){
   out = 100.0;
   disc = 5;
   ClassXYZ.doThis(cont,out);
} else
if ((cont == 0.0) && (log == true)
                  && (enum_1 == Speed)){
   out = 0.0;
   disc = 10;
   ClassABC.doThat();
} else
if ((cont > 0.0) && (cont < 1.0) && (log == false)
                 && (enum_1 == Distance)){
   out = -100.0;
   disc = ClassX.GetA();
   ClassABC.doThat();
   ClassXYZ.doThis(out,cont);
} else {
   out = 3.14159;
   disc = 0;
}
```

条件テーブルは、トリガメソッドを呼び出すことによって起動され、その際に現在の状態に一致する条件が定義された行が検索され、その行に定義された処理が実行されます。

条件テーブルは ESDL クラスですが、条件テーブル用の専用エディタで編集します。条件と処理は、それぞれ 1 つまたは複数の条件列（“Condition”列）と処理列（“Instruction”列）に定義され、処理列と条件列の列数と表示色（57 ページの「テーブルエディタのオプション」を参照してください）は自由に設定できます。

条件テーブルを作成する：

- コンポーネントマネージャで、新しいテーブルを作成するフォルダを選択します。
- **Insert → Conditional Table** を選択します。

または

- **Insert Conditional Table** ボタンをクリックします。

- コンポーネントの名前を入力してから **<Enter>** を押します。

  新しい条件テーブルが作成されます。
- **Component → Edit Item** を選択します。

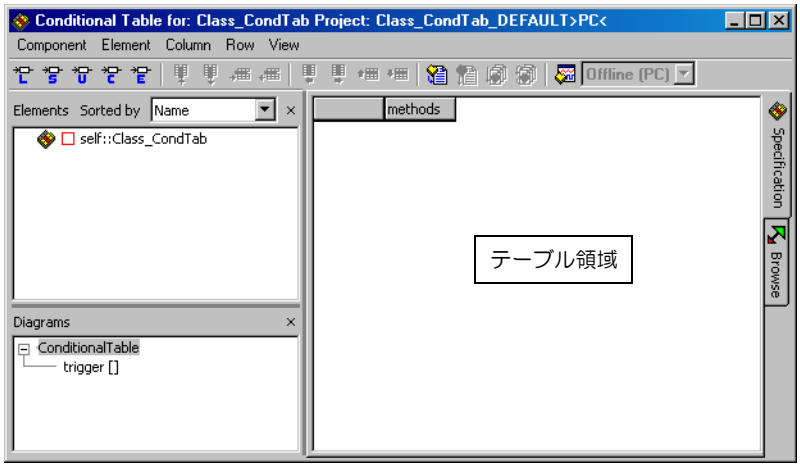
または

- **<Enter>** を押します。

または

- 選択されたテーブルをダブルクリックします。

  条件テーブルエディタが開きます。



条件テーブルを作成すると、自動的に `trigger` という名前のトリガ用パブリックメソッドが作成され、このメソッドについては名前の変更や削除はできません。またその他のメソッドやダイアグラムの追加も行えません。

テーブル内には、“methods”という列が自動的に作成されます。これは必ず処理（“Instruction”）を定義する列の中で最後（右端）に位置している必要があるため、位置の変更や削除は行えません。

条件テーブルの設定方法は 4.7.1 項、条件と処理を定義する方法は 4.7.2 項、また条件テーブルの実験を行う方法は 4.7.3 項に説明されています。

メニューコマンドの概要

- **Component**
  - *Clean code generation directory*

    コード生成ディレクトリ内のファイルをすべて削除します。

— *Touch*

次回のコード生成時に強制的にコードが再生成されるようにします。
→ *Flat* — 編集対象のコンポーネントのみ
→ *Recursive* — 被参照コンポーネントも含める

— *Generate Code*

コンポーネントのコードを生成します。

— *Compile*

生成されたコードをコンパイルします。

— *Open Experiment*

実験を開始します。

— *Default Project*

コンポーネントの実験に使用されるデフォルトプロジェクトの編集を行います。

→ *Edit* — デフォルトプロジェクト用のエディタを開きます。
→ *Resolve Globals* — コンポーネント内のインポートエレメントのうち、それに対応するエクスポートエレメントがないものに対して、グローバルエレメントが生成されます。
→ *Delete Unused Globals* — 実際には使用されていないグローバルエレメントを削除します。

— *Edit Layout*

レイアウトエディタを開きます。

— *Edit Data*

コンポーネント用のデータエディタを開きます。コンポーネントデータの検索が可能です。

— *Edit Implementation*

インプリメンテーションエディタを開きます。コンポーネントインプリメンテーションの検索が可能です。

— *Edit Notes*

注釈エディタを開きます。このエディタでは、コンポーネントについての注釈を入力することができます。

— *Check Dependency*

仮パラメータのモデルパラメータへの割り当てが正しいかどうかを調べます。

— *Export Data*

コンポーネントデータセットをエクスポートします。

— *Show Path*

選択されている内部コンポーネントのパスを表示します。

– *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* – モデルパス
→ *Model asd:// link* – ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* – データベースパス
→ *Database asd:// link* – ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

– *File out Generated Code*

生成されたコードをファイルに保存します。
→ *Flat* – 編集対象コンポーネントのみ
→ *Recursive* – 被参照コンポーネントも含める

– *View Generated Code*

コンポーネント用のコードを生成し、それをテキストエディタで表示します。
テキストエディタは、ASCET オプションウィンドウの“ASCET Editor”ノード（60 ページ参照）

– *File Out Generic Code For External Make*

後で外部ツールを使用して行う Make やビルド処理などを行うために、生成されるコードを任意のディレクトリに保存します。

– *Exit*

ブロックダイアグラムエディタを終了します。

- **Element**

（“Elements”ペインのショートカットメニューからでも実行可能）

注記

以下のコマンドのうち、**Add Item**、**Rename**、**Delete**、**Edit** コマンドは、ASCET-MD がインストールされている場合にのみ使用できます。

– *View → Collapse all*

“Elements”ペイン内のツリー構造をすべて格納し、コンポーネントのみが表示されるようにします。

– *View → Expand all*

ツリー構造を展開し、各コンポーネントの内容すべてが表示されるようにします。

– *Add Item*

コンポーネントを複合エレメントとして組み込みます。

— *Rename*（**<F2>**）

選択されているダイアグラムエレメントの名前を変更します。

— *Delete*（**<Del>**）

選択されているダイアグラムエレメントを削除します。

— *Show Occurrences*

エレメントのすべての「オカレンス」を表示します。ここで「オカレンス」とは、ダイアグラム上に配置されたダイアグラムアイテムを指します。（ブロックダイアグラムエディタでのみ有効）

— *Edit*

選択されているエレメントのプロパティを編集します。

— *Edit Data*

選択されているエレメントのデータを編集します。

— *Edit Implementation*

選択されているエレメントのインプリメンテーションを編集します。

— *Set Cache Locking*

このコマンドは ASCET-RP 用のものです。詳しくは ASCET-RP のユーザーズガイドを参照してください。

— *Edit Component*

選択されている内部コンポーネント（現在編集中のコンポーネントに内包されるコンポーネント）のコンポーネントエディタを開きます。

— *Replace Component*

コンポーネントを別のコンポーネントで置き換えます。コンポーネント名は古いコンポーネントのものがそのまま使われます。

— *Show Path*

選択されている内部コンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *Notes*

選択されている内部コンポーネントの注釈エディタを開きます。

— *Edit Distribution / Edit Max Size*

条件テーブルにはスカラ基本エレメントしか含めることができないため、これらのメニューコマンドは無効です。

— *Copy Elements*（**<Ctrl> + <C>**）

“Elements”ペインから、選択されている 1 つまたは複数のエレメントをコピーします。

— *Paste Elements*（**<Ctrl> + <V>**）

コピーされた 1 つまたは複数のエレメントを“Elements”ペインにペーストします。

— *File Out Data / File In Data*

条件テーブルにはスカラ基本エレメントしか含めることができないため、これらのメニューコマンドは無効です。

— *Export Data*

内部コンポーネントからデータセットをエクスポートします。

- **Column**
（テーブル領域のショートカットメニューからでも実行可能））

  — *Add*

  新しい列を追加します。

  → *Condition* — 条件を定義する列
  → *Instruction* — 処理を定義する列

  — *Delete*

  選択された列を削除します。この列に関連するエレメントは削除されません。

  — *Move Left*

  選択された列を左に移動します。

  — *Move Right*

  選択された列を右に移動します。

- **Row**
（テーブル領域のショートカットメニューからでも実行可能））

  — *Add*

  新しい行を追加します。

  — *Delete*

  選択された行を削除します。

  — *Move Up*

  選択された行を上に移動します。

— *Move Down*

選択された行を下に移動します。

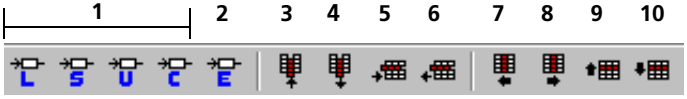
- **View**

— *Show/Hide*

エディタまたはコンポーネントの一部の表示／非表示を切り替えます。

→ *Elements List* — “Elements”ペイン
→ *Diagram List* — “Diagrams”ペイン

ツールバーの説明



1. 変数（論理、符号付き離散、符号なし離散、連続）
2. 列挙型データ
3. 列の追加
4. 列の削除
5. 行の追加
6. 行の削除
7. 列を左に移動
8. 列を右に移動
9. 行を上に移動
10. 行を下に移動

11. コード生成
12. 選択されているターゲット用の実験を開く

番号の付いていないボタンは、条件テーブルエディタにおいては常に無効になっています。

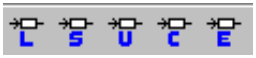
4.7.1 条件テーブルの設定

新しく作成された条件テーブルにはエレメントは含まれず、“methods”列以外は空の状態となっています。ここにエレメントや、列／行を追加します。列を追加するには 1 つ以上のエレメントが追加されている必要があり、また行を追加するには 1 つ以上の条件列が追加されていることが条件となります。

条件テーブル内で条件や処理の定義に使用するエレメントとしては、スカラ値の基本エレメント、列挙型データ、およびクラスを使用できます。前述の `trigger` という名前のメソッドは引数を持つことができますが、戻り値は持てません。エレメントの追加は以下のようにして行います。

エレメントを追加する：

1. 基本エレメント、列挙型データを追加する場合：



- 該当する変数ボタンまたは列挙型ボタンをクリックして、基本エレメントまたは列挙型データを追加します。

  “Elements”リストにエレメントが追加されます。
- エレメント名を入力して **<Enter>** を押します。

2. 複合エレメントを追加する場合：
   - 215 ページの方法でクラスを追加します。

   注記

   ここでは特殊なクラス（ステートマシン、CT ブロック、論理テーブル、条件テーブル）は使用できません。

3. `trigger` メソッドの引数やメソッドローカル変数を追加する場合：
   - “Diagrams”リストから `trigger` メソッドを選択します。
   - ショートカットメニューから **Edit** を選択します。

   または
   - メソッド名をダブルクリックします。

     インターフェースエディタが開きます。`trigger` メソッドは戻り値を持てないので、“Return”タブは表示されません。
   - 必要に応じて 191 ページの方法で引数を追加します。
   - 必要に応じて 193 ページの方法でメソッドローカル変数を追加します。

     メソッドローカル変数は条件テーブル内でしか使えません。

エレメントのデータとインプリメンテーションの編集は、4.11「データの編集」および 4.12「インプリメンテーションの編集」の項に説明されている通常の方法で行えます。

条件テーブルの外部からエレメントにアクセスできるようにするためには、そのエレメントの Get ／ Set ポートを有効にして、エレメントをグローバル定義する必要があります。その他の方法では行えません。

エレメントを外部からアクセス可能にする：

- “Elements”リストからエレメントを選択します。
- **Element** → **Edit** を選択します。

または

- ショートカットメニューから **Edit** を選択します。

エレメントエディタが開きます。

エディタの内容は、選択されたエレメントの型に応じて異なりますが（4.10.1 項参照）、“Scope”フィールドと **Set()** ／ **Get()** オプションは必ず設定可能になっています。

1. Get ／ Set ポート：

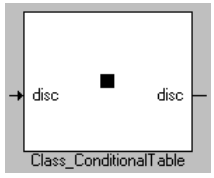
- **Set()** オプションをオンにして、エレメントに入力ポートを追加します。

  外部からのエレメントへの書き込みが可能になります。

- **Get()** オプションをオンにして、エレメントに出力ポートを追加します。

  外部からのエレメントの読み込みが可能になります。

  追加された入出力は、条件テーブルのレイアウト上に以下のようなピンで表示されます。



2. グローバルエレメント：

- そのエレメントが別のコンポーネントや別のプロジェクトで定義されたものである場合は、“Scope”フィールドで **Imported** オプションを選択します。
- そのエレメントが現在編集中の条件テーブル内で定義されたものである場合は、“Scope”フィールドで **Exported** オプションを選択します。

1 つ以上のエレメントを追加すると、条件テーブルの列（条件列と処理列）を作成するためのメニューコマンドが有効になります。

注記

Add a Column ボタンは、条件列と処理列のいずれを作成する場合にも使用されます。新しい列を作成したい位置を選択すると、このボタンが有効になります。

列を作成する際は、その列に割り当てる変数、列挙型データ、または引数を指定します。複合エレメントは列に割り当てることはできず、条件または処理を記述する際にのみ使用できます。

条件を表わす列は、以下のようにして、最大 100 列まで作成できます。

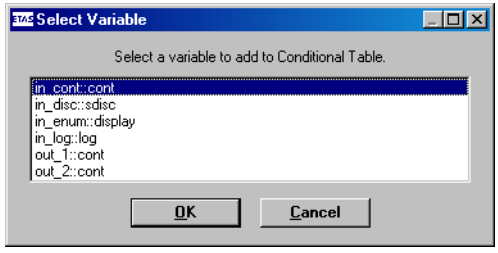
条件列を作成する：

- メニューバーまたはテーブル領域のショートカットメニューから **Column** → **Add** → **Condition** を選択します。

または

- 新しい列を追加したい位置の列を選択して **Insert a Column** ボタンをクリックします。

  “Select Variable”ダイアログボックスが開き、まだ他の条件列に割り当てられていない変数の一覧が表示されます。

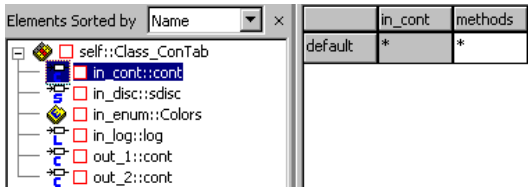


- 変数を選択します。

  どの変数も、条件列の中では 1 つの列にのみ割り当てることができます。

- **OK** をクリックしてダイアログボックスを閉じます。

  選択された変数の名前をもつ条件列が追加されます。その列が、その条件テーブル内で初めて追加された列であった場合、同時に“Default”という名前の行が追加されます。



メニューのコマンド（**Column → Add → Condition**）で条件列を作成した場合は、新しい列は条件列の右端に追加され、**Insert a Column** ボタンで作成した場合は、選択されていた列の右隣に追加されます。

処理を表わす列も、以下のようにして最大 100 列まで作成できます。

処理列を作成する：

- メニューバーまたはショートカットメニューから **Column → Add → Instruction** を選択します。

または

- 新しい列を追加したい位置の列を選択して **Insert a Column** ボタンをクリックします。

  “Select Variable”ダイアログボックスが開き、まだ他の処理列に割り当てられていない変数の一覧が表示されます。

- 変数を選択します。

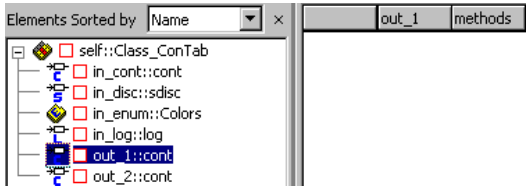
  どの変数も、処理列の中では 1 つの列にのみ割り当てることができます。

注記

`trigger` メソッドは、処理列に割り当てることはできません。

- **OK** をクリックしてダイアログボックスを閉じます。

  選択された変数の名前をもつ処理列が追加されます。条件列の場合とは異なり、その列がその条件テーブル内で初めて追加された列であっても、行は追加されません。



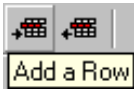
メニューコマンド（**Column → Add → Condition**）で処理列を作成した場合は、新しい列は“method”列の左隣に追加され、**Insert a Column** ボタンで作成した場合は、選択されていた列の右隣に追加されます。

条件テーブル内に 1 つ以上の条件列が作成されると、行を作成することが可能となります。1 つの条件テーブル内には“dafult”という行が自動的に作成され、ユーザーが追加できるのは最大 99 列までです。

行を作成する：

- 新しい行を追加したい位置の行を選択します。
- メニューバーまたはショートカットメニューから **Row → Add** を選択します。

または



- **Add a Row** ボタンをクリックします。

  選択されていた行の下に新しい行が追加されます。どの行も選択されていなかった場合、または“defualt”行が選択されていた場合は、“default”行の直前の位置に追加されます。

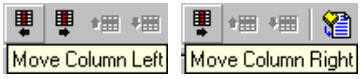
	in_cont	in_disc	out_1	methods
1.	*	*	*	*
default	*	*	*	*

列の移動は、同じ種類の列（条件列または処理列）の中でのみ可能です。

列を移動する：

- 移動したい列を選択します。
- メニューバーまたはショートカットメニューから **Column → Move Left** または **Column → Move Right** を選択します。

または



- **Move Column Left** または **Move Column Right** ボタンをクリックします。

  列が 1 つ左または 1 つ右に移動します。

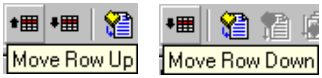
“method”列は常に処理列の一番左に位置し、移動はできません。

列の移動は、同じ種類の列（条件列または処理列）の中でのみ可能です。

各行は、上に位置するほど優先的に処理されるため、行を上下に移動することによって処理の優先順位を変更することができます。

行を移動する：

- 移動したい行を選択します。
- メニューバーまたはショートカットメニューから **Row → Move Up** または **Row → Move Down** を選択します。



または

- **Move Row Up** または **Move Row Down** ボタンをクリックします。

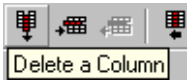
  行が 1 行上または下に移動します。

“default”行は常に一番下に位置し、移動はできません。

列や行、およびエレメントの削除は、以下のようにして行います。

列を削除する：

- 削除したい列を選択します。
- メニューバーまたはショートカットメニューから **Column → Delete** を選択します。



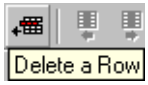
または

- **Delete a Column** ボタンをクリックします。

  選択された列が削除されます。ただしこの列に割り当てられていたエレメントは、そのまま“Elements”リスト内に残り、再度、列に割り当てることができます。

行を削除する：

- 削除したい行を選択します。
- メニューバーまたはショートカットメニューから **Row → Delete** を選択します。



または

- **Delete a Row** ボタンをクリックします。

  選択された行が削除されます。

エレメントを削除する：

- “Elements”リストから削除したいエレメントを選択します。
- **Element → Delete** を選択します。

または

- ショートカットメニューから **Delete** を選択します。

または

- **<Del>** を押します。

  確認のダイアログボックスが開きます。
- **OK** で確定します。

注記

削除されたエレメントが条件または処理の定義に使用されていた場合、その記述内容は自動的には削除されないので、必要な修正をマニュアル操作で行ってください。

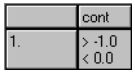
4.7.2 条件テーブルの定義

新しく作成された列と行の各セルには、アスタリスク（*）が表示されるので、ここに実際の条件と処理をマニュアル入力します。

条件： 条件列のセルには以下のフォーマットで条件文を入力します。

```
== b, <= b, >= b, < b, > b, != b, *
```

- b は、値、エレメント、または被参照クラスのパブリックメソッド名のいずれかです。またこれらを使用して ESDL 構文で記述された式も使用できます。
- アスタリスクは、いずれの値でも許されることを意味し、その列の値がその行の条件判定において影響しない場合に使用します。



- 1 つのセルに複数の条件を入力することができます。各条件は改行で区切られ、AND 条件で判定されます。

注記

“defaults”行の条件列には必ずアスタリスクである必要があるため、編集することはできません。

処理： s 処理列のセルには代入文（またはメソッド呼び出し）を入力します。

```
= b, class.method(<variables>), *
```

- b は、値、エレメント、または被参照クラスのパブリックメソッド名のいずれかです。またこれらを使用して ESDL 構文で記述された式も使用できます。
- `class.method(<variables>)` は、メソッド呼び出しを表わします。これは“methods”列のセルにのみ使用できます。被参照クラスのすべてのパブリックメソッドを呼び出せます。
- アスタリスクはそのセルに何も処理が定義されていないことを表わします。

条件や処理の定義に使用できるエレメントとクラスは、“Elements”リストに含まれているものだけです。その他のものが使用されていると、コード生成時にエラーが発生します。

注記

入力された条件や処理について、構文や内容の妥当性チェックは行われません。入力は正確に行い、特にメソッド名は間違いのないようにしてください。正しくない入力が行われていても、コード生成時までエラーは発生しません。

条件または処理を入力する：

- 編集したいセルをダブルクリックします。

  セルが入力モードになります。
- 条件または処理を入力します。

  条件または処理は、純粋な文字列として入力されます。

注記

複数の行に同じ条件が重複して入力されていても、特にチェックは行われません。

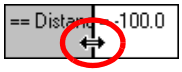
処理列のセルも複数行の入力が可能です。各行がそれぞれ代入処理（下図の“out”列参照）またはメソッド呼び出し（“methods”列参照）として解釈されます。

out	methods
= -100.0 = out + cont = MathFcn.cos(out)	ClassABC.doThat() ClassXYZ.doThis(out,cont)

- 入力内容を確定するには、テーブル内の別の場所をクリックします。

各カラムの幅は、以下のようにして任意に変えることができます。

カラム幅を変更する：



- 幅を変えたいカラムの右側の境界線までマウスポインタを移動します。

  マウスポインタが左のような左右の矢印に変わります。
- マウスボタンを押したまま、境界線を左右にドラッグします。

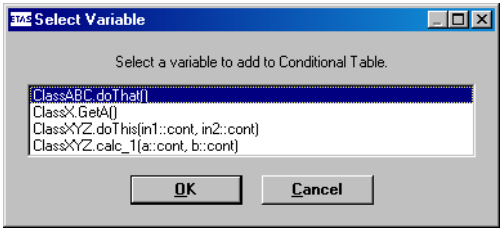
  1 度エディタを閉じると、カラム幅はデフォルト幅に戻ります。

以下のようにして、タイプミスによるメソッド名の入力エラーを回避することができます。

条件／処理の定義にメソッドを使用する：

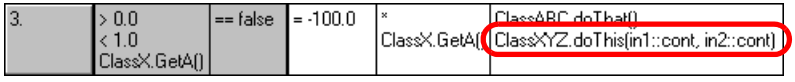
- メソッドを使用したいセルを選択します。
- ショートカットメニューから **Insert Method** を選択します。

  “Select Variable”ダイアログボックスが開き、すべての被参照クラスに含まれるすべてのパブリックメソッドが一覧表示されます。



- 使用したいメソッドを選択します。
- **OK** をクリックします。

  メソッド名が、選択されているセル内の最後の行に挿入されます。メソッドに引数がある場合、それらの名前も付加されます。

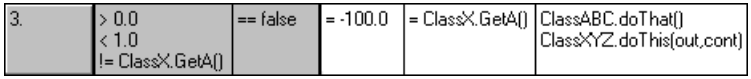


- 以下のようにして、メソッド呼び出し文を編集します。

  — 挿入された引数名を、条件テーブル内の変数名に置き換えます。

— 条件列のセルに、比較演算子を挿入します（359 ページ参照）

— 必要に応じて処理列（“methods”列以外）のセルのアスタリスクを削除して、代入演算子 = を挿入します。



4.7.3 条件テーブルの検証

テーブルの定義が終了したら、他のコンポーネントと同様に（4.1.10「コンポーネントの検証」を参照してください）、コードを生成して実験を行うことができます。ブロックダイアグラムと ESDL コンポーネントとの違いは、実験を開始する際に使用できるコマンドが **Diagram → Analyze Diagram** のみである点です。

条件テーブルのコード生成時には、以下のような問題が発生する可能性があります。

- 条件／処理に無効な情報（エレメント名など）が入力されていると、エラーが発生します。
- 複数の行に同じ条件が入力されていると、それらの行ごとに `If…Then…` 文が生成されますが、実行時にはそのうちの最初の行のみが実行されます。記述内容の妥当性についてのチェックは一切行われないため、このような場合、ワーニングやエラーはまったく発生しません。
- 条件テーブルで使用されている列挙型（Enumeration）に含まれる列挙子（Enumerator）の削除、追加、移動を行うと、条件テーブル内の情報の矛盾が発生する場合があります。この場合、コード生成時にエラーが発生します。
- 条件テーブルで使用されている列挙型（Enumeration）をデータベースから削除すると、使用されている列挙型はコード生成時に未定義エレメントとして扱われ、エラーが発生します。

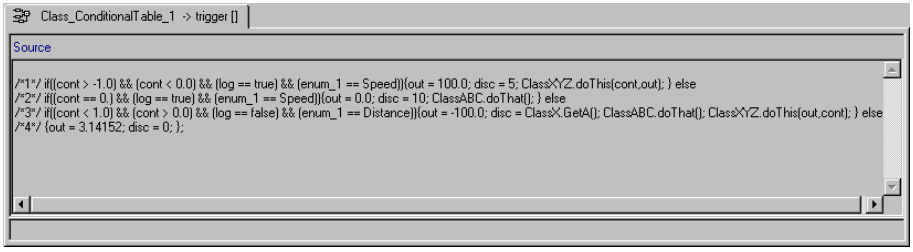
条件テーブルは `trigger` メソッドによって呼び出されて実行され、その際は毎回以下のような手順が実行されます。

1. `trigger` メソッドの引数と各エレメントに代入されている値が評価されます。
2. 定義された条件が現在の状態に一致する行が検索され、一致する行が見つかった時点で検索は終了します。それより下にも一致する行があっても、それらは無視されます。

   一致する行がない場合、デフォルト行（“default”行）が選択されます。
3. 選択された行に定義されている処理が実行されます。

   `trigger` は戻り値を持たないため、処理結果は、Get ポート（354 ページ参照）で読み取る必要があります。データの一貫性を保持するため、グローバル変数の使用は避けてください。

生成された ESDL コードは、以下のように“Physical Experiment”ウィンドウ内に表示されます。条件テーブルの ESDL コードは、ここでしか見ることができません。



実験のセットアップや実行の方法は、6.1「実験環境」の項を参照してください。

4.8 プロジェクトエディタ

ASCET のプロジェクトは、組み込みソフトウェアシステム全体の機能を定義するもので、組み込みシステムで実行されるコードを生成する単位となります。プロジェクト内に定義された機能はさまざまな実験ターゲット上でリアルタイムに実行することができます。また実際のマイクロコントローラターゲットにあわせた実装情報を定義して固定小数点コードを生成し、そのコードをフルパスまたはバイパス実験としてマイクロコントローラターゲット上で実行することもできます。

また、プロジェクトを使って、CT ブロックとモジュールのコンビネーションで構成される制御ループのモデリングを行うこともできます。その場合は、プロセスモデルの記述に CT ブロックを使い、コントローラの機能記述にモジュールを使います。このようにして作成されたモデルを実験ターゲット上で実験に用いることができます。

プロジェクトの開発は、まず個々のコンポーネントを開発してテストしてから、それらを組み合わせてプロジェクトを構築する、というようなモジュール化された手順で行います。具体的には以下のような手順で行われます。

- システムを構成するコンポーネントを選択します。コンポーネントは参照によって組み込まれるため、コンポーネントの内容が変更されると、その変更内容はそのコンポーネントを使用するすべてのプロジェクトに直接影響します。
- リアルタイムオペレーティングシステムをセットアップして、組み込みソフトウェアシステムの全体的な制御フローを定義します。
- ターゲットコントローラ用の固定小数点演算コードを生成するために必要な「インプリメンテーション」、つまり実装情報を定義します。インプリメンテーションを定義するには、変換式をプロジェクトに組み込む必要があります。

プロジェクトにも、コンポーネントの場合と同様、複数のデータセットとインプリメンテーションを定義することができるので、機能定義から実行形式のコードを生成する際には、それらの適切な組合せを選択します。つまり、プロジェクトのコードバリエーションは、データセット（コンポーネント単位の実験でも選択可能です）とインプリメンテーションの組み合わせによって定義されます。

インプリメンテーションは、固定小数点演算のコードを生成する際のみに必要です。浮動小数点演算しか必要ない場合（たとえばバイパス環境など）には、インプリメンテーションを定義する必要はありません。

コード生成時には、プロジェクトごとにコードバリエーションと対象ターゲットを選択できます。コードの算術演算機構は、浮動小数点、固定小数点、量子化浮動小数点の 3 種類から選択でき、このうち量子化浮動小数点演算は、浮動小数点演算をベースにしながら固定小数点演算をシミュレートするもので、これによって量子化の影響を調べることができ、コードを実行しながら量子化と値の限界値を対話形式で調整することができます。

新しいプロジェクトを作成する：

- コンポーネントマネージャで、新しいプロジェクトを作成するフォルダを選択します。
- **Insert** → **Project** を選択します。

または

- **Insert Project** ボタンをクリックします。
- 新しいプロジェクトの名前を入力してから **<Enter>** を押します。

  新しいプロジェクトが作成されます。
- メニューバーから **Component** → **Edit Item** を選択します。

または

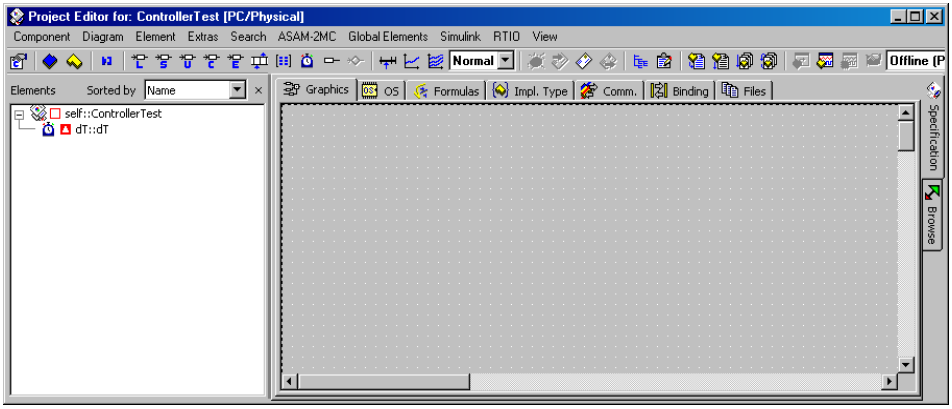
- **<Enter>** を押します。

または

- “1 Database”リスト内のプロジェクト名をダブルクリックします

  プロジェクトエディタが開きます。

エディタで生成されるテストプロジェクト以外のすべてのプロジェクトは、コンポーネントと同様にデータベースに格納されます。個々のプロジェクトはプロジェクトエディタで構築されてセットアップされ、実験に用いられます。以上の処理は、すべてプロジェクトエディタから行うことができます。



“Elements”ペインには、プロジェクトを構成するコンポーネントがツリー形式で表示されます。このツリーには、プロジェクトが参照するコンポーネントがすべて含まれ、それらを任意に展開／省略表示することができます。各コンポーネントの間の通信に用いられるエレメントは、プロジェクト内でグローバルに使用されます。

4.8.1 コンポーネント用のデフォルトプロジェクト

各種エディタで作成されるコンポーネントには、それぞれテスト用のデフォルトプロジェクトを作成することができます。このデフォルトプロジェクトはエディタ上で直接作成されるので、コンポーネントマネージャには表示されませんが、プロジェクトエディタ上で編集したり実験を行うことができます。プロジェクトエディタの機能と操作について、以降の項で説明します。

デフォルトプロジェクトを編集する：

- コンポーネントエディタで、**Component → Default Project → Edit** を選択します。

  デフォルトプロジェクト用のプロジェクトエディタが開きます。

- **Component → Default Project → Resolve Globals** を選択します。

  コンポーネント内のインポートエレメントのうち、それに対応するエクスポートエレメントがないものに対して、グローバルエレメントが生成されます（380 ページの「グローバル通信の定義」も参考にしてください）。

  この時、プロジェクトエディタは開きません。
- **Component → Default Project → Delete Unused Globals** を選択します。

  コンポーネント内で実際に使用されていないグローバルエレメントが、コンポーネントのデフォルトプロジェクトから削除されます。

コンポーネントの実験に使用されるオフライン実験環境は、デフォルトプロジェクト用のオフライン実験環境に組み込まれます。またこの他にプロジェクト用に実行できるのオンライン実験においては、プロジェクトはリアルタイムオペレーティングシステムに定義された条件でリアルタイムに実行されます。

4.8.2 メニューコマンドの概要

- **Component**
  - — *Clean code generation directory*

    コード生成ディレクトリ内のファイルをすべて削除します。
  - — *Touch*

    次回のコード生成時に強制的にコードが再生成されるようにします。
    → Flat — 編集対象のコンポーネントのみ、
    → Recursive — 被参照コンポーネントも
  - — *Generate Code*（**<Ctrl> + <F7>**）

    コンポーネントのコードを生成します。
  - — *Compile*

    生成されたコードをコンパイルします。
  - — *Link Only*

    プロジェクトのオブジェクトファイルを実行形式の HEX ファイルにリンクします。
  - — *Build*（**<F7>**）

    選択されているターゲット用のコードを生成して保存し、実行形式のコードを作成します。結果はデータベースに保存されます。

— *Build from directory*（**<Alt> + <F7>**）

選択されているターゲット用のコードを生成して保存し、実行形式のコードを作成します。結果は所定のディレクトリに書き込まれ、データベースには保存されません。

— *Rebuild All*（**<Shift> + <F7>**）

プロジェクト全体（そのすべてのコンポーネントを含みます）の完全なビルドを行います。**Component → Touch → Recursive** と **Component → Build** を続けて実行する場合と同じ結果が得られます。

— *Transfer Project*

プロジェクトを選択された実験用に移行します。

このコマンドは、ビルドオプションで `ES1130`、`ES1135`、`Prototyping` のいずれかがターゲットとして選択され、さらにプロジェクトエディタの“Experiment Target”コンボボックスで `INCA` または `INTECRIO` が選択されている場合にのみ有効です。

— *Open Experiment*

実験を開始します。

— *Flash Target*

ASCET で生成されたコードを、実験ターゲットのフラッシュメモリに書き込みます。コードには、フラッシュメモリからのブートを行うためのスタートアップルーチンが組み込まれ、ターゲットハードウェアをリセットするたびに ASCET モデルが実行されるようになります。

— *Recconect to Experiment*

選択された実験ターゲット上で稼動している実験への再接続を行います。

— *Default Project*

コンポーネントの実験に用いるデフォルトプロジェクトを編集します。（コンポーネントエディタでのみ有効）

— *Edit Data*

プロジェクト用のプロジェクトエディタを開きます。その際、プロジェクトのコンポーネントデータを検索することができます。

— *Edit Implementation*

プロジェクト用のインプリメンテーションエディタを開きます。その際、プロジェクトのコンポーネントインプリメンテーションを検索することができます。

— *Edit Notes*

注釈エディタを開きます。このエディタでは、コンポーネントについての注釈を入力することができます。

— *Edit Project Properties*

“Project Properties”ダイアログボックスを開き、プロジェクトのオプション設定を行います（400 ページの「プロジェクトの設定」を参照してください）。

— *Check Dependency*

仮パラメータのモデルパラメータへの割り当てが正しいかどうかを調べます。

— *Export Data*

コンポーネントまたはプロジェクトのデータセットをエクスポートします。

— *Show Path*

内容されたコンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *File out Generated Code*

生成されたコードをファイルに保存します。
→ *Flat* — 編集対象コンポーネントのみ
→ *Recursive* — 被参照コンポーネントも含める

— *View Generated Code*

コンポーネント用のコードを生成し、それをテキストエディタで表示します。
テキストエディタは、ASCET オプションウィンドウの“ASCET Editor”ノード（60 ページ参照）

— *File Out Generic Code For External Make*

後で外部ツールを使用して行う Make やビルド処理などを行うために、生成されるコードを任意のディレクトリに保存します。

— *Exit*

プロジェクトエディタを終了します。

- **Diagram**

— *Store To Cache*

コンポーネント定義をキャッシュメモリに保存します（データベースには保存されません）。

— *Analyze Diagram*

現在のダイアグラムを解析します。

— *Load Diagram*

ダイアグラムをロードします。

- **Element**

— *View → Collapse all*

“Elements”ペイン内のツリー構造をすべて格納し、コンポーネントのみが表示されるようにします。

— *View → Expand all*

ツリー構造を展開し、各コンポーネントの内容すべてが表示されるようにします。

— *Add Item*

コンポーネントをプロジェクトに組み込みます。

— *Rename*（**<F2>**）

コンポーネントの名前を変更します。

— *Delete*（**<Del>**）

コンポーネントのインスタンスを削除します。

— *Show Occurrences*

エレメントのすべてのオカレンスを表示します（ブロックダイアグラムディタのみ）。

— *Edit*

エレメントのプロパティを編集します。

— *Edit Data*

エレメントのデータを編集します。

— *Edit Implementation*

エレメントのインプリメンテーションを編集します。

— *Set Cache Locking*

このコマンドは ASCET-RP 用のものです。詳しくは ASCET-RP のユーザーズガイドを参照してください。

— *Edit Component*

内容されたコンポーネントを定義するためのエディタを開きます。

— *Replace Component*

コンポーネントを別のコンポーネントに置き換えます。コンポーネント名は古いコンポーネントのものがそのまま使われます。

— *Show Path*

内容されたコンポーネントのパスを表示します。

— *Copy Path to Clipboard*

内包されるコンポーネントのパスをクリップボードにコピーします。
→ *Model Path* — モデルパス
→ *Model asd:// link* — ASCET プロトコルフォーマットのパス（モデル内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）
→ *Database Path* — データベースパス
→ *Database asd:// link* — ASCET プロトコルフォーマットのパス（データベース内の ASCET コンポーネントをハイパーリンクとして開いたり参照したりできます）

— *Notes*

内容されたコンポーネントの注釈を編集するためのエディタを開きます。

— *Edit Distribution*

ディストリビューション用のエディタを開きます（グループカーブ／マップの場合のみ）。

— *Edit Max Size*

配列、マトリックス、特性カーブ／マップの最大サイズを定義するためのエディタを開きます。

— *Copy Elements*（**<Ctrl> + <C>**）

“Elements”リストからエレメントをコピーします。

— *Paste Elements*（**<Ctrl> + <V>**）

コピーされたエレメントを“Elements”リストにペーストします。

— *File Out Data*

配列、マトリックス、特性カーブ／マップからファイルにデータを書き出します。

— *File In Data*

配列、マトリックス、特性カーブ／マップのデータをファイルから読み込みます。

— *Export Data*

内容されたコンポーネントからデータセットをエクスポートします。

- **Extras**

— *Copy C-Code From*

他のターゲットから C コードをコピーします。

— *Global Replace Formula*

変換式をリカーシブに置換します。

— *Update Implementations*

プロジェクトのすべてのインプリメンテーションを更新します。

- **Search**

検索範囲が限定される以外は、132 ページの「データベースのブラウズ」で説明されている検索機能と同じです。

— *Component*

プロジェクト内のコンポーネントを検索します。

— *Reference To Component*

プロジェクト内で、指定のコンポーネントへの参照を検索します。

— *Declaration of Methods*

プロジェクト内で、指定のメソッドまたはプロセスを宣言しているコンポーネントを検索します。

— *References of Methods*

プロジェクト内で、指定のメソッドまたはプロセスを使用しているコンポーネントを検索します。

— *Declaration of elements*

プロジェクト内で、指定のエレメントを宣言しているコンポーネントを検索します。

— *References of elements*

プロジェクト内で、指定のエレメントを使用しているコンポーネントを検索します。

— *Sender Of Message*

プロジェクト内で、指定のメッセージを送信しているモジュールを検索します。

— *Receiver Of Message*

プロジェクト内で、指定のメッセージを受信しているモジュールを検索します。

— *Find/Replace ESDL or C code*

C コードまたは ESDL コンポーネントに含まれる文字列を検索／置換します。

- **ASAM-2MC**
  - *Write*

    適合用ファイルを作成します。
  - *Read Hex File*

    適合に使用するファイルを読み取ります。
- **Global Elements**
  - *Resolve Globals*

    コンポーネント内のインポートエレメントのうち、それに対応するエクスポートエレメントがないものに対して、グローバルエレメントが生成されます。
  - *Delete Unused Globals*

    使用されていないグローバルエレメントを削除します。

上記の共通コマンドのほか、それぞれのタブが選択されている場合にだけ使用できるコマンドがあります。

“Graphics” タブ

- **View**
  - *Show/Hide → Element List*

    “Elements” リストの表示／非表示を切り替えます。
  - *Redraw*

    ダイアグラムを再描画します。
  - *Rebuild Connections*

    演算子インプリメンテーションの自動変換により煩雑になった接続線を、スムーズな線に再描画します。
  - *Undo*（**<Ctrl> + <Z>**）

    最後の操作を取り消します。
  - *Redo*（**<Ctrl> + <Y>**）

    Undo コマンドを取り消します。
  - *Show Hierarchy Path*

    最高レベルの階層パスを表示します。
  - *Parent Component*

    内容されたコンポーネント用のエディタを開きます。このオプションは、内容されたコンポーネントの編集中に限り使用できます。
  - *Parent Hierarchy Level*

    階層パスを表示します。

- *Page frame Portrait*

  ダイアグラムを縦長のフォーマットで表示します。
- *Page frame Landscape*

  ダイアグラムを横長のフォーマットで表示します。
- *Grid*

  描画領域のグリッドを変更します。
- *Print Diagram*

  “Graphic”タブの内容を印刷します。
- *Save as Postscript/Bitmap/RTF/GIF*

  指定されたフォーマットでダイアグラムを保存します。

“OS”タブ

- **Operating System**
  - *Copy From Target*

    選択されているターゲット用から現在のターゲット用にプロジェクトを変換します。
  - *Copy To Target*

    選択されているターゲット用にプロジェクトを変換します。
- **Application Mode**

注記

これらのメニューコマンドは、“Application Modes”フィールドのショートカットメニューからも実行できます。

  - *Add*

    新しいアプリケーションモード（運用モード）を作成します。
  - *Rename*

    アプリケーションモードの名前を変更します。
  - *Delete*

    アプリケーションモードを削除します。
  - *As Start Mode*

    アプリケーションモードを開始モードとして定義します。
  - *As Deault CT-Mode*

    アプリケーションモードを CT ブロック用の標準モードとして定義します。

— *Assign*

アプリケーションモードをタスクに割り当てます。

— *Show in Tasks*

選択されているアプリケーションモードを割り当てるタスクをすべて選択します。

— *Show Init Tasks*

アプリケーションモードを割り当てる初期化タスクをすべて選択します。

- **Process**

注記

これらのメニューコマンドは、“Process”ペインのショートカットメニューからも実行できます。

— *Assign*

プロセスをタスクに割り当てます。

— *Show in Tasks*

“Tasks”フィールド内のプロセスのオカレンスをすべて表示します。

- **Task**

注記

これらのメニューコマンドは、“Task”ペインのショートカットメニューからも実行できます。

— *Add*

タスクを作成します。

— *Rename*

タスク名を変更します。

— *Delete*

タスクを削除します。

— *Move Up*

プロセスを、タスク内の上位に移動します。

— *Move Down*

プロセスを、タスク内の下位に移動します。

— *Deassign Application Modes*

アプリケーションモードを現在のタスクから削除します。

— *Deassign Processes*

プロセスをタスクから削除します。

— *Delete Undefined Processes*

定義されていないプロセスをタスクから削除します。

— *Open Module*

選択されたプロセスを含むモジュールを開きます。

— *Show in Processes*

“Processes”フィールド内のプロセスのオカレンスをすべて表示します。

— *Show in Application Modes*

タスクに属しているすべてのアプリケーションモードを選択します。

— *Set Cache Locking*

このコマンドは ASCET-RP 用のものです。詳しくは ASCET-RP のユーザーズガイドを参照してください。

“Formulas” タブ

- **Global Formulas**

注記

これらのメニューコマンドは、タブ上のショートカットメニューからも実行できます。

— *Add*

変換式を追加します。

— *Rename*

変換式名を変更します。

— *Delete*

変換式を削除します。

— *Edit*

フォーミュラエディタを開きます。

— *File Out*

変換式をファイルに格納します。
→*All* ーすべての変換式、
→*Selected* ー選択された変換式のみ

— *File In*

変換式をファイルから追加します。
→ *All* ーファイル内のすべての変換式、
→ *Selected* ーファイル内の選択された変換式のみ

— *Select All*（**<Ctrl> + <A>**）

すべての変換式を選択します。

"Impl. Type" タブ

- **Global Implementation**

注記

これらのメニューコマンドは、タブ上のショートカットメニューからも実行できます。

— *Add*

インプリメンテーション型を追加します。
→ *cont* ーモデルデータ型 `cont`
→ *sdisc* ーモデルデータ型 `sdisc`
→ *udisc* ーモデルデータ型 `udisc`

— *Rename*

インプリメンテーション型を変更します。

— *Delete*

インプリメンテーション型を削除します。

— *Edit*

インプリメンテーション型エディタを開きます。

— *Copy implementation*

選択されているインプリメンテーション型の設定内容をデータベースのクリップボードにコピーします。

— *Paste implementation*

選択されているインプリメンテーション型に、データベースのクリップボードにコピーされている設定内容を貼り付けます。

— *File Out*

インプリメンテーション型を XML ファイルに格納します。
→ *All* ーすべてのインプリメンテーション型
→ *Selected* ー選択されたインプリメンテーション型のみ

— *File In*

インプリメンテーション型を XML ファイルから読み込みます。
→ *All* ーファイル内のすべてのインプリメンテーション型
→ *Selected* ーファイル内の選択されたインプリメンテーション型のみ

— *Select All*（**<Ctrl> + <A>**）

すべてのインプリメンテーションを選択します。

“Files”タブ

- **Project Files**

> **注記**
>
> これらのメニューコマンドは、タブ上のショートカットメニューからも実行できます。

— *View*

プロジェクトファイルを、そのファイルに割り当てられているアプリケーションで表示します。

— *Edit*

プロジェクトファイルを、そのファイルに割り当てられているアプリケーションで編集します。

— *Add*

外部ファイルを追加します。

— *Delete*（**<Del>**）

ファイルを削除します。

— *Update*

選択されたプロジェクトファイルを更新します。

— *Write Selected Files(s)*

選択されたプロジェクトファイルをデフォルトディレクトリに書き込みます。

— *Write Selected Files(s) to*

選択されたプロジェクトファイルを任意のディレクトリに書き込みます。

— *Write All Files*

プロジェクトに含まれるすべてのプロジェクトファイルをデフォルトディレクトリに書き込みます。

— *Replace With Backup Contents*

選択されたプロジェクトファイルを、データベースにバックアップされているバージョンのものに置き換えます。

— *Show Default Path*

デフォルトディレクトリのパスを表示します。

— *Exlore Default Path*

デフォルトディレクトリをエクスプローラで開きます。

— *Copy From Target*

プロジェクトを、選択されたターゲット用から現在のターゲット用に変換します。

— *Copy To Target*

プロジェクトを、選択されたターゲット用に変換します。

— *Select All*（**<Ctrl> + <A>**）

すべてのファイルを選択します。

4.8.3 プロジェクトの定義

プロジェクトにモジュールや CT ブロックを組み込むには、コンポーネントエディタの場合と同様、プロジェクトエディタの“Elements”ペインにモジュールや CT ブロックを追加します。

コンポーネントをプロジェクトに組み込む：

- プロジェクトエディタで、**Element → Add Item** を選択します。

  “Select Item...”ダイアログボックスが開きます。
- “Select Items...”ダイアログボックスの“1 Database”リストから、必要なコンポーネントを選択します。
- **OK** をクリックします。

  選択したコンポーネントがプロジェクトの“Elements”ペインに追加されます。

コンポーネントエディタにおけるエレメントの扱いと同様、プロジェクト内で実際に使用される各コンポーネントにはデフォルトのインスタンス名が割り当てられます。この名前は、そのプロジェクト内で使用されるコンポーネントのインスタンスに固有のものなので、元のコンポーネント自体には影響しません。

コンポーネントの名前を変更する：

- “Elements”ペインから、名前を変更したいコンポーネントを選択します。
- **Element → Rename** を選択します。
- 新しい名前を入力してから **<Enter>** を押します。

コンポーネントのインスタンスを削除する：

- “Elements”ペインで、削除したいコンポーネントを選択します。

- **Element → Delete** を選択します。

  そのコンポーネントのインスタンスが削除されます。これは、データベース内のコンポーネントには影響しません。

プロジェクトエディタから、そのプロジェクトを構成するコンポーネントのエディタを開いて各モジュールを編集することができます。これは、コンポーネントマネージャから各エディタを開いて編集するのと同じです。つまり、これによって機能記述が変更され、その変更内容はコンポーネントのすべてのインスタンスに反映されます。

プロジェクトエディタ内からコンポーネント用エディタを開くと、現在のプロジェクトの設定がそのエディタ内でも有効になります。たとえば、現在のプロジェクトのコード生成オプションが固定小数点コードに設定されている場合、エディタもやはり固定小数点コード用に設定されます。それに対し、エディタをコンポーネントマネージャから起動した場合は浮動小数点コードの生成しか行えません。

プロジェクト内のコンポーネントを編集する：

- プロジェクトエディタの“Elements”ペインから、編集したいコンポーネントを選択します。
- コンポーネント名をダブルクリックします。

または

- **Element → Edit Component** を選択します。

または

- **<Enter>** を押します。

  編集したいコンポーネント用のエディタが開きます。

エレメントのデータとインプリメンテーションをプロジェクト内で編集することができます。これらのコマンドは、すべてブロックダイアグラムエディタのコマンドと同じです。

プロジェクト、または内包されるコンポーネントに対して、注釈を加えることができます。これらの注釈は、自動生成されるドキュメントに出力されます。

プロジェクト用の注釈を編集する：

- プロジェクトの注釈を編集するには、**Component → Notes** を選択します。

または

- “Emelents”ペインまたは“Graphics”タブ上で、注釈を編集したいコンポーネントを選択します。

- **Element → Notes** を選択します。

  選択されたデータベースアイテムについて、注釈用エディタが開きます。詳細は 675 ページの「注釈」を参照してください。

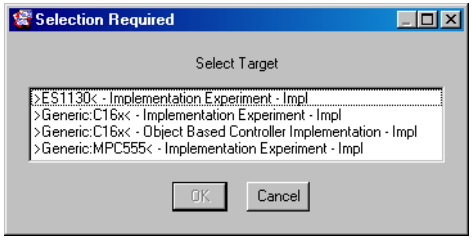
異なるターゲット、実験タイプ、インプリメンテーション用に作成された C コードとオペレーティングシステム設定（385 ページ参照）は、現在の環境用にコピーすることができます。

プロジェクト全体の C コードをコピーする：

あるプロジェクトに含まれるすべてのクラスとモジュールを別のターゲット、実験タイプ、インプリメンテーション用としてコピーするには、以下のようにします。

- プロジェクトエディタで、ターゲットとコード生成オプションを実際に使用するターゲットに合わせて設定します。設定方法は 4.8.8 項に説明されています。
- **Extra → Copy C-Code From** を選択します。

  “Selection Required”ダイアログボックスが開きます。



- コピー元とするターゲットを選択して **OK** をクリックします。

  C コードが、現在の環境用にコピーされます。

4.8.4 グローバル通信の定義

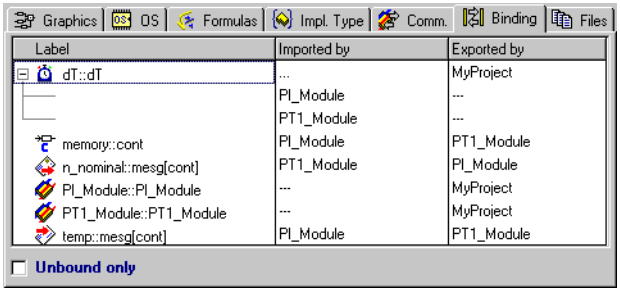
コンポーネントの場合、データの流れはコンポーネントのインターフェースエレメントを通じて、つまり入力と引数を定義し、出力と戻り値を読み取ることで実現されます。一方、プロジェクト内の通信は、グローバル変数のように機能する「インポートエレメント」と「エクスポートエレメント」により実現されます。これらのエレメントはそれぞれ名前によって結び付けが行われるので、結び付けられる 2 つのエレメントの名前は同じでなければなりません。そのため、それらのエレメントは、各モジュール内において適切な名前が割り当てられている必要があります。

エレメントは、コンポーネントまたはプロジェクトから「エクスポート」することができます。エクスポートされたエレメントは異なる複数のコンポーネントにインポートすることができますが、1 つのエレメントのエクスポートは 1 つのコンポーネントからしかできません。1 つのプロジェクト内の 2 つのコンポーネントに同じ名前のエクスポートエレメントが含まれていると、エラーメッセージが表示されます。

エレメントはプロジェクト内でも作成することができます。プロジェクト内で作成されたエレメントは、コンポーネント内の同じ名前を持つエレメントに結び付けられます。これは自動的に行われるため、結び付けるインポートエクスポートエレメントとエクスポートエレメントは、必ず同じ名前にしてください。

プロジェクト内の変数のインポート／エクスポート状態を表示する：

- プロジェクトエディタの“Binding”タブをクリックします。



プロジェクト用の変数のバインド状態が表示されます。

- このダイアログタブの一番下にある **Unbound Only** チェックボックスにチェックマークを付けて、表示項目をフィルタリングすることができます。

上の例には、内包されたモジュールが 2 つと、4 つのグローバルエレメントが表示されています。それぞれ以下のような内容です。

- `dT`：
  `MyProject` プロジェクトからエクスポートされる変数です。この変数はどのプロジェクトにもデフォルトで必ず作成され、プロジェクトが参照するすべてのモジュールにインポートされます。
- `PI_Module` および `PT1_Module`：
  内包される 2 つのモジュールです。
- `n_nominal`：
  モジュール `PI_Module` に定義されていて、ここからエクスポートされるメッセージです。このメッセージはモジュール `PT1_Module` にインポートされます。`PT1_Module` には同じ名前の受信メッセージが定義されています。

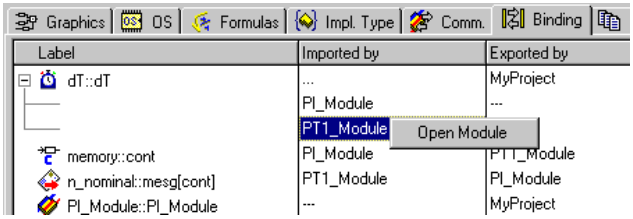
- `temp`：
  `PT1_Module` からエクスポートされ、`PI_Module` にインポートされるメッセージです。
- `memory`：
  `PT1_Module` からエクスポートされ、`PI_Module` にインポートされる変数です。

デフォルトで、送信メッセージと送受信メッセージはエクスポートエレメントとして定義され、受信メッセージはインポートエレメントとして定義されますが、これらの指定は変更することができます。変数は、インポートエレメントであるかエクスポートエレメントであるかを必ず明示的に宣言しなければなりません。1つのエレメントが複数のモジュールにインポートされる場合、上の例の変数 `dT` のように階層表示で明示されます。

もしもここで、同じエレメントが間違って 2 箇所以上でエクスポートされていることが発見された場合、“Binding”タブ上からエクスポート元のコンポーネントを開くことができます。

“Binding”タブからコンポーネントを開く；

- “Imported by”または“Exported by”カラムのコンポーネントを右クリックして、ショートカットメニューから **Open Module** を選択します。

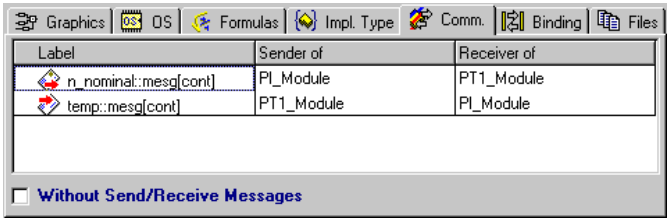


または

- 編集したいコンポーネントをダブルクリックします。

  選択されたコンポーネント用のエディタが開きます。

“Comm”タブをクリックすると、エレメントの対応関係の一覧を見ることができます。ここにはプロジェクト内で定義されているメッセージだけが表示され、送信メッセージから受信メッセージへの暗黙的なデータフローを確認することができます。



ここでは、メッセージと、そのメッセージが送信メッセージや受信メッセージとしてどのモジュール内で定義されているかだけが表示されます。ここに表示される項目も、このダイアログタブの一番下にあるフィールドを選択してフィルタリングすることができます。

モジュール内に、対応するエクスポートエレメントがないインポートエレメントがある場合、グローバルエレメントが自動的に作成され、結び付けが行われます。

プロジェクトのグローバルエレメントを定義する：

- **Global Elements → Resolve Globals** を選択すると、グローバルエレメントの対応付けが行われます。

注記

このコマンドを実行してからでないと、このプロジェクトの実験を開始することはできません。

Delete Unused Globals コマンドを使えば、プロジェクト内の使用されていないグローバルエレメントを削除することができます。

使用されていないグローバルエレメントを削除する：

- **Global Elements → Delete Unused Globals** を選択すると、使用されていないグローバルエレメントがすべて削除されます。

4.8.5 OS エディタでのスケジューリングの定義

プロジェクトエディタで行うもう 1 つの重要な作業として、プロセスのスケジューリングがあります。オペレーティングシステムは個々のプロセスを直接的には起動（「Activate」）しません。その代わり、プロセスは起動モードにより分類

され、タスクと呼ばれるシーケンスにまとめられます。これらのタスクがオペレーティングシステムにより起動され、そのタスクに割り当てられているプロセスのシーケンスが、指定されている順序で実行されます。

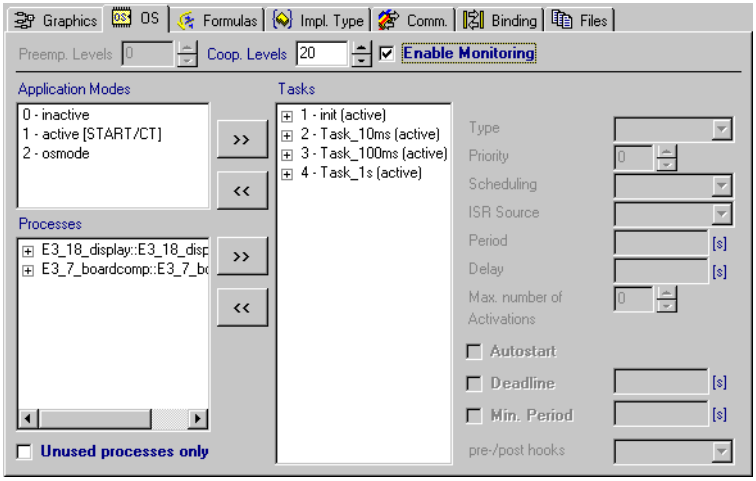
注記

オペレーティングシステムエディタの設定はすべてターゲット別に行う必要がありますが、ターゲット間で設定をコピーすることができます。

オペレーティングシステムの設定

オペレーティングシステムを設定する：

- プロジェクトエディタの“OS”タブをクリックして、オペレーティングシステムエディタを開きます。



- “Preemp. Levels”と“Coop. Levels”フィールドで、プロジェクトの協調レベル数とプリエンプティブレベル数を設定します。

  設定できる数は、ターゲットにより異なります。たとえばターゲットが PC の場合、PC のオペレーティングシステムではプリエンプティブマルチタスキングは行えないので、プリエンプティブレベル数は 0 です。

- デバッグ機能を使用する場合は、**Enable Monitoring** オプションをオンにします。

現在のターゲットのオペレーティングシステム設定を別のターゲットにコピーしたり、別のターゲットの設定を現在のターゲットにコピーできます。ターゲットの変更については、400 ページの「プロジェクトの設定」で説明されています。

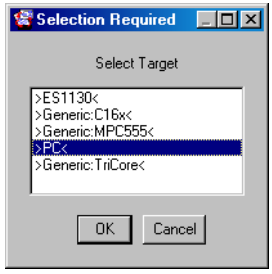
オペレーティングシステムの設定をコピーする：

- **Operating System → Copy To Target** を選択します。

または

- **Operating System → Copy From Target** を選択します。

  “Selection Required”ダイアログボックスが開きます。



- ターゲットを選択してから **OK** をクリックします。

  選択したコマンドに応じて、設定が現在のターゲットから選択したターゲットに、または選択したターゲットから現在のターゲットにコピーされます。

タスクとプロセス

本項では、オペレーティングシステムエディタを使用してオペレーティングシステムのスケジューリングを設定する方法について説明します。

タスクを作成する：

- **Task → Add** を選択します。

  新しいタスクが作成されます。

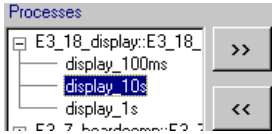
- “Tasks”フィールドで、新しいタスクの名前を入力してから **<Enter>** を押します。

必要なタスクを作成したら、各タスクにプロセスを登録します。

タスクにプロセスを割り当てる：

- “Tasks”フィールドから、プロセスを割り当てたいタスクを選択します。

- “Processes”フィールドから、そのタスクに割り当てたいプロセスを選択します。
- **Process → Assign** を選択します。



または

- >> ボタンをクリックします。

または

- ショートカットメニューから Assign を選択します。

プロセスがタスクに割り当てられます。このプロセスは、このタスクがスケジュールされるたびに実行されるようになります。

プロセスの割り当てを取り消すには、以下のようにします。

タスクへのプロセスの割り当てを取り消す：

- “Tasks”フィールドから、タスクへの割り当てを取り消したいプロセスを選択します。
- **Task → Deassign Processes** を選択します。

または

- **<<** ボタンを押します（上の画面を参照してください）。

または

- ショートカットメニューから **Deassign Processes** を選択します。

または

- プロセスを“Task”フィールドから“Processes”フィールドにドラッグします。

プロセスが、タスクから削除されます。

タスク内において各プロセスは、登録されている順番で実行されます。この順番の変更は、以下のように行います。

プロセスの順番を変更する：

- “Tasks”フィールドからプロセスを選択します。
- **Task → Move Up** または **Task → Move Down** を選択して、タスク内のプロセスの位置を移動させます。

プロセスを編集する場合は、“Tasks”フィールドから各コンポーネントを開くことができます。

“Tasks”フィールドからコンポーネントを開く：

- “Tasks”フィールドで、編集したいプロセスを選択します。
- メニューバーから**Task → Open Module**を選択します。

または

- ショートカットメニューから **Open Module** を選択します。

または

- プロセスをダブルクリックします。

  コンポーネント用エディタが開きます。

アプリケーションモード／タスク／プロセスの割り当てをチェックする：

1. タスクに割り当てられたプロセスが定義されているモジュールを確認する：
   - “Tasks”フィールドからプロセスを選択します。
   - そのプロセスを右クリックし、ショートカットメニューから **Show in Processes** を選択します。

   または

   - **Tasks → Show in Processes** を選択します。

     選択されたプロセスが、“Process”フィールド内で強調表示されます。
2. プロセスが割り当てられたタスクを確認する：
   - “Process”フィールドからプロセスを選択します。
   - そのプロセスを右クリックし、ショートカットメニューから **Show in Tasks** を選択します。

   または

   - **Process → Show in Tasks** を選択します。

     選択されたプロセスが、“Tasks”フィールド内で強調表示されます。
3. タスクに割り当てられたアプリケーションモードを確認する：
   - “Tasks”フィールドからタスクを選択します。
   - そのタスクを右クリックし、ショートカットメニューから **Show in Application Mode** を選択します。

   または

- **Tasks → Show in Application Mode** を選択します。

  選択されたタスクが割り当てられたアプリケーションモードが、“Application Mode”フィールド内で強調表示されます。

4. アプリケーションモードに割り当てられたタスクを確認する：
   - “Application Mode”フィールドからアプリケーションモードを選択します。
   - そのアプリケーションモードを右クリックし、ショートカットメニューから **Show in Tasks** を選択します。

   または

   - **Application Mode → Show in Tasks** を選択します。

     選択されたアプリケーションモードに割り当てられたタスクが、“Tasks”フィールド内で強調表示されます。

タスクの基本設定：

タスクのコンフィギュレーションとして、タスクの優先度、トリガモード、起動と実行に関するその他の詳細を指定します。

タスクの基本設定を行う：

1. トリガモード：

   以下のトリガモード（起動モード）から選択できます。

   - Alarm タスク：
     “Period”フィールドで定義された周期の初めごとに 1 回起動されます。
   - Init タスク：
     そのタスクが割り当てられているアプリケーションモードの開始時に 1 回だけ起動されます。
   - Software タスク：
     オペレーティングシステムコマンドにより起動されます。
   - Interrupt タスク：
     ハードウェアイベントにより起動されます。設定できるハードウェアイベントはターゲットにより異なります。たとえば、ターゲットがトランスピュータボードの場合、チャンネル 0 でデータを受信したときに Interrupt タスクを起動するように設定することができます。
   - Timetable タスク（マイクロコントローラターゲットで、OS は ERCOS^{EK} を使用する場合のみ）：
     タイマテーブルに書き込まれるアラームタスクです。ランタイムは短縮されますが、余分なメモリが必要となります。

- “Type”コンボボックスから上記のトリガモードを選択します。

2. スケジューリング：
   - “Scheduling”コンボボックスで、タスクのスケジューリングモードを選択します。

     このオプションは、Init タスクと Interrupt タスクには使用できません。

`cooperative`（協調）	協調タスク A 起動された時にそれよりも優先度の低いタスク B が実行中であった場合、実行中のプロセスが終了するまでタスク B はプリエンプト（中断）されません。
`preemptive`（プリエンプティブ）	プリエンプティブタスク A が起動されたときにそれよりも優先度の低いタスク B が実行中であった場合、そのタスク B は直ちにプリエンプトされ、タスク A の処理が終了した後、タスク B の処理が再開されます。
`non-preemptable`（ノンプリエンプタブル）	実行中のノンプリエンプタブルタスクは、より高い優先度レベルのタスクが起動されてもプリエンプトされません。

ERCOS^{EK} において、プリエンプティブタスクはすべての協調タスクよりも高い優先度を持つため、以下のような相互関係が成り立ちます。

- 協調タスクがプリエンプティブタスクをプリエンプトすることはありません。
- 協調タスク同士では、現在実行中のプロセスが終了後（タスクにプロセスが 1 つしかない場合はタスク終了後）にプリエンプトが行われます。
- 起動されたプリエンプティブタスクは、必ず他のタスクをすぐにプリエンプトします。

スケジューリングモードについての詳細は、『ASCET リファレンスガイド』の 1.1.1 項に記載されています。

注記

特定のマイクロコントローラ（MPC555 等）では、ノンプリエンプタブルタスクの使用には特殊な条件が必要となります（各ターゲットのリファレンスガイドを参照してください）。条件が満たされていない場合、コード生成時にエラーが発生します。
RTA-OSEK 用 ASCET-SE の場合は、ここで選択するモードの種類が異なります。詳しくは『ASCET-SE ユーザーズガイド』をお読みください。

3. 優先度：

   複数のタスクが起動されて同時にレディ状態となった時、タスクに割り当てられた優先度によってそれらのタスクを実際に実行する順番が決まります。

   たとえば、いつくかの アラームタスクに対して同時にトリガがかかり、これらのタスクが一度に起動されてレディとなった場合、それらのうちで最も優先度の高いタスクが最初に実行されます。また、実行されているタスクよりも優先度の高いプリエンプティブタスクまたはノンプリエンプタブルタスクがレディになると、そのタスクはプリエンプトされ、実行が保留されます。優先度機構については『ASCET リファレンスガイド』の 1.1.2 項を参照してください。

   - “Priority”フィールドに数値を入力して、タスクに優先度を割り当てます。

   注記

   ノンプリエンプタブルタスクとプリエンプティブタスクは同じ優先度領域を共有し、協調タスクは個別の優先度領域を使用します。

   このオプションは Init タスクの場合は無効です。

4. トリガ：

   この設定は、Interrupt タスクのトリガとしてどのイベントを使用するかを指定するもので、他のタイプのタスクには設定できません。
   またコンボボックスで選択できるオプションの種類は、現在のターゲットによっても異なります。

   - “ISR Source”コンボボックスで、トリガを選択します。

5. その他：

   タスクが起動される周期を指定します。たとえば、この値を 0.01 に設定すると、タスクは 100 分の 1 秒ごとに 1 回起動されます。

   - “Period”フィールドに周期を秒数で入力します。

     この設定は Alarm タスクと Timetable タスクについてのみ有効です。

   ディレイ時間が設定されていると、そのタスクはその時間が経過するまで起動されません。ディレイを 0 に設定すると、そのタスクは、プログラムの起動直後に 1 回、そしてその後は各周期の初めに 1 回ずつ起動されます。

- ディレイの値を“Delay”フィールドに秒数で入力します。

  この設定は Allarm タスクと Timetable タスクについてのみ有効です。

1 つのタスクを並行して起動できる回数を指定します。タスクが前回起動されてからその実行が完了する前にもう一度起動された場合、そのタスクは 2 回（2 重に）起動されていることになります。システムリソースを節約するために、各タスクを最高何回まで並行して起動できるかを限定しておくことができます。

- “Maximum No. of Activations”フィールドの値を調整します。

  この設定は Alarm、Timetable、Software タスクについてのみ有効です。

フックルーチンの設定：

各タスクごとに、デバッグ用のフックルーチンを生成することができます。

注記

この設定は、RTA-OSEK 用 ASCET-SE では使用されません。

タスクのフックルーチンを設定する：

- **Enable Monitoring** オプションがオンになっているかどうか（チェックボックスがチェックされているかどうか）を確認します。
- “pre-/post hooks”コンボボックスで、以下のデバッグ機能を選択します。

`none`	デバック機能なし
`monitoring`	完全なデバック機能

コード生成時には、ここで選択された条件にあったデータ構造体が作成され、必要な ERCOS^{EK} ライブラリが統合されます。その際は、ASCET によって生成される E_HOOKS という Make 変数が使用されます。以下に、オペレーティングシステムエディタでの設定内容と E_HOOKS 変数との関係を示します。

Enable Monitoring	pre-/post hooks	生成される E_HOOKS の値	コード内での dT の使用
オン	1 つ以上のタスクを“monitoring”に設定	MONITORING	可（その他のデバッグ情報も取得可能）
オフ	1 つ以上のタスクを“monitoring”に設定	DTONLY	可
オンまたはオフ	すべてのタスクが“none”	DTONLY	不可

実験ターゲットでのシミュレーションにおいては、monitoring オプションによって、モニタ用変数（393 ページ参照）の生成が制御されます。このオプションの機能はマイクロコントローラターゲットに依存するため、ASCET-SE のユーザーズガイドに詳しく説明されています。

OSEK 固有の設定：

OSEK 仕様に対応するために、OSEK 固有の設定項目も用意されています。これらの設定もマイクロコントローラターゲットに依存するものであるため、ASCET-SE ユーザーズガイドを参照してください。

デッドラインと最小起動周期を設定する：

- 同じタスクの起動間隔がある一定のリミットを超えて長くならないように指定する場合は、**Deadline** オプションをオンにします。

  チェックボックスをチェックしてオプションがオンになると、入力フィールドが有効になります。
- 連続する 2 回の起動の最大間隔を秒数で指定します。

  Deadline オプションは、Alarm、Timetable、Software タスクにのみ有効です。
- 逆に、同じタスクの起動間隔が、ある一定のリミットを下回らないように指定する場合は、**Min. Period** オプションをオンにします。

  チェックボックスをチェックしてオプションがオンになると、入力フィールドが有効になります。
- 連続する 2 回の起動の最小間隔を秒数で指定します。

  Min. Period オプションは、Interrupt タスクにのみ有効です。

Autostart オプションは、Software タスクと Alarm タスクについてのみ有効です。このオプションは、タスクのタイプにより異なる意味を持ちます。

オートスタートオプションを設定する：

- “Task”ペインで、Software タスクを選択します。
- **Autostart** オプションをオンにします。

  そのソフトウェアタスクは、対応するアプリケーションモードの初期化時に 1 回のみ実行されるようになります。

または

- “Task”ペインで、Alarm タスクを選択します。

- **Autostart** オプションをオンにします。

  そのタスク用のタイマのカウントが、対応するアプリケーションモードの初期化時に開始されるようになります。

モニタリングオプション

モニタリングオプションとして、プロジェクトの実験時に各タスクの監視やデバッグに使用できる 3 つの変数と全タスクの監視に使用できる 1 つの変数が用意されています。タスクの設定において、**Enable Monitoring** オプションがオンになっていてかつ“pre-/post hooks”コンボボックスで `monitoring` が選択されていると、そのタスク専用に以下の 3 つのモニタ用変数が生成されます。

`cycleStartTime_<taskname>`	タスクの前回の実行開始時刻を示します。
`cycleTime_<taskname>`	タスクの実行所要時間を示します。
`dt_<taskname>`	タスクの前回と今回の実行開始時刻の間隔を示します。

さらに全タスク用に以下の変数が生成されます。

`runtime_violation`	タスクがスケジューリングに対応できなかった回数を示します。

モニタリングオプションを有効にする：

- “Tasks”リストで、1 つまたは複数のタスクを選択します。
- タブの右下部分にある Monitoring オプションをオンにします。

  次回のコード生成時に、各タスク用のモニタ用変数が生成されます。これらの変数は、測定ウィンドウに表示したり、またオフライン実験の場合はデータロガーに書き込むことができます（582 ページの「データロガー」を参照してください）。

アプリケーションモード

オペレーティングシステムには複数のアプリケーションモードを設定することができ、各アプリケーションモードに一連のタスクが割り当てられます。このようにして、たとえばエンジン制御システムの場合、暖機運転用と通常運転用の処理を別モードとして定義するなど、制御対象システムの複数の動作モードをモデリングすることができます。

アプリケーションモードを作成する：

- **Application Mode → Add** を選択します。

  新しいアプリケーションモードが作成され、“Application Modes”フィールドに表示されます。
- 新しいアプリケーションモードの名前を入力してから **<Enter>** を押します。
- 新しいアプリケーションモードを開始モードにする場合には、**Application Mode → As Start Mode** を選択します。

  システムが起動されると、システムは、開始モードとして宣言されたアプリケーションモードになります。アプリケーションモードの切り替えは、オペレーティングシステムコマンドにより行われます。

タスクにアプリケーションモードを割り当てる：

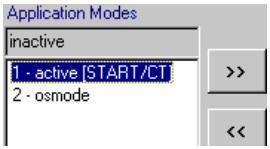
- “Application Modes”ペインから、1 つまたは複数のアプリケーションモードを選択します。

  <Ctrl> キーを押し下げたままの状態で複数のアプリケーションモードをクリックして、それらを一度に選択することができます。
- 選択したアプリケーションモードに割り当てたいタスクを、“Tasks”フィールドから選択します。
- **Application Mode → Assign** を選択します。

または

- “Application Modes”フィールドのアプリケーションモードを選択し、“Task”フィールドのタスクにマウスポインタでドラッグします。

  1 つのタスクに複数のアプリケーションモードを割り当てることができます。タスクは、それに割り当てられているアプリケーションモードのいずれかが有効になると、そのタスク設定に応じてスケジュールされます。



- アプリケーションモードの割り当てを解除するには、**Task → Deassign Application Modes** を選択します。

  メニューの代わりに右のボタンを使用することもできます。

注記

あるアプリケーションモードから別のアプリケーションモードへの切り替えは、オペレーティングシステムのサービスコールによって行われます。詳しくは ERCOS^{EK} マニュアルを参照してください。

4.8.6 外部プロジェクトファイルの管理

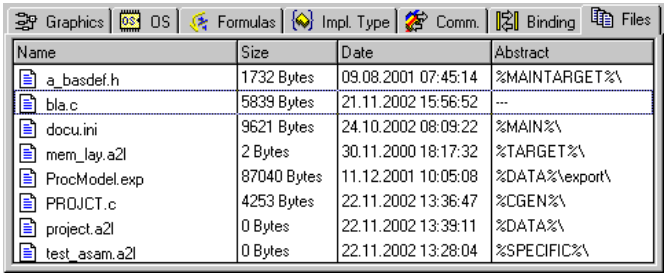
一般的に、「プロジェクトファイル」、つまり 1 つのプロジェクトに属するファイルとしては、仕様書や C コードファイル、またログファイルやその他の各種文書などがあります。これらの外部ファイルをプロジェクトエディタの“Files”タブで一括管理して保護することができます。

“Files”タブには、各ファイルの名前、サイズ（バイト数）、および作成日付の一覧が表示されます。ファイル名には絶対パスが含まれますが、ファイルが以下の 6 つのデフォルトパスのいずれかに含まれる場合のみ、相対パスが使用されます。その場合、“Abstract”カラムに以下のキーワードが表示されます。

キーワード	パス
%MAIN%	ETAS¥Ascet5.2（ASCET インストールディレクトリ）
%CGEN%	ETAS¥Ascet5.2¥cgen（コード生成用ターゲットディレクトリ）
%MAINTARGET%	ETAS¥Ascet5.2¥target（メインターゲットディレクトリ）
%TARGET%	ETAS¥Ascet5.2¥target¥<target name> （現在有効なターゲット用のターゲットディレクトリ） 例：PC ターゲットの場合は ETAS¥Ascet5.2¥target¥PC
%DATA%	ETASData¥Ascet5.2（ユーザーデータディレクトリ）
%SPECIFIC%	“Options”ウィンドウ（49 ページ参照）で設定されたユーザー定義のパス

外部プロジェクトファイルを追加する：

- プロジェクトエディタの“Project Files”タブを開きます。



このタブには、関連付けられているプロジェクトファイルがすべて表示されます。

- **Project Files → Add** を選択します。

  ファイル選択ダイアログボックスが開きます。

- ファイルを選択してから **<Enter>** を押します。

  選択したファイルが、プロジェクトエディタに関連付けられます。

“Files”タブから各ファイルを開くことができます。

プロジェクトファイルを表示する：

- “Files”タブで、表示したいプロジェクトファイルを選択します。
- メニューバーから **Project Files → View** を選択します。

または

- ショートカットメニューから **View** を選択します。

  プロジェクトファイルが、Windows で関連付けられたアプリケーションで開きます。
  アプリケーションが関連付けられていない場合、このコマンドは機能しません。

プロジェクトエディタのプロジェクトファイルを削除する：

- プロジェクトエディタから削除したいファイルを選択します。
- **Project Files → Delete** を選択します。

  選択したファイルがタブ上から削除されますが、ファイル自体は削除されません。

プロジェクトファイルを書き込む：

- プロジェクトエディタで、1 つまたは複数のファイルを選択します。
- **Project Files → Write Selected File(s)** を選択して、デフォルトのディレクトリに書き込みます。

または

- **Project Files → Write Selected File(s) to** を選択して、指定のディレクトリに書き込みます。

または

- **Project Files → Write All Files** を選択して、すべてのファイルをデフォルトディレクトリに書き込みます。
- プロジェクトファイルのデフォルトディレクトリの位置は、**Project Files → Show Default Path** で確認することができます。
- **Project Files → Explore Default Path** を選択すると、選択されたファイル用デフォルトディレクトリの内容が、エクスプローラで表示されます。

注記

1 つ以上のプロジェクトファイルが選択されている時は、**Project Files → Show Default Path** と **Project Files → Explore Default Path** は使用できません。

ディスクに保存されたプロジェクトファイルの内容が更新された場合、以下のようにしてファイルの情報を更新することができます。

プロジェクトファイルを更新する：

- プロジェクトエディタで、1 つまたは複数のファイルを選択します。
- **Project Files → Update** を選択します。

  データベース内のプロジェクトファイルの最新情報が表示されます。

データベースには常に 1 つ前のバージョンがバックアップされているので、以下のようにして、バックアップされたバージョンに戻ることができます。

プロジェクトファイルをバックアップされたバージョンに戻す：

- プロジェクトエディタで、1 つまたは複数のファイルを選択します。

- **Project Files → Replace With Backup Content** を選択します。

  選択されたプロジェクトファイルは、バックアップされていたバージョンに戻ります。

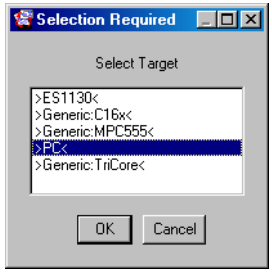
プロジェクトファイルをコピーする：

- **Project Files → Copy From Target**を選択します。

または

- **Project Files → Copy To Target** を選択します。

  “Selection Required”ダイアログボックスが開きます。



- ターゲットを選択してから **OK** をクリックします。

  選択したコマンドに応じて、ファイルが現在のターゲットから選択したターゲットに、または選択したターゲットから現在のターゲットにコピーされます。

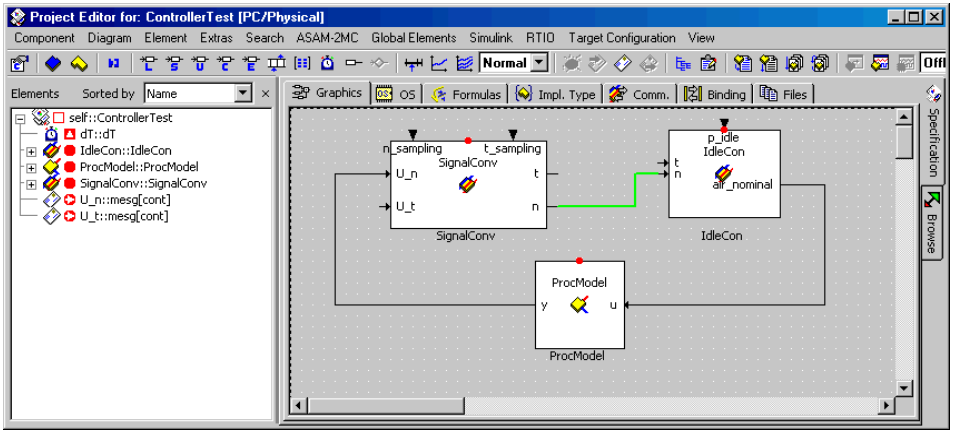
4.8.7 ハイブリッドプロジェクト

ハイブリッドプロジェクトは、連続系ブロック（CT ブロック）とモジュールの両方を内包するプロジェクトです。CT ブロックは、組み込み制御システムにより制御される物理プロセスをモデリングしたものです。

ハイブリッドプロジェクトをセットアップする：

- 本章で前述した方法で、プロジェクトのモジュールをセットアップします。
- 必要な CT ブロックを組み込みます。
- プロジェクトエディタの“Graphics”タブをクリックします。
- ハイブリッドプロジェクトに組み込もうとするCTブロックとモジュールを、描画領域にドラッグします。

- CT ブロックとモジュールを、ブロックダイアグラム内の一般的なエレメントの場合と同じ要領で接続します。



各コンポーネントのレイアウトは、226 ページの「内包されるコンポーネントのレイアウト」の項で説明されている方法で任意に変更できます。

CT ブロック間の通信や CT ブロックとモジュールの間の通信は、ブロックダイアグラム間の通信と同様、接続線に従って行われます。また現バージョンの ASCET では、モジュール同士もこの方法で接続はできますが、モジュール間の接続線は、モジュール間の実際の通信には使用されません（モジュール間の通信は、前述のとおり、名前での結び付けによって行われます）。つまり、モジュール間の接続線は「コメント線」として解釈され、ダイアグラムの視認性を高めるためにのみ使用されます。コメント線の色は、ASCET ユーザーオプションで変更できます（43 ページの「表示オプション」を参照してください）。

CT ブロックを使用するために必要なタスクはすべて自動的に生成されるので、任意にタスクを追加することはできません。1 つのハイブリッドプロジェクトについて、`initialize` および `terminate_CT` という 2 つのタスクが生成され、またプロジェクト内の各 CT ブロック用に `simualte` タスクと `event` タスクがそれぞれ生成されます。これらのタスクは、他のタスクと同様の設定が行えます（388 ページ参照）。

CT ブロックのスケジューリングを定義する：

- “OS” タブをクリックします。
- “Task” ペインの CT タスクを選択します。
- トリガモードなどの各項目を設定します。

これで、ハイブリッドプロジェクトの実験を、モジュールのみを含むプロジェクトの実験と同様に行うことができるようになりましたが、さらに、実験環境において各 CT ブロックにソルバを割り当てる必要があります。1 つのプロジェクト内

の各ブロックに異なるソルバを割り当てることにより、マルチレート実験を行うこともできます。ソルバの割り当てについては、333 ページの「連続系ブロックの検証」という項で詳しく説明されています。

4.8.8 プロジェクトの設定

ASCET では、さまざまなターゲットプラットフォーム上（PC, 実験ターゲット、マイクロコントローラターゲット）で実験を行うことができ、各ターゲットごとに、さまざまなコード生成オプション（ビルドオプション、実験および製品用コードコードオプション）や、最適化オプション、さらに ASAM-MCD-2MC ファイルの生成オプションなどが用意されています。

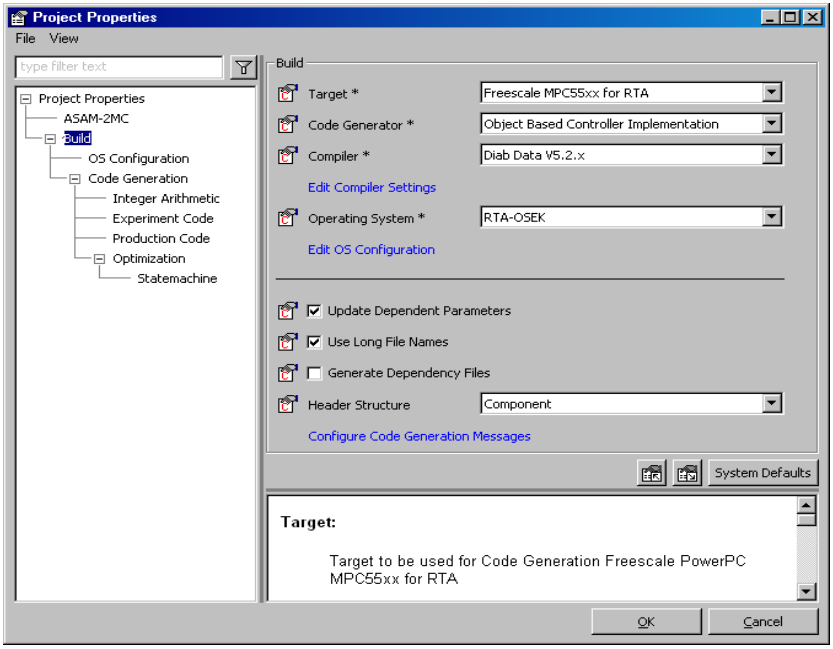
注記

ASCET の個々のモジュールに関して、プロジェクトなしで単体でコード生成とシミュレーションを行えるのは、コードジェネレータとして `Physical Experiment`（405 ページ参照）が選択されている場合のみです。他のコードジェネレータを使用して実験を行う場合、モジュールは必ずプロジェクトに組み込まれている必要があります。このため、各クラスやモジュールには、デフォルトプロジェクトと呼ばれるダミーのプロジェクトが定義され、これを利用しないとインプリメンテーション（実装情報）にアクセスできません。プロジェクトが存在しないと、インポートされるエレメントの変換式やインプリメンテーションは失われてしまいます。

プロジェクトのオプションを設定する：

- **Project Properties** ボタンをクリックします。

  “Project Properties”ウィンドウが開き、“Build”ノードが表示されます。



- 変更したいオプションのコンボボックスを開いて設定内容を選択します。

または

- チェックボックスでオン／オフを切り替えます。

または

- 入力フィールドに値を入力します。

または

- リンク（青いテキスト）をクリックして関連する他のオプションを設定します。
- 変更したいすべてのオプションについて、上記の操作を行います。

- デフォルト設定に戻すには、**System Default** をクリックします。

注記

System Default ボタンをクリックするとすべてのタブのオプションがデフォルト設定に戻ります。

- さらに別のタブに切り替えて設定を行います。

または

- **OK** をクリックして設定を確定し、ダイアログボックスを閉じます。

“Build”ノードのいくつかのサブノードにおけるオプション設定により、データベース内の生成済みコードが無効になる場合があります。使用されるサブノードは、選択されたコードジェネレータ（405 ページ参照）に応じて異なります。以下の表を参照してください。

コードジェネレータ	サブノード
`Physical Experiment` `Quantized Physical Experiment`	"Code Generation"（407 ページ） "Experiment Code"（412 ページ） "Optimization"（413 ページ）
`Implementation Experiment`	"Code Generation" "Integer Arithmetic"（410 ページ） "Experiment Code" "Optimization"
`Object Based Controller Implementation`	"Code Generation" "Integer Arithmetic" "Production Code"（413 ページ） "Optimization"

オプションアイテムをフィルタリングすることにより、プロジェクト設定をよりわかりやすく表示することができます。

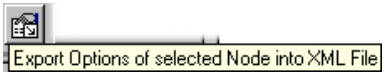
プロジェクト設定をフィルタリングする：

- 41 ページの「オプションをフィルタリングする：」の方法で、プロジェクト設定をフィルタリングします。

設定内容は XML ファイルに保存することができ、保存された設定を任意に読み込むことができます。この操作を行うには、“Project Properties”の最上位ノードを開いておく必要があります。

プロジェクト設定の保存と読み込みを行う：

1. プロジェクト設定の保存



- “Project Properties”ウィンドウで、“Project Properties”ノードを選択します。
- **Export Options of Selected Node into XML File** ボタンをクリックします。

  Windows のファイル選択ダイアログボックスが開きます。ファイルフォーマットには `*.xml` が選択されています。
- ファイルのパスと名前を指定します。
- **Save** をクリックします。

  現在の設定がファイルに保存されます。

2. プロジェクト設定の読み込み

- “Project Properties”ウィンドウで、“Project Properties”ノードを選択します。
- **Import Options of Selected Node from XML File** ボタンをクリックします。

  設定を上書きしてよいかを確認するダイアログボックスが開きます。
- 今後、この確認ダイアログが表示されないようにするには、**Show next time** オプションをオフにします（46 ページ参照）。
- 上書きしてよければ **OK** をクリックします。

  Windows のファイル選択ダイアログボックスが開きます。
- XML ファイルを選択します。
- **Open** をクリックします。

  指定したファイルの内容が読み込まれます。

プロジェクトの設定は、ASAM-MCD-2MC の設定（“ASAM-2MC”ノード）とビルドオプション（“Build”ノード）に分かれていて、ビルドオプションにはさらにいくつかのサブノードが含まれます。

ASM-MCD-2MC オプション（“ASAM-2MC”ノード）

“ASAM-2MC”ノードには以下のオプションが含まれます。

- *ROM Code*：実験ターゲット用のROMコードの生成を行うかどうかを切り替えます。

  このオプションがオンになっていると、実行ファイル（`*.cod` ファイル）に ROM コードと RAM コードの両方が格納され、スタンドアロン動作（ASCET-RP のユーザーズガイドを参照してください）が可能になります。1 つの実験ターゲットについて 1 つの実行ファイル（`*.cod` ファイル）が生成されます。

- *Hierarchical Names* ：ASAM-MCD-2MC ファイル内のエレメント名をフルネームにするかどうかを切り替えます。

  フルネームを使用すると、大きなプロジェクト内での名前の重複を避けることができます。ASAM-MCD-2MC ファイル生成時に、`<elementName>[.<className>][.<moduleName>]` というフルネームが付けられます。

- *Formulas with unit* ：変換式内に測定単位を書き込む（オプション = オン）かどうかを指定します。

- *Suppress exported parameters*：ASAM-MCD-2MC ファイル生成時に、エクスポートされるパラメータを省略する（オプション = オン）かどうかを指定します。

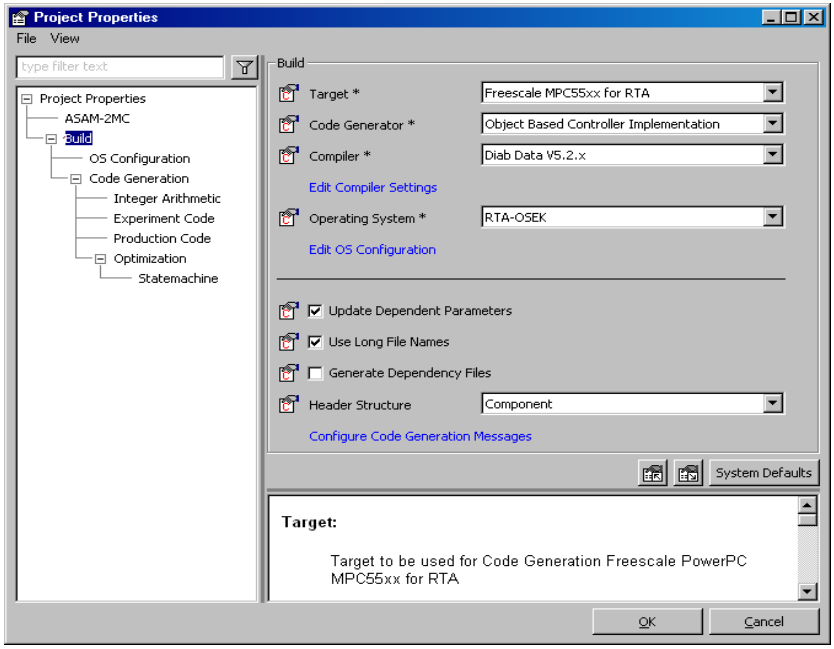
  コンパイラによっては、エクスポートされるパラメータへのアクセスを最適化し、値に置き換える場合があります。このようなパラメータを適合しようとすると、それをエクスポートするコンポーネントとインポートするコンポーネントとの間で矛盾が発生する可能性があるので、このオプションによってそのようなパラメータの適合処理を抑制します。

- *Suppress exported of prototypes*：ASAM-MCD-2MC ファイル生成時に、プロトタイプコンポーネントからエクスポートされるエレメントを無視する（オプション = オン）かどうかを指定します。

- *Suppress grouping information* ：ASAM-MCD-2MC ファイル生成時に、測定変数と適合変数と GROUP や SUBGROUP キーワードで構造化する（オプション = オン）かどうかを指定します。デフォルトではこのオプションはオフになっています。

ビルドオプション（“Build”ノード）

“Build”ノードには以下のオプションが含まれます。



- *Target*：コード生成の対象とするターゲット（実験ターゲットまたはマイクロコントローラ）を指定します。ASCET の基本システムで使用できるターゲットは PC のみで、ASCET-RP をインストールすることによって各実験ターゲットが追加され、さらに各マイクロコントローラ用の ASCET-SE によってマイクロコントローラターゲットが追加されます。
- *Code Generator*：生成されるコードに使用される算術演算機構の種類を規定します。以下の中から選択します。
  - `Physical Experiment`
  - `Quantized Physical Experiment`
  - `Implementation Experiment`

  マイクロコントローラ用の ASCET-SE がインストールされている場合、Target としてマイクロコントローラを選択すると、コードジェネレータは以下のものに固定されます。
  - `Object Based Controller Implementation`
- *Compiler*：プロジェクトのコード生成に使用するコンパイラを指定します。ASCET 製品には実験ターゲット用のコンパイラが含まれています。

- *Edit Compiler Settings*：このリンクをクリックすると ASCET オプションウィンドウが開き、現在 *Compiler* オプションで選択されているコンパイラのオプションを設定するためのサブノードが表示されます。
- *Operating System*：現在のターゲット用のオペレーティングシステムが表示されます。RTA-OSEK 用の ASCET-SE がインストールされている場合、Target として各マイクロコントローラを選択すると、コンボボックスで RTA-OSEK のバージョンを選択できます。
- *Edit OS Configuration*：“OS Cofiguration”ノードを開きます（407 ページを参照してください）。
- *Use Long File Names*：このオプションがオフになっていると、生成されたファイルに 8.3（“filename.ext”）パターンのファイル名が使用され。オンになっていると、ロングファイルネーム（エレメント名と同じ名前）が使用されます。

注記

このオプションは、ロングファイルネームをサポートするコンパイラについてのみ使用できます。DIAB 4.1 および Borland コンパイラが選択されていると、このオプションはグレイアウトされます。

- *Generate Dependency Files* ：このオプションがオンになっていると、自動生成されたすべてのヘッダファイル（つまりユーザーが `#include` 文を使用してインクルードしたヘッダファイル以外のもの）の依存関係が含まれる Make ファイルが生成されます。Make ファイルは、生成された `*.c` ファイルごとに生成されます。
- *Header Structure*：**Component → File Out Generated Code** コマンドによるヘッダファイル生成を制御します。

  以下の設定を選択できます。

  `Component` プロジェクト内の各クラス／モジュールごとに個別のヘッダファイルが生成されます。モジュールにクラスが含まれる場合、モジュールヘッダファイルにクラスのヘッダ情報が挿入されます。
  1 つのクラスがプロジェクト内の複数のモジュールに含まれる場合は、そのクラスのヘッダ情報が全モジュールのヘッダファイルに挿入されます。

  `Module` プロジェクト内の各モジュールごとに個別のヘッダファイルが生成されます。

  `Project` プロジェクト全体で 1 つのヘッダファイルが生成されます。このファイルに、プロジェクト内のすべてのモジュールとクラスの情報が含まれます。
  このヘッダファイルはすべての `*.c` ファイルにインクルードされる必要があります。

注記

このオプションは、内部／外部 C コードのヘッダと OS コンポーネントのヘッダには影響しません（外部 C コードクラスについては、**user header global** オプションがオンになっている場合を除く）。OS コンポーネントのヘッダの場合は、常に `conf.c` および `conf.h` ファイルに書き込まれます。

- *Configure Code Generation Messages*：このリンクをクリックすると、“CodeGen message Configuration”ダイアログボックスが開きます。ここではコード生成とビルド処理によって発行されるメッセージの表示設定を変更することができます。詳しくは 153 ページの「“CodeGen Message Configuration”ダイアログボックスでのメッセージ設定」を参照してください。

OS コンフィギュレーションオプション（“OS Configuration”ノード）

このノードは、RTA-OSEK 用 ASCET-SE がインストールされていて、かつビルドオプション（405 ページ参照）で所定のマイクロコントローラターゲットが選択されている場合にのみ使用されます。

コード生成オプション（“Code Generation”ノード）

このノードには以下のオプションが含まれます。

- *Protected Division*：このオプションがオンになっていると、生成されるコード内に、ゼロ除算を検知してその実行を止める機能が挿入されます。このオプション機能はパフォーマンスの低下を招くため、コードの効率性を最重要視する場合にはオフにしてください。
- *Generate Define Directives for Enum Values*：このオプションがオフ（デフォルト）になっていると、生成されたコード内で、整数値が列挙子として使用されます。

  このオプションがオンになっていると、コード内では各列挙子の名前（`Red`、`Green` など）が使用され、さらに、以下のようなマクロが生成されます。

  ```
  #define Red 0
  #define Green 1
  ```

  これらのマクロにより、各シンボル名に整数値が割り当てられます。
- *Add Comment with Implementation Information for each Assignment Statement*：このオプションがオンになっていると、各代入文とリターン文の前にコメントが生成されます。このコメントには、代入の右項の 実装情報が含まれます。

  例：

  ```
  /* return with expr from doAddition:          ↵
       min=-65536, max=65534, hex=1phys+0,       ↵
       limit=(maxBitLength: true, assign: true)*/
  ```

```
return ((sint32)input1 + input2);
```

- *Add Comment with Specification Source for each Statement*：このオプションがオンになっていると、各代入文の前にコメントが生成されます。このコメントには、代入文のソース内の位置を示す情報が含まれます。
  - ESDL の場合：行番号
  - ブロックダイアグラムの場合：シーケンスコール番号
  - ステートマシンの場合：アクション／コンディション、およびステート／トランジションの名前

  例（ESDL の場合）：

```
/* doAddition: line #1 */
self->cont->val = input1 + input2;
```

- *Add Comment with Generation Information for each Component*：このオプションがオンになっていると、各コンポーネントのコードの前にコメントが生成されます。最初のコメントにはコンポーネントと ASCET についての情報が含まれ、2 番目のコメントにはビルドオプションおよび実験／製品用コードオプションが含まれます。

  例：

```
/***********************************************
 * BEGIN: Generation Information
 *---------------------------------------------
 * Component:..............Project
 * Name:..................."ControllerTest"
 * Implementation:........."Impl"
 * Dataset:................"Data"
 * Specification:.........<not applicable>
 * Version:...............<empty String>
 *---------------------------------------------
 * Generation Date:.......21.03.2006
 * Generation Time:.......16:34:44
 *---------------------------------------------
 * ASCET-MD Version:......V5.1.4
 * ASCET-RP Version:......V5.4.1
 * ASCET-SE Version:......V5.2.1 CID[514]
 *---------------------------------------------
 * END: Generation Information
 ***********************************************/
/***********************************************
 * BEGIN: Project Options "Build"/"Code"
 *---------------------------------------------
 *    Build
```

```
*-----------------------------------------------
* Code Generator:.......Implementation Experiment
* Compiler:............Borland-C V4.5
* Operating System:....<not applicable>
* Target:..............PC
*-----------------------------------------------
*    Code
*-----------------------------------------------
* add comment with generation information for  ↵
    each component [true]:.true
...
*-----------------------------------------------
*    Code.Experiment
*-----------------------------------------------
* data logging [false]:.....................false
...
*-----------------------------------------------
*    Code.FixedPoint
*-----------------------------------------------
* allow double bit size for division numerators ↵
    [true]:.............true
...
*-----------------------------------------------
*    Code.Optimizations
*-----------------------------------------------
* common subexpression elimination [true]:....true
...
*-----------------------------------------------
* END: Project Options "Build"/"Code"
************************************************/
```

- *Force parenthesis for binary logical operators*: 2 進論理演算が必ず括弧で囲まれます。これは、安全性が重要視されるシステム内での C 言語の使用方法を規定した MISRA[1] -C 2004 Guidelines を順守するためのものです。
- *Add parentheses for readability*：このオプションがオンになっていると、コードを読みやすくするための括弧が追加されます。

  例：このオプションがオフの場合に生成されるコードの例です。

```
x = (a > b) ? a - b : a + b;
x = a + b * c;
```

[1] **M**otor **I**ndustry **S**oftware **R**eliability **A**ssociation, http://www.misra.org.uk

```
if (a + b == c) {.. }
```

オプションをオンにすることにより、上記のコードは以下のようになります。

```
x = (a > b) ? (a - b) : (a + b);
x = a + (b * c);
if ((a + b) == c) {.. }
```

整数演算オプション（"Integer Arithmetic" ノード）

このノードには以下のオプションが含まれます。

- *Arithmetic Services Set*：使用する算術演算サービスセット（4.14 項を参照してください）を選択します。コンボボックスには、現在のターゲットの `services.ini` ファイルに含まれるすべてのセットと、算術演算サービスを無効にするための `<None>` という項目が表示されます。

  Edit ボタンをクリックすると、算術演算サービスのインターフェースエディタが開きます（4.14.5 項を参照してください）。

- *Result on division by zero combo box*；ゼロ除算発生時の処理を選択します。`numerator`（デフォルト）が選択されていると、演算結果の分子は 0 になり、`user defined` を選択すると、ユーザー定義の C マクロ `protDiv0(num, result_type_max, result_type_min)` の呼び出しが行われます。このマクロは、インクルードされるヘッダファイルのうちのいずれかに定義されます。

  注記

  `user defined` を使用する場合は、`protDiv0` マクロが実際に存在していることを確認してください。このマクロの自動生成や構文チェックは一切行なわれません。

  マクロの引数の意味は以下のとおりです。

  — `num`：現在の分子の値

  — `result_type_max`：結果の値のデータ型で許容される最大値

  — `result_type_min`：結果の値のデータ型で許容される最小値

  2 番目と 3 番目の引数は、計算において、その値のデータ型の最小値と最大値を使用する際に使用されます。このマクロは以下のように定義しておくことも可能です。

  ```
  #define protDiv0(num,max,min) (num<0?min:max)
  ```

  このようにすると、現在の分子の値に応じて、データ型の最大値または最小値が得られます。

- *Maximum bit length (int)*：変数（整数）用コードの固定小数点演算の式に 8、16、32 ビット演算のうちのどれを使用するかを指定します。

- *Maximum bit length (float)*：変数（浮動小数点）用コードの浮動小数点演算の式に 32、64 ビット演算のどちらを使用するかを指定します。

- *Allow Limit Service for Assignment Limitation*：リミッタ機能付き算術演算サービスを使用する条件を指定します。このオプションがオフになっていると、プロセッサの最大ビット幅のデータの演算におけるオーバフローの発生を防ぐ目的でのみリミッタ機能が有効になり、オンになっていると、すべてのデータ型の演算（例：32 ビットプロセッサでの 16 ビット演算）のオーバーフローについてリミッタ機能が有効になります。

  古いバージョンの ASCET プロジェクトについては、バージョン変換時にこのオプションは自動的にオフになります。

- G*enerate Limiters (may be changed locally)*：計算された値がその代入先の変数のインプリメンテーションに指定されているインターバル（実装範囲）を超えた場合の処理、つまり値の範囲を制限するリミッタ処理のコードを生成するかどうかを指定します。このオプションがオフになっていると、リミッタは生成されません。オン（デフォルト）になっていると、エレメントのローカル設定が適用されます。

- *Allow Double bit Size for Division Numerators*：乗算の中間結果について、次に除算が行われる場合は *Minimum bit length* に設定されたビットサイズを 2 倍にします（デフォルト値：オン）。

- *Use SHIFT Operation on Signed Values instead of DIV Operation*：符号付きの値の除算のコード生成を行う際、実際の除算の代わりに右シフトを行うコードを生成します（デフォルト値：オン）。
  コンパイラによっては算術シフトの代わりに論理シフトを行うものがあり、そのような場合、このオプションがオンに設定されていると、符号エラーが発生する可能性があります。

- *Use SHIFT Operation on Signed Values instead of MUL Operation*：符号付きの値の乗算のコード生成を行う際、実際の乗算の代わりに左シフトを行うコードを生成します（デフォルト値：オン）。
  コンパイラによっては算術シフトの代わりに論理シフトを行うものがあり、そのような場合、このオプションがオンに設定されていると、符号エラーが発生する可能性があります。

- *Generate Round Operation on float to integer Assignment*：浮動小数点型変数を整数型変数に代入する時の丸め処理をオンまたはオフにします。丸め処理は、0.5 を加算（正の値の場合）または減算（負の値の場合）することによって行われます。このオプションがオフ（デフォルト）になっていると、小数部はそのまま切り取られます。

- *Temp Vars always 32 bit (integer)*：すべての整数型のテンポラリ変数を 32 ビットにします。このオプションがオフ（デフォルト）になっていると、ビット幅は実際の要件に合わせて調整されます。

- *Use power of 2 approximation of literals*：このオプションがオンになっていると、リテラルの乗算や除算のコードを生成する際、そのリテラルを表す分数に近似し、かつ分母が 2 のべき乗である分数が検索されます。該当する分数が存在する場合、その分数が近似式として使用され、シフト演算コードが生成されます。

  この最適化が行われるのは、元の分数との偏位が 1%未満の近似式が存在し、かつ近似誤差がオーバーフローを招かない場合のみです。

  例：このオプションがオフ（デフォルト）の場合、`0.866` というリテラルの乗算について以下のようなコードが生成されます。

  ```
  result = input * (sint32)433 / (sint16)500;
  ```

  オプションをオンにすることにより、上記のコードは以下のようになります。

  ```
  result = input * 28377 >> 15;
  ```

  上記の 2 つの式の精度はほとんど変わりませんが、後者では、処理時間が長くかかる除算の代わりに高速なシフト演算が行われます。

実験用コードオプション（“Experiment Code”ノード）

注記

マイクロコントローラターゲットにはこれらのオプションは使用されません。

実験用コード生成に使用される以下のようなオプションを設定します。

- *Max Number of Loop Iterations*：生成されるコードにおけるループの繰り返し回数を制限することができます。このオプションによって、無限ループを防ぐことができます。
- *Prefix for Component Names*：ここに入力した文字列が、コード生成時にコンポーネント名のプレフィックスとして挿入されます。
  INTECRIO において、1 つのコンポーネントについて複数のバージョンを使用する場合、各バージョンのコンポーネント名を同一にすることはできないので、このオプションが役立ちます。
- *Data Logging*：データロガーを過渡モード（値の変化時にのみサンプリングを行うモード）で使用する場合には、このオプションをオンにしておいてください。データロガーについては、582 ページの「データロガー」の項を参照してください。
- *Protected Vector Indices*：このオプションがオンになっていると、生成されるコード内に、範囲外のインデックス値による配列、マトリックス、特性カーブ／マップへのアクセスを監視する機能が挿入されます。これによって、たとえば要素が 5 個しかない配列について、6 番目の要素に対するアクセスが行われたために発生するような障害が阻止されます。この監視機能はパフォーマンスの低下を招くため、コードの効率性を最重要視する場合にはオフにしてください。

- *Use OID for Generation of Component Names*：コード生成時において、コンポーネント名として、コンポーネント OID を使用するか（オプション = オン）、またはモデル名を使用するか（オプション = オフ）を指定します。このオプションをオフにする場合は、ユーザーの責任において名前の衝突を防ぐようにしてください。
- *Add Comment with Specification Info for each Element*: このオプションがオンになっていると、各エレメントのモデル記述情報が、コメントとして挿入されます。

  例：

```
/*----Local variables object structure-------*/
struct MODULE_C_CODE_IMPL_Obj {
   ASDObjectHeader objectHeader;
   /* model: name=cont, type=cont,            ↵
             kind=variable, scope=local,      ↵
             memory=volatile */
   real64_Obj *cont;
};
```

- *Cache Loking, Cache Lock Code, Cache Lock Data, Used Cache Size*：これらのオプションは、ASCET-RP 用のものです。ASCET-RP のユーザーズガイドを参照してください。

製品用コードオプション（"Production Code" ノード）

これらのオプションは、ASCET-SE がインストールされていて、ビルドオプションでマイクロコントローラがターゲットとして選択されている場合にのみ使用されます（405 ページ参照）。詳しい情報は ASCET-SE ユーザーズガイドに記載されています。

最適化オプション（"Optimization" タブ）

このノードには以下のオプションが含まれます。

- *Constant Folding on Arithmetic Operations*：このオプションがオンになっていると、結果が決まっている算術演算が 1 つの定数に置き換えられます。たとえば、2 つの定数の加算（`C1 + C2`）は定数 `C3`（`C3:=C1+C2`）に置き換えられ、また 2 つの定数の比較（`C1 >= C2`）は `TRUE`（`C1>=C2` の場合）または `FALSE`（`C1<C2` の場合）に置き換えられます。
- *Constant Folding on Logical Operations*：このオプションがオンになっていると、結果が決まっている論理演算が 1 つの定数に置き換えられます。たとえば、`X_log AND TRUE` は `X_log` に置き換えられ、`FALSE OR X_log` は `X_log` に、また `FALSE != FALSE` は `FALSE` に置き換えられます。

- *Constant Folding on Control Flow* ：このオプションがオンになっていると、`If` 文の式の結果（`true` または `false`）が決まっている場合、コード生成時にその式が `TRUE` または `FALSE` に置き換えられます。
- *Operator Simplification with Respect to Intervals* ：このオプションがオンになっていると、インターバル計算によって式の結果が決まる場合、その式が単純化されます。たとえば、`X` のインターバル（実装範囲）が `[0,100]` で `Y` のインターバルが `[120,150]` である場合、`Y` は常に `X` より大きいため、式 `X < Y` は `TRUE` に置き換えられます。
- *Operator Simplification (general)* ：このオプションがオンになっていると、演算式が単純化されます。たとえば、`X - NEG(Y)` は `X` に置き換えられ、また `NOT(NOT(X_log))` は `X_log` に置き換えられます。
- *Common Subexpression Elimination* ：このオプションがオンになっていると、省略可能な部分が含まれる演算式が単純化されます。たとえば、`X / X` は `1` に置き換えられ、また `X < X` は `FALSE` に、`X_log AND X_log` は `X_log` に置き換えられます。
- *Optimize Direct Access Methods (One Level)*：このオプションがオンになっていると、生成されるコード内に直接アクセスメソッドが展開されます。オフになっていると、それぞれの直接アクセスメソッドの関数呼び出しが生成されます。コード生成時に問題が発生した場合には、このオプションをオフにしてください。

  注記

  メソッド引数として定義されたクラスの場合は、そのクラスエレメントへの ***optimized direct access***（最適化された直接アクセス）はサポートされていません。

- *Optimize Direct Access Methods (Multiple levels)*：このオプションがオンになっていると、コード生成時に、ネストされた直接アクセスメソッドがコード内に展開されます。直接アクセスメソッドのネスト化は、ESDL でコンポーネントを記述する場合にのみ可能です。このオプションの最適化効果は、上記の単一レベルのメソッド呼び出しの最適化の場合と同じです。

ステートマシンオプション（"Statemachine" ノード）

このノードには以下のオプションが含まれます。

- *Outline Generated Methods (may be changed locally)*：このオプションがオンになっていると、同じ階層ステートからの複数のトランジション内で同じコードが使用される場合、所定の条件が満たされていれば、この部分が独立したメソッドとしてコード生成（アウトライニング）されます。

  このようなメソッドとメソッド呼び出しが生成されないようにするには、このオプションをオフにしてくだい。その場合、アクションコードは、必要な箇所すべてに挿入されます。

- *Auto-inline private methods (smaller code-size - may be changed locally)*：サイズの小さなプライベート関数や呼び出し元が少ない関数をインライニングすることにより、コードサイズと実行時間が節約できます。このオプションがオンになっていると、コードジェネレータはこのような条件に当てはまる関数を検知し、コンパイラに対してインライニングを行うよう指示します。
- *Hierarchical Code-Generation (may be changed locally)*：ステートマシン用のコードは、フラット（ベースステートごとに 1 つの `switch` 文）または階層的（ステート階層に応じてネストされた `switch` 文）に生成されます。フラットコード（オプションがオンの場合）では実行時間が最適化され、階層コード（オプションがオフの場合）ではコードサイズが最適化されます。
- *Optimize Static Actions (Restricted Modeling)*：このオプションがオフ（デフォルト）になっていると、階層ステートの Static アクションは、階層ステートを離脱しない各トランジションごとに個別に生成されます。

  このオプションをオンにすると、階層ステートの Static アクションは、各サブステートごとに 1 つのみ生成されます。そのためコードサイズが縮小されます。

  注記

  このオプションによる最適化を行うと、アクションの実行とコンディションの評価の順番が変わるため、ステートマシンの挙動が変化する可能性があります。さらにこの最適化は、モデルの構造によっては適用できない場合があります。
  詳しくは ASCET リファレンスガイドの「ステートマシンの最適化」の項を参照してください。

- *Generate well-formed switch*：このオプションがオンになっていると、MISRA[1]（ルール 14.7、15.2、15.3）に準拠するステートマシンコードが生成されます。
- *Initialize history variable with zero*：このオプションがオンになっていると、プロジェクト内の階層ステートのヒストリ変数は、開始ステートの番号ではなく 0 に初期化されます。

  この場合、コードに代入文が追加され、MISRA ルール 15.3 に準拠しなくなる可能性があります。

4.8.9 固定小数点演算用のインプリメンテーションの定義

ECU に用いられるマイクロコントローラの多くは浮動小数点演算をサポートしないので、製品用のコード生成においては、多くの場合、ASCET の物理記述を固定小数点演算コードに変換する必要があります。この対応付けは、「インプリメンテーション」と呼ばれる実装情報により定義されます。

1. **M**otor **I**ndustry **S**oftware **R**eliability **A**ssociation, http://www.misra.org.uk

データセットと同様、コンポーネントやプロジェクトには任意の数のインプリメンテーションを定義することができます。また、インプリメンテーションは、データセットと同様に表示、編集、名前変更、削除、ブラウズ、エクスポート（フラットおよびリカーシブ）、追加、コピーが可能です。

さらにデータセットの場合と同様に、プロジェクトまたはコンポーネントの階層ツリー構造の末端の葉の部分となる基本エレメントには、専用のエディタがあります。`continuous` または `discrete` 型の基本エレメントのインプリメンテーションについては、物理値の「インターバル」（実装コードで使用する範囲）と変換式を定義する必要があります。

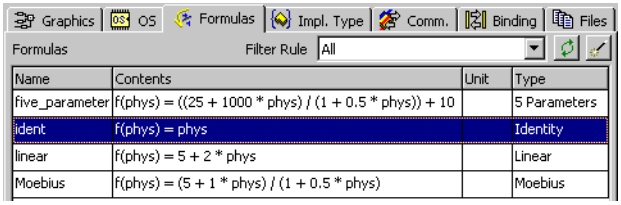
一般的には、たとえば組み込みソフトウェアで処理される車両の走行速度など、プロジェクト内の多くのエレメントに同じ変換式を定義することになりますが、そのような場合は、プロジェクト内で変換式を一度だけ定義し、それをすべてのエレメントで共有することができます。また、インポート／エクスポートのしくみを利用して、変換式をプロジェクト間で交換することもできます。

ASCET は、量子化浮動小数点コード生成の中間ステップのインプリメンテーションもサポートします。この種のコードでは固定小数点演算がエミュレートされますが、この固定小数点演算のリンクと量子化は、コード実行中に対話形式で変更することができます。この中間ステップを利用すれば、プロジェクトの各エレメントについて最適化されたコード実装を行うことができます。

変換式

変換式を追加する：

- プロジェクトエディタの“Formulas”タブをクリックします。



`ident` という変換式は必ず表示されています。新しく作成されたインプリメンテーションの場合、デフォルトとしてこの変換式が選択されています。

- 新しい変換式を作成するために、**Global Formulas → Add** を選択します。
- 式の名前を入力してから **<Enter>** を押します。

変換式を編集する：

- “Formulas”タブで、編集したい変換式を選択します。

- **Global Formulas** → **Edit** を選択します。

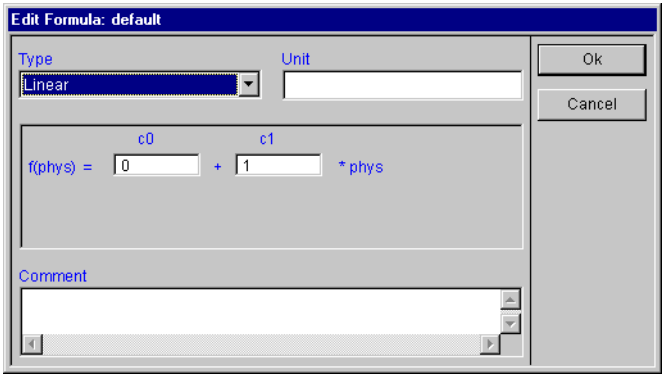
または

- ショートカットメニューから **Edit** を選択します。

または

- 変換式をダブルクリックします。

  グローバル変換式用のフォーミュラエディタが開きます。



- “Type”コンボボックスで、編集する変換式のタイプを選択します。
- 必要な値をフィールドに入力します。
- “Unit”フィールドに単位を入力します。

  単位はドキュメント生成時にのみ使用され、機能には影響しません。
- “Comment”フィールドにコメントを入力します。
- **OK** をクリックします。

変換式 `ident` は、名前を変更したり削除しないようにしてください。この変換式を編集して“Type”コンボボックスで別のタイプを選択すると、以下のようなワーニングメッセージが表示されます。

```
The <ident> formula is used for newly created elements to
provide an identity conversion and changing this formula
may corrupt your implementation. Do you really want to
change the <ident> formula?
```

`ident` のタイプの変更を行う必要がある場合は、**Yes** をクリックし、それ以外の場合は **No** をクリックしてください。

インプリメンテーションに使用できる変換式は、一次式のみです。4 種類の形式がサポートされていて、この形式に応じてフォーミュラエディタの表示形式も異なります。

- *Identity* 式は、恒等変換を表します。

  f(phys) = phys

- *Linear* 式は、係数 c0（オフセット）および c1（傾き）により規定される線形変換を表します。

  f(phys) = c0: 0 + c1: 1 * phys

- *Moebius* 式は、有理変換を係数 c0（分子のオフセット）、c1（分子の傾き）、d0（分母のオフセット）、および d1（分母の傾き）により規定される 2 つの線形変換の商として表します。Moebius 式は主に、一次式の定義が Moebius 式の形式で与えられた場合に Linear 式の形式に変換されるのを防ぐために使用します。

  $$f(phys) = \frac{c0: 0 + c1: 1 * phys}{d0: 1 + d1: 0 * phys}$$

- *Five Parameters* 式も、同じ種類の有理変換を係数 p1、p2、p3、p4 および p56 による別の形式で表すものです。この式も、一次式の定義が Five Parameters 式の形式で与えられた場合に Linear 式の形式に変換されるのを防ぐために使用します。

  $$f(phys) = \frac{p2: 0 + p1: 1 * phys}{p4: 1 + p3: 0 * phys} + p56: 0$$

変換式の名前を変更する：

- “Formulas” タブで変換式を選択します。
- **Global Formulas → Rename** を選択します。

または

- ショートカットメニューから **Rename** を選択します。

変換式を削除する：

- “Formulas” タブで 1 つまたは複数の変換式を選択します。

- **Global Formulas** → **Delete** を選択します。

または

- ショートカットメニューから **Delete** を選択します。
- “Confirm”ダイアログボックスで、**OK** で操作を確定します。

  変換式が削除されます。削除された変換式名が ASCET モニタウィンドウに表示されます。

変換式の表示をソートする：

- いずれかの列のタイトル部分をクリックします。

  変換式がソートされます。クリックした列の内容が、アルファベット順にソートされますが、“Type”列に限り、以下の順にソートされます。

  Identity → Linear → Moebius → 5 Parameters

  もう一度同じ列でソートすると、逆順ソートが行われます。

“Formulas”タブに表示される変換式を、以下のようにしてフィルタ表示することができます。

変換式をフィルタ表示する：

- “Filter Rule”コンボボックスでフィルタを選択します。

  以下のフィルタから選択できます。

フィルタ	表示される変換式
`All`	すべての変換式
`Used in ACTIVE implementations`	プロジェクト内で、エレメントのインプリメンテーションに使用されている変換式
`Not used in ACTIVE implementations`	プロジェクト内で、エレメントのインプリメンテーションに使用されていない変換式
`Used in ANY implementation`	プロジェクト内で、エレメントのインプリメンテーションに使用されているか、または現在使用されていないインプリメンテーション内で使用されている変換式
`Not used`	未使用の変換式

  たとえば `Not used` フィルタを使用して、容易に未使用のフィルタを削除することができます。

- **Update** をクリックして変換式表示を更新します。

  プロジェクトまたはそのコンポーネントのインプリメンテーションを別のエディタで編集した際に、この操作が必要です。

プロジェクト内のエレメントが、プロジェクト内で定義されていない変換式を使用している場合、コード生成時にエラーが発生します。このような「足りない」変換式は、以下のようにして追加することができます。

足りない変換式を追加する：

- プロジェクトエディタの“Formulas”タブを開きます。
- **Global Formulas → Add missing** を選択します。

または

- **Add missing formulas** をクリックします。

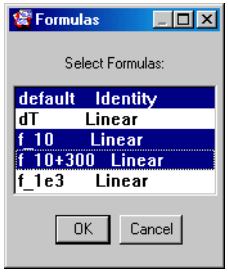
  現在のインプリメンテーション内にある未定義の変換式について、同じ名前の新しい変換式（恒等式）が追加されます。

変換式をインポートする：

- プロジェクトエディタの“Formulas”タブを開きます。
- **Global Formulas → File In → Selected**（または **All**）を選択します。

  Windows のファイル選択ダイアログボックスが開きます。
- 追加したい変換式が入っているファイルを選択します。
- **Open** をクリックします。

All コマンドを使用した場合は、選択したファイルに入っているすべての変換式がインポートされ、**Selected** コマンドを使用した場合は、インポートする変換式をダイアログボックスで選択することができます。



変換式をファイルに書き込む：

- **Global Formulas → File Out → Selected**（または **All**）を選択します。

  Windows のファイルダイアログボックスが開きます。
- ディレクトリとファイル名を選択します。
- **Save** をクリックします。

All コマンドを使用した場合、現在のプロジェクト内のすべての変換式がファイルに書き込まれ、**Selected** コマンドを使用した場合には、選択した変換式だけがエクスポートされます。変換式は ASCII フォーマットでファイルに書き込まれるので、任意のテキストエディタで開くことができます。

配列とマトリックスには、スカラエレメントの場合と同じように 1 つのインプリメンテーションが定義されます。特性カーブの場合は、入力値と出力値にそれぞれ 1 つずつ、合計 2 つのインプリメンテーションが定義され、また特性マップの場合は、入力値に 2 つと出力値に 1 つ、合計 3 つのインプリメンテーションが定義されます。

インプリメンテーションの一括変更

インプリメンテーションに含まれる変換式を編集したり、別の変換式に置き換えた場合、その変更内容をすべてのインプリメンテーションに反映させることができます。また、プロジェクト内のすべてのインプリメンテーション内の変換式を置き換えてインプリメンテーションを一括変更することもできます。

1 つのインプリメンテーションに加えられた変更は、自動的には他のインプリメンテーションに反映されません。ある変換式の置き換えや編集を行ったときには、必ずすべてのインプリメンテーションを更新する必要があります。

1 つのインプリメンテーション内の特定の変換式をすべて置き換える：

- プロジェクトエディタの“Formulas”タブを開いて、プロジェクト内に存在する変換式を一覧表示します。
- **Extras → Global Replace Formula** を選択します。

  ダイアログボックスが開き、元の変換式と新しい変換式の名前の入力が要求されます。
- 名前を適宜に入力してから、**OK** をクリックします。

  インプリメンテーション内で、変換式が置き換わります。

すべてのインプリメンテーションを更新する：

- プロジェクトエディタの **Extras → Update Implementations** を選択すると、変換式の変更がすべてのインプリメンテーションに反映されます。

  変更履歴が ASCET モニタウィンドウに書き込まれます。インプリメンテーションの変更は、各基本エレメントのインプリメンテーションエディタのマスタページのデータを基準として行われます。

インプリメンテーション型

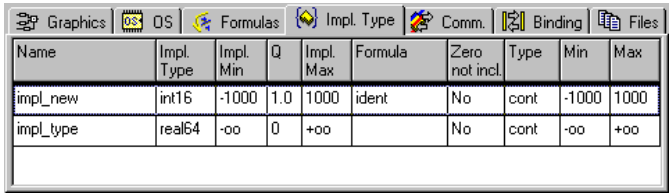
個々の変数のインプリメンテーションの編集を容易にするため、また同じインプリメンテーションを 1 度に複数のエレメントに割り当てることができるように、プロジェクト内に「インプリメンテーション型」と呼ばれるものを定義すること

ができます。これは、クラスやモジュールのデフォルトプロジェクト（4.8.1 項を参照してください）の場合も同じです。インプリメンテーション型にはインプリメンテーションの基本的な部分が含まれ、インプリメンテーションエディタを使用して個々のエレメントに割り当てることができます。

この項では、インプリメンテーション型の作成および設定方法について説明し、インプリメンテーション型を実際に使用する方法については、482 ページ「インプリメンテーション型の使用方法」の項を参照してください。

インプリメンテーション型を追加する：

- プロジェクトエディタの“Impl. Type”タブを開きます。



Name	Impl. Type	Impl. Min	Q	Impl. Max	Formula	Zero not incl.	Type	Min	Max
impl_new	int16	-1000	1.0	1000	ident	No	cont	-1000	1000
impl_type	real64	-oo	0	+oo		No	cont	-oo	+oo

- **Global Implementations → Add → *<model type>*** を選択します。

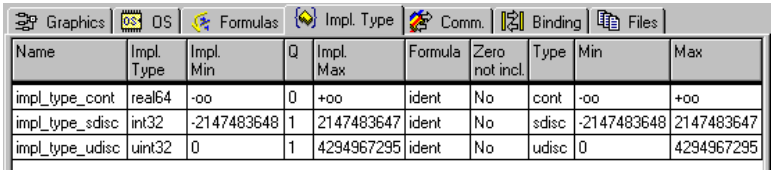
または

- ショートカットメニューから **Add → *<model type>*** を選択します。i

注記

インプリメンテーション型の作成時に選択されたモデルデータ型によって、そのインプリメンテーション型を使用できる変数が決まります（482 ページ「インプリメンテーション型の使用方法」の項を参照してください）。

新しいインプリメンテーション型がデフォルト名で作成され、***<model type>*** で選択された型に対応する情報（下の図を参照してください）が自動的に設定されます。



Name	Impl. Type	Impl. Min	Q	Impl. Max	Formula	Zero not incl.	Type	Min	Max
impl_type_cont	real64	-oo	0	+oo	ident	No	cont	-oo	+oo
impl_type_sdisc	int32	-2147483648	1	2147483647	ident	No	sdisc	-2147483648	2147483647
impl_type_udisc	uint32	0	1	4294967295	ident	No	udisc	0	4294967295

インプリメンテーション型の名前の変更または削除を行う：

複数のインプリメンテーション型を同時に削除することができますが、名前の変更は 1 つずつ行う必要があります。

- インプリメンテーション型を選択します。
- **Global Implementation → Rename** を選択します。

または

- ショートカットメニューから **Rename** を選択します。

  選択されたインプリメンテーション型の“Name”列のセルが入力モードになります。
- 新しい名前を入力して **<Enter>** を押します。
- **Global Implementation → Delete** を選択します。

または

- ショートカットメニューから **Delete** を選択します。

  選択されたインプリメンテーション型が削除されます。

インプリメンテーション型を編集する：

- “Impl. Type”タブから、編集したいインプリメンテーション型を選択します。
- **Global Implementation → Edit** を選択します。

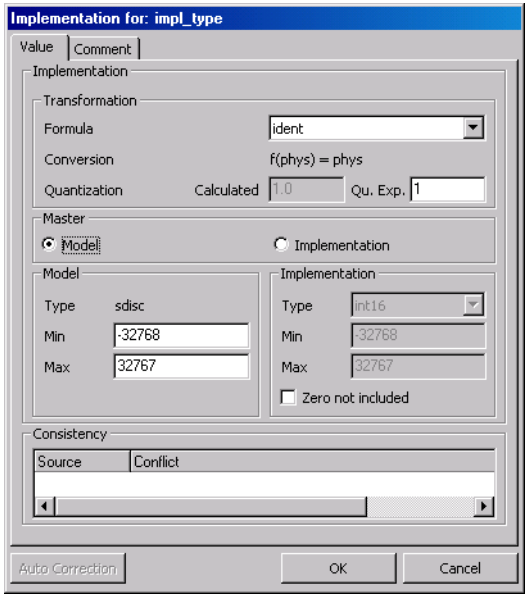
または

- ショートカットメニューから **Edit** を選択します。

または

- 選択されたインプリメンテーション型をダブルクリックします。

  インプリメンテーションエディタの“Value”タブが開きます。



- “Formula”フィールドで変換式を選択します。
- 必要に応じて、“Qu. Exp.”フィールドで量子化を指定します。

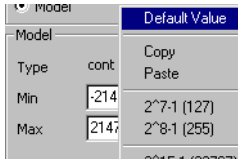
  このフィールドは、ビルドオプションでコードジェネレータとして `Quantized Physical Experiment`（405 ページ参照）が選択されている場合にのみ設定できます。
- “Master”フィールドで、インプリメンテーションエディタのマスタページを指定します。

1. “Model”をマスタとした場合の設定：

   物理モデルのプロパティを基準にインプリメンテーションを設定する場合、モデル側の設定をマスタにします。

   この設定になっていると、“Model”側のフィールドに含まれる値が編集可能となりますが、その中で、モデルデータ型（“Type”）のみはインプリメンテーション型が作成された時に自動的に決定されるため、編集できません。
   “Implementation”側のフィールドは、**Zero not included** 以外は編集できません。

- “Min.”および“Max.”フィールドにインターバル（インターバルの最小／最大値）を入力します。



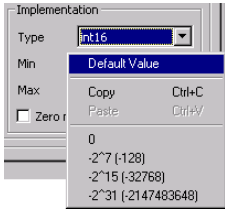
- これらのフィールドにモデルデータ型の上下限値（たとえば“cont”型の場合は“∞”）を指定するには、入力フィールド内のショートカットメニューから **Dafault Value** を選択します。

  これにより、実装側（“Implementation”側）の値は自動的に変更されます。

2. “Implementation”をマスタとした場合の設定：

   実装コードのプロパティを基準にインプリメンテーションを設定する場合、実装側の設定をマスタにします。
   この設定になっていると、“Implementation”側のフィールドに含まれる値が編集可能となり、“Model”側は編集できなくなります。

- “Type”コンボボックスで実装データ型を選択します。



- “Min.”および“Max.”フィールドに値のインターバル（インターバルの最小／最大値）を入力します。
- 選択されている実装データ型のデフォルトの上下限値を指定するには、入力フィールド内のショートカットメニューから **Dafault Value** を選択します。

  これにより、モデル側の値は自動的に変更されます。

3. その他の設定：

- インターバルに 0 が含まれない場合は、**Zero not included** オプションをオンにします。

注記

Zero not included オプションがオンになっていると、実行時に、コード内の除算の分母が 0 であるかどうかのチェックが行われず、これにより深刻な障害が発生する可能性があるため、通常このオプションはオフにしておいてください。

- コメントを入力する場合は、“Comment”タブを選択してください。
- “Comment”タブのテキストボックスにコメントを入力します。

  インプリメンテーション型をエレメントに割り当てる際、このコメントは除外されます。

- **OK** をクリックして設定を確定します。

インプリメンテーション型に設定できる内容は上記の項目のみです。その他の情報（代入時のリミッタ機能の設定やメモリ領域についての設定）は、インプリメンテーション型をエレメントに割り当てた後、個々のエレメントごとに設定します。

インプリメンテーション型を作成したら、その設定を別のインプリメンテーション型にコピーすることができます。

インプリメンテーション型の設定をコピーする：

- 設定のコピー元とするインプリメンテーション型を選択します。
- **Global Implementation → Copy** を選択します。

または

- ショートカットメニューから **Copy** を選択します。

  設定内容がデータベースのクリップボードにコピーされます。
- 設定のコピー先とするインプリメンテーション型を選択します。
- **Global Implementation → Paste** を選択します。

または

- ショートカットメニューから **Paste** を選択します。

  データベースのクリップボードから設定内容がコピーされます。

インプリメンテーション型とデータベース内のスカラエレメントの設定を入れ替えることもできます。

インプリメンテーション型とスカラエレメントの設定を入れ替える：

- プロジェクトエディタの“Impl. Type”タブで、設定のコピー元とするインプリメンテーション型を選択します。
- **Global Implementation → Copy** を選択します。

または

- ショートカットメニューから **Copy** を選択します。

  設定内容がデータベースのクリップボードにコピーされます。

  続いて、以下の 3 通りの方法で設定をエレメントにコピーします。

1. コンポーネントマネージャの“Implementation”タブから行う場合：
   - コンポーネントマネージャの“1 Database”リストから、設定のコピー先とするスカラエレメントを含むコンポーネントを選択します。
   - “3 Contents”ペインの“Implementation”タブを開きます。
   - 設定のコピー先とするエレメントを選択します。
   - ショートカットメニューから **Paste** を選択します。

   または

   - **<Ctrl> + <V>** を押します。

     データベースのクリップボードから設定内容がコピーされます。
2. コンポーネントのインプリメンテーションエディタから行う場合：
   - 設定のコピー先とするスカラエレメントを含むコンポーネントのインプリメンテーションエディタを開きます。
   - コピー先のエレメントが含まれるタブを開き、そのエレメントを選択します。
   - Element → Paste Implementation From Buffer を選択します。

   または

   - ショートカットメニューから **Paste Implementation From Buffer** を選択します。

     データベースのクリップボードからスカラエレメントに設定内容がコピーされます。

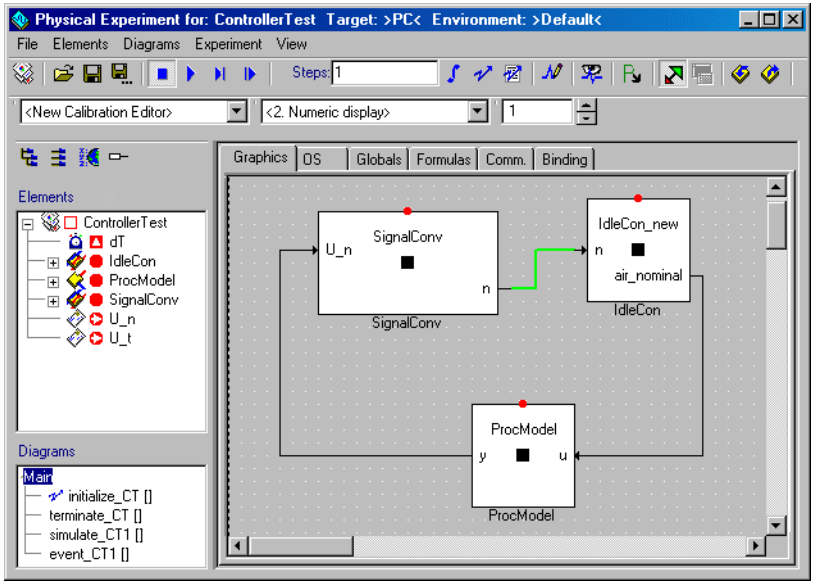
スカラエレメントのインプリメンテーションに設定された情報をインプリメンテーション型にコピーするには、上記の手順の逆の手順で行ってください。つまりコンポーネントマネージャまたはインプリメンテーションエディタからインプリメンテーションの設定をコピーし、それをプロジェクトエディタの“Impl. Type”タブに表示されたインプリメンテーション型に貼り付けます。

4.8.10 プロジェクトの実験

プロジェクトのコード生成と実験に関する機能は、コンポーネントの場合と比べてはるかに強力です。コンポーネントについては浮動小数点演算機構（ビルドオプション：`Physical Experiment`、405 ページ参照）によるオフライン実験しか行うことができませんが、プロジェクトについては浮動小数点演算、量子化浮動小数点演算（“Code Generator”オプション：`Quantized Physical Experiment`）、または固定小数点演算（同オプション：`Implementation Experiment`）の演算機構を用いて、オンライン実験を行ったり、オフライン実験とオンライン実験を組み合わせて行うことができます。またコンポーネントの実験をプロジェクト内で行えば、プロジェクトのコード生成機能をコンポーネント用にも利用することが可能です。

オンライン実験とオフライン実験

プロジェクトも、コンポーネントのようにオフラインで実験することができ、コンポーネントのオフライン実験で使用できる機能はすべて利用可能です。これは、コンポーネントのオフライン実験も、実際にはそのコンポーネントが属するデフォルトプロジェクトの単位で実験が行われるためです。オフライン実験についての詳しい説明は、547 ページの「実験」という項を参照してください。



コンポーネントとプロジェクトのオフライン実験の大きな違いは、プロジェクトの場合、イベントジェネレータにより「スティミュレート」される対象、つまり一連のデータが刺激信号として与えられる対象が、コンポーネント内のメソッドやプロセスではなく、オペレーティングシステムエディタで作成されたタスクである、という点です。

オンライン実験では、プロジェクトを一層現実的な条件下でリアルタイムにテストすることができます。オフライン実験では、ASCET が生成したコードが PC または実験ターゲット上で実行されますが、リアルタイムには実行されません。オンライン実験においては、コードは実験ターゲット上で常にリアルタイムに実行されるため、その環境は、ASCET モデルを実際の環境で実行させる際の基礎になります。オフライン実験はシステムの機能仕様の検証が主な目的ですが、オンライン実験においては、オペレーティングシステムのスケジューリングとそれに対応する制御システムのリアルタイムな挙動が検証されます。

オンライン実験は、オフライン実験と比べて主に以下の点が異なります。

- オンライン実験は、ES1130 、ES1135 などのリアルタイムハードウェア上でしか実行できません。
- オンライン実験ではイベントジェネレータやデータジェネレータは使用されません。スケジューリングはオペレーティングシステムによって決定され、データは物理プロセスモデルから読み取られます。
- 各測定ウィンドウ用に起動するタスクを選択して、データ測定のサンプリングレートを明示的に指定する必要があります。
- 実験と測定をそれぞれ独立して開始することができます。

オンライン実験を行うには、ASCET-RP がインストールされている必要があります。詳しくは『ASCET-RP ユーザーズガイド』をお読みください。

実験を行う前に、実験環境を起動せずにプロジェクトのコード生成だけを行うこともできます。これは、生成されたコードの構文が正しいかどうかを調べるのに有効です。コードが正しくない場合、コンパイラが発行したエラーが表示されます。

コードを生成する：

- プロジェクトエディタで **Component → Generate Code** を選択します。

または

- **Generate Code** ボタンをクリックします。

  プロジェクトのコードが生成され、エラーがあった場合にはメッセージウィンドウにエラーメッセージが出力されます。

または

- **Component → View Generated Code** を選択します。

  C コードが生成され、自動的にその内容がテキストエディタで開きます。
  使用されるエディタは、ASCET オプションウィンドウの“ASCII Editor node”ノードで選択できます（60 ページ参照）。

生成されたコードを保存する：

- プロジェクトのコードをファイルに書き込むには、**Component → File Out Generated Code → Flat**（または **Recursive**）を選択します。

  Windows の“Path selection”ダイアログボックスが開きます。

- パスを選択します。
- **OK** をクリックすると、選択したディレクトリにコードが書き込まれます。

  生成されたファイルの名前は、モニタウィンドウに出力されます。保存されたファイルは、任意のテキストエディタや C コードエディタで開くことができます。

実行コードを生成する：

- プロジェクトエディタで **Component → Build Executable Code [store to database]** を選択します。

または

- **Build Executable Code** ボタンをクリックします。

  プロジェクト全体のコードが生成され、リンクされます。ここでエラーが発生しなければ、実行形式のファイルが生成されます。
  このプロセスで生成されたソースコードとオブジェクトコードは、ASCET データベースに保存されます。

または

- **Component → Build from directory** で実行ファイルを作成すると、生成されたコードは ASCET データベースには保存されません。

  これにより、時間のかかるファイル保存処理を省略することができます。

  この方法は、充分な空きメモリがない PC において大きなプロジェクトのコード生成を行うような場合に、特に有効です。

実行ファイルが生成されると、ソースコードを含むすべてのファイルが、`.¥cgen¥` ディレクトリ（デフォルト）に書き込まれます。

注記

“Options”ウィンドウの“Code Generation”タブの **Keep files in Code Generation Directory** オプション（49 ページ参照）がオフに設定されていると、ASCET コード生成ディレクトリ `.¥cgen¥` の内容は、ASCET のセッションを終了するたびに削除されます。この場合、ファイルを保管しておくには、431 ページの方法でデータベースに保存しておくようにしてください。
このオプションを変更しても、現在実行中のセッションには影響しません。

必要に応じて、既存のオブジェクトファイルを、コード生成やコンパイルを行わずに実行ファイルにリンクすることができます。この機能は、プロジェクトで外部の C コードファイルを使用するような場合に有効です。

オブジェクトファイルを実行ファイルにリンクする：

- 外部の C ソースファイルを変更した後、オブジェクトファイルを作成します。
- プロジェクトエディタで **Component → Link Only** を選択します。

  データベース内に保存されているプロジェクトの内部オブジェクトファイルが、外部オブジェクトファイルと共にハードディスクに書き込まれ、実行ファイルにリンクされます。

注記

Link Only 機能は、**Component → Build** コマンドまたは **Build Executable File** ボタンによってすでに内部ファイルが作成されている場合にのみ有効です。
ファイルがデータベース内に保存されるのは、この操作が行われたときだけです。

量子化浮動小数点コードの実験

量子化浮動小数点演算のコードは、浮動小数点演算のコードと同じ物理モデルから生成されます。コード生成の設定を切り替えるだけで、量子化浮動小数点演算コードを生成することができます。

量子化浮動小数点演算に切り替える：

- **Project Properties** ボタンをクリックします。
- “Code Generator”オプションに `Quantized Physical Experiment` を選択します。
- **OK** をクリックします。

  次に実験環境が起動される際には、この設定に基づいてコードが生成されます。

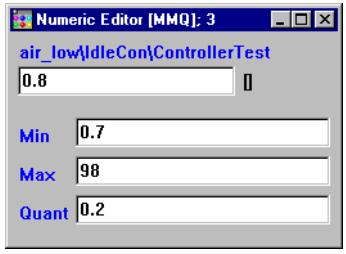
量子化浮動小数点演算を行うモデルの実験は、標準的な浮動小数点演算を行うモデルの実験と同様ですが、インターバルの上下限値と値の量子化を調整するための特別な適合ウィンドウがある点が異なります。

量子化適合ウィンドウを開く：

- 実験環境の“Elements”ペインで、適合したいエレメントを選択します。
- **Elements → Calibrate** を選択します。

  使用するエディタ（適合ウィンドウ）のタイプを指定するダイアログボックスが開きます。
- リストから `[Numeric Editor [MMQ]]` を選択し、**OK** をクリックします。

  量子化適合ウィンドウが開き、そこに現在のデータセットの値と現在のインプリメンテーションの情報が表示されます。



量子化浮動小数点演算は、変数への代入にのみ影響します。その変数に対して書き込みが行われたり、適合ウィンドウで変数の値が変更されるたびに、その値が量子化情報に従って調整されます。

量子化適合ウィンドウで適合を行う：

- 値のフィールドを選択します。
- 値を適宜に変更します。
- **<Enter>** を押します。

  ウィンドウに表示される値は、所定の量子化とインプリメンテーションに合わせて自動調整されます。
- “Min”、“Max”または“Quant”のいずれかのフィールドを選択し、値を適宜に変更することができます。
- **<Enter>** を押すか、別のフィールドを選択します。

  新しい設定に従って値が調整されます。

量子化浮動小数点演算はコンポーネントの実験にも利用することができますが、それは、C コードではなくブロックダイアグラムか ESDL で定義されたコンポーネントの場合のみです。また実験に用いるコンポーネントは、コンポーネントマネージャからではなくプロジェクトエディタから開く必要があります。

量子化浮動小数点演算を用いて、コンポーネントの実験を行う：

- プロジェクトエディタで **Specify Code Generation Settings** ボタンをクリックします。
- “Code Generator”オプションに `Quantized Physical Experiment` を選択します。
- “Elements”ペインからコンポーネントを選択します。
- **Element → Edit Component** を選択します。
- 所定のコンポーネントエディタが開きます。
- コンポーネントの実験を行います。

4.8.11 適合用データの生成

プロジェクトエディタでは、適合システムに接続されているマイクロコントローラターゲット上でコードを実行して INCA 等の適合システムで測定／適合を行うために必要なすべてのファイルを生成することができます。ASCET は適合システムに必要な情報を生成する際、ASAM-MCD-2MC という標準フォーマットを使用します。ASAM-MCD-2MC ファイルは、ASAM-MCD-2MC フォーマットをサポートする適合システムへのインターフェースとなります。

適合用データを生成する：

- プロジェクトエディタで **ASAM-2MC → Write** を選択して、システム用の ASAM-MCD-2MC ファイルを作成します。

  ファイル選択ダイアログボックスが開きます。
- ASAM-MCD-2MC ファイルのパスを選択し、ファイル名を入力します。
- **Save** をクリックして、適合ファイルをファイルシステムに書き込みます。

  ASCET は、適合用 ASAM-MCD-2MC データファイルと、プログラムコードが格納された HEX ファイルを生成します。

ASAM-MCD-2MC ファイルの生成オプションは、“Project Properties”ウィンドウの“ASAM-2MC”ノードで設定します。詳しくは、400 ページの「プロジェクトの設定」の項を参照してください。

4.9 コンテナ

ASCET V5.0 で導入された「コンテナ」は、プロジェクトやクラス、モジュールを格納するための「容器」のようなもので、異なるデータベースのアイテムを共通のバージョン管理下に置くために使用されます。旧バージョンの ASCET プロジェクトの場合、V5.2 用に変換される際に、既存の「ネットワーク」がこの「コンテナ」に変換されます 4.9.2 項を参照してください)。ただし「コンテナ」には「ネットワーク」ほど多くの機能はありません。

コンテナには、フォルダ以外のすべてのデータベースアイテムを格納できます。コンテナ内にコンテナを格納することもでき、この際、自分自身を格納したり、2 つのコンテナが互いに相手のコンテナを格納することも可能です。1 つのコンテナ内に同じアイテムを二重に格納することはできません。

4.9.1 コンテナの扱い

コンテナは、列挙型と同様、専用のエディタではなくコンポーネントマネージャ上で編集されます。“Container complnents”タブのコンテナビューには、他のビューモードと同様のショートカットメニューが用意されています。

コンテナビューを表示する：

- コンポーネントマネージャの“1 Database”リストから、既存のコンテナを選択します。

または

- 新しいコンテナを作成します（77 ページ参照）

  “3 Contents”ペインに“Container components”タブが表示されます。
- “Container components”タブをクリックしてコンテナビューを前面に表示します。

コンテナにデータベースアイテムを格納する（メニューコマンドまたはキーボードを使用）：

- ショートカットメニューから **Add** を選択します。

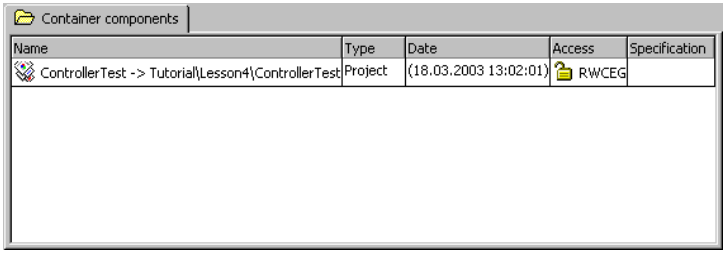
または

- **<Ins>** キーを押します。

  “Select Item”ダイアログボックスが開きます。
- “1 Database”リストから、必要なアイテムを選択します。

- **OK** をクリックして“Selected Items”ダイアログボックスを閉じます。

  アイテムがコンテナ内に追加され、“Container components”タブ上に表示されます。

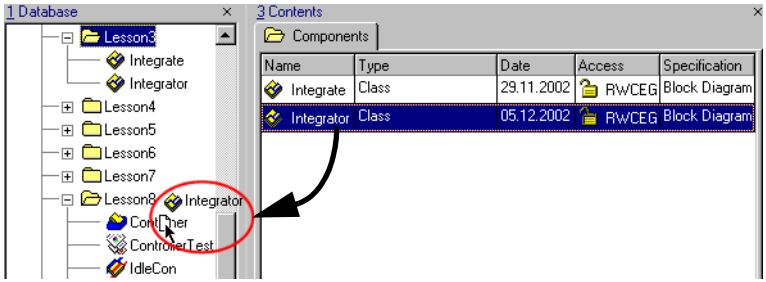


コンテナにデータベースアイテムを格納する（ドラッグ&ドロップで行う）：

- コンポーネントマネージャの“1 Database”リストで、アイテムを格納したいコンテナを含むフォルダを展開表示します。
- “1 Database”リストで、コンテナに格納したいアイテムが含まれるフォルダを選択します。

  フォルダの内容が“3 Contents”ペインに表示されます。

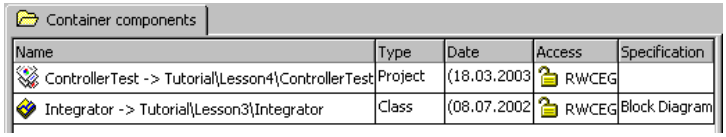
- “3 Contents”ペイン上のアイテムを、“1 Database”リスト内のコンテナにドラッグします。



注記

“1 Database”リストからアイテムをコンテナにドラッグすることはできません。

アイテムがコンテナ内に追加されます。



同じ方法で、あるコンテナ内のアイテムを別のコンテナにドラッグすることができます。この際、このアイテムは、元のコンテナから削除されません。

アイテムの名前を変更する：

注記

コンテナ内のアイテムの名前を変更すると、その元のデータベースアイテムの名前も変更されます。ただし、コンポーネントやプロジェクトにすでに使用されているアイテムの場合は、変更されません。

- “Container components”タブから、名前を変更したいアイテムを選択します。
- ショートカットメニューから **Rename** を選択します。

または

- **<F2>** キーを押します。

  アイテムの名前のフィールドが入力モードになります。
- 名前を入力して **<Enter>** を押します。

  データベース内およびコンテナ内のアイテムの名前が変更されます。

コンテナ内のアイテムを編集する：

アイテムをコンテナ内から直接開いて編集することができます。

- 編集したいアイテムを選択します。
- **Component → Edit Item** を選択します。

または

- ショートカットメニューから **Edit** を選択します。

注記

上記の 2 つの方法は、コンテナ内に含まれるコンテナについては利用できません。

または

- **<Enter>** キーを押します。

または

- アイテムをダブルクリックします。

  選択されたアイテムがコンテナであった場合、そのコンテナが“1 Database”内で選択され、その内容が“3 Container components”タブに表示されます。

コンテナ内のアイテムを削除する：

- “3 Container components”タブで削除したいアイテムを選択します。
- すべてのアイテムを選択するには、ショートカットメニューから **Select All** を選択します。

または

- **<Ctrl> + <A>** キーを押します。
- 選択されたアイテムを削除するには、ショートカットメニューから **Delete** を選択します。

または

- **<Del>** キーを押します。

  コンテナ内のアイテムが削除されます。しかしデーベースツリー内の実際のアイテムは削除されません。

コンテナビューをソートする：

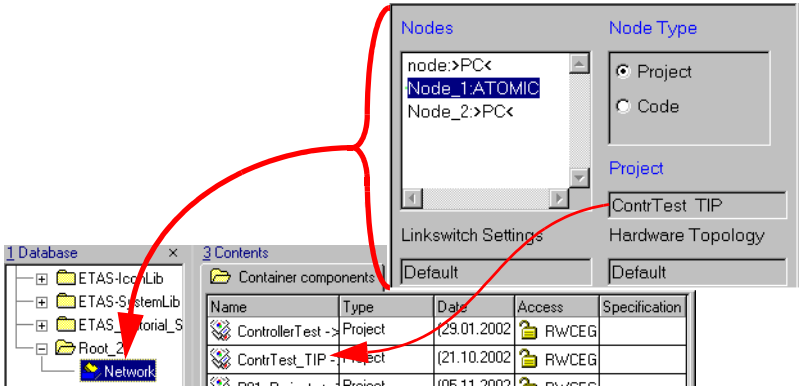
- ショートカットメニューから **Sort by → *<column>*** を選択します。

または

- 各カラムのヘッダをクリックすると、その列を基準にソートが行われます。

4.9.2 コンテナとネットワーク

「コンテナ」は、旧バージョンの ASCET で使用されていた「ネットワーク」に代わって導入されたものです。ネットワークが含まれる旧バージョンの ASCET データベースを ASCET V5.2 で開くと、既存のネットワークは自動的にコンテナに変換されます。異なるネットワークノードに割り当てられていたプロジェクトがすべてコンテナに格納され、その他の情報は格納されません。



4.10 エレメントプロパティの編集

エレメントの種類には、インターフェースエレメント、変数、特性カーブ／マップ、配列、マトリックス、および、複合エレメント（他のコンポーネントに参照されるコンポーネント）があります。

各エレメントにはさまざまなプロパティ（型、スコープ、アクセス、メモリ配置など）があり、その数や内容はエレメントの種類によって異なります。これらをモデルの要件に合わせて設定することができます。

1 つのエレメントに対して行われた変更は、コンポーネント内にあるそのエレメントのオカレンスすべてに適用されます。

インスタンスとオカレンス

1 つのエレメントは複数の「オカレンス」を持つことができます。エレメントの「オカレンス」とは、テキストベースのプログラミング言語においてそのエレメントの名前をプログラム内で記述することと同じです。あるエレメントの 1 つのオカレンスに変更を加えると、同じエレメントの他のすべてのオカレンスに影響が及びます。

被参照コンポーネントをエレメントとして使用する場合には、注意が必要です。すべてのコンポーネントは複数の「インスタンス」（実体）を持つ可能性があり、さらにその各インスタンスが複数のオカレンスを持つ可能性があります。

4.10.1 エレメントの設定

エレメントのプロパティはエレメントエディタで編集します。エレメントエディタは、さまざまな機能記述エディタ（プロジェクトエディタおよび各種コンポーネントエディタ）またはコンポーネントマネージャから起動することができます。

エレメントエディタを開く：

- 各機能記述エディタの“Elements”ペインで、編集したいエレメントを強調表示し、以下のいずれかの操作を行います。
  - **Element → Edit** を選択します。

  または
  - エレメントのショートカットメニューから **Edit** を選択します。

  または
  - エレメントをダブルクリックします。

または

- ブロックダイアグラムエディタの場合、描画領域内のエレメントのショートカットメニューから **Edit** を選択します。

または

- コンポーネントマネージャ、または各機能記述エディタのブラウザビュー（“Browse”タブ）で“Elements”タブを開き、編集したいエレメントを選択して以下のいずれかの操作を行います。
  - **<Enter>** を押します。

  または
  - “Elements”タブのショートカットメニューから **Edit** を選択します。

  “Element editor”ダイアログボックスが開きます。

ASCET オプションウィンドウで **Edit Primitive Elements**（または **Edit Implementation Cast**）オプション（46 ページの「確認ダイアログボックスに関するオプション」を参照してください）がオンになっていると、新しいエレメントを作成する際、自動的にエレメントエディタが開きます。

また、上記のオプションの設定よりも、各エレメントエディタ上の **Always show editor for new elements** オプションの設定が優先されます。

選択されたエレメントの型に応じて、エレメントエディタで設定する項目の数は異なりますが、同じ項目については、どの型のエレメントに対しても同じ機能を持ちます。

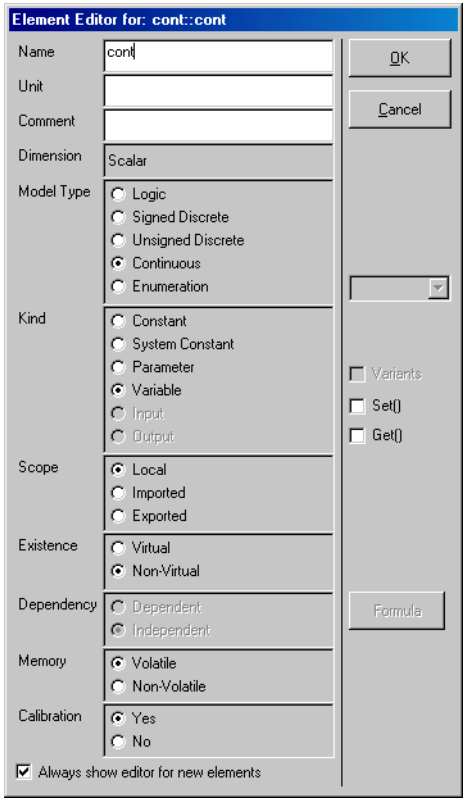


図 4-2　基本エレメントのエレメントエディタ（エレメントの型に応じて、一部のフィールドが無効になります。）

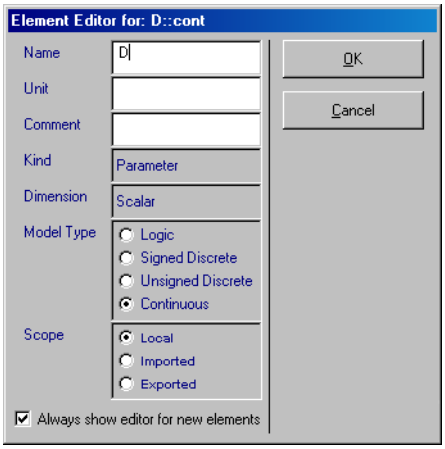


図 4-3　CT ブロックのエレメントエディタ

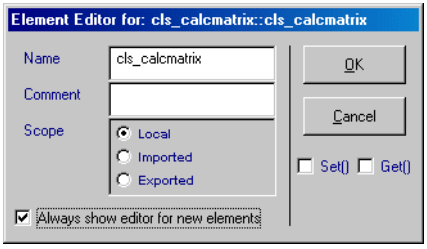


図 4-4　内包されるコンポーネント（被参照コンポーネント）のエレメントエディタ

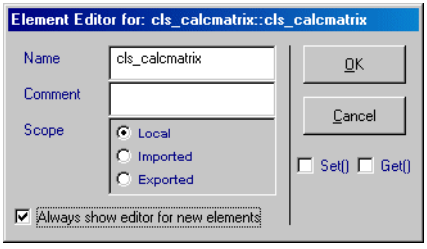


図 4-5　インプリメンテーションキャストのエレメントエディタ

エレメントのプロパティの変更は、以下のように行います。

エレメントのプロパティを編集する：

- エレメントエディタの“Unit”フィールドに単位を入力します。

  各エレメントに単位を割り当てることができますが、これはドキュメント生成時にのみ使用されるので、エレメントの機能には影響しません。
- “Comment”フィールドにコメントを入力します。

  生成されたコンポーネントのドキュメントに、このコメントが出力されます。
- “Model Type”フィールドにエレメントの型を指定します。

  Enumeration（列挙型）を選択すると、このフィールドの隣のコンボボックスがアクティブになり（図 4-2 を参照してください）、そこにデータベース内に存在するすべての列挙型が表示されます。
- “Kind”フィールドにエレメントの種類を指定します。

  ここでは、定数、システム定数（『ASCET リファレンスガイド』の「エレメントの種類」の項を参照してください）のいずれかを設定できます。

  Input および **Output** は、ステートマシンの入力と出力にのみ選択可能です。
- “Scope”フィールドで、エレメントのスコープ（『ASCET リファレンスガイド』の「エレメントのスコープ」の項を参照してください）を指定します。
- “Existence”フィールドで、エレメントが仮想エレメントであるかどうかを指定します。

  このフィールドは、変数およびパラメータ（列挙型を除く）の場合にのみ設定可能です。
- “Dependency”フィールドで、パラメータが、他のエレメントに依存しているかどうかを指定します。

  このフィールドは、パラメータ（列挙型を除く）にのみ設定可能です。

- “Memory”フィールドでは、エレメントが配置されるECUメモリが、揮発性の領域であるかどうか（**Volatile** または **Non-Volatile**）を指定します。

  ECUの不揮発性メモリに格納されたデータは、初期化時に上書きされません。

注記

ステートマシンのステート変数 `sm` は、エレメントエディタで編集できません。この変数に `Non-Volatile` 属性を割り当てるには、“`Elements`”ペインのショートカットメニューから **Settings → Non-Volatile** を選択します。

- “Calibration”フィールドでは、そのパラメータを適合システムで適合できるようにするかどうかを指定します。

  No を選択すると、ASAM-MCD-2MC ファイルの生成時に、このパラメータに対して READ ONLY の属性が付加されます。

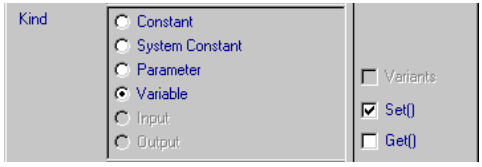
  このフィールドの設定内容は、その他のエレメントやオフライン実験には適用されません。

- **OK** をクリックして設定内容を確定します。

各エレメントについて、コンポーネント外部からのアクセスを有効にするかどうかを指定することができます。外部からのアクセスが有効になっていると、パラメータと定数の場合は外部からの読み出しのみが可能となり、変数の場合は読み出し／書き込みの両方が可能となります。

エレメントへの外部からのアクセスを有効／無効にする：

- エレメントエディタを開きます。



- **Set()** オプションをオンにします。

  エレメントへのコンポーネント外部からのアクセスが可能になり、コンポーネント外部から書き込みできるようになります。

- 外部からの書き込みを不可にするには、**Set()** オプションをオフにします。

- 出力を追加するには **Get()** オプションをオンにします。
- **OK** をクリックします。

  レイアウト上に、そのエレメント用の入出力が追加されます。

注記

モジュールレイアウトに入出力を表示するには、メッセージの **Get()** や **Set()** を有効にする必要がありますが、これは視覚化のためだけのもので、実際の代入はメッセージの名前を用いて行われます。

コード最適化オプションの **Optimize Direct Access Methods (...)**（413 ページ参照）がオフになっている状態で Get/Set ポートを使用すると、コード生成時において、エレメントへのダイレクトアクセス用に個別のメソッドが生成され、それらが関数として呼び出されます。
Optimize Direct Access Methods (one level) オプションがオンになっていると、個別のメソッドの代わりにダイレクトアクセスが使用され、**Optimize Direct Access Methods (multiple levels)** オプションがオンになっていると、それがネストされたクラスについても適用されます。

外部からのアクセスが有効になっているエレメントがあると、コンポーネントのレイアウト上に、そのエレメントの種類に応じたピンが表示されます。パラメータ（特性カーブとマップを含む）、定数、システム定数、送信メッセージの場合は表示できるピンは出力ピンのみで、受信メッセージの場合は、入力ピンのみです。また変数、および送受信メッセージの場合は、入力ピンと出力ピンの両方を追加することができます。

4.10.2 依存エレメント

パラメータは、他のエレメントに依存する「依存エレメント」にすることができます。

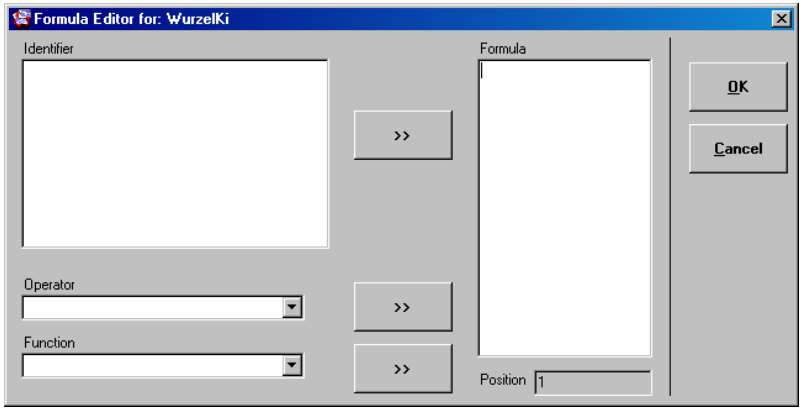
依存パラメータ用の変換式を作成する：

- エレメントエディタで、“Dependency”グループボックスから **Dependent** オプションを選択します。

  Formula ボタンが有効になります。

- **Formula** ボタンをクリックします。

  依存パラメータ用のフォーミュラエディタ（“Formula Editor”ダイアログ）が開きます。



  ここで、エレメントの依存関係を表わす式を C の構文で入力することができます。“Formula”フィールド内のカーソルの位置は、“Position”フィールドに表示されます。

- “Identifier”フィールドを右クリックし、ショートカットメニューから **Add** を選択します。

  これで、仮パラメータが作成されます。仮パラメータはモデルに依存しません。

- “Identifier”フィールドに名前を入力します。

  “Operator”コンボボックスに、使用できる演算子のリストが表示されます。
  “Function”コンボボックスに、使用できる数学関数のリストが表示されます。

  “Formula”フィールドで、これらの仮パラメータと演算子および関数を使用して変換式を作成することができます。

  これには 2 通りの方法があります。

1. “Formula”フィールドに式を直接入力します。

または

2. 以下のように **>>** ボタンを使用します。
   - “Function”コンボボックスで、関数を選択します。

— 隣の **>>** ボタンをクリックします。

選択された関数が“Formula”フィールドのカーソル位置に表示されます。

— “Operator”コンボボックスで、演算子を選択します。

— 隣の **>>** ボタンをクリックして、演算子を式に加えます。

— “Identifier”フィールドで、仮パラメータを選択します。

— **>>** ボタンで、パラメータを式に加えます。

演算子、関数、および仮パラメータは、いくつでも追加できますが、一度には 1 つづつしか追加できません。

依存パラメータの変換式は、以下のようにして後から変更することも可能です。

依存パラメータの変換式を編集する：

- 変換式を変更したい依存パラメータを、フォーミュラエディタで開きます。
- 前項の方法で、変換式を編集します。
- 仮パラメータの名前を変更するには、“Identifier”フィールドで仮パラメータを選択し、ショートカットメニューから **Edit** を選択します。
- 新しい名前を入力します。
- 仮パラメータを削除するには、“Identifier”フィールドで仮パラメータを選択し、ショートカットメニューから **Delete** を選択します。

注記

仮パラメータに関して行った名前の変更や削除処理は、“Identifier”フィールド上でのみ有効です。“Formula”フィールドの変更は、マニュアル操作で行う必要があります。

モデル内のエレメントの依存関係を有効にするためには、モデルに応じて仮パラメータを定義する必要があります。

仮パラメータを定義する：

- “Elements”ペインで、編集したいエレメントを強調表示します。

- **Element → Edit Data** を選択します。

または

- “Elements”ペインか描画領域でエレメントを右クリックし、ショートカットメニューから **Edit Data** を選択します。

または

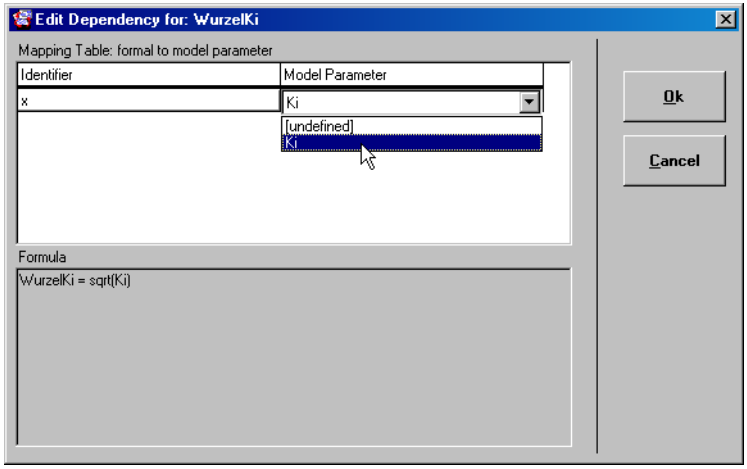
- ブラウザビューで、“Data”タブで編集したいエレメントを選択して、以下の操作を行います。

  — **<Enter>** を押します。

  または

  — “Data”タブのショートカットメニューから **Edit** を選択します。

  “Edit Dependency”ダイアログボックスが開きます。



- “Model Parameter”列の `[undefined]` と表示されたフィールドを右クリックします。

  コンボボックスが開いて、コンポーネント内のパラメータのリストが表示されます。

- リストの中からエレメントを選択します。

  このようにして、依存パラメータの値を決定するモデルパラメータを、仮パラメータに割り当てます。

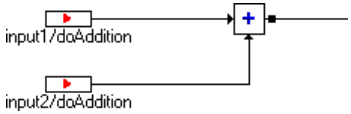
- “Edit Dependency”ダイアログボックスを閉じます。

「テンポラリ変数」を使用してそこに処理の結果を格納すると、1 つのメソッドまたはプロセス内で同じ処理が何度も実行されるのを回避することができます（テンポラリ変数については、『ASCET リファレンスガイド』の「エレメントの種類」の項を参照してください）。

テンポラリ変数を使用する：

- テンポラリ変数を使用したい演算子を右クリックします。
- ショートカットメニューから、**Temporary Variable** を選択します。

  演算子の出力ピンの部分に塗りつぶされた長方形が表示され、この演算結果がテンポラリ変数に格納されることを示します。



- 一度設定したテンポラリ変数の使用を解除するには、同じコマンドをもう一度選択します。

4.11 データの編集

コンポーネントを定義する際は、その中の各エレメントに初期値を指定することができ、これらの値は後で変更することもできます。また「データセット」、つまり初期値の値の組み合せを何通りか定義しておき、実験中にデータセットを切り替えることによって異なる値のセットに切り替えたり、個々の値を別のデータセットの値に切り替えることができます。

本項では、各種エレメント用のさまざまなデータエディタについて説明します。

データエディタは、通常、まず最初に、エレメントに初期値を設定するためにブロックダイアグラムエディタなどのコンポーネントエディタから起動します。そして同じエディタを実験環境から開くことにより、実験中にエレメントの値を適合することができます（606 ページの「数値エディタ」を参照してください）。また、データエディタを使用してコンポーネントやプロジェクト用のデータセットを定義することもできます。

データセットは、インプリメンテーション（4.12 項を参照してください）には直接は依存しませんが、場合によっては両者間で矛盾が生じる可能性もあります。これはたとえば、1 つのプロジェクト内において、あるデータセットに定義され

たエレメントの値が、インプリメンテーションで定義された値の幅を超える可能性があるためです。このようなデータに関する一貫性の保持は、ユーザーの責任範囲となります。

注記

コード生成時において、基本エレメントの値がインプリメンテーションで定義されたインターバルを超えていると、パラメータの場合はエラーが発生し、変数の場合はワーニングが発生します。

データエディタを開く：

- コンポーネントエディタの“Elements”ペインでエレメントを選択します。
- メニューバーの **Element** → **Edit Data** を選択します。

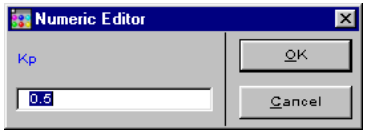
  エレメントエディタの場合と同様に（4.10.1 項を参照してください）、さまざまな方法でエディタを開くことができます。

  選択したエレメントに応じて、以降に説明されているエディタのうちの 1 つが開きます。

4.11.1 スカラ型エレメント用エディタ

数値エディタ

数値エディタには数値エレメントの値が表示され、その値を編集することができます。

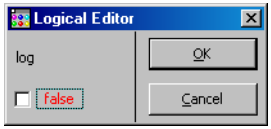


値を編集する：

- 表示されている数値をクリックして入力モードにします。
- 値を編集します。
- **OK** をクリックすると、変更内容が確定されます。

論理値エディタ

論理値エディタには論理エレメントの値が表示され、その値を編集することができます。

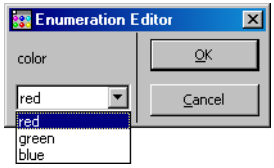


論理値を編集する：

- 値を `True` にするには、チェックボックスにチェックマークを付けます。
- 値を `False` にするには、チェックボックスのチェックマークをはずします。
- **OK** をクリックすると、変更内容が確定されます。

列挙型データ用エディタ

列挙データ用エディタではエレメントに代入される列挙アイテムが定義され、列挙型データの値を選択することができます。



列挙型データの値を選択する：

- コンボボックスをクリックします。
- 希望の値を選択します。
- **OK** をクリックして、変更を確定します。

4.11.2 集合型エレメント用エディタ（テーブルエディタ）

「テーブル」とは、配列、マトリックス、特性カーブ／マップなどの集合型エレメントを定義するための ASCET データ構造体で、モデルの入力（測定値など）に対する出力（制御量など）を、数学関数で求めるのではなく、いくつかのデータポイント（「ブレイクポイント」と呼ばれます）として定義された入力値と出力値の組合せを用いて、実際の出力を求めるためののもです。

テーブルの一般的な作成方法は、208 ページ「配列／マトリックス／特性カーブ／特性マップ」の項を参照してください。テーブルエレメントの編集は、“Elements”ペインから行います。テーブルエディタには、メニューバー、“v:”コンボボックス（現在編集中のテーブル名）、データ表示フィールド、およびテー

ブルサイズや補間モードを変更するためのフィールドが含まれます。以下に各種テーブルエディタについて簡単に説明します。詳しくは、6.2.3「適合ウィンドウの使用方法」の項を参照してください。

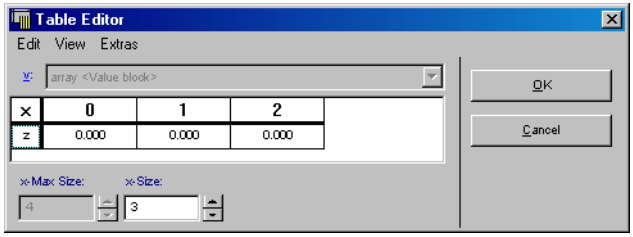
機能記述中、または実験中に、ブレイクポイントの数を増やすことができますが、テーブル作成時に定義された最大ポイント数を超えることはできません。値の最大数を変更する場合は、コンポーネントエディタの **Element → Max Size** コマンドを使用します。

以下に、いくつかの集合型エレメント用のテーブルエディタについて説明します。

配列エディタ

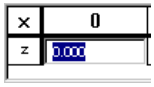
配列は、x 座標の値が固定された 1 次元テーブルです。つまり、x 値は必ず 0 から始まり、1 ずつ増加します。ASCET の配列は、一般的なプログラミング言語の 1 次元配列に相当します。

配列エディタの内容は、テーブルエディタの中で最も単純です。



エディタ中央のテーブルフィールドには、x 値（インデックス値）とそれに対応する出力値が表示され、その下のフィールドに現在の配列サイズが表示されます。

配列を編集する：

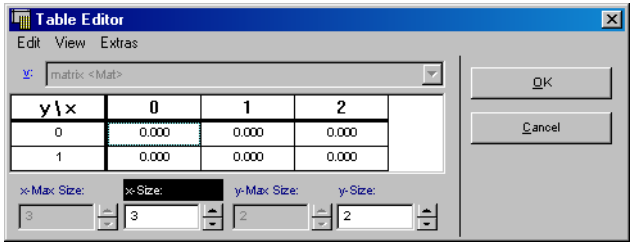


- “x-Size”フィールドで、x 座標の有効ポイント数を設定します。

  ここに指定された数だけ、セルが作成されます。実験中でも、x 座標ポイントを最大ポイント数まで追加することができます。
- エディタウィンドウの z 座標のセルをクリックします。

  セルが入力モードになります。
- 値を入力して **<Enter>** を押します。
- 他のセルについても同様に値を設定します。

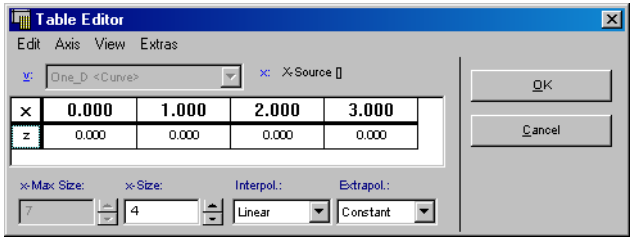
2 次元配列は、マトリックス（行列）の形で表示されます。



テーブルフィールドには入力値（x 軸と Y 軸のインデックス）とそれに対応する出力値が表示され、その下にテーブルサイズに関する 4 つのフィールドが表示されます。

特性カーブ用テーブルエディタ

特性カーブ（“One D Table”とも呼ばれます）には、複数のブレイクポイントが定義され、それぞれに対する入力値（x 座標ポイント）と出力値（Z 値）が定義されます。たとえば、ある特性カーブにおいて、入力値が 0.5 のブレイクポイントに出力値 2 が定義されている場合、プログラム実行時にその特性カーブに 2 が入力された場合、その特性カーブからは必ず 2 が出力されます。2 つの x 座標ポイントの間の値が入力された場合は、それら 2 つの x 座標ポイントに定義された出力値が補間によって求められます。



テーブルフィールドに各ブレイクポイントの入力値と出力値が表示され、その下にテーブルサイズに関する 2 つのフィールドと補間に関する 2 つのフィールドが表示されます。

特性カーブ用テーブルエディタの編集方法は配列エディタと似ていますが、入力値（x 軸）と出力値（y 軸）の両方の座標値を編集できる点が異なります。

特性カーブを編集する：

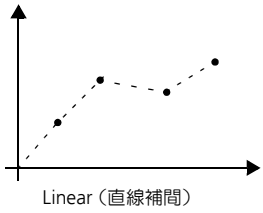
- ブロックダイアグラムエディタで、特性カーブエレメントを右クリックし、ショートカットメニューから **Edit Data** を選択します。

または

- “Elements”ペインの特性カーブエレメントを強調表示し、メニューバーの **Element → Edit Data** を選択します。

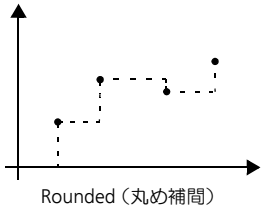
  特性カーブ用テーブルエディタが開きます。
- 452 ページの方法で、z 座標の値を修正します。
- “x-Size”ボックスで、ブレイクポイント数を調整します。

  これにより、指定された個数（n 個）の座標ポイントを x 軸上に持つ特性カーブが作成されます。デフォルトでは、これらの x 座標ポイントの値には `0` から `n-1` までの値が割り当てられます。
- “Interpol.”コンボボックスから補間モードを選択します。

  Linear（直線補間）

  — `Linear` を選択すると、隣り合うブレイクポイント同士を直線で結んだグラフが想定されます。たとえば、x 座標ポイントの値が 1 であるブレイクポイントの出力値が 2 で、かつ x 座標ポイントの値が 2 であるブレイクポイントの出力値が 4 である場合、入力値 1.5 に対する出力値は 3 になります。

  Rounded（丸め補間）

  — `Rounded` を選択すると、2 つの x 座標ポイントの間の入力値に対する出力値は、小さい方のブレイクポイントの出力値と同じになります。`Linear` の例と同じ 2 つのブレイクポイントがある場合、1 より大きく 2 より小さい入力値に対する出力値は、すべて 2 になります。
- **OK** をクリックします。

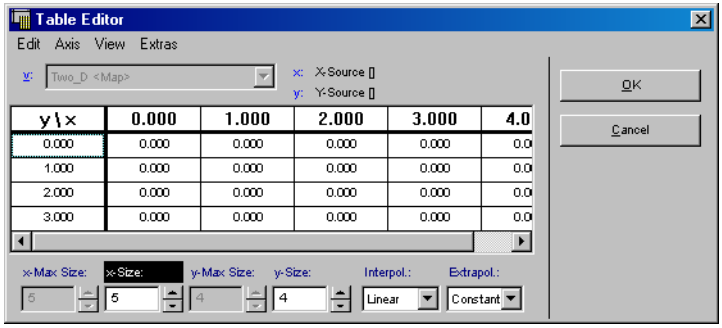
補外モードは `Constant` のみです。つまり、特性カーブに定義されている最大の x 値より大きい値が x 値として入力された場合、特性カーブ内で最大の x 値に対応する z 値が出力されます。また逆に特性カーブに定義されている最小の x 値より小さい値が入力された場合は、最小の x 値に対応する z 値が出力されます。

ブレイクポイントの入力値は特性カーブ内の x 軸に表示され、出力値は z 軸に表示されます。コンポーネントエディタで特性カーブを作成すると、デフォルトでは、“Max Size for: ”ダイアログボックスで指定された最大数のブレイクポイントが作成されます。

特性マップ用テーブルエディタ

特性マップ（“Two D Table”とも呼ばれます）用テーブルエディタは、前項で説明した特性カーブ用テーブルエディタとは異なり、ブレイクポイントが 1 次元ではなく 2 次元で表わされます。つまり 2 次元テーブルでは、2 つの入力値に対して 1 つの出力値が定義されます。

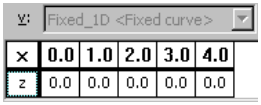
ブレイクポイントの x 値と y 値は、テーブルフィールドの一番上の行と一番左の列にそれぞれ表示されます。特性マップ用テーブルエディタは、テーブルエディタの中では最も構造が複雑です。



固定カーブ／マップ

固定カーブ／マップは、各座標ポイント間の間隔がすべて同じであるテーブルです。間隔は任意の値を指定でき、この値がすべてのポイント間に適用されます。たとえば、ポイント間隔が 3 でオフセットが 4 であれば、1 番目のブレイクポイントの入力値は 4、2 番目は 7、3 番目は 10、というようになります。

座標ポイントを設定する：

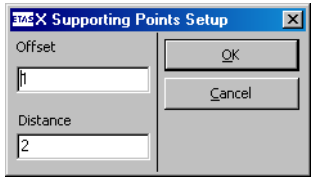


- テーブルエディタで、編集したい固定カーブを開きます。

  左は簡単な固定カーブの例を示しています。

- テーブルエディタで **Axis → X Supporting Points Setup** を選択します。

  “X Supporting Point Setup”ダイアログボックスが開きます。

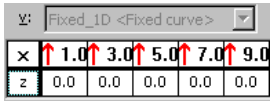


- “Offset”フィールドに、オフセット、つまり 1 番目の座標ポイントの値を入力します。上の例では 1 が入力されています。

- “Distance”フィールドに、各座標ポイント間の距離を入力します。上の例では 2 が入力されています。

注記

両方のフィールドには整数を入力してください。それ以外の数値を入力すると、インプリメンテーション使用時にエラーが発生します。



- **OK** をクリックします。

  入力した値が確定され、座標ポイントの位置が調整されます。

または

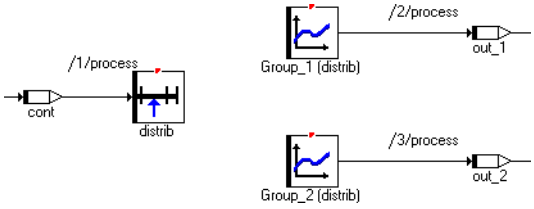
- **Cancel** をクリックして入力を取り消し、元の値に戻ります。

固定マップの Y 座標ポイントは、**Axis → Y Supporting Points Setup** で調整します。エディタの機能は、個々の座標ポイントを変更できない点を除き、他のテーブルエディタの機能と同じです。

グループカーブ／グループマップ

グループカーブとグループマップは、共通の座標ポイントを持ち、出力値のみが異なる複数のカーブ（またはマップ）を指します。これを使えば、1 つの入力に対して複数の戻り値を定義することができます。座標ポイントと個々のグループカーブ（またはマップ）は別のコンポーネントとして定義され、前者は「ディストリビューション」と呼ばれます。

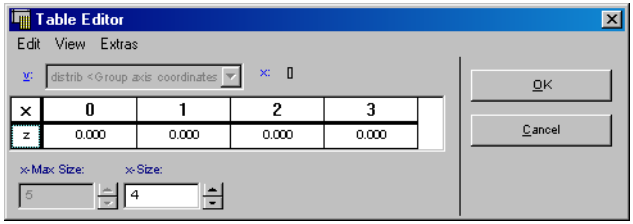
ディストリビューションには座標ポイントのみが定義され、出力値は定義されません。ディストリビューションの定義を行ってから、そのディストリビューションを使用するグループカーブ（またはマップ）を定義します。下図の例のように、ブロックダイアグラムにおいてはディストリビューションには 1 つの入力だけがあり、グループカーブには 1 つの出力だけがあります。これはグループマップの場合も同様です。



グループカーブ／マップ用テーブルエディタ： グループカーブ／マップ用テーブルエディタの使用方法は、通常のテーブルエディタとほぼ同じですが、グループカーブ／マップの作成時には、参照するディストリビューションを指定すること

が要求されます。指定されたディストリビューションはそのカーブ（またはマップ）に対応付けられ、ディストリビューション内に定義されたすべての座標ポイントがグループカーブ（またはマップ）の座標ポイントとして使用されます。グループマップ（またはカーブ）テーブルエディタにおいて X 軸（マップの場合は XY 軸）の値を変更することはできませんが、**Element → Edit Distribution** で、使用するディストリビューションを変更することができます。Z 軸（出力値）への値の割り当ては、通常のテーブルエディタと同様に行います。

ディストリビューションエディタ： ディストリビューションエディタは前出の配列エディタと似ています。主な相違点は、テーブルの Z 軸に座標ポイントの値が定義される点です。X 軸はポイントの番号を表わしているだけなので、この値を変更することはできません。そのためこのエディタでは、X 座標値の変更に関するコマンドは一切使用できません。



4.11.3 データセット

コンポーネントやプロジェクトには、複数のデータセットを定義することができます。データセットには、コンポーネントまたはプロジェクトに含まれるすべてのエレメント（パラメータと変数）の初期値が 1 つずつ設定されているため、複数のデータセットを定義することにより、コンポーネントやプロジェクトのバリエーションを定義することができます。

データセットの扱いはコンポーネントの場合でもプロジェクトの場合も同じです。コンポーネントの各データセットには、そのコンポーネントで使用されるすべての基本エレメントの初期値が含まれています。独自のデータセットを持つ複合エレメントの場合、コンポーネントのデータセットには、その複合エレメントのいずれかのデータセットへの参照が含まれます。

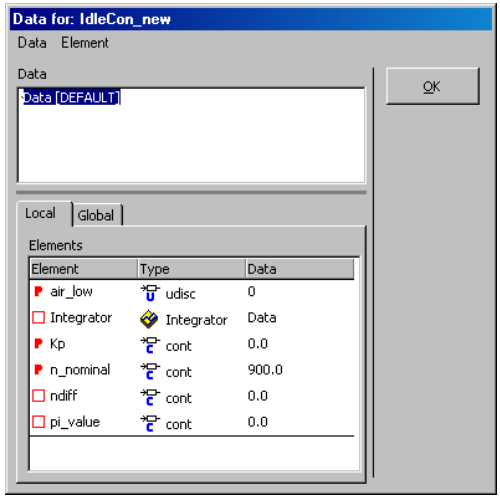
1 つのコンポーネントには複数のデータセットを定義することができます。コンポーネントが新規に作成されると、同時に `Data` という名前のデフォルトのデータセットが作成されます。

データセットを表示する：

- 現在編集中のプロジェクトまたはコンポーネントのデータを表示するには、プロジェクトまたはコンポーネントエディタで **Component → Edit Data** を選択します。

  “Data for: <*Component*>”ダイアログボックスが開きます。

  プロジェクトのすべてのデータセットの名前が“Data”ペインに表示されます。
  またプロジェクトまたはコンポーネントに含まれるすべてのコンポーネントが、その中に含まれるエレメントと共にダイアログボックスの下半分にある“Elements”ペインに表示されます。基本エレメントについては、その値がエレメントの名前と型の後ろに表示され、複合エレメントについては参照するデータセットの名前が表示されます。



- 値を表示したいデータセットを“Data”ペインから選択します。

  選択したデータセットに定義されている値が、“Elements”ペインに表示されます。

- **OK** をクリックします。

データエディタを閉じる時点で選択されていたデータセット、つまり最後に値が表示されていたデータセットが有効になり、その中に定義された値が各エレメントの初期値として使用されます。データセットエディタは、ブラウザの“Data”タブに表示されている被参照コンポーネントを選択して開くこともできます。

新しいデータセットを作成／コピーする：

- **Data → Add** を選択します。

  新しいデータセットが作成されます。すべてのエレメントにはデフォルト値が割り当てられます。基本エレメントのデフォルト値は 0.0 か `true`、複合エレメントのデフォルト値はその複合エレメントのデフォルトデータセットの値です。

または

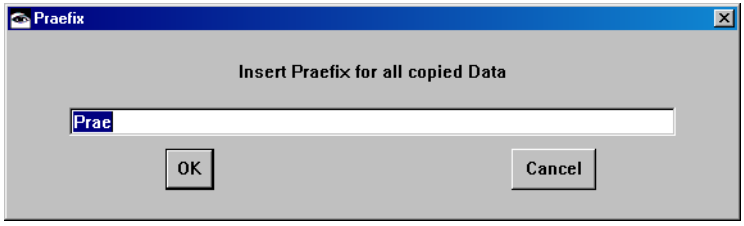
- **Data → Copy → Flat** を選択します。

  現在有効になっているデータセットのコピーが作成されます。新しく作成されるデータセットには同じ値が設定され、他のデータセットへの参照がある場合には、それらもコピーされます。

または

- **Data → Copy → Recursive** を選択します。

  入力ダイアログボックスが開きます。



- 作成されるデータセットのコピーの名前（プレフィックスのみ）を入力して **OK** をクリックします。

  現在有効になっているデータセットがコピーされ、同時にすべての被参照データセットのリカーシブコピーが行われます。つまり、被参照データセットについてもコピーが作成され、それらへの参照が、新しいデータセットに設定されます。

注記

この処理は、ローカル変数についてだけ行われます。

データセットを削除する：

- “Data for: <*Component*>”ダイアログボックスで、削除するデータセットを選択します。

- **Data** → **Delete** を選択します。

  選択したデータセットが削除されます。削除されたデータセットが他のデータセットにより参照されていた場合には、削除後に有効になったデータセットが代わりに参照されます。またデフォルトデータセットは削除できません。

データセットの名前を変更する：

- 名前を変更するデータセットを選択します。
- **Data** → **Rename** を選択します。
- 新しい名前を入力してから **<Enter>** を押します。

“Data for: <*Component*>”ダイアログボックスでデータセットを選択すると、それが有効なデータセットになります。しかし、コンポーネントエディタまたはプロジェクトエディタを 1 度閉じてから再度開くと、デフォルトのデータセットが再び有効になります。コンポーネントやプロジェクトには必ず 1 つの「デフォルトデータセット」が定義されていて、コンポーネントまたはプロジェクトが作成された時には、その時に自動的に作成されたデータセットがデフォルトデータセットとなります。

任意のデータセットをデフォルトデータセットにする：

- “Data for: <*Component*>”ダイアログボックスを開きます。
- デフォルトデータセットにしたいデータセットを選択します。
- **Data** → **Become Default** を選択します。

  選択したデータセットがデフォルトデータセットになります。以降、現在のコンポーネントまたはプロジェクトが開かれると、必ずこのデータセットが有効になります。

データセットに他のデータセットへの参照が含まれている場合には、参照先のデータセットの内容を次々に見る、つまりデータセットをブラウズすることができます。

データセットをブラウズする：

- ブラウズしたいデータセットを選択します。
- ブラウズしたいデータセットの複合エレメント（コンポーネント）を選択します。
- **Element** → **Edit** を選択します。

または

- “Elements”ペインの、ブラウズしたい複合エレメントをダブルクリックします。

  被参照データセットについての“Data for: *<Component>*”ダイアログボックスが開きます。
- この操作を繰り返すことにより、データセット全体を階層構造に従ってブラウズすることができます。

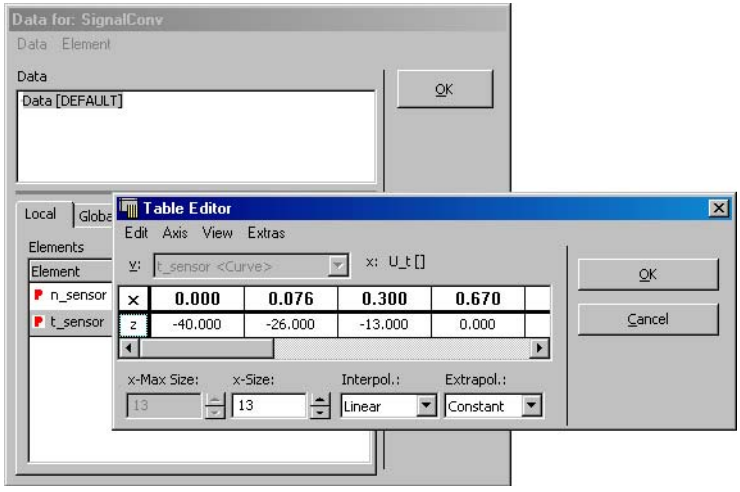
個々のエレメントのデータを編集するには、以下のようにいくつかの方法があります。

データセット内のデータを編集する：

1. データエディタを開く
   - 編集したい基本エレメントを選択します。

     被参照データセット内の基本エレメントを編集する場合は、まず前述の方法でそのデータセットを開く必要があります。
   - **Element → Edit** を選択します。

     エレメントの型に適したデータエディタが開きます。たとえば、特性カーブを選択した場合は特性カーブ用テーブルエディタが開きます。



2. データをコピーする
   - 値のコピー元としたい基本エレメントを選択します。

- **Element → Copy Data To Buffer** を選択します。

または

- ショートカットメニューから **Copy Data To Buffer** を選択します。

  エレメントの値がデータベースのクリップボードにコピーされます。
- 値のコピー先としたい基本エレメントを選択します。

注記

データの交換は、同じ種類のエレメント間でしか行えません。複合型（配列、マトリックス、テーブル）の場合は、両者のサイズが同じである必要があります。

- **Element → Paste Data From Buffer** を選択します。

または

- ショートカットメニューから **Paste Data From Buffer** を選択します。

  エレメントに、データベースのクリップボードから値がコピーされます。

プロジェクトまたはコンポーネントのデータセットのみを個別にエクスポートすることができます。この機能を使えば、データセットと機能定義を並行して開発することも可能です。

データセットをエクスポートする：

- “Data for: <Component>”ダイアログボックスで、エクスポートするデータセットを選択します。
- **Data → Export** を選択します。

  Windows のファイル選択ダイアログボックスが開きます。
- データセット名を入力して **<Enter>** を押します。

  データセットがエクスポートされます。

上記の操作は、プロジェクトまたはコンポーネントエディタのメインメニューの **Component → Export Data** コマンドを使用しても行えます。

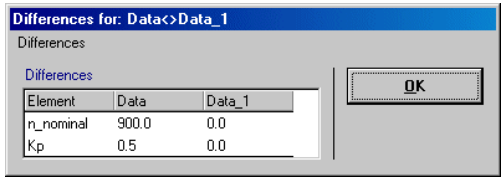
2 つのデータセットの相違を表示する：

- “Data” ペインからコンポーネントまたはプロジェクトのデータセットを 2 つ選択します。

  <Ctrl> キーを押し下げたまま複数のデータセットをクリックすれば、それらを一度に選択することができます。

- **Data → Show Differences → Flat** を選択します。

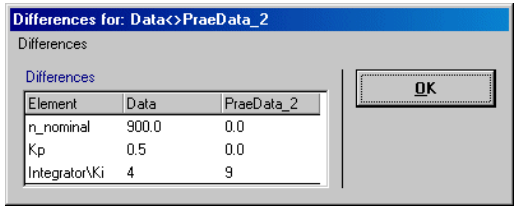
  データセットの第 1 レベル、つまり、プロジェクトまたはコンポーネント内の基本エレメントの値だけが比較されます。参照されるデータセットは比較されません。“Differences” ダイアログボックスが開き、そこに両データセット間の相違が表示されます。スカラエレメントについては、その内容が表示され、その他のエレメントについては、相違が見られたデータの一覧のみが表示されます。



または

- **Data → Show Differences → Recursive** を選択します。

  データセットがリカーシブに比較されます。つまり、参照されるすべてのコンポーネントの基本エレメントの値がすべて比較されます。



- 比較の結果をクリップボードにコピーするには、“Differences” ダイアログボックスで **Differences → Copy To Clipboard** を選択します。

  コピーしたデータは、他のアプリケーションで利用できます。

テーブルまたは配列のデータをファイルに書き込み、それを再び読み取ることができます。データはタブで区切られた ASCII フォーマットでファイルに書き込まれます。

配列またはテーブルのデータをファイルに書き込む：

- データエディタまたはプロジェクト／コンポーネントエディタの“Elements”ペインから、テーブルまたは配列を選択します。
- **Element → File Out Data** を選択します。

  Windows のファイル選択ダイアログボックスが開きます。
- パスおよびファイル名を選択します。
- **Save** をクリックします。

  選択したテーブルまたは配列のデータがファイルに書き込まれます。

配列またはテーブルのデータをファイルから読み取る：

- “Elements”ペインからテーブルまたは配列を選択します。
- **Element → File In Data** を選択します。

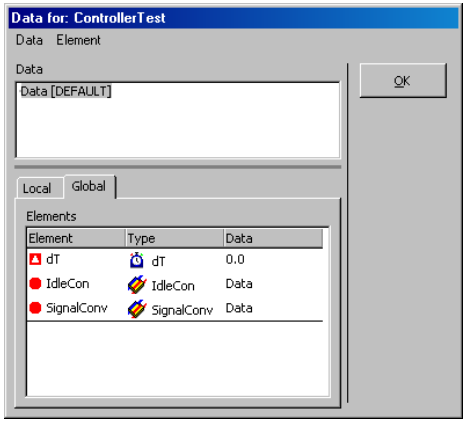
  Windows のファイル選択ダイアログボックスが開きます。
- 読み取りたいデータが入っているファイルを選択します。
- **Open** をクリックします。

このとき、ファイルのフォーマットが選択されているエレメントに対応していない場合は、データセットへの読み込みは行われず、メッセージが表示されます。これはたとえば、ファイルに書き込まれているテーブルの各座標ポイント値が順に増加していないような場合に起こります。またファイル内に欠けている数値がある場合（たとえば、入力値の数がブレイクポイントの数と同じでない場合）、その数値はゼロとして扱われます。また、ファイルに不適当な文字が含まれている場合も、それらはゼロとして解釈されます。エレメントのサイズが一致しない場合は、読み込まれるデータに合わせてエレメントのサイズが調整されます。

コンポーネントとプロジェクトには、データセットの扱いに関して大きな相違点が 1 つあります。プロジェクトには、そのプロジェクトが参照するコンポーネントと同じグローバルエレメントが定義されていますが、1 つのプロジェクトにコンポーネント用のデータセットを複数定義することができるのに対し、このグローバルエレメント用のデータセットは 1 つしか定義することができません。

グローバルエレメントのデータセットの内容を表示する：

- プロジェクトエディタで **Component → Edit Data** コマンドを選択し、データセットエディタを開きます。
- このダイアログボックスの“Global”タブをクリックすると、プロジェクトのグローバルエレメントの値が表示されます。



“Data”ペインに複数のデータセットが表示された場合も、それを選択したり編集することはできません。

ASCET には、データ交換に利用できるツールセットが含まれています。データセット内のパラメータを、各種フィルタやデータ交換オプションを利用して、フラットまたはリカーシブにさまざまなファイルフォーマットで読み取り／書き込みを行うことができます。

データ交換オプションを設定する：

- コンポーネントマネージャから ASCET オプションウィンドウを開きます。
- “Data Exchange”ノードで、オプション設定を行います。

  データ交換オプションについての説明は、65 ページの「データ交換オプション」の項を参照してください。

システム用のデータ交換オプションの設定が済んだら、データ交換ツールセットを使用してデータセットのインポートやエクスポートを行うことができます。このツールセットには、データセットエディタの対応するメニューアイテムを通じてアクセスすることができます。

データ交換ツールセットを使用する：

- データセットエディタで、編集したいデータセットを選択します。
- 選択したデータセットとすべての被参照データセットをファイルに書き込むには、**Data → File Out Recursive** を選択します。
- 選択したデータセットをファイルに書き込むには、**Data → File Out** を選択します。
- データセットとすべての被参照データセットをファイルから読み取るには、**Data → File In Recursive** を選択します。
- データセットをファイルから読み取るには、**Data → File In** を選択します。
- データファイルを表示するには、**Data → Show Data File** を選択します。
- 読み込み処理が記録されたログファイルを表示するには、**Data → Show File In Log File** を選択します。
- 書き込み処理が記録されたログファイルを表示するには、**Data → Show File Out Log File** を選択します。

4.12 インプリメンテーションの編集

インプリメンテーションエディタには、コンポーネントとプロジェクト用、基本エレメント（変数、パラメータ、システム定数）用、およびメソッドとプロセス用の 3 種類があります。エレメントの種類によって設定できる項目が異なり、タブの数も変わりますが、基本エレメント用インプリメンテーションエディタは常に同じ構成です。

インプリメンテーションは、データセット（4.11 項を参照してください）とは独立して定義されますが、場合によっては両者間で矛盾が生じる可能性もあります。これはたとえば、1 つのプロジェクト内において、あるデータセットに定義されたエレメントの値が、インプリメンテーションで定義された値の幅を超える可能性があるためです。このようなデータに関する一貫性の保持は、ユーザーの責任範囲となります。

注記

コード生成時において、基本エレメントの値がインプリメンテーションで定義された範囲を超えていると、パラメータの場合はエラーが発生し、変数の場合はワーニングが発生します。

リテラルについては、最適なインプリメンテーションをコード生成機能が自動的に導き出すので、ユーザーがインプリメンテーションを定義する必要はありません。これらのインプリメンテーションは、最初に代入された値から導き出されます。これは定数の場合も同様です。

4.12.1 コンポーネント／プロジェクトのインプリメンテーション

コンポーネントまたはプロジェクトのインプリメンテーションは、以下のいずれかのウィンドウからインプリメンテーションエディタを開いて設定します。

- コンポーネントマネージャ（87 ページ「インプリメンテーションを編集する：」の項を参照してください）
- 機能記述エディタ（プロジェクトエディタまたは各種コンポーネントエディタ）
- 内包されるコンポーネントのインプリメンテーションエディタ

コンポーネント用のインプリメンテーションエディタを開く：

1. 現在編集中のコンポーネント／プロジェクトの場合：
   - コンポーネントまたはプロジェクトの機能記述エディタから、**Component → Edit Implementation** を選択します。
2. 内包されるコンポーネントの場合（機能記述エディタから開く）：
   - 機能記述エディタの“Elements”リストから、内包されているコンポーネントを選択します。
   - **Element → Implementation** を選択します。

   または
   - ショートカットメニューから **Edit Implementation** を選択します。

   または
   - 以下のように、機能記述エディタのブラウザビューを使用します。

     — **Browse** タブをクリックします。

     — ブラウザビューの“Implementation”タブから内包されるコンポーネントのインプリメンテーションエディタを開きます。87 ページに示されている方法を参考にしてください。
3. 内包されるコンポーネントの場合（インプリメンテーションエディタから開く）：
   - インプリメンテーションエディタの“Elements”リストから、内包されているコンポーネントを選択します。

- **Element → Edit** を選択します。

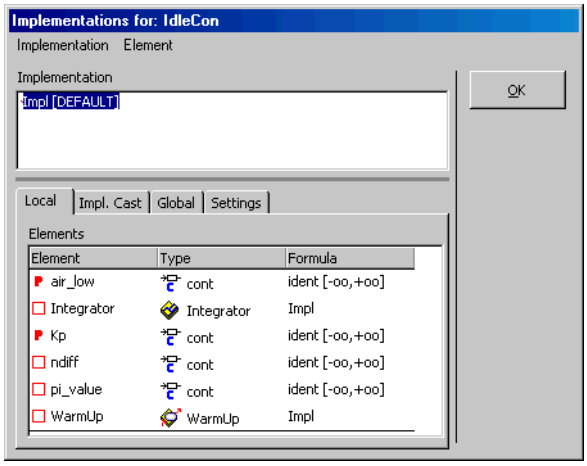
または

- コンポーネントをダブルクリックします。

または

- **<Enter>** キーを押します。

選択されたコンポーネントインプリメンテーションエディタが開きます。



このコンポーネントまたはプロジェクトに定義されているすべてのインプリメンテーションの名前が“Implementation”ペインに表示されます。

プロジェクトまたはコンポーネントに含まれるすべてのコンポーネントとローカルエレメントが“Locals”タブに表示され、インプリメンテーションキャストが“Impl. Cast”タブに、またグローバルエレメントが“Globals”タブに表示されます。基本エレメントには、エレメント名と型の後に変換式が表示されます。

また“Setting”タブには、いくつかのオプション設定項目が含まれます。

インプリメンテーションを選択する：

- “Implementation”ペインで、編集したいインプリメンテーションを選択します。

  選択されたインプリメンテーションの変換式が“Elements”ペインに表示されます。

  ここに表示されているエレメントや内包エレメントをダブルクリックすると、それらのインプリメンテーションエディタが開きます。

- **OK** をクリックするとダイアログボックスが閉じます。

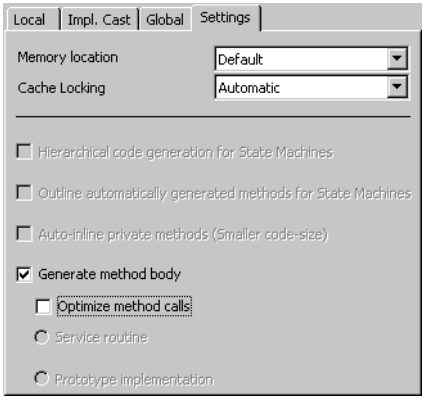
データセットについて実行できる操作（4.11.3「データセット」を参照してください）は、インプリメンテーションについても同様に実行できます。インプリメンテーションエディタのタブを切り替えて、ローカルエレメントとグローバルエレメントの両方のインプリメンテーションを編集することができます。

インプリメンテーションを編集する：

注記

“Settings”タブは、ビルドオプションで `Object Based Controller Implementation` が選択されている場合にのみ機能します。

- “Settings”タブを選択します。



“Memory location”および“Cache Locking コンボボックスはどのコンポーネント（クラス、モジュール、プロジェクト）についても有効ですが、他のオプションは、モジュールとプロジェクトの場合は無効になっています。

- “Memory location”コンボボックスで、コンポーネントのデータをどのメモリ領域に配置するかを指定します。
  C コード用のメモリクラスは、それぞれのメソッドまたはプロセスのインプリメンテーションエディタで指定されます（490 ページの 4.12.8 を参照してください）。

- クラスのインプリメンテーションを編集する際は、**Generate method body** オプションがオンになっていてクラスのコードが生成されるようになっていることを確認してください。

  Hierarchical code generation for State Machines、**Outline automatically generated methods for State Machines**、**Auto-inline private methods (Smaller code-size)** の各オプションは、ステートマシンについてのみ使用されます。これらのオプションの機能については、「アクション／コンディションのアウトライニングのオン／オフを切り替える：」（296 ページ）と「自動インライニングのオン／オフを切り替える：」（285 ページ）を参照してください。

キャッシュロックについては ASCET-RP、その他の設定については ASCET-SE のユーザーズガイドを参照してください。

4.12.2 スカラ型の非論理エレメントのインプリメンテーション

非論理エレメントのインプリメンテーションは、無限のモデル定義（`continuous` または `discrete`）から有限の実装コードへの変換方法を定義します。つまり、モデル内に記述された値に「インターバル」（最小値と最大値）を定義することによりその値を実装値に変換する範囲を規定し、さらに、物理値から実装値への変換を表す一次式を定義する必要があります。

解像度が限りなく精密な `continuous` モデルデータ型の場合、変換式は、固定小数点演算機構を使用する実装コードの量子化を規定します。実装コードは整数演算を利用すると仮定されるため、量子化は一次式の傾きの逆数となります。

基本エレメント用のインプリメンテーションエディタを開く：

1. コンポーネントマネージャから開く場合：
   - “1 Database”リストから、プロジェクトまたはコンポーネントを選択します。
   - “3 Contents”ペインの“Implementation”タブで、インプリメンテーションを編集したい基本エレメントを選択します。
   - **Component → Edit Item** を選択します。

   または
   - エレメントをダブルクリックします。

   または
   - ショートカットメニューから **Edit** を選択します。

   または
   - **<Enter>** キーを押します。
2. 機能記述エディタから開く場合：

- “Elements”リストから、インプリメンテーションを編集したい基本エレメントを選択します。
- **Element → Edit Implementation** を選択します。

または

- ショートカットメニューから **Edit Implementation** を選択します。

または

- 以下のように、機能記述エディタのブラウザビューを使用します。

  - **Browse** タブをクリックします。
  - ブラウザビューの“Implementation”タブからインプリメンテーションエディタを開きます。87 ページに示されている方法を参考にしてください。

3. コンポーネント／プロジェクトのインプリメンテーションエディタから開く場合：

- “Elements”リストから、インプリメンテーションを編集したい基本エレメントを選択します。
- **Element → Edit** を選択します。

または

- エレメントをダブルクリックします。

または

- ショートカットメニューから **Edit** を選択します。

または

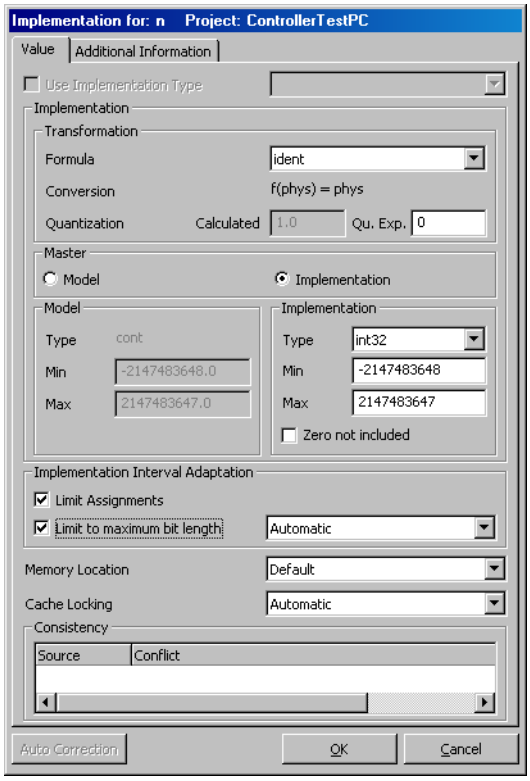
- **<Enter>** キーを押します。

基本エレメントのインプリメンテーションエディタが開きます。

エレメントエディタ（4.10.1 項を参照してください）と同じように、基本エレメントのインプリメンテーションは、さまざまな方法で開くことができます。

新しく作成されたエレメントには、プロジェクトで定義されたインプリメンテーション型（422 ページの「インプリメンテーション型」を参照してください）ではなくローカルなインプリメンテーションが割り当てられ、その際、モデルデータ型 `cont` および `sdisc` に対しては、オプションダイアログボックスの“Implementation”ノード（52 ページ参照）で設定された実装データ型が割り当てられます。

スカラ型の非論理エレメントのインプリメンテーションエディタは、以下のようになっています。



この“Implementaion for”ダイアログボックスでは、**Use Implementation Type** オプションに定義済みのインプリメンテーション型を選択するか（422 ページの「インプリメンテーション型」を参照してください）、または“Implementation”フィールドに独自の実装情報を指定します。どちらの場合も、値のリミッタ機能（“Implementation Interval Adaptation”フィールド）と、マイクロコントローラターゲットの場合にエレメントを配置するメモリ領域（“Memory Location”コンボボックス）は、個々に設定できます。“Consistency”フィールドには、ワーニングやエラーメッセージが表示されます。

“Additional Information”タブには、必要に応じて個々のコードジェネレータに関連する情報を入力します。ここに入力される情報は、ターゲットとコードジェネレータの種類によって異なります。

スカラ型の基本エレメントの場合、**Use Implementation Type** オプションはデフォルト状態においてはオフになっていて、コンボボックスはグレイアウトされています。インプリメンテーションを個々に編集する場合は、このままの設定にしておいてください。

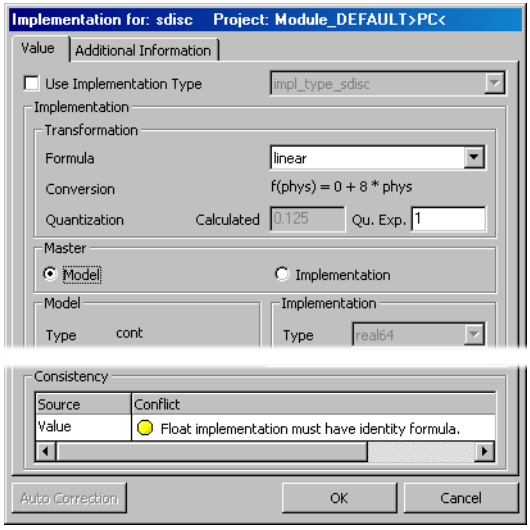
変換式を選択する：

- “Formula”コンボボックスで変換式を選択します。ここではプロジェクトに含まれるすべての変換式から選択できます。

  変換式の内容に従って、“Calculated”フィールドの値が自動的に決定されます。実装データ型が `real*` の場合は、`1` が表示されます。

  また、コード生成オプションが `Quantized Physical Experiment` の場合にのみ使用される量子化の値を、“Qu. exp.”のフィールドに入力することができます。

モデルデータ型が `cont` で実装データ型が `real32` または `real64` である変数の場合、またはモデルデータ型が `cont` 以外である変数の場合は、使用できる変換式は恒等式（`ident`）のみです。これは、コードジェネレータがサポートする変換式が恒等式のみであるためです。別の変換式を選択すると、ワーニングメッセージが表示されます。



- 変換式を恒等式に変更するには、“Formula”コンボボックスから `ident` を選択します。

- **OK** をクリックしてインプリメンテーションエディタを閉じます。

または

- `ident` を選択せずに **OK** をクリックします。

  変換式が確定されます。

または

- **Cancel** をクリックしてダイアログボックスを閉じ、元の設定に戻ります。

モデルデータ型が `sdisc` または `udisc` で、かつ旧バージョンの ASCET において恒等式以外の変換式が割り当てられているエレメントを編集すると、エラーメッセージが表示されます。

注記

旧バージョンの ASCET モデルはそのまま使用できますが、離散型エレメントに変換式が割り当てられていると、コード生成時にワーニングが発生します。

定義済み変換式の内容が変更された場合、またはプロジェクトのインプリメンテーションで変換式が一括して置き換えられた場合、すべてのインプリメンテーションについて自動的にインターバルとデータ型が調整されます。変換式の置き換えについては、422 ページの「インプリメンテーションの一括変更」を参照してください。

インプリメンテーションのマスタページを指定する：

インプリメンテーションエディタでは、モデル側と実装側のどちらか一方の設定値を基準にして、他方の設定を自動更新することができます。どちらを基準（マスタ）にするかは、インプリメンテーションエディタ内の“Master”フィールドで指定します。また ASCET オプションウィンドウの“Implementation”ノードで、デフォルトのマスタを指定しておくことができます（52 ページの「インプリメンテーションオプション」参照）。

- モデル側の設定をマスタにするには、インプリメンテーションエディタの **Model** オプションをオンにします。

  左側の“Model”フィールドの“Min”および“Max”フィールドが入力可能になります。

  “Type”フィールドに表示されるモデルデータ型は、エレメントによって決まっているので、変更できません。

  右側の“Implementation”フィールドの **Zero not included** 以外の各フィールドは、変更できなくなります。

または

- 実装側の設定をマスタにするには、インプリメンテーションエディタの **Implementation** オプションをオンにします。

  右側の“Impelmentation”フィールドの各オプションが入力可能になります。

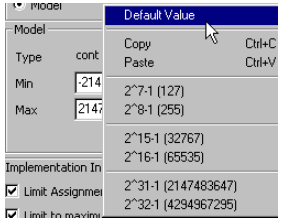
  左側の“Model”フィールドの各オプションの値は、変更できなくなります。

物理モデルを基準にして実装情報を設定する場合は、“Master”フィールドで **Model** オプションを選択してから、以下のように設定を行います。

インプリメンテーションを定義する（モデル側をマスタとする場合）：

- “Min”および“Max”フィールドに物理値のインターバルを入力します。
- モデルデータ型に応じたデフォルトのインターバルを使用するには、各フィールドのショートカットメニューから **Default Value** を選択します。

  モデルデータ型で扱える最大範囲が自動的に入力されます。



Type	Min.	Max.
cont	-oo	-oo
sdisc	-2147483648	2147483647
udisc	0	4294967295

  設定された値に従い、自動的に実装側の値が更新されます。

コード実装を行う際の制約条件（プロセッサのビット幅など）を基準にして実装情報を設定する場合は、“Master”フィールドで **Implementation** オプションを選択してから、以下のように設定を行います。

インプリメンテーションを定義する（実装側をマスタとする場合）：

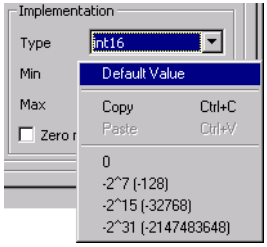
- “Type”コンボボックスで、実装データ型を選択します。

  使用可能なすべての型から選択できます。

- “Min”および“Max”フィールドに実装値のインターバルを入力します。

注記

実装データ型に real* を選択した場合、コード生成時においてはここで設定されたインターバルは無視され、モデル側と実装側で、ともに±∞が使用されます。



- 実装データ型に応じたデフォルトのインターバルを設定するには、各フィールドのショートカットメニューから **Default Value** を選択します。

  選択されている実装データ型で扱える最大範囲の値が自動的に入力されます。

Type	Min.	Max.
real64[a]	-oo	-oo
real32[a]	-oo	-oo
int32	-2147483648	2147483647
uint32	0	4294967295
int16	-32768	-32767
uint16	0	65535
int8	-128	127
uint8	0	255

a: モデルデータ型が cont の場合のみ

設定された値に従い、自動的にモデル側の値が更新されます。

モデル側または実装側に入力された内容は、変換式との間で矛盾がないかどうかがチェックされます。

- モデル側がマスタの場合、実装側のインターバル（“Min”と“Max”）は、変換式を用いて自動的に調整されます。

  モデル側の設定から計算された実装データ型のインターバルが、選択されている実装データ型の最大範囲を超えてしまった場合は、実装データ型が自動的に調整されます（ただし実装データ型が real* の場合を除きます）。この際、ASCET オプションで設定されている最小の型（52 ページ

の「インプリメンテーションオプション」を参照してください）またはそれより大きな型のうち、ここで入力されたインターバルをカバーできる最も小さな型が割り当てられます。

例：実装データ型に `uint8` が選択されている場合、そのインターバルがモデル側で `[-100..255]` に変更されると、実装データ型は以下のように調整されます。

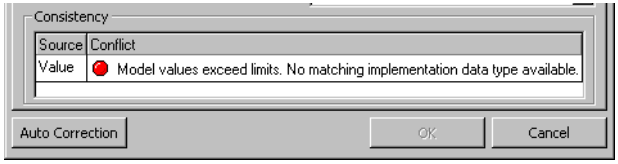
— “Minimum cont Data Type” が `int16` またはそれ以下の型に設定されている場合は、`int16` になります。

— “Minimum cont Data Type” が `int16` より大きい型に設定されている場合、その型が選択されます。

なお、データ型のサイズは以下の順です。

`int8` → `uint8` → `int16` → `uint16` → `int32` → `uint32`

`uint32` でも小さすぎる場合は、エラーメッセージが表示されます。たとえエレメントのモデルデータ型が `cont` で、実装データ型に `real32` または `real64` が使用できる場合も、同様です。



エラーメッセージに対して **Auto Correction** をクリックすると、実装側の設定を基準にしてマスタ側の設定が調整されます。つまりいずれかの 32 ビット整数フォーマットの最大値が適用されます。また **Cancel** をクリックすると、エディタが閉じて元の設定に戻ります。

この後にモデル側でより小さいインターバルが設定されると、実装側のデータ型もそれに合わせて小さくなりますが、“Minimum...Data Type” で指定されている型より小さくはなりません。

- 実装側がマスタの場合、モデル側の最小値と最大値は、変換式を用いて自動的に調整されます。

  新たに作成されたエレメントには、ASCET オプション（52 ページの「インプリメンテーションオプション」を参照してください）で設定されているデフォルトの型が割り当てられます。実装側にこの型で扱える値の範囲を超えるインターバルが入力されると、実装データ型が自動的に調整されます。この際、入力されたインターバルを扱える最小の型が割り当てられます。

  例：`uint8` 型のエレメントに対して、実装側で `uint8` が選択されている場合、入力されたインターバルに応じて実装データ型は以下のように調整されます。

  — `[-100..255]` と入力されると、自動的に `uint16` に変わります。

— `[0..100]` と入力されると、型は `uint8` のまま変わりません。

なお、ここでもデータ型のサイズは以下の順で扱われます。

```
int8 → uint8 → int16 → uint16 → int32 → uint32
```

`uint32` でも小さすぎる場合、コード生成時には、いずれかの 32 ビット整数フォーマットで扱える最大範囲が自動的に使用され、エラーやワーニングは発生しません。

- 新しく設定された値が他方の値に適していない場合、つまり、入力された最小値が最大値よりも大きいような場合は、インプリメンテーションエディタ下部の“Consistency”フィールドにエラーメッセージが表示されるので、**Auto Correction** または **Cancel** をクリックします。
- 実装側がマスタになっていて、かつモデルデータ型が `udisc` で実装データ型が `int*` になっている場合、実装側のインターバルに負の値を入力すると、以下のようなエラーメッセージが“Consistency”フィールドに表示されます。

  ```
  Model type udisc minimum cannot store
       <negative value>.
  ```

- 計算された最小値と最大値が、現在保存されている値と一致しない場合、システムは現在の変換式が間違っていると判断し、以下のエラーメッセージを表示します。

  ```
  Values for min/max are not consistent with
       current formula.
  ```

  上記のエラーが発生するのは、たとえば、プロジェクト内で変換式を編集した後、インプリメンテーションを更新していないような場合です（422 ページ参照）。

以降の設定は、どちらがマスタになっているかには関係なく行えます。

デフォルト状態においては **Zero not included** オプションはオフになっているため、実装値のインターバルにゼロが含まれている、という前提でコード生成が行われます。そのため、除算において分母（除数）がゼロであるかどうかがチェックされ、必要に応じて実行時のゼロ除算を避けるための C コードが生成されます。このチェックが不要な場合、以下のようにしてこの機能を無効にすることができます。

実装値のインターバルからゼロを除外する：

- 設定されている実装値のインターバルにゼロが含まれない場合は、**Zero not included** オプションをオンにします。

  コード生成時において、この変数はゼロの値をとらないという前提条件のもとに、この変数が分母として使用されていてもチェックは行われません。

注記

ゼロ除算が発生すると、システムに深刻な障害が発生します。このチェック機能を無効にした場合、つまり **Zero not included** オプションをオンにした場合は、ユーザーの責任範囲において ASCET モデル内に適切な対策を設け、ゼロ除算が絶対に発生しないようにしてください。

旧バージョンのモデルでは、インプリメンテーションの定義された演算子に接続されていないすべてのエレメントについては **Zero not included** がオフになっています。旧バージョンのモデルに含まれる演算子インプリメンテーションの扱いについては、4.12.11「演算子のインプリメンテーション」の項を参照してください。

演算処理によって得られた値が、その値が代入される変数のインターバルを超えてしまう場合があります。リミッタ機能を有効にしておくと、その変数へのすべての代入処理において、代入される値の範囲がチェックされます。つまり、代入時に値の範囲を制限するコードが生成されます。これによって、個々の変数への代入時に値の範囲をチェックする機能をマニュアル操作で追加する必要がなくなります。

スカラエレメントのリミッタを設定する：

注記

ここで使用される **Limit Assignments** オプションは、新しく作成されたエレメントについてはデフォルトでオンになりますが、モデルデータ型 `udisc`（443 ページ参照）のエレメントに限り、このオプションは自動的にオフに設定されます。

エレメントの実装データ型が `real64` または `real32` に設定されている場合は、このオプションは変更できません。

- **Limit Assignments** オプションをオンにします。

  これによって、コードジェネレータが代入のコードを生成する際、その変数に定義された“Min”や“Max”の値が考慮されます。

注記

コードジェネレータに `Physical Experiment` が選択されている場合（405 ページ参照）は、このオプションの設定は無視されます。

代入される値は、定義された値のインターバルの範囲に制限されます。代入される値がインターバル外であった場合、代わりに“Min”または“Max”の値が代入されます。

または

- **Limit Assignments** オプションをオフにします。

  この場合、コードジェネレータが代入のコードを生成する際、その変数に定義された“Min”や“Max”の値は考慮されません。

  実装データ型（`int8`、`int16` など）が扱える最大範囲の値まで代入される可能性があります。

旧バージョンの ASCET のインプリメンテーションエディタでは、上記の **Limit Assignments** オプションの代わりに、“Use Limiters”フィールドの **Yes** ／ **No** オプションが使用されていました。旧バージョンのモデルを ASCET V5.2 で使用すると、**Limit Assignments** オプションは、この旧エディタの設定内容に応じて設定されます。

以下のようなオーバフロー発生時の処理を定義するオプションも用意されています。これらは主に算術演算サービスに関連するものです（4.14「算術演算サービス」の項を参照してください）。

オーバフロー処理を設定する：

- 演算処理結果がオーバーフローしないようにするには、**Limit to maximum bit length** オプションをオンにします。

  結果の値が“Integer Arithmetic”ノード（410 ページ参照）で指定された範囲を超える場合、オーバーフローを回避するコードが生成されます。

  回避する方法は、右側のコンボボックスから以下の処理モードを選択できます。

  — リミッタ処理において値の精度が低くなることを許可するには、`Reduce resolution` を選択します。

  必要に応じて、入力値を右シフトすることによりオーバーフローが回避されます。シフトの必要性は自動的に決定されます。

  — リミッタ処理において値の精度が低くならないようにする場合は、`Keep resolution` を選択します。

  算術演算サービスが使用されていると、必要に応じてリミッタサービスによるオーバーフローの回避が行われます。
  使用されていないと、コード生成は中断されてエラーメッセージが出力されます。

  — 算術演算サービスが使用されているかどうかによってオーバーフロー処理が自動的に選択されるようにするには、`Automatic` を選択します。

  算術演算サービスが使用されている場合は精度が保持（`Keep resolution`）され、ない場合は精度の低下（`Reduce resolution`）が行われます。

旧バージョンのモデルを ASCET V5.2 で使用すると、**Limit to maximum bit length** オプションは、“Use Limiters”フィールドの設定内容に応じて自動的に設定され、処理モードには `Automatic` が選択されます。旧バージョンの ASCET で使用されていた演算子インプリメンテーションの扱いについては、4.12.11 項を参照してください。

メモリ位置を選択する：

- “Memory Location”コンボボックスで、エレメントを配置するメモリ領域を選択します。

  指定されているターゲットマイコンに応じて、使用可能なメモリ領域がすべて表示されます。

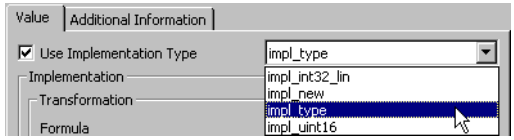
インプリメンテーション型の使用方法

エレメントのインプリメンテーションを個々に設定する代わりに、定義済みのインプリメンテーション型を割り当てることができます。インプリメンテーション型の作成方法は 422 ページの「インプリメンテーション型」に説明されていて、ここではその使い方を説明します。

現在のプロジェクトに定義されているすべてのインプリメンテーション型が利用できます。この「プロジェクト」は、コンポーネントエディタからインプリメンテーションを編集している場合は、「デフォルトプロジェクト」（4.8.1 項を参照してください）を指し、プロジェクトから編集している場合は、そのプロジェクト自身を指します。

インプリメンテーション型を割り当てる：

- 基本エレメントのインプリメンテーションエディタを開きます。
- **Use Implementation Type** オプションをオンにします。



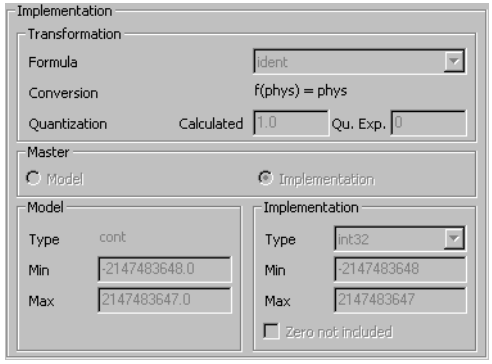
利用可能なインプリメンテーション型を選択するためのコンボボックスが有効になります。

- 割り当てる型を選択します。

注記

1 つのインプリメンテーション型は、1 種類のモデルデータ型の変数に対してしか割り当てできません。たとえば、`cont` 型に割り当てられたインプリメンテーション型を `sdisc` や `udisc` 型の変数に割り当てることはできません。

モデル側と実装側の設定が自動的に設定されます。“Implementaion" フィールドはロックされて個別に変更できなくなります。



- 480 ページの方法で、リミットオプションを設定します。
- “Memory Location” コンボボックスでエレメントを配置するメモリ領域を選択します。

インプリメンテーション型は、基本エレメントのインプリメンテーションから参照されます。インプリメンテーション型のコピーを使用して個別に変更を加えたい場合には、まずインプリメンテーション型を選択してその設定を読み込み、その後にインプリメンテーション型の割り当てを無効にします。すると、インプリメンテーション型から読み込まれた設定を、個別に変更することが可能になります。

注記

上記の操作を行った後に再びインプリメンテーション型を有効にすると、個別に変更された設定は破棄されます。

4.12.3 メソッドローカル／プロセスローカル変数のインプリメンテーション

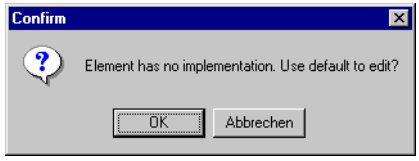
メソッドローカルおよびプロセスローカル変数のインプリメンテーションは、自動的（デフォルト）または明示的に指定されます。デフォルトでは、メソッド／プロセスローカル変数に最初に代入された変数のインプリメンテーションが使用されます。この「インプリメンテーションの自動割り当て」機能を無効にするには、以下のようにします。

メソッド／プロセルローカル変数のインプリメンテーションを定義する：

- コンポーネントのインプリメンテーションエディタで、“Local”タブを選択します。
- “Elements”フィールドで、メソッド／プロセスローカル変数を選択します。

  明示的なインプリメンテーションが定義されていない時は、メニューコマンドの **Element → Use automatic Implementation** は、チェックマークが付き、グレーアウトされた状態になっています。
- メソッド／プロセスローカル変数のインプリメンテーションエディタを開きます。

  インプリメンテーションの自動割り当て機能が有効になっていると、以下のワーニングメッセージが表示されます。



- **OK** をクリックして確定します。

  インプリメンテーションエディタが開きます。
- 470 ページの「スカラ型の非論理エレメントのインプリメンテーション」の項に書かれてある方法で、変数のインプリメンテーションを編集します。

  これで、インプリメンテーションの自動割り当て機能が無効になってメニューコマンド **Element → Use automatic Implementation** のチェックマークが削除され、このコマンドが使用できるようになります。

メソッド／プロセスローカル変数のインプリメンテーションの自動割り当て機能を再び有効にする操作は、コンポーネントのインプリメンテーションエディタ内で行います。

インプリメンテーションの自動割り当て機能を有効にする：

- “Elements”フィールドで、ローカル変数を選択します。
- **Element → Use automatic Implementation** を選択します。

または

- 変数名を右クリックし、ショートカットメニューから **Element → Use automatic Implementation** を選択します。

これによって、メソッド／プロセスローカル変数へのインプリメンテーションの自動割り当て機能が有効になります。**Element → Use automatic Implementation** メニューにはチェックマークが付いて、操作不可能となります。

変数のインプリメンテーションを明示的に設定することにより、インプリメンテーションの自動割り当て機能は無効になります。

4.12.4 テンポラリ変数のインプリメンテーション

演算子や複合エレメントの出力に「テンポラリ変数」を定義することができますが、このテンポラリ変数のインプリメンテーションは、コードジェネレータによって自動的に決定されます。そのテンポラリ変数がインプリメンテーションが定義された変数に初めて代入される際に、適切な変換式をインターバルが与えられ、その変換式とインターバルに適したインプリメンテーション型が選択されます。

注記

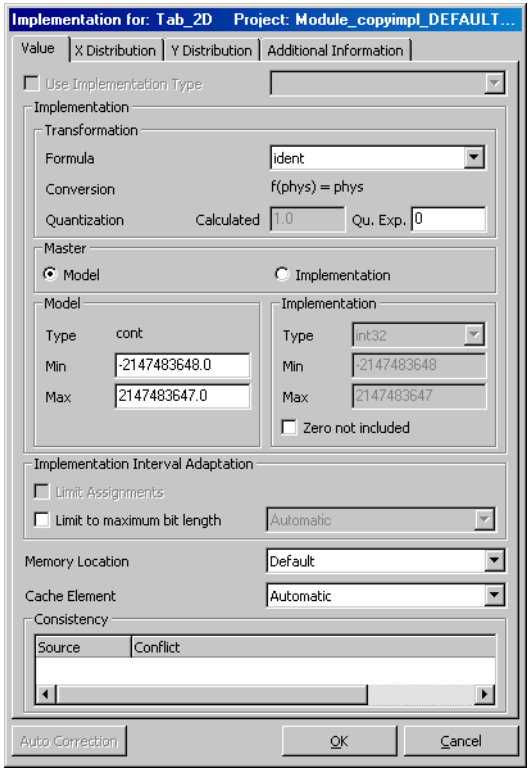
算術演算式内にテンポラリ変数を挿入しても、この式の算術演算のコード生成には影響しません。またテンポラリ変数は、制御フロー内の別の枝で使用する（例：`If` 文の一方の枝のテンポラリ変数を他方の枝で使用する）ことはできません。枝によって、結果とインプリメンテーション（例：量子化）が異なる場合があるためです。これが守られていないと、生成されたコード内に重大な計算エラーが含まれることになります。

4.12.5 配列／マトリックス／特性カーブ／特性マップのインプリメンテーション

配列とマトリックスには、1 つのインプリメンテーションが、スカラエレメントと同じように定義されます。また特性カーブには、入力値用と出力値用にそれぞれ 1 つずつ、合計 2 つのインプリメンテーションが定義され、特性マップには、2 つの入力値と 1 つの出力値用に、合計 3 つのインプリメンテーションが定義されます。

特性カーブ／マップのインプリメンテーションを定義する：

- 特性カーブまたは特性マップ用のインプリメンテーションエディタを開きます。



エディタ上部の“Value”、“X Distribution”、さらに特性マップの場合は“Y Distribution”タブで、それぞれのインプリメンテーションを編集します。

- 編集したいインプリメンテーションのタブを選択します。
- インプリメンテーションを編集します。

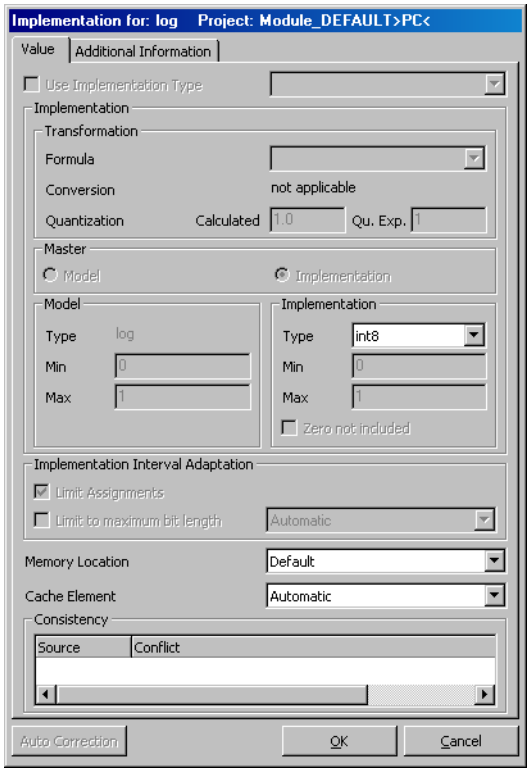
  編集の方法は、スカラエレメントの場合と同様です（470 ページの「スカラ型の非論理エレメントのインプリメンテーション」を参照してください）。
- 必要に応じて、他のタブで別のインプリメンテーションを編集します。

- **OK** をクリックします。

4.12.6 論理エレメントのインプリメンテーション

論理エレメントのインプリメンテーションを定義する：

- 470 ページの「基本エレメント用のインプリメンテーションエディタを開く：」に説明されている方法で、論理エレメント用のインプリメンテーションエディタを開きます。



“Value”タブのすべてのフィールド（“Type”、“Memory Loacation”、“Cache Locking”コンボボックスを除く）は、論理エレメントのインプリメンテーションには必要ないため、無効になっています。

- “Type”コンボボックスで実装データ型を選択します。

  使用できる型は、`bit`、`int8`、`int16`、`int32`、`uint8`、`uint16`、`uint32` のいずれかです。

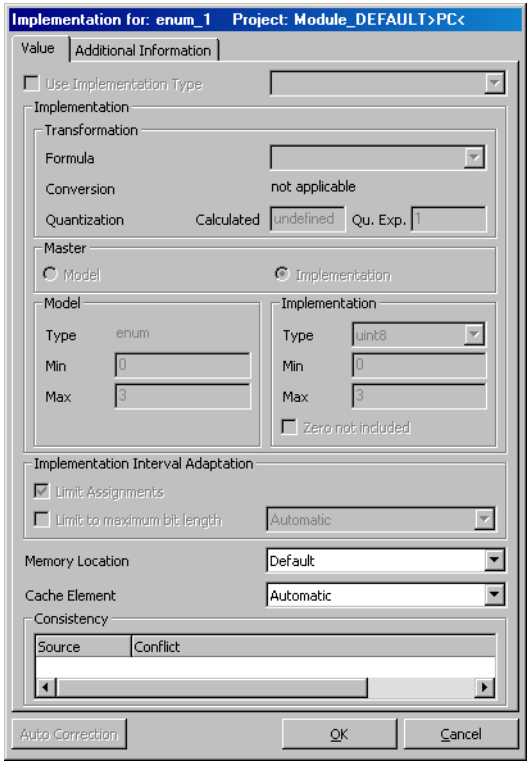
- “Memory Location”コンボボックスで、エレメントを配置するメモリ領域を選択します。

  指定されているターゲットマイコンに応じて、使用可能なメモリ領域がすべて表示されます。マイコンターゲット他のターゲットが指定されている場合、この設定は無視されます。
- “Additional Information”タブには、必要に応じて、使用するコードジェネレータに関連する情報を入力します。

  ここに入力できる情報は、ターゲットとコードジェネレータにより異なります。

列挙型のインプリメンテーションを定義する：

- 470 ページの「基本エレメント用のインプリメンテーションエディタを開く：」に説明されている方法で、列挙型用のインプリメンテーションエディタを開きます。



“Value”タブのすべてのフィールド（“Type”、“Memory Loacation”、“Cache Locking”コンボボックスを除く）は列挙型のインプリメンテーションには必要ないため、無効になっています。

- “Memory Location”コンボボックスで、エレメントを配置するメモリ領域を選択します。

  指定されているターゲットマイコンに応じて、使用可能なメモリ領域がすべて表示されます。マイコンターゲット他のターゲットが指定されている場合、この設定は無視されます。

- “Additional Information”タブには、必要に応じて、使用するコードジェネレータに関連する情報を入力します。

  ここに入力できる情報は、ターゲットとコードジェネレータにより異なります。

4.12.8 メソッドとプロセスのインプリメンテーション

ASCET では、データセット以外にも、クラス内やモジュール内に定義されているメソッドやプロセスについてもインプリメンテーションを定義することができます。これにより、システム全体のパフォーマンスを向上させることができます。

メソッド／プロセスのインプリメンテーションは、コンポーネントエディタで定義します。プロジェクトエディタから直接定義することはできません。プロセスとメソッドのインプリメンテーションはブロックダイアグラムまたは ESDL で記述されたコンポーネントで使用できます。

メソッドまたはプロセスのインプリメンテーションは、そのメソッド／プロセスからどのような C コードが生成され、またそれらがどのメモリ領域に配置されるかを規定するものです。

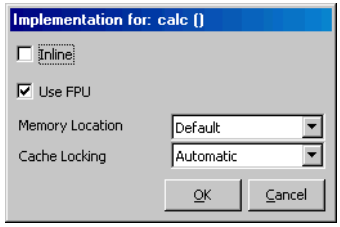
メソッド／プロセスのインプリメンテーションを編集する：

- 編集したいクラス／モジュール用のコンポーネントエディタを開きます。
- “Diagrams”ペインから、編集したいメソッドまたはプロセスを選択します。
- **Diagram → Edit Implementation** を選択します。

または

- ショートカットメニューから **Edit Implementation** を選択します。

  メソッド／プロセス用のインプリメンテーションエディタが開きます。



- メソッドコードがモデルコード内に直接展開して挿入されるようにするには、**Inline** オプションをオンにします。

注記

プロセスの場合、**Inline** オプションはありません。

これによって、メソッド呼び出しが行われなくなるため、実行速度は速くなりますが、同じメソッドが複数の個所で使用される場合は、それに応じたメモリが必要となります。

- タスク切り替え時において、マイクロコントローラターゲットの浮動小数点レジスタが保存されるようにするには、**Use FPU** オプションをオンにします。

  このオプションは ERCOS^{EK} オペレーティングシステムの機能の一部で、マイクロコントローラに依存する部分であるため、詳しい機能は『ASCET-SE V5.2 ユーザーズガイド』および『ERCOS^{EK} ユーザーズガイド』に説明されています。

- “Memory Location”リストでコードが実行されるメモリ領域を選択します。

  一般に、パフォーマンスを向上させるには、処理時間が長く呼び出しの頻度が低い処理を外部メモリで実行し、処理時間が短く頻繁に呼び出される処理を内部メモリで実行します。

4.12.9 メソッドの引数と戻り値のインプリメンテーション

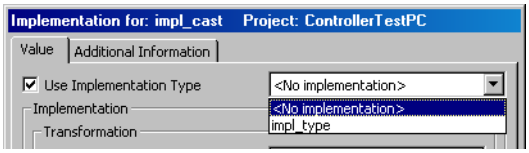
メソッドの引数と戻り値のインプリメンテーションは、同じタイプのエレメントと同様に定義します（4.12.1 項、4.12.2 項、4.12.4 項、4.12.5 項、4.12.6 項、4.12.7 項をそれぞれ参照してください）。

スカラ型と論理型の引数と戻り値、および `<enumeration>` 型の引数と戻り値については、“Memory location”コンボボックスは無効になります。これらのエレメントは STACK メモリクラスに配置されます。

ただし引数と戻り値に「参照」が用いられる場合、つまり `<array[*]>`、`<mat[*]>`、`<user defined>` のいずれかが選択されている場合、“Memory location”コンボボックスでメモリクラスを選択することができます。この場合、選択したメモリクラスは、「参照」に対して適用されるのではなく、参照されるエレメントに適用されます。

4.12.10 インプリメンテーションキャストのインプリメンテーション

インプリメンテーションキャストも、スカラ型エレメントと同様に（4.12.2 項を参照してください）、インプリメンテーションを定義することができます。他のエレメントの場合と異なる点は、“Implementation Type”コンボボックスで <No implementation> というオプションを選択することにより、「インプリメンテーションを使用しない」という設定を行うことができる点です。新しく作成されたインプリメンテーションキャストには、この <No implementation> がデフォルトとして選択されています。



注記

<No implementation> が選択されているインプリメンテーションキャストっは、コード生成時には、それが存在しないものとして扱われます。

4.12.11 演算子のインプリメンテーション

旧バージョンの ASCET では、ブロックダイアグラム内の演算子にもインプリメンテーションを定義することができ、これは浮動小数点演算の精度を高めるために利用されていました。

注記

現バージョンの ASCET では、新しいインプリメンテーションオプションである **Limit to maximum bit length** および **Zero not included** オプションが、演算子インプリメンテーションの役割を果たしています。さらに、インプリメンテーションキャストを使用することによって、連鎖的に処理される算術演算の演算子の量子化を、余分なメモリスペースを増やすことなく定義することが可能です。
このため、現在の ASCET には新しい演算子イプリメンテーションを作成する機能はありません。また旧バージョンで作成された演算子インプリメンテーションは、参照したり、インプリメンテーションキャストに置き換えたり（497 ページの「演算子インプリメンテーションの自動変換」を参照してください）、削除することはできますが、編集することはできません。

演算子はグラフィック上では名前がなく“Elements”リストにも表示されないので、演算子用のインプリメンテーションビューアには、グラフィックブロック内の演算子アイコンからしかアクセスできません。

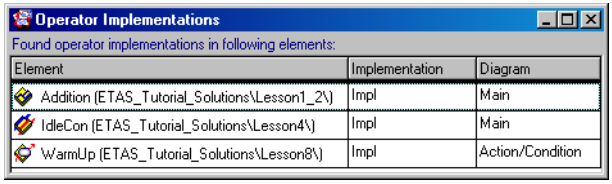
インプリメンテーションが定義された演算子は、そのアイコンの左上に短い斜線が表示されて区別されます（  ）。

以下のようにコンポーネントマネージャの **Tools → Database → List Operator Implementation** コマンドを使用して、データベース内の演算子インプリメンテーションを検索することができます。

演算子のインプリメンテーションを検索する：

- コンポーネントマネージャで、**Tools → Database → List Operator Implementations** を選択します。

  データベース内の検索が行われますが、データベースの大きさによってはこの処理には数秒かかる場合があります。検索された演算子のインプリメンテーションは“Operator Implementation”ウィンドウに一覧表示されます。



  “Element”列には演算子インプリメンテーションが含まれるコンポーネントまたはプロジェクトが表示されます。また“Implementation”列にインプリメンテーションの名前、“Diagram”列に演算子インプリメンテーションを含むダイアグラム名が表示されます。

- “Elements”列に表示されたコンポーネント名をダブルクリックします。

  コンポーネント用のエディタが開きます。演算子インプリメンテーションが定義された演算子が強調表示され、このインプリメンテーションを参照したり削除することができます。

個々の演算子のインプリメンテーションを削除する：

- 削除したい演算子インプリメンテーションを含むブロックダイアグラムを開きます。
- 演算子インプリメンテーションが定義されている演算子を右クリックし、ショートカットメニューから **Implementation → Reset** を選択します。

  そのインプリメンテーションが削除され、演算子の左上の斜線がなくなります。

コンポーネントまたはデータベース全体のインプリメンテーションを削除する：

- コンポーネントマネージャの“1 Database”フィールドからコンポーネントを選択します。
- コンポーネント内の演算子インプリメンテーションを削除するには、**Component → Remove Operator Implementation → Flat** を選択します。
- コンポーネント内、およびそのコンポーネントが参照するコンポーネント内の演算子インプリメンテーションを削除するには、**Component → Remove Operator Implementation → Recursive** を選択します。

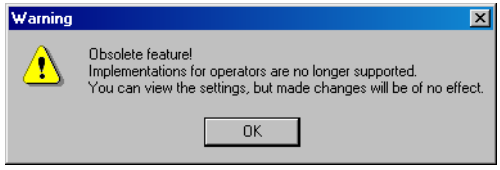
または

- データベースに含まれるすべての演算子インプリメンテーションを削除するには、**Tools → Database → Convert → Reset Operator Implementations** を選択します。

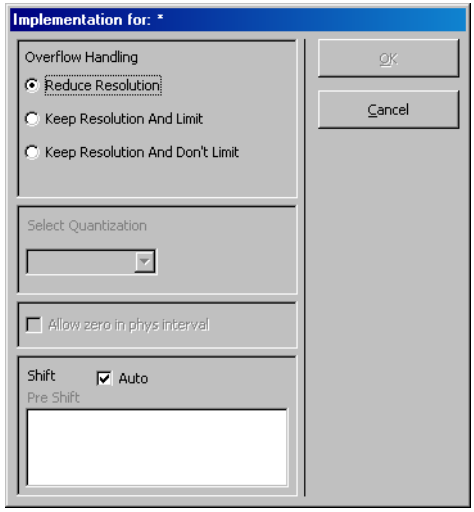
演算子のインプリメンテーションの内容を表示する：

- 目的の演算子インプリメンテーションを含むブロックダイアグラムを開きます。
- 演算子を右クリックし、ショートカットメニューから **Implementation → View** を選択します。

  演算子インプリメンテーションはすでにサポートされていないことを示す警告メッセージが表示されます。



- **OK** をクリックして、“Implementation for” ダイアログボックスを開きます。



このダイアログは表示専用で、設定内容の変更を行うことはできません。

- **OK** をクリックします。

演算子のインプリメンテーションは、演算結果におけるオーバフローや量子化の有無を規定するものです。変数の場合は、モジュールのインプリメンテーションに応じて内容が異なります。以下に演算子の種類に応じた情報の内容を示します。

注記

整数コードジェネレータにおいては、各基本演算の扱いはそれぞれ異なるため、演算の種類に応じて、演算子インプリメンテーションの情報が異なります。

加算と減算：

Overflow Handling（オーバーフロー対策）：

- *Reduce Resolution* - 両方の入力を必ず右シフトして、オーバーフローを防ぎます。
- *Keep Resolution And Limit* - シフトは行いません。クリッピングを行う算術演算サービスがあればそれを適用します。
- *Keep Resolution And Don't Limit* - シフトを行わないで通常の算術演算を行うので、オーバーフローが起こる可能性があります。この方法はたとえば、周期的にオーバーフローを起こさせるカウンタなどに利用します。

Select Quantization（量子化の選択）：

- *Auto* - 最適化の方針に基づく量子化を行います。
- *1* - 第 1 の入力の量子化と型を適用します。
- *2* - 第 2 の入力の量子化と型を適用します。

乗算 :

Overflow Handling：

- *Reduce Resolution* - 両方の入力を必ず右シフトして、オーバーフローを防ぎます。
- *Keep Resolution And Limit* - シフトは行わず、算術演算サービスルーチンを使用できる場合はそれを使用します。
- *Keep Resolution And Don't Limit* - シフトを行わないで通常の算術演算を行うので、オーバーフローが起こる可能性があります。

Pre-Shift（演算前のシフト処理）

演算の前に両方のオペランドをシフトしてオーバーフローを回避し、各オペランドを数値的に意義が失われないようにスケーリングし直すことができます。また、この「プレシフト」処理により、値の範囲をフルに活用することができます。どちらのオペランドについても、-31 から 31 までの整数値を入力することができます。正数は左シフトを、負数は右シフトを表し、ゼロはシフトを行わないことを表わします。

- *Auto* オプションがオンになっていると、プレシフトを行いません。
- *Auto* オプションがオフになっていると、“Pre Shift”フィールドの設定に従って以下のようなプレシフトが行われます。
  - 1 行目 - 第 1 オペランドのシフト（-31 ～ 31）を指定します。
  - 2 行目 - 第 2 オペランドのシフト（-31 ～ 31）を規定します。

除算 :

Ofervlow Handling：

- *Reduce Resolution* - 両方の入力を必ず右シフトして、オーバーフローを防ぎます。
- *Keep Resolution And Limit* - シフトは行わず、算術演算サービスルーチンを使用できる場合はそれを使用します。
- *Keep Resolution And Don't Limit* - シフトを行わないで通常の算術演算を行うので、オーバーフローが起こる可能性があります。

分母のインターバルにゼロが含まれているのにかかわらずゼロ除算をチェックするコードが必要でない場合は、**Allow zero in phys. interval** オプションがオンになっています。この場合は、ユーザーの責任範囲において、分母がゼロにならないようにする注意が必要です。

注記

このオプションを不用意に使用すると、ECU の重大な障害を招く可能性があります。

Pre-Shift（演算前のシフト処理）：

数値の精度を上げるために、分子を自動的に左シフトして最大化するかどうかを示します。

- *Auto* オプションがオンになっていると、分子を自動的に左シフトします。
- *Auto* オプションがオフになっていると、“Pre Shift”フィールドの設定に従って以下のようなプレシフトが行われます。
  - 1 行目 - 第 1 オペランドのシフト（-31 ～ 31）を指定します。
  - 2 行目 - 第 2 オペランドのシフト（-31 ～ 31）を規定します。

マルチプレクサ、最大、最小：

Select Quantization（量子化の選択）：

- *Auto* - 最適化の方針に基づく量子化を行います。
- *1* - 第 1 の入力の量子化と型を適用します。
- *2* - 第 2 の入力の量子化と型を適用します。

これらの演算では数値計算は行われないので、オーバーフロー対策は必要ありません。

演算子インプリメンテーションの自動変換

旧バージョンのモデルに存在する演算子インプリメンテーションは、ASCET V5.1 で新しく導入された「インプリメンテーションキャスト」（『ASCET リファレンスガイド』の「インプリメンテーションキャスト」の項を参照してください）に自動的に変換することができます。この自動変換はデータベース全体を対象に行われ、コンポーネント単位では行えません。

自動変換の規則： 演算子インプリメンテーションの自動変換を行うには、以下の条件が満たされている必要があります。

- 演算子インプリメンテーションに、Auto 以外の量子化オプションが設定されていないこと（加算、除算、MIN、MAX、MUX の場合、495 ページ参照）

注記

MIN、MAX、MUX について自動変換を行うために必要な条件は、上記の 1 点のみです。その他の条件は、加減乗除演算にのみ適用されます。

- 演算子の出力が接続されていること
- 演算子の出力が基本エレメントにのみ接続されていること

  コンポーネント、演算子、階層ブロックなどに接続されている場合は自動変換できません。

- インプリメンテーションキャストが演算子の出力に接続されている場合、そのインプリメンテーションキャストの **Use Implementation Type** オプションの隣のコンボボックスで <No implementation> 以外の型が選択されていること
- 演算子インプリメンテーションに特別なプレシフトが指定されていないこと（乗算と除算のみ、496 ページ参照）
- 除算演算子についてその演算子インプリメンテーションの **Allow zero in phys. interval** オプションがオンになっている場合、分母の入力には以下の条件が満たされている必要があります。
  - 分母の入力が接続されていること
  - 分母の入力が基本エレメントにのみ接続されていること

    コンポーネント、演算子、階層ブロックなどに接続されている場合は自動変換できません。
  - インプリメンテーションキャストが分母の入力に接続されている場合、このインプリメンテーションキャストの **Use Implementation Type** オプションの隣のコンボボックスで <No implementation> 以外の型が選択されていること

コンポーネント内に、上記の条件が完全に満たされていないインプリメンテーションが含まれている場合（4.12.1 項を参照してください）、それらについては個々にマニュアル操作で変換する必要があります。

演算子のインプリメンテーションをインプリメンテーションキャストに変換する：

- コンポーネントマネージャで、**Tools → Database → Convert → Operator Implementations to Impl. Casts** を選択します。

  データベース内に含まれるすべての演算子インプリメンテーションが、上記の条件のもとにインプリメンテーションキャストに変換されます。

演算子（MIN、MAX、MUX 以外）は、以下の規則に従って自動変換されます。

- 演算子の出力の各接続線ごとにインプリメンテーションキャストが作成されます。
- 演算子インプリメンテーションの**Allow zero in phys. interval**オプションがオンになっている除算演算子の場合、分母の入力への接続線上にインプリメンテーションキャストが作成されます。
- インプリメンテーションが定義された演算子の出力ごとに作成されるインプリメンテーションキャストには、その演算子の後方に続くエレメントのインプリメンテーション情報が適用されます。

- 除算演算子の分母の入力に作成されるインプリメンテーションキャストには、その演算子の前方のエレメントのインプリメンテーション情報（モデルデータ型以外）が適用されます。

注記

コンポーネントのインプリメンテーション（4.12.1 項を参照してください）のうち、演算子にインプリメンテーションが定義されていないものについては、新しく作成されたインプリメンテーションキャストには <No implementation> が選択されています。

- オーバフロー対策のオプションについては、以下の表のとおりに変換されます。

インプリメンテーションキャスト / 演算子インプリメンテーション	Limit to maximum bit length	Reduce Resolution	Keep Resolution
Reduce resolution	○	○	
Keep resolution and limit	○		○
Keep resolution and don't limit			○

  各行に示された演算子インプリメンテーションの設定は、演算子インプリメンテーションキャストの上記のような設定に変換されます。

- 演算子インプリメンテーションは削除されます。

自動変換時において、MIN、MAX、MUX 演算子については、その演算子インプリメンテーションが削除されるだけで、それに代わるインプリメンテーションキャストは作成されません。

上記の規則に従い、場合によっては、演算子の出力に接続されたエレメントと同じ設定を持つインプリメンテーションキャストが作成されます。この場合、インプリメンテーションキャストは挿入されません。

自動変換できない演算子インプリメンテーションについては、以下のような処理が行われます。

- 演算子の出力の接続線（コンポーネントや演算子などに接続されたものも含みます）ごとにインプリメンテーションキャストが作成され、コンポーネントのすべてのインプリメンテーションに含まれるインプリメンテーションキャストにおいて、<No implementation> が選択されます。これらのインプリメンテーションキャストには、マニュアル操作で個別に適切な情報を定義します。

  ただしいずれかの接続線上にすでにこのようなインプリメンテーションキャストが存在している場合は、この接続線上にはこれ以上インプリメンテーションキャストは追加されません。

- 除算演算子の演算子インプリメンテーションにおいて**Allow zero in phys. interval** オプションがオンになっている場合、分母の入力の接続線上に、<No implementation> が選択されたインプリメンテーションキャストが作成されます。

  すでにこのようなインプリメンテーションキャストが存在している場合は、これ以上インプリメンテーションキャストは追加されません。

- 演算子インプリメンテーションは、削除されずにそのまま残ります。

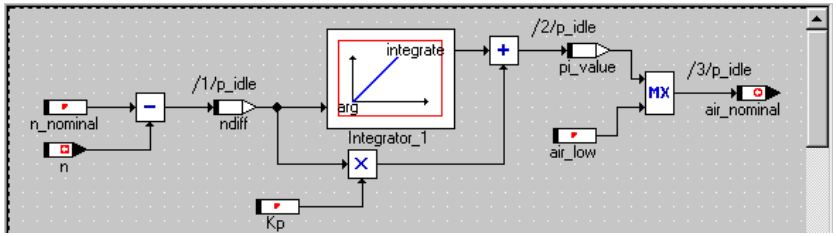
自動変換処理において、変換できない演算子インプリメンテーションがあった場合は、以下のメッセージが表示されます。

```
Not all operator implementations could be replaced
automatically. Please do the conversion manually.
```

このメッセージが表示されたら **OK** をクリックしてダイアログボックスを閉じます。すると“Operator Implementations”ダイアログボックスが開き（493 ページ参照）、まだ残っている演算子インプリメンテーションの一覧が表示されます。これらはマニュアル操作で変換してください。

4.13 コンポーネントのレイアウトの編集

コンポーネントをブロックダイアグラムの内部に含めると、それはブロックダイアグラムエディタまたはプロジェクトエディタの描画領域にグラフィックブロックとして表示され、そのインターフェースエレメントは入力ピンおよび出力ピンとして表示されます。



このようなグラフィックブロックの外観は、レイアウトエディタで変更することができます。アイコンの追加、入力ピンや出力ピンの表示位置の変更、ブロックのサイズや色を変えることができます。コンポーネントのタイプによっては、そのパブリックインターフェースを規定するメソッドまたはプロセスを有効にしたり無効にしたりすることもできます。

レイアウトエディタは、任意のコンポーネントエディタ、またはコンポーネントマネージャから起動することができます。

レイアウトエディタを開く：

- レイアウトを変更したいコンポーネントのコンポーネントエディタを開きます。

- **Component → Edit Layout** を選択します。

または

- ブラウザビューで、“Layout”タブを選択します。
- グラフィックブロックをダブルクリックします。

  選択されているコンポーネント用のレイアウトエディタが開きます。

あるコンポーネントが別のコンポーネントやプロジェクトに含まれる場合、含まれるコンポーネントのレイアウトをブロックダイアグラムエディタやプロジェクトエディタで編集できるようにすることができます（フレキシブルレイアウト機能）。この機能は、ASCET オプションウィンドウの **Activate flexible layout** オプション（54 ページの「ブロックダイアグラムエディタのオプション」を参照してください）で、共通オプションとして設定でき、また、これとは別にクラスやモジュール単位でこの機能の有効／無効を設定することもできます。デフォルトでは、フレキシブルレイアウト機能はすべて無効になっています。

4.13.1 クラスのレイアウトの編集

レイアウトエディタでは、ブロックの外観を変更したり、個々のメソッドを有効／無効にすることができます。レイアウトエディタにおいてあるメソッドを無効にすると、別のコンポーネントがそのクラスを参照する際にそのメソッドを使用することができなくなります。

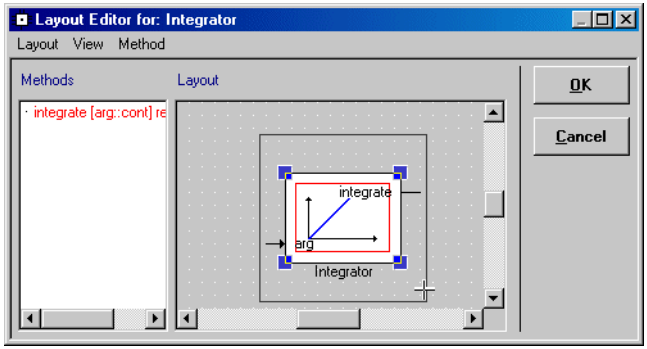
ダイアグラムブロックを修正する：

- レイアウトエディタで、描画領域内のブロックをクリックします。

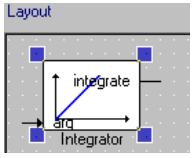
  アイコンが割り当てられたレイアウトを編集する際、ブロックとアイコンのサイズが同じ場合は、ブロックの代わりにアイコンが選択されてしまうことがありますが、その場合は、レイアウトエディタで編集することができません。
  このような場合、ブロックの境界線上をクリックしてください。

  クリックされたブロックが選択され、その四隅にサイズ変更ハンドルが表示されます。

- ハンドルをドラッグしてブロックのサイズを変更します。



ブロックのサイズをアイコンのサイズよりも小さくしてしまうと、アイコンしか選択できなくなってしまうため、ブロックのサイズを変更することができなってしまいます。



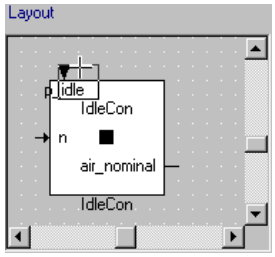
ブロックのサイズを変更するには、そのコンポーネントをブロックダイアグラム内に追加して描画領域に配置し、226 ページの「内包されるコンポーネントのレイアウト」に説明されている方法でレイアウトを変更してから、231 ページの方法で変更されたレイアウトをデフォルトレイアウトとして設定します。

デフォルトでは、入力ポートはブロックの左側、出力ポートは右側、またメソッドは上部に配置されていますが、これらの位置は以下のようにしてすべて変更できます。

ポートを移動する：

- 入力と出力ポートをブロックの辺に沿って任意の位置にドラッグします。

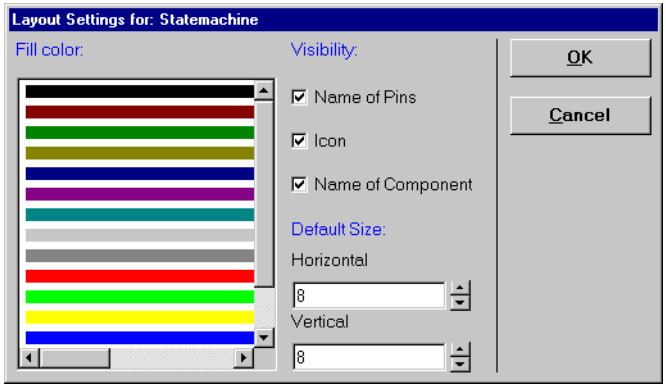
- メソッドをブロックの辺に沿って任意の位置にドラッグします。



すでに別のポートが置かれている位置に別のポートを配置することはできません。

ブロックの属性を修正する：

- レイアウトエディタの **Layout → Modify Attributes** を選択して、“Layout Settings”ダイアログボックスを開きます。



- “Fill color”ボックスから塗りつぶし用の色を選択します。
- “Horizontal”および“Vertical”ボックスから数値を選択して、ブロックのサイズを調整します。これにより、サイズ変更ハンドルをドラッグするのと同じことが行えます。
- 以下の表示オプションの表示／非表示を、それぞれのチェックボックスで指定します。
  - **Name of Pins**：入力ピンおよび出力ピンに、インターフェースエレメント名のラベルが付きます。

- **Icon**：コンポーネントが参照するアイコンが、ブロックの中央に表示されます。
- **Name of Component**：コンポーネントマネージャにおいてコンポーネントに付けられた名前です。インスタンス名、つまり、他のコンポーネントから参照されるときにそのコンポーネントの個々のインスタンスに付けられる名前を、描画領域に表示したり非表示にすることができます。

- **OK** をクリックして、現在のブロックの属性を修正します。

データベース内に新規に作成されるすべてのブロックに適用されるデフォルト属性を設定することもできます。

新規ブロック用のデフォルト属性を変更する：

- **Layout → Edit Default Attributes** を選択します。

  “Layout Settings”ダイアログボックスが表示されます。前述と同じ設定を行えますが、ここで設定された属性は、既存のブロックには反映されず、新規に作成されるすべてのコンポーネントに適用されるデフォルト属性として使用されます。

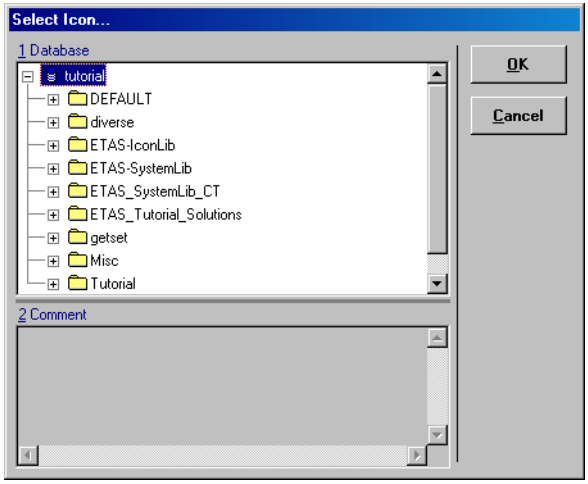
- デフォルト属性を既存のグラフィックブロックに適用するには、**Layout → Set Attributes to Default** を選択します。

コンポーネントにアイコンを割り当てて、そのアイコンをコンポーネントのグラフィックブロック内に表示することができます。アイコンはデータベース内に格納され、コンポーネントマネージャからアクセスすることができます。

アイコンをクラスに割り当てる：

- レイアウトエディタで **Layout** → **Select Icon** を選択します。

  “Select Icon...” ダイアログボックスが開きます。



- “1 Database” リストから、ブロックに割り当てたいアイコンを選択します。
- **OK** をクリックします。

  アイコンが、レイアウトエディタに表示されているコンポーネントに割り当てられます。

アイコンの作成と編集については、544 ページの「アイコンエディタ」の項を参照してください。

レイアウトエディタには、表示設定に関するいくつかの機能があります。

レイアウトエディタの表示内容を変更する：

- 再描画を行うには、**View** → **Redraw** を選択します。
- “Grid Option” ダイアログボックスを開くには、**View** → **Grid** を選択します。

  このダイアログボックスについては、262 ページの「ダイアグラムの表示と印刷」で説明されています。
- レイアウトを印刷するには、**View** → **Print** を選択します。
- レイアウトエディタを終了するには、**OK** をクリックします。

メソッドおよび変数に対するパブリックアクセスを有効／無効にして、パブリックインターフェースを変更することができます。

メソッドを有効／無効にする：

- レイアウトエディタの“Methods”リストから、有効または無効にしたいメソッドを選択します。
- 選択されたメソッドへのパブリックアクセスを無効にするには、**Method → Disable** を選択します。
- 選択されたメソッドへのパブリックアクセスを有効にするには、**Method → Disable** または **Enable** を選択します。

  デフォルトでは、すべてのメソッドへのパブリックアクセスは有効になっていて、レイアウトエディタの“Methods”リストに赤色で表示されています。無効になったメソッドはレイアウトペインには表示されず、“Methods”リストには黒色で表示されます。

4.13.2 その他のコンポーネントのレイアウトの編集

- *モジュール*：モジュールのレイアウトの編集の方法はクラスの場合と同じですが、メソッドの代わりにプロセスを、またエレメントの代わりにメッセージを有効／無効にする点が異なります。
- *連続系ブロック（CT ブロック）*：レイアウトエディタでメソッドやエレメントを有効／無効にすることはできません。ブロックの外観は、クラスと同様に修正できます。
- *ステートマシンと条件テーブル*：クラスの場合と同じです。
- *論理テーブル*：クラスの場合と同じですが、メソッドを無効にすることはできません。

4.14 算術演算サービス

ASCET では、*「算術演算サービス」*と呼ばれる機能を使用して、加算などの基本的な演算に対して、最適化を行ったり、数値範囲制限やオーバーフロー処理などの特別な処理を適用することができます。

算術演算サービスは C 関数で作成されています。算術演算サービスが有効になっていると、通常の演算処理（以下、「標準演算」と記します）の代わりに、それに相当する関数呼び出しを行うコードが生成されます。たとえば、

```
c = a + b;
```

上のような通常の加算の代わりに、下のような関数呼び出し文が生成されます。

```
c = add(a, b);
```

また、以下のような標準演算については、ユーザーが定義したサービスルーチンの呼び出しを行うコードを生成することもできます。

- 加算、減算、乗算、除算
- 否定と絶対値
- 補間点の検索と補間
- 複合演算（乗算の直後に除算を続けて行うなど）

算術演算サービスは、すべての物理記述形式（ESDL、ブロックダイアグラム）において、すべての複合エレメント（クラス、モジュール、有限ステートマシン）について使用できます。すべてのタイプのターゲット（マイクロコントローラターゲット、ラピッドプロトタイピング用実験ターゲット、PC）でのオフライン実験またはオンライン実験において、任意の算術演算機構を用いたコードを実行できます（405 ページを参照してください）。

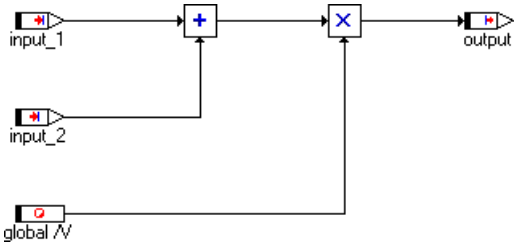
ASCET は、サービスが利用可能であるかどうか、および可能な場合はそのタイプについての情報を `services.ini` ファイルから見つけます。このファイルは、各ターゲット専用のターゲットディレクトリ（`Ascet5.2¥target¥<target>`）に格納されています。これらのファイルは、Windows の `*.ini` ファイルの形式で書かれているので、一般的なテキストエディタや ASCET に添付されている「AS エディタ」を使用して、作成したり編集したりすることができます（4.14.5 項を参照してください）が、ASCET から直接編集することはできません。

また算術演算サービスに使用する各関数は、ユーザーの開発環境にあわせて最適な形式（ライブラリ、オブジェクトコード、C コード、または ASCET の C コードエディタで作成された C コード）で作成することができます。

4.14.1 算術演算サービスの機能

ASCET では、クラスやモジュールなどを C コードで記述する以外に図（ブロックダイアグラム）や抽象言語（ESDL）で記述することもできます。これらの記述形式から任意のマイクロコントローラ用または実験用の実行コードを生成する際に

は、ASCET の「コードジェネレータ」によって、記述内容が C コードに変換されます。下の例は、コードジェネレータがコード生成の過程で算術演算サービスを必要とする理由と、その使用方法を示しています。.



上のブロックダイアグラムには、加算と乗算の 2 つの演算が定義されています。この演算を C コードに変換する方法としては、次の 2 通りが考えられます。

1. C 言語の標準演算である + と * を使用する
2. 加算と乗算の関数を使用する

乗算の標準演算は以下のようになります。

```
output := input_1 + input_2;
```

これに 2. の方法を適用すると、以下のようになります。

```
uint8 add(uint8 summand_1, uint8 summand_2)
{
   return summand_1 + summand_2
}
...
output := add(input_1, input_2);
```

ASCET のデフォルト設定においては、単純な加算には標準の + 演算子が使用されますが、この + 演算子に対応する算術演算サービスをユーザー定義することにより、第 2 の方法を適用することができます。

このためには、以下の手順を実行する必要があります。

1. まず、`uint8 add (uint8 summand_1, uint8 summand_2)` という関数のコードを作成します。
   この関数は、通常は ASCET の外部に、たとえば関数ライブラリのような形で作成しますが、必要に応じてモデル内に作成することもできます。
2. 次に、関数呼び出し文を定義します。

上の例の単純な加算の例の場合、関数呼び出し文のコードは次のようになります。

```
+|u8|u8|u8 = add(%i1%, %i2%)
```

この文の構文については、以降の項で説明します。

この文を元にして、コードジェネレータは、「2 つの uint8 の値を加算して 1 つの uint8 の結果を得る」というすべての加算について、標準の + 演算を用いる代わりに `add (%i1%, %i2%)` という関数呼び出し文を生成します。この際、仮引数 `%i1%` および `%i2%` は、各演算の実引数（たとえば、`input_1` と `input_2`）に置き換わります。

4.14.2 算術演算サービスの定義

算術演算サービスは、「関数呼び出し文の定義」と「関数自体のコード」という 2 つの部分で構成されます。関数は C コードで作成されたもので、かつ ASCET 環境に組み込まれていなければならない（4.3.3 項を参照してください）ということ以外、厳密な規則に従う必要はまったくありませんが、これらの関数呼び出し文の定義は、ASCET が理解できる形式になっていなければなりません。算術演算サービスの関数呼び出し文は、以下に示す厳密な構文に従って記述します。

```
operation|opType1|opType2|opType3|resType=function
```

上記の関数呼び出し文文は、以下の 2 つの部分から構成されます。

- 関数キー
  `- operation|opType1|opType2|opType3|resType`
- 関数呼び出し（C 言語の構文）
  `- function`

「関数キー」は、標準演算を算術演算サービスに置き換える際の変数の割り当てを定義し、「関数呼び出し」は、コードジェネレータにより生成される関数の呼び出し文を定義するものです。

各標準演算は、種類ごとに 1 つのキーにより定義され、1 つのキーに定義できるサービスは 1 つだけです。

関数キー

関数キーは、必ず以下の構文に従って記述します。

```
operation|opType1|opType2|opType3|resType
```

キーの各要素は以下の意味を持ちます。

- `operation`： 演算を識別する文字列
- `opType`： オペランドの型を定義する文字列
- `resType`：演算結果の型を定義する文字列

定義済みの算術演算サービス

ASCET には、ASCET 内で記述されるすべての計算に対応きる算術演算サービスが用意されていて、これらを具体的なニーズに合わせてカスタマイズすることも可能です。下の表は、ASCET で使用できるすべての算術演算サービス用の関数キーと、それに含まれる各引数の数と型（0 ＝演算、1, 2, 3 ＝オペランド 1, 2, 3、R

＝結果）、および関数の機能をまとめたものです。さらに、AS エディタで自動生成される標準的な関数呼び出し文（4.14.5 項を参照してください）も記載されています。

関数キー	標準的な関数呼び出し文	引数	機能
abs|*|*	abs_%t1%_%tr% (%i1%)	O|1|R	絶対値の算出
neg|*|*	neg_%t1%_%tr% (%i1%)	O|1|R	値の否定
+|*|*|*	add_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値の加算
+l|*|*|*	addl_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値の加算して結果を飽和[a]
-|*|*|*	sub_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値の減算
-l|*|*|*	subl_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	値の減算して結果を飽和 a
||*|*	mul_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値の乗算
l||*|*	mull_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	値を乗算して結果を飽和 a
/|*|*|*	div_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値の除算
/l|*|*|*	divl_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値の除算して結果を飽和 a
%|*|*|*	mod_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値のモジュロ計算
%l|*|*|*	modl_%t1%%t2%_%tr% (%i1%, %i2%)	O|1|2|R	2 つの値のモジュロ計算、結果を飽和 a
>||*|*|*	mul_r_%t1%%t2%_%tr% (%i1%, %i2%, %i3%)	O|1|2|3|R	2 つの値を乗算して右シフト
>l||*|*|*	mul_rl_%t1%%t2%_%tr% (%i1%, %i2%, %i3%)	O|1|2|3|R	2 つの値を乗算して右シフトし、結果の範囲を制限
/>|*|*|*|*	div_r_%t1%%t2%_%tr% (%i1%, %i2%, %i3%)	O|1|2|3|R	2 つの値の除算して右シフト
/>l|*|*|*|*	div_rl_%t1%%t2%_%tr% (%i1%, %i2%, %i3%)	O|1|2|3|R	2 つの値の除算して右シフト、結果を飽和 a
/||*|*|*	muldiv_%t1%%t2%%t3%_%tr%(%i1%, %i2%, %i3%)	O|1|2|3|R	2 つの値を乗算してから除算

関数キー	標準的な関数呼び出し文	引数	機能
/l||*|*|	muldivl_%t1%%t2%%t3%_%tr%(%i1%, %i2%, %i3%)	O|1|2|3|R	2 つの値の乗算してから除算、結果を飽和 a
getAt1|*|*	CharTable1_getAt_%t1%_%tr%(%ct%, %x%)	O|1|R	特性カーブへのアクセス
getAt1R|*|*	CharTable1_getAtR_%t1%_%tr%(%ct%, %x%)	O|1|R	特性カーブへの丸めアクセス
getAt2|*|*|*	CharTable2_getAt_%t1%&t2%_%tr%(%ct%, %x%, %y%)	O|1|2|R	特性マップへのアクセス
getAt2R|*|*|*	CharTable2_getAtR_%t1%&t2%_%tr%(%ct%, %x%, %y%)	O|1|2|R	特性マップへの丸めアクセス
getAtFixed1|*|*	CharTable1_getAtFixed_%t1%_%tr%(%ct%, %x%)	O|1|R	固定カーブへのアクセス
getAtFixed1R|*|*	CharTable1_getAtFixedR_%t1%_%tr%(%ct%, %x%)	O|1|R	固定カーブへの丸めアクセス
getAtFixed2|*|*|*	CharTable2_getAtFixed_%t1%&t2%_%tr%(%ct%, %x%, %y%)	O|1|2|R	固定マップへのアクセス
getAtFixed2R|*|*|*	CharTable2_getAtFixedR_%t1%&t2%_%tr%(%ct%, %x%, %y%)	O|1|R	固定マップへの丸めアクセス
interpolGroup1|*|*	GroupTable1_getAt_%t1%_%tr%(%ct%, %x%)	O|1|R	グループカーブの補間
interpolGroup1R|*|*	GroupTable1_getAtR_%t1%_%tr%(%ct%, %x%)	O|1|2|R	グループカーブの補間、丸め
interpolGroup2|*|*|*	GroupTable2_getAt_%t1%&t2%_%tr%(%ct%, %x%, %y%)	O|1|2|R	グループマップへのアクセス、補間
interpolGroup2R|*|*|*	GroupTable2_getAtR_%t1%&t2%_%tr%(%ct%, %x%, %y%)	O|1|2|R	グループマップの補間、丸め
searchDistrib|*|*	Distribution_search_%t1%_%i1%)	O|1	ディストリビューション（共通座標ポイント）の検索
getHighPart	_(%i1%)	O	値の最上位ビットを返す

a. 飽和は、結果の値に対してリミッタを設け、その型で使用できる範囲がすべて使用されるようにすることです。

表 4-1　ASCET でサポートされている算術演算サービス

注記

表 4-1 に示した引数の数と順序がデフォルト状態です。基本的には、引数の種類、数および順序については、C 言語の構文に従っている限り制限がないので、必要に応じてさらに引数を追加することができます（535 ページの「エントリを定義する：」を参照してください）。

許容される型

文字列	型
u8	8 ビットの符号なし整数
u16	16 ビットの符号なし整数
u32	32 ビットの符号なし整数
s8	8 ビットの符号付き整数
s16	16 ビットの符号付き整数
s32	32 ビットの符号付き整数
*	上記のすべての型が許容されます

表 4-2　オペランドおよび結果として許容される型

結果用データの型の有無とオペランドの数は、選択された演算により異なります。たとえば、加算にはオペランドは 2 つあり、否定には 1 つしかありませんが、乗算を行ってから右シフトを行う演算の場合は、3 つ必要です。

関数の宣言

関数の宣言は、標準演算の代わりに使用される関数呼び出しの内容を C 言語の構文に従って記述したものです。たとえば、ある関数が以下のような C の文で定義されている場合、

```
int Add(int operand1, int operand2){...},
```

その関数に対応する関数呼び出しは以下のようになります。

```
Add(operand1, operand2);
```

`services.ini` ファイル内の関数宣言において、`operand1` と `operand2` に対応する具体的な文字列はわからないので、それらの文字列の代わりに仮引数が用いられます。コードジェネレータがコード生成プロセスでこれらのプレースホルダを認識し、対応する文字列に置き換えます。`services.ini` ファイル内では、上の関数呼び出しは以下のように記述します。

```
Add(%i1%, %i2%);
```

表 4-3、表 4-4、および表 4-5 に、コードジェネレータがサポートするすべての引数を示します。

引数	意味
%i1%	第 1 オペランド
%i2%	第 2 オペランド
%i3%	第 3 オペランド
%t1%	第 1 オペランドの short 型（例、u8）
%t2%	第 2 オペランドの short 型（例、u8）
%t3%	第 3 オペランドの short 型（例、u8）
%ft1%	第 1 オペランドの long 型（例、uint8）
%ft2%	第 2 オペランドの long 型（例、uint8）
%ft3%	第 3 オペランドの long 型（例、uint8）
%tr%	結果の short 型（例、u8）
%ftr%	結果の long 型（例、uint8）
%c1%	自動的に決定される第 1 オペランドの型のキャスト
%c2%	自動的に決定される第 2 オペランドの型のキャスト
%c3%	自動的に決定される第 3 オペランドの型のキャスト
%op%	演算の名前
%il%	入力のビット長

表 4-3 Mul と Div（シフトと結果の範囲制限のあり／なし）、Add、Sub、Muldiv（結果の範囲制限のあり／なし）、Mod、Neg、Abs、getHighPart 演算の仮引数

引数	意味
%ct%	特性カーブまたはマップ
%x%	x 補間ポイント
%y%	y 補間ポイント（特性マップの場合のみ）
%tx%	x 値の short 型（例、u8）
%ty%	y 値の short 型（例、u8）（特性マップの場合のみ）
%tv%	結果の short 型（例、u8）
%ftx%	x 値の long 型（例、uint8）（固定カーブ／マップの場合のみ）
%fty%	y 値の long 型（例、uint8）（固定カーブ／マップの場合のみ）
%ftv%	結果の long 型（例、uint8）（固定カーブ／マップの場合のみ）
%fmst%	テーブルの最大サイズ（maxSize）により決まる long 型（固定カーブ／マップの場合のみ）
%xd%	x 座標ポイント（グループカーブ／マップの場合のみ）
%yd%	y 座標ポイント（グループマップの場合のみ）
%xlen%	x 座標ポイントの数
%ylen%	y 座標ポイントの数（特性マップの場合のみ）
%xdind%	下側のブレイクポイントのインデックス（x 次元、ディストリビューションとグループカーブ／マップの場合のみ）
%xdoff%	下側のブレイクポイントと上側のブレイクポイントの間のオフセット（x 次元、ディストリビューションとグループカーブ／マップの場合のみ）
%xddis%	下側のブレイクポイントと補間ポイントとの距離（x 次元、ディストリビューションとグループカーブ／マップの場合のみ）
%ydind%	下側のブレイクポイントのインデックス（y 次元、グループマップの場合のみ）
%ydoff%	下側のブレイクポイントと上側のブレイクポイントの間のオフセット（y 次元、グループマップの場合のみ）
%yddis%	下側のブレイクポイントと補間点との距離（y 次元、固定以外の特性マップの場合）

表 4-4 Getat1、Getat2、Getatfixed1、Getatfixed2、Interpolgroup1、Interpolgroup2 演算の仮引数

引数	意味
%i1%	ディストリビューションの入力値
%t1%	ディストリビューション入力値の short 型
%ft1%	ディストリビューション入力値の long 型
%c1%	自動的に決定される、ディストリビューションの入力値の型のキャスト
%d%	ディストリビューション
%xdind%	x ディストリビューション内の値のインデックス
%xdoff%	x ディストリビューション内の値のオフセット
%xddis%	x ディストリビューション内の 2 つの値の間の距離

表 4-5 Searchdistrib 演算の仮引数

各引数は、以下のような C 言語の関数呼び出しの構文に使用されます。

```
(%ftr%)functionname(%i1%, %i2%, %i3%);
```

さらに、コードジェネレータは、以下の構文から実際の関数名を生成します。

```
%op%_%t1%%t2%%t3%_%tr%
```

関数名は任意の方法で記述することができますが、通常は上記の構文を用いるようにしてください。

4.14.3 算術演算サービスの作成と保存

定義された算術演算サービスの保守は、ASCET 外部の services.ini ファイル内で行われます。このファイルは、Windows の *.ini ファイルに準じたフォーマットで記述され、各ターゲットごとに、専用のディレクトリにそれぞれ格納されます。

1 つの services.ini ファイル内に算術演算サービスのセットをいくつか定義して管理することができるため、ターゲット固有の様々な要件に合わせて ASCET コードジェネレータの設定を手早く変更し、定義されたる算術演算サービスルーチンを柔軟に使用することができます。ASCET の起動時に、現在のターゲット用のすべての情報がこのファイルからロードされるので、コードジェネレータ実行時に、定義されているセットの中から任意のものを適用することができます。

services.ini ファイル内のエントリは、以下の構文（BNF フォーマットで記述されています）に従って記述してください。

```
SERVICE-FILE ::= [ENTRY]+
ENTRY ::= [FUNCTION] | [COMMENT] | [SET]
COMMENT ::= ";" Σ*
SET ::= "["Σ+"]"
FUNCTION ::= [OPERATION]("|"[OPERAND])+ "=" Σ*
```

```
OPERATION ::= "abs" | "neg" | "+" | "-" | "*" | "/"
   | "%" | "+l" | "-l" | "*l" | "/l" | "*>" | "/>"
   | "*>l" | "/>l" | "*/" | "*/l" | "getAt1"
   | "getAt1R" | "getAt2" | "getAt2R" | "getAtFixed1"
   | "getAtFixed1R" | "getAtFixed2" | "getAtFixed2R"
   | "interpolGroup" | "interpolGroup1"
   | "interpolGroup1R" | "interpolGroup2"
   | "interpolGroup2R" | "searchDistrib"
   | "searchDistribR" | "getHighPart"
OPERAND ::= "*" | "u8" | "u16" | "u32" | "s8" | "s16"
   | "s32"
```

services.ini ファイルの詳細と、このファイルの作成や編集方法については、4.14.5「算術演算サービス用のインターフェースエディタ」の項を参照してください。

4.14.4 ASCET で算術演算サービスを使用する

ASCET では、以下の環境において算術演算サービスを使用できます。

- 記述形式：ESDL およびブロックダイアグラムでの物理記述
- コンポーネント：クラス、モジュール、ステートマシン
- 実験：オフラインおよびオンライン
- ターゲット：マイクロコントローラターゲット、ラピッドプロトタイピング用実験ターゲット、PC

算術演算サービスは、コード生成処理に直接影響します。コードジェネレータは、演算の入力および出力として使用されているエレメントの型に応じて、使用する関数を選択します。

たとえば、ある加算の 2 つの入力が「8 ビットの符号なし整数」型（uint8）で、出力が「16 ビットの符号なし整数」型（uint16）である場合、コードジェネレータは、その組合せから +|u8|u8|u16 というキーを検索します。そしてそのキーが見つかると、そこに記述されている関数呼び出しを使用し、見つからなかった場合には、演算の型に応じて、標準演算を使用するか、またはエラーメッセージを出力します。

算術演算サービスセットの選択

算術演算サービスのセットの選択は、ASCET のプロジェクト単位で行われます。

算術演算サービスのセットを選択する：

- 算術演算サービスを適用したいプロジェクトを開きます。
- 

プロジェクトエディタの **Project Properties** ボタンをクリックします。

  “Project Properties”ダイアログボックスの“Build”ノードが開きます。

- “Code Generator”コンボボックスから `Implementation Experiment` または `Object Based Controller` という設定を選択します。

注記

`Physical Experiment` や `Quantized Physical Experiment` が選択されている場合は、“Fixed Point Codegeneration”オプションは設定**できません。**

- **Integer Arithmetic** ノードを開きます。

  “Arithmetic Service set”コンボボックスに、使用できるすべての算術演算サービスセットが表示されます。
- 使用するセットをコンボボックスから選択します。

注記

新しく作成されたプロジェクトについては、ASCET 起動時に読み込まれている `services.ini` ファイル内で最初に見つかったセットが自動的に適用されます。`services.ini` ファイルにセットが 1 つも定義されていない場合や、空のセットしか定義されていない場合、またはファイル内の先頭のセットが `[None]` というラベルの付いた空のセットである場合には、“Arithmetic Service set”コンボボックスで `<None>` が自動的に選択されています。

使用可能なセットが services.ini ファイル内に 2 つ以上定義されている場合は、コードジェネレータが実行されるたびに異なるセットを使用することができます。また、`<None>` を選択することによりセットを使用しないようにすることもでき、その場合は標準演算が使用されます。

ASCET は各プロジェクトについて、最後に選択されて使用されたセットに関する情報を保存します。そしてそのセットが、現在読み込まれている `services.ini` ファイルに格納されていない場合（たとえば、別の `services.ini` ファイルが使用されている他の PC 上のプロジェクトがロードされたような場合）には、以下のエラーメッセージが出力されます。

```
The selected set of arithmetic services <...> is not
available. Please check the services.ini file.
```

コードジェネレータがコードを生成するために必要な関数がセット内に定義されていない場合には、演算の型に応じて、エラーメッセージが出力されるか、または標準演算が使用されてワーニングが出力されます。これについての詳細は、518 ページの「発生する可能性のあるエラー」という項を参照してください。

演算内で定数が使用されている場合、算術演算サービスの使用時に特別な処理が行われます。ASCET で記述された定数とリテラルには特定の「型」がないため、算術演算サービスを使用するように設定されている場合、これらの値にはコード生成の過程で自動的に型が割り当てられます。この場合、コードジェネレータは下のリストの中から定数の値に適した型を検索し、最初に見つかったものを使用します。

```
sint8, uint8, sint16, uint16, sint32, uint32
```

この規則の例外となるのは、未定義型の定数と定義済み型の変数の両方を入力する 2 項演算子の場合です。

- 定数の値が正で変数の型が `unsigned` の場合には、定数には変数と同じ型が割り当てられます。この例外的な処理は、できるだけ多くの演算について入力の型を統一させることを目的として行われるものです。
- 定数の値が負の場合には、その定数には符号付き型を選択する必要があります。次の 2 通りの場合が考えられます。
  - 変数が `unsigned` 型の場合：この場合、型の統一は不可能です。
  - 変数が `signed` 型の場合：定数の値が変数の型に適合する場合には、変数の型がそのまま定数にも割り当てられます。そうでない場合には、型の統一は不可能です。

発生する可能性のあるエラー

算術演算サービスの使用が有効になっている場合には、コードジェネレータは、適切な算術演算サービスの存在の有無やその内容の妥当性に基いて処理を進めます。

ある演算のために算術演算サービスが必要であるにもかかわらず、選択されているセット内にそのサービスが見つからない場合には、以下のエラーメッセージがすべてのターゲットについて出力されます。

```
Arithmetic service <name> required but not defined
```

このメッセージが出力されるのは、純粋な算術演算（+、-、*、/、abs、neg）から得られる演算の場合だけです。

注記

旧バージョンの ASCET とは異なり、現バージョンでは、このような場合に標準演算は使用**されません**。

各演算に対応する算術演算サービスが正しく定義されるようにするために、ワイルドカードを使用して、いくつかの標準演算をまとめて定義することができます。この場合は、それぞれの算術演算（+、-、*、/、abs、neg）に使用する各演算サービス用エントリをあらかじめ設定しておく必要があります。

加算についてのエントリは以下のようになります。

```
+|*|*|*=(%i1% + %i2%)
```

乗算／除算の複合演算についてのエントリは以下のようになります。

```
*/|*|*|*|*=((%i1% * %i2%) / %i3%)
```

このように記述しておくと、コードジェネレータがワイルドカード（*）を理論上可能なあらゆる型の組み合わせに置き換えます。

注記

ある特別な型の組み合わせ（`+|u8|s8|s8 = add_u8s8_s8(%i1%, %i2%)` など）についての定義が現在のセット内に存在する場合は、そちらの内容が、ワイルドカードで定義されたものよりも優先されます。

算術演算（+、-、*、/、abs、neg）にインプリメンテーションキャストが直接接続されていて、それぞれに異なる量子化が定義されている場合は、ASCET は以下のワーニングを出力します。

```
Usage of arithmetic service <name> requires extra code
for requantization
```

注記

サービス `getAt..`、`searchDistrib`、`interpolGroup..`、`getHighPart` および `modulo` についての関数が現在の算術演算サービスセット内に定義されていない場合には、コードジェネレータはワーニングメッセージや通知メッセージを出力**しないで** ASCET の標準演算を使用します。

4.14.5 算術演算サービス用のインターフェースエディタ

ASCET には、算術演算サービスインターフェースエディタ（以下、「AS エディタ」と呼びます）が付属しています。これは、算術演算サービス用のインターフェース定義が格納される `services.ini` ファイル（以下、「AS ファイル」とも呼ばれます）の作成と編集に使用されるツールです。`services.ini` ファイルはターゲットごとに作成し、ASCET の各ターゲットのディレクトリに置いておく必要があります。

算術演算サービス用ファイル（「AS ファイル」）には、ASCET でコードを生成するために必要なインターフェース定義が格納されます。AS ファイルの構造は Windows の `*.ini` ファイルのフォーマットに準じています。このファイルには、セットの定義、インターフェース定義、およびコメントだけが格納されます。

セット定義は、角括弧（[ ]）で囲まれた文字列で、またインターフェース定義は、あらかじめ定義されている構文に準じた任意の文字列で記述します。コメントは、セミコロンで始まる任意の文字列です。

AS ファイルには 1 ～ 255 個の算術演算サービスセットを格納できます。1 つのセットについては、最初にセット名を宣言し、その直後から次のセットの始まりまたはファイルの終わりまでに、すべてのインターフェース定義を記述します。

AS ファイルは以下のようなレイアウトになります。

```
[Set name]
Function
```

```
Function
...
[Set name]
;Comment
Function
Function
...
```

AS ファイルレイアウトの構文についての詳しい説明は、4.14.3 項を参照してください。

AS エディタの機能

AS エディタでは、AS ファイルの作成、編集、保存などを行えます。

またさらに、AS エディタには以下ような機能も含まれています。

- 新しい空のセットを作成する
- セットを削除する
- セットを複製する
- セットの名前を変更する
- 新しい空のインターフェース定義を作成する
- インターフェース定義を削除する
- インターフェース定義のすべての部分を修正する
- インターフェース定義用の標準的な関数呼び出しを作成する
- 記述文の構文を検証する
- インターフェース定義をコメントにする
- コメントをインターフェース定義にする
- セット内のインターフェース定義を検索する

インターフェース定義の変更は、定義済みの値を選択リストから選択することによって行われます。AS ファイルの内容を AS エディタからマニュアル入力することはできません。

ユーザーが AS エディタで行うことができるのは、構文の正しいエントリを作成することだけです。このエディタでは、ASCET コードジェネレータの要件に完全に適合した AS ファイルしか作成できません。

AS エディタの起動

AS エディタは独立したツールです。ASCET とは無関係に起動することも、ASCET 内から起動することもできます。

AS エディタを ASCET から起動する：

- コンポーネントマネージャから、**Tools → Arithmetic Service Editor** を選択します。

  このメニュー項目を選択すると、その右側に、現在 PC に登録されているすべてのターゲットが表示されます。**>PC<** というエントリは常に選択可能です。
- **Tools → Arithmetic Service Editor → >PC<** のように操作して、AS ファイルを編集するターゲットを選択します。

  AS エディタが起動され、選択されたターゲット用の AS ファイルが開きます。

  選択されたターゲットのディレクトリに `services.ini` ファイルがない場合は、AS エディタは新しいファイルを自動的に作成します。

AS エディタを、ASCET とは無関係に Windows Explorer から起動することができます。

AS エディタを Windows Explorer から起動する：

- Windows Explorer を開きます。
- ASCET がインストールされているディレクトリから、`ETAS¥Ascet5.2¥ASEditor` サブディレクトリを開きます。
- そこから、`Editor.exe` アプリケーションを実行します。

  AS エディタが起動されます。

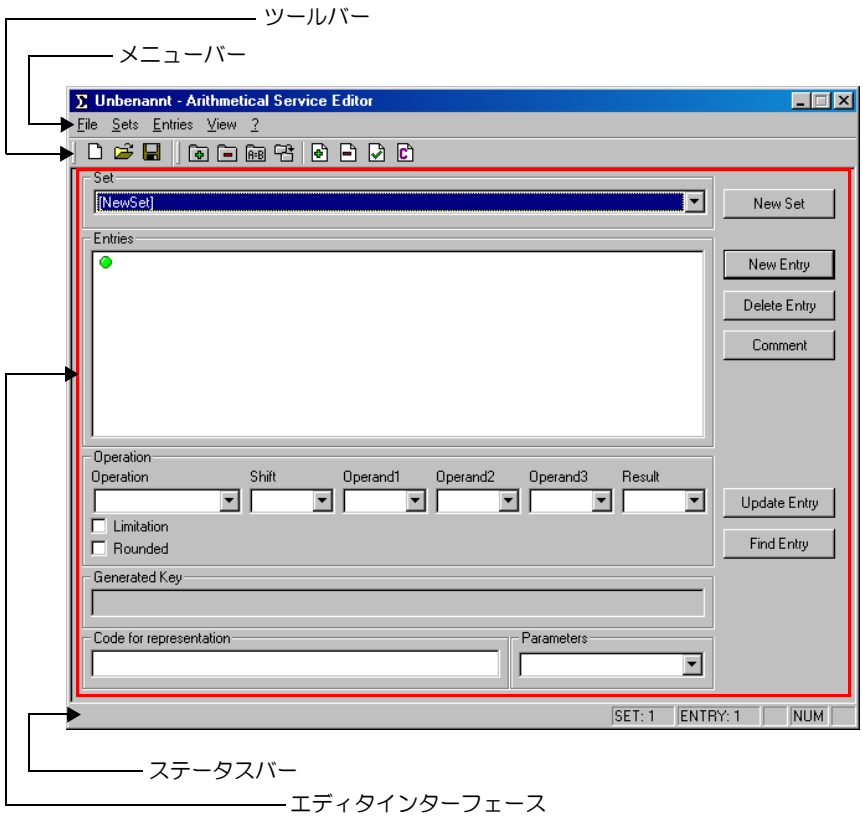
第 3 の方法として、AS エディタを Windows のスタートメニューから起動することもできます。

AS エディタを ASCET の Start メニューから起動する：

- Windows タスクバーの**スタート**をクリックします。
- **ETAS** プログラムグループから **ASCET5.2 → AS-Editor** エントリを選択します。

  AS エディタが起動されます。

AS エディタを起動すると、下の図に示されるようなウィンドウが開きます。



メニューコマンドの概要：

メニューバーから、ファイル関連操作（**File**）、算術演算サービスセットの編集（**Sets**）、エントリの編集（**Entries**）、AS エディタの画面表示の変更（**View**）、およびプログラムについての情報の呼び出し（**?**）に関するすべてのコマンドを実行できます。

- **File**
  - *New*（**<Ctrl>** + **<N>**）
    新しい空の AS ファイルを作成します。
  - *Open*（**<Ctrl>** + **<O>**）
    既存の AS ファイルを開きます。

- *Save*（**<Ctrl>** + **<S>**）

  現在の AS ファイルを保存します。
- *Save as*

  現在の AS ファイルを新しい名前で保存します。
- *1 .. 4*

  最近開かれた 4 つの AS ファイルの 1 つを開きます。
- *Exit*

  AS エディタを閉じます。

- **Sets**
  - *New*

    新しい空のセットを作成します。
  - *Copy*

    選択されているセットのコピーを作成します。
  - *Rename*

    選択されているセットの名前を変更します。
  - *Delete*

    選択されているセットを削除します。
- **Entries**
  - *New*

    新しい空のエントリを作成します。
  - *Update*

    選択されているエントリについての変更を適用します。
  - *Delete*

    選択されているエントリを削除します。
  - *Comment*

    選択されているエントリをコメントにします。
  - *Find*

    エントリを検索します。
- **View**
  - *Symbolbar*

    標準ツールバーの表示／非表示を切り替えます。
  - *Statusbar*

    ステータスバーの表示／非表示を切り替えます。

— *Toolbar*

専用ツールバーの表示／非表示を切り替えます。

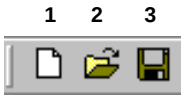
- **?**

— *About Editor*

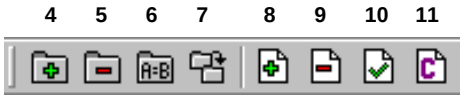
プログラムについての情報を表示します。

ツールバーの説明：

AS エディタには 2 つのツールバーがあります。1 つは標準の Windows ツールバーで、もう 1 つは AS エディタ固有のコマンドが含まれているツールバーです。



1. 新しい空の AS ファイルを作成します。
2. 既存の AS ファイルを開きます。
3. 現在の AS ファイルを保存します。



4. 新しい空のセットを作成します。
5. 選択されているセットを削除します。
6. 選択されているセットの名前を変更します。
7. 選択されているセットのコピーを作成します。
8. 新しい空のエントリを作成します。
9. 選択されているエントリを削除します。
10. 選択されているエントリについての変更を適用します。
11. 選択されているエントリをコメントにします。

ステータスバー：

ステータスバーには、ツールバーのボタンの機能説明、および現在選択されているセットとエントリのインデックスが表示されます。また、**<Num>**、**<Caps Lock>** および **<Scroll Lock>** キーのステータスも表示されます。

以下のように、**View** メニューを使用して AS エディタのビューを切り替えることができます。

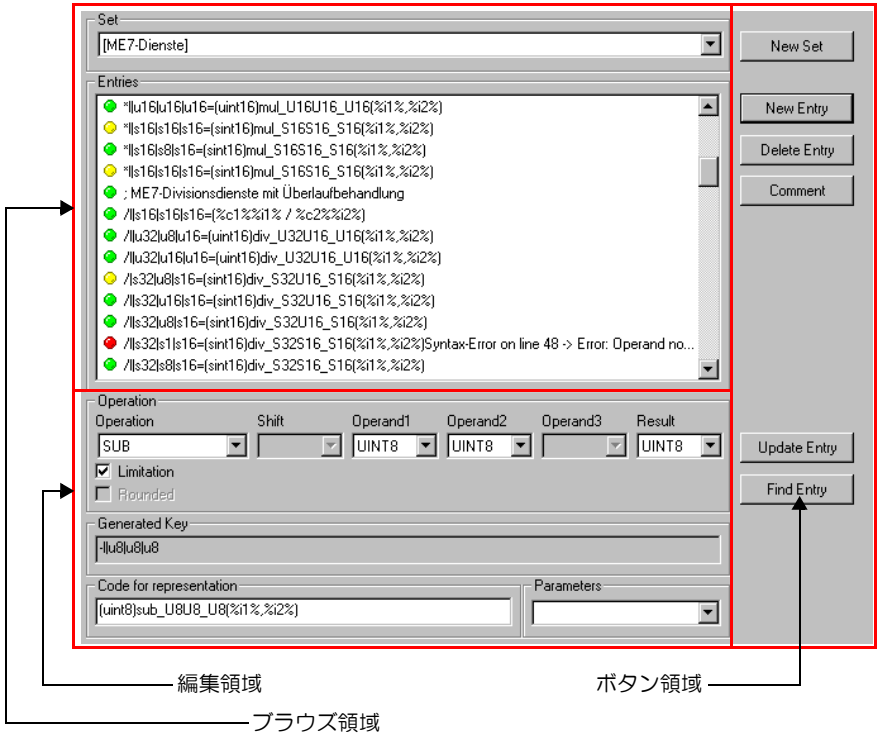
AS エディタのビューを変更する：

- メニューの **View** → **Symbolbar** をクリックすると、標準ツールバーの表示／非表示が切り替わります。

  ツールバーが表示されているときには、このメニュー項目にはチェックマークが付いています。
- メニューの **View** → **Toolbar** をクリックすると、このエディタのツールバーの表示／非表示が切り替わります。
- メニューの**View** → **Statusbar**をクリックすると、ステータスバーの表示／非表示が切り替わります。

エディタインターフェース

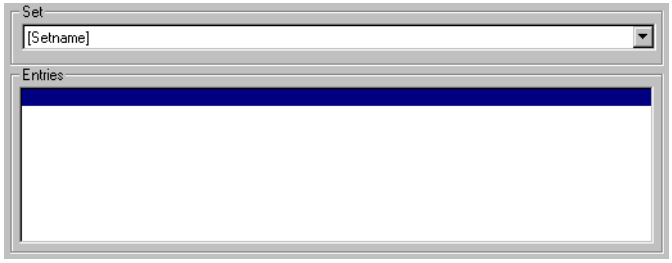
このエディタウィンドウは、3 つの領域に分かれています。



*ブラウズ領域*には、AS ファイルのすべての情報とエントリが表示されます。これらの内容の変更する際は*編集領域*で行い、*ボタン領域*にある各ボタンを使用してエントリの作成、更新、削除を行います。

ブラウズ領域 :

*ファイルが正しく読み込まれると、ブラウズ領域*に現在の AS ファイルに含まれる情報が表示されます。具体的には、“Set”コンボボックスにそのファイル内に定義されているすべてのセットの名前が表示され、そのコンボボックスで選択されているセットに属するすべての情報が“Entries”領域に表示されます。コンボボックスで別のセットを選択すると、それに応じて“Entries”フィールドの内容も更新されます。



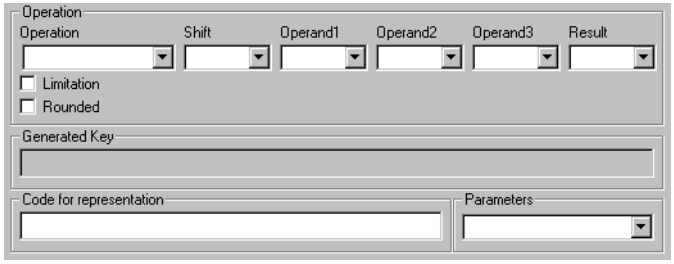
“Set”フィールドのコンボボックスに表示されるセット名は、エディタプログラムにより出力されるものであるため、ユーザーがこれを直接編集することはできません。

“Entries”フィールドはリストボックスで、リスト内のエントリ（行）を 1 つ選択して編集を行いますが、このリストボックスの内容もエディタプログラムにより出力されるものであるため、これを直接編集することはできません。

各行の先頭に表示される LED シンボルは、その行のステータスを示します。緑色は構文が正しいことを示し、赤色は構文エラーが含まれていることを示しています。黄色は、構文は正しいけれども AS ファイルの規則が守られていないこと（たとえば同じ行が重複している場合など）を示しています。

編集領域 :

編集領域には、現在選択されているエントリ（行）についてのデータが表示され、ここでエントリを構成する任意の値を修正することができます。



“Operation”フィールドのコンボボックスには、使用可能なすべての演算が表示され、この中から使用する演算を 1 つ選択します。すると、その演算には使用されない他のフィールドはすべて自動的に無効になります。選択可能な演算は以下のとおりです。

```
ABS, NEG, ADD, SUB, MUL, DIV, MOD, MULDIV, GETAT,
GETATFIXED, INTERPOLGROUP, SEARCHDISTRIB, GETHIGHPART
```

“Shift”、“Operand1”、“Operand2”、“Operand3”および“Result”というフィールドもコンボボックスで、これら 5 つのフィールドにはすべて、使用できる型が表示されます。以下の型が使用可能です。

```
ALL, UINT8, UINT16, UINT32, SINT8, SINT16, SINT32
```

ALL 型は、特定の型ではなく、ワイルドカードとして使用されます。また型を選ばないという選択肢もできます。ただし、このオプションは、オプション引数を使用できる演算についてだけ選択できます。

演算によっては、**Limitation** および **Rounded** というオプションもオンになります。

“Generated Key”フィールドには、ASCET が使用する関数キー（509 ページ「関数キー」の項を参照してください）が表示されます。これは、エディタプログラムが自動的に生成したもので、このフィールドを直接編集することはできません。

“Code for representation”フィールドには、ASCET がコード生成に使用される関数呼び出し文が表示されます（512 ページ「関数の宣言」の項を参照してくださいという項を参照してください）。原則として、このフィールドの内容には特定の構文が定義されていないため、内容が正しいかどうかをエディタで検証することはできません。

“Parameters”フィールドはコンボボックスで、ここに表示される引数（`%i1%` など）を選択して“Code for representation”に追加することができます。

ボタン領域のエレメント：

この領域には、頻繁に使用されるコマンド用のボタンがあります。

- **New Set**：新しい空のセットを作成します。
- **New Entry**：新しい空のエントリを作成します。
- **Delete Entry**：カレントエントリを削除します。
- **Comment**：カレントエントリをコメントにします。
- **Uncomment**（コメントが選択されると **Comment** ボタンの代わりに表示されます）：現在のコメントを有効なエントリにします。
- **Update Entry**：カレントエントリについての変更を適用します。
- **Find Entry**：エントリを検索します。

注記

各ボタンは、エディタの状態に応じて有効／無効が切り替わります。

AS エディタの使用法

ファイルをロードする：

- **Open** ボタンをクリックします。

または

- **File** → **Open** メニューコマンドを選択します。
  ファイル選択ダイアログボックスが開きます。
- 開きたいファイルを選択します。
- **Open** ボタンをクリックしてファイルをロードします。

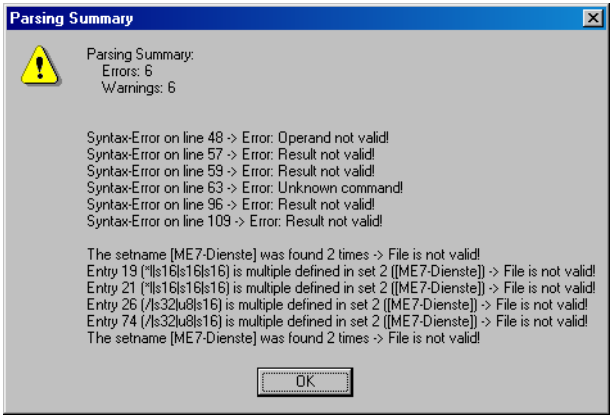
または

- **Cancel** をクリックしてこの操作を取り消します。

注記

最近開かれた 4 つのファイルが **File** メニューに表示されるので、それらのファイルを選択して直接開くこともできます。

ファイルを開くと、エディタプログラムはそのファイルの内容をチェックし、エラーを検知すると、ファイルのロードが完了したときに“Parsing Summary”ウィンドウに解析結果を表示します。



エラーが含まれていたエントリはコメント行に変更され、エラーについての説明が付加されます。このようなエントリは“Entries”フィールドに赤または黄色のLED 付きで表示されるので、エラー個所を容易に見つけ出すことができます。

注記

ファイルをロードする際、AS ファイル内の最初のセット定義より前（つまり、最初の左角括弧“[”より上）に記述されている内容はすべて無視されます。また、空の行もすべて無視され、ロードされません。

すべてのエントリがロードされると、ファイル内に最初に記述されているセットに属するエントリが“Entries”フィールドに表示されます。

ファイルを保存する：

- 既存のファイルに上書き保存するには、ツールバーの **Save** ボタンをクリックします。

または

- メニューから **File** → **Save** を選択します。

新しく作成したファイルを保存しようとすると、ファイルを保存するための標準ダイアログボックスが開くので、以下のように操作します。

- 新しいファイルの名前とディレクトリを選択します。
- **Save** をクリックして保存を実行します。

または

- **Cancel** をクリックして操作を取り消します。

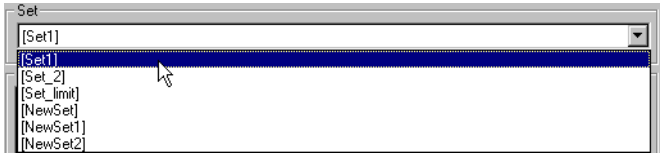
セットの管理：

ファイルがロードされると、そのファイルに含まれているすべてのセットが“Set”コンボボックスに表示されます。

セットを選択する：

- “Set”コンボボックスを直接、またはその右にある矢印ボタンをクリックします。

- ドロップダウンリストが開くので、そこからセットを選択します。



セットの数が多いためにドロップダウンリストに一度に表示しきれない場合は、リストの右にスクロールバーが表示されます。

選択したセットがアクティブになり、そのセットに含まれるエントリが“Entries”フィールドに表示されます。

セットを作成する：

- **Sets → New** メニューコマンドを選択します。

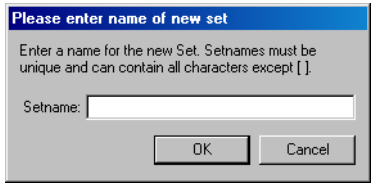
または

- ツールバーの **New Set** ボタンをクリックします。

または

- ボタン領域の **New Set** ボタンをクリックします。

“Please enter name of new set”ダイアログボックスが開きます。



- 新しいセットの名前を入力します。

注記

セットの識別はセット名により行われるので、各セットにはそれぞれ異なる名前を付ける必要があります。セット名には 1 ～ 255 文字を使用でき、[または] を含めることはできません。

- **OK** をクリックして入力を確定します。

  新しいセットがリストの最後に追加され、アクティブなセットとして選択されます。この新しいセットにはまだエントリが入力されていないので、“Entries”フィールドは空です。

または

- **Cancel** をクリックして操作を取り消します。

セットを削除する：

- **Sets** → **Delete** メニューコマンドを選択します。

または

- ツールバーの **Delete Set** ボタンをクリックします。

  そのセットとそれに含まれるすべてのエントリが削除される旨を知らせるダイアログボックスが開きます。

- **OK** をクリックしてセットの削除を実行します。

または

- **Cancel** をクリックして操作を取り消します。

注記

この操作を実行すると、セットとその中のすべてのエントリは、永久的に削除されます。

セットの複製を作成する：

- **Sets** → **Copy** を選択します。

または

- **Copy Set** ボタンをクリックします。

  “Please enter name of new set”ダイアログボックスが開きます。

- 新しいセットの名前を入力します。
- **OK** をクリックして操作を確定します。

  カレントセットのコピーが作成されて新しい名前が付けられ、その新しいセットがセットリストの最後に追加され、さらにそのセットがアクティブセットとして自動的に選択されます。

または

- **Cancel** をクリックして操作を取り消します。

セットの名前を変更する：

- **Sets** → **Rename** を選択します。

または

- **Rename** ボタンをクリックします。
- “Please enter name of new set”ダイアログボックスに、セットの新しい名前を入力します。
- **OK** をクリックして操作を確定します。

または

- **Cancel** をクリックして操作を取り消します。

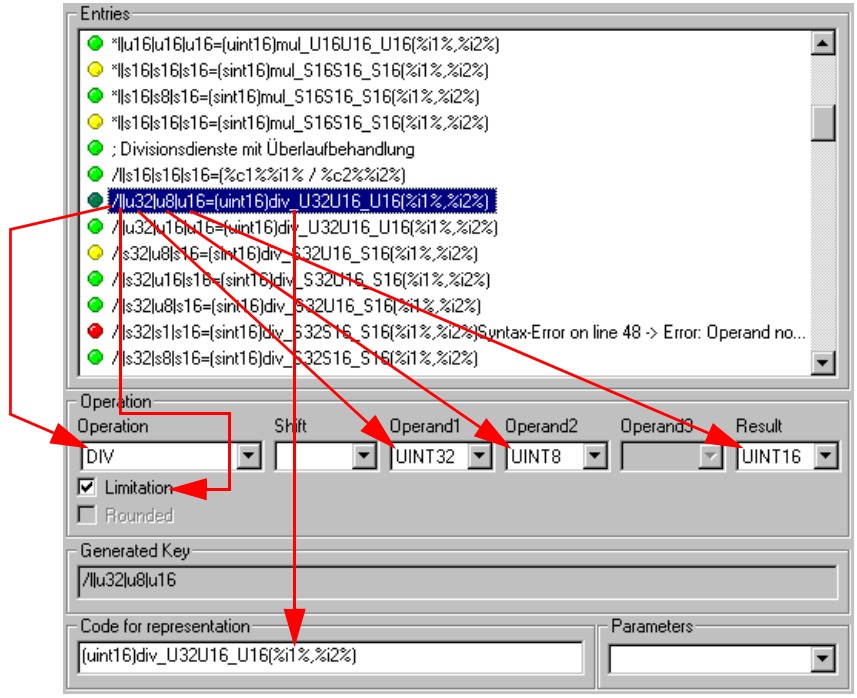
エントリの管理：

ファイルがロードされてセットが選択されると、そのセットに属するすべてのエントリが“Entries”フィールドに表示されます。エントリの数が多くてリストボックスに表示しきれない場合は、そのフィールドの右端にスクロールバーが表示されます。1 つのエントリをマウスでクリックして選択すると、そのエントリの内容がウィンドウ下部の*編集領域*内の各フィールドに表示され、これらを編集することが可能となります。

エントリを選択する：

- “Entries”フィールドのエントリをクリックします。

  そのエントリを構成する値が編集領域の各フィールドに表示されます。下図の矢印は、エントリのどの要素が編集領域のどのフィールドに表示されているかを示しています。



エントリにエラーが含まれている場合（LED シンボルが黄色か赤色になっている場合）、そのエントリは無条件にコメントとして扱われます。

各エントリは、「関数」または「コメント」という 2 種類のタイプに分類されます。コメントは“;”で始まります。選択したエントリがコメントである場合、編集領域は、“Code for representation”フィールドと“Parameters”フィールド以外のフィールドはすべて無効になり、関数を選択した場合は、演算の種類に応じて各フィールドが有効または無効になります。有効なフィールドには、エントリ内で定義された値が表示されます。

エントリを作成する：

- **Entry → New** を選択します。

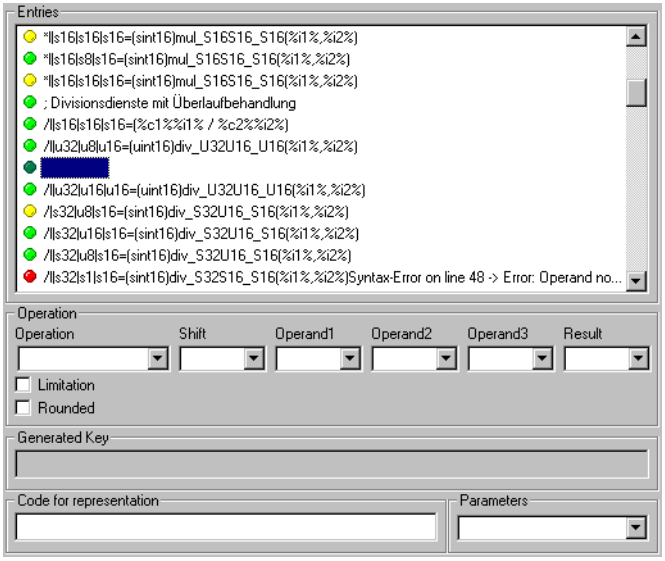
または

- ツールバーの **New Entry** ボタンをクリックします。

または

- ボタン領域の **New Entry** ボタンをクリックします。

  直前に選択されていたエントリの下に新しい空のエントリが作成され、編集領域のすべてのフィールドが有効になります。



新しく作成されたエントリは、「関数」として作成されます。コメントを作成するには、作成したエントリをコメントに変更します（538 ページの「エントリをコメントにする：」を参照してください）。

AS エディタでは、エントリの文字列を直接編集することはできません。その代わり、いくつかの入力オプションを選択することにより、エントリに含まれる個々の引数を変更することができます。また、エントリの内容が誤った内容に変更されてしまうのを防ぐために、入力された内容は、マニュアル操作で更新処理を実行することにより、初めて適用されます。

エントリを定義する：

- “Entries”フィールドから、変更したいエントリを選択します。

  選択されたエントリの内容が、編集領域の各コンボボックスに表示されます。この値を変更するには、コンボボックスから他の項目を選択します。
- “Operation”コンボボックスで、算術演算を選択します。

注記

選択されている種類の算術演算で使用されるフィールドのみが有効になります。

- “Shift”コンボボックスで、シフトの種類を選択します。

  シフトを使用しない場合には、空の項目を選択します。
- 複数の“Operand*”コンボボックスで、演算の各オペランド（演算項）の型を選択します。

  特性テーブルにアクセスする演算子（`GETAT`、`GETATFIXED`、`INTERPOLGROUP`）が選択されている場合、“Operand1”と“Operand2”の 2 つのフィールドが有効になっています。*特性マップ*にアクセスする場合には、これら 2 つのフィールドの値をそれぞれ選択し、*特性カーブ*にアクセスする場合は、“Operand2”フィールドを空のままにしておいてください。
- “Result”コンボボックスから結果の型を選択します。
- 演算結果の値に範囲制限を加えたい場合には、**Limitation** オプションをオンにします。
- 特性カーブ／マップへの丸めアクセスを行うには、**Rounded** オプションをオンにします。
- 関数呼び出しのコード、またはコメントのテキストを、“Code for representation”フィールドに記述します。

または

- 標準的な関数呼び出しを使用するには、フィールドの内容をすべて削除します（537 ページを参照してください）。

- コードに引数を追加する必要がある場合には、“Parameters”コンボボックスから引数を 1 つずつ選択します。

  選択された引数が、“Code for representation”フィールド内の文字列の最後に挿入されます。

変更作業がすべて終わったら、以下に示す方法で、エントリを更新して入力内容を確定します。

注記

“Generated Key”フィールドの値は、変更内容が確定されたときにエディタプログラムにより自動的に生成されるので、この内容を任意に編集することはできません。

エントリを更新する：

- **Entry → Update** を選択します。

または

- ツールバーの **Update Entry** ボタンをクリックします。

または

- ボタン領域の **Update Entry** ボタンをクリックします。

  編集領域の値が適用され、構文チェックが行われます。入力内容がすべて正しければ、その内容が現在選択されているエントリに適用されます。

エントリの更新により変更内容が適用される際には、以下のような点について構文チェックが行われます。

- 必要な項目がすべて入力されていること
- 入力された構文が正しいこと
- 現在選択されているセット内に同じ内容のエントリがないこと

上記の条件がすべて満たされている場合にのみ、エントリの内容が更新され、そうでない場合には、エラーメッセージが出力され、エントリは変更されません。

“Code for representation”フィールドを空のままにしてエントリの更新を行い、更新が正しく行われると、このフィールドには他の入力フィールドの情報から導き出される標準的な関数呼び出しが表示されます。この内容は推奨例として扱うことができ、これに変更を加えることも可能です。

注記

“Code for representation”フィールドの値を任意に編集した場合、その内容のチェックは行われません。またこのフィールドは、ユーザーの責任において正しく定義し、ASCET のコード生成時に適切な算術演算サービスが正しく使用されるようにしてください。

このエディタで生成されるコードは、509 ページの「定義済みの算術演算サービス」の項にまとめて掲載されています。下の例は、変換規則ををわかりやすく示した例です。

関数キー：	生成される標準的な関数呼び出し：
`+l|*|*|*`	`addl_%t1%%t2%_%tr%(%i1%, %i2%)`
`+l|*|u8|*`	`addl_%t1%u8_%tr%(%i1%, %i2%)`
`+l|u16|u8|*`	`addl_u16u8_%tr%(%i1%, %i2%)`
`+l|u16|u8|u32`	`addl_u16u8_u32(%i1%, %i2%)`

エディタが関数呼び出しを生成する際の変換規則は、任意に変更することができます。このためには、エディタと同じディレクトリ（デフォルトでは `..¥ETAS¥Ascet5.2¥ASEditor`）にある `operations.ini` および `type.ini` という 2 つのファイルを編集する必要があります。以下の例のように、各演算または各型に対応する文字列を定義して、それを標準文字列の代わりに使用されるようにします。

注記

`operations.ini` および `type.ini` という 2 つのファイルを編集するときには、これらのファイル名を変更しないように注意してください。変更してしまうと AS エディタはこれらのファイルを見つけることができなくなり、標準設定を使用することになってしまいます。

例： `operations.ini` ファイル内のエントリ

```
add = add_
```

を

```
add = doAddition_
```

に変更すると、

```
+|*|*|*
```

というキーに対応する標準コードは

```
add_%t1%%t2%_%tr%(%i1%, %i2%)
```

ではなく、以下のようになります。

```
doAddition_%t1%%t2%_%tr%(%i1%, %i2%)
```

エントリを削除する：

注記

削除操作を行う際、確認のためのプロンプトは表示**されません**。削除されたエントリを復元することはできないので、この操作は慎重に行ってください。

- **Entries → Delete** メニューコマンドを選択します。

または

- **Delete Entry** ボタンをクリックします。

  確認のプロンプトは表示されずに、選択されたエントリが削除されます。

エントリを削除すると、そのエントリの次（直下の行）に位置していたエントリが自動的に選択され、リスト内の最後のエントリを削除した場合には、その直前のエントリが選択されるので、複数のエントリを連続して削除することができます。

AS エディタでは、コメントを AS ファイルに挿入することができます。また、既存のエントリの先頭に“;”を挿入するだけで、そのエントリをコメントに変更することもできます。“;”から始まる行は、ASCET でも AS エディタでもコメントとして扱われます。

エントリをコメントにする：

- “Entries”フィールドのエントリのリストからエントリを選択します。
- **Entries → Comment** メニューコマンドを選択します。

または

- ツールバーの **Make Comment** ボタンをクリックします。

または

- ボタン領域の**Comment**ボタンをクリックします。

  選択されたエントリがコメントになります。

コメント行の先頭の“;”を取り除くだけでその行が有効なエントリになるような場合は、そのコメントを有効なエントリに変えることができます。この機能は、エラーが含まれているエントリの訂正にも利用できます。

コメントを有効なエントリに変える操作を行うと、“Code for representation” フィールドに表示されているテキストの構文チェックが行われ、エラーが検知されなければ、そのコメントはエントリに変わります。一方、エラーが発見された場合は、そのコメントはコメントのままとなります。

コメントをエントリに変える：

- “Entries” フィールドからコメントを選択します。

  ボタン領域の **Comment** ボタンの名前が **Uncomment** に切り替わります。
- **Entries → Comment** メニューコマンドを選択します。

または

- ボタン領域の **Uncomment** ボタンをクリックします。

  選択されたコメントのテキストがエントリの構文として正しければ、そのコメントはエントリに変わりますが、正しくなければコメントのままで何も変わりません。

セット内のある特定のエントリを探し出す必要がある場合、関数キーを検索キーとしてエントリを検索することができます。

エントリを検索する：

- 編集領域の “Operation”、“Operand1”、“Operand2”、“Operand3”、“Shift” および “Result” のフィールドでそれぞれ値を選択し、検索したい関数キーの条件を設定します。
- 検索したい関数キーの数値範囲制限や丸め機能の有無に合わせて **Limitation** と **Rounded** オプションを設定します。
- **Entries → Find** を選択します。

または

- ボタン領域の **Find Entry** ボタンをクリックします。

  指定した設定と一致するエントリがあれば、そのエントリが選択されて “Entries” フィールドに表示されます。

一致するエントリが存在しない場合や、一致するエントリがすでに選択されている場合には、その状況に応じたメッセージが表示されます。

5 シグナルとアイコン

本章では、ASCET データベースにおける「シグナル」と「アイコン」の扱いについて説明します。シグナルとアイコンは、実験や機能記述を行う際に補足的に使用されるデータで、他のコンポーネントやプロジェクトと共に ASCET データベースに格納されます。

シグナルやアイコンは、ファイルシステムにエクスポートしたり、そのファイルをデータベースにインポートすることができます。

シグナルは、シミュレーション実行時に「スティミュレート」される、つまり外部からの刺激信号としてモデルに与えられる一連の信号です。

アイコンは、ブロックダイアグラムにおいてエレメントや階層に割り当てられるシンボルです。

5.1 シグナルビューア

「シグナルビューア」は、保存された測定信号データを ASCET データベースの「シグナル」というアイテムにロードしたり、データの内容を参照するためのものです。このビューアは、さまざまな信号フォーマットを読み取り、別のフォーマットに変換することができます。

ASCET は、以下の測定データフォーマットをサポートしています。

- タブで区切られた ASCII フォーマット（一般的なスプレッドシートやデータベースを扱うアプリケーションで読み書きできます。）
- ETAS フォーマット（非常に効率的なバイナリフォーマットで、ASCET 内部でのみ使用できます。）
- FAMOS チャンネルフォーマット
- FAMOS レコードフォーマット
- MATLAB フォーマット（MATLAB/Simulink 用の ASCII フォーマットです。）
- MDF フォーマット

上記のフォーマットのうち最後の 4 つは他社の測定データフォーマットです。詳細は、それぞれのドキュメントを参照してください。ASCET は ASCII、ETAS、FAMOS チャンネル、および FAMOS レコードのフォーマットを処理することができ、また、MDF と MATLAB フォーマットの読み書きができます。

新しいシグナルを作成する、または既存のシグナルを開く：

- コンポーネントマネージャで、**Insert → Signal** を選択します。

  新しいシグナルが作成されます。
- “1 Database”リストに表示されたシグナルをダブルクリックします。

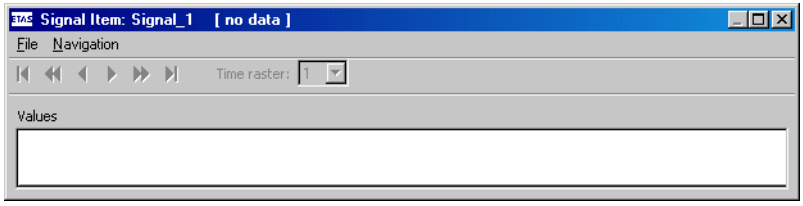
または

- シグナルを選択して **Component** → **Edit Item** を選択します

または

- **<Enter>** キーを押します。

シグナルビューアが開きます。



測定データをシグナルにインポートする：

- シグナルをシグナルビューアで開きます。
- シグナルビューアで **File** → **File In Data** → **<format>** を選択します。

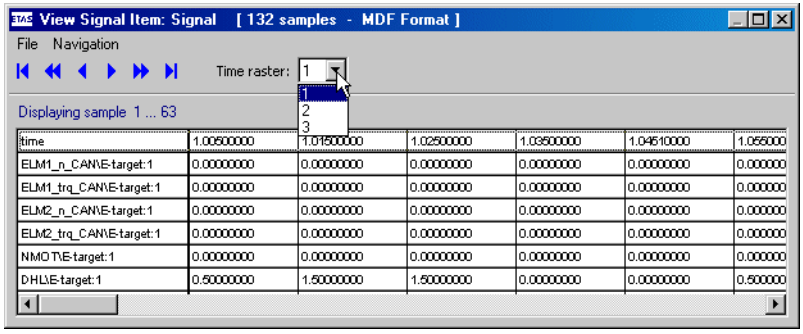
  Windows のファイル選択ダイアログボックスが開きます。

- インポートしたいデータファイルのパスとファイル名を選択します。

注記

INCA や ASCET での実験中に作成された測定ファイルを使用できます。

データがインポートされ、指定のフォーマットに自動的に変換されます。変換後、データがビューアに表示されます。



MDF または FAMOS フォーマットをインポートした場合は、異なるタイムフレームでデータを表示することが可能です。

測定データを表示する：

- 次のデータ列を表示するには、**Navigation → Show Next** を選択します。

または

- **Show Next Column** ボタンをクリックします。
- 次の 1 画面文のデータを表示するには、**Navigation → Fast Forward** を選択します。

または

- **Fast Forward** ボタンをクリックします。

  画面に 5 列分のデータが表示されていた場合、次の 5 列のデータが表示されます。
- 最後のデータ列を表示するには、**Navigation → Show Last** を選択します。

または

- **Show Last Column** ボタンをクリックします。

上記の 3 つのボタンに対して、反対方向への移動を行うためのボタンも用意されています。

5.2 アイコンエディタ

コンポーネントのレイアウトにアイコンを割り当てると、そのコンポーネントが他のコンポーネントに内包される場合、ブロックダイアグラムエディタにおいて、内包されるコンポーネントを表わすブロック内にそのアイコンが表示されます。あらかじめ用意されているアイコンを選んで使用したり、ユーザー独自のアイコンを作成して使用することもできます。

アイコンエディタを起動する：

- コンポーネントマネージャで、**Insert → Icon** を選択します。

または

- データベースから既存のアイコンを選択します。
- 選択されたアイコンをダブルクリックします。

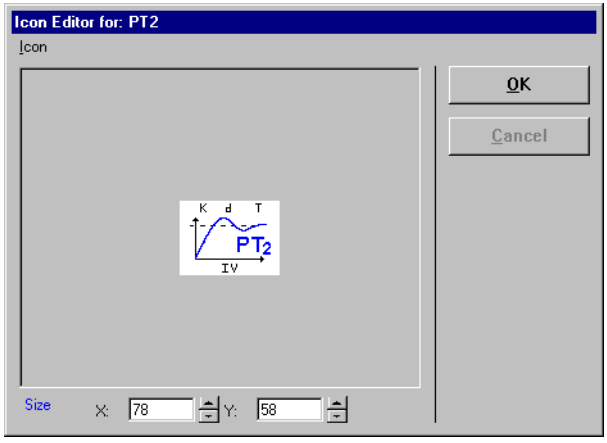
または

- **Component → Edit Item** を選択します。

または

- **<Enter>** キーを押します。

アイコンエディタが開きます。



新しく作成されたアイコンの場合、描画領域は空になっています。既存のアイコンを開いた場合は、アイコンが描画領域に表示されます。

ここに任意のビットマップファイル（BMP、PCX、TIFF）をロードして、ASCET用アイコンとして使用することができます。この際、描画プログラムで作成したユーザー独自のアイコンを使用したり、既存のアイコンを使用することができます。ただし独自のアイコンを使用する際は、そのビットマップが適切なサイズであることを確認し、アイコンのサイズが、その割り当て先とするグラフィックブロックよりも大きくならないようにしてください。ASCET のグリッドの 1 単位は 5 ピクセルに相当します。

アイコンをロードする：

- **Icon → Load** を選択します。

  “Image File”ダイアログボックスが開きます。
- ロードしたいアイコンファイルを選択します。
- **OK** をクリックします。

  選択されたアイコンがアイコンエディタに表示されます。

アイコンのサイズは、割り当てるブロックの大きさに合わせて拡大／縮小することができます。ただし、拡大によってアイコンの画質は低下するので、高倍率の拡大は避けてください。

アイコンを拡大／縮小する：

- 拡大／縮小したいアイコン用のアイコンエディタを開きます。
- アイコンペインの下にある 2 つの “Size” ボックスを使用して、アイコンの縦と横の長さを調整します。

  “Size”ボックスの値を調整すると、すぐにアイコンのサイズが変わります。
- **OK** をクリックします。

ASCET のアイコンはビットマップファイルとして保存することができます。これは、たとえば、コンポーネント用に作成したドキュメントに図を掲載するような場合に利用できます。

アイコンをファイルに保存する：

- ファイルに保存したいアイコンを、アイコンエディタで開きます。
- **Item → File Out → <file format>** を選択します。

  “Image File”ダイアログボックスが開きます。
- アイコンを保存したいファイル名とパスを選択します。

- **OK** をクリックします。

  アイコンが指定のフォーマット（BMP、PCX または TIFF）で保存されます。

6 実験

6.1 実験環境

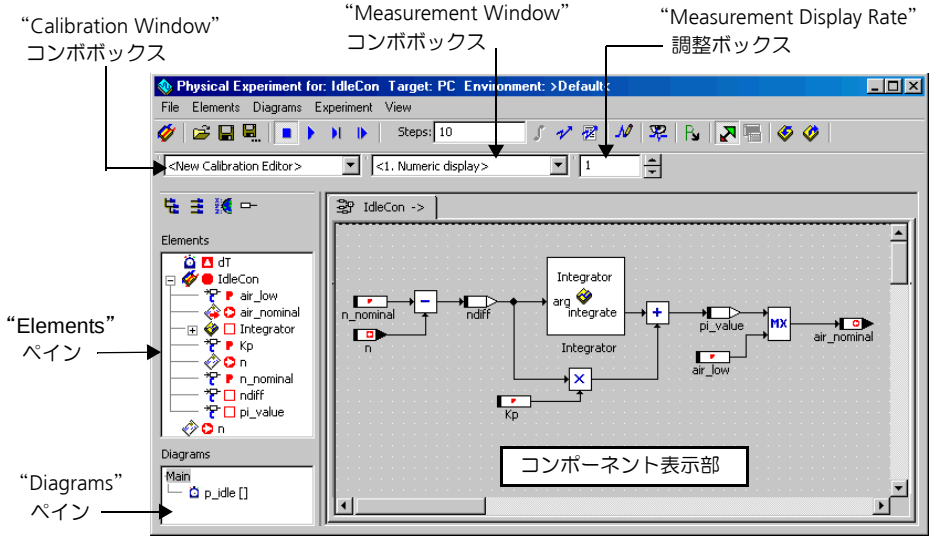
ASCET では、モジュール単位でのソフトウェア開発が可能です。プロジェクトに含まれる各コンポーネントを、個別に開発して徹底的なテストを行い、それをシステムとして組み立てていくことができます。ASCET には、このようなコンポーネント単位、またはプロジェクト全体のテストを行うための実験環境が用意されています。

プロジェクトのテストは、「オフライン実験」と「オンライン実験」が可能です。プロジェクトを外部ハードウェアと接続してリアルタイムに実行させる場合は、オンライン実験のみが可能です。

注記

オンライン実験を行うには、ASCET-RP がインストールされている必要があります。オンライン実験についての詳しい説明は、ASCET-RP のユーザーズガイドを参照してください。

オフライン実験では、メソッドやプロセス、およびコンポーネントまたはプロジェクトのエレメントに、外部からの入力信号の代わりとなる一連のシグナル（「スティミュラスシグナル」と呼ばれます）を与えることができます。エレメントの値をさまざまな表示形式で読み取ったり、ファイルに書き込んだり、また対話形式で適合することもできます。実験ウィンドウの構成（オフライン実験の場合）を、下図に示します。



コンポーネントに含まれるエレメントは、ウィンドウの左側部分にある“Elements”ペインに表示されます。コンポーネントがブロックダイアグラムで記述されたものである場合、ウィンドウ中央のコンポーネント表示部にはブロックダイアグラムが表示され、C コードまたは ESDL で記述されたコンポーネントについては、そのコードが表示されます。また、プロジェクトの場合はプロジェクトエディタのタブが表示されます。

“Elements”ペインの下には“Diagrams”ペインがあり、そこには現在のコンポーネントに含まれるすべてのダイアグラムが一覧表示されます。ツールバーの下にある“Calibration Window”コンボボックスには使用可能なすべての適合ウィンドウがリストアップされ、その隣の“Measurement Window”コンボボックスには測定ウィンドウがリストアップされます。タイトルバーには現在のコンポーネントまたはプロジェクトの名前と、現在のターゲットが表示されます。

6.1.1 メニューコマンドの概要

- **File:**
  - *Load Environment*

    実験の環境設定を、現在の実験環境上にロードします。
  - *Save Environment*

    実験の環境設定を保存します。
  - *Save Environment As*

    実験の環境設定を、別の名前で保存します。
  - *Export Environment*

    実験の環境設定を、ファイルにエクスポートします。
  - *Exit*

    実験環境を終了します。
- **Elements:**
  - *Show Component*

    選択された被参照コンポーネントを表示します。
  - *Calibrate*（**<Ctrl> + <C>**）

    選択されたエレメント用の適合ウィンドウを開きます。
  - *Stimulate*（**<Ctrl> + <S>**）

    注記

    このコマンドは、**オフライン**実験でのみ使用できます。

    選択されたエレメント用の“Stimulus”ダイアログボックスを開きます。

- *Measure*（**<Ctrl> + <M>**）

  選択されたエレメント用の測定ウィンドウを開きます。

- *Log*（**<Ctrl> + <L>**）

  選択されたエレメントをデータロガーに追加します。

- *Log All*（**<Ctrl> + <A>**）

  すべてのエレメントをデータロガーに追加します。

- *Reinitialize*

  コンポーネントで現在使用されているデータセットから、エレメントの値を読み込みます。
  → *Variables* ― 変数
  → *Parameters* ― パラメータ
  → *Both* ― 変数とパラメータ

- *Write Back Data*

  コンポーネントで現在使用されているデータセットに、選択されたエレメントの値を書き込みます。
  → *Selected Elements* ― 選択されたエレメント、
  → *Calibrated Elements* ― 実験で適合されるエレメント

- *Show Element Implementation*（**<Ctrl> + <L>**）

  選択されたエレメントのインプリメンテーションを表示します。

- *Update Dependent Parameters*（**<Ctrl> + <U>**）

  依存パラメータの値を更新します。

- *Load Data*

  ファイルからパラメータのデータを読み取ります。詳細は個別のダイアログボックスで設定します。

- *Save Data*

  選択されたパラメータまたは変数のデータをファイルに書き込みます。詳細は個別のダイアログボックスで設定します。

- **Diagrams:**
  - *Show Diagram*

    選択されたダイアグラムを表示します。

  - *Stimulate*

    注記

    このコマンドは、**オフライン**実験でのみ使用できます。

    選択されたメソッドまたはプロセスの“Event”ダイアログボックスを開きます。ここでイベントを設定します。

- **Experiment**:

注記

これらのコマンドは、**オフライン**実験でのみ使用できます。

— *Event Generator →Open*

イベントジェネレータを開きます。

— *Event Generator →Step Mode*

ステップモードのタイプを選択します。
→ *Steps* — シングルステップモード（デフォルト）、
→ *Timed [s]* — 指定時間モード、
→ *Break At Condition* —ブレークポイントモード

— *Event Generator →Edit Break Condition*

ブレークポイントコンディションエディタを開きます。

— *Event Tracer*

データのトレースを行う“Event Tracer”ダイアログボックスを開きます。

— *Data Genarator*

データジェネレータを開きます。

— *Data Logger*

データロガーを開きます。

— *Automatic Stop*

一連のスティミュラスシグナルが終了した時点で、自動的に実験を終了します（567 ページを参照してください）。

— *Stop Experiment*

実験を終了します。

— *Start Experiment*

実験を開始します。

— *Pause Experiment*

実験を一時停止します。

— *Step Experiment*

実験をステップモードで実行します。

— *Open Target Debugger*

C コードコンポーネントのデバッグウィンドウを開きます。

— *Update Calibration Windows*

適合ウィンドウの内容を更新します。

— *Close Calibraion Windows*

すべての適合ウィンドウを閉じます。

— *Close Measure Windows*

すべての測定ウィンドウを閉じます。

- **View:**
  — *Redraw*

  ブロックダイアグラムを再描画します。

  — *Hide Seq. Calls*

  ブロックダイアグラム上のシーケンスコールを非表示状態にします。

  — *Show Seq. Calls*

  ブロックダイアグラム上にシーケンスコールを表示します。

  — *Monitor All*

  ブロックダイアグラム上のすべてのエレメントの値をモニタ表示します。

  — *Delete Monitors*

  ブロックダイアグラム上のすべてのモニタ表示を消去します。

  — *Show Parent Component*

  上位のコンポーネントを表示します。

6.1.2 実験環境の起動とセットアップ

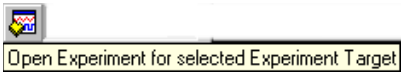
コンポーネントまたはプロジェクトの実験環境は、コンポーネントエディタまたはプロジェクトエディタから起動します。

オフライン実験用の実験環境を開く：

- コンポーネントまたはプロジェクトを開きます。
- プロジェクトのオフライン実験を行うには、ビルドオプションで、`PC` をターゲットとして選択します。

  “Experiment Target”コンボボックスに `Offline (PC)` と表示されます。
  コンポーネントの場合、このコンボボックスは無効になります。
- **Component → Open Experiment** を選択します。

または



- **Open Experiment for selected Experiment target** ボタンをクリックします。

  コンポーネントまたはプロジェクト用にデフォルトの実験環境が開きます。複数の環境がすでに保存されている場合には、どれを開くかを選ぶことができます。詳細は、579 ページの「環境設定のロードと保存」という項を参照してください。

コンポーネントの実験を行う際、デフォルトプロジェクトにおいて、グローバルエレメント（380 ページの「グローバル通信の定義」を参照してください）が正しく更新されない場合があります。この時は、以下のエラーメッセージが ASCET モニタウィンドウに表示されます。

```
Error: need export for import element <name> with type
<type>
```

このエラーを修正するには、以下のようにしてください。

デフォルトプロジェクト内でグローバルエレメントを定義する：

- ブロックダイアグラムエディタで、**Component → Default Project → Resolve Globals** を選択して、グローバルエレメントを更新します。

  この後、実験を開いてください。

これで実験を開始することができるようになりました。オフライン実験のセットアップは、以下の 4 つのステップで行います。

- コンポーネントマネージャまたは各コンポーネントエディタから、実験環境を起動します。実験環境は、すべてのタイプのコンポーネントについて同じように機能します。
- イベントジェネレータをセットアップします。イベントジェネレータによって、指定のメソッドまたはプロセスが、指定のモードで起動されます。プロジェクトの実験を行う場合は、メソッドやプロセスではなく、タスクが起動されます。
- データジェネレータをセットアップします。データジェネレータを介し、任意に設定可能な一連のデータが、コンポーネント内の指定のエレメントにスティミュラスシグナルとして与えられます。
- 測定／適合ウィンドウをセットアップします。すべてのエレメントの値を数値ディスプレイやオシロスコープなどのさまざまな形式で表示することができます。

6.1.3 イベントジェネレータ

オンライン実験では、リアルタイムオペレーティングシステムがプロジェクトに含まれる各タスクとプロセスをスケジューリングしますが、オフライン実験の場合は、イベントジェネレータによってスケジューリングが行われます。このため、イベントジェネレータでイベントを設定し、各イベントによって、実験対象であ

るコンポーネントのどのメソッドまたはプロセスがどのような順序やモードで起動されるか、ということを定義します。イベントは、起動される各メソッドまたはプロセスごとに必要です。

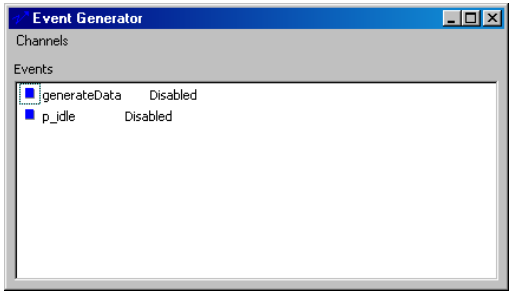
イベントジェネレータをセットアップする：

- **Experiment** → **Event Generator** → **Open**を選択します。

または

- **Open Event Generator** ボタンをクリックします。

  “Event Generator”ウィンドウが開きます。このウィンドウには、コンポーネント内に定義されているすべてのメソッド（クラスの場合）またはプロセス（モジュールの場合）に対応するイベントがそれぞれ表示されています。デフォルト状態においてはすべてのイベントが「無効」、つまりイベントが発生しない状態になっているので、まず、実験で使用したいイベントを有効にする必要があります。



- イベントを選択します。
- **Channels** → **Enable** を選択します。

または

- イベントを右クリックし、ショートカットメニューから **Enable** を選択します。
- 有効にしたいイベントについて、それぞれ以上の操作を繰り返します。
- 有効になっているイベントを無効にするには、イベントを選択して **Channels** → **Enable** を再度選択します。

該当するイベントが有効になっていないメソッドやプロセスは起動されないので、実験に影響を与えることはありません。

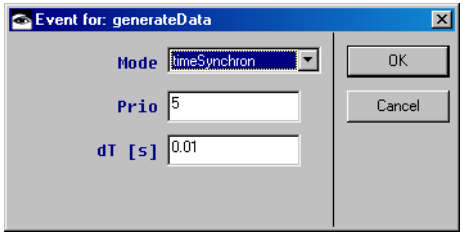
イベントジェネレータには、メソッドやプロセス用の各イベントの他に、`generateData` というデフォルトイベントが必ず含まれます。このイベントは、データジェネレータによるデータの生成を開始するためのイベントです。データの生成が必要な実験を行う場合は、必ずこのイベントを有効にしてください。

各イベントのイベントオプションには、デフォルト値が設定されているので、必要に応じてこれらの設定を、以下のようにして変更します。

イベントをセットアップする：

- セットアップしたいイベントを選択します。
- **Channels → Edit** を選択します。

  “Edit Event”ダイアログボックスが開きます。



- イベントオプションを調整します。

  各オプションの意味については、555 ページを参照してください。
- **OK** をクリックします。
- セットアップしたいすべてのイベントについて、以上の操作を繰り返します。

イベントを直接有効にしてセットアップする：

- “Diagrams”ペインから、イベントを有効にしたいメソッドまたはプロセスを選択します。
- 選択されたメソッドまたはプロセスを右クリックし、ショートカットメニューから **Stimulate** を選択します。

または

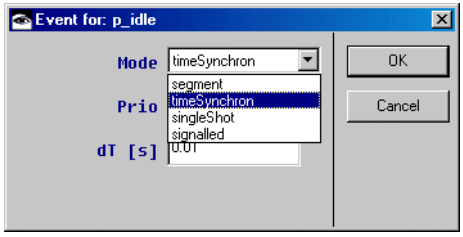
- **Diagrams → Stimulate** を選択します。

  そのイベント用の“Event for:”ダイアログボックスが開きます。必要に応じて内容を編集してダイアログボックスを閉じると、設定された内容でそのイベントが有効になります。

イベントオプション

“Event for:”ダイアログボックスでは、各イベントについてのオプションを設定することができます。

- “Mode”オプション：イベントのモードを、以下に示す 4 種類のうちから選択します。

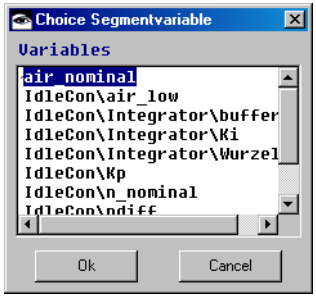


  - 時間同期イベント（`timeSynchronous`）：各インターバルの初めに一度発生します。インターバルの長さは、“dT [s]”フィールドで設定します。
  - セグメントイベント（`segment`）：自動車制御システムにおいて、エンジンの回転に連動してイベントを発生させる必要があるような場合に使用されます。詳細は、次項の「セグメントイベントをセットアップする：」を参照してください。
  - シングルショットイベント（`singleShot`）：シミュレーション開始時に一度だけ発生します。これは、初期化処理などに使用できます。“Event Generator”ウィンドウの **Channels → Reactivate Event** コマンドを実行すると、このイベントを再び発生させることができます。
  - 非同期 イベント（`signalled`）：モデルに与えられるスティミュラスシグナルによってトリガされるイベントです。このオプションを使用すれば、オフライン実験であっても実際に測定されたデータをトリガとして使用することができます。非同期イベントの設定については、558 ページを参照してください。
- “Prio”オプション：イベントの発生順序は、この優先度 によって決まります。複数のイベントに同じインターバル（例、10 ミリ秒）が設定されることがよくありますが、そのような場合、実際には、優先度の最も高いイベント、つまりここに設定された値が最も大きいイベントから順に発生します。
- “dT [s]”オプション：イベントを発生させるインターバルを決定する `dT` 値を入力します。`dT` 値が 0.01 秒なら、そのイベントは 10 ミリ秒ごとに発生します。`dT` に設定できる最小の値は 1 マイクロ秒（10^{-6} 秒）です。

セグメントイベントをセットアップする：

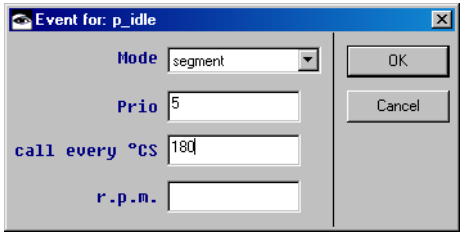
- イベント用の“Event for:”ダイアログボックスを開きます。

- “Mode”コンボボックスから Segment を選択します。
- ダイアログボックスが開きます。そこには、コンポーネントに含まれるすべての変数がリストアップされています。



- セグメント変数として使用する変数を選択し、**OK** をクリックします。

  「セグメント変数」とは、毎分の回転数を表す変数です。
- “Event for:”ダイアログボックスに“call every °CS”フィールドが表示されるので、そこにクランク角を入力します。
- **OK** をクリックします。



これで、セグメントイベントは、tseg（単位は秒）のセグメントごとに発生するようになります。tseg の値は、以下の式で算出されます。

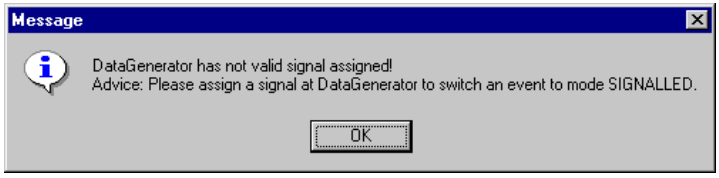
$$t_{seg}[s] = \frac{CS[deg]}{n\left[\frac{1}{min}\right] \cdot 6}$$

ここで、n は毎分の回転数で、CS はクランク角です。

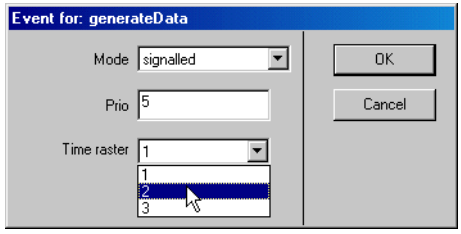
非同期イベントをセットアップする：

- イベントを“Event for:”ダイアログボックスで開きます。
- “Mode”コンボボックスから、`signalled` を選択します。

  このとき、データジェネレータ（558 ページの「データジェネレータ」を参照してください）で何もシグナルが選択されていないと、以下のエラーメッセージが表示されます。



- エラーを解除するには、以下のようにします。
  - **OK** をクリックして、メッセージを閉じます。
  - イベントを直接セットアップする（554 ページ参照）場合は、“Event for:”ダイアログボックスを閉じます。
  - 
    “Physical Experiment”ウィンドウで、**Open Data Generator**”ボタンをクリックしてデータジェネレータを開きます。
  - 564 ページの「インポートされたシグナルを実験に割り当てる：」の方法で、シグナルを定義します。
  - 最初に戻ってイベントの設定を行います。



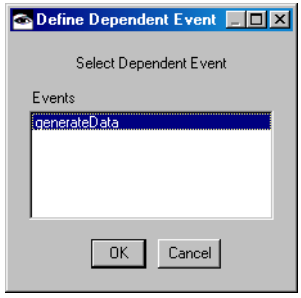
  - シグナルがマルチラスタ（複数のサンプリングレートを含むシグナルデータ）であった場合、使用するラスタを選択して、
- **OK** をクリックして設定を確定します。

「依存イベント」として指定されたイベントは、他のイベントが発生したときに連動して同時に発生します。ただし、「依存イベント」として指定されていても、そのイベント自身の設定内容（モードや周期など）に基き、独立的なイベントとしても発生します。

依存イベントをセットアップする：

- イベントジェネレータウィンドウで、イベントを選択します。
- **Channels → Dependent Event** を選択します。

  “Define Dependent Event”ダイアログボックスが開きます。



- 先のイベントが依存するイベントを選択します。

  ここで選択されたイベントに連動して、先のイベントが発生します。
- **OK** をクリックします。

6.1.4 データジェネレータ

オフライン実験において、データジェネレータは、コンポーネントに含まれるエレメントに対して「スティミュラスシグナル」、つまりシミュレーションを実行するために必要な外部からの刺激信号を供給します。オンライン環境でソフトウェアが実行される際は、このようなデータは、コンポーネントのインターフェースエレメントに対して供給されますが、データジェネレータを使えば、どのエレメントに対してもスティミュラスを与えることができます。データジェネレータは、正弦波やパルスなど、さまざまなシグナルを生成することができます。

以下に示す方法で「データチャンネル」を作成し、各チャンネルごとにシグナルモードと供給先のエレメントを指定します。

注記

オフライン実験でコンポーネント単位の実験を行う際は、オンライン実験においては「モデル値」、つまり他のコンポーネントから出力される値を使用する変数やパラメータに対して、スティミュレーション（つまりスティミュラスシグナルの供給）が必要となる場合があります。必要に応じて設定を行ってください

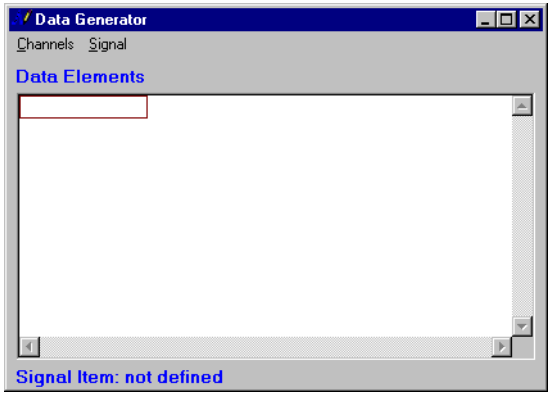
データジェネレータをセットアップする：

- 実験環境ウィンドウのツールバーの **Open Data Generator** ボタンをクリックします。

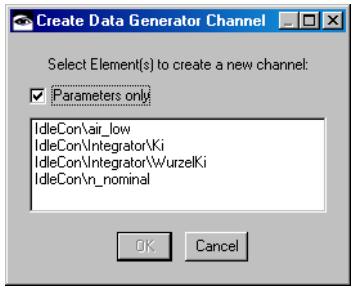
または

- **Experiment → Data Generator** を選択します。

“Data Generator”ウィンドウが開きます。



- **Channels → Create** を選択して、スティミュレーションを行うエレメントを選択するダイアログボックスを開きます。



  デフォルト状態において、このダイアログボックスには現在のコンポーネントに含まれるエレメントのうち、パラメータだけが表示されますが、必要に応じて、すべての基本エレメントに対してスティミュレーションを行うことができます。

- すべての基本エレメントを表示するには、**Parameters only** オプションをオフにします。
- データチャンネルを割り当てるエレメント、つまりスティミュラスシグナルを供給する対象とするエレメント（1 つまたは複数）を選択します。

  <Ctrl> キーを押したまま複数のエレメントをクリックすると、それらを一度に選択することができます。

- **OK** をクリックします。
- データチャンネルを作成したいすべてのエレメントについて、以上の操作を繰り返します。

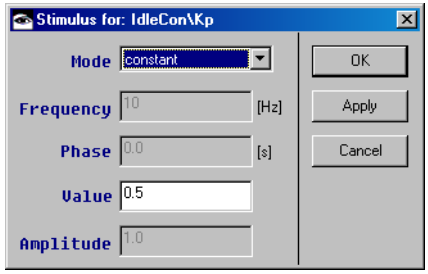
データジェネレータのチャンネルをセットアップする：

- “Data Generator”ウィンドウから、チャンネルを選択します。
- **Channels → Edit** を選択します。

または

- 選択されたチャンネルをダブルクリックします。

  “Stimulus for:”ダイアログボックスが開きます。そのタイトルバーには、選択されているチャンネルの名前（エレメント名と同じです）が表示されます。



- または、実験環境の“Elements”リストからエレメントを選択します。
- ショートカットメニューから **Stimulate** を選択します。

または

- **Elements** → **Stimulate** を選択します。

  これにより、そのエレメント用のチャンネルが“Data Generator”ウィンドウに追加され、“Stimulus for:”ダイアログボックスが開きます。
- チャンネルのモードを選択します。このモードにより、生成されるカーブの種類が決まります。選択できるモードは、`constant`（定数）、`sinus`（正弦波）、`ramp`（ランプ）、`pulse`（パルス）、`step`（ステップ）、`table`（テーブル）、`matrix`（マトリックス）、`random`（乱数）、および `gaussian`（ガウス）です。

スティミュレーションモードを選択する：

- データジェネレータの各チャンネルごとにセットアップをおこないます。

注記

実験中にもシグナルチャンネルの設定を変更することができます。実験中に設定を変更して **Apply** をクリックすると、ダイアログは開いたまま変更内容が適用され、シグナルが変化します。

1. 定数モード（`constant`）

- “Stimulus for:”ダイアログボックスの“Value”フィールドに定数を入力します。

  他の設定フィールドはすべてオフにします。
- **OK** をクリックすると、値が確定されてウィンドウが閉じます。
- **Apply** をクリックしても値が確定されますが、ウィンドウは閉じません。
- **Cancel** をクリックすると、変更内容が取り消されてウィンドウが閉じます。

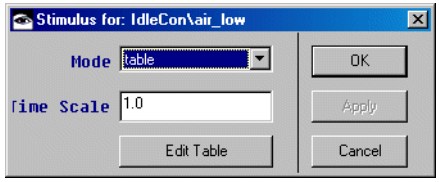
2. 周期モード（sinus、ramp、puls、step）およびランダムモード（random: 等分散）
   - “Frequency”フィールドに周波数を設定します。
   - “Phase”フィールドに波形の位相を設定します。

     ここに入力する値は、基準時間からのオフセットです。たとえば 0.5 秒に設定すると、シグナルの波形は 0.5 秒分だけ遅れて出力されます。

     注記

     “Frequency”および“Phase”フィールドは、random モードでは使用できません。
   - “Offset”フィールドにシグナル値のオフセットを入力します。
   - “Amplitude”フィールドに増幅を設定します。
   - **OK** または **Apply** をクリックします。
3. テーブルモード（table）

   table モードでは、テーブルに格納されたデータがスティミュラスシグナルとして使用されます。



   - Edit Table ボタンをクリックします。

     特性カーブ用のテーブルエディタが開きます。
   - テーブルエディタで、値を直接入力します。

   または

- **Edit** → **File In Data** メニューで既存のテーブルをロードします。

  テーブルエディタの使用方法は、451 ページの「集合型エレメント用エディタ（テーブルエディタ）」および 604 ページの「適合ウィンドウの使用方法」に詳しく説明されています。
- “Time Scale”フィールドに時間の係数を入力します。

  テーブルの X 軸の値にこの係数を掛けた値がスティミュレーションのタイムステップとなります。
- **OK** または **Apply** をクリックします。

注記

テーブルの場合、テーブルに設定された一連の値がすべて出力された時点で、スティミュレーションは終了します。再度スティミュレーションを行うには、実験を再スタートする必要があります。

4. マトリックスモード（`matrix`）
   - 564 ページの「インポートされたシグナルを実験に割り当てる：」以降の 2 つの項で説明されている方法で設定します。
5. ガウスモード（`gaussian`）
   このモードは、数値変数に対してのみ使用できます。ガウス分布に基づいて生成された乱数の値が使用されます。
   - “Mean”フィールドに、ガウス分布の平均値を入力します。
   - “Variance”フィールドにガウス分布の分散を入力します。
   - **OK** または **Apply** をクリックします。

チャンネルを削除する：

- 選択されたチャンネルをデータジェネレータから削除するには、**Channels** → **Delete** を選択します。
- すべてのチャンネルをデータジェネレータから削除するには、**Channels** → **Delete All** を選択します。

データジェネレータから供給されるスティミュラスシグナルとして、実際の測定データを使用することができます。これにより、コンポーネントのオフライン実験において、現実的なデータに基づくモデルの挙動を検証することができます。

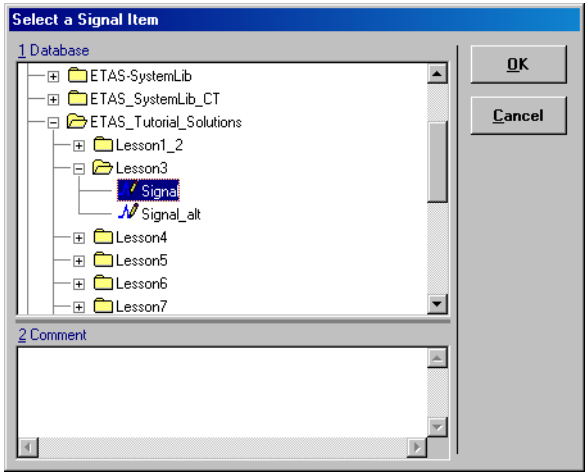
各種フォーマットのファイルに保存された測定データを、データベースの「シグナル」というアイテムインポートして使用します。シグナルにデータをインポートする方法については、541 ページの「シグナルとアイコン」の項を参照してください。1 つのシグナルに含まれる個々のチャンネルを、データジェネレータの別々のチャンネルに割り当てることができます。

インポートされたシグナルを実験で使用するには、シグナルを実験に割り当てる必要があります。この割り当ては、実験を開始する前に以下のように行ってください。

インポートされたシグナルを実験に割り当てる：

- データジェネレータで **Signal** → **Define Signal** を選択します。

  “Select a Signal Item”ダイアログボックスが開きます。



- “1 Database”ペインからインポート済みシグナルを選択します。
- **OK** をクリックします。

  これで、実験にシグナルが割り当てられました。各実験に割り当てられるシグナルアイテムは 1 つだけです。

インポートされたシグナルチャンネルをデータジェネレータのチャンネルに割り当てる：

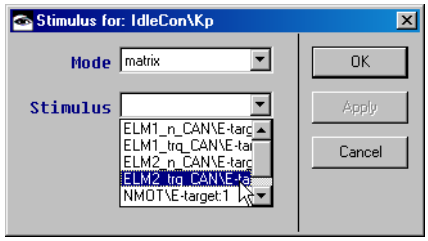
- データジェネレータウィンドウでチャンネルを選択します。

- **Channels → Edit** を選択します。

または

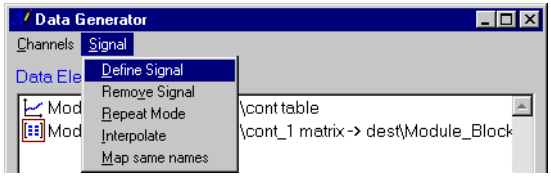
- チャンネルをダブルクリックします。
- “Stimulus for:” ダイアログボックスの Mode コンボボックスから、`matrix` を選択します。

  “Mode” コンボボックスの下に “Stimulus” コンボボックスが開き、現在の実験に割り当てられたインポート済みシグナルに含まれるシグナルチャンネルが一覧表示されます。



- データジェネレータのチャンネルに割り当てたいシグナルチャンネルを選択します。
- **OK** をクリックします。

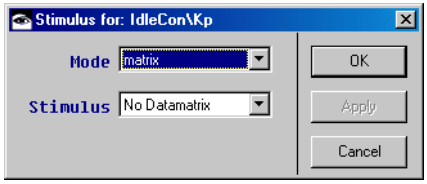
シグナルを実験に割り当てる（564 ページ参照）と、データジェネレータの **Signal** メニューに新しいメニューアイテムが追加されます。このメニューは実験が開始されていない時にのみ使用できます。



シグナルの割り当てを解除する：

- データジェネレータで **Signal → Remove Signal** を選択します。

  シグナルの割り当てが解除されます。そのシグナルでスティミュレートされていたチャンネルは `matrix` モードのままですが、“Stimulus”コンボボックスの設定内容はリセットされ、`No Datamatrix` と表示されます。



シグナルチャンネルとデータジェネレータチャンネルの名前が同じである場合、“Stimulus”コンボボックスで個々のチャンネルにシグナルチャンネルを選択する（564 ページ参照）必要はありません。同じ名前のチャンネルは自動的に割り当てることができます。

同じ名前のチャンネルを割り当てる：

- “Stimulus for:”ダイアログボックスで、シグナルチャンネルを自動的に割り当てるデータジェネレータチャンネルを `matrix` モードに設定します。
- データジェネレータで、**Signal → Map same names** を選択します。

  各データジェネレータチャンネルに、同じ名前のシグナルチャンネルがそれぞれ割り当てられます。

実験中にスティミュラスデータが生成されるたびに（554 ページ参照）、その値が評価されます。テーブルに定義された 2 点のブレイクポイントの間が実際のサンプリングポイントとなった場合、デフォルトでは低い方の値がチャンネルの値として有効になりますが、必要に応じて直線補間を行うこともできます。

シグナルの補間を行う：

- データジェネレータで **Signal → Interpolate** を選択します。
- 実験を開始します。

  サンプリングポイントを挟む 2 点のブレイクポイントの値が直線補間されてチャンネルに供給されます。

他に指定がなければ、シグナルが最後まで使用されると、実験（シミュレーション）はそのまま継続されますが、スティミュラスチャンネルは最後の値を保持したままとなります。これを、同じシグナルが繰り返し使用されるようにしたり（「繰り返しモード」）、またシグナル終了時に実験が自動的に終了するようにする（「自動終了モード」）ことができます。

注記

繰り返しモードと自動終了モードが両方とも有効になっている場合は、自動終了モードが優先され、シグナルが 1 回終了した時点で実験は終了します。

シグナルを繰り返しモードに設定する：

- データジェネレータで **Signal → Repeat Mode** を選択します。
- 実験を開始します。

  シグナルの最終ポイントに達すると、再度同じシグナルを使用してスティミュレーションが続行されます。

以下の操作は、最低 1 つのデータジェネレータチャンネルがシグナルによってスティミュレートされている場合に限って行えます。

自動終了モードに設定する：

- “Physical Experiment”ウィンドウで、**Experiment → Automatic Stop** を選択します。

  このメニュー項目にチェックマークが付きます。実験を開始すると、シグナルの最終ポイントに達した時点で実験（シミュレーションの実行）は自動的に終了します。
- **Experiment → Automatic Stop** をもう一度選択すると、自動終了モードが無効になります。

Experiment → Automatic Stop の設定は、実験が終了して、実験環境を閉じた後も保持され、次に同じコンポーネントの実験を行うと、同じ設定が有効になります。

6.1.5 測定システム

実験環境に組み込まれている「測定システム」を利用して、オシロスコープ、棒グラフディスプレイや数値ディスプレイなど、さまざまな形式でエレメント（「測定変数」）の測定値を測定ウィンドウに表示することができます。各表示形式にはさまざまな表示オプションが用意されていて、実験のセットアップ時だけでなく実験の実施中にもその設定を変更することができます。新しい測定ウィンドウを作成するには、実験環境ウィンドウにおいて、測定したいエレメントを任意のタイプの測定ウィンドウに割り当てます。測定ウィンドウの種類と機能の詳細については、631 ページの「測定ウィンドウ」の項に詳しく説明されています。

エレメントを新しい測定ウィンドウに割り当てる：

- エレメントを割り当てたい測定ウィンドウのタイプを、“Measurement Window”コンボボックスから選択します。

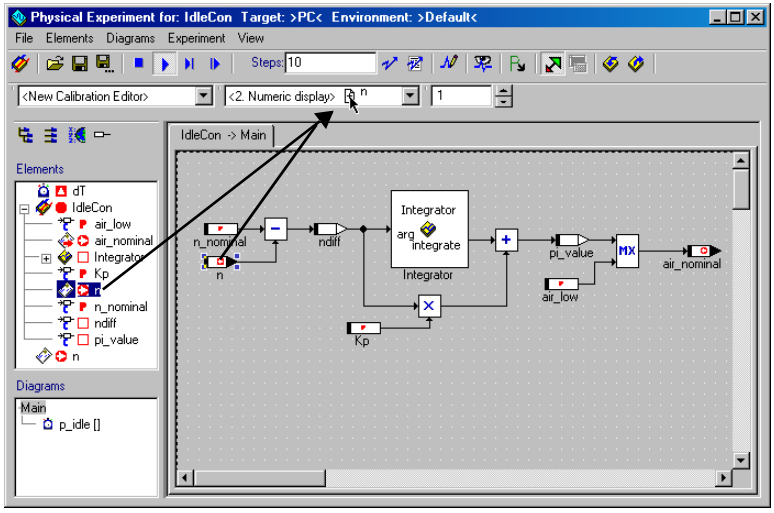
  選択可能なタイプは、たとえば <1. Oscilloscope> のように、< > に囲まれてリストアップされています。
- “Elements”ペインから、測定ウィンドウに割り当てたいエレメントを選択します。
- **Elements → Measure** を選択します。

または

- 割り当てたいエレメントを“Elements”ペインから“Measurement Window”コンボボックスにドラッグします。

または

- ブロックダイアグラム上に表示されているエレメントのオカレンスを“Measurement Window”コンボボックスにドラッグします。



新しい測定ウィンドウが開き、選択されたエレメントがそのウィンドウ上に表示されます。**<Ctrl>** キーを押したまま複数のエレメントをクリックすれば、選択された複数のエレメントを同時に同じウィンドウに割り当てることができます。

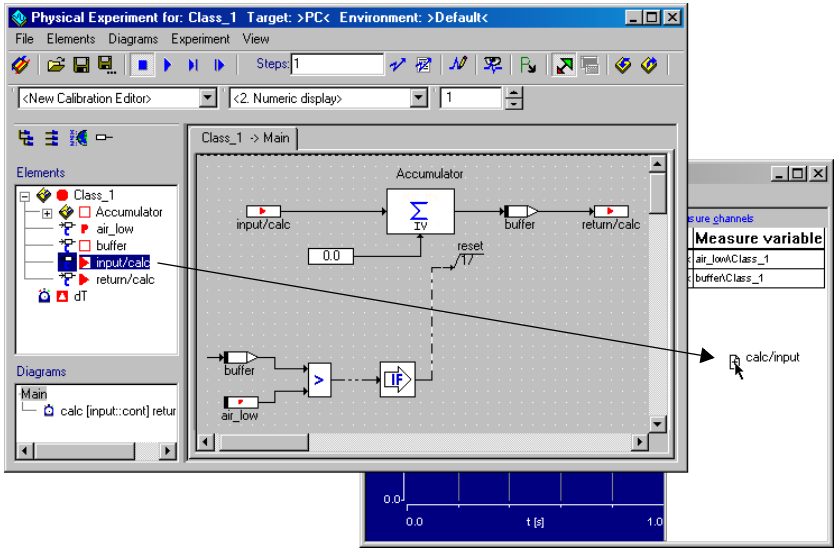
“Measurement Window”コンボボックスには、すでに開いている測定ウィンドウのタイトルも表示されます。開いているウィンドウのタイトルは、たとえば `Oscilloscope; 1` のように、角かっこ（< >）なしで表示されます。測定ウィンドウのタイトルを変更した場合には、このコンボボックスに表示される名前を変わります。

エレメントを既存の測定ウィンドウに割り当てる：

- 既存の測定ウィンドウを、“Measurement Window”コンボボックスから選択します。
- 1 つまたは複数のエレメントを“Measurement Window”コンボボックスにドラッグします。

または

- 1 つまたは複数のエレメントを測定ウィンドウにドラッグします。



以下のようにしてモニタ表示を有効にすると、描画領域上のオカレンスの上に、そのエレメントの現在値が表示されます。この機能の詳細は、659 ページの「モニタ」という項を参照してください。

モニタ機能をエレメントに割り当てる：

- 描画領域の、エレメントのオカレンスを右クリックします。

- ショートカットメニューから **Monitor** を選択します。

  エレメントの現在の値がコンポーネント表示部上に表示されます。

すべての測定ウィンドウを閉じる：

- **Experiment → Close Measurement Windows** を選択して、現在開いているすべての測定ウィンドウを閉じます。

6.1.6 適合システム

適合システムは、オンライン／オフライン実験で同じように機能します。ツールバーの“Calibration Window”コンボボックスには現在開いているすべての適合ウィンドウがリストアップされ、ここで新しい適合ウィンドウを開くこともできます。1 つのウィンドウに同じ型の複数のエレメントを割り当てることができます。つまり、1 つのテーブルエディタに複数のテーブルを割り当てたり、1 つの数値エディタに複数のスカラエレメントを割り当てることが可能です。割り当てる方法は、測定ウィンドウの場合と同様です。適合ウィンドウの種類や機能の詳細は、599 ページの「適合ウィンドウ（適合エディタ）」を参照してください。

エレメントを適合する：

- 実験環境の“Elements”ペインから、適合を行うエレメントを選択し、いずれかの操作を行います。

  — **Elements → Calibrate** を選択します。

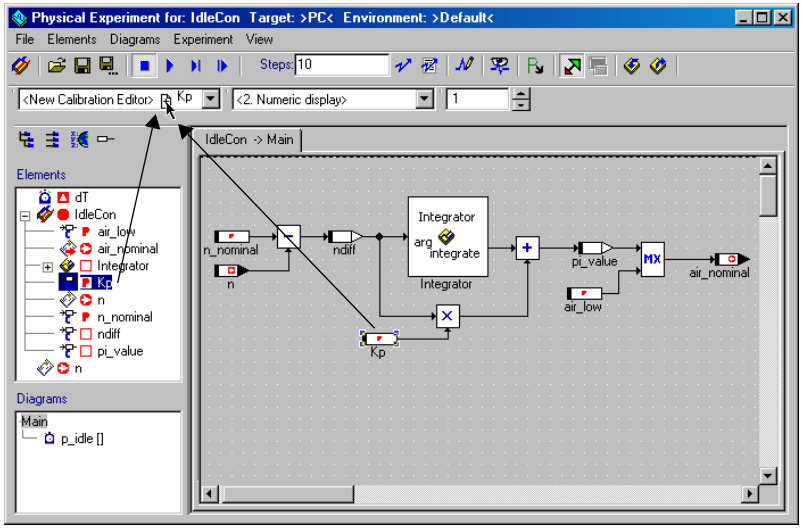
  または

  — エレメントをダブルクリックします。

  エレメントの型に応じたデータエディタが開きます。

または

- 適合したいエレメントを、“Elements”ペインまたは描画領域からドラッグし、“Calibration Window”コンボボックスにドロップします。



<New Calibration Window> というエントリにエレメントをドラッグすると、新しいウィンドウが開き、既存のウィンドウ名にドラッグすると、エレメントはそのウィンドウに追加されます。またエレメントを既存の適合ウィンドウに直接ドラッグすることもできます。

- すべての適合ウィンドウの内容を更新するには、**Experiment → Update Calibration Windows** を選択します。
- すべての適合ウィンドウを閉じるには、**Experiment → Close Calibration Windows** を選択します。

6.1.7 オフライン実験の実行

セットアップが終了したら、実験を開始します。オフライン実験が実行されている間も、測定ウィンドウの表示オプションの変更、測定ウィンドウの追加／削除、データジェネレータやイベントジェネレータの設定変更、適合システムを用いたデータ値の変更などを行えます。

オフライン実験を開始する：

- 実験の対象とするコンポーネントまたはプロジェクトの実験環境を開きます。

- “Measurement Rate”ボックスでサンプリングレートを調整します。

  レートが 1 に設定されていると、すべてのサンプル値が表示されます。それより大きな値が設定されていると、その値の倍数にあたる回の値のみ表示されます。たとえば、10 が設定されていると、測定値は 10 回のサンプリングごとに 1 回だけ表示されます。

- 実験環境のコンポーネント表示を隠すには、**Expand/Collapse Window** ボタンをクリックします。

  実験環境のメニューバーとツールバーだけが表示される状態となります。このボタンをもう一度クリックすると、コンポーネントが再び表示されます。

  また、左の **Always on top** ボタンが有効になります。

- **Always on top** ボタンをクリックすると、“Physical Experiment”ウィンドウが常に最前面に表示されます。

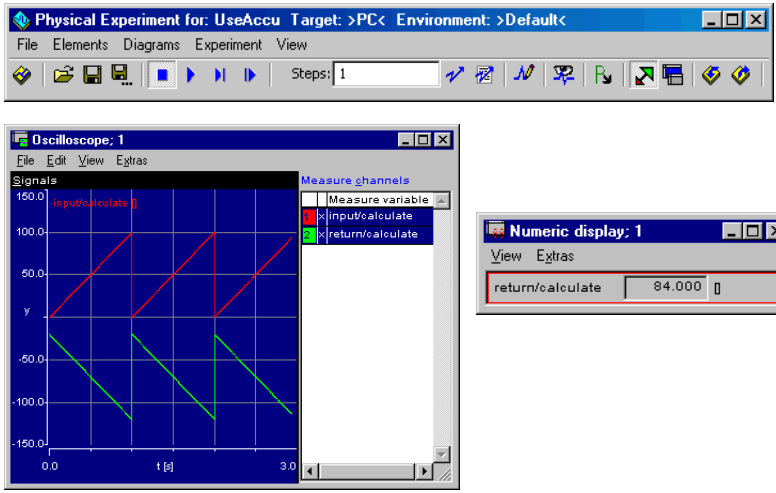
- **Experiment → Start Experiment** を選択します。

または

- **Start Offline Experiment** ボタンをクリックします。

  ASCET オプションウィンドウの“Experiment”ノードでの設定に応じて（59 ページの「実験オプション」を参照してください）、変数とパラメータが初期化されます。

実験（シミュレーション）が開始されて測定システムが稼動し、各測定ウィンドウに、割り当てられたエレメントの値が表示されます。



- データジェネレータ、イベントジェネレータ、測定システム、または適合システムを必要に応じて調整しながら実験を進めます。

オフライン実験を終了する：

- **Experiment** → **Stop Experiment** を選択します。

または

- **Stop Offline Experiment** ボタンをクリックします。

  実験は終了しますが、すべての設定は有効のままです。測定データはオシロスコープウィンドウに残っているので、データを分析することができます（646 ページの「分析モードに切り替える：」という項を参照してください）。実験を再開すると、時間軸の値は 0 にリセットされます。またオプション設定に応じてパラメータと変数が初期化されます（59 ページ参照）。

実験環境を閉じる：

- **Exit to Component** ボタンをクリックします。

または

- **File** → **Exit** を選択します。

  実験環境が閉じ、シミュレートしていたコンポーネントのエディタに戻ります。

  この際、現在の実験環境を保存するかどうかを尋ねる“Save Experiment”ダイアログボックスが開きます。
- 今後、同じタイプの質問に毎回同じ回答を行いたい場合は、**Remember my Decision** オプションをオンにします。

  上記の設定を行うと、以後“Save Experiment”ダイアログボックスは開かなくなります。この設定は、ASCET オプションウィンドウの“Confirmation Dialogs”ノードで確認／変更できます（46 ページの「確認ダイアログボックスに関するオプション」参照）。
- 保存を行うには **OK** をクリックします。

  実験環境が所定の名前で保存され、コンポーネントのエディタに戻ります。

または

- 保存を行わない場合は **No** をクリックします。

  実験環境が所定の名前で保存され、コンポーネントのエディタに戻ります。

または

- 実験環境を閉じる操作を取り消すには、**Cancel** をクリックします。

以下のようにして、実験を任意のステップ単位で実行することができます。各ステップでは、優先度に従って 1 つのイベントが起動されます。

実験をステップ実行する：

- **Experiment** → **Pause Experiment** を選択します。

または

- **Pause Offline Experiment** ボタンをクリックします。

  実験がポーズ状態になります。つまり、シミュレーションの実行は停止しますが、各設定はすべて有効で、データも保持されます。実験を再開すると、停止した時点の状態からシミュレーションの実行が再開されます。

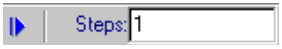
- **Experiment → Stop Experiment** を選択します。

または

- **Step Offline Experiment** ボタンをクリックします。

  実験がシングルステップモードで実行されます。

  ボタンをクリックするたびに 1 ステップずつシミュレーションが実行され、再びポーズ状態になります。また実験を開始する前でもこのステップ実行を行うことができ、その場合、実験の状態は「非稼動状態」から「ポーズ 状態」に自動的に変化します。



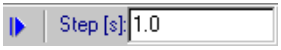
- 実験をステップ実行する前に、実行するステップ数を“Steps”ボックスで調整することができます。

  たとえば、ステップ数を 5 にすると、5 ステップが実行されます。

現バージョンの ASCET では、3 種類のステップモードから選択することができます。ステップ実行を行うと、選択されたモードに応じて、指定のステップ数または指定の秒数だけ、またはブレークポイントに到達するまで実行されます。

タイムステップモードに切り替える：

- 実験環境で、**Experiment → Event Generator → Step Mode → Timed [s]** を選択して、タイムステップモードに入ります。



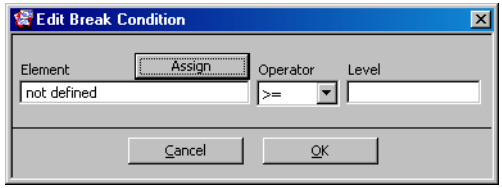
  ツールバーに、1 ステップ分の時間（秒数）を入力するためのテキストフィールドが表示されます。秒数を指定してから、上記の手順で実験をステップ実行します。

3 つめのステップモード、つまりブレークポイント条件を使用して実験をステップ実行する方法を利用すれば、コンポーネント内のエレメントの条件（例、「回転数が 5,000 より大きい」）を指定することができます。実験はブレークポイント条件が真になるまで実行されます。

ブレークポイント条件をセットアップする：

- 実験環境で **Experiment → Event Generator → Step Mode → Break At Condition** を選択して、ブレークポイントステップモードに入ります。

  “Edit Breakpoint Condition”ダイアログボックスが開きます。このダイアログボックスで、どの時点まで実験を実行されるか、という条件を指定します。



  このエディタは、後で **Experiment → Event Generator → Edit Break condition** コマンドで開くとこができます。

- **Assign** ボタンをクリックして、条件に使用するエレメントの名前を選択します。

  選択ダイアログボックスには、エレメントが一覧表示されます。

- 希望のエレメントを選択して **OK** をクリックし、そのエレメント名を使用します。
- リストから演算子を選択してオペランドの値を入力し、条件を完成させます。
- **OK** をクリックして確定します。

  ツールバーにブレーク条件を示すテキストフィールドが表示されます。前述の手順で、実験をステップ実行します。

実験の実行中／停止中に関わらず、実験の対象となっているコンポーネント、およびその中に含まれるエレメントのインプリメンテーションを表示することができます。

インプリメンテーションを表示する：

- “Elements”リストで、インプリメンテーションを表示したいエレメントまたはコンポーネントを表示します。

- **Elements → Show Implementation** を選択します。

注記

このコマンドは、2 つ以上のエレメントが選択されている場合は使用できません。

選択されたオブジェクトのインプリメンテーションエディタが開きます。このエディタの使用方法は、4.12「インプリメンテーションの編集」を参照してください。ただしここでは内容の変更は行えないため、**OK** ボタンは無効になっています。

C コードで記述されているコンポーネントについては、デバッグ情報やエラーメッセージを C コードに組み込んでおき、実験中にそれらのメッセージを画面に表示させることができます。デバッグ情報は、実験中に開くことのできる「ターゲットデバッグビューア」に表示されます。エラーメッセージは ASCET モニタウィンドウに出力されます。

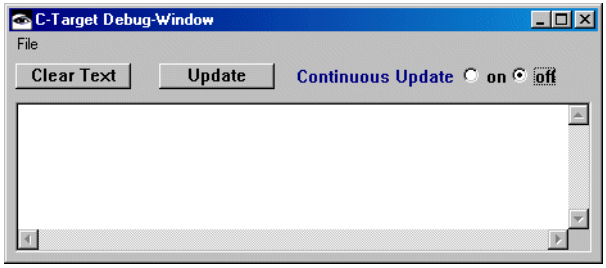
実験中に任意の情報を表示するには、C コード内に `asdWriteUserDebug()` および `asdWriteUserError()` という関数を組み込みます。どちらの関数も、表示されるメッセージを含む文字列を引数として受け取ります。典型的なコードは、以下のようになります。

```
asdWriteUserError("Overflow: ¥n Upper Limit exceeded.")
```

文字列引数は標準の ANSI C の規則に従って扱われるため、上の例は 2 行のメッセージとして出力されます。

デバッグ情報を表示する：

- **Experiment → Open Target Debugger** を選択し、“C-Target Debug-Window” を開きます。



1. 自動表示

- “C-Target Debug-Window” ウィンドウで、**on** オプションをオンにします。

または

- **File → Continuous Update** メニューをオンにします。

  生成されたデバッガ情報が直ちに表示され、継続的に更新されます。

2. マニュアル表示

- **off** オプションをオンにします。

または

- **File → Continuous Update** メニューをオフにします。

  自動更新が行われなくなり、更新操作を行った時にのみ新しいデバッグ情報が表示されるようになります。

- 最新の情報を表示するには、**Update** ボタンをクリックします。

または

- **File → Update Text** を選択します。

3. テキストウィンドウのクリア

- **Clear Text** ボタンをクリックします。

または

- **File → Clear Text** を選択します。

4. テキストウィンドウの保存

- **File → Save as** を選択します。

  ファイル選択ダイアログボックスが開きます。

- パスとファイル名を指定します。
- **Save** をクリックします。

  デバッガウィンドウの内容が、指定のファイルに書き込まれます。

「イベントトレーサ」は、どのイベントがどのような順序で発生したかを監視するためのものです。いずれかのイベントが発生すると、そのポイントの時間が秒単位で表示され、同じポイントにおいて複数のイベントが発生した場合は、発生順序に基き各イベントが一覧表示されます。

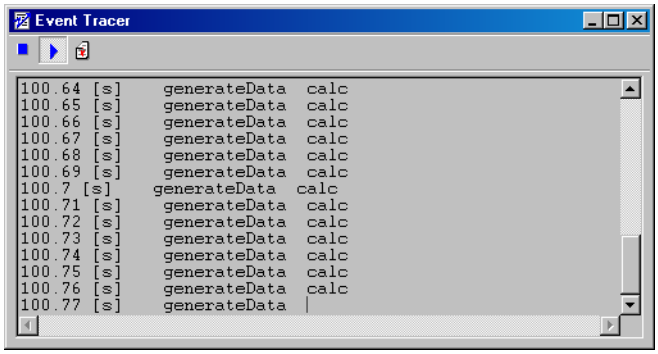
イベントトレーサでイベントを監視する：

- 任意のコンポーネントの実験をセットアップします。

- **Open Event Tracer** ボタンをクリックします。

または

- **Experiment** → **Event Tracer** を選択して、“Event Tracer” ウィンドウを開きます。



- **Start Tracer** ボタンをクリックして、イベントトレーサを起動します。
- 実験を開始します。

- イベントのトレースを停止するには、**Stop Tracer** ボタンをクリックします。

  実験の実行中に、任意にイベントのトレースを開始／停止することができます。

- “Event Tracer” ウィンドウの表示内容をクリアするには、**Reset Tracer** ボタンをクリックします。

6.1.8 環境設定のロードと保存

複雑なモデルの実験において多くの測定／適合ウィンドウや測定チャンネルが使用さる場合、実験環境の設定には手間がかかります。そのような場合、1 つの実験の環境設定を保存し、別の実験セッションでその設定を再利用して環境設定の手間を省くことができます。環境設定は、各コンポーネントごとに複数保存することができます。

環境設定を保存すると、データジェネレータ、イベントジェネレータ、およびデータロガーのすべての設定が保存され、さらに、開いているすべての適合ウィンドウや測定ウィンドウが、それぞれの設定を保ったままで保存されます。後に、実験環境を開く時に任意の環境設定をロードすると、その設定が保存された時と同じ環境が復元されます。保存されている環境設定がない場合は、実験環境はデフォルト設定で開きます。

環境設定を保存する：

- 実験環境を開き、必要な調整を行います。

- **Save Environment** ボタンをクリックします。

または

- **File → Save Environment** を選択して、現在の環境設定を保存します。

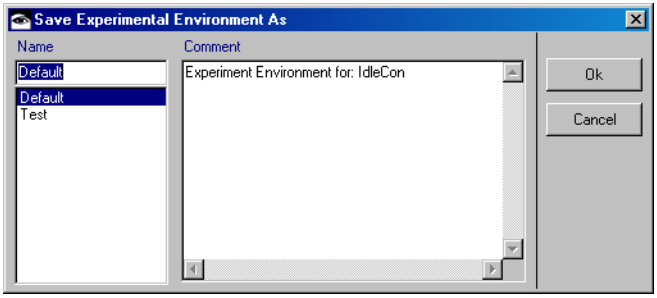
環境設定を別の名前で保存する：

- **Save Environment As** ボタンをクリックします。

または

- **File → Save Environment As** を選択します。

  “Save Environment As”ダイアログボックスが開きます。



- 環境設定の名前を入力します。
- “Comment”ペインに、実験の環境について説明するコメントを入力します。
- **OK** をクリックします。

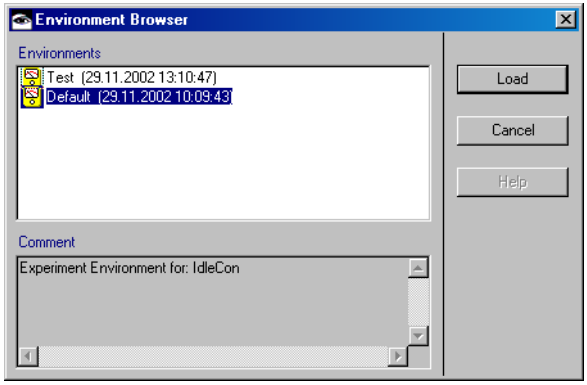
  実験の環境設定が、入力された名前で格納されます。

環境設定をロードする：

- 実験環境を開きます。

  デフォルトの環境設定以外に環境設定が保存されていない場合、実験環境を開くとそのデフォルト環境が自動的にロードされます。コンポーネントについて複数の環境設定が保存されている場合に

は、“Environment Browser”ダイアログボックスが開き、使用可能なすべての環境設定がリストアップされます。



- ロードしたい環境設定を選択します。
- **Load** をクリックします。

  選択された環境設定で実験環境が開きます。

実験環境がすでに開いた状態で、他の環境設定をロードすることができます。その場合、現在開いている測定／適合ウィンドウは開いたままですが、ロードされた環境設定に格納されていたその他の設定が有効になり、そこに保存されているすべてのウィンドウが開きます。

別の環境設定に切り替える：

- **Load Environment** ボタンをクリックします。

または

- **File → Load Environment** を選択します。

  “Browse Environment”ダイアログボックスが開きます。
- 使用したい環境設定を選択します。
- **Load** をクリックします。

環境設定をエクスポートする：

- **File → Export Experiment** を選択します。

  “Export File”ダイアログボックスが開きます。
- エクスポートファイルのパスとファイル名を入力します。

- **OK** をクリックします。

  現在使用されている環境設定がエクスポートされます。環境設定をインポートするには、コンポーネントマネージャのインポート機能を使用します。環境設定のインポートは、他のデータのインポートと同じ方法で行います。

6.1.9 データロガー

「データロガー」は、オフライン実験においてはコンポーネントまたはプロジェクトの変数、またオンライン実験においてプロジェクトの変数の値をロギングします。値はファイルに書き込まれ、測定データ分析ツール（「MDA」など）を用いて分析することができます。次の 3 種類のロギングモードがあります。

- 過渡サンプリング（Transient Sampling）

  「過渡サンプリング」モードを使用するには、コード生成オプションを調整し、ロギング対象とする変数の値が変わるたびにその変化を自動的に記録するためのコードが生成されるようにする必要があります。これによって、1 つの変数のすべての変化をロギングすることができますが、このためのコードが追加されるため、モデルのリアルタイムな挙動に影響します。またこのモードでは、ロギング対象の各変数ごとにタイムスタンプが生成されるため、ログデータのデータ量が多くなります。

- 周期サンプリング（Periodic Sampling）

  「周期サンプリング」モードの場合、コードの変更は必要ないため、モデルのリアルタイムな挙動に及ぶ影響は最小限で済みます。ロギング処理は特定のタスクにより起動されます。つまり、そのタスクが起動されるたびに、ロギング対象の全チャンネルの値が記録されます。このロギングはタスクが終了してから行われるので、タスクのリアルタイムな挙動に影響しません。1 回のロギングが行われてから次にロギングが行われるまでの間に、変数の値が複数回変わった場合は、最後の変化だけが記録されます。このモードでは、あらかじめ定義された固定のインターバルで行われるので、タイムスタンプは必要ありません。

  実験ターゲットを使用している場合、ロギングされたデータはターゲット内のリングバッファに格納され、記録が終了した時点でホスト PC 上の別のリングバッファに書き込まれます。ターゲット上のリングバッファは、あらかじめ定義された数の値を常に保存していて、その数を超えると、現在格納されている値は、先入れ先出しの原則で上書きされます。つまり 1 つのデータは限られた時間しか保存されないため、データが失われるのを防ぐには、「連続ポーリング」を行う必要があります。このモードにおいては、ロギングを行っている間、データは定期的に PC に転送されます。ただし、PC との頻繁な通信によってターゲットプロセッサの処理に影響が出る場合があるため、注意が必要です。

- 周期保存（Periodic to File）

  長時間の記録が必要で、ターゲット上の記録用 RAM の量が不十分な場合、「周期保存」モードを使用します。このモードでは、ロギングの開始は周期サンプリングの場合と同様に特定のタスクによって開始されますが、ロギングが行われている間、データは一定の間隔でホスト PC に送信されるため、データが失われることはありません。

ロギング処理が終了すると、上記の 3 通りのいずれのロギングモードにおいても、PC に集められたデータはファイルに保存されます（592 ページ参照）。

ロギングできるチャンネル数と、データを記録するレートは、以下の 3 つの要因により制約されます。

- ロギングされるデータはターゲット上の RAM の一部に格納されるため、ターゲットの RAM 容量が大きいほど、より多くの値をロギングすることができます。
- PC とターゲットの間のデータ転送レート が、データのロギングに影響します。たとえば、ターゲット上の RAM 容量が少ない場合でも、データを PC に高速転送することにより、より多くの値を記録することができます。
- ロギングされた値は PC 上の RAM（物理および仮想）に格納されるため、PC の RAM の空き容量が大きいほど、より多くのデータを記録することができます。

過渡サンプリングモードでデータをロギングするためには、コード生成オプションを変更する必要があります。

注記

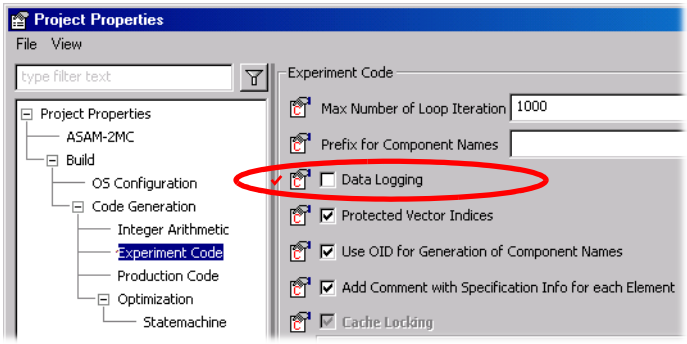
過渡サンプリングモードは、プロジェクト単位でしか使用できません。

過渡サンプリングの準備を行う：

- 実験を行うプロジェクトを開きます。
- プロジェクトエディタで、**Project Properties** ボタンをクリックします。

  “Project Properties”ウィンドウの“Build”ノードが開きます。
- "Experiment Code”ノードを開きます。

- データロギングを有効にするための **Data Logging** オプションをオンにします。



- **OK** をクリックします。
- プロジェクトのコードを生成し、実験環境を開きます。

周期サンプリングモードまたは 周期保存モード でロギングを行う際は、上記の設定を行う必要はありません。コード生成時において、データロギングオプションはどちらの設定（有効／無効）になっていてもかまいませんが、有効になっている状態でコード生成を行うと、実際にデータロギングを行わなくても、コードの実行速度は遅くなります。

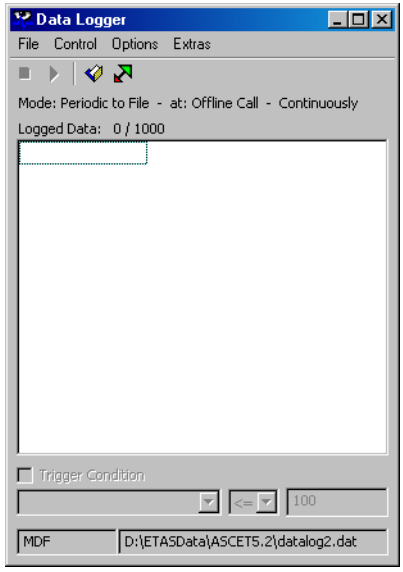
データロガーを開く：

- コンポーネントまたはプロジェクトの実験環境を開きます。

- **ASCET Data Logger** ボタンをクリックします。

または

- **Experiment → Data Logger** を選択します。
  “Data Logger” ウィンドウが開きます。



このダイアログボックスには以下のエレメントが含まれています。

- **File** メニュー
  - *Open* （**<Ctrl> + <O>**）

    ロギングされるエレメントのリスト（INCA 用 `*.lab` フォーマット）を開きます（587 ページ参照）。
  - *Save* （**<Ctrl> + <S>**）

    ロギングされるエレメントのリスト（INCA 用 `*.lab` フォーマット）を保存します（587 ページ参照）。
  - *Save As* （**<Ctrl> + <S>**）

    ロギングされるエレメントのリスト（INCA 用 `*.lab` フォーマット）を任意の名前で保存します（587 ページ参照）。
  - *Import*

    ロギングされるエレメントのリスト（`*.csv`、または SelectX - `*.cfg` - フォーマット）を任意の名前で保存します（587 ページ参照）。
  - *Exit*

    データロガーを閉じます。

- **Control** メニュー
  - — *Enable Logging*

    ロギングを開始します。
  - — *Disable Logging*

    ロギングを終了します。
- **Option** メニュー
  - — *Logging Options*

    “Logging Options”ダイアログボックスを開きます（588 ページ参照）。
  - — *Change Filename*

    ログファイルのデフォルト名を変更します。
- **Extras** メニュー
  - — *Convert MDF to FAMOS*

    MDF フォーマットのデータファイルを FAMOS フォーマットに変換します（593 ページ参照）。
- ツールバー

   ロギングの終了

   ロギングの開始

   ログファイルのデフォルト名の変更

   メニューバーとツールバーの下の部分の表示／非表示切り替え

- “Logged Data”フィールド

  ロギングされるエレメントのリストが表示されます。
- **Trigger Condition** オプション

  トリガ条件の有効／無効を切り替えます。
- トリガ条件定義フィールド

  2 つのコンボボックスと 1 つの入力フィールドで、トリガ条件を定義します（591 ページ参照）。
- ステータスバー

  データロガーのステータス情報が表示されます。

ロギングするチャンネルをセットアップする：

- 実験環境の“Elements”ペインから、ロギングの対象にしたいエレメントを選択します。

- **Elements** → **Log** を選択して、ロギングを有効にします。

  まだデータロガーが開いていない場合は、ここで開きます。選択されたエレメントが“Logged Data”フィールドに表示されます。
- エレメントは、データロガーに直接ドラッグすることもできます。
- **Elements** → **Log All** を選択すると、全エレメントについてのロギングが行われます。

注記

ロギングを行えるエレメントの数は、いくつかの要因によって制限されます。583 ページを参照してください。

データロガーを閉じると、登録されたエレメントについての情報は失われます。次回の実験での設定作業を容易にするためには、現在データロガーに登録されているエレメントのリストをラベルリスト（`*.lab`）として保存しておくことをおすすめします。このラベルリストは、INCA と互換であるため、同じエレメントが両ツールで使用されているかを確認したりする際に利用できます。

ラベルリストを使用する：

- データロガーを開きます（584 ページ参照）。
- ロギングを行うチャンネルを設定します（586 ページ）。
- **File** → **Save** を選択して、エレメントリストを保存します。

  初回の保存時には、パスとファイル名を尋ねられます。
- エレメントリストを任意の名前で保存するには、**File** → **Save As** を選択します。
- データロガーにエレメントリストを読み込むには、**File** → **Open** を選択します。

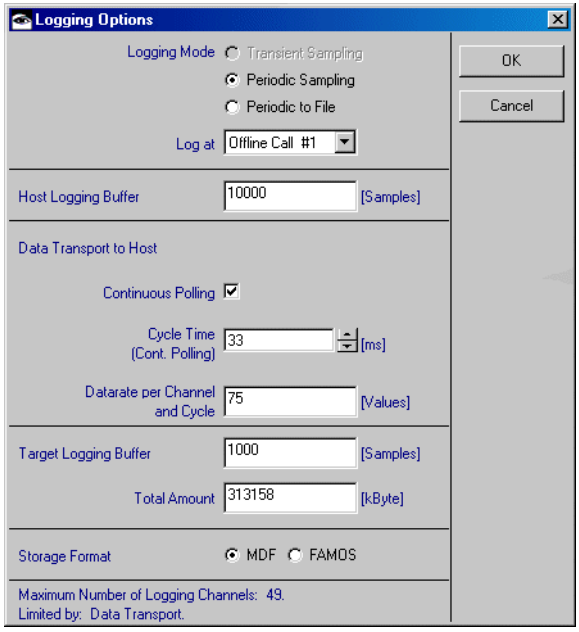
  読み込まれたリスト内に、現在の実験に含まれていないエレメントが含まれていた場合、モニタウィンドウにワーニングメッセージが出力されます。
- `*.csv`（**c**omma-**s**eparated **v**alues）または `*.cfg`（SelectX フォーマット）からリストをインポートするには、**File** → **Import** を選択します。

  ASCET 外部でラベルリストを作成／管理するには、これらのフォーマットを使用してください。

ロギングオプションを調整する：

1. ロギングモード

   - データロガーで **Options** → **Logging Options** を選択して、“Logging Options”ダイアログボックスを開きます。



   - **Transient Sampling**（過渡サンプリング）、**Periodic Sampling**（周期サンプリング）、**Periodic to File**（周期保存）のいずれかを選択します。

     コード生成オプションでデータロギングが有効になっている場合には、**Transient Sampling** しか選択できません（583 ページ参照）。

- “Log at”コンボボックスからタスクを選択します。

注記

“Log at”コンボボックスは、**Transient Sampling** が選択されている場合は表示されません。

オンライン実験の場合、このコンボボックスにはプロジェクト内に定義されているすべてのタスクが表示されます。ここで選択されたタスクが起動されるたびにロギングが行われます。

オフライン実験の場合、イベントジェネレータで定義された各イベントの発生ごとにロギングが行われるので、ここでは何も選択する必要はありません。

- “Host Logging Buffer”フィールドに、PC のロギングバッファに保存するサンプル数を入力します。

注記

“Host Logging Buffer”フィールドは、**Periodic to File** が選択されている場合は入力できません。

この設定により、各チャンネルにつき何個の値が PC 上のリングバッファに格納されるかが決まります。

2. PC へのデータ転送

- **Continuous Polling**（連続ポーリング）チェックボックスを設定します。

注記

Continuous Polling は、**Periodic to File** が選択されると自動的に有効になります。

Continuous Polling が行われない時は、データはターゲットバッファ上に記録され、ロギング終了時にのみ PC 上で保存されます。

- “Cycle Time”フィールドで、連続ポーリングの周期を調整します。

注記

“Cycle Time (Cont. Polling)”フィールドは、**Continuous Polling** が有効な場合にのみ入力できます。

Continuous Polling がオンになっていると、ロギングされてターゲット上のリングバッファに格納されたデータは、“Cycle Time”フィールドで指定された周期ごとに PC 上で転送されて保存されます。

転送に時間がかかり過ぎて設定された周期が守られない場合、データロガーウィンドウのヘッダ部分にその旨が表示されます。これには、CPU の処理速度、ターゲット上の負荷、通信状態などが大きく影響するため、最適な設定を行う必要があります。

- “Data Rate per Channel and Cycle”フィールドに、転送レートを入力します

  この値によって、1 回のポーリングにおいてチャンネルあたりいくつのデータを読み取るかが決定されます。

  Continuous Polling がオフになっていると、ポーリングの周期はロギングが行われる総時間に応じて決まり、**Continuous Polling** がオンになっていると、“Cycle Time (Cont. Polling)”フィールドで指定された周期が使用されます。

3. ターゲットロギングバッファ

- “Target Logging Buffer”と“Total Amount”フィールドで、ターゲット上のロギングバッファの使用量を調整します。

  ターゲットの RAM 領域は、ロギングデータ用バッファとして最大限使用できるようになっています。この割合を少なくするには、この設定を調整してください。この設定により、1 チャンネルあたり何個のデータをターゲットリング上のリングバッファに格納するかが決まります。

4. 出力フォーマット

- **FAMOS** または **MDF** のいずれかのオプションボタンをクリックして、ログファイルのフォーマットを選択します。

- **OK** をクリックします。

このダイアログボックスでの設定に応じて、一度にロギングできるチャンネル数が決まります。この数はダイアログボックスの一番下に表示され、その下の行に制約理由も表示されます。この表示はダイアログボックス内の設定内容が変更されるたびに更新されます。ユーザーが入力した値が他の設定または使用可能なリソースにより設定された限界値を超える場合には、エラーメッセージが表示され、設定は変更されません。

データをロギングする：

- 実験を開始します。
- データロガーで、**Enable Logging** ボタンをクリックします。

または

- **Control → Enable Logging** を選択します。

選択されたエレメントのロギングが開始されます。

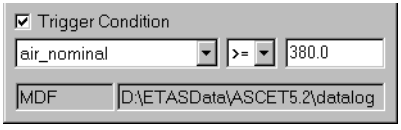
実験を開始する前にこのコマンドを選択してロギング処理を開始することはできますが、実際にデータがロギングされるのは、実験が実行されている間だけです。

上記の方法で、実験開始後、直ちにロギングを開始することができます。また **Trigger Condition** オプションによって、データロギング開始条件を設定することもできます。以下のように行ってください。

トリガ条件を設定する：

- データロガーで、**Trigger Condition** オプションをオンにします。

このオプションは、ロギングが開始されていない時にのみ設定できます。このとき、“Logged Data”フィールドに 1 つ以上のエレメントを指定しておいてください。

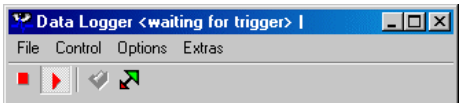


- 左側のコンボボックスで、ロギングする変数の中から 1 つ（上図では、`air_nominal`）を選択します。
- 中央のコンボボックスで、比較演算子を選択します。

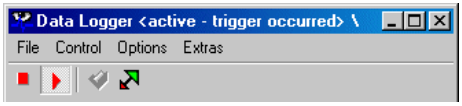
>= または <= を選択できます。

- 右側のフィールドで、しきい値（例：380）を入力します。

上記の設定を行った後にデータロギングを開始すると、設定されたトリガ条件が満たされるまで、実際のロギングは行われません。データロガーのヘッダ部に、トリガの状態が表示されます。



条件が満たされると実際にロギングが開始され、ヘッダ部の状態が更新されます。この後は、再度条件が満たされない状態になっても、データロギングはユーザーが終了するまで継続されます。



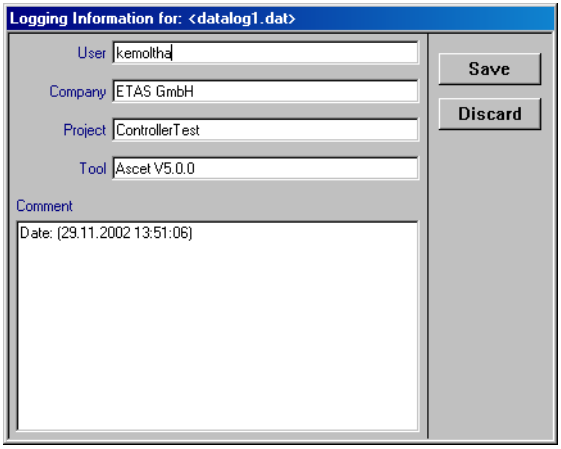
ロギングを終了する：

- **Disable Logging** ボタンをクリックします。

または

- **Control → Disable Logging** を選択します。

データロギングが終了し、“Logging Information” ダイアログボックスが開きます。



- 必要に応じて、情報フィールドにデータを入力します。

  このダイアログボックスの情報は、必ずしも入力する必要はありません。
- **Save** ボタンをクリックしてファイルを保存します。

または

- **Discard** ボタンをクリックしてデータを破棄します。

ロギング処理が終了されるたびに、データがログファイルに保存されます。ASCET オプションウィンドウの“Datalogger”ノードに含まれる **Auto Increment DataLogger File Name** オプションがオンになっていると、ファイル名は、`datalog<n>`（<n> は整数で、データが保存されるたびにインクリメントされます）となります（58 ページ参照）。このオプションをオフにしておくと、ロギングを繰り返し行うことによってログファイルは上書きされます。

ログファイルは、ASCET のデータディレクトリ（例： `ETASData¥Ascet5.2`）内に保存されます。

ログファイルのデフォルト名とパスは、以下のようにして変更できます。

ログファイルを変更する：

- **Change Filename** ボタンをクリックします。

または

- **Options → Change Filename** を選択します。

  “File Selection”ダイアログボックスが開きます。
- 新しいパスおよびファイル名を設定します。
- **OK** をクリックして、ログファイルを変更します。

  データロガーウィンドウの一番下に表示されるファイル名が変わり、データはここに表示される新しいファイルに書き込まれるようになります。

  ASCET オプションウィンドウで所定のオプションがオンになっていると（58 ページ参照）、指定されたファイル名に整数の番号が付加され、この番号はデータが保存されるたびにインクリメントされます。

MDF データを FAMOS フォーマットに変換する：

- **Extras → Convert MDF to FAMOS** を選択します。

  入力ファイルを選択するためのファイル選択ダイアログボックスが開きます。

- 変換したい MDF ファイルを選択します。

  FAMOS 出力ファイルを選択するためのファイル選択ダイアログボックスが開きます。
- FAMOSファイルのパスおよびファイル名を選択します。

  MDF ファイルが読み取られて FAMOS に変換されてから、選択された出力ファイルに書き込まれます。

6.1.10 実験環境の補助機能

ブロックダイアグラム間のナビゲーション

現在のコンポーネントまたはプロジェクトに他のコンポーネントが内包されている場合、現在の実験環境において、内包されているコンポーネントの内容を表示することができます。

ブロックダイアグラム間をナビゲートする：

1. 下位レベルのダイアグラムを表示する
   - “Elements”ペインから、ダイアグラムを表示したいコンポーネントを選択します。
     - **Elements → Show Component** を選択します。

     または

     - **Navigate down to child component** ボタンをクリックします

     または

     - コンポーネント表示部で、内包されるコンポーネントをダブルクリックします。

     選択されたコンポーネントの内容が実験環境上に表示されます。

2. 上位レベルのダイアグラムを表示する
   - **View → Show Parent Component** を選択します。

   または

   - **Navigate up to parent component** ボタンをクリックします

   または

- コンポーネント表示部の何も表示されていない場所をダブルクリックします

注記

ダブルクリックによる上下レベルへのナビゲーションは、ブロックダイアグラムまたはステートマシンでのみ可能です。

上位のコンポーネントが表示されます。

1 つのコンポーネント内の別のダイアグラムを表示する：

- 表示したいダイアグラムを、実験環境の“Diagrams”ペインから選択します。
- **Diagrams → Show Diagram** を選択します。
- 選択されたダイアグラムが表示されます。

表示オプション

ブロックダイアグラムの表示内容を変更する：

- 現在のブロックダイアグラムのシーケンスコールを表示するには、**View → Show Seq. Calls** を選択します。

  デフォルトでは、シーケンスコールは表示されません。
- シーケンスコールを非表示にするには、**View → Hide Seq. Calls** を選択します。
- ブロックダイアグラムを再描画するには、**View → Redraw** を選択します。

Elements ペインの上にある 4 つのボタンを使用して、コンポーネントのエレメントリストをさまざまな形式で表示することができます。

エレメントリストの表示形式を変更する：

- エレメントリストを階層表示（デフォルト）に切り替えるには、**Elements Hierarchical** ボタンをクリックします。

- エレメントリストををフラット表示に切り替えるには、**Elements Flat** ボタンをクリックします。

- エクスポートされているエレメント（メッセージおよびグローバルエレメント）だけをリストに表示するには、**Exported** ボタンをクリックします。

- パラメータエレメントだけを表示するには、**Parameter Elements** ボタンをクリックします。

6.1.11 データの扱い

実験中は、モデルのパラメータの値を変更しながら適切な値を見つけ、見つかった値をコンポーネントのデータセットに保存することができます。データセットについての詳細は、457 ページの「データセット」という項を参照してください。

データセットの読み取り／書き込みを行う：

- 現在選択されているデータセットに値を書き込みたいエレメントを、“Elements”ペインから選択します。
- **Elements → Write Back Data → Selected Elements**（または **Calibrated Elements**）を選択します。

  変数の一覧が表示されます。**Selected Elements** コマンドを選択した場合、この一覧には“Elements”ペインで現在選択されているすべてのエレメントが表示されます。

  Calibrated Elements を選択した場合、実験中に適合された（つまり値が変更された）すべてのエレメントが表示されます。最初は、すべての変数が選択されています。
- 値を書き込みたくない変数があればそれらの選択を解除し、**OK** をクリックします。

  選択されたエレメントの値が、コンポーネントの現在のデータセットに書き込まれ、同じエレメントの値は上書きされます。「現在のデータセット」とは、実験開始時にコンポーネントエディタ上で選択されていたデータセットです。
- 現在のデータセットからデータを読み取るには、**Elements → Reinitialise → Variables**（または **Parameters**、**Both**）を選択します。

  すべての変数またはパラメータ、またはその両方に、現在のデータセット内の値が代入され、実験中の値に上書きされます。このコマンドは、実験が実行されているときには使用できません。

また、パラメータの値を外部ファイルに書き込んだり、外部ファイルから読み込んだりすることもできます。

注記

シミュレーションボード ES1135 を使用して実験を行う場合、Non-Volatile 変数のデータを外部ファイルに書き込むこともできます。

データ交換オプションを設定する：

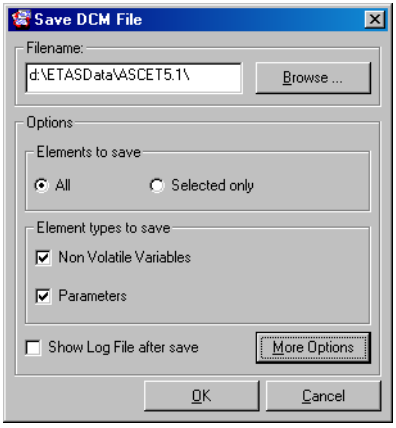
- コンポーネントマネージャで、ASCET オプションウィンドウを開きます。
- “Data Exchange”ノードでオプションを設定します。
- **Elements → Data Exchange Options** を選択します。

  オプションについては 65 ページの「データ交換オプション」を参考にしてください。

データを外部ファイルに書き込む：

- “Elements”ペインで、値をファイルに書き込みたいエレメントを選択します。
- **Elements → Save Data** を選択します。

  “Save DCM File”ダイアログボックスが開きます。



- オプションを設定します。
- **OK** をクリックします。

  選択されたパラメータの値が指定のファイルに書き込まれます。また設定に応じて書き込みログファイルの内容が表示されます。

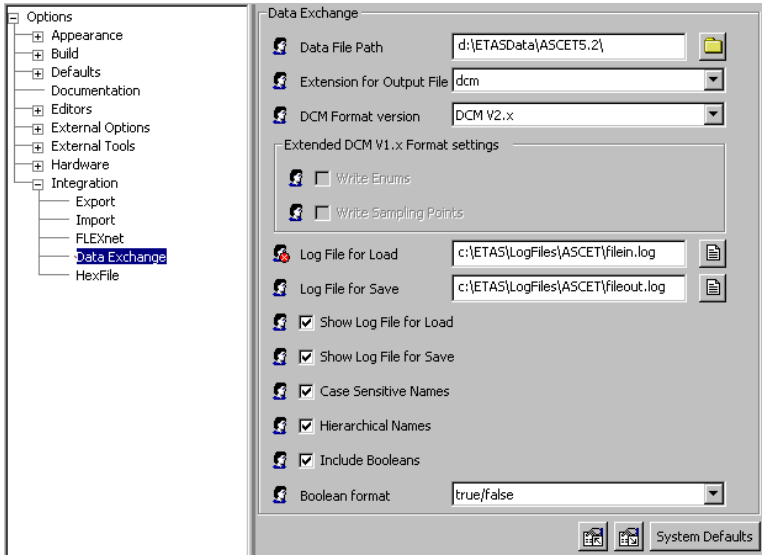
“Save DCM File”ダイアログボックスで設定するオプションは、以下のとおりです。

注記

ここでは、ファイル名は明示的に指定し、また **Non Volatile Variables** と **Parameters** の少なくとも一方を選択する必要があります。これらが指定されていないと、書き込み処理は行われません。

- “Filename”フィールドで、ファイルのパスと名前を入力するか、または **Browse** ボタンでファイルを指定します。

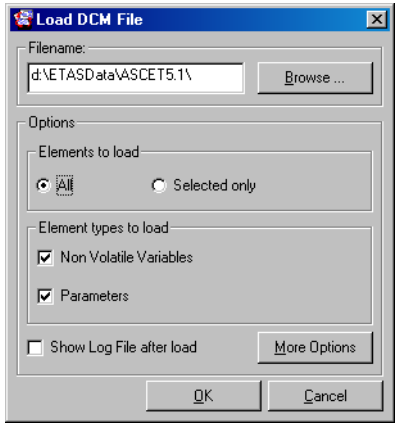
  ファイルフォーマットは、ASCET オプションウィンドウの“Data Exchange”ノード（65 ページ参照）で指定します。
- **All** または **Selected only** オプションで、すべてのエレメントのデータを保存するか、それとも選択されたエレメントのデータだけを保存するかを指定します。
- **Non Volatile Variables** または **Parameters** オプションで、データを保存するエレメントのタイプを指定します。
- **Show Log File after save** オプションで、書き込み終了後にログファイルの内容を表示するかどうかを指定します。
- **More Options** ボタンをクリックすると、ASCET オプションウィンドウの“Data Exchange”ノードが開きます。このノードに含まれるオプションについては 65 ページの「データ交換オプション」を参照してください。



データを外部ファイルから読み込む：

- **Elements → Load Data** を選択します。

  “Load DCM File” ダイアログボックスが開きます。



  ここに表示されるオプションは、“Save DCM File” ダイアログボックス（598 ページ参照）と同じですが、**All** および **Selected only** オプションは、ここでは意味を持ちません。

- オプションを設定します。
- **OK** をクリックします。

  選択されたパラメータの値が指定のファイルから読み込まれます。また設定に応じて読み込みログファイルの内容が表示されます。

6.2 適合ウィンドウ（適合エディタ）

適合システムを使用して、実験中のコンポーネントに含まれる基本エレメント（通常は「パラメータ」）の値を変更（つまり「適合」）することができます。値の変更は、実験のセットアップ時にも、実行中にも行うことができます。コンポーネントエディタでエレメントを定義した時にそのデフォルト値を指定できますが、適合システム内では、変更後の値の方がデフォルト値よりも優先されます。

適合システムでエレメントの値を変更すると、その値は、コンポーネント内の計算で変更されるか、またはデータジェネレータからスティミュレートされる値により上書きされるまで変わりません。適合ウィンドウに表示されるデータエディタは、コンポーネントの定義に用いられるものと同じです。これらのエディタについては、449 ページの「データの編集」という項で説明しています。

コンポーネントを定義するときに、「データセット」、つまり各エレメントのデフォルト値のセットをいくつか定義しておいて、実験中に必要に応じて他のセットに切り替えてたり、個々の値を変更したりできます。本項では、各種エレメント用のエディタについて説明します。

通常、データエディタはブロックダイアグラムエディタなどのコンポーネント開発環境から起動され、エレメントにデフォルト値を割り当てるために使用されます。その後、実験環境から適合ウィンドウ上に同じデータエディタを開き、実験中のエレメントの値を適合します。また、データエディタを使用してコンポーネントまたはプロジェクト用のデータセットを定義することもできます。どこから開いた場合でも、データエディタは同じように機能します。

オフライン実験環境から適合ウィンドウを開く方法は、570 ページの「適合システム」の項を参照してください。

6.2.1 メニューコマンドの概要

ここで説明するメニューコマンドのうち、実際に使用できるコマンドは、エディタやエレメントの種類、また実験環境によって異なります。

- **Edit**：

注記

Edit メニューは、3D グラフィックエディタでは使用できません。

  - *Undo Last Change*（**<Ctrl> + <U>**）
    最後の変更を取り消します。
  - *Redo Last Change*
    最後の変更を復元します。
  - *Copy*（**<Ctrl> + <C>**）
    エレメントの値をクリップボードにコピーします。
  - *Paste*（**<Ctrl> + <V>**）
    クリップボード上の値を選択されているエレメントの値に貼り付けます。
  - *Copy Entire Data Into Clipboard*
    テーブルエディタに表示されているエレメント（カーブまたはマップ）の内容をクリップボードにコピー、他のアプリケーション（Excel など）に貼り付けられるようにします。
  - *Select All Values*（**<Ctrl> + <A>**）
    すべての値を選択します。
  - *Block Selection*（**<Ctrl> + <B>**）
    カーブ上の複数の値を選択できるようにします。
  - *Decrement*（**<Ctrl> + <N>**）
    選択されている値をプリセットされた値だけ小さくします。

- *Increment*（**<Ctrl> + <M>**）
  選択されている値をプリセットされた値だけ大きくします。
- *Add Offset...*
  選択されている値にオフセット値を加えます。
- *Multiply By Factor...*
  選択されている値に値を掛けます。
- *Fill With Values...*
  選択されている値に別の値を代入します。
- *Decrement X Axis Point*
  x 座標ポイントの値をデクリメントします。つまり x 座標ポイントを左にシフトします。
- *Increment X Axis Point*
  x 座標ポイントの値をデクリメントします。つまり x 座標ポイントを右にシフトします。
- *Add X Axis Point*
  x 座標ポイントを追加します。
- *Remove X Axis Point*
  x 座標ポイントを削除します。
- *File In Data*
  配列またはテーブル用のデータをファイルから読み取ります。
- *File Out Data*
  配列またはテーブルのデータをファイルに書き込みます。

- **Axis：**

注記

Axis メニューは、特定カーブ／マップにしか使用できません。

- *X Supporting Points Setup...*
  すべての x 座標ポイントを等間隔にします。先頭の値のオフセットも指定できます。
- *Decrement X Axis Point*（**<Ctrl> + <J>**）
  x 座標ポイントの値を小さくします。
- *Increment X Axis Point*（**<Ctrl> + <K>**）
  x 座標ポイントの値を大きくします。
- *Edit X Axis Point...*（**<Ctrl> + <X>**）
  x 座標ポイントに任意の値を代入します。
- *Add X Axis Point...*
  x 座標ポイントを追加します。
- *Remove X Axis Point*
  x 座標ポイントを削除します。

- *Y Supporting Points Setup...*
  すべての y 座標ポイントを等間隔にします。先頭の値のオフセットも指定できます。
- *Decrement Y Axis Point*（**<Ctrl> + <R>**）
  y 座標ポイントの値をデクリメントします。
- *Increment Y Axis Point*（**<Ctrl> + <T>**）
  y 座標ポイントの値をインクリメントします。
- *Edit Y Axis Point...*（**<Ctrl> + <Y>**）
  y 座標ポイントに値を割り当てます。
- *Add Y Axis Point...*
  y 座標ポイントを追加します。
- *Remove Y Axis Point*
  y 座標ポイントを削除します。

- **View：**
  - *Larger font*
    適合後の値を大きいフォントで表示します。このコマンドをもう一度選択すると、元の大きさのフォントで値が表示されます。
  - *Display unit*
    値の単位の表示／非表示を切り替えます。
  - *Reset Change Marks*
    変化マークをリセットします。
  - *Show Process Point*
    現在のプロセスポイント（出力ポイント）をマークします。
  - *Set Editor on Process Point*（**<Ctrl> + <W>**）
    現在のプロセスポイントまでスクロールします。
  - *Grid*
    表示グリッドの表示／非表示を切り替えます。
  - *xz-Viewpoint*
    xz 表示に切り替えます。
  - *yz-Viewpoint*
    yz 表示に切り替えます。
  - *Show Key Help*
    主なキーボードコマンドの一覧を適合ウィンドウの最下部に表示します。

- **Extras：**
  - *Change Title...*
    強調表示されている適合ウィンドウの名前を変更します。
  - *Optimize Size*
    適合ウィンドウのサイズを最適化します。

— *Display Setup...*（**<Ctrl> + <S>**）
セットアップダイアログボックスを開きます（“Setup”ダイアログボックスの内容は、エディタの種類によって異なります）。

— Colors
グラフィカルカーブエディタの表示色を選択します。
→ Black & white — モノクロ
→ Default colors — デフォルト色
→ Invert colors — 反転色

— *Physical Representation*（**<Ctrl> + <P>**）
選択されているエレメント（1 つまたは複数）を物理値で表示します。

— *Hexadec. representation*（**<Ctrl> + <H>**）
選択されているエレメント（1 つまたは複数）を 16 進数で表示します。

— *Decimal Representation*（**<Ctrl> + <Z>**）
選択されているエレメント（1 つまたは複数）を 10 進数で表示します。

— *Binary Representation*（**<Ctrl> + <R>**）
選択されているエレメント（1 つまたは複数）を 2 進数で表示します。

— *Move Variable Into Window...*
選択されているエレメント（1 つまたは複数）を別の適合ウィンドウタに移動します。1 つの変数を一度に複数のウィンドウに割り当てることはできません。

— *Remove Variable*
選択されているエレメント（1 つまたは複数）を適合ウィンドウから削除します。

— *About Variable...*（**<Ctrl> + <I>**）
選択されているエレメント（1 つまたは複数）についての情報ウィンドウを開きます。

— *Move*（1 つのウィンドウに複数のエレメントが含まれている場合のみ）
→ *Up* — 選択されている 1 つのエレメントの表示位置を 1 つ上に移動します。
→ *Down* — 選択されている 1 つのエレメントの表示位置を 1 つ下に移動します。

6.2.2 適合用データエディタ

適合作業は、各種エディタ上で行います。数値エディタとテーブルエディタの場合は数値を直接変更し、カーブエディタとマップエディタの場合は、ブレイクポイントを移動させてカーブを直観的に変化させることができます。

どのエディタの場合も、適合を行った値には、その値の横に赤色の上向きまたは下向き矢印が表示されます。これは、現在の値が適合によって大きくまたは小さくなったことを示しています。

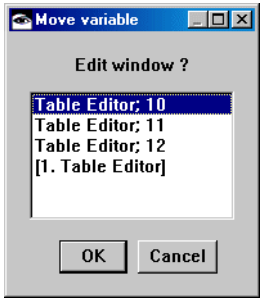
6.2.3 適合ウィンドウの使用方法

どの適合ウィンドウにも、複数のエレメント（変数）を表示することができ、各エレメントをウィンドウ間で移動させることもできます。ただしどのエレメントも 1 つのウィンドウにしか表示できないため、エレメントを他のウィンドウに移動すると、そのエレメントは元のウィンドウからは削除されます。エレメントの移動は、以下のようにして行います。

エレメントを別の適合ウィンドウに移動する：

- 他のウィンドウに移動させたいエレメント（1 つまたは複数）を選択します。
- **Extras → Move Variable Into Window** を選択します。

  “Move variable”ダイアログボックスが開き、そのエレメントの型に合った適合ウィンドウが表示されます。



- 移動先のウィンドウを選択します。新しいウィンドウを選択することもできます。
- **OK** をクリックします。

  選択されたエレメントが指定のウィンドウ（新規または既存）に移動し、元のウィンドウから削除されます。元のウィンドウに他のエレメントが残っていない場合、そのウィンドウも閉じます。

適合ウィンドウ内のエレメントの削除は、以下のように行います。

適合ウィンドウ内のエレメントを削除する：

- 適合ウィンドウから削除したいエレメント（1 つまたは複数）を選択します。
- **Extras → Remove Variables** を選択します。

  選択されたエレメントが適合ウィンドウから削除されます。ウィンドウに他のエレメントが残っていない場合、そのウィンドウも閉じます。

適合ウィンドウのタイトルの変更は、次のように行います。

適合ウィンドウのタイトルを変更する：

- タイトルを変更したい適合ウィンドウで、**Extras → Change Title** を選択します。
- “Change title”ダイアログボックスが開くので、新しいタイトルを入力します。
- **OK** をクリックします。

  適合ウィンドウのヘッダ部に、新しいタイトルが表示されます。

また以下のようにして、1 つまたは複数のエレメントについての情報適合ウィンドウを開くことができます。

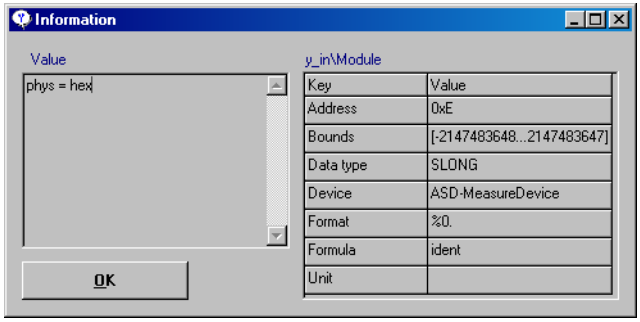
エレメントについての情報を表示する：

- 適合ウィンドウで、1 つまたは複数のエレメントを選択します。
- 同じウィンドウで、**Extras → About Variable** を選択します。

または

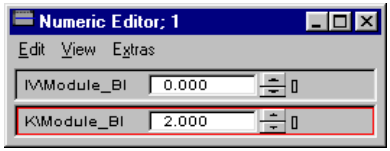
- **<Ctrl> + <I>** を押します。

  選択されたエレメントごとに、“Information" ウィンドウが開きます。



- **OK** をクリックしてウィンドウを閉じます。

以下に、適合ウィンドウで使用される各種適合エディタについて詳しく説明します。



数値エディタをセットアップする：

- 数値を物理値で表示するには、**Extras → Physical Representation** を選択します。

または

- **<Ctrl> + <P>** を押します。
- 数値を 16 進数で表示するには、**Extras → Hexadec. Representation** を選択します。

または

- **<Ctrl> + <H>** を押します。
- 数値を 2 進数で表示するには、**Extras → Binary Representation** を選択します。

または

- **<Ctrl> + <R>** を押します。
- 数値を 10 進数で表示するには、**Extras → Decimal Representation** を選択します。

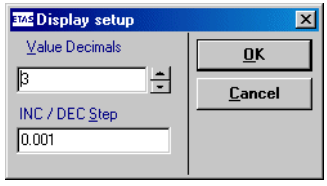
または

- **<Ctrl> + <Z>** を押します。
- 表示設定ダイアログボックスを開くには、**Extras → Display Setup** を選択します。

または

- **<Ctrl> + <S>** を押します。

  “Display setup”ダイアログボックスが開きます。



- 小数部に表示する桁数を設定します。

  これにより、値の小数部の桁数が決まります。

- “INC/DEC Step”フィールドの値を調整します。

  これにより、値のインクリメントやデクリメントを行う際のステップサイズが決まります。
- **OK** をクリックします。

1 つの数値を編集する：

- このウィンドウの数値ディスプレイ内をクリックします。
- 値を編集してから **<Enter>** を押して、変更を確定します。

  または、値の右に表示されている矢印キーを使用して値をインクリメントしたりデクリメントすることもできます。この操作により、セットアップダイアログボックスで指定したステップサイズずつ数値が大きく、または小さくなります。
- 最後の変更が行われる前の値に戻すには、**Edit → Undo Last Change** を選択します。

または

- **<Ctrl> + <U>** を押します。
- Undo 操作を取り消すには、**Edit → Redo Last Change** を選択します。

または

- **<Ctrl> + <D>** を押します。

注記

各エレメントごとに、10 個までの操作が記憶されます。

1 つの数値エディタウィンドウ内に複数のエレメントが含まれている場合、それらの値を同時に変更することができます。

複数の数値を編集する：

- 値を変更したいエレメント（1 つまたは複数）を選択します。
- 選択されたすべてのエレメントの値をセットアップダイアログボックスで設定されたステップサイズだけインクリメントするには、**Edit → Increment** を選択します。

または

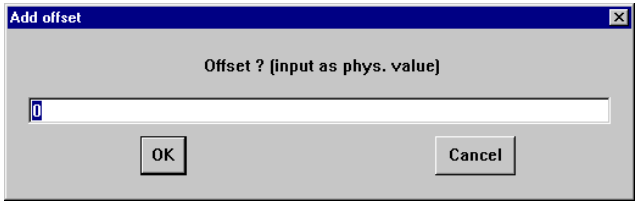
- **<Ctrl> + <M>** を押します。

- 選択されたすべてのエレメントの値をセットアップダイアログボックスで設定されたステップサイズだけデクリメントするには、**Edit → Decrement** を選択します。

または

- **<Ctrl> + <N>** を押します。
- 選択されたすべてのエレメントの値に任意の値を加えるには、**Edit → Add Offset** を選択します。

  オフセット値を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべてのエレメントに加算されます。



- 選択されたすべての値に任意の値を掛けるには、**Edit → Multiply By Factor** を選択します。

  係数を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべての変数に乗算されます。
- 選択されたすべての値に任意の値を上書きするには、**Edit → Fill With Values** を選択します。

  値を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべてのエレメントに代入されます。

注記

複数の値を上記のような操作で変更した場合、Undo 操作は行えません。

Windows のクリップボードを介して、他のアプリケーションとの間でデータを交換することができます。データの交換は、異なる適合ウィンドウ間や、適合ウィンドウと他のアプリケーションプログラム（スプレッドシートやデータベースなど）との間でも行えます。

- クリップボードにコピーしたい値を選択して強調表示します。

注記

数値エディタには複数のエレメントを表示できますが、同時にコピーできるのは 1 つのエレメントだけです。

- **Edit → Copy** を選択します。

または

- **<Ctrl> + <C>** を押します。

  選択された値がクリップボードにコピーされます。
- 値を数値エディタ内の別のエレメントに貼り付けるには、貼り付け先のエレメントを選択します。
- テーブルの 1 つのセルに値を貼り付けるには、貼り付け先のセルを選択します。

注記

テーブルの複数のセルを選択しても、値は先頭のセルにのみ貼り付けられます。

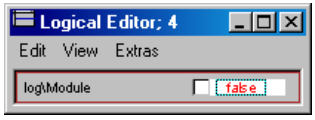
- **Edit → Paste** を選択します。

または

- **<Ctrl> + <V>** を押します。

  クリップボード上の値が、選択されたエレメントまたはセルに貼り付けられます。
- クリップボード上の値は、他のアプリケーションに貼り付けることもできます。

論理値エディタ

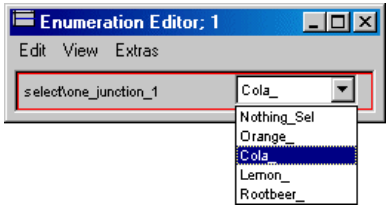


論理値エディタの各メニューコマンドの機能は、数値エディタの同名のコマンドと同じです。

論理値を編集する：

- 値を `True` にするには、チェックボックスにチェックマークを付けます。
- 値を `False` にするには、チェックボックスのチェックマークをはずします。

列挙型データ用エディタ

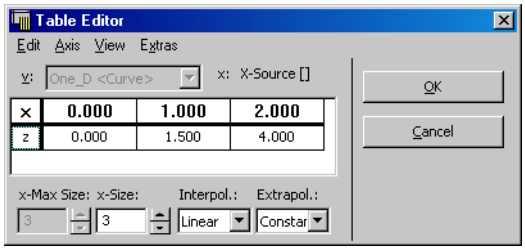


列挙型データ用エディタの各メニューコマンドの機能も、数値エディタの同名のコマンドと同じです。

列挙型データの値を選択する：

コンボボックスには、使用できる列挙子がすべて表示されます。

- コンボボックスから列挙子を選択します。
- 希望の値を選択します。**OK** をクリックして、変更を確定します。



配列エディタは、テーブルエディタウィンドウ（“Table Editor”ウィンドウ）上に開きます。１つのウィンドウに複数の配列が開いている場合、“v:”コンボボックスにそれらの配列の一覧が表示されるので、任意の配列を選択して表示／編集することができます。また、このテーブルエディタウィンドウには、配列と特性カーブ／マップを混在させて開くこともできます。

注記

2 次元配列は、マトリックスの形で表示されます。

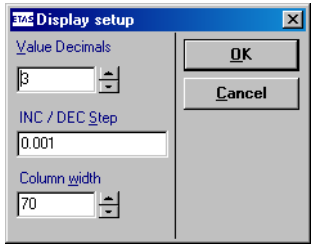
配列エディタをセットアップする：

- **Extras → Display Setup** を選択します。

または

- **<Ctrl> + <S>** を押します。

  “Display setup”ダイアログボックスが開きます。



- 配列エディタに表示される数値の小数部分の桁数を設定します。
- 値のインクリメントおよびデクリメントのステップサイズを設定します。
- 配列エディタの列幅を設定します。
- **OK** をクリックして、“Display setup”ダイアログボックスを閉じます。

1 つの出力値を編集する：

- 編集したい z 値のセルをクリックします。

  値が強調表示され、セルが編集モードになります。
- 新しい値を入力します。
- **<Enter>** を押します。

  値が確定され、新しい値の横には、変更済みであることを示す上または下向きの矢印が表示されます。
- 最後の変更を取り消すには、**Edit → Undo Last Change** を選択します。
- Undo 処理を取り消すには、**Edit → Redo Last Change** を選択します。
- 表示されている矢印を消すには、**View → Reset Change Marks** を選択します。

複数の出力値を編集する：

- z 軸上のすべての出力値（z 値）を選択するには、**Edit → Select All Values** を選択します。

または

- **<Ctrl> + <A>** を押します。
- 選択されたすべての z 値をインクリメントするには、**Edit → Increment** を選択します。

または

- **<Ctrl> + <M>** を押します。
- 選択されたすべての z 値をデクリメントするには、**Edit → Decrement** を選択します。

または

- **<Ctrl> + <N>** を押します。
- 選択されたすべての値に任意の値を加えるには、**Edit → Add Offset** を選択します。

  オフセット値を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべての z 値に加算されます。
- 選択されたすべての値に任意の値を掛けるには、**Edit → Multiply By Factor** を選択します。

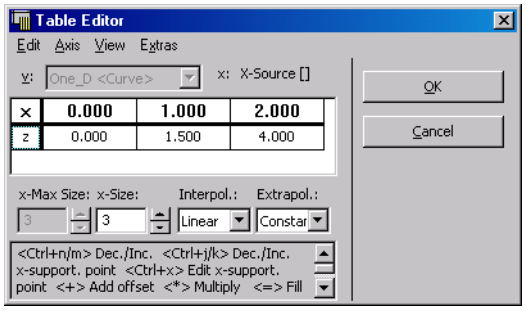
  係数を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべての z 値に乗算されます。

- 選択されたすべての値に任意の値を上書きするには、**Edit → Fill With Values** を選択します。

  値を入力するダイアログボックスが開き、そこに入力した値が、択されているすべての z 値に代入されます。

表示を変更する：

- 配列エディタのキーボードコマンドをウィンドウの下部に表示するには、**View → Show Key Help** を選択します。



- 別の配列に切り替えるたびに、その配列のサイズにあわせてウィンドウサイズが自動調整されるようにするには、**Extras → Optimize Size** を選択します。

  このコマンドは、デフォルトでオンになっています。

Windows のクリップボードを介して、他のアプリケーションとの間でデータを交換することができます。データの交換は、異なるテーブル間や、適合ウィンドウと他のアプリケーションプログラム（スプレッドシートやデータベースなど）との間でも行えます。

他のアプリケーションとデータを交換する：

- クリップボードにコピーしたい z 値を選択します。

または

- すべての z 値を選択するには、**Edit → Select All Values** を選択します。
- **Edit → Copy** を選択します。

または

- **<Ctrl> + <C>** を押します。

  選択された値がクリップボードにコピーされます。

注記

特性カーブの場合、z 値のみがコピーされ、x 値（ブレイクポイントの x 座標ポイント値）はコピーされません。

- 配列内のすべてのデータをクリップボードにコピーするには、**Edit → Copy Entire Data Into Clipboard** を選択します。
- コピーした値を他ののテーブルに貼り付けるには、貼り付け先とするセルの範囲を正確に選択してください。つまり、コピーした値の数と同じ数のセルを選択してください。
- **Edit → Paste** を選択します。

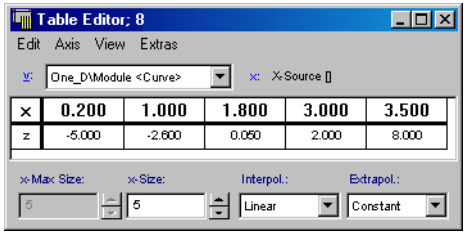
または

- **<Ctrl> + <V>** を押します。

  クリップボード上の値が、選択されたセルに貼り付けられます。

コピーしたデータは Windows のクリップボードにコピーされるため、他の Windows アプリケーションに貼り付けることができます。

特性カーブ用テーブルエディタ



配列エディタと同様、1 つのテーブルエディタウィンドウに複数の特性カーブを割り当てて適合を行うことができます。1 つのウィンドウに複数の特性カーブが開いている場合、“v:” コンボボックスにそれらのテーブルの一覧が表示されるので、任意のテーブルを選択して表示／編集することができます。

特性カーブ用テーブルエディタの機能は、前述の配列エディタとほぼ同じですが、X 座標ポイントの値も編集できる点が異なります。ここでは、この相違点についてのみ説明しますので、その他の操作については「配列エディタ)」の項を参照してください。

特性カーブ用テーブルエディタをセットアップする：

- 配列エディタと同じ方法でセットアップします。611 ページの「配列エディタをセットアップする：」を参照してください。

特性カーブやマップの場合、実験中に現在のプロセスポイント（実際の出力値を表わすポイント）を表示することができます。

プロセスポイントを表示する

- プロセスポイント表示を有効にするには、**View → Show Process Point** を選択します。

  実験が開始されると、現在の実際の出力値に隣接するブレイクポイントのセルが赤枠で示されます。
- 現在のプロセスポイントに最も近いブレイクポイントの出力値を編集するには、**View → Set Editor on Pocess Point** を選択します。

または

- **<Ctrl> + <W>** を押します。

既存のブレイクポイントを編集する：

- 変更したいブレイクポイントの座標ポイント（x 値）をクリックします。

  プロンプトボックスが開き、ブレイクポイントの現在の x 値が表示されます。
- 新しい値を入力します。

  入力する値は前後のブレイクポイントの間の値でなければなりません。それ以外の値が入力されると、変更は行われません。
- **OK** をクリックします。

  値が変更されると、その旨を示す赤い矢印が値の隣に表示されます。
- 選択された座標ポイントの値をインクリメントするには、**Axis → Increment X Axis Point** を選択します。

または

- **<Ctrl> + <K>** を押します。

- 選択された座標ポイントの値をデクリメントするには、**Axis → Decrement X Axis Point** を選択します。

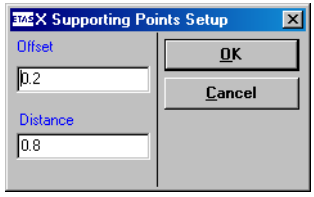
または

- **<Ctrl> + <J>** を押します。
- 表示されている矢印を消すには、**View → Reset Change Marks** を選択します。
- 最後のアクションを取り消すには、**Edit → Undo Last Change** を選択します。
- Undo 処理を取り消すには、**Edit → Redo Last Change** を選択します。

すべてのブレイクポイントを一度に編集する：

- **Axis → X Supporting Points Setup** を選択します。

  “X Supporting Points Setup”ダイアログボックスが開きます。



- オフセットと間隔を、それぞれのフィールドに入力します。
- **OK** をクリックして変更を確定します。

  テーブル全体の X 座標ポイントが、指定 0 どおりの値になります。1 番目のポイントの値はオフセットとして指定された値になり、以降のポイントの値は、指定された間隔の値が順に加算された値となります。

新しいブレイクポイントを挿入する：

- 新しいブレイクポイントを挿入するには、**Axis → Add X Axis Point** を選択します。

注記

コンポーネントエディタにおいてテーブル作成時に指定された最大サイズを超える数のブレイクポイントは作成できません。

プロンプトボックスが開きます。

- 挿入するブレイクポイントの x 値を入力します。
- **OK** をクリックします。

新しいブレイクポイントが、指定された値に応じてテーブル内の適切な位置に挿入されます。ブレイクポイントの z 値は、左隣のブレイクポイントの値と同じになります。

- 選択した座標ポイントを削除するには、**Axis → Remove X Axis Point** を選択します。

1 つの出力値を編集する：

- 編集したい x 値のセルをクリックします。

そのセルが編集モードになります。

- 新しい値を入力します。
- **<Enter>** を押します。

複数の出力値を編集する：

- 編集したい出力値（z 値）のセルを選択します。

または

- **Edit → Select All Values** を選択して、z 軸上のすべての出力値を強調表示します。
- 選択された値をインクリメントするには、**Edit → Increment** を選択します。

または

- **<Ctrl> + <M>** を押します。
- 選択された値をデクリメントするには、**Edit → Decrement** を選択します。

または

- **<Ctrl> + <N>** を押します。

- 選択されたすべての値に任意の値を加えるには、**Edit → Add Offset** を選択します。

  オフセット値を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべての z 値に加算されます。

- 選択されたすべての値に任意の値を掛けるには、**Edit → Multiply By Factor** を選択します。

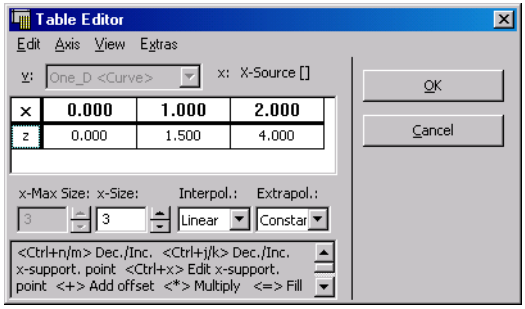
  係数を入力するダイアログボックスが開き、そこに入力した値が、選択されているすべての z 値に乗算されます。

- 選択されたすべての値に任意の値を上書きするには、**Edit → Fill With Values** を選択します。

  値を入力するダイアログボックスが開き、そこに入力した値が、択されているすべての z 値に代入されます。

表示を変更する：

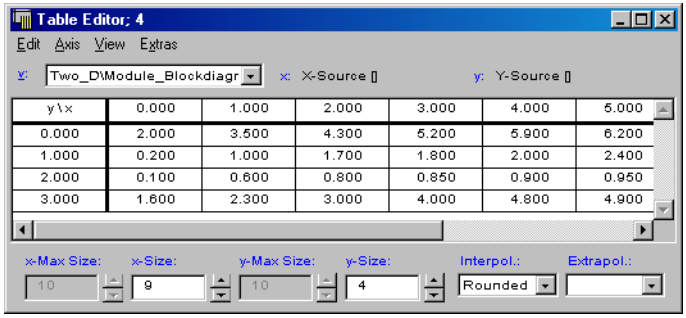
- 特性カーブ用テーブルエディタのキーボードコマンドをウィンドウの下部に表示するには、**View → Show Key Help** を選択します。



- 別のテーブルに切り替えるたびに、そのテーブルのサイズにあわせてウィンドウサイズが自動調整されるようにするには、**Extras → Optimize Size** を選択します。

  このコマンドは、デフォルトでオンになっています。

特性マップ用テーブルエディタ

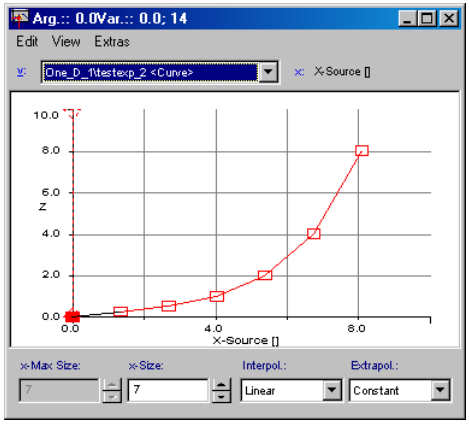


特性マップ用テーブルエディタの機能は、前述の特性カーブ用テーブルエディタとほぼ同じで、ブレイクポイントが 2 次元で表わされるという点のみが異なります。

従って、ブレイクポイントや出力値の編集方法は特性カーブとほとんど同じですが、**Axis** メニューの各コマンドは、x 座標ポイントと y 座標ポイント用にそれぞれ個別に用意されています。また、座標ポイントの数を指定するフィールドも、それぞれの座標用のものがあります。

固定マップとグループマップも、特性カーブと同じように扱われます。

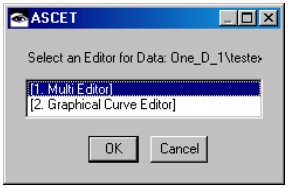
グラフィックカーブエディタ



特性カーブと特性マップは、各種グラフィックエディタを使用して視覚的に編集を行うことがことができます。ここでは、特性カーブをグラフィックカーブエディタで編集する方法を例に説明します。

以下に説明されていない操作については、特性カーブ用テーブルエディタと同じ方法で行えますので、614 ページの「特性カーブ用テーブルエディタ」の項を参照してください。グラフィックエディタは、実験環境でのみ使用できます。

グラフィックカーブエディタウィンドウを開く：



- オフライン実験環境で、編集したい特性カーブを右クリックし、ショートカットメニューから **Calibrate** を選択します。

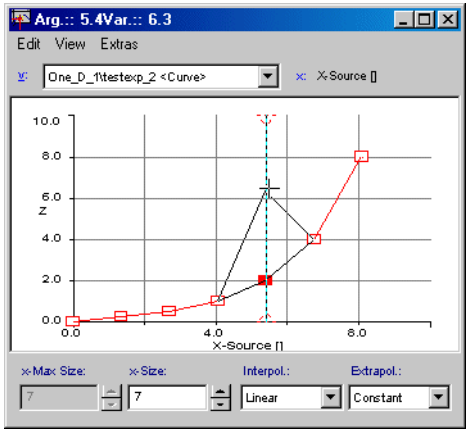
  使用できるエディタの一覧を示すダイアログボックスが開きます。
- 一覧から `Graphical Curve Editor` を選択します。
- **OK** をクリックしてエディタウィンドウを開きます。

  グラフィックカーブエディタでは、x 座標ポイントが小さい正方形で示され、それらが赤い線で接続されたグラフとして表示されます。

グラフィックカーブエディタで特性カーブを編集する：

- 編集したいブレイクポイントを示す赤い正方形をクリックします。
- その x 座標ポイントを上下にドラッグして、現在のブレイクポイントの出力値（z 値）を変更します。

  現在の z 値が、エディタウィンドウのタイトルバーに表示されます。



- ブレイクポイントの x 座標値を変更するには、選択したブレイクポイント上に表示された縦線カーソル左右に移動します。

  縦線カーソルの動きに応じて、x 座標値が調整されます。縦線カーソルは、両隣のブレイクポイントまで動かすことができます。

または

- **Edit → Decrement X Axis Point** を選択し選択して x 座標値をデクリメントします。

または

- **Edit → Increment X Axis Point** を選択して x 座標値をインクリメントします。

- 2 つ以上のブレイクポイントを同時に変更する「ブロック選択モード」に切り替えるには、**Edit → Block Selection** を選択します。

または

- **<Ctrl> + <B>** を押します。

  変更したいすべてのブレイクポイントを囲むようにマウスカーソルをドラッグします。複数の値を一度に調整できるのは、z 座標値だけです。

- ブロック選択モードを解除するには、もう一度 **Edit → Block Selection** を選択します。

ブレイクポイントを追加／削除する：

- 選択されたブレイクポイントを削除するには、**Edit → Remove X Axis Point** を選択します。
- 新しいブレイクポイントを挿入するには、**Edit → Add X Axis Point** を選択します。

注記

コンポーネントエディタにおいてテーブル作成時に指定された最大サイズを超える数のブレイクポイントは作成できません。

プロンプトボックスが開きます。

- 挿入するブレイクポイントの x 値を入力します。

- **OK** をクリックします。

  新しいブレイクポイントが、指定された値に応じてグラフ内の適切な位置に挿入されます。ブレイクポイントの z 値は、左隣のブレイクポイントの値と同じになります。

グラフの表示を変更する：

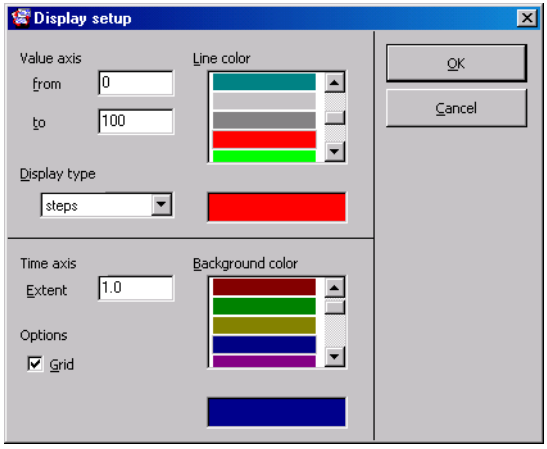
- グラフ上のグリッドの表示／非表示を切り替えるには、**View → Grid** を選択します。
- エディタをモノクロ表示にするには、**Extras → Colors → Black & white** を選択します。
- デフォルトの色に戻すには、**Extras → Colors → Default colors** を選択します。
- 表示色を反転するには、**Extras → Colors → Invert colors** を選択します。

注記

座標軸とラベル（座標軸の目盛りの位置を表わす値）の色を変更することはできません。これらは通常は黒色で表示され、**Black & White** か **Invert colors** を選択した場合には白色で表示されます。

グラフィックカーブエディタの表示をセットアップする：

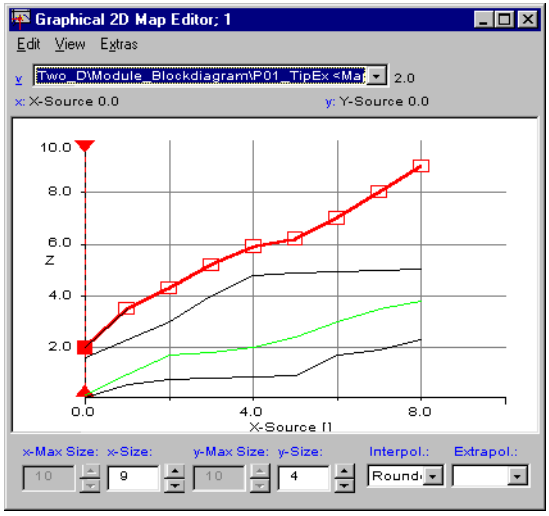
- **Extras → Display Setup...** を選択して、“Display setup”ダイアログボックスを開きます。



- “Value axis”および“X-axis”フィールドで、z 値と x 軸の上下限値を設定します。

- “Line Color”フィールドで、グラフの表示色を選択します。
- “Background Color”フィールドで、背景色を選択します。
- “INC/DEC Step”フィールドで、インクリメントとデクリメントのステップサイズを調整します。
- グリッドを表示する場合は、**Grid** チェックボックスにチェックマークを付けます。
- **OK** をクリックして変更を確定します。

2D グラフィックマップエディタ



2D グラフィックマップエディタでは、特性マップを 2D グラフ上で編集することができます。このエディタはグラフィックカーブエディタと似ていて、特性マップを複数の特性カーブとして表示します。

2D グラフィックマップエディタウィンドウを開く：

- オフライン実験環境で、編集したい特性マップを右クリックし、ショートカットメニューから **Calibrate** を選択します。

  使用できるエディタの一覧を示すダイアログボックスが開きます。
- 一覧から`Graphical 2D Map Editor`を選択します。

- **OK** をクリックしてエディタウィンドウを開きます。

2D グラフィックマップエディタで特性マップを編集する：

- 編集したいカーブをクリックして選択します。

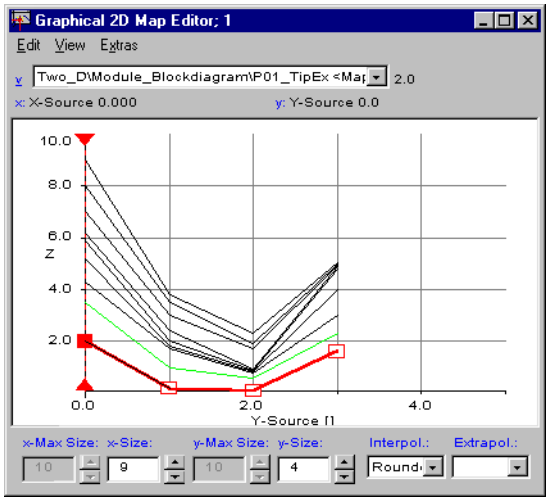
  選択したカーブには、ブレイクポイントを示す長方形が表示されます。
- 選択したカーブのブレイクポイントを、グラフィックカーブエディタの場合と同じようにドラッグして z 値を調整します（619 ページの「グラフィックカーブエディタ」を参照してください）。

  2D グラフィックマップエディタの場合、z 値のインクリメント／デクリメント用のメニューコマンドは、**Edit → Increment Value** および **Edit → Decrement Value** です。またブレイクポイントを左右に移動すると、x 座標ポイントまたは y 座標ポイント（次の「2D グラフィックマップエディタの視点を切り替える：」を参照してください）が移動します。
- 他のカーブに切り替えるには、**< ↑ >** かまたは **< ↓ >** キーを押すか、または選択したいカーブを直接クリックします。

2D グラフィックマップエディタでは、ブレイクポイントの追加／削除はできません。

2D グラフィックマップエディタの視点を切り替える：

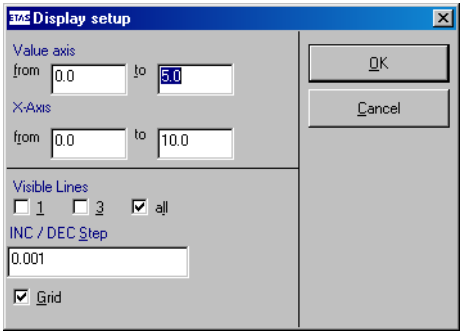
- 座標系を変えて別の視点からグラフィック表示するには、**View → yz-Viewpoint** を選択します。



- デフォルトの視点に戻すには、**View → xz-Viewpoint** を選択します。

2D グラフィックマップエディタの表示をセットアップする：

- **Extras → Display Setup...** を選択して、“Display setup”ダイアログボックスを開きます。

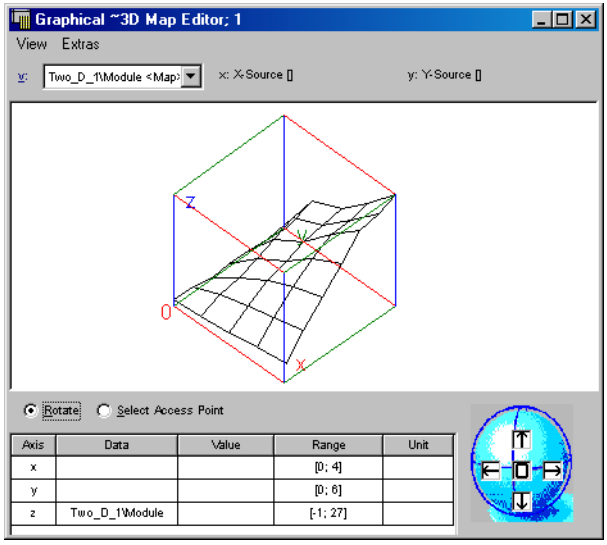


- z 値（出力値）と x 軸の範囲を、それぞれのフィールドに入力します。

- 同時に表示されるカーブの数を設定します。

  1 を選択すると、現在選択されているカーブだけが表示されます。この場合、< ↑ > と < ↓ > でカーブを切り替えることができます。
- グリッドを表示したくない場合には、**Grid** オプションをオフにします。
- インクリメントとデクリメントのステップサイズを調整します。
- **OK** をクリックして変更を確定します。

3D グラフィックマップエディタ



3D グラフィックマップエディタでは、特性マップを 3 次元グラフとして表示します。グラフを全方向に回転させ、マップ全体を直感的に把握しながら編集することができます。3D グラフィックマップエディタは、実験環境からしか使用できません。

3D グラフィックマップエディタウィンドウを開く：

- オフライン実験環境で、編集したい特性マップを右クリックし、ショートカットメニューから **Calibrate** を選択します。

  選択したテーブルの編集に使用できるエディタの一覧を示すダイアログボックスが開きます。

- 一覧から`Graphical 3D Map Editor`を選択します。
- **OK** をクリックします。

  3D グラフィックマップエディタウィンドウが開きます。

3D グラフ表示の下にリストが表示されます。**Select Access Point** オプションをオンにすると、このリストには現在選択されて強調表示されているネットポイントの値が表示されます。この値を直接変更して、ネットポイントを移動させることができます。

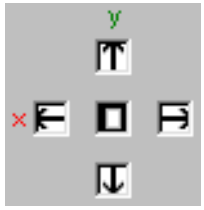
3D グラフィックマップエディタのネットポイントを選択する：

- リストの上にある、**Select Access Point** オプションを選択します。

または

- **<S>** キーを押します。

  4 つの方向を指す矢印で構成される十字のシンボルが、リストの右隣に表示されます。



  また、3D 座標系の原点に最も近いネットポイントが選択され、そのポイントと隣接するポイントとを結ぶ線が灰色になります。
- シンボルの矢印をクリックして、ポイントを上下左右に移動します。

または

- ←↑↓→のカーソルキーを使用します。

  x 軸と y 軸の各座標ポイントとそれに対応する z 軸の値（出力値）が、テーブルの“Value”列に表示されます。テーブルには、各軸の範囲も表示されます。
- 編集したいポイントまで移動します。

- リストの“Value”列に新しい値を入力します。

注記

グループマップや固定マップの場合、変更できるのは z 値だけです。

- 最初のポイントに戻るには、シンボルの中心にある正方形をクリックします。

または

- **<0>** キーを押します。

ネットの目が詰まっていて希望のポイントを選択しにくい場合、座標系を回転させることができます。

座標系を回転させる：

- リストの上にある **Rotate** オプションを選択します。

または

- **<R>** キーを押します。

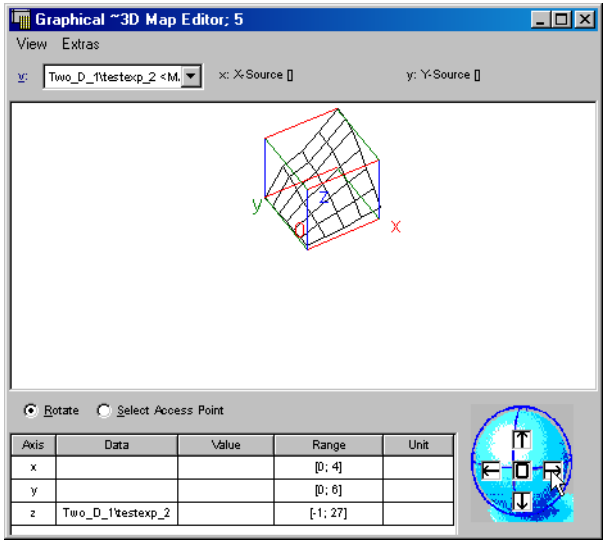
- 3Dグラフィックマップエディタウィンドウの右下にある、回転コントロールの矢印の 1 つをクリックします。

  表示を縦横の任意の方向に回転させることができます。

- 元のアングルに戻すには、**0** ボタンをクリックします。

- 高速で回転させるには、矢印ボタンをクリックしてそのままマウスボタンをホールドします。

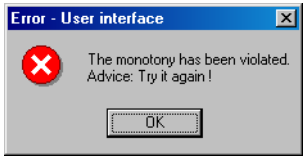
  3D グラフのサイズが縮小され、回転が速まります。



6.2.4 テーブルエディタでのブレイクポイント適合時の注意点

テーブルエディタには、前述の **Edit** メニューの適合コマンド以外に、ブレイクポイントの座標ポイント（x 値または y 値）を適合するためのコマンドが **Axis** メニューに用意されています。

座標ポイントの値を変更しようとすると、入力された値と前後の値との連続性が保たれているかがチェックされます。新しい値が前後の値の間でなかった場合、以下のようなダイアログボックスが開きます。



特性カーブまたはマップの座標ポイントの値を変更するには、テーブルエディタの x または y のテーブルセルの値を変更します。複数の座標ポイントを同時に変更することはできません。

以下にカーブのブレイクポイントを適合するための x 座標ポイントの変更方法を説明しますが、マップの場合は、y 座標ポイントについても同様の操作を行えます。

テーブルエディタで座標ポイントを適合する：

- 編集したい x 座標ポイントのセルをクリックします。

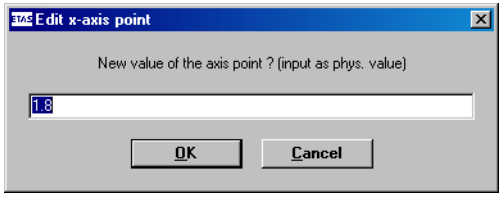
または

- **<Ctrl> + <X>** キーを押します。

または

- **Axis → Edit X Axis Point** を選択します。

  以下のダイアログボックスが開きます。



- 新しい値を入力します。
- **OK** で確定します。

または

- **Cancel** ボタンを使用して操作を取り消します。
- プリセットされている量だけ値をデクリメントするには、
  - **Axis → Decrement X Axis Point** コマンドを選択します。

  または
  - **<Ctrl> + <J>** キーを押します。
- プリセットされている量だけ値をインクリメントするには、
  - **Axis → Increment X Axis Point** コマンドを選択します。

または

- **<Ctrl> + <K>** キーを押します。
- 選択したブレイクポイントを削除するには、**Axis → Remove X Axis Point** コマンドを選択します。
- ブレイクポイントを追加するには、**Axis → Add X Axis Point** コマンドを選択します。

y 座標ポイントの編集は、上記の手順で、y 軸用のメニューコマンドを使用して行います。

固定カーブ／マップやグループカーブ／マップ、およびディストリビューションの場合は、座標ポイントの編集は行えません。その場合、**Axis** メニューはグレイアウトされて無効になるか、または表示されません。

6.3 測定ウィンドウ

エレメント（測定変数）の値を測定ウィンドウに表示するには、適切なウィンドウを選択し、表示や測定に関するオプション設定を行います。測定ウィンドウに割り当てられたエレメントは、ASCET の実験中にその値が読み取られ、そのウィンドウ内に表示されます。各エレメントごとに 1 つの測定チャンネルが作成されますが、1 つのエレメントを 2 つの測定ウィンドウに表示する必要がある場合は、そのエレメント用に複数の測定チャンネルを作成することが可能です。また、測定データを保存しないで表示のみを行ったり、また、表示中に記録を開始／終了したり、表示せずに記録のみを行うこともできます。

なお、オフライン実験の場合、リアルタイムな測定は行われません。

6.3.1 測定ウィンドウの選択

実験環境では、以下のようなディスプレイタイプの測定ウィンドウを使用することができます。

- オシロスコープ
- 数値ディスプレイ
- 垂直棒グラフディスプレイ
- 水平棒グラフディスプレイ
- ビットディスプレイ
- レコーダ

測定ウィンドウを開く方法は、567 ページの「測定システム」を参照してください。

6.3.2 メニューコマンドの概要

使用できるコマンドは、ウィンドウにより異なります。

- **Extras：**
  - *Change title...*
    強調表示されている測定ウィンドウの名前を変更します。
  - *Message when out of bounds*
    定義されている範囲外の値が測定された場合にユーザーへの通知が行われるようにします。
  - *Setup...*（**<Ctrl> + <S>**）
    セットアップダイアログボックスを開きます。

— Colors
測定ウィンドウの表示色を選択します。
→ Black & white — モノクロ
→ Default colors — デフォルト色
→ Invert colors — 反転色

— *Physical Representation*（**<Ctrl> + <P>**）
選択されているエレメント（1 つまたは複数）を物理値で表示します。

— *Hexadec. representation*（**<Ctrl> + <H>**）
選択されているエレメント（1 つまたは複数）を 16 進数で表示します。

— *Decimal Representation*（**<Ctrl> + <Z>**）
選択されているエレメント（1 つまたは複数）を 10 進数で表示します。

— *Binary Representation*（**<Ctrl> + <R>**）
選択されているエレメント（1 つまたは複数）を 2 進数で表示します。

— *Copy variable to window...*
選択されているエレメント（1 つまたは複数）を、別の測定ウィンドウにコピーします。

— *Move variable to window...*
強調表示されているエレメント（1 つまたは複数）を、別の測定ウィンドウに移動します。

— *Remove Variable*
選択されているエレメント（1 つまたは複数）を測定ウィンドウから削除します。

— *About Variable*（**<Ctrl> + <I>**）
選択されているエレメント（1 つまたは複数）についての情報ウィンドウを開きます。

— *Attributes*
→ *Copy* — 強調表示されているエレメントの表示オプションをコピーします。
→ *Paste* — 強調表示されているエレメントに表示オプションを割り当てます（このコマンドは、直前に Copy コマンドが実行されている場合に限り実行できます）。

— *Move*（1 つのウィンドウに複数のエレメントが含まれている場合のみ）
→ *Up* — 選択されている 1 つのエレメントの表示位置を 1 つ上に移動します。
→ *Down* — 選択されている 1 つのエレメントの表示位置を 1 つ下に移動します。
→ *Left* — 選択されている 1 つのエレメントの表示位置を 1 つ上に移動します。（垂直棒グラフの場合のみ）
→ *Right* — 選択されている 1 つのエレメントの表示位置を 1 つ下に移動します。（垂直棒グラフの場合のみ）

- File：
  - *Print*
    オシロスコープまたはレコーダウィンドウの内容を印刷するためのダイアログボックスを開きます。
  - *Copy To Clipboard*
    オシロスコープまたはレコーダウィンドウのスクリーンコピーをクリップボードにコピーします。クリップボードにコピーされたデータは、任意のアプリケーションにペーストすることができます。
  - *Save Selected Channels*
    選択した測定チャンネルのデータを保存します。
    → *MDF* ー 測定チャンネルのデータを MDF フォーマットで保存します。
    → *FAMOS* ー 測定チャンネルのデータを FAMOS フォーマットで保存します。
  - *Save All Channels*
    すべての測定チャンネルのデータを保存します。
    → *MDF* ー すべての測定チャンネルのデータを MDF フォーマットで保存します。
    → *FAMOS* ー すべての測定チャンネルのデータを FAMOS フォーマットで保存します。
- **Edit：**
  - *Define trigger*
    トリガ条件を定義します（表示だけに影響するトリガです）。
  - *Activate trigger*
    トリガを有効／無効にします。
  - *Trigger manually*
    トリガシグナルを手操作で発生させます。
  - *Autoscale*
    y 軸のスケーリングを、強調表示されている測定チャンネルに合わせて調整します。
  - *Autoscale all channels*
    y 軸のスケーリングを、すべての測定チャンネルに合わせて調整します。
  - *Undo last scaling*
    最後のスケーリングコマンドを取り消します。
  - *Autodistribution*
    強調表示されている複数のチャンネルを、それぞれ別の表示領域に分散させます。
  - *Analyze measure data*
    分析モードに切り替えます。

- *Analysis setup*
  分析モードの設定を定義します（分析モードでだけ有効）。
- *Analyze next point*
  測定カーソルを次の測定ポイントまで移動します（分析モードでだけ有効）。
- *Analyze previous point*
  測定カーソルを前の測定ポイントまで移動します（分析モードでだけ有効）。

- **View：**
  - *Show selected channel*
    強調表示されている測定チャンネル（1 つまたは複数）の表示／非表示を切り替えます。
  - *Grid*
    背景のグリッドを定義します。“none”、“dynamic”、“fixed”から選択できます。
  - *Show measure channel lists*
    測定チャンネルのリストの表示／非表示を切り替えます。
  - *Min/Max*
    Min./Max. 値の測定チャンネルリストへの表示／非表示を切り替えます。
  - *Rate*
    サンプルレートの測定チャンネルリストへの表示／非表示を切り替えます。
  - *Value at active cursor*
    アクティブカーソルの位置の測定チャンネル値の測定チャンネルリストへの表示／非表示を切り替えます（分析モードでだけ有効）。
  - *Differences between cursors*
    測定チャンネルリストのエレメントについて、2 つのカーソル位置の値の差の表示／非表示を切り替えます（分析モードでだけ有効）。
  - *Show key help*
    最も重要なキーボードコマンドのフッタへの表示／非表示を切り替えます。
  - *Show Setup*
    設定オプションのフッタへの表示／非表示を切り替えます。
  - *Larger font*
    大きいフォントでの表示に切り替えます。

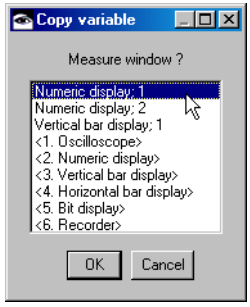
6.3.3 測定ウィンドウの一般的な使用方法

1 つの測定ウィンドウには複数の測定チャンネルのデータを表示することができ、ウィンドウ間で測定チャンネルをコピー／移動することができます。チャンネルを他のウィンドウに移動すると、そのチャンネルは元のウィンドウから削除され、移動先のウィンドウに追加されます。チャンネルをコピーすると、同じエレメントの新しい測定チャンネルがコピー先のウィンドウに作成されます。

測定ウィンドウ間でチャンネルをコピーする：

- 他の測定ウィンドウにコピーしたいチャンネル（1 つまたは複数）をクリックして選択します。
- **Extras → Copy variable to window** を選択します。

  “Copy variable”ダイアログボックスが開き、使用できるすべての測定ウィンドウがリストアップされています。



- コピー先のウィンドウを選択します。
- **OK** をクリックします。

  新しいウィンドウを選択した場合には、そのウィンドウが開き、選択されたチャンネルがそこに表示されます。そうでない場合には、選択されたチャンネルが既存のウィンドウにコピーされます。

測定ウィンドウ間でチャンネルを移動する：

- 他の測定ウィンドウに移動したいチャンネル（1 つまたは複数）をクリックして選択します。
- **Extras → Move variable to window** を選択します。

  “Move variable”ダイアログボックスが開き、使用できるすべての測定ウィンドウがリストアップされています。

- 移動先のウィンドウを選択します。
- **OK** をクリックします。

  チャンネルが、指定された測定ウィンドウ（新規または既存のウィンドウ）に移動し、元のウィンドウからは削除されます。それによって元のウィンドウが空になった場合、そのウィンドウも削除されます。

測定ウィンドウ内のチャンネルを削除するには、以下のように行ってください。

測定ウィンドウからチャンネルを削除する：

- 削除したいチャンネル（1 つまたは複数）をクリックして選択します。
- **Extras → Remove variable** を選択します。

または

- **<Delete>** キーを押します。

  チャンネルが、ウィンドウから削除されます。それによってウィンドウが空になった場合、そのウィンドウも削除されます。

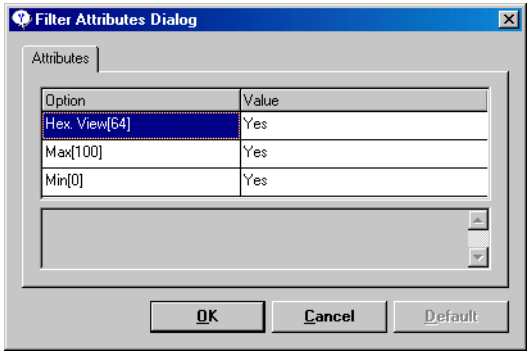
測定ウィンドウの表示設定は、他のウィンドウにコピーすることができ、これによって煩雑なセットアップ作業を簡潔にすることができます。設定された各表示属性のうち、コピー先のウィンドウに適用できるものだけがコピーされ、またそのうちで実際にコピーする属性を任意に選択することができます。

測定ウィンドウ間で属性をコピーする：

- 表示設定のコピー元にしたい測定ウィンドウで、**Extras → Attributes → Copy** を選択します。

- 表示設定のコピー先にしたい測定ウィンドウで、**Extras → Attributes → Paste** を選択します。

  “Filter Attributes Dialog”ダイアログボックスが開きます。そこには、コピー先の測定ウィンドウに適用可能なすべての属性が、元の測定ウィンドウでの設定値とともに表示されます。



- 実際にコピーする属性を選択するには、各属性の行の“Value”列のフィールドをクリックします。

  フィールドがコンボボックスになります。
- コピーする属性については、Yes を選択します。
- コピーしない属性については、No を選択します。

  No が選択された属性は、コピー先の測定ウィンドウに適用されません。
- **OK** をクリックします。

  選択された属性が、コピー先の測定ウィンドウにコピーされます。

以下に、測定ウィンドウの表示設定を変更するためのメニューコマンドについて説明します。

測定ウィンドウの表示を変更する：

- 測定ウィンドウ内で、表示設定を変えたいチャンネル（1 つまたは複数）を選択します。
- 選択されたチェンネルを物理値で表示するには、**Extras → Physical representation** を選択します。

- 選択されたチェンネルを 16 進数で表示するには、**Extras → Hexadec. representation** を選択します。

  この表現は、固定小数点コードを用いる実験など便利です。
- 測定値がそのチャンネルに設定された監視限界値を超えるたびにメッセージウィンドウが表示されるようにするには、**Extras → Message when out of bounds** を選択します。

  この操作はオシロスコープやレコーダでは行えません。

測定ウィンドウのタイトルは、以下のようにして変更します。

測定ウィンドウのタイトルを変更する：

- タイトルを変更した測定ウィンドウで、**Extras → Change Title** を選択します。
- 入力ダイアログボックスが開くので、新しいタイトルを入力します。
- **OK** をクリックします。

  新しいタイトルが測定ウィンドウのヘッダ部に表示されます。

測定チャンネルに割り当てられているエレメント（変数）の情報を表示するには、以下のように操作します。

エレメントについての情報を表示する：

- 測定ウィンドウで、1 つまたは複数のエレメントを選択します。
- 同じウィンドウで、**Extras → About Variable** を選択します。

または

- **<I>** を押します。

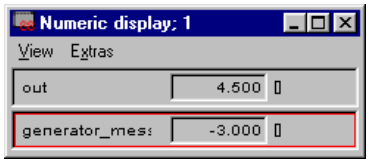
  選択されたエレメントごとに、“Information" ウィンドウが開きます。
- **OK** をクリックしてウィンドウを閉じます。

6.3.4 測定ウィンドウで使用される各種ディスプレイ

前項では、測定ウィンドウについての一般的な操作方法を説明しましたので、本項で、測定ウィンドウに表示されるさまざまなスタイルのディプレイごとに、その機能と操作方法を詳しく説明します。

数値ディスプレイ

数値ディスプレイには、測定チャンネルの値が 10 進、2 進または 16 進数で表示されます。



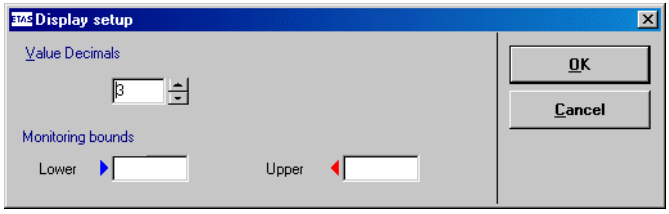
数値ディスプレイをセットアップする：

- セットアップしたい数値ディスプレイをクリックして選択します。

  <Ctrl> を押し下げたままで同じウィンドウ内の複数の表示をクリックすれば、それらを一度に選択することができます。

- **Extras** → **Setup** を選択します。

  以下のダイアログボックスが開きます。



- “Value Decimals”フィールドに、測定値を表示する際の小数部の桁数を設定します。
- “Monitoring bounds”の 2 つのフィールドでは、監視用の上限値と下限値を定義することができます。
- **OK** をクリックしてダイアログボックスを閉じます。

  この監視範囲を超える値が測定されると、タイトルバーの赤いアラームランプが点灯します。さらに、測定値が下限値を下回っているときには測定値の左に青い正方形が表示され、上限値を上回っている時には赤い正方形が表示されます。

それに加えてメッセージウィンドウも開き、このウィンドウは、測定値が上下限値内に戻ると自動的に閉じます。

数値ディスプレイの表示オプションを変更する：

- 測定値の数値表現を変更するには、**Extras → <Format> representation** を選択します（637 ページ参照）。
- 選択した測定チャンネルの表示位置を、ウィンドウ内で上下に移動するには、**Extras → Move → Up** または **Down** を選択します。

  このコマンドは、1 つのウィンドウに複数のチャンネルが設定されている場合に限り使用できます。
- オフライン実験環境では、**View → Larger font** を選択して、測定値を表示するフォントを大きくすることができます。

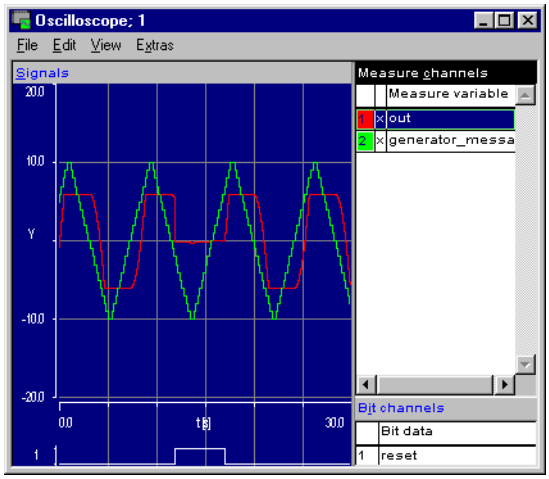
オシロスコープ

オシロスコープには、実際のオシロスコープと同じような、非常に柔軟な表示機能が用意されています。1 つのオシロスコープに複数のチャンネルを表示することができますが、測定するチャンネル数が多い場合は、いくつかのウィンドウに分けて表示するほうが、視認性がよくなります。

オシロスコープを使用する場合は、まず、表示オプションをセットアップして、ウィンドウ全体のレイアウトや、各チャンネルをどのように配置して表示するかを決めます。実験中には、表示されるデータの保存や分析を行ったり、トリガを使用した測定や保存を行うなど、さまざまな操作を行うことができます。

ウィンドウ左側の“Signals”ペインには、測定された数値の波形を表すカーブが、アナログチャンネル（上部）とビットチャンネル（下部）ごとに表示されます。また“Signals”ペインの右の“Measure Channels”リストにはアナログチャンネ

ルの情報（名前やその他任意に選択可能な項目）が表示され、その下の“Bit Channels”リストに論理値の測定チャンネル（「ビットチャンネル」）についての情報が表示されます。



まず、オシロスコープ内の各測定チャンネルについてのセットアップの方法を説明します。必要に応じて、複数のチャンネルやすべてのチェンネルを選択し、それらのチャンネルを同時にセットアップすることもできます。

測定チャンネルをセットアップする：

- セットアップしたいチャンネルを、“Measure Channels”ペインから選択します。

  <Ctrl> キーを押し下げたまま複数のチャンネルをクリックすれば、それらを一度に選択することができます。

注記

ビットチャンネルのセットアップは行えません。

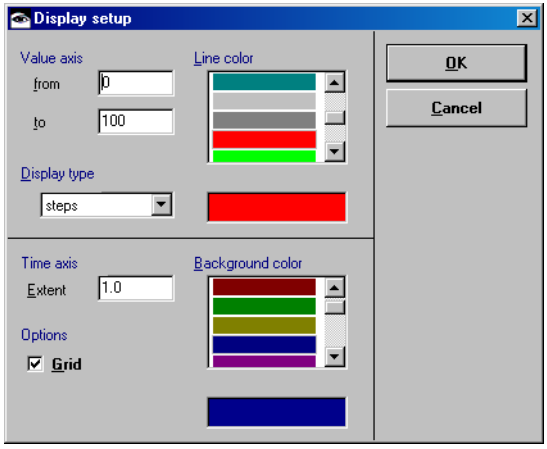
- **Extras → Setup** を選択します。

または

- チャンネルをダブルクリックします。

または

- **<S>** キーを押します。
  “Display setup”ダイアログボックスが開きます。



- “from”および“to”フィールドで、y 軸の下限値と上限値を調整します。
  ここに入力される値の範囲がオシロスコープの y 軸として表示されます。デフォルトの範囲は 0 ～ 100 です。
- “Line Color”フィールドでシグナル波形の表示色を選択します。

注記

このオプションは、1 つの測定チャンネルが選択されている場合にのみ、設定可能です。

  表示の視認性をよくするためには、各チャンネルにそれぞれ異なる色を割り当ててください。デフォルト状態においても、各チャンネルに固有の色が設定されます。
- “Display type”コンボボックスで線のスタイルを選択します。
  `steps` を選択すると、各サンプリングポイントの値がステップ表示、つまり階段状に接続され、`line` を選択すると、各値が直線で結ばれます。
- **OK** をクリックして、設定を有効にします。

“Display Setup”ダイアログボックスには、上記の個々のチャンネルについての設定以外に、オシロスコープウィンドウ全体の表示をセットアップする機能もあります。

オシロスコープウィンドウをセットアップする：

- 上述のようにして、“Display Setup”ダイアログボックスを開きます。

  ここでは、どのチャンネルが選択されていても、以下の操作には影響しません。
- “Time Axis”フィールドに時間軸の表示範囲を入力します。

  ここに入力した範囲の時間の幅が、オシロスコープの横軸として表示されます。たとえば 1 を入力すると 1 秒間、または 0.5 を入力すると 0.5 秒間のシグナル値がオシロスコープに表示されます。デフォルト値は 1 秒です。

注記

オフライン実験ではリアルタイムなシミュレーションは行われないため、オシロスコープに表示される時間は、実際の経過時間とは一致しません。

- “Background Color”フィールドから、オシロスコープの背景の色を選択します。
- **Grid** オプションをオンにすると、オシロスコープにグリッド線が表示されます。
- **OK** をクリックして設定を確定します。

オシロスコープの色を変更する：

- オシロスコープウィンドウをモノクロ表示にするには、**Extras → Colors → Black & white** を選択します。
- オシロスコープウィンドウをデフォルトの色にリセットするには、**Extras → Colors → Default colors** を選択します。
- 現在の表示色を反転するには、**Extras → Colors → Invert colors** を選択します。

以下の操作は、チャンネルのシグナル波形表示のスケーリングや、測定の開始／停止などに関するものです。自動スケーリングは、現在表示されているシグナル波形を元に最適なスケーリングを行うもので、1 つ、または複数のチャンネルについて行えます。

測定チャンネルをセットアップする：

- **Edit → Autoscale** を選択します。

  選択されたチャンネルについて、現在表示されている値の範囲を元に y 軸の表示範囲が変更されます。

- **Edit → Autoscale all channels** を選択すると、一度にすべてのチャンネルについて自動スケーリングが行われます。

  これは、1 チャンネルずつ個別に **Edit → Autoscale** を実行するのと同じことです。

- **Edit → Autodistribution** を選択すると、選択されているチャンネルが、独立した表示領域に分かれて表示されます。

  各チャンネルについて、互いにシグナル波形が重ならないように、別の y 軸領域が割り当てられます。

注記

全チャンネルを選択しないで**Edit → Autoscale all channels** を実行した場合、選択されていなかったチャンネルについては自動スケーリングはおこなわれません。

- 最後のスケーリングを取り消すには、**Edit → Undo last scaling** を選択します。

または

- **<Ctrl> + <U>** キーを押します。

オシロスコープに表示されているチャンネルを、そのウィンドウから削除することなく、シグナル波形表示のみを一時的に消すことができます。以下のように行います。

測定チャンネルの表示／非表示を切り替える：

- 1 つまたは複数のチャンネルを選択します。
- **View → Show selected channel** を選択します。

または

- **<X>** を押します。

  選択されたチャンネルのシグナル波形の表示／非表示が切り替わりますが、“Measure channels” リストには常に表示されます。

“Measure channels”および“Bit channels”リストの表示／非表示の切り替えは、以下のように行います。

リストの表示／非表示を切り替える

- **View → Show measure channel lists** を選択します。

または

- **<L>** キーを押します。

  “Measure Channels”および“Bit channel”リストの表示／非表示が切り替わります。

注記

個々のリストについて別々に表示／非表示を切り替えることはできません。

“Measure channels”および“Bit channels”リストのセットアップは、以下のように行います。

チャンネルリストをセットアップする：

- **View → Min/Max** を選択します。

  “Measure channels”および“Bit channel”リストに“Min..Max”列が追加され、各チャンネルについて定義されている最小値と最大値が表示されます。

- **View → Rate** を選択します。

  両リストに“Rate”列が追加され、各チャンネルのサンプリングレートが表示されます。

オシロスコープウィンドウにキーボードコマンドの一覧を表示することができます。

キーボードコマンドを表示する：

- **View → Show key help** を選択します。

  オシロスコープウィンドウの最下部にキーボードコマンドが底部に表示されます。

  ここには、現在使用できるキーボードコマンドのみが表示されます。たとえば、測定モードのときには分析モードのコマンドは表示されません。

測定チャンネルの現在のセットアップ内容を見るには、以下のようにします。

チャンネルのセットアップ内容を表示する：

- チャンネルを選択します。
- **View → Show Setup** を選択します。

  チャンネルの表示範囲に関する設定内容が、ウィンドウの下部に表示されます。

  続いて別のチャンネルを選択して同じ操作を行うと、新しいチャンネルの情報に変わります。
- 情報が表示されているチャンネルを選択して同じ操作を行うと、情報が消去されます。

デフォルト状態においては、オシロスコープは「測定モード」になっていて、実験中に測定された値が順次表示されますが、この他に、データを詳しく分析するための、「分析モード」があります。分析モードは実験が終了またはポーズしている時に限り利用できます。

測定モードから分析モードへの切り替えは、以下のようにして行います。

分析モードに切り替える：

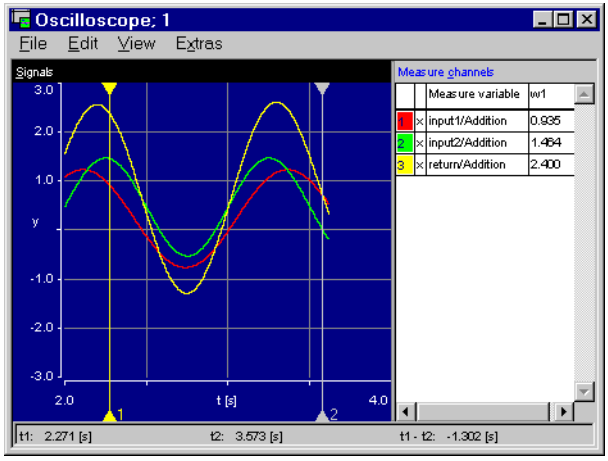
- **Edit → Analyse measure data** を選択します。

または

- **<Ctrl> + <V>** を押します。

  オシロスコープ表示領域に縦線が 2 本表示され、デフォルトでは左側に 1、右側に 2、という番号が付きます。これらの線は「分析用カーソル」と呼ばれ、この 2 つのポイントでデータを読み取ることができます。2 本のうち、アクティブなカーソルは黄色で表示され、もう一方のカーソルはグレーで表示されます。

オシロスコープウィンドウの最下部には、時間軸に関する値（両カーソルの x 軸の値とその間隔）が表示されます。



分析モードにおいては、“Measure channels”および“Bit channels”リストの表示項目を以下のように設定できます。

チャンネルリストの表示項目を設定する：

- **View → Value at active cursor** を選択します。

  “Measure Channels”および“Bit channels”リストに“w1”（カーソル 2 がアクティブな場合は“w2”）列が追加され、アクティブカーソルの位置の測定値が表示されます。

- **View → Difference between cursors** を選択します。

  両リストに“w1 - w2”（カーソル 2 がアクティブな場合は“w2 - w1”）列が追加され、アクティブカーソルの位置の測定値からもう一方のカーソル位置の測定値を引いた値が表示されます。

- 追加された情報が見えない場合は、ウィンドウサイズを調整します。
- 必要に応じて各リストの列の間の縦線を左右にドラッグし、すべての情報を読み取れるようにします。

オシロスコープに表示された測定値の分析は、以下のような手順で行います。

測定データを分析する：

- オシロスコープウィンドウを分析モードに切り替えます。
- 使用する分析カーソルを選択します。

  選択されたカーソルがアクティブカーソルになり、“Measure channel”および“Bit channels”リストの“w1”（または“w2”）列の情報が更新されます。
- 分析カーソルを、値を読み取りたい位置にドラッグします。

または

- **<←>** / **<→>**キーを使用して分析カーソルを移動します。

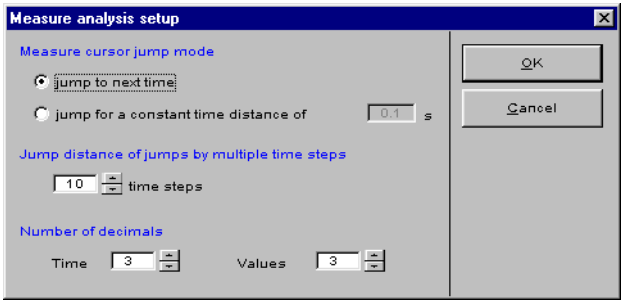
  “Measure channels”リストに分析カーソル位置の測定値（y 軸の値）が表示されます。またオシロスコープウィンドウの最下部には、分析カーソル位置の時間（x 軸の値）が表示されます。
- 必要に応じてオシロスコープウィンドウのサイズを調整します。
- さらに、必要に応じてリストの各列の境界線を左右にドラッグして、すべての情報を読み取れるようにします。

測定値を分析する際には、さまざまなオプションを使用できます。これらのオプションの設定作業は、オシロスコープが分析モードになっている時に限り行えます。

分析のセットアップを行う：

- **Edit → Analysis setup** を選択します。

  “Measure analysis setup”ダイアログボックスが開きます。



- “measure cursor jump mode”と書かれた部分で、分析カーソルを動かす際のジャンプモードを選択します。
  - **Jump to next time** を選択すると、カーソルは 1 つずつ隣の値（サンプリングポイント）に移動します。
  - **Jump for a constant time distance of...** を選択すると、カーソルは指定のステップサイズずつ移動します。たとえば、ステップサイズに 0.1 を設定すると、カーソルは x 軸上を 0.1 秒分ずつ移動します。
- “Jump distance of jumps by multiple time steps”と書かれた下の“time steps”入力フィールドに、高速ジャンプを行う際の乗数を定義します。

  たとえば、ステップサイズが 0.1 で、乗数が 3 に設定されていると、カーソルを高速モード（**<Ctrl>** キーを押しながらカーソルを移動させる）で移動する際、カーソルは x 軸上を 0.3 秒ずつ移動します。
- “Number of Decimals”と書かれた下の“Time”フィールドに、時間値の小数部の桁数を入力します。
- 隣の”Value”フィールドに、測定値の小数部の桁数を入力します。
- **OK** をクリックして設定を確定します。

  新しい設定は、いずれかの測定カーソルを動したときに有効になります。

表示開始のトリガ条件が設定されている場合、オシロスコープは、その条件が満たされるまで、測定値を表示しません。このトリガは、オシロスコープの表示にのみ関係するため、実験の実行やデータの保存にはまったく影響しません。

トリガ条件を設定できるのは、実験が停止しているときだけです。

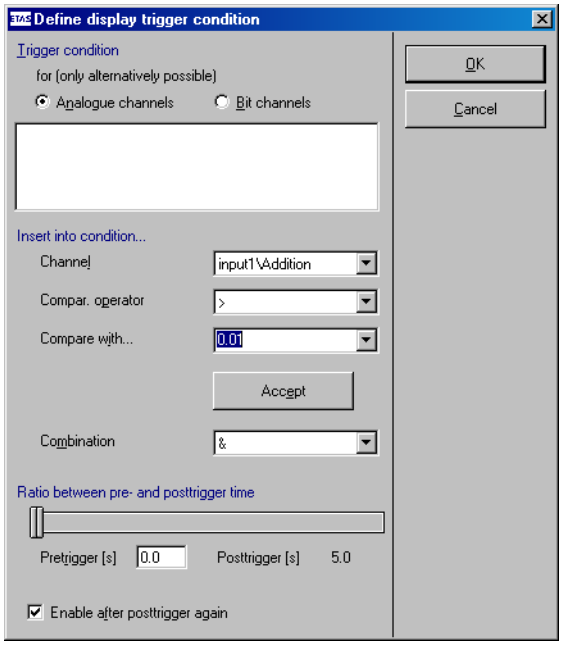
以下に、簡単なトリガの設定例を紹介します。

単純なトリガイベントを定義する：

- オシロスコープウィンドウで **Edit** → **Define trigger** を選択します。

  “Define display trigger condition” ダイアログボックスが開きます。

  トリガを定義できるのは、実験が終了またはポーズ状態になっているときだけです。



- **Analogue channels** または **Bit Channels** オプションをクリックして、トリガモードを選択します。

  トリガ条件は、アナログチャンネルかビットチャンネルのいずれか一方についてしか定義できません。

- “Channel” コンボボックスでチャンネルを選択します。
- “Compar.operator” コンボボックスで比較演算子を選択します。

  使用できる演算子は、==、>、<、exceeds、false のいずれかです。

- “Compare with”コンボボックスで、最初に選択したチャンネルと比較するチャンネルを選択します。

または

- 数値、`true`、`false` のいずれかを入力します。
- **Accept** をクリックします。

  ここまでの操作で定義された条件式がテキストフィールドに書き込まれます。
- 上記のようにコンボボックスを使用して定義する以外に、テキストフィールドに直接条件式を入力することもできます。
- **OK** をクリックしてダイアログボックスを閉じます。

  これで、トリガ条件が有効になり、実験が開始されると、オシロスコープは自動的にこの条件待ちの状態となります。

上の図の例では、`input1/addition` というチャンネルの値が `0.01` を超えたときに表示開始トリガが発生するように設定されていますが、“Compare With”コンボボックスで数値の代わりに他のチャンネルを選択し、そのチャンネルの値を超えたときにトリガを発生させることもできます。

また、複数の条件文を `and` や `or` で連結して複雑な条件を作成することもできます。

複合的なトリガイベントを定義する：

- “Defile display trigger codition”ダイアログボックスを開きます。
- **Analogue channels** または **Bit Channels** オプションをクリックして、トリガモードを選択します。

  トリガ条件は、アナログチャンネルかビットチャンネルのいずれか一方についてしか定義できません。
- 単純なトリガを定義する場合と同じ方法で、トリガ条件に使用する最初の文を定義します（650 ページ参照）。
- “Comibination”コンボボックスで、条件文を連結するための比較演算子（`&` または `|`）を選択します。
- 2 番目の条件文を定義します。

  必要に応じて、条件文を追加します。

- **OK** をクリックします。

  これで、トリガ条件が有効になり、実験が開始されると、オシロスコープは自動的にこの条件待ちの状態となります。

トリガが発生する前でも、オシロスコープのバッファには測定値が格納されているため、「プリトリガ時間」を設定することにより、トリガが発生して表示が開始される際に、その時点から所定の時間分だけさかのぼった時点からの測定値も表示されるようにすることができます。また「ポストトリガ時間」は、トリガが発生してから表示を終了するまでの時間を規定します。このプリトリガ時間とポストトリガ時間は、オシロスコープウィンドウの時間軸に対する割合で指定します。たとえば、オシロスコープウィンドウの時間軸の範囲が 2 秒に設定されている場合、プリトリガ時間とポストトリガ時間の比率を 0.4/0.6 と設定すると、実際のプリトリガ時間は 0.8 秒、ポストトリガ時間は 1.2 秒になります。

プリトリガおよびポストトリガ時間の設定は、オシロスコープウィンドウが測定モードになっていて、かつ実験がポーズまたは終了している時に、以下のようにして行います。

プリトリガ／ポストトリガ時間を設定する：

- “Define display trigger condition” ダイアログボックスを開きます。
- トリガの条件文を定義します。
- **Ratio between pre- and posttrigger time** スライダで、プリトリガ時間とポストトリガ時間の割合を調節します。

または

- スライダの下の “Pritrigger[s]” フィールドに、プリトリガ時間を秒単位で入力します。

  プリトリガ時間には、オシロスコープウィンドウの時間幅を超える値を設定することはできません。ここに入力された時間をオシロスコープの時間幅から差し引いた時間が自動的にポストトリガ時間となります。
- 測定中において、ポストトリガ時間が経過した後に再度同じトリガ条件を有効にするには、**Enable after posttrigger again** オプションをオンにします。
- **OK** をクリックします。

  設定内容が確定されます。

トリガ条件が確定されると、そのトリガ条件が自動的に有効になり、次に実験が開始されると、オシロスコープはすぐにそのトリガ待ち状態となります。

トリガ条件は、マニュアル操作で有効／無効にすることができます。

トリガ条件を有効／無効にする：

- オシロスコープウィンドウで、**Edit** メニューを開きます。

  メニューオプションの **Activate trigger** にチェックマークが付いているときは、トリガ条件が有効になっています。実験を開始すると、オシロスコープウィンドウはトリガ待ち状態となります。
- トリガ条件を無効にするには、**Activate trigger** を選択してチェックマークを消します。

  実験を開始すると、無条件で直ちに表示が開始されます。
- もう 1 度トリガ条件を有効にするには、再度 **Activate trigger** を選択します。

実験が開始され、オシロスコープが測定モードになっている場合、表示開始トリガをマニュアル操作で発生させて任意の時点で表示を開始することができます。

トリガをマニュアル操作で発生させる：

- **Edit → Trigger manually** を選択します。

  トリガ条件が満たされた場合と同じようにトリガが発生し、表示が開始されます。

オシロスコープウィンドウのイメージをクリップボードにコピーして、それを他のアプリケーションに貼り付けることができます。さらに、オシロスコープウィンドウを印刷することもできます。

オシロスコープウィンドウをコピーする：

- 実験を終了します。
- クリップボードにコピーしたいオシロスコープウィンドウを選択します。
- **File → Copy to Clipboard** を選択します。

  オシロスコープウィンドウのイメージがクリップボードにコピーされます。

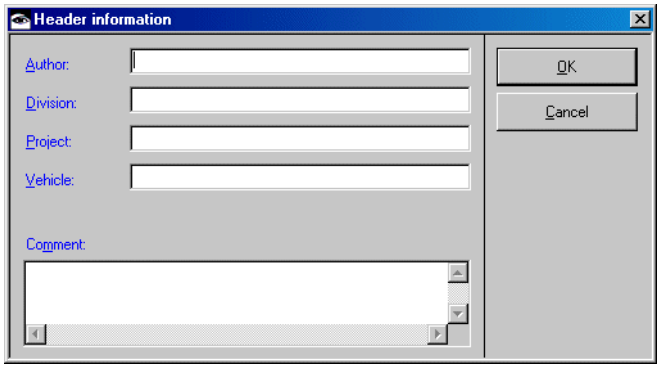
オシロスコープウィンドウのシグナル波形表示領域の内容を印刷するには、以下のようにします。

オシロスコープの表示データを印刷する：

- 実験を終了します。

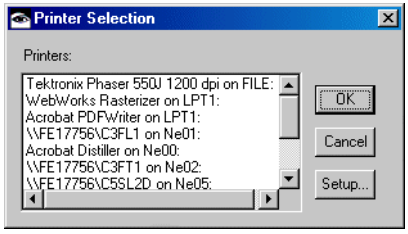
- オシロスコープウィンドウの **File** → **Print** を選択します。

  “Header information”ダイアログボックスが開きます。



- 必要に応じて、作成者、部門、プロジェクト名、車両名についての情報を入力します。
- “Comment”フィールドにはその他の情報を入力します。
- **OK** をクリックします。

  入力内容が確定され、“Printer Selection”ウィンドウが開きます。



- プリンタを選択して **OK** をクリックします。

  プリンタに、オシロスコープウィンドウの波形表示の部分が印刷されます。
  “Measure channels”および“Bit channels”リストは印刷されません。

測定されたデータは、MDF または FAMOS フォーマットのファイルに格納することができます。これらのファイルフォーマットは、他の ETAS ツールでもサポートされています。

Windows のマルチタスキング機能の制約により、データセットをファイルに保存する処理がオフライン実験中に完了しない場合があります。正確を期するために、オンライン実験環境のデータロギング機能（582 ページの「データロガー」という項を参照してください）をご利用ください。

オシロスコープのデータをファイルに保存する：

- 実験を終了します。
- 保存したいチャンネル（1 つまたは複数）を選択します。
- **File → Save Selected Channels → <format>** を選択します。

  MDF または FAMOS フォーマットを選択できます。

または

- すべてのチャンネルを保存するには、**File → Save All Channels → <format>** を選択します。

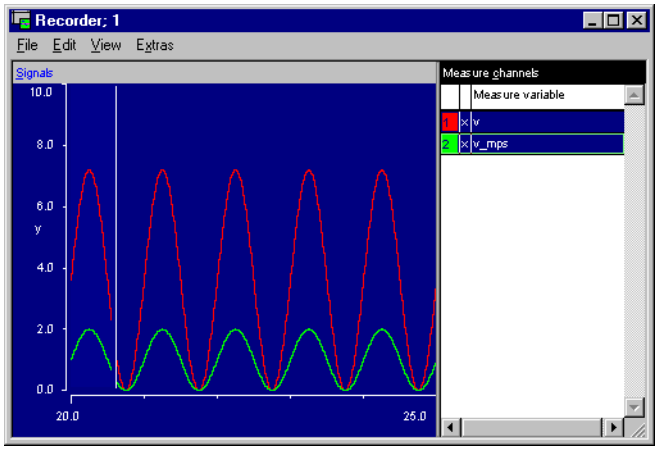
  “Store measure data”ダイアログボックスが開きます。
- 出力ファイルのパスとファイル名を指定します。
- **OK** をクリックします。

  オシロスコープウィンドウに表示されたデータがファイルに保存されます。

レコーダ

レコーダの機能はオシロスコープと似ていますが、大きな相違点は、波形表示のリフレッシュに関する点です。シグナル波形の表示がオシロスコープの右端まで到達すると、オシロスコープの場合は、それまで表示されていた波形が消去され、再度左端から新しい表示が行われますが、レコーダの場合は、すでに表示された

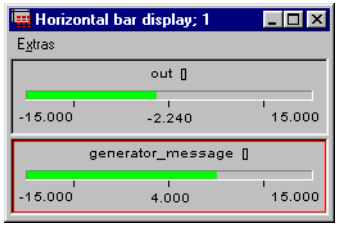
波形は画面上に残り、その上に新しいシグナル波形が上書きされます。白い縦線のカーソルが、現在の描画ポイントを示します。つまり、このカーソルより右側に表示されている波形は、前回描かれた波形です。

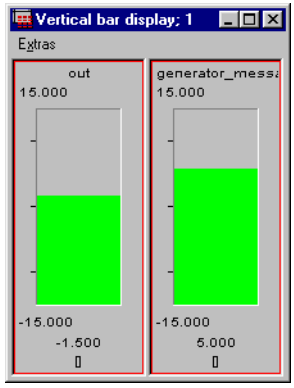


その他のオシロスコープとの相違点は、グリッドを表示できない点です。これは、レコーダの場合、現在のポイントが移動するので、それと共にグリッドを移動させなくてはならなくなるためです。表示オプション、測定データ分析およびトリガ機能は、オシロスコープと同じです。ただし、分析を行えるのは、カーソルよりも左の部分だけです。

水平および垂直棒グラフディスプレイ

棒グラフディスプレイでは、測定データがカラーバーで表示されます。各ディスプレイには上限値と下限値を指定でき、値が下限値を下回っている時は、カラーバーは表示されません。測定値はバーの下の中央に、数値でも表示されます。垂直棒グラフディスプレイは、オフライン実験環境でだけ使用できます。

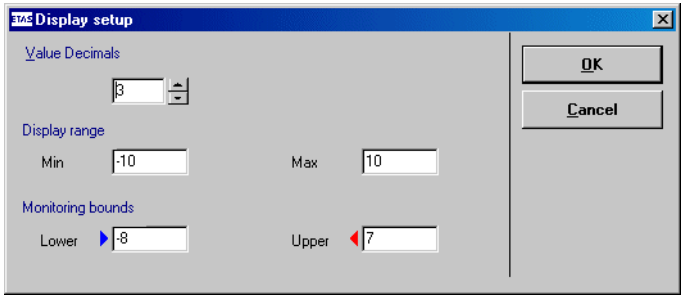




棒グラフディスプレイをセットアップする：

- セットアップしたい棒グラフディスプレイを、クリックして選択します。
- **<Ctrl>** キーを押し下げたまま複数の棒グラフディスプレイを選択すれば、それらを一度に選択することができます。
- **Extras → Setup** を選択します。

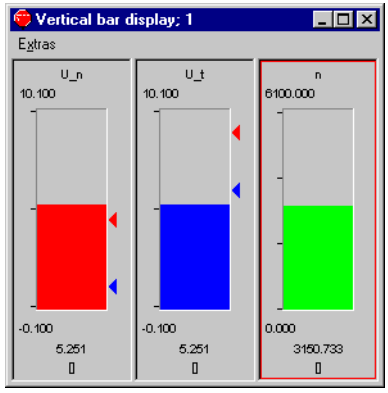
  以下のダイアログボックスが開きます。



- “Value Decimals”フィールドに、表示される数値の小数部の桁数を選択します。
- “Min”および“Max”フィールドに、表示の上下限値を設定します。
- “Lower”および“Upper”フィールドに、監視用の限界値を定義できます。

  棒グラフディスプレイ上には、監視用限界値が青と赤の三角形のマークで示されます。また測定値が監視用下限値を下回るとバーの色が青色に変わ

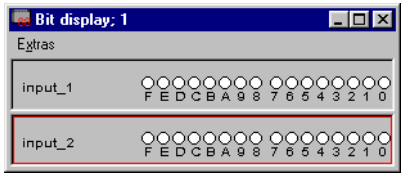
り、監視用上限値を上回ると赤色に変わります。また測定値がそれらの上下限値の範囲内であれば、バーの色は緑です。



- **OK** をクリックして設定を有効にします。

ビットディスプレイ

ビットディスプレイでは、測定値がビット列として表わされます。バイナリチャンネルの値を表示したり、測定を停止させて各ビットの状態をすばやく読み取る必要がある場合に便利です。



ビットディスプレイに関してセットアップする項目はありませんが、他のディスプレイと同様、チャンネルのコピー（635 ページ）、移動（635 ページ）、削除（636 ページ）、属性の交換（636 ページ）、ウィンドウタイトルの変更（638 ページ）、チャンネルの情報の表示（638 ページ）が可能です。

ビットディスプレイ内のチャンネルの位置を変える：

- 選択されたチャンネルをビットディスプレイウィンドウ内で上に移動させるには、**Extras → Move → Up** を選択します。

- 選択されたチャンネルをビットディスプレイウィンドウ内で下に移動させるには、**Extras → Move → Down** を選択します。

  このコマンドは、1 つのウィンドウ内に複数のチャンネルが表示されている場合にのみ有効です。

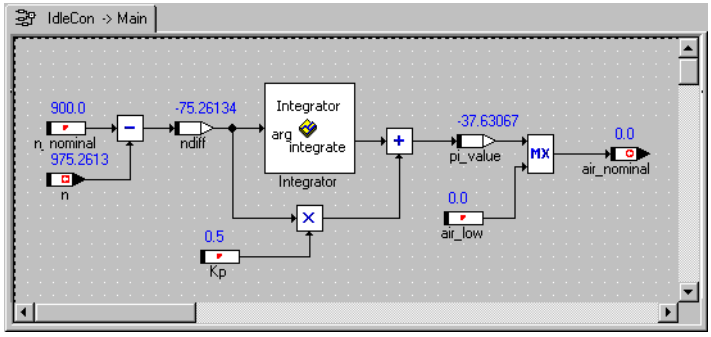
モニタ

モニタ機能を利用すれば、各エレメントの現在の数値や論理値を、ブロックダイアグラム上で参照することができます。この機能は、多くのエレメントが使用されている複雑なダイアグラムにおいて、それぞれの値が互いに影響しあう様子を追跡するような場合に特に便利です。

エレメントにモニタを割り当てる：

- オフライン実験環境の描画領域で、モニタ機能を割り当てたいエレメントを右クリックします。
- ショートカットメニューから **Monitor** を選択します。

  実験を開始すると、“Physical Experiment”ウィンドウのエレメントの上に値が表示されます。



- モニタ機能を無効にするのは、もう一度 **Monitor** コマンドを選択します。
- すべてのエレメントにモニタ機能を割り当てるには、**View → Monitor All** を選択します。
- ダイアグラム内のすべてのモニタ機能を無効にするには、**View → Delete Monitors** を選択します。

7 ドキュメントの自動生成

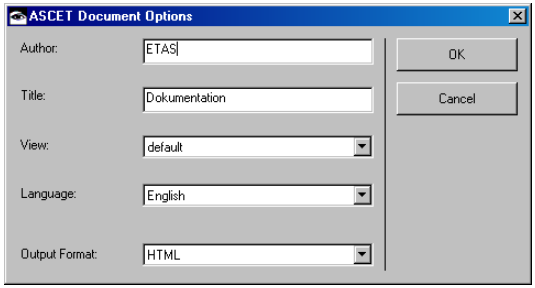
ASCET では、フォルダまたはデータベースアイテムごとにドキュメントファイルを自動生成することができます。ドキュメントファイルには、そのアイテムについての情報、つまり、インターフェースや定義ダイアグラム、および任意に入力された注釈が含まれます。ドキュメントは RTF、HTML、ASCII、Postscript などのフォーマットで作成でき、印刷したり、他のドキュメント内で使用したりすることもできます。

7.1 ドキュメントの生成

ドキュメントを生成するには、あらかじめドキュメントに含めるフォルダまたはアイテムを登録しておく必要があります。ドキュメントの生成は、1 つまたは複数のフォルダ全体、あるいは 1 つまたは複数のデータベースアイテムを対象として行えます。

ドキュメント生成用オプションを設定する：

- コンポーネントマネージャから **Tools → Documentation → Options** を選択して、“ASCET Document Options”ダイアログボックスを開きます。



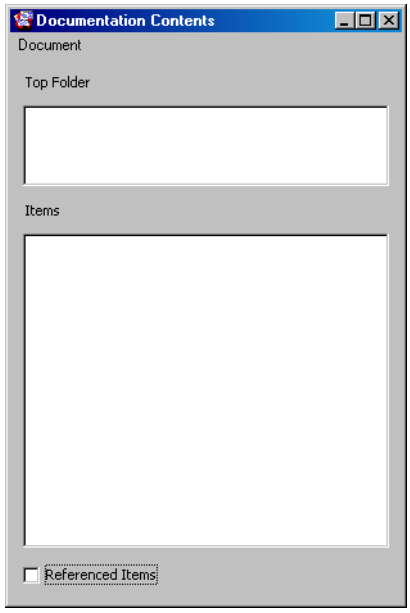
- ユーザー名を“Author”フィールドに入力し、生成するドキュメントのタイトルを“Title”フィールドに入力します。
- “View”コンボボックスで、ドキュメントファイルのビューを選択します（詳細は 7.3 項を参照してください）。
- “Language”コンボボックスで、ドキュメントの言語（英語またはドイツ語）を選択します。
- “Output Format”コンボボックスで、ドキュメントのフォーマットを選択します。

  ドキュメントファイルは、ASCII、RTF、HTML のいずれかのフォーマットで作成できます。

- **OK** をクリックします。

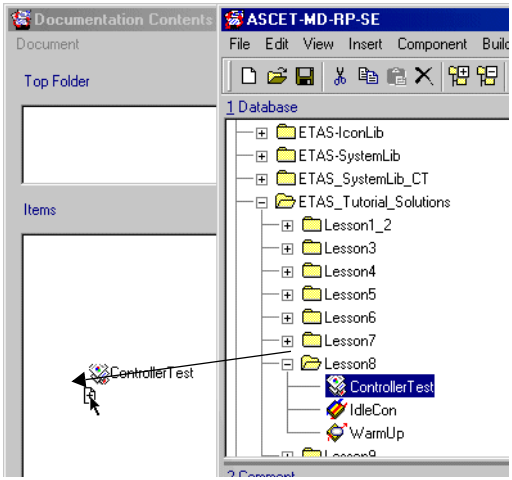
ドキュメントに含めるアイテムを選択する：

- コンポーネントマネージャから **Tools** → **Documentation** → **Contents** を選択して、“Documentation Contents”ダイアログボックスを開きます。



- ダイアログボックスの一番下にある **Referenced Items** オプションをオンにすると、新しいアイテムを追加したときに、それが参照するアイテムの情報もドキュメントに含まれます。

- コンポーネントマネージャからアイテムまたはフォルダをドラッグして“Documentation Content”ダイアログボックスの“Items”ペインにドロップします。



このダイアログボックスにフォルダをドラッグすると、そのフォルダに含まれるすべてのアイテムが“Items”ペインに表示され、トップフォルダの名前はその上の“Top Folder”ペインに表示されます。

ドキュメントに含めるアイテムを変更する：

- “Documentation Contents”ダイアログボックスを開きます。
- “Items”ペインからデータベースアイテムを選択します。
- アイテムを右クリックし、ショートカットメニューから **Delete** を選択します。

  選択されたアイテムが“Items”ペインから削除され、生成されるドキュメントに含まれなくなります。
- アイテムの“Items”ペイン内の位置を変更するには、**Move Up** または **Move Down** を選択します。

  生成されるドキュメントには、各アイテムはこのペインに表示されている順に掲載されます。

生成されるドキュメントに含まれるアイテムのリストを、出力される順番どおりにコンフィギュレーションファイルに保存しておくことができます。

ドキュメントのコンフィギュレーションを保存／ロードする：

- “Documentation Contents”ダイアログボックスで **Document → Save Document Items to File** を選択します。

  ファイル選択ダイアログボックスが開きます。ファイル拡張子として `*.cfg` が選択されています。
- コンテンツファイルのパスとファイル名を選択してから **Save** をクリックします。

  ファイルが保存されます。
- 保存してあるコンフィギュレーションをロードするには、**Document → Load Document Items From File** を選択します。

ドキュメントを生成する：

- “Documentation Contents”ダイアログボックスの **Document → Generate Document** を選択します。

  ドキュメントファイル名を入力するダイアログボックスが開きます。
- ファイル名を入力して **OK** をクリックします。

  “Documentation Contents”ダイアログボックスに表示されているすべてのアイテムについて、ドキュメントが生成されます。

生成されたファイルは、ASCET のドキュメントディレクトリに格納されます。このディレクトリは“Options”ウィンドウの“Documentation”ノードで指定します（53 ページの 2.2.5 項を参照してください）。ドキュメントファイルを同じ名前で再度保存すると、古いファイルは上書きされます。

生成されたドキュメントを参照する：

- “Documentation Contents”ダイアログボックスの **Document → Preview Document** を選択します。

  生成されたファイルが、そのドキュメントタイプに適したビューアで開きます。

7.2 ドキュメントファイルの出力フォーマット

生成されるドキュメントは、すべてドキュメントディレクトリに書き込まれ、このディレクトリは、“Options”ウィンドウの“Documentation”ノードで指定します（53 ページの 2.2.5 項を参照してください）。ASCET がドキュメントファイル名として使用する名前は、デフォルトでは常に同じなので、古いファイルが同じディレクトリに残っていると、それらは上書きされてしまいます。

ASCII フォーマット

出力フォーマットとして ASCII を選択すると、生成されるテキストはすべて `text.txt` というファイルに書き込まれます。ドキュメント化されるデータベースアイテムにダイアグラムが含まれている場合、それらは 1 つずつ別々の Postscript ファイルに書き込まれ、テキストファイルには各ファイルの名前が格納されます。クラスに C コードが含まれている場合は、それはテキストファイルに格納されます。

ASCET により生成された ASCII テキストを任意のエディタやワードプロセッサにロードして、ASCET で開発したシステムに関するドキュメントを作成することができます。

RTF フォーマット

RTF は、フォーマットされたドキュメントを交換する際の標準的なフォーマットで、Microsoft Word や WordPerfect など、ほとんどのワードプロセッサプログラムで読み込んで編集することができます。このフォーマットのドキュメントは、`docu.rtf` というファイルに書き込まれます。

HTML フォーマット

HTML は、World Wide Web で情報を交換する際の標準的なフォーマットです。HTML ドキュメントは Netscape や Internet Explorer など、一般的な World Wide Web ブラウザで見ることができます。生成された HTML ソーステキストは `docu.htm` というファイルに書き込まれます。ダイアグラムはすべて、GIF フォーマットで格納されます。

Postscript フォーマット

出力フォーマットとして Postscript を選択すると、すべての情報が 1 つの Postscript ファイルに書き込まれ、ダイアグラムはテキスト内に配置されます。出力ファイルは `docmain.ps` という名前になります。

7.3 ビュー

ASCET のブロックダイアグラムエディタとステートマシンエディタでは「ビュー」と呼ばれる機能が利用できます。いくつかの「ビュー」を定義してこれを切り替えることにより、ダイアグラムや自動生成されるドキュメントに出力される情報を、状況に応じて任意に限定することができます。

個々のビューでは、ブロックダイアグラム、ステートマシン、ドキュメントへの出力（表示）について、一般的な設定やエレメントタイプごとの設定を行えます。

以下のような情報の表示 / 非表示を切り替えることができます。

- シーケンスコールと接続線
- メソッドローカル エレメントのメソッド名と、プロセスローカル エレメントのプロセス名
- グラフィカルコメント
- 各ビューごとに、個々のエレメントグループのデフォルト表示モードを定義可能（例：すべての `cont` パラメータを `As Line` モードで表示）

各エレメントの表示設定については、224 ページの「ダイアグラムアイテムの各ビューでの表示モードを編集する：」を参照してください。

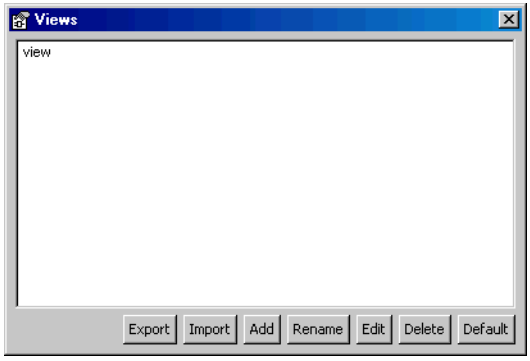
あるビューを選択すると、そのビューに定義された一般的な設定とエレメント固有の設定が、すべて有効になります。

一般事項

ビューを管理する：

- コンポーネントマネージャから **Tools → Views** を選択します。

  “Views” ウィンドウを開きます。



  最初は `view` という名前のビューだけが表示されます。
- **Add** ボタンをクリックします。
- 新しいビューの名前を入力し、**<Enter>** を押します。
- ビューの名前を変更するには、**Rename** をクリックします。

- ビューを削除するには、**Delete** ボタンをクリックします。

注記

Add、**Rename**、**Edit**、**Delete**、**Default** ボタンと同じ機能のコマンドがショートカットメニューにも含まれています。

- 入力内容を確定してダイアログボックスを閉じるには、**OK** をクリックします。
- 入力内容をキャンセルしてダイアログボックスを閉じるには、**Cancel** をクリックします。

複数のビューが存在する場合、そのうちの任意のビューをデフォルトビューとして指定することができます。ブロックダイアグラムでコンポーネントを開くと、このデフォルトビューが有効となります。

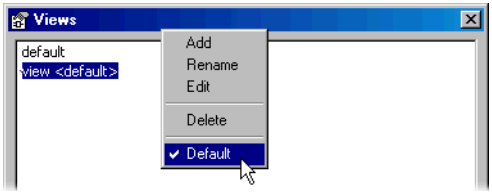
デフォルトビューを選択する：

- “Views”ウィンドウで、いずれかのビューを選択します。
- **Default** をクリックします。

または

- ショートカットメニューから **Default** を選択します。

  選択されたビューがデフォルトビューに指定され、そのビューのショートカットメニューの **Defualt** コマンドにチェックマークが付きます。



- デフォルト指定を解除するには、以下の操作を行います。
  - 現在のデフォルトビューを選択します。
  - **Default** ボタンをクリックします。

    そのビューのデフォルトビュー指定が解除され、どのビューもデフォルトビューでなくなります。

- 別のビューをデフォルトビューにするには、以下の操作を行います。
  - 別のビューを選択します。
  - **Default** ボタンをクリックします。

    それまでデフォルトビューに指定されていたビューのデフォルト指定が解除され、新しいビューがデフォルトビューになります。

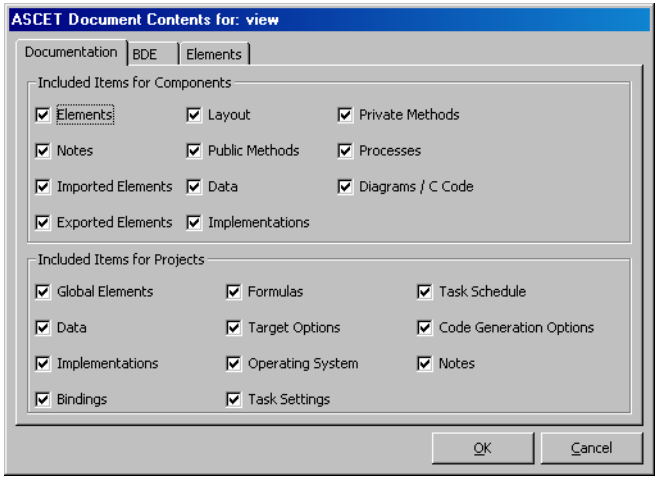
ブロックダイアグラムエディタまたはステートマシンエディタで、以下のようにビューを変更できますが、これによってビューのデフォルト指定が影響を受けることはありません。

各ビューにおける表示項目は、以下のようにして設定します。

ビューを編集する：

- “Views”ウィンドウで、編集したいビューを選択します。
- **Edit** ボタンをクリックします。

  “ASCET Document Contents for”ダイアログボックスが開きます。このダイアログボックスには 3 つのタブがあり、それぞれドキュメント、ブロックダイアグラム／ステートマシンエディタ、エレメントについて、表示／非表示を設定できます。



- 各タブで、ビューの設定を編集します。

  各オプションについては“671 ページの「ドキュメント用オプション」、672 ページの「エディタ用オプション」、673 ページの「エレメントタイプ用オプション」を参照してください。
- **OK** をクリックします。

上記のようにして設定したビューは、データベースアイテムのエクスポート時にエクスポートの対象となりません。その代わり、“Views”ウィンドウにおいて、ビュー（1 つまたは複数）を 1 つの XML ファイルに出力することができます。

ビューをエクスポートする：

- “Views”ウィンドウで、エクスポートしたいビュー（複数可）を選択します。
- **Export** ボタンをクリックします。

  “Export File”ダイアログボックスが開き、ASCET のエクスポートディレクトリ（61 ページ参照）に保存されている XML ファイルがすべて表示されます。
- エクスポート先のパスとファイル名（拡張子は `*.xml`）を指定します。
- **Save** をクリックします。

  選択されていたビューがファイルに書き込まれます。

ビューのエクスポートファイルの内容は、非常に単純な構成になっています。ビューの名前が `<View>` エレメントに格納され、“ASCET Document Contents”ダイアログボックスの各タブの設定内容が `<Components>` および `<Projects>` エレメントに格納されます。これらの各タブ内の各オプションは、それぞれのエレメント内の各属性に格納されます。この際の順番は、まずタブ上の左側の列のオプションが上から順に格納され、次に右側の列が格納されます。以下に例を示します。

```
<Views>
<View name="view">
<DocumentationSettings>
  <Components notes="true"
        layout="true"
        publicMethods="true"
        privateMethods="true"
        processes="true"
        graphic="true"
        elements="true"
        importedElements="true"
```

```
            exportedElements="true"
            data="true"
            implementation="true" />
      <Projects notes="true"
            codegenOptions="true"
            targetOptions="true"
            operatingSystem="true"
            taskSettings="true"
            taskSchedule="true"
            elements="true"
            bindings="true" data="true"
            implementation="true"
            formulas="true" />
   </DocumentationSettings>
   <BDESettings sequenceCalls="true"
         processNameLocals="false"
         graphicalComments="true" />
   <ElementDefaults>
      <AsLine>
         <Element type="Scalar"
               modelType="Continuous"
               kind="Parameter"
               scope="*"
               existence="Non-Virtual"
               dependency="*"
               memory="*"
               calibration="*"/>
      </AsLine>
      <Invisible></Invisible>
      <Contour></Contour>
      <HideContents></HideContents>
   </ElementDefaults>
   </View>
   </Views>
```

ファイル内の必須アイテムは、ビューの名前が格納される <View> エレメントのみで、他のエレメントと属性は省略できます。省略された属性は、ファイルのインポート時には true として扱われます。

1 つのファイルに複数のビューをエクスポートすると、ビューごとに <View> エレメントが作成されます。

エクスポートされたビューは、以下のようにしてインポートすることができます。エクスポートファイルに複数のビューが含まれる場合、それらすべてがインポートされます。

ビューをインポートする：

- “Views”ウィンドウで、**Import** ボタンをクリックします。

  インポートされるビューと同名のビューがすでに存在している場合、それらのビューは上書きされるので、ワーニングメッセージが表示されます。
- **Yes** をクリックしてワーニングメッセージを閉じます。

  “Inport File”ダイアログボックスが開き、ASCET のエクスポートディレクトリ（61 ページ参照）に保存されている XML ファイルがすべて表示されます。
- インポートしたいファイルを選択します。
- **Open** をクリックします。

  ファイルに格納されているビューが ASCET データベースにインポートされます。

ドキュメント用オプション

“ASCET Document Contents for”ダイアログボックスの“Documentation”タブで、コンポーネント用のアイテム（上部）とプロジェクト用のアイテム（下部）について表示／非表示を設定します。

コンポーネント用設定アイテム：

Elements	コンポーネントのすべてのエレメント
Notes	ビューに関連付けられている注釈
Imported Elements	すべてのインポートエレメント（メッセージとグローバル変数）
Exported Elements	すべてのエクスポートエレメント
Layout	コンポーネントのブロックレイアウト
Public methods	パブリックダイアグラム内にあるすべてのメソッド
Data	すべてのデータセット
Implementations	すべてのインプリメンテーション
Private Methods	プライベートダイアグラム内にあるクラスのすべてのメソッド
Processes	すべてのプロセス（モジュールの場合のみ）

Diagrams / C Code	コンポーネント内のブロックダイアグラムまたはステートマシンダイアグラム（含まれている場合のみ）、またはクラスの C コード（含まれている場合のみ）

プロジェクト用設定アイテム：

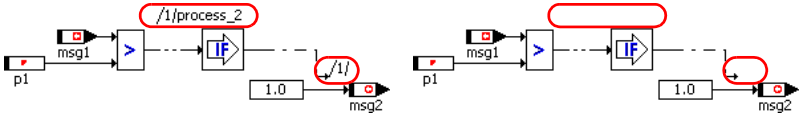
Global Elements	プロジェクト内で定義されているすべてのグローバルエレメント
Data	プロジェクト内のすべてのデータセット
Implementations	プロジェクト内のすべてのインプリメンテーション
Binding	プロジェクト内で宣言されているグローバルエレメントがどのエレメントに結びつけられているかという情報
Formulas	プロジェクト内で宣言されているすべてのグローバル変換式
Target Options	ターゲットオプション
Operating System	オペレーティングシステムエディタでのタスクとプロセスに関する設定
Task Settings	各タスクの設定
Task Schedule	タスクのスケジューリングについての情報
Code Generation Options	コード生成オプション
Notes	ビューに関連づけられている注釈内のセグメント

エディタ用オプション

“BDE”タブでは、ブロックダイアグラムエディタとステートマシンエディタについて、以下の表示オプションを設定します。

- **Show Sequence Calls**

  このオプションがオフ（無効）になっていると、そのビューにおいては、シーケンスウコールと接続線がすべて非表示となります。

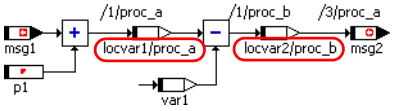


オプション = オン（有効）　　オプション = オフ（無効）

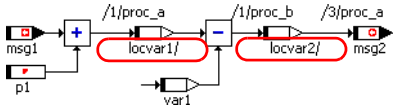
このオプションは、メニューコマンド **Sequence Calls → Show → *** と **Sequence Calls → Hide → *** の機能とは異なります。現在のビューにおいて **Show Sequence Calls** が無効になっている場合、これらのメニューコマンドは使用できません。

- **Show Method/Process for Locals**

  このオプションがオフ（無効）になっていると、そのビューにおいては、メソッドローカル エレメント内のメソッド名と、プロセスローカル エレメント内のプロセス名がすべて非表示となります。



オプション = オン（有効）

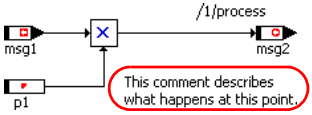


オプション = オフ（無効）

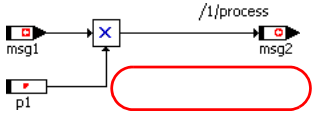
- **Show Graphical Comments**

  このオプションがオフ（無効）になっていると、そのビューにおいては、ダイアグラム内のコメントがすべて非表示となります。

  このオプションをオン（有効）にすると、コメントが表示されます。ただしエレメントオプションで 1 つ以上のコメントが `Invisible` モードに設定されていると、それらのコメントは無条件に非表示となります。各エレメントの表示設定については、224 ページの「ダイアグラムアイテムの各ビューでの表示モードを編集する：」を参照してください



オプション = オン（有効）



オプション = オフ（無効）

エレメントタイプ用オプション

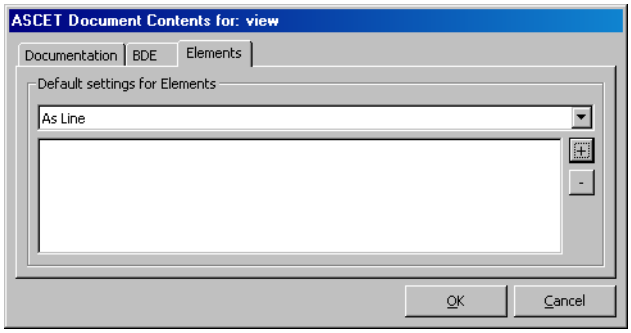
“Elements”タブでは、ブロックダイアグラムエディタとステートマシンエディタで使用されるいくつかのエレメントグループについての表示設定を行います。この設定はグローバル設定で、データベース内で該当するエレメントグループに属するエレメントのうち、個別の表示設定が行われていないものすべてに適用されます。

注記

ダイアグラムエレメントのうち `Normal` 以外のモードが設定されているエレメント（224 ページの「ダイアグラムアイテムの各ビューでの表示モードを編集する：」を参照してください）は、このグローバル設定の影響を受けません。

エレメントグループのグローバル設定を行う：

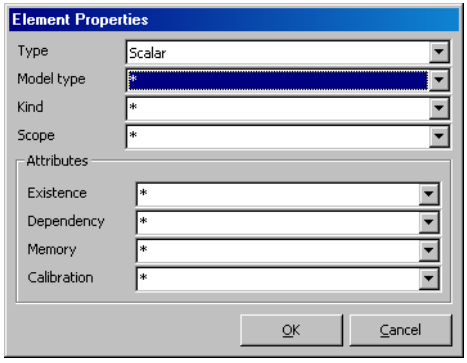
- 設定したいビューの“ASCET Document Content for”ダイアログボックスを開き、“Elements”タブを選択します。



- コンボボックスで、エレメントグループに割り当てる表示モードを選択します（224 ページを参照してください）。

- + ボタンをクリックして、エレメントグループを追加します。

  “Element Properties”ダイアログボックスが開きます。



- タイプ、モデル型、種類（kind）、スコープ、属性を選択します。

  これらのフィールドに表示される名前は、エレメントエディタ（441 ページの図 4-2 参照）に表示されるものと同じです。

  特定のアイテムが選択されていない場合、つまりフィールドが * のままになっている場合は、“Elements”タブで選択されている表示モードがすべてのアイテムに対して適用されます。
- **OK** をクリックして設定を確定し、ダイアログボックスを閉じます。

  “ASCET Document Content”ダイアログボックスの下側のフィールドに、選択されたエレメントが表示されます。
- **OK** をクリックして設定を確定し、ダイアログボックスを閉じます。

同じエレメントを複数のグループに割り当てた場合、実際にそのエレメントがどのように表示されるかは、ASCET がビューオプションを処理する順番で決まるため、明確ではありません。たとえば、エクスポートされる（`Exported`）システム定数が `As Line` グループに割り当てられていて、論理（`Logic`）システム定数が `Hide` グループに割り当てられている場合、エクスポートされる論理システム定数は、ラインとして表示される場合もあれば、非表示となる場合もあります。

注記

エレメントグループを定義する際は、同じエレメントを複数のグループに割り当てないよう、注意してください。

7.4 注釈

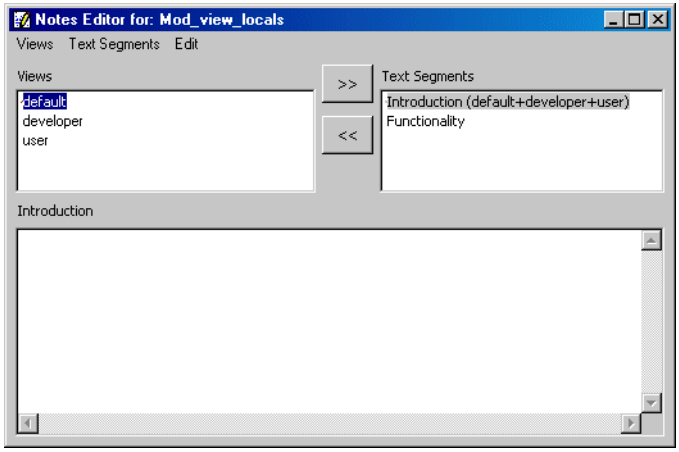
フォルダやデータベースアイテムに対して、「注釈」を付加することができます。「コメント」はブロックダイアグラムに組み込まれ、そのブロックダイアグラムと共に印刷されますが、注釈は、フォルダまたはデータベースアイテムに割り当てられ、生成されるドキュメントのテキストの一部になります。注釈には、機能記述に含めることのできない情報を含めることができます。

注釈を作成する：

- 注釈を付加するフォルダまたはデータベースアイテムを選択します。

- **Component → Notes** を選択して、新しい注釈を追加します。

  注釈用エディタが開きます。



注釈は複数のテキストセグメントで構成され、各テキストセグメントを 1 つまたは複数のビューに割り当てることができ、これによって、同じテキストを何度も入力する必要がなくなります。生成されるドキュメントには、選択されているビューに属するすべてのテキストセグメントが含まれます。

テキストセグメントを作成する：

- **Text Segments → Add** を選択します。
- 作成するテキストセグメントの名前を入力してから **<Enter>** を押します。

  入力したテキストセグメントの名前が“Text Segments”ペインにリストアップされます。新しいテキストセグメントは、現在選択されているすべてのビューに割り当てられます。
- “Text Segments”ペインの下にあるテキスト入力ペインに、セグメントのテキストを入力します。

  現在のテキストセグメントの名前が、テキスト入力ペインのタイトルとして表示されます。
- テキストセグメントの名前を変更するには、**Text Segment → Rename** を選択します。
- テキストセグメントを削除するには、**Text Segment → Delete** を選択します。

テキストセグメントを編集する：

- ウィンドウ内のテキストの編集には、**Edit → Cut**、**Copy**、**Paste** を使用できます。

  これらの操作では Windows の標準クリップボードが用いられるので、他の Windows アプリケーションとテキストを交換することができます。
- テキスト入力ペイン内のすべてのテキストを選択するには、**Edit → Select All** を選択します。
- 外部ファイルからデータを読み込むには、**Edit → Read from File** を選択します。

  この操作ではソーステキストのコピーが生成されるので、ソースファイルの内容は変更されません。
- 注釈のテキストを外部ファイルに書き込むには、**Edit → Write to File** を選択します。

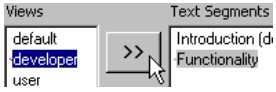
  外部ファイルのパスとファイル名の入力が要求され、ユーザーがこれに応えて入力した位置に外部ファイルが格納されます。
- 現在のテキストセグメントを保存するには、**Edit → Save** を選択します。

作成されたテキストセグメントは、“Views”ペインで選択されているすべてのビューに割り当てられます。

テキストセグメントをビューに割り当てる：

- “Text Segments”ペインからテキストセグメントを選択します。
- “Views”ペインからビューを選択します。
- **Views → Assign Views** を選択します。

または



- **>>** ボタンをクリックします。

  1 つのテキストセグメントを複数のビューに割り当てるには、**<Ctrl>** キーを押し下げたまま“Views”ペインから複数のビューを選択してます。

ビューの割り当てを解除する：

- テキストセグメントを選択します。
- “Views”ペインからビューを選択します。
- **Text Segment → Deassign Views** を選択します。

または

- **<<** ボタンをクリックします。

テキストセグメントの割り当て状況を表示する：

- “Views”ペインからビューを選択します。
- **Views** → **Select Text Segments** を選択します。

  “Text Segments”ペインで、選択されたビューに割り当てられているテキストセグメントが強調表示されます。
- “Text Segments”ペインからテキストセグメントを選択します。
- **Text Segments** → **Select Assigned Views** を選択します。

  “Views”ペイン内で、選択されたテキストセグメントが割り当てられているすべてのビューが強調表示されます。

索引

お

さ

し

へ

ほ

ま

め