User's Manual

CubeSuite+ V1.00.00

統合開発環境

ユーザーズマニュアル　V850 ビルド編

対象デバイス

V850 マイクロコントローラ

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2011.04

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

このマニュアルの使い方

このマニュアルは，V850 マイクロコントローラ用アプリケーション・システムを開発する際の統合開発環境である CubeSuite+について説明します。

CubeSuite+は，V850 マイクロコントローラの統合開発環境（ソフトウエア開発における，設計，実装，デバッグなどの各開発フェーズに必要なツールをプラットフォームである IDE に統合）です。統合することで，さまざまなツールを使い分ける必要がなく，本製品のみを使用して開発のすべてを行うことができます。

対 象 者　このマニュアルは，CubeSuite+を使用してアプリケーション・システムを開発するユーザを対象としています。

目　　的　このマニュアルは，CubeSuite+の持つソフトウエア機能をユーザに理解していただき，これらのデバイスを使用するシステムのハードウエア，ソフトウエア開発の参照用資料として役立つことを目的としています。

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

読 み 方　このマニュアルを読むにあたっては，電気，論理回路，マイクロコンピュータに関する一般的知識が必要となります。

凡　　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称に上線）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文中の補足説明
数の表記	：10 進数　…　××××
	16 進数　…　0x××××

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

資　料　名		資料番号	
		和　文	英　文
CubeSuite+ 統合開発環境 ユーザーズ・マニュアル	起動編	R20UT0545J	R20UT0545E
	78K0 設計編	R20UT0546J	R20UT0546E
	78K0R 設計編	R20UT0547J	R20UT0547E
	RL78 設計編	R20UT0548J	R20UT0548E
	V850 設計編	R20UT0549J	R20UT0549E
	R8C 設計編	R20UT0550J	R20UT0550E
	78K0 コーディング編	R20UT0551J	R20UT0551E
	RL78,78K0R コーディング編	R20UT0552J	R20UT0552E
	V850 コーディング編	R20UT0553J	R20UT0553E
	コーディング編（CX コンパイラ）	R20UT0554J	R20UT0554E
	R8C コーディング編	R20UT0576J	R20UT0576E
	78K0 ビルド編	R20UT0555J	R20UT0555E
	RL78,78K0R ビルド編	R20UT0556J	R20UT0556E
	V850 ビルド編	このマニュアル	R20UT0557E
	ビルド編（CX コンパイラ）	R20UT0558J	R20UT0558E
	R8C ビルド編	R20UT0575J	R20UT0575E
	78K0 デバッグ編	R20UT0559J	R20UT0559E
	78K0R デバッグ編	R20UT0560J	R20UT0560E
	RL78 デバッグ編	R20UT0561J	R20UT0561E
	V850 デバッグ編	R20UT0562J	R20UT0562E
	R8C デバッグ編	R20UT0574J	R20UT0574E
	解析編	R20UT0563J	R20UT0563E
	メッセージ編	R20UT0407J	R20UT0407E

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料を使用してください。

〔メ　モ〕

〔メ　モ〕

〔メ　モ〕

目　　次

第1章 概 説

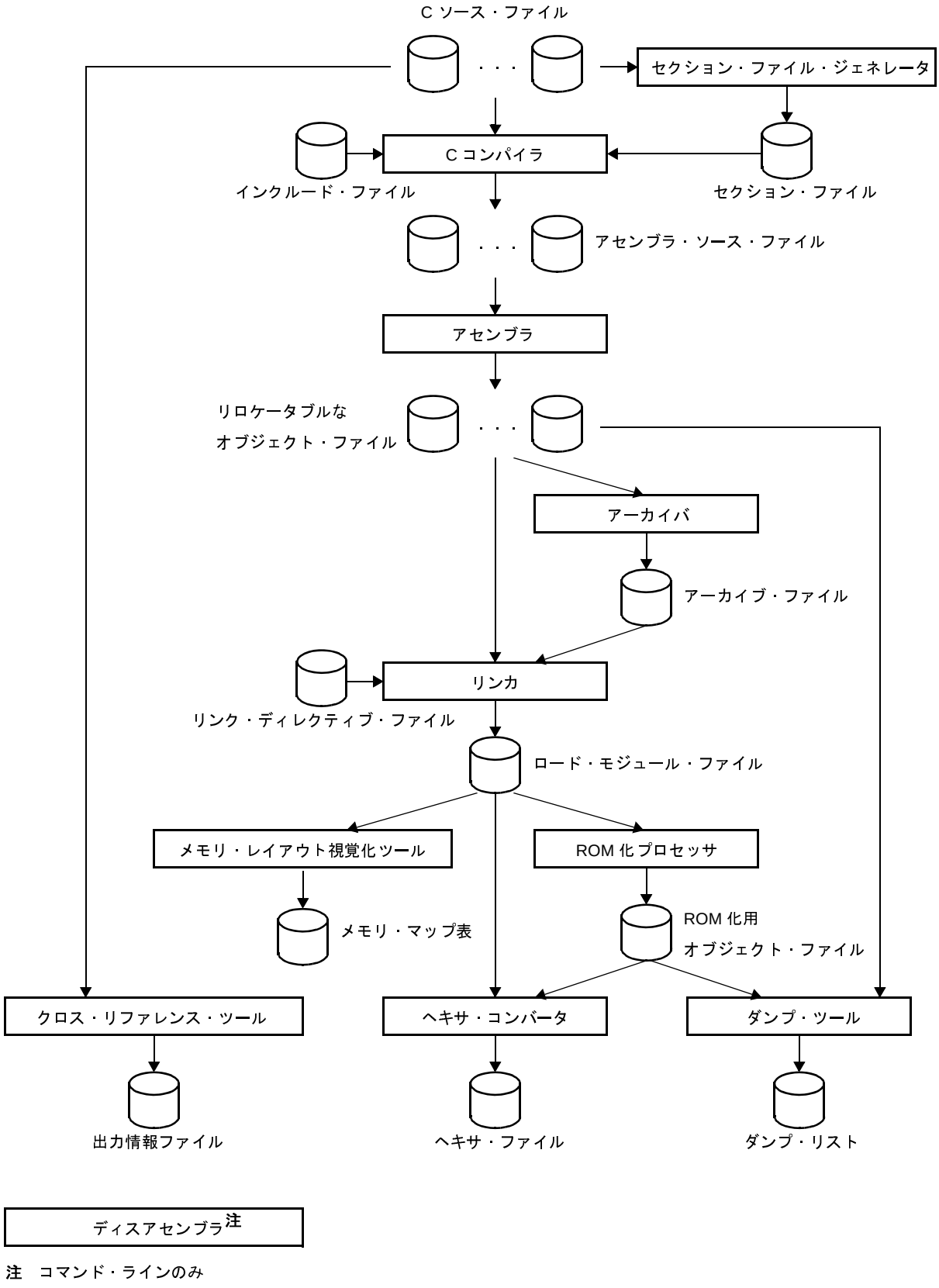
この章では，ビルド・ツールの概要について説明します。

1.1 概 要

ビルド・ツールは，本製品が提供しているコンポーネントの一種であり，GUI ベースで各種情報を設定することにより，ソース・ファイルから ROM 化用オブジェクト・ファイル，ロード・モジュール・ファイル，ヘキサ・ファイル，アーカイブ・ファイルなどを，目的に応じて生成することができます。

以下に，ビルド・ツールの処理の流れを示します。

図 1—1 ビルド・ツールの処理の流れ



注 コマンド・ラインのみ

1.2 特 長

以下に，ビルド・ツールの特長を示します。

- 最適化機能

コンパイル時に，コード・サイズ優先，実行速度優先などの最適化を行うことにより，効率の良いオブジェクト・モジュール・ファイルを生成することができます。

6 段階の最適化レベルを選択でき，また，ソースごとに最適化レベルを設定することも可能です。

- 組み込みシステムの開発に最適な機能

割り込み処理やリアルタイム OS のタスクを，C 言語で記述することが可能です。

マイコンの周辺ハードウエアへのアクセスを通常の変数アクセスのように扱うことができます。

C コンパイラが使用する汎用レジスタの本数を制限することにより，割り込み時のレジスタ退避／復帰処理のオーバヘッドを軽減します（レジスタ・モード）。

アライメントによる構造体／共用体のメンバ間のホールを詰め，すでに決められたアライメントによる構造体，共用体を扱うことが可能です（構造体／共用体パッキング機能）。

第2章 機 能

この章では，CubeSuite+ を使用したビルドの手順，およびビルドに関する主な機能について説明します。

2.1 概 要

ここでは，ロード・モジュール，およびユーザ・ライブラリの作成手順について説明します。

2.1.1 ロード・モジュールを作成する

ロード・モジュールの作成手順を以下に示します。

(1) プロジェクトの作成／読み込み

プロジェクトの新規作成，または既存のプロジェクトの読み込みを行います。

備考 プロジェクトの新規作成，および既存のプロジェクトの読み込みについての詳細は，「CubeSuite+ 起動編」を参照してください。

(2) ビルド対象プロジェクトの設定

ビルド対象とするプロジェクトを設定します（「2.17 ビルドの設定をする」参照）。

備考 1. プロジェクトにサブプロジェクトがない場合，そのプロジェクトは常にアクティブになります。

2. ビルド・モードを設定する場合は，ビルド・モードを追加してください（「2.17.5 ビルド・モードを追加する」参照）。

(3) ビルド対象ファイルの設定

ビルド対象ファイルの追加／削除，依存関係の更新などを行います（「2.3 ビルド対象ファイルを設定する」参照）。

備考 1. ユーザ・ライブラリのプロジェクトへの追加方法については，「2.7.1 ユーザ・ライブラリを追加する」を参照してください。

2. オブジェクト・モジュール・ファイル，およびライブラリ・ファイルのリンク順は，ユーザが設定することもできます（「2.17.1 ファイルのリンク順を設定する」参照）。

(4) ロード・モジュールの出力指定

生成するロード・モジュールの種類を設定します（「2.4 出力ファイルの種類を設定する」参照）。

(5) ビルド・オプションの設定

コンパイラ，アセンブラ，リンカなどに対するオプションを設定します（「2. 5 コンパイル・オプションを設定する」，「2. 6 アセンブル・オプションを設定する」，「2. 7 リンク・オプションを設定する」参照）。

(6) ビルドの実行

ビルドを実行します（「2. 18 ビルドを実行する」参照）。

ビルドには，次の種類があります。

- ビルド（「2. 18. 1 更新ファイルのビルドを実行する」参照）
- リビルド（「2. 18. 2 すべてのファイルのビルドを実行する」参照）
- ラピッド・ビルド（「2. 18. 3 他の処理と平行してビルドを実行する」参照）
- バッチ・ビルド（「2. 18. 4 ビルド・モードを一括してビルドを実行する」参照）

備考 ビルド処理前，およびビルド処理後に実行したいコマンドがある場合は，プロパティ パネルの［共通オプション］タブの［その他］カテゴリにおいて，［ビルド前に実行するコマンド］プロパティ，および［ビルド後に実行するコマンド］プロパティを設定してください。
ファイル単位でビルド処理前，およびビルド処理後に実行したいコマンドがある場合は，［個別コンパイル・オプション］タブ（C ソース・ファイルの場合），および［個別アセンブル・オプション］タブ（アセンブラ・ソース・ファイルの場合）において設定することができます。

(7) プロジェクトの保存

プロジェクトの設定内容をプロジェクト・ファイルに保存します。

備考 プロジェクトの保存についての詳細は，「CubeSuite+ 起動編」を参照してください。

2. 1. 2 ユーザ・ライブラリを作成する

ユーザ・ライブラリの作成手順を以下に示します。

(1) プロジェクトの作成／読み込み

プロジェクトの新規作成，または既存のプロジェクトの読み込みを行います。

プロジェクトを新規作成する際は，ライブラリ用のプロジェクトを設定します。

備考 プロジェクトの新規作成，および既存のプロジェクトの読み込みについての詳細は，「CubeSuite+ 起動編」を参照してください。

(2) ビルド対象プロジェクトの設定

ビルド対象とするプロジェクトを設定します（「2. 17 ビルドの設定をする」参照）。

備考 1. プロジェクトにサブプロジェクトがない場合，そのプロジェクトは常にアクティブになります。

2. ビルド・モードを設定する場合は，ビルド・モードを追加してください（「2. 17. 5 ビルド・モードを追加する」参照）。

(3) ビルド対象ファイルの設定

ビルド対象ファイルの追加／削除，依存関係の更新などを行います（「2.3 ビルド対象ファイルを設定する」参照）。

(4) ビルド・オプションの設定

コンパイラ，アセンブラ，アーカイバに対するオプションを設定します（「2.5 コンパイル・オプションを設定する」，「2.6 アセンブル・オプションを設定する」，「2.10 アーカイブ・オプションを設定する」参照）。

備考 デバイス共通のライブラリを作成する場合は，プロパティ パネルの［共通オプション］タブの［出力ファイルの種類と場所］カテゴリにおいて，［デバイス共通オブジェクトを出力する］プロパティを設定してください。

(5) ビルドの実行

ビルドを実行します（「2.18 ビルドを実行する」参照）。

ビルドには，次の種類があります。

- ビルド（「2.18.1 更新ファイルのビルドを実行する」参照）
- リビルド（「2.18.2 すべてのファイルのビルドを実行する」参照）
- ラピッド・ビルド（「2.18.3 他の処理と平行してビルドを実行する」参照）
- バッチ・ビルド（「2.18.4 ビルド・モードを一括してビルドを実行する」参照）

備考 ビルド処理前，およびビルド処理後に実行したいコマンドがある場合は，プロパティ パネルの［共通オプション］タブの［その他］カテゴリにおいて，［ビルド前に実行するコマンド］プロパティ，および［ビルド後に実行するコマンド］プロパティを設定してください。

ファイル単位でビルド処理前，およびビルド処理後に実行したいコマンドがある場合は，［個別コンパイル・オプション］タブ（C ソース・ファイルの場合），および［個別アセンブル・オプション］タブ（アセンブラ・ソース・ファイルの場合）において設定することができます。

(6) プロジェクトの保存

プロジェクトの設定内容をプロジェクト・ファイルに保存します。

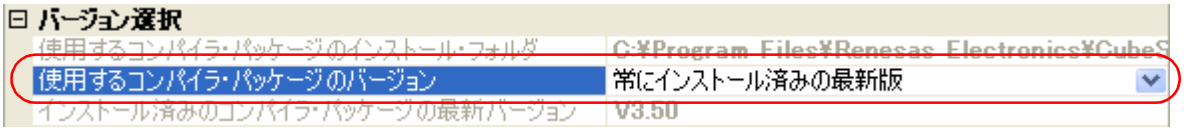
備考 プロジェクトの保存についての詳細は，「CubeSuite+ 起動編」を参照してください。

2.2 ビルド・ツールのバージョンを変更する

プロジェクト（メイン・プロジェクト，またはサブプロジェクト）で使用するビルド・ツール（コンパイラ・パッケージ）のバージョンを変更することができます。

プロジェクト・ツリー でビルド・ツール・ノードを選択したのち，プロパティ パネルの［共通オプション］タブを選択します。［バージョン選択］カテゴリの［使用するコンパイラ・パッケージのバージョン］プロパティで［常にインストール済みの最新版］，または該当バージョンを選択してください。

図 2—1 ［バージョン選択］カテゴリ



備考 1. メイン・プロジェクト，およびサブプロジェクトで使用するビルド・ツールが同じ場合，それらのビルド・ツール・ノードをすべて選択し，プロパティを設定することで，ビルド・ツールのバージョンをまとめて変更することができます。

2. 他の実行環境で作成したプロジェクトを開いた場合など，インストールしていないコンパイラ・パッケージを選択している場合は，そのバージョンも表示します。

3. コンパイラ・パッケージによってオプションに変更がある場合は，選択したバージョンにあわせて，ビルド・ツールの各プロパティの表示を切り替えます。

バージョンの変更により非表示になるプロパティについては，プロジェクト・ファイル中に設定値を残しておき，再表示の際に値を再現します。

なお，オプションの変更は，以下の規則に基づいて行い，変更情報は出力 パネルに表示します。

- 旧バージョンから新バージョンへ変更した場合は，オプションの設定の引き継ぎ，および変換（必要な場合のみ）を行います。
- 新バージョンから旧バージョンへ変更した場合は，同一オプションの設定の引き継ぎのみを行います。旧バージョンのみに存在するオプションについては，デフォルト値を設定します。

2. 3　ビルド対象ファイルを設定する

ビルドを実行する前に，ビルド対象となるファイル（C ソース・ファイル，アセンブラ・ソース・ファイルなど）をプロジェクトに追加しておく必要があります。

ここでは，プロジェクトにおけるファイルの設定に関する操作を説明します。

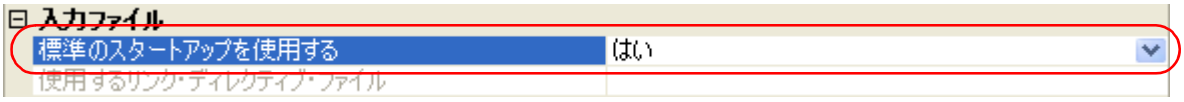
2. 3. 1　スタートアップ・ルーチンを設定する

（1）標準のスタートアップ・ルーチンを使用する場合

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［リンク・オプション］タブを選択します。

標準のスタートアップ・ルーチンを使用するには，［入力ファイル］カテゴリの［標準のスタートアップを使用する］プロパティで［はい］を選択してください。

図 2―2　［標準のスタートアップを使用する］プロパティ



標準のスタートアップ・ルーチンは，［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティの値によって，以下のファイルを使用します。

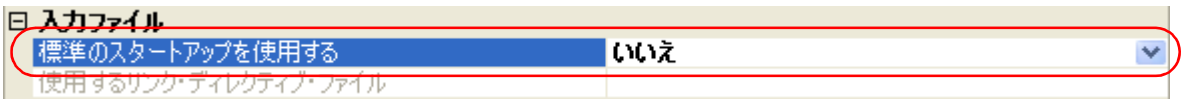
［レジスタ・モードの選択］プロパティの値	標準のスタートアップ・ルーチン
32 レジスタ・モード (なし)	*使用するコンパイラ・パッケージのインストール・フォルダ*¥ lib850 ¥ r32 ¥ crtE.o
26 レジスタ・モード (-reg26)	*使用するコンパイラ・パッケージのインストール・フォルダ*¥ lib850 ¥ r26 ¥ crtE.o
22 レジスタ・モード (-reg22)	*使用するコンパイラ・パッケージのインストール・フォルダ*¥ lib850 ¥ r22 ¥ crtE.o

（2）標準以外のスタートアップ・ルーチンを使用する場合

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［リンク・オプション］タブを選択します。

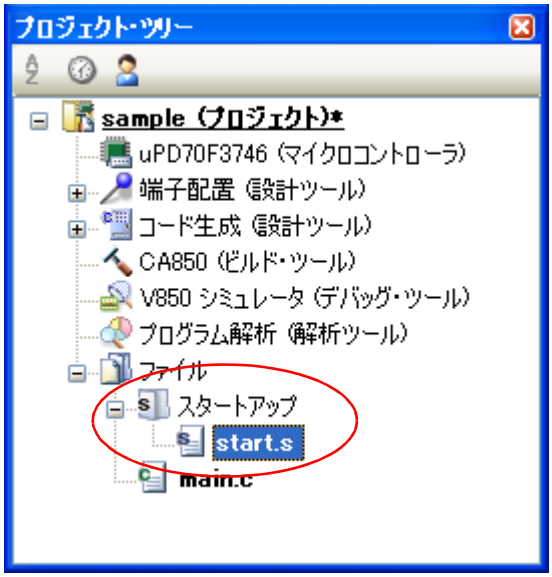
標準以外のスタートアップ・ルーチンを使用するには，［入力ファイル］カテゴリの［標準のスタートアップを使用する］プロパティで［いいえ］を選択してください（デフォルトでは，［はい］が選択されています）。

図 2―3　［標準のスタートアップを使用する］プロパティ



次に，スタートアップ・ルーチンを記述したファイル（スタートアップ・ファイル）をプロジェクト・ツリーのスタートアップ・ノードに追加してください。プロジェクト・ツリーへのファイルの追加方法については，「2. 3. 3　プロジェクトにファイルを追加する」を参照してください。

図 2—4　プロジェクト・ツリー パネル（スタートアップ・ファイル追加後）



注意　プロジェクト・ツリーのスタートアップ・ノード直下に追加しているファイルのうち，ビルド対象ファイルがスタートアップ・ファイルとみなされます。スタートアップ・ノード以下のカテゴリに追加した場合は，スタートアップ・ファイルとはみなされません。
スタートアップ・ファイルをスタートアップ・ノードに追加する際，すでにスタートアップ・ファイルを追加している場合は，追加する最新のスタートアップ・ファイルのみがビルド対象となり，追加済みのスタートアップ・ファイルはビルド対象外となります。
ビルド対象外となっているスタートアップ・ファイルをビルド対象に設定する際，ほかにもスタートアップ・ファイルを追加している場合は，ビルド対象に設定したスタートアップ・ファイルのみがビルド対象となり，それ以外のスタートアップ・ファイルはビルド対象外となります。

備考　スタートアップ・ルーチンを新たに作成する場合には，次のサンプルをコピーして，プロジェクトに追加後，編集してください。

レジスタ・モード	スタート・アップ・ルーチンのサンプル
32 レジスタ・モード	*使用するコンパイラ・パッケージのインストール・フォルダ*¥ lib850 ¥ r32 ¥ crtE.s
26 レジスタ・モード	*使用するコンパイラ・パッケージのインストール・フォルダ*¥ lib850 ¥ r26 ¥ crtE.s
22 レジスタ・モード	*使用するコンパイラ・パッケージのインストール・フォルダ*¥ lib850 ¥ r22 ¥ crtE.s

スタートアップ・ルーチンは，アセンブリ言語で記述する必要があります。
なお，スタートアップ・ルーチンについての詳細は，「CubeSuite+ V850 コーディング編」を参照してください。

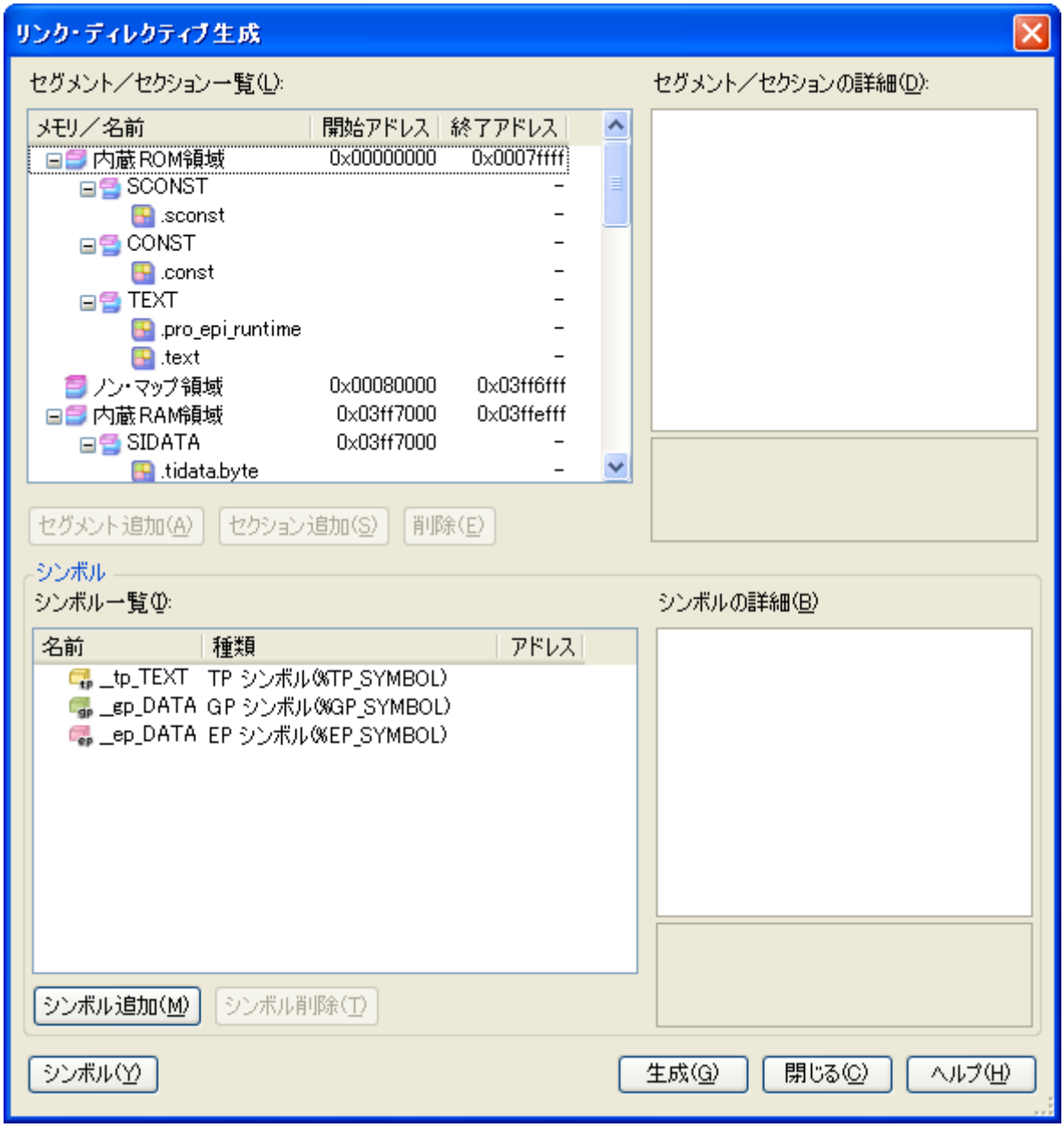
2. 3. 2 リンク・ディレクティブを自動生成する

リンク・ディレクティブ・ファイルはユーザが作成してプロジェクトに追加しますが，CubeSuite+ 上で自動生成することもできます。

備考 リンク・ディレクティブの詳細，およびリンク・ディレクティブ・ファイルの作成方法については，「CubeSuite+ V850 コーディング編」を参照してください。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，コンテキスト・メニューの［リンク・ディレクティブ・ファイルを生成する ...］を選択すると，リンク・ディレクティブ生成 ダイアログがオープンします。

図 2—5 リンク・ディレクティブ生成 ダイアログ



ダイアログ上で，セグメント／セクション，シンボルの編集を行います。

(1) セグメント／セクションの編集

［セグメント／セクション一覧］エリアには，デバイスのメモリ配置情報と，現在設定されているセグメントとセクションの一覧が表示されています。

一覧において，セグメント／セクションを選択すると，［セグメント／セクションの詳細］エリアに該当セグメント／セクションの詳細情報が表示されます。［セグメント／セクションの詳細］エリアにおいて，各項目を編集してください。

備考　予約セクションについては，編集不可の項目（自動で値が設定される項目）があります。
各項目の詳細，および予約セクションの扱いについては，「付録 A　ウインドウ・リファレンス」の「リンク・ディレクティブ生成 ダイアログ」を参照してください。

図 2—6　セグメントの詳細（SCONST を選択した場合）

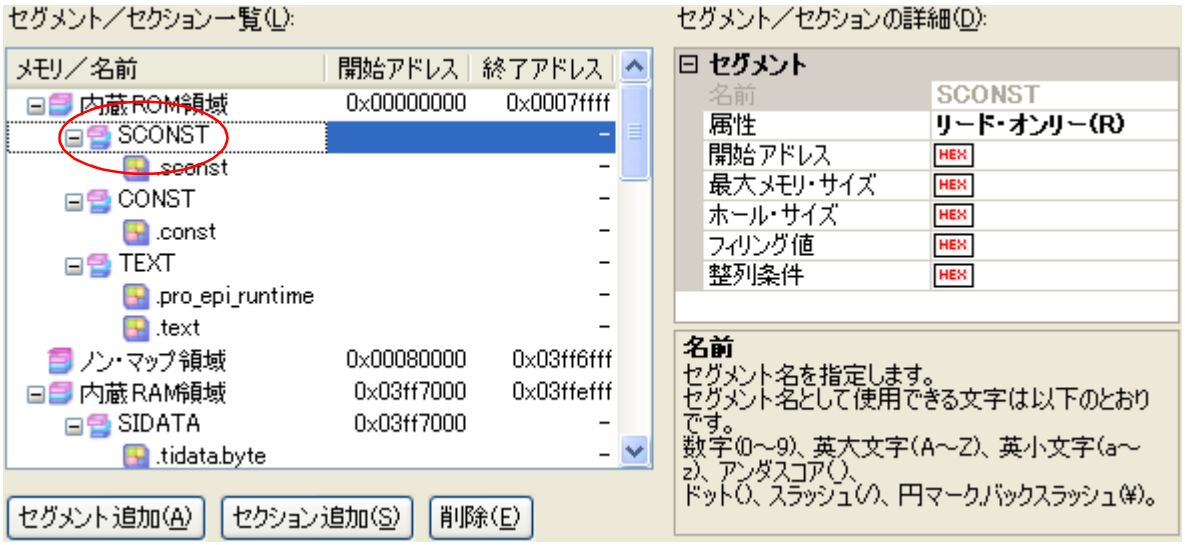
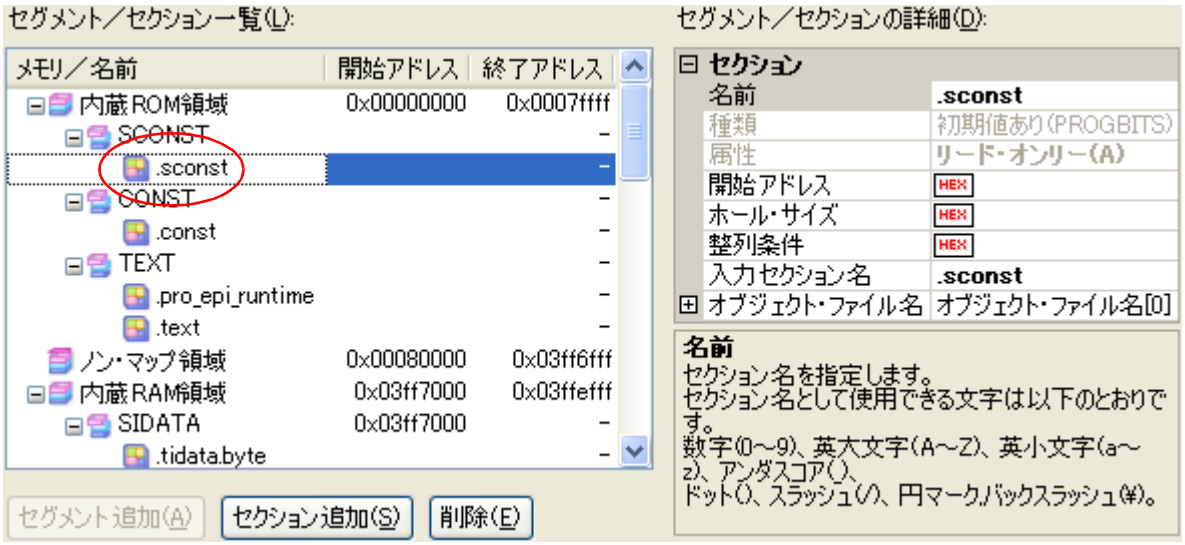


図 2—7　セクションの詳細（.sconst を選択した場合）

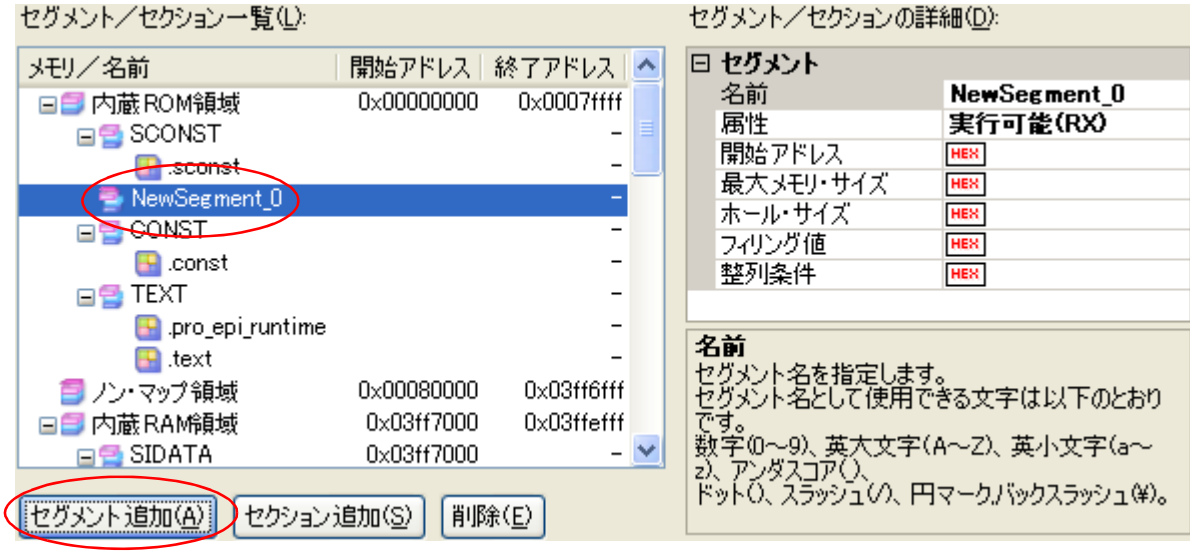


セグメント／セクションは，追加することもできます。

［セグメント追加］ボタンをクリックすると，一覧で選択している行の直下に新しいセグメント“NewSegment_*XXX*”を追加します（*XXX*：0 ～ 255 の 10 進数）。［セグメント／セクションの詳細］エリアにおいて，各項目を編集してください。デフォルトでは，［属性］に［実行可能 (RX)］（内蔵 ROM 領域，またはノン・マップ領域に追加した場合），または［リード／ライト可能 (RW)］（内蔵 RAM 領域に追加した場合）が選択されています。

注意　一覧でセクションの行を選択している場合，［セグメント追加］ボタンは無効となります。

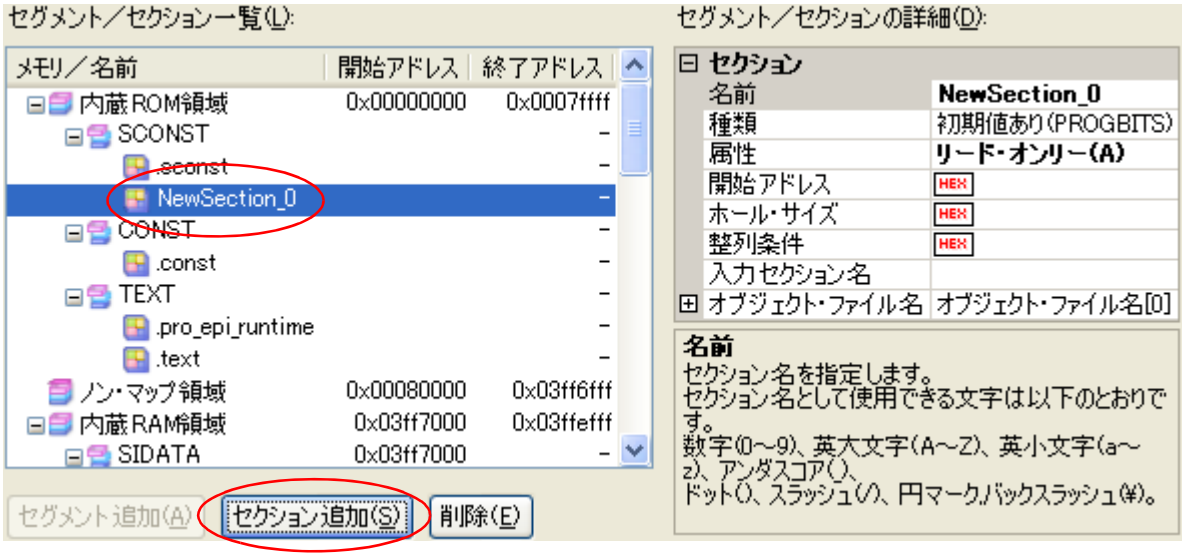
図 2—8　セグメントの追加



［セクション追加］ボタンをクリックすると，一覧で選択している行の直下に新しいセクション“NewSection_*XXX*”を追加します（*XXX*：0 ～ 255 の 10 進数）。［セグメント／セクションの詳細］エリアに

おいて，各項目を編集してください。デフォルトでは，［種類］は［初期値あり (PROGBITS)］が選択され，［属性］は親セグメントの属性を引き継ぎます。

図 2—9 セクションの追加

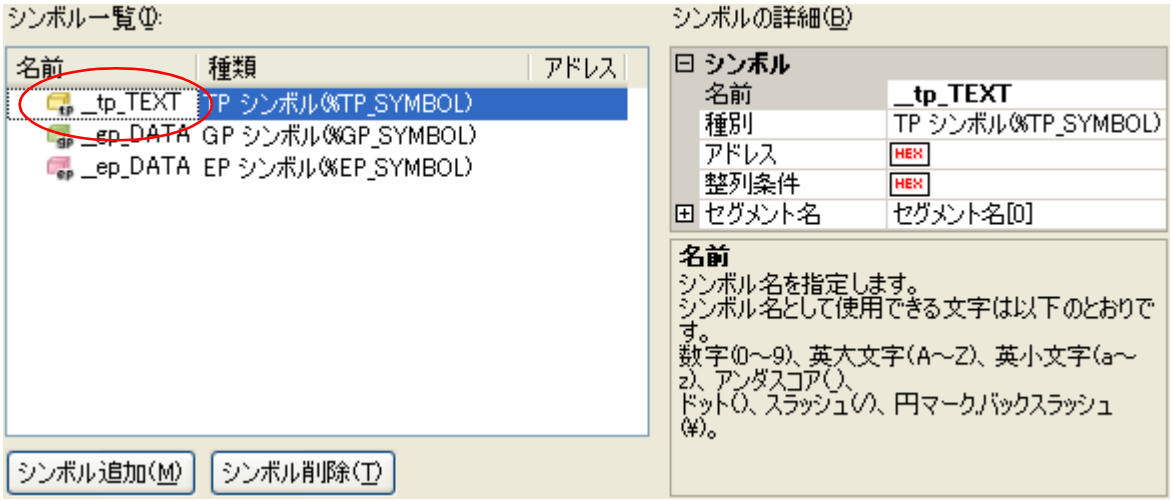


(2) シンボルの編集

［シンボル一覧］エリアには，現在設定されているシンボルの一覧が表示されています。

一覧において，シンボルを選択すると，［シンボルの詳細］エリアに該当シンボルの詳細情報が表示されます。［シンボルの詳細］エリアにおいて，各項目を編集してください。

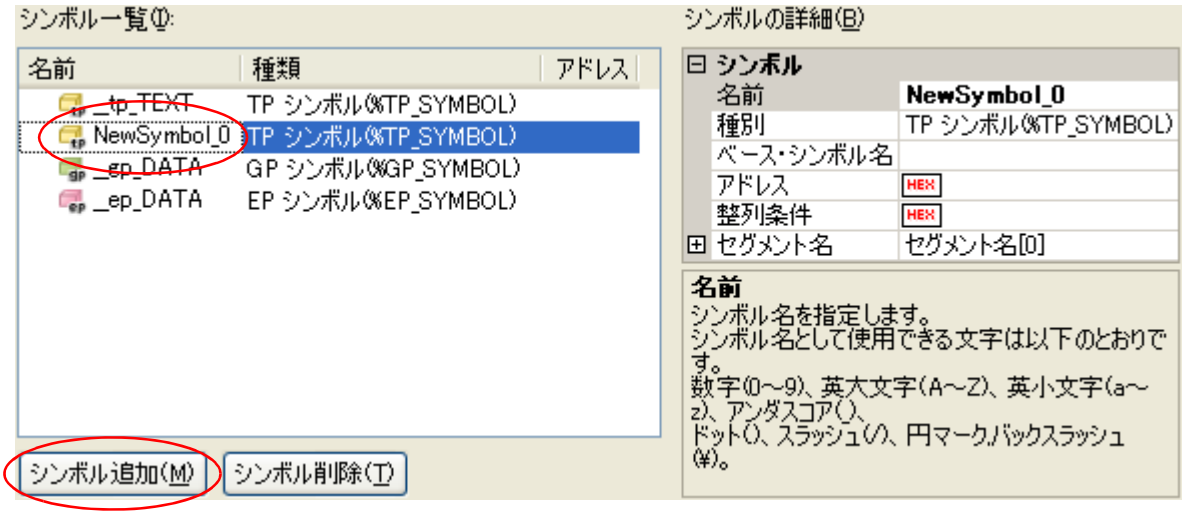
図 2—10 シンボルの詳細（_tp_TEXT を選択した場合）



シンボルは，追加することもできます。

［シンボル追加］ボタンをクリックすると，一覧で選択している行の直下に新しいシンボル“NewSymbol_*XXX*”を追加します（*XXX*：0 ～ 255 の 10 進数）。［シンボルの詳細］エリアにおいて，各項目を編集してください。デフォルトでは，［種別］に［TP シンボル (%TP_SYMBOL)］が選択されています。

図 2—11 シンボルの追加

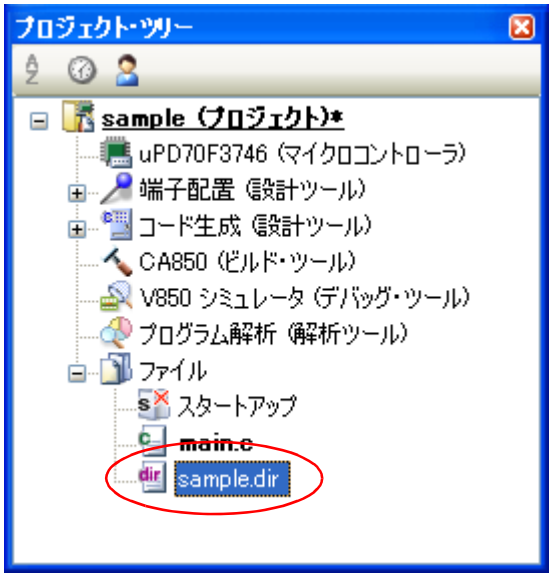


セグメント／セクション，シンボルの編集後，［生成］ボタンをクリックしてください。

指定したメモリ，セグメント／セクション，シンボルの配置情報を元に，リンク・ディレクティブ・ファイル（ファイル名 : *プロジェクト名*.dir）を生成し，プロジェクトに登録します。

リンク・ディレクティブ・ファイルの生成先は，プロジェクト・フォルダとなります。生成したリンク・ディレクティブ・ファイルは，プロジェクト・ツリーのファイル・ノードにも表示されます。

図 2—12 プロジェクト・ツリー パネル（リンク・ディレクティブ・ファイル生成後）



注意 生成したリンク・ディレクティブ・ファイルはビルド対象となります。すでにリンク・ディレクティブ・ファイルをプロジェクトに登録していた場合，登録済みのリンク・ディレクティブ・ファイルはビルド対象外となります。

2. 3. 3 プロジェクトにファイルを追加する

プロジェクトにファイルを追加するには，次の方法があります。

- 既存のファイルを追加する場合
- 空のファイルを作成して追加する場合

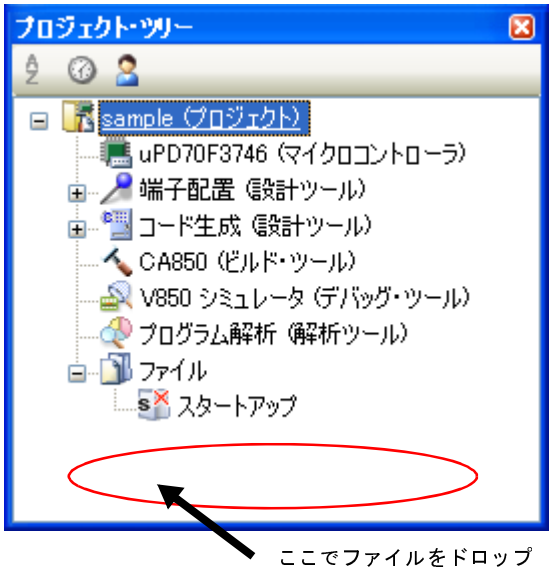
(1) 既存のファイルを追加する場合

(a) ファイル単位で追加する

エクスプローラなどからファイルをドラッグし，プロジェクト・ツリー下部の空白部分にドロップしてください。

ファイルの追加先はファイル・ノード以下となります。

図 2—13 プロジェクト・ツリー パネル（ファイルのドロップ位置）



注意 標準以外のスタートアップ・ルーチンを追加する場合は，スタートアップ・ノード上でファイルをドロップしてください。標準以外のスタートアップ・ルーチンの使用についての詳細は，「2. 3. 1 スタートアップ・ルーチンを設定する」を参照してください。

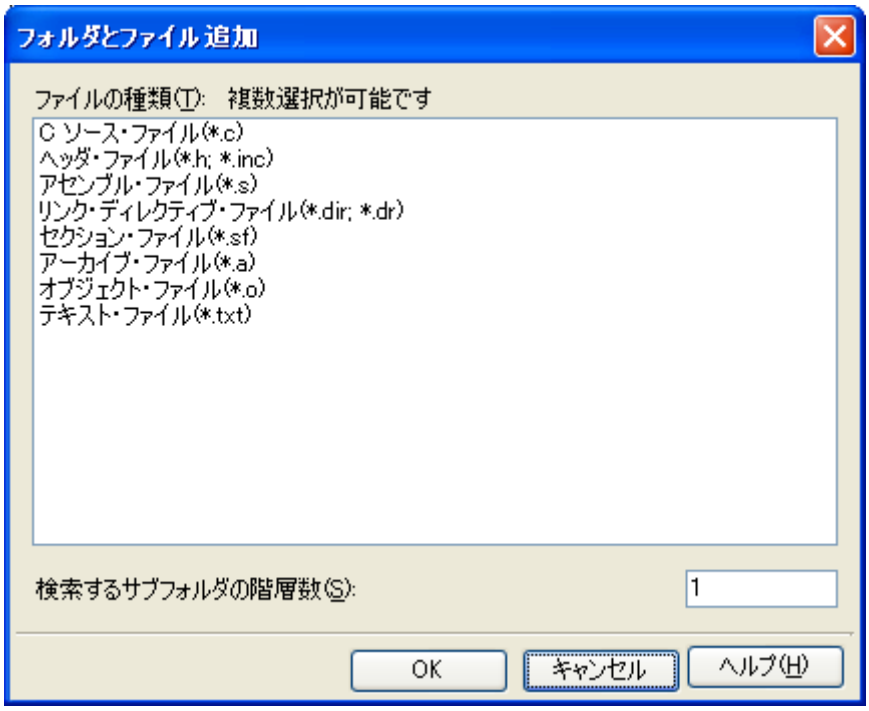
(b) フォルダ単位で追加する

エクスプローラなどからフォルダをドラッグし，プロジェクト・ツリー下部の空白部分にドロップすると，フォルダとファイル追加 ダイアログがオープンします。

備考 複数のフォルダを同時にドラッグし，プロジェクト・ツリーにドロップすることにより，複数のフォルダを同時にプロジェクトに追加することもできます。

注意 フォルダ名が 200 文字を越えるフォルダをドロップした場合，201 文字目以降は切り捨てたカテゴリ名で，プロジェクト・ツリーに追加します。

図 2—14 フォルダとファイル追加 ダイアログ



ダイアログ上で，プロジェクトに追加するファイルの種類を選択し，プロジェクトに追加するサブフォルダの階層数を指定したのち，［OK］ボタンをクリックしてください。

備考 ファイルの種類は，［Ctrl］キー＋左クリック，または［Shift］キー＋左クリックにより，複数選択することができます。
何も選択しない場合は，すべての種類を選択したものとみなします。

フォルダの追加先はファイル・ノード以下となります。
なお，フォルダはプロジェクト・ツリーではカテゴリとなります。

備考 ユーザが作成したカテゴリ・ノードが存在する場合，カテゴリ・ノード上でファイルをドロップすると，カテゴリ・ノード以下に追加することができます（カテゴリ・ノードについては，「2. 3. 6 ファイルをカテゴリに分類する」を参照してください）。

(2) 空のファイルを作成して追加する場合

プロジェクト・ツリー でプロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノードのいずれかを選択し，コンテキスト・メニューの［追加］→［新しいファイルを追加 ...］を選択すると，ファイル追加 ダイアログがオープンします。

図 2—15 ファイル追加 ダイアログ

ダイアログ上で，新しく作成するファイルを指定し，［OK］ボタンをクリックしてください。

ファイルの追加先はファイル・ノード以下となります。

ファイル追加後のプロジェクト・ツリーは，以下のようになります。

図 2—16 プロジェクト・ツリー パネル（ファイル main.c 追加後）

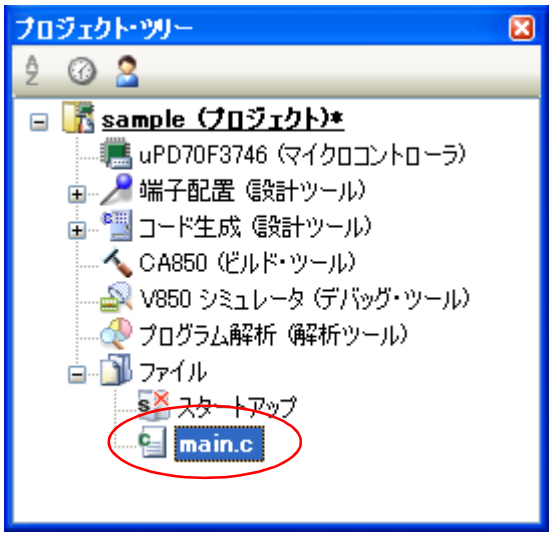
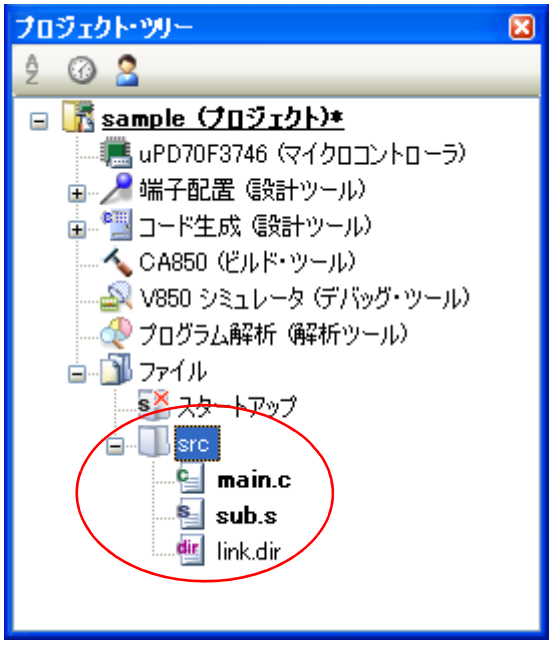


図 2—17　プロジェクト・ツリー パネル（フォルダ src 追加後）



備考　ファイル・ノード以下におけるファイルの追加位置は，現在のファイルの表示順の設定に依存します。
ファイルの表示順の変更方法については，「2. 3. 7　ファイルの表示順を変更する」を参照してください。

注意 1.　パスが異なれば，同名のソース・ファイルを追加することができます。ただし，それらの出力ファイル名の設定がデフォルトのままの場合，出力ファイル名が同名になるため，ビルドを正しく実行することができません（例えば，D:￥sample1￥func.c，D:￥sample2￥func.c を追加した場合，これらの出力ファイル名は，デフォルトではどちらも func.o となります）。
この問題を回避するために，ソース・ファイルの個別ビルド・オプションで，出力ファイル名をそれぞれ異なる名前に設定してください。
C ソース・ファイルの出力ファイル名の変更は，［個別コンパイル・オプション］タブの［出力ファイル］カテゴリの［オブジェクト・ファイル名］プロパティで行います。アセンブラ・ソース・ファイルの出力ファイル名の変更は，［個別アセンブル・オプション］タブの［出力ファイル］カテゴリの［オブジェクト・ファイル名］プロパティで行います。個別ビルド・オプションの設定方法については，「2. 15. 2　ファイル単位でコンパイル／アセンブル・オプションを設定する」を参照してください。

2. 同名のソース・ファイルを追加した場合，デバッグ時に対象のソースをオープンすることができません。
3. 拡張子が“dr”，“dir”のファイルをプロジェクトに追加した場合，そのファイルはリンク・ディレクティブ・ファイルとみなされます。スタートアップ・ノード以下に追加した場合もリンク・ディレクティブ・ファイルとみなされます。
リンク・ディレクティブ・ファイルをプロジェクトに追加する際，すでにリンク・ディレクティブ・ファイルを追加している場合は，追加する最新のリンク・ディレクティブ・ファイルのみがビルド対象となり，追加済みのリンク・ディレクティブ・ファイルはビルド対象外となります。
ビルド対象外となっているリンク・ディレクティブ・ファイルをビルド対象に設定する際，ほかにもリンク・ディレクティブ・ファイルを追加している場合は，ビルド対象に設定したリンク・ディレクティ

ブ・ファイルのみがビルド対象となり，それ以外のリンク・ディレクティブ・ファイルはビルド対象外となります。

4. **プロジェクトに追加可能なファイル数は，メイン・プロジェクト，およびサブプロジェクトごとに 5000 個までです。**

新しいファイルを追加した場合，ファイル追加 ダイアログで指定した場所に，空のファイルが作成されます。

プロジェクト・ツリー でファイル名をダブルクリックすることにより，エディタ パネルをオープンし，ファイルを編集することができます。

以下に，エディタ パネルでオープン可能なファイルを示します。

- C ソース・ファイル（.c）
- アセンブラ・ソース・ファイル（.s）
- ヘッダ・ファイル（.h，.inc）
- リンク・ディレクティブ・ファイル（.dr，.dir）
- セクション・ファイル（.sf）
- マップ・ファイル（.map）
- ヘキサ・ファイル（.hex）
- テキスト・ファイル（.txt）

備考 1. 以下のいずれかの方法により，上記以外のファイルもエディタ パネルでオープンすることができます。

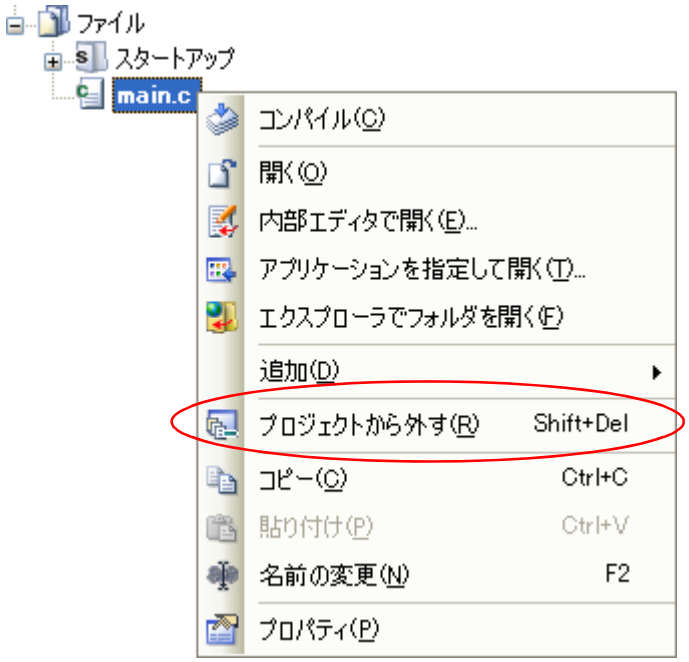
- ファイルをドラッグし，エディタ パネルにドロップする。
- ファイルを選択し，コンテキスト・メニューの［内部エディタで開く ...］を選択する。

2. オプション ダイアログで，外部エディタを使用する設定になっている場合は，設定している外部エディタでオープンします。それ以外のファイルは，ホスト OS で関連付けられているアプリケーションで起動します。

2. 3. 4 プロジェクトからファイルを外す

プロジェクトに追加しているファイルをプロジェクトから外すには，プロジェクト・ツリー でプロジェクトから外すファイルを選択し，コンテキスト・メニューの［プロジェクトから外す］を選択してください。

図 2—18 ［プロジェクトから外す］項目



2. 3. 5 ファイルをビルド対象から外す

プロジェクトに追加しているファイルのうち，特定のファイルをビルド対象から外すことができます。

プロジェクト・ツリー でビルド対象から外すファイルを選択したのち，プロパティ パネルの［ビルド設定］タブを選択します。［ビルド］カテゴリの［ビルドの対象とする］プロパティで［いいえ］を選択してください。

図 2—19 ［ビルドの対象とする］プロパティ

備考 この機能を適用できるファイルは，C ソース・ファイル，アセンブラ・ソース・ファイル，リンク・ディレクティブ・ファイル，セクション・ファイル，オブジェクト・ファイル，アーカイブ・ファイルです。

2. 3. 6 ファイルをカテゴリに分類する

プロジェクトに追加しているファイルをプロジェクト・ツリー上で見やすくしたり，機能ごとに管理しやすくするために，ファイル・ノード以下にカテゴリ・ノードを作成して，ファイルを分類することができます。

カテゴリ・ノードを作成するには，プロジェクト・ツリー でプロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノードのいずれかを選択し，コンテキスト・メニューの［追加］→［新しいカテゴリを追加］を選択してください。

図 2—20 ［新しいカテゴリを追加］項目（ファイル・ノードの場合）

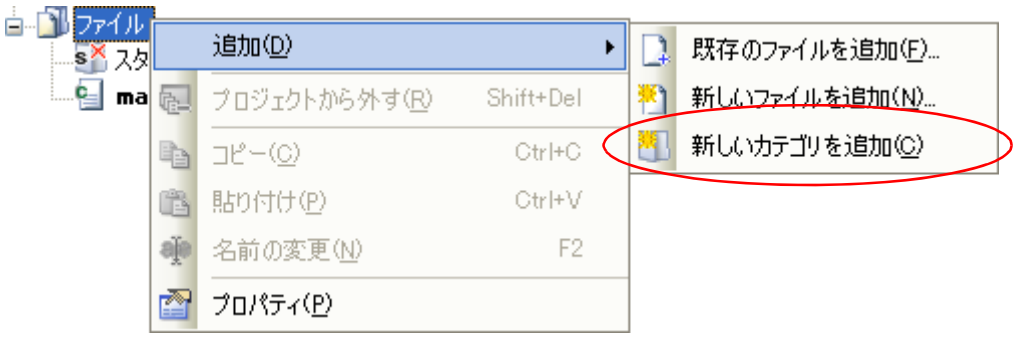
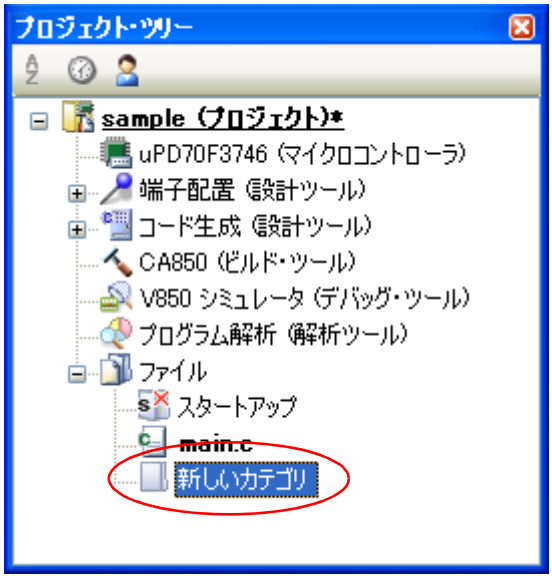


図 2—21 プロジェクト・ツリー パネル（カテゴリ・ノード追加後）



備考 1. カテゴリ名は，デフォルトで“新しいカテゴリ”となります。
カテゴリ名の変更は，カテゴリ・ノードのコンテキスト・メニューの［名前の変更］から行うことができます。

2. すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。

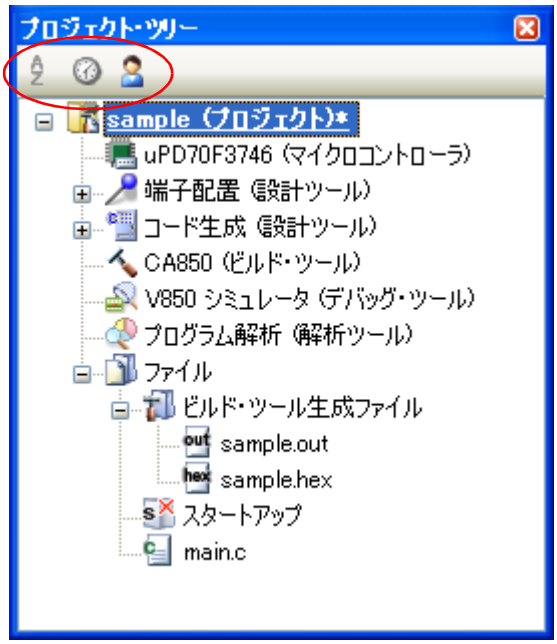
3. カテゴリのネスト数の上限は 20 です。

作成したカテゴリ・ノードにファイルを分類するには，ファイルのドラッグ・アンド・ドロップにより行うことができます。

2. 3. 7 ファイルの表示順を変更する

プロジェクト・ツリー上のボタンで，ファイル，およびカテゴリ・ノードの表示順を変更することができます。

図 2—22 ツールバー（プロジェクト・ツリー パネル）



プロジェクト・ツリー パネルのツールバーで，以下のいずれかのボタンを選択してください。

ボタン	説明
	カテゴリ・ノード，およびファイルを名前順でソートします。  : 昇順  : 降順  : 昇順
	カテゴリ・ノード，およびファイルをタイムスタンプ順でソートします。  : 降順  : 昇順  : 降順
	カテゴリ・ノードとファイルをユーザが指定した順で表示します（デフォルト）。 カテゴリ・ノード，およびファイルをドラッグ・アンド・ドロップすることにより，表示順を任意に変更することができます。

2. 3. 8 ファイルの依存関係を更新する

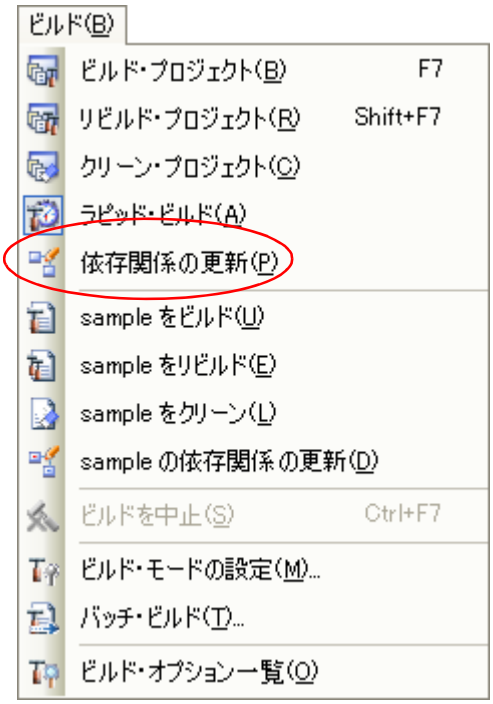
コンパイル・オプションの設定，アセンブル・オプションの設定で，ファイルの依存関係に影響する変更（インクルード・ファイルのパスの変更，C ソース・ファイル，およびアセンブラ・ソース・ファイル中にヘッダ・ファイルのインクルード文を追加など）を行った場合は，該当ファイルの依存関係を更新する必要があります。

ファイルの依存関係の更新は，プロジェクト全体（メイン・プロジェクト，およびサブプロジェクト），またはアクティブ・プロジェクトに対して行います。

(1) プロジェクト全体の場合

［ビルド］メニュー→［依存関係の更新］を選択してください。

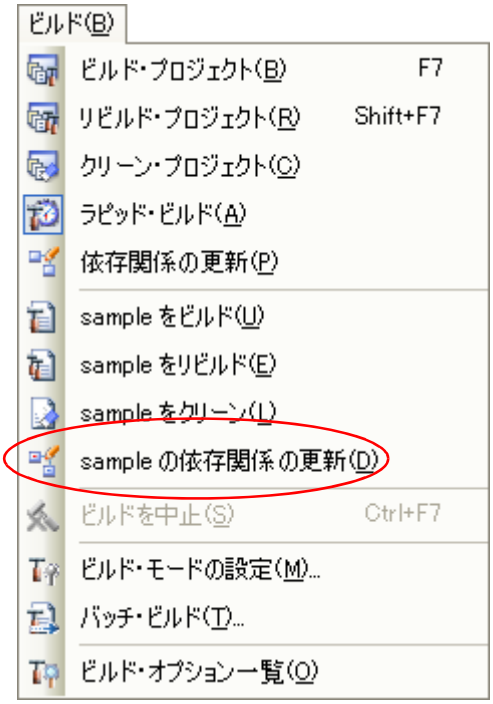
図 2—23 ［依存関係の更新］項目



(2) アクティブ・プロジェクトの場合

［ビルド］メニュー→［*アクティブ・プロジェクト*の依存関係の更新］を選択してください。

図 2—24 ［アクティブ・プロジェクトの依存関係の更新］項目



備考 ファイルの依存関係を更新する際，エディタ パネルで編集中のファイルがある場合は，該当ファイルを一括して保存します。

注意 1. CubeSuite+ は，インクルード・ファイルの依存関係のチェックにおいて，#if などの条件文やコメントを無視します。そのため，ビルドに不要なインクルード・ファイルを，必要なファイルであると誤認します（以下の例において，header1.h，header5.h は，ビルドに必要であると判断します）

```
#if         0
#include    "header1.h"     /* 依存関係ありと判断する */
#else                       /* ! zero */
#include    "header2.h"     /* 依存関係あり */
#endif

#define     AAA
#ifdef      AAA
#include    "header3.h"     /* 依存関係あり */
#else
#include    "header4.h"     /* 依存関係あり */
#endif

/*
#include    "header5.h"     /* 依存関係ありと判断する */
*/
```

2. **CubeSuite+ は，インクルード・ファイルの依存関係のチェックにおいて，コメント文のあとに記述したインクルード文を無視します。そのため，ビルドに必要なインクルード・ファイルを，不要なファイルであると誤認します（以下の例において，header6.h，header7.h は，ビルドに不要であると判断します）。**

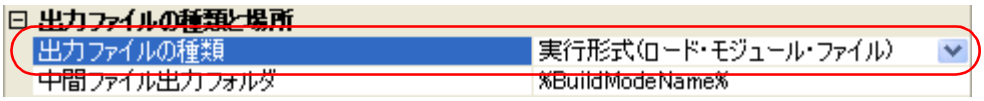
```
/* comment */   #include    "header6.h"     /* 依存関係なしと判断する */


/*
comment
*/  #include    "header7.h"                 /* 依存関係なしと判断する */
```

2. 4　出力ファイルの種類を設定する

ビルドの生成物として出力するファイルの種類を設定します。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。［出力ファイルの種類と場所］カテゴリの［出力ファイルの種類］プロパティにおいて，ファイルの種類を選択してください。

図 2—25　［出力ファイルの種類］プロパティ



(1)［実行形式（ROM 化用モジュール・ファイル)］を選択した場合

ROM 化用モジュール・ファイルを生成します。

［ROM 化プロセス・オプション］タブの［出力ファイル］カテゴリで設定しているファイルがデバッグ対象となります。

(2)［実行形式（ロード・モジュール・ファイル)］を選択した場合（デフォルト）

ロード・モジュール・ファイルを生成します。

［リンク・オプション］タブの［出力ファイル］カテゴリで設定しているファイルがデバッグ対象となります。

(3)［実行形式（ヘキサ・ファイル)］を選択した場合

ヘキサ・ファイルも含めて生成します。

［ヘキサ・コンバート・オプション］タブの［出力ファイル］カテゴリで設定しているファイルがデバッグ対象となります。

注意　ライブラリ用のプロジェクトの場合，本プロパティは常に［ライブラリ形式］となり，変更することはできません。

2. 4. 1　出力ファイル名を変更する

ビルド・ツールが出力する ROM 化用モジュール・ファイル，ロード・モジュール・ファイル，ヘキサ・ファイル，アーカイブ・ファイルのファイル名は，デフォルトで次の名前が設定されています。

“%ProjectName%”は，埋め込みマクロで，プロジェクト名に置換します。

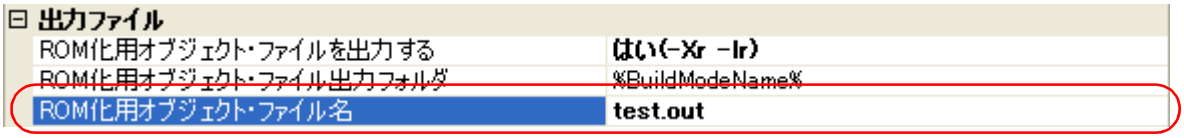
ROM 化用モジュール・ファイル名	：romp.out
ロード・モジュール・ファイル名	：%ProjectName%.out
ヘキサ・ファイル名	：%ProjectName%.hex
アーカイブ・ファイル名	：lib%ProjectName%.a

これらのファイル名の変更方法を，以下に示します。

(1) ROM 化用モジュール・ファイル名を変更する場合

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ROM 化プロセス・オプション］タブを選択します。［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイル名］プロパティにおいて，変更するファイル名を入力してください。

図 2—26　［ROM 化用オブジェクト・ファイル名］プロパティ（ROM 化用モジュール・ファイルの場合）

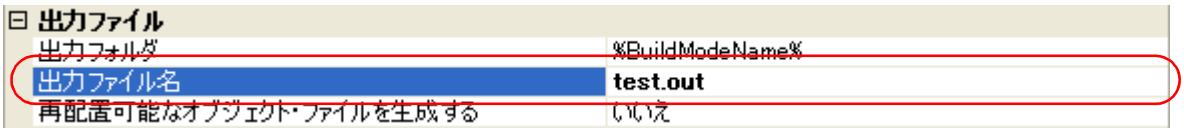


備考　［共通オプション］タブの［よく使うオプション（ROM 化プロセッサ）］カテゴリの［ROM 化用オブジェクト・ファイル名］プロパティでも，同様に変更することができます。

(2) ロード・モジュール・ファイル名を変更する場合

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［リンク・オプション］タブを選択します。［出力ファイル］カテゴリの［出力ファイル名］プロパティにおいて，変更するファイル名を入力してください。

図 2—27　［出力ファイル名］プロパティ（ロード・モジュール・ファイルの場合）

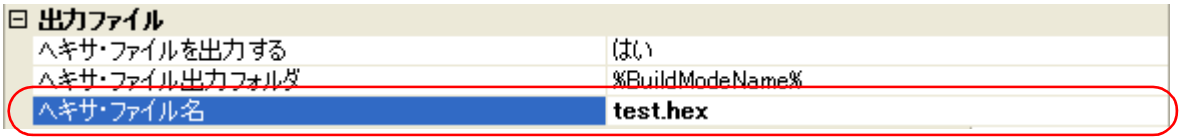


備考　［共通オプション］タブの［よく使うオプション（リンカ）］カテゴリの［出力ファイル名］プロパティでも，同様に変更することができます。

(3) ヘキサ・ファイル名を変更する場合

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ヘキサ・コンバート・オプション］タブを選択します。［出力ファイル］カテゴリの［ヘキサ・ファイル名］プロパティにおいて，変更するファイル名を入力してください。

図 2—28　［ヘキサ・ファイル名］プロパティ

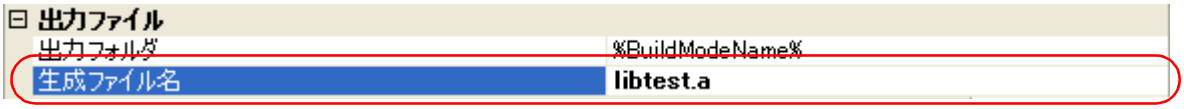


備考　［共通オプション］タブの［よく使うオプション（ヘキサ・コンバータ）］カテゴリの［ヘキサ・ファイル名］プロパティでも，同様に変更することができます。

(4) アーカイブ・ファイル名を変更する場合

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アーカイブ・オプション］タブを選択します。［出力ファイル］カテゴリの［生成ファイル名］プロパティにおいて，変更するファイル名を入力してください。

図 2—29 ［生成ファイル名］プロパティ（アーカイブ・ファイルの場合）



2. 4. 2 アセンブル・リストを出力する

アセンブル結果はアセンブル・リスト・ファイルに出力されます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アセンブル・オプション］タブを選択します。アセンブル・リストを出力するには，［アセンブル・リスト］カテゴリの［アセンブル・リスト・ファイルを出力する］プロパティで［はい (-a -l)］を選択してください。

図 2—30 ［アセンブル・リスト・ファイルを出力する］プロパティ

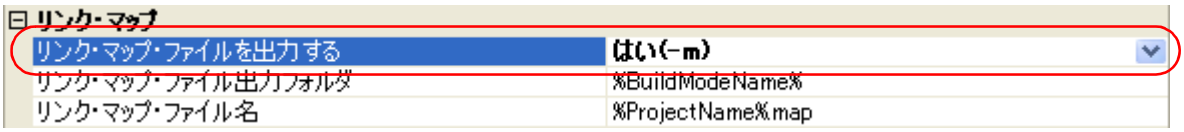
備考 アセンブル・リストについては，「3. 1 アセンブラ」を参照してください。

2. 4. 3 マップ情報を出力する

マップ情報（セクションの配置に関する情報）はリンク・マップ・ファイルに出力されます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［リンク・オプション］タブを選択します。リンク・マップ・ファイルを出力するには，［リンク・マップ］カテゴリの［リンク・マップ・ファイルを出力する］プロパティで［はい (-m)］を選択してください。

図 2—31 ［リンク・マップ・ファイルを出力する］プロパティ



リンク・マップ・ファイルを出力する場合，出力フォルダ，および出力ファイル名を設定することができます。

(1) 出力フォルダの設定

［リンク・マップ・ファイル出力フォルダ］プロパティにおいて，テキスト・ボックスへの直接入力，または［...］ボタンにより行います。テキスト・ボックスには 247 文字まで指定可能です。デフォルトでは，“%BuildModeName%”が設定されています。“%BuildModeName%”は，埋め込みマクロで，ビルド・モード名に置換します。

(2) 出力ファイル名の設定

［リンク・マップ・ファイル名］プロパティにおいて，テキスト・ボックスへの直接入力により行います。テキスト・ボックスには 259 文字まで指定可能です。デフォルトでは，“%ProjectName%.map”が設定されています。“%ProjectName%”は，埋め込みマクロで，プロジェクト名に置換します。

備考 マップ情報については，「3. 2 リ ン カ」を参照してください。

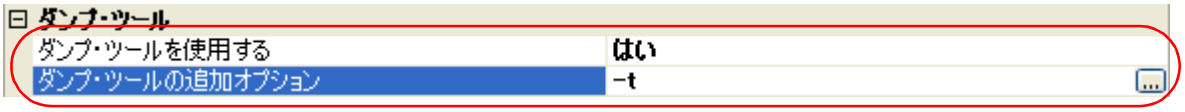
2. 4. 4 シンボル情報を出力する

入力モジュール内で定義されているシンボル情報を出力するには，ダンプ・ツールの -t オプションを使用します。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ダンプ・オプション］タブを選択します。

-t オプションの設定は，［ダンプ・ツール］カテゴリで行います。［ダンプ・ツールを使用する］プロパティで［はい］を選択すると，［ダンプ・ツールの追加オプション］プロパティが表示されます。

図 2—32 ［ダンプ・ツールを使用する］，および［ダンプ・ツールの追加オプション］プロパティ



［ダンプ・ツールの追加オプション］プロパティにおいて，“-t”を指定してください。

備考 1. 出力するシンボル情報については，「(8) シンボル・テーブル」を参照してください。

2. ［ダンプ・ツールの追加オプション］プロパティにおいて，“-t *num*”と指定すると，*num* 番目のシンボル・テーブル・エントリから表示します。また，“-v”も指定すると，セクション属性などの値を数字ではなく，文字列で表示することができます。

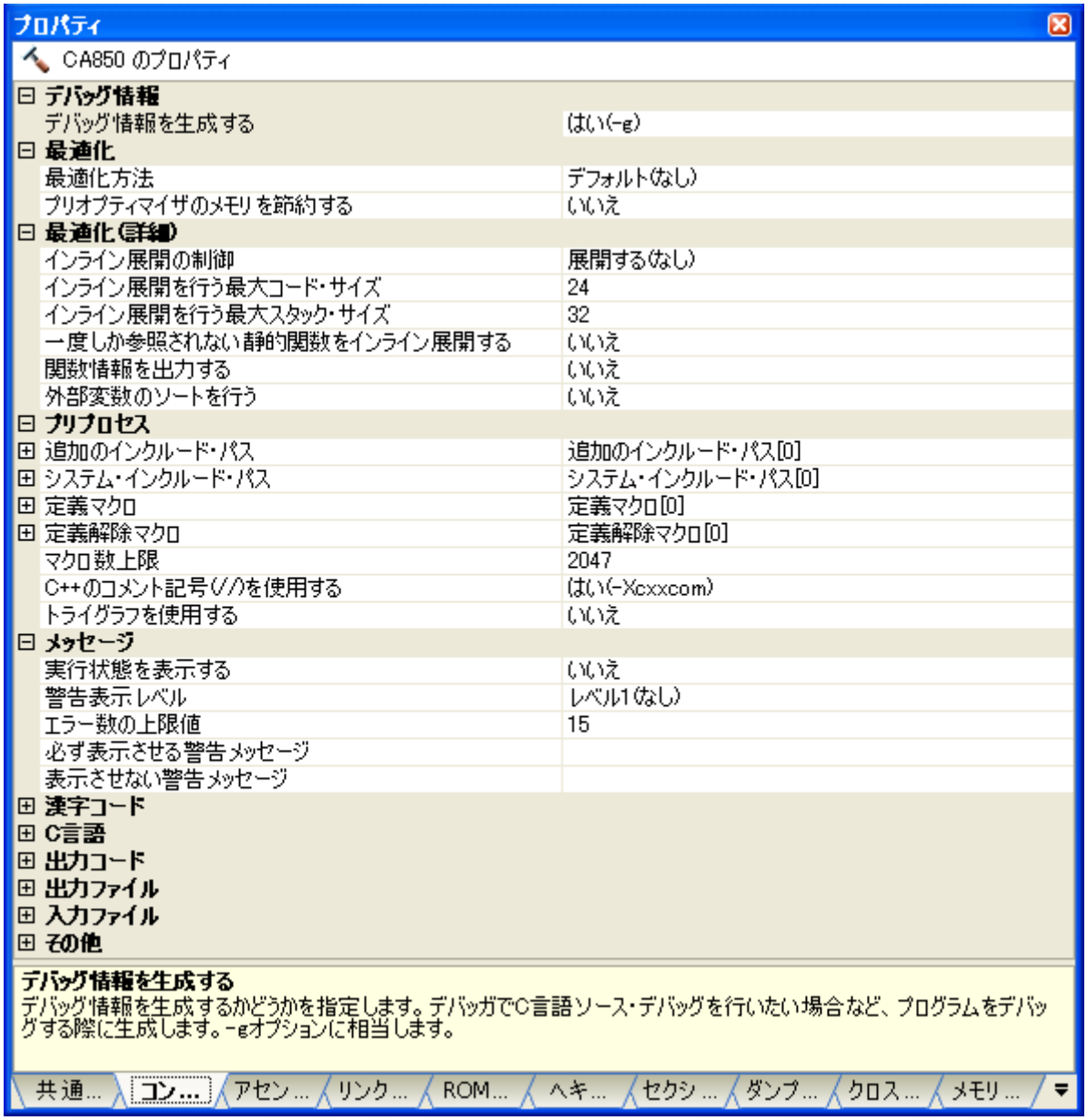
各オプションについての詳細は，「B. 8. 2 オプション」を参照してください。

2.5 コンパイル・オプションを設定する

コンパイラに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するコンパイル・オプションを設定することができます。

図 2—33 プロパティ パネル：［コンパイル・オプション］タブ



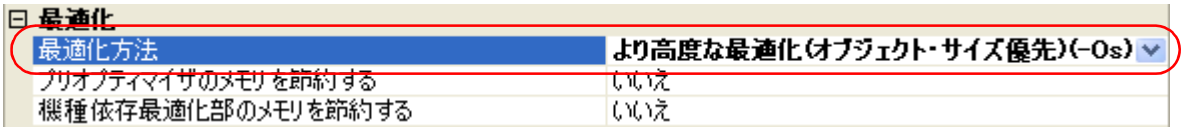
備考 よく使うオプションについては，［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリにまとめられています。

2. 5. 1 コード・サイズを優先した最適化を行う

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択します。

コード・サイズを優先した最適化を行うには，［最適化］カテゴリの［最適化方法］プロパティで［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］を選択してください（デフォルトでは，［デフォルト (なし)］が選択されています）。

図 2—34 ［最適化方法］プロパティ（コード・サイズ優先の場合）



備考 1. ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［最適化方法］プロパティでも，同様に設定することができます。

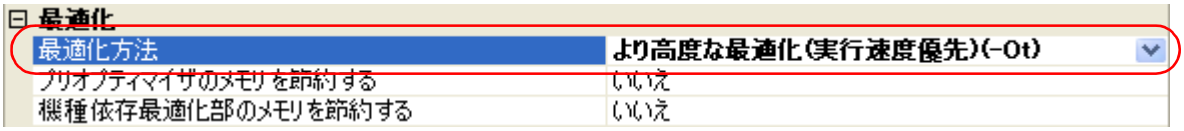
2. 最適化についての詳細は，「(3) 効率的な最適化の仕方」を参照してください。

2. 5. 2 実行速度を優先した最適化を行う

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択します。

実行速度を優先した最適化を行うには，［最適化］カテゴリの［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択してください（デフォルトでは，［デフォルト (なし)］が選択されています）。

図 2—35 ［最適化方法］プロパティ（実行速度優先の場合）



備考 1. ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［最適化方法］プロパティでも，同様に設定することができます。

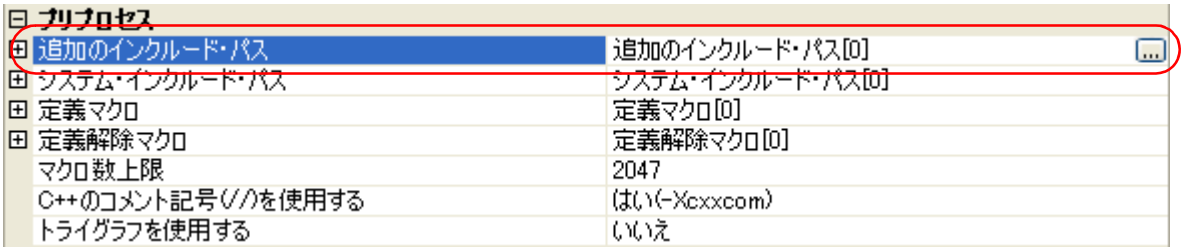
2. 最適化についての詳細は，「(3) 効率的な最適化の仕方」を参照してください。

2. 5. 3 インクルード・パスを追加する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択します。

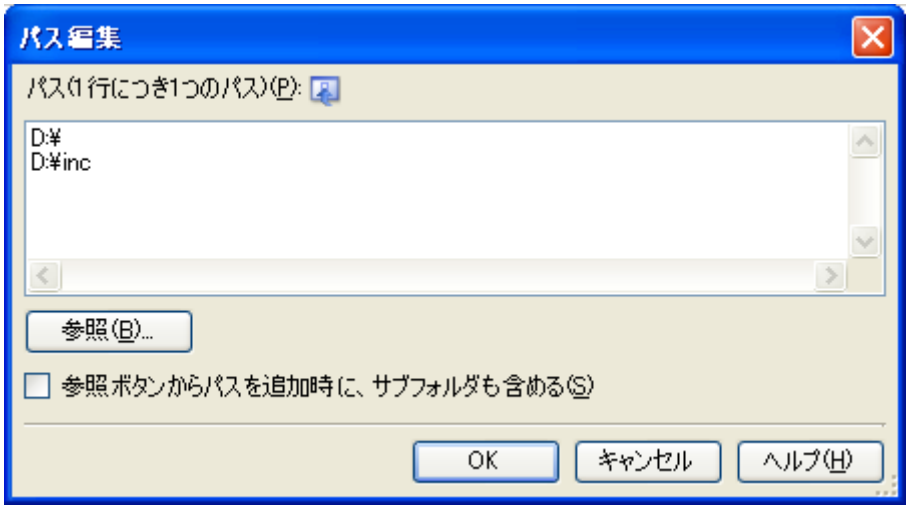
インクルード・パスの設定は，［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで行います。

図 2—36 ［追加のインクルード・パス］プロパティ



［...］ボタンをクリックすると，パス編集 ダイアログがオープンします。

図 2—37 パス編集 ダイアログ

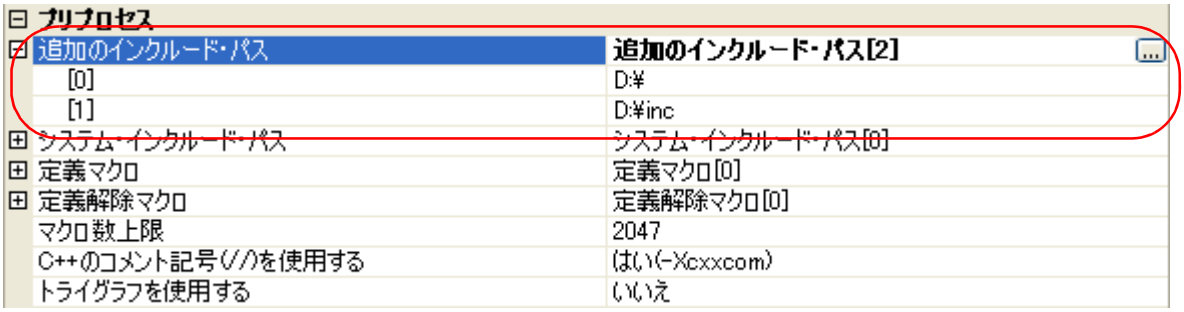


［パス (1 行につき 1 つのパス)］にインクルード・パスを 1 行に 1 つずつ入力します。1 行に 259 文字まで，64 行まで指定可能です。

備考 インクルード・パスは，エクスプローラなどからフォルダのドラッグ・アンド・ドロップ，または［参照 ...］ボタンから指定することも可能です。［参照ボタンからパスを追加時に、サブフォルダも含める］をチェックしたのち，［参照 ...］ボタンからパスの指定を行うと，指定したパスとそのサブフォルダ 5 階層分までのパスが［パス (1 行につき 1 つのパス)］に追加されます。

［OK］ボタンをクリックすると，入力したインクルード・パスがサブプロパティとして表示されます。

図 2—38 ［追加のインクルード・パス］プロパティ（インクルード・パス追加後）



インクルード・パスの変更は，［...］ボタン，またはサブプロパティのテキスト・ボックスへの直接入力により行うことができます。

また，プロジェクト・ツリーにインクルード・ファイルを追加すると，そのインクルード・パスをサブプロパティの最初に自動で追加します。

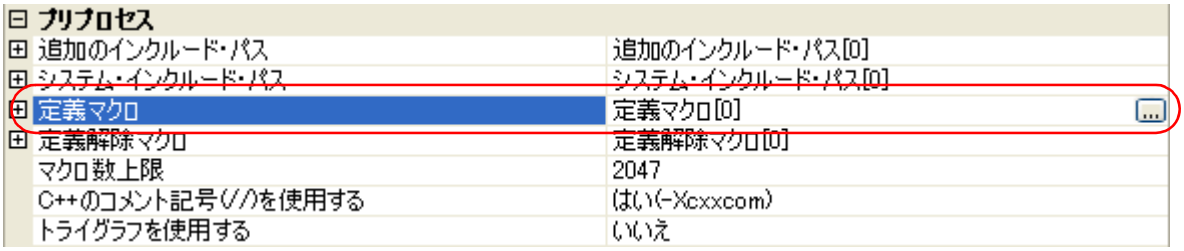
備考 ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［追加のインクルード・パス］プロパティでも，同様に設定することができます。

2. 5. 4 定義マクロを設定する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択します。

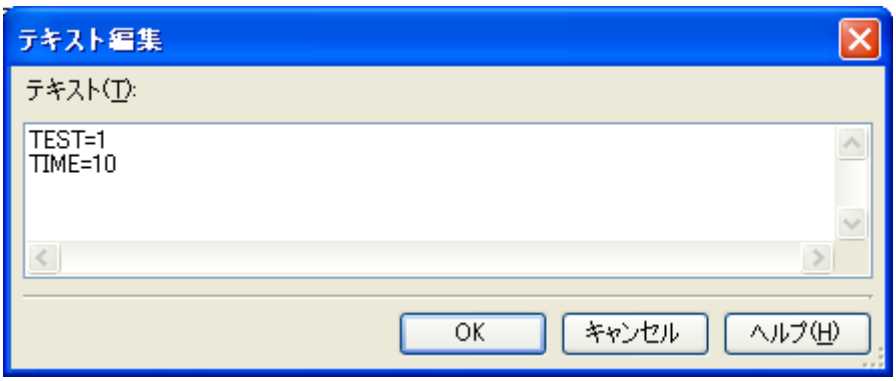
定義マクロの設定は，［プリプロセス］カテゴリの［定義マクロ］プロパティで行います。

図 2—39 ［定義マクロ］プロパティ



［...］ボタンをクリックすると，テキスト編集 ダイアログがオープンします。

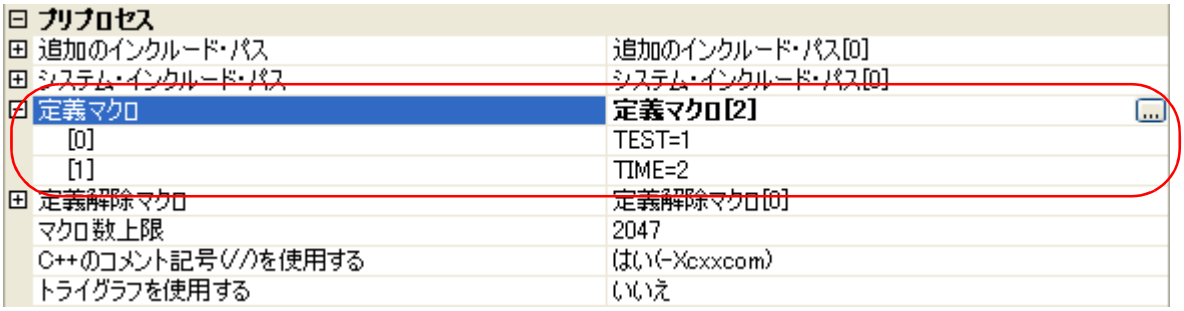
図 2—40 テキスト編集 ダイアログ



［テキスト］に定義マクロを「*マクロ名* = *定義値*」の形式で 1 行に 1 つずつ入力します。1 行に 256 文字まで，30 行まで指定可能です。「= *定義値*」の部分は省略可能で，省略した場合，定義値を 1 とします。

［OK］ボタンをクリックすると，入力した定義マクロがサブプロパティとして表示されます。

図 2—41 ［定義マクロ］プロパティ（定義マクロ設定後）



定義マクロの変更は，［...］ボタン，またはサブプロパティのテキスト・ボックスへの直接入力により行うことができます。

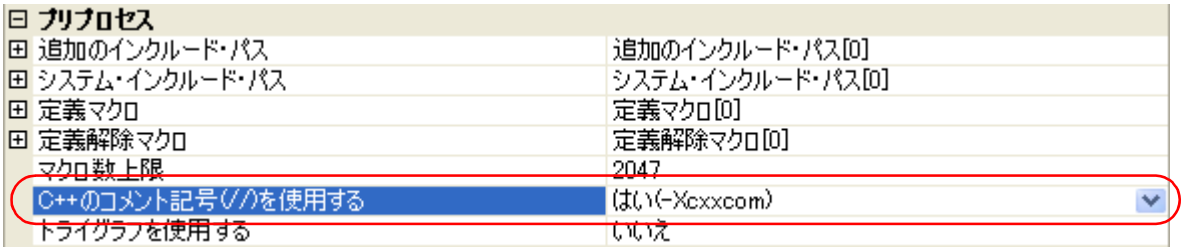
備考 ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［定義マクロ］プロパティでも，同様に設定することができます。

2. 5. 5 C++ のコメントを有効にする

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択します。

C++ のコメントを有効にするには，［プリプロセス］カテゴリの［C++ のコメント記号 (//) を使用する］プロパティで［はい (-Xcxxcom)］を選択してください（デフォルト）。

図 2—42　［C++ のコメント記号 (//) を使用する］プロパティ



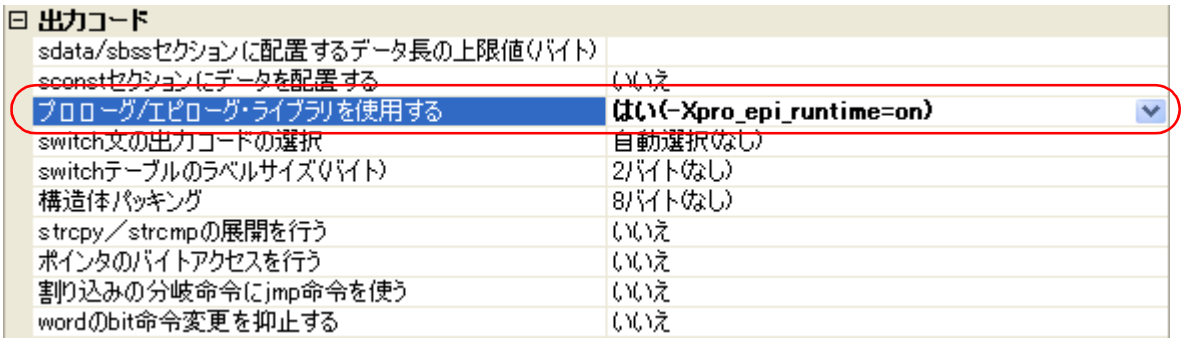
2. 5. 6　コード・サイズを削減する（プロローグ／エピローグ・ランタイム呼び出しを行う）

関数のプロローグ／エピローグ処理の一部をランタイム・ライブラリ呼び出し方式に変更することで，コード・サイズを削減することができます。ただし，callt 命令によってランタイム呼び出しが行われるため，実行時間にオーバーヘッドがかかります。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［コンパイル・オプション］タブを選択します。

関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にするには，［出力コード］カテゴリの［プロローグ／エピローグ・ライブラリを使用する］プロパティで，［はい (-Xpro_epi_runtime=on)］を選択してください。

図 2—43　［プロローグ／エピローグ・ライブラリを使用する］プロパティ

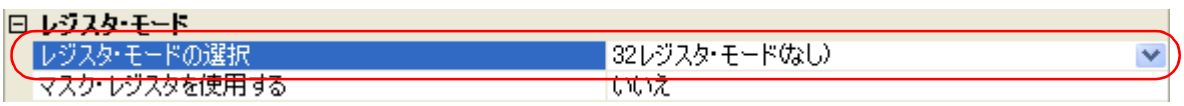


2. 5. 7　レジスタ・モードを変更する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。

［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで，レジスタ・モードを変更してください。

図 2—44　［レジスタ・モードの選択］プロパティ



以下のレジスタ・モードを選択することができます。

レジスタ・モード	作業用レジスタ	レジスタ変数用レジスタ
32 レジスタ・モード (なし) （デフォルト）	r10 ～ r19	r20 ～ r29
26 レジスタ・モード (-reg26)	r10 ～ r16	r23 ～ r29
22 レジスタ・モード (-reg22)	r10 ～ r14	r25 ～ r29

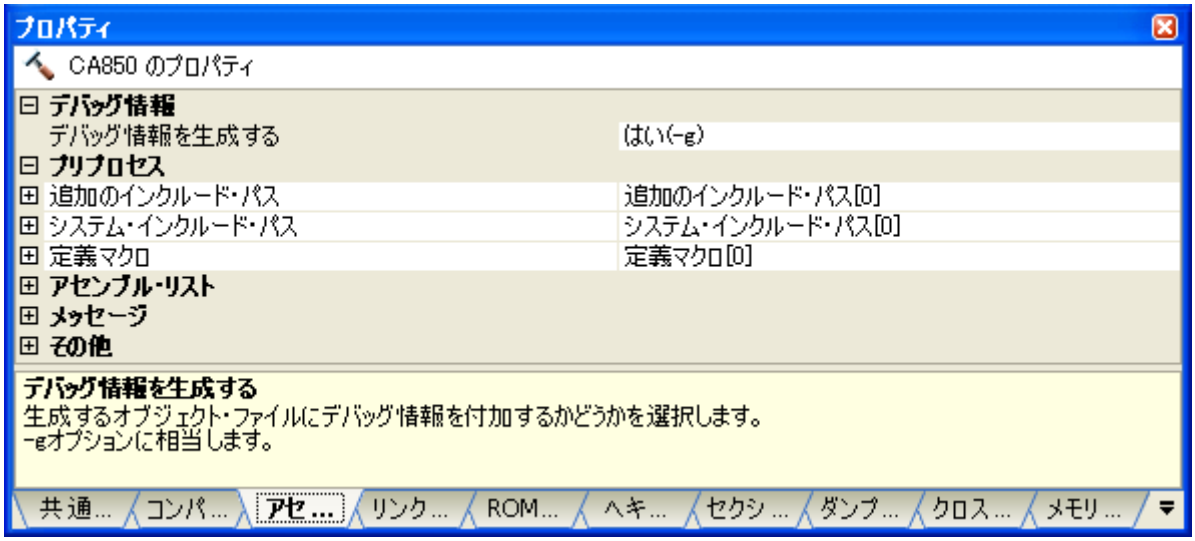
備考 レジスタ・モードについての詳細は，「CubeSuite+ V850 コーディング編」を参照してください。

2. 6 アセンブル・オプションを設定する

アセンブラに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アセンブル・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するアセンブル・オプションを設定することができます。

図 2—45 プロパティ パネル：［アセンブル・オプション］タブ



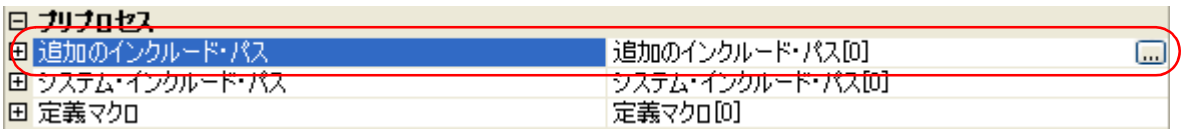
備考 よく使うオプションについては，［共通オプション］タブの［よく使うオプション（アセンブラ）］カテゴリにまとめられています。

2. 6. 1 インクルード・パスを追加する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アセンブル・オプション］タブを選択します。

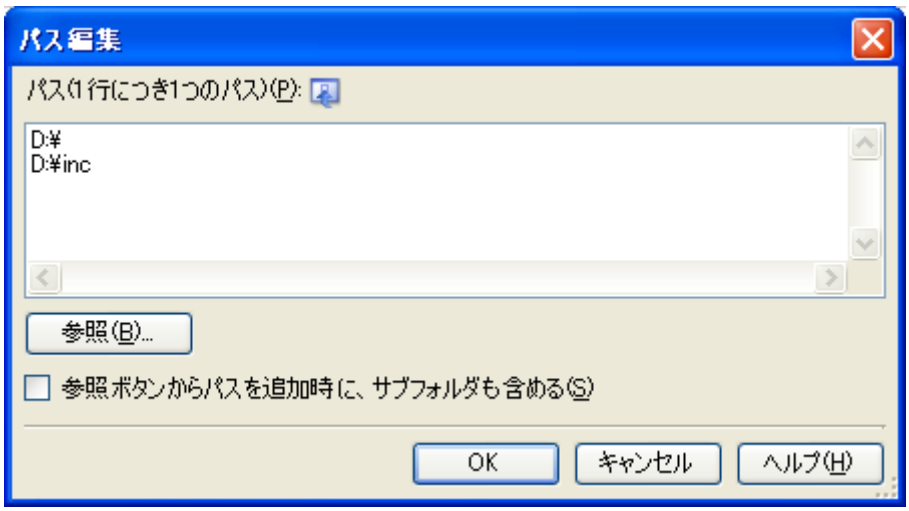
インクルード・パスの設定は，［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで行います。

図 2—46 ［追加のインクルード・パス］プロパティ



［...］ボタンをクリックすると，パス編集 ダイアログがオープンします。

図 2—47 パス編集 ダイアログ



［パス (1 行につき 1 つのパス)］にインクルード・パスを 1 行に 1 つずつ入力します。1 行に 259 文字まで，64 行まで指定可能です。

備考 インクルード・パスは，エクスプローラなどからフォルダのドラッグ・アンド・ドロップ，または［参照 ...］ボタンから指定することも可能です。［参照ボタンからパスを追加時に、サブフォルダも含める］をチェックしたのち，［参照 ...］ボタンからパスの指定を行うと，指定したパスとそのサブフォルダ 5 階層分までのパスが［パス (1 行につき 1 つのパス)］に追加されます。

［OK］ボタンをクリックすると，入力したインクルード・パスがサブプロパティとして表示されます。

図 2—48 ［追加のインクルード・パス］プロパティ（インクルード・パス追加後）

インクルード・パスの変更は，［...］ボタン，またはサブプロパティのテキスト・ボックスへの直接入力により行うことができます。

また，プロジェクト・ツリーにインクルード・ファイルを追加すると，そのインクルード・パスをサブプロパティの最初に自動で追加します。

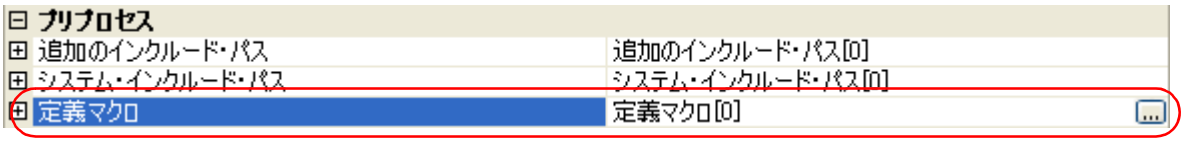
備考 ［共通オプション］タブの［よく使うオプション（アセンブラ）］カテゴリの［追加のインクルード・パス］プロパティでも，同様に設定することができます。

2. 6. 2 定義マクロを設定する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アセンブル・オプション］タブを選択します。

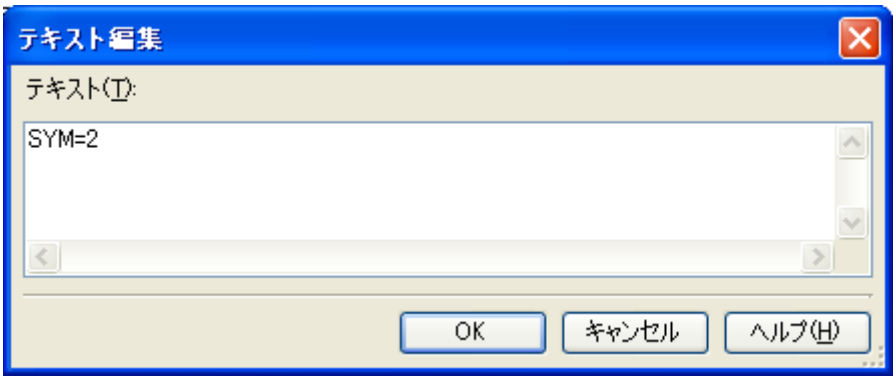
定義マクロの設定は，［プリプロセス］カテゴリの［定義マクロ］プロパティで行います。

図 2—49 ［定義マクロ］プロパティ



［...］ボタンをクリックすると，テキスト編集 ダイアログがオープンします。

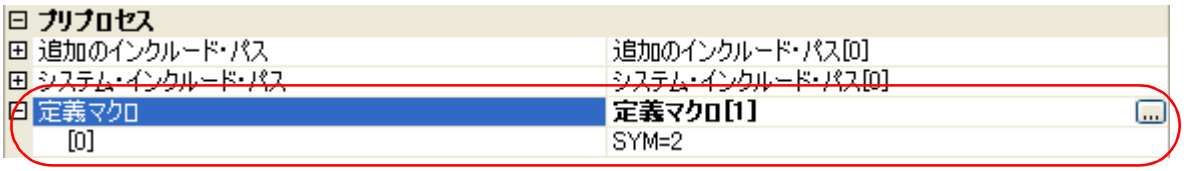
図 2—50 テキスト編集 ダイアログ



［テキスト］に定義マクロを「*マクロ名* = *定義値*」の形式で 1 行に 1 つずつ入力します。1 行に 31 文字まで，30 行まで指定可能です。「= *定義値*」の部分は省略可能で，省略した場合，定義値を 1 とします。

［OK］ボタンをクリックすると，入力した定義マクロがサブプロパティとして表示されます。

図 2—51 ［定義マクロ］プロパティ（定義マクロ設定後）



定義マクロの変更は，［...］ボタン，またはサブプロパティのテキスト・ボックスへの直接入力により行うことができます。

備考 ［共通オプション］タブの［よく使うオプション（アセンブラ）］カテゴリの［定義マクロ］プロパティでも，同様に設定することができます。

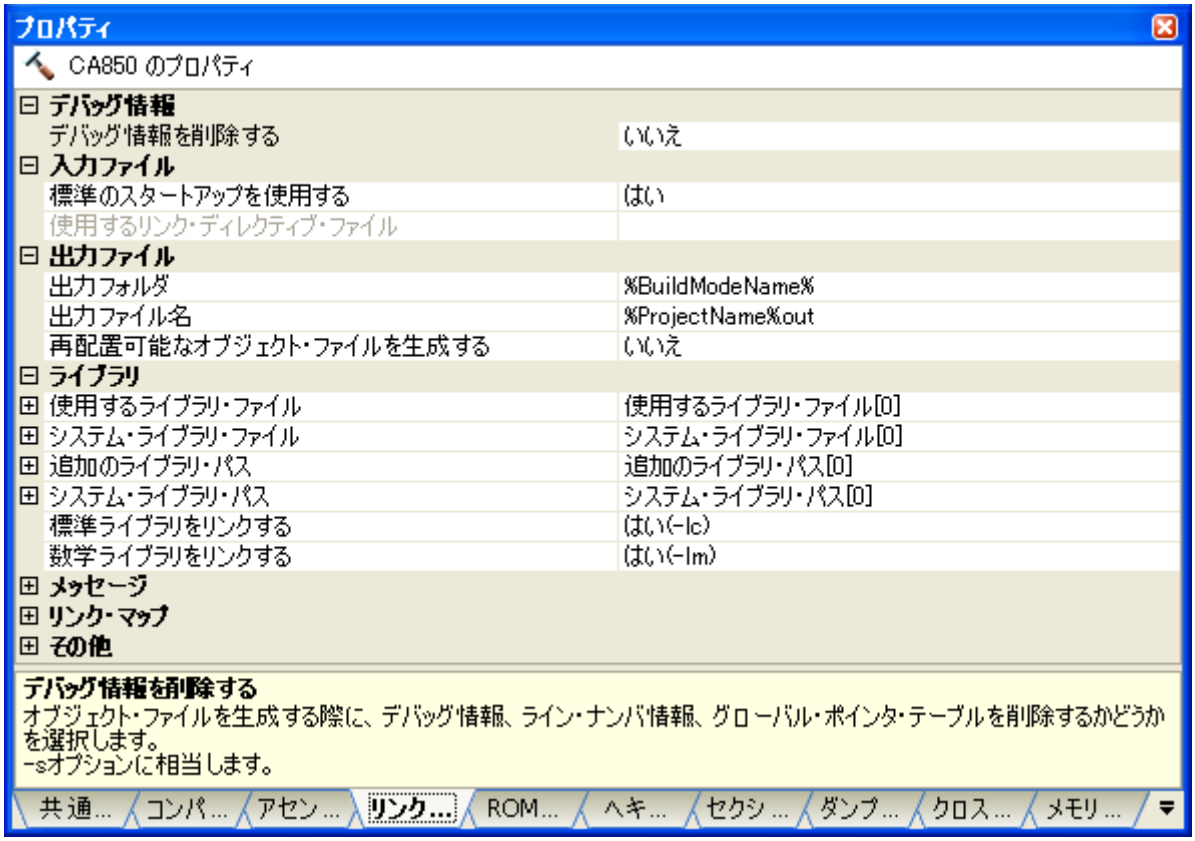
2.7 リンク・オプションを設定する

リンカに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［リンク・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するリンク・オプションを設定することができます。

注意　本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 2—52　プロパティ パネル：［リンク・オプション］タブ



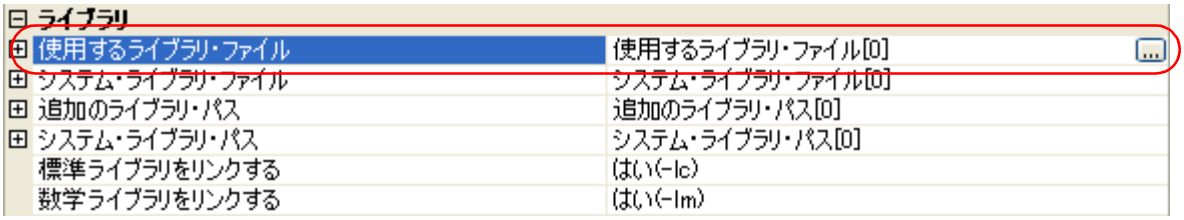
備考　よく使うオプションについては，［共通オプション］タブの［よく使うオプション（リンカ）］カテゴリにまとめられています。

2. 7. 1 ユーザ・ライブラリを追加する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［リンク・オプション］タブを選択します。

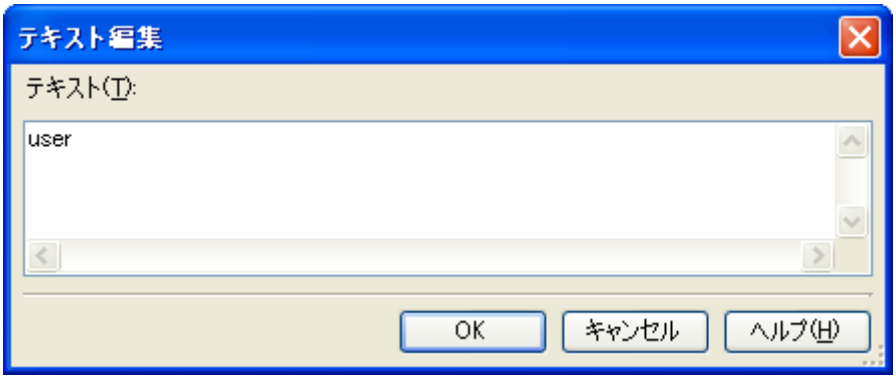
ユーザ・ライブラリの追加は，［ライブラリ］カテゴリの［使用するライブラリ・ファイル］プロパティで行います。

図 2—53 ［使用するライブラリ・ファイル］プロパティ



［...］ボタンをクリックすると，テキスト編集 ダイアログがオープンします。

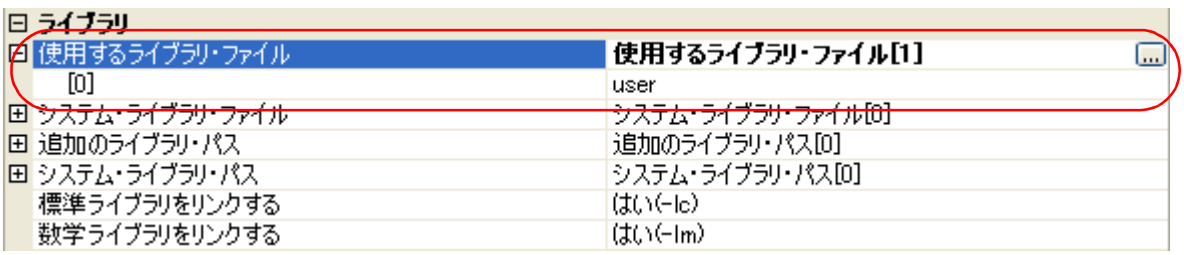
図 2—54 テキスト編集 ダイアログ



［テキスト］にライブラリ・ファイル名“lib*string*.a”のうち，*string* のみを指定します（例：“user”と指定すると，libuser.a を指定したものとみなされます）。1 行に 1 つずつ入力します。1 行に 63 文字まで，256 行まで指定可能です。

［OK］ボタンをクリックすると，入力したライブラリ・ファイルがサブプロパティとして表示されます。

図 2—55 ［使用するライブラリ・ファイル］プロパティ（ライブラリ・ファイル設定後）



ライブラリ・ファイルの変更は，［...］ボタン，またはサブプロパティのテキスト・ボックスへの直接入力により行うことができます。

備考 ［共通オプション］タブの［よく使うオプション（リンカ）］カテゴリの［使用するライブラリ・ファイル］プロパティでも，同様に設定することができます。

なお，ライブラリ・ファイルはライブラリ・パスから検索します。ライブラリ・パスを追加する場合は，［追加のライブラリ・パス］プロパティを設定してください。

注意 ライブラリ・ファイルは，プロジェクトに直接追加することでもリンクされます。その場合は，追加したライブラリ・ファイルの絶対パスで直接リンクするため，ライブラリ・パスからは検索しません。

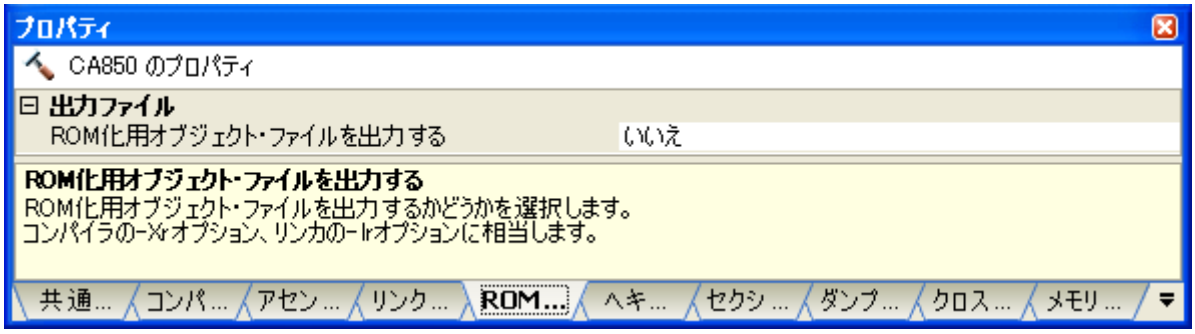
2. 8 ROM 化プロセス・オプションを設定する

ROM 化プロセッサに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ROM 化プロセス・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応する ROM 化プロセス・オプションを設定することができます。

注意 本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 2—56 プロパティ パネル：［ROM 化プロセス・オプション］タブ



備考 よく使うオプションについては，［共通オプション］タブの［よく使うオプション（ROM 化プロセッサ）］カテゴリにまとめられています。

2. 8. 1 ROM 化用オブジェクトを作成する

デフォルトで用意されている ROM 化用領域確保コード（rompcrt.o）を使用して，ROM 化用オブジェクトを作成する方法を以下に示します。

ROM 化プロセッサは，data 属性セクションの変数の初期値情報や RAM 上に配置するプログラムを，1 つのセクションにパッキングします。デフォルトでは，このセクションは“rompsec セクション”となります。rompsec セクションを ROM 上に配置し，コピー関数を呼び出すことによって，初期値情報やプログラムを RAM 上へ展開することができます。

備考 ROM 化用オブジェクトの作成方法の詳細については，「B. 4. 3 ROM 化用オブジェクトの作成」を参照してください。

(1) アプリケーションからコピー関数の呼び出し

プログラムにおいて，ROM から RAM へコピーしたいセクションを，コピー関数（_rcopy，_rcopy1，_rcopy2，_rcopy4）を使用して指定します。

コピー関数の第一引数には，rompsec セクションの先頭アドレスを指すラベル“__S_romp”（rompcrt.o で定義されているラベル）を指定します。

備考 コピー関数は，スタートアップ・ルーチン内や main 関数の先頭など，プログラムのなるべく始めの方で呼び出してください。

(2) リンク・ディレクティブの作成

ROM 化時には，.text セクションの直後に rompsec セクションが追加されます。リンク・ディレクティブにおいて，.text セクションを ROM の最後に配置することにより，ROM の終端までの rompsec セクションを配置することができます。

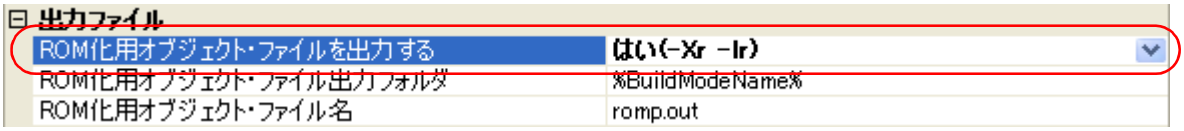
(3) ROM 化プロセス・オプションの設定

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ROM 化プロセス・オプション］タブを選択します。

(a) ROM 化用オブジェクトの出力設定

ROM 化用オブジェクトを作成するには，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択してください。

図 2—57 ［ROM 化用オブジェクト・ファイルを出力する］プロパティ



ROM 化用オブジェクト・ファイルを出力する場合，出力フォルダ，および出力ファイル名を設定することができます。

- 出力フォルダの設定

［ROM 化用オブジェクト・ファイル出力フォルダ］プロパティにおいて，テキスト・ボックスへの直接入力，または［...］ボタンにより行います。テキスト・ボックスには 247 文字まで指定可能です。デフォルトでは，“%BuildModeName%”が設定されています。“%BuildModeName%”は，埋め込みマクロで，ビルド・モード名に置換します。

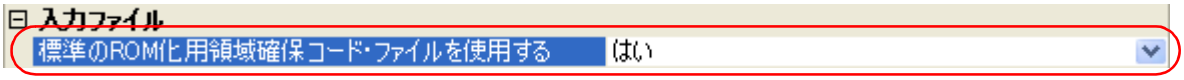
- 出力ファイル名の設定

［ROM 化用オブジェクト・ファイル名］プロパティにおいて，テキスト・ボックスへの直接入力により行います。テキスト・ボックスには 259 文字まで指定可能です。デフォルトでは，“romp.out”が設定されています。

(b) 標準の ROM 化用領域確保コード・ファイルの使用設定

標準の ROM 化用領域確保コード・ファイルを使用するには，［入力ファイル］カテゴリの［標準の ROM 化用領域確保コード・ファイルを使用する］プロパティで［はい］（デフォルト）を選択してください。

図 2―58 ［標準の ROM 化用領域確保コード・ファイルを使用する］プロパティ

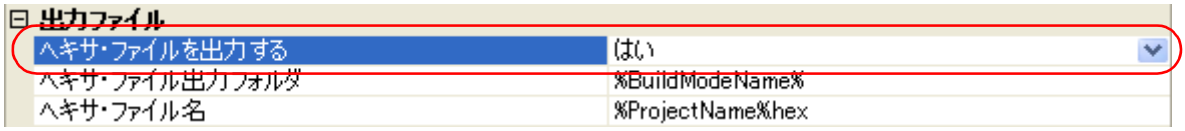


(4) ビルドの実行

ビルドを実行することにより，“__S_romp”というラベルが rompsec セクションの先頭アドレスを指すコードが生成され，ROM 化用領域確保コード（rompcrt.o）とコピー関数が格納されている ROM 化用ライブラリ（libr.a）をリンクします。そして，生成されたロード・モジュール・ファイルから，ROM 化用オブジェクト・ファイルが生成されます。

また，プロパティ パネルの［ヘキサ・コンバート・オプション］タブの［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで［はい］を選択している場合，ヘキサ・ファイルも生成されます。

図 2―59 ［ヘキサ・ファイルを出力する］プロパティ



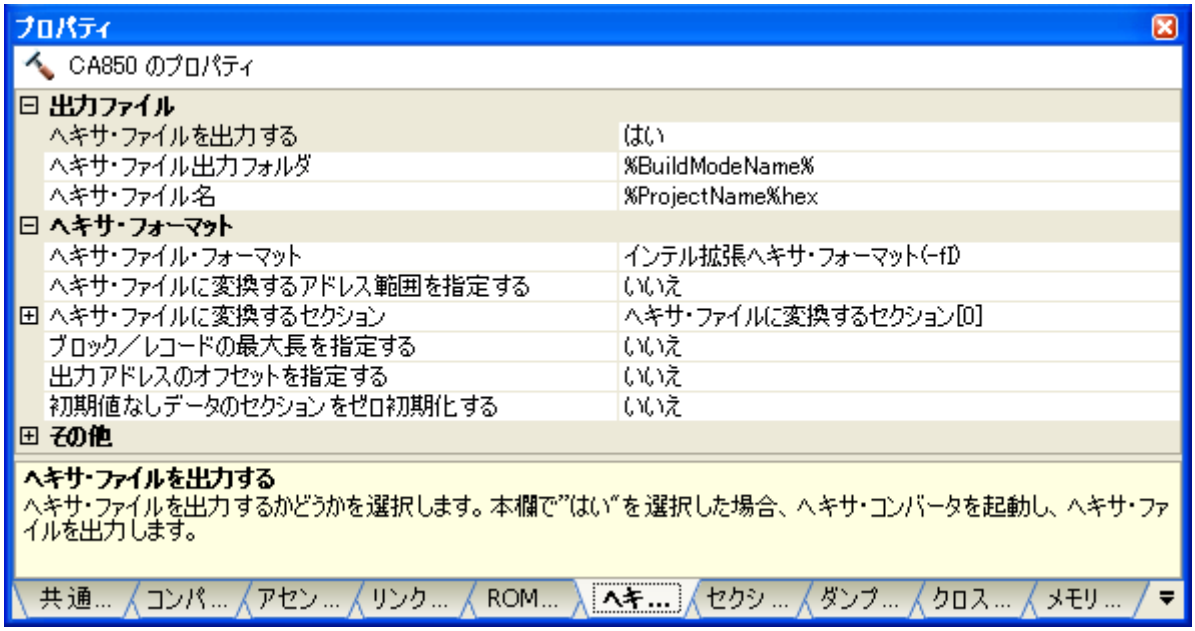
2. 9　ヘキサ・コンバート・オプションを設定する

ヘキサ・コンバータに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ヘキサ・コンバート・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するヘキサ・コンバート・オプションを設定することができます。

注意　本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 2—60　プロパティ パネル：［ヘキサ・コンバート・オプション］タブ



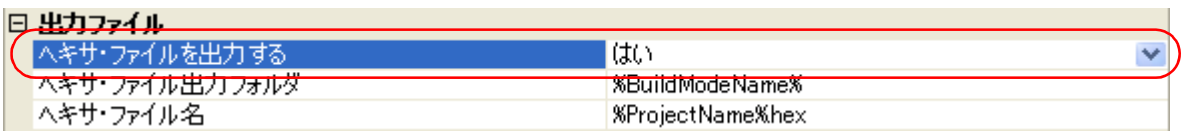
備考　よく使うオプションについては，［共通オプション］タブの［よく使うオプション（ヘキサ・コンバータ）］カテゴリにまとめられています。

2. 9. 1　ヘキサ・ファイルの出力を設定する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ヘキサ・コンバート・オプション］タブを選択します。

ヘキサ・ファイルの出力の設定は，［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで行います。ヘキサ・ファイルを出力する場合は［はい］（デフォルト），出力しない場合は［いいえ］を選択してください。

図 2—61　［ヘキサ・ファイルを出力する］プロパティ



ヘキサ・ファイルを出力する場合，出力フォルダ，および出力ファイル名を設定することができます。

(1) 出力フォルダの設定

［ヘキサ・ファイル出力フォルダ］プロパティにおいて，テキスト・ボックスへの直接入力，または［...］ボタンにより行います。テキスト・ボックスには 247 文字まで指定可能です。デフォルトでは，“%BuildModeName%”が設定されています。“%BuildModeName%”は，埋め込みマクロで，ビルド・モード名に置換します。

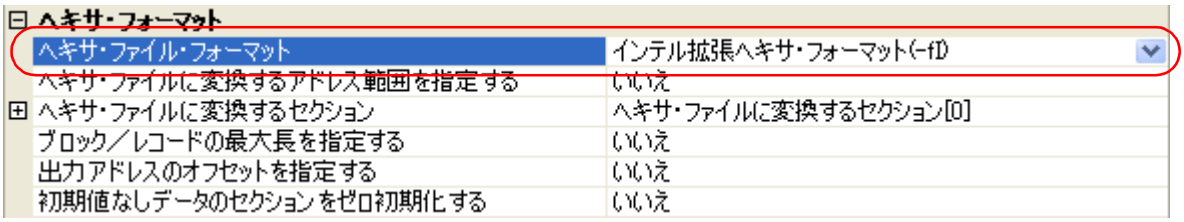
(2) 出力ファイル名の設定

［ヘキサ・ファイル名］プロパティにおいて，テキスト・ボックスへの直接入力により行います。テキスト・ボックスには 259 文字まで指定可能です。デフォルトでは，“%ProjectName%.hex”が設定されています。“%ProjectName%”は，埋め込みマクロで，プロジェクト名に置換します。

また，ヘキサ・ファイルのフォーマットを設定することができます。

［ヘキサ・フォーマット］カテゴリの［ヘキサ・ファイル・フォーマット］プロパティで，フォーマットを選択してください。

図 2—62　［ヘキサ・ファイル・フォーマット］プロパティ



以下のフォーマットを選択することができます。

フォーマット	構成
インテル拡張ヘキサ・フォーマット (-fI)（デフォルト）	スタート・アドレス・レコード，拡張アドレス・レコード，データ・レコード，およびエンド・レコードの 4 種類のレコード
モトローラ S タイプ・フォーマット（スタンダード・アドレス）(-fS)	ヘッダ・レコードである S0 レコード，データ・レコードである S2 レコード，エンド・レコードである S8 レコード
モトローラ S タイプ・フォーマット（32 ビット・アドレス）(-fs)	ヘッダ・レコードである S0 レコード，データ・レコードである S3 レコード，エンド・レコードである S7 レコード
拡張テクトロニクス・ヘキサ・フォーマット (-fT)	データ・ブロック，シンボル・ブロック，およびターミネーション・ブロックの 3 種類のブロック

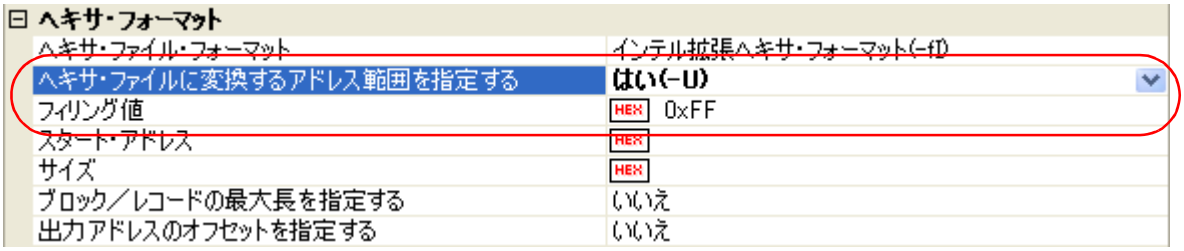
備考　ヘキサ・ファイル・フォーマットについての詳細は，「3.3　ヘキサ・コンバータ」を参照してください。

2. 9. 2 空き領域を充てんする

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ヘキサ・コンバート・オプション］タブを選択します。

空き領域の充てんについての設定は，［ヘキサ・フォーマット］カテゴリで行います。［ヘキサ・ファイルに変換するアドレス範囲を指定する］プロパティで［はい (-U)］を選択すると，［フィリング値］プロパティが表示されます。

図 2―63 ［ヘキサ・ファイルに変換するアドレス範囲を指定する］，および［フィリング値］プロパティ



テキスト・ボックスに空き領域の充てん値を直接入力してください。指定可能な値の範囲は 0x00 ～ 0xFF（16 進数）です。デフォルトでは，“0xFFFF”が設定されています。

なお，ヘキサ・ファイルに変換する領域のアドレス範囲は，［スタート・アドレス］プロパティ，［サイズ］プロパティで設定します。指定可能な値の範囲は，［スタート・アドレス］プロパティは 0x0 ～ *デバイスで扱うことができるアドレスの最大値*（16 進数），［サイズ］プロパティは 0x1 ～ *デバイスで扱うことができるアドレスの最大値*（16 進数）です。デフォルトでは，デバイス・ファイルで定義されている内蔵 ROM 領域の先頭アドレスと，そのサイズが設定されています。

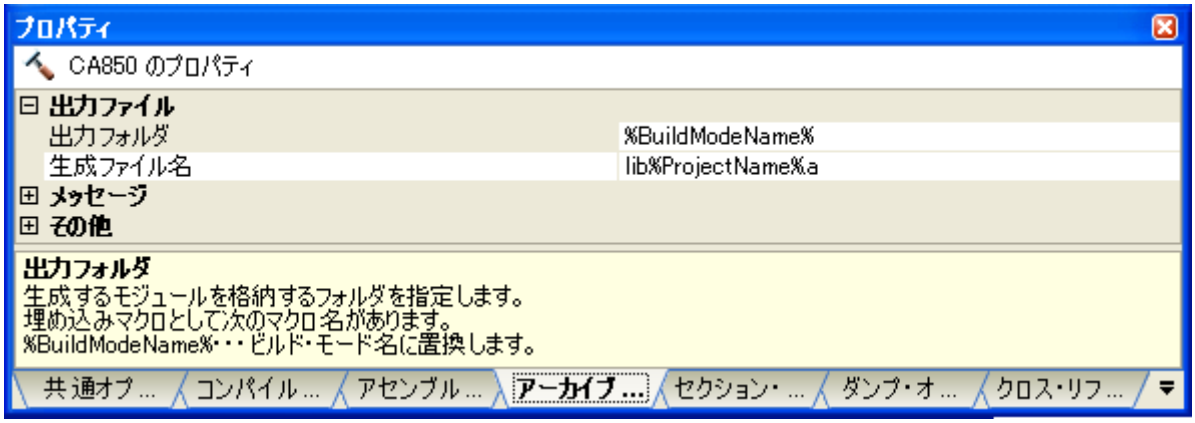
2. 10 アーカイブ・オプションを設定する

アーカイバに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アーカイブ・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するアーカイブ・オプションを設定することができます。

注意 本タブは，ライブラリ用のプロジェクトの場合のみ表示されます。

図 2—64 プロパティ パネル：［アーカイブ・オプション］タブ

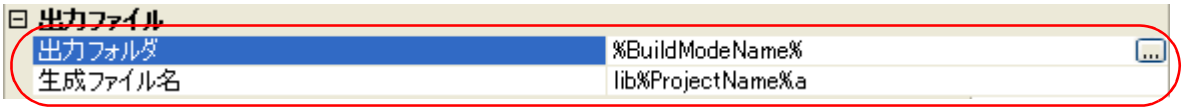


2. 10. 1 アーカイブ・ファイルの出力を設定する

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［アーカイブ・オプション］タブを選択します。

アーカイブ・ファイルの出力の設定は，［出力ファイル］カテゴリで行います。

図 2—65 ［出力ファイル］カテゴリ



(1) 出力フォルダの設定

［出力フォルダ］プロパティにおいて，テキスト・ボックスへの直接入力，または［...］ボタンにより行います。テキスト・ボックスには 247 文字まで指定可能です。デフォルトでは，“%BuildModeName%”が設定されています。“%BuildModeName%”は，埋め込みマクロで，ビルド・モード名に置換します。

(2) 出力ファイル名の設定

［生成ファイル名］プロパティにおいて，テキスト・ボックスへの直接入力により行います。テキスト・ボックスには 259 文字まで指定可能です。デフォルトでは，“%ProjectName%.a”が設定されています。“%ProjectName%”は，埋め込みマクロで，プロジェクト名に置換します。

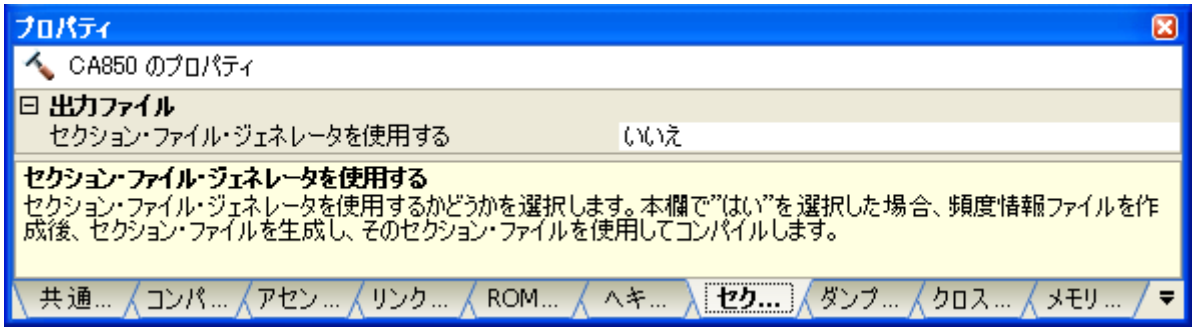
リンカのオプションで指定できるように，出力ファイル名は，先頭に“lib”を付加して，“lib%ProjectName%.a”としてください。

2. 11 セクション・ファイル・ジェネレート・オプションを設定する

セクション・ファイル・ジェネレータに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［セクション・ファイル・ジェネレート・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するセクション・ファイル・ジェネレート・オプションを設定することができます。

図 2—66 プロパティ パネル：［セクション・ファイル・ジェネレート・オプション］タブ



2. 11. 1 静的解析により変数を自動配置する

静的解析により変数を自動配置するには，セクション・ファイル・ジェネレータを使用します。セクション・ファイル・ジェネレータにより，セクション・ファイル（変数を配置するセクションを定義するファイル）を生成し，そのセクション・ファイルを使用してコンパイルを行うことで，変数が指定したセクションに配置されます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［セクション・ファイル・ジェネレート・オプション］タブを選択します。

［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択すると，空のセクション・ファイルを生成し，プロジェクトに追加します（プロジェクト・ツリーのファイル・ノードにも表示されます）。ファイルの出力先は，［セクション・ファイル出力フォルダ］プロパティ，および［セクション・ファイル名］プロパティで設定されているものとなります。

備考 すでに同名のセクション・ファイルが存在する場合は，ビルド対象に設定します。

図 2—67 ［セクション・ファイル・ジェネレータを使用する］プロパティ

図 2―68 プロジェクト・ツリー パネル（セクション・ファイル生成後）

備考 生成するセクション・ファイルの書式については，「3. 4 セクション・ファイル・ジェネレータ」を参照してください。

セクション・ファイルの出力フォルダ，および出力ファイル名の設定は，変更することもできます。

(1) 出力フォルダの設定

［セクション・ファイル出力フォルダ］プロパティにおいて，テキスト・ボックスへの直接入力，または［...］ボタンにより行います。テキスト・ボックスには 247 文字まで指定可能です。デフォルトでは，“%BuildModeName%”が設定されています。“%BuildModeName%”は，埋め込みマクロで，ビルド・モード名に置換します。

(2) 出力ファイル名の設定

［セクション・ファイル名］プロパティにおいて，テキスト・ボックスへの直接入力により行います。テキスト・ボックスには 259 文字まで指定可能です。デフォルトでは，“%ProjectName%.sf”が設定されています。“%ProjectName%”は，埋め込みマクロで，プロジェクト名に置換します。

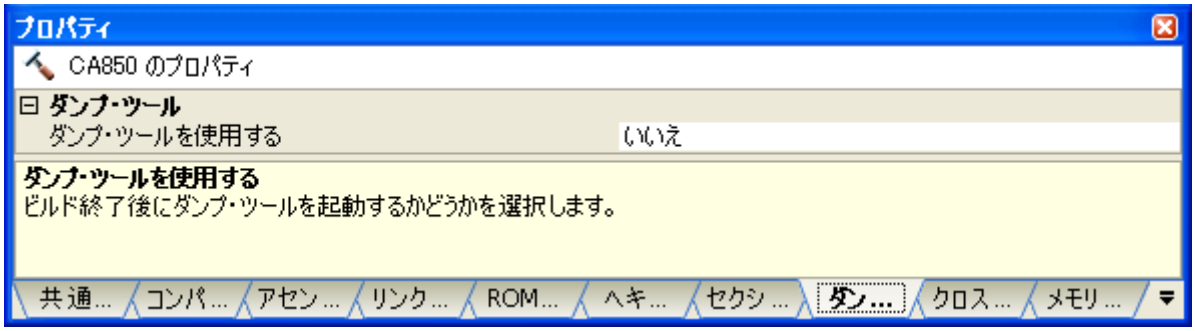
本プロパティを変更すると，空のセクション・ファイルを生成し，プロジェクトに追加します（プロジェクト・ツリーのファイル・ノードにも表示されます）。

2. 12 ダンプ・オプションを設定する

ダンプ・ツールに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ダンプ・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するダンプ・オプションを設定することができます。

図 2—69 プロパティ パネル：［ダンプ・オプション］タブ



2. 12. 1 ダンプ・ツールを使用する

ダンプ・ツールを使用すると，オブジェクト・ファイルやアーカイブ・ファイル中の，セクション／セグメントのアドレスや属性，シンボル名などの情報を出力することができます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ダンプ・オプション］タブを選択します。

ダンプ・ツールを使用するには，［ダンプ・ツール］カテゴリの［ダンプ・ツールを使用する］プロパティで［はい］を選択してください（デフォルトでは，［いいえ］が選択されています）。

図 2—70 ［ダンプ・ツールを使用する］プロパティ

備考 ダンプ・ツールが出力する各情報についての詳細は，「3. 5 ダンプ・ツール」を参照してください。

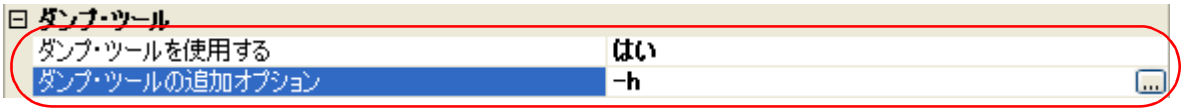
2. 12. 2 セクション情報を参照する

入力モジュール内で定義されているセクション情報を出力するには，ダンプ・ツールの -h オプションを使用します。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［ダンプ・オプション］タブを選択します。

-h オプションの設定は，［ダンプ・ツール］カテゴリで行います。［ダンプ・ツールを使用する］プロパティで［はい］を選択すると，［ダンプ・ツールの追加オプション］プロパティが表示されます。

図 2—71 ［ダンプ・ツールを使用する］，および［ダンプ・ツールの追加オプション］プロパティ



［ダンプ・ツールの追加オプション］プロパティにおいて，“-h”を指定してください。

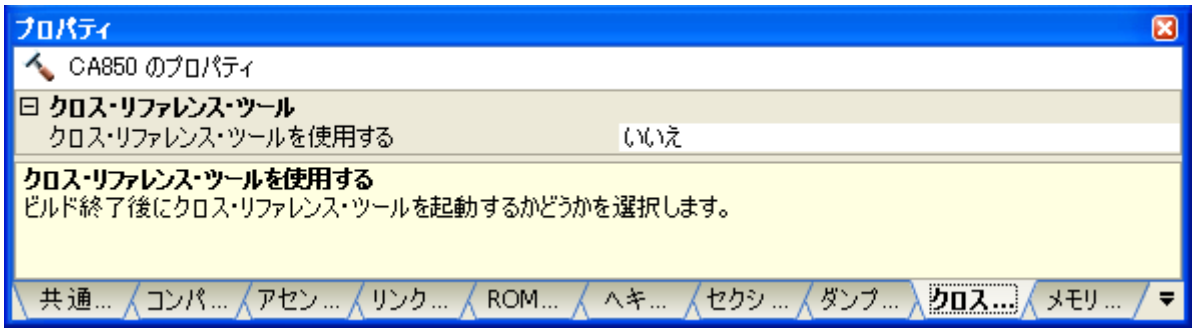
備考 出力するセクション情報については，「3.5 ダンプ・ツール」を参照してください。

2. 13 クロス・リファレンス・オプションを設定する

クロス・リファレンス・ツールに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［クロス・リファレンス・オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するクロス・リファレンス・オプションを設定することができます。

図 2—72 プロパティ パネル：［クロス・リファレンス・オプション］タブ



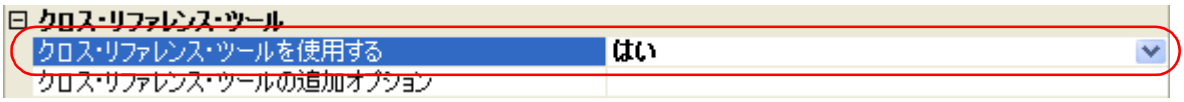
2. 13. 1 クロス・リファレンス・ツールを使用する

クロス・リファレンス・ツールを使用すると，プロジェクトに登録しているすべての C ソース・ファイルを入力とし，すべての情報（クロス・リファレンス情報，タグ・ジャンプ情報，コール・ツリー，関数計量，コール・データベース）をテキスト形式，および CSV 形式の各ファイルに出力することができます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［クロス・リファレンス・オプション］タブを選択します。

クロス・リファレンス・ツールを使用するには，［クロス・リファレンス・ツール］カテゴリの［クロス・リファレンス・ツールを使用する］プロパティで［はい］を選択してください（デフォルトでは，［いいえ］が選択されています）。

図 2—73 ［クロス・リファレンス・ツールを使用する］プロパティ



備考 クロス・リファレンス・ツールが出力する各情報についての詳細は，「3. 7 クロス・リファレンス・ツール」を参照してください。

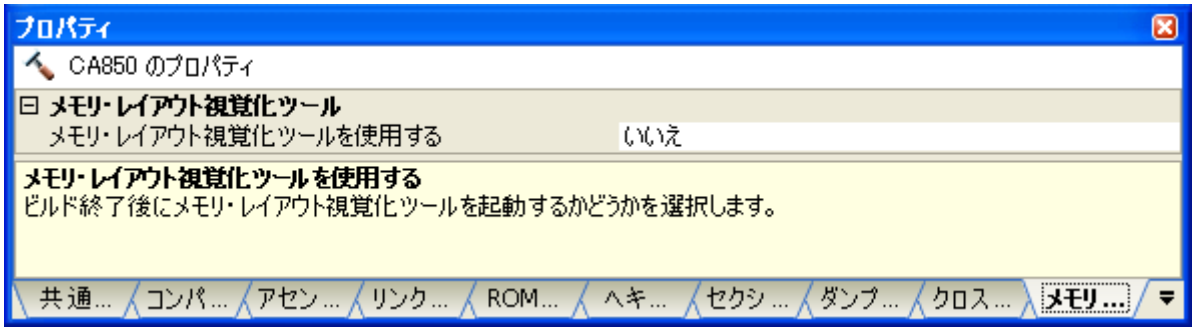
2. 14 メモリ・レイアウト視覚化オプションを設定する

メモリ・レイアウト視覚化ツールに対するオプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［メモリ・レイアウト視覚化オプション］タブを選択してください。

タブ上で各プロパティを設定することにより，対応するメモリ・レイアウト視覚化オプションを設定することができます。

注意 本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 2—74 プロパティ パネル：［メモリ・レイアウト視覚化オプション］タブ



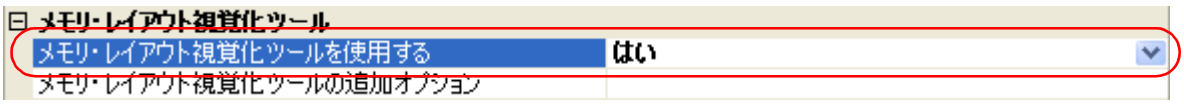
2. 14. 1 メモリ・レイアウト視覚化ツールを使用する

メモリ・レイアウト視覚化ツールを使用すると，オブジェクト・ファイル（*.out）を入力とし，メモリ・マップ表（変数のメモリ配置情報）をテキスト形式，および CSV 形式のファイルに出力することができます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［メモリ・レイアウト視覚化オプション］タブを選択します。

メモリ・レイアウト視覚化ツールを使用するには，［メモリ・レイアウト視覚化ツール］カテゴリの［メモリ・レイアウト視覚化ツールを使用する］プロパティで［はい］を選択してください（デフォルトでは，［いいえ］が選択されています）。

図 2—75 ［メモリ・レイアウト視覚化ツールを使用する］プロパティ



備考 メモリ・マップ表についての詳細は，「3. 8 メモリ・レイアウト視覚化ツール」を参照してください。

2. 15 個別にビルド・オプションを設定する

ビルド・オプションの設定は，プロジェクト単位，またはファイル単位で行います。

- プロジェクト単位 → 「2. 15. 1 プロジェクト単位でビルド・オプションを設定する」参照
- ファイル単位 → 「2. 15. 2 ファイル単位でコンパイル／アセンブル・オプションを設定する」参照

2. 15. 1 プロジェクト単位でビルド・オプションを設定する

プロジェクト（メイン・プロジェクト，またはサブプロジェクト）に対するビルド・オプションを設定するには，プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルを表示します。

コンポーネントに対する各タブを選択し，必要なプロパティを設定することにより，ビルド・オプションを設定することができます。

コンパイラ	→ ［コンパイル・オプション］タブ
アセンブラ	→ ［アセンブル・オプション］タブ
リンカ	→ ［リンク・オプション］タブ
ROM 化プロセッサ	→ ［ROM 化プロセス・オプション］タブ
ヘキサ・コンバータ	→ ［ヘキサ・コンバート・オプション］タブ
アーカイバ	→ ［アーカイブ・オプション］タブ
セクション・ファイル・ジェネレータ	→ ［セクション・ファイル・ジェネレート・オプション］タブ
ダンプ・ツール	→ ［ダンプ・オプション］タブ
クロス・リファレンス・ツール	→ ［クロス・リファレンス・オプション］タブ
メモリ・レイアウト視覚化ツール	→ ［メモリ・レイアウト視覚化オプション］タブ

2. 15. 2 ファイル単位でコンパイル／アセンブル・オプションを設定する

プロジェクトに追加している各ソース・ファイルに対して，コンパイル・オプション，またはアセンブル・オプションを個別に設定することができます。

(1) C ソース・ファイルにコンパイル・オプションを設定する場合

プロジェクト・ツリー で C ソース・ファイルを選択し，プロパティ パネルの［ビルド設定］タブを選択します。［ビルド］カテゴリの［個別コンパイル・オプションを設定する］プロパティで［はい］を選択すると，「図 2—77 メッセージ ダイアログ」のメッセージ ダイアログがオープンします。

図 2—76 ［個別コンパイル・オプションを設定する］プロパティ

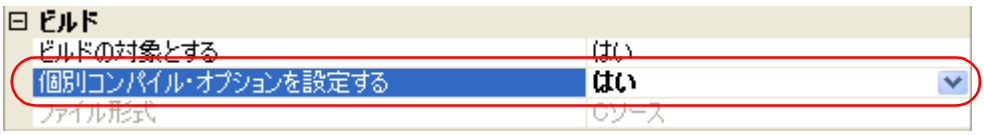
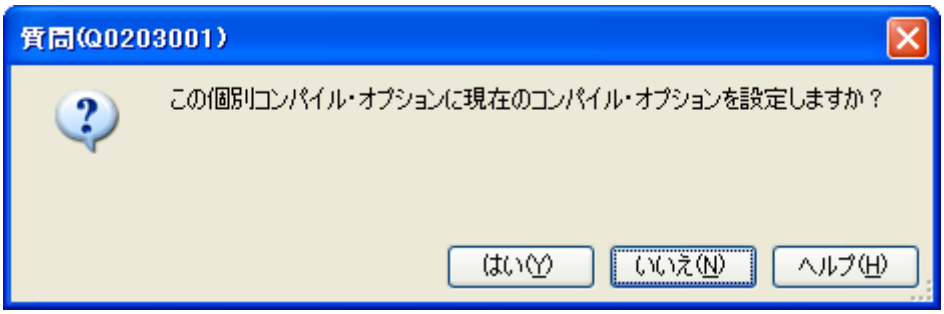
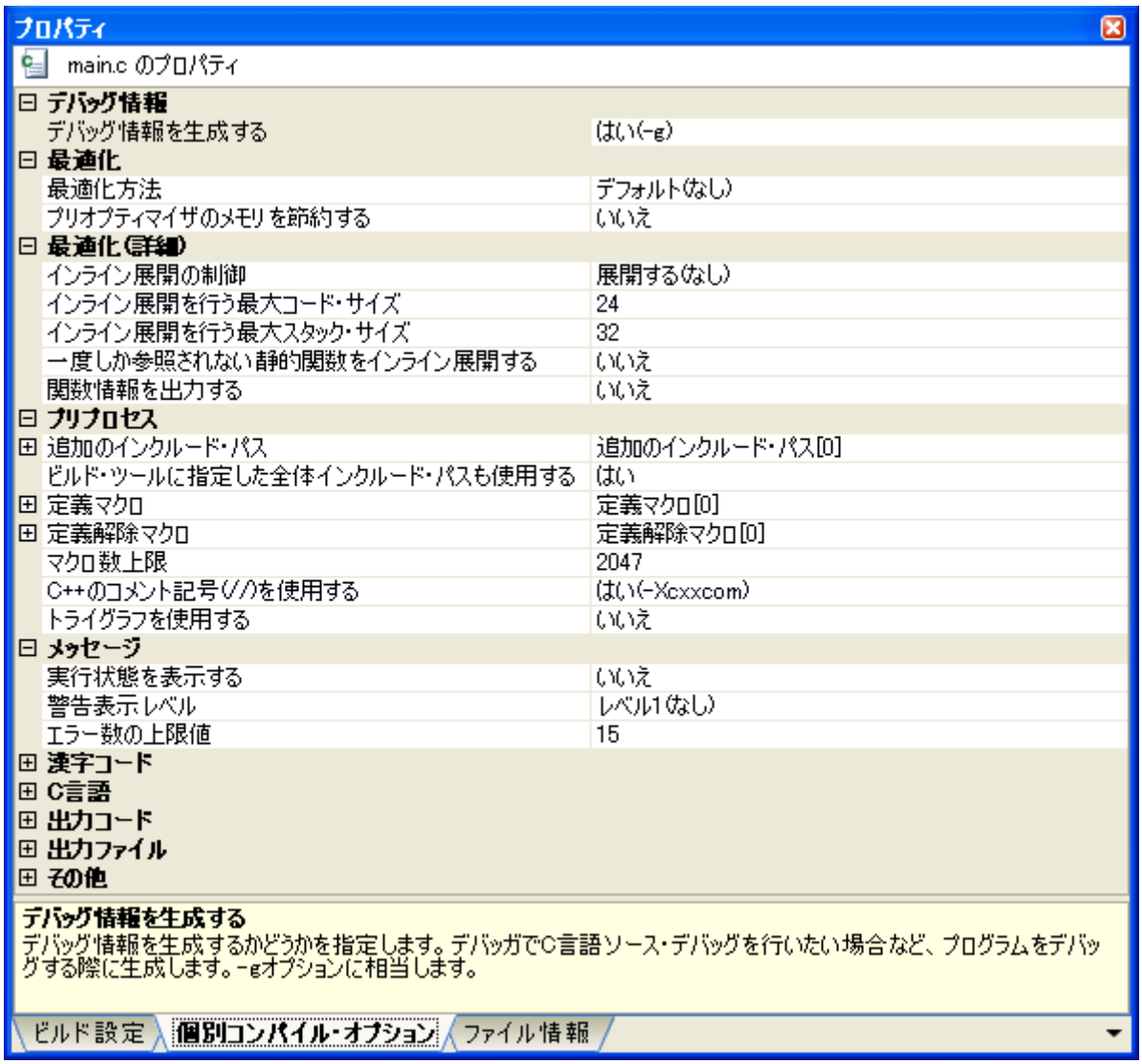


図 2—77 メッセージ ダイアログ



ダイアログ上で［はい］をクリックすると，［個別コンパイル・オプション］タブが表示されます。

図 2—78 プロパティ パネル：［個別コンパイル・オプション］タブ



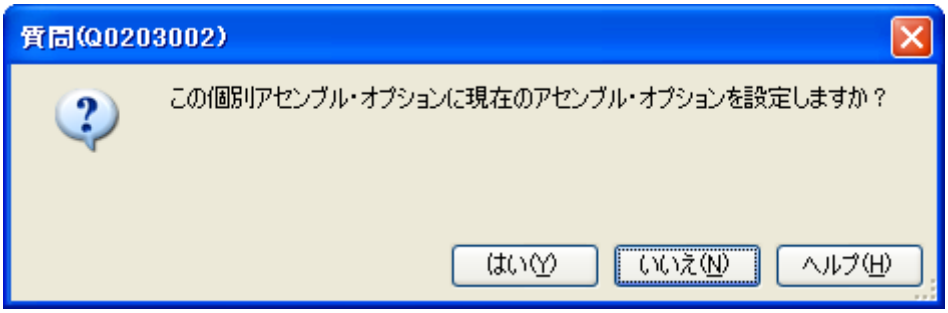
タブ上で必要なプロパティを設定することにより，C ソース・ファイルに対するコンパイル・オプションを設定することができます。なお，本タブは，デフォルトでは［コンパイル・オプション］タブの設定内容を継承します。

(2) アセンブラ・ソース・ファイルにアセンブル・オプションを設定する場合

プロジェクト・ツリー でアセンブラ・ソース・ファイルを選択し，プロパティ パネルの［ビルド設定］タブを選択します。［ビルド］カテゴリの［個別アセンブル・オプションを設定する］プロパティで［はい］を選択すると，「図 2—80　メッセージ ダイアログ」のメッセージ ダイアログがオープンします。

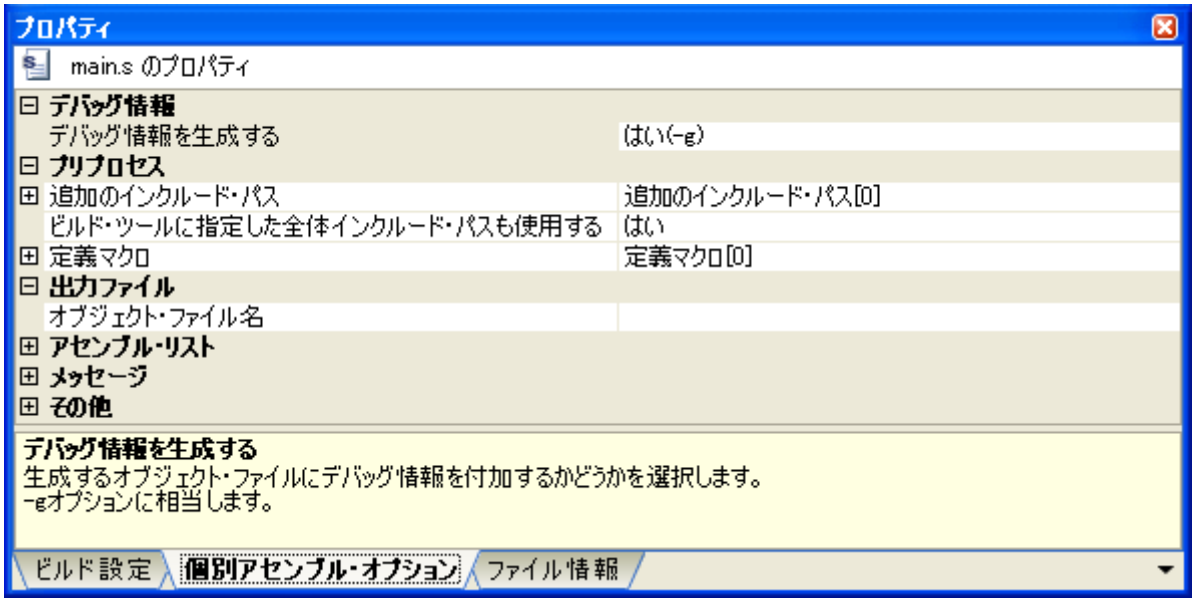
図 2—79 ［個別アセンブル・オプションを設定する］プロパティ

図 2—80　メッセージ ダイアログ



ダイアログ上で［はい］をクリックすると，［個別アセンブル・オプション］タブが表示されます。

図 2—81　プロパティ パネル：［個別アセンブル・オプション］タブ



タブ上で必要なプロパティを設定することにより，アセンブラ・ソース・ファイルに対するアセンブル・オプションを設定することができます。なお，本タブは，デフォルトでは［アセンブル・オプション］タブの設定内容を継承します。

備考　C ソース・ファイルから生成されるアセンブラ・ソース・ファイルに対してもアセンブル・オプションを設定することができます。プロジェクト・ツリー で C ソース・ファイルを選択し，プロパティ パネルの［個別コンパイル・オプション］タブを選択します。［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択すると，［個別アセンブル・オプション］タブが表示されます。

2. 16 ブートーフラッシュの再リンク機能を実現するための準備をする

システムによっては，書き換え／取り換えが不可能な領域（ブート領域）に加え，フラッシュや外付け ROM といった，書き換え／取り換えが可能な領域（フラッシュ領域）を使用することがあります。

このようなシステムにおいて，フラッシュ領域上のプログラムのみを変更したい場合，ブート領域上のプログラムの再構築を行わず，ブート領域とフラッシュ領域間の関数呼び出しを正常に行う機能を“再リンク機能”と呼んでいます。

ブート領域側，フラッシュ領域側のロード・モジュール・ファイルを作成することにより，再リンク機能が実現されます。再リンク機能の実現方法を以下に示します。

備考 再リンク機能とその実現方法についての詳細は，「B. 3. 3 ブートーフラッシュ再リンク機能」を参照してください。

2. 16. 1 ビルド対象ファイルを準備する

（1）リンク・ディレクティブ・ファイルの用意

リンク・ディレクティブ・ファイルを，ブート領域側，フラッシュ領域側のプロジェクト用にそれぞれ用意します。

備考 ブート領域側，フラッシュ領域側で同じリンク・ディレクティブ・ファイルを使用してもかまいませんが，記述が煩雑になるため，それぞれの領域用にリンク・ディレクティブ・ファイルを使用することを推奨します。

（2）.ext_func 疑似命令の記述

アセンブラ・ソース・ファイルに，.ext_func 疑似命令を記述します。

.ext_func 疑似命令で，対象となる関数（実体がフラッシュ領域に存在し，ブート領域から呼び出される関数）について，ID 値を指定します。

備考 記述漏れやソース間の矛盾が生じることを防ぐため，.ext_func 疑似命令の記述は 1 つのファイルにまとめ，ブート領域側，フラッシュ領域側を問わず，全アセンブラ・ソース・ファイルに .include 疑似命令でインクルードすることを推奨します。

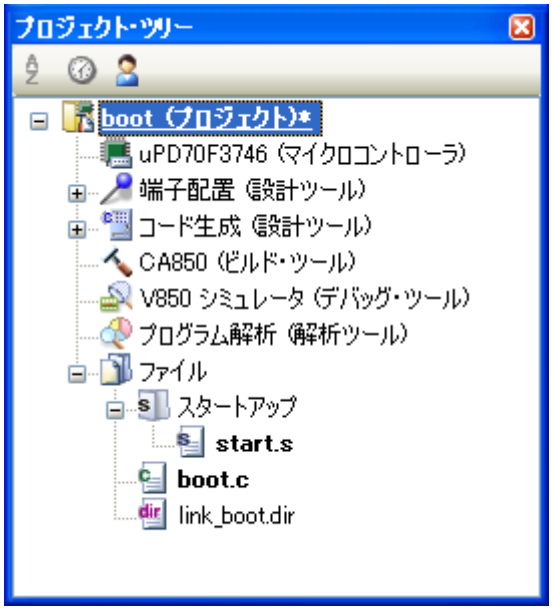
2. 16. 2 ブート領域側のプロジェクトを設定する

（1）ブート領域側のプロジェクトの作成

ブート領域側のプロジェクトを作成し，ビルド対象ファイルをプロジェクトに追加します。

スタートアップ・ルーチンは，スタートアップ・ノードに追加します。

図 2—82 ブート領域側のプロジェクト

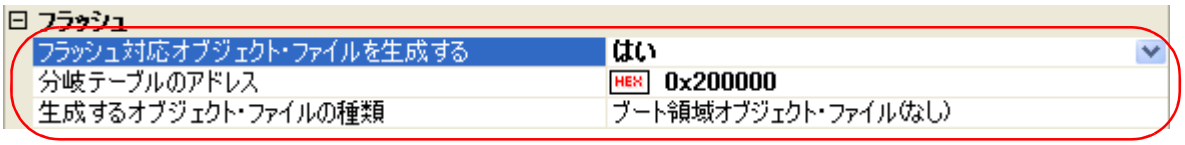


(2) ブート領域側のプロジェクトのビルド・オプションの設定

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。ビルド・オプションの設定は，［フラッシュ］カテゴリで行います。

［フラッシュ対応オブジェクト・ファイルを生成する］プロパティで［はい］を選択すると，［分岐テーブルのアドレス］プロパティと［生成するオブジェクト・ファイルの種類］プロパティが表示されます。

図 2—83 ブート領域側の［フラッシュ対応オブジェクト・ファイルを生成する］，［分岐テーブルのアドレス］，および［生成するオブジェクト・ファイルの種類］プロパティ



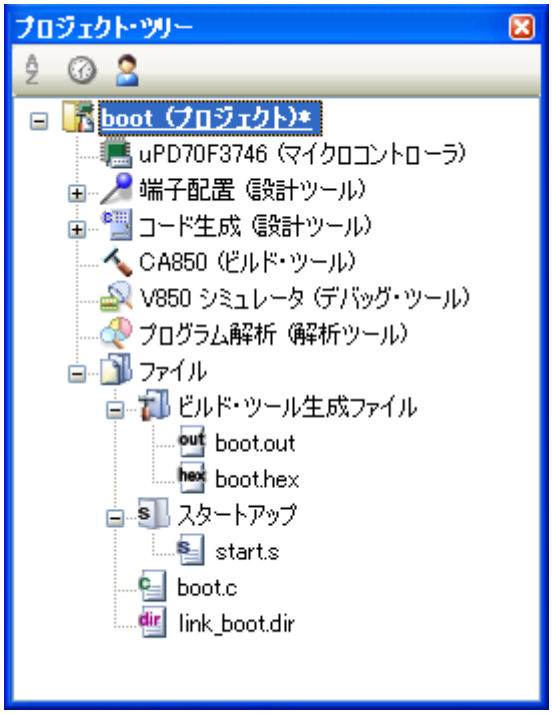
［分岐テーブルのアドレス］プロパティで，分岐テーブルの先頭アドレス（フラッシュ領域内のアドレス）を指定します。指定可能な値の範囲は，0x0 ～ 0xffffffff（16 進数）です。デフォルトでは，“0x0”が設定されています。

また，［生成するオブジェクト・ファイルの種類］プロパティで，［ブート領域オブジェクト・ファイル (なし)］を選択します（デフォルト）。

(3) ブート領域側のプロジェクトのビルドの実行

ブート領域側のプロジェクトのビルドを実行すると，ロード・モジュール・ファイルが生成されます。

図 2—84 ブート領域側の生成ファイル



2. 16. 3 フラッシュ領域側のプロジェクトを設定する

(1) フラッシュ領域側のプロジェクトの作成

フラッシュ領域側のプロジェクトを作成し，ビルド対象ファイルをプロジェクトに追加します。

スタートアップ・ルーチンは，スタートアップ・ノードに追加します。

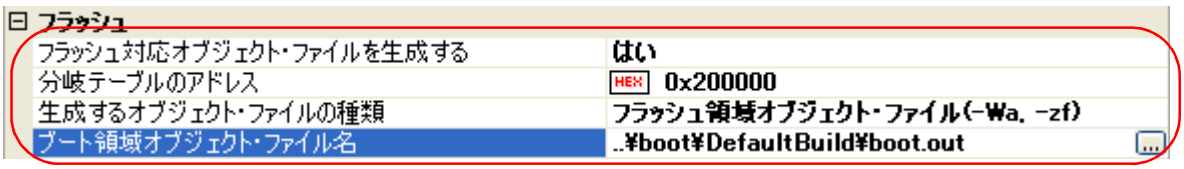
図 2—85 フラッシュ領域側のプロジェクト

(2) フラッシュ領域側のプロジェクトのビルド・オプションの設定

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。ビルド・オプションの設定は，［フラッシュ］カテゴリで行います。

［フラッシュ対応オブジェクト・ファイルを生成する］プロパティで［はい］を選択すると，［分岐テーブルのアドレス］プロパティと［生成するオブジェクト・ファイルの種類］プロパティが表示されます。

図 2—86 ［フラッシュ対応オブジェクト・ファイルを生成する］，［分岐テーブルのアドレス］，［生成するオブジェクト・ファイルの種類］，および［ブート領域オブジェクト・ファイル名］プロパティ



［分岐テーブルのアドレス］プロパティで，分岐テーブルの先頭アドレス（ブート領域側と同じアドレス）を指定します。

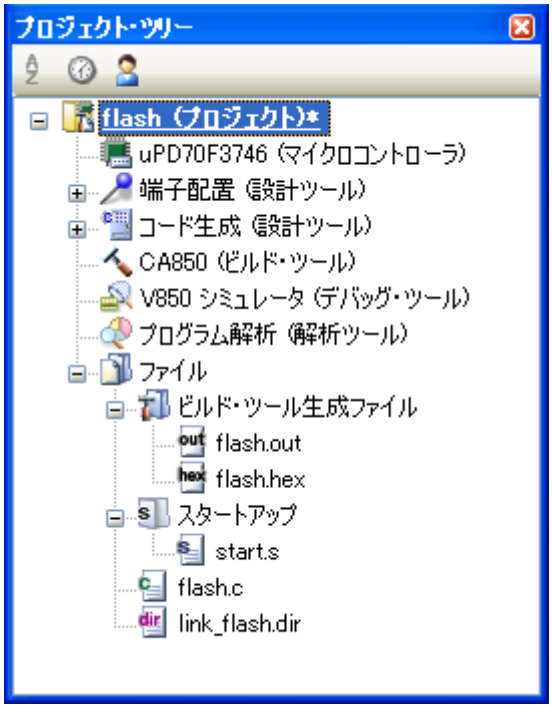
また，［生成するオブジェクト・ファイルの種類］プロパティで［フラッシュ領域オブジェクト・ファイル (-Wa, -zf)］を選択すると，［ブート領域オブジェクト・ファイル名］プロパティが表示されます。ここで，ブート領域側のオブジェクト・ファイルを指定します。

注意 ここで指定するものは，リンカが出力したオブジェクト・ファイルです。ROM 化プロセッサが出力したオブジェクト・ファイルを指定すると，エラーとなるので注意してください。

(3) フラッシュ領域側のプロジェクトのビルドの実行

フラッシュ領域側のプロジェクトのビルドを実行することにより，再リンク機能を実現したロード・モジュール・ファイルが生成されます。

図 2—87 フラッシュ領域側の生成ファイル



2. 17 ビルドの設定をする

ここでは，ビルドに関する以下の操作を説明します。

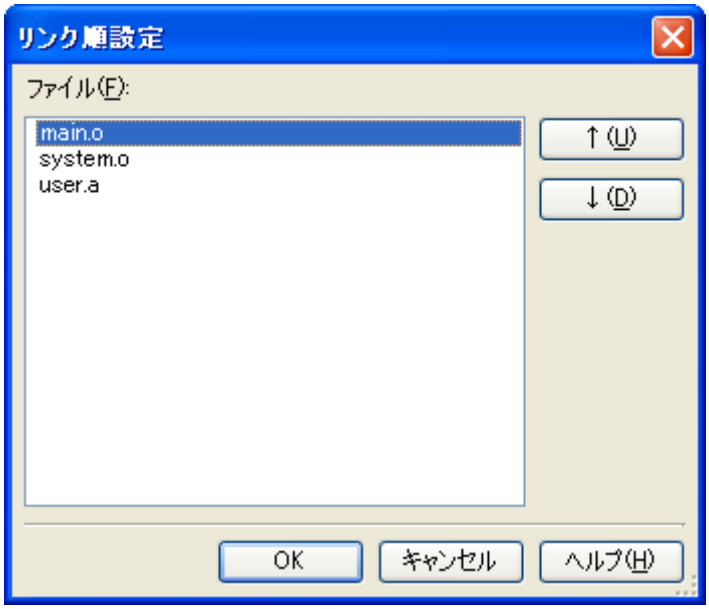
- ファイルのリンク順を設定する
- サブプロジェクトのビルド順を変更する
- ビルド・オプションを一覧表示する
- ビルド対象プロジェクトを変更する
- ビルド・モードを追加する
- ビルド・モードを変更する
- ビルド・モードを削除する
- 現在のビルド・オプションをプロジェクトの標準に設定する

2. 17. 1 ファイルのリンク順を設定する

オブジェクト・モジュール・ファイル，およびライブラリ・ファイルのリンク順は，自動で決定されますが，ユーザが設定することもできます。

プロジェクト・ツリーでビルド・ツール・ノードを選択し，コンテキスト・メニューの［リンク順を設定する...］を選択すると，リンク順設定 ダイアログがオープンします。

図 2—88 リンク順設定 ダイアログ



［ファイル］には，以下のファイルのファイル名一覧が，リンカへの入力順で表示されます。

- 選択しているメイン・プロジェクト，またはサブプロジェクトに追加されているソース・ファイルから生成されるオブジェクト・モジュール・ファイル
- 選択しているメイン・プロジェクト，またはサブプロジェクトのプロジェクト・ツリーに直接追加したオブジェクト・モジュール・ファイル
- 選択しているメイン・プロジェクト，またはサブプロジェクトのプロジェクト・ツリーに直接追加したライブラリ・ファイル

備考 デフォルトでは，プロジェクトに追加されている順番となります。
新規に追加したソース・ファイルから生成されるオブジェクト・モジュール・ファイル，および新規に追加したオブジェクト・モジュール・ファイルは，一覧の最後のオブジェクト・モジュール・ファイルの次に追加されます。新規に追加したライブラリ・ファイルは，一覧の最後に追加されます。

ファイルの表示順を変更することにより，リンカへのファイルの入力順を設定することができます。
表示順の変更は，［↑］，および［↓］ボタン，またはファイル名のドラッグ・アンド・ドロップにより行います。表示順を変更したのち，［OK］ボタンをクリックしてください。

2. 17. 2 サブプロジェクトのビルド順を変更する

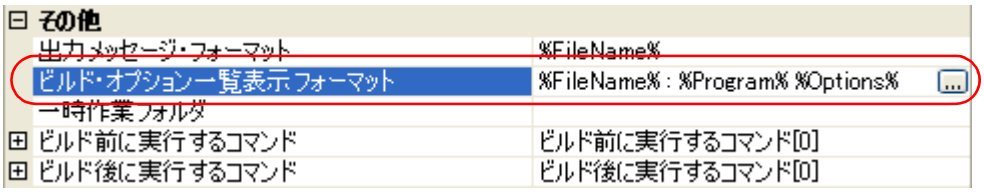
ビルドの実行は，サブプロジェクト，メイン・プロジェクトの順で行いますが，複数のサブプロジェクトを追加している場合，サブプロジェクトのビルド順はプロジェクト・ツリーでの表示順となります。
プロジェクト・ツリーでのサブプロジェクトの表示順を変更するには，移動するサブプロジェクトをドラッグし，移動先でドロップしてください。

2. 17. 3 ビルド・オプションを一覧表示する

プロジェクト（メイン・プロジェクト，およびサブプロジェクト）に対して，プロパティ パネルで現在設定しているビルド・オプションを一覧表示することができます。
［ビルド］メニュー→［ビルド・オプション一覧］を選択すると，プロジェクトに対する現在のオプションの設定内容が，出力 パネルの［ビルド・ツール］タブにビルド順に表示されます。

備考 ビルド・オプション一覧の表示フォーマットは，変更することができます。
プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。［その他］カテゴリの［ビルド・オプション一覧表示フォーマット］プロパティを設定してください。

図 2—89 ［ビルド・オプション一覧表示フォーマット］プロパティ



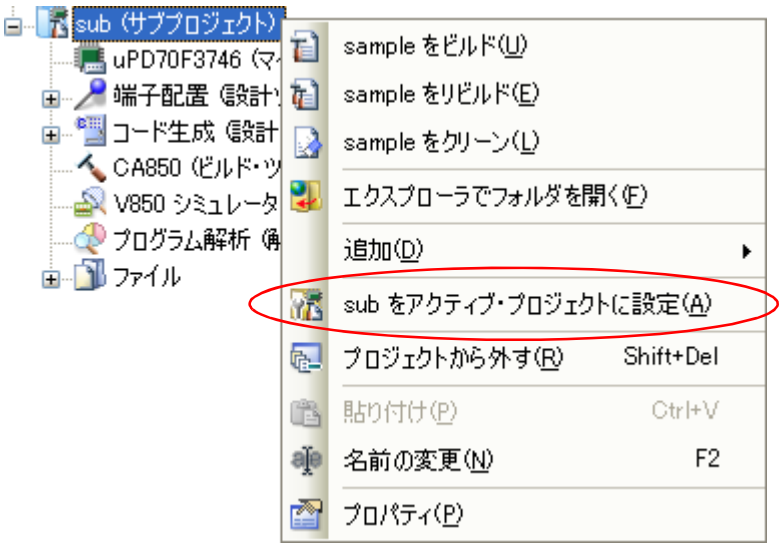
デフォルトでは，“%FileName% : %Program% %Options%”が設定されています。
“%FileName%”，“%Program%”，“%Options%”は，埋め込みマクロで，それぞれビルド中のファイル名，実行中のプログラム名，ビルド時のコマンド・ライン・オプションに置換します。

2. 17. 4 ビルド対象プロジェクトを変更する

特定のプロジェクト（メイン・プロジェクト，またはサブプロジェクト）を対象にビルドを行う場合，そのプロジェクトを“アクティブ・プロジェクト”として設定する必要があります。

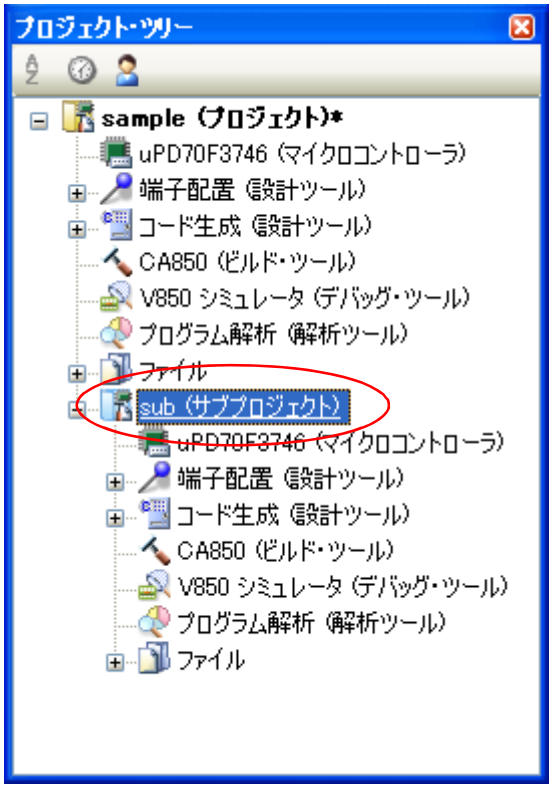
アクティブ・プロジェクトを設定するには，プロジェクト・ツリーでアクティブ・プロジェクトに設定するメイン・プロジェクト・ノード，またはサブプロジェクト・ノード選択し，コンテキスト・メニューの［*選択しているプロジェクト*をアクティブ・プロジェクトに設定］を選択してください。

図 2—90 ［*選択しているプロジェクト*をアクティブ・プロジェクトに設定］項目



アクティブ・プロジェクトを設定すると，そのプロジェクトには下線が付加されます。

図 2—91 アクティブ・プロジェクト



備考 1. プロジェクトの作成直後は，メイン・プロジェクトがアクティブ・プロジェクトとなります。

2. アクティブ・プロジェクトに設定しているサブプロジェクトをプロジェクトから外した場合は，メイン・プロジェクトがアクティブ・プロジェクトとなります。

2. 17. 5 ビルド・モードを追加する

ビルドの目的に応じてビルド・オプションや定義マクロを変更したい場合，それらの設定を一括して変更することができます。ビルド・オプションや定義マクロの設定をまとめたものをビルド・モードと呼び，ビルド・モードを変更することにより，ビルド・オプションや定義マクロの設定を毎回変更する必要がなくなります。

ビルド・モードは，デフォルトでは“DefaultBuild”のみ用意していますので，ビルドの目的に応じてユーザが追加してください。

以下に，ビルド・モードの追加方法を示します。

(1) ビルド・モードの新規作成

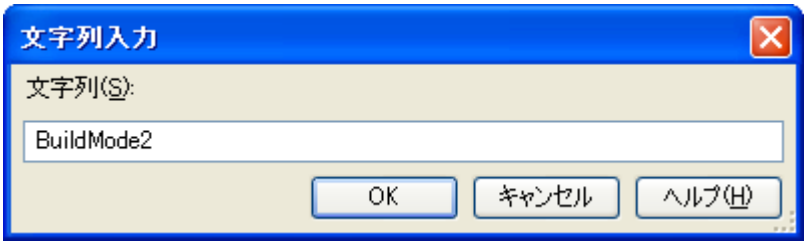
新規のビルド・モードの作成は，既存のビルド・モードの複製により行います。

［ビルド］メニュー→［ビルド・モードの設定 ...］を選択すると，ビルド・モード設定 ダイアログがオープンします。

図 2—92 ビルド・モード設定 ダイアログ

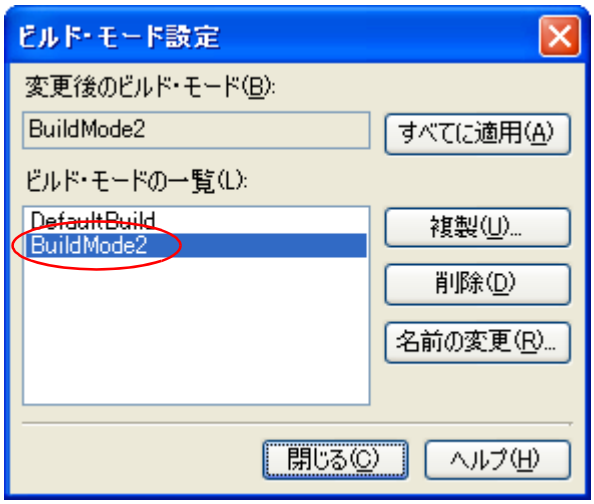
ビルド・モードの一覧から複製元のビルド・モードを選択したのち，［複製 ...］ボタンをクリックすると，文字列入力 ダイアログがオープンします。

図 2—93 文字列入力 ダイアログ



ダイアログ上で新規作成するビルド・モードの名前を入力し，［OK］ボタンをクリックすると，その名前でビルド・モードを複製します。プロジェクトに属するメイン・プロジェクト，およびすべてのサブプロジェクトのビルド・モードに，作成したビルド・モードが追加されます。

図 2—94 ビルド・モード設定 ダイアログ（ビルド・モード追加後）



(2) ビルド・モードの変更

ビルド・モードを新規に作成したビルド・モードに変更します（「2. 17. 6 ビルド・モードを変更する」参照）。

(3) ビルド・モードの設定内容の変更

プロジェクト・ツリー でビルド・ツール・ノードを選択したのち，プロパティ パネルでビルド・オプションや定義マクロの設定を変更します。

備考 ビルド・モードの作成は，プロジェクトの変更とみなされます。プロジェクトを閉じる際に，ビルド・モードを保存するかどうかの確認を行います。

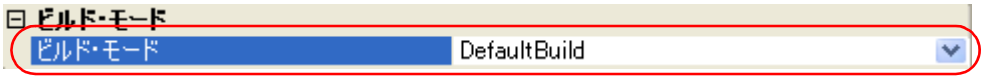
2. 17. 6 ビルド・モードを変更する

ビルドの目的に応じてビルド・オプションや定義マクロを変更したい場合，それらの設定を一括して変更することができます。ビルド・オプションや定義マクロの設定をまとめたものをビルド・モードと呼び，ビルド・モードを変更することにより，ビルド・オプションや定義マクロの設定を毎回変更する必要がなくなります。

(1) メイン・プロジェクト，またはサブプロジェクトのビルド・モードを変更する場合

プロジェクト・ツリー で対象プロジェクトのビルド・ツール・ノードを選択したのち，プロパティ パネルの［共通オプション］タブを選択します。［ビルド・モード］カテゴリの［ビルド・モード］プロパティで変更するビルド・モードを選択してください。

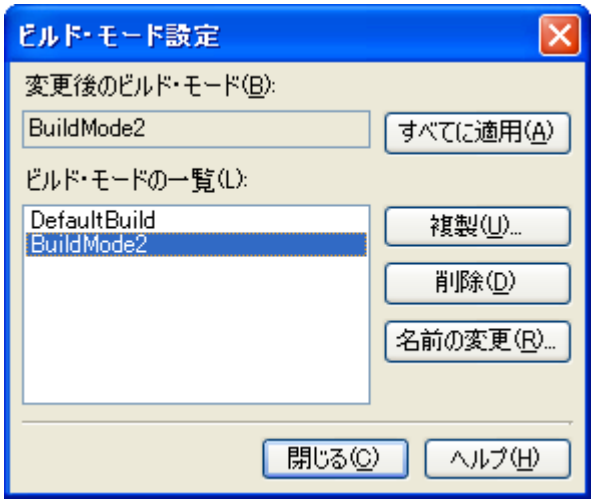
図 2—95 ［ビルド・モード］プロパティ



(2) プロジェクト全体のビルド・モードを変更する場合

［ビルド］メニュー→［ビルド・モードの設定 ...］を選択すると，ビルド・モード設定 ダイアログがオープンします。

図 2—96 ビルド・モード設定 ダイアログ



ビルド・モードの一覧から変更するビルド・モードを選択すると，［変更後のビルド・モード］に選択したビルド・モードが表示されます。［すべてに適用］ボタンをクリックすると，プロジェクトに属するメイン・プロジェクト，およびすべてのサブプロジェクトのビルド・モードを，ダイアログ上で選択したビルド・モードに変更します。

注意 選択したビルド・モードが存在しないプロジェクトについては，“DefaultBuild”を選択したビルド・モード名で複製し，複製したビルド・モードに変更します。

備考 1. ビルド・モードは，デフォルトでは“DefaultBuild”のみ用意されています。ビルド・モードの追加方法については，「2. 17. 5 ビルド・モードを追加する」を参照してください。

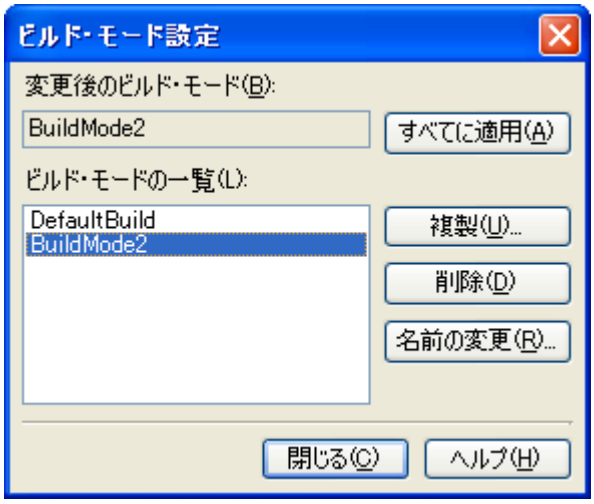
2. ビルド・モードの一覧でビルド・モードを選択したのち，［名前の変更］ボタンをクリックすることにより，ビルド・モードの名前を変更することができます。ただし，“DefaultBuild”は名前を変更することができません。

2. 17. 7 ビルド・モードを削除する

ビルド・モードの削除は，ビルド・モード設定 ダイアログで行います。

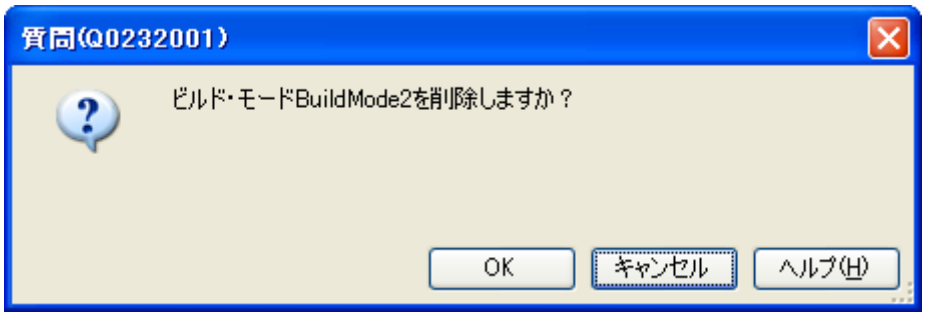
［ビルド］メニュー→［ビルド・モードの設定 ...］を選択すると，ダイアログがオープンします。

図 2—97 ビルド・モード設定 ダイアログ



ビルド・モードの一覧で削除するビルド・モードを選択したのち，［削除］ボタンをクリックすると，以下のメッセージ ダイアログがオープンします。

図 2—98 メッセージ ダイアログ



処理を継続するには，ダイアログ上で［OK］をクリックしてください。

選択したビルド・モードをプロジェクトから削除します。

注意 “DefaultBuild”を削除することはできません。

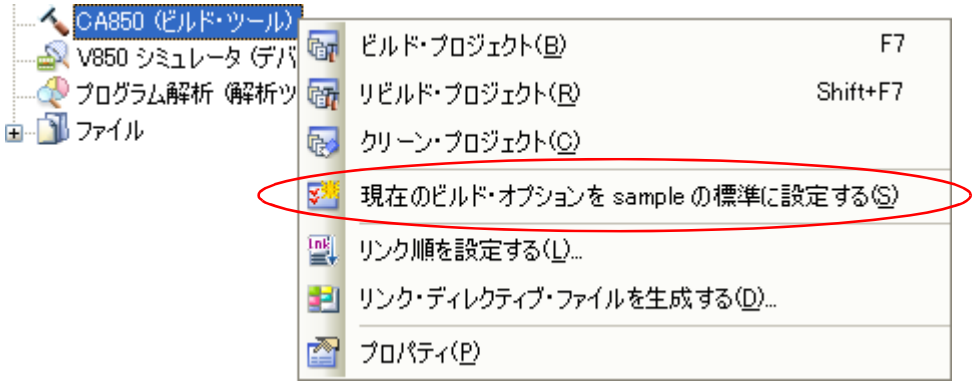
2. 17. 8 現在のビルド・オプションをプロジェクトの標準に設定する

プロパティ パネルにおいて，標準ビルド・オプションの設定に変更を加えると，プロパティの値が太字表示されます。

図 2—99 プロパティ パネル（標準ビルド・オプション変更後）

現在選択しているプロジェクト（メイン・プロジェクト，またはサブプロジェクト）のビルド・オプションを標準ビルド・オプションとする（太字表示を解除する）には，プロジェクト・ツリーでビルド・ツール・ノードを選択し，コンテキスト・メニューの［現在のビルド・オプションを*選択しているプロジェクト*の標準に設定する］を選択してください。

図 2—100 ［現在のビルド・オプションを*選択しているプロジェクト*の標準に設定する］項目



標準ビルド・オプションに設定後のプロパティの値は，以下のようになります。

図 2—101 プロパティ パネル（標準ビルド・オプション設定後）

注意 メイン・プロジェクトを選択している場合，メイン・プロジェクトの設定のみ行います。サブプロジェクトを追加していても，サブプロジェクトの設定は行いません。

2. 18 ビルドを実行する

ここでは，ビルドの実行に関する操作を説明します。

(1) ビルドの種類

ビルドには，次の種類があります。

表 2—1 ビルドの種類

種類	説明
ビルド	ビルド対象ファイルのうち，更新されたファイルのみビルドを実行します。 →「2. 18. 1 更新ファイルのビルドを実行する」参照
リビルド	ビルド対象のすべてのファイルのビルドを実行します。 →「2. 18. 2 すべてのファイルのビルドを実行する」参照
ラピッド・ビルド	ビルド設定の変更と平行してビルドを実行します。 →「2. 18. 3 他の処理と平行してビルドを実行する」参照
バッチ・ビルド	プロジェクトが持つビルド・モードを一括してビルドを実行します。 →「2. 18. 4 ビルド・モードを一括してビルドを実行する」参照

備考 1. ビルドの実行は，サブプロジェクト，メイン・プロジェクトの順で行います。
サブプロジェクトは，プロジェクト・ツリーでの表示順にビルドを行います（「2. 17. 2 サブプロジェクトのビルド順を変更する」参照）。

2. ビルド，リビルド，バッチ・ビルドを実行する際，エディタ パネルで編集中のファイルがある場合は，該当ファイルを一括して保存します。

(2) 実行結果の表示

ビルドの実行結果（ビルド・ツールの出力メッセージ）は，出力 パネルの各タブに表示されます。

- ビルド，リビルド，バッチ・ビルドの場合
  → ［すべてのメッセージ］タブ，および［ビルド・ツール］タブ
- ラピッド・ビルドの場合
  → ［ラピッド・ビルド］タブ

図 2—102 ビルドの実行結果（ビルド，リビルド，バッチ・ビルドの場合）

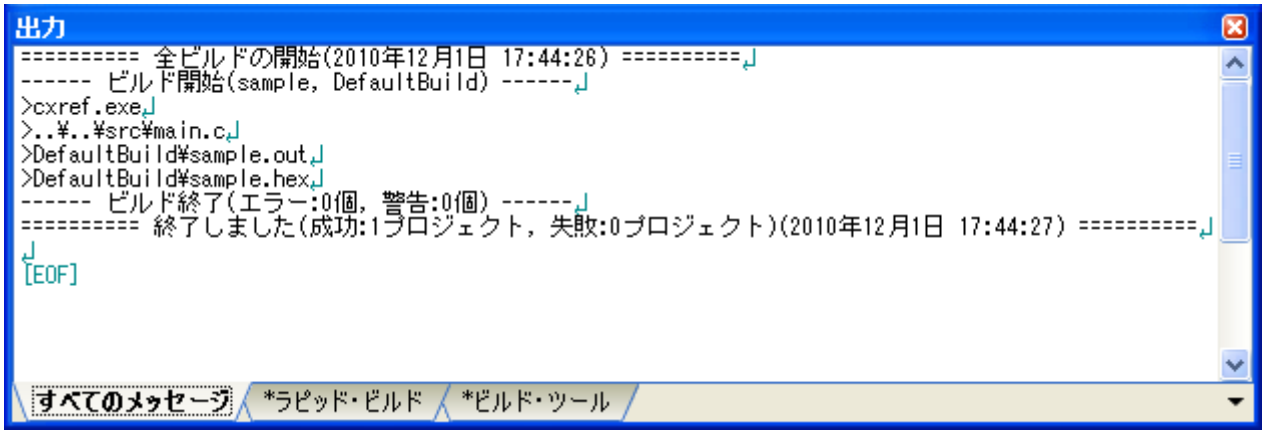
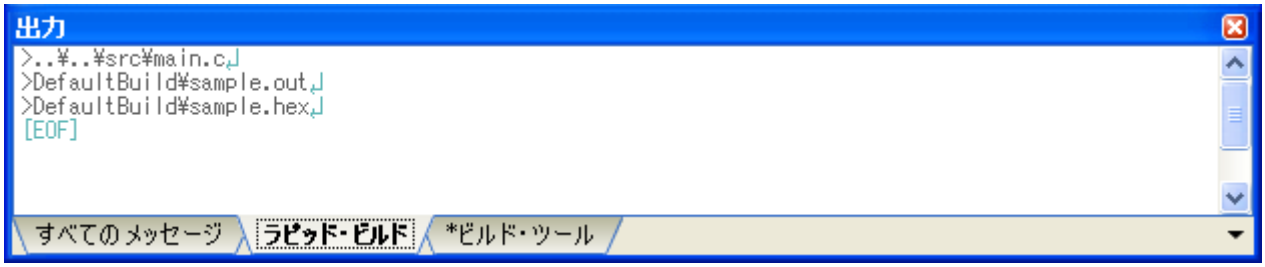


図 2—103 ビルドの実行結果（ラピッド・ビルドの場合）



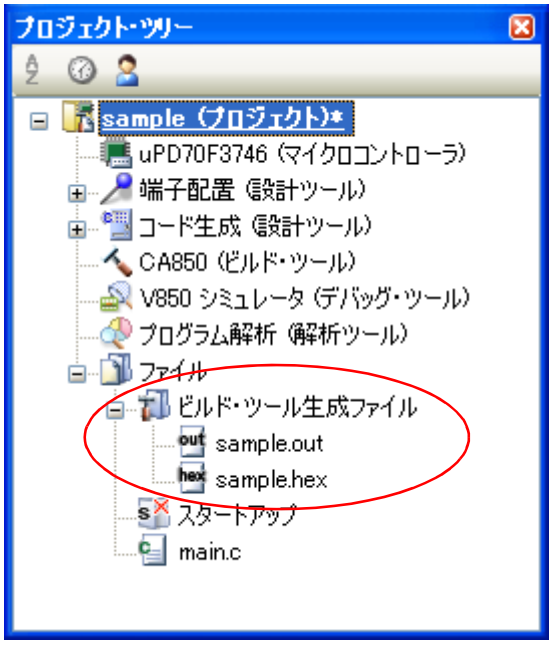
備考 1. ［ラピッド・ビルド］タブの表示文字列は，淡色表示になります。

2. 出力されたメッセージからファイル名／行番号を獲得できる場合，メッセージ上でダブルクリックすると，ファイルの該当する行へジャンプすることができます。

3. 警告メッセージ，またはエラー・メッセージを表示している行にキャレットがある状態で，［F1］キーを押下すると，その行のメッセージに関するヘルプを表示することができます。

ビルド・ツールの生成ファイルは，プロジェクト・ツリー パネルのビルド・ツール生成ファイル・ノードに表示されます。

図 2—104 ビルド・ツールの生成ファイル



備考 ビルド・ツール生成ファイル・ノードに表示されるのは，以下のファイルです。

- ライブラリ用のプロジェクト以外の場合

ロード・モジュール・ファイル（*.out）
リンク・マップ・ファイル（*.map）
ヘキサ・ファイル（*.hex）
ダンプ・リスト（dump.txt）
クロス・リファレンス情報（cxref）
タグ情報（ctags）
コール・ツリー情報（ccalltre.csv，ccalltre.lst）
関数計量情報（cmeasure.csv，cmeasure.lst）
コール・データベース情報 （cprofile.csv，cprofile.dat）
メモリ・マップ表 （rammap.csv）

- ライブラリ用のプロジェクトの場合

アーカイブ・ファイル (*.a)
ダンプ・リスト（dump.txt）
クロス・リファレンス情報（cxref）
タグ情報（ctags）
コール・ツリー情報（ccalltre.csv，ccalltre.lst）
関数計量情報（cmeasure.csv，cmeasure.lst）
コール・データベース情報 （cprofile.csv，cprofile.dat）

注意 ビルド・ツール生成ファイル・ノードは，ビルド時に作成されるノードです。
ビルド後にプロジェクトの再読み込みを行った場合，本ノードは表示されなくなります。

2. 18. 1 更新ファイルのビルドを実行する

ビルド対象ファイルのうち，更新されたファイルのみビルドを実行します（以降，“ビルド”と呼びます）。

ビルドの実行は，プロジェクト全体（メイン・プロジェクト，およびサブプロジェクト），またはアクティブ・プロジェクト（「2. 17. 4 ビルド対象プロジェクトを変更する」参照）に対して行います。

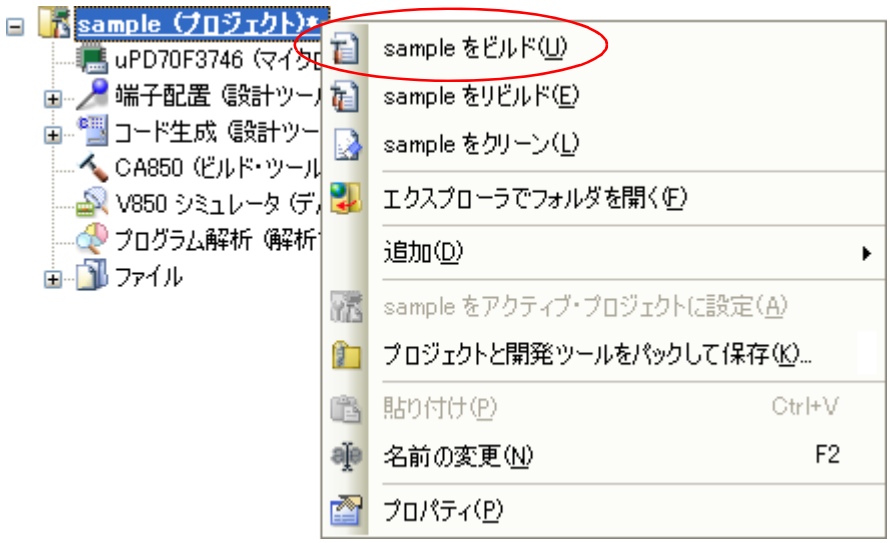
(1) プロジェクト全体のビルドを実行する場合

ツールバーの  ボタンをクリックしてください。

(2) アクティブ・プロジェクトのビルドを実行する場合

プロジェクトを選択し，コンテキスト・メニューの［*アクティブ・プロジェクト*をビルド］を選択してください。

図 2—105 ［*アクティブ・プロジェクト*をビルド］項目



備考 ヘッダ・ファイルを編集後にビルドを実行してもインクルードしているソース・ファイルがビルドされない場合は，ファイルの依存関係を更新してください（「2. 3. 8 ファイルの依存関係を更新する」参照）。

2. 18. 2　すべてのファイルのビルドを実行する

ビルド対象のすべてのファイルのビルドを実行します（以降，“リビルド”と呼びます）。

リビルドの実行は，プロジェクト全体（メイン・プロジェクト，およびサブプロジェクト），またはアクティブ・プロジェクト（「2. 17. 4　ビルド対象プロジェクトを変更する」参照）に対して行います。

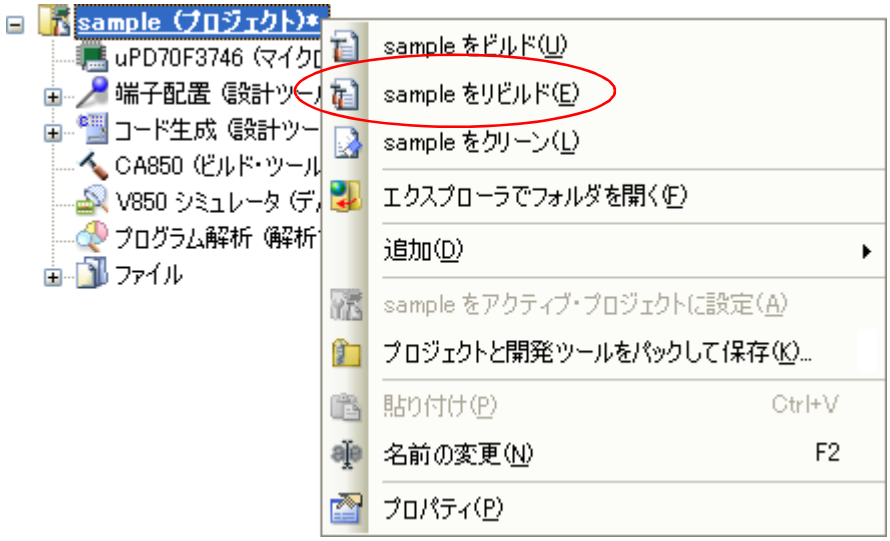
(1) プロジェクト全体のリビルドを実行する場合

ツールバーの  ボタンをクリックしてください。

(2) アクティブ・プロジェクトのリビルドを実行する場合

プロジェクトを選択し，コンテキスト・メニューの［*アクティブ・プロジェクト*をリビルド］を選択してください。

図 2—106　［*アクティブ・プロジェクト*をリビルド］項目



2. 18. 3　他の処理と平行してビルドを実行する

以下のタイミングでビルドを自動で開始する機能があります（以降，“ラピッド・ビルド”と呼びます）。

- プロジェクトに追加しているＣソース・ファイル，アセンブラ・ソース・ファイル，ヘッダ・ファイル，リンク・ディレクティブ・ファイル，セクション・ファイル，オブジェクト・モジュール・ファイル，およびライブラリ・ファイルを更新したとき
- プロジェクトにビルド対象ファイルを追加，または削除したとき
- オブジェクト・モジュール・ファイル，およびライブラリ・ファイルのリンク順を変更したとき
- ビルド・ツール，およびビルド対象ファイルのプロパティを変更したとき

（ただし，［ダンプ・オプション］タブ，［クロス・リファレンス・オプション］タブ，［メモリ・レイアウト視覚化オプション］タブのプロパティを変更した場合を除きます。）

ラピッド・ビルドを有効にすることにより，上記の操作と平行してビルドを行うことができます。

ラピッド・ビルドの有効／無効は，［ビルド］メニュー→［ラピッド・ビルド］の選択により，切り替えます。デフォルトでは，有効となっています。

図 2—107 ［ラピッド・ビルド］項目（ラピッド・ビルドが有効の場合）

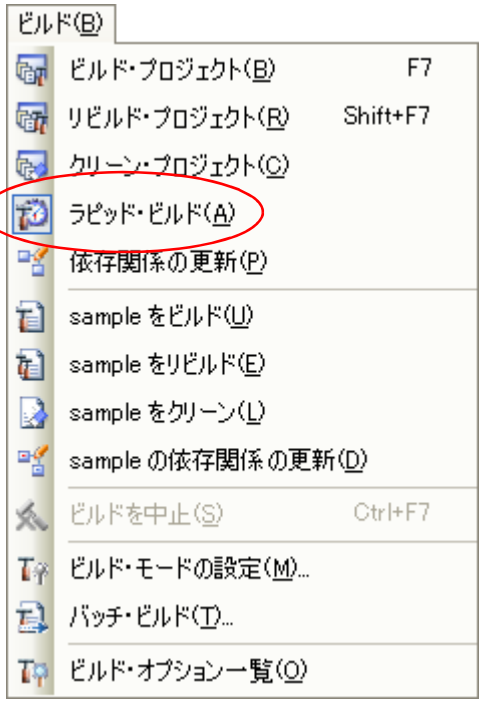
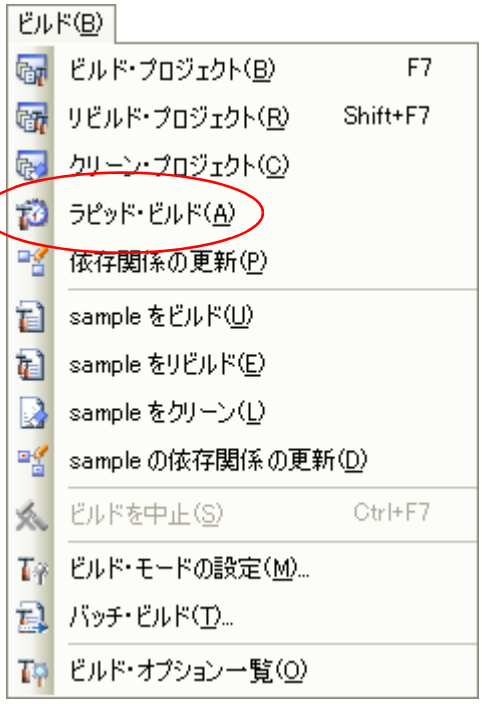


図 2—108 ［ラピッド・ビルド］項目（ラピッド・ビルドが無効の場合）



備考 1. ソース・ファイル編集後，［Ctrl］＋［S］キーの押下により，こまめに上書き保存することを推奨します。

2. ラピッド・ビルドの有効／無効は，プロジェクト全体（メイン・プロジェクト，およびサブプロジェクト）に対して設定されます。

3. ラピッド・ビルドの実行中に，ラピッド・ビルドを無効に切り替えた場合は，その場でラピッド・ビルドの実行を中止します。

注意 この機能は，ソース・ファイルの編集をエディタ パネルで行った場合のみ有効です。

2. 18. 4 ビルド・モードを一括してビルドを実行する

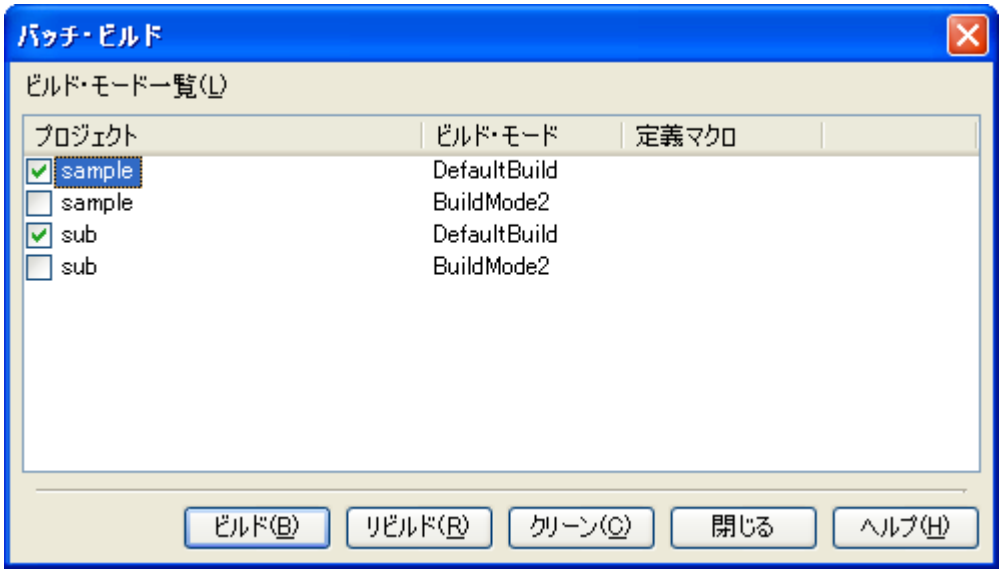
プロジェクト（メイン・プロジェクト，およびサブプロジェクト）が持つビルド・モードを一括して，ビルド／リビルド／クリーンを行うことができます（以降，“バッチ・ビルド”と呼びます）。

備考 ビルド，リビルド，クリーンについては，それぞれ以下を参照してください。

- ビルド → 「2. 18. 1 更新ファイルのビルドを実行する」参照
- リビルド → 「2. 18. 2 すべてのファイルのビルドを実行する」参照
- クリーン → 「2. 18. 8 中間ファイル，生成ファイルを削除する」参照

［ビルド］メニュー→［バッチ・ビルド ...］を選択すると，バッチ・ビルド ダイアログがオープンします。

図 2—109 バッチ・ビルド ダイアログ



ダイアログ上には，現在開いているプロジェクトが持つメイン・プロジェクト，およびサブプロジェクトの名前と，それらが持つビルド・モード，定義マクロの組み合わせの一覧が表示されます。

バッチ・ビルドを行いたいメイン・プロジェクト，およびサブプロジェクトとビルド・モードの組み合わせをチェック・ボックスにより選択し，［ビルド］／［リビルド］／［クリーン］ボタンをクリックしてください。

備考 バッチ・ビルド順は，プロジェクトのビルド順に従い，サブプロジェクト，メイン・プロジェクトの順となります。
1 つのメイン・プロジェクト，またはサブプロジェクトについて複数のビルド・モードを選択した場合は，そのサブプロジェクトで選択されているすべてのビルド・モードでビルドを行ったのち，次のサブプロジェクト，またはメイン・プロジェクトのビルドを行います。

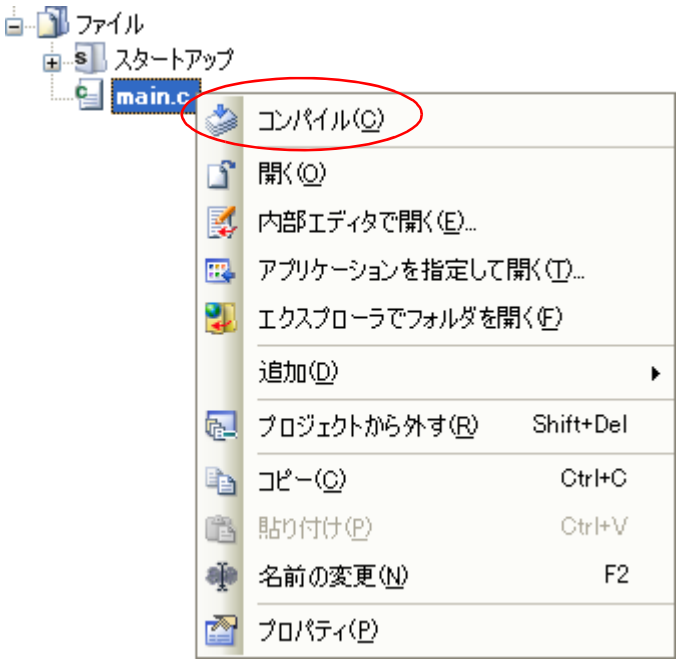
2. 18. 5 ファイル単位でコンパイル／アセンブルする

プロジェクトに追加している各ソース・ファイルに対して，コンパイル，またはアセンブルのみを行うことができます。

(1) C ソース・ファイルをコンパイルする場合

プロジェクト・ツリー で C ソース・ファイルを選択し，コンテキスト・メニューの［コンパイル］を選択してください。

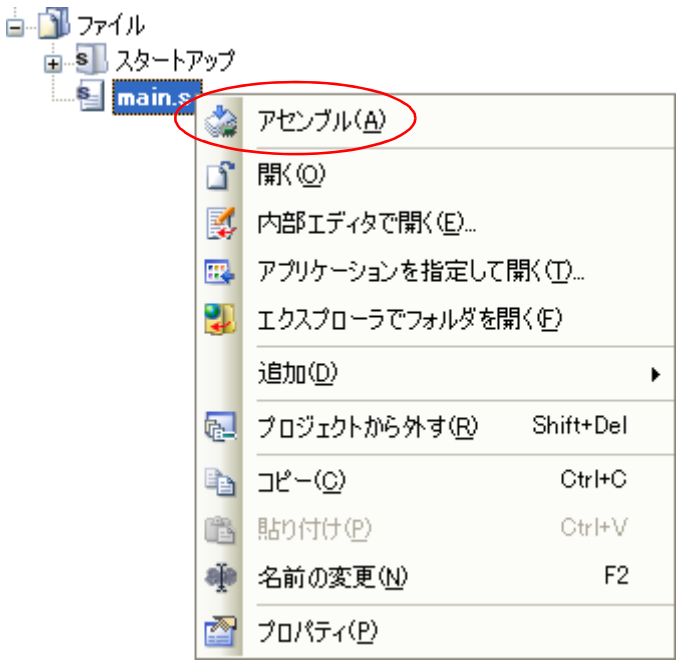
図 2—110 ［コンパイル］項目



(2) アセンブラ・ソース・ファイルをアセンブルする場合

プロジェクト・ツリー でアセンブラ・ソース・ファイルを選択し，コンテキスト・メニューの［アセンブル］を選択してください。

図 2—111 ［アセンブル］項目



2. 18. 6 ビルドの実行を中止する

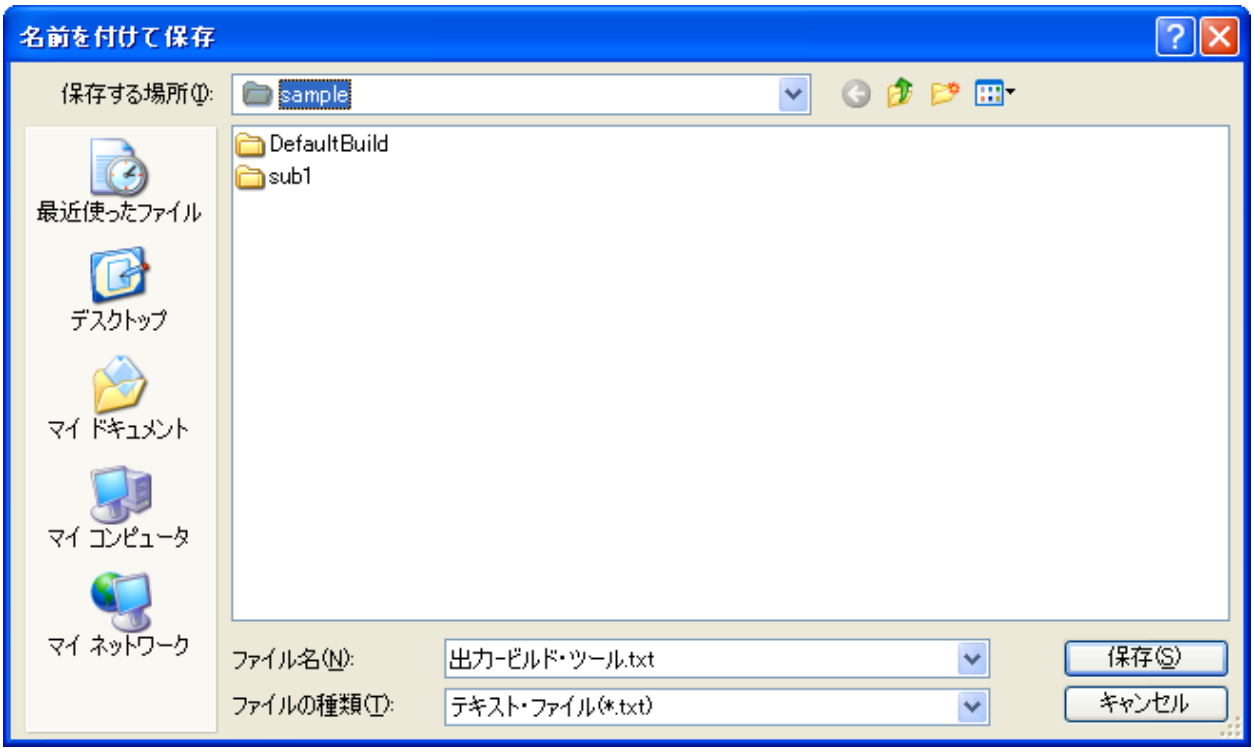
実行中のビルド，リビルド，バッチ・ビルドを中止するには，ツールバーの  ボタンをクリックしてください。

2. 18. 7 ビルド結果をファイルに保存する

出力 パネルに表示されるビルドの実行結果（ビルド・ツールの出力メッセージ）をテキスト・ファイルに保存することができます。

パネル上で［ビルド・ツール］タブを選択し，［ファイル］メニュー→［名前を付けて 出力－ビルド・ツール を保存 ...］を選択すると，名前を付けて保存 ダイアログがオープンします。

図 2—112 名前を付けて保存 ダイアログ



ダイアログ上で，保存するテキスト・ファイル名と保存場所を指定し，［保存］ボタンをクリックしてください。

2. 18. 8 中間ファイル，生成ファイルを削除する

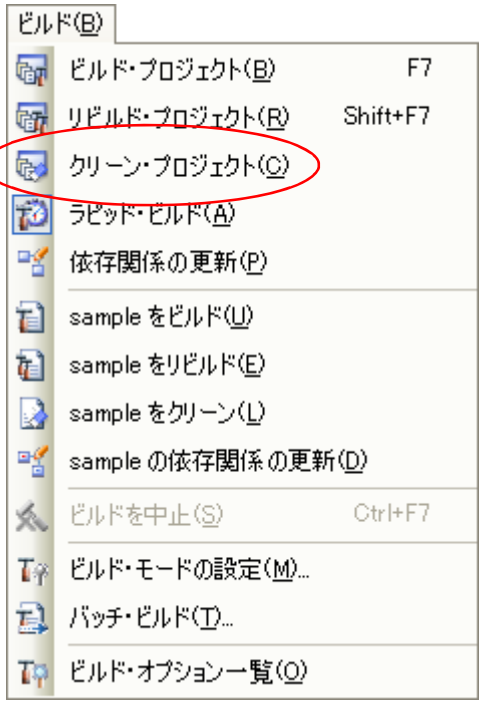
ビルドの実行により出力された中間ファイル，生成ファイルをすべて削除することができます（以降，“クリーン”と呼びます）。

クリーンの実行は，プロジェクト全体（メイン・プロジェクト，およびサブプロジェクト），またはアクティブ・プロジェクト（「2. 17. 4 ビルド対象プロジェクトを変更する」参照）に対して行います。

(1) プロジェクト全体のクリーンを実行する場合

［ビルド］メニュー→［クリーン・プロジェクト］を選択してください。

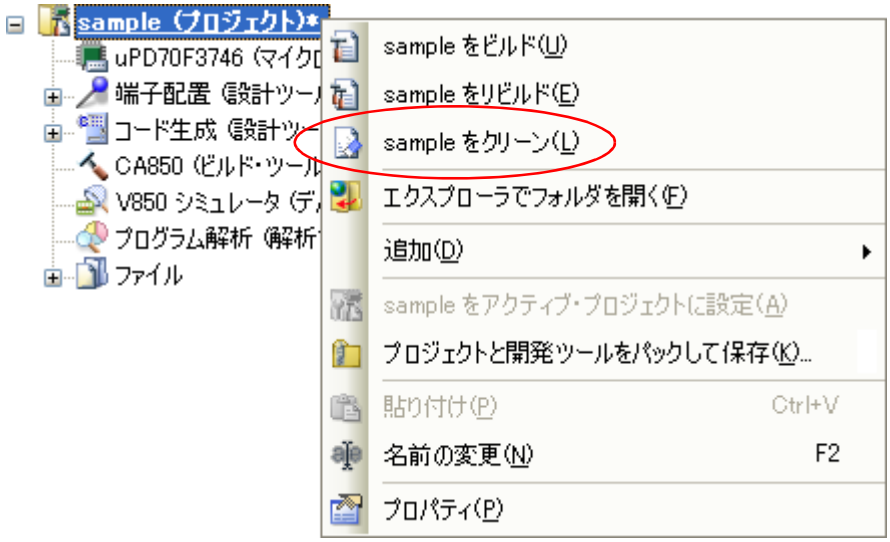
図 2—113 ［クリーン・プロジェクト］項目



(2) アクティブ・プロジェクトのクリーンを実行する場合

プロジェクトを選択し，コンテキスト・メニューの［*アクティブ・プロジェクト*をクリーン］を選択してください。

図 2—114 ［*アクティブ・プロジェクト*をクリーン］項目



2. 19 スタックを見積もる

スタックを見積もるには，スタック見積もりツールを使用します。

スタック見積もりツールでは，静的に解析処理を行うことにより，関数の呼び出し関係をツリー形式で表示するとともに，関数単位のスタック情報（関数名，スタック・サイズ，単体スタック・サイズ，加算サイズ，ファイル名）をリスト形式で表示します。

2. 19. 1 起動と終了

スタック見積もりツールの起動は，メイン・ウインドウの［ツール］メニュー→［スタック見積もりツールの起動］を選択することにより行います。

なお，スタック見積もりツールの起動が完了した際には，関数の呼び出し関係，および関数単位のスタック情報が Stack Usage Tracer ウインドウのツリー表示エリア／リスト表示エリアに表示されます。

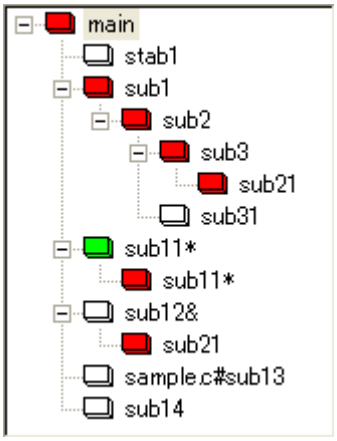
図 2—115 スタック見積もりツールの起動イメージ

また，スタック見積もりツールの終了は，Stack Usage Tracer ウインドウの［ファイル］メニュー→［sk850 の終了］を選択することにより行います。

2. 19. 2 呼び出し関係を確認する

関数の呼び出し関係については，Stack Usage Tracer ウインドウのツリー表示エリアで確認することができます。

図 2—116 ツリー表示エリア



備考 関数名の直前に表示されているアイコンは，以下の意味を持ちます。
なお，アイコンの表示優先度は，高い：  ～ 低い：  となります。

	同じ関数から直接呼び出される関数の中でスタック・サイズが最大となる関数
	スタックサイズ変更 ダイアログ，またはスタック・サイズ指定ファイルにより情報（加算サイズ，再帰回数，呼び出し関数）の変更が行われた関数
	再帰関数
	スタック見積もりツールがスタック情報を取得できていない関数
	上記以外の関数

2. 19. 3 スタック情報を確認する

関数単位のスタック情報（関数名，スタック・サイズ，単体スタック・サイズ，加算サイズ，ファイル名）については，Stack Usage Tracer ウインドウのリスト表示エリアで確認することができます。

- スタック・サイズ（呼び出し関数のスタック・サイズを含む）
- 単体スタック・サイズ（呼び出し関数のスタック・サイズを含まない）
- 加算サイズ（単体スタック・サイズに対して強制的に加算する値）

図 2—117 リスト表示エリア

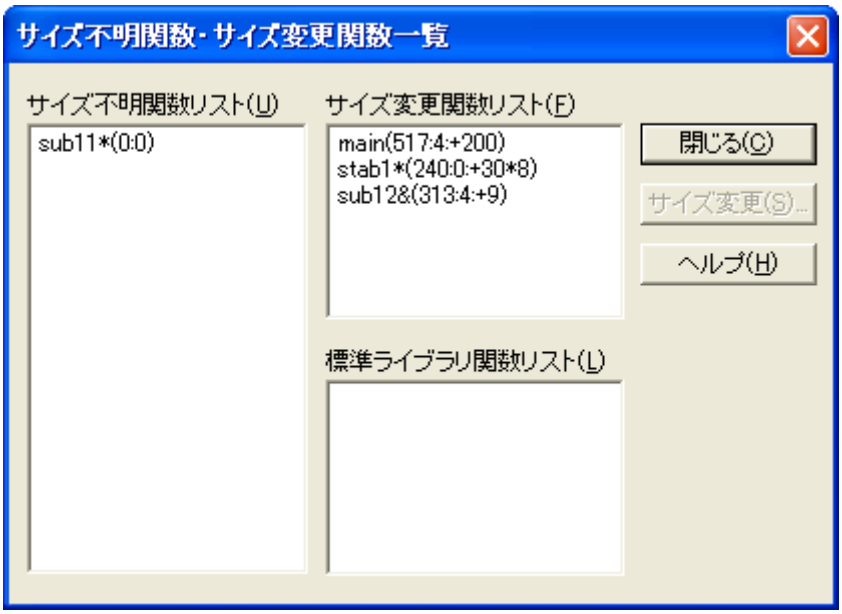
関数	スタックサイズ	単体スタックサイズ	加算サイズ	ファイル名
main	304	4		D:¥proj¥STK¥...
stab1	0	0		D:¥proj¥STK¥...
sub1	300	216		D:¥proj¥STK¥...
sub11*	0	0		D:¥proj¥STK¥...
sub12&	4	4		D:¥proj¥STK¥...
sample.c...	0	0		D:¥proj¥STK¥...
sub14	0	0		D:¥proj¥STK¥...

備考 スタック見積もりツールの起動中に，プロジェクトに登録されているファイルに対してスタック・サイズが変わるような記述変更などを行った際には，リビルド後，ボタンをクリックし，表示内容の更新を行ってください。

2. 19. 4 不明関数を確認する

スタック見積もりツールがスタック情報を取得できていない関数については，サイズ不明関数・サイズ変更関数一覧 ダイアログの［サイズ不明関数リスト］で確認することができます。

図 2—118 サイズ不明関数・サイズ変更関数一覧 ダイアログ



備考 以下の場合，［サイズ不明関数リスト］に該当関数が表示されます。

- 単体スタック・サイズを計測することができなかった関数
- スタックサイズ変更 ダイアログで再帰回数の設定が行われていない再帰関数
- スタックサイズ変更 ダイアログで呼び出し関数の設定が行われていない間接関数呼び出しを含む関数

2. 19. 5 スタック・サイズを変更する

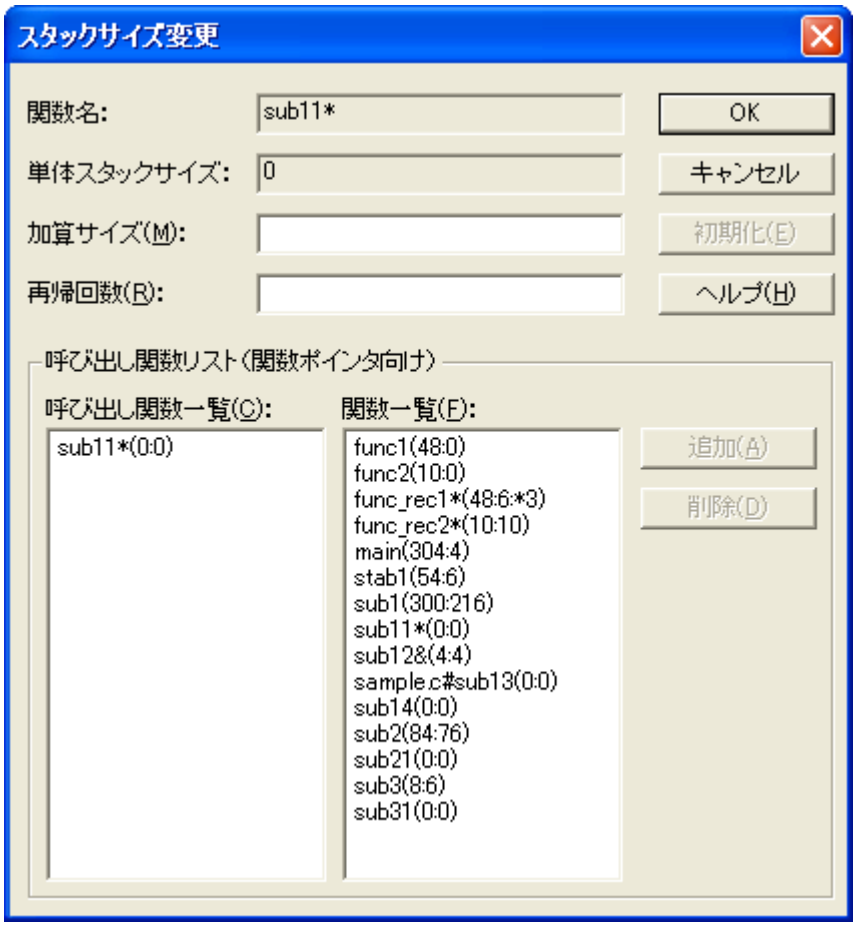
スタック見積もりツールがスタック情報を取得できていない関数，および意図的に情報を変更したい関数については，スタックサイズ変更 ダイアログ，またはスタック・サイズ指定ファイルを用いることにより，該当値を動的に設定することができます。

(1) スタックサイズ変更 ダイアログを用いる場合

スタックサイズ変更 ダイアログを用いる場合は，以下の操作手順となります。

- Stack Usage Tracer ウインドウのツリー表示エリアで該当関数を選択したのち，ツールバー→  ボタンをクリックし，スタックサイズ変更 ダイアログをオープン

図 2—119 スタックサイズ変更 ダイアログ



- ［加算サイズ］，［再帰回数］，［呼び出し関数一覧］を設定したのち，［OK］ボタンをクリック

（2）スタック・サイズ指定ファイルを用いる場合

スタック・サイズ指定ファイルを用いる場合は，以下の操作手順となります。

- スタック・サイズ指定ファイルの作成

スタック・サイズ指定ファイルでは，動的に設定したい関数を以下の形式で記述します。

関数名［, ADD= 加算サイズ］［, RECTIME= 再帰回数］［, CALL= 呼び出し関数］...

図 2—120 スタック・サイズ指定ファイルの記述イメージ

```
# アセンブリ言語で記述された関数 _flib の単体スタック・サイズを 50 に設定
[flib], ADD=50

#C 言語で記述された関数 sub2 の単体スタック・サイズを 100 に設定
sub2, ADD=100

#C 言語で記述された再帰関数 sub3 の再帰回数を 123 に設定
sub3, RECTIME=123
```

- Stack Usage Tracer ウインドウの［ファイル］メニュー→［スタックサイズ指定ファイルを開く］を選択することによりオープンするファイルを開く ダイアログで該当スタック・サイズ指定ファイルを指定したのち，［開く］ボタンをクリック

第３章　ビルドの出力リスト

この章では，ビルドにより各コマンドが出力する各種リストのフォーマットなどについて説明します。

3.1　アセンブラ

ここでは，アセンブル・リストについて説明します。アセンブル・リストとは，ソースをコンパイル，アセンブルして出力されるコードをリスト形式にしたものです。これによってコンパイル，アセンブルした結果が，どのようなコードになっているかを確認することができます。

備考　アセンブラの入出力ファイルについては，「B. 2. 1　入出力ファイル」を参照してください。

3.1.1　出力方法

アセンブル・リストの出力方法は次のとおりです。

（1）コマンド入力の場合

-a オプションを指定すると，アセンブル・リストを標準出力に出力します。-a オプションと同時に -l オプションで出力ファイル名を指定すると，そのファイルにアセンブル・リストを出力します。

C コンパイラで C ソース・ファイルのコンパイル時に「アセンブル・リストの出力」を指定し，-Xc オプションで「ソースのコメントの出力」を指定すると，アセンブル・リストには，そのコードに対応した C ソース行をコメント表示します。

ただし，最適化を強力にした場合，コード行とソース行が一致しないことがあります。

（2）CubeSuite+ の場合

プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［アセンブル・オプション］タブを選択します。アセンブル・リストを出力するには，［アセンブル・リスト］カテゴリの［アセンブル・リスト・ファイルを出力する］プロパティで［はい (-a -l)］を選択します。出力先は，［アセンブル・リスト・ファイル出力フォルダ］プロパティで指定します。リストはファイルに出力され，そのファイル名は，拡張子を .v で置き換えたものとなります。

C ソース・ファイルのコンパイル時に，［コンパイル・オプション］タブをオープンし，［出力ファイル］カテゴリの［アセンブル・リストを出力する］プロパティで［はい (-Fv)］を選択し，さらに［出力コード］カテゴリの［アセンブリ言語ソースにコメントを出力する］プロパティで［はい (-Xc)］を選択すると，アセンブル・リストには，そのコードに対応した C ソース行をコメント表示します。

ただし，最適化を強力にした場合，コード行とソース行が一致しないことがあります。

3. 1. 2 出 力 例

アセンブル・リストの出力例を示します。

例にある C ソース・ファイルをコンパイルし，さらに出力されたアセンブラ・ソース・ファイルをアセンブルすることにより出力されたアセンブル・リストの一例を次に示します。

- C ソース・ファイル

```
void main(void)
{
int     a;
}
```

- アセンブル・リストの出力

```
(1)     (2)         (3)      (4)                (5)
         :
A-X- 00000000              41             .file “c:￥work￥src￥a.c“
A-X- 00000000              42             .align 4
A-X- 00000000              43             #@BF
A-X- 00000000              44             .frame _main, .s2
A-X- 00000000              45             .globl _main
A-X- 00000000              46     _main:
A-X- 00000000              47             #@B_PROLOGUE
A-X- 00000000 D505         48             jbr     .L15
A-X- 00000002              49     .L16:
A-X- 00000002              50     .G17:
A-X- 00000002              51     .G18:
A-X- 00000002              52     .G9:
A-X- 00000002              53     .G11:
A-X- 00000002              54     .G19:
A-X- 00000002              55             #@B_EPILOGUE
A-X- 00000002 23FF0100     56             ld.w    -4+.F2[sp], lp
A-X- 00000006 441A         57             add     .S2, sp
A-X- 00000008 7F00         58             jmp     [lp]    --0
A-X- 0000000A              59             #@E_EPILOGUE
A-X- 0000000A              60     .L15:
A-X- 0000000A 5C1A         61             add     -.S2, sp
A-X- 0000000C 63FF0100     62             st.w    lp, -4+.F2[sp]
A-X- 00000010              63             #@E_EPILOGUE
A-X- 00000010 95F0         64             jbr     .L16
A-X- 00000012              65             #@FUNC_ARG
A-X- 00000012              66     .G5:
A-X- 00000012              67             .set    .S2, 0x4
A-X- 00000012              68             .set    .F2, 0x4
A-X- 00000012              69             .set    .A2, 0x0
```

```
A-X- 00000012              70            .set    .T2, 0x0
A-X- 00000012              71            .set    .P2, 0x0
A-X- 00000012              72            .set    .R2, 0x0
A-X- 00000012              73            .set    .X2, 0x0
        :
```

項番	説明
(1)	セクション属性 その行が格納されるセクションのセクション属性です。 セクション属性とその意味は次のようになります。 A：メモリを占有するセクション W：書き込み可能なセクション X：実行可能なセクション G：グローバル・ポインタ（gp）と 16 ビットのディスプレースメントを用いて参照することのできるメモリの範囲に割り付けるセクション
(2)	ロケーション・カウンタ値 コードの先頭に対するロケーション・カウンタ値です。
(3)	コード 生成されたコードです。16 進数で表記されます。
(4)	行番号 その行の行番号です。10 進数で表記されます。
(5)	ソース・プログラム その行のアセンブラ・ソース・プログラムです。その行の命令に対して命令展開が生じた場合，その命令展開によって生成された命令列を“--”以降に示します。また，その行のアセンブラ・ソース・プログラムに対する C ソース・プログラムも，このエリアに表示されます。

3.2 リンカ

ここでは，リンカの出力するリンク・マップについて説明します。

リンク・マップとは，リンク結果の情報が書かれたもので，セクションの配置アドレスなどの情報を知ることができます。

3.2.1 出力方法

リンク・マップの出力方法は次のとおりです。

（1）コマンド入力の場合

-m オプションを指定すると，リンク終了時に標準出力にリンク・マップを表示します。また，-mo オプションを指定すると，CA850 Ver.2.60 以前の旧形式での表示となります。ファイルに出力したい場合は，-m=*file* オプション，または -mo=*file* オプションでファイル名を指定して出力します。

（2）CubeSuite+ の場合

プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［リンク・オプション］タブを選択します。リンク・マップを出力するには，［リンク・マップ］カテゴリの［リンク・マッ

プ・ファイルを出力する］プロパティで［はい (-m)］を選択します。出力先は，［リンク・マップ・ファイル出力フォルダ］プロパティ，および［リンク・マップ・ファイル名］プロパティで指定します。また，プロジェクト・ツリーのビルド・ツール生成ファイル・ノードにも表示されます。

3. 2. 2　リンク・マップの出力例

次にリンク・マップの出力例を示します。
以下のオブジェクトをリンクした場合に出力されるリンクマップの一例を示します。

- オブジェクト

crtN.o
main.o
func.o
libc.a（標準ライブラリ）

- リンク・マップの出力例

```
                ********** MEMORY ALLOCATION MAP **********

(1)OUTPUT     (2)SEGMENT      (3)VIRTUAL      (4)SIZE(16)     (5)SIZE(10)
   SEGMENT      ATTRIBUTE       ADDRESS

   TEXT         RX              0x00000000      0x00000082          130
   DATA         RW              0x00000088      0x00000018           24

                ********** LINK EDITOR ALLOCATION MAP **********

(6)OUTPUT     (7)INPUT        (8)VIRTUAL      (9)SIZE        (10)INPUT
   SECTION      SECTION         ADDRESS                        FILE

   .text                        0x00000000      0x00000082
                .text           0x00000000      0x0000001a     crtN.o
                .text           0x0000001c      0x0000002c     main.o
                .text           0x00000048      0x00000018     func.o
                .text           0x00000060      0x00000022     strcmp.o(..¥lib850¥

   .sdata                       0x00000088      0x0000000e
                .sdata          0x00000088      0x0000000e     main.o

   .sbss                        0x00000098      0x00000008
                .sbss           0x00000098      0x00000004     func.o
                .sbss           0x0000009c      0x00000004     *(GpCommon)*
```

項番	説明
(1)	出力セグメント 生成されるオブジェクト・ファイルを構成する出力セグメントの名前（生成されるオブジェクト・ファイルには，出力セグメントの名前は格納されません）
(2)	セグメント属性 R：読み出し可能 W：書き込み可能 X：実行可能
(3)	アドレス 出力セグメントの先頭アドレス
(4)	サイズ（16 進数） セクション間の整列条件やアラインホールを含めたメモリのサイズ（16 進数）
(5)	サイズ（10 進数） セクション間の整列条件やアラインホールを含めたメモリのサイズ（10 進数）
(6)	出力セクション ロード・モジュールに出力されたセクション名（12 文字まで表示）
(7)	入力セクション 出力セクションを構成する入力セクション名（12 文字まで表示）
(8)	アドレス 出力セクション，または入力セクションの先頭アドレス
(9)	サイズ 出力セクション，または入力セクションのサイズ
(10)	入力ファイル 入力セクションの属しているオブジェクト・ファイル名

アセンブリ言語で .comm 疑似命令を用いて領域確保した場合，全ファイル共通の領域となり，そのセクションは“*(Common)*”，または“*(GpCommon)*”と表示されます。入力セクションの属しているオブジェクト・ファイルがアーカイブ・ファイル（ライブラリ）内のオブジェクト・ファイルである場合，

- オブジェクト・ファイル名（アーカイブ・ファイル名）

の形式で，アーカイブ・ファイルの名前も表示されます。

なお，-mo オプションを指定して，CA850 Ver.2.60 以前の旧形式での表示を指定した場合，リンカによって作られたセクション，およびアセンブラによって作られる .symtab，.strtab，および .shstrtab などのセクションに対しては *(nil)* が表示されます。

備考 *(nil)* 表示について

.sbss，sdata などのデータ領域に *(nil)* が表示されているものがあります。これは「グローバルに宣言された，初期値を持たない変数が配置されている」ことを指しています。これは，別のファイルに同名の変数があっても，最終的にロード・モジュールに集約されるため，変数の存在するファイル名が不明となってしまうため，リンク・マップ上では *(nil)* という表示になっています。

ただし，#pragma section“data”命令などを使用して，初期値なしデータを宣言した場合は，存在箇所が明らかになるため，*(nil)* ではなくファイル名が表示されます。

3. 3　ヘキサ・コンバータ

ここでは，ヘキサ・コンバータの出力ファイルの形式について説明します。

CubeSuite+ でヘキサ・ファイルを出力するには，プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［ヘキサ・コンバート・オプション］タブを選択し，［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで［はい］を選択します。出力先は，［ヘキサ・ファイル出力フォルダ］プロパティ，および［ヘキサ・ファイル名］プロパティで指定します。出力ファイルの形式は，［ヘキサ・フォーマット］カテゴリの［ヘキサ・ファイル・フォーマット］プロパティで設定します。なお，ヘキサ・ファイルは，プロジェクト・ツリーのビルド・ツール生成ファイル・ノードにも表示されます。

備考　ヘキサ・コンバータの入出力ファイルについては，「B. 5. 1　入出力ファイル」を参照してください。

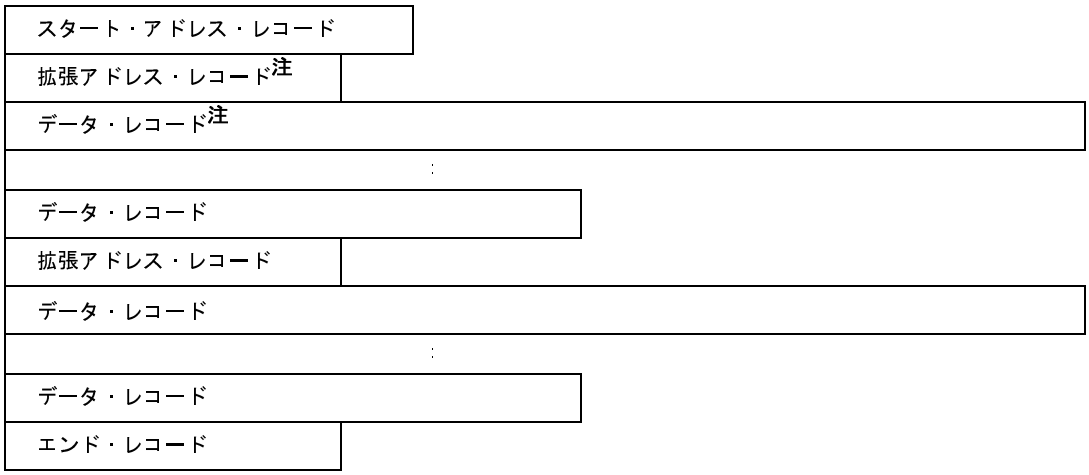
3. 3. 1　インテル拡張

インテル拡張ヘキサ・フォーマットのファイルは，スタート・アドレス・レコード，拡張アドレス・レコード，データ・レコード，およびエンド・レコードの 4 種類のレコード[注]により構成されます。

注　各レコードは，ASCII コードで出力されます。

次図にインテル拡張ヘキサ・フォーマットのファイル構成を示します。

図 3—1　インテル拡張ヘキサ・フォーマットのファイル構成



注　拡張アドレス・レコード，およびデータ・レコードは繰り返されます。

各レコードは，各種フィールドにより次の形に構成されます。

:	CC	AAAA	TT	[フィールド]...	SS	NL
(1)	(2)	(3)	(4)		(5)	(6)

項番	説明
(1)	レコード・マーク
(2)	バイト数 [フィールド]... の 2 桁ずつの 16 進数で表されるバイトのバイト数
(3)	ロケーション・アドレス
(4)	レコード・タイプ 03：スタート・アドレス・レコード，02：拡張アドレス・レコード，00：データ・レコード，01：エンド・レコード
(5)	チェック・サム ：，SS，NL を除くレコード内の 2 桁ずつの 16 進数で表される値を初期値 0 から順に減算し，その下位 1 バイトを 2 桁の 16 進数で表したもの
(6)	ニュー・ライン（¥ n）

- スタート・アドレス・レコード

エントリ・ポイント・アドレスを示します。

```
:        04       0000     03       PPPP     OOOO     SS       NL
         (1)      (2)      (3)      (4)      (5)
```

項番	説明
(1)	バイト数 04 で固定です。
(2)	0000 で固定です。
(3)	レコード・タイプ 03
(4)	エントリ・ポイント・アドレスのパラグラフ値[注]
(5)	エントリ・ポイント・アドレスのオフセット値

注　アドレスは（パラグラフ値 <<4）＋オフセット値で求められます。

- 拡張アドレス・レコード

ロード・アドレスのパラグラフ値を示します[注]。

注　（データ・レコードを出力する際）セグメントの先頭で，またはデータ・レコードのロード・アドレスのオフセット値が最大値 0xffff を越えてセグメントが新しくなる際，出力されます。

```
:        02       0000     02       PPPP     SS       NL
         (1)      (2)      (3)      (4)
```

項番	説明
(1)	バイト数 02 で固定です。
(2)	0000 で固定です。
(3)	レコード・タイプ 02
(4)	セグメントのパラグラフ値

- データ・レコード

コードの値を示します。

```
:       CC      AAAA    00      DD...DD         SS      NL
        (1)     (2)     (3)       (4)
```

項番	説明
(1)	バイト数[注]
(2)	ロケーション・アドレス
(3)	レコード・タイプ 00
(4)	コード コードの 1 バイトごとを 2 桁の 16 進数で表したもの

注 0x1 から 0xff までの範囲に限られます（1 つのデータ・レコードで示されるコードのバイト数の最小値は 1 で最大値は 255 です）。

例

```
:       04      0100    00      3C58E01B        6C      NL
        (1)     (2)     (3)       (4)           (5)
```

項番	説明
(1)	3C58E01B の 2 桁ずつの 16 進数で表されるバイトのバイト数
(2)	ロケーション・アドレス
(3)	レコード・タイプ 00
(4)	コード コードの 1 バイトごとを 2 桁の 16 進数で表したもの
(5)	チェック・サム 04 ＋ 01 ＋ 00 ＋ 00 ＋ 3C ＋ 58 ＋ E0 ＋ 1B ＝ 194 の 2 の補数 E6C の下位 1 バイトを 2 桁の 16 進数で表したもの

- エンド・レコード

コードの終わりを示します。

```
:        00       0000     01       FF       NL
         (1)       (2)      (3)      (4)
```

項番	説明
(1)	バイト数 00 で固定です。
(2)	0000 で固定です。
(3)	レコード・タイプ 01
(4)	チェック・サム FF で固定です。

備考 インテルヘキサについて

インテルヘキサ・フォーマットのロケーション・アドレスは 2 バイト（16 ビット）です。したがって，64K の空間しか直接指定はできません。それを拡張するために，16 ビットの拡張アドレスを追加して 1M（20 ビット）の空間まで扱えるようにしたのがインテル拡張ヘキサ・フォーマットです。

具体的には，16 ビットの拡張アドレスを指定するレコードタイプを追加しています。この追加された拡張アドレスの 4 ビットをシフトしてロケーション・アドレスと加算することで，20 ビットのアドレスを表現できるようになっています。

たとえば，FFFFFH を示す場合には，拡張アドレスに F000H を設定し，ロケーション・アドレスに FFFFH を指定します。

このようにインテル拡張ヘキサ・フォーマットでは 0 ～ FFFFFH までしかアドレッシングできません。100000H のような場合には別のオブジェクト形式を使用する必要があります。

ヘキサ・コンバータでは，このアドレス，サイズにおいて，このフォーマットの規定に違反していた場合，メッセージを出力します。

インテル拡張ヘキサの場合，表現可能な値が 20 ビット，つまり，1M（0x100000）バイトになります。

```
W8737 : The start address of convert area exceeds the maximum value of the address
that can be expressed in the Intel expanded hex format
```

“W8737”のメッセージが出力された場合は，ヘキサ変換する領域のスタート・アドレスが，1 M バイトを越えている場合になります。

```
W8735: The address of convert area exceeds the maximum value of the address that can
be expressed in the Intel expanded hex format
```

“W8735”のメッセージが出力された場合は，ヘキサ変換を行おうとするアドレスが 1M（20 ビット）を越えた場合になります。

次の例のような場合には，1M を越えなくても，上記エラーが発生します。

例 1. -d オプションで指定したアドレスを起点とするオフセットとしない
→ 絶対アドレスをヘキサ・フォーマットに格納する

2. 20 ビットで表現可能なアドレスの上限付近にセクションを配置
→スタート・アドレスは 20 ビットに収まっているが，セクションの途中から 20 ビットを越える

この 2 パターンに合致する場合には，変換する領域がわずかに 4 バイトであるとしても，“W8735”のメッセージ が発生します。

3. 3. 2　モトローラ S タイプ

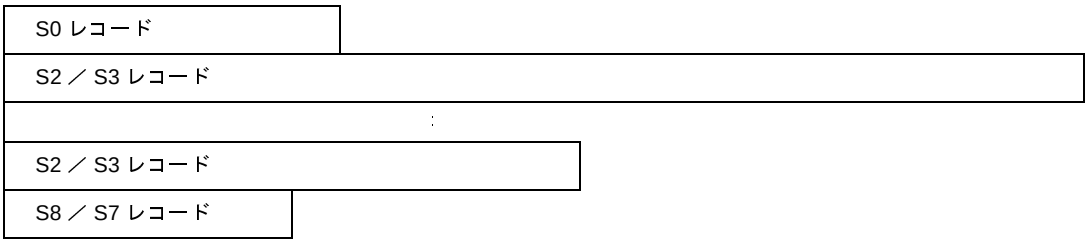
モトローラ S タイプ・ヘキサ・フォーマットのファイルは，ヘッダ・レコードである S0 レコード，データ・レコードである S2 ／ S3 レコード，エンド・レコードである S8 ／ S7 レコードの 5 種類のレコード**注 1** により構成されます**注 2**。

次図にモトローラ S タイプ・ヘキサ・フォーマットのファイル構成を示します。

注 1. 各レコードは，ASCII コードで出力されます。

2. モトローラ S タイプ・ヘキサ・フォーマットには，（24 ビット）スタンダード・アドレスのものと，32 ビット・アドレスのものとが存在し，スタンダード・アドレスのフォーマットは S0，S2，および S8 レコード，32 ビット・アドレスのフォーマットは S0，S3，および S7 レコードによって構成されます。

図 3―2　モトローラ S タイプ・ヘキサ・フォーマットのファイル構成



各レコードは，各種フィールドにより次の形に構成されます。

ST	LL	フィールド [フィールド] . . .	SS	NL
(1)	(2)		(3)	(4)

項番	説明
(1)	レコード・タイプ
(2)	レコード長 フィールド [フィールド]... の 2 桁ずつの 16 進数で表されるバイトのバイト数＋ SS で表されるバイト数**注**
(3)	チェック・サム ST，SS，NL を除くレコード内の 2 桁ずつの 16 進数で表されるバイトの値を合計したものの 1 の補数を取り，その下位 1 バイトを 2 桁の 16 進数で表したもの
(4)	ニュー・ライン（¥ n）

注 1 です。

- S0 レコード

ファイル名を示します。

```
S0      LL      FF...FF         SS      NL
(1)               (2)
```

項番	説明
(1)	レコード・タイプ S0
(2)	ファイル名 指定されたファイル名の ASCII コード表示

- S2 レコード

コードの値を示します。

```
S2      LL      AAAAAA          DD...DD         SS      NL
(1)               (2)             (3)
```

項番	説明
(1)	レコード・タイプ S2
(2)	ロード・アドレス 24 ビット**注**
(3)	コード コードの 1 バイトごとを 2 桁の 16 進数で表したもの

注 0x0 ～ 0xffffff の範囲です。

- S3 レコード

コードの値を示します。

```
S3      LL      AAAAAAAA        DD...DD         SS      NL
(1)               (2)             (3)
```

項番	説明
(1)	レコード・タイプ S3
(2)	ロード・アドレス 32 ビット**注**
(3)	コード コードの 1 バイトごとを 2 桁の 16 進数で表したもの

注 0x0 ～ 0xffffffff の範囲です。

- S7 レコード

エントリ・ポイント・アドレスを示します。

```
S7      LL      AAAAAAAA        SS      NL
(1)               (2)
```

項番	説明
(1)	レコード・タイプ S7
(2)	エントリ・ポイント・アドレス 32 ビット**注**

注 0x0 ～ 0xffffffff の範囲です。

- S8 レコード

エントリ・ポイント・アドレスを示します。

```
S8      LL      AAAAAA          SS      NL
(1)               (2)
```

項番	説明
(1)	レコード・タイプ S8
(2)	エントリ・ポイント・アドレス 24 ビット**注**

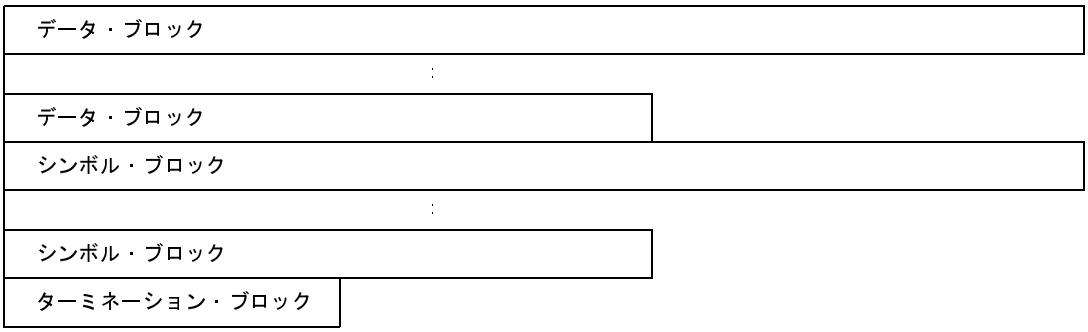
注 0x0 ～ 0xffffff の範囲です。

3. 3. 3 拡張テクトロニクス

拡張テクトロニクス・ヘキサ・フォーマットのファイルは，データ・ブロック，シンボル・ブロック，およびターミネーション・ブロックの 3 種類のブロックにより構成されます。

次図に拡張テクトロニクス・ヘキサ・フォーマットのファイル構成を示します。

図 3—3　拡張テクトロニクス・ヘキサ・フォーマットのファイル構成



各ブロックは，各種フィールドにより次の形に構成されます。

%	LL	T	SS	フィールド [フィールド]...	NL
(1)	(2)	(3)	(4)		(5)

項番	説明
(1)	ヘッダ文字
(2)	ブロック長 %，NL を除くブロック内のすべての文字の数
(3)	ブロックの種類**注 1**
(4)	チェック・サム %，SS，NL を除くブロック内のすべての文字に対する値**注 2** を合計したものを 256 で割った余りの値を，2 桁の 16 進数で表したもの
(5)	ニュー・ライン（¥n）

注 1.　6 ：データ・ブロック，3 ：シンボル・ブロック，8 ：ターミネーション・ブロックです。

2.　各文字に対する値は次のように定められています（0 ～ 9 : 0 ～ 9，A ～ Z : 10 ～ 35，$: 36，% : 37，. : 38，- : 39，a ～ z : 40 ～ 65）。

- データ・ブロック

コードの値を示します。

%	LL	T	SS	L A...A	D...D	NL
		(1)		(2)	(3)	

項番	説明
(1)	ブロックの種類
(2)	ロード・アドレスの桁数とロード・アドレス
(3)	コード コードの 1 バイトごとを 2 桁の 16 進数で表したもの

例

```
%       15      6       1C      3 100           020202020202    NL
        (1)     (2)     (3)       (4)               (5)
```

項番	説明
(1)	ブロック長
(2)	ブロックの種類
(3)	チェック・サム 1＋5＋6＋3＋1＋0＋0＋0＋2＋0＋2＋0＋2＋0＋2＋0＋2＋0＋2＝28 を 256 で割った余りの値を 2 桁の 16 進数で表したもの
(4)	ロード・アドレスの桁数は 3，ロード・アドレスは 100
(5)	コード

- シンボル・ブロック

シンボルの値を示します。

```
%       LL      T       SS      L N...N [SEDF]          SYDF [SYDF]     NL
                (1)               (2)     (3)           (4)
```

項番	説明
(1)	ブロックの種類
(2)	セクション名の文字数とセクション名
(3)	セクション定義フィールド（SEDF）**注 1**
(4)	シンボル定義フィールド（SYDF）**注 2**

注 1. セクション定義フィールドはセクションごとに 1 つ存在しなければならず，任意の数のシンボル定義フィールドの前，または後ろに続けることができます。

```
0       L B...B         L L...L
(1)       (2)             (3)
```

項番	説明
(1)	このフィールドがセクション定義フィールドであることを示す
(2)	セクションのベース・アドレスの桁数とセクションのベース・アドレス
(3)	セクションの長さの桁数とセクションの長さ

2. シンボル定義フィールド

```
T       L S...S         L V...V
(1)       (2)             (3)
```

項番	説明
(1)	シンボルの種類 1 : グローバル・アドレス（バインディング・クラス GLOBAL と ABS 以外のタイプを持つシンボル） 2 : グローバル・スカラ（バインディング・クラス GLOBAL とタイプ ABS を持つシンボル） 5 : ローカル・アドレス（バインディング・クラス LOCAL と ABS 以外のタイプを持つシンボル） 6 : ローカル・スカラ（バインディング・クラス LOCAL とタイプ ABS を持つシンボル）
(2)	シンボルの文字数とシンボル
(3)	シンボルの値の桁数とシンボルの値

例 1.

```
%   37  3   60  8SVCSTUFF   02402C6 22CR1D14OPEN25014READ25815WRITE260 NL
    (1) (2) (3)     (4)       (5)                   (6)
```

項番	説明
(1)	ブロック長
(2)	ブロックの種類
(3)	チェック・サム
(4)	セクション名の文字数は 8，セクション名は SVCSTUFF
(5)	セクション定義フィールド セクションのベース・アドレスの桁数は 2，セクションのベース・アドレスは 40，セクションの長さの桁数は 2，セクションの長さは C6
(6)	シンボル定義フィールド 22CR1D ／ 14OPEN250 ／ 14READ258 ／ 15WRITE260

2.

```
%   37  3   C8  8SVCSTUFF   15CLOSE26814EXIT27029BUFLENGTH28013BUF278 NL
    (1) (2) (3)     (4)                       (5)
```

項番	説明
(1)	ブロック長
(2)	ブロックの種類
(3)	チェック・サム
(4)	セクション名の文字数は 8，セクション名は SVCSTUFF
(5)	シンボル定義フィールド 15CLOSE268 ／ 14EXIT270 ／ 29BUFLENGTH280 ／ 13BUF278

- ターミネーション・ブロック

エントリ・ポイント・アドレスを示します。

```
%       LL      T       SS      L A...A         NL
                (1)             (2)
```

項番	説明
(1)	ブロックの種類
(2)	エントリ・ポイント・アドレスの桁数とエントリ・ポイント・アドレス

例

```
%       08      8       1A      2 80    NL
        (1)     (2)     (3)     (4)
```

項番	説明
(1)	ブロック長
(2)	ブロックの種類
(3)	チェック・サム
(4)	エントリ・ポイント・アドレスの桁数は 2，エントリ・ポイント・アドレスは 80

3.4　セクション・ファイル・ジェネレータ

セクション・ファイルは，コンパイル時に入力し，変数の配置セクションを変更するテキスト・ファイルです。これにより，C ソース・ファイルに変更を加えずに，変数の配置を変更することができます。セクション・ファイルによる配置指定は，C ソース・プログラムにおける #pragma section 指令よりも優先されます。

CubeSuite+ でセクション・ファイルを出力するには，プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［セクション・ファイル・ジェネレート・オプション］タブを選択し，［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択します。出力先は，［セクション・ファイル出力フォルダ］プロパティ，および［セクション・ファイル名］プロパティで指定します。また，プロジェクト・ツリーのファイル・ノードにも表示されます。

なお，C コンパイラでは，コンパイル時にセクション・ファイル・ジェネレータにより出力されたセクション・ファイルを指定することができます。セクション・ファイル・ジェネレータは入力された複数のファイルから情報をマージし，C コンパイラのオプションで指定する「セクション・ファイル」を 1 つ出力します。

セクション・ファイル・ジェネレータが出力するセクション・ファイルの例を以下に示します。

```
[tidata]
// [file:[func:]]variable  // section size total_freq Byte_freq Word_freq
"main.c:val1"  // data 4 10 10 0
"main.c:val2"  // data 4 8 8 0
"main.c:func1:val3" // -4 5 5 0
"i" // -4 3 3 0
"j" // -2 1 1 0
```

ファイルにおいて「//」以降はコメントとして扱われます。

セクション・ファイルにおける変数の表示は次のような形になっています。

```
[ セクション種別 ]
" ファイル名 : 関数名 : 変数名 " // コメント
" ファイル名 : 変数名 " // コメント
" 変数名 " // コメント
```

変数の表示には上のように 3 種類があり，変数の種類によります。その種類は次のようになります。

表 3—1　変数の種類と表示

表示	意味
ファイル名 : 関数名 : 変数名	関数内で宣言された静的変数。その関数名とファイル名も表示される。
ファイル名 : 変数名	ファイル内で宣言された静的変数。そのファイル名も表示される。
変数名	外部変数。変数名だけが表示される。

また，コメントは次のような形で出力されます。

```
section size total_freq Byte_freq Word_freq
```

それぞれの意味は次のようになります。

表 3—2 変数の表示とその意味

表示	意味
section	その変数の割り当てが明示的に指示されたセクション。 明示的に指示されていない変数に対しては“-”が表示されます。
size	変数のサイズ（バイト単位）。サイズが不明な場合は 0 となります。
total_freq	変数の参照頻度。その変数に対するロード／ストア命令の出現回数を意味します。
Byte_freq	変数の参照頻度中，バイト単位で参照された回数を意味します。
Word_freq	変数の参照頻度中，ワード単位で参照された回数を意味します。

セクション・ファイル・ジェネレータは，すべての変数を .tidata セクションへ配置するようなセクション・ファイルを出力します。.tidata セクションのメモリ・サイズは 256 バイトであるため，その範囲に収まらない場合は，使用する側の判断で修正する必要があります。

ただし，“-O オプション”を指定すると，利用頻度の順に，範囲に収まる分だけの変数を判断して出力するため，そのまま C コンパイラに対する入力として用いることができます。また，“-O オプション”を指定した場合，[tidata] の代わりに [tidata_word] ／ [tidata_byte] に分けて出力します。

“-O オプション”を指定したときに出力されるセクション・ファイルの例を以下に示します。

```
[tidata_byte]
// [file:[func:]]variable  // section size total_freq Byte_freq Word_freq
"a.c:si1"       // - 4 10 10 0
"a.c:si2"       // - 4 8 8 0
"a.c:f1:sfil"   // - 4 5 2 3
"j"     // - 2 2 1 1
"i"     // - 4 3 3 0
[tidata_word]
"a.c:si3"       // - 4 10 0 10
"a.c:si4"       // - 4 8 0 8
"a.c:f1:sfi2"   // - 4 5 0 5
"l"     // - 4 3 0 3
"m"     // - 2 1 0 1
```

“-O2 オプション”を指定したときに出力されるセクション・ファイルの例を以下に示します。

```
[tidata_byte]
// [file:[func:]]variable  // section size total_freq Byte_freq Word_freq
"a.c:si1"       // - 4 10 10 0
"a.c:si2"       // - 4 8 8 0
"a.c:f1:sfil"   // - 4 5 2 3
"j"     // - 2 2 1 1
"i"     // - 4 3 3 0
[tidata_word]
"a.c:si3"       // - 4 10 0 10
"a.c:si4"       // - 4 8 0 8
"a.c:f1:sfi2"   // - 4 5 0 5
"1"     // - 4 3 0 3
"m"     // - 2 1 0 1
[sidata]
"huge3" // - 30000 3 3 0
[sedata]
"huge1" // - 30000 2 2 0
[sdata]
"huge2" // - 30000 1 1 0
```

tidata 属性，tidata.word 属性，tidata.byte 属性，sidata 属性，sedata 属性，sdata 属性のセクションに限らず，他の属性のセクションもセクション・ファイルには記述することができます。

セクション種別に指定可能な文字列は次のとおりです。

表 3—3 C コンパイラで指定可能なセクション種別

種別指定文字列	割り当てられるセクション
tidata	初期値を設定したバイト・データは .tidata.byte セクション，初期値を設定したハーフ・ワード以上のデータは .tidata.word セクションに割り当てられます。 初期値を設定しないバイト・データは .tibss.byte セクション，初期値を設定しないハーフ・ワード以上のデータは .tibss.word に割り当てられます。
data	初期値を設定した場合，.data セクションに割り当てられ，設定しない場合，.bss セクションに割り当てられます。
sdata	初期値を設定した場合，.sdata セクションに割り当てられ，設定しない場合，.sbss セクションに割り当てられます。
sedata	初期値を設定した場合，.sedata セクションに割り当てられ，設定しない場合，.sebss セクションに割り当てられます。
sidata	初期値を設定した場合，.sidata セクションに割り当てられ，設定しない場合，.sibss セクションに割り当てられます。
const	.const セクションに割り当てられます。
sconst	.sconst セクションに割り当てられます。

3. 4. 1 注意事項

- セクション名を指定する [ ] 内で，セクション名の前後に空白を挿入することはできません。
  たとえば，[tidata] の場合，“tidata”の前後に空白を挿入することはできません。
- 変数名は"（ダブルクォーテーション）で囲んでください（V2.60 以前の書式も有効となります）。
- 変数は 1 行につき 1 つのみです。1 行に 2 つ以上となる修正，および 1 つの変数指定が複数行で示されるような修正はできません。
- ‘:’の前後に空白を挿入することはできません。
- ファイル名にはパスを含めることができません。
- 関数や変数の定義がヘッダ・ファイル内にある場合，セクション・ファイルにおける“ファイル名”は，ヘッダ・ファイル名ではなく，ヘッダ・ファイルをインクルードしている C ソース・ファイル名となります。
- “/* */”，または“//”形式のコメントを挿入することができます。
  ただし，セクション名や変数名をコメントで区切ることはできません。また，変数名の直後には空白が必要です。なお，コメントには，ASCII 文字，および EUC 日本語コードを使用することができます。
- セクション・ファイルでセクション種別に“data”を指定した変数を，他のアセンブラ・ソース・ファイルで参照する場合，data ／ bss 属性であることをアセンブラに通知するため，.option 疑似命令を使い“.option data”と指定して参照してください。また，“sdata”を指定した変数を，他のアセンブラ・ソース・ファイルで参照する場合，sdata ／ sbss 属性であることをアセンブラに通知するため，.option 疑似命令を使い“.option sdata”と指定して参照してください。
  記述例を次に示します。

```
// セクション・ファイル
[data]
"a.c:dat1"  // 初期値ありの場合，.data セクションとなる
"b.c:dat2"  // 初期値なしの場合，.bss セクションとなる
[sdata]
"a.c:sdat1" // 初期値ありの場合，.sdata セクションとなる
"b.c:sdat2" // 初期値なしの場合，.sbss セクションとなる

# アセンブラ・ソース・ファイル
        .option data _dat1
        .text
        ld.w    $_dat1, r11     -- .data セクションとみなし，命令展開する
        .option data _dat2
        .text
        ld.w    $_dat2, r12     -- .bss セクションとみなし，命令展開する
        .option sdata _sdat1
        .text
        ld.w    $_sdat1, r13    -- .sdata セクションとみなし，命令展開しない
        .option sdata _sdat2
        .text
        ld.w    $_sdat2, r14    -- .sbss セクションとみなし，命令展開しない
```

3.5 ダンプ・ツール

ここでは，ダンプ・ツールの出力形式について説明します。

CubeSuite+ でダンプ・ツールを使用するには，プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［ダンプ・オプション］タブを選択し，［ダンプ・ツール］カテゴリの［ダンプ・ツールを使用する］プロパティで［はい］を選択します。出力ファイル名は，“dump.txt”です。なお，プロジェクト・ツリーのビルド・ツール生成ファイル・ノードにも表示されます。

3.5.1 ダンプ・リストの表示内容

(1) アーカイブ・ヘッダ

アーカイブ・ヘッダの内容を表示します。

```
                        ***ARCHIVE HEADER***
(1)Date         (2)Uid      (3)Gid      (4)Mode     (5)Size     (6)Member Name
   0x3158DE73     0           0           0100664     0x2B8       atof.o
```

項番	説明
(1)	メンバの更新年月日
(2)	ユーザ ID
(3)	グループ ID
(4)	メンバのパーミッション
(5)	メンバの総バイト数
(6)	メンバ名

(2) アーカイブ・シンボル・テーブル

アーカイブ・シンボル・テーブルの内容を表示します。

```
                      ***ARCHIVE SYMBOL TABLE***
        (1)Offset       (2)Name
           0x1f3c          _abs
```

項番	説明
(1)	シンボルを含んでいるメンバへのファイル内オフセット
(2)	シンボル名

(3) アーカイブ・ストリング・テーブル

アーカイブ・シンボル・テーブルの内容を表示します。

```
                    ***ARCHIVE STRING TABLE***
        (1)Offset       (2)Name
           0x1100          foo.o
```

項番	説明
(1)	オフセット
(2)	メンバ名

(4) ELF ヘッダ

ELF ヘッダの内容を表示します。

```
                        ***ELF HEADER***
(1) Class       (2) Data        (3) Type        (4) Machine     (5) Version
(6) Entry       (7) Phoff       (8) Shoff       (9) Flags       (10)Ehsize
(11)Phentsize   (12)Phnum       (13)Shentsz     (14)Shnum       (15)Shstrndx
    1               1               1               070377          1
    0x0             0x0             0x2A4           0x84            0x34
    0x20            0               0x28            6               5
```

項番	説明
(1)	クラス
(2)	バイト・オーダ
(3)	種類
(4)	プロセッサ
(5)	版番号
(6)	エントリ・ポイント・アドレス
(7)	プログラム・ヘッダ・テーブルのファイル内オフセット
(8)	セクション・ヘッダ・テーブルのファイル内オフセット
(9)	フラグ
(10)	ELF ヘッダのサイズ
(11)	プログラム・ヘッダ・テーブルのエントリ・サイズ
(12)	プログラム・ヘッダ・テーブルのエントリの数
(13)	セクション・ヘッダ・テーブルのエントリのサイズ
(14)	セクション・ヘッダ・テーブルのエントリの数
(15)	セクション名を保持しているストリング・テーブルのセクション・ヘッダ・テーブル・インデックス

(5) プログラム・ヘッダ・テーブル

プログラム・ヘッダ・テーブルの内容を表示します。

```
                    ***PROGRAM HEADER***
(1)No.      (2)Type     (3)Offset   (4)Vaddr    (5)Paddr
            (6)Filesz   (7)Memsz    (8)Flags    (9)Align
  1.          0           0x0         0x0         0x0
              0x0         0x0         0x0         0x0
```

項番	説明
(1)	インデックス
(2)	セグメント・タイプ
(3)	ファイル内オフセット
(4)	仮想アドレス
(5)	物理アドレス
(6)	ファイル・サイズ
(7)	メモリ・サイズ
(8)	セグメント属性
(9)	整列条件

(6) セクション・ヘッダ・テーブル

セクション・ヘッダ・テーブルの内容を表示します。

```
                    ***SECTION HEADER***
(1)No.      (2)Type     (3)Flags    (4) Addr    (5) Offset   (6)Size     (7)Name
            (8)Link     (9)Info     (10)Adralgn (11)Entsize
  1.          0x1         0x6         0x0         0x1         0x7556      .text
              0x0         0x0         0x4         0x0
```

項番	説明
(1)	インデックス
(2)	セクション・タイプ
(3)	セクション属性
(4)	先頭アドレス
(5)	ファイル内オフセット
(6)	サイズ
(7)	セクション名
(8)	セクション・ヘッダ・テーブル・インデックス・リンク
(9)	情報
(10)	整列条件
(11)	エントリのサイズ

(7) ストリング・テーブル

ストリング・テーブルの内容を表示します。

```
                        ***STRING TABLE INFORMATION***
        (1)Index        (2)String
           0x1             .text
```

項番	説明
(1)	インデックス
(2)	文字列

(8) シンボル・テーブル

シンボル・テーブルの内容を表示します。

```
                        ***SYMBOL TABLE INFORMATION***
(1)No.   (2)Value   (3)Size    (4)Bind    (5)Type    (6)Other   (7)Shndx   (8)Name
   1.       0x0        0x0        0          3          0          0x1        .text
```

項番	説明
(1)	インデックス
(2)	値
(3)	サイズ
(4)	バインディング・クラス
(5)	タイプ
(6)	未使用
(7)	セクション・ヘッダ・テーブル・インデックス
(8)	シンボル名

(9) リロケーション情報

リロケーション情報（リロケーション・エントリの並び）の内容を表示します。

```
                        ***RELOCATION INFORMATION***
        (1)Offset       (2)Sym          (3)Type         (4)Addend
           0x20            6               0x23            0x0
```

項番	説明
(1)	オフセット
(2)	シンボル・デーブル・インデックス
(3)	リロケーション・タイプ
(4)	加数定数

（10）レジスタ・モード情報

レジスタ・モード情報の内容を表示します。

```
                        ***REGISTER MODE INFORMATION***
        (1)SymIdx       (2)TmpReg       (3)ParReg
           0x1             0x5             0x5
```

項番	説明
(1)	シンボル・テーブル・インデックス
(2)	作業用レジスタの数
(3)	レジスタ変数用レジスタの数

（11）グローバル・ポインタ・テーブル

グローバル・ポインタ・テーブルの内容を表示します。

```
                        ***GPTAB INFORMATION***
        (1)Gnum         (2)Gsize
           0x4             0xc
```

項番	説明
(1)	-G*num* で指定した値，またはシンボルの最大サイズ
(2)	0，またはセクションのサイズ

（12）ライン・ナンバ情報

ライン・ナンバ情報の内容を表示します。

```
                        ***LINE NUMBER INFORMATION***
(1)Bfunc (2)Maddr (3)Daddr (4)Pad (5)Function Name (6)Num (7)Snum (8)Offset (9)Flags
   0x0      0xA2     0xE28    0x0     _main            0x5     0x0      0x0        0x1
```

項番	説明
(1)	サブセクションの始まり
(2)	関数のアドレス
(3)	デバッグ情報のアドレス
(4)	パディング
(5)	関数名
(6)	行番号
(7)	文の位置
(8)	オフセット
(9)	フラグ

（13）デバッグ情報

デバッグ情報の内容を表示します。

```
                    ***DEBUG INFORMATION***
      (1)Tag        (2)Attr       (3)Aux
         0x0016
         size          0x00000026
                       0x000c        0x00000E1C
```

項番	説明
(1)	タグ
(2)	属性
(3)	補助情報

（14）PROGBITS データ

PROGBITS データの内容を表示します。

```
                    ***PROGBITS DATA in HEX***
0x00000000 : 40 0E 00 00 21 2E 00 00 ...
```

セクション・タイプ PROGBITS のセクションのロウ・データの内容を 16 進数で表示します。

3. 5. 2 要素の値と意味

-v オプションを指定した場合，次の情報では，一部の要素に対し，数値ではなく，その値の示す文字列で意味を表示します。

- ELF ヘッダ
- プログラム・ヘッダ・テーブル
- セクション・ヘッダ・テーブル
- シンボル・テーブル
- リロケーション情報
- デバッグ情報

次に，-v 指定で文字列表示となる要素のうち，特徴的な要素について，値[注]，-v 指定時の表示，その意味を示します。

注 ダンプ・ツールの出力する進数のままで値を示します。

（1）ELF ヘッダの「Flags」

値	-v 指定時の表示	意味
0x1	L__________	.vline セクションが存在する。
0x2	_D_________	.vdebug セクションが存在する。

値	-v 指定時の表示	意味
0x4	__P_____	PIC（Position Independent Code）オブジェクトである。
0x10	___R_____	22，または 26 レジスタ・モードである。
0x20	____d____	異なるレジスタ・モードが混在する。
0x40	_____r____	ROM 化プロセッサが出力したオブジェクトである。
0x80	______N___	デフォルトの関数呼び出し仕様である（旧仕様の呼び出しではない）。
0x100	_____M__	マスク・レジスタ機能を使用している。
0x200	_____U_	プロローグ／エピローグ・ランタイムの callt 呼び出しコードが出力される可能性がある。
0x400	_____S	プロローグ／エピローグ・ランタイムの callt 呼び出し向けに CTBP が設定されている。

（2）プログラム・ヘッダ・テーブルの「Type」

値	-v 指定時の表示	意味
1	Load	メモリにロードされるセグメント
4	Note	補助的な情報を入れたセグメント

（3）セクション・ヘッダ・テーブルの「Type」

値	-v 指定時の表示	意味
0x1	Progbits	オブジェクト・ファイル内に実際の値を持っているもの（機械語命令や初期値ありデータ）に対するセクション
0x2	Symtab	シンボル・テーブル
0x3	Strtab	ストリング・テーブル
0x4	Rela	リロケーション情報
0x8	Nobits	オブジェクト・ファイル内に実際の値を持っていないもの（初期値なしデータ）に対するセクション
0x9	Rel	リロケーション情報
0x70000000	Gptab	グローバル・ポインタ・テーブル（最初のエントリは C コンパイラ，およびアセンブラに対して指定された *-Gnum* の *num* と 0，2 番目以降のエントリはデータのサイズとワードで整列した場合のサイズ）
0x70000001	Regmode	レジスタ・モード機能を用いて生成した，リンク可能なオブジェクト・ファイルに存在するセクション（C コンパイラが内部的に使用したレジスタの本数に関する情報が格納されている）

（4）シンボル・テーブルの「Bind」

値	-v 指定時の表示	意味
0	Local	外部参照の解決において用いられることのないシンボル
1	Global	外部参照の解決において用いられるシンボル

（5）シンボル・テーブルの「Type」

値	-v 指定時の表示	意味
1	Object	通常のオブジェクト（ラベル）
2	Func	関数名
3	Section	セクション
4	File	通常のファイル名
13	Devfile	デバイス・ファイル名

（6）シンボル・テーブルの「Shndx」

値	-v 指定時の表示	意味
0x0	Undef	未定義なシンボル
0xFF00	GpCommon	グローバル・ポインタ（gp）と 16 ビットのディスプレースメントを用いて参照される未定義外部シンボル
0xFFF1	Abs	定数を示すシンボル
0xFFF2	Common	グローバル・ポインタ（gp）と 32 ビットのディスプレースメントを用いて参照される未定義外部シンボル

なお，オブジェクト・ファイルの形式については「3. 9　オブジェクト・ファイルの形式」も参考にしてください。

3. 6　ディスアセンブラ

ディスアセンブラの出力例を次に示します。

```
C:￥>dis850 -A a.out
```

```
  Address      Offset       Opecode
                              _main:
0x00000000 : 0x00000000 : 45D5           br    _main + 0x8a
0x00000002 : 0x00000002 : D800           mov   zero, r27
0x00000004 : 0x00000004 : E6230000       movea  0, sp, r28
0x00000008 : 0x00000008 : 301C           mov   r28, r6
0x0000000A : 0x0000000A : FF800176       jarl  _getToken[pc], lp
0x0000000E : 0x0000000E : 580A           mov   r10, r11
0x00000010 : 0x00000010 : 5A7F           cmp   -0x1, r11
0x00000012 : 0x00000012 : 1D92           bz    _main + 0x44
0x00000014 : 0x00000014 : EE2300E8       movea 0x3e8, zero, r29
0x00000018 : 0x00000018 : D9FD           cmp   r29, r27
0x0000001A : 0x0000001A : 15DE           bge   _main + 0x44
0x0000001C : 0x0000001C : 301C           mov   r28, r6
0x0000001E : 0x0000001E : FF800000       jarl  0[pc], lp
0x00000022 : 0x00000022 : 580A           mov   r10, r11
0x00000024 : 0x00000024 : 501B           mov   r27, r10
0x00000026 : 0x00000026 : 52C2           shl   0x2, r10
0x00000028 : 0x00000028 : 66230020       movea 0x20, sp, r12
0x0000002C : 0x0000002C : 61CA           add   r10, r12
0x0000002E : 0x0000002E : 5F6C0001       st.w  r11, 0[r12]
0x00000032 : 0x00000032 : DA41           add   0x1, r27
0x00000034 : 0x00000034 : 301C           mov   r28, r6
0x00000036 : 0x00000036 : FF80014A       jarl  _getToken[pc], lp
0x0000003A : 0x0000003A : 580A           mov   r10, r11
0x0000003C : 0x0000003C : 5A7F           cmp   -0x1, r11
0x0000003E : 0x0000003E : 05B2           bz    _main + 0x44
        :
```

a.out に含まれている情報のうち，アドレス，オフセット，命令フォーマット形式に従ったコード，タイトルをアセンブリ言語の命令とともに表示し，レジスタは別名で表示します。

3.7　クロス・リファレンス・ツール

ここでは，クロス・リファレンス・ツールの各出力形式の詳細について説明します。

CubeSuite+ でクロス・リファレンス・ツールを使用するには，プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［クロス・リファレンス・オプション］タブを選択し，［クロス・リファレンス・ツール］カテゴリの［クロス・リファレンス・ツールを使用する］プロパティで［はい］を選択します。各情報ファイルの出力先は，［共通オプション］タブの［出力ファイルの種類と場所］カテゴリの［中間ファイル出力フォルダ］プロパティで設定したフォルダです。また，プロジェクト・ツリーのビルド・ツール生成ファイル・ノードにも表示されます。

備考　クロス・リファレンス・ツールの入出力については，「B. 10. 1　入 出 力」を参照してください。

3.7.1　クロス・リファレンス

クロス・リファレンス・ツールは，ファイルごとに，そのファイル内で使用されている変数，および関数のクロス・リファレンス情報を出力します。出力先は“標準出力（デフォルト）”，または“テキスト・ファイル”です。ファイルに出力する場合，デフォルトの出力ファイル名は“cxref”です。

- クロス・リファレンス出力例

```
C:¥>cxref -x apli.c
```

```
**** apli.c
 G V NULL    20 30 43 90 91 199 204 205 235
 G F combine #163 187 190
 G F delete  #216 257
 G V deleted #22 203 220 222
        ...
 L V printtree:depth #232 236 242
 G F removeitem #118 178 209
 G F restore #182 208 212
 G V root #20 42 113 115 115 221 223 224 224 224 261
        ...
```

識別子のアルファベット順に出力されます。各行の左から順に，4 種類の情報が出力されます。

(1) リンケージと記憶クラス

次の記号で示します。

G	外部リンケージを持つ静的外部変数，関数
L	内部リンケージを持つ静的変数，関数，関数内静的変数
?	不明

(2) 種別

次の記号で示します。

F	関数
V	変数
?	不明

(3) 識別子名

関数名，または変数名そのもの。

ただし，関数内で定義された変数については，名前の重複の可能性があるため，“関数名：変数名”の形式で示します。

(4) 行番号

定義行番号，および参照行番号が，次の記号付きで列記されます。

! 行番号	宣言行
# 行番号	定義行
? 行番号	宣言・定義であるか，参照であるか不明
記号なし	参照行

3. 7. 2 タグ情報

クロス・リファレンス・ツールは，変数，および関数について，その定義ファイル名と行番号の情報（タグ・ジャンプ情報）を出力します。出力先は“標準出力（デフォルト）”，または“テキスト・ファイル”です。ファイルに出力する場合，デフォルトの出力ファイル名は“ctags”です。

- タグ情報出力例

```
C:￥>cxref -t apli.c
```

```
apli.c          163     G F combine
apli.c          216     G F delete
apli.c          22      G V deleted
apli.c          194     G F deletesub
apli.c          22      G V done
apli.c          108     G F insert
apli.c          54      G F insertitem
apli.c          86      G F insertsub
apli.c          21      G V key
       ...
```

識別子のアルファベット順に出力されます。各行の左から順に，5 種類の情報が出力されます。

(1) ファイル名

変数，または関数の定義が記述されているファイル名を示します。

(2) 行番号

変数，または関数の定義の記述位置を示します。

(3) リンケージと記憶クラス

次の記号で示します。

G	外部リンケージを持つ静的外部変数，関数
L	内部リンケージを持つ静的変数，関数，関数内静的変数
?	不明

(4) 種別

次の記号で示します。

F	関数
V	変数
?	不明

(5) 識別子名

関数名，または変数名そのもの。

ただし，関数内で定義された変数については，名前の重複の可能性があるため，“関数名：変数名”の形式で示します。

3. 7. 3　コール・ツリー

クロス・リファレンス・ツールは，-c などのコール・ツリー情報出力オプションを指定すると，“ある関数がどの関数を呼び出しているか”を，ツリー状に出力します。出力ファイルの形式は，テキスト形式，または CSV 形式です。主な情報をそのまま参照する場合はテキスト形式に，詳細な情報を表にして参照する場合は CSV 形式に出力すると便利です。

(1) テキスト形式の出力例

-c オプションを指定すると，コール・ツリーがテキスト形式で出力されます。デフォルトの出力ファイル名は“ccalltre.lst”です。

テキスト形式の出力は，次のようになります。

- コール・ツリーのテキスト形式出力例

```
C:¥>cxref -c apli.c
```

```
1       @newpage
2       |---malloc?
3       |---printf?
4       +---exit?
5       @search
6       @insertitem
7       @split
8       |---newpage...(1)
9       |---insertitem...(6)
10      +---insertitem...(6)
11      @insertsub
12      |---insertsub*
13      |---insertitem...(6)
14      +---split...(7)
       ...
```

- 処理対象の関数群が，ツリー状に出力されます。
- ツリーのルートとなる関数名の先頭には‘@’が付きます。
- ツリーには，提供ライブラリの関数も含まれます。
- 関数名の後ろに表示される記号の意味は，次のとおりです。

?	処理対象のファイル中に定義がない関数であることを示します。
...（数値）	一度出力されているため，以降の出力が省略されたことを示し，数値は，一度目の出力時の行数を示します。 定義されているソース・ファイルを示します。
*	自分自身を呼び出す再帰呼び出し関数のため，以降の出力が中止されたことを示します。

（2）CSV 形式の出力例

-cc オプションを指定すると，コール・ツリーが CSV 形式で出力されます。CSV 形式のファイルは，Microsoft Excel などの表計算ソフトで読み込み可能です。デフォルトの出力ファイル名は“ccalltre.csv”です。

CSV 形式の出力は，次のようになります。

- コール・ツリーの CSV 形式出力例

```
C:¥>cxref -cc apli.c
```

```
[SrcFileList]
 No,SrcFileName,FilePath
 1,apli.c,

 [Funcs]
 No,FuncName,SrcFileNo,LineNo,Ret1,Arg1,Ret2,Arg2
 1,free,0,0,,,,
 2,main,1,248,int,(),,
 3,scanf,0,0,,,,
 4,delete,1,217,void,(void),,
 5,search,1,38,void,(void),,
        ...
 [Calltree]
 No,FuncNo,FuncAttr,TopFlg,ElimNo,ChildPtr,ChildCnt,RefFileNo,RefLine
 1,8,0,1,0,1,3,0,0
 2,7,0x21,0,0,0,0,1,30
 3,12,0x21,0,0,0,0,1,31
 4,9,0x21,0,0,0,0,1,32
 [ChildFuncs]
 No,CalltreeNo
 1,2
 2,3
 3,4
 4,8
 5,9
 6,10
 7,12
 8,13
 9,14
10,16
        ...
```

（a）[SrcFileList]

プログラムで使用されている関数の定義されているソース・ファイル名が出力されます。

FileName	ソース・ファイル名
FilePath	ソース・ファイルのパス 入力したファイルにパスを指定した場合にのみ出力されます。

（b）[Funcs]

プログラムで使用されているすべての関数が出力されます。

FuncName	関数名

SrcFileNo	ソース・ファイル番号 [SrcFileList] の“No”の値を用いて，その関数が定義されているソース・ファイルを示します。
LineNo	行番号 ソース・ファイル中で，その関数の定義が始まる行を示します。
Ret1,Ret2	関数の戻り値 解析できなかった場合は，何も出力されません。
Arg1,Arg2	関数の引数 解析できなかった場合は，何も出力されません。

（c）[Calltree]

コール・ツリーが出力されます。

FuncNo	関数番号 [Funcs] の“No”の値を用いて，その関数を示します。
FuncAttr	関数属性 ツリーの属性を，次の数値の組み合わせで示します。属性がない場合，0 が出力されます。 0x0001 ： プログラム記述がない。 0x0002 ： 再帰関数である。 0x0004 ： 以降のツリーの出力を省略する。 0x0008 ： ソース・ファイル名，記述開始行を出力する。 0x0010 ： 戻り値，引数を出力する。 0x0020 ： 参照情報を出力する。
TopFlag	トップ・フラグ その関数がツリーのルートの場合，1 が出力され，ルートでない場合，0 が出力されます。
ElimNo	以前出力された際のツリー番号 その関数が“FuncAttr”で“0x0004”に該当する場合，その関数が以前出力されたツリーを，[Calltree] の“No”で示す。 該当しない場合，0 が出力されます。
ChildPtr	子関数表示の開始位置 [ChildFuncs] で，その関数の最初の子関数が出力される位置を，[ChildFuncs] の“No”で示します。
ChildCnt	子関数の数 [ChildFunc] に登録された子関数の数を示す。子関数がない場合，0 が出力されます。

（d）[ChildFuncs]

子関数情報として，その子関数が存在するツリーが出力されます。

CallTreeNo	ツリー番号 [Calltree] の“No”の値を用いて，その子関数が存在するツリーを示します。

3. 7. 4 関数計量

クロス・リファレンス・ツールは，-m などの関数計量情報出力オプションを指定すると，関数単位の情報を出力します。出力ファイルの形式は，テキスト形式，または CSV 形式です。主な情報をそのまま参照する場合はテキスト形式に，詳細な情報を表にして参照する場合は CSV 形式に出力すると便利です。

（1）テキスト形式の出力例

-m オプションを指定すると，関数計量がテキスト形式で出力されます。デフォルトの出力ファイル名は“cmeasure.lst”です。

テキスト形式の出力は，次のようになります。

- 関数計量のテキスト形式出力例

```
C:¥>cxref -m apli.c
```

```
                Flie            Line   Called
newpage         apli.c           27      2
search          apli.c           38      1
insertitem      apli.c           55      3
split           apli.c           68      1
insertsub       apli.c           87      2
insert          apli.c          109      1
removeitem      apli.c          119      2
moveright       apli.c          128      1
moveleft        apli.c          146      1
combin          apli.c          164      2
restore         apli.c          183      2
deletesub       apli.c          195      3
delete          apli.c          217      1
printtree       apli.c          231      3
main            apli.c          248      0
          ...
```

（a）Flie

ファイル名。その関数が定義されているソース・ファイル名を示します。

（b）Line

開始行。その関数が，ソース・ファイルで何行目に定義されているかを示します。

（c）Called

コール・ヒストグラム。その関数が呼び出される頻度を示します。関数呼び出しの記述 1 つに対し，1 回呼び出されたと想定した頻度が出力されます。

（2）CSV 形式の出力例

-mc オプションを指定すると，関数計量が CSV 形式で出力されます。CSV 形式のファイルは，Microsoft Excel などの表計算ソフトで読み込み可能です。デフォルトの出力ファイル名は“cmeasure.csv”です。

CSV 形式の出力は，次のようになります。

- 関数計量の CSV 形式出力例

```
C:¥>cxref -mc apli.c
```

```
[SrcFileList]
 No,SrcFileName,FilePath
 1,apli.c,


 [Funcs]
 No,FuncName,SrcFileNo,LineNo,Ret1,Arg1,Ret2,Arg2
 1,free,0,0,,,,
 2,main,1,248,int,(),,
 3,scanf,0,0,,,,
 4,delete,1,217,void,(void),,
 5,search,1,38,void,(void),,
        ...
 [Measure]
 No,FuncNo,FuncSz,Clk,TClk,Stk,TStk,CalledCnt,StkUp,StkUpPtr,StkUpCnt,ClkUp,
ClkUpPtr,ClkUpCnt,StkDw,StkDwPtr,StkDwCnt,ClkDw,ClkDwPtr,ClkDwCnt
 1,8,64,37,37,12,68,2,68,1,4,496,5,4,12,0,0,37,0,0
 2,5,208,118,118,12,24,1,24,9,1,237,10,1,12,0,0,118,0,0
 3,19,148,71,71,16,72,3,72,11,4,530,15,4,16,0,0,71,0,0
        ...
```

（a）[SrcFileList]

プログラムで使用されている関数の定義されているソース・ファイル名が出力されます。

FileName	ソース・ファイル名
FilePath	ソース・ファイルのパス 入力したファイルにパスを指定した場合にのみ出力されます。

（b）[Funcs]

プログラムで使用されているすべての関数が出力されます。

FuncName	関数名
SrcFileNo	ソース・ファイル番号 [SrcFileList] の“No”の値を用いて，その関数が定義されているソース・ファイルを示します。
LineNo	行番号 ソース・ファイル中で，その関数の定義が始まる行を示します。

Ret1,Ret2	関数の戻り値 解析できなかった場合は，何も出力されません。
Arg1,Arg2	関数の引数 解析できなかった場合は，何も出力されません。

（c）[Measure]

関数計量情報が出力されます。

FuncNo	関数番号 [Funcs] の“No”の値を用いて，その関数を示します。
CalledCnt	コール・ヒストグラム その関数が呼び出される頻度を示します。関数呼び出しの記述１つに対し，１回呼び出されたと想定した頻度が出力されます。

3. 7. 5 コール・データベース

クロス・リファレンス・ツールは，-b などのコール・データベース情報出力オプションを指定すると，“どの関数がどの関数を何回呼び出しているか”を出力します。出力ファイルの形式は，テキスト形式，または CSV 形式です。主な情報をそのまま参照する場合はテキスト形式に，詳細な情報を表にして参照する場合は CSV 形式に出力すると便利です。

（1）テキスト形式の出力例

-b オプションを指定すると，コール・データベースがテキスト形式で出力されます。デフォルトの出力ファイル名は“cprofile.dat”です。

テキスト形式の出力は，次のようになります。

- コール・データベースのテキスト形式出力例

```
C:¥>cxref -b apli.c
```

```
newpage,apli.c,malloc,0,1
newpage,apli.c,printf,0,1
newpage,apli.c,exit,0,1
split,apli.c,newpage,apli.c,1
split,apli.c,insertitem,apli.c,2
insertsub,apli.c,insertsub,apli.c,1
insertsub,apli.c,insertitem,apli.c,1
insertsub,apli.c,split,apli.c,1
insert,apli.c,insertsub,apli.c,1
insert,apli.c,newpage,apli.c,1
combine,apli.c,removeitem,apli.c,1
combine,apli.c,free,0,1
       ...
```

各行の左から順に，5 種類の情報が出力されます。

（a）呼び出し元の関数名

（b）呼び出し元の関数が定義されているソース・ファイル名

解析できない場合，“???”が出力されます。

（c）呼び出し先の関数名

（d）呼び出し先の関数が定義されているソース・ファイル名

ライブラリ中の関数の場合，ソース・ファイル名は不明のため，0 が出力されます。

（e）呼び出し元の関数中で，呼び出し先の関数が呼び出されている回数

（2）CSV 形式の出力例

-bc オプションを指定すると，コール・データベースが CSV 形式で出力されます。CSV 形式のファイルは，Microsoft Excel などの表計算ソフトで読み込み可能です。デフォルトの出力ファイル名は，“cprofile.csv”です。

CSV 形式の出力は，次のようになります。

- コール・データベースの CSV 形式出力例

```
C:￥>cxref -bc apli.c
```

```
[SrcFileList]
No,SrcFileName,FilePath
1,apli.c,

[Funcs]
No,FuncName,SrcFileNo,LineNo,Ret1,Arg1,Ret2,Arg2
1,free,0,0,,,,
2,main,1,248,int,(),,
3,scanf,0,0,,,,
4,delete,1,217,void,(void),,
5,search,1,38,void,(void),,
        ...
[CallDataBase]
No,FuncNo,ChildFuncNo,CallCnt
1,8,7,1
2,8,12,1
3,8,9,1
4,11,8,1
5,11,19,2
        ...
```

（a）[SrcFileList]

プログラムで使用されている関数の定義されているソース・ファイル名が出力されます。

FileName	ソース・ファイル名
FilePath	ソース・ファイルのパス -p オプション指定時にのみ出力されます。

（b）[Funcs]

プログラムで使用されているすべての関数が出力されます。

FuncName	関数名
SrcFileNo	ソース・ファイル番号 [SrcFileList] の“No”の値を用いて，その関数が定義されているソース・ファイルを示します。
LineNo	行番号 ソース・ファイル中で，その関数の定義が始まる行を示します。
Ret1,Ret2	関数の戻り値 解析できなかった場合は，何も出力されません。
Arg1,Arg2	関数の引数 解析できなかった場合は，何も出力されません。

（c）[CallDataBase]

コール・データベース情報が出力されます。

FuncNo	呼び出し元関数番号 [Funcs] の“No”の値を用いて，呼び出し元の関数を示します。
ChildFuncNo	呼び出し先関数番号 [Funcs] の“No”の値を用いて，呼び出し先の関数を示します。
CalledCnt	呼び出されている回数 呼び出し元の関数中で，呼び出し先の関数が呼び出されている回数。

3. 8　メモリ・レイアウト視覚化ツール

ここでは，メモリ・レイアウト視覚化ツールの出力形式の詳細について説明します。

CubeSuite+ でメモリ・レイアウト視覚化ツールを使用するには，プロジェクト・ツリー パネルでビルド・ツール・ノードを選択したのち，プロパティ パネルの［メモリ・レイアウト視覚化オプション］タブを選択し，［メモリ・レイアウト視覚化ツール］カテゴリの［メモリ・レイアウト視覚化ツールを使用する］プロパティで［はい］を選択します。各情報ファイルの出力先は，［共通オプション］タブの［出力ファイルの種類と場所］カテゴリの［中間ファイル出力フォルダ］プロパティで設定したフォルダです。また，プロジェクト・ツリーのビルド・ツール生成ファイル・ノードにも表示されます。

備考　メモリ・レイアウト視覚化ツールの入出力については，「B. 11. 1　入 出 力」を参照してください。

3. 8. 1　メモリ・マップ表

メモリ・レイアウト視覚化ツールは，変数名，サイズ，メモリ配置を示すメモリ・マップ表を出力します。出力先は，“標準出力”，または“ファイル”です。ファイルに出力する場合，出力ファイルの形式は，テキスト形式，または CSV 形式です。主な情報をそのまま参照する場合はテキスト形式に，詳細な情報を表にして参照する場合，CSV 形式に出力すると便利です。

- メモリ・マップ表は，1 行あたり 16 バイトです。
- 変数名は，C ソース・ファイルでの名前を“name”とすると，次の形式で表示されます。

外部変数	_name
ファイル内ローカル変数	ファイル名 @_name
関数内静的変数，文字列定数	ファイル名 @LL 数字

- サイズは，変数名の後ろに，“(10 進数表記のバイト数)”の形式で表示されます。

(1) テキスト形式の出力例

-m オプションを指定すると，メモリ・マップ表がテキスト形式で出力されます。デフォルトの出力ファイル名は“rammap.txt”です。

テキスト形式の出力は，次のようになります。

- メモリ・マップ表のテキスト形式出力例

```
C:¥>rammap -m a.out
```

```
Address    +0  +1  +2  +3  +4  +5  +6  +7  +8  +9  +A  +B  +C  +D  +E  +F
-----------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
0x00000000  |
:
0x00FFE000  |crtN.s@__argc(>|crtN.s@__argv(>|               |test.c@LL29(5)-
0x00FFE010  |-->|           |test.c@_svar(4>|_var(4)------->|_gAppName(8)---
```

```
0x00FFE020  |-------------->|_c>|           |_tmp(4)------->|_buf(100)------
:
0x00FFE080  |-------------------------------------------------------------->|
0x00FFE090  |_var2(4)------>|_c>|           |crtN.s@__stack(512)------------
:
0x00FFE290  |------------------------------>|
:
0xFFFFFFF0  |
------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:
```

- 変数名とサイズは，該当するアドレスの先頭に，左詰めで表示されます。
- メモリ配置枠に入りきらない変数名は，入るところまで表示されます。
- 変数名を１つも持たない行は，コロン（:）が出力され，省略されます。未使用領域，text 属性セクション，大きな変数の内部が，これに該当します。

（2）CSV 形式の出力例

-mc オプションを指定すると，メモリ・マップ表が CSV 形式で出力されます。CSV 形式のファイルは，Microsoft Excel などの表計算ソフトで読み込み可能です。デフォルトの出力ファイル名は“rammap.csv”です。

CSV 形式の出力は，次のようになります。

- メモリ・マップ表の CSV 形式出力例

```
C:¥>rammap -mc a.out
```

```
Address,0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
0x00000000,,,,,,,,,,,,,,,,
:
0x00FFE000,crtN.s@__argc(4),,,,crtN.s@__argv(4),,,,,,,,test.c@LL29(5),,,
0x00FFE010,,,,,test.c@_svar(4),,,,_var(4),,,,_gAppName(8),,,
0x00FFE020,,,,,_cInput(1),,,,_tmp(4),,,,_buf(100),,,
:
0x00FFE090,_var2(4),,,,_c(1),,,,crtN.s@__stack(512),,,,,,,
:
0xFFFFFFF0,,,,,,,,,,,,,,,,
...
```

- 変数名を１つも持たない行は，コロン（:）が出力され，省略されます。未使用領域，text 属性セクション，大きな変数の内部が，これに該当します。

3. 9　オブジェクト・ファイルの形式

ここでは，C コンパイラにおいて用いられるオブジェクト・ファイルの形式について説明します。

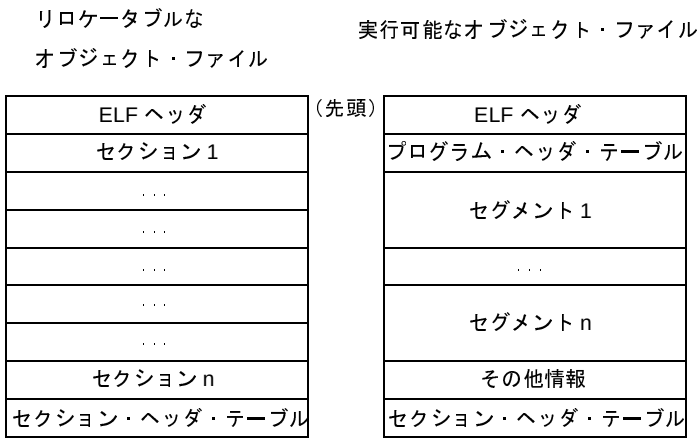
3. 9. 1　オブジェクト・ファイルの構造

C コンパイラにおいて用いられるオブジェクト・ファイルの形式は，標準的なオブジェクト・ファイルの形式の 1 つである ELF 形式に準拠しています。

この形式におけるオブジェクト・ファイルの構造は，リロケータブルなオブジェクト・ファイルと実行可能なオブジェクト・ファイルにおいて多少異なっています（次図参照）。リロケータブルなオブジェクト・ファイルは，実行可能なオブジェクト・ファイルを作成するうえで必要とされる情報を持っており，実行可能なオブジェクト・ファイルは，オブジェクト・ファイルを実行するうえで必要とされる情報を持っています。

以降，この形式のオブジェクト・ファイルの構成要素である ELF ヘッダ，プログラム・ヘッダ・テーブル，セクション・ヘッダ・テーブル，セクション，およびセグメントについて説明します。

図 3—4　オブジェクト・ファイルの構造



3. 9. 2　ELF ヘッダ

ここでは，オブジェクト・ファイル形式の構成要素である ELF ヘッダについて説明します。

ELF ヘッダは，オブジェクト・ファイルの先頭に位置し，そのオフジェクト・ファイルを解釈するための情報やそのオブジェクト・ファイルに含まれる他の構成要素をアクセスするための情報（「図 3—4　オブジェクト・ファイルの構造」参照）を持っています。

表 3—4　ELF ヘッダの構成要素とその意味

構成要素	意味
ident[CLASS]	オブジェクト・ファイルのクラス
ident[DATA]	オブジェクト・ファイル内のデータのバイト・オーダ（ビッグ・エンディアンである場合 2MSB ／リトル・エンディアンである場合 2LSB）
type	オブジェクト・ファイルの種類
machine	オブジェクト・ファイルの対象とするプロセッサ

構成要素	意味
version	オブジェクト・ファイル形式の版番号
entry	エントリ・ポイント・アドレス
phoff	プログラム・ヘッダ・テーブルのファイル内オフセット
shoff	セクション・ヘッダ・テーブルのファイル内オフセット
flags	オブジェクト・ファイルが動作するプロセッサに固有なフラグ
ehsize	本 ELF ヘッダのバイト・サイズ
phentsize	プログラム・ヘッダ・テーブルのエントリのサイズ
phnum	プログラム・ヘッダ・テーブルのエントリの数
shentsize	セクション・ヘッダ・テーブルのエントリのサイズ
shnum	セクション・ヘッダ・テーブルのエントリの数
shstrndx	セクション名を保持しているストリング・テーブル .shstrtab のセクション・ヘッダ・テーブル・インデックス

3. 9. 3 プログラム・ヘッダ・テーブル

ここでは，オブジェクト・ファイル形式の構成要素であるプログラム・ヘッダ・テーブルについて説明します。

プログラム・ヘッダ・テーブルは，そのオブジェクト・ファイルに含まれるすべてのセグメントに関する情報（次表参照）を持つプログラム・ヘッダ・テーブル・エントリの配列です。

この配列に対するインデックス（添え字）をプログラム・ヘッダ・テーブル・インデックスと呼び，プログラム・ヘッダ・テーブル・エントリの参照は，このプログラム・ヘッダ・テーブル・インデックスを用いて行われます。

表 3—5 プログラム・ヘッダ・テーブル・エントリの構成要素とその意味

構成要素	意味
type	対応するセグメントのセグメント・タイプ（メモリにロードされるセグメントである場合 LOAD ／補助的な情報を入れたセグメントである場合 NOTE）
offset	対応するセグメントのファイル内オフセット
vaddr	対応するセグメントの仮想アドレス
paddr	対応するセグメントの物理アドレス
filesz	対応するセグメントのファイル上でのサイズ**注**
memsz	対応するセグメントのメモリ上でのサイズ
flags	対応するセグメントのセグメント属性（読み出し可能なセグメントである場合 R ／書き込み可能なセグメントである場合 W ／実行可能なセグメントである場合 X）
align	対応するセグメントの整列条件

注 対応するセグメントに対し NOBITS のセクション・タイプを持つセクション（オブジェクト・ファイル内に実際の値を持たないセクション）を割り付けた場合，memsz と異なる値が設定されます。

3. 9. 4　セクション・ヘッダ・テーブル

ここでは，オブジェクト・ファイル形式の構成要素であるセクション・ヘッダ・テーブルについて説明します。

セクション・ヘッダ・テーブルは，そのオブジェクト・ファイルに含まれるすべてのセクションに関する情報を持つセクション・ヘッダ・テーブル・エントリの配列です。この配列に対するインデックス（添え字）をセクション・ヘッダ・テーブル・インデックスと呼び，セクション・ヘッダ・テーブル・エントリの参照は，このセクション・ヘッダ・テーブル・インデックスを用いて行われます。

表 3—6　セクション・ヘッダ・テーブル・エントリの構成要素とその意味

構成要素	意味
name	対応するセクションの名前（セクション名を保持しているストリング・テーブル .shstrtab に対するインデックス）
type	対応するセクションのセクション・タイプ（「(1) セクション・タイプ」参照）
flags	対応するセクションのセクション属性（メモリを占有するセクションである場合 A ／書き込み可能なセクションである場合 W ／実行可能なセクションである場合 X ／グローバル・ポインタ（gp）と 16 ビットのディスプレースメントを用いて参照することのできるメモリの範囲に割り付けるセクションである場合 G）
addr	対応するセクションの先頭アドレス
offset	対応するセクションのファイル内オフセット
size	対応するセクションのサイズ
link	対応するセクションのセクション・ヘッダ・テーブル・インデックス・リンク（「(2) セクション・タイプに依存する要素（link ／ info)」参照）
info	対応するセクションのセクション・タイプに依存する情報（「(2) セクション・タイプに依存する要素（link ／ info)」参照）
addralign	対応するセクションの整列条件
entsize	対応するセクションのエントリのサイズ

(1) セクション・タイプ

次表にセクション・ヘッダ・テーブルの構成要素 type で示されるセクション・タイプとその意味を示します。

表 3—7　セクション・タイプとその意味

セクション・タイプ	意味
GPTAB	グローバル・ポインタ・テーブル（最初のエントリは C コンパイラ，およびアセンブラに対して指定された *-Gnum* の *num* と 0，2 番目以降のエントリはデータのサイズとワードで整列した場合のサイズ）
NOBITS	オブジェクト・ファイル内に実際の値を持っていないもの（たとえば，初期値の指定されていないデータ）に対するセクション
PROGBITS	オブジェクト・ファイル内に実際の値を持っているもの（たとえば，機械語命令や初期値の指定されているデータ）に対するセクション

セクション・タイプ	意味
REGMODE	レジスタ・モード機能注を用いて生成した，リンク可能なオブジェクト・ファイルに存在するセクション（C コンパイラが内部的に使用したレジスタの本数に関する情報が格納されている）
REL（未サポート）	リロケーション情報
RELA	リロケーション情報
SYMTAB	シンボル・テーブル（「(1) シンボル・テーブル」参照）
STRTAB	ストリング・テーブル（「(2) ストリング・テーブル」参照）

注　C コンパイラのレジスタ・モード指定オプション（-reg）を参照してください。

(2) セクション・タイプに依存する要素（link ／ info）

セクション・タイプ type に依存するセクション・ヘッダ・テーブルの構成要素 link と info の意味を以下に示します。

表 3―8　link と info の意味

セクション・タイプ	link の意味	info の意味
GPTAB	---	対応するデータの割り付けられているセクションのセクション・ヘッダ・テーブル・インデックス
REL（未サポート）	対応するシンボル・テーブルのセクション・ヘッダ・テーブル・インデックス	リロケートされるセクションのセクション・ヘッダ・テーブル・インデックス
RELA	対応するシンボル・テーブルのセクション・ヘッダ・テーブル・インデックス	リロケートされるセクションのセクション・ヘッダ・テーブル・インデックス
SYMTAB	対応するストリング・テーブルのセクション・ヘッダ・テーブル・インデックス	最初に現れるローカルでないシンボルのシンボル・テーブル・インデックス

3. 9. 5　セクション

ここでは，オブジェクト・ファイル形式の構成要素であるセクションについて説明します。

セクションは，機械語命令，データ，シンボル・テーブル，ストリング・テーブル，デバッグ情報，ライン・ナンバ情報などをその内容として含むオブジェクト・ファイル形式における主要な構成要素です。

セクションは，次に示す条件を満たします。

- セクションごとにセクション・ヘッダ・テーブルに対応するセクション・ヘッダ・テーブル・エントリが 1 つ存在します。
- セクション・ヘッダ・テーブル・エントリのみが存在しオブジェト・ファイル内に実際の値を持っていないセクションが存在する場合があります（NOBITS のセクション・タイプを持つセクション）。

- オブジェクト・ファイル内に実際の値を持っているセクションは，オブジェクト・ファイル内に連続した領域を占めます。
- セクションはオブジェクト・ファイル内で領域を共有することはありません。つまり，複数のセクションに属している領域は存在しません。

(1) シンボル・テーブル

ここでは，セクションの 1 つの種類であるシンボル・テーブルについて説明します。

シンボル・テーブルは，SYMTAB のセクション・タイプを持つセクションで，そのオブジェクト・ファイルに含まれるすべてのシンボルに関する情報を持つシンボル・テーブル・エントリの配列です。

この配列に対するインデックス（添え字）をシンボル・テーブル・インデックスと呼び，シンボル・テーブル・エントリの参照は，このシンボル・テーブル・インデックスを用いて行われます注。

注 シンボル・テーブル・インデックスが 0 のエントリは予約されており，各構成要素は 0 の値を持ちます。

表 3—9 シンボル・テーブル・エントリの構成要素とその意味

構成要素	意味
name	対応するシンボルの名前（ストリング・テーブル .strtab に対するインデックス）
value	対応するシンボルの値
size	対応するシンボルのサイズ
BIND（info）	対応するシンボルのバインディング・クラス（外部参照の解決において用いられるシンボルである場合 GLOBAL ／外部参照の解決において用いられることのないシンボルである場合 LOCAL）
TYPE（info）	対応するシンボルのタイプ（通常のファイル名である場合 FILE ／関数名である場合 FUNC ／未定義なシンボルである場合 NOTYPE ／通常のラベルを示すシンボルである場合 OBJECT ／セクション名である場合 SECTION ／デバイス・ファイル名である場合 DEVFILE）
other	---
shndx	対応するシンボルに対応するセクションのセクション・ヘッダ・テーブル・インデックス（定数を示すシンボルである場合 ABS ／グローバル・ポインタ（gp）と 32 ビットのディスプレースメントを用いて参照される未定義外部シンボルである場合 COMMON ／グローバル・ポインタ（gp）と 16 ビットのディスプレースメントを用いて参照される未定義外部シンボルである場合 GPCOMMON ／未定義なシンボルである場合 UNDEF の値を取ります）

(2) ストリング・テーブル

ここでは，セクションの 1 つの種類であるストリング・テーブルについて説明します。

ストリング・テーブルは，STRTAB のセクション・タイプを持つセクションで，null 文字（¥0）が終端である文字列の並びで構成されます。この文字列は，ストリング・テーブルの先頭からのオフセットであるインデックスを用いて参照されます注。

オブジェクト・ファイル形式は，シンボルの名前やセクションの名前の保持するためにこれらの文字列を用いており，たとえば，セクション・ヘッダ・テーブル・エントリの構成要素 name は，セクション名を保持しているストリング・テーブル .shstrtab に対するインデックスを持っています。

注　インデックス 0 で表される先頭の 1 バイトは null 文字と定められています。

表 3—10　ストリング・テーブルにおけるインデックスと文字列の関係

インデックス	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0	¥0	n	a	m	e	.	¥0	V	a	r
+10	i	a	b	l	e	¥0	a	b	l	e
+20	¥0	¥0	x	x	¥0					

インデックス	文字列
0	null 文字
+1	name
+7	Variable
+11	able
+16	able
+24	null 文字

(3) 予約セクション

オブジェクト・ファイル形式では，いくつかのセクションが予約セクションとして予約されています。
次表に予約されているセクションの名前とそれらのセクション・タイプ，およびセクション属性を示します。

表 3—11　予約セクション一覧

名前[注1]	内容	セクション・タイプ	セクション属性
.bss	.bss セクション	NOBITS	AW
.const	.const セクション	PROGBITS	A
.data	.data セクション	PROGBITS	AW
.ext_info .ext_info_boot	フラッシュ／外付け ROM 再リンク機能用情報セクション	PROGBITS	なし
.ext_table	フラッシュ／外付け ROM 再リンク機能用分岐テーブル・セクション	PROGBITS	AX
.ext_tgsym	フラッシュ／外付け ROM 再リンク機能用情報セクション	PROGBITS	なし
.gptabname	グローバル・ポインタ・テーブル[注2]	GPTAB	なし
.pro_epi_runtime	プロローグ／エピローグ・ランタイム呼び出しセクション	PROGBITS	AX
.regmode	レジスタ・モード情報	REGMODE	なし

名前[注1]	内容	セクション・タイプ	セクション属性
.relname	リロケーション情報	REL	なし
.relaname	リロケーション情報	RELA	なし
.sbss	.sbss セクション	NOBITS	AWG
.sconst	.sconst セクション	PROGBITS	A
.sdata	.sdata セクション	PROGBITS	AWG
.sebss	.sebss セクション	NOBITS	AW
.sedata	.sedata セクション	PROGBITS	AW
.shstrtab	セクション名を保持しているストリング・テーブル	STRTAB	なし
.sibss	.sibss セクション	NOBITS	AW
.sidata	.sidata セクション	PROGBITS	AW
.strtab	ストリング・テーブル	STRTAB	なし
.symtab	シンボル・テーブル	SYMTAB	なし
.text	.text セクション	PROGBITS	AX
.tibss	.tibss セクション	NOBITS	AW
.tibss.byte	.tibss.byte セクション	NOBITS	AW
.tibss.word	.tibss.word セクション	NOBITS	AW
.tidata	.tidata セクション	PROGBITS	AW
.tidata.byte	.tidata.byte セクション	PROGBITS	AW
.tidata.word	.tidata.word セクション	PROGBITS	AW
.vdbstrtab	デバッグ情報用シンボル・テーブル	STRTAB	なし
.vdebug	デバッグ情報	PROGBITS	なし
.version	バージョン情報セクション	PROGBITS	なし
.vline	ライン・ナンバ情報	PROGBITS	なし

注 1. gptabname，.relname，および .relaname の name の部分は，それぞれそのセクションに対応するセクションの名前を示します。

2. リンカにおける -A オプションの処理において用いられる情報です。

付録 A　ウインドウ・リファレンス

ここでは，ビルドに関するウインドウ／パネル／ダイアログについての詳細を説明します。

A. 1　説　　明

以下に，ビルドに関するウインドウ／パネル／ダイアログの一覧を示します。

表 A—1　ウインドウ／パネル／ダイアログ一覧

ウインドウ／パネル／ダイアログ名	機能概要
メイン・ウインドウ	CubeSuite+ を起動した際，最初にオープンするウインドウ
プロジェクト・ツリー パネル	プロジェクトの構成要素をツリー表示
プロパティ パネル	プロジェクト・ツリー パネルで選択しているビルド・ツール・ノード，ファイル，カテゴリ・ノードについて，詳細情報を表示，および設定を変更
エディタ パネル	テキスト・ファイル／ソース・ファイルを表示／編集
出力 パネル	ビルド・ツールから出力するメッセージを表示
ファイル追加 ダイアログ	新規ファイルを作成，およびプロジェクトに追加
フォルダとファイル追加 ダイアログ	既存のファイルとフォルダ構成をプロジェクトに追加
文字列入力 ダイアログ	1 行分の文字列を入力，編集
テキスト編集 ダイアログ	複数行のテキストを入力，編集
パス編集 ダイアログ	パス，またはパスを含むファイル名を編集，追加
システム・インクルード・パス順設定 ダイアログ	コンパイラに対して指定するシステム・インクルード・パスを参照，および指定順を設定
ビルド時の警告メッセージ設定 ダイアログ	ビルド・ツールが出力する警告メッセージを設定
ファイルの保存設定 ダイアログ	エディタ パネルで編集中のファイルのエンコードと改行コードを設定
リンク・ディレクティブ生成 ダイアログ	リンク・ディレクティブ・ファイルを生成
オブジェクト・ファイル指定 ダイアログ	本ダイアログの呼び出し元に設定するオブジェクト・ファイルを選択
セグメント指定 ダイアログ	本ダイアログの呼び出し元に設定するセグメントを選択
リンク順設定 ダイアログ	リンカに入力するファイルを参照，およびリンク順を設定
ビルド・モード設定 ダイアログ	ビルド・モードの追加と削除，および現在のビルド・モードを一括設定
バッチ・ビルド ダイアログ	プロジェクトが持つビルド・モードを一括して，ビルド，リビルド，クリーンを実行
指定位置へ移動 ダイアログ	指定した位置にキャレットを移動
処理中表示 ダイアログ	処理の進捗状況を表示
オプション ダイアログ	各種環境を設定

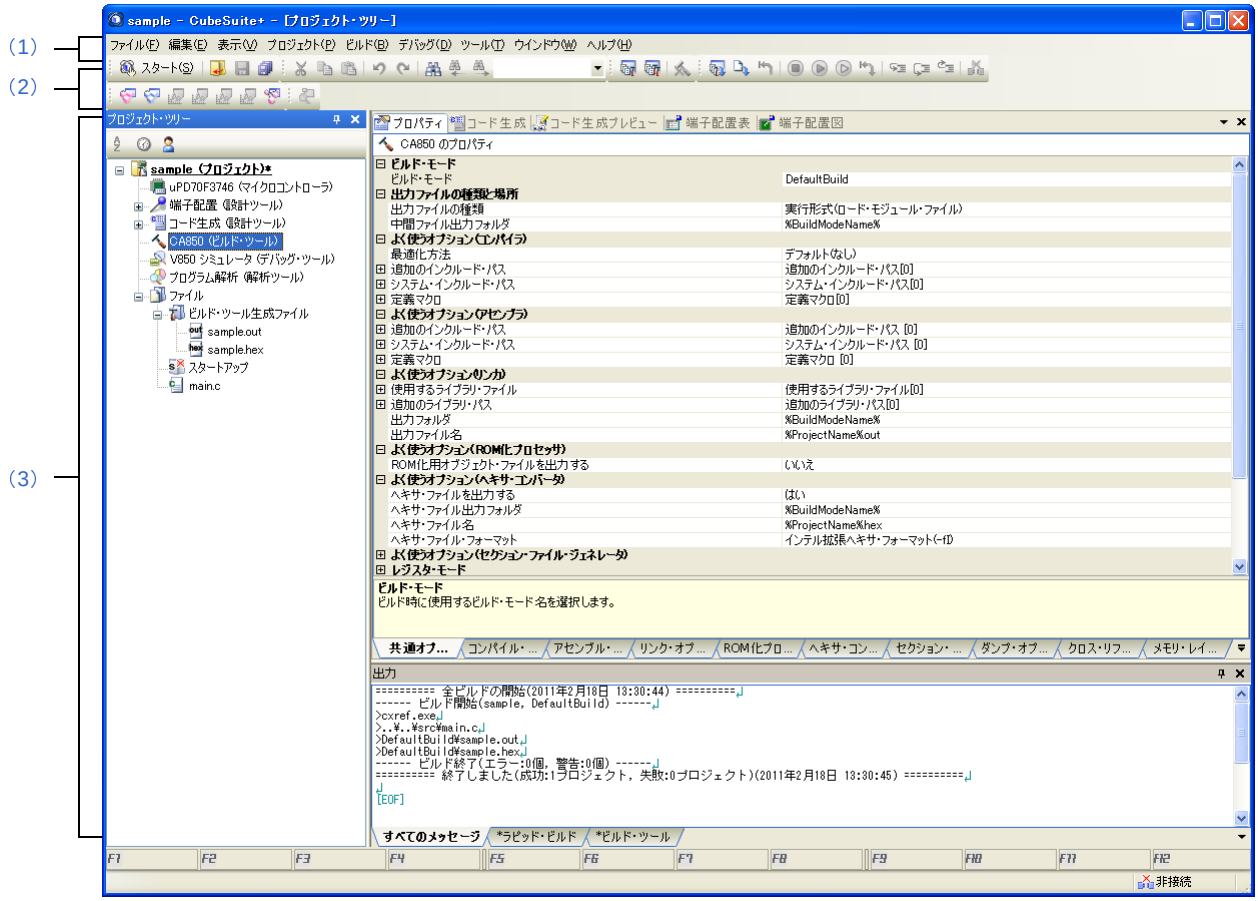
ウインドウ／パネル／ダイアログ名	機能概要
既存のファイルを追加 ダイアログ	プロジェクトに既存のファイルを追加
フォルダの参照 ダイアログ	本ダイアログの呼び出し元に設定するフォルダを選択
ブート領域オブジェクト・ファイルを指定 ダイアログ	本ダイアログの呼び出し元に設定するブート領域オブジェクト・ファイルを選択
関数情報ファイルを指定 ダイアログ	本ダイアログの呼び出し元に設定する関数情報ファイルを選択
外部変数ソート用中間言語ファイルを指定 ダイアログ	本ダイアログの呼び出し元に設定する外部変数ソート用中間言語ファイルを選択
Far Jump ファイルを指定 ダイアログ	本ダイアログの呼び出し元に設定する Far Jump ファイルを選択
ROM 化用領域確保コード・ファイルを指定 ダイアログ	本ダイアログの呼び出し元に設定する ROM 化用領域確保コード・ファイルを選択
名前を付けて保存 ダイアログ	編集中のファイル，または各パネルの内容をファイルに保存
プログラムから開く ダイアログ	ファイルを開くアプリケーションを選択
Stack Usage Tracer ウインドウ	スタック見積もりツールを起動した際，最初にオープンするウインドウ
サイズ不明関数・サイズ変更関数一覧 ダイアログ	スタック見積もりツールがスタック情報を取得できていない関数，意図的に情報の変更が行われた関数，およびスタック見積もりツールが強制的に加算サイズの設定を行った関数を一覧表示
スタックサイズ変更 ダイアログ	選択関数に対する情報を変更
ファイルを開く ダイアログ	既存のスタック・サイズ指定ファイルをオープン

メイン・ウインドウ

CubeSuite+ を起動した際，最初にオープンするウインドウです。

ビルドを行う際は，本ウインドウからユーザ・プログラムの実行制御，および各パネルのオープンを行います。

図 A―1　メイン・ウインドウ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- Windows の［スタート］→［すべてのプログラム］→［Renesas Electronics CubeSuite+］→［CubeSuite+］を選択

［各エリアの説明］

(1) メニューバー

ビルド関連のメニューを示します。

(a) ［プロジェクト］

［プロジェクト］メニューでは，プロジェクト関連を操作するメニュー項目を表示します。

新しいプロジェクトを作成 ...	現在のプロジェクトを閉じて，新しいプロジェクトを作成するために，プロジェクト作成 ダイアログをオープンします。 開いているプロジェクト，またはファイルを変更し，保存していない場合は，それらを保存するかどうかの確認を行います。
プロジェクトを開く ...	現在のプロジェクトを閉じて，既存のプロジェクトを開くために，プロジェクトを開く ダイアログをオープンします。 開いているプロジェクト，またはファイルを変更し，保存していない場合は，それらを保存するかどうかの確認を行います。
お気に入りのプロジェクト	お気に入りのプロジェクトを開く，または登録するためのカスケード・メニューを表示します。
1 パス	［お気に入りのプロジェクト］→［1 お気に入りのプロジェクトに登録］で登録したプロジェクトを開きます。 プロジェクトを登録していない場合は，“お気に入りのプロジェクト”が表示されます。
2 パス	［お気に入りのプロジェクト］→［2 お気に入りのプロジェクトに登録］で登録したプロジェクトを開きます。 プロジェクトを登録していない場合は，“お気に入りのプロジェクト”が表示されます。
3 パス	［お気に入りのプロジェクト］→［3 お気に入りのプロジェクトに登録］で登録したプロジェクトを開きます。 プロジェクトを登録していない場合は，“お気に入りのプロジェクト”が表示されます。
4 パス	［お気に入りのプロジェクト］→［4 お気に入りのプロジェクトに登録］で登録したプロジェクトを開きます。 プロジェクトを登録していない場合は，“お気に入りのプロジェクト”が表示されます。
1 お気に入りのプロジェクトに登録	現在開いているプロジェクトのパスを［お気に入りのプロジェクト］→［1 パス］に登録します。
2 お気に入りのプロジェクトに登録	現在開いているプロジェクトのパスを［お気に入りのプロジェクト］→［2 パス］に登録します。
3 お気に入りのプロジェクトに登録	現在開いているプロジェクトのパスを［お気に入りのプロジェクト］→［3 パス］に登録します。
4 お気に入りのプロジェクトに登録	現在開いているプロジェクトのパスを［お気に入りのプロジェクト］→［4 パス］に登録します。

追加	プロジェクトにサブプロジェクトを追加するためのカスケード・メニューを表示します。
既存のサブプロジェクトを追加 ...	プロジェクトに既存のサブプロジェクトを追加するために，既存のサブプロジェクトを追加 ダイアログをオープンします。
新しいサブプロジェクトを追加 ...	プロジェクトに新しいサブプロジェクトを追加するために，プロジェクト作成 ダイアログをオープンします。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	ファイル・ノードの直下にカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
*選択しているプロジェクト，またはサブプロジェクト名*をアクティブ・プロジェクトに設定	選択しているプロジェクト，またはサブプロジェクトをアクティブ・プロジェクトに設定します。
プロジェクトを閉じる	現在開いているプロジェクトを閉じます。 開いているプロジェクト，またはファイルを変更し，保存していない場合は，それらを保存するかどうかの確認を行います。
プロジェクトを保存	現在開いているプロジェクトの設定情報をプロジェクト・ファイルに保存します。
名前を付けてプロジェクトを保存 ...	現在開いているプロジェクトの設定情報を別名のプロジェクト・ファイルに保存するために，名前を付けてプロジェクトを保存 ダイアログをオープンします。
プロジェクトから外す	選択しているサブプロジェクト，またはファイルをプロジェクトから外します。 サブプロジェクト・ファイル，およびファイル自体はファイル・システム上からは削除されません。
プロジェクトと開発ツールをパックして保存 ...	本製品一式とプロジェクト一式を指定したフォルダにコピーして，一つのフォルダにまとめて保存します。

(b) ［ビルド］

［ビルド］メニューでは，ビルド関連を操作するメニュー項目を表示します。

ビルド・プロジェクト	プロジェクトのビルドを行います。サブプロジェクトを追加している場合は，サブプロジェクトのビルドも行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。

リビルド・プロジェクト	プロジェクトのリビルドを行います。サブプロジェクトを追加している場合は，サブプロジェクトのリビルドも行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
クリーン・プロジェクト	プロジェクトのクリーンを行います。サブプロジェクトを追加している場合は，サブプロジェクトのクリーンも行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
ラピッド・ビルド	ラピッド・ビルド機能の有効（デフォルト）／無効を選択します（トグル）。
依存関係の更新	プロジェクトのビルド対象ファイルの依存関係を更新します。サブプロジェクトを追加している場合は，サブプロジェクトのビルド対象ファイルの依存関係も更新します。
*アクティブ・プロジェクト*をビルド	アクティブ・プロジェクトのビルドを行います。 アクティブ・プロジェクトがメイン・プロジェクトの場合，サブプロジェクトのビルドは行いません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
*アクティブ・プロジェクト*をリビルド	アクティブ・プロジェクトのリビルドを行います。 アクティブ・プロジェクトがメイン・プロジェクトの場合，サブプロジェクトのリビルドは行いません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
*アクティブ・プロジェクト*をクリーン	アクティブ・プロジェクトのクリーンを行います。 アクティブ・プロジェクトがメイン・プロジェクトの場合，サブプロジェクトのクリーンは行いません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
*アクティブ・プロジェクト*の依存関係の更新	アクティブ・プロジェクトのビルド対象ファイルの依存関係を更新します。
ビルドを中止	実行中のビルド，リビルド，バッチ・ビルド，クリーンを中止します。
ビルド・モードの設定 ...	ビルド・モードの変更，追加等を行うために，ビルド・モード設定 ダイアログをオープンします。
バッチ・ビルド ...	バッチ・ビルドを行うために，バッチ・ビルド ダイアログをオープンします。
ビルド・オプション一覧	現在設定しているビルド・オプションを出力 パネルに一覧表示します。

(2) ツールバー

ビルド関連のボタン群を示します。

(a) ビルド・ツールバー

ビルド・ツールバーでは，ビルド関連を操作するボタン群を表示します。

	プロジェクトのビルドを行います。サブプロジェクトを追加している場合は，サブプロジェクトのビルドも行います。 なお，本ボタンは，ビルド・ツールが実行中の場合は無効となります。

	プロジェクトのリビルドを行います。サブプロジェクトを追加している場合は，サブプロジェクトのリビルドも行います。 なお，本ボタンは，ビルド・ツールが実行中の場合は無効となります。
	実行中のビルド，リビルド，バッチ・ビルド，クリーンを中止します。

（3）パネル表示エリア

以下のパネルを表示するエリアです。

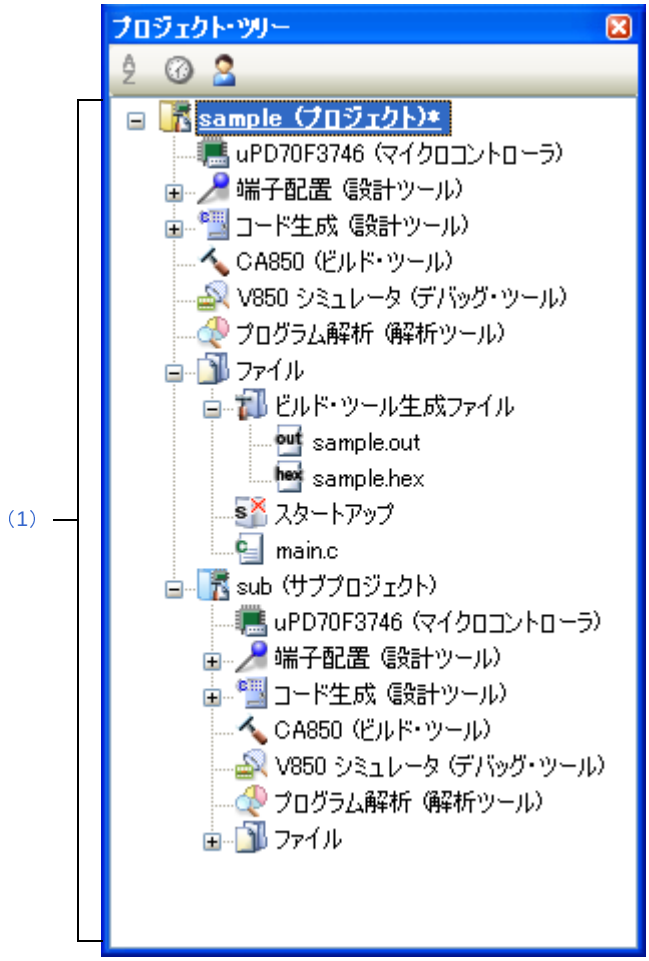
- プロジェクト・ツリー パネル
- プロパティ パネル
- エディタ パネル
- 出力 パネル

表示内容についての詳細は，各パネルの項を参照してください。

プロジェクト・ツリー パネル

プロジェクトを構成するビルド・ツール，ソース・ファイル等の構成要素をツリー表示します。

図 A―2　プロジェクト・ツリー パネル



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［［編集］メニュー（プロジェクト・ツリー パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- ［表示］メニュー→［プロジェクト・ツリー］を選択

［各エリアの説明］

(1) プロジェクト・ツリー エリア

プロジェクトの構成要素を以下のノードでツリー表示します。

ノード	説明
プロジェクト名 (プロジェクト) （以降，“プロジェクト・ノード”と呼びます。）	プロジェクトの名前です。
ビルド・ツール名 (ビルド・ツール) （以降，“ビルド・ツール・ノード”と呼びます。）	使用するビルド・ツール（コンパイラ，アセンブラ等）です。
ファイル （以降，“ファイル・ノード”と呼びます。）	プロジェクトに追加している以下のファイルが，直下に表示されます。 - C ソース・ファイル（*.c） - アセンブラ・ソース・ファイル（*.s） - ヘッダ・ファイル（*.h，*.inc） - オブジェクト・ファイル（*.o） - ライブラリ・ファイル（*.a） - リンク・ディレクティブ・ファイル（*.dr，*dir） - セクション・ファイル（*.sf） - その他のファイル（*.doc，*.xml 等）

ノード	説明
ビルド・ツール生成ファイル （以降，“ビルド・ツール生成ファイル・ノード”と呼びます。）	ビルド時に生成されるノードで，ビルド・ツールによって生成されたファイルのうち，以下のものが直下に表示されます。 - ライブラリ用のプロジェクト以外の場合 ロード・モジュール・ファイル (*.out) リンク・マップ・ファイル (*.map) ヘキサ・ファイル (*.hex) ダンプ・リスト（dump.txt） クロス・リファレンス情報（cxref） タグ情報（ctags） コール・ツリー情報（ccalltre.csv，ccalltre.lst） 関数計量情報（cmeasure.csv，cmeasure.lst） コール・データベース情報 （cprofile.csv，cprofile.dat） メモリ・マップ表 （rammap.csv） - ライブラリ用のプロジェクトの場合 アーカイブ・ファイル (*.a) ダンプ・リスト（dump.txt） クロス・リファレンス情報（cxref） タグ情報（ctags） コール・ツリー情報（ccalltre.csv，ccalltre.lst） 関数計量情報（cmeasure.csv，cmeasure.lst） コール・データベース情報 （cprofile.csv，cprofile.dat） 本ノードに表示されているファイルは，名前の変更，削除，移動を行うことができません。 なお，本ノードは常にファイル・ノード以下に生成されます。 ビルド後にプロジェクトの再読み込みを行った場合，本ノードは表示されなくなります。
スタートアップ （以降，“スタートアップ・ノード”と呼びます。）	プロジェクトに標準以外のスタートアップ・ルーチンを追加するためのノードです。 なお，本ノードは常にファイル・ノード以下に表示されます。
カテゴリ名 （以降，“カテゴリ・ノード”と呼びます。）	ファイルを分類するためにユーザが作成するカテゴリです（「2. 3. 6　ファイルをカテゴリに分類する」参照）。 なお，本ノードは常にファイル・ノード以下に作成されます。
サブプロジェクト名 (サブプロジェクト) （以降，“サブプロジェクト・ノード”と呼びます。）	プロジェクトに追加しているサブプロジェクトです。

各構成要素（ノード，またはファイル）を選択すると，その詳細情報（プロパティ）がプロパティ パネルに表示され，設定の変更を行うことができます。

備考　複数の構成要素を選択している場合は，その構成要素に共通するタブのみ表示されます。
なお，複数のファイルを選択し，共通するプロパティの値が異なる場合，その値は空欄となります。

本エリアは，次の機能を備えています。

(a) ファイルの追加

以下のいずれかの方法により，ファイルの追加を行うことができます。
ファイルの追加先はファイル・ノード以下となります。

- 既存のファイルを追加する場合
  - プロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノード，ファイルのいずれかを選択し，［ファイル］メニュー→［追加］→［既存のファイルを追加 ...］を選択する。既存のファイルを追加 ダイアログがオープンし，追加するファイルを選択する。
  - プロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノード，ファイルのいずれかのコンテキスト・メニューの［追加］→［既存のファイルを追加 ...］を選択する。既存のファイルを追加 ダイアログがオープンし，追加するファイルを選択する。
  - エクスプローラなどでファイルをコピーし，本エリアにフォーカスを移動したのち，［編集］メニュー→［貼り付け］を選択する。
  - エクスプローラなどからファイルをドラッグし，本エリア上のファイルを追加したい位置にドロップする。

備考 エクスプローラなどからファイルをドラッグし，プロジェクト・ツリー下部の空白部分にドロップした場合は，メイン・プロジェクト上にドロップしたものとみなされます。

- 新しいファイルを追加する場合
  - プロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノードのいずれかを選択し，［ファイル］メニュー→［追加］→［新しいファイルを追加 ...］を選択する。ファイル追加 ダイアログがオープンし，新しく作成するファイルを指定する。
  - プロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノードのいずれかのコンテキスト・メニューの［追加］→［新しいファイルを追加 ...］を選択する。ファイル追加 ダイアログがオープンし，新しく作成するファイルを指定する。

備考 ファイル追加 ダイアログで指定した場所に，空のファイルが作成されます。

(b) プロジェクトからファイルを外す

以下のいずれかの方法により，プロジェクトからファイルを外すことができます。
ファイル自体はファイル・システム上からは削除されません。

- プロジェクトから外すファイルを選択し，［プロジェクト］メニュー→［プロジェクトから外す］を選択する。
- プロジェクトから外すファイルを選択し，コンテキスト・メニューの［プロジェクトから外す］を選択する。

(c) ファイルの移動

以下の方法により，ファイルの移動を行うことができます。

ファイルの移動先はファイル・ノード以下となります。

- 移動するファイルをドラッグし，移動先でドロップする。

備考 1. メイン・プロジェクト内，またはサブプロジェクト内でドロップした場合は，ファイルに設定した個別オプションは保持されます。

2. 異なるプロジェクト間，または同一プロジェクトのメイン・プロジェクトかサブプロジェクトにドロップした場合は，ファイルは移動ではなく，コピーされます。なお，ファイルに設定した個別オプションは保持されません。

(d) カテゴリの追加

以下のいずれかの方法により，カテゴリ・ノードの追加を行うことができます。

カテゴリ・ノードの追加先はファイル・ノード以下となります。

- ［プロジェクト］メニュー→［新しいカテゴリを追加］を選択する。
- プロジェクト・ノード，サブプロジェクト・ノード，またはファイル・ノードのコンテキスト・メニューの［新しいカテゴリを追加］を選択する。

備考 1. カテゴリ名は，デフォルトで“新しいカテゴリ”となります。

2. すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。

(e) カテゴリの移動

以下の方法により，カテゴリ・ノードの移動を行うことができます。

カテゴリ・ノードの移動先はファイル・ノード以下となります。

- 移動するカテゴリ・ノードをドラッグし，移動先でドロップする。

備考 1. メイン・プロジェクト内，またはサブプロジェクト内でドロップした場合は，カテゴリ・ノードに含まれるファイルに設定した個別オプションは保持されます。

2. 異なるプロジェクト間，または同一プロジェクトのメイン・プロジェクトかサブプロジェクトにドロップした場合は，カテゴリ・ノードは移動ではなく，コピーされます。なお，カテゴリ・ノードに含まれるファイルに設定した個別オプションは保持されません。

(f) フォルダの追加

以下の方法により，エクスプローラなどからフォルダの追加を行うことができます。

フォルダの追加先はファイル・ノード以下となります。

なお，フォルダはカテゴリとして追加されます。

- エクスプローラなどからフォルダをドラッグし，移動先でドロップする。フォルダとファイル追加ダイアログがオープンし，フォルダに含まれているファイルのうち追加するファイルの種類と，フォルダの階層を指定する。

注意　フォルダとファイルを同時にドラッグして，本エリアにドロップすることはできません。

(g) サブプロジェクトのビルド順の表示編集

サブプロジェクトは，ビルド順に上から表示されます。そのため，サブプロジェクトの表示位置を変更することで，ビルド順も変更することができます。

なお，プロジェクトのビルドは，すべてのサブプロジェクト，メイン・プロジェクトの順で行います。

(h) 標準ビルド・オプションの設定

プロパティ パネルにおいて，標準ビルド・オプションの設定に変更を加えると，プロパティの値が太字表示されます。

以下の方法により，現在設定しているビルド・オプションを標準ビルド・オプションとする（太字表示を解除する）ことができます。

- ビルド・ツール・ノードを選択し，コンテキスト・メニューの［現在のビルド・オプションをプロジェクトの標準に設定する］を選択する。

備考　標準ビルド・オプションの設定は，プロジェクト全体（メイン・プロジェクト，およびサブプロジェクト）に対して行います。

(i) ファイル，およびカテゴリのソート

以下の方法により，ファイル，およびカテゴリ・ノードをファイル名順，タイムスタンプ順，ユーザ指定順でソートすることができます。

- ツールバーのいずれかのボタンを選択する。

以下に，各ボタンの説明を示します。

なお，デフォルトでは  が選択されます。

ボタン	説明
	カテゴリ・ノード，およびファイルを名前順でソートします。  ：昇順  ：降順  ：昇順
	カテゴリ・ノード，およびファイルをタイムスタンプ順でソートします。  ：降順  ：昇順  ：降順
	カテゴリ・ノードとファイルをユーザが指定した順で表示します（デフォルト）。 カテゴリ・ノード，およびファイルをドラッグ・アンド・ドロップすることにより，表示順を任意に変更することができます。

(j) 編集中ファイルの表示

プロジェクトに追加しているファイルをエディタ パネルで編集し，未保存の場合，ファイル名の後ろに“*”を表示します。ファイルを保存すると，“*”は消去されます。

通常のファイル	main.c
編集後，未保存のファイル	main.c*

(k) 個別ビルド・オプションを設定しているソース・ファイルの強調表示

プロジェクトの全体オプションとは異なるビルド・オプション（個別コンパイル・オプション，個別アセンブル・オプション）を設定しているソース・ファイルのアイコンは，通常とは異なるアイコンに変更されます。

通常のファイル	main.c
個別ビルド・オプションを設定しているファイル	main.c

(l) 読み取り専用属性ファイルの強調表示

プロジェクトに追加している読み取り専用属性ファイルは，イタリック表示されます。

通常のファイル	main.c
読み取り専用属性ファイル	*main.c*

(m) 存在しないファイルの強調表示

プロジェクトに追加しているファイルで，存在しないファイルは，グレー表示され，アイコンは淡色表示となります。

通常のファイル	main.c
存在しないファイル	main.c

(n) ビルド対象ファイルの強調表示

- ビルド（ラピッド・ビルド），リビルド，コンパイル，アセンブル時にエラーが検出されたファイルは，以下の例のように強調表示されます。

エラーもワーニングも検出されなかったファイル	main.c
エラーが検出されたファイル	main.c
ワーニングが検出されたファイル	main.c

備考 1. エラーとワーニングの両方が検出されたファイルは，赤色表示されます。

2. 強調表示は，ビルド・オプション（全体オプション，または個別オプション）の変更，およびビルド・モードの変更により解除されます。

- 以下のファイルは，太字表示されます。
  - 編集後，コンパイルしていないソース・ファイル
  - クリーンを実行した場合のソース・ファイル
  - ビルド・ツールのオプションを変更した場合のソース・ファイル
  - ビルド・モードを変更した場合のソース・ファイル

備考 プロジェクトを開いた直後は，すべて太字表示となり，ビルドを行うことで太字表示は解除されます。

(o) ビルド対象外ファイルの強調表示

ビルドの対象外に設定しているファイルは，以下の例のように強調表示されます。

通常のファイル	 main.c
ビルド対象外ファイル	 main.c

(p) 変更したプロジェクトの強調表示

プロジェクトに追加しているファイル構成を変更した場合，およびプロジェクトの構成要素のプロパティを変更した場合，プロジェクト名に“*”が付加され，太字表示されます。

強調表示は，プロジェクトを保存すると解除されます。

通常のプロジェクト	 sample (プロジェクト)
変更したプロジェクト	 **sample (プロジェクト)***

(q) アクティブ・プロジェクトの強調表示

アクティブ・プロジェクトには，下線が付加されます。

通常のプロジェクト	 **sample (プロジェクト)***
アクティブ・プロジェクト	 **sample (プロジェクト)***

(r) エディタの起動

特定の拡張子を持つファイルをエディタ パネルでオープンします。オプション ダイアログで，外部エディタを使用する設定をしている場合は，設定している外部エディタでオープンします。それ以外のファイルは，ホスト OS で関連付けられているアプリケーションで起動します。

注意 ホスト OS で関連付けられていない拡張子のファイルは表示されません。

以下のいずれかの方法により，エディタをオープンすることができます。

- ファイルをダブルクリックする。
- ファイルを選択し，コンテキスト・メニューの［開く］を選択する。
- ファイルを選択し，［Enter］キーを押下する。

以下に，エディタ パネルでオープンできるファイルを示します。

- C ソース・ファイル（*.c）
- アセンブラ・ソース・ファイル（*.s）
- ヘッダ・ファイル（*.h，*.inc）
- リンク・ディレクティブ・ファイル（*.dr，*.dir）
- セクション・ファイル（*.sf）
- マップ・ファイル（*.map）
- ヘキサ・ファイル（*.hex）
- テキスト・ファイル（*.txt）

備考　以下のいずれかの方法により，上記以外のファイルもエディタ パネルでオープンすることができます。

- ファイルをドラッグし，エディタ パネルにドロップする。
- ファイルを選択し，コンテキスト・メニューの［内部エディタで開く ...］を選択する。

[[編集]メニュー（プロジェクト・ツリー パネル専用部分）]

コピー	選択しているファイル，カテゴリ・ノードをクリップ・ボードにコピーします。 ファイル名，カテゴリ名を編集中の場合は，選択している文字列をクリップ・ボードにコピーします。 なお，本メニューは，ファイル，カテゴリ・ノードを選択している場合のみ有効となります。
貼り付け	クリップ・ボードの内容をプロジェクト・ツリー上で選択しているノードの直下に挿入します。 ファイル名，カテゴリ名を編集中の場合は，クリップ・ボードの内容を挿入します。 なお，本メニューは，クリップボートの内容が同一プロジェクトに存在する場合，ファイル，カテゴリ・ノードを複数選択している場合，およびビルド・ツールが実行中の場合は無効となります。
名前の変更	選択しているプロジェクト，サブプロジェクト，ファイル，カテゴリ・ノードの名前が編集可能な状態になります。[Enter]キーの押下により編集を確定し，[ESC]キーの押下により編集をキャンセルすることができます。 ファイルを選択している場合は，実際のファイル名も変更されます。 ファイルを選択し，そのファイルを他のプロジェクトにも追加している場合は，それらの名前も変更されます。 なお，本メニューは，プロジェクト，サブプロジェクト，ファイル，カテゴリ・ノードを選択している場合のみ有効となります。ただし，ビルド・ツールが実行中の場合は無効となります。

[コンテキスト・メニュー]

(1) プロジェクト・ノードを選択している場合

*アクティブ・プロジェクト*をビルド	アクティブ・プロジェクトのビルドを行います。 アクティブ・プロジェクトがメイン・プロジェクトの場合，サブプロジェクトのビルドは行いません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
*アクティブ・プロジェクト*をリビルド	アクティブ・プロジェクトのリビルドを行います。 アクティブ・プロジェクトがメイン・プロジェクトの場合，サブプロジェクトのリビルドは行いません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
*アクティブ・プロジェクト*をクリーン	アクティブ・プロジェクトのクリーンを行います。 アクティブ・プロジェクトがメイン・プロジェクトの場合，サブプロジェクトのクリーンは行いません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
エクスプローラでフォルダを開く	選択しているプロジェクトのプロジェクト・ファイルが存在しているフォルダをエクスプローラでオープンします。
追加	プロジェクトにサブプロジェクト，ファイルを追加するためのカスケード・メニューを表示します。

既存のサブプロジェクトを追加 ...	既存のサブプロジェクトを追加 ダイアログをオープンし，選択したサブプロジェクトをプロジェクトに追加します。
新しいサブプロジェクトを追加 ...	プロジェクト作成 ダイアログをオープンし，作成したサブプロジェクトをプロジェクトに追加します。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	ファイル・ノードの直下にカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，200 文字まで指定可能です。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
選択しているプロジェクトをアクティブ・プロジェクトに設定	選択しているプロジェクトをアクティブ・プロジェクトに設定します。
プロジェクトと開発ツールをパックして保存 ...	本製品一式とプロジェクト一式を指定したフォルダにコピーして，一つのフォルダにまとめて保存します。
貼り付け	本メニューは常に無効です。
名前の変更	選択しているプロジェクトの名前が編集可能な状態になります。
プロパティ	選択しているプロジェクトのプロパティをプロパティ パネルに表示します。

(2) サブプロジェクト・ノードを選択している場合

アクティブ・プロジェクトをビルド	アクティブ・プロジェクトのビルドを行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
アクティブ・プロジェクトをリビルド	アクティブ・プロジェクトのリビルドを行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
アクティブ・プロジェクトをクリーン	アクティブ・プロジェクトのクリーンを行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
エクスプローラでフォルダを開く	選択しているサブプロジェクトのサブプロジェクト・ファイルが存在しているフォルダをエクスプローラでオープンします。
追加	プロジェクトにサブプロジェクト，ファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。

既存のサブプロジェクトを追加 ...	既存のサブプロジェクトを追加 ダイアログをオープンし，選択したサブプロジェクトをプロジェクトに追加します。 サブプロジェクトにサブプロジェクトを追加することはできません。
新しいサブプロジェクトを追加 ...	プロジェクト作成 ダイアログをオープンし，作成したサブプロジェクトをプロジェクトに追加します。 サブプロジェクトにサブプロジェクトを追加することはできません。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	ファイル・ノードの直下にカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，200 文字まで指定可能です。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
*選択しているサブプロジェクト*をアクティブ・プロジェクトに設定	選択しているサブプロジェクトをアクティブ・プロジェクトに設定します。
プロジェクトから外す	選択しているサブプロジェクトをプロジェクトから外します。 サブプロジェクト・ファイル自体はファイル・システム上からは削除されません。 選択しているサブプロジェクトがアクティブ・プロジェクトの場合は，プロジェクトから外すことはできません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
貼り付け	本メニューは常に無効です。
名前の変更	選択しているサブプロジェクトの名前が編集可能な状態になります。
プロパティ	選択しているサブプロジェクトのプロパティをプロパティ パネルに表示します。

(3) ビルド・ツール・ノードを選択している場合

ビルド・プロジェクト	選択しているプロジェクト（メイン・プロジェクト，またはサブプロジェクト）のビルドを行います。サブプロジェクトを追加している場合は，サブプロジェクトのビルドも行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
リビルド・プロジェクト	選択しているプロジェクト（メイン・プロジェクト，またはサブプロジェクト）のリビルドを行います。サブプロジェクトを追加している場合は，サブプロジェクトのリビルドも行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。

クリーン・プロジェクト	選択しているプロジェクト（メイン・プロジェクト，またはサブプロジェクト）のクリーンを行います。サブプロジェクトを追加している場合は，サブプロジェクトのクリーンも行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
現在のビルド・オプションを*選択しているプロジェクト*の標準に設定する	現在のビルド・オプションを選択しているプロジェクトの標準に設定します。サブプロジェクトを追加している場合，サブプロジェクトの設定は行いません。 標準と異なるビルド・オプションを設定した場合，そのプロパティは太字表示されます。
リンク指定順を設定する ...	リンク順設定 ダイアログをオープンし，オブジェクト・モジュール・ファイル，ライブラリ・ファイルの表示，およびリンク順の設定を行います。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
リンク・ディレクティブ・ファイルを生成する ...	リンク・ディレクティブ生成 ダイアログをオープンし，リンク・ディレクティブ・ファイルの生成を行います。
プロパティ	選択しているビルド・ツールのプロパティをプロパティ パネルに表示します。

（4）ファイル・ノードを選択している場合

追加	プロジェクトにファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。追加先は，本ノードの直下です。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。追加先は，本ノードの直下です。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	本ノードの直下にカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，200 文字まで指定可能です。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
プロジェクトから外す	本メニューは常に無効です。
コピー	本メニューは常に無効です。
貼り付け	クリップ・ボードの内容を本ノードの直下に挿入します。 ただし，クリップボートの内容が同一プロジェクトに存在する場合は，無効となります。
名前の変更	本メニューは常に無効です。
プロパティ	本ノードのプロパティをプロパティ パネルに表示します。

(5) ファイルを選択している場合

コンパイル	選択しているC ソース・ファイルをコンパイルします。 なお，本メニューは，C ソース・ファイル（ビルド対象外のファイルを除く）を選択している場合のみ表示されます。 ただし，ビルド・ツールが実行中の場合は無効となります。
アセンブル	選択しているアセンブラ・ソース・ファイルをアセンブルします。 なお，本メニューは，アセンブラ・ソース・ファイル（ビルド対象外のファイルを除く）を選択している場合のみ表示されます。 ただし，ビルド・ツールが実行中の場合は無効となります。
開く	ファイルの拡張子に割り当てられたアプリケーションで選択しているファイルをオープンします（「(r) エディタの起動」参照）。 なお，本メニューは，複数のファイルを選択している場合は無効となります。
内部エディタで開く ...	エディタ パネルで選択しているファイルをオープンします。 なお，本メニューは，複数のファイルを選択している場合は無効となります。
アプリケーションを指定して開く ...	プログラムから開く ダイアログをオープンし，指定したアプリケーションで選択しているファイルをオープンします。 ただし，複数のファイルを選択している場合は無効となります。
エクスプローラでフォルダを開く	選択しているファイルが存在しているフォルダをエクスプローラでオープンします。
追加	プロジェクトにファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。追加先は，選択しているファイルと同じレベルです。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。追加先は，選択しているファイルと同じレベルです。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	選択しているファイルと同じレベルにカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，200 文字まで指定可能です。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
プロジェクトから外す	選択しているファイルをプロジェクトから外します。 ファイル自体はファイル・システム上からは削除されません。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
コピー	選択しているファイルをクリップ・ボードにコピーします。 ファイル名を編集中の場合は，選択している文字列をクリップ・ボードにコピーします。
貼り付け	本メニューは常に無効です。

名前の変更	選択しているファイルの名前が編集可能な状態になります。 実際のファイル名も変更されます。 選択しているファイルを他のプロジェクトにも追加している場合は，それらの名前も変更されます。
プロパティ	選択しているファイルのプロパティをプロパティ パネルに表示します。

(6) ビルド・ツール生成ファイル・ノードを選択している場合

プロパティ	本ノードのプロパティをプロパティ パネルに表示します。

(7) スタートアップ・ノードを選択している場合

追加	プロジェクトにファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。
既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。追加先は，本ノードの直下です。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。追加先は，本ノードの直下です。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	本ノードの直下にカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，200 文字まで指定可能です。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
プロジェクトから外す	本メニューは常に無効です。
コピー	本メニューは常に無効です。
貼り付け	クリップ・ボードの内容を本ノードの直下に挿入します。 ただし，クリップボートの内容が同一プロジェクトに存在する場合は，無効となります。
名前の変更	本メニューは常に無効です。
プロパティ	本ノードのプロパティをプロパティ パネルに表示します。

(8) カテゴリ・ノードを選択している場合

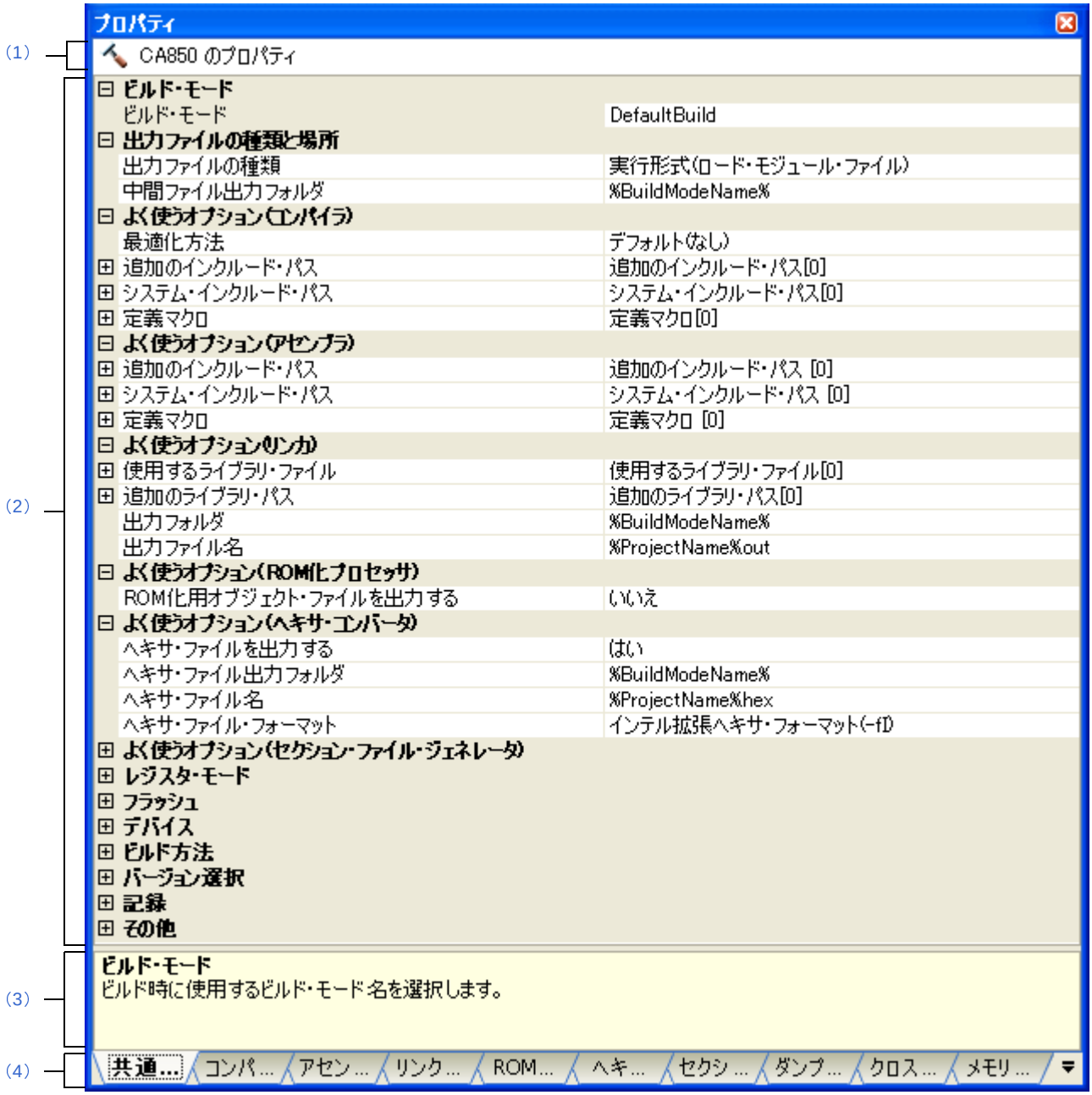
追加	プロジェクトにファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。

既存のファイルを追加 ...	既存のファイルを追加 ダイアログをオープンし，選択したファイルをプロジェクトに追加します。追加先は，本ノードの直下です。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいファイルを追加 ...	ファイル追加 ダイアログをオープンし，選択した種類でファイルを作成し，プロジェクトに追加します。追加先は，本ノードの直下です。 追加したファイルはファイルの拡張子に割り当てられたアプリケーションでオープンされます。
新しいカテゴリを追加	本ノードの直下にカテゴリ・ノードを追加し，カテゴリ名が編集可能な状態になります。 カテゴリ名は，200 文字まで指定可能です。 カテゴリ名は，デフォルトで“新しいカテゴリ”となります。すでに存在するカテゴリ・ノードと同名のカテゴリ・ノードを追加することもできます。 なお，本メニューは，ビルド・ツールが実行中の場合，およびカテゴリのネスト数が 20 の場合は無効となります。
プロジェクトから外す	選択しているカテゴリ・ノードをプロジェクトから外します。 なお，本メニューは，ビルド・ツールが実行中の場合は無効となります。
コピー	選択しているカテゴリ・ノードをクリップ・ボードにコピーします。 カテゴリ名を編集中の場合は，選択している文字列をクリップ・ボードにコピーします。
貼り付け	クリップ・ボードの内容を本ノードの直下に挿入します。 ただし，クリップボートの内容が同一プロジェクトに存在する場合は，無効となります。 カテゴリ名を編集中の場合は，クリップ・ボードの内容を挿入します。
名前の変更	選択しているカテゴリ・ノードの名前が編集可能な状態になります。
プロパティ	選択しているカテゴリ・ノードのプロパティをプロパティ パネルに表示します。

プロパティ パネル

プロジェクト・ツリー パネルで選択しているビルド・ツール・ノード，ファイル，カテゴリ・ノードについて，カテゴリ別に詳細情報の表示，および設定の変更を行います。

図 A―3　プロパティ パネル



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［[編集］メニュー（プロパティ パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- プロジェクト・ツリー パネルにおいて，ビルド・ツール・ノード，ファイル，カテゴリ・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択，またはコンテキスト・メニュー→［プロパティ］を選択

備考　すでにプロパティ パネルがオープンしている場合，プロジェクト・ツリー パネルにおいて，ビルド・ツール・ノード，ファイル，カテゴリ・ノードを選択することで，選択した項目の詳細情報を表示します。

［各エリアの説明］

（1）対象名エリア

プロジェクト・ツリー パネルで選択しているノードの名称を表示します。

複数のノードを選択している場合，本エリアは空欄となります。

（2）詳細情報表示／変更エリア

プロジェクト・ツリー パネルで選択しているビルド・ツール・ノード，ファイル，カテゴリ・ノードの詳細情報を，カテゴリ別のリスト形式で表示し，設定の変更を直接行うことができるエリアです。

⊟マークは，そのカテゴリ内に含まれているすべての項目が展開表示されていることを示し，また，⊞マークは，カテゴリ内の項目が折りたたみ表示されていることを示します。展開／折りたたみ表示の切り替えは，このマークのクリック，またはカテゴリ名のダブルクリックにより行うことができます。

HEXマークは，そのプロパティのテキスト・ボックスが 16 進数入力専用であることを示します。

カテゴリ，およびそれに含まれる項目の表示内容／設定方法についての詳細は，該当するタブの項を参照してください。

（3）プロパティの説明エリア

詳細情報表示／変更エリアで選択したカテゴリや項目の簡単な説明を表示します。

（4）タブ選択エリア

タブを選択することにより，詳細情報を表示するカテゴリが切り替わります。

本パネルには，次のタブが存在します（各タブ上における表示内容／設定方法についての詳細は，該当するタブの項を参照してください）。

（a）プロジェクト・ツリー パネルでビルド・ツール・ノードを選択している場合

- ［共通オプション］タブ
- ［コンパイル・オプション］タブ

- ［アセンブル・オプション］タブ
- ［リンク・オプション］タブ
- ［ROM 化プロセス・オプション］タブ
- ［ヘキサ・コンバート・オプション］タブ
- ［アーカイブ・オプション］タブ
- ［セクション・ファイル・ジェネレート・オプション］タブ
- ［ダンプ・オプション］タブ
- ［クロス・リファレンス・オプション］タブ
- ［メモリ・レイアウト視覚化オプション］タブ

（b）プロジェクト・ツリー パネルでファイルを選択している場合

- ［ビルド設定］タブ（C ソース・ファイル，アセンブラ・ソース・ファイル，リンク・ディレクティブ・ファイル，セクション・ファイル，オブジェクト・ファイル，ライブラリ・ファイルの場合）
- ［個別コンパイル・オプション］タブ（C ソース・ファイルの場合）
- ［個別アセンブル・オプション］タブ（アセンブラ・ソース・ファイルの場合）
- ［ファイル情報］タブ

（c）プロジェクト・ツリー パネルでカテゴリ・ノード，ファイル・ノード，ビルド・ツール生成ファイル・ノード，スタートアップ・ノードを選択している場合

- ［カテゴリ情報］タブ

備考　プロジェクト・ツリー パネルで複数の構成要素を選択している場合は，その構成要素に共通するタブのみ表示されます。プロパティの値の変更は，選択している複数の構成要素に共通に反映されます。

［［編集］メニュー（プロパティ パネル専用部分）］

元に戻す	直前に行ったプロパティの値の編集作業を取り消します。
切り取り	プロパティの値を編集中の場合，選択している文字列を切り取ってクリップ・ボードに移動します。
コピー	選択しているプロパティの値文字列をクリップ・ボードにコピーします。
貼り付け	プロパティの値を編集中の場合，クリップ・ボードの内容を挿入します。
削除	プロパティの値を編集中の場合，選択している文字列を削除します。
すべて選択	プロパティの値を編集中の場合，選択しているプロパティの値文字列をすべて選択します。

[コンテキスト・メニュー]

元に戻す	直前に行ったプロパティの値の編集作業を取り消します。
切り取り	プロパティの値を編集中の場合，選択している文字列を切り取ってクリップ・ボードに移動します。
コピー	選択しているプロパティの値文字列をクリップ・ボードにコピーします。
貼り付け	プロパティの値を編集中の場合，クリップ・ボードの内容を挿入します。
削除	プロパティの値を編集中の場合，選択している文字列を削除します。
すべて選択	プロパティの値を編集中の場合，選択しているプロパティの値文字列をすべて選択します。
デフォルトに戻す	選択している項目の設定値をプロジェクトに設定しているデフォルト値に戻します。 ただし，［個別コンパイル・オプション］タブ，［個別アセンブル・オプション］タブにおいては，全体オプションの設定値に戻します。
すべてデフォルトに戻す	現在表示しているタブの設定値をすべてプロジェクトに設定しているデフォルト値に戻します。 ただし，［個別コンパイル・オプション］タブ，［個別アセンブル・オプション］タブにおいては，全体オプションの設定値に戻します。

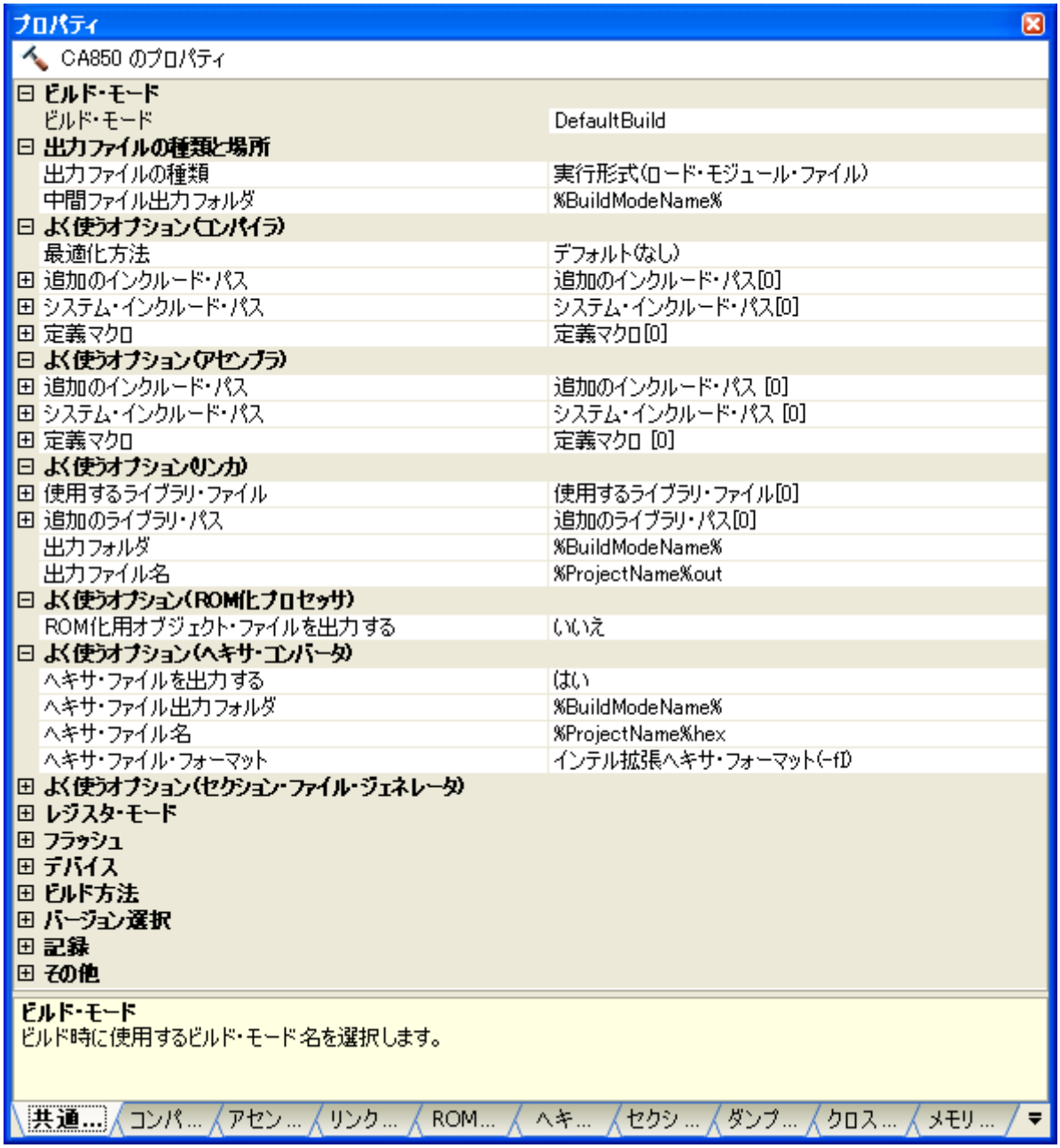
［共通オプション］タブ

本タブでは，ビルド・ツールに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

備考　［よく使うオプション］カテゴリのプロパティを変更した場合，それらに対応するタブの同名のプロパティの値も連動して変更されます。

［共通オプション］タブのカテゴリ	対応するタブ
［よく使うオプション（コンパイラ）］カテゴリ	［コンパイル・オプション］タブ
［よく使うオプション（アセンブラ）］カテゴリ	［アセンブル・オプション］タブ
［よく使うオプション（リンカ）］カテゴリ	［リンク・オプション］タブ
［よく使うオプション（ROM 化プロセッサ）］カテゴリ	［ROM 化プロセス・オプション］タブ
［よく使うオプション（ヘキサ・コンバータ）］カテゴリ	［ヘキサ・コンバート・オプション］タブ
［よく使うオプション（セクション・ファイル・ジェネレータ）］カテゴリ	［セクション・ファイル・ジェネレート・オプション］タブ

図 A―4　プロパティ パネル：［共通オプション］タブ



［各カテゴリの説明］

(1) ［ビルド・モード］

ビルド・モードに関する詳細情報の表示，および設定の変更を行います。

ビルド・モード	ビルド時に使用するビルド・モードを選択します。 なお，コンテキスト・メニューの［すべてデフォルトに戻す］は，本プロパティには適用されません。		
	デフォルト	DefaultBuild	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	DefaultBuild	プロジェクトの新規作成時にデフォルトで設定されるビルド・モードでビルドを行います。
		プロジェクトに登録しているビルド・モード（DefaultBuild 以外）	プロジェクトに登録しているビルド・モード（DefaultBuild 以外）でビルドを行います。

(2)［出力ファイルの種類と場所］

出力ファイルの種類と場所に関する詳細情報の表示，および設定の変更を行います。

出力ファイルの種類	ビルド時に生成するファイルの種類を選択します。 ここで設定したファイルの種類がデバッグ対象となります。 なお，ライブラリ用のプロジェクト以外の場合は［実行形式（ROM 化用モジュール・ファイル）］，［実行形式（ロード・モジュール・ファイル）］，［実行形式（ヘキサ・ファイル）］のみが表示されます。ただし，［ヘキサ・コンバート・オプション］タブの［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで［いいえ］を選択した場合は，［実行形式（ROM 化用モジュール・ファイル）］，［実行形式（ロード・モジュール・ファイル）］のみが表示されます。［ROM 化プロセス・オプション］タブの［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［いいえ］を選択した場合は，［実行形式（ロード・モジュール・ファイル）］，［実行形式（ヘキサ・ファイル）］のみが表示されます。 ライブラリ用のプロジェクトの場合は［ライブラリ形式］のみが表示されます。		
	デフォルト	- ライブラリ用のプロジェクト以外の場合 実行形式（ロード・モジュール・ファイル） - ライブラリ用のプロジェクトの場合 ライブラリ形式	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	実行形式（ROM 化用モジュール・ファイル）	ビルド時に生成するファイルを実行形式（ROM 化用モジュール・ファイル）とします。
		実行形式（ロード・モジュール・ファイル）	ビルド時に生成するファイルを実行形式（ロード・モジュール・ファイル）とします。
		実行形式（ヘキサ・ファイル）	ビルド時に生成するファイルを実行形式（ヘキサ・ファイル）とします。
		ライブラリ形式	ビルド時に生成するファイルをライブラリ形式（ライブラリ・ファイル）とします。

<table>
<tr><td rowspan="6">デバイス共通オブジェクトを出力する</td><td colspan="3">デバイス共通のオブジェクトを出力するかどうかを選択します。
コンパイラ，およびアセンブラのオプション -cn，-cnv850e，-cnv850e2 に相当します。
なお，本プロパティは，ライブラリ用のプロジェクトの場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ (デバイス固有)(なし)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="4">指定可能値</td><td>はい (V850 コア共通)(-cn)</td><td>V850 コアで共通に使用することができるオブジェクトを出力します。
V850/V850ES/V850E1/V850E2 コアのオブジェクトとリンクが可能です。</td></tr>
<tr><td>はい (V850E/ES コア共通)(-cnv850e)</td><td>V850E/ES コアで共通に使用することができるオブジェクトを出力します。
V850ES/V850E1/V850E2 コアのオブジェクトとリンクが可能です。</td></tr>
<tr><td>はい (V850E2 コア共通)(-cnv850e2)</td><td>V850E2 コアで共通に使用することができるオブジェクトを出力します。
V850E2 コアのオブジェクトとリンクが可能です。</td></tr>
<tr><td>いいえ (デバイス固有)(なし)</td><td>指定したデバイス固有の情報を持つオブジェクトを出力します。
ライブラリに SFR 名や割り込みを用いた記述が可能です。</td></tr>
<tr><td rowspan="4">中間ファイル出力フォルダ</td><td colspan="3">中間ファイルを出力するフォルダへのパスを指定します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
空欄の場合は，プロジェクト・フォルダを指定したものとみなします。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%BuildModeName%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
</table>

(3)［よく使うオプション（コンパイラ）］

コンパイラでよく使うオプションに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td>最適化方法</td><td colspan="3">コンパイルの最適化の種類を選択します。
コンパイラのオプション -O* に相当します。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">デフォルト (なし)</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td></td><td>指定可能値</td><td>デバッグ優先 (-Od)</td><td>デバッグを優先して最適化を行います。
ROM 容量や実行速度に着目せず，ソース・デバッグに着眼したコードを生成します。</td></tr>
<tr><td></td><td></td><td>デフォルト (なし)</td><td>ソース・デバッグに着眼したコードを生成します。
ソース・デバッグに影響のない範囲で最適化を行います。</td></tr>
<tr><td></td><td></td><td>標準最適化 (-Og)</td><td>適度な最適化を行います。
ほとんどの場合で C ソース・デバッグが可能となる最適化を行います。</td></tr>
<tr><td></td><td></td><td>高度な最適化 (-O)</td><td>高度な最適化を行います。
ROM 容量に着目した最適化を行います。</td></tr>
<tr><td></td><td></td><td>より高度な最適化 (オブジェクト・サイズ優先)(-Os)</td><td>より高度な最適化（オブジェクト・サイズ優先）を行います。
ROM 容量を最も重視した最大限の最適化を行います。</td></tr>
<tr><td></td><td></td><td>より高度な最適化 (実行速度優先)(-Ot)</td><td>より高度な最適化（実行速度優先優先）を行います。
実行速度を最も重視した最大限の最適化を行います。</td></tr>
<tr><td>追加のインクルード・パス</td><td colspan="3">コンパイル時の追加のインクルード・パスを指定します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%　：ビルド・モード名に置換します。
%ProjectName%　：プロジェクト名に置換します。
%MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。
このオプションを省略した場合，コンパイラの標準フォルダのみ検索します。なお，パスはプロジェクト・フォルダを基点とします。
コンパイラのオプション -I に相当します。
指定したインクルード・パスはサブプロパティとして表示されます。
なお，プロジェクト・ツリーにインクルード・ファイルを追加すると，そのインクルード・パスをサブプロパティの最初に追加します。
インクルード・パスに大文字，小文字の区別はありません。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">追加のインクルード・パス [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，パス編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td colspan="2">259 文字までの文字列
64 個まで指定可能です。</td></tr>
</table>

システム・インクルード・パス	コンパイル時にシステムが設定するインクルード・パスを表示します。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%　：ビルド・モード名に置換します。 %ProjectName%　：プロジェクト名に置換します。 %MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。 システム・インクルード・パスは，追加のインクルード・パスより低い優先度で検索します。 パスはプロジェクト・フォルダを基点とします。 コンパイラのオプション -I に相当します。 インクルード・パスはサブプロパティとして表示します。	
	デフォルト	システム・インクルード・パス [*定義数*]
	変更方法	[...] ボタンをクリックし，システム・インクルード・パス順設定 ダイアログによる編集
	指定可能値	変更不可（インクルード・パスの設定順の変更のみ可能）
定義マクロ	定義したいマクロ名を指定します。 「*マクロ名* = *定義値*」の形式で 1 行に 1 つずつ指定します。「= *定義値*」の部分は省略可能で，省略した場合，定義値を 1 とします。 コンパイラのオプション -D に相当します。 指定したマクロはサブプロパティとして表示されます。	
	デフォルト	定義マクロ [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

(4) [よく使うオプション（アセンブラ）]

アセンブラでよく使うオプションに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td>追加のインクルード・パス</td><td colspan="2">アセンブル時の追加のインクルード・パスを指定します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%　：ビルド・モード名に置換します。
%ProjectName%　　：プロジェクト名に置換します。
%MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。
このオプションを省略した場合，アセンブラの標準フォルダのみ検索します。なお，パスはプロジェクト・フォルダを基点とします。
アセンブラのオプション -I に相当します。
指定したインクルード・パスはサブプロパティとして表示されます。
なお，プロジェクト・ツリーにインクルード・ファイルを追加すると，そのインクルード・パスをサブプロパティの最初に追加します。
インクルード・パスに大文字，小文字の区別はありません。</td></tr>
<tr><td></td><td>デフォルト</td><td>追加のインクルード・パス [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>[...] ボタンをクリックし，パス編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td>259 文字までの文字列
64 個まで指定可能です。ただし，連携するツールが使用するパスの数も含みます。</td></tr>
<tr><td>システム・インクルード・パス</td><td colspan="2">アセンブル時にシステムが設定するインクルード・パスを表示します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%　：ビルド・モード名に置換します。
%ProjectName%　　：プロジェクト名に置換します。
%MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。
システム・インクルード・パスは，追加のインクルード・パスより低い優先度で検索します。
パスはプロジェクト・フォルダを基点とします。
アセンブラのオプション -I に相当します。
インクルード・パスはサブプロパティとして表示します。</td></tr>
<tr><td></td><td>デフォルト</td><td>システム・インクルード・パス [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>[...] ボタンをクリックし，システム・インクルード・パス順設定 ダイアログによる編集</td></tr>
<tr><td></td><td>指定可能値</td><td>変更不可（インクルード・パスの設定順の変更のみ可能）</td></tr>
<tr><td>定義マクロ</td><td colspan="2">定義したいマクロ名を指定します。
「(マクロ名)=(定義値)」の形式で 1 行に 1 つずつ指定します。「=(定義値)」の部分は省略可能で，省略した場合，定義値を 1 とします。
アセンブラのオプション -D に相当します。
指定したマクロはサブプロパティとして表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td>定義マクロ [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td>256 文字までの文字列
256 個まで指定可能です。</td></tr>
</table>

(5)［よく使うオプション（リンカ）］

リンカでよく使うオプションに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，ライブラリ用のプロジェクトの場合は表示されません。

<table>
<tr><td rowspan="4">使用するライブラリ・ファイル</td><td colspan="2">標準ライブラリ以外の使用するライブラリ・ファイル名（libstring.a）を指定します。
string のみを指定してください（例：“abc”と指定すると，libabc.a を指定したものとみなされます）。
1 行に 1 ファイルずつ指定します。
ライブラリ・ファイルはライブラリ・パスから検索します。
リンカのオプション -l に相当します。
指定したライブラリ・ファイル名はサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td>使用するライブラリ・ファイル [定義数]</td></tr>
<tr><td>変更方法</td><td>［...］ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td>63 文字までの文字列
256 個まで指定可能です。</td></tr>
<tr><td rowspan="4">追加のライブラリ・パス</td><td colspan="2">標準ライブラリ以外の使用するライブラリ・ファイルの検索フォルダを指定します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%　：ビルド・モード名に置換します。
%ProjectName%　　：プロジェクト名に置換します。
%MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。
ライブラリ・ファイルはライブラリ・パスから検索します。なお，相対パスを指定した場合，プロジェクト・フォルダを基点とします。
リンカのオプション -L に相当します。
指定したライブラリ・パス名はサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td>追加のライブラリ・パス [定義数]</td></tr>
<tr><td>変更方法</td><td>［...］ボタンをクリックし，パス編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td>259 文字までの文字列
256 個まで指定可能です。</td></tr>
<tr><td rowspan="4">出力フォルダ</td><td colspan="2">生成するロード・モジュール・ファイルを格納するフォルダを指定します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%：ビルド・モード名に置換します。
空欄の場合は，プロジェクト・フォルダを指定したものとみなします。</td></tr>
<tr><td>デフォルト</td><td>%BuildModeName%</td></tr>
<tr><td>変更方法</td><td>テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td>247 文字までの文字列</td></tr>
</table>

出力ファイル名	生成するロード・モジュールのファイル名を指定します。 “.out”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.out”が自動的に付加されます。 リンカのオプション -o に相当します。 埋め込みマクロとして次のマクロ名があります。 %ProjectName%：プロジェクト名に置換します。	
	デフォルト	%ProjectName%.out
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

（6）［よく使うオプション（ROM 化プロセッサ）］

ROM 化プロセッサでよく使うオプションに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，ライブラリ用のプロジェクトの場合は表示されません。

ROM 化用オブジェクト・ファイルを出力する	ROM 化用オブジェクト・ファイルを出力するかどうかを選択します。 コンパイラのオプション -Xr，リンカのオプション -lr に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xr -lr)	ROM 化用オブジェクト・ファイルを出力します。
		いいえ	ROM 化用オブジェクト・ファイルを出力しません。
ROM 化用オブジェクト・ファイル出力フォルダ	ROM 化用オブジェクト・ファイルを格納するフォルダを指定します。 ROM 化プロセッサのオプション -o に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択した場合のみ表示されます。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	

ROM 化用オブジェクト・ファイル名	ROM 化用オブジェクト・ファイル名を指定します。 “.out”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.out”が自動的に付加されます。 ROM 化プロセッサのオプション -o に相当します。 なお，本プロパティは，［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択した場合のみ表示されます。	
	デフォルト	romp.out
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(7)［よく使うオプション（ヘキサ・コンバータ）］

ヘキサ・コンバータでよく使うオプションに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，ライブラリ用のプロジェクトの場合は表示されません。

ヘキサ・ファイルを出力する	ヘキサ・ファイルを出力するかどうかを選択します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ヘキサ・ファイルを出力します。
		いいえ	ヘキサ・ファイルを出力しません。
ヘキサ・ファイル出力フォルダ	ヘキサ・ファイルを格納するフォルダを指定します。 ヘキサ・コンバータのオプション -o に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［ヘキサ・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	

ヘキサ・ファイル名	ヘキサ・ファイル名を指定します。 ヘキサ・コンバータのオプション -o に相当します。 拡張子は自由に指定可能です。 埋め込みマクロとして次のマクロ名があります。 %ProjectName% ：プロジェクト名に置換します。 なお，本プロパティは，［ヘキサ・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	%ProjectName%.hex	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	259 文字までの文字列	
ヘキサ・ファイル・フォーマット	生成するヘキサ・ファイルのフォーマットを選択します。 ヘキサ・コンバータのオプション -f に相当します。 なお，本プロパティは，［ヘキサ・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	インテル拡張ヘキサ・フォーマット (-fI)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	インテル拡張ヘキサ・フォーマット (-fI)	生成するヘキサ・ファイルをインテル拡張ヘキサ・フォーマットとします。
		モトローラ S タイプ・フォーマット (スタンダード・アドレス)(-fS)	生成するヘキサ・ファイルをモトローラ S タイプ・フォーマット（スタンダード・アドレス）とします。
		モトローラ S タイプ・フォーマット (32 ビット・アドレス)(-fs)	生成するヘキサ・ファイルをモトローラ S タイプ・フォーマット（32 ビット・アドレス）とします。
		拡張テクトロニクス・ヘキサ・フォーマット (-fT)	生成するヘキサ・ファイルを拡張テクトロニクス・ヘキサ・フォーマットとします。

(8)［よく使うオプション（セクション・ファイル・ジェネレータ）］

セクション・ファイル・ジェネレータでよく使うオプションに関する詳細情報の表示，および設定の変更を行います。

セクション・ファイル・ジェネレータを使用する	セクション・ファイル・ジェネレータを使用するかどうかを選択します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	セクション・ファイル・ジェネレータを使用します。 なお，セクション・ファイルは，ラピッド・ビルドの対象外となります。
		いいえ	セクション・ファイル・ジェネレータを使用しません。

セクション・ファイル出力フォルダ	セクション・ファイルを格納するフォルダを指定します。 セクション・ファイル・ジェネレータのオプション -o に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択した場合のみ表示されます。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列
セクション・ファイル名	セクション・ファイル名を指定します。 “.sf”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.sf”が自動的に付加されます。 セクション・ファイル・ジェネレータのオプション -o に相当します。 埋め込みマクロとして次のマクロ名があります。 %ProjectName%：プロジェクト名に置換します。 なお，本プロパティは，［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択した場合のみ表示されます。	
	デフォルト	%ProjectName%.sf
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(9)［レジスタ・モード］

レジスタ・モードに関する詳細情報の表示，および設定の変更を行います。

レジスタ・モードの選択	ソフトウエア・レジスタ・バンク機能のレジスタ・モード（C コンパイラが使用するレジスタの本数）注を選択します。 コンパイラ，リンカのオプション -reg に相当します。		
	デフォルト	32 レジスタ・モード (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	32 レジスタ・モード (なし)	レジスタ・モードを 32 とします。
		26 レジスタ・モード (-reg26)	レジスタ・モードを 26 とします。
		22 レジスタ・モード (-reg22)	レジスタ・モードを 22 とします。

マスク・レジスタを使用する	r20 レジスタと r21 レジスタをマスク・レジスタとして使用するかどうかを選択します。 コンパイラのオプション -Xmask_reg，アセンブラのオプション -m，リンカのオプション -mask_reg に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xmask_reg, -m, -mask_reg)	r20 レジスタに 8 ビットのマスク値 0xff，r21 レジスタに 16 ビットのマスク値 0xffff が設定されているものとしてコードを出力します。 ［レジスタ・モードの選択］プロパティで［32 レジスタ・モード (なし)］を選択した場合は，マスク・レジスタ機能用のライブラリを参照します。
		いいえ	マスク・レジスタ機能を使用しません。

注 C コンパイラが提供しているレジスタ・モードを示します。

レジスタ・モード	作業用レジスタ	レジスタ変数用レジスタ
22 レジスタ・モード	r10 ～ r14	r25 ～ r29
26 レジスタ・モード	r10 ～ r16	r23 ～ r29
32 レジスタ・モード	r10 ～ r19	r20 ～ r29

（10）［フラッシュ］

フラッシュに関する詳細情報の表示，および設定の変更を行います。

フラッシュ対応オブジェクト・ファイルを生成する	フラッシュ対応オブジェクト・ファイルを生成するかどうかを選択します。 フラッシュ領域側とブート領域側の双方での指定が必要です。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	フラッシュ対応オブジェクト・ファイルを生成します。
		いいえ	フラッシュ対応オブジェクト・ファイルを生成しません。
分岐テーブルのアドレス	分岐テーブルの先頭アドレスを指定します。 フラッシュ領域側とブート領域側の双方で同じアドレスを指定します。 リンカのオプション -ext_table に相当します。 なお，本プロパティは，［フラッシュ対応オブジェクト・ファイルを生成する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	0x0	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xffffffff（16 進数）	

生成するオブジェクト・ファイルの種類	生成するオブジェクト・ファイルの種類を選択します。 コンパイラのオプション -Wa, -zf，アセンブラのオプション -zf，リンカのオプション -zf に相当します。 なお，本プロパティは，［フラッシュ対応オブジェクト・ファイルを生成する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	ブート領域オブジェクト・ファイル (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	ブート領域オブジェクト・ファイル (なし)	ブート領域オブジェクト・ファイルを生成します。
		フラッシュ領域オブジェクト・ファイル (-Wa, -zf)	フラッシュ領域オブジェクト・ファイルを生成します。
ブート領域オブジェクト・ファイル名	ブート領域オブジェクト・ファイル名を指定します。 リンカのオプション -zf に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 なお，本プロパティは，［生成するオブジェクト・ファイルの種類］プロパティで［フラッシュ領域オブジェクト・ファイル (-Wa, -zf)］を選択した場合のみ表示されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，ブート領域オブジェクト・ファイルを指定 ダイアログによる編集	
	指定可能値	259 文字までの文字列	

（11）［デバイス］

デバイスに関する詳細情報の表示，および設定の変更を行います。

メモリ空間を 256M バイトとして扱う	物理アドレス空間を 256M バイトもつデバイスで，64M バイトを越えた 256M バイトまでのアドレス空間を使用するプログラムを作成するかどうかを選択します。 コンパイラ，アセンブラ，リンカのオプション -X256M に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-X256M)	メモリ空間を 256M バイトとして扱います。
		いいえ	メモリ空間を 64M バイトとして扱います。

プログラマブル I/O 領域開始アドレス	プログラマブル I/O 領域の使用と開始アドレスを指定します。 なお，アドレスは 16K バイトでアラインされます。 コンパイラのオプション -Xbpc，アセンブラのオプション -bpc に相当します。 CubeSuite V1.20 未満で保存した値は，設定可能範囲外の場合があります。設定可能範囲外の値を復帰した場合は，本プロパティは空欄となります。 なお，本プロパティは，プログラマブル I/O 機能を持たないデバイスの場合は表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	16 進数（選択したデバイスに依存します）
セキュリティ ID	フラッシュ・メモリ搭載デバイスのセキュリティ ID を指定します。 リンカのオプション -Xsid に相当します。 なお，本プロパティは，セキュリティ ID 機能を持たないデバイスの場合は表示されません。	
	デフォルト	0xffffffffffffffffffff
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0x00000000000000000000 ～ 0xffffffffffffffffffff （20 桁（10 バイト）の 16 進数）

（12）［ビルド方法］

ビルド方法に関する詳細情報の表示，および設定の変更を行います。

インクルード・ファイルが存在しないソースの扱い	ソース・ファイルがインクルードしているファイルが存在しない場合，そのソース・ファイルを再コンパイル／アセンブルするかどうかを選択します。		
	デフォルト	再コンパイル／アセンブルする	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	再コンパイル／アセンブルする	インクルード・ファイルが存在しない場合，ソース・ファイルを再コンパイル／アセンブルします。
		再コンパイル／アセンブルしない	インクルード・ファイルが存在しない場合，ソース・ファイルを再コンパイル／アセンブルしません。

（13）［バージョン選択］

ビルド・ツールのバージョン選択に関する詳細情報の表示，および設定の変更を行います。

使用するコンパイラ・パッケージのインストール・フォルダ	使用するコンパイラ・パッケージがインストールしているフォルダを表示します。	
	デフォルト	*インストール・フォルダ名*
	変更方法	変更不可

使用するコンパイラ・パッケージのバージョン	使用するコンパイラ・パッケージのバージョンを選択します。 この設定はすべてのビルド・モードで共通です。 他の実行環境で作成したプロジェクトを開いた場合など，インストールしていないコンパイラ・パッケージを選択している場合は，そのバージョンも表示します。 コンパイラ・パッケージによってオプションに変更がある場合は，選択したバージョンにあわせて，ビルド・ツールの各プロパティの表示を切り替えます。		
	デフォルト	常にインストール済みの最新版	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	常にインストール済みの最新版	インストールしているコンパイラ・パッケージの内，最新バージョンを使用します。
		インストールしているコンパイラ・パッケージのバージョン	選択したバージョンのコンパイラ・パッケージを使用します。
インストール済みのコンパイラ・パッケージの最新バージョン	［使用するコンパイラ・パッケージのバージョン］プロパティで［常にインストール済みの最新版］を選択した場合に使用するコンパイラ・パッケージのバージョンを表示します。 この設定はすべてのビルド・モードで共通です。 なお，本プロパティは，［使用するコンパイラ・パッケージのバージョン］プロパティで［常にインストール済みの最新版］を選択した場合のみ表示されます。		
	デフォルト	*インストールしているコンパイラ・パッケージの最新バージョン*	
	変更方法	変更不可	

（14）［記録］

記録に関する詳細情報の表示，および設定の変更を行います。

メモ	このビルド・ツールにメモを追加します。 1 行に 1 項目ずつ指定します。 この設定はすべてのビルド・モードで共通です。 追加したメモはサブプロパティとして表示します。	
	デフォルト	メモ [*項目数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

（15）［その他］

ビルド・ツールに関するその他の詳細情報の表示，および設定の変更を行います。

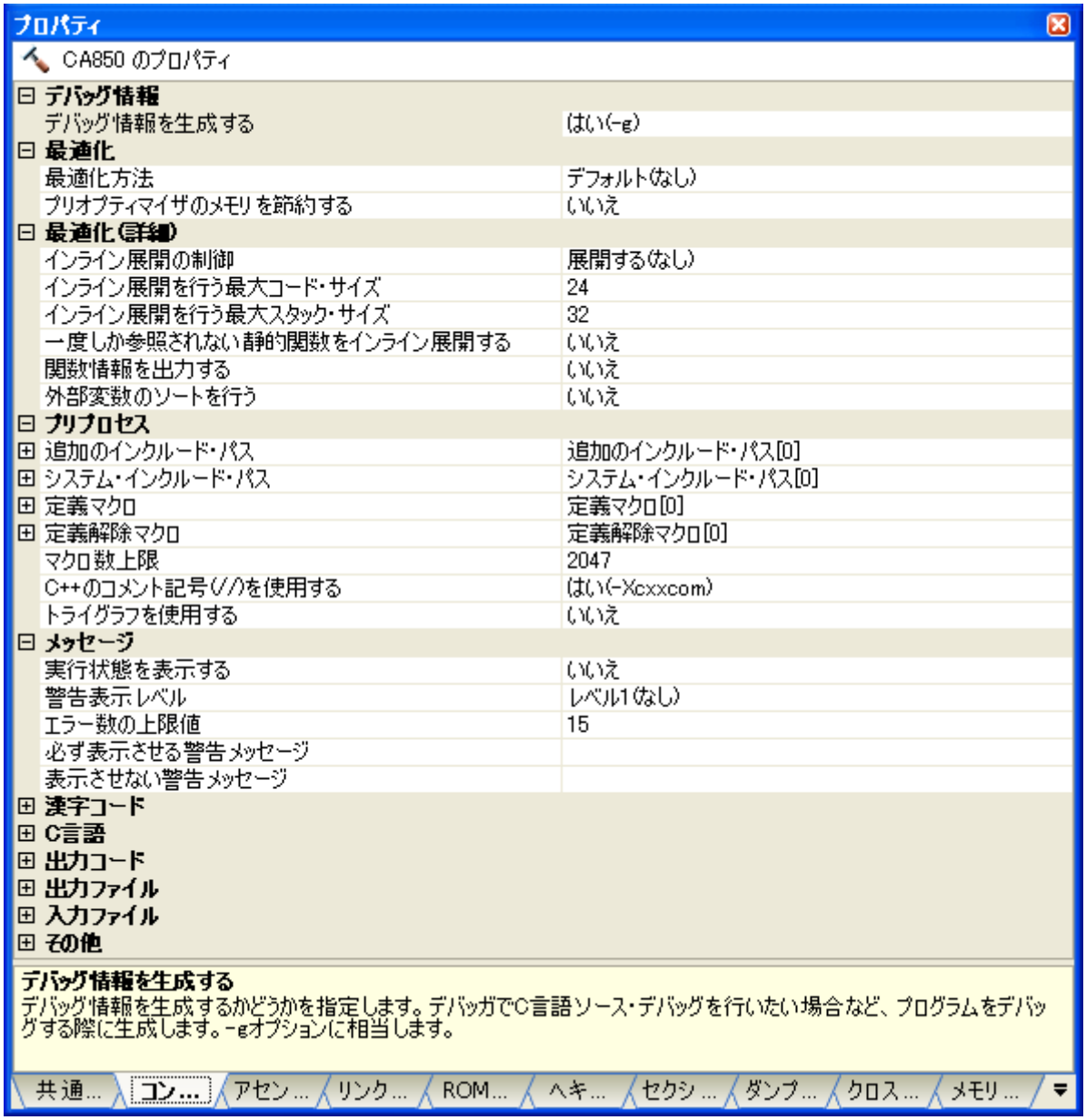
<table>
<tr><td rowspan="5">出力メッセージ・フォーマット</td><td colspan="3">ビルド中のメッセージのフォーマットを指定します。
対象となるのは，使用するビルド・ツール，およびプラグインによって追加されたコマンドの出力メッセージです。
［ビルド前に実行するコマンド］プロパティ，および［ビルド後に実行するコマンド］プロパティなどで指定したコマンドの出力メッセージは対象外です。
埋め込みマクロとして次のマクロ名があります。
%Program% ：実行中のプログラム名に置換します。
%Options% ：ビルド実行時のコマンド・ライン・オプションに置換します。
%FileName% ：ビルド中のファイル名に置換します。
空欄の場合は，%Program% %Options% を指定したものとみなします。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%FileName%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力（256 文字までの文字列），またはドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="3">指定可能値</td><td>%FileName%</td><td>出力メッセージにファイル名を表示します。</td></tr>
<tr><td>%FileName%: %Options%</td><td>出力メッセージにファイル名とコマンド・ライン・オプションを表示します。</td></tr>
<tr><td>%Program% %Options%</td><td>出力メッセージにプログラム名とコマンド・ライン・オプションを表示します。</td></tr>
<tr><td rowspan="4">ビルド・オプション一覧表示フォーマット</td><td colspan="3">ビルド・オプション一覧（「2. 17. 3 ビルド・オプションを一覧表示する」参照）の表示フォーマットを指定します。
対象となるのは，使用するビルド・ツール，およびプラグインによって追加されたコマンドのオプションです。
［ビルド前に実行するコマンド］プロパティ，および［ビルド後に実行するコマンド］プロパティなどで指定したコマンドのオプションは対象外です。
埋め込みマクロとして次のマクロ名があります。
%Program% ：実行中のプログラム名に置換します。
%Options% ：ビルド時のコマンド・ライン・オプションに置換します。
%FileName% ：ビルド中のファイル名に置換します。
空欄の場合は，%FileName% : %Program% %Options% を指定したものとみなします。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%FileName% : %Program% %Options%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">256 文字までの文字列</td></tr>
</table>

一時作業フォルダ	ビルド・ツールの各コマンドが実行中に生成する一時的なファイルの格納先フォルダを指定します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	200 文字までの文字列
ビルド前に実行するコマンド	ビルド処理前に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder%　：出力フォルダの絶対パスに置換します。 %OutputFile%　：出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示します。	
	デフォルト	ビルド前に実行するコマンド [*定義数*]
	変更方法	［...］ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。
ビルド後に実行するコマンド	ビルド処理後に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder%　：出力フォルダの絶対パスに置換します。 %OutputFile%　：出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示します。	
	デフォルト	ビルド後に実行するコマンド [*定義数*]
	変更方法	［...］ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。

［コンパイル・オプション］タブ

本タブでは，コンパイラに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

図 A―5　プロパティ パネル：［コンパイル・オプション］タブ



[各カテゴリの説明]

(1) [デバッグ情報]

デバッグ情報に関する詳細情報の表示，および設定の変更を行います。

デバッグ情報を生成する	ソース・デバッガ用のシンボル情報を出力し，ソース・レベル・デバッグを可能にするかどうかを選択します。 コンパイラのオプション -g に相当します。		
	デフォルト	はい (-g)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-g)	ソース・デバッガ用のシンボル情報を出力します。
		いいえ	ソース・デバッガ用のシンボル情報を出力しません。

(2) [最適化]

最適化に関する詳細情報の表示，および設定の変更を行います。

最適化方法	コンパイルの最適化の種類を選択します。 コンパイラのオプション -O* に相当します。		
	デフォルト	デフォルト (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	デバッグ優先 (-Od)	デバッグを優先して最適化を行います。 ROM 容量や実行速度に着目せず，ソース・デバッグに着眼したコードを生成します。
		デフォルト (なし)	ソース・デバッグに着眼したコードを生成します。 ソース・デバッグに影響のない範囲で最適化を行います。
		標準最適化 (-Og)	適度な最適化を行います。 ほとんどの場合で C ソース・デバッグが可能となる最適化を行います。
		高度な最適化 (-O)	高度な最適化を行います。 ROM 容量に着目した最適化を行います。
		より高度な最適化 (オブジェクト・サイズ優先)(-Os)	より高度な最適化（オブジェクト・サイズ優先）を行います。 ROM 容量を最も重視した最大限の最適化を行います。
		より高度な最適化 (実行速度優先)(-Ot)	より高度な最適化（実行速度優先優先）を行います。 実行速度を最も重視した最大限の最適化を行います。

プリオプティマイザのメモリを節約する	コンパイル時のプリオプティマイザのメモリ使用量を節約するかどうかを選択します。 マシンのメモリが不足し，コンパイルが正常に終了しない場合に，本プロパティを指定します。 コンパイラのオプション -Wp,-D に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wp,-D)	コンパイル時のプリオプティマイザのメモリ使用量を節約します。 ただし，コンパイル速度は低下します。
		いいえ	コンパイル時のプリオプティマイザのメモリ使用量の節約を指定しません。
機種依存最適化部のメモリを節約する	コンパイル時の機種依存最適化部のメモリ使用量を節約するかどうかを選択します。 マシンのメモリが不足し，コンパイルが正常に終了しない場合に，本プロパティを指定します。 コンパイラのオプション -Wi,-D に相当します。 なお，本プロパティは，［最適化方法］プロパティで［デバッグ優先 (-Od)］，［デフォルト (なし)］，［標準最適化 (-Og)］のいずれかを選択した場合は表示されません。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wi,-D)	コンパイル時の機種依存最適化部のメモリ使用量を節約します。 ただし，コンパイル速度は低下します。
		いいえ	コンパイル時の機種依存最適化部のメモリ使用量の節約を指定しません。

(3)［最適化 (詳細)］

最適化に関する詳細情報の表示，および設定の変更を行います。

インライン展開の制御	インライン展開を行うかどうかを選択します。 コンパイラのオプション -Wp,-inline に相当します。		
	デフォルト	展開する (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	展開する (なし)	インライン展開を行います。
		‘inline’関数のみ展開する (-Wp,-inline)	#pragma inline 指定した関数のみインライン展開を行います。
		展開しない (-Wp,-no_inline)	#pragma inline 指定した関数を含む，すべての関数のインライン展開を指定しません。

インライン展開を行う最大コード・サイズ	インライン展開を行う関数の中間言語サイズの最大値を指定します。 指定サイズより大きい関数はインライン展開を行いません。 コンパイラのオプション -Wp,-N に相当します。 サイズの目安は，［関数情報を出力する］プロパティの指定により出力した関数情報ファイルを参照してください。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	- ［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合 128 - ［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］以外を選択した場合 24	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 9999（10 進数）	
インライン展開を行う最大スタック・サイズ	インライン展開を行う関数の中間言語でのスタック・サイズの最大値（バイト）を指定します。 指定サイズより大きい関数はインライン展開を行いません。 コンパイラのオプション -Wp,-G に相当します。 サイズの目安は，［関数情報を出力する］プロパティの指定により出力した関数情報ファイルを参照してください。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	32	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 9999（10 進数）	
一度しか参照されない静的関数をインライン展開する	一度しか参照されない静的関数をインライン展開するかどうかを選択します。 コンパイラのオプション -Wp,-S に相当します。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wp,-S)	一度しか参照されない静的関数のインライン展開を行います。
		いいえ	一度しか参照されない静的関数のインライン展開を指定しません。

<table>
<tr><td rowspan="5">関数情報を出力する</td><td colspan="3">各関数の中間言語でのコード・サイズとスタック・サイズをファイルに出力するかどうかを選択します。
出力する情報は，［インライン展開を行う最大コード・サイズ］プロパティ，および［インライン展開を行う最大スタック・サイズ］プロパティで指定する値の目安となります。
コンパイラのオプション -Wp,-I に相当します。
なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Wp,-I)</td><td>各関数の中間言語でのコード・サイズとスタック・サイズをファイルに出力します。</td></tr>
<tr><td>いいえ</td><td>各関数の中間言語でのコード・サイズとスタック・サイズのファイルへの出力を指定しません。</td></tr>
<tr><td rowspan="4">関数情報ファイル名</td><td colspan="3">各関数の中間言語でのコード・サイズとスタック・サイズを出力するファイル名を指定します。
コンパイラのオプション -Wp,-I に相当します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%：ビルド・モード名に置換します。
空欄の場合は，%BuildModeName%￥FunctionData.txt を指定したものとみなします。
なお，本プロパティは，［関数情報を出力する］プロパティで［いいえ］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%BuildModeName%￥FunctionData.txt</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，関数情報ファイルを指定 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">259 文字までの文字列</td></tr>
<tr><td rowspan="5">ループの展開を行う</td><td colspan="3">for，while などのループの展開を行うかどうかを選択します。
コンパイラのオプション -Wo,-Ol,-Xlo に相当します。
なお，本プロパティは，［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい (展開数は自動で調整)(-Wo,-Ol)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="3">指定可能値</td><td>はい (展開数は自動で調整)(-Wo,-Ol)</td><td>［ループ展開最大数］プロパティで指定した回数以下でコード・サイズが最小になるよう，ループの展開を行います。</td></tr>
<tr><td>はい (展開数は固定)(-Wo,-Ol,-Xlo)</td><td>［ループ展開最大数］プロパティで指定した回数でループの展開を行います。</td></tr>
<tr><td>いいえ (-Wo,-Ol0)</td><td>ループの展開を指定しません。</td></tr>
</table>

<table>
<tr><td rowspan="4">ループ展開最大数</td><td colspan="3">for，while などのループを展開する最大数を指定します。
コンパイラのオプション -Wo,-Ol に相当します。
なお，本プロパティは，［ループの展開を行う］プロパティで［いいえ (-Wo,-Ol0)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">4</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td>指定可能値</td><td colspan="2">0 ～ 999（10 進数）</td></tr>
<tr><td rowspan="5">外部変数のソートを行う</td><td colspan="3">const ／ sconst セクション以外のセクションに配置されている外部変数をアライメントが大きい順に並び替えるかどうかを選択します。
コンパイラのオプション -Wo,-Op に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Wo,-Op)</td><td>const ／ sconst セクション以外のセクションに配置されている外部変数をアライメントが大きい順に並び替えます。</td></tr>
<tr><td>いいえ</td><td>外部変数のアライメントが大きい順への並び替えを指定しません。</td></tr>
<tr><td rowspan="4">外部変数ソート用中間言語ファイル名</td><td colspan="3">外部変数のソート後に作成される中間言語ファイルのファイル名（.ic）を指定します。
各ソース・ファイル内での外部変数のソートでなく，プロジェクト中の外部変数をまとめてソートしたい場合に，本プロパティを指定します。
コンパイラのオプション -Wo,-Op に相当します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
なお，本プロパティは，［外部変数のソートを行う］プロパティで［いいえ］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">空欄</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，外部変数ソート用中間言語ファイルを指定 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">259 文字までの文字列</td></tr>
</table>

<table>
<tr><td rowspan="5">サイズ優先で分岐命令を出力する</td><td colspan="3">分岐命令をコード・サイズ優先で並べてコードを出力するかどうかを選択します。
コンパイラのオプション -Wo,-XFo に相当します。
なお，本プロパティは，［最適化方法］プロパティで［デバッグ優先 (-Od)］，または［デフォルト (なし)］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Wo,-XFo)</td><td>分岐命令をコード・サイズ優先で並べてコードを出力します。</td></tr>
<tr><td>いいえ</td><td>分岐命令に対してデバッグ情報を優先したコードを出力します。</td></tr>
<tr><td rowspan="5">アライメントを詰める</td><td colspan="3">分岐先ラベルを整列する最適化を抑止するかどうかを選択します。
コンパイラのオプション -Wi,-P に相当します。
なお，本プロパティは，［最適化方法］プロパティで［高度な最適化 (-O)］，［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］，または［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合のみ表示されます。
ただし，［高度な最適化 (-O)］，または［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］を選択した場合は，この機能が含まれるため，常に［はい (-Wi,-P)］が選択されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">- ［最適化方法］プロパティで［高度な最適化 (-O)］，または［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］を選択した場合
［はい (-Wi,-P)］
- ［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合
いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Wi,-P)</td><td>分岐先ラベルを整列する最適化を抑止します。
実行コード・サイズを小さくすることができます。</td></tr>
<tr><td>いいえ</td><td>分岐先ラベルを整列する最適化の抑止を指定しません。</td></tr>
<tr><td rowspan="5">強力な最適化を行う</td><td colspan="3">データ・フロー解析を厳密に行い，最も強力な最適化を行うかどうかを選択します。
高度な最適化を行う場合に，さらに強力なデータ・フロー解析を行いたい場合に，本プロパティを指定します。
コンパイラのオプション -Wi,-O4 に相当します。
なお，本プロパティは，［最適化方法］プロパティで［高度な最適化 (-O)］，［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］，または［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Wi,-O4)</td><td>データ・フロー解析を厳密に行い，最も強力な最適化を行います。
ただし，コンパイル速度はかなり低下します。</td></tr>
<tr><td>いいえ</td><td>強力な最適化を指定しません。</td></tr>
</table>

(4)［プリプロセス］

プリプロセスに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="4">追加のインクルード・パス</td><td colspan="2">コンパイル時の追加のインクルード・パスを指定します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%　：ビルド・モード名に置換します。
%ProjectName%　　：プロジェクト名に置換します。
%MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。
このオプションを省略した場合，コンパイラの標準フォルダのみ検索します。なお，パスはプロジェクト・フォルダを基点とします。
コンパイラのオプション -I に相当します。
指定したインクルード・パスはサブプロパティとして表示されます。
なお，プロジェクト・ツリーにインクルード・ファイルを追加すると，そのインクルード・パスをサブプロパティの最初に追加します。
インクルード・パスに大文字，小文字の区別はありません。</td></tr>
<tr><td>デフォルト</td><td>追加のインクルード・パス [定義数]</td></tr>
<tr><td>変更方法</td><td>[...] ボタンをクリックし，パス編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td>259 文字までの文字列
64 個まで指定可能です。ただし，連携するツールが使用するパスの数も含みます。</td></tr>
<tr><td rowspan="4">システム・インクルード・パス</td><td colspan="2">コンパイル時にシステムが設定するインクルード・パスを表示します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName%　：ビルド・モード名に置換します。
%ProjectName%　　：プロジェクト名に置換します。
%MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。
システム・インクルード・パスは，追加のインクルード・パスより低い優先度で検索します。
パスはプロジェクト・フォルダを基点とします。
コンパイラのオプション -I に相当します。
インクルード・パスはサブプロパティとして表示します。</td></tr>
<tr><td>デフォルト</td><td>システム・インクルード・パス [定義数]</td></tr>
<tr><td>変更方法</td><td>[...] ボタンをクリックし，システム・インクルード・パス順設定 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td>変更不可（インクルード・パスの設定順の変更のみ可能）</td></tr>
</table>

<table>
<tr><td rowspan="4">定義マクロ</td><td colspan="3">定義したいマクロ名を指定します。
「マクロ名 = 定義値」の形式で 1 行に 1 つずつ指定します。「= 定義値」の部分は省略可能で，省略した場合，定義値を 1 とします。
コンパイラのオプション -D に相当します。
指定したマクロはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">定義マクロ [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">256 文字までの文字列
256 個まで指定可能です。</td></tr>
<tr><td rowspan="4">定義解除マクロ</td><td colspan="3">定義解除したいマクロ名を指定します。
「マクロ名」の形式で 1 行に 1 つずつ指定します。
コンパイラのオプション -U に相当します。
指定したマクロはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">定義解除マクロ [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">256 文字までの文字列
256 個まで指定可能です。</td></tr>
<tr><td rowspan="3">マクロ数上限</td><td colspan="3">マクロ識別子の上限を指定します。
コンパイラのオプション -Xm に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">2047</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td>指定可能値</td><td colspan="2">1 ～ 999999（10 進数）</td></tr>
<tr><td rowspan="4">C++ のコメント記号 (//) を使用する</td><td colspan="3">通常のコメントのほかに，C++ のコメント・スタイル（“//”から行末まで）を有効にするかどうかを選択します。
コンパイラのオプション -Xcxxcom に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい (-Xcxxcom)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td>指定可能値</td><td>はい (-Xcxxcom)</td><td>通常のコメントのほかに，C++ のコメント・スタイル（“//”から行末まで）を有効にします。</td></tr>
<tr><td></td><td>いいえ</td><td>C++ のコメント・スタイル（“//”から行末まで）を有効にしません。</td></tr>
</table>

プリプロセッサの出力ファイルにコメントを保存する	C 言語のソース・プログラムの前処理の出力に，ソース・プログラムのコメントを含めるかどうかを選択します。 コンパイラのオプション -C に相当します。 なお，本プロパティは，［出力ファイル］カテゴリの［プリプロセス処理したソースを出力する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-C)	C 言語のソース・プログラムの前処理の出力に，ソース・プログラムのコメントを含めます。
		いいえ	C 言語のソース・プログラムの前処理の出力に，ソース・プログラムのコメントを含めません。
トライグラフを使用する	トライグラフ系列を置換するかどうかを選択します。 トライグラフとは，ANSI 規格で規定された，単一文字に置換される 3 文字（トライグラフ）系列のことです。 コンパイラのオプション -t に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-t)	トライグラフ系列を置換します。
		いいえ	トライグラフ系列を置換しません。

(5)［メッセージ］

メッセージに関する詳細情報の表示，および設定の変更を行います。

実行状態を表示する	ビルド時にコンパイラの実行状態を出力 パネルに表示するかどうかを選択します。 コンパイラのオプション -v に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-v)	ビルド時にコンパイラの実行状態を表示します。
		いいえ	ビルド時にコンパイラの実行状態を表示しません。
警告表示レベル	コンパイル時の警告表示レベルを選択します。 コンパイラのオプション -w に相当します。		
	デフォルト	レベル 1(なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	出力しない (-w)	警告メッセージを出力しません。
		レベル 1(なし)	通常の警告メッセージを出力します。
		レベル 2(-w2)	詳細な警告メッセージを出力します。
エラー数の上限値	エラー・メッセージの最大出力数を指定します。 コンパイラのオプション -err_limit に相当します。		
	デフォルト	15	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	15 ～ 50（10 進数）	

必ず表示させる警告メッセージ	［警告表示レベル］プロパティの設定にかかわらず，表示させたい警告メッセージの番号を指定します。 複数指定する場合は，メッセージ番号をカンマで区切って指定します（例：2042,2107）。また，ハイフンを使用して，区間設定を行うこともできます（例：2222-2554,2699-2782）。 ［表示させない警告メッセージ］プロパティと同一の番号を指定した場合は，本プロパティを優先します。 コンパイラのオプション -won に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，ビルド時の警告メッセージ設定 ダイアログによる編集
	指定可能値	2048 文字までの文字列
表示させない警告メッセージ	［警告表示レベル］プロパティの設定にかかわらず，表示させない警告メッセージの番号を指定します。 複数指定する場合は，メッセージ番号をカンマで区切って指定します（例：2042,2107）。また，ハイフンを使用して，区間設定を行うこともできます（例：2222-2554,2699-2782）。 ［必ず表示させる警告メッセージ］プロパティと同一の番号を指定した場合は，［必ず表示させる警告メッセージ］プロパティを優先します。 コンパイラのオプション -woff に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，ビルド時の警告メッセージ設定 ダイアログによる編集
	指定可能値	2048 文字までの文字列

（6）［漢字コード］

漢字コードに関する詳細情報の表示，および設定の変更を行います。

ソースの漢字コード	入力ファイル中の日本語コメント，文字列に対し，使用する漢字コードを選択します。 コンパイラのオプション -Xk に相当します。		
	デフォルト	Shift_JIS(なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	Shift_JIS(なし)	ソースの漢字コードを Shift_JIS と解釈します。
		なし (-Xk=none)	ソースに漢字コードがないと解釈します。 コードを保証しません。
		EUC-JP(-Xk=euc)	ソースの漢字コードを EUC-JP と解釈します。

ターゲットの漢字コード	日本語の文字列に対し，変換する漢字コードを選択します。 アプリケーション開発時に使用した漢字コードをターゲットで変更したい場合に，本プロパティを設定します。 コンパイラのオプション -Xkt に相当します。		
	デフォルト	無変換 (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	無変換 (なし)	ターゲットの漢字コードの変換を行いません。 コードを保証しません。
		Shift_JIS(-Xkt=sjis)	ターゲットの漢字コードを Shift_JIS に変換します。
		EUC-JP(-Xkt=euc)	ターゲットの漢字コードを EUC-JP に変換します。

（7）［C 言語］

C 言語に関する詳細情報の表示，および設定の変更を行います。

ビット・フィールドの符号	型指定子（signed，unsigned）の付かない int 型のビット・フィールドに対し，符号付きとするか，符号なしとするかを選択します。 コンパイラのオプション -Xbitfield に相当します。		
	デフォルト	符号付き	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	符号付き	型指定子の付かない int 型のビット・フィールドに対し，符号付きとします。
		符号なし (-Xbit-field=unsigned)	型指定子の付かない int 型のビット・フィールドに対し，符号なしとします。
char の符号	型指定子（signed，unsigned）の付かない char 型に対し，符号付きとするか，符号なしとするかを選択します。 コンパイラのオプション -Xchar に相当します。		
	デフォルト	符号付き	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	符号付き	型指定子の付かない char 型に対し，符号付きとします。
		符号なし (-Xchar=unsigned)	型指定子の付かない char 型に対し，符号なしとします。

enum の型	列挙型に対し，どの整数型と整合するかを選択します。 コンパイラのオプション -Xenum_type に相当します。		
	デフォルト	int(なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	int(なし)	列挙型に対し，int 型と整合します。
		signed char(-Xenum_type=char)	列挙型に対し，signed char 型と整合します。
		unsigned char(-Xenum_type=uchar)	列挙型に対し，unsigned char 型と整合します。
		short(-Xenum_type=short)	列挙型に対し，short 型と整合します。
		unsigned short(-Xenum_type=ushort)	列挙型に対し，unsigned short 型と整合します。
ANSI 規格に厳密に合わせてコンパイルする	コンパイラの処理を ANSI 規格に厳密に合わせ，規格に反する記述に対してエラーや警告メッセージを表示するかどうかを選択します。 コンパイラのオプション -ansi に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-ansi)	コンパイラの処理を ANSI 規格に厳密に合わせ，規格に反する記述に対してエラーや警告メッセージを表示します。
		いいえ	従来の C 言語の仕様との両立性を持たせ，警告メッセージを表示してコンパイラの処理を続行します。
CC78K の拡張機能を使用する	78K マイクロコントローラ C コンパイラ CC78K 互換の拡張機能を有効にするかどうかを選択します。 コンパイラのオプション -cc78k に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-cc78k)	CC78K 互換の拡張機能を有効にします。
		いいえ	CC78K 互換の拡張機能を有効にしません。
整数演算を厳密に行う	ANSI 規格に厳密な乗除算処理を行うため，16 ビット・データ以下の整数に対し，mulh，divh 命令を使用せず，ランタイム・ライブラリ ___mul ／ ___mulu，___div ／ ___divu，または mul，mulu，div，divu 命令を使用するかどうかを選択します。 コンパイラのオプション -Xe に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xe)	16 ビット・データ以下の整数に対し，ランタイム・ライブラリ ___mul ／ ___mulu，___div ／ ___divu を使用します。
		いいえ	16 ビット・データ以下の整数に対し，mulh，divh 命令を使用します。

仮定義を定義に変更する	変数の仮定義を定義として扱うかどうかを選択します。 コンパイラのオプション -Xdefvar に相当します。		
	デフォルト	はい (-Xdefvar)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xdefvar)	変数の仮定義を定義として扱います。
		いいえ	変数の仮定義を定義として扱いません。

(8)［出力コード］

出力コードに関する詳細情報の表示，および設定の変更を行います。

sdata/sbss セクションに配置するデータ長の上限値 (バイト)	.sdata/.sbss セクションに配置するデータ長の上限サイズ（バイト）を指定します。 ただし，#pragma section 指令やセクション・ファイルで .sdata/.sbss セクションが指定されたデータは，そのサイズに関係なく，.sdata/.sbss セクションに配置します。 コンパイラのオプション -G に相当します。 なお，本プロパティを変更した場合，［アセンブル・オプション］タブの［その他］カテゴリの［sdata/sbss セクションに配置するデータ長の上限値 (バイト)] プロパティの値も連動して変更されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 32767（10 進数）	
sconst セクションにデータを配置する	const 属性のデータ，文字列リテラルを .sconst セクションに配置するかどうかを選択します。 コンパイラのオプション -Xsconst に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xsconst)	const 属性のデータ，文字列リテラルを .sconst セクションに配置します。
		いいえ	const 属性のデータ，文字列リテラルを .const セクションに配置します。
sconst セクションのデータ長の上限値 (バイト)	.const 属性のデータ，文字列リテラルを .sconst セクションに配置する上限サイズ（バイト）を指定します。 ただし，#pragma section 指令やセクション・ファイルで .sconst セクションが指定されたデータは，そのサイズに関係なく，.sconst セクションに配置します。 コンパイラのオプション -Xsconst に相当します。 なお，本プロパティは，［sconst セクションにデータを配置する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	32767	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 32767（10 進数）	

プロローグ／エピローグ・ライブラリを使用する	関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にするかどうかを選択します。 コンパイラのオプション -Xpro_epi_runtime に相当します。		
	デフォルト	自動選択 (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	自動選択 (なし)	［最適化］カテゴリの［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合は［いいえ (-Xpro_epi_runtime=off)］，それ以外を選択した場合は［はい (-Xpro_epi_runtime=on)］となります。
		いいえ (-Xpro_epi_runtime=off)	関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にしません。
		はい (-Xpro_epi_runtime=on)	関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にします。
switch 文の出力コードの選択	プログラム中の switch 文のコード出力方式を選択します。 コンパイラのオプション -Xcase に相当します。		
	デフォルト	自動選択 (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	自動選択 (なし)	コンパイラが最適と思われる形式を自動判断します。
		if-else(-Xcase=ifelse)	プログラム中の switch 文を case 文の並びに沿った if-else 文と同じ形で出力します。 上から順に比較するので，合致する頻度が多い順に case 文を書いているときやラベル数が少ないときは余計な比較が減り，実行速度の向上につながります。
		バイナリ・サーチ (-Xcase=binary)	プログラム中の switch 文をバイナリ・サーチ形式で出力します。 バイナリ・サーチ・アルゴリズムに用いて合致する case 文を探すため，ラベル数が多いときは，どの case 文も同じくらいの速さで見つけることができます。
		テーブル分岐 (-Xcase=table)	プログラム中の switch 文をテーブル・ジャンプ方式で出力します。 case 文の値を基にインデックス化したテーブルを参照し，switch 文の値により case ラベルを選択し，処理を行います。どの case 文にも同じくらい速く分岐します。ただし，case 値が連続していないときは無駄な領域ができます。

switch テーブルのラベルサイズ (バイト)	switch 文の case ラベルに対する分岐テーブルの 1 ラベルあたりのサイズを選択します。 コンパイラのオプション -Xword_switch に相当します。		
	デフォルト	2 バイト (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	2 バイト (なし)	switch 文の case ラベルに対する分岐テーブルを 1 ラベルあたり 2 バイトで生成します。
		4 バイト (-Xword_switch)	switch 文の case ラベルに対する分岐テーブルを 1 ラベルあたり 4 バイトで生成します。 長い switch 文のため，コンパイル・エラーとなる場合に選択します。
構造体パッキング	構造体のパッキング値を選択します。 構造体メンバをメンバ型に応じてアライメントすることなく，指定したアライメント値を用いることができます。データ・サイズは小さくすることができますが，コード・サイズは大きくなります。 コンパイラのオプション -Xpack に相当します。		
	デフォルト	8 バイト (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	1 バイト (-Xpack=1)	構造体メンバを 1 バイトのアライメントで整列します。
		2 バイト (-Xpack=2)	構造体メンバを 2 バイトのアライメントで整列します。
		4 バイト (-Xpack=4)	構造体メンバを 4 バイトのアライメントで整列します。
		8 バイト (なし)	構造体メンバを 8 バイトのアライメントで整列します。
strcpy ／ strcmp の展開を行う	配列（文字列を含む），および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開するかどうかを選択します。 オブジェクトの実行速度は高速になりますが，コード・サイズは増大します。 コンパイラのオプション -Xi に相当します。 なお，本プロパティは，［構造体パッキング］プロパティで［8 バイト (なし)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xi)	配列（文字列を含む），および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開します。
		いいえ	関数 strcpy()，または strcmp() の呼び出しをインライン展開しません。

ポインタのバイトアクセスを行う	構造体の間接アドレス・アクセスをバイト単位でアクセスするかどうかを選択します。 構造体パッキング機能で，制限に該当するような場合に，本プロパティを設定します。 コンパイラのオプション -Xbyte に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xbyte)	構造体の間接アドレス・アクセスをバイト単位でアクセスします。
		いいえ	構造体の間接アドレス・アクセスをバイト単位でアクセスしません。
アセンブリ言語ソースにコメントを出力する	出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力するかどうかを選択します。 コンパイラのオプション -Xc に相当します。 なお，本プロパティは，［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択した場合，または［アセンブル・リストを出力する］プロパティで［はい (-Fv)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xc)	出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力します。
		いいえ	出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力しません。
割り込みの分岐命令に jmp 命令を使う	C 言語で定義された割り込み関数に対し，jmp 命令を用いるかどうかを選択します。 コンパイラのオプション -Xj に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xj)	C 言語で定義された割り込み関数に対し，jmp 命令を用います。
		いいえ	C 言語で定義された割り込み関数に対し，jr 命令を用います。
word の bit 命令変更を抑止する	ld.w/ld.h，st.w/st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を禁止するかどうかを選択します。 コンパイラのオプション -Xno_word_bitop に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xno_word_bitop)	ld.w/ld.h，st.w/st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を禁止します。
		いいえ	ld.w/ld.h，st.w/st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を行います。

(9)［出力ファイル］

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

アセンブリ・ファイルを出力する	C ソースのコンパイル結果のアセンブラ・ソース・ファイルを出力するかどうかを選択します。 コンパイラのオプション -Fs に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Fs)	アセンブラ・ソース・ファイルを出力します。
		いいえ	アセンブラ・ソース・ファイルを出力しません。
アセンブリ・ファイルの出力フォルダ	アセンブラ・ソース・ファイルの出力先フォルダを指定します。 アセンブラ・ソース・ファイルは，ソース・ファイルの拡張子を .s で置き換えたファイル名で保存します。 コンパイラのオプション -Fs に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% ：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択した場合のみ表示されます。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	
アセンブル・リストを出力する	C ソースのコンパイル結果のアセンブル・リストを出力するかどうかを選択します。 コンパイラのオプション -Fv に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Fv)	アセンブル・リストを出力します。
		いいえ	アセンブル・リストを出力しません。

<table>
<tr><td rowspan="4">アセンブル・リストの出力フォルダ</td><td colspan="3">アセンブル・リストの出力先フォルダを指定します。
アセンブル・リストは，ソース・ファイルの拡張子を .v で置き換えたファイル名で保存します。
コンパイラのオプション -Fv に相当します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
空欄の場合は，プロジェクト・フォルダを指定したものとみなします。
なお，本プロパティは，［アセンブル・リストを出力する］プロパティで［はい (-Fv)］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">%BuildModeName%</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
<tr><td rowspan="4">頻度情報ファイルを出力する</td><td colspan="3">セクション・ファイル・ジェネレータで使用する変数の頻度情報ファイルを出力するかどうかを選択します。
コンパイラのオプション -Xcre_sec_data に相当します。
なお，本プロパティは，［セクション・ファイル・ジェネレート・オプション］タブの［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Xcre_sec_data)</td><td>変数の頻度情報ファイルを出力します。</td></tr>
<tr><td>いいえ</td><td>変数の頻度情報ファイルを出力しません。</td></tr>
</table>

頻度情報ファイルの出力フォルダ	頻度情報ファイルの出力先フォルダを指定します。 頻度情報ファイルは，ソース・ファイルの拡張子を .sec で置き換えたファイル名で保存します。 コンパイラのオプション -Xcre_sec_data に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% ：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［頻度情報ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	
プリプロセス処理したソースを出力する	C ソース・プログラムに対し，前処理（プリプロセス処理）のみ実行するコマンドをコンパイル前に実行するかどうかを選択します。 結果は，C ソース・ファイルの拡張子 .c を .i で置き換えたファイル名で出力します。 ただし，ソース・プログラムの行番号表示やファイル名表示は出力しません。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-P)	C ソース・プログラムに対し，プリプロセス処理のみ実行した結果を出力します。
		いいえ	C ソース・プログラムに対し，プリプロセス処理のみ実行した結果を出力しません。

（10）［入力ファイル］

入力ファイルに関する詳細情報の表示，および設定の変更を行います。

セクション・ファイル名	コンパイラ起動時にグローバル変数／スタティック変数を配置するセクションを定義するためのセクション・ファイル名を表示します。 プロジェクトに登録している有効なセクション・ファイルを検索して使用します。 コンパイラのオプション -Xsec_file に相当します。 指定したセクション・ファイル名はサブプロパティとして表示されます。 なお，本プロパティは，［セクション・ファイル・ジェネレート・オプション］タブの［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択した場合は表示されません。	
	デフォルト	セクション・ファイル名 [*プロジェクトに登録している有効なセクション・ファイル数*]
	変更方法	変更不可

Far Jump ファイル名	Far Jump ファイル名を指定します。 Far Jump ファイルには，ファイルに記述された関数への分岐に対して，jmp 命令を使用したコードを出力します。関数本体が jarl，および jr 命令では分岐できない範囲（± 2M バイト以上）にあり，リンカがエラーを出力する場合，このオプションを用いてコンパイルし直します。 なお，拡張子は“.fjp”としてください。 コンパイラのオプション -Xfar_jump に相当します。 指定した Far Jump ファイル名はサブプロパティとして表示されます。	
	デフォルト	Far Jump ファイル名 [*設定数*]
	変更方法	[...] ボタンをクリックし，Far Jump ファイルを指定 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	259 文字までの文字列 5000 個まで指定可能です。

（11）［外部変数レジスタ］

外部変数レジスタに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで［32 レジスタ・モード (なし)］を選択した場合は表示されません。

r15 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r15 に相当します。 なお，本プロパティは，［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで［26 レジスタ・モード (-reg26)］を選択した場合は表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r16 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r16 に相当します。 なお，本プロパティは，［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで［26 レジスタ・モード (-reg26)］を選択した場合は，表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r17 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r17 に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列

r18 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r18 に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r19 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r19 に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r20 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r20 に相当します。 なお，本プロパティは，［共通オプション］タブの［レジスタ・モード］カテゴリの［マスク・レジスタを使用する］プロパティで［はい (-Xmask_reg, -m, -mask_reg)］を選択した場合は，表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r21 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r21 に相当します。 なお，本プロパティは，［共通オプション］タブの［レジスタ・モード］カテゴリの［マスク・レジスタを使用する］プロパティで［はい (-Xmask_reg, -m, -mask_reg)］を選択した場合は，表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r22 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r22 に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列
r23 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r23 に相当します。 なお，本プロパティは，［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで［26 レジスタ・モード (-reg26)］を選択した場合は，表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列

r24 レジスタに割り当てる外部変数	レジスタに割り付ける外部変数（“_”を除くシンボル名）を指定します。 コンパイラのオプション -r24 に相当します。 なお，本プロパティは，［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで［26 レジスタ・モード (-reg26)］を選択した場合は，表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1022 文字までの文字列

（12）［その他］

コンパイルに関するその他の詳細情報の表示，および設定の変更を行います。

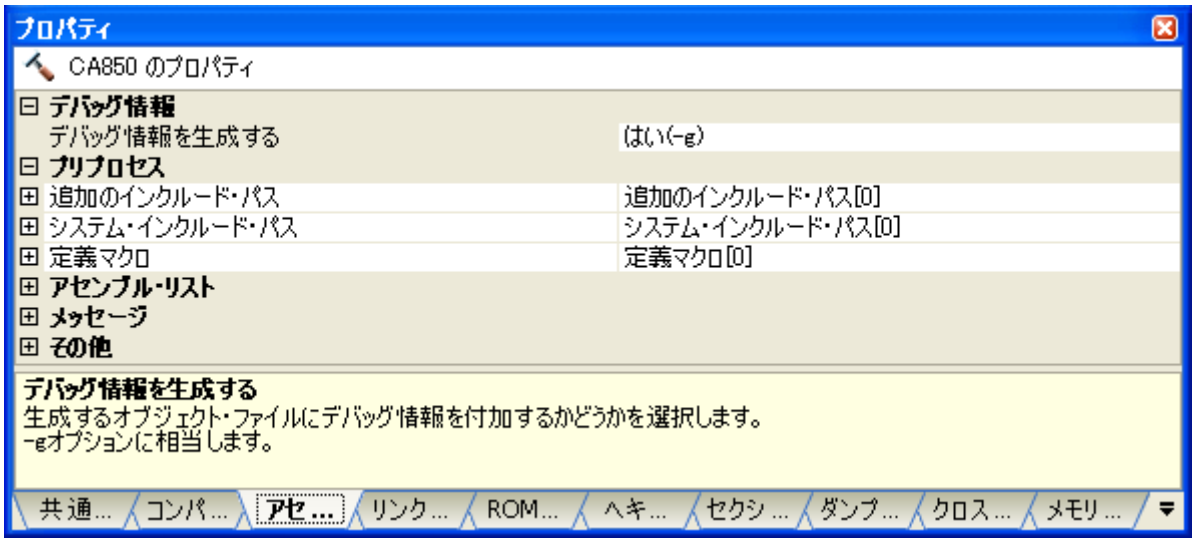
コンパイル前に実行するコマンド	コンパイル処理前に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder% ：出力フォルダの絶対パスに置換します。 %OutputFile% ：出力ファイルの絶対パスに置換します。 %InputFile% ：コンパイル対象ファイルの絶対パスに置換します。 %CompiledFile% ：コンパイル時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。	
	デフォルト	コンパイル前に実行するコマンド [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。
コンパイル後に実行するコマンド	コンパイル処理後に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder% ：出力フォルダの絶対パスに置換します。 %OutputFile% ：出力ファイルの絶対パスに置換します。 %InputFile% ：コンパイル対象ファイルの絶対パスに置換します。 %CompiledFile% ：コンパイル時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。	
	デフォルト	コンパイル後に実行するコマンド [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。

その他の追加オプション	その他に追加するコンパイラのオプションを入力します。 なお，ここで設定したオプションは，コンパイラのオプション群の最後に付加されます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	259 文字までの文字列

［アセンブル・オプション］タブ

本タブでは，アセンブラに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

図 A―6　プロパティ パネル：［アセンブル・オプション］タブ



［各カテゴリの説明］

（1）［デバッグ情報］

デバッグ情報に関する詳細情報の表示，および設定の変更を行います。

デバッグ情報を生成する	生成するオブジェクト・ファイルにデバッグ情報を付加し，ソース・レベル・デバッグを可能にするかどうかを選択します。 アセンブラのオプション -g に相当します。		
	デフォルト	はい (-g)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-g)	生成するオブジェクト・ファイルにデバッグ情報を付加します。
		いいえ	生成するオブジェクト・ファイルにデバッグ情報を付加しません。

(2) [プリプロセス]

プリプロセスに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td>追加のインクルード・パス</td><td colspan="2">アセンブル時の追加のインクルード・パスを指定します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ： ビルド・モード名に置換します。
%ProjectName% ： プロジェクト名に置換します。
%MicomToolPath% ： 本製品のインストール・フォルダの絶対パスに置換します。
このオプションを省略した場合，アセンブラの標準フォルダのみ検索します。なお，パスはプロジェクト・フォルダを基点とします。
アセンブラのオプション -I に相当します。
指定したインクルード・パスはサブプロパティとして表示されます。
なお，プロジェクト・ツリーにインクルード・ファイルを追加すると，そのインクルード・パスをサブプロパティの最初に追加します。
インクルード・パスに大文字，小文字の区別はありません。</td></tr>
<tr><td></td><td>デフォルト</td><td>追加のインクルード・パス [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>[...] ボタンをクリックし，パス編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td>259 文字までの文字列
64 個まで指定可能です。ただし，連携するツールが使用するパスの数も含みます。</td></tr>
<tr><td>システム・インクルード・パス</td><td colspan="2">アセンブル時にシステムが設定するインクルード・パスを表示します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ： ビルド・モード名に置換します。
%ProjectName% ： プロジェクト名に置換します。
%MicomToolPath% ： 本製品のインストール・フォルダの絶対パスに置換します。
システム・インクルード・パスは，追加のインクルード・パスより低い優先度で検索します。
パスはプロジェクト・フォルダを基点とします。
アセンブラのオプション -I に相当します。
インクルード・パスはサブプロパティとして表示します。</td></tr>
<tr><td></td><td>デフォルト</td><td>システム・インクルード・パス [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>[...] ボタンをクリックし，システム・インクルード・パス順設定 ダイアログによる編集</td></tr>
<tr><td></td><td>指定可能値</td><td>変更不可（インクルード・パスの設定順の変更のみ可能）</td></tr>
</table>

定義マクロ	定義したいマクロ名を指定します。 「*マクロ名* = *定義値*」の形式で 1 行に 1 つずつ指定します。「= *定義値*」の部分は省略可能で，省略した場合，定義値を 1 とします。 アセンブラのオプション -D に相当します。 指定したマクロはサブプロパティとして表示されます。	
	デフォルト	定義マクロ [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

(3) [アセンブル・リスト]

アセンブル・リストに関する詳細情報の表示，および設定の変更を行います。

アセンブル・リスト・ファイルを出力する	アセンブル・リスト・ファイルを出力するかどうかを選択します。 アセンブラのオプション -a -l に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-a -l)	アセンブル・リスト・ファイルを出力します。
		いいえ	アセンブル・リスト・ファイルを出力しません。
アセンブル・リスト・ファイル出力フォルダ	アセンブル・リスト・ファイルの出力先フォルダを指定します。 アセンブル・リスト・ファイルは，アセンブラ・ソース・ファイルの拡張子 .s を .v で置き換えたファイル名で保存します。 アセンブラのオプション -l に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% : ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，[アセンブル・リスト・ファイルを出力する] プロパティで [はい (-a -l)] を選択した場合のみ表示されます。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または [...] ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	

(4) [メッセージ]

メッセージに関する詳細情報の表示，および設定の変更を行います。

実行状況を表示する	ビルド時にアセンブラの実行状況を出力 パネルに表示するかどうかを選択します。 アセンブラのオプション -v に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-v)	ビルド時にアセンブラの実行状況を表示します。
		いいえ	ビルド時にアセンブラの実行状況を表示しません。
r0 のデスティネーション・レジスタ使用を警告する	r0 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示するかどうかを選択します。 アセンブラのオプション -wr0+，-wr0- に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-wr0+)	r0 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示します。 本プロパティの設定は，［警告を表示する］プロパティより優先します。
		いいえ (-wr0-)	r0 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示しません。 本プロパティの設定は，［警告を表示する］プロパティより優先します。
		いいえ	［警告を表示する］プロパティの設定に従います。
r1 レジスタ使用を警告する	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定しているとき，警告を表示するかどうかを選択します。 アセンブラのオプション -wr1+，-wr1- に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-wr1+)	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定しているとき，警告を表示します。 本プロパティの設定は，［警告を表示する］プロパティより優先します。
		いいえ (-wr1-)	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定しているとき，警告を表示しません。 本プロパティの設定は，［警告を表示する］プロパティより優先します。
		いいえ	［警告を表示する］プロパティの設定に従います。

警告を表示する	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定したとき，r0 レジスタをデスティネーション・レジスタとして指定したとき，およびマスク・レジスタ機能使用時に r20 レジスタ，または r21 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示するかどうかを選択します。 アセンブラのオプション -w に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定したとき，r0 レジスタをデスティネーション・レジスタとして指定したとき，およびマスク・レジスタ機能使用時に r20 レジスタ，または r21 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示します。
		いいえ (-w)	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定したとき，r0 レジスタをデスティネーション・レジスタとして指定したとき，およびマスク・レジスタ機能使用時に r20 レジスタ，または r21 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示しません。

(5) [その他]

アセンブルに関するその他の詳細情報の表示，および設定の変更を行います。

sdata/sbss セクションに配置するデータ長の上限値 (バイト)	.sdata/.sbss セクションに配置するデータ長の上限（バイト）を指定します。 アセンブラのオプション -G に相当します。 なお，本プロパティを変更した場合，[コンパイル・オプション] タブの [出力コード] カテゴリの [sdata/sbss セクションに配置するデータ長の上限値 (バイト)] プロパティの値も連動して変更されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 32767（10 進数）	
最適化を行う	命令を並べ替えて，レジスタ／フラグのハザードを回避する最適化を行うかどうかを選択します。 アセンブラのオプション -O に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-O)	レジスタ／フラグのハザードを回避する最適化を行います。
		いいえ	レジスタ／フラグのハザードを回避する最適化を行いません。

<table>
<tr><td rowspan="4">32 ビット分岐命令を使用する</td><td colspan="3">命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定するかどうかを選択します。
アセンブラのオプション -Xfar_jump に相当します。
なお，本プロパティは，デバイスの品種に V850E2 コアのデバイスを指定している場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい (-Xfar_jump)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td>指定可能値</td><td>はい (-Xfar_jump)</td><td>命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定します。</td></tr>
<tr><td></td><td>いいえ</td><td>命令に 22/32 を記述しない分岐命令（jarl，jr）は通常分岐命令となります。</td></tr>
<tr><td rowspan="4">アセンブル前に実行するコマンド</td><td colspan="3">アセンブル処理前に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%InputFile%　：アセンブル対象ファイルの絶対パスに置換します。
%AssembledFile%　：アセンブル時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">アセンブル前に実行するコマンド [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td rowspan="4">アセンブル後に実行するコマンド</td><td colspan="3">アセンブル処理後に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%InputFile%　：アセンブル対象ファイルの絶対パスに置換します。
%AssembledFile%　：アセンブル時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">アセンブル後に実行するコマンド [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">1023 文字までの文字列
64 個まで指定可能です。</td></tr>
</table>

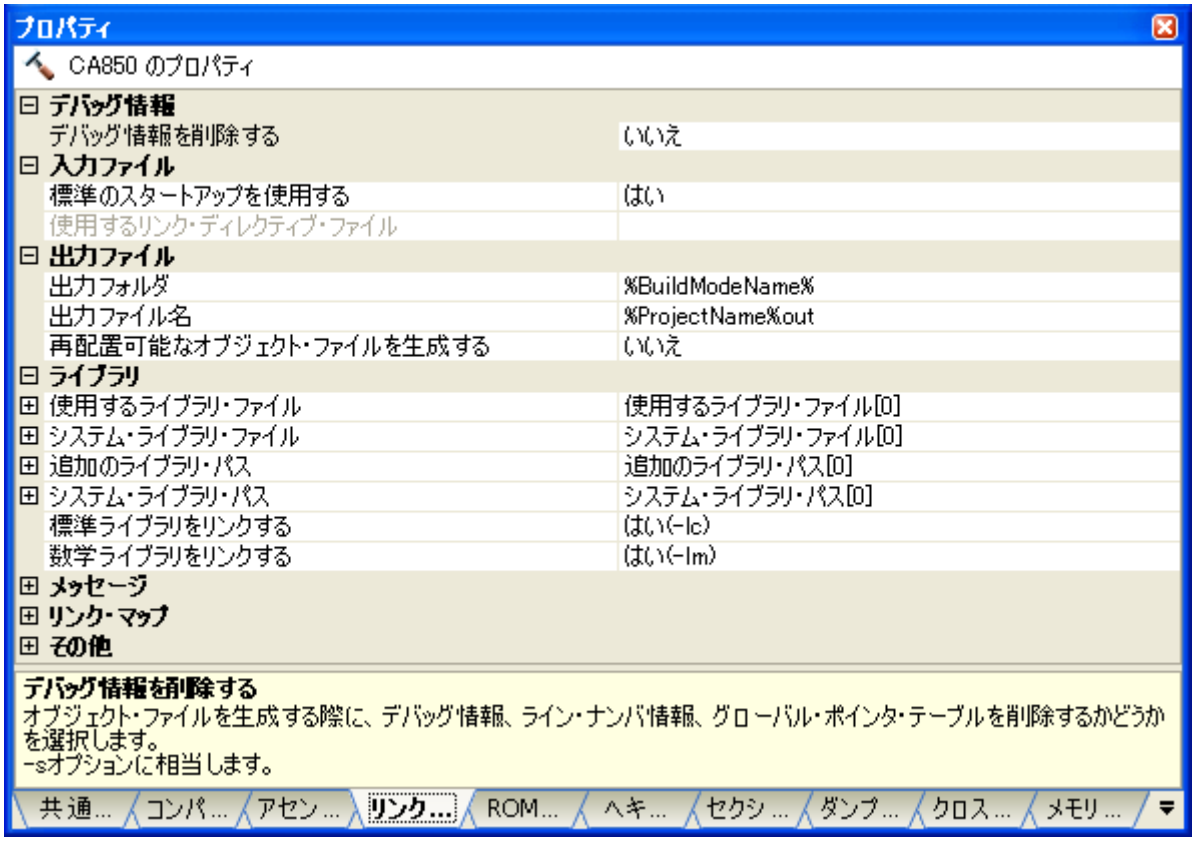
その他の追加オプション	その他に追加するアセンブラのオプションを入力します。 なお，ここで設定したオプションは，アセンブラのオプション群の最後に付加されます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	259 文字までの文字列

［リンク・オプション］タブ

本タブでは，リンカに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

注意　本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 A—7　プロパティ パネル：［リンク・オプション］タブ



[各カテゴリの説明]

(1) [デバッグ情報]

デバッグ情報に関する詳細情報の表示，および設定の変更を行います。

デバッグ情報を削除する	オブジェクト・ファイルを生成する際に，デバッグ情報，ライン・ナンバ情報，グローバル・ポインタ・テーブルを削除するかどうかを選択します。 リンカのオプション -s に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-s)	オブジェクト・ファイルを生成する際に，デバッグ情報，ライン・ナンバ情報，グローバル・ポインタ・テーブルを削除します。
		いいえ	オブジェクト・ファイルを生成する際に，デバッグ情報，ライン・ナンバ情報，グローバル・ポインタ・テーブルを削除しません。

(2) [入力ファイル]

入力ファイルに関する詳細情報の表示，および設定の変更を行います。

標準のスタートアップを使用する	標準的なスタートアップ・ルーチンが書かれているコンパイラ付属のオブジェクト・モジュール・ファイルをリンク時にリンクするかどうかを選択します。 ただし，プロジェクトに C ソース・ファイルを追加していない場合，およびスタートアップ・ノードにビルド対象ファイルを追加した場合は，コンパイラ付属のオブジェクト・モジュール・ファイルをリンクしません。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	コンパイラ付属のオブジェクト・モジュール・ファイルをリンクします。 スタートアップ・ノードに追加済みのファイルは，ビルド対象外となります。
		いいえ	コンパイラ付属のオブジェクト・モジュール・ファイルをリンクしません。
使用するリンク・ディレクティブ・ファイル	リンクに使用するリンク・ディレクティブ・ファイル名を表示します。 リンカのオプション -D に相当します。		
	デフォルト	*プロジェクトに登録しているリンク・ディレクティブ・ファイル名*	
	変更方法	変更不可	

(3) [出力ファイル]

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

出力フォルダ	生成するモジュール・ファイルを格納するフォルダを指定します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	
出力ファイル名	生成するロード・モジュールのファイル名を指定します。 “.out”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.out”が自動的に付加されます。 リンカのオプション -o に相当します。 埋め込みマクロとして次のマクロ名があります。 %ProjectName%：プロジェクト名に置換します。 空欄の場合は，%ProjectName%.out を指定したものとみなします。		
	デフォルト	%ProjectName%.out	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	259 文字までの文字列	
再配置可能なオブジェクト・ファイルを生成する	再配置可能なオブジェクト・ファイルを生成するかどうかを選択します。 リンカのオプション -r に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-r)	再配置可能なオブジェクト・ファイルを生成します。
		いいえ	再配置可能なオブジェクト・ファイルを生成しません。

(4)［ライブラリ］

ライブラリ生成に関する詳細情報の表示，および設定の変更を行います。

使用するライブラリ・ファイル	標準ライブラリ以外の使用するライブラリ・ファイル名（lib*string*.a）を指定します。 *string* のみを指定してください（例：“user”と指定すると，libuser.a を指定したものとみなされます）。 1 行に 1 ファイルずつ指定します。 ライブラリ・ファイルはライブラリ・パスから検索します。 リンカのオプション -l に相当します。 指定したライブラリ・ファイル名はサブプロパティとして表示されます。	
	デフォルト	使用するライブラリ・ファイル [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	63 文字までの文字列 256 個まで指定可能です。
システム・ライブラリ・ファイル	システムが使用するライブラリ・ファイルの名前を表示します。 システム・ライブラリ・ファイルは，使用するライブラリ・ファイルより低い優先度で検索されます。 ライブラリ・ファイル名はサブプロパティとして表示します。	
	デフォルト	システム・ライブラリ・ファイル [*定義数*]
	変更方法	変更不可
追加のライブラリ・パス	標準ライブラリ以外の使用するライブラリ・ファイルの検索フォルダを指定します。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% ：ビルド・モード名に置換します。 %ProjectName% ：プロジェクト名に置換します。 %MicomToolPath% ：本製品のインストール・フォルダの絶対パスに置換します。 ライブラリ・ファイルはライブラリ・パスから検索します。なお，相対パスを指定した場合，プロジェクト・フォルダを基点とします。 リンカのオプション -L に相当します。 指定したライブラリ・パス名はサブプロパティとして表示されます。	
	デフォルト	追加のライブラリ・パス [*定義数*]
	変更方法	[...] ボタンをクリックし，パス編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	259 文字までの文字列 256 個まで指定可能です。
システム・ライブラリ・パス	システム・ライブラリ・ファイルの検索フォルダを表示します。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% ：ビルド・モード名に置換します。 %ProjectName% ：プロジェクト名に置換します。 %MicomToolPath% ：本製品のインストール・フォルダの絶対パスに置換します。 相対パスを表示している場合は，プロジェクト・フォルダを基点とします。 リンカのオプション -L に相当します。 ライブラリ・パス名はサブプロパティとして表示します。	
	デフォルト	システム・ライブラリ・パス [*定義数*]
	変更方法	変更不可

標準ライブラリをリンクする	標準ライブラリ（libc.a）をリンクするかどうかを選択します。 リンカのオプション -lc に相当します。		
	デフォルト	はい (-lc)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-lc)	標準ライブラリをリンクします。
		いいえ	標準ライブラリをリンクしません。
数学ライブラリをリンクする	数学ライブラリ（libm.a）をリンクするかどうかを選択します。 リンカのオプション -lm に相当します。 なお，本プロパティは，［標準ライブラリをリンクする］プロパティで［はい (-lc)］を選択した場合のみ表示されます。		
	デフォルト	はい (-lm)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-lm)	数学ライブラリをリンクします。
		いいえ	数学ライブラリをリンクしません。

(5)［メッセージ］

メッセージに関する詳細情報の表示，および設定の変更を行います。

実行状況を表示する	ビルド時にリンカの実行状況を出力 パネルに表示するかどうかを選択します。 リンカのオプション -v に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-v)	ビルド時にリンカの実行状況を表示します。
		いいえ	ビルド時にリンカの実行状況を表示しません。
警告を表示する	警告を出力 パネルに表示するかどうかを選択します。 リンカのオプション -w に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	警告を表示します。
		いいえ (-w)	警告を表示しません。

(6)［リンク・マップ］

リンク・マップに関する詳細情報の表示，および設定の変更を行います。

リンク・マップ・ファイルを出力する	リンク・マップ・ファイルを出力するかどうかを選択します。 リンカのオプション -m に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-m)	リンク・マップ・ファイルを出力します。
		いいえ	リンク・マップ・ファイルを出力しません。

リンク・マップ・ファイル出力フォルダ	リンク・マップ・ファイルの出力先フォルダを指定します。 リンカのオプション -m に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［リンク・マップ・ファイルを出力する］プロパティで［はい (-m)］を選択した場合のみ表示されます。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列
リンク・マップ・ファイル名	リンク・マップ・ファイルのファイル名を指定します。 リンカのオプション -m に相当します。 拡張子は“.map”を指定してください。拡張子を省略した場合は，“.map”が自動的に付加されます。 埋め込みマクロとして次のマクロ名があります。 %ProjectName%：プロジェクト名に置換します。 空欄の場合は，%ProjectName%.map を指定したものとみなします。 なお，本プロパティは，［リンク・マップ・ファイルを出力する］プロパティで［はい (-m)］を選択した場合のみ表示されます。	
	デフォルト	%ProjectName%.map
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(7)［その他］

リンクに関するその他の詳細情報の表示，および設定の変更を行います。

エントリ・シンボル	オブジェクト・ファイルのエントリ・ポイント・アドレスとして設定するシンボルを指定します。 空欄の場合は，次の順序でエントリ・ポイント・アドレスを決定します。 (1) シンボル“__start”が存在する場合は，そのアドレス (2) text 属性のセクションが存在する場合は，生成されるオブジェクト・ファイル内の最下位に割り付けられた text 属性のセクションの先頭アドレス (3) アドレス 0 リンカのオプション -e に相当します。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	1022 文字までの文字列

ホールのフィリング値を指定する	生成されるオブジェクト内のセクション間のアライン・ホールのフィリング値を指定するかどうかを選択します。 リンカのオプション -f に相当します。 なお，本プロパティは，［2 パス・モードでリンクする］プロパティで［はい (-B)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-f)	ホールのフィリング値を指定します。
		いいえ	ホールのフィリング値を指定しません。
ホールのフィリング値	生成されるオブジェクト内のセクション間のアライン・ホールのフィリング値を指定します。 リンカのオプション -f に相当します。 なお，本プロパティは，［ホールのフィリング値を指定する］プロパティで［はい (-f)］を選択した場合のみ表示されます。		
	デフォルト	0x0000	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0000 ～ 0xffff（4 桁の 16 進数）	
GP 情報を表示する	［アセンブル・オプション］タブの［その他］カテゴリの［sdata/sbss セクションに配置するデータ長の上限値 (バイト)］プロパティの数値設定において目安として用いるための情報を出力 パネルに表示するかどうかを選択します。 リンカのオプション -A に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-A)	［sdata/sbss セクションに配置するデータ長の上限値 (バイト)］プロパティの数値設定において目安として用いるための情報を表示します。
		いいえ	［sdata/sbss セクションに配置するデータ長の上限値 (バイト)］プロパティの数値設定において目安として用いるための情報を表示しません。
2 パス・モードでリンクする	2 パス・モードでリンクするかどうかを選択します。 2 パス・モードは 1 パス・モードよりも低速ですが，より大きなサイズのファイルを処理することができます。 リンカのオプション -B に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-B)	2 パス・モードでリンクを行います。
		いいえ	1 パス・モードでリンクを行います。

リロケーション不正を無視する	リロケーション処理において，次の不正箇所があった場合，エラーとはせず，警告メッセージを出力してリンクの処理を続行するかどうかを選択します。 - 未解決な外部参照のアドレス計算結果が不正 - 配置されるセクションとの関係が不正 リンカのオプション -E に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-E)	リロケーション処理において，不正箇所があった場合，警告メッセージを出力してリンクの処理を続行します。
		いいえ	リロケーション処理において，不正箇所があった場合，警告メッセージを出力してリンクの処理を中止します。
すべての多重定義シンボルを検出する	多重定義されたすべての外部シンボルに対してメッセージを出力し，リンクの処理を中止するかどうかを選択します。 リンカのオプション -M に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-M)	多重定義されたすべての外部シンボルに対してメッセージを出力し，リンクの処理を中止します。
		いいえ	多重定義された最初の外部シンボルに対してメッセージを出力し，リンクの処理を中止します。
未定義外部シンボル不正をチェックする	未定義外部シンボルをリンクする際，シンボルのサイズ，および整列条件の不正をチェックするかどうかを選択します。 リンカのオプション -t に相当します。 なお，本プロパティは，［メッセージ］カテゴリの［警告を表示する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	未定義外部シンボルをリンクする際，シンボルのサイズ，および整列条件の不正をチェックします。
		いいえ (-t)	未定義外部シンボルをリンクする際，シンボルのサイズ，および整列条件の不正をチェックしません。
外部シンボル不正をチェックする	外部シンボルをリンクする際，シンボルのサイズ，および整列条件の不正をチェックするかどうかを選択します。 リンカのオプション -T に相当します。 なお，本プロパティは，［メッセージ］カテゴリの［警告を表示する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	外部シンボルをリンクする際，シンボルのサイズ，および整列条件の不正をチェックします。
		いいえ (-T)	外部シンボルをリンクする際，シンボルのサイズ，および整列条件の不正をチェックしません。

マスク・レジスタ機能の使用／未使用の混在をチェックする	C ソース・ファイルから作成されたオブジェクト・ファイルのリンクの際，マスク・レジスタ機能を使用しているファイルと使用していないファイルの混在をチェックするかどうかを選択します。 リンカのオプション -mc に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-mc)	マスク・レジスタ機能を使用しているファイルと使用していないファイルの混在をチェックします。
		いいえ	マスク・レジスタ機能を使用しているファイルと使用していないファイルの混在をチェックしません。
レジスタ・モードをチェックする	すべての入力オブジェクト・ファイルに対し，異なるレジスタ・モードが混在している場合に詳細情報を表示するかどうかを選択します。 リンカのオプション -rc に相当します。		
	デフォルト	はい (-rc)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-rc)	すべての入力オブジェクト・ファイルに対し，異なるレジスタ・モードが混在している場合に詳細情報を表示します。
		いいえ	すべての入力オブジェクト・ファイルに対し，異なるレジスタ・モードが混在している場合に詳細情報を表示しません。
ライブラリ・ファイルを再スキャンする	［ライブラリ］カテゴリの［使用するライブラリ・ファイル］プロパティ，および［システム・ライブラリ・ファイル］プロパティで指定したライブラリ・ファイルの再参照を行うかどうかを選択します。 本プロパティを指定すると，ライブラリのリンク順によるシンボル未解決を防ぐことができます。 リンカのオプション -rescan に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-rescan)	使用するライブラリ・ファイルの再参照を行います。
		いいえ	使用するライブラリ・ファイルの再参照を行いません。
内蔵 ROM 領域への配置をチェックする	内蔵 ROM 領域への配置に対して，チェックを行うかどうかを選択します。 ROM レス・モード使用時は，［いいえ (-rom_less)］を選択してください。 リンカのオプション -rom_less に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	内蔵 ROM 領域への配置に対して，チェックを行います。
		いいえ (-rom_less)	内蔵 ROM 領域への配置に対して，チェックを行いません。

<table>
<tr><td rowspan="5">内蔵メモリ・オーバーフローの扱い</td><td colspan="3">内蔵 ROM ／ RAM 領域への配置時にオーバーフローが発生した場合，警告メッセージを表示して処理を継続するか，エラー・メッセージを表示して処理を中止するかを選択します。
リンカのオプション -Ximem_overflow=warning に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">エラー (なし)</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>エラー (なし)</td><td>内蔵 ROM ／ RAM 領域への配置時にオーバーフローが発生した場合，エラー・メッセージを表示して処理を中止します。</td></tr>
<tr><td>警告 (-Ximem_overflow=warning)</td><td>内蔵 ROM ／ RAM 領域への配置時にオーバーフローが発生した場合，警告メッセージを表示して処理を継続します。</td></tr>
<tr><td rowspan="4">リンク前に実行するコマンド</td><td colspan="3">リンク処理前に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%LinkedFile%　：リンク処理時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">リンク前に実行するコマンド [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td rowspan="4">リンク後に実行するコマンド</td><td colspan="3">リンク処理後に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%LinkedFile%　：リンク時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">リンク後に実行するコマンド [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">1023 文字までの文字列
64 個まで指定可能です。</td></tr>
</table>

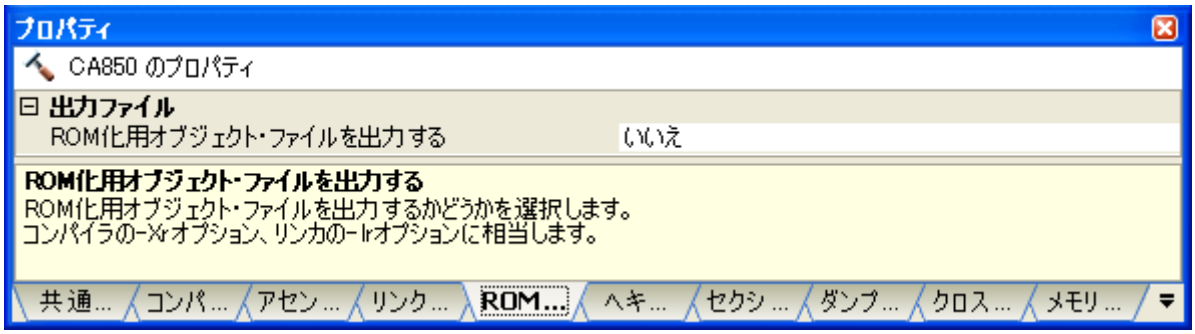
その他の追加オプション	その他に追加するリンカのオプションを入力します。 なお，ここで設定したオプションは，リンカのオプション群の最後に付加されます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	259 文字までの文字列

[ROM 化プロセス・オプション] タブ

本タブでは，ROM 化プロセッサに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

注意　本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 A—8　プロパティ パネル：[ROM 化プロセス・オプション] タブ



[各カテゴリの説明]

（1）[出力ファイル]

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

ROM 化用オブジェクト・ファイルを出力する	ROM 化用オブジェクト・ファイルを出力するかどうかを選択します。 コンパイラのオプション -Xr，リンカのオプション -lr に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xr -lr)	ROM 化用オブジェクト・ファイルを出力します。
		いいえ	ROM 化用オブジェクト・ファイルを出力しません。

ROM 化用オブジェクト・ファイル出力フォルダ	ROM 化用オブジェクト・ファイルを格納するフォルダを指定します。 ROM 化プロセッサのオプション -o に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択した場合のみ表示されます。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列
ROM 化用オブジェクト・ファイル名	ROM 化用オブジェクト・ファイル名を指定します。 “.out”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.out”が自動的に付加されます。 ROM 化プロセッサのオプション -o に相当します。 なお，本プロパティは，［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択した場合のみ表示されます。	
	デフォルト	romp.out
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(2)［入力ファイル］

入力ファイルに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。

標準の ROM 化用領域確保コード・ファイルを使用する	［共通オプション］タブの［レジスタ・モード］カテゴリの［レジスタ・モードの選択］プロパティで選択したレジスタ・モードに対応する標準の ROM 化用領域確保コード・ファイル（rompcrt.o）を使用するかどうかを選択します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	標準の ROM 化用領域確保コード・ファイルを使用します。
		いいえ	標準の ROM 化用領域確保コード・ファイルを使用しません。 ROM 化用領域確保コード・ファイルを作成し，［ROM 化用領域確保コード・ファイル名］プロパティにファイル名を指定してください。

ROM 化用領域確保コード・ファイル名	ROM 化用領域確保コード・ファイル名を指定します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 空欄の場合は，リンク時にエラーとなるため，ファイル名は必ず指定してください。 なお，本プロパティは，［標準の ROM 化用領域確保コード・ファイルを使用する］プロパティで［いいえ］を選択した場合のみ表示されます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，ROM 化用領域確保コード・ファイルを指定 ダイアログによる編集
	指定可能値	259 文字までの文字列

(3)［セクション・リスト］

セクション・リストに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。

rompsec セクションに格納する順番	ROM 化するセクション名と rompsec セクションに格納する順番を指定します。 「*セクション名 オプション属性*」の形式で 1 行に 1 つずつ指定します。 ［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択している場合，本プロパティの編集操作確定時に，ROM 化用セクション・ファイルを出力します。 オプション属性の形式を以下に示します。 - -p 追加するセクションが data 属性，sdata 属性の場合に指定します。 このオプション属性による指定を省略した場合，data 属性，または sdata 属性を持つすべてのセクション，および内蔵命令 RAM に配置されるセクションを指定したものとみなします。 - -t 追加するセクションが text 属性，const 属性の場合に指定します。 このオプション属性による指定を省略した場合，内蔵命令 RAM に配置される各セクションを指定したものとみなします。 なお，内蔵命令 RAM 搭載のデバイス・ファイルを指定してリンクされた入力ファイルに対して，このオプション属性で特定のセクションを指定した場合，指定されなかった内蔵命令 RAM に配置されたセクションは，rompsec セクションに格納されないだけでなく，出力ファイル中からも削除されます。 ROM 化プロセッサのオプション -t,-p に相当します。 指定したセクション名はサブプロパティとして表示されます。	
	デフォルト	rompsec セクションに格納する順番 [*設定数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1022 文字までの文字列 1024 個まで指定可能です。

ROM 化用セクション・ファイルを出力する	ROM 化用セクション・ファイルを出力するかどうかを選択します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	［rompsec セクションに格納する順番］プロパティの編集操作確定時に，ROM 化用セクション・ファイルを出力します。
		いいえ	ROM 化用セクション・ファイルを出力しません。
ROM 化用セクション・ファイル出力フォルダ	ROM 化用セクション・ファイルを格納するフォルダを指定します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	%BuildModeName%	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	
ROM 化用セクション・ファイル名	ROM 化用セクション・ファイル名を指定します。 拡張子は自由に指定可能です。 なお，本プロパティは，［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	259 文字までの文字列	

(4)［メモリ・マップ］

メモリ・マップに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。

メモリ・マップ・ファイルを出力する	メモリ・マップ・ファイルを出力するかどうかを選択します。 ROM 化プロセッサのオプション -m に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-m)	メモリ・マップ・ファイルを出力します。
		いいえ	メモリ・マップ・ファイルを出力しません。

メモリ・マップ・ファイル出力フォルダ	メモリ・マップ・ファイルを格納するフォルダを指定します。 ROM 化プロセッサのオプション -m に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［メモリ・マップ・ファイルを出力する］プロパティで［はい (-m)］を選択した場合のみ表示されます。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列
メモリ・マップ・ファイル名	メモリ・マップ・ファイル名を指定します。 “.map”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.map”が自動的に付加されます。 ROM 化プロセッサのオプション -m に相当します。 埋め込みマクロとして次のマクロ名があります。 %ProjectName%：プロジェクト名に置換します。 空欄の場合は，romp.map を指定したものとみなします。 なお，本プロパティは，［メモリ・マップ・ファイルを出力する］プロパティで［はい (-m)］を選択した場合のみ表示されます。	
	デフォルト	romp.map
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(5)［その他］

ROM 化プロセスに関するその他の詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。

エントリ・ラベル	生成する rompsec セクションの先頭アドレスとして用いるエントリ・ラベルを指定します。 ROM 化プロセッサのオプション -b に相当します。 空欄の場合は，__S_romp を指定したものとみなします。	
	デフォルト	__S_romp
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	1022 文字までの文字列

ROM 化用オブジェクト・ファイル中に text 属性セクションを含める	ROM 化用オブジェクト・ファイル中に text 属性セクションを含めるかどうかを選択します。 ROM 化プロセッサのオプション -d に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ROM 化用オブジェクト・ファイル中に text 属性セクションを含めます。
		いいえ (-d)	ROM 化用オブジェクト・ファイル中に text 属性セクションを含めません。
アドレスの重複をチェックする	入力ファイル（実行可能オブジェクト・ファイル），および出力ファイル（ROM 化用オブジェクト・ファイル）のアドレスの重複チェックを行うかどうかを選択します。 ROM 化プロセッサのオプション -i に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	入力ファイル，および出力ファイルのアドレスの重複チェックを行います。
		いいえ (-i)	入力ファイル，および出力ファイルのアドレスの重複チェックを行いません。
内蔵 ROM 領域への配置をチェックする	内蔵 ROM 領域への配置に対して，チェックを行うかどうかを選択します。 ROM レス・モード使用時は，［いいえ (-rom_less)］を選択してください。 ROM 化プロセッサのオプション -rom_less に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	内蔵 ROM 領域への配置に対して，チェックを行います。
		いいえ (-rom_less)	内蔵 ROM 領域への配置に対して，チェックを行いません。
内蔵メモリ・オーバーフローの扱い	内蔵 ROM ／ RAM 領域への配置時にオーバーフローが発生した場合，警告メッセージを表示して処理を継続するか，エラー・メッセージを表示して処理を中止するかを選択します。		
	デフォルト	エラー (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	エラー (なし)	内蔵 ROM ／ RAM 領域への配置時にオーバーフローが発生した場合，エラー・メッセージを表示して処理を中止します。
		警告 (-Ximem_overflow=warning)	内蔵 ROM ／ RAM 領域への配置時にオーバーフローが発生した場合，警告メッセージを表示して処理を継続します。

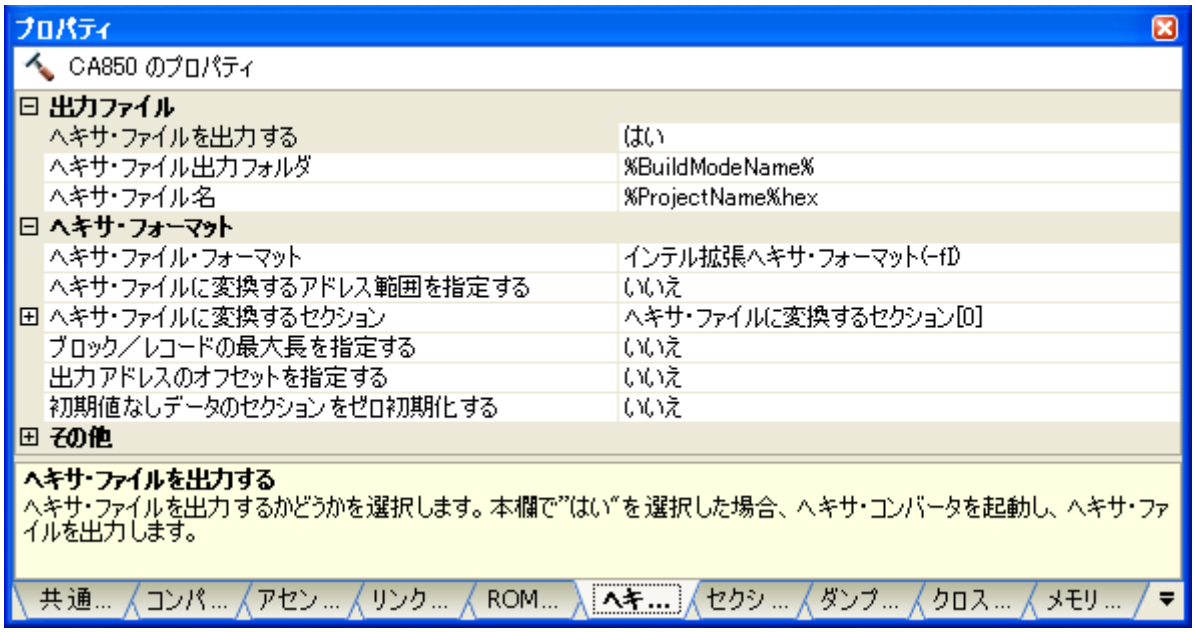
<table>
<tr><td>ROM 化前に実行するコマンド</td><td colspan="2">ROM 化処理前に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%RomizedFile%　：ROM 化処理時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td>ROM 化前に実行するコマンド [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>［...］ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td>1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td>ROM 化後に実行するコマンド</td><td colspan="2">ROM 化処理後に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%RomizedFile%　：ROM 化時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td>ROM 化後に実行するコマンド [定義数]</td></tr>
<tr><td></td><td>変更方法</td><td>［...］ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td>1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td>その他の追加オプション</td><td colspan="2">その他に追加する ROM 化プロセッサのオプションを入力します。
なお，ここで設定したオプションは，ROM 化プロセッサのオプション群の最後に付加されます。</td></tr>
<tr><td></td><td>デフォルト</td><td>空欄</td></tr>
<tr><td></td><td>変更方法</td><td>テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集</td></tr>
<tr><td></td><td>指定可能値</td><td>259 文字までの文字列</td></tr>
</table>

[ヘキサ・コンバート・オプション]タブ

本タブでは，ヘキサ・コンバータに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

注意　本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 A—9　プロパティ パネル：[ヘキサ・コンバート・オプション]タブ



[各カテゴリの説明]

(1) [出力ファイル]

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

ヘキサ・ファイルを出力する	ヘキサ・ファイルを出力するかどうかを選択します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ヘキサ・ファイルを出力します。
		いいえ	ヘキサ・ファイルを出力しません。

ヘキサ・ファイル出力フォルダ	ヘキサ・ファイルを格納するフォルダを指定します。 ヘキサ・コンバータのオプション -o に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［ヘキサ・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列
ヘキサ・ファイル名	ヘキサ・ファイル名を指定します。 ヘキサ・コンバータのオプション -o に相当します。 拡張子は自由に指定可能です。 埋め込みマクロとして次のマクロ名があります。 %ProjectName%：プロジェクト名に置換します。 空欄の場合は，%ProjectName%.hex を指定したものとみなします。 なお，本プロパティは，［ヘキサ・ファイルを出力する］プロパティで［はい］を選択した場合のみ表示されます。	
	デフォルト	%ProjectName%.hex
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(2)［ヘキサ・フォーマット］

ヘキサ・フォーマットに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。

ヘキサ・ファイル・フォーマット	生成するヘキサ・ファイルのフォーマットを選択します。 ヘキサ・コンバータのオプション -f に相当します。		
	デフォルト	インテル拡張ヘキサ・フォーマット (-fI)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	インテル拡張ヘキサ・フォーマット (-fI)	生成するヘキサ・ファイルをインテル拡張ヘキサ・フォーマットとします。
		モトローラ S タイプ・フォーマット (スタンダード・アドレス)(-fS)	生成するヘキサ・ファイルをモトローラ S タイプ・フォーマット（スタンダード・アドレス）とします。
		モトローラ S タイプ・フォーマット (32 ビット・アドレス)(-fs)	生成するヘキサ・ファイルをモトローラ S タイプ・フォーマット（32 ビット・アドレス）とします。
		拡張テクトロニクス・ヘキサ・フォーマット (-fT)	生成するヘキサ・ファイルを拡張テクトロニクス・ヘキサ・フォーマットとします。
ヘキサ・ファイルに変換するアドレス範囲を指定する	ヘキサ・ファイルに変換するアドレス範囲を指定するかどうかを選択します。 ヘキサ・コンバータのオプション -U に相当します。 なお，本プロパティは，［ヘキサ・ファイル・フォーマット］プロパティで［拡張テクトロニクス・ヘキサ・フォーマット (-fT)］を選択した場合は表示されません。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-U)	ヘキサ・ファイルに変換するアドレス範囲を指定します。
		いいえ	ヘキサ・ファイルに変換するアドレス範囲を指定しません。
フィリング値	ヘキサ・ファイルに変換する領域のうち，未使用領域のフィリング値を指定します。 ヘキサ・コンバータのオプション -U に相当します。 なお，本プロパティは，［ヘキサ・ファイルに変換するアドレス範囲を指定する］プロパティで［はい (-U)］を選択した場合のみ表示されます。		
	デフォルト	0xFF	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0000 ～ 0xFFFF（2 桁，または 4 桁の 16 進数）	
スタート・アドレス	ヘキサ・ファイルに変換する領域のスタート・アドレスを指定します。 ヘキサ・コンバータのオプション -U に相当します。 なお，本プロパティは，［ヘキサ・ファイルに変換するアドレス範囲を指定する］プロパティで［はい (-U)］を選択した場合のみ表示されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ *デバイスで扱うことができるアドレスの最大値*（16 進数）	

サイズ	ヘキサ・ファイルに変換する領域のサイズを指定します。 ヘキサ・コンバータのオプション -U に相当します。 なお，本プロパティは，［ヘキサ・ファイルに変換するアドレス範囲を指定する］プロパティで［はい (-U)］を選択した場合のみ表示されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x1 ～ *デバイスで扱うことができるアドレスの最大値*（16 進数）	
ヘキサ・ファイルに変換するセクション	ヘキサ・ファイルに変換するセクションを指定します。 1 行に 1 つずつ指定します。 本プロパティの設定を省略した場合，NOBITS 以外のセクション・タイプとセクション属性 A を持つすべてのセクションをヘキサ・ファイルに変換します。 ヘキサ・コンバータのオプション -I に相当します。 指定したセクション名はサブプロパティとして表示されます。 なお，本プロパティは，［ヘキサ・ファイルに変換するアドレス範囲を指定する］プロパティで［いいえ］を選択した場合のみ表示されます。		
	デフォルト	ヘキサ・ファイルに変換するセクション [*設定数*]	
	変更方法	［...］ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能	
	指定可能値	1022 文字までの文字列 1024 個まで指定可能です。	
ブロック／レコードの最大長を指定する	ヘキサ・ファイルのブロック／レコードの最大長を指定するかどうかを選択します。 ヘキサ・コンバータのオプション -b に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-b)	ブロック／レコードの最大長を指定します。
		いいえ	ブロック／レコードの最大長を指定しません。

<table>
<tr><td rowspan="4">ブロック／レコードの最大長</td><td colspan="2">ヘキサ・ファイルのブロック／レコードの最大長を指定します。
ヘキサ・コンバータのオプション -b に相当します。
［ヘキサ・ファイル・フォーマット］プロパティを変更したとき，本プロパティが変更後のヘキサ・ファイル・フォーマットの指定可能値の範囲外となる場合，本プロパティは該当フォーマットのデフォルトの値に変更されます。
なお，本プロパティは，［ブロック／レコードの最大長を指定する］プロパティで［はい (-b)］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td>- ［ヘキサ・ファイル・フォーマット］プロパティで［インテル拡張ヘキサ・フォーマット (-fI)］を選択し，本プロパティのコンテキスト・メニューの［デフォルトに戻す］を選択した場合
31
- ［ヘキサ・ファイル・フォーマット］プロパティで［モトローラ S タイプ・フォーマット (スタンダード・アドレス)(-fS)］を選択し，本プロパティのコンテキスト・メニューの［デフォルトに戻す］を選択した場合
80
- ［ヘキサ・ファイル・フォーマット］プロパティで［モトローラ S タイプ・フォーマット (32 ビット・アドレス)(-fs)］を選択し，本プロパティのコンテキスト・メニューの［デフォルトに戻す］を選択した場合
80
- ［ヘキサ・ファイル・フォーマット］プロパティで［拡張テクトロニクス・ヘキサ・フォーマット (-fT)］を選択し，本プロパティのコンテキスト・メニューの［デフォルトに戻す］を選択した場合
255</td></tr>
<tr><td>変更方法</td><td>テキスト・ボックスによる直接入力</td></tr>
<tr><td>指定可能値</td><td>- ［ヘキサ・ファイル・フォーマット］プロパティで［インテル拡張ヘキサ・フォーマット (-fI)］を選択した場合
1 ～ 255（10 進数），または 0x01 ～ 0xff（16 進数）
- ［ヘキサ・ファイル・フォーマット］プロパティで［モトローラ S タイプ・フォーマット (スタンダード・アドレス)(-fS)］を選択した場合
1 ～ 251（10 進数），または 0x01 ～ 0xfb（16 進数）
- ［ヘキサ・ファイル・フォーマット］プロパティで［モトローラ S タイプ・フォーマット (32 ビット・アドレス)(-fs)］を選択した場合
1 ～ 250（10 進数），または 0x01 ～ 0xfa（16 進数）
- ［ヘキサ・ファイル・フォーマット］プロパティで［拡張テクトロニクス・ヘキサ・フォーマット (-fT)］を選択した場合
16 ～ 255（10 進数），または 0x10 ～ 0xff（16 進数）</td></tr>
</table>

出力アドレスのオフセットを指定する	ヘキサ・ファイルに変換する際の出力アドレスのオフセットを指定するかどうかを選択します。 ヘキサ・コンバータのオプション -d に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-d)	出力アドレスのオフセットを指定します。
		いいえ	出力アドレスのオフセットを指定しません。
出力アドレスのオフセット	ヘキサ・ファイルに変換する際の出力アドレスのオフセットを指定します。 ヘキサ・コンバータのオプション -d に相当します。 なお，本プロパティは，［出力アドレスのオフセットを指定する］プロパティで［はい (-d)］を選択した場合のみ表示されます。		
	デフォルト	0x0	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xfffffffe（16 進数）	
初期値なしデータのセクションをゼロ初期化する	ヘキサ・ファイルに変換する際に，初期値なしデータのセクション（セクション・タイプ NOBITS とセクション属性 A を持つセクション）をゼロ初期化するかどうかを選択します。 ヘキサ・コンバータのオプション -z に相当します。 なお，本プロパティは，［ヘキサ・ファイルに変換するアドレス範囲を指定する］プロパティで［はい (-U)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-z)	初期値なしデータのセクションをゼロ初期化します。
		いいえ	初期値なしデータのセクションをゼロ初期化しません。

(3)［シンボル・テーブル］

シンボル・テーブルに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで［いいえ］を選択した場合，および［ヘキサ・フォーマット］カテゴリの［ヘキサ・ファイル・フォーマット］プロパティで［拡張テクトロニクス・ヘキサ・フォーマット (-fT)］以外を選択した場合は表示されません。

シンボル・テーブルを変換する	ヘキサ・ファイルに変換する際に，シンボル・テーブルを変換するかどうかを選択します。 ヘキサ・コンバータのオプション -S -x に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (グローバル・シンボルとローカル・シンボルを変換する)(-S -x)	グローバル・シンボルとローカル・シンボルを変換します。
		はい (グローバル・シンボルのみ変換する)(-S)	グローバル・シンボルのみ変換します。
		いいえ	シンボル・テーブルを変換しません。

(4)［その他］

ヘキサ・コンバートに関するその他の詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［ヘキサ・ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。

内蔵 ROM 領域オーバフロー時に警告を表示する	ヘキサ・ファイルに変換する領域が内蔵 ROM 領域をオーバフローした場合，警告メッセージを表示するかどうかを選択します。 ヘキサ・コンバータのオプション -rom_less に相当します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ヘキサ・ファイルに変換する領域が内蔵 ROM 領域をオーバフローした場合，警告メッセージを表示します。
		いいえ (-rom_less)	ヘキサ・ファイルに変換する領域が内蔵 ROM 領域をオーバフローした場合，警告メッセージを表示しません。
ヘキサ・コンバート前に実行するコマンド	ヘキサ・コンバート処理前に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder% ：出力フォルダの絶対パスに置換します。 %OutputFile% ：出力ファイルの絶対パスに置換します。 %InputFile% ：ヘキサ・コンバート処理時の入力ファイルの絶対パスに置換します。 %HexConvertedFile% ：ヘキサ・コンバート処理時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。		
	デフォルト	ヘキサ・コンバート前に実行するコマンド [*定義数*]	
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能	
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。	

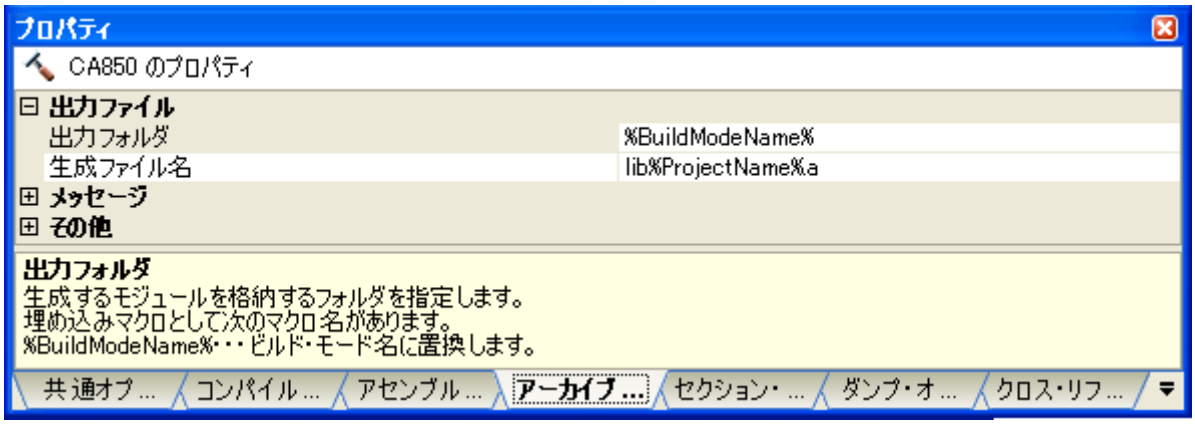
<table>
<tr><td rowspan="4">ヘキサ・コンバート後に実行するコマンド</td><td colspan="2">ヘキサ・コンバート処理後に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder% ：出力フォルダの絶対パスに置換します。
%OutputFile% ：出力ファイルの絶対パスに置換します。
%InputFile% ：ヘキサ・コンバート処理時の入力ファイルの絶対パスに置換します。
%HexConvertedFile% ：ヘキサ・コンバート処理時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td>ヘキサ・コンバート後に実行するコマンド [定義数]</td></tr>
<tr><td>変更方法</td><td>[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td>1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td rowspan="4">その他の追加オプション</td><td colspan="2">その他に追加するヘキサ・コンバータのオプションを入力します。
なお，ここで設定したオプションは，ヘキサ・コンバータのオプション群の最後に付加されます。</td></tr>
<tr><td>デフォルト</td><td>空欄</td></tr>
<tr><td>変更方法</td><td>テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td>259 文字までの文字列</td></tr>
</table>

［アーカイブ・オプション］タブ

本タブでは，アーカイバに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

注意　本タブは，ライブラリ用のプロジェクトの場合のみ表示されます。

図 A—10　プロパティ パネル：［アーカイブ・オプション］タブ



［各カテゴリの説明］

（1）［出力ファイル］

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

出力フォルダ	生成するアーカイブ・ファイルを格納するフォルダを指定します。 アーカイバのキー q に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列

生成ファイル名	生成するアーカイブ・ファイルのファイル名を指定します。 アーカイバのキー q に相当します。 “.a” 以外の拡張子を指定することはできません。拡張子を省略した場合は，“.a” が自動的に付加されます。 埋め込みマクロとして次のマクロ名があります。 %ProjectName% : プロジェクト名に置換します。	
	デフォルト	lib%ProjectName%.a
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(2) [メッセージ]

メッセージに関する詳細情報の表示，および設定の変更を行います。

実行状況を表示する	ビルド時にアーカイバの実行状況[注]を出力 パネルに表示するかどうかを選択します。 アーカイバのオプション v に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (v)	ビルド時にアーカイバの実行状況を表示します。
		いいえ	ビルド時にアーカイバの実行状況を表示しません。

注 実行状況の出力の意味を以下に示します。

出力形式	意味
q - *ファイル名*	アーカイブ・ファイルの新規作成，またはメンバの追加

(3) [その他]

アーカイブに関するその他の詳細情報の表示，および設定の変更を行います。

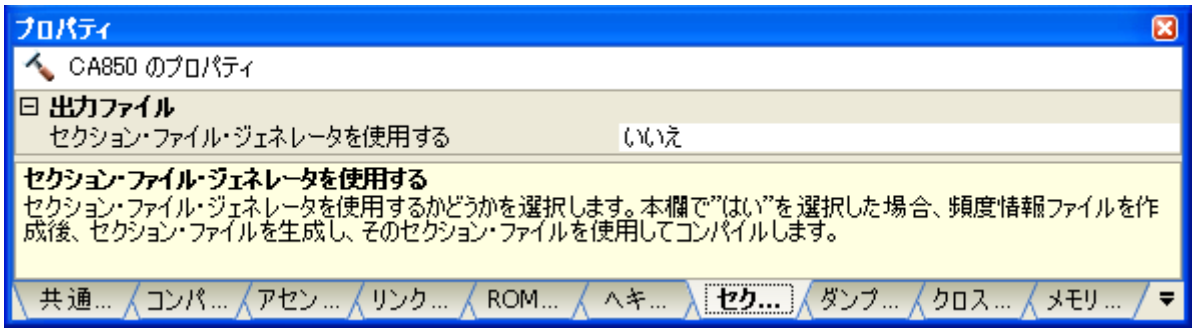
アーカイブ前に実行するコマンド	アーカイブ処理前に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例 : call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder% : プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder% : 出力フォルダの絶対パスに置換します。 %OutputFile% : 出力ファイルの絶対パスに置換します。 %ArchivedFile% : アーカイブ処理時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。	
	デフォルト	アーカイブ前に実行するコマンド [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。

<table>
<tr><td rowspan="4">アーカイブ後に実行するコマンド</td><td colspan="2">アーカイブ処理後に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder%　：出力フォルダの絶対パスに置換します。
%OutputFile%　：出力ファイルの絶対パスに置換します。
%ArchivedFile%　：アーカイブ処理時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td>アーカイブ後に実行するコマンド[定義数]</td></tr>
<tr><td>変更方法</td><td>[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td>1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td rowspan="4">その他の追加オプション</td><td colspan="2">その他に追加するアーカイバのオプションを入力します。
なお，ここで設定したオプションは，アーカイバのオプション群の最後に付加されます。</td></tr>
<tr><td>デフォルト</td><td>空欄</td></tr>
<tr><td>変更方法</td><td>テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，
文字列入力 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td>259 文字までの文字列</td></tr>
</table>

［セクション・ファイル・ジェネレート・オプション］タブ

本タブでは，セクション・ファイル・ジェネレータに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

図 A―11　プロパティ パネル：［セクション・ファイル・ジェネレート・オプション］タブ



［各カテゴリの説明］

（1）［出力ファイル］

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

セクション・ファイル・ジェネレータを使用する	ビルド時にセクション・ファイル・ジェネレータを使用するかどうかを選択します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	頻度情報ファイルを生成後，セクション・ファイルを生成し，そのセクション・ファイルを使用してコンパイルを行います。 なお，セクション・ファイルは，ラピッド・ビルドの対象外となります。
		いいえ	ビルド時に頻度情報ファイルを作成せず，セクション・ファイル・ジェネレータを使用しません。

セクション・ファイル出力フォルダ	セクション・ファイルを格納するフォルダを指定します。 セクション・ファイル・ジェネレータのオプション -o に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% ：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択した場合のみ表示されます。	
	デフォルト	%BuildModeName%
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	指定可能値	247 文字までの文字列
セクション・ファイル名	セクション・ファイル名を指定します。 “.sf”以外の拡張子を指定することはできません。拡張子を省略した場合は，“.sf”が自動的に付加されます。 セクション・ファイル・ジェネレータのオプション -o に相当します。 埋め込みマクロとして次のマクロ名があります。 %ProjectName% ：プロジェクト名に置換します。 空欄の場合は，%ProjectName%.sf を指定したものとみなします。 なお，本プロパティは，［セクション・ファイル・ジェネレータを使用する］プロパティで［はい］を選択した場合のみ表示されます。	
	デフォルト	%ProjectName%.sf
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(2) ［メッセージ］

メッセージに関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［いいえ］を選択した場合は表示されません。

実行状況を表示する	ビルド時にセクション・ファイル・ジェネレータの実行状況を出力 パネルに表示するかどうかを選択します。 セクション・ファイル・ジェネレータのオプション -v に相当します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-v)	ビルド時にセクション・ファイル・ジェネレータの実行状況を表示します。
		いいえ	ビルド時にセクション・ファイル・ジェネレータの実行状況を表示しません。

(3)［変数の配置］

変数の配置に関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［いいえ］を選択した場合は表示されません。

変数ソートのキー	セクション・ファイル中に出力する変数ソートのキーを選択します。 セクション・ファイル・ジェネレータのオプション -ns，-sname，-ssection，-ssize，-O，-O2 に相当します。		
	デフォルト	利用頻度 (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	ソートしない (-ns)	セクション・ファイル中に出力する変数のソートを行いません。
		利用頻度 (なし)	セクション・ファイル中に出力する変数を利用頻度順にソートします。
		変数名 (-sname)	セクション・ファイル中に出力する変数を変数名の辞書順にソートします。
		セクション名 (-ssection)	セクション・ファイル中に出力する変数をセクション名の辞書順にソートします。
		変数のサイズ (-ssize)	セクション・ファイル中に出力する変数をサイズの小さい順にソートします。
		最適配置 (-O)	セクション・ファイル中に出力する変数を利用頻度順にソートし，.tidata セクションに配置可能な分だけの変数を判断して出力します。
		全セクションを最適配置 (-O2)	セクション・ファイル中に出力する変数を変数のサイズあたりの利用頻度の高い順に，.tidata，.sidata，.sedata，.sdata セクションに順番で配置できるように選択し，配置可能な分だけの変数を判断して出力します。
最適化の対象としないセクションの指定	セクション・ファイル生成時に最適化の対象としないセクションを選択します。 セクション・ファイル・ジェネレータのオプション -Xcs に相当します。 なお，本プロパティは，［変数ソートのキー］プロパティで［最適配置 (-O)］，または［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。		
	デフォルト	すべてのセクションを最適化の対象とする (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	すべてのセクションを最適化の対象とする (なし)	セクション・ファイル生成時にすべてのセクションを最適化の対象とします。
		すべてのセクションを最適化の対象としない (-Xcs)	セクション・ファイル生成時にすべてのセクションを最適化の対象としません。
		最適化の対象としないセクションを指定する (-Xcs)	セクション・ファイル生成時に最適化の対象としないセクションを指定します。

<table>
<tr><td>最適化の対象としないセクション</td><td colspan="3">セクション・ファイル生成時に最適化の対象としないセクションを指定します。
1 行に 1 つずつ指定します。
本プロパティの設定を省略した場合，すべてのセクションを最適化の対象としません。
セクション・ファイル・ジェネレータのオプション -Xcs に相当します。
指定したセクション名はサブプロパティとして表示されます。
なお，本プロパティは，［最適化の対象としないセクションの指定］プロパティで［最適化の対象としないセクションを指定する (-Xcs)］を選択した場合のみ表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">最適化の対象としないセクション [設定数]</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">［...］ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td></td><td>指定可能値</td><td colspan="2">1022 文字までの文字列
1024 個まで指定可能です。</td></tr>
<tr><td>tidata セクションへの割り付け可能サイズを指定する</td><td colspan="3">.tidata.word/.tidata.byte セクションへの割り付け可能サイズを指定するかどうかを選択します。
セクション・ファイル・ジェネレータのオプション -size_tidata に相当します。
なお，本プロパティは，［変数ソートのキー］プロパティで［最適配置 (-O)］，または［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td></td><td rowspan="2">指定可能値</td><td>はい (-size_tidata)</td><td>.tidata.word/.tidata.byte セクションへの割り付け可能サイズを指定します。</td></tr>
<tr><td>いいえ</td><td>.tidata.word/.tidata.byte セクションへの割り付け可能サイズを指定しません。</td></tr>
<tr><td>tidata セクションへの割り付け可能サイズ</td><td colspan="3">.tidata.word/.tidata.byte セクションへの割り付け可能サイズを指定します。
セクション・ファイル・ジェネレータのオプション -size_tidata に相当します。
なお，本プロパティは，［tidata セクションへの割り付け可能サイズを指定する］プロパティで［いいえ］を選択した場合は表示されません。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">256</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td></td><td>指定可能値</td><td colspan="2">0 ～ 256（10 進数）</td></tr>
<tr><td>tidata.byte セクションへの割り付け可能サイズを指定する</td><td colspan="3">.tidata.byte セクションへの割り付け可能サイズを指定するかどうかを選択します。
セクション・ファイル・ジェネレータのオプション -size_tidata_byte に相当します。
なお，本プロパティは，［変数ソートのキー］プロパティで［最適配置 (-O)］，または［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td></td><td rowspan="2">指定可能値</td><td>はい (-size_tidata_byte)</td><td>.tidata.byte セクションへの割り付け可能サイズを指定します。</td></tr>
<tr><td>いいえ</td><td>.tidata.byte セクションへの割り付け可能サイズを指定しません。</td></tr>
</table>

tidata.byte セクションへの割り付け可能サイズ	.tidata.byte セクションへの割り付け可能サイズを指定します。 セクション・ファイル・ジェネレータのオプション -size_tidata_byte に相当します。 なお，本プロパティは，［tidata.byte セクションへの割り付け可能サイズを指定する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	128	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 128（10 進数）	
sidata セクションへの割り付け可能サイズを指定する	.sidata セクションへの割り付け可能サイズを指定するかどうかを選択します。 セクション・ファイル・ジェネレータのオプション -size_sidata に相当します。 なお，本プロパティは，［変数ソートのキー］プロパティで［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-size_sidata)	.sidata セクションへの割り付け可能サイズを指定します。
		いいえ	.sidata セクションへの割り付け可能サイズを指定しません。
sidata セクションへの割り付け可能サイズ	.sidata セクションへの割り付け可能サイズを指定します。 セクション・ファイル・ジェネレータのオプション -size_sidata に相当します。 なお，本プロパティは，［sidata セクションへの割り付け可能サイズを指定する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	32768	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 32768（10 進数）	
sedata セクションへの割り付け可能サイズを指定する	.sedata セクションへの割り付け可能サイズを指定するかどうかを選択します。 セクション・ファイル・ジェネレータのオプション -size_sedata に相当します。 なお，本プロパティは，［変数ソートのキー］プロパティで［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-size_sedata)	.sedata セクションへの割り付け可能サイズを指定します。
		いいえ	.sedata セクションへの割り付け可能サイズを指定しません。
sedata セクションへの割り付け可能サイズ	.sedata セクションへの割り付け可能サイズを指定します。 セクション・ファイル・ジェネレータのオプション -size_sedata に相当します。 なお，本プロパティは，［sedata セクションへの割り付け可能サイズを指定する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	32768	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 32768（10 進数）	

sdata セクションへの割り付け可能サイズを指定する	.sdata セクションへの割り付け可能サイズを指定するかどうかを選択します。 セクション・ファイル・ジェネレータのオプション -size_sdata に相当します。 なお，本プロパティは，［変数ソートのキー］プロパティで［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-size_sdata)	.sdata セクションへの割り付け可能サイズを指定します。
		いいえ	.sdata セクションへの割り付け可能サイズを指定しません。
sdata セクションへの割り付け可能サイズ	.sdata セクションへの割り付け可能サイズを指定します。 セクション・ファイル・ジェネレータのオプション -size_sdata に相当します。 なお，本プロパティは，［sdata セクションへの割り付け可能サイズを指定する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	65536	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 65536（10 進数）	
最適化の対象としない変数	セクション・ファイル生成時に最適化の対象としない変数を指定します。 1 行に 1 つずつ指定します。 セクション・ファイル・ジェネレータのオプション -Xcv に相当します。 指定した変数名はサブプロパティとして表示されます。 なお，本プロパティは，［変数ソートのキー］プロパティで［最適配置 (-O)］，または［全セクションを最適配置 (-O2)］を選択した場合のみ表示されます。		
	デフォルト	最適化の対象としない変数 [*設定数*]	
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能	
	指定可能値	1022 文字までの文字列 1024 個まで指定可能です。	

（4）［その他］

セクション・ファイル・ジェネレートに関するその他の詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，［出力ファイル］カテゴリの［セクション・ファイル・ジェネレータを使用する］プロパティで［いいえ］を選択した場合は表示されません。

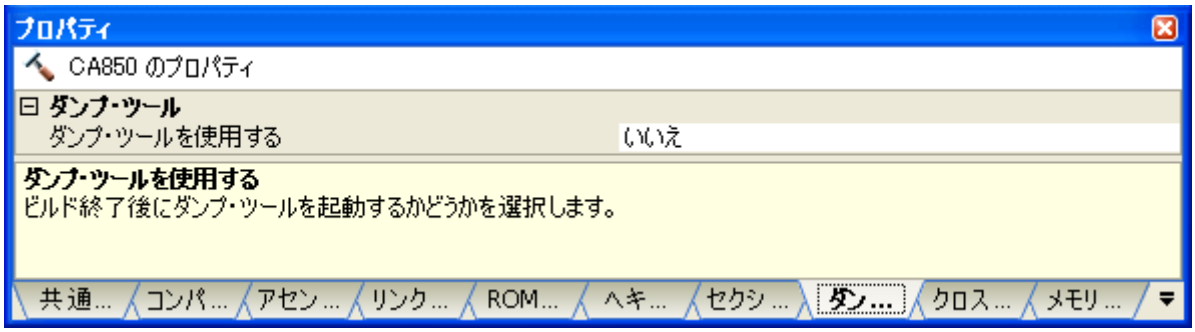
コメント・レベル	セクション・ファイル中に出力するコメントのレベルを選択します。 セクション・ファイル・ジェネレータのオプション -cl に相当します。		
	デフォルト	レベル 1(なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	出力しない (-cl 0)	セクション・ファイル中にコメントを出力しません。
		レベル 1(なし)	セクション・ファイル中にコメント（日付などのファイル生成情報，および変数情報とその説明）を出力します。変数情報は，セクション名，サイズ，変数利用頻度です。
		レベル 2(-cl 2)	レベル 1 に加え，書式ガイドを出力します。
その他の追加オプション	その他に追加するセクション・ファイル・ジェネレータのオプションを入力します。 なお，ここで設定したオプションは，セクション・ファイル・ジェネレータのオプション群の最後に付加されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集	
	指定可能値	259 文字までの文字列	

[ダンプ・オプション] タブ

本タブでは，ダンプ・ツールに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

(1) [ダンプ・ツール]

図 A—12 プロパティ パネル：[ダンプ・オプション] タブ



[各カテゴリの説明]

(1) [ダンプ・ツール]

ダンプ・ツールに関する詳細情報の表示，および設定の変更を行います。

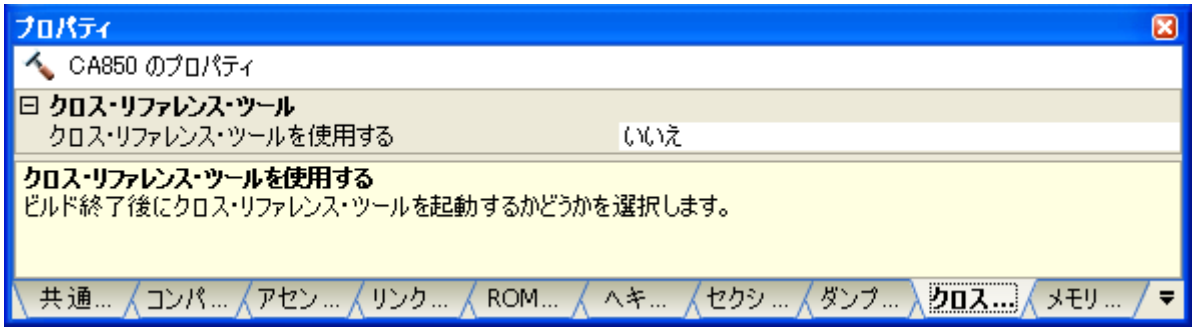
ダンプ・ツールを使用する	ビルド終了後にダンプ・ツールを起動するかどうかを選択します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ビルド終了後にロード・モジュール・ファイルに対してダンプ・ツールを起動します。
		いいえ	ビルド終了後にダンプ・ツールを起動しません。
ダンプ・ツールの追加オプション	追加するダンプ・ツールのオプションを入力します。 ここで設定したオプションは，ダンプ・ツールのオプション群の最後に付加されます。 なお，本プロパティは，[ダンプ・ツールを使用する] プロパティで [はい] を選択した場合のみ表示されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力，または [...] ボタンをクリックし，文字列入力 ダイアログによる編集	
	指定可能値	259 文字までの文字列	

［クロス・リファレンス・オプション］タブ

本タブでは，クロス・リファレンス・ツールに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

（1）［クロス・リファレンス・ツール］

図 A―13　プロパティ パネル：［クロス・リファレンス・オプション］タブ



［各カテゴリの説明］

（1）［クロス・リファレンス・ツール］

クロス・リファレンス・ツールに関する詳細情報の表示，および設定の変更を行います。

クロス・リファレンス・ツールを使用する	ビルド終了後にクロス・リファレンス・ツールを起動するかどうかを選択します。 クロス・リファレンス・ツールを起動する場合，プロジェクトに登録しているすべてのC ソース・ファイルを入力とし，すべての情報（クロス・リファレンス情報，タグ・ジャンプ情報，コール・ツリー，関数計量，コール・データベース）をテキスト形式，および CSV 形式の各ファイルに出力します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ビルド終了後にクロス・リファレンス・ツールを起動します。
		いいえ	ビルド終了後にクロス・リファレンス・ツールを起動しません。
クロス・リファレンス・ツールの追加オプション	追加するクロス・リファレンス・ツールのオプションを入力します。 なお，ここで設定したオプションは，クロス・リファレンス・ツールのオプション群の最後に付加されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集	
	指定可能値	259 文字までの文字列	

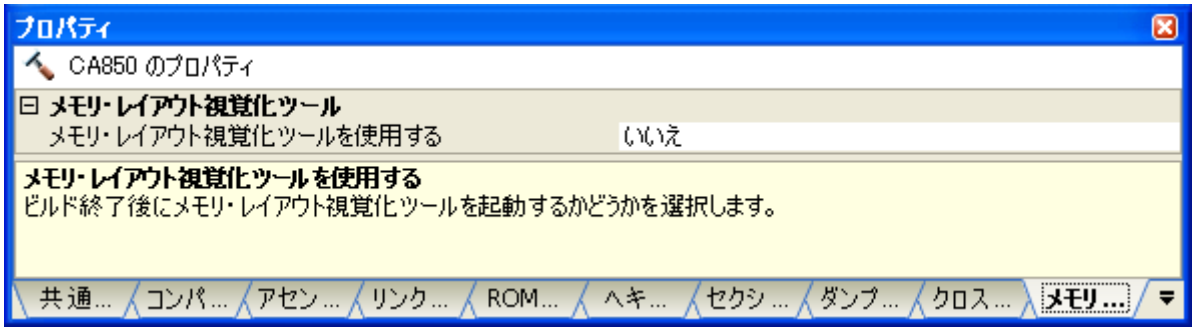
［メモリ・レイアウト視覚化オプション］タブ

本タブでは，メモリ・レイアウト視覚化ツールに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

（1）［メモリ・レイアウト視覚化ツール］

注意　本タブは，ライブラリ用のプロジェクトの場合は表示されません。

図 A―14　プロパティ パネル：［メモリ・レイアウト視覚化オプション］タブ



［各カテゴリの説明］

（1）［メモリ・レイアウト視覚化ツール］

メモリ・レイアウト視覚化ツールに関する詳細情報の表示，および設定の変更を行います。

メモリ・レイアウト視覚化ツールを使用する	ビルド終了後にメモリ・レイアウト視覚化ツールを起動するかどうかを選択します。 メモリ・レイアウト視覚化ツールを起動する場合，オブジェクト・ファイル（*.out）を入力とし，メモリ・マップ表をテキスト形式，および CSV 形式のファイルに出力します。 入力ファイルは，リンカが出力したオブジェクト・ファイル（*.out）となります。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ビルド終了後にメモリ・レイアウト視覚化ツールを起動します。
		いいえ	ビルド終了後にメモリ・レイアウト視覚化ツールを起動しません。

メモリ・レイアウト視覚化ツールの追加オプション	追加するメモリ・レイアウト視覚化ツールのオプションを入力します。 なお，ここで設定したオプションは，メモリ・レイアウト視覚化ツールのオプション群の最後に付加されます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	259 文字までの文字列

［ビルド設定］タブ

本タブでは，各 C ソース・ファイル，アセンブラ・ソース・ファイル，リンク・ディレクティブ・ファイル，セクション・ファイル，オブジェクト・ファイル，アーカイブ・ファイルに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

（1）［ビルド］

図 A—15　プロパティ パネル：［ビルド設定］タブ（C ソース・ファイルを選択した場合）

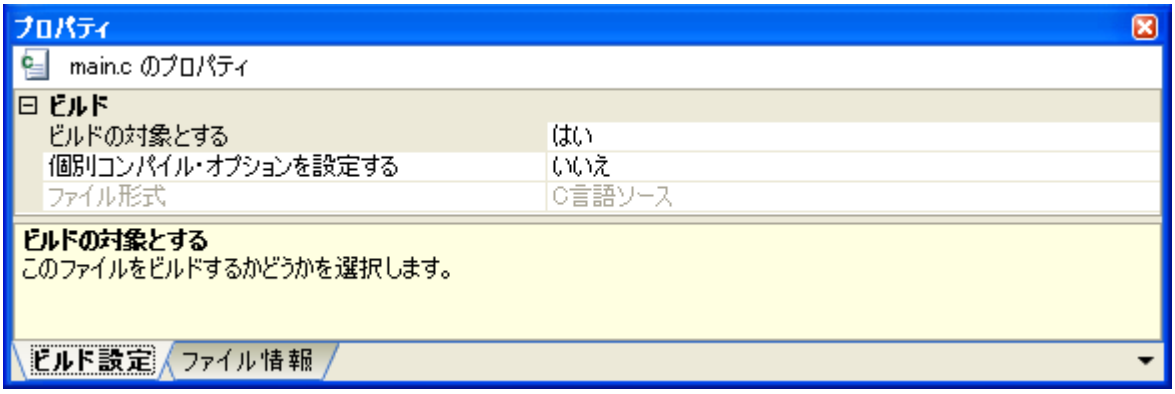


図 A—16　プロパティ パネル：［ビルド設定］タブ（アセンブラ・ソース・ファイルを選択した場合）

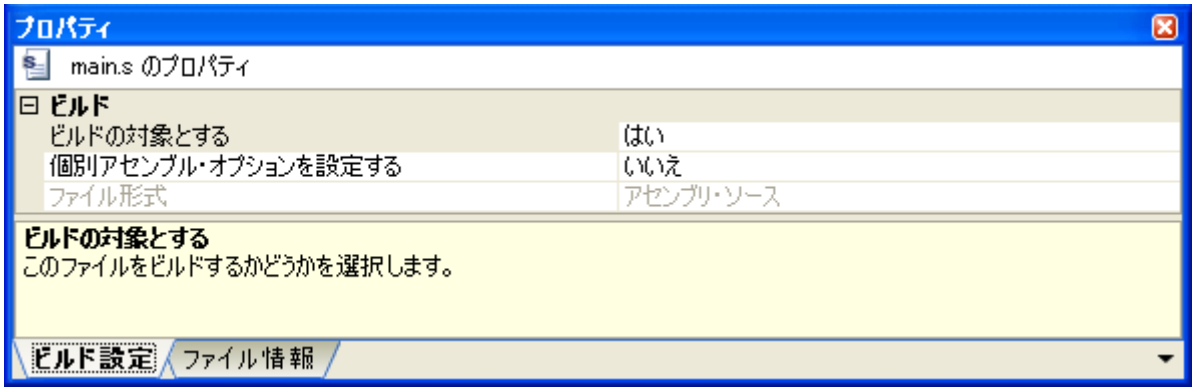


図 A—17 プロパティ パネル：［ビルド設定］タブ（リンク・ディレクティブ・ファイルを選択した場合）

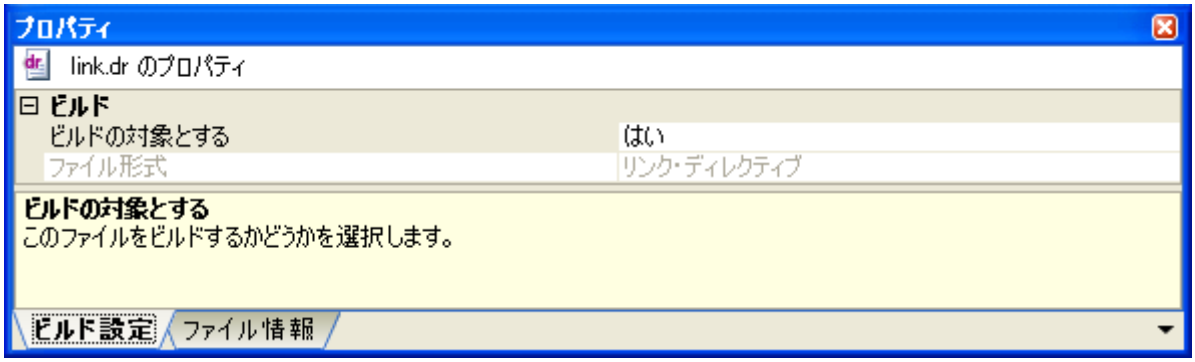


図 A—18 プロパティ パネル：［ビルド設定］タブ（セクション・ファイルを選択した場合）

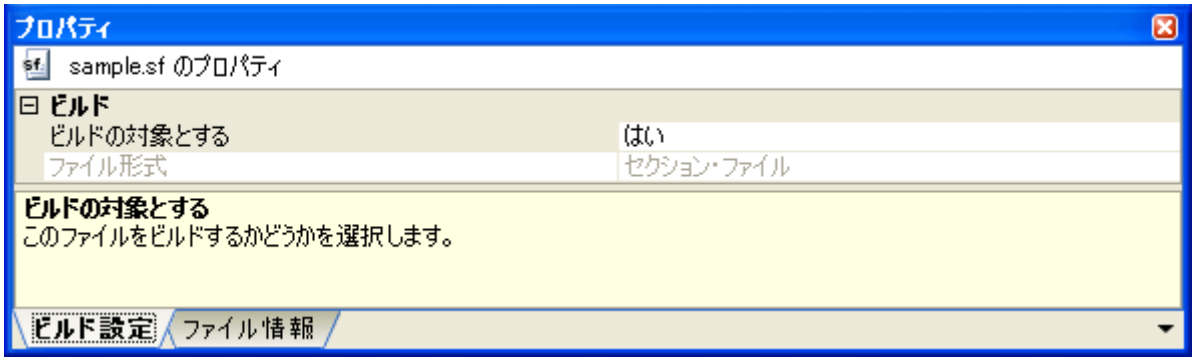


図 A—19 プロパティ パネル：［ビルド設定］タブ（オブジェクト・ファイルを選択した場合）

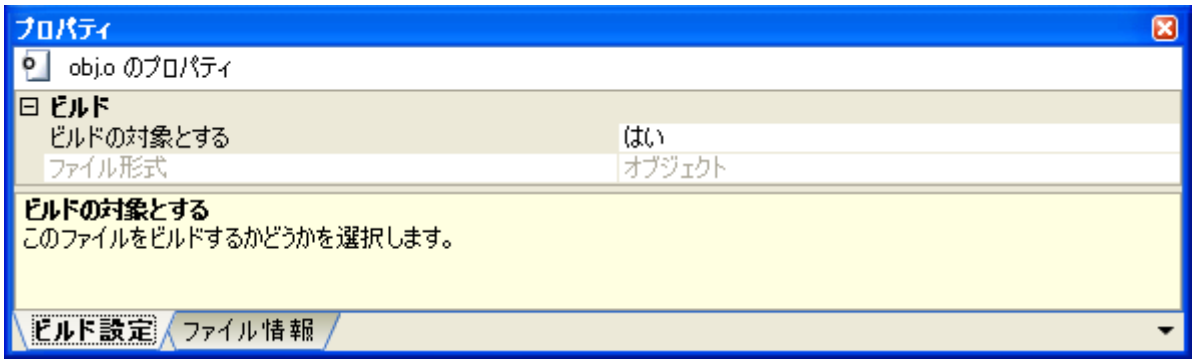
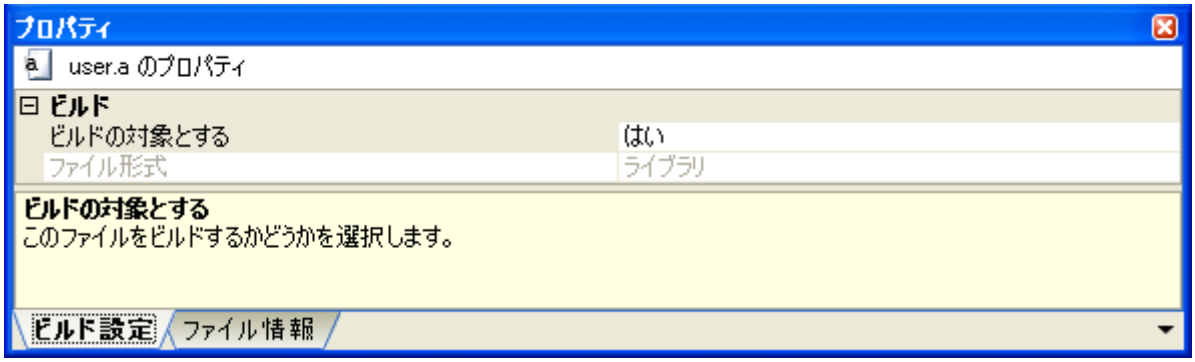


図 A―20　プロパティ パネル：［ビルド設定］タブ（アーカイブ・ファイルを選択した場合）



［各カテゴリの説明］

(1) ［ビルド］

ビルドに関する詳細情報の表示，および設定の変更を行います。

ビルドの対象とする	選択しているファイルをビルド対象とするかどうかを選択します。		
	デフォルト	はい	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	選択しているファイルをビルド対象とします。
		いいえ	選択しているファイルをビルド対象としません。
個別コンパイル・オプションを設定する	選択しているＣソース・ファイルにプロジェクトの設定とは異なるコンパイル・オプションを設定するかどうかを選択します。 なお，本プロパティは，プロジェクト・ツリー パネルでＣソース・ファイルを選択し，［ビルドの対象とする］プロパティで［はい］を選択した場合のみ表示されます。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	選択しているＣソース・ファイルにプロジェクトの設定とは異なるオプションを設定します。
		いいえ	選択しているＣソース・ファイルにプロジェクトの設定とは異なるオプションを設定しません。

<table>
<tr><td rowspan="5">個別アセンブル・オプションを設定する</td><td colspan="3">選択しているアセンブラ・ソース・ファイルにプロジェクトの設定とは異なるアセンブル・オプションを設定するかどうかを選択します。
なお，本プロパティは，プロジェクト・ツリー パネルでアセンブラ・ソース・ファイルを選択し，［ビルドの対象とする］プロパティで［はい］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">いいえ</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい</td><td>選択しているアセンブラ・ソース・ファイルにプロジェクトの設定とは異なるオプションを設定します。</td></tr>
<tr><td>いいえ</td><td>選択しているアセンブラ・ソース・ファイルにプロジェクトの設定とは異なるオプションを設定しません。</td></tr>
<tr><td rowspan="3">ファイル形式</td><td colspan="3">選択しているファイルの形式を表示します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">C 言語ソース（C ソース・ファイルを選択している場合）
アセンブリ・ソース（アセンブラ・ソース・ファイルを選択している場合）
リンク・ディレクティブ（リンク・ディレクティブ・ファイルを選択している場合）
セクション・ファイル（セクション・ファイルを選択している場合）
オブジェクト（オブジェクト・ファイルを選択している場合）
ライブラリ（アーカイブ・ファイルを選択している場合）</td></tr>
<tr><td>変更方法</td><td colspan="2">変更不可</td></tr>
</table>

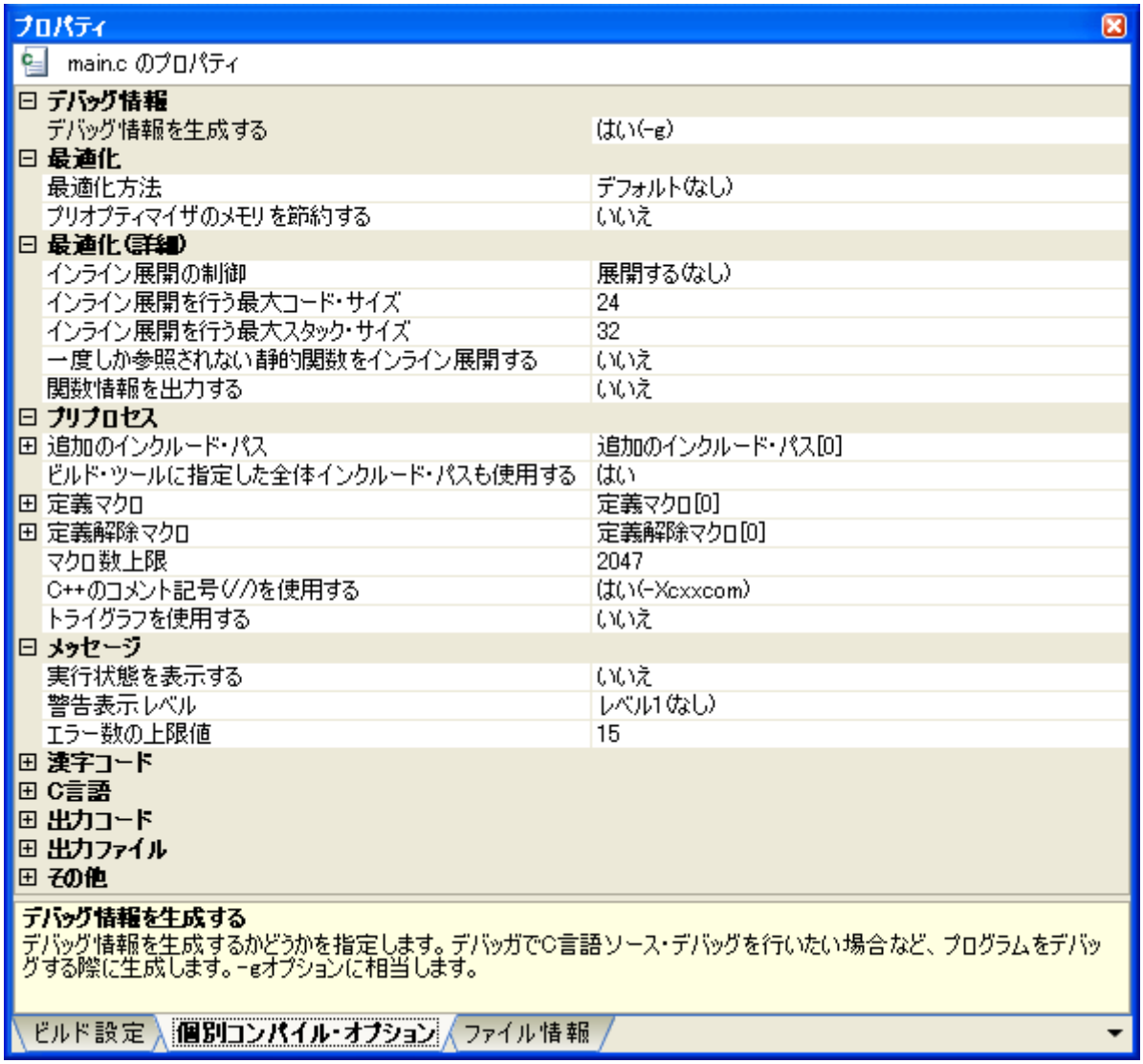
［個別コンパイル・オプション］タブ

本タブでは，１つのＣソース・ファイルに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

なお，本タブは，［コンパイル・オプション］タブの設定内容を継承します。［コンパイル・オプション］タブと異なる値を設定した場合は，プロパティが太字表示となります。

備考　本タブは，［ビルド設定］タブの［ビルド］カテゴリの［個別コンパイル・オプションを設定する］プロパティで［はい］を選択した場合のみ表示されます。

図 A―21　プロパティ パネル：［個別コンパイル・オプション］タブ



［各カテゴリの説明］

(1)［デバッグ情報］

デバッグ情報に関する詳細情報の表示，および設定の変更を行います。

デバッグ情報を生成する	ソース・デバッガ用のシンボル情報を出力し，ソース・レベル・デバッグを可能にするかどうかを選択します。 コンパイラのオプション -g に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-g)	ソース・デバッガ用のシンボル情報を出力します。
		いいえ	ソース・デバッガ用のシンボル情報を出力しません。

(2)［最適化］

最適化に関する詳細情報の表示，および設定の変更を行います。

最適化方法	コンパイルの最適化の種類を選択します。 コンパイラのオプション -O* に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	デバッグ優先 (-Od)	デバッグを優先して最適化を行います。 ROM 容量や実行速度に着目せず，ソース・デバッグに着眼したコードを生成します。
		デフォルト (なし)	ソース・デバッグに着眼したコードを生成します。 ソース・デバッグに影響のない範囲で最適化を行います。
		標準最適化 (-Og)	適度な最適化を行います。 ほとんどの場合で C ソース・デバッグが可能となる最適化を行います。
		高度な最適化 (-O)	高度な最適化を行います。 ROM 容量に着目した最適化を行います。
		より高度な最適化 (オブジェクト・サイズ優先)(-Os)	より高度な最適化（オブジェクト・サイズ優先）を行います。 ROM 容量を最も重視した最大限の最適化を行います。
		より高度な最適化 (実行速度優先)(-Ot)	より高度な最適化（実行速度優先優先）を行います。 実行速度を最も重視した最大限の最適化を行います。
プリオプティマイザのメモリを節約する	コンパイル時のプリオプティマイザのメモリ使用量を節約するかどうかを選択します。 マシンのメモリが不足し，コンパイルが正常に終了しない場合に，本プロパティを指定します。 コンパイラのオプション -Wp,-D に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wp,-D)	コンパイル時のプリオプティマイザのメモリ使用量を節約します。 ただし，コンパイル速度は低下します。
		いいえ	コンパイル時のプリオプティマイザのメモリ使用量の節約を指定しません。

機種依存最適化部のメモリを節約する	コンパイル時の機種依存最適化部のメモリ使用量を節約するかどうかを選択します。 マシンのメモリが不足し，コンパイルが正常に終了しない場合に，本プロパティを指定します。 コンパイラのオプション -Wi,-D に相当します。 なお，本プロパティは，［最適化方法］プロパティで［デバッグ優先 (-Od)］，［デフォルト (なし)］，［標準最適化 (-Og)］のいずれかを選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wi,-D)	コンパイル時の機種依存最適化部のメモリ使用量を節約します。 ただし，コンパイル速度は低下します。
		いいえ	コンパイル時の機種依存最適化部のメモリ使用量の節約を指定しません。

(3)［最適化 (詳細)］

最適化に関する詳細情報の表示，および設定の変更を行います。

インライン展開の制御	インライン展開を行うかどうかを選択します。 コンパイラのオプション -Wp,-N に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	展開する (なし)	インライン展開を行います。
		‘inline’関数のみ展開する (-Wp,-inline)	#pragma inline 指定した関数のみインライン展開を行います。
		展開しない (-Wp,-no_inline)	#pragma inline 指定した関数を含む，すべての関数のインライン展開を指定しません。
インライン展開を行う最大コード・サイズ	インライン展開を行う関数の中間言語サイズの最大値を指定します。 指定サイズより大きい関数はインライン展開を行いません。 コンパイラのオプション -Wp,-N に相当します。 サイズの目安は，［関数情報を出力する］プロパティの指定により出力した関数情報ファイルを参照してください。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 9999（10 進数）	

インライン展開を行う最大スタック・サイズ	インライン展開を行う関数の中間言語でのスタック・サイズの最大値（バイト）を指定します。 指定サイズより大きい関数はインライン展開を行いません。 コンパイラのオプション -Wp,-G に相当します。 サイズの目安は，［関数情報を出力する］プロパティの指定により出力した関数情報ファイルを参照してください。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0 ～ 9999（10 進数）	
一度しか参照されない静的関数をインライン展開する	一度しか参照されない静的関数をインライン展開するかどうかを選択します。 コンパイラのオプション -Wp,-S に相当します。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wp,-S)	一度しか参照されない静的関数のインライン展開を行います。
		いいえ	一度しか参照されない静的関数のインライン展開を指定しません。
関数情報を出力する	各関数の中間言語でのコード・サイズとスタック・サイズをファイルに出力するかどうかを選択します。 出力する情報は，［インライン展開を行う最大コード・サイズ］プロパティ，および［インライン展開を行う最大スタック・サイズ］プロパティで指定する値の目安となります。 コンパイラのオプション -Wp,-I に相当します。 なお，本プロパティは，［インライン展開の制御］プロパティで［展開しない (-Wp,-no_inline)］を選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wp,-I)	各関数の中間言語でのコード・サイズとスタック・サイズをファイルに出力します。
		いいえ	各関数の中間言語でのコード・サイズとスタック・サイズのファイルへの出力を指定しません。

<table>
<tr><td>関数情報ファイル名</td><td colspan="3">各関数の中間言語でのコード・サイズとスタック・サイズを出力するファイル名を指定します。
コンパイラのオプション -Wp,-I に相当します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
空欄の場合は，全体オプションの設定値を指定したものとみなします。
なお，本プロパティは，［関数情報を出力する］プロパティで［いいえ］を選択した場合は表示されません。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，関数情報ファイルを指定 ダイアログによる編集</td></tr>
<tr><td></td><td>指定可能値</td><td colspan="2">259 文字までの文字列</td></tr>
<tr><td>ループの展開を行う</td><td colspan="3">for，while などのループの展開を行うかどうかを選択します。
コンパイラのオプション -Wo,-Ol,-Xlo に相当します。
なお，本プロパティは，［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合のみ表示されます。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td></td><td rowspan="3">指定可能値</td><td>はい (展開数は自動で調整)(-Wo,-Ol)</td><td>［ループ展開最大数］プロパティで指定した回数以下でコード・サイズが最小になるよう，ループの展開を行います。</td></tr>
<tr><td>はい (展開数は固定)(-Wo,-Ol,-Xlo)</td><td>［ループ展開最大数］プロパティで指定した回数でループの展開を行います。</td></tr>
<tr><td>いいえ (-Wo,-Ol0)</td><td>ループの展開を指定しません。</td></tr>
<tr><td>ループ展開最大数</td><td colspan="3">for，while などのループを展開する最大数を指定します。
コンパイラのオプション -Wo,-Ol に相当します。
なお，本プロパティは，［ループの展開を行う］プロパティで［いいえ (-Wo,-Ol0)］を選択した場合は表示されません。</td></tr>
<tr><td></td><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td></td><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力</td></tr>
<tr><td></td><td>指定可能値</td><td colspan="2">0 ～ 999（10 進数）</td></tr>
</table>

サイズ優先で分岐命令を出力する	分岐命令をコード・サイズ優先で並べてコードを出力するかどうかを選択します。 コンパイラのオプション -Wo,-XFo に相当します。 なお，本プロパティは，［最適化方法］プロパティで［デバッグ優先 (-Od)］，または［デフォルト (なし)］を選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wo,-XFo)	分岐命令をコード・サイズ優先で並べてコードを出力します。
		いいえ	分岐命令に対してデバッグ情報を優先したコードを出力します。
アライメントを詰める	分岐先ラベルを整列する最適化を抑止するかどうかを選択します。 なお，本プロパティは，［最適化方法］プロパティで［高度な最適化 (-O)］，［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］，または［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合のみ表示されます。 ただし，［高度な最適化 (-O)］，または［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］を選択した場合は，この機能が含まれるため，常に［はい (-Wi,-P)］が選択されます。 コンパイラのオプション -Wi,-P に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wi,-P)	分岐先ラベルを整列する最適化を抑止します。 実行コード・サイズを小さくすることができます。
		いいえ	分岐先ラベルを整列する最適化の抑止を指定しません。
強力な最適化を行う	データ・フロー解析を厳密に行い，最も強力な最適化を行うかどうかを選択します。 高度な最適化を行う場合に，さらに強力なデータ・フロー解析を行いたい場合に，本プロパティを指定します。 なお，本プロパティは，［最適化方法］プロパティで［高度な最適化 (-O)］，［より高度な最適化 (オブジェクト・サイズ優先)(-Os)］，または［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合のみ表示されます。 コンパイラのオプション -Wi,-O4 に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Wi,-O4)	データ・フロー解析を厳密に行い，最も強力な最適化を行います。 ただし，コンパイル速度はかなり低下します。
		いいえ	強力な最適化を指定しません。

(4)［プリプロセス］

プリプロセスに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="4">追加のインクルード・パス</td><td colspan="3">コンパイル時の追加のインクルード・パスを指定します。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
%ProjectName% ：プロジェクト名に置換します。
%MicomToolPath% ：本製品のインストール・フォルダの絶対パスに置換します。
このオプションを省略した場合，コンパイラの標準フォルダのみ検索します。なお，パスはプロジェクト・フォルダを基点とします。
コンパイラのオプション -I に相当します。
指定したインクルード・パスはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">追加のインクルード・パス [定義数]</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，パス編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">259 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td rowspan="5">ビルド・ツールに指定した全体インクルード・パスも使用する</td><td colspan="3">使用するビルド・ツールの［コンパイル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで指定したインクルード・パスも使用してコンパイルするかどうかを選択します。
コンパイラのオプション -I に相当します。
以下の順番で，-i オプションにパスを追加します。
- ［追加のインクルード・パス］プロパティに指定したパス
- ［コンパイル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで指定したパス
- ［コンパイル・オプション］タブの［プリプロセス］カテゴリの［システム・インクルード・パス］プロパティで指定したパス</td></tr>
<tr><td>デフォルト</td><td colspan="2">はい</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい</td><td>使用するビルド・ツールのプロパティで指定したインクルード・パスも使用してコンパイルします。</td></tr>
<tr><td>いいえ</td><td>使用するビルド・ツールのプロパティで指定したインクルード・パスを使用しません。</td></tr>
<tr><td rowspan="4">定義マクロ</td><td colspan="3">定義したいマクロ名を指定します。
「マクロ名 = 定義値」の形式で 1 行に 1 つずつ指定します。「= 定義値」の部分は省略可能で，省略した場合，定義値を 1 とします。
コンパイラのオプション -D に相当します。
指定したマクロはサブプロパティとして表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">[...] ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td colspan="2">256 文字までの文字列
256 個まで指定可能です。</td></tr>
</table>

定義解除マクロ	定義解除したいマクロ名を指定します。 「*マクロ名*」の形式で 1 行に 1 つずつ指定します。 コンパイラのオプション -U に相当します。 指定したマクロはサブプロパティとして表示されます。		
	デフォルト	*全体オプションの設定値*	
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能	
	指定可能値	256 文字までの文字列 256 個まで指定可能です。	
マクロ数上限	マクロ識別子の上限を指定します。 コンパイラのオプション -Xm に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	1 ～ 999999（10 進数）	
C++ のコメント記号 (//) を使用する	通常のコメントのほかに，C++ のコメント・スタイル（“//”から行末まで）を有効にするかどうかを選択します。 コンパイラのオプション -Xcxxcom に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xcxxcom)	通常のコメントのほかに，C++ のコメント・スタイル（“//”から行末まで）を有効にします。
		いいえ	C++ のコメント・スタイル（“//”から行末まで）を有効にしません。
プリプロセッサの出力ファイルにコメントを保存する	C 言語のソース・プログラムの前処理の出力に，ソース・プログラムのコメントを含めるかどうかを選択します。 コンパイラのオプション -C に相当します。 なお，本プロパティは，［出力ファイル］カテゴリの［プリプロセス処理したソースを出力する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-C)	C 言語のソース・プログラムの前処理の出力に，ソース・プログラムのコメントを含めます。
		いいえ	C 言語のソース・プログラムの前処理の出力に，ソース・プログラムのコメントを含めません。

トライグラフを使用する	トライグラフ系列を置換するかどうかを選択します。 トライグラフとは，ANSI 規格で規定された，単一文字に置換される 3 文字（トライグラフ）系列のことです。 コンパイラのオプション -t に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-t)	トライグラフ系列を置換します。
		いいえ	トライグラフ系列を置換しません。

(5)［メッセージ］

メッセージに関する詳細情報の表示，および設定の変更を行います。

実行状態を表示する	ビルド時にコンパイラの実行状態を出力 パネルに表示するかどうかを選択します。 コンパイラのオプション -v に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-v)	ビルド時にコンパイラの実行状態を表示します。
		いいえ	ビルド時にコンパイラの実行状態を表示しません。
警告表示レベル	コンパイル時の警告表示レベルを選択します。 コンパイラのオプション -w に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	出力しない (-w)	警告メッセージを出力しません。
		レベル 1(なし)	通常の警告メッセージを出力します。
		レベル 2(-w2)	詳細な警告メッセージを出力します。
エラー数の上限値	エラー・メッセージの最大出力数を指定します。 コンパイラのオプション -err_limit に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	15 ～ 50（10 進数）	

(6)［漢字コード］

漢字コードに関する詳細情報の表示，および設定の変更を行います。

ソースの漢字コード	入力ファイル中の日本語コメント，文字列に対し，使用する漢字コードを選択します。 コンパイラのオプション -Xk に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	Shift_JIS(なし)	ソースの漢字コードを Shift_JIS と解釈します。
		なし (-Xk=none)	ソースに漢字コードがないと解釈します。 コードを保証しません。
		EUC-JP(-Xk=euc)	ソースの漢字コードを EUC-JP と解釈します。
ターゲットの漢字コード	日本語の文字列に対し，変換する漢字コードを選択します。 アプリケーション開発時に使用した漢字コードをターゲットで変更したい場合に，本プロパティを設定します。 コンパイラのオプション -Xkt に相当します。		
	デフォルト	無変換 (なし)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	無変換 (なし)	ターゲットの漢字コードの変換を行いません。 コードを保証しません。
		Shift_JIS(-Xkt=sjis)	ターゲットの漢字コードを Shift_JIS に変換します。
		EUC-JP(-Xkt=euc)	ターゲットの漢字コードを EUC-JP に変換します。

(7) [C 言語]

C 言語に関する詳細情報の表示，および設定の変更を行います。

ビット・フィールドの符号	型指定子（signed，unsigned）の付かない int 型のビット・フィールドに対し，符号付きとするか，符号なしとするかを選択します。 コンパイラのオプション -Xbitfield に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	符号付き	型指定子の付かない int 型のビット・フィールドに対し，符号付きとします。
		符号なし (-Xbit-field=unsigned)	型指定子の付かない int 型のビット・フィールドに対し，符号なしとします。
char の符号	型指定子（signed，unsigned）の付かない char 型に対し，符号付きとするか，符号なしとするかを選択します。 コンパイラのオプション -Xchar に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	符号付き	型指定子の付かない char 型に対し，符号付きとします。
		符号なし (-Xchar=unsigned)	型指定子の付かない char 型に対し，符号なしとします。

enum の型	列挙型に対し，どの整数型と整合するかを選択します。 コンパイラのオプション -Xenum_type に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	int(なし)	列挙型に対し，int 型と整合します。
		signed char(-Xenum_type=char)	列挙型に対し，signed char 型と整合します。
		unsigned char(-Xenum_type=uchar)	列挙型に対し，unsigned char 型と整合します。
		short(-Xenum_type=short)	列挙型に対し，short 型と整合します。
		unsigned short(-Xenum_type=ushort)	列挙型に対し，unsigned short 型と整合します。
ANSI 規格に厳密に合わせてコンパイルする	コンパイラの処理を ANSI 規格に厳密に合わせ，規格に反する記述に対してエラーや警告メッセージを表示するかどうかを選択します。 コンパイラのオプション -ansi に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-ansi)	コンパイラの処理を ANSI 規格に厳密に合わせ，規格に反する記述に対してエラーや警告メッセージを表示します。
		いいえ	従来の C 言語の仕様との両立性を持たせ，警告メッセージを表示してコンパイラの処理を続行します。
CC78K の拡張機能を使用する	78K マイクロコントローラ C コンパイラ CC78K 互換の拡張機能を有効にするかどうかを選択します。 コンパイラのオプション -cc78k に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-cc78k)	CC78K 互換の拡張機能を有効にします。
		いいえ	CC78K 互換の拡張機能を有効にしません。
整数演算を厳密に行う	ANSI 規格に厳密な乗除算処理を行うため，16 ビット・データ以下の整数に対し，mulh，divh 命令を使用せず，ランタイム・ライブラリ ___mul ／ ___mulu，___div ／ ___divu，または mul，mulu，div，divu 命令を使用するかどうかを選択します。 コンパイラのオプション -Xe に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xe)	16 ビット・データ以下の整数に対し，ランタイム・ライブラリ ___mul ／ ___mulu，___div ／ ___divu を使用します。
		いいえ	16 ビット・データ以下の整数に対し，mulh，divh 命令を使用します。

（8）［出力コード］

出力コードに関する詳細情報の表示，および設定の変更を行います。

<table>
<tr><td rowspan="5">プロローグ／エピローグ・ライブラリを使用する</td><td colspan="3">関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にするかどうかを選択します。
コンパイラのオプション -Xpro_epi_runtime に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="3">指定可能値</td><td>自動選択 (なし)</td><td>［最適化］カテゴリの［最適化方法］プロパティで［より高度な最適化 (実行速度優先)(-Ot)］を選択した場合は［いいえ (-Xpro_epi_runtime=off)］，それ以外を選択した場合は［はい (-Xpro_epi_runtime=on)］となります。</td></tr>
<tr><td>いいえ (-Xpro_epi_runtime=off)</td><td>関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にしません。</td></tr>
<tr><td>はい (-Xpro_epi_runtime=on)</td><td>関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にします。</td></tr>
<tr><td rowspan="6">switch 文の出力コードの選択</td><td colspan="3">プログラム中の switch 文のコード出力方式を選択します。
コンパイラのオプション -Xcase に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="4">指定可能値</td><td>自動選択 (なし)</td><td>コンパイラが最適と思われる形式を自動判断します。</td></tr>
<tr><td>if-else(-Xcase=ifelse)</td><td>プログラム中の switch 文を case 文の並びに沿った if-else 文と同じ形で出力します。
上から順に比較するので，合致する頻度が多い順に case 文を書いているときやラベル数が少ないときは余計な比較が減り，実行速度の向上につながります。</td></tr>
<tr><td>バイナリ・サーチ (-Xcase=binary)</td><td>プログラム中の switch 文をバイナリ・サーチ形式で出力します。
バイナリ・サーチ・アルゴリズムに用いて合致する case 文を探すため，ラベル数が多いときは，どの case 文も同じくらいの速さで見つけることができます。</td></tr>
<tr><td>テーブル分岐 (-Xcase=table)</td><td>プログラム中の switch 文をテーブル・ジャンプ方式で出力します。
case 文の値を基にインデックス化したテーブルを参照し，switch 文の値により case ラベルを選択し，処理を行います。どの case 文にも同じくらい速く分岐します。ただし，case 値が連続していないときは無駄な領域ができます。</td></tr>
</table>

switch テーブルのラベルサイズ (バイト)	switch 文の case ラベルに対する分岐テーブルの 1 ラベルあたりのサイズを選択します。 コンパイラのオプション -Xword_switch に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	2 バイト (なし)	switch 文の case ラベルに対する分岐テーブルを 1 ラベルあたり 2 バイトで生成します。
		4 バイト (-Xword_switch)	switch 文の case ラベルに対する分岐テーブルを 1 ラベルあたり 4 バイトで生成します。 長い switch 文のため，コンパイル・エラーとなる場合に選択します。
構造体パッキング	構造体のパッキング値を選択します。 構造体メンバをメンバ型に応じてアライメントすることなく，指定したアライメント値を用いることができます。データ・サイズは小さくすることができますが，コード・サイズは大きくなります。 コンパイラのオプション -Xpack に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	1 バイト (-Xpack=1)	構造体メンバを 1 バイトのアライメントで整列します。
		2 バイト (-Xpack=2)	構造体メンバを 2 バイトのアライメントで整列します。
		4 バイト (-Xpack=4)	構造体メンバを 4 バイトのアライメントで整列します。
		8 バイト (なし)	構造体メンバを 8 バイトのアライメントで整列します。
strcpy ／ strcmp の展開を行う	配列（文字列を含む），および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開するかどうかを選択します。 オブジェクトの実行速度は高速になりますが，コード・サイズは増大します。 コンパイラのオプション -Xi に相当します。 なお，本プロパティは，［構造体パッキング］プロパティで［8 バイト (なし)］を選択した場合のみ表示されます。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xi)	配列（文字列を含む），および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開します。
		いいえ	関数 strcpy()，または strcmp() の呼び出しをインライン展開しません。

<table>
<tr><td rowspan="5">ポインタのバイト・アクセスを行う</td><td colspan="3">構造体の間接アドレス・アクセスをバイト単位でアクセスするかどうかを選択します。
構造体パッキング機能で，制限に該当するような場合に，本プロパティを設定します。
コンパイラのオプション -Xbyte に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Xbyte)</td><td>構造体の間接アドレス・アクセスをバイト単位でアクセスします。</td></tr>
<tr><td>いいえ</td><td>構造体の間接アドレス・アクセスをバイト単位でアクセスしません。</td></tr>
<tr><td rowspan="5">アセンブリ言語ソースにコメントを出力する</td><td colspan="3">出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力するかどうかを選択します。
コンパイラのオプション -Xc に相当します。
なお，本プロパティは，［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択した場合，または［アセンブル・リストを出力する］プロパティで［はい (-Fv)］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Xc)</td><td>出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力します。</td></tr>
<tr><td>いいえ</td><td>出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力しません。</td></tr>
<tr><td rowspan="5">割り込みの分岐命令に jmp 命令を使う</td><td colspan="3">C 言語で定義された割り込み関数に対し，jmp 命令を用いるかどうかを選択します。
コンパイラのオプション -Xj に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Xj)</td><td>C 言語で定義された割り込み関数に対し，jmp 命令を用います。</td></tr>
<tr><td>いいえ</td><td>C 言語で定義された割り込み関数に対し，jr 命令を用います。</td></tr>
<tr><td rowspan="5">word の bit 命令変更を抑止する</td><td colspan="3">ld.w/ld.h，st.w/st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を禁止するかどうかを選択します。
コンパイラのオプション -Xno_word_bitop に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Xno_word_bitop)</td><td>ld.w/ld.h，st.w/st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を禁止します。</td></tr>
<tr><td>いいえ</td><td>ld.w/ld.h，st.w/st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を行います。</td></tr>
</table>

(9)［出力ファイル］

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

オブジェクト・ファイル名	コンパイル後に生成されるオブジェクト・ファイルのファイル名を指定します。 ".o" 以外の拡張子を指定することはできません。拡張子を省略した場合は，".o" が自動的に付加されます。 空欄の場合は，C ソース・ファイルの拡張子 .c を .o で置き換えたファイル名で保存します。 コンパイラのオプション -o に相当します。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	259 文字までの文字列	
アセンブリ・ファイルを出力する	C ソースのコンパイル結果のアセンブラ・ソース・ファイルを出力するかどうかを選択します。 コンパイラのオプション -Fs に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Fs)	アセンブラ・ソース・ファイルを出力します。
		いいえ	アセンブラ・ソース・ファイルを出力しません。
アセンブリ・ファイルの出力フォルダ	アセンブラ・ソース・ファイルの出力先フォルダを指定します。 アセンブラ・ソース・ファイルは，ソース・ファイルの拡張子を .s で置き換えたファイル名で保存します。 コンパイラのオプション -Fs に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName% ：ビルド・モード名に置換します。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択した場合のみ表示されます。		
	デフォルト	*全体オプションの設定値*	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	
アセンブル・リストを出力する	C ソースのコンパイル結果のアセンブル・リストを出力するかどうかを選択します。 コンパイラのオプション -Fv に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Fv)	アセンブル・リストを出力します。
		いいえ	アセンブル・リストを出力しません。

<table>
<tr><td rowspan="4">アセンブル・リストの出力フォルダ</td><td colspan="3">アセンブル・リストの出力先フォルダを指定します。
アセンブル・リストは，ソース・ファイルの拡張子を .v で置き換えたファイル名で保存します。
コンパイラのオプション -Fv に相当します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
空欄の場合は，プロジェクト・フォルダを指定したものとみなします。
なお，本プロパティは，［アセンブル・リストを出力する］プロパティで［はい (-Fv)］を選択した場合のみ表示されます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
<tr><td rowspan="4">頻度情報ファイルを出力する</td><td colspan="3">セクション・ファイル・ジェネレータで使用する変数の頻度情報ファイルを出力するかどうかを選択します。
コンパイラのオプション -Xcre_sec_data に相当します。</td></tr>
<tr><td>デフォルト</td><td colspan="2">全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td colspan="2">ドロップダウン・リストによる選択</td></tr>
<tr><td rowspan="2">指定可能値</td><td>はい (-Xcre_sec_data)</td><td>変数の頻度情報ファイルを出力します。</td></tr>
<tr><td>いいえ</td><td>変数の頻度情報ファイルを出力しません。</td></tr>
<tr><td rowspan="4">頻度情報ファイルの出力フォルダ</td><td colspan="3">頻度情報ファイルの出力先フォルダを指定します。
頻度情報ファイルは，ソース・ファイルの拡張子を .sec で置き換えたファイル名で保存します。
コンパイラのオプション -Xcre_sec_data に相当します。
相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。
絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。
埋め込みマクロとして次のマクロ名があります。
%BuildModeName% ：ビルド・モード名に置換します。
空欄の場合は，プロジェクト・フォルダを指定したものとみなします。
なお，本プロパティは，［頻度情報ファイルを出力する］プロパティで［いいえ］を選択した場合は表示されません。</td></tr>
<tr><td>デフォルト</td><td colspan="2">空欄</td></tr>
<tr><td>変更方法</td><td colspan="2">テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td colspan="2">247 文字までの文字列</td></tr>
</table>

プリプロセス処理したソースを出力する	C ソース・プログラムに対し，前処理（プリプロセス処理）のみ実行するコマンドをコンパイル前に実行するかどうかを選択します。 結果は，C ソース・ファイルの拡張子 .c を .i で置き換えたファイル名で出力します。 ただし，ソース・プログラムの行番号表示やファイル名表示は出力しません。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-P)	C ソース・プログラムに対し，プリプロセス処理のみ実行した結果を出力します。
		いいえ	C ソース・プログラムに対し，プリプロセス処理のみ実行した結果を出力しません。

（10）［その他］

コンパイルに関するその他の詳細情報の表示，および設定の変更を行います。

コンパイル前に実行するコマンド	コンパイル処理前に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder% ：出力フォルダの絶対パスに置換します。 %OutputFile% ：出力ファイルの絶対パスに置換します。 %InputFile% ：コンパイル対象ファイルの絶対パスに置換します。 %CompiledFile% ：コンパイル時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。	
	デフォルト	コンパイル前に実行するコマンド [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。
コンパイル後に実行するコマンド	コンパイル処理後に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder% ：出力フォルダの絶対パスに置換します。 %OutputFile% ：出力ファイルの絶対パスに置換します。 %InputFile% ：コンパイル対象ファイルの絶対パスに置換します。 %CompiledFile% ：コンパイル時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。	
	デフォルト	コンパイル後に実行するコマンド [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。

その他の追加オプション	その他に追加するコンパイラのオプションを入力します。 なお，ここで設定したオプションは，コンパイラのオプション群の最後に付加されます。	
	デフォルト	*全体オプションの設定値*
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	259 文字までの文字列

［個別アセンブル・オプション］タブ

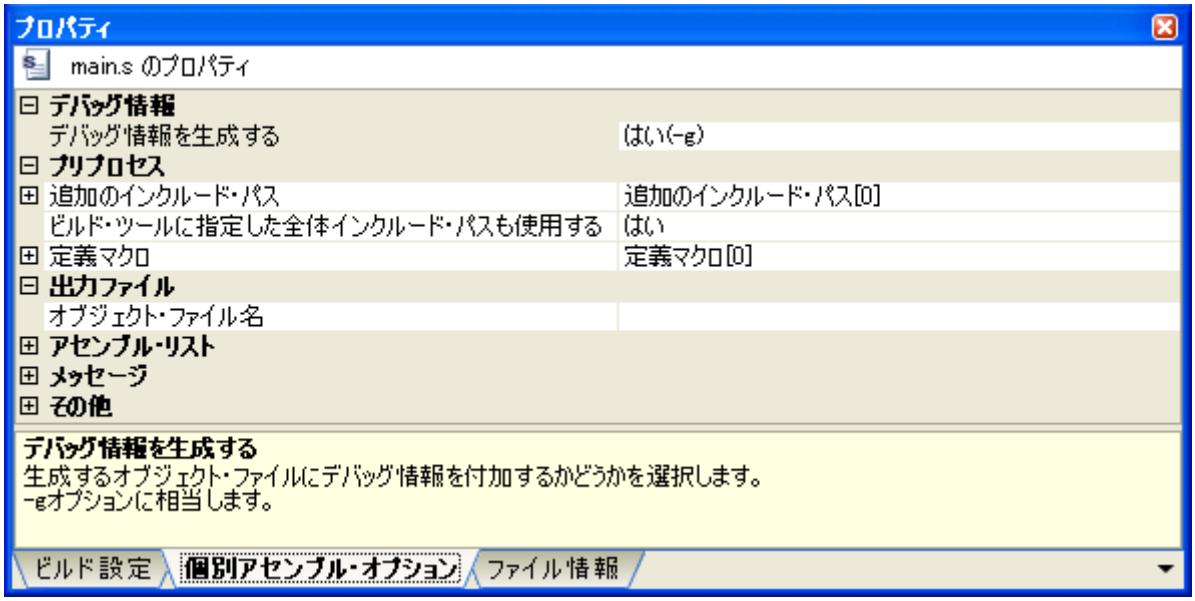
本タブでは，１つのアセンブラ・ソース・ファイルに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

なお，本タブは，［アセンブル・オプション］タブの設定内容を継承します。［アセンブル・オプション］タブと異なる値を設定した場合は，プロパティが太字表示となります。

備考 1. 本タブは，［ビルド設定］タブの［ビルド］カテゴリの［個別アセンブル・オプションを設定する］プロパティで［はい］を選択した場合に表示されます。

2. 本タブは，C ソース・ファイルを選択し，［個別コンパイル・オプション］タブの［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択した場合にも表示されます。

図 A—22　プロパティ パネル：［個別アセンブル・オプション］タブ



［各カテゴリの説明］

(1)［デバッグ情報］

デバッグ情報に関する詳細情報の表示，および設定の変更を行います。

デバッグ情報を生成する	生成するオブジェクト・ファイルにデバッグ情報を付加し，ソース・レベル・デバッグを可能にするかどうかを選択します。 アセンブラのオプション -g に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-g)	生成するオブジェクト・ファイルにデバッグ情報を付加します。
		いいえ	生成するオブジェクト・ファイルにデバッグ情報を付加しません。

(2)［プリプロセス］

プリプロセスに関する詳細情報の表示，および設定の変更を行います。

追加のインクルード・パス	アセンブル時の追加のインクルード・パスを指定します。 埋め込みマクロとして次のマクロ名があります。 %BuildModeName%　：ビルド・モード名に置換します。 %ProjectName%　：プロジェクト名に置換します。 %MicomToolPath%　：本製品のインストール・フォルダの絶対パスに置換します。 このオプションを省略した場合，アセンブラの標準フォルダのみ検索します。なお，パスはプロジェクト・フォルダを基点とします。 アセンブラのオプション -I に相当します。 指定したインクルード・パスはサブプロパティとして表示されます。	
	デフォルト	追加のインクルード・パス [*定義数*]
	変更方法	[...] ボタンをクリックし，パス編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	259 文字までの文字列 64 個まで指定可能です。ただし，連携するツールが使用するパスの数も含みます。

ビルド・ツールに指定した全体インクルード・パスも使用する	使用するビルド・ツールの［アセンブル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで指定したインクルード・パスも使用してアセンブルするかどうかを選択します。 アセンブラのオプション -I に相当します。 以下の順番で，-i オプションにパスを追加します。 - ［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで指定したパス - ［アセンブル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで指定したパス - ［アセンブル・オプション］タブの［プリプロセス］カテゴリの［システム・インクルード・パス］プロパティで指定したパス	
	デフォルト	はい
	変更方法	ドロップダウン・リストによる選択
定義マクロ	定義したいマクロ名を指定します。 「*マクロ名* = *定義値*」の形式で 1 行に 1 つずつ指定します。「= *定義値*」の部分は省略可能で，省略した場合，定義値を 1 とします。 アセンブラのオプション -D に相当します。 指定したマクロはサブプロパティとして表示されます。	
	デフォルト	定義マクロ [*定義数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

(3) ［出力ファイル］

出力ファイルに関する詳細情報の表示，および設定の変更を行います。

オブジェクト・ファイル名	アセンブル後に生成されるオブジェクト・ファイルのファイル名を指定します。 ".o" 以外の拡張子を指定することはできません。拡張子を省略した場合は，".o" が自動的に付加されます。 空欄の場合は，アセンブラ・ソース・ファイルの拡張子 .s を .o で置き換えたファイル名で保存します。 アセンブラのオプション -o に相当します。 なお，本プロパティは，C ソース・ファイルを選択し，［個別コンパイル・オプション］タブの［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択することにより［個別アセンブル・オプション］タブを表示した場合は，表示されません。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	259 文字までの文字列

(4) ［アセンブル・リスト］

アセンブル・リストに関する詳細情報の表示，および設定の変更を行います。

アセンブル・リスト・ファイルを出力する	アセンブル・リスト・ファイルを出力するかどうかを選択します。 アセンブラのオプション -a -l に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-a -l)	アセンブル・リスト・ファイルを出力します。
		いいえ	アセンブル・リスト・ファイルを出力しません。
アセンブル・リスト・ファイル出力フォルダ	アセンブル・リスト・ファイルの出力先フォルダを指定します。 アセンブル・リスト・ファイルは，アセンブラ・ソース・ファイルの拡張子 .s を .v で置き換えたファイル名で保存します。 アセンブラのオプション -l に相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。 なお，本プロパティは，［アセンブル・リスト・ファイルを出力する］プロパティで［はい (-a -l)］を選択した場合のみ表示されます。		
	デフォルト	*全体オプションの設定値*	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集	
	指定可能値	247 文字までの文字列	

(5)［メッセージ］

メッセージに関する詳細情報の表示，および設定の変更を行います。

実行状況を表示する	ビルド時にアセンブラの実行状況を出力 パネルに表示するかどうかを選択します。 アセンブラのオプション -v に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-v)	ビルド時にアセンブラの実行状況を表示します。
		いいえ	ビルド時にアセンブラの実行状況を表示しません。
r0 のデスティネーション・レジスタ使用を警告する	r0 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示するかどうかを選択します。 アセンブラのオプション -wr0+，-wr0- に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-wr0+)	r0 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示します。
		いいえ (-wr0-)	r0 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示しません。
		いいえ	r0 レジスタのデスティネーション・レジスタ指定にかかわらず，警告を表示します。

r1 レジスタ使用を警告する	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定しているとき，警告を表示するかどうかを選択します。 アセンブラのオプション -wr1+，-wr1- に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-wr1+)	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定しているとき，警告を表示します。
		いいえ (-wr1-)	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定しているとき，警告を表示しません。
		いいえ	r1 レジスタのソース・レジスタ，またはデスティネーション・レジスタ指定にかかわらず，警告を表示します。
警告を表示する	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定したとき，r0 レジスタをデスティネーション・レジスタとして指定したとき，およびマスク・レジスタ機能使用時に r20 レジスタ，または r21 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示するかどうかを選択します。 アセンブラのオプション -w に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定したとき，r0 レジスタをデスティネーション・レジスタとして指定したとき，およびマスク・レジスタ機能使用時に r20 レジスタ，または r21 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示します。
		いいえ (-w)	r1 レジスタをソース・レジスタ，またはデスティネーション・レジスタとして指定したとき，r0 レジスタをデスティネーション・レジスタとして指定したとき，およびマスク・レジスタ機能使用時に r20 レジスタ，または r21 レジスタをデスティネーション・レジスタとして指定しているとき，警告を表示しません。

（6）［その他］

アセンブルに関するその他の詳細情報の表示，および設定の変更を行います。

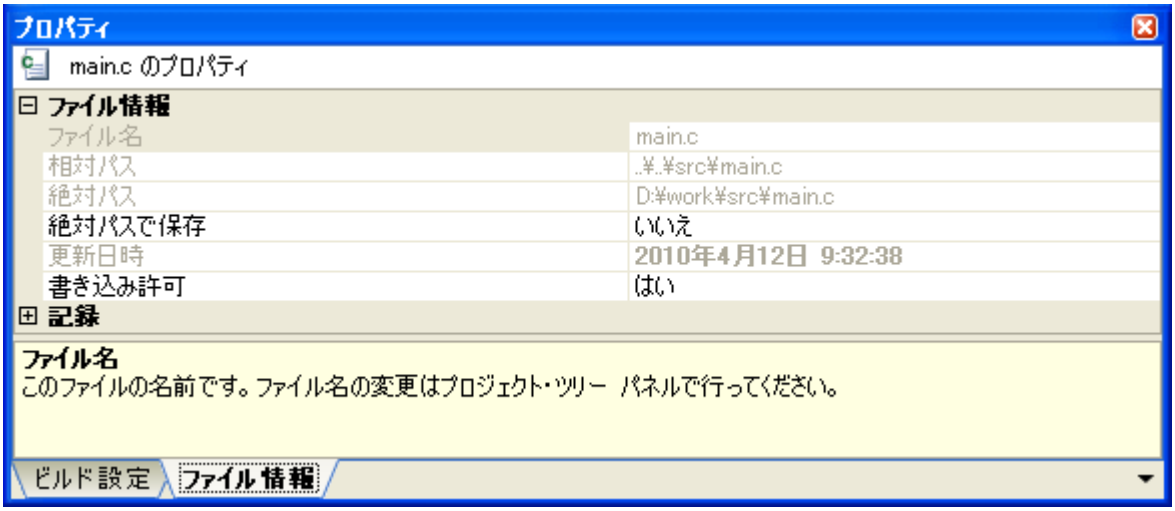
最適化を行う	命令を並べ替えて，レジスタ／フラグのハザードを回避する最適化を行うかどうかを選択します。 アセンブラのオプション -O に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-O)	レジスタ／フラグのハザードを回避する最適化を行います。
		いいえ	レジスタ／フラグのハザードを回避する最適化を行いません。
32 ビット分岐命令を使用する	命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定するかどうかを選択します。 アセンブラのオプション -Xfar_jump に相当します。 なお，本プロパティは，デバイスの品種に V850E2 コアのデバイスを指定している場合のみ表示されます。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-Xfar_jump)	命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定します。
		いいえ	命令に 22/32 を記述しない分岐命令（jarl，jr）は通常分岐命令となります。
アセンブル前に実行するコマンド	アセンブル処理前に実行するコマンドを指定します。 バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。 埋め込みマクロとして次のマクロ名があります。 %ProjectFolder%　：プロジェクト・フォルダの絶対パスに置換します。 %OutputFolder%　：出力フォルダの絶対パスに置換します。 %OutputFile%　：出力ファイルの絶対パスに置換します。 %InputFile%　：アセンブル対象ファイルの絶対パスに置換します。 %AssembledFile%　：アセンブル時の出力ファイルの絶対パスに置換します。 指定したコマンドはサブプロパティとして表示されます。 なお，本プロパティは，C ソース・ファイルを選択し，［個別コンパイル・オプション］タブの［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択することにより［個別アセンブル・オプション］タブを表示した場合は，表示されません。		
	デフォルト	アセンブル前に実行するコマンド [*定義数*]	
	変更方法	［...］ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能	
	指定可能値	1023 文字までの文字列 64 個まで指定可能です。	

<table>
<tr><td rowspan="4">アセンブル後に実行するコマンド</td><td colspan="2">アセンブル処理後に実行するコマンドを指定します。
バッチファイルを指定する場合は，call 命令を使用してください（例：call a.bat）。
埋め込みマクロとして次のマクロ名があります。
%ProjectFolder% ：プロジェクト・フォルダの絶対パスに置換します。
%OutputFolder% ：出力フォルダの絶対パスに置換します。
%OutputFile% ：出力ファイルの絶対パスに置換します。
%InputFile% ：アセンブル対象ファイルの絶対パスに置換します。
%AssembledFile% ：アセンブル時の出力ファイルの絶対パスに置換します。
指定したコマンドはサブプロパティとして表示されます。
なお，本プロパティは，C ソース・ファイルを選択し，［個別コンパイル・オプション］タブの［出力ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティで［はい (-Fs)］を選択することにより［個別アセンブル・オプション］タブを表示した場合は，表示されません。</td></tr>
<tr><td>デフォルト</td><td>アセンブル後に実行するコマンド [定義数]</td></tr>
<tr><td>変更方法</td><td>［...］ボタンをクリックし，テキスト編集 ダイアログによる編集
サブプロパティはテキスト・ボックスによる直接入力も可能</td></tr>
<tr><td>指定可能値</td><td>1023 文字までの文字列
64 個まで指定可能です。</td></tr>
<tr><td rowspan="4">その他の追加オプション</td><td colspan="2">その他に追加するアセンブラのオプションを入力します。
なお，ここで設定したオプションは，アセンブラのオプション群の最後に付加されます。</td></tr>
<tr><td>デフォルト</td><td>全体オプションの設定値</td></tr>
<tr><td>変更方法</td><td>テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集</td></tr>
<tr><td>指定可能値</td><td>259 文字までの文字列</td></tr>
</table>

［ファイル情報］タブ

本タブでは，各ファイルに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

図 A―23　プロパティ パネル：［ファイル情報］タブ



［各カテゴリの説明］

（1）［ファイル情報］

ファイルに関する詳細情報の表示，および設定の変更を行います。

ファイル名	ファイル名を表示します。 ファイル名の変更は，プロジェクト・ツリー パネルで行ってください。	
	デフォルト	*ファイル名*
	変更方法	変更不可
相対パス	ファイルのプロジェクト・フォルダからの相対パス名を表示します。	
	デフォルト	*ファイルのプロジェクト・フォルダからの相対パス名*
	変更方法	変更不可
絶対パス	ファイルの絶対パス名を表示します。	
	デフォルト	*ファイルの絶対パス名*
	変更方法	変更不可

絶対パスで保存	ファイルの場所を絶対パスで保存するかどうかを選択します。		
	デフォルト	いいえ	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ファイルの場所を絶対パスで保存します。
		いいえ	ファイルの場所を相対パスで保存します。
更新日時	ファイルが最後に変更された日時を表示します。		
	デフォルト	*ファイルの更新日時*	
	変更方法	変更不可	
書き込み許可	ファイルに書き込みを許可するかどうかを選択します。		
	デフォルト	はい（ファイルに書き込みが許可されている場合） いいえ（ファイルに書き込みが許可されていない場合）	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい	ファイルに書き込みを許可します。
		いいえ	ファイルに書き込みを許可しません。

(2) [記録]

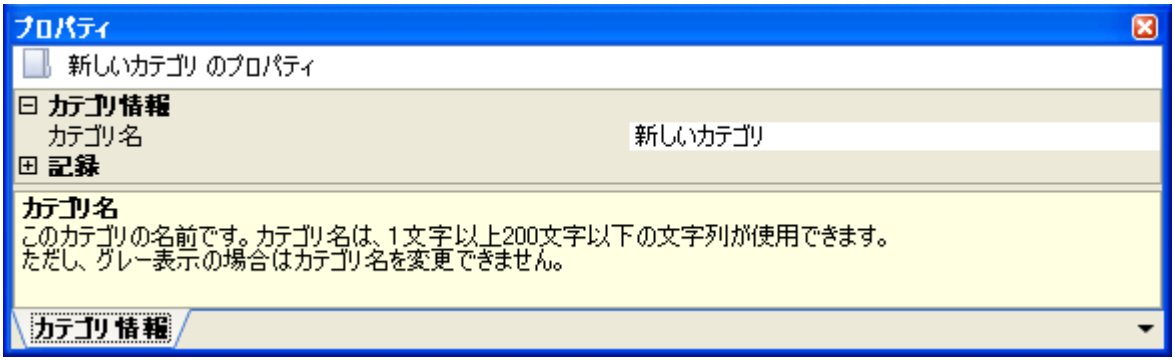
記録に関する詳細情報の表示，および設定の変更を行います。

メモ	ファイルにメモを追加します。 1 行に 1 項目ずつ指定します。 追加したメモはサブプロパティとして表示します。	
	デフォルト	メモ [*項目数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

［カテゴリ情報］タブ

本タブでは，カテゴリ・ノード（ユーザが追加したファイルのカテゴリ），ファイル・ノード，ビルド・ツール生成ファイル・ノード，スタートアップ・ノードに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

図 A―24　プロパティ パネル：［カテゴリ情報］タブ



[各カテゴリの説明]

（1）［カテゴリ情報］

カテゴリに関する詳細情報の表示，および設定の変更を行います。

カテゴリ名	ファイルを分類するためのカテゴリ名を指定します。 なお，本プロパティは，ファイル・ノード，ビルド・ツール生成ファイル・ノード，スタートアップ・ノードについてはグレー表示となり，変更することはできません。	
	デフォルト	*ファイルのカテゴリ名*
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	1 ～ 200 文字までの文字列

（2）［記録］

記録に関する詳細情報の表示，および設定の変更を行います。

なお，本カテゴリは，ファイル・ノード，ビルド・ツール生成ファイル・ノード，スタートアップ・ノードについては表示されません。

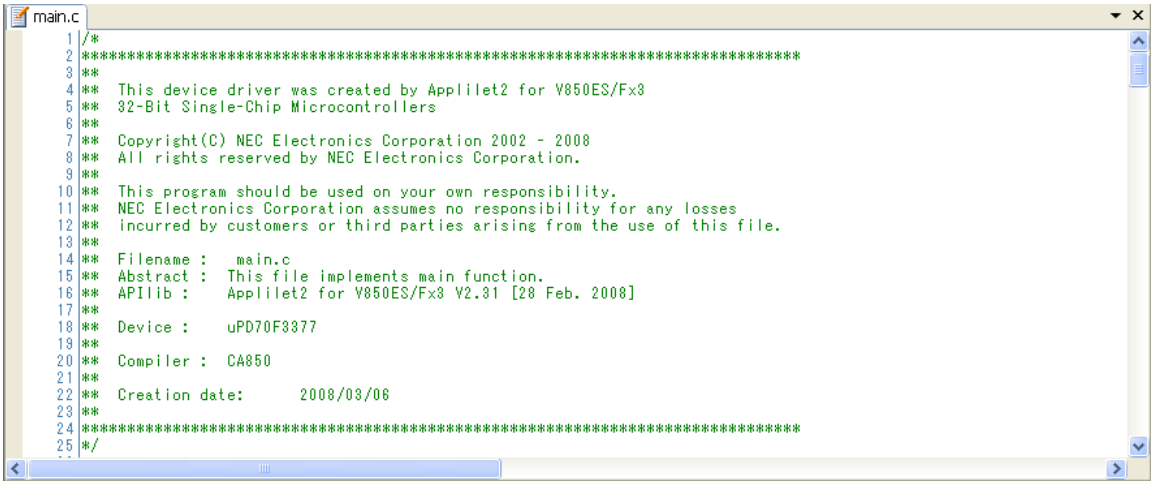
メモ	ファイルのカテゴリにメモを追加します。 1 行に 1 項目ずつ指定します。 追加したメモはサブプロパティとして表示します。	
	デフォルト	メモ [*項目数*]
	変更方法	[...] ボタンをクリックし、テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

エディタ パネル

テキスト・ファイル／ソース・ファイルの表示／編集を行います。

本パネルについての詳細は，「CubeSuite+ V850 コーディング編」を参照してください。

図 A—25　エディタ パネル

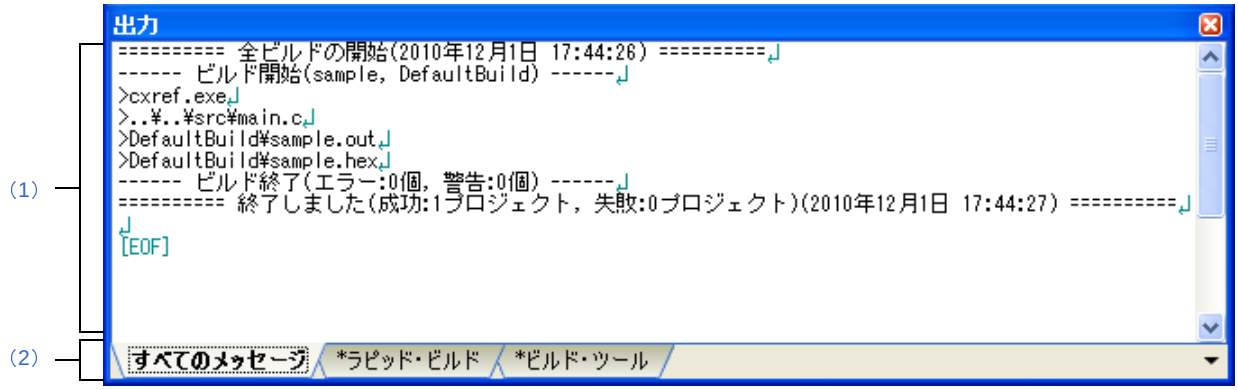


出力 パネル

ビルド・ツールから出力されるメッセージの表示を行います。

メッセージは，出力元のツールごとに分類されたタブ上でそれぞれ個別に表示されます。

図 A―26　出力 パネル



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［［ファイル］メニュー（出力 パネル専用部分）］
- ［［編集］メニュー（出力 パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

（1）メッセージ・エリア

各ツールから出力されたメッセージ，および検索結果を表示します。

ビルド結果／検索結果（一括検索）の表示では，ビルド／検索を行うごとに，以前のメッセージをクリアしたのち新しいメッセージを表示します（［すべてのメッセージ］タブを除く）。

備考　メッセージの最大表示行数は 500000 行です。500001 行以上のメッセージが出力された場合は，古い行から削除されます。

なお，メッセージの表示色は，出力メッセージの種別により，次のように異なります（表示の際の文字色／背景色は，オプション ダイアログにおける［全般 - フォントと色］カテゴリ項目の設定に依存します）。

メッセージ種別	表示例（デフォルト）			説明
通常メッセージ	AaBbCc	文字色	黒	何らかの情報を通知する際に表示されます。
		背景色	白	
警告メッセージ	AaBbCc	文字色	青	操作に対して，何らかの警告を通知する際に表示されます。
		背景色	標準色	
エラー・メッセージ	AaBbCc	文字色	赤	致命的なエラー，または操作ミスにより実行が不可能な場合に表示されます。
		背景色	薄グレー	

本エリアは，次の機能を備えています。

(a) タグ・ジャンプ

出力されたメッセージをダブルクリック，またはメッセージにキャレットを合わせて［Enter］キーを押下することにより，エディタ パネルをオープンして該当ファイルの該当行番号を表示します。

これにより，ビルド時に出力されたエラー・メッセージなどから，ソース・ファイルの該当するエラー行へジャンプすることができます。

(b) ヘルプの表示

警告メッセージ，またはエラー・メッセージを表示している行にキャレットがある状態で，コンテキスト・メニューの［メッセージに関するヘルプ］を選択するか，または［F1］キーを押下することにより，その行のメッセージに関するヘルプを表示します。

(c) ログの保存

［ファイル］メニュー→［名前を付けて出力 - *タブ名*を保存 ...］を選択することにより，名前を付けて保存 ダイアログをオープンし，現在選択しているタブ上に表示されている内容をテキスト・ファイル（*.txt）に保存することができます（非選択状態のタブ上のメッセージは保存の対象となりません）。

(2) タブ選択エリア

メッセージの出力元を示すタブを選択します。

表示されるタブは次のとおりです。

タブ名	説明
すべてのメッセージ	すべてのメッセージを出力順に一括して表示します（ラピッド・ビルド実行時を除く）。
ラピッド・ビルド	ラピッド・ビルドの実行により，ビルド・ツールから出力されたメッセージを表示します。
ビルド・ツール	ビルド，リビルド，バッチ・ビルドの実行により，ビルド・ツールから出力されたメッセージを表示します。

注意 新たなメッセージが非選択状態のタブ上に出力されても，自動的なタブの表示切り替えは行いません。この場合，タブ名の先頭に  マークが付加し，新たなメッセージが出力されていることを示します。

[[ファイル］メニュー（出力 パネル専用部分)]

出力 パネル専用の［ファイル］メニューは次のとおりです（その他の項目は共通です)。

出力 - *タブ名*を保存	現在選択しているタブ上に表示されている内容を，前回保存したテキスト・ファイル（*.txt）に保存します（「(c) ログの保存」参照)。 なお，起動後に初めて本項目を選択した場合は，［名前を付けて*タブ名*を保存 ...］の選択と同等の動作となります。
名前を付けて出力 - *タブ名*を保存 ...	現在選択しているタブ上に表示されている内容を，指定したファイル（*.txt）に保存するために，名前を付けて保存 ダイアログをオープンします（「(c) ログの保存」参照)。

[[編集］メニュー（出力 パネル専用部分)]

出力 パネル専用の［編集］メニューは次のとおりです（その他の項目はすべて無効となります)。

コピー	選択している文字列をクリップ・ボードにコピーします。
すべて選択	本パネルに表示しているすべてのメッセージを選択状態にします。
検索 ...	検索・置換 ダイアログを［クイック検索］タブが選択状態でオープンします。
置換 ...	検索・置換 ダイアログを［一括置換］タブが選択状態でオープンします。

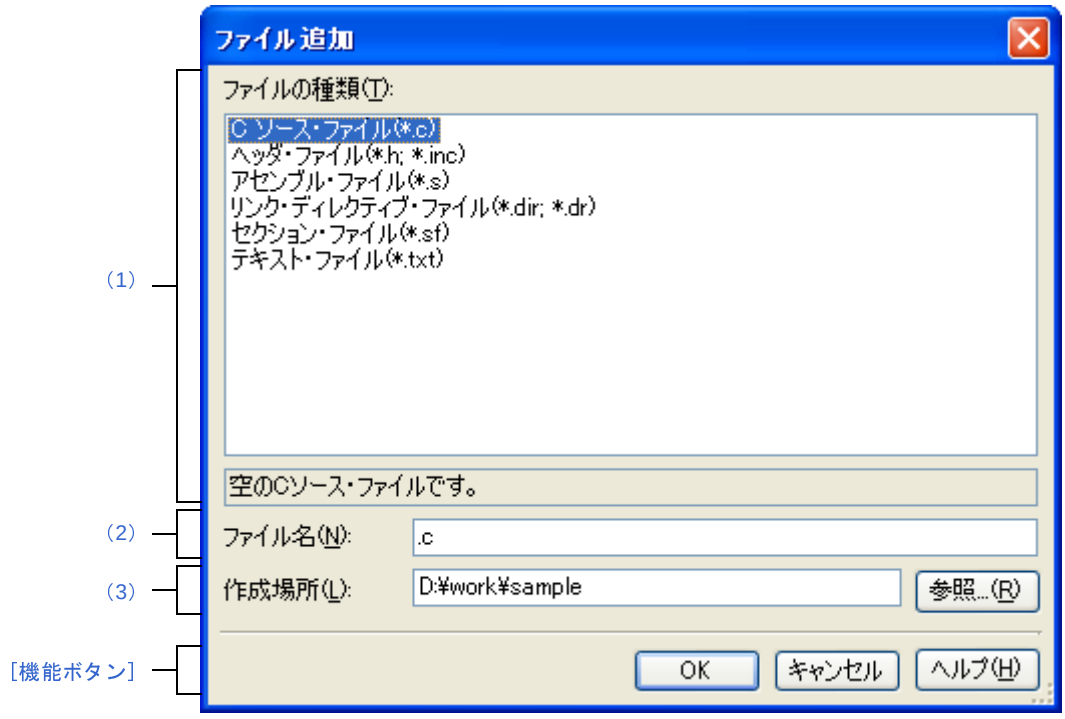
[コンテキスト・メニュー]

コピー	選択している文字列をクリップ・ボードにコピーします。
すべて選択	本パネルに表示しているすべてのメッセージを選択状態にします。
クリア	本パネルに表示しているすべてのメッセージを消去します。
タグ・ジャンプ	キャレット行のメッセージに対応するエディタ（ファイル，行，桁）へジャンプします。
メッセージに関するヘルプ	現在のキャレット位置のメッセージに関するヘルプを表示します。 ただし，警告メッセージ／エラー・メッセージのみが対象となります。

ファイル追加 ダイアログ

新規にファイルを作成し，プロジェクトへの追加を行います。

図 A—27　ファイル追加 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ファイル］メニュー→［追加］→［新しいファイルを追加 ...］を選択
- プロジェクト・ツリー パネルにおいて，プロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノード，カテゴリ・ノードのいずれかを選択したのち，コンテキスト・メニュー→［追加］→［新しいファイルを追加 ...］を選択

［各エリアの説明］

(1)［ファイルの種類］エリア

作成するファイルの種類を選択します。

ファイルの種類を選択すると，下部のボックスにその説明を表示します。

表示されるファイルの種類を以下に示します。

- C ソース・ファイル（*.c）
- ヘッダ・ファイル（*.h; *.inc）
- アセンブル・ファイル（*.s）
- リンク・ディレクティブ・ファイル（*.dir; .*dr）
- セクション・ファイル（*.sf）
- テキスト・ファイル（*.txt）

(2)［ファイル名］エリア

作成するファイルの名前を直接入力します。

デフォルトでは，“.txt”を表示します。

備考　拡張子を指定しなかった場合は，［ファイルの種類］エリアで選択した拡張子が付加されます。また，［ファイルの種類］エリアと異なる拡張子を指定した場合も，［ファイルの種類］エリアで選択した拡張子が付加されます（例えば，ファイル名に“aaa.txt”，ファイルの種類に“C ソース・ファイル (*.c)”を指定した場合，ファイル名は“aaa.txt.c”となります）。

(3)［作成場所］エリア

ファイルの作成場所のパスをテキスト・ボックスに直接入力，または［参照 ...］ボタンから選択します。

デフォルトでは，プロジェクト・フォルダのパスを表示します。

(a) ボタン

参照 ...	フォルダの参照 ダイアログをオープンします。 フォルダを選択すると，テキスト・ボックスにパスが追加されます。

備考 1.　テキスト・ボックスが空欄の場合は，プロジェクト・フォルダを指定したものとみなします。

2.　相対パスで指定した場合は，プロジェクト・フォルダからの相対パスとみなします。

備考　［ファイル名］エリア，［作成場所］エリアで指定可能な文字数は，パス名とファイル名をあわせて 259 文字までです。入力内容が正しくない場合，以下のメッセージが［ファイル名］エリアにツールチップ表示されます。

メッセージ	説明
パスを含むファイル名が長すぎます。259 文字以内にしてください。	パスを含むファイル名が 259 文字を越えています。

メッセージ	説明
指定したパスに存在しないフォルダが含まれています。	パスに存在しないフォルダが含まれています。
ファイル名，もしくは，パス名が不正です。文字 (¥ , /, :, *, ?, ", <, >, |) は使用できません。	不正なパスを含むファイル名が指定されました。ファイル名，およびフォルダ名に文字 (¥ , /, :, *, ?, ", <, >, |) は使用できません。

[機能ボタン]

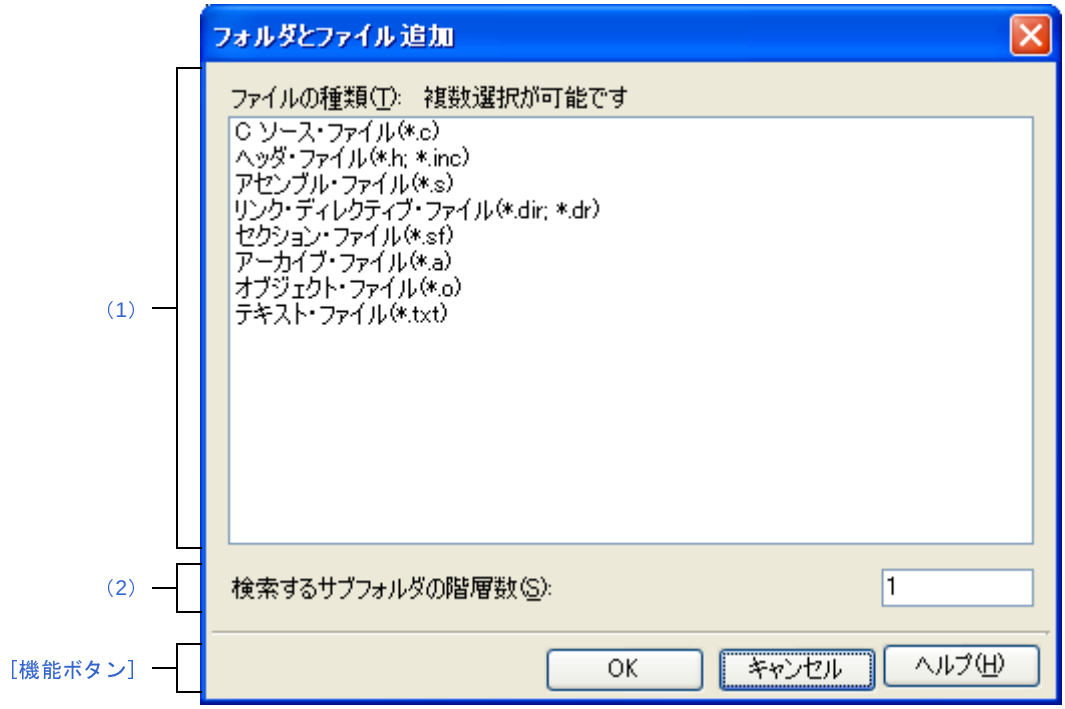
ボタン	機能
OK	入力したファイル名でファイルを作成して，プロジェクトに追加し，エディタ パネルでオープンします。そののち，本ダイアログをクローズします。
キャンセル	ファイルの生成を行わずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

フォルダとファイル追加 ダイアログ

既存のファイルとフォルダ構成のプロジェクトへの追加を行います。

フォルダはカテゴリとして追加します。

図 A―28　フォルダとファイル追加 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- エクスプローラなどからフォルダをドラッグし，プロジェクト・ツリー パネル上でドロップ

［各エリアの説明］

(1)［ファイルの種類］エリア

プロジェクトに追加するファイルの種類を選択します。

［Ctrl］キー＋左クリック，または［Shift］キー＋左クリックにより，複数選択することができます。

何も選択しない場合は，すべての種類を選択したものとみなします。

表示されるファイルの種類を以下に示します。

- C ソース・ファイル（*.c）
- ヘッダ・ファイル（*.h; *.inc）
- アセンブル・ファイル（*.s）
- リンク・ディレクティブ・ファイル（*.dir; .*dr）
- セクション・ファイル（*.sf）
- アーカイブ・ファイル（*.a）
- オブジェクト・ファイル（*.o）
- テキスト・ファイル（*.txt）

(2)［検索するサブフォルダの階層数］エリア

プロジェクトに追加するサブフォルダの階層数を直接入力します。

デフォルトでは，“1”を表示します。

備考　入力可能な値は 10 までの 10 進数です。入力内容が正しくない場合，以下のメッセージがツールチップ表示されます。

メッセージ	説明
0 未満もしくは 10 を越える値を指定できません。	サブフォルダの指定階層数が，10 を越えています。
10 進数で指定してください。	10 進数以外の数値や文字列が指定されました。

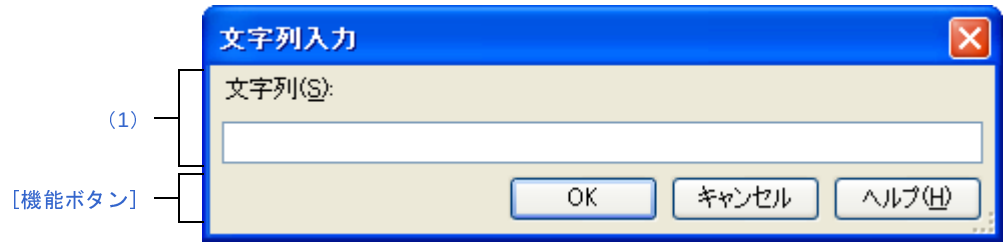
［機能ボタン］

ボタン	機能
OK	ドラッグ・アンド・ドロップしたフォルダとそのフォルダ下に存在するファイルをプロジェクトに追加します。そののち，本ダイアログをクローズします。
キャンセル	フォルダとファイルの追加を行わずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

文字列入力 ダイアログ

1 行分の文字列の入力，編集を行います。

図 A—29　文字列入力 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［共通オプション］タブの［その他］カテゴリの［ビルド・オプション一覧表示フォーマット］
  - ［コンパイル・オプション］タブの［メッセージ］カテゴリの［必ず表示させる警告メッセージ］，［表示させない警告メッセージ］，［その他］カテゴリの［その他の追加オプション］
  - ［アセンブル・オプション］タブの［その他］カテゴリの［その他の追加オプション］
  - ［リンク・オプション］タブの［その他］カテゴリの［エントリ・シンボル］，［その他の追加オプション］
  - ［ROM 化プロセス・オプション］タブの［その他］カテゴリの［エントリ・ラベル］，［その他の追加オプション］
  - ［ヘキサ・コンバート・オプション］タブの［その他］カテゴリの［その他の追加オプション］
  - ［アーカイブ・オプション］タブの［その他］カテゴリの［その他の追加オプション］
  - ［セクション・ファイル・ジェネレート・オプション］タブの［その他］カテゴリの［その他の追加オプション］
  - ［ダンプ・オプション］タブの［ダンプ・ツール］カテゴリの［ダンプ・ツールの追加オプション］
  - ［クロス・リファレンス・オプション］タブの［クロス・リファレンス・ツール］カテゴリの［クロス・リファレンス・ツールの追加オプション］
  - ［メモリ・レイアウト視覚化オプション］タブの［メモリ・レイアウト視覚化ツール］カテゴリの［メモリ・レイアウト視覚化ツールの追加オプション］
  - ［個別コンパイル・オプション］タブの［その他］カテゴリの［その他の追加オプション］
  - ［個別アセンブル・オプション］タブの［その他］カテゴリの［その他の追加オプション］
- リンク・ディレクティブ生成 ダイアログの［セグメント／セクション一覧］エリアでセグメント，またはセクションを選択したのち，［セグメント／セクションの詳細］エリアの［名前］において，［...］ボタンをクリック

- リンク・ディレクティブ生成 ダイアログの［セグメント／セクション一覧］エリアでセクションを選択したのち，［セグメント／セクションの詳細］エリアの［入力セクション名］において，［...］ボタンをクリック
- リンク・ディレクティブ生成 ダイアログの［シンボル一覧］エリアでシンボルを選択したのち，［シンボルの詳細］エリアの［名前］において，［...］ボタンをクリック
- リンク・ディレクティブ生成 ダイアログの［シンボル一覧］エリアでシンボルを選択したのち，［シンボルの詳細］エリアの［ベース・シンボル名］において，［...］ボタンをクリック
- オプション ダイアログの［全般 - 外部ツール］カテゴリにおいて，新規登録エリアの［起動時に引数を入力する］をチェックしたのち，［ツール］メニューより外部ツールの起動時に自動的にオープン

［各エリアの説明］

（1）［文字列］エリア

1 行分の文字列の入力を行います。

デフォルトでは，本ダイアログの呼び出し元の内容が反映されます。

なお，改行することはできません。

備考　入力可能な文字数は，32767 文字までです。入力内容が正しくない場合，以下のメッセージがツールチップ表示されます。

メッセージ	説明
*呼び出し元で指定されている最大文字数*文字を越える文字を指定できません。	入力された文字列の文字数が，呼び出し元で指定されている最大文字数を越えています。

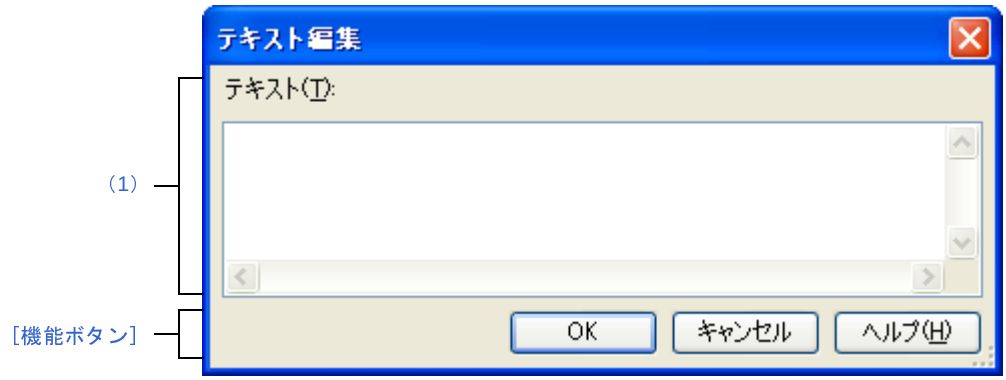
［機能ボタン］

ボタン	機能
OK	入力した文字列を本ダイアログの呼び出し元に反映し，本ダイアログをクローズします。
キャンセル	入力した文字列を本ダイアログの呼び出し元に反映せずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

テキスト編集 ダイアログ

複数行のテキストの入力，編集を行います。

図 A—30　テキスト編集 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［定義マクロ］，［よく使うオプション（アセンブラ）］カテゴリの［定義マクロ］，［よく使うオプション（リンカ）］カテゴリの［使用するライブラリ・ファイル］，［記録］カテゴリの［メモ］，［その他］カテゴリの［ビルド前に実行するコマンド］，［ビルド後に実行するコマンド］
  - ［コンパイル・オプション］タブの［プリプロセス］カテゴリの［定義マクロ］，［定義解除マクロ］，［その他］カテゴリの［コンパイル前に実行するコマンド］，［コンパイル後に実行するコマンド］
  - ［アセンブル・オプション］タブの［プリプロセス］カテゴリの［定義マクロ］，［その他］カテゴリの［アセンブル前に実行するコマンド］，［アセンブル後に実行するコマンド］
  - ［リンク・オプション］タブの［ライブラリ］カテゴリの［使用するライブラリ・ファイル］，［その他］カテゴリの［リンク前に実行するコマンド］，［リンク後に実行するコマンド］
  - ［ROM 化プロセス・オプション］タブの［セクション・リスト］カテゴリの［rompsec セクションに格納する順番］，［その他］カテゴリの［ROM 化前に実行するコマンド］，［ROM 化後に実行するコマンド］
  - ［ヘキサ・コンバート・オプション］タブの［ヘキサ・フォーマット］カテゴリの［ヘキサ・ファイルに変換するセクション］，［その他］カテゴリの［ヘキサ・コンバート前に実行するコマンド］，［ヘキサ・コンバート後に実行するコマンド］
  - ［アーカイブ・オプション］タブの［その他］カテゴリの［アーカイブ前に実行するコマンド］，［アーカイブ後に実行するコマンド］

- ［セクション・ファイル・ジェネレート・オプション］タブの［変数の配置］カテゴリの［最適化の対象としないセクション］，［最適化の対象としない変数］
- ［個別コンパイル・オプション］タブの［プリプロセス］カテゴリの［定義マクロ］，［定義解除マクロ］，［その他］カテゴリの［コンパイル前に実行するコマンド］，［コンパイル後に実行するコマンド］
- ［個別アセンブル・オプション］タブの［プリプロセス］カテゴリの［定義マクロ］，［その他］カテゴリの［アセンブル前に実行するコマンド］，［アセンブル後に実行するコマンド］
- ［ファイル情報］タブの［記録］カテゴリの［メモ］
- ［カテゴリ情報］タブの［記録］カテゴリの［メモ］

［各エリアの説明］

(1)［テキスト］エリア

複数行のテキストの編集を行います。

デフォルトでは，本ダイアログの呼び出し元の内容が反映されます。

備考 入力可能な行数は 65535 行まで，文字数は 65535 文字までです。入力内容が正しくない場合，以下のメッセージがツールチップ表示されます。

メッセージ	説明
*呼び出し元で指定されている最大文字数*文字を越える文字を指定できません。制限を越えた行頭のかっこ内に今の文字数を表示しました。	入力された文字列の文字数が，呼び出し元で指定されている最大文字数を越えています。

［機能ボタン］

ボタン	機能
OK	入力したテキストを本ダイアログをオープンしたテキスト・ボックスに反映し，本ダイアログをクローズします。
キャンセル	入力したテキストを本ダイアログをオープンしたテキスト・ボックスに反映せずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

パス編集 ダイアログ

パス，またはパスを含むファイル名の編集，追加を行います。

図 A—31　パス編集 ダイアログ（パスを編集する場合）

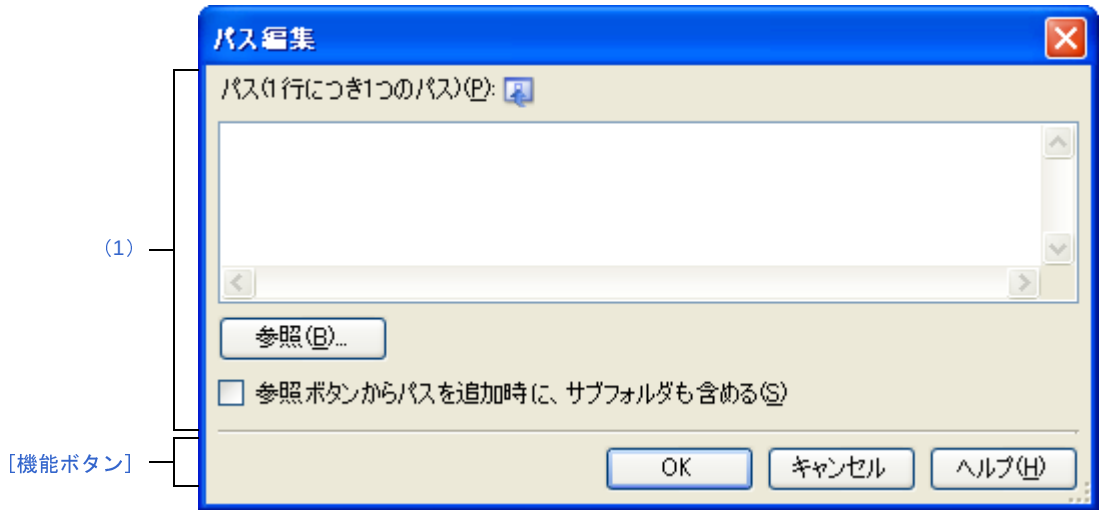
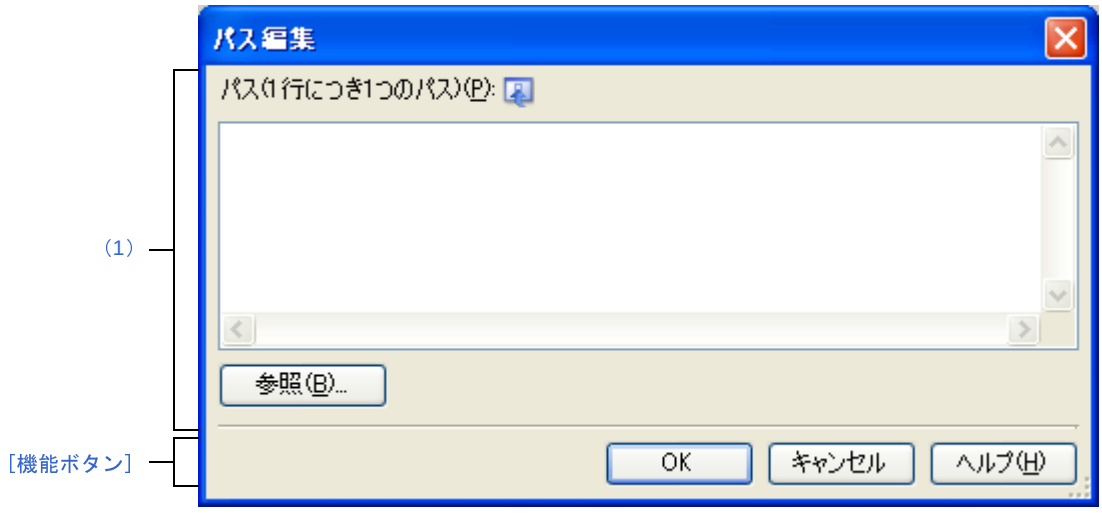


図 A—32　パス編集 ダイアログ（パスを含むファイル名を編集する場合）



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

[オープン方法]

- プロパティ パネルにおいて，以下のプロパティを選択したのち，[...] ボタンをクリック
  - ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［追加のインクルード・パス］，［よく使うオプション（アセンブラ）］カテゴリの［追加のインクルード・パス］，［よく使うオプション（リンカ）］カテゴリの［追加のライブラリ・パス］
  - ［コンパイル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］，［入力ファイル］カテゴリの［Far jump ファイル名］
  - ［アセンブル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］
  - ［リンク・オプション］タブの［ライブラリ］カテゴリの［追加のライブラリ・パス］
  - ［個別コンパイル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］
  - ［個別アセンブル・オプション］タブの［プリプロセス］カテゴリの［追加のインクルード・パス］

[各エリアの説明]

（1）パス編集エリア

パス，またはパスを含むファイル名の編集，追加を行います。

（a）［パス (1 行につき 1 つのパス)］

直接入力により，パス，またはパスを含むファイル名の編集，追加を行います。

パス，またはパスを含むファイル名は複数行指定可能です。1 行につき 1 つのパス，またはパスを含むファイル名を指定してください。

デフォルトで，本ダイアログをオープンしたテキスト・ボックスの内容が反映されます。

パスの追加は，以下の方法でも行うことができます。

- ［参照 ...］ボタンをクリックし，フォルダの参照 ダイアログによるフォルダの選択
- エクスプローラなどからフォルダをドラッグ・アンド・ドロップ

パスを含むファイル名の追加は，以下の方法でも行うことができます。

- ［参照 ...］ボタンをクリックし，Far Jump ファイルを指定 ダイアログによるファイルの選択
- エクスプローラなどからファイルをドラッグ・アンド・ドロップ

注意　絶対パスで非常に長いパスを相対パスで指定すると，［OK］ボタンのクリック時にエラーになる場合があります。その場合は，絶対パスで指定してください。

備考　入力可能な行数は 10000 行まで，文字数は Windows のパスの最大文字数までです。入力内容が正しくない場合，以下のメッセージがツールチップ表示されます。

メッセージ	説明
パスを指定してください。	空欄になっています。

メッセージ	説明
パスが長すぎます。*呼び出し元で指定されている最大文字数*文字以下のパスを指定してください。	パスを含むファイル名が呼び出し元で指定されている最大文字数を越えています。
指定したパスに存在しないフォルダが含まれています。	パスに存在しないフォルダが含まれています。
ファイル名，もしくは，パス名が不正です。文字 (¥ , /, :, *, ?, ", <, >, |) は使用できません。	不正なパスを含むファイル名が指定されました。ファイル名，およびフォルダ名に文字 (¥ , /, :, *, ?, ", <, >, |) は使用できません。
*呼び出し元で指定されている最大パス数，またはファイル数*行を越える行を指定できません。	入力されたパス，またはファイルの総数が呼び出し元で指定されている最大パス数，またはファイル数を越えています。

（b）ボタン

参照 ...	- パスを追加する場合 フォルダの参照 ダイアログをオープンします。 フォルダを選択すると，［パス (1 行につき 1 つのパス)］にパスが追加されます。 - パスを含むファイル名を追加する場合 Far Jump ファイルを指定 ダイアログをオープンします。 ファイルを選択すると，［パス (1 行につき 1 つのパス)］にファイル名が追加されます。

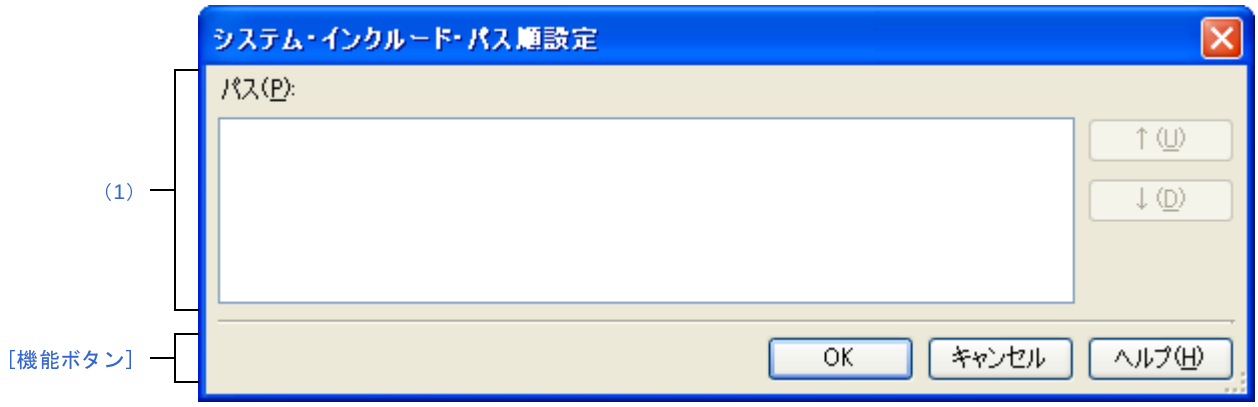
［機能ボタン］

ボタン	機能
OK	入力したパスを本ダイアログをオープンしたテキスト・ボックスに反映し，本ダイアログをクローズします。
キャンセル	入力したパスを本ダイアログをオープンしたテキスト・ボックスに反映せずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

システム・インクルード・パス順設定 ダイアログ

コンパイラに対して指定するシステム・インクルード・パスの参照，および指定順の設定を行います。

図 A―33　システム・インクルード・パス順設定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［システム・インクルード・パス］，［よく使うオプション（アセンブラ）］カテゴリの［システム・インクルード・パス］
  - ［コンパイル・オプション］タブの［プリプロセス］カテゴリの［システム・インクルード・パス］
  - ［アセンブル・オプション］タブの［プリプロセス］カテゴリの［システム・インクルード・パス］

［各エリアの説明］

(1) パス一覧表示エリア

コンパイラに対して指定するシステム・インクルード・パスの一覧を表示します。

(a) ［パス］

システム・インクルード・パス名の一覧を，コンパイラへの指定順に表示します。

デフォルトでは，プロジェクトに登録されている順番となります。

パスの表示順を変更することにより，コンパイラへの指定順を設定することができます。

表示順の変更は，［↑］，および［↓］ボタン，またはパス名のドラッグ・アンド・ドロップにより行います。

備考 1. パス名にマウス・カーソルをあわせると，そのパスを絶対パスでポップアップ表示します。

2. 新規に追加されたシステム・インクルード・パスは，一覧の最後のパスの次に追加されます。

3. パス名をドラッグ・アンド・ドロップする際，連続して並んでいるパス名のみ複数選択することができます。

（b）ボタン

↑	選択しているパスを上へ移動します。
↓	選択しているパスを下へ移動します。

備考　上記のボタンは，パスを選択していない場合は無効となります。

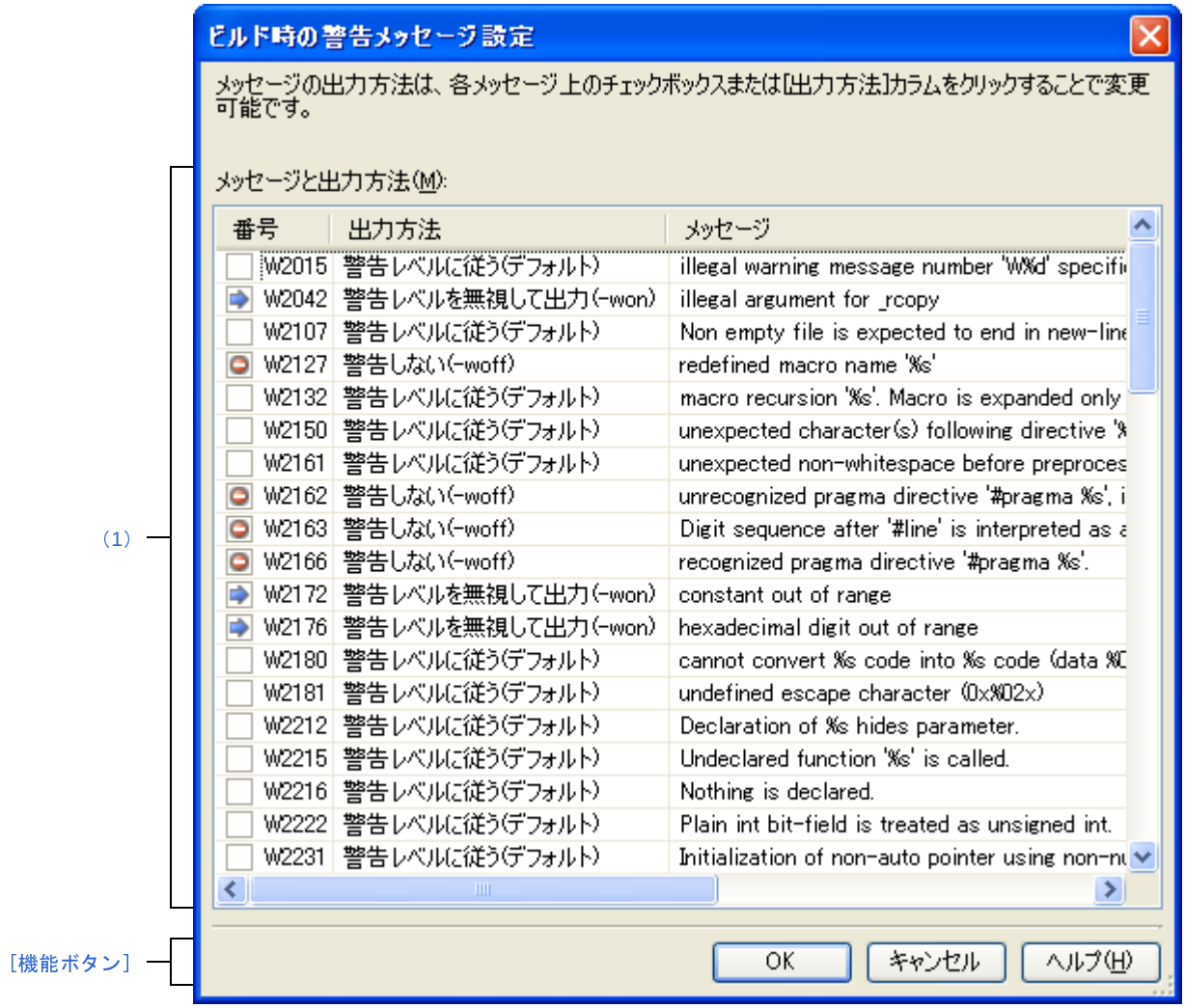
［機能ボタン］

ボタン	機能
OK	コンパイラへのパスの指定順をパス一覧表示エリアの表示順に設定し，本ダイアログをクローズします。
キャンセル	パスの指定順の設定をキャンセルし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

ビルド時の警告メッセージ設定 ダイアログ

ビルド・ツールが出力する警告メッセージの設定を行います。

図 A—34　ビルド時の警告メッセージ設定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［コンパイル・オプション］タブの［メッセージ］カテゴリの［必ず表示させる警告メッセージ］，［表示させない警告メッセージ］

［各エリアの説明］

（1）［メッセージと出力方法］エリア

ビルド・ツールが出力する警告メッセージの一覧を表示します。

（a）［番号］

警告メッセージの番号を表示します。

チェックボックスには，［出力方法］に対応するアイコンを表示します。

アイコン	［出力方法］
	警告レベルに従う (デフォルト)
	警告レベルを無視して出力 (-won)
	警告しない (-woff)

備考　チェックボックスをクリックすることにより，［出力方法］の項目を変更することができます。

（b）［出力方法］

ドロップダウン・リストを選択することにより，警告メッセージの出力方法の設定を行います。

項目	説明
警告レベルに従う (デフォルト)	［警告表示レベル］プロパティの設定に従って，警告メッセージを表示します。
警告レベルを無視して出力 (-won)	［警告表示レベル］プロパティの設定にかかわらず，警告メッセージを表示します。
警告しない (-woff)	［警告表示レベル］プロパティの設定にかかわらず，警告メッセージを表示しません。

デフォルトでは，［必ず表示させる警告メッセージ］プロパティ，および［表示させない警告メッセージ］プロパティの内容が反映されます。

ただし，［必ず表示させる警告メッセージ］プロパティと［表示させない警告メッセージ］プロパティで同一の番号を指定している場合は，［必ず表示させる警告メッセージ］プロパティを優先します。

（c）［メッセージ］

警告メッセージを表示します。

備考 1.　各ヘッダ（［番号］／［出力方法］／［メッセージ］）をクリックすることにより，番号順，出力方法順，メッセージ順で警告メッセージの一覧を昇順／降順ソートすることができます。
デフォルトでは，番号順で昇順ソートされます。

2.　警告メッセージは，［Ctrl］キー，または［Shift］キーの押下により，複数選択することができます。

複数選択している場合，以下の方法により，選択しているすべての警告メッセージの出力方法を変更することができます。

- チェックボックスをクリック
- ［Ctrl］キー，または［Shift］キーを押下したまま，［出力方法］のドロップダウン・リストを選択

［機能ボタン］

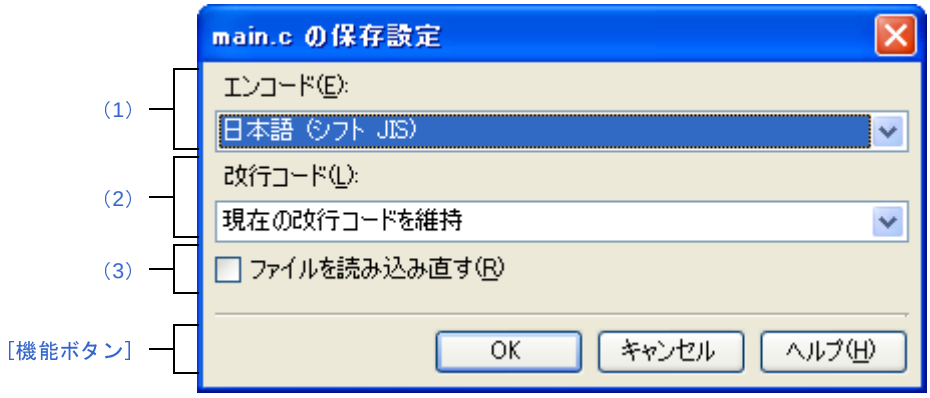
ボタン	機能
OK	設定内容を［必ず表示させる警告メッセージ］プロパティ，および［表示させない警告メッセージ］プロパティに反映し，本ダイアログをクローズします。
キャンセル	設定内容を［必ず表示させる警告メッセージ］プロパティ，および［表示させない警告メッセージ］プロパティに反映せずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

ファイルの保存設定 ダイアログ

エディタ パネルで編集中のファイルのエンコードと改行コードの設定を行います。

備考　タイトルバーには，設定対象ファイルの名前が表示されます。

図 A—35　ファイルの保存設定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- エディタ パネルにフォーカスがある状態で，［ファイル］メニュー→［*ファイル名*を保存 ...］を選択

［各エリアの説明］

(1)［エンコード］

設定するエンコードをドロップダウン・リストにより選択します。

ドロップダウン・リストの項目は，以下の順番で表示されます。

ただし，同じエンコード名，および現在の OS が対応していないエンコード名は表示されません。

- *現在のファイルのエンコード名*（デフォルト）
- *現在の OS の既定のエンコード名*
- *コード・ページ932（SJIS）のエンコード名*
- *コード・ページ50222（JIS）のエンコード名*
- *コード・ページ51932（EUC）のエンコード名*
- *コード・ページ65001（UTF8）のエンコード名*
- *現在の OS が対応する上記以外のエンコード名*

(2)［改行コード］

設定する改行コードをドロップダウン・リストにより選択します。

以下の項目を選択することができます。

- 現在の改行コードを維持
- Windows (CR LF)
- Macintosh (CR)
- Unix (LF)

デフォルトでは，“現在の改行コードを維持”が選択されます。

なお，改行コードを変更した後は，設定した改行コードがデフォルトで選択されます。

(3)［ファイルを読み込み直す］

［OK］ボタンのクリック時に，選択したエンコード，および改行コードでファイルを読み込み直すかどうかをチェック・ボックスにより選択します。

デフォルトでは，チェック・ボックスをチェックしません。

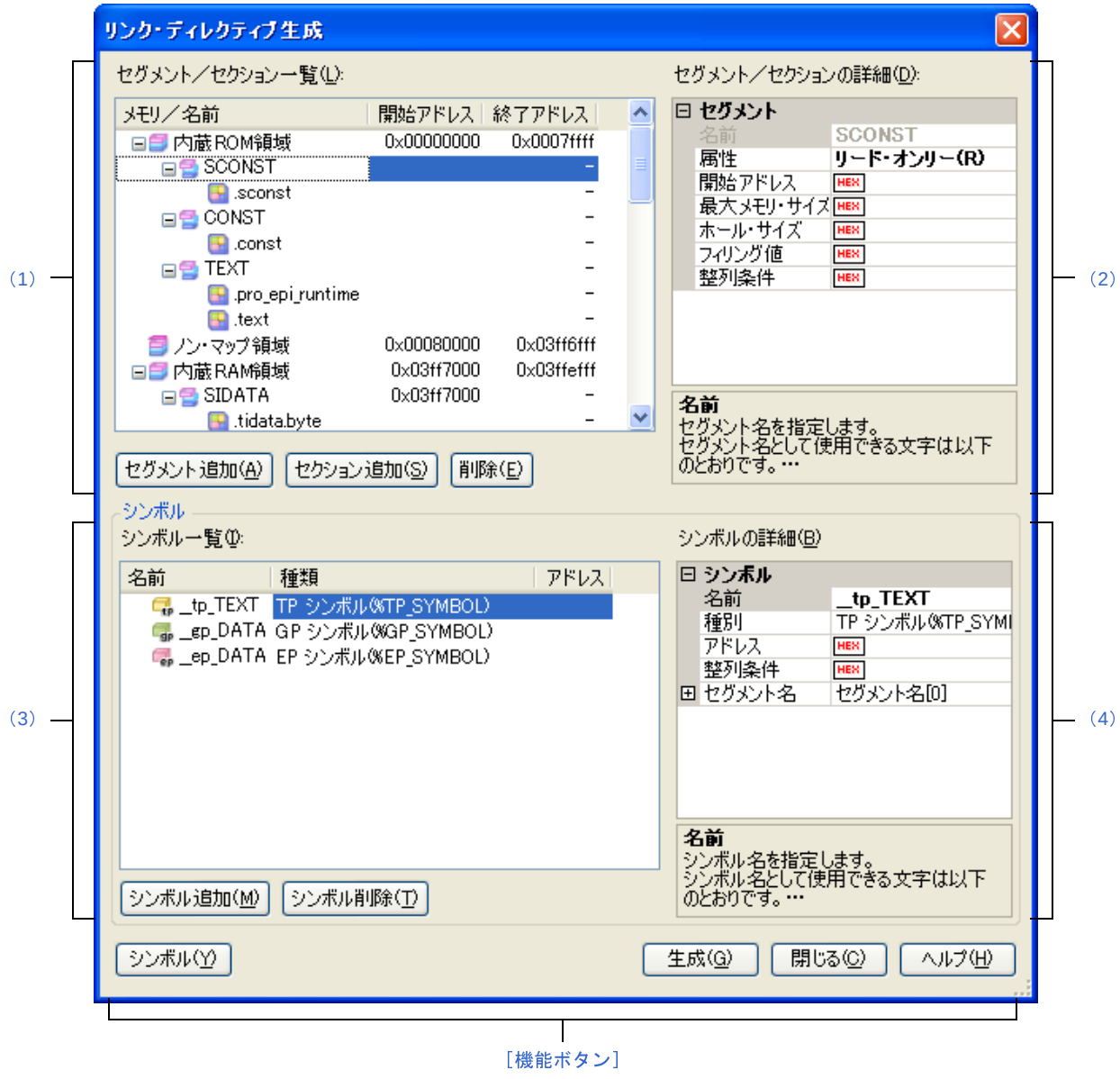
［機能ボタン］

ボタン	機能
OK	選択したエンコード，および改行コードを対象ファイルに設定し，本ダイアログをクローズします。 ［ファイルを読み込み直す］をチェックした場合は，選択したエンコード，および改行コードを対象ファイルに設定し，ファイルを読み込み直します。そののち，本ダイアログをクローズします。
キャンセル	エンコード，および改行コードの設定をキャンセルし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

リンク・ディレクティブ生成 ダイアログ

指定したメモリ，セグメント，セクション，シンボルの配置情報から，リンク・ディレクティブ・ファイルを生成します。

図 A―36 リンク・ディレクティブ生成 ダイアログ



ここでは，次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [機能ボタン]

[オープン方法]

- プロジェクト・ツリー パネル上において，ビルド・ツール・ノードを選択したのち，コンテキスト・メニュー→［リンク・ディレクティブ・ファイルを生成する ...］を選択

[各エリアの説明]

(1)［セグメント／セクション一覧］エリア

デバイスのメモリ配置情報と，現在設定されているセグメントとセクションの一覧を表示します。

(a)［メモリ／名前］

メモリ領域，セグメント，およびセクションの名前を表示します。

メモリ領域は，以下のうち，該当するメモリ領域の名前を表示します。

- 内蔵 ROM 領域
- ノン・マップ領域
- 内蔵 RAM 領域
- データフラッシュ領域

セグメント，およびセクションについては，この項目を直接編集することができます。セグメント名，およびセクション名を変更すると，［セグメント／セクションの詳細］エリアの［名前］も変更されます。

注意　セグメント名，およびセクション名は，予約セクションの扱いによって，編集できない場合があります。詳細については，［セグメント／セクションの詳細］エリアの備考を参照してください。

(b)［開始アドレス］

メモリ領域，セグメント，セクションの開始アドレスを表示します。

セグメント，およびセクションについては，この項目を直接編集することができます。開始アドレスを変更すると，［セグメント／セクションの詳細］エリアの［開始アドレス］も変更されます。

(c)［終了アドレス］

メモリ領域の終了アドレスを表示します。

セグメント，およびセクションの行については，“ー”が表示されます。

(d) ボタン

セグメント追加	一覧で選択している行の直下に，新しいセグメントを追加します。 セグメント名は，デフォルトで“NewSegment_*XXX*”となります（*XXX*：0 ～ 255 の 10 進数）。 セグメントの詳細設定は，［セグメント／セクションの詳細］エリアで行います。 なお，このボタンは，セクションの行を選択している場合，および一覧に 256 個のセグメントを登録している場合は無効となります。

セクション追加	一覧で選択している行の直下に，新しいセクションを追加します。 セクション名は，デフォルトで“NewSection_*XXX*”となります（*XXX*：0～255の10進数）。 セクションの詳細設定は，［セグメント／セクションの詳細］エリアで行います。 なお，このボタンは，一覧に256個のセクションを登録している場合は無効となります。
削除	一覧で選択しているセグメント，またはセクションを削除します。 セグメントを削除する場合は，セグメントに含まれているセクションも削除します。

また，このエリアは，次の機能を備えています。

- 行の展開／折りたたみ表示の切り替え

  行をダブルクリック，または行の先頭にあるマーク／マークをクリックすることにより，各行の展開／折りたたみ表示の切り替えを行うことができます。

- セグメント，およびセクションの行の移動

  ドラッグ・アンド・ドロップにより，セグメント，およびセクションの行を移動することができます。

  備考　セグメントを移動する場合は，セグメントに含まれるセクションも移動します。

- セグメント，およびセクションのコピー

  セグメント，またはセクションを選択したのち，［Ctrl］＋［C］キーの押下によりコピー，［Ctrl］＋［V］キーの押下により貼り付けを行うことができます。
  貼り付け位置は，［Ctrl］＋［V］キーの押下時に選択している行の直下となります。
  コピー後のセグメント，およびセクションの名前には，先頭に“Copy_”が付加されます。

  備考 1.　セグメントをコピーする場合，セグメントに含まれるセクションはコピーしません。
  2.　コピー後のセグメント，およびセクションの開始アドレスは，空欄となります。
  3.　コピー先のセグメントの属性により，コピーできない場合は，エラーとなります。

(2)［セグメント／セクションの詳細］エリア

［セグメント／セクション一覧］エリアで選択したセグメント／セクションの詳細情報の表示，および編集を行います。

（a）セグメントの詳細情報

名前	セグメント名を指定します。 使用可能な文字は，数字（0 ～ 9），英大文字（A ～ Z），英小文字（a ～ z），アンダスコア（_），ドット（.），スラッシュ（/），円マーク，バックスラッシュ（\）です。		
	デフォルト	NewSegment_*XXX*（*XXX*：0 ～ 255 の 10 進数）	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集	
	指定可能値	1022 文字までの文字列	
属性	セグメントの属性を選択します。 セグメントが予約セクションを含んでいる場合は，そのセクションの属性により，セグメントの属性も設定できるものが限られる場合があります。その場合，設定できない属性については，ドロップダウン・リストに項目が表示されません。		
	デフォルト	- 内蔵 ROM 領域，またはノン・マップ領域に追加した場合 実行可能 (RX) - 内蔵 RAM 領域に追加した場合 リード／ライト可能 (RW) - データフラッシュ領域に追加した場合 リード・オンリー (R)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	実行可能 (RX)	読み出し，実行が可能なセグメントに設定します。
		リード・オンリー (R)	読み出しが可能なセグメントに設定します。
		リード／ライト可能 (RW)	読み出し，書き込みが可能なセグメントに設定します。
		すべて可能 (RWX)	読み出し，書き込み，実行が可能なセグメントに設定します。
開始アドレス	セグメントを配置する開始アドレスを指定します。 空欄の場合は，コンパイラのリンク機能により直前のセグメントの後に配置されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xFFFFFFFF（16 進数）	
最大メモリ・サイズ	セグメントの最大メモリ・サイズを指定します。 空欄の場合は，コンパイラのリンク機能により 0x100000（バイト）として扱われます。 リンク時に，指定した最大メモリ・サイズを越えた場合は，エラーとなります。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xFFFFFFFF（16 進数）	

ホール・サイズ	セグメント間のホール・サイズを指定します。 空欄の場合は，コンパイラのリンク機能により 0x0（バイト）として扱われます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0x0 ～ 0xFFFFFFFF（16 進数）
フィリング値	セグメント間のホールを埋める値（フィリング値）を指定します。 空欄の場合は，コンパイラのリンク機能により 0x0000 として扱われます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0x0000 ～ 0xFFFF（16 進数）
整列条件	セグメントの整列条件（アライメント値）を指定します。 奇数の値を指定した場合は，自動的に 1 を足して偶数の値に変更します。 空欄の場合は，コンパイラのリンク機能により 0x8 として扱われます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0x0 ～ 0xFF（16 進数）

（b）セクションの詳細情報

名前	セクション名を指定します。 使用可能な文字は，数字（0 ～ 9），英大文字（A ～ Z），英小文字（a ～ z），アンダスコア（_），ドット（.），スラッシュ（/），円マーク，バックスラッシュ（\）です。	
	デフォルト	NewSection_*XXX*（*XXX*：0 ～ 255 の 10 進数）
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	1022 文字までの文字列
種類	セクションの種類を選択します。 オブジェクト・ファイル内に実際の値を持っているセクション（.text，.data 等）の場合は［初期値あり (PROGBITS)］，実際の値を持っていないセクション（.bss，.sbss 等）の場合は［初期値なし (NOBITS)］を選択します。	
	デフォルト	初期値あり (PROGBITS)
	変更方法	ドロップダウン・リストによる選択
	指定可能値	初期値あり (PROGBITS)：初期値ありセクションに設定します。
		初期値なし (NOBITS)：初期値なしセクションに設定します。

属性	セクションの属性を選択します。		
	デフォルト	- 親セグメントの属性が［実行可能 AX)］の場合 実行可能 (AX) - 親セグメントの属性が［リード・オンリー (A)］の場合 リード・オンリー (A) - 親セグメントの属性が［リード／ライト可能 (AW)］の場合 リード／ライト可能 (AW) - 親セグメントの属性が［すべて可能 (AWX)］の場合 すべて可能 (AWX)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	実行可能 (AX)	メモリを占有し，実行が可能なセクションに設定します。 なお，この項目は，親セグメントの属性が［リード・オンリー (R)］の場合は表示されません。
		リード・オンリー (A)	メモリを占有するセクションに設定します。
		リード／ライト可能 (AW)	メモリを占有し，書き込みが可能なセクションに設定します。 なお，この項目は，親セグメントの属性が［リード／ライト可能 (RW)］，または［すべて可能 (RWX)］の場合のみ表示されます。
		GP 相対 1 命令アクセス (AWG)	メモリを占有し，書き込みが可能，およびグローバル・ポインタ（gp）と 16 ビットのディスプレースメントを用いて参照が可能なメモリ範囲内に割り付けるセクションに設定します。 なお，この項目は，親セグメントの属性が［リード／ライト可能 (RW)］，または［すべて可能 (RWX)］の場合のみ表示されます。
		すべて可能 (AWX)	メモリを占有し，書き込み，実行が可能なセクションに設定します。 なお，この項目は，親セグメントの属性が［すべて可能 (RWX)］の場合のみ表示されます。
開始アドレス	セクションを配置する開始アドレスを指定します。 空欄の場合は，コンパイラのリンク機能により直前のセクションの後に配置されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xFFFFFFFF（16 進数）	

ホール・サイズ	セクション間のホール・サイズを指定します。 空欄の場合は，コンパイラのリンク機能により 0x0（バイト）として扱われます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0x0 ～ 0xFFFFFFFF（16 進数）
整列条件	セクションの整列条件を指定します。 奇数の値を指定した場合は，自動的に 1 を足して偶数の値に変更します。 空欄の場合は，コンパイラのリンク機能により 0x4 として扱われます。 ただし，セクション名が“.tidata.byte”，または“.tibss.byte”の場合は，奇数の値を指定することができます。また，空欄の場合は，コンパイラのリンク機能により 0x1 として扱われます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0x0 ～ 0xFF（16 進数）
入力セクション名	入力セクション名を指定します。 使用可能な文字は，数字（0 ～ 9），英大文字（A ～ Z），英小文字（a ～ z），アンダスコア（_），ドット（.），スラッシュ（/），円マーク，バックスラッシュ（\）です。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	1022 文字までの文字列
オブジェクト・ファイル名	入力セクションを含んでいるオブジェクト・ファイル名を指定します。 指定したオブジェクト・ファイル名はサブプロパティとして表示されます。	
	デフォルト	オブジェクト・ファイル名 [*設定数*]
	変更方法	［...］ボタンをクリックし，オブジェクト・ファイル指定 ダイアログによる編集

備考　予約セクションについては，以下のように扱います。

- C コンパイラで予約セクションとして定義されているセクションを［名前］，または［入力セクション名］に指定した場合，［種類］，および［属性］は編集不可となり，自動で値が設定されます。
  予約セクション名と自動で設定する値の組み合わせを，以下に示します。

予約セクション名	種類	属性
.pro_epi_runtime	初期値あり (PROGBITS)	実行可能 (AX)
.text	初期値あり (PROGBITS)	実行可能 (AX)
.data	初期値あり (PROGBITS)	リード／ライト可能 (AW)
.sedata	初期値あり (PROGBITS)	リード／ライト可能 (AW)
.sidata	初期値あり (PROGBITS)	リード／ライト可能 (AW)

予約セクション名	種類	属性
.tidata	初期値あり (PROGBITS)	リード／ライト可能 (AW)
.tidata.byte	初期値あり (PROGBITS)	リード／ライト可能 (AW)
.tidata.word	初期値あり (PROGBITS)	リード／ライト可能 (AW)
.bss	初期値なし (NOBITS)	リード／ライト可能 (AW)
.sebss	初期値なし (NOBITS)	リード／ライト可能 (AW)
.sibss	初期値なし (NOBITS)	リード／ライト可能 (AW)
.tibss	初期値なし (NOBITS)	リード／ライト可能 (AW)
.tibss.byte	初期値なし (NOBITS)	リード／ライト可能 (AW)
.tibss.word	初期値なし (NOBITS)	リード／ライト可能 (AW)
.sdata	初期値あり (PROGBITS)	GP 相対 1 命令アクセス (AWG)
.sbss	初期値あり (PROGBITS)	GP 相対 1 命令アクセス (AWG)
.const	初期値あり (PROGBITS)	リード・オンリー (A)
.sconst	初期値あり (PROGBITS)	リード・オンリー (A)

- 以下の予約セクションは，リンカにより，割り当て可能なセグメント名が固定されています。

セクション名	セグメント名
.sidata，.sibss，.tidata，.tibss，.tidata byte，.tibss.byte，.tidata.word，.tibss.word	SIDATA
.sedata，.sebss	SEDATA
.sconst	SCONST

これらのセクション名を［名前］に指定した場合，親セグメントの名前を参照します。
なお，これらのセクションは，セグメント内で移動することはできますが，他のセグメントへ移動することはできません。

- 以下の予約セクションは，リンカにより，出力セクション名と入力セクション名の対応が固定されているため，入力セクション名を省略してもリンカによって自動で割り付けられます。
  .pro_epi_runtime，.tidata，.tibss，.tidata.byte，.tibss.byte，.tidata.word，.sidata，.sibss，.sedata，.sebss

(3)［シンボル一覧］エリア

現在設定されているシンボルの一覧を表示します。

(a)［名前］

シンボルの名前を表示します。

この項目は直接編集することもできます。シンボル名を変更すると，［シンボルの詳細］エリアの［名前］も変更されます。

(b)［種別］

シンボルの種別を表示します。

この項目は直接編集することもできます。種別を変更すると，［シンボルの詳細］エリアの［種別］も変更されます。

(c)［アドレス］

シンボルを配置する開始アドレスを表示します。

この項目は直接編集することもできます。アドレスを変更すると，［シンボルの詳細］エリアの［アドレス］も変更されます。

(d) ボタン

シンボル追加	一覧で選択している行の直下に，新しいシンボルを追加します。 シンボル名は，デフォルトで“NewSymbol_*XXX*”となります（*XXX*：0 ～ 255 の 10 進数）。 シンボルの詳細設定は，［シンボルの詳細］エリアで行います。 なお，このボタンは，一覧に 256 個のシンボルを登録している場合は無効となります。
シンボル削除	一覧で選択しているシンボルを削除します。

また，このエリアは，次の機能を備えています。

- シンボルの行の移動

ドラッグ・アンド・ドロップにより，シンボルの行を移動することができます。

(4)［シンボルの詳細］エリア

［シンボル一覧］エリアで選択したシンボルの詳細情報の表示，および編集を行います。

名前	シンボル名を指定します。 使用可能な文字は，数字（0 ～ 9），英大文字（A ～ Z），英小文字（a ～ z），アンダスコア（_），ドット（.），スラッシュ（/），円マーク，バックスラッシュ（\）です。	
	デフォルト	NewSymbol_*XXX*（*XXX*：0 ～ 255 の 10 進数）
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	1022 文字までの文字列

種別	シンボルの種別を選択します。		
	デフォルト	TP シンボル (%TP_SYMBOL)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	TP シンボル (%TP_SYMBOL)	TP シンボルに設定します。
		GP シンボル (%GP_SYMBOL)	GP シンボルに設定します。
		EP シンボル (%EP_SYMBOL)	EP シンボルに設定します。
ベース・シンボル名	ベース・シンボル（GP シンボル値を定める際に用いる TP シンボル）として，すでに存在する TP シンボルを指定します。 ベース・シンボル名を指定すると，TP シンボル値からのオフセット値が GP シンボル値となります。 使用可能な文字は，数字（0 ～ 9），英大文字（A ～ Z），英小文字（a ～ z），アンダスコア（_），ドット（.），スラッシュ（/），円マーク，バックスラッシュ（\）です。 なお，本プロパティは，［種別］プロパティで［GP シンボル (%GP_SYMBOL)］を選択した場合のみ表示されます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集	
	指定可能値	1022 文字までの文字列	
アドレス	シンボルを配置する開始アドレスを指定します。 空欄の場合は，コンパイラのリンク機能のアドレス決定規則により，自動的にアドレスが割り振られます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xFFFFFFFF（16 進数）	
整列条件	シンボルの整列条件（アライメント値）を指定します。 奇数の値を指定した場合は，自動的に 1 を足して偶数の値に変更します。 空欄の場合は，コンパイラのリンク機能により 0x4 として扱われます。		
	デフォルト	空欄	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0x0 ～ 0xFF（16 進数）	
セグメント名	TP シンボル値，GP シンボル値の参照対象とするセグメント名を指定します。 指定したセグメント名はサブプロパティとして表示されます。 なお，本プロパティは，［種別］プロパティで［EP シンボル (%EP_SYMBOL)］を選択した場合は表示されません。		
	デフォルト	セグメント名 [*設定数*]	
	変更方法	[...] ボタンをクリックし，セグメント指定 ダイアログによる編集	

［機能ボタン］

ボタン	機能
シンボル	［シンボル一覧］エリア，および［シンボルの詳細］エリアの表示／非表示を切り替えます。
生成	指定したメモリ，セグメント／セクション，シンボルの配置情報を元に，リンク・ディレクティブ・ファイル（ファイル名：*プロジェクト名*.dir）を生成し，プロジェクトに登録します。 リンク・ディレクティブ・ファイルの生成先は，プロジェクト・フォルダとなります。 生成したリンク・ディレクティブ・ファイルは，プロジェクト・ツリーのファイル・ノードにも表示されます。 生成したリンク・ディレクティブ・ファイルはビルド対象となります。すでにリンク・ディレクティブ・ファイルをプロジェクトに登録していた場合，登録済みのリンク・ディレクティブ・ファイルはビルド対象外となります。
閉じる	本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

オブジェクト・ファイル指定 ダイアログ

本ダイアログの呼び出し元に設定するオブジェクト・ファイルを，プロジェクトに追加されているオブジェクト・ファイル，およびライブラリ・ファイルの中から選択します。

図 A―37　オブジェクト・ファイル指定 ダイアログ

ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- リンク・ディレクティブ生成 ダイアログの［セグメント／セクション一覧］エリアでセクションを選択したのち，［セグメント／セクションの詳細］エリアの［オブジェクト・ファイル名］において，［...］ボタンをクリック

［各エリアの説明］

(1)［オブジェクト・ファイル一覧］エリア

リンク・ディレクティブ生成 ダイアログをオープンしたプロジェクトに追加されているオブジェクト・ファイル，およびライブラリ・ファイルと，リンク・ディレクティブ生成 ダイアログでそれらを指定しているセクションの一覧を表示します。

(a)［オブジェクト・ファイル］

以下のファイル名一覧を表示します。

本ダイアログをオープンしたリンク・ディレクティブ生成 ダイアログの［セグメント／セクションの詳細］エリアの［オブジェクト・ファイル名］に設定するファイルをチェック・ボックスにより選択します。

- プロジェクトに追加されているソース・ファイルから生成されるオブジェクト・モジュール・ファイル
- プロジェクト・ツリーに直接追加したオブジェクト・モジュール・ファイル
- プロジェクト・ツリーに直接追加したライブラリ・ファイル

備考 1. ファイル名にマウス・カーソルをあわせると，そのファイルの絶対パスをポップアップ表示します。

2. 本ダイアログをオープンしたリンク・ディレクティブ生成 ダイアログの［セグメント／セクションの詳細］エリアの［オブジェクト・ファイル名］において，すでにオブジェクト・ファイルを設定していた場合は，該当するオブジェクト・ファイルのチェック・ボックスはデフォルトでチェック状態となります。

(b)［セクション］

リンク・ディレクティブ生成 ダイアログで該当オブジェクト・ファイルを指定しているセクションを表示します。

オブジェクト・ファイルを複数のセクションから指定している場合は，カンマで区切って表示します。

オブジェクト・ファイルを指定しているセクションが存在しない場合は，空欄となります。

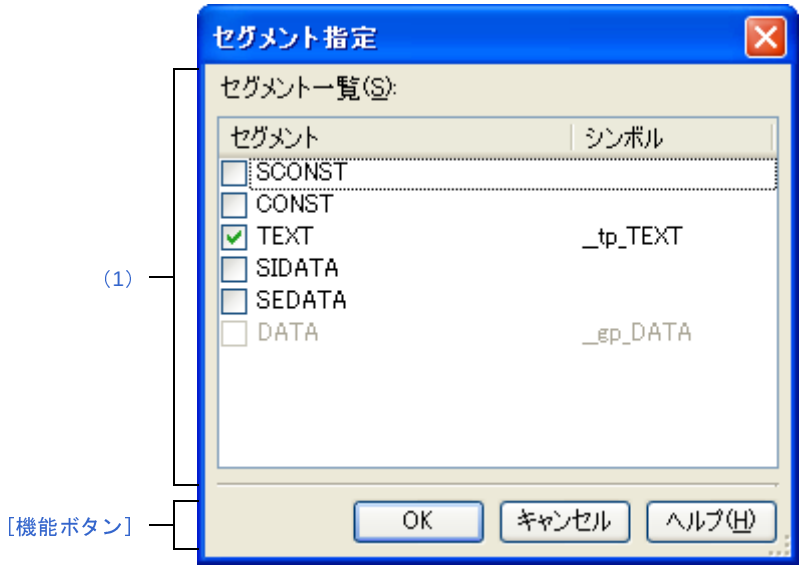
［機能ボタン］

ボタン	機能
OK	本ダイアログをクローズし，選択したファイルをリンク・ディレクティブ生成 ダイアログの［セグメント／セクションの詳細］エリアの［オブジェクト・ファイル名］に設定します。
キャンセル	ファイルの選択をキャンセルし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

セグメント指定 ダイアログ

本ダイアログの呼び出し元に設定するセグメントを，リンク・ディレクティブ生成 ダイアログで現在設定しているセグメントの中から選択します。

図 A―38　セグメント指定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- リンク・ディレクティブ生成 ダイアログの［シンボル一覧］エリアでシンボルを選択したのち，［シンボルの詳細］エリアの［セグメント名］において，［...］ボタンをクリック

［各エリアの説明］

(1)［セグメント一覧］エリア

リンク・ディレクティブ生成 ダイアログで現在設定しているセグメントと，それらを指定しているシンボルの一覧を表示します。

(a)［セグメント］

リンク・ディレクティブ生成 ダイアログで現在設定しているセグメント名一覧を表示します。

本ダイアログをオープンしたリンク・ディレクティブ生成 ダイアログの［シンボルの詳細］エリアの［セグメント名］に設定するセグメントをチェック・ボックスにより選択します。

備考 1. ファイル名にマウス・カーソルをあわせると，そのファイルの絶対パスをポップアップ表示します。

2. 本ダイアログをオープンしたリンク・ディレクティブ生成 ダイアログの［シンボルの詳細］エリアの［セグメント名］において，すでにセグメントを設定していた場合は，該当するセグメントのチェック・ボックスはデフォルトでチェック状態となります。

3. 本ダイアログをオープンしたシンボル以外のシンボルを指定しているセグメントの場合は，該当するセグメントのチェック・ボックスはチェック不可状態となります。

（b）［シンボル］

表示しているセグメントを指定しているシンボルを表示します。

セグメントを指定しているシンボルが存在しない場合は，空欄となります。

［機能ボタン］

ボタン	機能
OK	本ダイアログをクローズし，選択したセグメントをリンク・ディレクティブ生成 ダイアログの［シンボルの詳細］エリアの［セグメント名］に設定します。
キャンセル	ファイルの選択をキャンセルし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

リンク順設定 ダイアログ

リンカに入力するオブジェクト・モジュール・ファイル，およびライブラリ・ファイルの参照，およびリンク順の設定を行います。

図 A―39　リンク順設定 ダイアログ

ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロジェクト・ツリー パネルにおいて，ビルド・ツール・ノードを選択したのち，コンテキスト・メニュー→［リンク順を設定する ...］を選択

［各エリアの説明］

(1) ファイル一覧表示エリア

リンカに入力するファイルの一覧を表示します。

(a)［ファイル］

以下のファイルのファイル名一覧を，リンカへの入力順に表示します。

- 選択しているメイン・プロジェクト，またはサブプロジェクトに追加されているソース・ファイルから生成されるオブジェクト・モジュール・ファイル

- 選択しているメイン・プロジェクト，またはサブプロジェクトのプロジェクト・ツリーに直接追加したオブジェクト・モジュール・ファイル
- 選択しているメイン・プロジェクト，またはサブプロジェクトのプロジェクト・ツリーに直接追加したライブラリ・ファイル

デフォルトでは，プロジェクトに追加されている順番となります。
ファイルの表示順を変更することにより，リンカへのファイルの入力順を設定することができます。
表示順の変更は，［↑］，および［↓］ボタン，またはファイル名のドラッグ・アンド・ドロップにより行います。

備考 1. ファイル名にマウス・カーソルをあわせると，そのファイルの存在する場所がプロジェクト・ファイルと同一のドライブの場合は相対パスで，異なるドライブの場合は絶対パスでポップアップ表示します。

2. 新規に追加したソース・ファイルから生成されるオブジェクト・モジュール・ファイル，および新規に追加したオブジェクト・モジュール・ファイルは，一覧の最後のオブジェクト・モジュール・ファイルの次に追加されます。新規に追加したライブラリ・ファイルは，一覧の最後に追加されます。

3. ファイル名をドラッグ・アンド・ドロップする際，連続して並んでいるファイル名のみ複数選択することができます。

（b）ボタン

↑	選択しているファイルを上へ移動します。
↓	選択しているファイルを下へ移動します。

備考　上記のボタンは，ファイルを選択していない場合は無効となります。

［機能ボタン］

ボタン	機能
OK	リンカへのファイルの入力順をファイル一覧表示エリアの表示順に設定し，本ダイアログをクローズします。
キャンセル	リンク順の設定をキャンセルし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

ビルド・モード設定 ダイアログ

ビルド・モードの追加と削除，および現在のビルド・モードの一括設定を行います。

図 A―40　ビルド・モード設定 ダイアログ

ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ビルド］メニュー→［ビルド・モードの設定 ...］を選択

［各エリアの説明］

(1) ［変更後のビルド・モード］エリア

［ビルド・モードの一覧］エリアで選択しているビルド・モードを表示します。

(a) ボタン

すべてに適用	表示しているビルド・モードを現在開いているプロジェクトのメイン・プロジェクト，およびすべてのサブプロジェクトに設定します。

(2) ［ビルド・モードの一覧］エリア

現在開いているプロジェクト（メイン・プロジェクト，およびサブプロジェクト）に存在するすべてのビルド・モードを一覧表示します。

デフォルトでは，すべてのプロジェクトの現在のビルド・モードが一致している場合は，そのビルド・モードが選択されます。一致していない場合は，“DefaultBuild”が選択されます。

一部のメイン・プロジェクト，およびサブプロジェクトのみに存在するビルド・モードには，“*”が付加されます。

なお，ビルド・モードには，あらかじめ“DefaultBuild”が用意されており，常に先頭に表示されます。

(a) ボタン

複製 ...	選択しているビルド・モードを複製します。 文字列入力 ダイアログがオープンし，入力した名前でビルド・モードを複製し，現在開いているプロジェクトのメイン・プロジェクト，およびすべてのサブプロジェクトに追加します。 なお，“*”が付加されているビルド・モードを複製する場合，そのビルド・モードがメイン・プロジェクト，およびサブプロジェクトに存在しなければ，DefaultBuild を複製します。 登録可能なビルド・モード数は，20 個までです。
削除	選択しているビルド・モードを削除します。 ただし，DefaultBuild を削除することはできません。
名前の変更 ...	選択しているビルド・モードの名前を変更します。 文字列入力 ダイアログがオープンし，入力した名前でビルド・モードの名前を変更します。

注意 ビルド・モードを複製，およびビルド・モードの名前を変更する場合，すでに存在するビルド・モードと同名の名前を使用することはできません。

備考 1. ビルド・モード名として指定可能な文字数は 127 文字までです。入力内容が正しくない場合，以下のメッセージがツールチップ表示されます。

メッセージ	説明
同名のビルド・モードがすでに存在します。	同名のビルド・モードがすでに存在します。
127 文字を越える文字を指定できません。	長い名前（128 文字以上）のビルド・モードが指定されました。
ビルド・モード名が不正です。文字 (¥ , /, :, *, ?, ", <, >, |) は使用できません。	不正なビルド・モード名が指定されました。ビルド・モード名のフォルダを作成するため，文字 (¥ , / , :, *, ?, ", <, >, |) は使用できません。

2. 登録可能なビルド・モード数は，20 個までです。入力内容が正しくない場合，以下のメッセージがツールチップ表示されます。

メッセージ	説明
1 つのプロジェクト／サブプロジェクトに設定できるビルド・モード数は，20 個までです。	登録するビルド・モード数が 20 個を越えました。

［機能ボタン］

ボタン	機能
閉じる	本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

バッチ・ビルド ダイアログ

プロジェクト（メイン・プロジェクト，およびサブプロジェクト）が持つビルド・モードを一括して，ビルド，リビルド，クリーンを行います。

備考　バッチ・ビルド順は，プロジェクトのビルド順に従い，サブプロジェクト，メイン・プロジェクトの順となります。
1 つのメイン・プロジェクト，またはサブプロジェクトについて複数のビルド・モードを選択した場合は，そのサブプロジェクトで選択されているすべてのビルド・モードでビルドを行ったのち，次のサブプロジェクト，またはメイン・プロジェクトのビルドを行います。

図 A—41　バッチ・ビルド ダイアログ

ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ビルド］メニュー→［バッチ・ビルド ...］を選択

［各エリアの説明］

(1)［ビルド・モード一覧］エリア

現在開いているプロジェクトが持つメイン・プロジェクト，およびサブプロジェクトの名前と，それらが持つビルド・モード，定義マクロの組み合わせの一覧を表示します。

(a)［プロジェクト］

現在開いているプロジェクトが持つメイン・プロジェクト，およびサブプロジェクトを表示します。

ビルドを行うメイン・プロジェクト，およびサブプロジェクトとビルド・モードの組み合わせをチェック・ボックスにより選択します。

プロジェクトを作成後，最初に本ダイアログをオープンした場合は，すべてのチェック・ボックスをチェックしません。2 回目以降は前回のチェック状態を保持します。

(b)［ビルド・モード］

メイン・プロジェクト，およびサブプロジェクトが持つビルド・モードを表示します。

(c)［定義マクロ］

メイン・プロジェクト，およびサブプロジェクトとそのビルド・モードの組み合わせに対して，プロパティ パネルの［コンパイル・オプション］タブ，および［アセンブル・オプション］タブで設定している定義マクロを“｜”で区切って表示します。

なお，コンパイル・オプションの定義マクロ，アセンブル・オプションの定義マクロの順で表示し，コンパイル・オプションの定義マクロとアセンブル・オプションの定義マクロの間は“,”で区切って表示します。

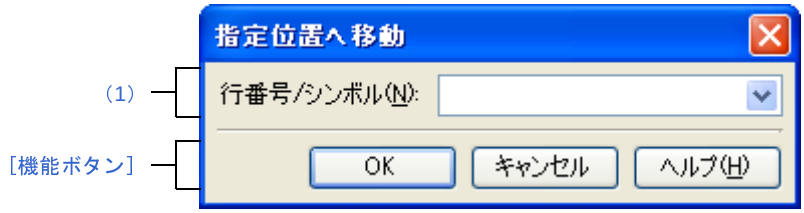
［機能ボタン］

ボタン	機能
ビルド	本ダイアログをクローズし，選択しているプロジェクトをそのビルド・モードでビルドします。ビルドの実行結果は，出力 パネルに表示されます。 ビルド完了後，ビルド・モードは本ダイアログをオープンする前の設定に戻ります。 なお，本ボタンは，プロジェクトを選択していない場合は無効となります。
リビルド	本ダイアログをクローズし，選択しているプロジェクトをそのビルド・モードでリビルドします。リビルドの実行結果は，出力 パネルに表示されます。 リビルド完了後，ビルド・モードは本ダイアログをオープンする前の設定に戻ります。 なお，本ボタンは，プロジェクトを選択していない場合は無効となります。
クリーン	本ダイアログをクローズし，選択しているプロジェクトのそのビルド・モードでビルドしたファイルを削除します。クリーンの実行結果は，出力 パネルに表示されます。 クリーン完了後，ビルド・モードは本ダイアログをオープンする前の設定に戻ります。 なお，本ボタンは，プロジェクトを選択していない場合は無効となります。
閉じる	本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

指定位置へ移動 ダイアログ

指定した位置にキャレットを移動します。

図 A—42　指定位置へ移動 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［編集］メニュー→［移動 ...］を選択

［各エリアの説明］

（1）［行番号 / シンボル］エリア

キャレットを移動したい箇所の行番号（10 進数），またはシンボル名を指定します。

テキスト・ボックスに直接入力するか，またはドロップダウン・リストより入力履歴項目を選択します（最大履歴数：10 個）。

［機能ボタン］

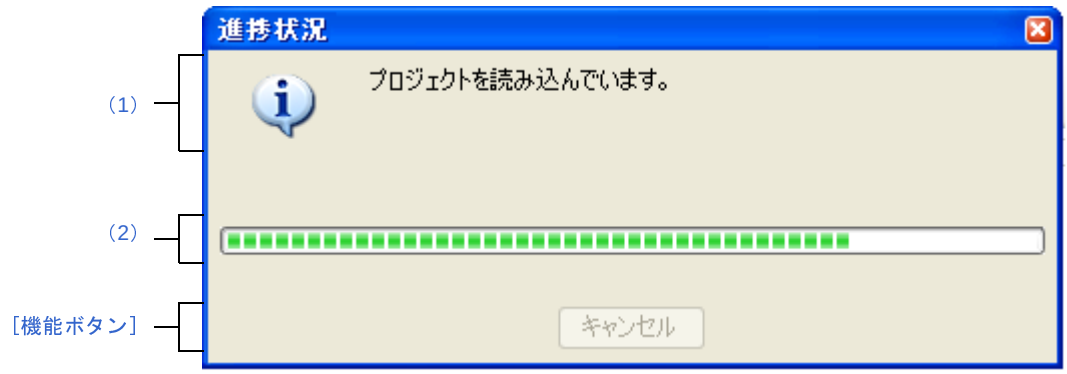
ボタン	機能
OK	指定した位置を対象パネルの表示の中央とし，キャレットを移動します。 なお，開いているファイルを登録しているプロジェクトがアクティブ，かつライブラリ用のプロジェクト以外の場合のみ有効です。
キャンセル	設定を無効とし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

処理中表示 ダイアログ

時間を要する処理を行っている際に，その進捗状況の表示を行います。

本ダイアログは，実行中の処理が完了した場合，自動的にクローズします。

図 A―43　処理中表示 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- 時間を要する処理において，メッセージが発生した際に自動的に表示

［各エリアの説明］

(1) メッセージ表示エリア

処理中に発生したメッセージを表示します（編集不可）。

(2) プログレスバー

現在実行中の処理の進捗状況をバーの長さで表示します。

なお，進捗率が 100% に達した場合（右端までバーの長さが達した場合），本ダイアログは自動的にクローズします。

［機能ボタン］

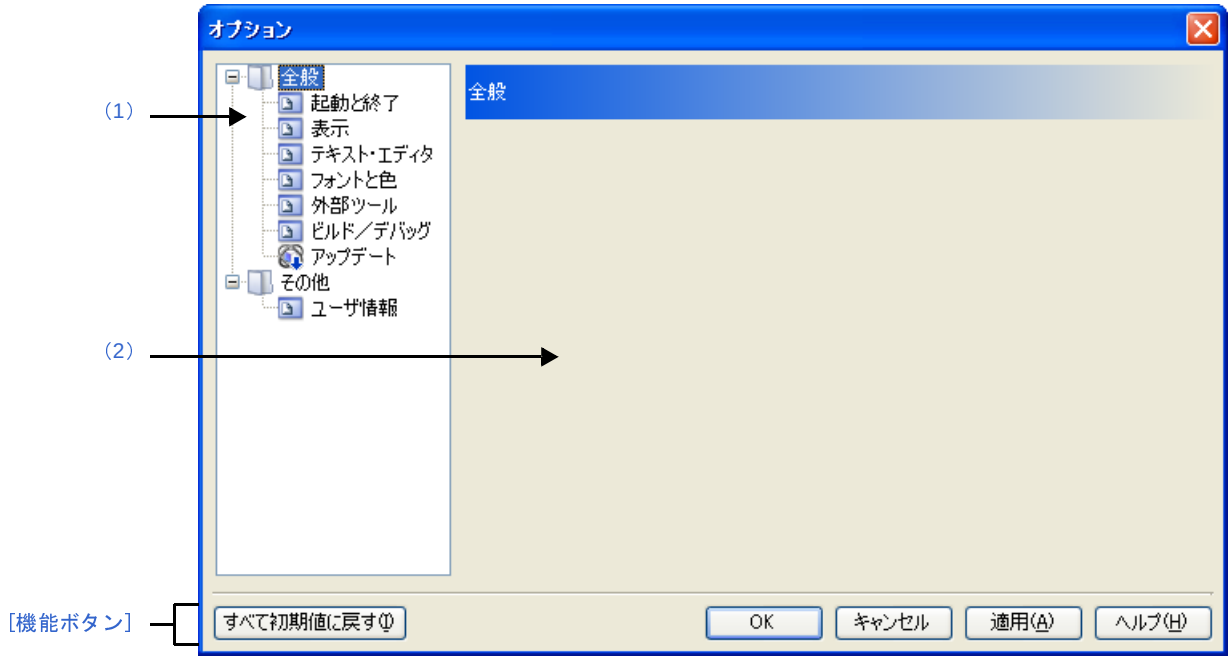
ボタン	機能
キャンセル	現在実行中の処理を中断し，本ダイアログをクローズします。 ただし，実行中の処理の中断が不可能な場合，本ボタンは無効となります。

オプション ダイアログ

CubeSuite+ の各種環境設定を行います。

本ダイアログでの設定は，使用中のユーザの設定として保存されます。

図 A―44　オプション ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ツール］メニュー→［オプション ...］を選択

［各エリアの説明］

(1) カテゴリ選択エリア

設定したい項目を次のカテゴリから選択します。

カテゴリ	設定内容
［全般 - 起動と終了］カテゴリ	起動，または終了時に関連した設定を行います。
［全般 - 表示］カテゴリ	表示に関連した設定を行います。
［全般 - テキスト・エディタ］カテゴリ	テキスト・エディタに関連した設定を行います。

カテゴリ	設定内容
［全般 - フォントと色］カテゴリ	各パネルで表示するフォントと色に関連した設定を行います。
［全般 - 外部ツール］カテゴリ	外部ツールを起動する際の設定を行います。
［全般 - ビルド／デバッグ］カテゴリ	ビルド，またはデバッグに関連した設定を行います。
［全般 - アップデート］カテゴリ	アップデートに関連した設定を行います。
［その他 - ユーザ情報］カテゴリ	ユーザ情報に関連した設定を行います。

備考　［全般 - ビルド／デバッグ］以外のカテゴリについては，「CubeSuite+ 起動編」を参照してください。

（2）設定エリア

選択したカテゴリに対して，各種オプションを設定するエリアです。

各カテゴリの設定方法についての詳細は，該当するカテゴリの項を参照してください。

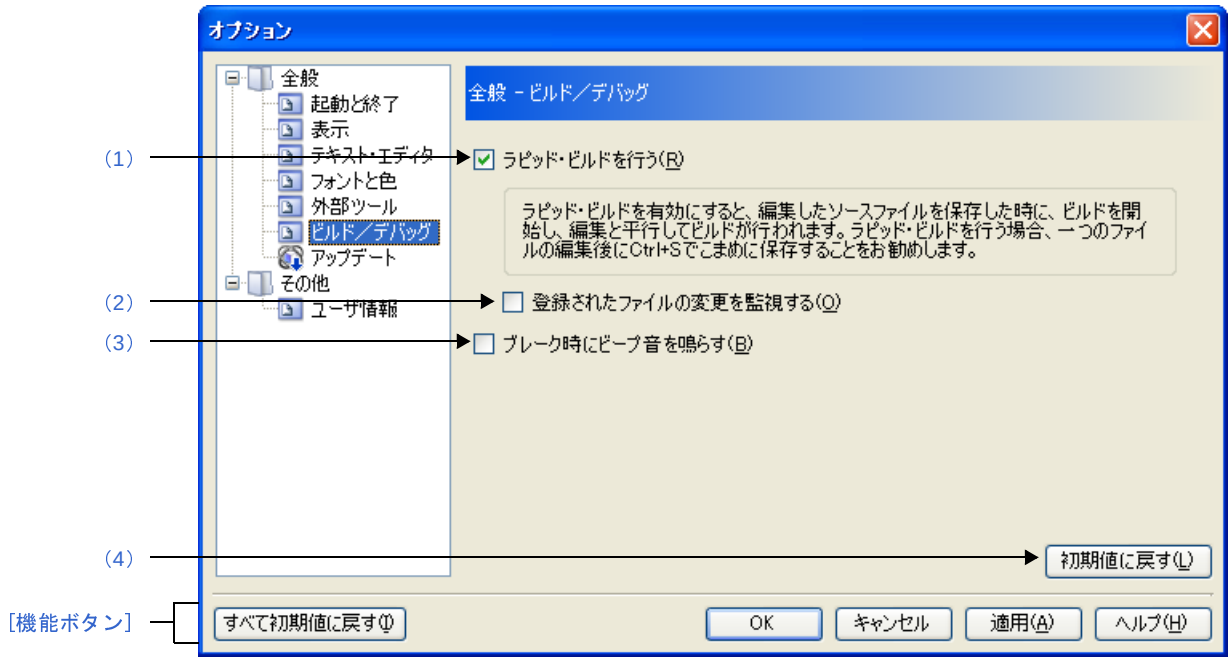
［機能ボタン］

ボタン	機能
すべて初期値に戻す	本ダイアログのすべての設定項目をデフォルトの状態に戻します。 ただし，［全般 - 外部ツール］カテゴリでは，新規登録した内容の削除は行いません。
OK	変更した設定内容を適用し，本ダイアログをクローズします。
キャンセル	変更した設定内容を無効とし，本ダイアログをクローズします。
適用	変更した設定内容を適用します（本ダイアログをクローズしません）。
ヘルプ	本ダイアログのヘルプを表示します。

［全般 - ビルド／デバッグ］カテゴリ

全般に関わる設定のうち，ビルド，またはデバッグに関連した設定を行います。

図 A―45　オプション ダイアログ（［全般 - ビルド／デバッグ］カテゴリ）



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ツール］メニュー→［オプション ...］を選択

［各エリアの説明］

(1) ［ラピッド・ビルドを行う］

☑	ラピッド・ビルド機能[注]を有効にします（デフォルト）。
☐	ラピッド・ビルド機能を使用しません。

注　編集したソース・ファイルの保存時に，ビルドを自動で開始する機能です。
本機能を有効にすることにより，ソース・ファイルの編集と同時にビルドを行うことができます。
なお，本機能を使用する場合，ソース・ファイル編集後，こまめに上書き保存することを推奨します。

(2)［登録されたファイルの変更を監視する］

☑	プロジェクトに登録されたソース・ファイルの変更を監視し，外部エディタなどで編集／保存されたときに，ラピッド・ビルドを開始します。
☐	プロジェクトに登録されたソース・ファイルの変更を監視し，外部エディタなどで編集／保存されたときに，ラピッド・ビルドを開始しません（デフォルト）。

備考 ［ラピッド・ビルドを行う］チェック・ボックスにチェックが付いている場合のみ有効です。

注意 1. 本項目をチェックし，かつ，ラピッド・ビルドの対象となったファイルをビルド前に実行するコマンド，ビルド後に実行するコマンドなどで自動で編集／上書きするように登録した場合，ラピッド・ビルドが終了しなくなります。
ラピッド・ビルドが終了しなくなった場合は，本項目のチェックを外して，ラピッド・ビルドを停止してください。

2. 本項目をチェックし，かつ，プロジェクトに登録されたソース・ファイルで存在しないファイル（プロジェクト・ツリーでグレー表示されたファイル）がある場合，エクスプローラなどでファイルを再登録しても，監視状態にはなりません。
監視状態にするためには，プロジェクト・ファイルを読み込み直すか，または本項目のチェックを一旦外してダイアログを閉じた後，再度本項目をチェックしてください。

(3)［ブレーク時にビープ音を鳴らす］

☑	プログラムの実行が，ブレーク・イベント（ハードウエア・ブレーク／ソフトウエア・ブレーク）により停止した際，ビープ音を鳴らします。
☐	プログラムの実行が，ブレーク・イベント（ハードウエア・ブレーク／ソフトウエア・ブレーク）により停止した際，ビープ音を鳴らしません（デフォルト）。

(4) ボタン・エリア

初期値に戻す	現在表示している項目をすべてデフォルトに戻します。

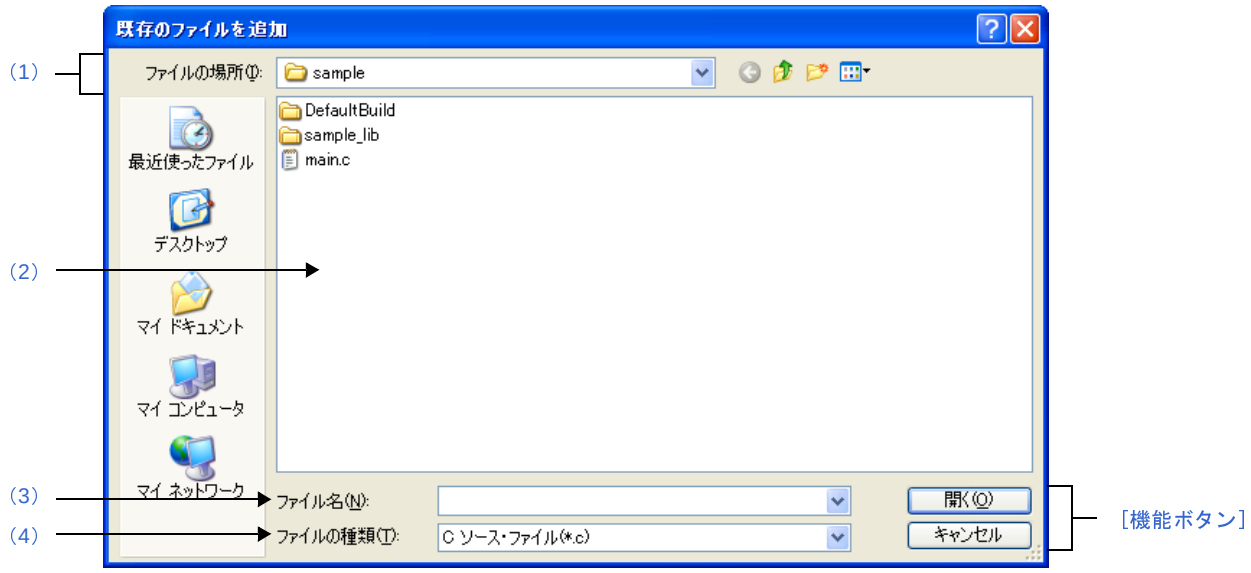
［機能ボタン］

ボタン	機能
すべて初期値に戻す	本ダイアログのすべての設定項目をデフォルトの状態に戻します。 ただし，［全般 - 外部ツール］カテゴリでは，新規登録した内容の削除は行いません。
OK	変更した設定内容を適用し，本ダイアログをクローズします。
キャンセル	変更した設定内容を無効とし，本ダイアログをクローズします。
適用	変更した設定内容を適用します（本ダイアログをクローズしません）。
ヘルプ	本ダイアログのヘルプを表示します。

既存のファイルを追加 ダイアログ

プロジェクトに追加する既存のファイルの選択を行います。

図 A―46　既存のファイルを追加 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ファイル］メニュー→［追加］→［既存のファイルを追加 ...］を選択
- プロジェクト・ツリー パネルにおいて，プロジェクト・ノード，サブプロジェクト・ノード，ファイル・ノード，ファイルのいずれかを選択したのち，コンテキスト・メニュー→［追加］→［既存のファイルを追加 ...］を選択

［各エリアの説明］

(1)［ファイルの場所］エリア

プロジェクトに追加するファイルが存在するフォルダを選択します。

デフォルトでは，プロジェクト・フォルダが選択されます。

(2) ファイルの一覧エリア

［ファイルの場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

プロジェクトに追加するファイルのファイル名を指定します。

(4)［ファイルの種類］エリア

プロジェクトに追加するファイルのファイルの種類（ファイル・タイプ）を選択します。

C ソース・ファイル (*.c)	C ソース・ファイル
ヘッダ・ファイル (*.h; *.inc)	ヘッダ・ファイル
アセンブル・ファイル (*.s)	アセンブラ・ソース・ファイル
リンク・ディレクティブ・ファイル (*.dir; .*dr)	リンク・ディレクティブ・ファイル
セクション・ファイル（*.sf）	セクション・ファイル
アーカイブ・ファイル (*.a)	アーカイブ・ファイル
オブジェクト・ファイル (*.o)	オブジェクト・ファイル
テキスト・ファイル (*.txt)	テキスト形式
すべてのファイル (*.*)	すべての形式（デフォルト）

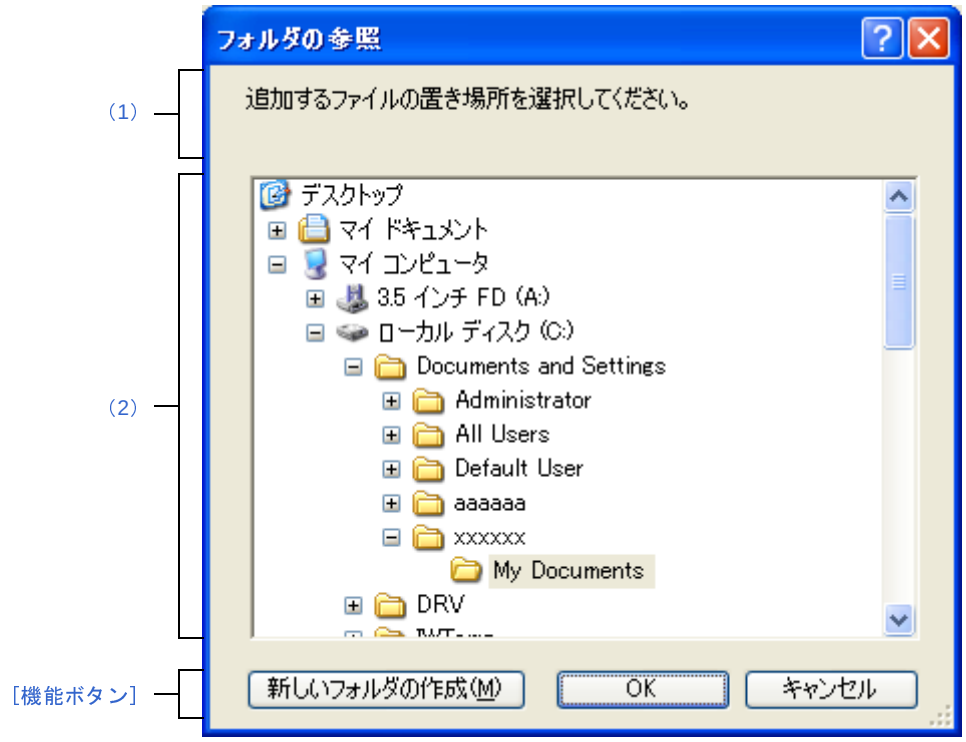
［機能ボタン］

ボタン	機能
開く	指定したファイルをプロジェクトに追加します。
キャンセル	本ダイアログをクローズします。

フォルダの参照 ダイアログ

本ダイアログの呼び出し元に設定するフォルダの選択を行います。

図 A—47　フォルダの参照 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ファイル追加 ダイアログにおいて，［作成場所］エリア内の［参照 ...］ボタンをクリック
- パス編集 ダイアログにおいて，パス編集エリア内の［参照 ...］ボタンをクリック
- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［共通オプション］タブの［出力ファイルの種類と場所］カテゴリの［中間ファイル出力フォルダ］，［よく使うオプション（リンカ）］カテゴリの［出力フォルダ］，［よく使うオプション（ROM 化プロセッサ）］カテゴリの［ROM 化用オブジェクト・ファイル出力フォルダ］，［よく使うオプション（ヘキサ・コンバータ）］カテゴリの［ヘキサ・ファイル出力フォルダ］，［よく使うオプション（セクション・ファイル・ジェネレータ）］カテゴリの［セクション・ファイル出力フォルダ］，［その他］カテゴリの［一時作業フォルダ］
  - ［コンパイル・オプション］タブの［出力ファイル］カテゴリの［アセンブリ・ファイルの出力フォルダ］，［アセンブル・リストの出力フォルダ］，［頻度情報ファイルの出力フォルダ］

- ［アセンブル・オプション］タブの［アセンブル・リスト］カテゴリの［アセンブル・リスト・ファイル出力フォルダ］
- ［リンク・オプション］タブの［出力ファイル］カテゴリの［出力フォルダ］，［リンク・マップ］カテゴリの［リンク・マップ・ファイル出力フォルダ］
- ［ROM 化プロセス・オプション］タブの［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイル出力フォルダ］，［セクション・リスト］カテゴリの［ROM 化用セクション・ファイル出力フォルダ］，［メモリ・マップ］カテゴリの［メモリ・マップ・ファイル出力フォルダ］
- ［ヘキサ・コンバート・オプション］タブの［出力ファイル］カテゴリの［ヘキサ・ファイル出力フォルダ］
- ［アーカイブ・オプション］タブの［出力ファイル］カテゴリの［出力フォルダ］
- ［セクション・ファイル・ジェネレート・オプション］タブの［出力ファイル］カテゴリの［セクション・ファイル出力フォルダ］
- ［個別コンパイル・オプション］タブの［出力ファイル］カテゴリの［アセンブリ・ファイルの出力フォルダ］，［アセンブル・リストの出力フォルダ］，［頻度情報ファイルの出力フォルダ］
- ［個別アセンブル・オプション］タブの［アセンブル・リスト］カテゴリの［アセンブル・リスト・ファイル出力フォルダ］

［各エリアの説明］

（1）メッセージ・エリア

本ダイアログで選択するフォルダに関するメッセージを表示します。

（2）フォルダの場所エリア

本ダイアログの呼び出し元に設定するフォルダを選択します。

デフォルトでは，呼び出し元に設定しているフォルダが選択されます。

備考 呼び出し元が空欄，または存在しないパスを設定している場合は，“C:￥Documents and Settings￥*ユーザ名*￥My Documents”が選択されます。

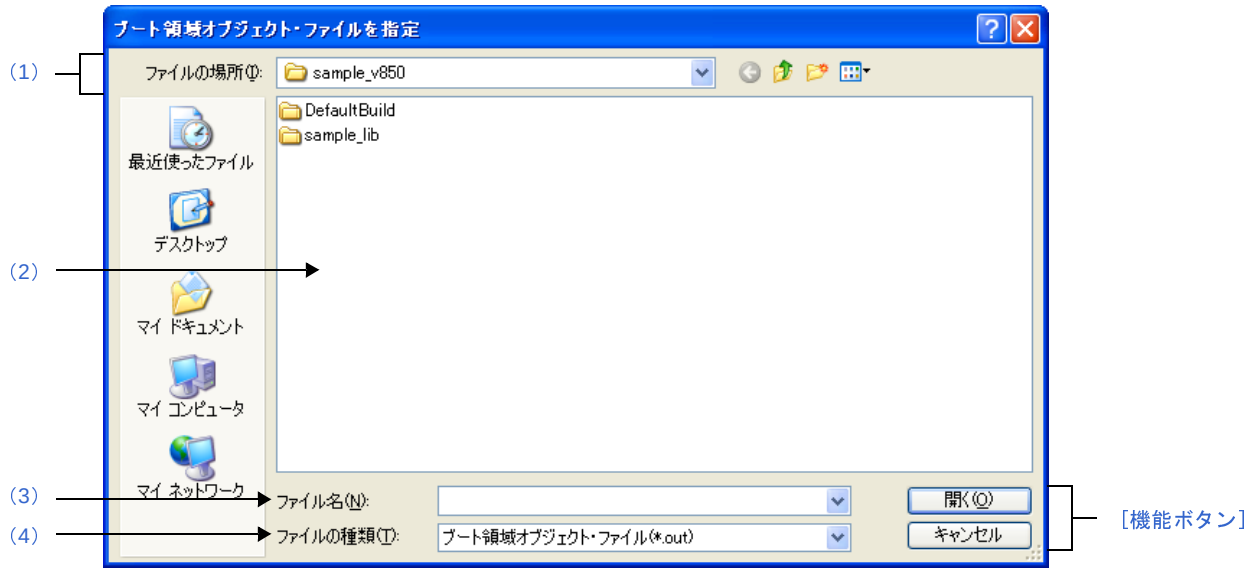
［機能ボタン］

ボタン	機能
新しいフォルダの作成	選択したフォルダの直下に新しいフォルダを作成します。 フォルダ名は，デフォルトで“新しいフォルダ”となります。
OK	指定したフォルダのパスを本ダイアログの呼び出し元に設定します。
キャンセル	本ダイアログをクローズします。

ブート領域オブジェクト・ファイルを指定 ダイアログ

本ダイアログの呼び出し元に設定するブート領域オブジェクト・ファイルの選択を行います。

図 A―48　ブート領域オブジェクト・ファイルを指定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［共通オプション］タブの［フラッシュ］カテゴリの［ブート領域オブジェクト・ファイル名］

［各エリアの説明］

(1)［ファイルの場所］エリア

本ダイアログの呼び出し元に設定するファイルが存在するフォルダを選択します。

デフォルトでは，プロジェクト・フォルダが選択されます。

(2) ファイルの一覧エリア

［ファイルの場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

本ダイアログの呼び出し元に設定するファイルの名前を指定します。

（4）［ファイルの種類］エリア

本ダイアログの呼び出し元に設定するファイルの種類（ファイル・タイプ）を選択します。

ブート領域オブジェクト・ファイル (*.out)	ブート領域オブジェクト・ファイル（デフォルト）
すべてのファイル (*.*)	すべての形式

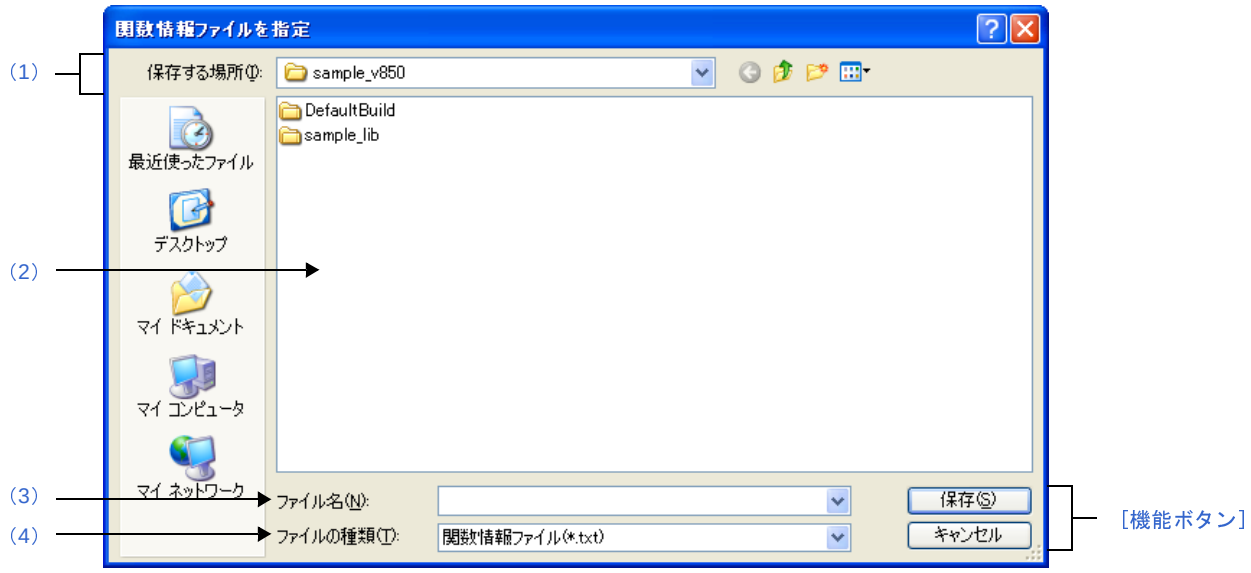
［機能ボタン］

ボタン	機能
開く	本ダイアログの呼び出し元に指定したファイルを設定します。
キャンセル	本ダイアログをクローズします。

関数情報ファイルを指定 ダイアログ

本ダイアログの呼び出し元に設定する関数情報ファイルの選択を行います。

図 A―49 関数情報ファイルを指定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［コンパイル・オプション］タブの［最適化 (詳細)］カテゴリの［関数情報ファイル名］
  - ［個別コンパイル・オプション］タブの［最適化 (詳細)］カテゴリの［関数情報ファイル名］

［各エリアの説明］

(1)［保存する場所］エリア

本ダイアログの呼び出し元に設定するファイルが存在するフォルダを選択します。

デフォルトでは，プロジェクト・フォルダが選択されます。

(2) ファイルの一覧エリア

［保存する場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

本ダイアログの呼び出し元に設定するファイルの名前を指定します。

(4)［ファイルの種類］エリア

本ダイアログの呼び出し元に設定するファイルの種類（ファイル・タイプ）を選択します。

関数情報ファイル (*.txt)	関数情報ファイル（デフォルト）
すべてのファイル (*.*)	すべての形式

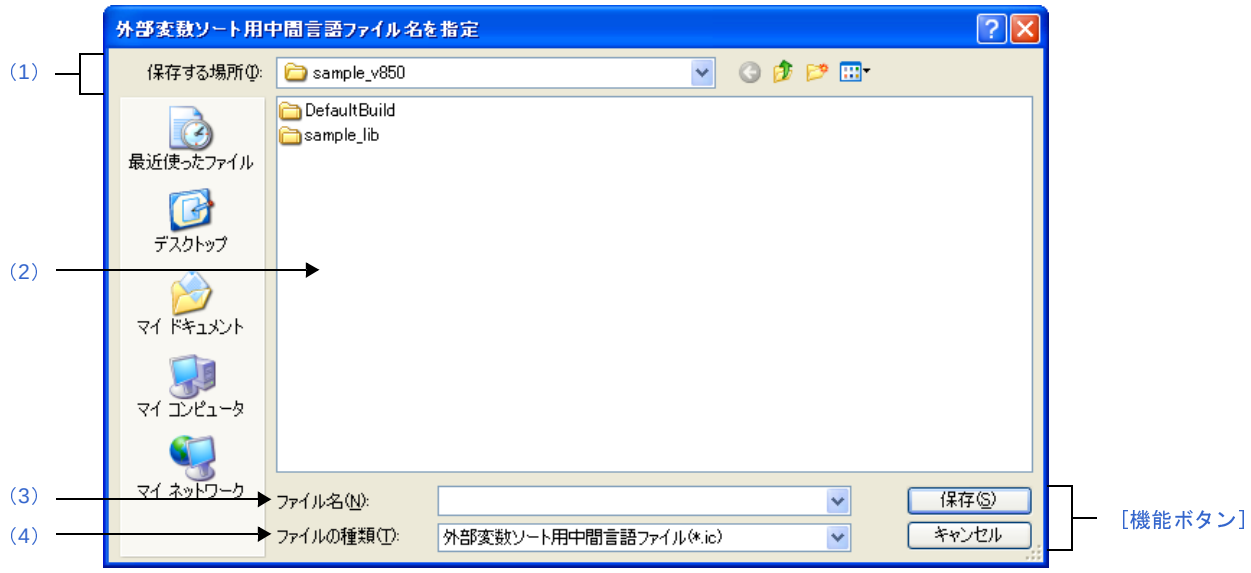
［機能ボタン］

ボタン	機能
保存	本ダイアログの呼び出し元に指定したファイルを設定します。
キャンセル	本ダイアログをクローズします。

外部変数ソート用中間言語ファイルを指定 ダイアログ

本ダイアログの呼び出し元に設定する外部変数ソート用中間言語ファイルの選択を行います。

図 A―50 外部変数ソート用中間言語ファイルを指定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［コンパイル・オプション］タブの［最適化 (詳細)］カテゴリの［外部変数ソート用中間言語ファイル名］

［各エリアの説明］

(1)［保存する場所］エリア

本ダイアログの呼び出し元に設定するファイルが存在するフォルダを選択します。

デフォルトでは，プロジェクト・フォルダが選択されます。

(2) ファイルの一覧エリア

［保存する場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

本ダイアログの呼び出し元に設定するファイルの名前を指定します。

（4）［ファイルの種類］エリア

本ダイアログの呼び出し元に設定するファイルの種類（ファイル・タイプ）を選択します。

外部変数ソート用中間言語ファイル (*.ic)	外部変数ソート用中間言語ファイル

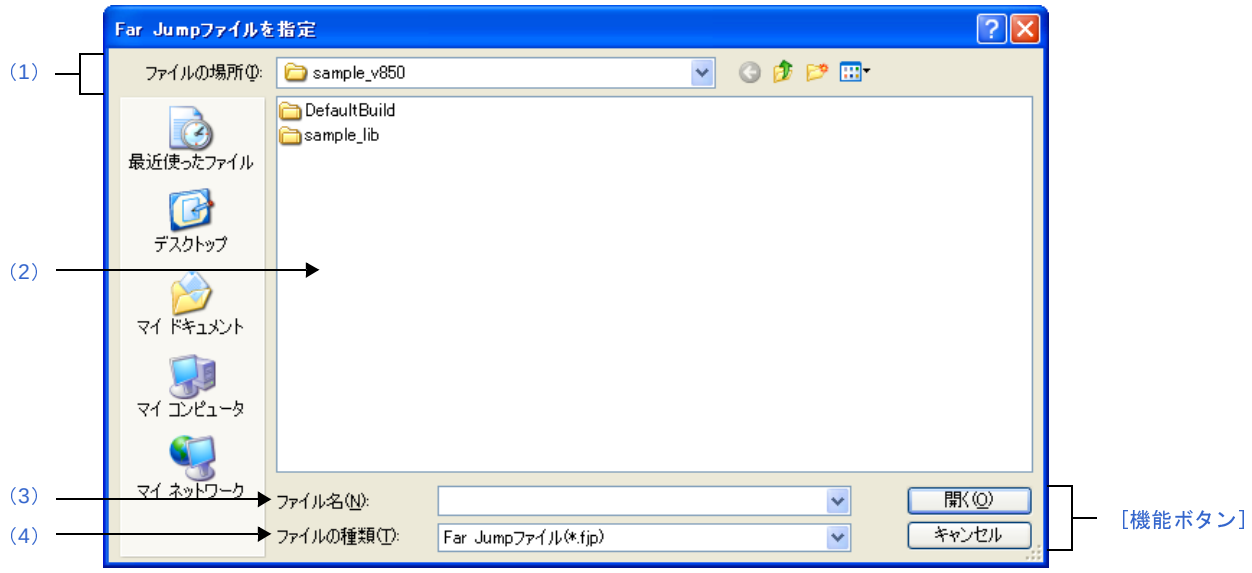
［機能ボタン］

ボタン	機能
保存	本ダイアログの呼び出し元に指定したファイルを設定します。
キャンセル	本ダイアログをクローズします。

Far Jump ファイルを指定 ダイアログ

本ダイアログの呼び出し元に設定する Far Jump ファイルの選択を行います。

図 A—51 Far Jump ファイルを指定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，［コンパイル・オプション］タブの［入力ファイル］カテゴリの［Far Jump ファイル名］プロパティを選択したのち，［...］ボタンをクリックしてパス編集 ダイアログをオープン
→ダイアログ上で［参照 ...］ボタンをクリック

［各エリアの説明］

(1) ［ファイルの場所］エリア

本ダイアログの呼び出し元に設定するファイルが存在するフォルダを選択します。

デフォルトでは，プロジェクト・フォルダが選択されます。

(2) ファイルの一覧エリア

［ファイルの場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

本ダイアログの呼び出し元に設定するファイルの名前を指定します。

(4)［ファイルの種類］エリア

本ダイアログの呼び出し元に設定するファイルの種類（ファイル・タイプ）を選択します。

Far Jump ファイル (*.fjp)	Far Jump ファイル

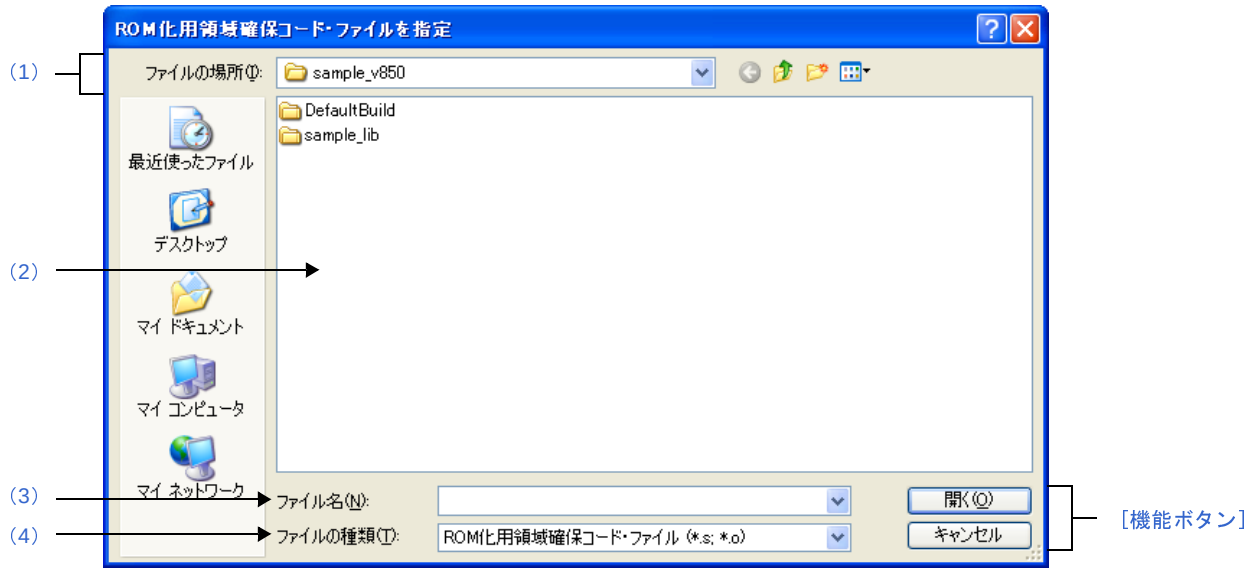
［機能ボタン］

ボタン	機能
開く	本ダイアログの呼び出し元に指定したファイルを設定します。
キャンセル	本ダイアログをクローズします。

ROM 化用領域確保コード・ファイルを指定 ダイアログ

本ダイアログの呼び出し元に設定する ROM 化用領域確保コード・ファイルの選択を行います。

図 A—52　ROM 化用領域確保コード・ファイルを指定 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルにおいて，以下のプロパティを選択したのち，［...］ボタンをクリック
  - ［ROM 化プロセス・オプション］タブの［入力ファイル］カテゴリの［ROM 化用領域確保コード・ファイル名］

［各エリアの説明］

(1)［ファイルの場所］エリア

本ダイアログの呼び出し元に設定するファイルが存在するフォルダを選択します。

デフォルトでは，プロジェクト・フォルダが選択されます。

(2) ファイルの一覧エリア

［ファイルの場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

本ダイアログの呼び出し元に設定するファイルの名前を指定します。

(4)［ファイルの種類］エリア

本ダイアログの呼び出し元に設定するファイルの種類（ファイル・タイプ）を選択します。

ROM 化用領域確保コード・ファイル名 (*.s; *.o)	ROM 化用領域確保コード・ファイル名（デフォルト）

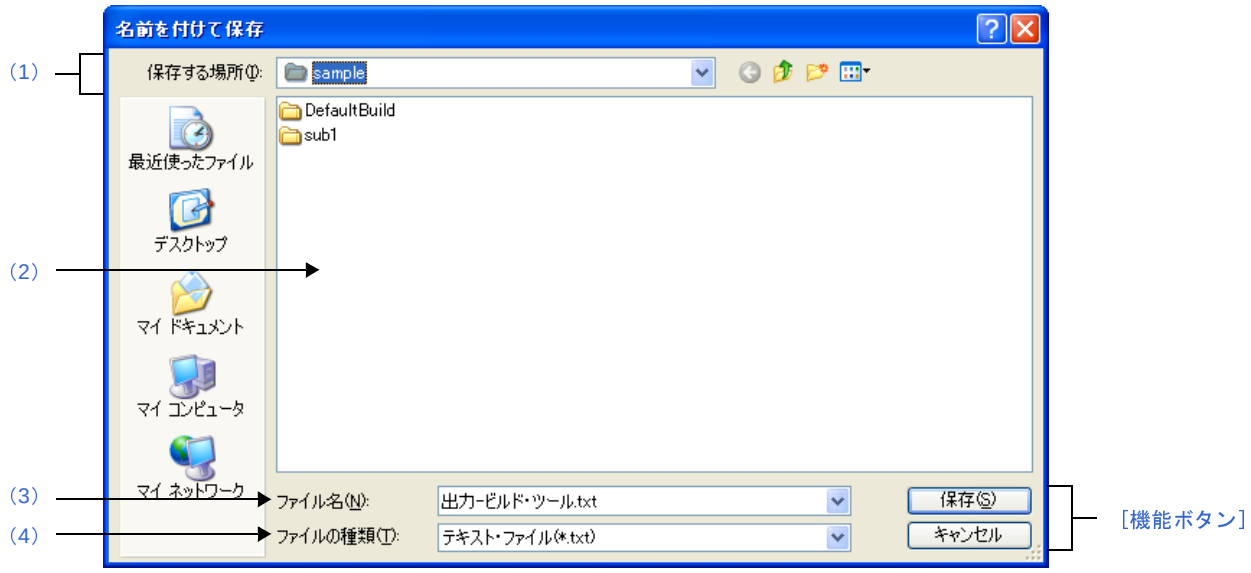
［機能ボタン］

ボタン	機能
開く	本ダイアログの呼び出し元に指定したファイルを設定します。
キャンセル	本ダイアログをクローズします。

名前を付けて保存 ダイアログ

編集中のファイル，または各パネルの内容を名前を付けてファイルに保存します。

図 A—53　名前を付けて保存 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- エディタ パネルにフォーカスがある状態で，［ファイル］メニュー→［名前を付けて *ファイル名*を保存 ...］を選択
- 出力 パネルにフォーカスがある状態で，［ファイル］メニュー→［名前を付けて *タブ名*を保存 ...］を選択

［各エリアの説明］

(1)［保存する場所］エリア

パネルに表示している内容をファイルに保存するためのフォルダを選択します。

デフォルトでは，以下のフォルダが選択されます。

(a) エディタ パネルの場合

現在編集しているファイルが存在しているフォルダ

(b) 出力 パネルの場合

初めて保存する場合は，プロジェクト・フォルダ，2 回目以降は前回選択したフォルダ

(2) ファイルの一覧エリア

［保存する場所］エリア，および［ファイルの種類］エリアで選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

保存する際のファイル名を指定します。

(4)［ファイルの種類］エリア

(a) エディタ パネルの場合

編集中のファイルの種類に依存して，次のファイルの種類（ファイル・タイプ）が表示されます。

テキスト・ファイル (*.txt)	テキスト形式
C ソース・ファイル (*.c)	C ソース・ファイル
ヘッダ・ファイル (*.h; *.inc)	ヘッダ・ファイル
アセンブル・ファイル (*.s)	アセンブラ・ソース・ファイル
リンク・ディレクティブ・ファイル (*.dir; .*dr)	リンク・ディレクティブ・ファイル
セクション・ファイル（*.sf）	セクション・ファイル
マップ・ファイル (*.map)	マップ・ファイル
ヘキサ・ファイル (*.hex)	ヘキサ・ファイル

(b) 出力 パネルの場合

次のファイルの種類（ファイル・タイプ）が表示されます。

テキスト・ファイル (*.txt)	テキスト形式

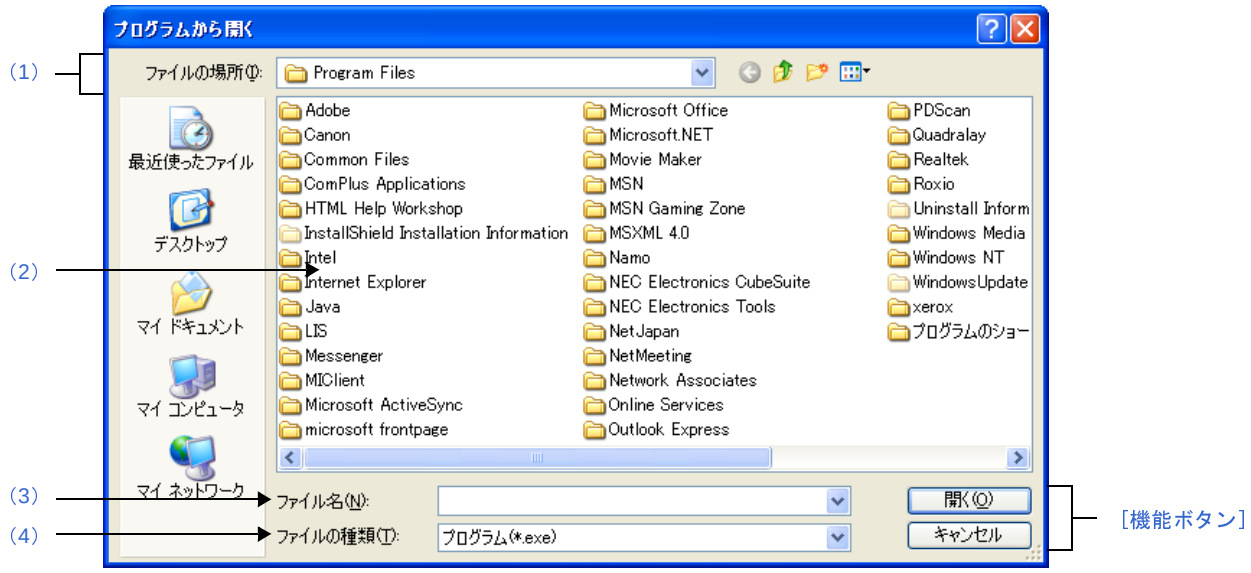
［機能ボタン］

ボタン	機能
保存	指定したファイル名でファイルを保存します。
キャンセル	本ダイアログをクローズします。

プログラムから開く ダイアログ

プロジェクト・ツリー上で選択しているファイルを開くアプリケーションの選択を行います。

図 A—54 プログラムから開く ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロジェクト・ツリー パネルにおいて，ファイルを選択したのち，コンテキスト・メニュー→［アプリケーションを指定して開く ...］を選択

［各エリアの説明］

(1)［ファイルの場所］エリア

ファイルを開くアプリケーションが存在するフォルダを選択します。

デフォルトでは，プログラム・フォルダ（Windows XP の場合は“C:￥Program Files”）が選択されます。

(2) ファイルの一覧エリア

［ファイルの場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

ファイルを開くアプリケーションの実行ファイル名を指定します。

（4）［ファイルの種類］エリア

ファイルを開くアプリケーションの実行ファイルの種類（ファイル・タイプ）を選択します。

プログラム (*.exe)	実行形式（デフォルト）
すべてのファイル (*.*)	すべての形式

［機能ボタン］

ボタン	機能
開く	指定したアプリケーションでファイルをオープンします。
キャンセル	本ダイアログをクローズします。

Stack Usage Tracer ウインドウ

スタック見積もりツールを起動した際，最初にオープンするウインドウです。

スタックの使用量を関数単位に確認／変更する際は，本ウインドウから行います。

図 A—55　Stack Usage Tracer ウインドウ

ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［注意］

［オープン方法］

- ［ツール］メニュー→［スタック見積もりツールの起動］を選択

［各エリアの説明］

(1) メニューバー

本エリアは，以下に示したメニュー群から構成されています。

(a)［ファイル］メニュー

選択した関数の最大経路の保存 ...	ツリー表示エリア／リスト表示エリアで選択した関数のスタック・サイズ（呼び出し関数のスタック・サイズを含む）が最大となる呼び出し経路を出力結果ファイルとして保存するための名前を付けて保存 ダイアログをオープンします。  ボタンのクリックと同様です。
選択した関数の全経路保存 ...	ツリー表示エリア／リスト表示エリアで選択した関数の全呼び出し経路を出力結果ファイルとして保存するための名前を付けて保存 ダイアログをオープンします。
全ルート関数の最大経路の保存 ...	ツリー表示エリアに表示されている関数の中でスタック・サイズが最大となる呼び出し経路を出力結果ファイルとして保存するための名前を付けて保存 ダイアログをオープンします。
全ルート関数の全経路保存 ...	ツリー表示エリアに表示されている全関数の全呼び出し経路を出力結果ファイルとして保存するための名前を付けて保存 ダイアログをオープンします。
スタックサイズ指定ファイルを開く ...	スタック・サイズ指定ファイルを読み込むためのファイルを開く ダイアログをオープンします。
スタックサイズ指定ファイルの保存 ...	スタックサイズ変更 ダイアログで行った各種操作結果（関数に対する情報の変更など）をスタック・サイズ指定ファイルとして保存するための名前を付けて保存 ダイアログをオープンします。
sk850 の終了	本ウインドウをクローズします。

備考　出力結果ファイルの保存は，テキスト形式（*.txt），またはCSV 形式（*.csv）に限られます。

(b)［表示］メニュー

スタックサイズの再計算	スタック・サイズの再計算を行います。  ボタンのクリックと同様です。
中止	スタック見積もりツールが実行中の処理（スタック・サイズの再計算など）を強制的に中止します。  ボタンのクリックと同様です。

アイコンの整列	リスト表示エリアの関数表示順序を変更します。	
	関数名順	関数名順に並べ替えます。
	アイコン順	アイコンの表示優先度順（高い：  ～ 低い：  ）に並べ替えます。
	スタックサイズ順	スタック・サイズ順に並べ替えます。
	単体スタックサイズ順	単体スタック・サイズ順に並べ替えます。
	加算サイズ順	加算サイズ順に並べ替えます。
	ファイル名順	ファイル名順に並べ替えます。

（c）［オプション］メニュー

サイズ不明関数・サイズ変更関数一覧	単体スタックサイズが不明な関数，情報（加算サイズ，再帰回数，呼び出し関数）の変更が行われた関数，およびスタック見積もりツールが強制的に加算サイズの設定を行った関数を一覧表示するためのサイズ不明関数・サイズ変更関数一覧 ダイアログをオープンします。
スタックサイズ変更 ...	ツリー表示エリア／リスト表示エリアで選択した関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するためのスタックサイズ変更 ダイアログをオープンします。 選択関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するダイアログです。  ボタンのクリックと同様です。
指定関数を初期値に戻す	選択関数の情報（加算サイズ，再帰回数，呼び出し関数）を初期状態に戻します。 選択関数の情報が初期状態から何ら変更されていない場合，本ボタンはグレー表記となります。
全関数を初期値に戻す	全関数の情報（加算サイズ，再帰回数，呼び出し関数）を初期状態に戻します。 関数の情報が初期状態から何ら変更されていない場合，本ボタンはグレー表記となります。

（d）［ヘルプ］メニュー

sk850 のヘルプ	スタック見積もりツールのヘルプを表示します。  ボタンのクリックと同様です。
sk850 のバージョン情報	sk850 のバージョン情報 ダイアログをオープンします。

（2）ツールバー

本エリアは，以下に示したボタン群から構成されています。

	ツリー表示エリア／リスト表示エリアで選択した関数のスタック・サイズ（呼び出し関数のスタック・サイズを含む）が最大となる呼び出し経路を出力結果ファイルとして保存するための名前を付けて保存 ダイアログをオープンします。 ［ファイル］メニュー→［選択した関数の最大経路の保存 ...］の選択と同様です。
	スタック・サイズの再計算を行います。 ［表示］メニュー→［スタックサイズの再計算］の選択と同様です。
	スタック見積もりツールが実行中の処理（スタック・サイズの再計算など）を強制的に中止します。 ［表示］メニュー→［中止］の選択と同様です。
	ツリー表示エリア／リスト表示エリアで選択した関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するためのスタックサイズ変更 ダイアログをオープンします。 ［オプション］メニュー→［スタックサイズ変更 ...］の選択と同様です。
	スタック見積もりツールのヘルプを表示します。 ［ヘルプ］メニュー→［sk850 のヘルプ］の選択と同様です。

(3) ツリー表示エリア

関数の呼び出し関係をツリー形式で表示します。

なお，関数名の直前に表示されているアイコンは，以下の意味を持ちます。

	同じ関数から直接呼び出される関数の中でスタック・サイズが最大となる関数
	スタックサイズ変更 ダイアログ，またはスタック・サイズ指定ファイルにより情報（加算サイズ，再帰回数，呼び出し関数）の変更が行われた関数
	再帰関数
	スタック見積もりツールがスタック情報を取得できていない関数
	上記以外の関数

備考　アイコンの表示優先度は，高い： ～ 低い：となります。

(a) コンテキスト・メニュー

本エリアの関数を選択したのち，マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

スタックサイズ変更 ...	選択した関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するためのスタックサイズ変更 ダイアログをオープンします。

(4) リスト表示エリア

関数単位のスタック情報（関数名，スタック・サイズ，単体スタック・サイズ，加算サイズ，ファイル名）をリスト形式で表示します。

関数名	関数名を表示します。 なお，本エリアに表示される関数は，第１階層（選択関数）／第２階層（選択関数が直接呼び出している関数）に限られます。
スタック・サイズ	スタック・サイズ（呼び出し関数のスタック・サイズを含む，単位：バイト）を表示します。
単体スタック・サイズ	単体スタック・サイズ（呼び出し関数のスタック・サイズを含まない，単位：バイト）を表示します。
加算サイズ	単体スタック・サイズに対して強制的に加算する値（単位：バイト）を表示します。
ファイル名	ファイル名を表示します。

なお，関数名の直前に表示されているアイコンは，以下の意味を持ちます。

	同じ関数から直接呼び出される関数の中でスタック・サイズが最大となる関数
	スタックサイズ変更 ダイアログ，またはスタック・サイズ指定ファイルにより情報（加算サイズ，再帰回数，呼び出し関数）の変更が行われた関数
	再帰関数
	スタック見積もりツールがスタック情報を取得できていない関数
	上記以外の関数

(a) コンテキスト・メニュー

本エリアの関数を選択したのち，マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

スタックサイズ変更 ...	選択した関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するためのスタックサイズ変更 ダイアログをオープンします。	
アイコンの整列	リスト表示エリアの関数表示順序を変更します。	
	関数名順	関数名順に並べ替えます。
	アイコン順	アイコンの表示優先度順（高い：  ～ 低い：  ）に並べ替えます。
	スタックサイズ順	スタック・サイズ順に並べ替えます。
	単体スタックサイズ順	単体スタック・サイズ順に並べ替えます。
	加算サイズ順	加算サイズ順に並べ替えます。
	ファイル名順	ファイル名順に並べ替えます。

(5) メッセージ表示エリア

スタック見積もりツールの操作ログを表示します。

［注意］

- アセンブリ・ファイル

スタック見積もりツールでは，C コンパイラが中間ファイルとして出力する“デッバグ情報の付与されたアセンブリ・ファイル”から各種情報を収集し，スタック・サイズの計算を行っています。
したがって，スタック見積もりツールを使用して，関数単位のスタック情報を得るためには，コンパイル・オプションで“デッバグ情報の付与されたアセンブリ・ファイル”の出力設定が必要となります。

- 静的な解析処理の実行タイミング

スタック見積もりツールでは，起動時に静的な解析処理を実行し，関数の呼び出し関係，および関数単位のスタック情報を本ウインドウに表示しています。
したがって，関数の呼び出し関係，または関数単位のスタック情報が変わるようなこと（ファイルの追加，コンパイル・オプションの変更，ソース・コードの変更など）を行っても，本ウインドウの該当情報は，連動して変化しません。

- 解析対象関数

スタック見積もりツールの解析対象関数は，C コンパイラが中間ファイルとして出力した“デバッグ情報の付与されたアセンブリ・ファイル”，またはビルド・ツールが提供しているライブラリ・ファイルに内包されている関数に限られます。
したがって，ユーザが記述したアセンブラ・ソース・ファイル，およびユーザが作成したライブラリ・ファイルに内包されている関数については，解析対象外となるため，スタックサイズ変更 ダイアログを用いて該当情報を設定する必要があります。

- アイコンの表示色

本ウインドウのツリー表示エリア／リスト表示エリアに表示されているアイコンについては，表示優先度（高い：  ～ 低い：  ）が付与されています。
したがって，“同じ関数から直接呼び出される関数の中でスタック・サイズが最大となる関数”を意味する  が表示されてる場合であっても，“単体スタック・サイズが不明：  ”などといった優先度の低い情報は GUI 上から隠れるため，注意が必要です。

- 最大スタック・サイズの確定

スタック見積もりツールでは，スタック・サイズが最大となる経路を検出する際，解析対処外の関数については，スタック・サイズが“0 バイト”であるものとして，該当経路の検出を行います。
したがって，最大スタック・サイズを確定する際には，サイズ不明関数・サイズ変更関数一覧 ダイアログの［サイズ不明関数リスト］に関数が表示されていないことを確認する必要があります。

- 再帰関数のツリー表示

本ウインドウのツリー表示エリアでは，再開関数の表示を“2 回目の呼び出しまで”としています。
したがって，“3 回目以降の呼び出し”については，非表示となります。

- ライブラリ関数 bsearch，exit，qsort

スタック見積もりツールでは，ビルド・ツールが提供しているライブラリ・ファイルに内包されている関数であっても，bsearch，exit，qsort については，不明関数として扱います。
したがって，これらの関数を使用する際には，スタックサイズ変更 ダイアログにおいて，各種情報（再帰回数，呼び出し関数など）を設定する必要があります。

- 呼び出し関数

スタック見積もりツールでは，スタックサイズ変更 ダイアログで追加可能な“呼び出し関数”をＣソース・ファイルに内包されている関数，および明示的な呼び出しが行われている関数（ポインタを用いた呼び出しでない）に限定しています。
したがって，スタックサイズ変更 ダイアログの［関数一覧］には，上記の条件に合致した関数のみが表示されます。

- 複数の関数から呼び出される関数

スタック見積もりツールでは，複数の関数から呼び出される関数のスタック情報を一意としています。
したがって，該当関数のスタック情報を呼び出し元に応じて変化させることはできません。

例 本ウインドウのツリー表示エリアで func1 から呼び出される sub を選択してオープンしたスタックサイズ変更 ダイアログで各種設定を行った場合，func2 から呼び出される sub についても同様の情報が反映されます。

```
int     sub ( int i );
void    func1 ( void );
void    func2 ( void );

void main ( void ) {
    func1 ( );
    func2 ( );
}
int sub ( int i ) {
    i++;
    return ( i );
}

void func1 ( void ) {
    int ret, i = 0;
    ret = sub ( i );
}

void func2 ( void ) {
    int ret, i = 100;
    ret = sub ( i );
}
```

- C ソース内の ASM 文

C ソース内に ASM 文が記述されている際には，スタック見積もりツールが“W9432 : 不正なフォーマットがアセンブラ・ソース・モジュール・ファイル（*path name*）で見つかりました（*line number* 行）。ファイルを確認してください。”といったメッセージを出力する場合があります。
このような場合，“該当部を #if などを用いて無効化する”，または“該当部をコメントアウトする”といった対処を行ってください。

- 間接的な再帰関数の呼び出し

再帰経路が複数の関数から構成される際には，“スタック・サイズの計算”が正しく行われない場合があります。

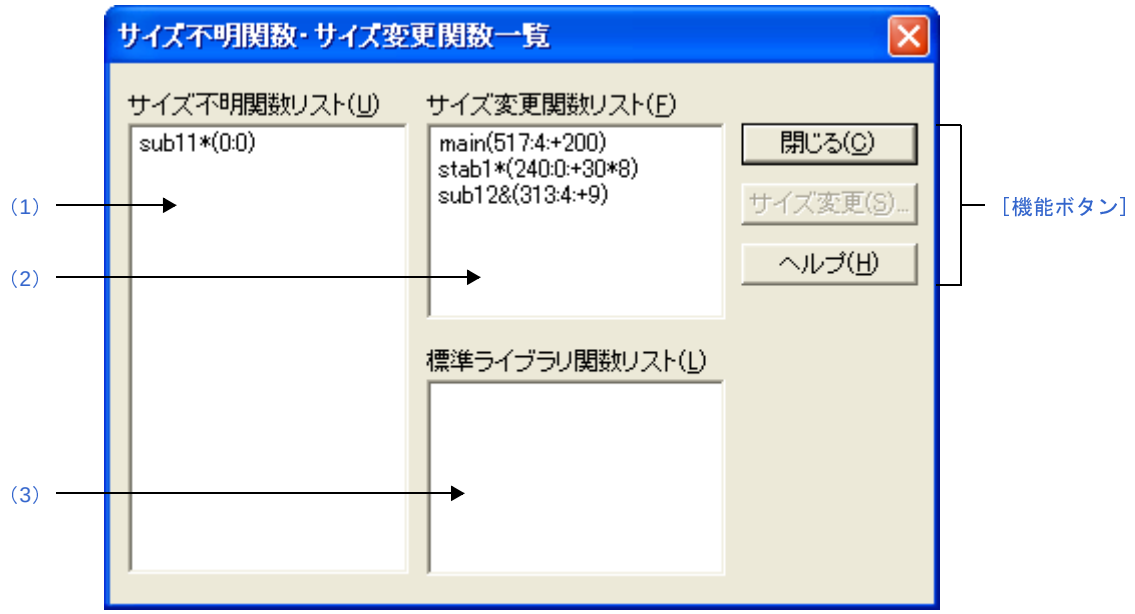
例 再帰関数 func_rec1 ／ func_rec2 の単体スタック・サイズを 8 バイトと仮定し，スタックサイズ変更 ダイアログにおいて，再帰関数 func_rec1 ／ func_rec2 の再帰回数を 3 回と設定した場合，func1 のスタック・サイズは“（8 ＋ 24）× 3”と正しく計算されますが， func2 のスタック・サイズについては“8 × 3”と func_rec1 の呼び出しが無視された計算が行われます。

```
void    func_rec1 ( int i );
void    func_rec2 ( int i );
void    func1 ( void );
void    func2 ( void );

void main ( void ) {
    func1 ( );
    func2 ( );
}
void func_rec1 ( int i ) {
    func_rec2 ( i );
}

void func_rec2 ( int i ) {
    if ( i ) {
        func_rec1 ( i - 1 );
    }
}

void func1 ( void ) {
    func_rec1 ( 2 );
}

void func2 ( void ) {
    func_rec2 ( 2 );
}
```

サイズ不明関数・サイズ変更関数一覧 ダイアログ

スタック見積もりツールがスタック情報を取得できていない関数，意図的に情報（加算サイズ，再帰回数，呼び出し関数）の変更が行われた関数，およびスタック見積もりツールが強制的に加算サイズの設定を行った関数を一覧表示します。

図 A―56　サイズ不明関数・サイズ変更関数一覧 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- Stack Usage Tracer ウインドウの［オプション］メニュー→［サイズ不明関数・サイズ変更関数一覧 ...］を選択

［各エリアの説明］

(1)［サイズ不明関数リスト］

スタック見積もりツールがスタック情報を取得できていない関数“不明関数”を一覧表示します。

なお，本エリアでは，不明関数を基本的に以下の形式で表示します。

関数名（スタック・サイズ：単体スタック・サイズ）

備考 1.　不明関数が“アセンブリ言語で記述された関数”の場合，シンボル名の先頭に付与されている“_”を削ったのち，“[]”で囲んだものを関数名として表示します。

2. 不明関数が“再帰関数”の場合，関数名の直後に“*”を表示します。
3. 不明関数が“関数ポインタを用いた間接呼び出しを含む関数”の場合，関数名の直後に“&”を表示します。
4. 不明関数が“スタティック関数”の場合，関数名の直前に“ファイル名 #”を表示します。

(2)［サイズ変更関数リスト］

スタックサイズ変更 ダイアログ，またはスタック・サイズ指定ファイルにより意図的に情報（加算サイズ，再帰回数，呼び出し関数）の変更が行われた関数“変更関数”を一覧表示します。

なお，本エリアでは，変更関数を基本的に以下の形式で表示します。

関数名（スタック・サイズ：単体スタック・サイズ：加算サイズ）

備考 1. 変更関数が“アセンブリ言語で記述された関数”の場合，シンボル名の先頭に付与されている“_”を削ったのち，“[]”で囲んだものを関数名として表示します。
2. 変更関数が“再帰関数”の場合，関数名の直後に“*”を表示します。
3. 変更関数が“関数ポインタを用いた間接呼び出しを含む関数”の場合，関数名の直後に“&”を表示します。
4. 変更関数が“スタティック関数”の場合，関数名の直前に“ファイル名 #”を表示します。
5. スタックサイズ変更 ダイアログにおいて，“呼び出し関数の追加”のみが行われた関数については，本エリアの表示内容が以下のようになります。

   関数名（スタック・サイズ：単体スタック・サイズ）

(3)［標準ライブラリ関数リスト］

単体スタック・サイズが不明な関数のうち，スタック見積もりツールが強制的に加算サイズの設定を行ったライブラリ関数“自動設定関数”を一覧表示します。

なお，本エリアでは，自動設定関数を基本的に以下の形式で表示します。

関数名（スタック・サイズ：？：加算サイズ）

備考 1. シンボル名の先頭に付与されている“_”を削ったのち，“[]”で囲んだものを関数名として表示します。
2. スタック見積もりツールが保有するデータ・ベースの中から該当ライブラリ関数に適切なスタック・サイズを“加算サイズ”として設定しています。

［機能ボタン］

ボタン	機能
閉じる	本ダイアログをクローズします。
サイズ変更 ...	［サイズ不明関数リスト］／［サイズ変更関数リスト］／［標準ライブラリ関数リスト］で選択した関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するためのスタックサイズ変更 ダイアログをオープンします。
ヘルプ	本ダイアログのヘルプを表示します。

スタックサイズ変更 ダイアログ

選択関数に対する情報（加算サイズ，再帰回数，呼び出し関数）を変更するダイアログです。

図 A—57　スタックサイズ変更 ダイアログ

ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- Stack Usage Tracer ウインドウのツリー表示エリア／リスト表示エリアにおいて，関数を選択したのち，［オプション］メニュー→［スタックサイズ変更 ...］を選択
- Stack Usage Tracer ウインドウのツリー表示エリア／リスト表示エリアにおいて，関数を選択したのち，ツールバー→  ボタンをクリック
- Stack Usage Tracer ウインドウのツリー表示エリア／リスト表示エリアにおいて，関数を選択したのち，コンテキスト・メニューから［スタックサイズ変更 ...］を選択

- サイズ不明関数・サイズ変更関数一覧 ダイアログの［サイズ不明関数リスト］／［サイズ変更関数リスト］／［標準ライブラリ関数リスト］において，関数を選択したのち，［サイズ変更 ...］ボタンをクリック

［各エリアの説明］

(1)［関数名］

選択関数の関数名を表示します。

備考 1. 選択関数が“アセンブリ言語で記述された関数”，または“ライブラリ関数”の場合，シンボル名の先頭に付与されている“_”を削ったのち，“[]”で囲んだものを関数名として表示します。

2. 選択関数が“再帰関数”の場合，関数名の直後に“*”を表示します。

3. 選択関数が“関数ポインタを用いた間接呼び出しを含む関数”の場合，関数名の直後に“&”を表示します。

4. 選択関数が“スタティック関数”の場合，関数名の直前に“ファイル名 #”を表示します。

(2)［単体スタックサイズ］

選択関数の単体スタック・サイズ（呼び出し関数のスタック・サイズを含まない，単位：バイト）を表示します。

備考 単体スタック・サイズが不明な場合は“?”を，限界値を越えている場合は“SIZEOVER”を表示します。

(3)［加算サイズ］

選択関数の単体スタック・サイズに対して強制的に加算する値（単位：バイト）を 10 進数，または“0x”／“0X”で始まる 16 進数で指定します。

(4)［再帰回数］

選択関数の再帰回数を 10 進数，または“0x”／“0X”で始まる 16 進数で指定します。

備考 選択関数が“再帰関数以外”の場合，本項目はグレー表記となります。

(5)［呼び出し関数リスト（関数ポインタ向け）］エリア

(a)［呼び出し関数一覧］

選択関数からの呼び出し関数（関数ポインタなどを用いて間接的に呼び出される関数）を一覧表示します。

なお，本エリアでは，呼び出し関数を基本的に以下の形式で表示します。

関数名（スタック・サイズ：単体スタック・サイズ：加算サイズ）

備考 1. 呼び出し関数が“アセンブリ言語で記述された関数”，または“ライブラリ関数”の場合，シンボル名の先頭に付与されている“_”を削ったのち，“[]”で囲んだものを関数名として表示します。

2. 呼び出し関数が“再帰関数”の場合，関数名の直後に“*”を表示します。

3. 呼び出し関数が“関数ポインタを用いた間接呼び出しを含む関数”の場合，関数名の直後に“&”を表示します。

4. 呼び出し関数が“スタティック関数”の場合，関数名の直前に“ファイル名 #”を表示します。

5. ［追加］ボタンのクリックにより，［関数一覧］から意図的に追加された関数については，関数名の直前に“+”を表示します。

（b）［関数一覧］

選択関数からの呼び出し関数として追加可能な関数を一覧表示します。

なお，本エリアでは，追加可能な関数を基本的に以下の形式で表示します。

関数名（スタック・サイズ：単体スタック・サイズ：加算サイズ）

備考 1. 追加可能な関数が“アセンブリ言語で記述された関数”，または“ライブラリ関数”の場合，シンボル名の先頭に付与されている“_”を削ったのち，“[]”で囲んだものを関数名として表示します。

2. 追加可能な関数が“再帰関数”の場合，関数名の直後に“*”を表示します。

3. 追加可能な関数が“関数ポインタを用いた間接呼び出しを含む関数”の場合，関数名の直後に“&”を表示します。

4. 追加可能な関数が“スタティック関数”の場合，関数名の直前に“ファイル名 #”を表示します。

（c）ボタン・エリア

追加	［関数一覧］で選択した関数を［呼び出し関数一覧］に追加します。 ［関数一覧］で関数が未選択の場合，本ボタンはグレー表記となります。
削除	［呼び出し関数一覧］で選択した関数を［呼び出し関数一覧］から削除します。 ［呼び出し関数一覧］で関数が未選択の場合，本ボタンはグレー表記となります。

備考 ［呼び出し関数一覧］から削除可能な関数は，関数名の直前に“+”が付与されているもの（［追加］ボタンのクリックにより，［関数一覧］から意図的に追加された関数）に限られます。

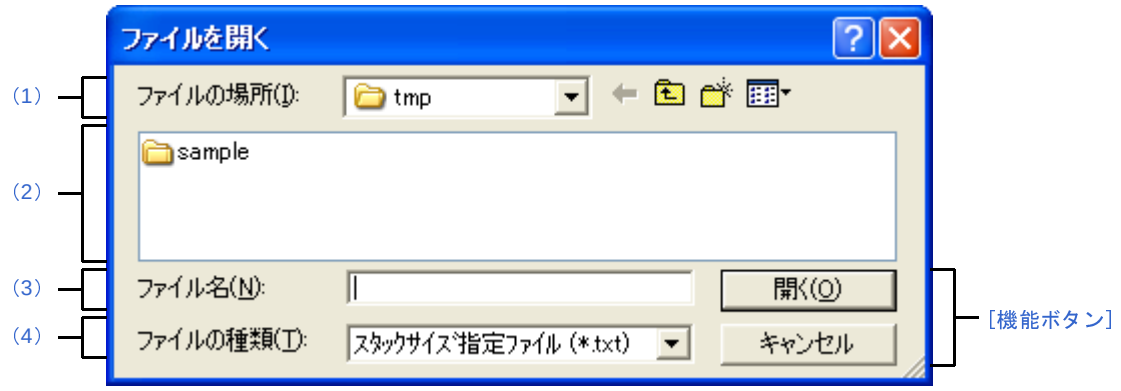
［機能ボタン］

ボタン	機能
OK	設定内容を Stack Usage Tracer ウインドウに反映／プロジェクト・ファイル（*.prj）に保存したのち，本ダイアログをクローズします。
キャンセル	設定内容を無効とし，本ダイアログをクローズします。
初期化	選択関数の情報（加算サイズ，再帰回数，呼び出し関数）を初期状態に戻します。 選択関数の情報が初期状態から何ら変更されていない場合，本ボタンはグレー表記となります。
ヘルプ	本ダイアログのヘルプを表示します。

ファイルを開く ダイアログ

既存のスタック・サイズ指定ファイルを開きます。

図 A—58　ファイルを開く ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- Stack Usage Tracer ウインドウの［ファイル］メニュー→［スタックサイズ指定ファイルを開く ...］を選択

［各エリアの説明］

(1)［ファイルの場所］エリア

開きたいスタック・サイズ指定ファイルが存在するフォルダを選択します。

(2) ファイルの一覧エリア

［ファイルの場所］エリア，および［ファイルの種類］エリアで選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］エリア

開くスタック・サイズ指定ファイルのファイル名を指定します。

(4)［ファイルの種類］エリア

開くファイルの種類（ファイル・タイプ）を選択します。

スタックサイズ指定ファイル（*.txt）	テキスト形式

［機能ボタン］

ボタン	機能
開く	指定されたファイルを開きます。
キャンセル	設定内容を無効とし，本ダイアログをクローズします。

付録 B　コマンド・リファレンス

ここでは，ビルド・ツールに含まれる各コマンドの仕様について，詳細を説明します。

B. 1　C コンパイラ

C コンパイラ（ca850）は，C ソース・ファイルに記述された C 言語のソース・プログラムから，リロケータブルなオブジェクト・ファイルや，ターゲット・システムで実行可能なオブジェクト・ファイルを生成します。

つまり，C コンパイラは，パッケージに含まれているモジュールのドライバとしての役目をし，マクロ展開やコメント処理，中間言語ファイルのマージ，最適化，アセンブラ・ソース・プログラムの生成から機械語命令への変換，オブジェクト・ファイルのリンクといった操作ができます。

C コンパイラは，次の順序で処理を行います。

ただし，「図 B—1　C コンパイラにおける動作の流れ」に示すように，最適化レベル指定によって処理の流れが若干異なります。

（1）フロントエンド（cafe）

C ソース・プログラムに対するマクロ展開，コメントの処理を行ったのち，中間言語プログラムに変換します。

（2）プリオプティマイザ（popt）

中間言語プログラム中の関数の並び替えを行います。

また，コマンド・ラインからの起動時に，マージ・オプション（-Om）を指定した場合，複数の中間言語プログラムを 1 つにマージします。

なお，より高度な最適化（実行速度優先）を指定した場合，さらに，中間言語プログラム内の関数をインライン展開します。

（3）広域最適化部（opt）

中間言語プログラムを最適化します。

（4）コード生成部（cgen）

中間言語プログラムをアセンブラ・ソース・プログラムに変換します。

（5）機種依存最適化部（impr）

アセンブラ・ソース・プログラムを最適化します。

（6）アセンブラ（as850）

アセンブラ・ソース・プログラムを機械語命令に変換し，リロケータブルなオブジェクト・ファイルを生成します。

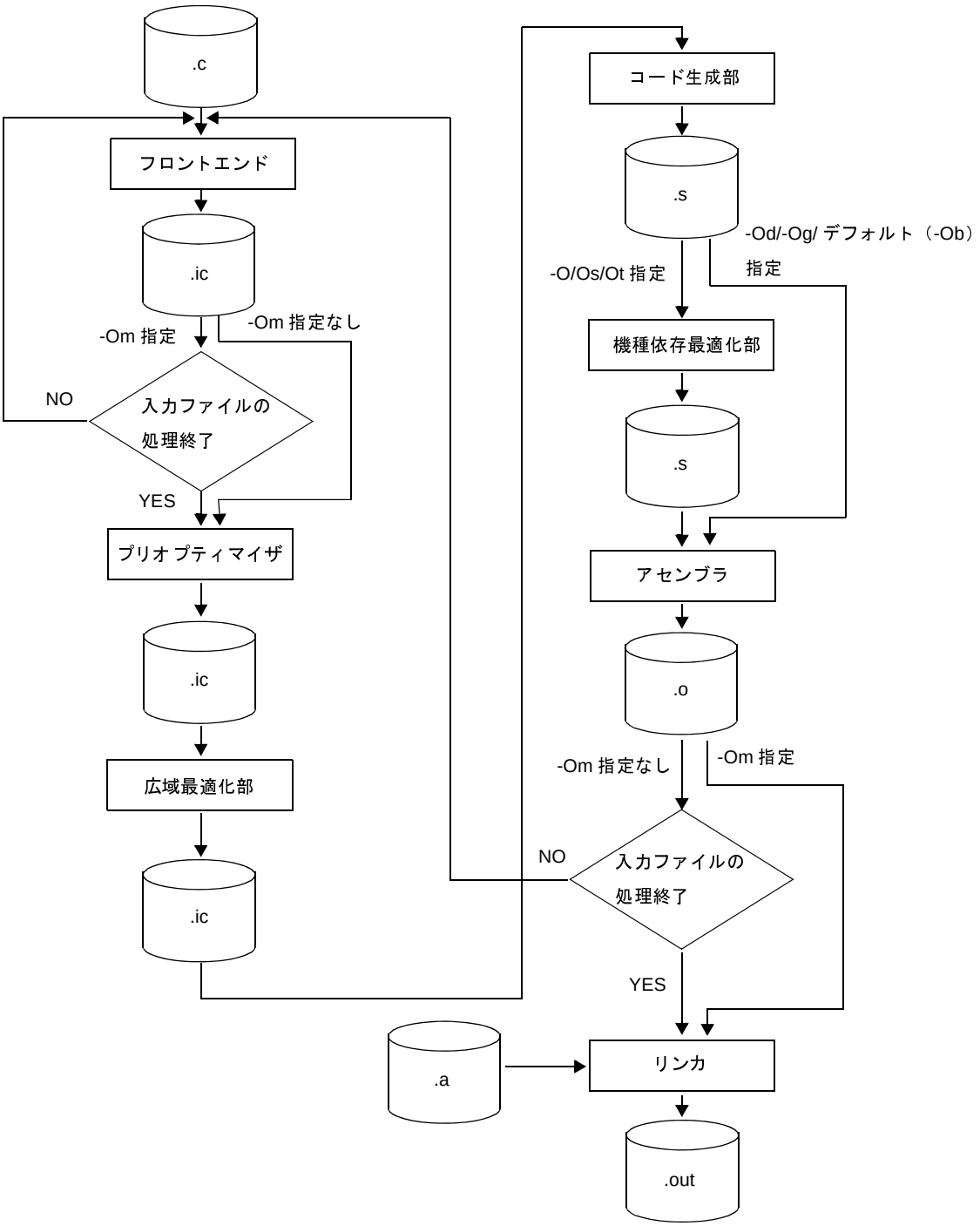
（7）リンカ（ld850）

リロケータブルなオブジェクト・ファイルをリンクし，実行可能なオブジェクト・ファイルを生成します。

広域最適化部，および機種依存最適化部は，最適化オプションが指定された場合にのみ起動されます。

なお，（1）～（5）のモジュールは，C コンパイラから起動されることを想定しています。したがって，これらを単独で起動した場合，動作は保証されません。

図 B—1　C コンパイラにおける動作の流れ



B. 1. 1　入出力ファイル

C コンパイラでは，次のファイルを入力ファイル，または出力ファイルとして指定できます。

file.c	C ソース・ファイル（.c ファイルと呼ばれる）
file.ic	中間言語ファイル（.ic ファイルと呼ばれる）
file.s	アセンブラ・ソース・ファイル（.s ファイルと呼ばれる）
file.o	オブジェクト・ファイル（.o ファイルと呼ばれる）
file.a	アーカイブ・ファイル（.a ファイルと呼ばれる）

- .s ファイルは，そのままアセンブラに渡されます（アセンブリ言語で直接記述されたソース・プログラムに対しては，機種依存最適化部を起動しません）。
- .a ファイルや .o ファイルなど，.c ファイル，.ic ファイル，および .s ファイル以外のファイルは，すべてそのままリンカに渡されます。

なお，入力ファイル名は Windows で認められるものであれば指定できますが，‘@’はコマンド・オプションと判断されるため‘@’をファイル名の先頭に使用できません。
また，ファイルの漢字コードが EUC の場合，ファイル名，フォルダ名に日本語は使用できません。

B. 1. 2　実行オブジェクト

C コンパイラは，アセンブラやリンカも起動するため，C ソース・ファイルを読み込んで，実行可能なオブジェクト・ファイルの生成まで一度に行うことができます。

また，オプション（-S）の指定により，アセンブラやリンカを起動する手前で処理を止め，コンパイラのコード出力やリロケータブルなオブジェクト・ファイルを生成することもできます（操作方法の詳細については，「B. 1. 3 操作方法」を参照してください）。

各コマンドのコマンド・ラインからの起動例を，次に示します（オプションの詳細については，「B. 1. 4　オプション」を参照してください）。

（1）C コンパイラからすべて行う場合

```
C:￥>ca850 -cpu 3201 file.c obj.o
```

デバイスに“-cpu 3201”（V850ES/SA2）を指定し，*file*.c と *obj*.o を読み込み，実行可能なオブジェクト・ファイル a.out を作成します。このとき，スタート・アップ・モジュールとして crtE.o をリンクし，標準ライブラリ libc.a と libm.a を参照します。

```
C:￥>ca850 -cpu 3201 -R org_crt.o file.c obj.o
```

file.c と *obj*.o を読み込み，実行可能なオブジェクト・ファイル a.out を作成します。このとき，スタート・アップ・モジュールとして *org_crt*.o をリンクし，標準ライブラリ libc.a と libm.a を参照します。

（2）C コンパイラからアセンブラまでを起動し，リンカは単独で起動する場合

```
C:¥>ca850 -cpu 3201 -c file.c asm.s
```

file.c，*asm*.s を読み込み，リロケータブルなオブジェクト・ファイル *file*.o，*asm*.o を作成します。

```
C:¥>ld850 -cpu 3201 org_crt.o file.o asm.o obj.o -lc
```

org_crt.o，*file*.o，*asm*.o，*obj*.o をリンクし，実行可能なオブジェクト・ファイル a.out を作成します。このとき標準ライブラリ libc.a を参照します。

（3）C コンパイラ，アセンブラ，リンカともに単独で起動する場合

```
C:¥>ca850 -cpu 3201 -c file.c
```

file.c を読み込み，リロケータブルなオブジェクト・ファイル *file*.o を作成します。

```
C:¥>as850 -cpu 3201 asm.s
```

asm.s を読み込み，リロケータブルなオブジェクト・ファイル *asm*.o を作成します。

```
C:¥>ld850 org_crt.o file.o asm.o -lc
```

org_crt.o，*file*.o，*asm*.o をリンクし，実行可能なオブジェクト・ファイル a.out を作成します。このとき標準ライブラリ libc.a を参照します。

B. 1. 3　操作方法

ここでは，C コンパイラの操作方法について説明します。

（1）コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:¥>ca850 ［オプション］ ... ファイル名 ［ファイル名，またはオプション］...
      ［ ］:    ［ ］内は省略できます。
      ... :    直前の［ ］内のパターンの繰り返しができます。
```

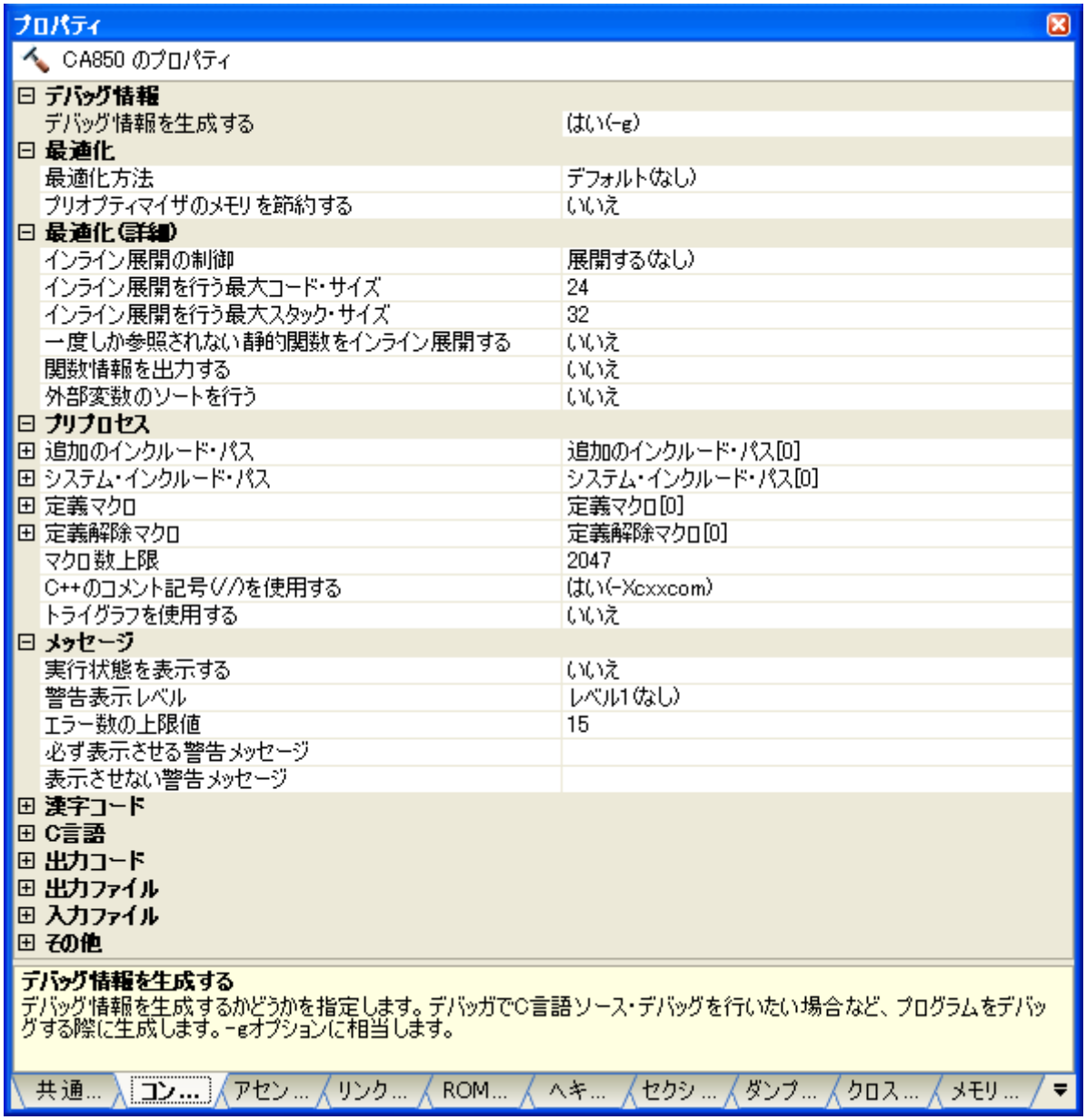
（2）CubeSuite+ でのオプション設定

CubeSuite+ からコンパイル・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネル上において，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［コンパイル・オプション］タブを選択します。

タブ上で必要なプロパティを設定することにより，各コンパイル・オプションを設定することができます。

図 B—2　プロパティ パネル：［コンパイル・オプション］タブ



B. 1. 4　オプション

ここでは，コンパイル・オプションについて説明します。

注意　コマンド・ラインからの起動時に「表 B—1　コンパイル・オプション」にないオプションが与えられた場合，それらはリンカのオプションとみなし，リンカを起動する際に渡されます。

コンパイル・オプションの分類と説明を示します。

表 B—1　コンパイル・オプション

分類	オプション	説明
バージョン／ヘルプ表示／動作状態	-V	バージョン情報を標準エラー出力に出力します。
	-help	オプションの説明を標準エラー出力に出力します。
	-v	コンパイル状況の詳細を標準エラー出力に出力します。
出力ファイル指定	-Fic	中間言語ファイルの保存先を指定します。
	-Fo	オブジェクト・ファイルの保存先を指定します。
	-Fs	アセンブリ言語ファイルの保存先を指定します。
	-Fv	アセンブル・リストの保存先を指定します。
	-o	出力ファイルを指定します。
	-temp	作業用フォルダを指定します。
ソース・デバッガ制御	-Xno_word_bitop	ld.w ／ ld.h，st.w ／ st.h 命令を 1 ビット操作命令へ置き換える動作を禁止します。
	-g	ソース・デバッガ用のシンボル情報を出力します。
デバイス指定	-X256M	メモリ空間を 256M バイトとして扱います。
	-Xbpc	プログラマブル周辺 I/O レジスタの上位アドレスを設定します。
	-cn	V850 コア共通のマジックナンバを埋め込みます。
	-cnv850e	V850Ex コア共通のマジックナンバを埋め込みます。
	-cnv850e2	V850E2 コア共通のマジックナンバを埋め込みます。
	-cpu	ターゲット・デバイスを指定します。
	-devpath	デバイス・ファイルの検索フォルダを指定します。
コンパイラ制御指定	-S	アセンブラ以降のモジュールを実行せず，アセンブラ・ソース・ファイルを出力します。
	-a	アセンブル・リストを出力します。
	-c	リンカまで起動せず，オブジェクト・ファイルの出力まで行います。
	-m	フロントエンドのみ実行し，.ic ファイルを生成して終了します。
ROM 化制御	-Xr	ROM 化用のオブジェクトを作成する場合に必要なオプションです。

分類	オプション	説明
プリプロセッサ処理設定	-C	前処理の出力に，ソース・プログラムのコメントも含めます。
	-D	C ソース・プログラムの前に #define が記述されたものとみなします。
	-E	C ソース・プログラムに対し前処理のみ実行し，結果を標準出力に出力します。
	-I	C ソース・プログラムのヘッダ・ファイルの検索フォルダを指定します。
	-P	C ソース・プログラムに対し前処理のみ実行して，結果をファイルに出力します。
	-U	C ソース・プログラムの前に #undef が記述されたものとみなします。
	-Wa,-D	アセンブラ・ソースの前に .set が記述されたものとみなします。
	-Wa,-I,	アセンブラ・ソース・ファイルのヘッダ・ファイルの検索フォルダを指定します。
	-Xcxxcom	通常のコメントのほかに，“//”から行末までをコメントとして扱います。
	-Xd	自動変数ではない変数のアドレス，または関数のアドレスを用いたポインタ型外部変数の初期化に対し，警告メッセージを出力します。
	-Xm	マクロ識別子数の上限を指定します。
	-t	トライグラフ系列の置換を行います。
コンパイル時のメモリ節約	-Wp,-D	コンパイル時のプリオプティマイザのメモリ使用量を減らします。
	-Wi,-D	コンパイル時の機種依存最適化のメモリ使用量を減らします。
エラー出力指定	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。
	-err_limit	エラー・メッセージの最大出力数を指定します。
拡張機能指定	-cc78k	78K マイクロコントローラ C コンパイラ CC78K*x* 互換の拡張機能を有効にします。
最適化	-Od	デバッグ優先オプションです。
	-Ob	デフォルト最適化オプションです。
	-Og	標準最適化オプションです。
	-O	高度な最適化オプションです。
	-Os	より高度な最適化（オブジェクト・サイズ優先）オプションです
	-Ot	より高度な最適化（実行速度優先）オプションです。
ターゲット・コード最適化	-Wi,-O4	データ・フロー解析を厳密に行い，最も強い最適化を行います。
	-Wi,-P	分岐先ラベルを整列する最適化を抑止します。
ファイル・マージ	-Om	複数ファイル同時指定時に，ファイルのマージを行います。

分類	オプション	説明
インライン展開最適化制御	-Wp,-G	インライン展開対象関数のスタック・サイズを，中間言語での大きさに制限し，大きいものはインライン展開しません。
	-Wp,-N	インライン展開対象関数の中間言語サイズを制限し，大きいものはインライン展開しません。
	-Wp,-S	一度しか参照されない静的な関数を無条件にインライン展開します。
	-Wp,-I	関数の情報を標準出力に出力，またはファイルに追加出力します。
	-Wp,-inline	#pragma inline 指定した関数のみインライン展開します。
	-Wp,-no_inline	#pragma inline 指定した関数を含む，すべての関数のインライン展開を抑止します。
	-Wp,-r	関数をエントリ関数として，そこから呼び出された関数のうち，展開後の不要な関数を削除します。
ループ展開最適化制御	-Wo,-Ol	for，while などで，ループを指定回展開します。
	-Wo,-Xlo	ループ展開数を指定した回数で固定してループ展開します。
strcpy，strcmp 展開	-Xi	配列，および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開します。
外部変数ソート	-Wo,-Op	外部変数をアライメントの大きい順に並び替えます。
分岐命令制御	-Wo,-XFo	分岐命令をコード・サイズ優先で並べてコードを出力します。
レジスタ使用制御	-r	指定した外部変数を指定レジスタに割り付けます。
	-reg	C コンパイラが使用するレジスタを制限します。
	-Xmask_reg	マスク・レジスタ機能を使用します。
プロローグ／エピローグ処理制御	-Xpro_epi_runtime	関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にするかどうかを設定します。
変数配置制御	-G	指定バイト以下のデータを .sdata セクション，または .sbss セクションに配置します。
	-Xsconst	const 属性のデータ，文字列リテラルを .sconst セクションに配置します。
	-Xcre_sec_data	セクション・ファイル・ジェネレータで使用する変数の頻度情報ファイルを出力します。
	-Xcre_sec_data_only	
	-Xsec_file	C コンパイラ起動時にデータのセクション割り当てを指定するためのセクション・ファイル名を指定します。
型制御	-Xbitfield	型指定子の付かない単なる int 型のビット・フィールドに対し，符号付きとするか符号なしとするかを指定します。
	-Xchar	型指定子の付かない単なる char 型に対し，符号付きとするか符号なしとするかを指定します。
	-Xenum_type	列挙型に対し，どの整数型と整合するかを指定します。
switch-case 文出力コード制御	-Xcase	switch 文のコード出力方式を指定します。
	-Xword_switch	switch 文の case ラベルに対する分岐テーブルを，1 ラベルあたり 4 バイトで生成します。

分類	オプション	説明
構造体パッキング制御	-Xbyte	構造体の間接アドレス・アクセスをバイト単位でアクセスします。
	-Xpack	構造体メンバのアライメントを指定します。
far jump 出力制御	-Xfar_jump	指定された関数への分岐に対して，jmp 命令を使用します。
	-Xj	C 言語で定義された通常の割り込み関数に対し，jmp 命令を用います。
コメント出力	-Xc	出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力します。
ANSI 規約	-Xe	16 ビット・データ以下の整数に対し mulh，divh 命令を利用せず，ランタイム・ライブラリを利用します。
	-Xdefvar	変数の仮定義を定義として扱います。
	-ansi	C コンパイラの処理を厳密に ANSI 規格にあわせ，規格に反する記述に対してエラーや警告メッセージを出力します。
日本語文字列制御	-Xk	入力ファイル中の日本語のコメント，文字列に対し，使用する文字コードを指定します。
	-Xkt	日本語文字列を，指定したコードに変換して出力します。
ライブラリ指定	-L	ライブラリの検索フォルダを指定します。
	-R	リンカまで起動する場合，使用するスタート・アップ・モジュールを指定します。
	-l	リンカで参照する，アーカイブ・ファイルを指定します。
警告メッセージ制御	-w	警告メッセージのレベル，出力，抑制を指定します。
	-won	指定した番号の警告メッセージを出力するようにします。
	-woff	指定した番号の警告メッセージを抑制します。
コマンド・ファイル指定	@	指定ファイルをコマンド・ファイルとして扱います。
CPU 不具合パッチ	-Xv850patch	CPU の障害に対応したコードを出力するため，C コンパイラが出力するアセンブラ・ソース・ファイルに対し，アセンブラに -p オプション指定を指示します。
各モジュール	-W	モジュールに対するオプションを指定します。
その他	+Oc	強力な最適化を行います。

表 B―2　オプション説明でのマーク

【V850E2】	V850E2 コア専用のオプション
【V850E】	V850Ex コア専用のオプション
【78K 互換】	78K マイクロコントローラ C コンパイラ CC78K*x* 互換オプション

バージョン／ヘルプ表示／動作状態

バージョン／ヘルプ表示／動作状態オプションには，次のものがあります。

- -V
- -help
- -v

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- C コンパイラのバージョン情報を標準エラー出力に出力します。

[使用例]

- C コンパイラのバージョン情報を標準エラー出力に出力します。

```
C:￥>ca850 -V
```

-help

[記述形式]

```
-help
```

- 省略時解釈

  なし

[機能説明]

- オプションの説明を標準エラー出力に出力します。

[使用例]

- オプションの説明を標準エラー出力に出力します。

```
C:¥>ca850 -help
```

-v

[記述形式]

```
-v
```

- 省略時解釈

  なし

[機能説明]

- コンパイル状況の詳細を標準エラー出力に出力します。

[使用例]

- コンパイル状況の詳細を標準エラー出力に出力します。

```
C:¥>ca850 -v prime.c
```

出力ファイル指定

出力ファイル指定オプションには，次のものがあります。

- -Fic
- -Fo
- -Fs
- -Fv
- -o
- -temp

-Fic

[記述形式]

```
-Fic[=outfile]
```

- 省略時解釈

  なし

[機能説明]

- コンパイル途中に生成される中間言語ファイルの保存先を指定します。

（1）*outfile* にファイル名を指定した場合

指定したファイル名でカレント・フォルダに *outfile* を保存します。

なお，*outfile* の拡張子は，.ic に限られます。

（2）*outfile* にフォルダを指定した場合

指定フォルダに .c を .ic で置き換えたファイル名で保存します。

（3）=*outfile* を省略した場合

カレント・フォルダに .c を .ic で置き換えたファイル名で保存します。

（4）出力ファイルが複数の場合

outfile に指定したフォルダを作成し，.c を .ic で置き換えたそれぞれのファイル名で保存します。

［使用例］

- 中間言語ファイルをフォルダ D:￥sample にファイル名 main.ic で保存します。

```
C:￥>ca850 -cpu f3719 -Fic=D:￥sample main.c
```

-Fo

[記述形式]

```
-Fo[=outfile]
```

- 省略時解釈

カレント・フォルダに .c，または .s を .o で置き換えたファイル名で保存します。

[機能説明]

- コンパイル途中に生成されるオブジェクト・ファイルの保存先を指定します。

(1) *outfile* にファイル名を指定した場合

指定ファイル名でカレント・フォルダに *outfile* を保存します。

(2) *outfile* にフォルダを指定した場合

指定フォルダに .c，または .s，または .ic を .o で置き換えたファイル名で保存します。

(3) *=outfile* を省略した場合

カレント・フォルダに .c，または .s，または .ic を .o で置き換えたファイル名で保存します。

(4) 出力ファイルが複数の場合

outfile に指定したフォルダを作成し，.c，または .s，または .ic を .o で置き換えたそれぞれのファイル名で保存します。

[使用例]

- オブジェクト・ファイルをファイル名 sample.o で保存します。

```
C:¥>ca850 -cpu f3719 -Fo=sample.o main.c
```

-Fs

[記述形式]

```
-Fs[=outfile]
```

- 省略時解釈

  なし

[機能説明]

- コンパイル途中に生成されるアセンブリ言語ファイルの保存先を指定します。

(1) *outfile* にファイル名を指定した場合

指定ファイル名でカレント・フォルダに *outfile* を保存します。

(2) *outfile* にフォルダを指定した場合

指定フォルダに .c，または .ic を .s で置き換えたファイル名で保存します。

(3) *=outfile* を省略した場合

カレント・フォルダに .c，または .ic を .s で置き換えたファイル名で保存します。

(4) 出力ファイルが複数の場合

outfile に指定したフォルダを作成し，.c，または .ic を .s で置き換えたそれぞれのファイル名で保存します。

[使用例]

- アセンブリ言語ファイルをフォルダ D:¥sample にファイル名 main.s で保存します。

```
C:¥>ca850 -cpu f3719 -Fs=D:¥sample main.c
```

-Fv

[記述形式]

```
-Fv[=outfile]
```

- 省略時解釈

　なし

[機能説明]

- コンパイル途中に生成されるアセンブル・リストの保存先を指定します。

（1）*outfile* にファイル名を指定した場合

　指定ファイル名でカレント・フォルダに *outfile* を保存します。

（2）*outfile* にフォルダを指定した場合

　指定フォルダに .c，または .s，または .ic を .v で置き換えたファイル名で保存します。

（3）=*outfile* を省略した場合

　カレント・フォルダに .c，または .s，または .ic を .v で置き換えたファイル名で保存します。

（4）出力ファイルが複数の場合

　outfile に指定したフォルダを作成し，.c，または .s，または .ic を .v で置き換えたそれぞれのファイル名で保存します。

- このオプションと -a オプションを指定しない場合は，アセンブル・リストは生成されません。

[使用例]

- アセンブル・リストをファイル名 sample.v で保存します。

```
C:¥>ca850 -cpu f3719 -Fv=sample.v main.c
```

-o

[記述形式]

```
-o outfile
```

- 省略時解釈

  カレント・フォルダに保存します。

[機能説明]

- 出力ファイルを *outfile* で指定します。

(1) -S オプションと同時に指定した場合

outfile は，アセンブリ言語ファイル（.s）の指定となります。

(2) -c オプションと同時に指定した場合

outfile は，リロケータブル・オブジェクト・ファイル（.o）の指定となります。

(3) -m オプションと同時に指定した場合

outfile は，フロントエンド出力ファイル（.ic）の指定となります。

(4) 上記以外の場合

outfile は，実行可能なオブジェクト・ファイル（.out）の指定となります。デフォルトは a.out です。

(5) 出力ファイルが複数の場合

エラーとなります。

- コンパイラ制御オプション -S，-c，-m を指定しコンパイルを途中で止めた場合にもこのオプションは有効となります。

[使用例]

- 実行可能なオブジェクト・ファイルをファイル名 sample.out で出力します。

```
C:￥>ca850 -cpu f3719 -o sample.out main.c
```

-temp

[記述形式]

```
-temp=dir
```

- 省略時解釈

  テンポラリ・ファイルは，環境変数 TEMP で指定されたフォルダ，またはカレント・ドライブのルート・フォルダに生成されます。

[機能説明]

- 内部的に用いるテンポラリ・ファイルを生成する作業用フォルダを指定します。
- ハード・ディスクの容量不足などで，テンポラリ・ファイルを生成することができないためのエラーが発生した場合，このオプションで回避できます。

[使用例]

- フォルダ D:¥tmp をテンポラリ・ファイルを生成する作業用フォルダとします。

```
C:¥>ca850 -cpu f3719 -temp=D:¥tmp main.c
```

ソース・デバッガ制御

ソース・デバッガ制御オプションには，次のものがあります。

- -Xno_word_bitop
- -g

-Xno_word_bitop

[記述形式]

```
-Xno_word_bitop
```

- 省略時解釈

  ld.w ／ ld.h，st.w ／ st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作をします。

[機能説明]

- ld.w ／ ld.h，st.w ／ st.h 命令を 1 ビット操作命令（set1，clr1，tst1，not1）へ置き換える動作を禁止します。
- デバッグ時に変数の read ／ write イベントを設定する場合，1 ビット操作命令に置き換わっていると，イベントが発生しないことがあります。この場合，このオプションを指定すると，ld.w ／ ld.h，st.w ／ st.h 命令のままとなり，デバッグがしやすくなります。

[使用例]

- ld.w ／ ld.h，st.w ／ st.h 命令の 1 ビット操作命令（set1，clr1，tst1，not1）への置き換えを禁止します。

```
C:¥>ca850 -cpu f3719 -Xno_word_bitop main.c
```

-g

[記述形式]

```
-g
```

- 省略時解釈

  ソース・デバッガ用のシンボル情報を出力しません。

[機能説明]

- ソース・デバッガ用のシンボル情報を出力します。

  つまり，このオプションの指定により，C ソース・レベルでデバッグが可能となります。
- C コンパイラからアセンブラを起動する場合，このオプションの指定により，アセンブラの -g も指定したものとみなされます。これにより，デバッガでアセンブラ・ソース・レベルでデバッグが可能となります。

[使用例]

- ソース・デバッガ用のシンボル情報を出力し，C ソース・レベルでのデバッグを可能とします。

```
C:¥>ca850 -cpu f3719 -g main.c
```

デバイス指定

デバイス指定オプションには，次のものがあります。

- -X256M
- -Xbpc
- -cn
- -cnv850e
- -cnv850e2
- -cpu
- -devpath

-X256M

[記述形式]

```
-X256M
```

- 省略時解釈

  メモリ空間を 64M バイトとして扱い，アドレス解決します。

[機能説明]

【V850E】

- メモリ空間を 256M バイトとして扱います。
- 使用するチップセットにあわせて設定してください。V850Ex コアでは，物理アドレス空間が 256M バイト持つ場合が多く，64M を越えた 256M までの空間を使用するアプリケーションを作成する場合，このオプションを指定してください。

[使用例]

- メモリ空間を 256M バイトとします。

```
C:¥>ca850 -cpu f3719 -X256M main.c
```

-Xbpc

[記述形式]

```
-Xbpc=num
```

- 省略時解釈

  プログラマブル周辺 I/O レジスタの上位アドレスを 0 として扱います。

[機能説明]

- プログラマブル周辺 I/O レジスタの上位アドレスを設定します。
- *num* には，BPC レジスタの最上位ビットを除いたアドレス部分のみを指定してください。
- ターゲット・デバイスがプログラマブル周辺 I/O レジスタ機能を持ち（V850E/IA1 など），変更可能なアドレス部分（=BPC レジスタに設定する値）を設定したい場合，アプリケーションのコンパイル（アセンブル）時に，値を確定させる必要があります。
- このオプションを指定すると，指定した値を使用してコンパイル（アセンブル）します。このオプション指定時には必ず値を指定してください。

  値には 2 進数，8 進数，10 進数，16 進数を使用できます。不正な値を指定した場合，および BPC レジスタに設定可能な範囲を越える値を指定した場合，警告メッセージが出力され，このオプションは無視されます。
- 設定する値はアプリケーション全体で 1 つです。ファイルごとのオプション設定で“-Xbpc”や“-bpc”を指定する場合，ファイル間で同じ値にしてください。

  ただし，プログラマブル周辺 I/O レジスタを使用しないファイルに対しては，このオプションを指定する必要はありません。
- プログラマブル周辺 I/O レジスタ機能を持たないターゲット・デバイスの場合，および V850 コア／ V850Ex コア／ V850E2 コア共通としてアセンブルする場合，このオプションを指定すると，警告メッセージが出力され，このオプションは無視されます。
- このオプションは，プログラマブル周辺 I/O レジスタのアドレスをコンパイル（アセンブル）時に確定するためのものであり，BPC レジスタに実際に値を反映させるものではありません。動作のためには，別途，スタート・アップ・モジュールなどで BPC レジスタに値を設定する必要があります。

  「CubeSuite+ V850 コーディング編」にスタート・アップ・ルーチンのサンプルが載っていますので，そちらを参照してください。また，パッケージに含まれるスタート・アップ・モジュールにも，サンプルが載っています（コメントアウトしてあります）。
- アセンブラでは，このオプションを指定するか，または省略していても，実際にはプログラマブル周辺 I/O レジスタを参照していた場合，予約セクションである .bpc セクションを出力します。

  このセクションは，リンク時のチェックのために使用されます。.bpc セクションは，情報用の特殊な予約セクションであり，メモリにロードされることはありません。したがって，通常のセクションのように，リンク・ディレクティブに記述する必要はありません。

［使用例］

- ターゲット・デバイスが V850E/IA1 の場合，下記オプション設定では，プログラマブル周辺 I/O レジスタ領域の先頭アドレスは，この値を 14 ビット左シフトした 0x48d0000 として扱われます。

```
C:￥>ca850 -cpu 3116 -Xbpc=0x1234 main.c
```

プログラマブル周辺 I/O レジスタの先頭アドレスの変更可能部分を“0x1234”とし，この機能の使用を許可するフラグ“0x8000”を設定する場合，次の記述をスタート・アップ・モジュールに記述します。

```
mov     0x9234,r10      -- 0x1234 | 0x8000 = 0x9234
st.h    r10, BPC
```

-cn

[記述形式]

```
-cn
```

- 省略時解釈

  なし

[機能説明]

- 生成するオブジェクトに V850 コア共通のマジックナンバを埋め込みます。

[使用例]

- オブジェクトに V850 コア共通のマジックナンバを埋め込みます。

```
C:¥>ca850 -cn -c main.c
```

-cnv850e

[記述形式]

```
-cnv850e
```

- 省略時解釈

  なし

[機能説明]

【V850E】

- 生成するオブジェクトに V850Ex コア共通のマジックナンバを埋め込みます。

[使用例]

- オブジェクトに V850Ex コア共通のマジックナンバを埋め込みます。

```
C:￥>ca850 -cnv850e -c main.c
```

-cnv850e2

[記述形式]

```
-cnv850e2
```

- 省略時解釈

  なし

[機能説明]

【V850E2】

- 生成するオブジェクトに V850E2 コア共通のマジックナンバを埋め込みます。

[使用例]

- オブジェクトに V850E2 コア共通のマジックナンバを埋め込みます。

```
C:￥>ca850 -cnv850e2 -c main.c
```

-cpu

[記述形式]

```
-cpu devicename
```

- 省略時解釈

  省略することはできません（-cn ／ -cnv850e ／ -cnv850e2 ／ #pragma cpu 指定時を除きます）。

[機能説明]

- ターゲット・デバイスを指定します注。

  注 “#pragma cpu *devicename*” と同じ機能です。
  -cpu オプション指定と #pragma 命令による指定の両方があり，指定の内容が異なっている場合，-cpu オプションによる指定内容が優先されます。

- このオプションを省略し，-cn ／ -cnv850e ／ -cnv850e2 オプション，または #pragma 指令による指定もない場合は，コンパイルは中止されます。

[使用例]

- ターゲット・デバイスとして，V850E を指定します。

```
C:¥>ca850 -cpu f3719 main.c
```

-devpath

[記述形式]

```
-devpath=dir
```

- 省略時解釈

  デバイス・ファイルを，標準フォルダから検索します。

[機能説明]

- デバイス・ファイルを，フォルダ *dir* から検索します。

[使用例]

- デバイス・ファイルをフォルダ D:￥dev から検索します。

```
C:￥>ca850 -cpu f3719  -devpath=D:￥dev main.c
```

コンパイラ制御指定

コンパイラ制御指定オプションには，次のものがあります。

- -S
- -a
- -c
- -m

-S

[記述形式]

```
-S
```

- 省略時解釈

  アセンブラ以降のフェーズも実行されます。

[機能説明]

- アセンブラ以降のモジュールを実行せず，生成されたアセンブラ・ソース・ファイルを出力します。
- 出力されるファイル名は，.c，または .ic を .s で置き換えたものになります。-o オプションを利用することで，出力ファイル名を指定できます（-o オプションの説明参照）。また，-Fs オプションでも出力ファイル名を指定できます。

[使用例]

- アセンブラ以降のモジュールを実行せず，アセンブラ・ソース・ファイル main.s を出力します。

```
C:¥>ca850 -cpu f3719 -S main.c
```

-a

[記述形式]

```
-a
```

- 省略時解釈

  アセンブル・リストを出力しません。

[機能説明]

- アセンブル・リストを出力します。ファイル名は .c，または .s，または .ic を .v に置き換えたものになります（「3. 1　アセンブラ」参照）。
- 最適化オプションで，“標準最適化（-Og）以上”を指定した場合，アセンブラによる最適化のための命令並び替えにより，アセンブル・リストの出力が，一部不正確になる場合があります。

[使用例]

- アセンブル・リスト main.v を出力します。

```
C:¥>ca850 -cpu f3719 -a main.c
```

-c

[記述形式]

```
-c
```

- 省略時解釈

  リンカまで起動します。

[機能説明]

- リンカまで起動せず，オブジェクト・ファイルの出力まで行います。
- ファイル名は，.c，または .s，または .ic を .o で置き換えたものになります。
- -o オプションを利用することで，出力ファイル名を指定できます（-o オプションの説明参照）。また，-Fo オプションでも出力ファイル名を指定できます。

[使用例]

- オブジェクト・ファイル main.o を出力します。

```
C:¥>ca850 -cpu f3719 -c main.c
```

-m

[記述形式]

```
-m
```

- 省略時解釈

  フロントエンド以降のモジュールも実行します。

[機能説明]

- フロントエンドのみ実行し，.ic ファイルを生成して終了します。

[使用例]

- フロントエンドのみ実行し，中間言語ファイル main.ic を出力します。

```
C:¥>ca850 -cpu f3719 -m main.c
```

ROM 化制御

ROM 化制御オプションには，次のものがあります。

- -Xr

-Xr

[記述形式]

```
-Xr
```

- 省略時解釈

  ROM 化情報を持たないオブジェクトを作成します。

[機能説明]

- ROM 化用のオブジェクトを作成する場合に必要なオプションです。
- コンパイラが行う処理は，次のようになります。

（1）先頭が“_rcopy”で始まる関数の第一引数のラベルが，オブジェクト内の .text セクションの終端を越える最初の（4 バイトの整列条件で整列された）アドレスを指すようにする。

（2）これにより，rompsec セクション用の領域確保コード（デフォルト名は rompcrt.o），および libr.a ファイルを，リンカでリンクするように指示。

- ROM 化オブジェクトの作成方法の詳細は「B. 4. 3　ROM 化用オブジェクトの作成」を参照してください。

[使用例]

- ROM 化情報を持つオブジェクト・ファイル a.out を出力します。

```
C:¥>ca850 -cpu f3719 -Xr main.c
```

プリプロセッサ処理設定

プリプロセッサ処理設定オプションには，次のものがあります。

- -C
- -D
- -E
- -I
- -P
- -U
- -Wa,-D
- -Wa,-I
- -Xcxxcom
- -Xd
- -Xm
- -t

-C

[記述形式]

```
-C
```

- 省略時解釈

  なし

[機能説明]

- C ソース・プログラムの前処理の出力に，ソース・プログラムのコメントも含めます。-E，または -P オプション指定時のみ有効です。

[使用例]

- 前処理の出力に，ソース・プログラムのコメントも含め，結果を標準出力に出力します。

```
C:￥>ca850 -cpu f3719 -C -E main.c
```

-D

[記述形式]

```
-Dname[=def]
```

- 省略時解釈

  なし

[機能説明]

- C ソース・プログラムの前に #define *name def* が記述されたものとみなします。
- *=def* の指定を省略した場合，*def* を 1 とみなします。256 個まで指定できます。

[使用例]

- C ソース・プログラムの前に #define sample 256 が記述されたものとします。

```
C:￥>ca850 -cpu f3719 -Dsample=256 main.c
```

-E

[記述形式]

```
-E
```

- 省略時解釈

  なし

[機能説明]

- C ソース・プログラムに対し前処理のみ実行し，結果を標準出力に出力します。
- 結果には，ソース・プログラムの行番号表示と，ファイル名表示を含みます。

[使用例]

- 前処理のみ実行し，結果を標準出力に出力します。

```
C:￥>ca850 -cpu f3719 -E main.c
```

-I

[記述形式]

```
-Idir
```

- 省略時解釈

C ソース・プログラムのヘッダ・ファイルを 標準フォルダからのみ検索します。
標準フォルダは，“インストール・フォルダ¥ CubeSuite+ ¥ CA850 ¥ V*x.xx* 注¥ inc850”フォルダです。

注 V*x.xx* は C コンパイラのバージョンです。

[機能説明]

- C ソース・プログラムのヘッダ・ファイルをフォルダ *dir*，標準フォルダの順で検索します。
100 個まで指定できます。
- #include " ヘッダ・ファイル名 " と記述された場合は，ソース・ファイルのあるフォルダを最初に検索します。

[使用例]

- C ソース・プログラムのヘッダ・ファイルをフォルダ D:¥ head，標準フォルダの順で検索します。

```
C:¥>ca850 -cpu f3719 -ID:¥head main.c
```

-P

[記述形式]

```
-P
```

- 省略時解釈

  なし

[機能説明]

- C ソース・プログラムに対し前処理のみ実行して，結果を C ソース・ファイル名の .c を .i に置き換えた名前のファイルに出力します。
- ソース・プログラムの行番号表示やファイル名表示は出力しません。

[使用例]

- 前処理のみ実行して，結果をファイル main.i に出力します。

```
C:￥>ca850 -cpu f3719 -P main.c
```

-U

[記述形式]

```
-Uname
```

- 省略時解釈

  なし

[機能説明]

- C ソース・プログラムの前に #undef *name* が記述されたものとみなします。

  256 個まで指定できます。

[使用例]

- C ソース・プログラムの前に #undef test が記述されたものとします。

```
C:￥>ca850 -cpu f3719 -Utest main.c
```

-Wa,-D

[記述形式]

```
-Wa,-Dname[=num]
```

- 省略時解釈

  なし

[機能説明]

- アセンブラ・ソースの前に .set *name*, *num* が記述されたものとみなします。
- “=*num*”の指定を省略した場合，*num* を 1 とみなします。

[使用例]

- 出力したアセンブラ・ソースの前に .set _sample, 256 が記述されたものとします。

```
C:￥>ca850 -cpu f3719 -Wa,-D_sample=256 main.c
```

-Wa,-I

[記述形式]

```
-Wa,-I,dir
```

- 省略時解釈

  アセンブラ・ソース・ファイルのヘッダ・ファイルを 標準フォルダからのみ検索します。

[機能説明]

- アセンブラ・ソース・ファイルのヘッダ・ファイルをフォルダ *dir*，標準フォルダの順で検索します。

  標準フォルダにもない場合は，アセンブラ・ソース・ファイルのあるフォルダ，C ソース・ファイルのあるフォルダの順で検索します。

[使用例]

- アセンブラ・ソース・ファイルのヘッダ・ファイルをフォルダ D:￥head，標準フォルダの順で検索します。

```
C:￥>ca850 -cpu f3719 -Wa,-I,D:￥head main.c
```

-Xcxxcom

[記述形式]

```
-Xcxxcom
```

- 省略時解釈

  なし

[機能説明]

- 通常のコメントのほかに，“//”から行末までをコメント（C++ のコメント・スタイル）として扱います。

[使用例]

- “//”から行末までをコメントとして扱います。

```
C:¥>ca850 -cpu f3719 -Xcxxcom main.c
```

-Xd

[記述形式]

```
-Xd
```

- 省略時解釈

  自動変数ではない変数のアドレス，または関数のアドレスを用いたポインタ型外部変数の初期化に対し，警告メッセージを出力しません。

[機能説明]

- 自動変数ではない変数のアドレス，または関数のアドレスを用いたポインタ型外部変数の初期化に対し，警告メッセージを出力します。

[使用例]

- 自動変数ではない変数のアドレス，または関数のアドレスを用いたポインタ型外部変数の初期化に対し，警告メッセージを出力します。

```
C:¥>ca850 -cpu f3719 -Xd main.c
```

-Xm

[記述形式]

```
-Xmnum
```

- 省略時解釈

  -Xm2047

[機能説明]

- マクロ識別子数の上限を指定します。*num* には，999999 までの 10 進数を指定します。
- このオプションはプリプロセッサで使用するバッファのサイズを大きくするものです。

  ただし，これによって何文字分のバッファが確保されるのかという具体的な数値を出すことはできません。

[使用例]

- マクロ識別子数の上限を 32000 とします。

```
C:￥>ca850 -cpu f3719 -Xm32000 main.c
```

-t

[記述形式]

```
-t
```

- 省略時解釈

  なし

[機能説明]

- トライグラフ系列の置換を行います。ANSI 規格で規定された，単一文字に置換される 3 文字（トライグラフ）系列です。

  詳細は ANSI 規格に関する文献を参照してください。

[使用例]

- トライグラフ系列の置換を行います。

```
C:￥>ca850 -cpu f3719 -t main.c
```

コンパイル時のメモリ節約

コンパイル時のメモリ節約オプションには，次のものがあります。

- -Wp,-D
- -Wi,-D

-Wp,-D

[記述形式]

```
-Wp,-D
```

- 省略時解釈

  なし

[機能説明]

- コンパイル時のプリオプティマイザのメモリ使用量を減らします。
- マシンのメモリが不足しコンパイルが正常に終了しないときにこのオプションを指定します。このオプションを指定するとコンパイル速度は低下します。

[使用例]

- コンパイル時のプリオプティマイザのメモリ使用量を減らします。

```
C:￥>ca850 -cpu f3719 -Wp,-D main.c
```

-Wi,-D

[記述形式]

```
-Wi,-D
```

- 省略時解釈

  なし

[機能説明]

- コンパイル時の機種依存最適化のメモリ使用量を減らします。
- マシンのメモリが不足しコンパイルが正常に終了しないときにこのオプションを指定します。
- このオプションを指定するとコンパイル速度は低下します。

[使用例]

- コンパイル時の機種依存最適化のメモリ使用量を減らします。

```
C:￥>ca850 -cpu f3719 -Wi,-D main.c
```

エラー出力指定

エラー出力指定オプションには，次のものがあります。

- +err_file
- -err_file
- -err_limit

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>ca850 -cpu f3719 +err_file=err main.c
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>ca850 -cpu f3719 -err_file=err main.c
```

-err_limit

[記述形式]

```
-err_limit=num
```

- 省略時解釈

  エラー・メッセージの最大出力数を 15 とします。

[機能説明]

- エラー・メッセージの最大出力数 *num* を指定します。
- *num* には，15 から 50 までの 10 進数を指定します。

[使用例]

- エラー・メッセージの最大出力数を 50 とします。

```
C:￥>ca850 -cpu f3719 -err_limit=50 main.c
```

拡張機能指定

拡張機能指定オプションには，次のものがあります。

- -cc78k

-cc78k

[記述形式]

```
-cc78k
```

- 省略時解釈

  78K マイクロコントローラ C コンパイラ CC78Kx 互換の拡張機能は無効になります。

[機能説明]

【78K 互換】

- 78K マイクロコントローラ C コンパイラ CC78K*x* 互換の拡張機能を有効にします。

[使用例]

- 78K マイクロコントローラ C コンパイラ CC78K*x* 互換の拡張機能を有効にします。

```
C:¥>ca850 -cpu f3719 -cc78k main.c
```

最適化

最適化オプションには，次のものがあります。

- -Od
- -Ob
- -Og
- -O
- -Os
- -Ot

-Od

[記述形式]

```
-Od
```

- 省略時解釈

  -Ob

[機能説明]

- デバッグ優先オプションです。
- ROM 容量や実行速度に着目せず，ソース・デバッグに着眼したコードを生成します。
- CA850 Ver.2.41 以前のデフォルト最適化に相当する機能です。

[使用例]

- デバッグ優先のソース・デバッグに着眼したコードを生成します。

```
C:￥>ca850 -cpu f3719 -Od main.c
```

-Ob

[記述形式]

```
-Ob
```

- 省略時解釈

  -Ob

[機能説明]

- デフォルト・オプションです。

  ソース・デバッグに着眼したコードを生成します。

- ソース・デバッグに影響のない範囲で最適化を行います。

[使用例]

- ソース・デバッグに影響のない範囲で最適化を行った，ソース・デバッグに着眼したコードを生成します。

```
C:￥>ca850 -cpu f3719 -Ob main.c
```

-Og

[記述形式]

```
-Og
```

- 省略時解釈

  -Ob

[機能説明]

- 標準最適化オプションです。

  適度な最適化を行います。
- ほとんどの場合で C ソース・デバッグが可能となる最適化を行います。
- 外部変数をレジスタに割り当てることから，実行速度，コード・サイズともにデフォルト・オプションより改善されます。

[使用例]

- 適度な最適化を行います。

```
C:￥>ca850 -cpu f3719 -Og main.c
```

-O

[記述形式]

```
-O
```

- 省略時解釈

  -Ob

[機能説明]

- 高度な最適化オプションです。

  ROM 容量に着目した最適化を行います。

[使用例]

- ROM 容量に着目した最適化を行います。

```
C:￥>ca850 -cpu f3719 -O main.c
```

-Os

[記述形式]

```
-Os
```

- 省略時解釈

  -Ob

[機能説明]

- より高度な最適化（オブジェクト・サイズ優先）オプションです。

  ROM 容量を最も重視した最大限の最適化を行います。

[使用例]

- ROM 容量を最も重視した最大限の最適化を行います。

```
C:￥>ca850 -cpu f3719 -Os main.c
```

-Ot

[記述形式]

```
-Ot
```

- 省略時解釈

  -Ob

[機能説明]

- より高度な最適化（実行速度優先）オプションです。

  ROM 容量よりも実行速度を最も重視した最大限の最適化を行います。

[使用例]

- 実行速度を最も重視した最大限の最適化を行います。

```
C:￥>ca850 -cpu f3719 -Ot main.c
```

ターゲット・コード最適化

ターゲット・コード最適化オプションには，次のものがあります。

- -Wi,-O4
- -Wi,-P

-Wi,-O4

[記述形式]

```
-Wi,-O4
```

- 省略時解釈

  なし

[機能説明]

- データ・フロー解析を厳密に行い，最も強い最適化を行います。
- 最適化オプション -O ／ -Os ／ -Ot を指定した上で，さらに強い最適化を行いたい場合に指定してください。
- この最適化では具体的に次のことを行います。
  - 分岐命令をまたいだレジスタの最適化
  - 絶対値演算の最適化
  - 分岐命令をまたいだ cmp 命令の最適化
  - 分岐命令をまたいだ復帰命令の最適化
- ソースによっては -O ／ -Os ／ -Ot の最適化結果と同じになることがあります。また，コンパイル時間は，-Os ／ -Ot 指定時より遅くなります。

[使用例]

- データ・フロー解析を厳密に行い，最も強い最適化を行います。

```
C:￥>ca850 -cpu f3719 -Os -Wi,-O4 main.c
```

-Wi,-P

[記述形式]

```
-Wi,-P
```

- 省略時解釈

  なし

[機能説明]

- 分岐先ラベルを整列する最適化を抑止します。
- 実行コードのサイズを小さくすることができます。
- このオプションは，より高度な最適化（実行速度優先）-Ot オプションを指定しているときに有効です。

[使用例]

- 実行速度優先の最適化を行っている場合に，分岐先ラベルを整列する最適化を抑止します。

```
C:￥>ca850 -cpu f3719 -Ot -Wi,-P main.c
```

ファイル・マージ

ファイル・マージ・オプションには，次のものがあります。

- -Om

-Om

[記述形式]

```
-Om
```

- 省略時解釈

  なし

[機能説明]

- 複数ファイル同時指定時に，ファイルのマージを行います。
- コンパイル時間は遅くなりますが，最適化オプション -O，-Os，および -Ot と同時に指定することで，インライン展開の適用範囲を広くすることができます。ただし，ソース・デバッグは難しくなります。

[使用例]

- 複数ファイル同時指定時に，ファイルのマージを行います。

```
C:¥>ca850 -cpu f3719 -Om -Os main.c sub.c
```

インライン展開最適化制御

インライン展開最適化制御オプションには，次のものがあります。

- -Wp,-G
- -Wp,-N
- -Wp,-S
- -Wp,-I
- -Wp,-inline
- -Wp,-no_inline
- -Wp,-r

-Wp,-G

[記述形式]

```
-Wp,-Gnum
```

- 省略時解釈

  -Wp,-G32

[機能説明]

- インライン展開対象関数のスタック・サイズを，中間言語での大きさ *num* に制限し，*num* より大きいものはインライン展開しません。
- *num* の目安については，後述の -Wp,-I オプションを参照してください。

[使用例]

- インライン展開対象関数のスタック・サイズを，中間言語での大きさ 64 に制限します。

```
C:¥>ca850 -cpu f3719 -Wp,-G64 main.c
```

-Wp,-N

[記述形式]

```
-Wp,-Nnum
```

- 省略時解釈

  より高度な最適化（実行速度優先）オプション指定時には -Wp,-N128 が指定されたものとみなし，それ以外では -Wp,-N24 が指定されたものとみなします。

[機能説明]

- インライン展開対象関数の中間言語サイズを *num* に制限し，*num* より大きいものはインライン展開しません。
- *num* の目安については，後述の -Wp,-l オプションを参照してください。

[使用例]

- インライン展開対象関数の中間言語サイズを 64 に制限します。

```
C:￥>ca850 -cpu f3719 -Wp,-N64 main.c
```

-Wp,-S

[記述形式]

```
-Wp,-S
```

- 省略時解釈

  なし

[機能説明]

- 一度しか参照されない静的な関数を無条件にインライン展開します。

[使用例]

- 一度しか参照されない静的な関数を無条件にインライン展開します。

```
C:¥>ca850 -cpu f3719 -Wp,-S -Os main.c
```

-Wp,-l

[記述形式]

```
-Wp,-l[=file]
```

- 省略時解釈

  関数の情報を出力しません。

[機能説明]

- 関数の情報を標準出力に出力，または *file* に追加出力します。
- 表示される情報は，上記の -Wp,-G，-Wp,-N オプションで指定する値の目安となります。たとえば，スタック・サイズでは，呼び出された関数の値が -Wp,-G で指定した値以下であればインライン展開されます。また，コード・サイズでは，呼び出された関数の値が -Wp,-N で指定した値以下であればインライン展開されます。
- このオプションによって出力されるスタック・サイズは，あくまでもプリオプティマイザが出力する中間言語でのサイズであるため，関数が実際に使用するスタック・サイズとは異なります。

[使用例]

- 関数の情報を標準出力に出力します。

```
C:￥>ca850 -cpu f3719 -Wp,-l main.c
```

-Wp,-inline

［記述形式］

```
-Wp,-inline
```

- 省略時解釈

  なし

［機能説明］

- #pragma inline 指定した関数のみインライン展開します。
- -Ot 指定時にはコンパイラが自動判別しインライン展開を行います。
- ユーザが指定した関数のみ展開する場合は，このオプションを指定してください。

［使用例］

- #pragma inline 指定した関数のみインライン展開します。

```
C:￥>ca850 -cpu f3719 -Wp,-inline -Ot main.c
```

-Wp,-no_inline

[記述形式]

```
-Wp,-no_inline
```

- 省略時解釈

  なし

[機能説明]

- #pragma inline 指定した関数を含む，すべての関数のインライン展開を抑止します。
- -Ot 指定時に，インライン展開機能をすべて抑止する場合に有効です。

[使用例]

- すべての関数のインライン展開を抑止します。

```
C:￥>ca850 -cpu f3719 -Wp,-no_inline -Ot main.c
```

-Wp,-r

[記述形式]

```
-Wp,-r[funcname]
```

- 省略時解釈

  なし

[機能説明]

- 関数 *funcname* をエントリ関数として，そこから呼び出された関数のうち，インライン展開後の不要な関数を削除します。
- *funcname* は，C 言語で記述された関数の先頭に‘_’を付けて指定します。*funcname* を指定しない場合は，“_main”が指定されたものとみなします。
- アセンブラ・ソースによってのみ呼び出される関数は，呼び出されていることが認識できないため，不要な関数として削除されます。

  ただし，割り込み関数とリアルタイム OS 用のタスクは，関数削除の対象から除外されています。

[使用例]

- 関数 func をエントリ関数として，そこから呼び出された関数のうち，インライン展開後の不要な関数を削除します。

```
C:¥>ca850 -cpu f3719 -Wp,-r_func -Om -Os main.c sub.c
```

ループ展開最適化制御

ループ展開最適化制御オプションには，次のものがあります。

- -Wo,-Ol
- -Wo,-Xlo

-Wo,-Ol

[記述形式]

```
-Wo,-Ol[num]
```

- 省略時解釈

  なし

[機能説明]

- for，while などで，ループを *num* 回展開します。
- 実行速度優先最適化の場合にのみ指定できます。
- 実行回数が N 回（N は定数）のループの実行と *num* 回展開されたコードを含むループの実行に変換されます。ただし，展開後のコード・サイズが大きいかループの実行回数が少ない場合は，展開数が少なくなったり展開されない場合があります。また，内側にループを含むような複雑な構造のループは，展開されない場合があります。
- *num* に 0，または 1 を指定した場合，展開が抑止されます[注]。また，*num* を指定しない場合，4 が指定されたものとみなします。なお，*num* は 10 進数で指定してください。

注　より高度な最適化（実行速度優先）指定時で，ループ展開を行いたくない場合に有効です。

[使用例]

- 実行回数が 10 のループを 4 回展開します。

```
C:¥>ca850 -cpu f3719 -Wo,-Ol4 -Ot main.c
```

下記のソースをコンパイルした場合，

```
i = 0;
while(i < 10) {
    /* 処理 */
    ++i;
}
```

以下のように展開されます。

```
i = 0;
/* 処理 */
i =1;
/* 処理 */
i = 2;
while(i < 10) {
    /* 処理 */
    ++i;
    /* 処理 */
    ++i;
    /* 処理 */
    ++i;
    /* 処理 */
    ++i;
}
```

-Wo,-Xlo

[記述形式]

```
-Wo,-Xlo
```

- 省略時解釈

  なし

[機能説明]

- ループ展開数を -Wo,-Ol*num* で指定した回数で固定してループ展開します。
- 実行速度優先最適化の場合にのみ指定できます。

[使用例]

- ループ展開数を 4 回固定でループ展開します。

```
C:￥>ca850 -cpu f3719 -Wo,-Xlo -Ot main.c
```

strcpy，strcmp 展開

strcpy，strcmp 展開オプションには，次のものがあります。

- -Xi

-Xi

[記述形式]

```
-Xi
```

- 省略時解釈

  関数 strcpy()，または strcmp() の呼び出しをインライン展開しません。

[機能説明]

- 配列（文字列を含む），および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開します。
- オブジェクトの実行速度は高速になりますが，コード・サイズは増大します。
- このオプションは，strcpy() の第二引数が文字列の場合，または strcmp() の場合のみ変換します。また，引数は，プログラム側で 4 バイトに整列されている必要があります（strcpy() の第二引数は文字列のため，C コンパイラが整列しています）。
- このオプションは，-Xpack オプションと同時指定はできません。

[使用例]

- 配列（文字列を含む），および構造体の整列条件を 4 バイトとし，関数 strcpy()，または strcmp() の呼び出しをインライン展開します。

```
C:￥>ca850 -cpu f3719 -Xi main.c
```

外部変数ソート

外部変数ソート・オプションには，次のものがあります。

- -Wo,-Op

-Wo,-Op

[記述形式]

```
-Wo,-Op[=file]
```

- 省略時解釈

  外部変数をアライメントの大きい順に並び替えません。

[機能説明]

- const ／ sconst セクション以外のセクションに配置されている外部変数をアライメントの大きい順に並び替えます。
- 中間言語ファイル *file* を指定した場合は，ソース・ファイル中の外部リンケージを持つ const ／ sconst セクション以外のセクションに配置されている変数の定義および仮定義を *file* に移動します。移動後のソース・ファイルの変数の定義および仮定義は，宣言と同じ扱いになります。最初に *file* が存在しなくてもエラーにはなりません。

[使用例]

- const ／ sconst セクション以外のセクションに配置されている外部変数をアライメントの大きい順に並び替えます。

```
C:¥>ca850 -cpu f3719 -Wo,-Op main.c
```

分岐命令制御

分岐命令制御オプションには，次のものがあります。

- -Wo,-XFo

-Wo,-XFo

[記述形式]

```
-Wo,-XFo
```

- 省略時解釈

  分岐命令に対してデバッグ情報を優先したコードを出力します。

[機能説明]

- 分岐命令をコード・サイズ優先で並べてコードを出力します。

  ただし，ソース・デバッグが難しくなります。
- このオプションは -Og ／ -O ／ -Os ／ -Ot 指定時に有効になります。

[使用例]

- 分岐命令をコード・サイズ優先で並べてコードを出力し，適度な最適化を行います。

```
C:￥>ca850 -cpu f3719 -Os -Wo,-XFo main.c
```

レジスタ使用制御

レジスタ使用制御オプションには，次のものがあります。

- -r
- -reg
- -Xmask_reg

-r

[記述形式]

```
-rnum=sym
```

- 省略時解釈

  外部変数を静的にレジスタに割り付けません。

[機能説明]

- 指定した外部変数 *sym* をレジスタ r*num* に割り付けます。
- *num* は -reg オプションを指定して空けたマスク・レジスタ以外のレジスタを指定します。
- *sym* は外部変数名（先頭の“_”を除く）で，volatile 変数，アドレス演算子を使用した変数，集成体，配列，内部リンケージを持つ変数，および周辺 I/O レジスタは指定できません。

[使用例]

- 外部変数 arg をレジスタ r18 に割り付けます（22 レジスタ・モード使用時）。

```
C:¥>ca850 -cpu f3719 -reg22 -r18=arg main.c
```

-reg

[記述形式]

```
-regn
```

- 省略時解釈

  -reg32

[機能説明]

- C コンパイラが使用するレジスタを *n* 本に制限します（レジスタ・モードを *n* とします）。

  n の値として指定できるのは次のとおりです。

表 B—3　レジスタ・モード

レジスタ・モード（*n*）	作業用レジスタ	レジスタ変数用レジスタ
22	r10 ～ r14	r25 ～ r29
26	r10 ～ r16	r23 ～ r29
32	r10 ～ r19	r20 ～ r29

- このオプションは，ソース・ファイルごとの指定では選択できません。常に全体のオプションとして設定します。
- このオプションによる設定はリンカにも認識されるため，適切なモードのライブラリが参照されます。
- このオプションを指定することで，ソフトウエア・レジスタ・バンク機能のレジスタ・モードを切り替えることができます。

[使用例]

- C コンパイラが使用するレジスタを 22 本に制限します。

```
C:¥>ca850 -cpu f3719 -reg22 main.c
```

-Xmask_reg

[記述形式]

```
-Xmask_reg
```

- 省略時解釈

  マスク・レジスタ機能は無効です。

[機能説明]

- マスク・レジスタ機能を使用します。
- この機能を使用すると，C コンパイラは r20 に 8 ビットのマスク値 0xff，r21 に 16 ビットのマスク値 0xffff が設定されているものとしてコードを出力します。マスク・レジスタ（r20，r21）へのマスク値の設定は，スタート・アップ・ルーチン等，ユーザ・プログラムで行う必要があります。
- V850 マイクロコントローラでは，バイト・データ，ハーフワード・データをメモリからレジスタにロードする場合，最上位ビットの値によりワード長へ符号拡張します。このため，unsigned char，unsigned short 型データの演算では，上位ビットのマスク・コードが生成される場合があります。

  演算結果をレジスタ変数へストアする場合，符号なしバイト・データ，符号なしハーフワード・データでは，上位ビットをクリアするためにマスク・コードが生成されます。

  どちらの場合も，ワード・データに切り替えれば回避できますが，ワード・データにできず，マスク・コードが生成される場合，マスク・レジスタ機能を用いることにより，コード・サイズの削減ができます。
- マスク・レジスタ機能を利用するかどうかの判断には，利用する側で次の点を十分考慮する必要があります。
  - マスク・コードが多く出力されるプログラムであるか。
  - マスク・レジスタとして使用するため，レジスタ変数用レジスタが 2 本少なくなるが，その影響はないか。
- このオプションを指定したとき，マスク・レジスタを使用したオブジェクトとしないオブジェクトが混在する場合，リンカでエラーとなります。
- 32 レジスタ・モードのとき，リンカに -mask_reg が渡されます。これにより，リンカによる標準ライブラリの検索は，標準フォルダより先に，マスク・レジスタ用フォルダ（lib850￥r32msk）で行います。

[使用例]

- マスク・レジスタ機能を使用します。

```
C:￥>ca850 -cpu f3719 -Xmask_reg main.c
```

プロローグ／エピローグ処理制御

プロローグ／エピローグ処理制御オプションには，次のものがあります。

- -Xpro_epi_runtime

-Xpro_epi_runtime

[記述形式]

```
-Xpro_epi_runtime[=on|=off]
```

- 省略時解釈

  -Xpro_epi_runtime=off（-Ot 指定）

  -Xpro_epi_runtime=on（-Ot 指定以外）

[機能説明]

- 関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しによる処理にするかどうかを設定します。
- on にした場合，関数のプロローグ／エピローグ処理がランタイム・ライブラリ呼び出しになります。
- [=on][=off] の指定をしなかった場合，[=on] が指定されたものと見なします。デフォルト時の動作は on となり，このオプションで [=off] を指定した場合，または -Ot オプションを指定した場合のみ off となります。

[使用例]

- 関数のプロローグ／エピローグ処理をランタイム・ライブラリ呼び出しにしません。

```
C:¥>ca850 -cpu f3719 -Xpro_epi_runtime=off main.c
```

変数配置制御

変数配置制御オプションには，次のものがあります。

- -G
- -Xsconst
- -Xcre_sec_data
- -Xcre_sec_data_only
- -Xsec_file

-G

[記述形式]

```
-Gnum
```

- 省略時解釈

  すべてのデータを，.sdata ／ .sbss セクションに配置します。

[機能説明]

- *num* バイト以下のデータを .sdata セクション，または .sbss セクションに配置します。
- #pragma section 指令，または「B. 7. 1　セクション・ファイル」で .sdata ／ .sbss セクションが指定されたデータは，そのサイズに関係なく .sdata ／ .sbss セクションに配置します。
- *num* は 10 進数で指定してください。*num* に設定する値の目安は，リンカの -A オプションで出力されます。

[使用例]

- 16 バイト以下のデータを .sdata セクション，または .sbss セクションに配置します。

```
C:￥>ca850 -cpu f3719 -G16 main.c
```

-Xsconst

[記述形式]

```
-Xsconst[=num]
```

- 省略時解釈

  すべての const 属性のデータ，文字列リテラルを，.const セクションに配置します。

[機能説明]

- const 属性のデータ，文字列リテラルを .sconst セクションに配置します。
- *num* を指定した場合，*num* バイト以下のデータを .sconst セクション配置し，*num* を省略した場合，そのサイズに関係なく配置します。
- *num* は 10 進数で指定してください。
- ファイルごとに異なるオプションを指定すると，変数の配置，および参照方法が異なるコードを生成しリンク時にもエラー，または警告メッセージが出力されることがあります。

[使用例]

- const 属性のデータ，文字列リテラルを .sconst セクションに配置します。

```
C:¥>ca850 -cpu f3719 -Xsconst main.c
```

-Xcre_sec_data

[記述形式]

```
-Xcre_sec_data[=outfile]
```

- 省略時解釈

  変数の頻度情報ファイルを出力しません。

[機能説明]

- セクション・ファイル・ジェネレータで使用する変数の頻度情報ファイルを出力します。

(1) *outfile* にファイル名を指定した場合

指定ファイル名でカレント・フォルダに *outfile* を保存します。

(2) *outfile* にフォルダを指定した場合

指定フォルダに .c，または .ic を .sec で置き換えたファイル名で保存します。

(3) *=outfile* を省略した場合

カレント・フォルダに .c，または .ic を .sec で置き換えたファイル名で保存します。

(4) 出力ファイルが複数の場合

outfile に指定したフォルダを作成し，.c，または .ic を .sec で置き換えたそれぞれのファイル名で保存します。

- C ソース・ファイルが複数存在し，それぞれのファイルに対してファイル名を指定して頻度情報ファイルを生成したい場合，コマンド・ラインでは，各 C ソース・ファイルに対して，“*=outfile*”付きでこのオプションを指定します。その際，C ソース・ファイルは，1 つずつ指定します。
- 変数の頻度情報ファイルは，C ソース・ファイル内の変数に対する ld，st 命令でのアクセス頻度情報を出力します。アセンブラ・ソース・ファイルに対しては何も行いません。
- このオプションと -Xcre_sec_data_only オプションを同時に指定した場合は，-Xcre_sec_data_only オプションが優先されます。

[使用例]

- 変数の頻度情報ファイル main.sec を出力します。

```
C:¥>ca850 -cpu f3719 -Xcre_sec_data main.c
```

-Xcre_sec_data_only

[記述形式]

```
-Xcre_sec_data_only[=outfile]
```

- 省略時解釈

  変数の頻度情報ファイルを出力しません。

[機能説明]

- セクション・ファイル・ジェネレータで使用する，変数の頻度情報ファイルを出力します。

  ただし，-Xcre_sec_data と違い，変数の頻度情報ファイルのみを出力するため，オブジェクトの生成までは行われません。
- 頻度情報ファイルのみを出力したい場合に使用します。

（1）*outfile* にファイル名を指定した場合

指定ファイル名でカレント・フォルダに *outfile* を保存します。

（2）*outfile* にフォルダを指定した場合

指定フォルダに .c，または .ic を .sec で置き換えたファイル名で保存します。

（3）=*outfile* を省略した場合

カレント・フォルダに .c，または .ic を .sec で置き換えたファイル名で保存します。

（4）出力ファイルが複数の場合

outfile に指定したフォルダを作成し，.c，または .ic を .sec で置き換えたそれぞれのファイル名で保存します。

- C ソース・ファイルが複数存在し，それぞれのファイルに対してファイル名を指定して頻度情報ファイルを生成したい場合，コマンド・ラインでは，各 C ソース・ファイルに対して，“=*outfile*”付きでこのオプションを指定します。その際，C ソース・ファイルは，（-c 指定で）1 つずつ指定します。
- 変数の頻度情報ファイルは，C ソース・ファイル内の変数に対する ld，st 命令でのアクセス頻度情報を出力します。アセンブラ・ソース・ファイルに対しては何も行いません。

[使用例]

- 変数の頻度情報ファイル main.sec のみを出力し，オブジェクトの生成は行いません。

```
C:￥>ca850 -cpu f3719 -Xcre_sec_data_only main.c
```

-Xsec_file

[記述形式]

```
-Xsec_file=file
```

- 省略時解釈

  なし

[機能説明]

- C コンパイラ起動時にデータのセクション割り当てを指定するためのセクション・ファイル（「B. 7. 1　セクション・ファイル」参照）名を指定します。必ずファイル名を指定してください。
- このオプションを複数指定し，複数のセクション・ファイルを入力することもできます。

[使用例]

- C コンパイラ起動時にデータのセクション割り当てを指定するためのセクション・ファイル section を指定します。

```
C:￥>ca850 -cpu f3719 -Xsec_file=section main.c
```

型制御

型制御オプションには，次のものがあります。

- -Xbitfield
- -Xchar
- -Xenum_type

-Xbitfield

[記述形式]

```
-Xbitfield=string
```

- 省略時解釈

  型指定子（signed，unsigned）の付かない単なる int 型の ビット・フィールドに対し，符号付きとして扱います。

[機能説明]

- 型指定子（signed，unsigned）の付かない単なる int 型のビット・フィールドに対し，符号付きとするか符号なしとするかを指定します。
- *string* に指定できるものは，次のとおりです。

s	符号付きとして扱う
signed	符号付きとして扱う
u	符号なしとして扱う
unsigned	符号なしとして扱う

- 符号なしとして扱う場合，警告メッセージを出力します。

[使用例]

- 型指定子（signed，unsigned）の付かない単なる int 型の ビット・フィールドに対し，符号付きとして扱います。

```
C:¥>ca850 -cpu f3719 -Xbitfield=s main.c
```

-Xchar

[記述形式]

```
-Xchar=string
```

- 省略時解釈

  型指定子（signed，unsigned）の付かない単なる char 型に対し，符号付きとして扱います。

[機能説明]

- 型指定子（signed，unsigned）の付かない単なる char 型に対し，符号付きとするか符号なしとするかを指定します。
- *string* に指定できるものは，次のとおりです。

s	符号付きとして扱う
signed	符号付きとして扱う
u	符号なしとして扱う
unsigned	符号なしとして扱う

[使用例]

- 型指定子（signed，unsigned）の付かない単なる char 型に対し，符号付きとして扱います。

```
C:￥>ca850 -cpu f3719 -Xchar=s main.c
```

-Xenum_type

[記述形式]

```
-Xenum_type=string
```

- 省略時解釈

  列挙型に対し，signed int として扱います。

[機能説明]

- 列挙型に対し，どの整数型と整合するかを指定します。
- *string* に指定できるものは，次のとおりです。

char	signed char として扱う
uchar	unsigned char として扱う
short	short として扱う
ushort	unsigned short として扱う

[使用例]

- 列挙型に対し，signed char として扱います。

```
C:￥>ca850 -cpu f3719 -Xenum_type=char main.c
```

switch-case 文出力コード制御

switch-case 文出力コード制御オプションには，次のものがあります。

- -Xcase
- -Xword_switch

-Xcase

［記述形式］

```
-Xcase=string
```

- 省略時解釈

  switch 文のコード出力に対して，コンパイラが最適と思われる形式を 自動判断します。

［機能説明］

- switch 文のコード出力方式を指定します。
- *string* に指定できるものは，次のとおりです。

ifelse	case 文の並びに沿った if-else 文と同じ形で出力します。 頻度が多い順に case 文を書いているときやラベル数が少ないときは，これを選択します。上から順に比較するので，頻繁に合致する case 文を先に記述すると余計な比較が減り，実行速度向上につながります。
binary	バイナリ・サーチ形式で出力します。 バイナリ・サーチ・アルゴリズムに用いて合致する case 文を探します。ラベル数が多いときにこれを選択すると，どの case 文も同じくらいの速さで見つけることができます。
table	テーブル・ジャンプ方式で出力します。 case 文の値を基にインデックス化したテーブルを参照し，switch 文の値により case ラベルを選択し，処理を行います。どの case 文にも同じくらい速く分岐します。ただし，case 値が連続していないときは無駄な領域ができます。

- 符号なしとして扱う場合，警告メッセージを出力します。

［使用例］

- switch 文のコード出力に対して，バイナリ・サーチ形式で出力します。

```
C:￥>ca850 -cpu f3719 -Xcase=binary main.c
```

-Xword_switch

[記述形式]

```
-Xword_switch
```

- 省略時解釈

  分岐テーブルを 2 バイトで生成します。

[機能説明]

- switch 文の case ラベルに対する分岐テーブルを，1 ラベルあたり 4 バイトで生成します。
- 長い switch 文のためコンパイル・エラーとなる場合，このオプションを指定します。

[使用例]

- 分岐テーブルを，1 ラベルあたり 4 バイトで生成します。

```
C:￥>ca850 -cpu f3719 -Xword_switch main.c
```

構造体パッキング制御

構造体パッキング制御オプションには，次のものがあります。

- -Xbyte
- -Xpack

-Xbyte

[記述形式]

```
-Xbyte
```

- 省略時解釈

  なし

[機能説明]

- 構造体の間接アドレス・アクセスをバイト単位でアクセスします。
- 構造体パッキング機能で，制限にかかるような場合に使用します。

[使用例]

- 構造体の間接アドレス・アクセスをバイト単位でアクセスします。

```
C:¥>ca850 -cpu f3719 -Xbyte main.c
```

-Xpack

[記述形式]

```
-Xpack=num
```

- 省略時解釈

  なし

[機能説明]

- このオプションを用いることにより，構造体メンバをメンバの型に応じてアライメントすることなく，指定したアライメントを用いることができます。
- データ・サイズは小さくすることができますが，コード・サイズは大きくなります。*num* には，1，2，4，8 が指定できます。デフォルトは 8 です。注
- C ソース・ファイル中に #pragma 指令で構造体パッキング指定がある場合に，このオプションを指定した場合，最初の #pragma 指令が出現するまでは，オプション指定値がすべての構造体に適用されます。それ以降は，#pragma 指令の値が適用されます。

  ただし，#pragma 指令の出現後でも指定がデフォルトになった部分は，オプション指定値が適用されます。
- このオプションは，-Xi オプションと同時指定はできません。
- V850 ／ V850Ex ／ V850E2 コア製品ミス・アライン・アクセス禁止の設定の CPU をご使用の場合，次の制限があります。この制限は，#pragma pack についても同様です。
  - 構造体メンバのアドレスを取得すると正しく取得できません。
  - ビット・フィールドへのアクセスは，そのメンバの型で読み込むため，データ領域もアクセスします。

    ビット・フィールドの幅が，メンバの型以下の場合，メンバの型で読み込むのでオブジェクトの外部にアクセスします。実行上，通常は問題がありませんが，I/O などがマップされていた場合，不正なアクセスとなる場合があります。

注 本バージョンでは *num* の値が“4”と“8”のときの動作は同じになります。

[使用例]

- 構造体メンバを，指定したアライメント 1 を用いてアラインメントします。

```
C:￥>ca850 -cpu f3719 -Xpack=1 main.c
```

far jump 出力制御

far jump 出力制御オプションには，次のものがあります。

- -Xfar_jump
- -Xj

-Xfar_jump

[記述形式]

```
-Xfar_jump=file
-Xfar_jump file
```

- 省略時解釈

  関数への分岐に対して，jarl 命令を使用します。

[機能説明]

- *file* に記述された関数への分岐に対して，jmp 命令を使用します。
- 関数本体が，jarl，および jr 命令では分岐できない範囲（± 2M バイト以上）にあり，リンカがエラーを出力する場合，このオプションを用いてコンパイルし直します。
- ファイル名には拡張子が必要となります。推奨する拡張子は“.fjp”です。
- Flash ／外付け ROM 再リンク機能でブート側からフラッシュ側の関数を呼び出す場合には，このオプションは指定できません。詳細は「B. 3. 3　ブート－フラッシュ再リンク機能」を参照してください。

[使用例]

- func.fjp に記述された関数への分岐に対して，jmp 命令を使用します。

```
C:¥>ca850 -cpu f3719 -Xfar_jump=func.fjp main.c
```

-Xj

[記述形式]

```
-Xj
```

- 省略時解釈

  C 言語で定義された通常の割り込み関数に対し，jr 命令を用います。

[機能説明]

- C 言語で定義された通常の割り込み関数に対し，jmp 命令を用います。
- 関数本体が，jr 命令では分岐できない範囲（± 1M バイト以上）にあり，リンカがエラーを出力する場合，このオプションを用いてコンパイルし直します。このオプションを省略した場合，jr 命令を用います。
- Flash ／外付け ROM 再リンク機能でブート側からフラッシュ側の関数を呼び出す場合には，このオプションは指定できません。詳細は「B. 3. 3　ブートーフラッシュ再リンク機能」を参照してください。

[使用例]

- C 言語で定義された通常の割り込み関数に対し，jmp 命令を用います。

```
C:￥>ca850 -cpu f3719 -Xj main.c
```

コメント出力

コメント出力オプションには，次のものがあります。

- -Xc

-Xc

[記述形式]

```
-Xc
```

- 省略時解釈

  出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力しません。

[機能説明]

- 出力するアセンブラ・ソース・ファイル中に C ソース・プログラムをコメントとして出力します。
- ただし，出力されるコメントは，あくまで参考であり，厳密にはコードと対応していない場合もあります。

  たとえば，グローバル変数とローカル変数，関数宣言などのコメントの出力位置がずれることがあります。また，

  最適化によりコードが削除され，コメントのみが残ることもあります。
- 本オプションは，-S，-a，-Fs，または -Fv のいずれかが指定されている必要があります。

[使用例]

- C ソース・プログラムをコメントとして，アセンブラ・ソース・ファイル main.s を出力します。

```
C:¥>ca850 -cpu f3719 -Xc -S main.c
```

ANSI 規約

ANSI 規約オプションには，次のものがあります。

- -Xe
- -Xdefvar
- -ansi

-Xe

[記述形式]

```
-Xe
```

- 省略時解釈

  16 ビット・データ以下の整数に対し mulh，divh 命令を利用します。

[機能説明]

- 16 ビット・データ以下の整数に対し mulh，divh 命令を利用せず，V850 製品の場合はランタイム・ライブラリ ___mul ／ ___mulu，___div ／ ___divu，V850E 製品の場合は mul ／ mulu，div ／ divu 命令を利用します。
- 実行速度は遅くなりますが，ANSI 規格に厳密な乗除算処理を行います。
- C コンパイラにおけるランタイム・ライブラリは，V850 マイクロコントローラのアーキテクチャにはない命令について，ANSI 規格を満たすために CA850 の標準ライブラリで用意されているものです。

[使用例]

- 16 ビット・データ以下の整数に対し，ランタイム・ライブラリ ___mul ／ ___mulu，___div ／ ___divu を利用します。

```
C:￥>ca850 -cpu f3719 -Xe main.c
```

-Xdefvar

[記述形式]

```
-Xdefvar
```

- 省略時解釈

  なし

[機能説明]

- 変数の仮定義を定義として扱います。
- このオプションを指定すると，複数のファイルで同名の仮定義が存在した場合に，リンク時に１つに結合されずに多重定義エラーにすることが可能となります。

[使用例]

- 変数の仮定義を定義として扱います。

```
C:￥>ca850 -cpu f3719 -Xdefvar main.c
```

-ansi

[記述形式]

```
-ansi
```

- 省略時解釈

  従来の C 言語の仕様との両立性を持たせ，警告メッセージを出力して処理を続行します。

[機能説明]

- C コンパイラの処理を厳密に ANSI 規格にあわせ，規格に反する記述に対してエラーや警告メッセージを出力します。
- _asm 形式以外の拡張記述は認められます。
- このオプション指定時は，マクロ名 __STDC__ を定義します。
- 言語仕様に厳密なコンパイル時の処理は次のようになります。

(1) トライグラフ系列

トライグラフを置換する。このオプションを指定しなかった場合，置換しない。

(2) ビット・フィールド

ビット・フィールドに int 型以外の型を指定した場合，エラーとする。このオプションを指定しなかった場合，警告メッセージを出力して許可する。

(3) 引数のスコープ

関数引数と同名の自動変数を宣言した場合，二重定義エラーとする。このオプションを指定しなかった場合，警告メッセージを出力して自動変数を有効とする。

(4) ポインタの代入

(a) 汎整数型変数へのポインタ型数値の代入はエラーとする。このオプションを指定しなかった場合，警告メッセージを出力し，キャストして代入する。

(b) 異なる型を指すポインタ同士の代入はエラーとする。このオプションを指定しなかった場合，警告メッセージを出力して許可する。

(5) 型変換

左辺値でない配列のポインタへの変換はエラーとする。このオプションを指定しなかった場合，警告メッセージを出力して許可する。

（6）比較演算子

算術型変数とポインタの比較はエラーとする。このオプションを指定しなかった場合，警告メッセージを出力して許可する。

（7）条件演算子

第 2 式と第 3 式がともに汎整数型，同じ構造体，同じ共用体，または代入先と同じ型へのポインタ型のいずれでもない場合，エラーとする。このオプションを指定しなかった場合，警告メッセージを出力し，キャストして代入する。

（8）# 行番号

エラーとする。このオプションを指定しなかった場合，“#line 行番号”と同様に扱う。

（9）行の途中の '#' 文字

エラーとする。このオプションを指定しなかった場合，警告メッセージを出力して許可する。

（10）_asm

警告メッセージを出力して関数呼び出しとして扱う。ただし __asm（“_”2 つ）は有効。このオプションを指定しなかった場合，アセンブラ挿入として扱う。

（11）__STDC__

値が 1 のマクロとして定義する。このオプションを指定しなかった場合，マクロとして定義しない。

（12）2 進定数

使用不可。このオプションを指定しなかった場合，“0b”，または“0B”と，その後ろに続く 1 個以上の“0”，または“1”の数字の並びを 2 進定数とします。

［使用例］

- C コンパイラの処理を厳密に ANSI 規格にあわせ，規格に反する記述に対してエラーや警告メッセージを出力します。

```
C:￥>ca850 -cpu f3719 -ansi main.c
```

日本語文字列制御

日本語文字列制御オプションには，次のものがあります。

- -Xk
- -Xkt

-Xk

[記述形式]

```
-Xk=code
```

- 省略時解釈

  使用する文字コードをシフト JIS とします。

[機能説明]

- 入力ファイル中の日本語のコメント，文字列に対し，使用する文字コードを指定します。
- 指定したコード以外のコードが存在する場合，エラー・メッセージを出力します。
- *code* に指定できるものは，次のとおりです。

e	EUC
euc	EUC
n	コードを保証しない
none	コードを保証しない
s	シフト JIS
sjis	シフト JIS

[使用例]

- 入力ファイル中の日本語のコメント，文字列に使用する文字コードを EUC とします。

```
C:¥>ca850 -cpu f3719 -Xk=e main.c
```

-Xkt

[記述形式]

```
-Xkt=code
```

- 省略時解釈

  日本語文字列を，コードを変換しません。

[機能説明]

- 日本語文字列を，指定したコードに変換して出力します。
- アプリケーション開発時に使用した漢字コードを，ターゲットで変更したい場合に有効です。
- *code* に指定できるものは，次のとおりです。

e	EUC
euc	EUC
n	コードを保証しない
none	コードを保証しない
s	シフト JIS
sjis	シフト JIS

- -Xc 指定により出力されるコメントは，このオプションの指定に関わらず，変換しません。

[使用例]

- 日本語文字列を，EUC コードに変換して出力します。

```
C:¥>ca850 -cpu f3719 -Xkt=euc main.c
```

ライブラリ指定

ライブラリ指定オプションには，次のものがあります。

- -L
- -R
- -l

-L

[記述形式]

```
-Ldir
```

- 省略時解釈

  ライブラリを，標準フォルダのみ検索します。

[機能説明]

- ライブラリをフォルダ *dir*，標準フォルダの順で検索します。
- 標準フォルダは，“インストール・フォルダ¥CubeSuite+¥CA850¥V*x.xx* 注¥lib850 フォルダ”，および“インストール・フォルダ¥CubeSuite+¥CA850¥V*x.xx* 注¥lib850¥r32 フォルダ”です。ただし，レジスタ・モードが指定された場合，r32 フォルダの代わりに，同じ並びの r22，または r26 フォルダで検索します。

  注　V*x.xx* は C コンパイラのバージョンです。

- リンカの -L オプションを参照してください。

[使用例]

- ライブラリをフォルダ lib，標準フォルダの順で検索します。

```
C:¥>ca850 -cpu f3719 -Llib main.c
```

-R

[記述形式]

```
-R file
```

- 省略時解釈

標準フォルダにある crtN.o，または crtE.o をスタート・アップ・モジュールとします。標準フォルダは，“インストール・フォルダ¥ CubeSuite+ ¥ CA850 ¥ V*x.xx* 注¥ lib850 ¥ r32（r26，r22）”です。

注　V*x.xx* は C コンパイラのバージョンです。

[機能説明]

- リンカまで起動する場合，使用するスタート・アップ・モジュールを *file* とするようリンカに指示します。

[使用例]

- 使用するスタート・アップ・モジュールを start.o とするようリンカに指示します。

```
C:¥>ca850 -cpu f3719 -R start.o main.c
```

-l

[記述形式]

```
-lstring
```

- 省略時解釈

  アーカイブ・ファイルを，参照しません。ただし，C コンパイラからリンカを起動する場合，C コンパイラは自動的に標準ライブラリ，および数学ライブラリのリンク（-lm -lc）指定をリンカに渡します。

[機能説明]

- リンカで参照する，アーカイブ・ファイルを指定します。

  ただし，C コンパイラからリンカを起動する場合，C コンパイラは自動的に標準ライブラリ，および数学ライブラリのリンク（-lm -lc）指定をリンカに渡します。
- アーカイブ・ファイルの指定方法については，リンカのライブラリ指定（-l）オプションを参照してください。

[使用例]

- リンカで参照する，アーカイブ・ファイル libarc.a を指定します。

```
C:¥>ca850 -cpu f3719 -larc main.c
```

警告メッセージ制御

警告メッセージ制御オプションには，次のものがあります。

- -w
- -won
- -woff

-w

[記述形式]

```
-wnum
-wstring+
-wstring-
```

- 省略時解釈

  -w*num* を省略した場合は，-w1。

  -w*string*+，-w*string*- を省略した場合は，警告メッセージの出力を -w*num* オプションのレベル指定に従います。

[機能説明]

- -w*num* では警告メッセージのレベルを指定します。
- *num* に指定できる数字は，次のとおりです。

0	メッセージを抑止する
1	通常の警告メッセージを出力する
2	詳細なメッセージを出力する

- *num* を省略した場合，-w0 が指定されたものとみなします。
- -w*string*+ および -w*string*- ではレベルに関わりなく，項目ごとに警告メッセージの出力，および抑止を指定します。‘+’ を付けた場合メッセージを出力し，‘-’ を付けた場合メッセージを抑止します。
- *string* に指定できる文字列は，次のとおりです。

bitfield_align	ビット・フィールドのメンバが整列条件の境界を越えるため，次の境界から割り当てられた場合
bitfield_type	ビット・フィールドに対し，ANSI 仕様で指定することのできない型が指定された場合
callnodecl	宣言のない関数が呼ばれた場合
cast_type	サイズが小さい型に変換した場合
comparison	比較式が常に真（または偽）になる場合
nopic	自動変数でない変数のアドレス，または関数のアドレスを用いたポインタ型外部変数の初期化の場合
pragma	実行不可能な #pragma 記述が現れた場合

sharp	ソース行中に#文字が現れた場合

- ‘+’，および‘-’を付けない場合，エラーとなります。

［使用例］

- 詳細な警告メッセージを出力します。

```
C:￥>ca850 -cpu f3719 -w2 main.c
```

- ビット・フィールドに対し，ANSI 仕様で指定することのできない型が指定された場合の警告メッセージを出力します。

```
C:￥>ca850 -cpu f3719 -wbitfield_type+ main.c
```

-won

[記述形式]

```
-won=num[,num]...
-won=num1-num2[,num3-num4]...
```

- 省略時解釈

  なし

[機能説明]

- *num* で指定した番号の警告メッセージを出力するようにします。
- *num* には 2000 番台の警告メッセージが指定できます。
- W2042 の警告メッセージを出力する場合は，-won=2042 になります。*num1-num2* の形式で指定すると，*num1* から *num2* までの警告メッセージを指定したことになります。*num* は省略できません。
- C コンパイラにない警告メッセージ番号を指定すると，警告メッセージが出力されます。

[使用例]

- W2042 の警告メッセージを出力します。

```
C:¥>ca850 -cpu f3719 -won=2042 main.c
```

-woff

[記述形式]

```
-woff=num[,num]...
-woff=num1-num2[,num3-num4]...
```

- 省略時解釈

  なし

[機能説明]

- *num* で指定した番号の警告メッセージを抑制します。
- *num* には 2000 番台の警告メッセージが指定できます。
- W2042 の警告メッセージを抑制する場合は，-woff=2042 になります。*num1-num2* の形式で指定すると，*num1* から *num2* までの警告メッセージを指定したことになります。*num* は省略できません。
- C コンパイラにない警告メッセージ番号を指定すると，警告メッセージが出力されます。

[使用例]

- W2042 の警告メッセージを抑制します。

```
C:¥>ca850 -cpu f3719 -woff=2042 main.c
```

コマンド・ファイル指定

コマンド・ファイル指定オプションには，次のものがあります。

- @

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイル（「(2) コマンド・ファイル」参照）として扱います。これにより，オプション文字列の長さの制限を意識せずに済みます。
- コマンド・ファイルでは，指定する引数を複数行に分けて記述できますが，オプションやファイル名などが 2 行に分かれないようにしてください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>ca850 @command main.c
```

CPU 不具合パッチ

CPU 不具合パッチ・オプションには，次のものがあります。

- -Xv850patch

-Xv850patch

[記述形式]

```
-Xv850patch[=num]
```

- 省略時解釈

  なし

[機能説明]

- CPU の障害に対応したコードを出力するため，C コンパイラが出力するアセンブラ・ソース・ファイルに対し，*num* に応じて，アセンブラに -p[*num*] オプション指定を指示します（「（2）CPU 不具合の回避オプション」参照）。
- *num* には 1，2，3，4，4a，5，6，7，8，9，10，11 が指定できます。5 ～ 10 は V850E/ES コアでのみ有効です。
- “=*num*”を省略した場合，*num* に 1，2，3，5，6，7，8，9，10 が指定されたものとみなします。
- このオプションは，CPU の障害回避のためのオプションです。使用している CPU に該当する障害であるかどうかは，CPU 添付の資料を参照してください。
- -Xv850patch=11 オプションのみ C コンパイラで処理します。-Xv850patch=11 オプションを指定すると，次の命令を出力しなくなります。
  - set1/clr1/not1
  - V850E/ES コアのミスアライン・アクセス（構造体パッキング時）

  ただし，asm 文とアセンブラ・ソース・ファイルに関してはチェックを行わないので，これらの箇所で使用した場合はそのまま出力されます。
- -Xv850patch=11 オプションを指定して，プログラム中で周辺 I/O レジスタへのビット・アクセスの記述をした場合には，周辺 I/O レジスタへワード（4byte）単位のアクセスになります。ビット・アクセスではなく，バイト／ハーフワード単位での操作の記述に変更してください。
- CPU コアと，バッチ・オプションに対応する不具合は，次のとおりです（保守品と廃品種を除く，最新バージョンの μPD(F)703xxx の場合）。

  なお，使用している CPU に該当する障害であるかどうかは，CPU の資料を参照してください。

表 B―4　CPU コアと -Xv850patch オプションに対応する不具合

CPU コア	-Xv850patch=11
V850 コア	―
V850E/MS1	×
V850E1 コア	×
V850ES コア	×
V850E2 コア	―

備考　×：該当
　　―：非該当

［使用例］

- CPU の障害に対応したコードを出力するため，C コンパイラが出力するアセンブラ・ソース・ファイルに対し，アセンブラに -p4a オプション指定を指示します。

```
C:￥>ca850 -cpu f3719 -Xv850patch=4a main.c
```

各モジュール

C コンパイラから各モジュールにオプションを渡すことができます。

- -W

-W

[記述形式]

```
-Wx,option
```

- 省略時解釈

  なし

[機能説明]

- *option* をモジュール *x* に対するオプションとしてを渡します。*option* がカンマを含む場合は，そのカンマで区切られた複数のオプションとして与えられます。
- モジュール *x* に指定できるものは，次のとおりです。

p	プリオプティマイザ（popt）
o	広域最適化部（opt）
i	機種依存最適化部（impr）
a	アセンブラ（as850）
l	リンカ（ld850）

（1）プリオプティマイザ（popt）

（a）-Wp,-D

コンパイル時のメモリ使用量を減らすことができます。

（b）-Wp,-G*num*

インライン展開対象関数のスタック・サイズを，中間言語での大きさ *num* に制限し，*num* より大きいものはインライン展開しません。

num の目安については，後述の -Wp,-I オプションを参照してください。

このオプションを指定しない場合，-Wp,-G32 が指定されたものとみなします。

（c）-Wp,-N*num*

インライン展開対象関数の中間言語サイズを *num* に制限し，*num* より大きいものはインライン展開しません。

num の目安については，後述の -Wp,-I オプションを参照してください。

このオプションを指定しない場合，より高度な最適化（実行速度優先）オプション指定時には -Wp,-N128 が指定されたものとみなし，それ以外では -Wp,-N24 が指定されたものとみなします。

（d）-Wp,-S

一度しか参照されない静的な関数を無条件にインライン展開します。

（e）-Wp,-I[=file]

関数の情報を標準出力に出力，または file に追加出力します。

表示される情報は，上記の -Wp,-G，-Wp,-N オプションで指定する値の目安となります。たとえば，スタック・サイズでは，呼び出された関数の値が -Wp,-G で指定した値以下であればインライン展開されます。また，コード・サイズでは，呼び出された関数の値が -Wp,-N で指定した値以下であればインライン展開されます。

なお，このオプションによって出力されるスタック・サイズは，あくまでもプリオプティマイザが出力する中間言語でのサイズであるため，関数が実際に使用するスタック・サイズとは異なります。

（f）-Wp,-r[_*funcname*]

funcname をエントリ関数として，そこから呼び出された関数のうち展開後に不要な関数を削除します。

funcname は，関数名の先頭に ‘_’ を付けて指定します。*funcname* を指定しない場合，“_main” が指定されたものとみなします。

なお，アセンブラ文によってのみ呼び出されている関数は，呼び出されていることが認識されず，不要な関数として削除されます。ただし，割り込み関数とリアルタイム OS 用のタスクは，関数削除の対象から除外されています。

（g）-Wp,-inline

#pragma inline 指定された関数のみインライン展開します。

（h）-Wp,-no_inline

#pragma inline 指定された関数を含むすべてのインライン展開を抑止します。

（2）広域最適化部（opt）

（a）-Wo,-Ol[*num*]

for，while などで，ループを *num* 回展開します。

実行速度優先最適化の場合にのみ指定できます。

実行回数が N 回（N は定数）のループの実行と *num* 回展開されたコードを含むループの実行に変換されます。ただし，展開後のコード・サイズが大きいか，ループの実行回数が少ない場合は，展開数が少な

くなったり展開されない場合があります。また，内側にループを含むような複雑な構造のループは，展開されない場合があります。

num に 0，または 1 を指定した場合，展開が抑止されます注。また，*num* を指定しない場合，4 が指定されたものとみなします。なお，*num* は 10 進数で指定してください。

注 より高度な最適化（実行速度優先）指定時で，ループ展開を行いたくない場合に有効です。

例

実行回数が 10 のループを 4 回展開する場合

```
i = 0;
while(i < 10) {
    /* 処理 */
    ++i;
}
```

```
i = 0;
/* 処理 */
i =1;
/* 処理 */
i = 2;
while(i < 10) {
    /* 処理 */
    ++i;
    /* 処理 */
    ++i;
    /* 処理 */
    ++i;
    /* 処理 */
    ++i;
}
```

（b）-Wo,-Op[=*file*]

const ／ sconst セクション以外のセクションに配置されている外部変数をアライメントの大きい順に並び替えます。

中間言語ファイル *file* を指定した場合は，ソース・ファイル中の外部リンケージを持つ const ／ sconst セクション以外のセクションに配置されている変数の定義および仮定義を *file* に移動します。移動後のソース・ファイルの変数の定義および仮定義は，宣言と同じ扱いになります。最初に *file* が存在しなくてもエラーにはなりません。

（c）-Wo,-XFo

分岐に関して，コード・サイズを優先したコードを出力します。

ただし，デバッグ情報への影響があります。このオプションは，-Og ／ -O ／ -Os ／ -Ot 指定時に有効になります。

このオプションを省略した場合，分岐に関して，デバッグ情報を優先したコードを出力します。

（d）-Wo,-Xlo

ループ展開を CA850 Ver.2.02 以前のバージョンの条件で展開します。

（3）機種依存最適化部（impr）

（a）-Wi,-D

コンパイル時のメモリ使用量を減らすことができます。

ただし，コンパイル速度は低下します。メモリを非常に多く使い，コンパイルが正常にできなくなるような場合に指定します。

（b）-Wi,-O4

データ・フロー解析を緻密に行い，次の最適化を実行します。

- 分岐命令をまたいだレジスタの最適化
- 絶対値演算の最適化
- 分岐命令をまたいだ cmp 命令の最適化
- 分岐命令をまたいだ復帰命令の最適化

ただし，コンパイル速度はかなり遅くなります。最適化オプション -O，-Os，または -Ot 指定をした上で，さらにデータ・フロー解析を強力に行いたい場合のみ指定してください。

（c）-Wi,-P

ラベルを整列する最適化を抑制します。これにより，コード・サイズを小さくできます。

（4）アセンブラ（as850）

「B. 2. 3　オプション」を参照してください。

（5）リンカ（ld850）

「B. 3. 2　オプション」を参照してください。

［使用例］

- データ・フロー解析を緻密に行い，最適化を実行します。

```
C:￥>ca850 -cpu f3719 -Wi,-O4 -Os main.c
```

その他

その他のオプションには，次のものがあります。

- +Oc

+Oc

[記述形式]

```
+Oc
```

- 省略時解釈

  なし

[機能説明]

- 強力な最適化を行います。
- V850E2 コアのデバイスを品種指定している場合には，本機能はデフォルトで有効となっています。

[使用例]

- 強力な最適化を行います。

```
C:¥>ca850 +Oc -Ot -Wi,-O4 main.c
```

B. 1. 5　注意事項

（1）オプションの複数指定

これらのオプションの中には，他のオプションと同時に指定された場合，無効になるものがあります。次に示した“＞”の右側に置かれたオプションは，左側に置かれたオプションと同時に指定された場合，無効になります。

- -E ＞ -P
- -U ＞ -D
- -E ／ -P ＞ -G ／ -L ／ -O ／ -R ／ -S ／ -Wc ／ -a ／ -c ／ -l ／ -m ／ -o

  前処理で終了するため，フロントエンド以降のモジュールに関するオプションは無効になります。
- -S ＞ -L ／ -R ／ -W[a|l] ／ -a ／ -c ／ -l

  コード生成部，または機種依存最適化部で終了するため，アセンブラ以降のモジュールに関するオプションは無効になります。
- -V ／ -help

  あとから指定された方が無効になります。また，このオプションを指定した場合，他のオプションはすべて無効になります。
- -c ＞ -L ／ -R ／ -Wl ／ -l

  アセンブラで終了するため，リンカ以降のモジュールに関するオプションは無効になります。
- -m ＞ -G ／ -L ／ -O ／ -R ／ -S ／ -Wc ／ -a ／ -c ／ -l

  フロントエンドで終了するため，プリオプティマイザ以降のモジュールに関するオプションは無効になります。
- -Og ／ -O ／ -Os ／ -Ot ＞ -a ／ -Fv

  -Og ／ -O ／ -Os ／ -Ot が指定された場合，正しく表示しない場合があります。
- -Od ／ -Ob ／ -Og ／ -O ／ -Os ／ -Ot

  あとから指定されたものが有効となります。
- -w ／ -w[1|2]

  先に指定された方が無効となります。

（2）コマンド・ファイル

コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなく，ファイルに記述して指定するものです。C コンパイラは，コマンド・ファイルの内容をコマンド・ラインの引数のように扱います。また，コマンド・ファイルでは，指定する引数を複数行に分けて記述できます。ただし，オプションやファイル名などが 2 行に分かれないようにしてください。また，コマンド・ファイルのネストはできません。

コマンド・ファイルは，次の文字については特殊文字として扱います。

"（ダブルクォーテーション）	次の "（ダブルクォーテーション）までの文字列を連続した文字列として扱います。
#（シャープ）	行頭に指定した場合は行末までをコメントとして扱います。
^（ハット）	直後の文字を特殊文字として扱いません。

なお，特殊文字自体はコマンド・ファイルを指定したCコンパイラのコマンド・ラインの中に含まれずに削除されます。

備考　as850，ar850，hx850，dump850，dis850，およびromp850は，"（ダブルクォーテーション）のみ使用できます。

- コマンド・ファイルの例

```
-Dtest          … #define test を記述
-o object       … オブジェクト・ファイル名を指定
a.c             … コンパイルするファイルを指定
```

- コマンド・ファイルの指定例

```
C:¥>type cfile
        -cpu 3201 -c -Os file.c ← コマンド・ファイルの内容
C:¥>ca850 @cfile ← ca850 -cpu 3201 -c -Os file.c と同じ動作をする
```

(3) 効率的な最適化の仕方

“最適化”は，アプリケーションの実行速度を向上したり，使用ROM容量を小さくしたりする処理です。最適化のかかり方は，最適化のレベルによって異なります。最適化レベルの高い最適化を選択した場合，コンパイル速度が低下したり，削除／変更されるCソース行や変数のレジスタ化の確率が高くなります。後者の場合，デバッガにおいてブレークポイントが設定できなくなる現象などが発生する可能性があり，デバッグ効率が悪くなることがあります。

ここでは，-Oオプションで指定可能な最適化処理の主な内容と，効率的に最適化を行うための指定を示します。

図B—3　最適化処理と項目

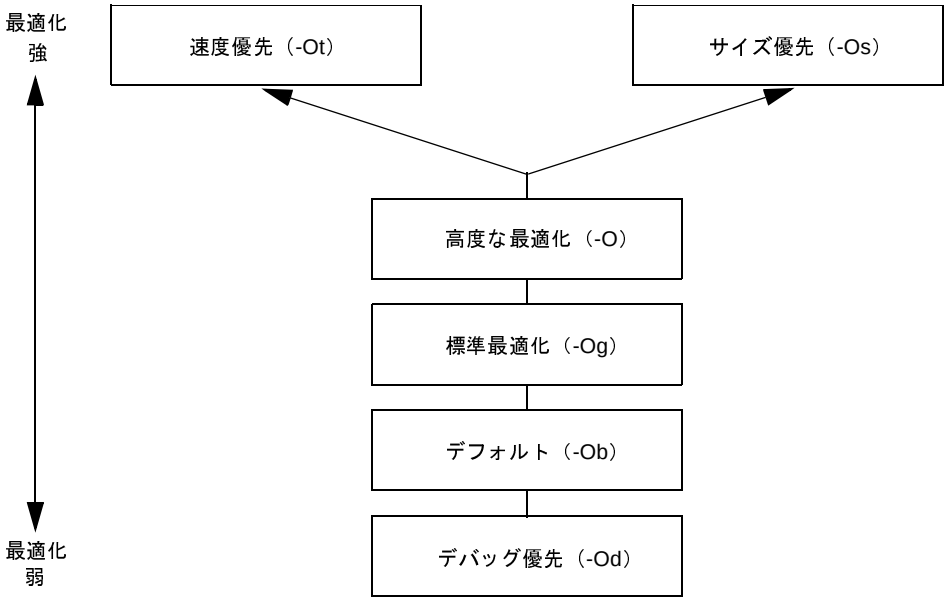


表 B―5　最適化処理と項目

オプション：最適化機能	効果			
	デバッグ	コード効率	実行速度	コンパイル時間
-Od：デバッグ優先	レベル 4	レベル 1	レベル 1	レベル 3
-Ob：デフォルト	レベル 3	レベル 2	レベル 2	レベル 3
-Og：標準最適化	レベル 3	レベル 3	レベル 3	レベル 3
-O：高度な最適化	レベル 2	レベル 4	レベル 4	レベル 2
-Os：より高度な最適化（サイズ優先）	レベル 1	レベル 5	レベル 4	レベル 2
-Ot：より高度な最適化（実行速度優先）	レベル 1	レベル 4	レベル 5	レベル 1

表中の表現は次のとおりです。

デバッグ	最適化が強くなると，C ソース行の削除や同一処理行を一箇所にまとめる最適化を行うことから，ブレークポイントが設定できる箇所が少なくなる傾向があります。変数もメモリからレジスタを割り当てる確率も向上します。 最適化によるブレークポイントの現象や変数のレジスタ割り当てが少ないものを“レベル 4”とし，最も傾向が強いものを“レベル 1”としています。“レベル 1”となっているものであっても，デバッグは可能です。
コード効率	ROM サイズ効率について，“レベル 1 ～レベル 5”で分類しています。 最も ROM サイズが小さくなるオプションは，-Os ですが，コンパイル時間がかかります。 ROM 容量に比較的余裕のある場合は，-Og，または -O オプションを指定してください。
実行速度	実行速度について，“レベル 1 ～レベル 5”で分類しています。 モジュール全体として ROM 容量を小さくしたいが，クリティカルな関数のみ実行速度をさらに向上したい場合は，ファイル単位で -Ot オプションを指定してください。
コンパイル時間	コンパイル時間について，“レベル 1 ～レベル 3”で分類しています。 -O，-Os，-Ot は強力な最適化を行うことから，コンパイル時間はこれら以外のオプションに比べて遅くなります。

（a）-Od：デバッグ優先

基本ブロック[注]内最適化を行います。基本ブロック内最適化とは，基本ブロック内で掌握できる情報を用いて行う最適化です。

- 定数計算や式の変形，
- 基本ブロック内の共通部分式の認識，
- 基本ブロック内の複写の伝播

などがあります。

なお，この基本ブロック内最適化は，コンパイル時にデフォルトで行われます。たとえば，“定数のみの演算式をコンパイル時に結果の定数に置き換える”という最適化を指します。

C コンパイラでは最も弱い最適化となります。この最適化は，CA850 Ver.2.4x のデフォルトの最適化と同等の最適化レベルとなります。

注　必ず先頭の命令から入り，最後の命令以外からは分岐しない最大の命令の並びです。

（b）-Ob：デフォルト

基本ブロック内最適化と自動変数のカラリング・レジスタ割り付けを行います。

- 自動変数をレジスタとして割り付けます。
  デバッグ時の影響はありません。

CA850 Ver.2.50 以降のデフォルトです。レジスタ割り付けが高機能となるため，-Od よりも無駄なコードが削除されます。

（c）-Og：標準最適化

基本ブロック内最適化，およびカラリング・レジスタ割り付けに加え，関数内で掌握できる情報を用いて，次の最適化を行います（代表的な物のみ記載）。

- 共通な演算を見つけ，まとめて処理する命令列を出力する。
- ループ内の値が変化しない代入文をループ外へ移動する。
  ステップ実行や，ブレークポイントがユーザの意図どおり設定できないことがあります。
- 冗長な代入文の削除。
  削除された行のブレークポイントは設定できません。
- 外部変数をレジスタ割り付けする。
  デバッグ時にメモリへの read ／ write ブレークが正確に行えないことがあります。
- C コンパイラにより命令を並び替えてレジスタ／フラグのハザードを回避する最適化を行います。
  デバッグ時の影響はありません。

高度な最適化を行う場合に比べ，コンパイル速度も速く，コード効率／実行速度は，C コンパイラの最適化で中間の性能となります。ROM 容量に比較的余裕のある場合は，このオプションの設定を推奨します。

（d）-O：高度な最適化

-Og までの最適化に加え，次の最適化を行います（代表的な物のみ記載）。

- 実行回数が 1 回のループのみは展開し，終了条件判断のオーバーヘッドを回避します。
  デバッグ時の影響はありません。
- ラベルの整列や関数の先頭での 4 バイト整列を抑制します。
  デバッグ時の影響はありません。
- 未参照ラベルの削除を行います。
  削除対象のラベルにブレークポイントが設定できなくなります。
- 不要命令の削除を行います。
  ブレークポイントやステップ実行がユーザの意図どおり設定できないことがあります。
- ピープホール最適化（効率の良い命令列への 5 命令以内の並び替え）を行います。
  ブレークポイントやステップ実行がユーザの意図どおり設定できないことがあります。

この最適化は，CA850 Ver.2.4x のオブジェクト・サイズ優先 -Os に相当します。

なお，このオプションは，CA850 Ver.2.4x で行われていた一度しか参照されない静的関数のインライン展開は行われません。

（e）-Os：より高度な最適化（サイズ優先）

-O の処理を最適化ができなくなるところまで，最適化を行います。C コンパイラがサポートしている最適化のうち，コード・サイズを増加させない最適化のすべてを行い，できるかぎりサイズを小さくする，最強のオブジェクト・サイズ優先最適化オプションです。

ただし，アプリケーションの内容によっては，-Os に加え，次のオプションや機能を使用することにより，最適化をさらに強化できる場合があります。

なお，アプリケーションの内容によっては，上記オプションに加えて，次のオプションや機能を使用することにより，最適化を強化できる場合があります。

- -Wi,-O4 を指定する

  データ・フロー解析を行い，最適化を強化します。ただし，コンパイル時間はかなり増大する可能性があります。

- マスク・レジスタを使用する

  unsigned char，unsigned short 型の演算のためにマスク・コードが多発するアプリケーションの場合，マスク・レジスタ機能を利用すると，コード・サイズを削減できます。

  ただし，マスク・レジスタ機能では，32 レジスタ・モードの場合は使用できるレジスタ変数用レジスタが 2 本少なくなり，32 レジスタ・モード以外の場合は空きレジスタが 2 本少なくなります。

- セクション・ファイルを使用する

  データは，内蔵メモリや，gp ／ r0 相対で 1 命令参照するセクションに割り当てると，コード・サイズが削減され，高速化できます。プログラムでデータのセクション割り当てを行っていない場合，コンパイル時にセクション・ファイル（「B. 7. 1　セクション・ファイル」参照）で，[tidata.byte] ／ [tidata.word] ／ [sidata] ／ [sedata] ／ [sconst] ／ [sdata] に割り当てます。

C コンパイラでコード・サイズに着目した最適化では最もサイズが小さくなります。CA850 Ver.2.4x のオブジェクト・サイズ優先 -Os ＋オプショナル最適化 -Ol に相当します。

なお，このオプションは，CA850 Ver.2.4x で行われていた一度しか参照されない静的関数のインライン展開は行われません。

（f）-Ot：より高度な最適化（実行速度優先）

実行速度優先で最適化を行います。データ処理アプリケーションなど，サイズを犠牲にしてでも実行時間を短縮したい場合に使用します。

-O までの最適化に加えて，

- ラベルの 4 バイト整列
- 関数の先頭での 4 バイト整列

の抑制の最適化を行い，さらに，

- テール・リカージョン最適化
- インライン展開
- ループ展開

を行います。

テール・リカージョン最適化では，関数の終わりの return 文が，その関数自身の呼び出しである場合，その関数をループに変換し，関数呼び出しによるスタック量を減らします。

インライン展開では，関数呼び出しの部分に関数本体を展開し，最適化の可能性を高め，呼び出しによるオーバーヘッドを防ぎます。

ループ展開では，ループ本体を複数回展開し，最適化の可能性を高め，条件判断や分岐によるオーバーヘッドを防ぎます。

インライン展開，ループ展開では，オブジェクト・サイズは大きくなりますが，実行速度の向上が期待できます。

なお，-Ot を指定し，ラベル定義している asm 文を含む関数を使用した場合，関数の定義部分とインライン展開された部分とで，同じラベルが定義されることになります。この場合，ラベルの多重定義エラーになるため注意が必要です。また，#pragma block_interrupt，#pragma interrupt，#pragma rtos_task，#pragma text 指定された関数は，インライン展開しません。その際，メッセージも出力しません。

スタック・フレームの操作のような，インライン展開されることを予期していない asm 文を含む関数を使用した場合には，不正な関数フレーム操作などが起こるために，実行エラーとなることがあります。

注意　より高度な最適化（実行速度優先）によってサイズが大きくなりすぎる場合，“-Wp,-G”や“-Wo,-Ol”オプションで，インライン展開やループ展開を調整してください。オプションと関係なく特定の関数のみインライン展開するには，#pragma inline を使用します。これにより“サイズ優先”を指定しつつ，特定の関数呼び出しのみ実行速度を優先できます。

なお，アプリケーションの内容によっては，-Os 指定のときと同様にマスク・レジスタなどで実行速度優先の最適化を強化できる場合があります。

また，次の機能によっても，強化できる場合があります。

- strcpy()，strcmp() を展開する

  文字列コピー関数の strcpy() を多用するアプリケーションの場合，“strcpy/strcmp の展開”を行う -Xi オプションを指定すると，実行時間が短縮されます。ただし，サイズは増大します。

- -Wp,-r を指定する

  ソース・ファイルをマージしたインライン展開の結果，不要な関数が生じる場合があります。そのような場合，“-Wp,-r”オプションを指定すると，不要な関数が削除され，サイズが削減される場合があります。

C コンパイラで実行速度に着目した最適化では最も実行速度が短くなります。CA850 Ver.2.4x の実行速度優先 -Ot ＋オプショナル最適化 -Ol に相当します。

以上のように，C コンパイラの最適化には，いくつかのレベル／項目がありますが，最適化を指定する場合，基準となるのは，次の選択です。

- サイズを重視する
- サイズを犠牲にしても実行速度を重視する

ほとんどの最適化機能は，サイズ削減と同時に実行速度を向上させますが，一部の機能を使用するかどうかで，サイズをより重視するか，実行速度をより重視するかが決まります。

（4）最適化によるデバッグへの影響

最適化を行うと，ソース・デバッガを使用する際，次のような影響があるので注意してください。

- 最適化による式の変形（複写の伝播や共通部分式の認識）によって，ソース・プログラム中での出現箇所で“変数参照”が起こらなくなり，変数の read ／ write イベントがユーザの意図どおり発生しない場合があります。
- 文の共通化や削除，並び替えが行われると，ステップ実行やブレークポイントがユーザの意図どおり設定できないことがあります。
- 変数の生存範囲（プログラム中でその変数を参照可能な範囲），変数の位置（レジスタやメモリ上の位置）が変更される可能性があります。
- 文の削除が起こった場合，その文にはブレークポイントを設定できません。
- 文の移動や分割，統合により，実行命令の順序が入れ替わった[注]場合，入れ替わった行とそれらの行の間にある行は，1 つのまとまりとして扱われ，途中のブレークポイントの設定やステップ実行ができなくなる場合があります。

注 あるソース行に対する実行命令のアドレスが，それ以前の行に対する実行命令のアドレスより小さくなった，あるいは，それ以後の行に対する実行命令のアドレスより大きくなったことをいいます。

- if-else の各場合で実行命令の順序が入れ替わったり，ループ展開で実行命令の順序が入れ替わった場合，文の共通化や削除，並び替えが行われた場合と同様にステップ実行などができなくなる場合があります。
- 自動変数はすべて有効範囲（スコープ）が関数全体とみなされます。しかし，その変数がレジスタへ割り付けられた場合，スコープ内であっても最適化により削除され，見えなくなる可能性があります。これは，スコープ内でその変数が“局所的に”使用されている場合や，最適化の結果として局所化された場合に起こります。

例

```
void f(void)
{
    int     a;      /* 関数内で有効 */
        :
    /* アドレス 1 */
        :           /* a はアドレス 1 ～アドレス 2 の範囲でのみ使用 */
    /* アドレス 2 */
        :
}
```

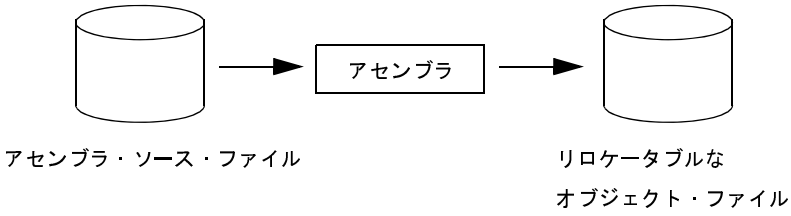
この例では，a のスコープは関数 f() 内全体です。しかし，a はアドレス 1 ～アドレス 2 間に限定して使用されています。このとき，a がレジスタに割り付けられ，最適化によりスタック・フレームから削除されると，a はアドレス 1 ～アドレス 2 の区間外では見えなくなります。この現象は，レジスタを効率的に使用するために，a の見える区間外では，a が割り付けられたレジスタに他の変数を割り付ける結果として生じます。

- コンパイル時，デバッグ情報の処理でメモリを大量に消費するため，“out of memory”となる可能性があります。
- インライン展開が行われた部分は１つのまとまりとして扱われ，ステップ実行はできません。
- ループ展開が行われた部分はループ本体部分が１つのまとまりとして扱われ，その内部をステップ実行はできません。また，本体部分のまとまりで停止する回数は，展開前のループ数ではなく，展開後のループ数となります。
- 外部変数にレジスタを割り付けた場合は指定した外部変数のデバッグ情報が削除されるため，最適化デバッグはできなくなります。

B. 2　アセンブラ

アセンブラ（as850）は，指定されたアセンブラ・ソース・ファイルをアセンブルし，リロケータブルなオブジェクト・ファイルを生成します。

図 B—4　アセンブラにおける動作の流れ



B. 2. 1　入出力ファイル

アセンブラでは，次に示したファイルを入力ファイルとして指定できます。

file.s	アセンブラ・ソース・ファイル（.s ファイルと呼ばれる）

アセンブラによって生成されるリロケータブルなオブジェクト・ファイルは，.s を .o で置き換えたファイル名となります。

なお，ファイル名は Windows で認められるものであれば指定できますが，‘@’はコマンド・オプションと判断されるため‘@’をファイル名の先頭に使用できません。また，ファイル／フォルダ名にスペースが含まれるものも使用できません。さらに，ファイルの漢字コードが EUC の場合，ファイル／フォルダ名に日本語は使用できません。

アセンブラによって生成されるリロケータブルなオブジェクト・ファイルは，その中に未解決な外部参照を含んでいる場合，それに対するリロケーションは未解決のままとなっています。

すべてのリロケーションを解決した実行可能なオブジェクト・ファイル（実行形式と呼ばれる）は，このリロケータブルなオブジェクト・ファイルをリンカでリンクすることによって生成されます。

出力リストについての詳細は，「3. 1　アセンブラ」を参照してください。

B. 2. 2　操作方法

ここでは，アセンブラの操作方法について説明します。

(1) コマンド入力による方法

アセンブラは，ca850 コマンドからデフォルトで起動されますが，次の形式により単独でも起動できます。

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>as850 [オプション]... ファイル名
      [ ]:    [ ] 内は省略できます。
      ...:    直前の [ ] 内のパターンの繰り返しができます。
```

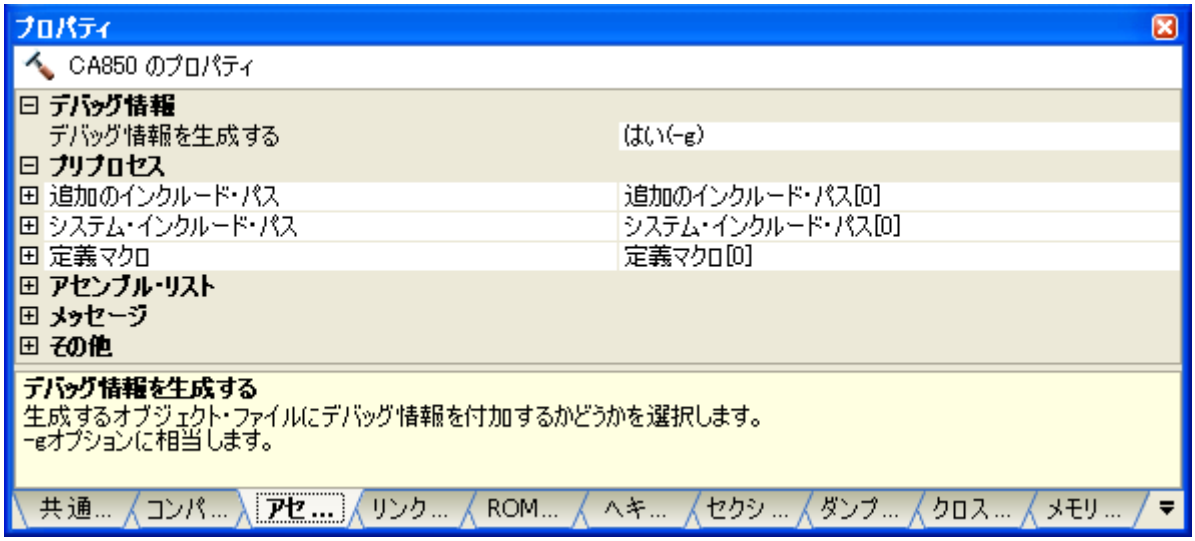
（2）CubeSuite+ でのオプション設定

CubeSuite+ からアセンブル・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネル上において，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［アセンブル・オプション］タブを選択します。

タブ上で必要なプロパティを設定することにより，各アセンブル・オプションを設定することができます。

図 B—5　プロパティ パネル：［アセンブル・オプション］タブ



B. 2. 3　オプション

ここでは，アセンブル・オプションについて説明します。

注意　アセンブラのオプションを ca850 からそのままアセンブラに渡すためには，ca850 に“-Wa”の指定が必要となります（「各モジュール」参照）。

アセンブル・オプションの分類と説明を示します。

表 B—6　アセンブル・オプション

分類	オプション	説明
ファイル	-a	アセンブル・リストを生成します。
	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。
	-l	-a オプションが指定された場合に生成されるアセンブル・リストを保存します。

分類	オプション	説明
アセンブラ	-D	定義したいマクロ名を指定します。
	-G	外部ラベルへのアクセスに対し，指定バイト以下のデータは sdata 属性，sbss 属性に割り付けられることを想定した機械語命令を生成します。
	-I	ファイル入力疑似命令で指定したファイルを，優先して検索するフォルダを指定します。
	-m	マスク・レジスタ機能を使用するという情報を持つオブジェクト・ファイルを作成します。
	-O	命令を並べ替えて，レジスタ／フラグのハザードを回避する最適化を行います。
	-v	アセンブラの実行状況の詳細を標準エラー出力に出力します。
	-w	警告メッセージのレベル，出力，抑制を指定します。
	-Xfar_jump	命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定します。
デバイス	-X256M	メモリ空間を 256M バイトとして扱います。
	-bpc	プログラマブル周辺 I/O レジスタの上位アドレスを設定します。
警告メッセージ制御	-woff	指定した番号の警告メッセージを抑制します。
その他	-cn	V850 コア共通のマジックナンバを埋め込みます。
	-cnv850e	V850Ex コア共通のマジックナンバを埋め込みます。
	-cnv850e2	V850E2 コア共通のマジックナンバを埋め込みます。
	-cpu	ターゲット・デバイスを指定します。
	-F	デバイス・ファイルの置かれているフォルダを指定します。
	-g	デバッグ情報を出力します。
	-o	アセンブルして出力するオブジェクト・ファイル名を指定します。
	-p	CPU の不具合を回避するためのコードを出力します。
	-V	アセンブラのバージョン情報を標準エラー出力に出力します。
	-zf	フラッシュ／外付け ROM 側のアセンブル処理を行います。
	@	指定ファイルをコマンド・ファイルとして扱います。

表 B―7　オプション説明でのマーク

【V850E2】	V850E2 コア専用のオプション
【V850E】	V850Ex コア専用のオプション

ファイル

アセンブラ・ソース・ファイルに対する前処理のオプションには，次のものがあります。

- -a
- +err_file
- -err_file
- -I

-a

[記述形式]

```
-a
```

- 省略時解釈

  アセンブル・リストを生成しません。

[機能説明]

- アセンブル・リストを生成します。
- -I オプションが指定されていない場合には，生成されるアセンブル・リストを標準出力に出力します。
- -O オプション（最適化オプション）と同時に指定された場合，命令の並び替えにより，アセンブル・リストの出力が，一部不正確になる場合があります。

[使用例]

- アセンブル・リストを生成します。

```
C:￥>as850 -cpu f3719 -a main.s
```

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>as850 -cpu f3719 +err_file=err main.s
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>as850 -cpu f3719 -err_file=err main.s
```

-l

[記述形式]

```
-l file
```

- 省略時解釈

  -a オプションが指定された場合，生成されるアセンブル・リストを標準出力に出力します。

[機能説明]

- -a オプションが指定された場合に生成されるアセンブル・リストをファイルに入れ，*file* をそのファイルの名前とします。
- -a オプションが指定されなかった場合は無効となります。

[使用例]

- アセンブル・リストをファイル asm に保存します。

```
C:￥>as850 -cpu f3719 -a -l asm main.s
```

アセンブラ

アセンブラ・ソース・ファイルに対するアセンブラのオプションには，次のものがあります。

-D

[記述形式]

```
-Dname[=def]
```

- 省略時解釈

  なし

[機能説明]

- 定義したいマクロ名を指定します。
- “=*def*”の部分を省略した場合，def は 1 とみなします。アセンブラ・ソース・プログラムの前に，.set *name*, *def* が記述されたものとみなします。

[使用例]

- アセンブラ・ソース・プログラムの前に，.set sample, 256 が記述されたものとします。

```
C:¥>as850 -cpu f3719 -Dsample=256 main.s
```

-G

[記述形式]

```
-Gnum
```

- 省略時解釈

  num= ∞とみなします。

[機能説明]

- 外部ラベルへのアクセスに対し，“*num* バイト以下のサイズのデータはすべて sdata 属性セクション，または sbss 属性セクションに割り付けられる”ことを想定した機械語命令を生成します。
- *num* は 10 進数で 0 ～ 32767 の範囲の整数が指定可能です。
- 疑似命令“.option sdata”で sdata が指定されたデータは，そのサイズに関係なく sdata 属性／ sbss 属性セクションに割り付けられることを想定したアセンブラ命令を生成します。
- ca850 から起動された場合，ca850 の起動時に指定された -G*num* が渡されます。

[使用例]

- 16 バイト以下のデータを .sdata セクション，または .sbss セクションに割り付けられることを想定した機械語命令を生成します。

```
C:￥>as850 -cpu f3719 -G16 main.s
```

-I

[記述形式]

```
-I dir
```

- 省略時解釈

  ソース・ファイルの置かれたフォルダ，C ソース・ファイルの置かれたフォルダ，カレント・フォルダの順で検索します。

[機能説明]

- ファイル入力疑似命令（.include ／ .binclude）で指定したファイルを，ソース・ファイルの置かれているフォルダより優先して検索するフォルダを指定します。
- 指定したフォルダを検索しても見つからない場合，およびこのオプションを省略した場合，ソース・ファイルの置かれたフォルダ，C ソース・ファイルの置かれたフォルダ，カレント・フォルダの順で検索します。

[使用例]

- ファイル入力疑似命令（.include ／ .binclude）で指定したファイルを，フォルダ D:￥head から検索します。

```
C:￥>as850 -cpu f3719 -I D:￥head main.s
```

-m

[記述形式]

```
-m
```

- 省略時解釈

  マスク・レジスタ機能は無効です。

[機能説明]

- マスク・レジスタ機能を使用するという情報を持つオブジェクト・ファイルを作成します。
- この機能を使用すると，アセンブラは r20 に 8 ビットのマスク値 0xff，r21 に 16 ビットのマスク値 0xffff が設定されているものとしてコードを出力します。
- マスク・レジスタ（r20，r21）へのマスク値の設定は，スタート・アップ・ルーチン等，ユーザ・プログラムで行う必要があります。
- マスク・レジスタ機能を利用するかどうかの判断には，利用する側で次の点を十分考慮する必要があります。
  - マスク・コードが多く出力されるプログラムであるか。
  - 32 レジスタ・モードの場合，マスク・レジスタとして使用するため，レジスタ変数用レジスタが 2 本少なくなるが，その影響はないか。
  - 32 レジスタ・モード以外の場合，マスク・レジスタとして使用するため，空きレジスタが 2 本少なくなるが，その影響はないか。

[使用例]

- マスク・レジスタ機能を使用するという情報を持つオブジェクト・ファイルを作成します。

```
C:￥>as850 -cpu f3719 -m main.s
```

-O

[記述形式]

```
-O
```

- 省略時解釈

  命令を並べ替える最適化は無効です。

[機能説明]

- 命令を並べ替えて，レジスタ／フラグのハザードを回避する最適化を行います。
- -g オプション（デバッガ用の情報出力）と同時に指定した場合，このオプションは無視され，-g オプションが有効となります。
- V850 コアのターゲット・デバイス指定，または V850 コア共通オブジェクト作成時に -p オプション（CPU の不具合回避オプション）と同時に指定した場合も，このオプションは無視され，-p オプションが有効となります。
- V850E ／ V850E1 ／ V850ES コアのターゲット・デバイス指定，または V850E ／ V850E1 ／ V850ES コア共通オブジェクト作成時に -p オプションと同時に指定した場合，このオプション，および -p オプションはともに有効となります。

[使用例]

- 命令を並べ替えて，レジスタ／フラグのハザードを回避する最適化を行います。

```
C:￥>as850 -cpu f3719 -O main.s
```

-v

[記述形式]

```
-v
```

- 省略時解釈

  なし

[機能説明]

- アセンブラの実行状況の詳細を標準エラー出力に出力します。

[使用例]

- アセンブラの実行状況の詳細を標準エラー出力に出力します。

```
C:¥>as850 -cpu f3719 -v main.s
```

-w

[記述形式]

```
-w
-wstring+
-wstring-
```

- 省略時解釈

  警告メッセージを抑止しません。

[機能説明]

- -w 指定の場合，次の場合に対し，警告メッセージを出力しません。
  - r1 をソース・レジスタ，またはデスティネーション・レジスタとして指定
  - r0 をデスティネーション・レジスタとして指定
  - マスク・レジスタ機能使用時に r20，または r21 をデスティネーション・レジスタとして指定
- -w*string+* および -w*string-* では，-w オプションの指定にかかわりなく，項目ごとに警告メッセージの出力，および抑止を指定します。“+”を付けた場合，メッセージを出力し，“-”を付けた場合，メッセージを抑止します。
- *string* に指定できる文字列は，次のとおりです。

r0	r0 をデスティネーション・レジスタとして指定
r1	r1 をソース・レジスタ，またはデスティネーション・レジスタとして指定

- “+”，および“-”を付けない場合，エラーとなります。

[使用例]

- 特定の指定の警告メッセージを出力しません。

```
C:¥>as850 -cpu f3719 -w main.s
```

- r0 をデスティネーション・レジスタとして指定した場合，警告メッセージを出力します。

```
C:¥>as850 -cpu f3719 -wr0+ main.s
```

-Xfar_jump

[記述形式]

```
-Xfar_jump
```

- 省略時解釈

  分岐命令に 22/32 を記述しない場合，通常分岐（far jump ではない）命令になります。

[機能説明]

【V850E2】

- アセンブラに，V850E2 コアのデバイスを品種指定している場合に，命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定します。
- 命令単位で変更する場合には，jarl22 / jarl32，または jr22 / jr32 と明示したものを記述してください。
- jmp 命令に関しては，-Xfar_jump オプションの影響を受けません。

[使用例]

- 命令に 22/32 を記述しない分岐命令（jarl，jr）に対して，far jump にするように指定します。

```
C:¥>as850 -cpu 3500 -Xfar_jump main.s
```

デバイス

アセンブラ・ソース・ファイルに対するアセンブラのデバイスに関するオプションには，次のものがあります。

- -X256M
- -bpc

-X256M

[記述形式]

```
-X256M
```

- 省略時解釈

  メモリ空間を 64M バイトとして扱い，アドレス解決します。

[機能説明]

【V850E】

- メモリ空間を 256M バイトとして扱います。
- このオプションは使用するチップセットにあわせて設定してください。
- V850Ex コアでは，物理アドレス空間が 256M バイト持つ場合が多く，64M を越えた 256M までの空間を使用するアプリケーションを作成する場合，このオプションを指定してください。

[使用例]

- メモリ空間を 256M バイトとして扱います。

```
C:￥>as850 -cpu f3719 -X256M main.s
```

-bpc

[記述形式]

```
-bpc=num
```

- 省略時解釈

  プログラマブル周辺 I/O レジスタの上位アドレスを 0 として扱います。

[機能説明]

- プログラマブル周辺 I/O レジスタの上位アドレスを設定します。
- *num* には，BPC レジスタの最上位ビットを除いたアドレス部分のみを指定してください。
- ターゲット・デバイスがプログラマブル周辺 I/O レジスタ機能を持ち（V850E/IA1 など），変更可能なアドレス部分（=BPC レジスタに設定する値）を設定したい場合，アプリケーションのアセンブル時に，値を確定させる必要があります。そこでこのオプションを指定すると，指定した値を使用してアセンブルします。
- このオプション指定時には必ず値を指定してください。値には 2 進数，8 進数，10 進数，16 進数を使用できます。
- 不正な値を指定した場合，および BPC レジスタに設定可能な範囲を越える値を指定した場合，警告メッセージが出力され，このオプションは無視されます。
- 設定する値はアプリケーション全体で 1 つです。ファイルごとのオプション設定で“-Xbpc”や“-bpc”を指定する場合，ファイル間で同じ値にしてください。
- プログラマブル周辺 I/O レジスタを使用しないファイルに対しては，このオプションを指定する必要はありません。
- プログラマブル周辺 I/O レジスタ機能を持たないターゲット・デバイスの場合，および V850 コア，V850Ex コア共通としてアセンブルする場合，このオプションを指定すると，警告メッセージが出力され，このオプションは無視されます。
- このオプションは，プログラマブル周辺 I/O レジスタのアドレスを，アセンブル時に確定するためのものであり，BPC レジスタに実際に値を反映させるものではありません。

  動作のためには，別途，スタート・アップ・モジュールなどで BPC レジスタに値を設定する必要があります。「CubeSuite+ V850 コーディング編」にスタート・アップ・ルーチンのサンプルが載っていますので，そちらを参照してください。また，パッケージに含まれるスタート・アップ・モジュールにも，サンプルが載っています（コメントアウトしてあります）。
- アセンブラは，このオプションを指定するか，省略してもプログラマブル周辺 I/O レジスタを参照している場合，特殊な予約セクションの .bpc セクションを出力します。

  このセクションは，リンク時のチェックのために使用されます。.bpc セクションは，情報用の特殊な予約セクションであり，メモリにロードされることはありません。したがって，通常のセクションのように，リンク・ディレクティブに記述する必要はありません。

［使用例］

- ターゲット・デバイスが V850E/IA1 の場合，下記オプション設定では，プログラマブル周辺 I/O レジスタ領域の先頭アドレスは，この値を 14 ビット左シフトした 0x48d0000 として扱われます。

```
C:￥>as850 -cpu 3116 -bpc=0x1234 main.s
```

プログラマブル周辺 I/O レジスタの先頭アドレスの変更可能部分を“0x1234”とし，この機能の使用を許可するフラグ“0x8000”を設定する場合，次の記述をスタート・アップ・モジュールに記述します。

```
mov     0x9234,r10      -- 0x1234 | 0x8000 = 0x9234
st.h    r10, BPC
```

警告メッセージ制御

警告メッセージ制御オプションには，次のものがあります。

- -woff

-woff

[記述形式]

```
-woff=num
```

- 省略時解釈

  なし

[機能説明]

- *num* で指定した番号の警告メッセージを抑制します。
- *num* には 3029，3030，3031 のいずれが指定できます。
- W3029 の警告メッセージを抑制する場合は，-woff=3029 になります。
- *num* は省略できません。

[使用例]

- W3029 の警告メッセージを抑制します。

```
C:￥>as850 -cpu f3719 -woff=3029 main.s
```

その他

その他のオプションには，次のものがあります。

- -cn
- -cnv850e
- -cnv850e2
- -cpu
- -F
- -g
- -o
- -p
- -V
- -zf
- @

-cn

[記述形式]

```
-cn
```

- 省略時解釈

  指定されたターゲット・デバイスに定義されたマジック・ナンバが埋め込まれます。

[機能説明]

- 生成するオブジェクトに，マジック・ナンバとして，V850 コア共通の“共通マジック・ナンバ”の値が埋め込まれます。これにより，V850 コア内で共通に使用可能なオブジェクトとなります。

[使用例]

- オブジェクトに V850 コア共通のマジックナンバを埋め込みます。

```
C:￥>as850 -cn main.s
```

-cnv850e

[記述形式]

```
-cnv850e
```

- 省略時解釈

  指定されたターゲット・デバイスに定義されたマジック・ナンバを設定します。

[機能説明]

【V850E】

- 生成するオブジェクトのマジック・ナンバとして，V850Ex コア共通の“共通マジック・ナンバ”の値を設定します。これにより，V850Ex コア内で共通に使用できるオブジェクトとなります。

[使用例]

- オブジェクトに V850Ex コア共通のマジックナンバを埋め込みます。

```
C:￥>as850 -cnv850e main.s
```

-cnv850e2

[記述形式]

```
-cnv850e2
```

- 省略時解釈

  指定されたターゲット・デバイスに定義されたマジック・ナンバを設定します。

[機能説明]

【V850E2】

- 生成するオブジェクトのマジック・ナンバとして，V850E2 コア共通の“共通マジック・ナンバ” の値を設定します。これにより，V850E2 コア内で共通に使用できるオブジェクトとなります。

[使用例]

- オブジェクトに V850E2 コア共通のマジックナンバを埋め込みます。

```
C:￥>as850 -cnv850e2 main.s
```

-cpu

[記述形式]

```
-cpu devicename
```

- 省略時解釈

  省略することはできません（-cn ／ -cnv850e ／ -cnv850e2 指定時を除きます）。

[機能説明]

- ターゲット・デバイスを指定します。
- このオプション指定は疑似命令“.option cpu”より優先されます。
- このオプション，または疑似命令“.option cpu”によるターゲット・デバイスの指定と -cn ／ -cnv850e ／ -cnv850e2 指定の両方がある場合は，ターゲット・デバイス固有の情報を含むコア共通オブジェクトを作成できます。
- 疑似命令“.option cpu”による指定，または -cn ／ -cnv850e ／ -cnv850e2 オプション指定がない場合，このオプションを省略するとアセンブルを中止します。

[使用例]

- ターゲット・デバイスとして，UPD70F3719 を指定します。

```
C:￥>as850 -cpu f3719 main.s
```

-F

[記述形式]

```
-F devpath
```

- 省略時解釈

  デバイス・ファイルの置かれているフォルダを標準フォルダとします。

[機能説明]

- デバイス・ファイルの置かれているフォルダを指定します。

[使用例]

- デバイス・ファイルの置かれているフォルダを D:￥dev から検索します。

```
C:￥>as850 -cpu f3719 -F D:￥dev main.s
```

-g

[記述形式]

```
-g
```

- 省略時解釈

  ソース・デバッガ用のシンボル情報を出力しません。

[機能説明]

- デバッグ情報を出力します。
- デバッガでアセンブラ・ソース・デバッグを行いたい場合など，プログラムをデバッグする際に指定してください。
- 最適化（-O）オプションを同時に指定した場合，ソース・ファイル中にデバッグ情報用セクションがあれば，このオプションは無視されます。デバッグ情報用のセクションがなければ，最適化（-O）が無視され，このオプションが有効になります。つまり，デバッグ情報がなければ，このオプションが優先されます。

[使用例]

- デバッグ情報を出力します。

```
C:￥>as850 -cpu f3719 -g main.s
```

-o

[記述形式]

```
-o ofile
```

- 省略時解釈

  オブジェクト・ファイル名は，ソース・ファイルの拡張子 .s を .o に置き換えたものとなります。

[機能説明]

- アセンブルして出力するオブジェクト・ファイル名を *ofile* とします。

[使用例]

- アセンブルして出力するオブジェクト・ファイル名を test.o とします。

```
C:¥>as850 -cpu f3719 -o test.o main.s
```

-p

[記述形式]

```
-p[num]
```

- 省略時解釈

  CPU の不具合を回避するためのコードを出力しません。

[機能説明]

- CPU の不具合を回避するためのコードを出力します。
- *num* には，出力するコードの種類（1 ～ 10，および 4a）を指定します。1 ～ 4 と 4a は V850 コアに，5 ～ 10 は V850E/ES コアに有効です。
- *num* を省略した場合，デバイス・ファイルから判断し，次の種類のコードを出力します。
  - ターゲット・デバイスが V850E/ES コア，またはアセンブル・オプションで（-cnv850e）マジック・ナンバに“V850E/ES コア共通”を指定した場合，5 ～ 10 のコードを出力。

    ターゲット・デバイスが V850 デバイスの場合，1 ～ 3 のコードを出力。
  - アセンブル・オプション（-cn）でマジック・ナンバに“V850 コア共通”を指定した場合，1 ～ 3，5 ～ 10 のコードを出力。

  このオプションにより出力されるコードについての詳細は，「（2）CPU 不具合の回避オプション」を参照してください。

[使用例]

- CPU の不具合を回避するための 4a のコード（“ロード命令（ld.[b|h|w] ／ sld.[b|h|w]）＋ロード・ストア命令（ld.[b|h|w] ／ sld.[b|h|w] ／ sst.[b|h|w] ／ st.[b|h|w]）”の組み合わせに対し，最初のロード命令の直後に nop 命令を挿入する）を出力します。

```
C:¥>as850 -cpu f3719 -p4a main.s
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- アセンブラのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- アセンブラのバージョン情報を標準エラー出力に出力します。

```
C:¥>as850 -cpu f3719 -V main.s
```

-zf

[記述形式]

```
-zf
```

- 省略時解釈

  フラッシュ／外付け ROM 再リンク機能を使用したアセンブラ・ソース・ファイルに対し，ブート／内蔵 ROM 側のアセンブル処理を行います。

[機能説明]

- フラッシュ／外付け ROM 再リンク機能を使用した .ext_func 疑似命令の記述があるアセンブラ・ソース・ファイルに対し，フラッシュ／外付け ROM 側のアセンブル処理を行います。
- フラッシュ／外付け ROM 再リンク機能を使用しないアセンブラ・ソース・ファイルに対し，このオプションを指定する必要はありません。指定した場合，機能が変化することはありません。また，警告メッセージを出力しません。
- フラッシュ／外付け ROM 再リンク機能の詳細は，「B. 3. 3　ブート－フラッシュ再リンク機能」を参照してください。

[使用例]

- フラッシュ／外付け ROM 再リンク機能を使用した .ext_func 疑似命令の記述があるアセンブラ・ソース・ファイルに対し，フラッシュ／外付け ROM 側のアセンブル処理を行います。

```
C:¥>as850 -cpu f3719 -zf main.s
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は，「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>as850 @command main.s
```

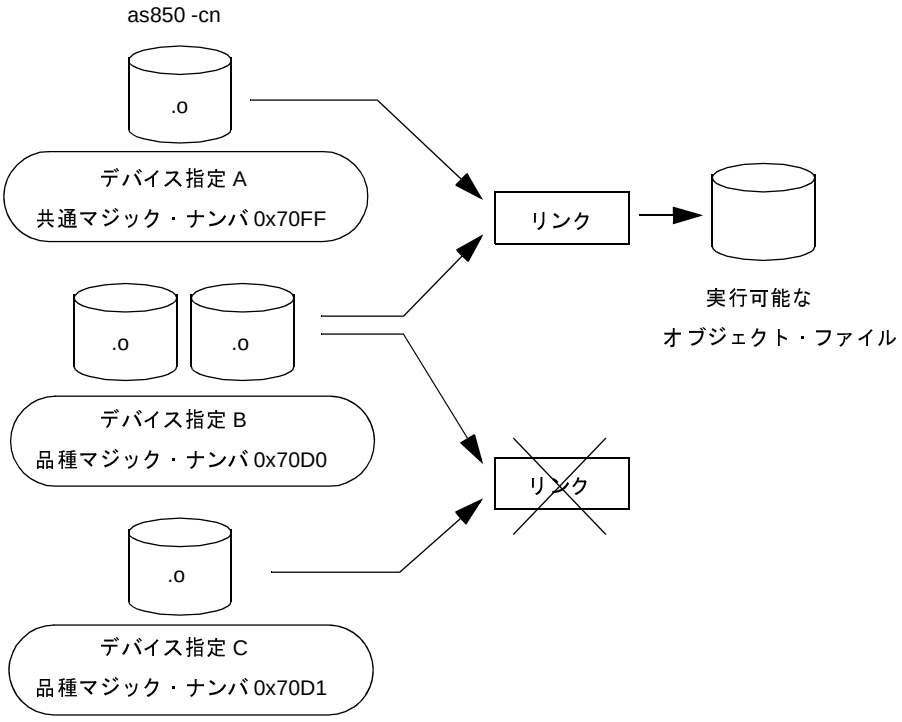
B. 2. 4　注意事項

(1) マジック・ナンバ

アセンブラが生成するオブジェクトには，そのオブジェクトが対象としているデバイス情報が自動的に埋め込まれます。この情報を“マジック・ナンバ”と呼びます。オブジェクトが，あるデバイス品種のみを対象としている場合，「品種固有のマジック・ナンバ」が埋め込まれ，コア全体を対象としている場合，「共通のマジック・ナンバ」が埋め込まれます。

アセンブラで -cn オプションを指定してアセンブルされたオブジェクトは，共通マジック・ナンバを持つため，同じコア内であれば，異なるデバイス品種を指定されたオブジェクトとリンク可能です（リンカによるリンク時にエラーとなりません）。このため，-cn オプションを指定して生成したオブジェクトは，コア内で共通して使用可能なオブジェクトとなります。

図 B―6　アセンブラにおける共通オブジェクト作成イメージ

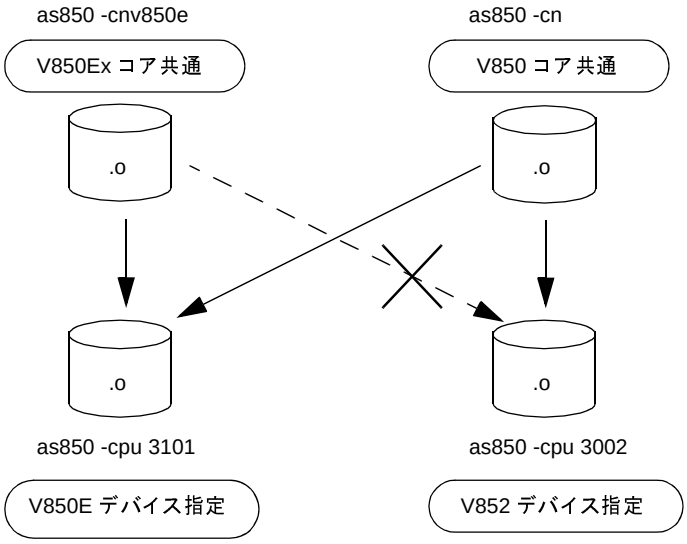


(a) 注意事項

- コア共通のマジック・ナンバと，品種固有のマジック・ナンバは，各デバイス・ファイルに定義され，対応付けられています。アセンブラはデバイス・ファイルを参照し，マジック・ナンバを埋め込みます。
- 品種固有の周辺機能レジスタなどを操作するオブジェクト・ファイルは，コア共通にしないでください。
- -cpu オプション，または .option 疑似命令によってターゲット・デバイスを指定したうえで，-cn ／ -cnv850e ／ -cnv850e2 オプションによる指定を行うと，ターゲット・デバイス固有の情報を含むコア共通オブジェクトが作成できます。
  ただし，ターゲット・デバイスと異なるデバイス固有情報を持たせたオブジェクトは正常に動作しないので，目的のターゲット・デバイスに流用可能なデバイス固有情報であることをあらかじめ確認しておいてください。

- V850Ex コアは，V850 コアに対して上位互換です。V850 コアで使用していたソース・ファイルを，V850Ex コアで使用できます。その場合，“-cn”，または“-cnv850e”オプションを指定してオブジェクトを作成してください。“-cn”で作成された V850 コア共通オブジェクトは，V850Ex コア・オブジェクトともリンク可能です。一方，“-cnv850e”で作成したオブジェクトは，V850 コア・オブジェクトとはリンクできません。
- V850E2 コアは，V850 ／ V850Ex コアに対して上位互換です。V850 ／ V850Ex コアで使用していたソース・ファイルを，V850E2 コアで使用できます。その場合，“-cn”，または“-cnv850e”オプションを指定してオブジェクトを作成してください。“-cn”で作成された V850 ／ V850Ex コア共通オブジェクトは，V850E2 コア・オブジェクトともリンク可能です。一方，“-cnv850e”で作成したオブジェクトは，V850 ／ V850Ex コア・オブジェクトとはリンクできません。

図 B—7 アセンブラにおける CPU コアによる互換関係の例（V850Ex コアと V850 コアの場合）



(2) CPU 不具合の回避オプション

C コンパイラでは，V850 コア／ V850E/ES コア CPU の不具合を回避するため，ca850 において -Xv850patch オプション，アセンブラにおいて -p オプションを提供しています。ca850 からアセンブラを起動する場合，ca850 の -Xv850patch オプションを指定すると，（ca850 が出力するアセンブラ・ソース・ファイルに対して）アセンブラで同じ *num* を持つ -p オプションが自動的に設定されます。

num には，出力するコードの種類（1 ～ 10，および 4a）を指定します。1 ～ 4 と 4a は V850 コアに，5 ～ 10 は V850E/ES コア有効です。*num* を省略した場合，デバイス・ファイルから判断し，次の種類のコードを出力します。

- ターゲット・デバイスが V850E/ES コア，またはアセンブル・オプション（-cnv850e）でマジック・ナンバに“V850E/ES コア共通”を指定した場合，5 ～ 10 のコードを出力。
- ターゲット・デバイスが V850 コアの場合，1 ～ 4 と 4a のコードを出力。
- アセンブル・オプション（-cn）でマジック・ナンバに“V850 コア共通”を指定した場合，1 ～ 10，および 4a のコードを出力。

注意事項を以下に示します。

- 使用している CPU に該当する不具合があるかどうかは，CPU 添付の資料を参照してください。
- V850 コアのターゲット・デバイス指定，または V850 コア共通オブジェクト作成時に -p オプションとアセンブラの最適化（-O）オプションを同時に指定した場合，-p が優先され，-O は無視されます。
- V850E/ES コアのターゲット・デバイス指定，または V850E/ES コア共通オブジェクト作成時に -p オプションとアセンブラの最適化（-O）オプションを同時に指定した場合，-p，および -O はともに有効になります。
- 不具合の発生するコード・パターンが，異なるセクションにまたがっている場合，このオプションの機能は無効です。
- -Xv850patch=11 オプションのみ ca850 で処理します。
- CPU コアと，回避オプションに対応する不具合は，次のとおりです（保守品と廃品種を除く，最新バージョンの µPD(F)703xxx の場合）。
  なお，使用している CPU に該当する障害であるかどうかは，CPU の資料を参照してください。

表 B—8 CPU コアと -p オプションに対応する不具合

CPU コア	-p1	-p2	-p3	-p4	-p4a	-p5	-p6	-p7	-p8	-p9	-p10
V850 コア	○	○	○	○	○	—	—	—	—	—	—
V850E/MS1	—	—	—	—	—	○	—	—	×	—	×
V850E1 コア	—	—	—	—	—	—	○	○	—	○	—
V850ES コア	—	—	—	—	—	—	—	—	—	—	—
V850E2 コア	—	—	—	—	—	—	—	—	—	—	—

備考 × ：該当
○ ：修正済み（保守品と廃品種を除く，最新バージョンの µPD(F)703xxx の場合）
— ：非該当

num の種類と意味は，次のとおりです。

なお，命令，およびレジスタについては，各デバイスのアーキテクチャ編のユーザーズ・マニュアルを参照してください。

（a）1（-Xv850patch=1 → -p1）

“ld.w 命令＋（st.[b|h|w] ／ sst.[b|h|w] ／ ld.[b|w] ／ sld.[b|w] 命令）＋分岐命令”の組み合わせに対し，最初の ld.w の直後に nop 命令を挿入する。

例

```
ld.w                    ld.w
sst.w                   nop
jarl                    sst.w
                        jarl
```

（b）2（-Xv850patch=2 → -p2）

“ld.w ／ sld.w ／ st.w ／ sst.w 命令＋分岐命令”の組み合わせに対し，ロード／ストア命令と分岐命令の間に nop 命令を挿入する。

例

ld.w jarl	ld.w nop jarl

なお，*num*=1 のパターンと同時に処理する場合，*num*=2 のパターンを先に検索して処理します。無駄な nop 命令を挿入することはありません。

（c）3（-Xv850patch=3 → -p3）

reti 命令の直前に，対応する割り込みの制御レジスタに対し，clr1 命令を挿入する。

例

reti	clr15, P0IC0 reti

（d）4（-Xv850patch=4 → -p4）

“ロード命令（ld.[b|h|w] ／ sld.[b|h|w]）＋ロード・ストア命令（ld.[b|h|w] ／ sld.[b|h|w] ／ sst.[b|h|w] ／ st.[b|h|w]）”の組み合わせに対し，最初のロード命令の直後に nop 命令を挿入する（入力ファイル中で周辺 I/O レジスタをアクセスしている場合に挿入する）。

例

ld.w ld.w	ld.w nop ld.w

（e）4a（-Xv850patch=4a → -p4a）

“ロード命令（ld.[b|h|w] ／ sld.[b|h|w]）＋ロード・ストア命令（ld.[b|h|w] ／ sld.[b|h|w] ／ sst.[b|h|w] ／ st.[b|h|w]）”の組み合わせに対し，最初のロード命令の直後に nop 命令を挿入する（周辺 I/O レジスタ・アクセスの有無にかかわらず挿入する）。

例

ld.w ld.w	ld.w nop ld.w

-p4 は，入力ファイル中で周辺 I/O レジスタ・アクセスがあった場合，4 のパッチを当てる。

-p4a は，周辺 I/O レジスタ・アクセスの有無に関わらず 4 のパッチを当てる。

（f）5（-Xv850patch=5 → -p5）

乗算命令に対し，無条件で直後に nop 命令を挿入する。

例

mulh jarl	mulh nop jarl

（g）6（-Xv850patch=6 → -p6）

“ロード命令（ld.[b|h|w|bu|hu] ／ sld.[b|h|w]）＋ jr ／ jarl ／ jcond（bcond）”の組み合わせに対し，ロード命令の直後に nop 命令を挿入する。

例

sld.bu jarl	sld.bu nop jarl

（h）7（-Xv850patch=7 → -p7）

callt 命令の直後に nop 命令を挿入する。また，switch 命令，および reti 命令の直前に“mov r31, r0”命令を挿入する。

例

switch	mov r31, r0 switch

（i）8（-Xv850patch=8 → -p8）

連続する sld 命令の間に nop 命令を挿入する。

例

sld.b sld.b	sld.b nop sld.b

(j) 9 (-Xv850patch=9 → -p9)

次の (A), (B), (C) に該当する命令が連続で存在した場合，sld.b 命令の直後に nop 命令を挿入する。

例

add sld and	add … (A) sld.b … (B) nop and … (C)

- (A)

2 バイト長の mov，not，satsubr，satsub，satadd，zxb，zxh，sxh，or，xor，and，subr，sub，add，shr，sar，shl 命令のうち r0, r30 以外にライト・バックする命令

例

```
add     0x1, r10
```

ただし，LABEL，式，または定義が参照より後にある .set シンボルを記述し，上記命令に命令展開される可能性のある命令が含まれます。
次の例では CPU の不具合パターンにはあたりませんが，パッチを当てる対象となります。

例

```
addi    SYM, r10, r10
.set    SYM, 0x123
```

- (B)

(A) の命令がライト・バックするレジスタと異なるレジスタにライト・バックする sld 命令

例

```
sld.b   %LABEL, r11
```

- (C)

(A) の命令がライト・バックするレジスタ値をロードする命令

例

```
add     r11, r10
```

ただし，LABEL，式，または定義が参照より後ろにある .set シンボルを記述し，(A) の命令でライト・バックしたレジスタ値をロードする命令を記述した場合が含まれます。

例

```
        addi    LABEL2-LABEL1, r10, r12
LABEL1:
        --（中略）
LABEL2:
```

この例では，LABEL2 と LABEL1 の相対値が 16 ビットで表現できる範囲を越えた場合，次のように命令展開されます。

```
mov     LABEL2-LABEL1, r12
and     r10, r12
```

ここで，（B）の命令の直後は mov 命令となり，r10 の値はロードされません。つまり，CPU のバグ・パターンには当てはまりませんが，パッチを当てる対象となります。

（k）10（-Xv850patch=10 → -p10）

“ストア命令（sst.[b|h|w] ／ st.[b|h|w]）＋ jcond（bcond）”の組み合わせに対し，ストア命令の直後に nop 命令を挿入する。

例

sst.b br	sst.b nop br

（l）*num* 指定なし（-Xv850patch → -p）

デバイス・ファイルより判断し，1 ～ 3，5 ～ 10 の組み合わせの各コードを出力する（前記参照）。

なお，パッチを当てる必要のないオブジェクト生成時にこのオプションを指定した場合，パッチを当てません。生成オブジェクトと各オプションの対応は次のとおりです。

表 B—9　生成オブジェクトと -p オプションの対応

生成するオブジェクト	-p1	-p2	-p3	-p4	-p4a	-p5	-p6	-p7	-p8	-p9	-p10
V850 デバイス固定	○	○	○	○	○	×	×	×	×	×	×
V850E/ES デバイス固定	×	×	×	×	×	○	○	○	○	○	○
V850E2 デバイス固定	×	×	×	×	×	×	×	×	×	×	×
V850 コア共通	○	○	○	○	○	○	○	○	○	○	○
V850E/ES コア共通	×	×	×	×	×	○	○	○	○	○	○
V850E2 コア共通	×	×	×	×	×	×	×	×	×	×	×

備考　○　：パッチを当てる

×　：パッチを当てない

B. 3 リンカ

アプリケーション・プログラムは，一般的に複数のソース・ファイルに分けられてコーディングされます。C 言語で書かれたソースはコンパイラ（ca850）／アセンブラ（as850）を，アセンブリ言語で書かれたソースはアセンブラ（as850）を起動し，オブジェクト・ファイルを出力します。

これらのオブジェクト・ファイル群を，リンク・ディレクティブ，およびデバイス・ファイルの情報に従ってアドレス解決し，実行可能な 1 つのオブジェクト・ファイル，つまり，ロード・モジュール・ファイルを生成するのが“リンカ（ld850）”です。

リンカがオブジェクト・ファイル群をリンクするとき，未解決な外部参照があった場合は，指定されたアーカイブ・ファイル（ライブラリ・ファイル）を検索して解決しようとします。そして解決に必要なオブジェクト・ファイルのみをリンクし，実行可能なオブジェクト・ファイルを生成します。また，-r オプションによってリロケータブルなオブジェクト・ファイルを生成することができます。

図 B—8 リンカにおける動作の流れ

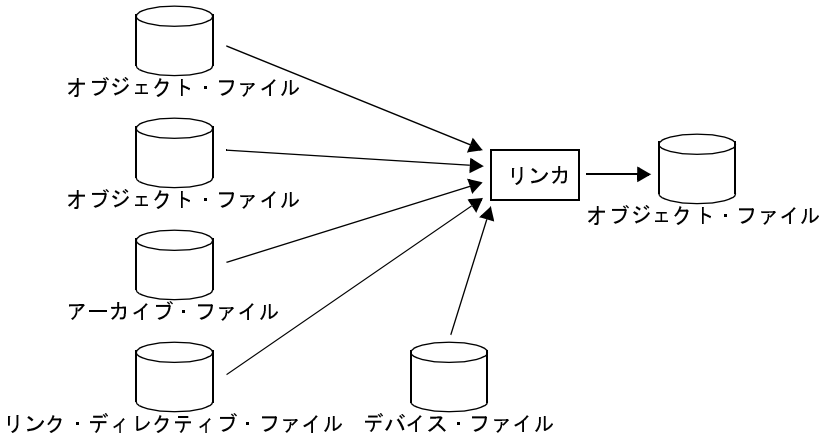
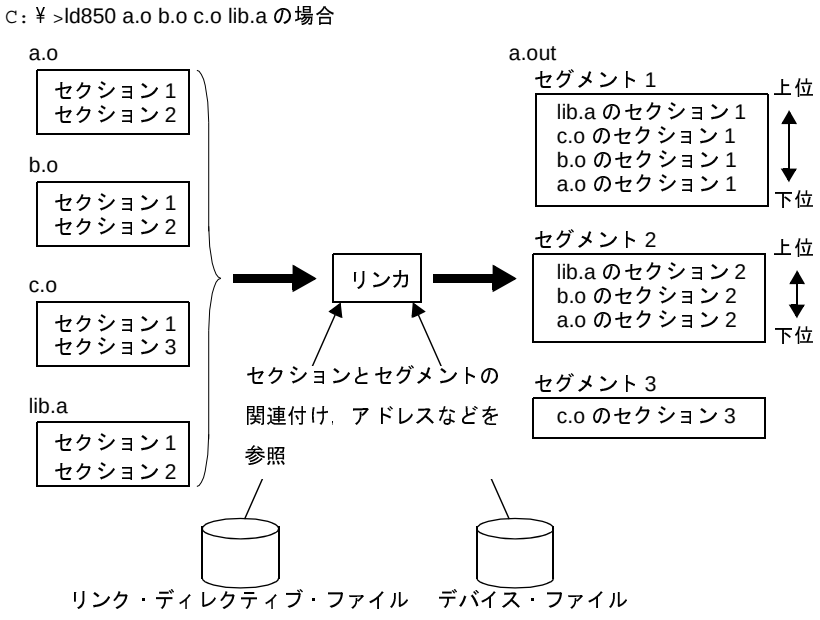


図 B—9 リンカにおける動作のイメージ例



なお，ca850 はドライバとして as850，リンカを内部で起動します。

ca850 を起動すると，ロード・モジュールまで生成することができます。つまり，as850 やリンカの起動を意識する必要がありません。

図 B—10　一括処理

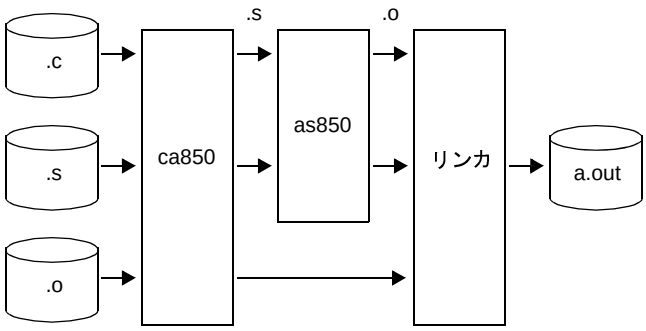
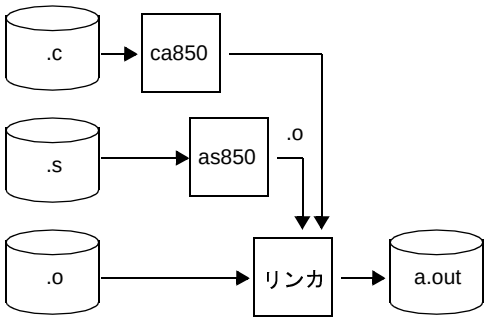


図 B—11　分割処理

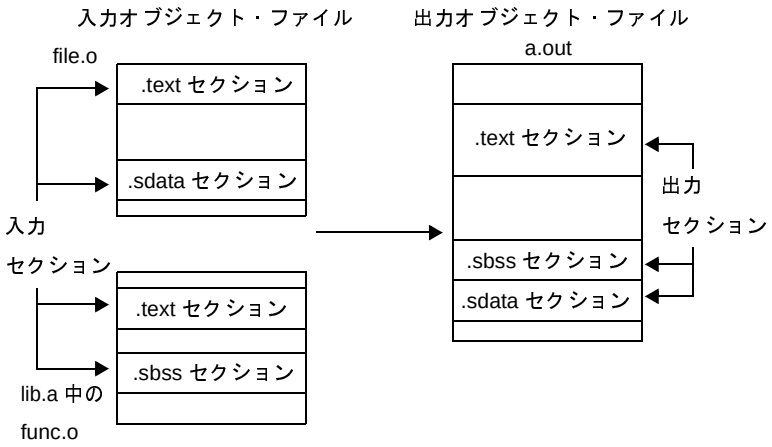


（1）リンクの手順

リンクの手順を次に示します。

（a）指定されたオブジェクト・ファイルに含まれるセクション（入力セクション）をリンク・ディレクティブ，およびデバイス・ファイルに従って結合し，生成されるオブジェクト・ファイルを構成する出力セクションを作成する（詳細については，「CubeSuite+ V850 コーディング編」を参照してください）。

図 B—12　出力セクションの作成

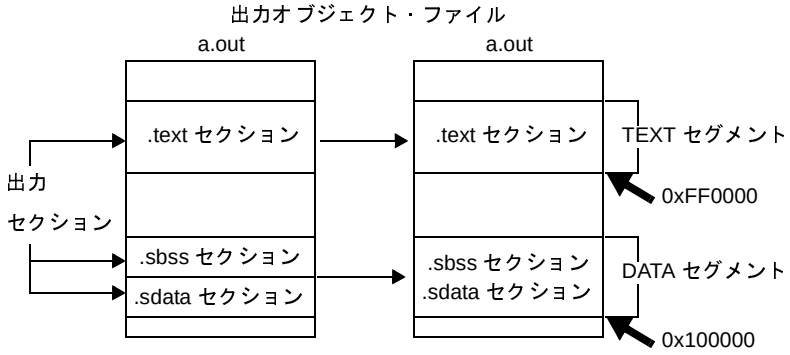


(b)（a）のステップにおいて作成された出力セクションをリンク・ディレクティブに従って結合し，セグメント[注]を作成する。

注　プログラムをメモリにロードする際の最小単位で，生成されるオブジェクト・ファイルのプログラム・ヘッダに反映されます。

(c)（b）のステップにおいて作成されたセグメントをリンク・ディレクティブ，およびデバイス・ファイルに従ってターゲット・マシンのメモリ空間に割り付ける。

図 B—13　メモリ空間への割り付け



(d) 出力セクション内の未解決な外部参照を解決する。

(e) リンク・ディレクティブ内のシンボル・ディレクティブに従い，次の 3 種類のシンボルを生成する[注]。

- テキスト・ポインタ（tp）に設定する値を持つテキスト・ポインタ・シンボル
- グローバル・ポインタ（gp）に設定する値を持つグローバル・ポインタ・シンボル
- エレメント・ポインタ（ep）に設定する値を持つエレメント・ポインタ・シンボル

注 これらのシンボルは（たとえば，スタート・アップ・モジュールにおいて），C コンパイラによって生成されたコードを実行する前にテキスト・ポインタ（tp），グローバル・ポインタ（gp），およびエレメント・ポインタ（ep）を適切な値に設定する場合に用いることができます。
エレメント・ポインタ値はユーザが設定することもできますが，省略した場合，エレメント・ポインタ・シンボルには，リンカがターゲット・デバイス固有の値（内蔵 RAM の先頭アドレス）を，指定したデバイス・ファイルより読み込み，設定します。

(f) 予約シンボルを生成する。予約シンボルには次のものがあります。

- 各出力セクションの先頭アドレス
- 各出力セクションの終端を越える最初の（4 バイトで整列された）アドレス
- 生成された実行可能なオブジェクト・ファイルの終端を越える最初の（4 バイトで整列された）アドレス

予約シンボルの詳細は，「(3) 予約シンボル」を参照してください。

B. 3. 1 操作方法

ここでは，リンカの操作方法について説明します。

(1) コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:¥>ld850 [オプション]... ファイル名 [ファイル名，またはオプション]...
      [ ]:    [ ] 内は省略できます。
      ...:    直前の [ ] 内のパターンの繰り返しができます。
```

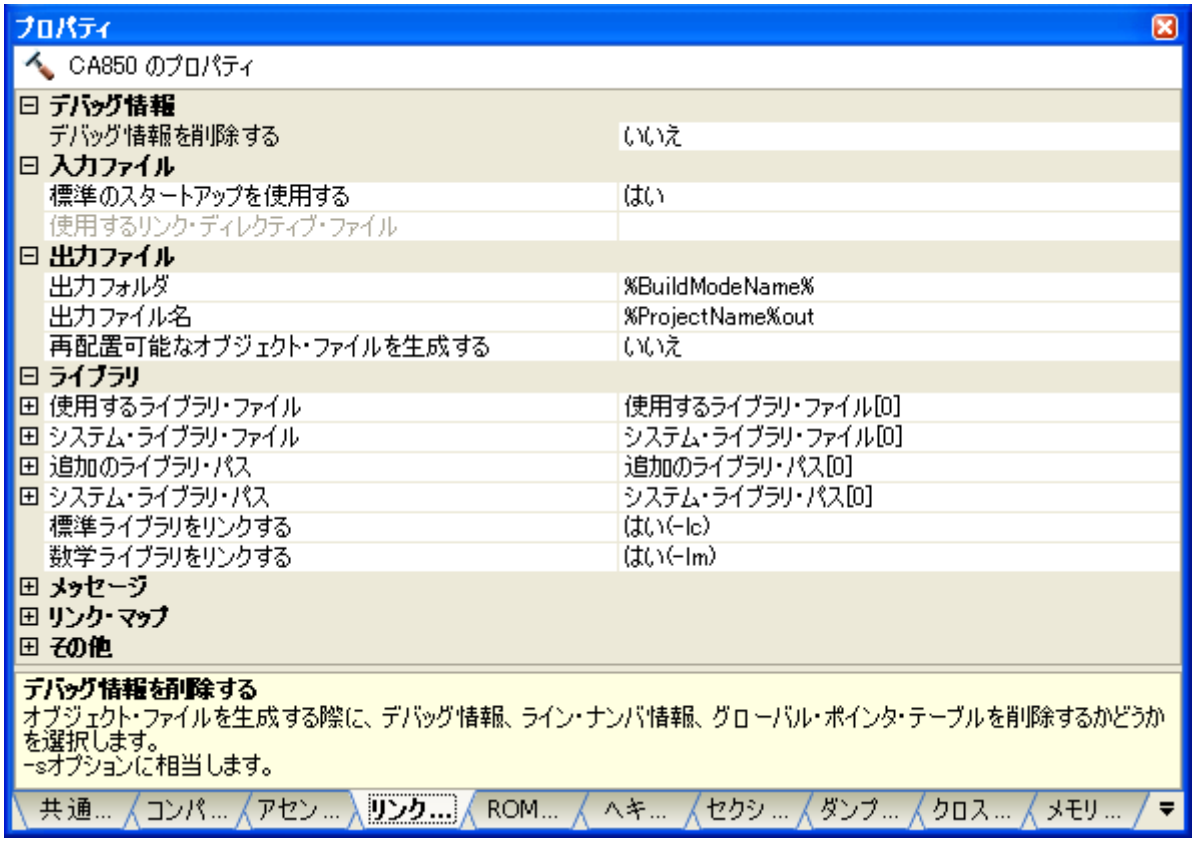
(2) CubeSuite+ でのオプション設定

CubeSuite+ からリンク・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネルにおいて，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［リンク・オプション］タブを選択します。

タブ上で各プロパティを設定することにより，対応するリンク・オプションを設定することができます。

図 B—14　プロパティ パネル：［リンク・オプション］タブ



B. 3. 2　オプション

ここでは，リンク・オプションについて説明します。

リンク・オプションの分類と説明を示します。

表 B—10　リンク・オプション

分類	オプション	説明
入力ファイル	-D	指定リンク・ディレクティブ・ファイル内のリンク・ディレクティブに従ってリンクします。
	-Xolddir	リンク・ディレクティブのフォーマットの，旧版との互換性を選択します。
出力ファイル	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。
	-o	生成するオブジェクト・ファイル名を指定します。
	-m	入力セクション，出力セクションのメモリ空間への割り付け状態を示すリンク・マップを出力します。
	-mo	入力セクション，出力セクションのメモリ空間への割り付け状態を示すリンク・マップを CA850 Ver.2.60 以前の旧形式で出力します。

分類	オプション	説明
ライブラリ	-L	-lオプションによって指定されたアーカイブ・ファイル（ライブラリ・ファイル）を，指定フォルダ，標準フォルダの順で探します。
	-lc	コンパイラの標準ライブラリ（libc.a）をリンクします。
	-lm	コンパイラの数学ライブラリ（libm.a）をリンクします。
	-l	未解決な外部シンボルの参照を解決する際，指定アーカイブ・ファイルを参照します。
フラッシュ	-ext_table	指定される値を分岐テーブル先頭アドレス値として，フラッシュ／外付けROM再リンク機能用のオブジェクト・ファイルを生成します。
	-zf	指定したオブジェクト・ファイルをブート領域側のオブジェクト・ファイルとして，フラッシュ側オブジェクト・ファイルを生成します。
デバイス	-X256M	メモリ空間を256Mバイトとして扱います。
	-Xsid	フラッシュ・メモリ搭載デバイスの“セキュリティID”を設定します。
	-Xob=none	デフォルトで生成されるオプション・バイトを抑止しします。

分類	オプション	説明
リンカ	-A	ca850，および as850 に対して指定する [sdata ／ sbss のデータ配置] オプションにおいて，目安として用いることのできる情報を標準出力に出力します。
	-B	2 パス・モードでリンクを行います。
	-E	リロケーション処理において，不正箇所があった場合，エラーとはせず，警告メッセージを出力してリンクを続行します。
	-M	多重定義されたすべての外部シンボルに対してメッセージを出力し，リンクの処理を中止します。
	-T	外部シンボルのリンクの際，サイズ，および整列条件のチェックを行いません。
	-Ximem_overflow=warning	内蔵 ROM ／ RAM をオーバーフローした際のチェックの制御を行います。
	-e	指定シンボル値を生成されるオブジェクト・ファイルのエントリ・ポイント・アドレス値とします。
	-f	生成されるオブジェクト内のセクション間のアライン・ホールのフィリング値を指定します。
	-mc	マスク・レジスタ機能を使用しているファイルと，使用していないファイルが混在していないかどうかチェックします。
	-rc	異なるレジスタ・モードが混在している場合に詳細な情報を出力します。
	-rescan	-l オプションで指定したライブラリ・ファイルの再参照を行います。
	-rom_less	内蔵 ROM 領域に対する配置に対して，チェックを行いません。
	-s	デバッグ情報，ライン・ナンバ情報，およびグローバル・ポインタ・テーブルを取ったオブジェクト・ファイルを生成します。
	-t	未定義外部シンボルのリンクにおいて，シンボルのサイズ，および整列条件のチェックを行いません。
	-v	リンカの実行状況の詳細を出力します。
	-w	警告メッセージを出力しません。

分類	オプション	説明
その他	-F	デバイス・ファイルを，指定フォルダから探します。
	-V	バージョン情報を標準エラー出力に出力します。
	-cpu	指定されたターゲット・デバイスのデバイス・ファイルを読み込みます。
	-fc	旧関数呼び出しと現バージョンの呼び出し仕様が混在していないかチェックします。
	-help	オプションの説明を標準エラー出力に出力します。
	-mask_reg	マスク・レジスタ機能用のライブラリを参照します。
	-r	リロケータブル（再配置可能）なオブジェクト・ファイルを生成します。
	-ro	リロケータブル（再配置可能）なオブジェクト・ファイルを旧マッピング方式（CA850 Ver.2.30 以前）で生成します。
	-reg	対応するレジスタ・モード用のライブラリを参照します。
	@	指定ファイルをコマンド・ファイルとして扱います。

表 B—11　オプション説明でのマーク

【V850E2】	V850E2 コア専用のオプション
【V850E】	V850Ex コア専用のオプション

入力ファイル

入力ファイルに関するオプションには，次のものがあります。

- -D
- -Xolddir

-D

[記述形式]

```
-D dfile
```

- 省略時解釈

  デフォルトのリンク・ディレクティブを用います。

[機能説明]

- リンク・ディレクティブ・ファイル *dfile* 内のリンク・ディレクティブに従ってリンクします。
- *dfile* の長さは，パスを指定する部分を含んだ長さで 127 文字以内，パスを指定する部分を含んでいない長さで 14 文字以内にしてください。
- 拡張子も必要です。推奨する拡張子は“.dir”です。
- リンク・ディレクティブ・ファイルについての詳細は，「CubeSuite+ V850 コーディング編」を参照してください。

[使用例]

- リンク・ディレクティブ・ファイル link.dir 内のリンク・ディレクティブに従ってリンクします。

```
C:¥>ld850 -D link.dir main.o
```

-Xolddir

[記述形式]

```
-Xolddir[=version]
```

- 省略時解釈

  なし

[機能説明]

- リンク・ディレクティブのフォーマットの，旧版との互換性を選択します。
- *version* には“V240”，“V250”，“V260”を指定できます。*version* 省略時には“V240”が指定されたものとして扱います。
- このオプションを指定しなかった場合は，最新のリンク・ディレクティブのフォーマットに対応します。

V240 指定時	セクション優先配置機能 OFF，セグメント・ソート OFF（CA850 Ver.2.40 相当）
V250 指定時	セクション優先配置機能 ON，セグメント・ソート OFF（CA850 Ver.2.50 相当）
V260 指定時	セクション優先配置機能 ON，セグメント・ソート ON（CA850 Ver.2.60 以降相当）

[使用例]

- リンク・ディレクティブのフォーマットを CA850 Ver.2.40 相当とします。

```
C:￥>ld850 -Xolddir=V240 main.o
```

出力ファイル

出力ファイルに関するオプションには，次のものがあります。

- +err_file
- -err_file
- -o
- -m
- -mo

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>ld850 +err_file=err main.o
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>ld850 -err_file=err main.o
```

-o

[記述形式]

```
-o ofile
```

- 省略時解釈

  生成するオブジェクト・ファイル名に a.out が指定されたものとみなします。

[機能説明]

- 生成するオブジェクト・ファイル名を *ofile* とします。

[使用例]

- 生成するオブジェクト・ファイル名を test.out とします。

```
C:¥>ld850 -o test.out main.o
```

-m

[記述形式]

```
-m[=mapfile]
```

- 省略時解釈

  リンク・マップを出力しません。

[機能説明]

- 入力セクション，出力セクションのメモリ空間への割り付け状態を示すリンク・マップを *mapfile* に出力します。
- *mapfile* を省略した場合，標準出力に出力します。
- リンク・マップに関しての詳細は「3. 2　リンカ」を参照してください。

[使用例]

- 入力セクション，出力セクションのメモリ空間への割り付け状態を示すリンク・マップを標準出力に出力します。

```
C:￥>ld850 -m main.o
```

-mo

[記述形式]

```
-mo[=mapfile]
```

- 省略時解釈

  リンク・マップを出力しません。

[機能説明]

- 入力セクション，出力セクションのメモリ空間への割り付け状態を示すリンク・マップを CA850 Ver.2.60 以前の旧形式で *mapfile* に出力します。
- *mapfile* を省略した場合，標準出力に出力します。
- リンク・マップに関しての詳細は，「3. 2　リ ン カ」を参照してください。

[使用例]

- 入力セクション，出力セクションのメモリ空間への割り付け状態を示すリンク・マップを CA850 Ver.2.60 以前の旧形式で標準出力に出力します。

```
C:¥>ld850 -mo main.o
```

ライブラリ

ライブラリに関するオプションには，次のものがあります。

- -L
- -lc
- -lm
- -l

-L

[記述形式]

```
-Ldir
```

- 省略時解釈

  -l オプションによって指定されたアーカイブ・ファイル（ライブラリ・ファイル）を標準フォルダから検索します。

[機能説明]

- このオプションとともに（コマンド・ラインからの場合このオプションのあとに）-l オプションが指定された場合，-l オプションによって指定されたアーカイブ・ファイル（ライブラリ・ファイル）を，フォルダ dir，標準フォルダの順で探します。

  このオプション以降に指定した -l オプションが対象となります。
- リンカは，CubeSuite+ のインストールされたフォルダから CubeSuite+￥CA850￥V*x.xx* 注￥lib850 の位置にあるフォルダ，および lib850￥r*XY* の位置にあるフォルダ（*XY*=[32 | 26 | 22]）をライブラリに対する標準フォルダとして扱います。

  注　V*x.xx* は C コンパイラのバージョンです。

[使用例]

- リンクするコンパイラの標準ライブラリ (libc.a) をフォルダ D:￥lib，標準フォルダの順で探します。

```
C:￥>ld850 -LD:￥lib main.o -lc
```

-lc

[記述形式]

```
-lc
```

- 省略時解釈

  コンパイラの標準ライブラリ（libc.a）をリンクしません。

[機能説明]

- コンパイラの標準ライブラリ（libc.a）をリンクします。

[使用例]

- コンパイラの標準ライブラリ（libc.a）をリンクします。

```
C:¥>ld850 main.o -lc
```

-lm

[記述形式]

```
-lm
```

- 省略時解釈

  コンパイラの数学ライブラリ（libm.a）をリンクしません。

[機能説明]

- コンパイラの数学ライブラリ（libm.a）をリンクします。
- 数学ライブラリは，標準ライブラリ内の関数も参照するため，-lc オプションを同時に設定する必要があります。
- C コンパイラで提供している数学ライブラリは，標準ライブラリの libc.a ファイルを参照しています。したがって，コマンド・ラインからの起動では，標準ライブラリの参照指定“-lc”は，数学ライブラリの参照指定“-lm”より後ろに指定してください。

[使用例]

- コンパイラの数学ライブラリ（libm.a）をリンクします。

```
C:¥>ld850 main.o -lm -lc
```

-l

[記述形式]

```
-lstring
```

- 省略時解釈

  アーカイブ・ファイルをリンクしません。

[機能説明]

- 未解決な外部シンボルの参照を解決する際，アーカイブ・ファイル lib*string*.a を参照します。
- このオプションで複数のアーカイブ・ファイルが指定された場合，指定順序に従って検索します。
- *string* の文字数は，64 文字未満にしてください。
- リンカは，このオプションの指定において，指定された時点で未解決な外部参照についてのみ指定されたアーカイブ・ファイルを参照します。したがって，コマンド・ラインからの起動では，このオプションは，指定するアーカイブ・ファイルを参照するオブジェクト・ファイルより，後ろに指定してください。

[使用例]

- 未解決な外部シンボルの参照を解決する際，アーカイブ・ファイル libtest.a を参照します。

```
C:￥>ld850 main.o -ltest
```

フラッシュ

フラッシュに関するオプションには，次のものがあります。

- -ext_table
- -zf

-ext_table

[記述形式]

```
-ext_table address
```

- 省略時解釈

  フラッシュ／外付け ROM 再リンク機能用のオブジェクト・ファイルを生成しません。

[機能説明]

- 8 桁の 16 進数 *address* で指定される値を分岐テーブル先頭アドレス値として，フラッシュ／外付け ROM 再リンク機能用のオブジェクト・ファイルを生成します（「B. 3. 3　ブート－フラッシュ再リンク機能」を参照）。
- ブート領域側指定時には，フラッシュ領域側への分岐処理となります。
  この際，分岐テーブルへの分岐となりますが，このアドレスが，このオプションで指定されるアドレスとなります。
- フラッシュ領域側指定時には，このオプションに指定したアドレスに，本来の分岐先への分岐命令を持つ分岐テーブルを生成します。
- このオプションに指定するアドレス値は，ブート領域側／フラッシュ領域側作成の際に同じ値とする必要があります。異なる値を指定した場合，正しく動作しません。また，エラー・チェックもしていません。
- このオプションに指定するアドレス値は，フラッシュ領域側 ROM 内である必要があります。指定したアドレスがどちらの領域であるかの判断ができないため，エラー・チェックはしていません。
- このオプションにより，フラッシュ領域側作成時には，指定されたアドレス値を先頭とする，サイズ（(ID 値[注]の最大 +1）×分岐テーブルのエントリサイズ）バイトのセクション .ext_table を自動作成します。このセクションは，リンク・ディレクティブ・ファイルで配置指定を行う必要はありませんが，配置するため領域を空けておく必要があります。

  注　アセンブラ・ソース・ファイルに .ext_func 疑似命令で指定された値

- このオプションは，-r オプションとの併用はできません。また，-r オプションにより生成したリロケータブル・オブジェクト・ファイルを入力した場合，正しく動作しません。
- フラッシュ／外付け ROM 再リンク機能の詳細は「B. 3. 3　ブート－フラッシュ再リンク機能」を参照してください。

［使用例］

- 分岐テーブル先頭アドレス値を 0x10000 としたブート領域側オブジェクト・ファイルを生成します。

```
C:￥>ld850 -ext_table 0x100000 boot.o
```

-zf

[記述形式]

```
-zf bootfile
```

- 省略時解釈

  フラッシュ／外付け ROM 再リンク機能用のオブジェクト・ファイルを生成しません。

  ただし，-ext_table 指定時には，ブート領域側オブジェクト・ファイルを生成します。

[機能説明]

- フラッシュ／外付け ROM 再リンク機能使用時に，指定したオブジェクト・ファイルをブート領域側のオブジェクト・ファイルとして，フラッシュ側オブジェクト・ファイルを生成します。
- ブート領域側オブジェクト・ファイルは，フラッシュ／外付け ROM 再リンク機能を指定して作成したものを指定してください。
- ここで指定するものは，リンカが出力したオブジェクトです。ROM 化プロセッサが出力したオブジェクトを指定すると，不正なオブジェクトが生成されるので注意してください。
- このオプションを使用する場合には，-ext_table オプションが指定されている必要があります。

[使用例]

- 分岐テーブル先頭アドレス値を 0x10000 としたフラッシュ領域側オブジェクト・ファイルを生成します。

  ブート領域側オブジェクト・ファイル名は boot.out とします。

```
C:¥>ld850 -zf boot.out -ext_table 0x100000 flash.o
```

デバイス

デバイスに関するオプションには，次のものがあります。

- -X256M
- -Xsid
- -Xob=none

-X256M

[記述形式]

```
-X256M
```

- 省略時解釈

  メモリ空間を 64M バイトとして扱い，アドレス解決します。

[機能説明]

【V850E】

- メモリ空間を 256M バイトとして扱います。
- このオプションは使用するチップセットにあわせて設定してください。
- V850Ex コアでは，物理アドレス空間が 256M バイト持つ場合が多く，64M を越えた 256M までの空間を使用するアプリケーションを作成する場合，このオプションを指定してください。

[使用例]

- メモリ空間を 256M バイトとして扱います。

```
C:￥>ld850 -X256M main.o
```

-Xsid

[記述形式]

```
-Xsid=id
```

- 省略時解釈

  -Xsid=0xffffffffffffffffffff（セキュリティ ID 搭載品種指定時）

[機能説明]

- フラッシュ・メモリ搭載デバイスの“セキュリティ ID”を設定します。
- セキュリティ ID 機能をサポートしていないデバイスの場合は利用できません。
- ID は，10 バイト以内の 16 進数（先頭の 0x は省略不可）で指定します。
  指定した値が 10 バイトより不足している場合は，上位ビットを 0 で埋めます。10 バイトを越えた場合はエラーを出力します。
- セキュリティ ID 機能をサポートしているデバイスに対して，本オプションの指定，またはアセンブラ記述 (.section "SECURITY_ID" 使用) によるセキュリティ ID の指定が省略された場合，“0xffffffffffffffffffff”が指定されたものとして扱います。
- 上記以外の方法でセキュリティ ID を設定した場合，リンカが生成したセキュリティ ID により多重指定したものと判断し，次のエラーとなります。

```
F4264: start address(0x00000070) of section "SECURITY_ID" overlaps previous section "任意のセクション名" ended before address (0xXXXXXXXX).
```

  このような場合には，+Xsid オプションを指定し，リンカによるセキュリティ ID の生成を抑止してください。
- セキュリティ ID 機能をサポートしていないデバイス用のオブジェクトをリンク時に指定した場合は，警告を出力し，指定は無視されます。

[使用例]

- セキュリティ・コード 0x112233445566778899aa （0x70 番地に 0x11，0x71 番地に 0x22，0x72 番地に 0x33，0x73 番地に 0x44，0x74 番地に 0x55，0x76 番地に 0x77，0x77 番地に 0x88，0x78 番地に 0x99，0x79 番地に 0xaa）を設定します。

```
C:¥>ld850 -Xsid=0x112233445566778899aa main.o
```

-Xob=none

[記述形式]

```
-Xob=none
```

- 省略時解釈

  オプション・バイトを生成します（オプション・バイト搭載品種指定時）

[機能説明]

- デフォルトで生成されるオプション・バイトを抑止します。
- 本オプションは，デバイス・ファイルに登録された初期値によるデフォルト生成のみを抑止します。
- アセンブラ・ソース・ファイルで .section "OPTION_BYTES" を使用して指定が行われた場合には，本オプションの指定の有無にかかわらず，.section "OPTION_BYTES" の指定が優先されます。
- 本オプションをオプション・バイト機能を持たないデバイスに対して指定した場合には，メッセージを出力せずに本オプションを無視します。

[使用例]

- デフォルトで生成されるオプション・バイトを抑止ししします。

```
C:￥>ld850 -Xob=none main.o
```

リンカ

リンカのオプションには，次のものがあります。

- -A
- -B
- -E
- -M
- -T
- -Ximem_overflow=warning
- -e
- -f
- -mc
- -rc
- -rescan
- -rom_less
- -s
- -t
- -v
- -w

-A

[記述形式]

```
-A
```

- 省略時解釈

  -G*num* オプションの *num* の設定において，目安として用いることのできる情報を出力しません。

[機能説明]

- ソース・ファイルのコンパイル時，およびアセンブル時に ca850，および as850 に対して指定する，sdata ／ sbss のデータ配置オプション（-G*num* オプションの *num* の設定）において，目安として用いることのできる情報を標準出力に出力します。
- *OK* と表示された数値を用いると，sdata ／ sbss の領域へは，その数値以下のサイズを持つデータが割り当てられます。
- ca850 から起動された場合，ca850 の起動時に指定された -A が渡されます。
- 詳細は，「（1）-A オプションの使い方」を参照してください。

[使用例]

- ca850，および as850 に対して指定する，sdata ／ sbss のデータ配置オプション（-G*num* オプションの *num* の設定）において，目安として用いることのできる情報を標準出力に出力します。

```
C:￥>ld850 -A main.o
```

-B

[記述形式]

```
-B
```

- 省略時解釈

  1パス・モードでリンクを行います。

[機能説明]

- 2パス・モードでリンクを行います。
- 2パス・モードは1パス・モードよりも低速ですが，より大きなサイズのファイルを処理できます。

[使用例]

- 2パス・モードでリンクを行います。

```
C:￥>ld850 -B main.o
```

-E

[記述形式]

```
-E
```

- 省略時解釈

  リロケーション処理において，不正箇所があった場合にメッセージを出力し，リンクの処理を中止します。

[機能説明]

- リロケーション処理において，次の不正箇所があった場合，
  - 未解決な外部参照のアドレス計算結果が不正な場合
  - 配置されるセクションとの関係が不正な場合

  エラーとはせず，警告メッセージを出力してリンクを続行します。
- 誤りとされた未解決な外部参照に対しては，不正と判断されたアドレスの計算結果の値は入れられず，元の値が残されます。

[使用例]

- リロケーション処理において，未解決な外部参照のアドレス計算結果が不正な場合，警告メッセージを出力してリンクを続行します。

```
C:￥>ld850 -E main.o
```

-M

[記述形式]

```
-M
```

- 省略時解釈

  多重定義された最初の外部シンボルに対してメッセージを出力し，リンクの処理を中止します。

[機能説明]

- 多重定義されたすべての外部シンボルに対してメッセージを出力し，リンクの処理を中止します。

[使用例]

- 多重定義されたすべての外部シンボルに対してメッセージを出力し，リンクの処理を中止します。

```
C:￥>ld850 -M main.o
```

-T

[記述形式]

```
-T
```

- 省略時解釈

  サイズのチェックを行い，サイズの違いが検出された場合，警告メッセージを出力し，リンクを続行します。
  この際，実際にシンボルが定義されているファイルのシンボル・サイズが有効となります。

[機能説明]

- 外部シンボルのリンクの際，サイズ，および整列条件のチェックを行いません。

[使用例]

- 外部シンボルのリンクの際，サイズ，および整列条件のチェックを行いません。

```
C:￥>ld850 -T main.o sub.o
```

-Ximem_overflow=warning

[記述形式]

```
-Ximem_overflow=warning
```

- 省略時解釈

  オーバーフロー時にはメッセージを出力し，リンクの処理を中止します。

[機能説明]

- 内蔵 ROM ／ RAM をオーバーフローした際のチェックの制御を行います。
- オーバーフロー時には警告メッセージを出力し，リンクを継続します。

[使用例]

- 内蔵 ROM ／ RAM をオーバーフローした際のチェックの制御を行います。

```
C:￥>ld850 -Ximem_overflow=warning main.o
```

-e

[記述形式]

```
-e symbol
```

- 省略時解釈

  次の規則でエントリ・ポイント・アドレス値を定めます。

  - シンボル __start が存在する場合は，その値
  - __start が存在しない場合は，生成されるオブジェクト・ファイル内の最下位に割り付けられた text 属性のセクションの先頭アドレス
  - text 属性セクションが存在しない場合は 0

[機能説明]

- シンボル *symbol* 値を生成されるオブジェクト・ファイルのエントリ・ポイント・アドレス値とします。
- 指定したシンボルが見つからない場合，リンカはメッセージを出力してリンクを中止します。
- シンボル名に空白は使用できません。

[使用例]

- エントリ・ポイント・アドレス値のシンボルを _main とします。

```
C:￥>ld850 -e _main main.o
```

-f

[記述形式]

```
-f num
```

- 省略時解釈

  -f 0x0000

[機能説明]

- 生成されるオブジェクト内のセクション間のアライン・ホールのフィリング値を，4 桁の 16 進数（2 バイト分）で指定します。
- このオプションを用いる場合は，-B オプションを指定して 2 パス・モードでリンクを行ってください。
- 先頭の“0x”は省略不可能です。
- このオプションによる指定は，リンク・ディレクティブにおけるフィリング値指定より優先します。
- 4 桁に満たない場合，満たない分の 0 が頭に指定されたものとみなします。
- ホールの大きさが 2 バイトに満たない場合，指定されたフィリング値の下位から必要な桁数分だけを取り出して初期化を行います。

[使用例]

- 生成されるオブジェクト内のセクション間のアライン・ホールのフィリング値を，0xffff とします。

```
C:￥>ld850 -B -f 0xffff main.o
```

-mc

[記述形式]

```
-mc
```

- 省略時解釈

  マスク・レジスタ機能を使用しているファイルと，使用していないファイルが混在していないかどうかチェックしません。

[機能説明]

- C ソース・ファイルから作成されたオブジェクト・ファイルのリンクの際，マスク・レジスタ機能を使用しているファイルと，使用していないファイルが混在していないかどうかチェックします。
- 混在している場合，リンクを中止します。

[使用例]

- リンクの際，マスク・レジスタ機能を使用しているファイルと，使用していないファイルが混在していないかどうかチェックします。

```
C:¥>ld850 -mc main.o sub.o
```

-rc

[記述形式]

```
-rc
```

- 省略時解釈

  すべての入力オブジェクト・ファイルに対し，異なるレジスタ・モードが混在している場合に詳細な情報を出力しません。

[機能説明]

- すべての入力オブジェクト・ファイルに対し，異なるレジスタ・モードが混在している場合に詳細な情報を出力します。
- -w オプションと同時指定された場合，このオプションは無視されます。

[使用例]

- すべての入力オブジェクト・ファイルに対し，異なるレジスタ・モードが混在している場合に詳細な情報を出力します。

```
C:¥>ld850 -rc main.o sub.o
```

-rescan

[記述形式]

```
-rescan
```

- 省略時解釈

  -l オプションで指定したライブラリ・ファイルの再参照を行いません。

[機能説明]

- -l オプションで指定したライブラリ・ファイルの再参照を行います。
- このオプションを指定すると，ライブラリのリンク順によるシンボル未解決を防ぐことができます。

[使用例]

- アーカイブ・ファイル libtest1.a，libtest2.a の再参照を行います。

```
C:￥>ld850 -rescan main.o -ltest1 -ltest2
```

-rom_less

[記述形式]

```
-rom_less
```

- 省略時解釈

  内蔵 ROM 領域となっているアドレスに，アプリケーションの配置がオーバラップしている場合，メッセージを出力し，リンクの処理を中止します。

[機能説明]

- 内蔵 ROM 領域に対する配置に対して，チェックを行いません。

  つまり，内蔵 ROM 領域となっているアドレスに，アプリケーションの配置がオーバラップしていても，警告メッセージを出力しません。
- アプリケーションを ROM レス・モードで作成する場合，このオプションを指定してください。

注意　シングルチップ・モード選択時の内蔵 ROM オーバのチェックには対応していません。このオプションを指定して内蔵 ROM オーバーフローのチェックを無効とし，リンク・マップで確認してください。

[使用例]

- 内蔵 ROM 領域に対する配置に対して，チェックを行いません。

```
C:¥>ld850 -rom_less main.o
```

-S

[記述形式]

```
-s
```

- 省略時解釈

  入力オブジェクトに，デバッグ情報，ライン・ナンバ情報，およびグローバル・ポインタ・テーブルが含まれる場合には，その情報を含んだオブジェクト・ファイルを生成します。

[機能説明]

- オブジェクト・ファイルの生成において，デバッグ情報，ライン・ナンバ情報，およびグローバル・ポインタ・テーブルを取ったオブジェクト・ファイルを生成します。

[使用例]

- オブジェクト・ファイルの生成において，デバッグ情報，ライン・ナンバ情報，およびグローバル・ポインタ・テーブルを取ったオブジェクト・ファイルを生成します。

```
C:¥>ld850 -s main.o
```

-t

[記述形式]

```
-t
```

- 省略時解釈

  シンボル・サイズ，および整列条件のチェックを行い，違いが検出された場合，警告メッセージを出力し，リンクを続行します。

[機能説明]

- 未定義外部シンボルのリンクにおいて，シンボルのサイズ，および整列条件のチェックを行いません。
- リンカは，未定義外部シンボルの多重定義をサポートしています。

  多重定義された未定義外部シンボルは，リンク後 .sbss，または .bss セクションに割り付けられます。この際，リンクされるシンボル・サイズ，または整列条件が異なっていた場合，サイズは，リンクされるシンボルのうちの最大サイズとし，整列条件は，リンクされるシンボルの整列条件の最小公倍数とします。

[使用例]

- 未定義外部シンボルのリンクにおいて，シンボルのサイズ，および整列条件のチェックを行いません。

```
C:￥>ld850 -t main.o
```

-v

[記述形式]

```
-v
```

- 省略時解釈

  なし

[機能説明]

- リンカの実行状況の詳細を出力します。リンクするオブジェクトの一覧等が表示されます。

[使用例]

- リンカの実行状況の詳細を出力し，リンクするオブジェクトの一覧等が表示されます。

```
C:¥>ld850 -v main.o
```

-w

[記述形式]

```
-w
```

- 省略時解釈

  警告メッセージを抑止しません。

[機能説明]

- 警告メッセージを出力しません。
- 致命的な誤りに対するメッセージのみを出力します。

[使用例]

- 致命的な誤りに対するメッセージのみを出力します。

```
C:￥>ld850 -w main.o
```

その他

その他のオプションには，次のものがあります。

- -F
- -V
- -cpu
- -fc
- -help
- -mask_reg
- -r
- -ro
- -reg
- @

-F

［記述形式］

```
-F devpath
```

- 省略時解釈

  デバイス・ファイルを，標準フォルダから探します。

［機能説明］

- リンカを単体で起動する場合，デバイス・ファイルを，フォルダ *devpath* から探します。
- ca850 から起動する場合に，デバイス・ファイルのパスを指定するには，ca850 の -devpath オプションを使用します。

［使用例］

- リンカを単体で起動する場合，デバイス・ファイルを，フォルダ D:￥dev から探します。

```
C:￥>ld850 -F D:￥dev main.o
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- リンカのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- リンカのバージョン情報を標準エラー出力に出力します。

```
C:￥>ld850 -V
```

-cpu

[記述形式]

```
-cpu devicename
```

- 省略時解釈

.o ファイル作成時に指定されたターゲット・デバイスのデバイス・ファイルを読み込みます。

[機能説明]

- *devicename* で指定されたターゲット・デバイスのデバイス・ファイルを読み込みます。

[使用例]

- ターゲット・デバイスとして UPD70F3719 を指定します。

```
C:¥>ld850 -cpu f3719 main.o
```

-fc

[記述形式]

```
-fc
```

- 省略時解釈

  C ソース・ファイルから作成されたオブジェクト・ファイルのみをチェックします。

[機能説明]

- すべての入力オブジェクト・ファイルに対し，旧関数呼び出しと現バージョンの呼び出し仕様が混在していないかチェックします。
- 旧関数呼び出し仕様は現バージョンではサポート対象外です。

[使用例]

- すべての入力オブジェクト・ファイルに対し，旧関数呼び出しと現バージョンの呼び出し仕様が混在していないかチェックします。

```
C:¥>ld850 -fc main.o sub.o
```

-help

[記述形式]

```
-help
```

- 省略時解釈

  なし

[機能説明]

- オプションの説明を標準エラー出力に出力します。

[使用例]

- オプションの説明を標準エラー出力に出力します。

```
C:¥>ld850 -help
```

-mask_reg

[記述形式]

```
-mask_reg
```

- 省略時解釈

  マスク・レジスタ機能を使用しないライブラリを参照します。

[機能説明]

- マスク・レジスタ機能用のライブラリを参照します。
- ca850 から起動する場合，-Xmask_reg オプションを使用します。
- マスク・レジスタ機能用のライブラリは，32 レジスタ・モード時に使用できるライブラリです。22，26 レジスタ・モード時に指定した場合，次の警告メッセージを出力し，後ろに指定したものを無視します。

```
W4857: "-reg22" option is illegal when "-mask_reg" option is specified, ignored "-reg22"
option.
```

[使用例]

- マスク・レジスタ機能用のライブラリを参照します。

```
C:¥>ld850 -mask_reg main.o
```

-r

[記述形式]

```
-r
```

- 省略時解釈

  未解決な外部参照が残されていた場合は，次のメッセージを出力し，リンクを中止します。この場合，オブジェクト・ファイル（ロード・モジュール・ファイル）は生成されません。

```
F4452 :  undefined symbol.
         symbol referenced in "file"
```

[機能説明]

- リロケータブル（再配置可能）なオブジェクト・ファイルを生成します。
- -ro オプションと同時に指定された場合，このオプションは無視されます。
- このオプションを指定すると，リンク終了後に未解決な外部参照が残されていてもメッセージを出力せずにリンクを正常終了します。
- リンカによって生成されたオブジェクト・ファイルを，リンカによる再リンクの対象として指定する場合，再リンクの対象となるオブジェクト・ファイルの生成には，このオプションを使用してください。

[注意事項]

- このオプションが指定されると，リンク・ディレクティブは，マッピング・ディレクティブの部分のタイプ／属性のみが有効になり，その他は無視されます。
- このオプションが指定されると，予約シンボルの作成を行いません。
- CA850 Ver.2.30 以前からは，-r オプションの仕様が変更されています。
  旧バージョンのマッピング方式を用いる場合，-r オプションの代わりに -ro を用いてください。

[使用例]

- リロケータブル（再配置可能）なオブジェクト・ファイルを生成します。

```
C:￥>ld850 -r main.o
```

-ro

[記述形式]

```
-ro
```

- 省略時解釈

  実行可能なオブジェクト・ファイルを生成します。

[機能説明]

- リロケータブル（再配置可能）なオブジェクト・ファイルを旧マッピング方式（CA850 Ver.2.30 以前）で生成します。
- -r オプションと同時指定された場合は，-r は無視されます。

[使用例]

- リロケータブル（再配置可能）なオブジェクト・ファイルを旧マッピング方式（CA850 Ver.2.30 以前）で生成します。

```
C:¥>ld850 -ro main.o
```

-reg

[記述形式]

```
-regnum
```

- 省略時解釈

  -reg32

[機能説明]

- 対応するレジスタ・モード用のライブラリを参照します。
- *num* には，22，26，または 32 が指定可能です。

  -reg の後ろに空白を入れないでください。

[使用例]

- 22 レジスタ・モード用のライブラリを参照します。

```
C:￥>ld850 -reg22 main.o
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。このオプションを指定している場合は，オプション文字列はコマンド・ファイルに出力されるため，文字列の制限を意識する必要がなくなります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>ld850 @command
```

B. 3. 3　ブートーフラッシュ再リンク機能

（1）再リンク機能とは

システムによっては，フラッシュ領域や，着脱可能な ROM を搭載していることがあります。

フラッシュ領域の場合は，書かれてある内容を書き換えたり，着脱可能な ROM の場合は，新しく書き換えた ROM 自体を取り替えることによって，プログラムのバージョン・アップ等を行います。

プログラムの一部でも変更する場合，基本的にプロジェクトそのものを再構築，つまり，“リビルド”して作成し直すことになります。しかし，バージョン・アップしたい箇所が，フラッシュ領域や外付け ROM だけに限られている場合は，再構築しないで済むと便利です。また，ブート部分は内蔵 ROM などに固定され，書き換え対象のフラッシュ領域との間に関数呼び出しがある場合，フラッシュ領域内の関数を修正することにより，関数の先頭アドレスがずれてしまうと，関数呼び出しが正常に行えなくなってしまいます。

このような状況を防ぎ，正常に関数呼び出しの実現をするのが“ブートーフラッシュ再リンク機能”（以下“再リンク機能”）です。

実現方法の概略は，次のようになります。

（a）フラッシュ領域に，フラッシュ領域内の関数群への分岐命令が書かれてある“分岐テーブル”を用意する

（b）ブート領域から，フラッシュ領域内の関数をコールするとき，いったんフラッシュ領域の分岐テーブルへジャンプし，その後，目的の関数への分岐命令を実行してジャンプする

これらの仕組みをユーザで用意して実現することもできますが，この“再リンク機能”を用いると比較的簡単に実現できます。

ただし，この機能を使う上で，ブート領域側を作成した時点で，フラッシュ領域側の呼び出す関数は決定している必要があります。あくまでも，フラッシュ領域側の関数に変更があっても，ブート領域側からその関数を問題なく呼び出すことができるようにする仕組みです。

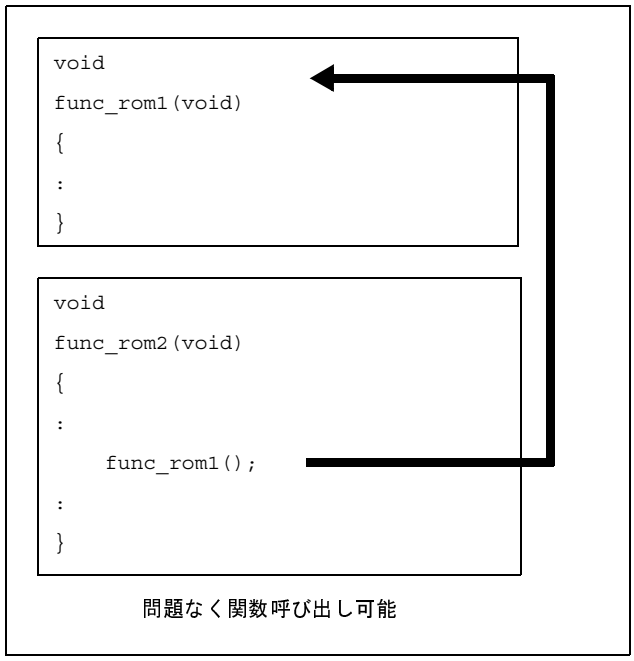
（2）再リンク機能のイメージ

再リンク機能を利用したときの，関数呼び出しのイメージは次のようになります。

（a）ブート領域内からブート領域内の関数を呼び出すとき

ブート領域に書き込む前に，すでにアドレス解決ができていることなので，問題なく関数呼び出しができます。

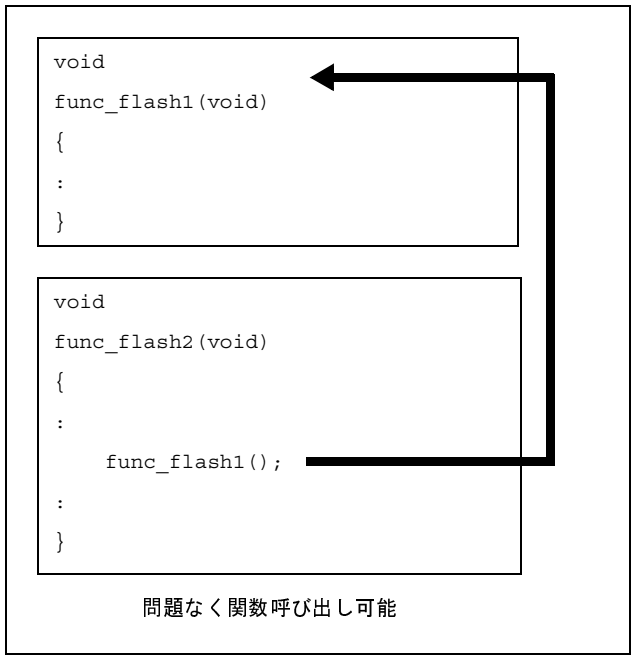
図 B—15　ブート領域内



（b）フラッシュ領域内からフラッシュ領域内の関数を呼び出すとき

フラッシュ領域内ではアドレス解決ができていることなので，問題なく関数呼び出しができます。

図 B—16　フラッシュ領域内

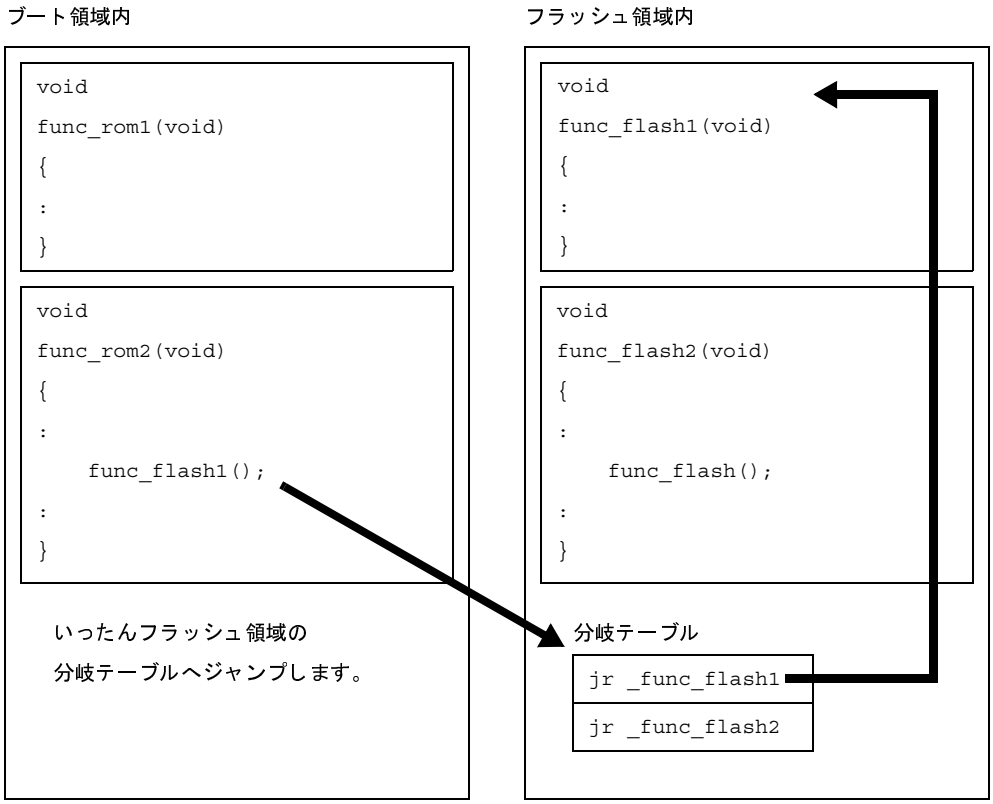


(c) ブート領域内からフラッシュ領域内の関数を呼び出すとき

ブート領域内からフラッシュ領域内にある関数を呼び出すとき，ブート領域内からは，フラッシュ領域内の関数サイズ等の変更により，アドレスがわかりません。つまり，フラッシュ領域内の関数を直接呼び出すことができません。これを解決するため，いったんフラッシュ領域内の分岐テーブルへジャンプします。

そのテーブルから該当する関数へのジャンプ命令を実行し，目的の関数へジャンプします。

図 B—17 ブート領域内からフラッシュ領域内



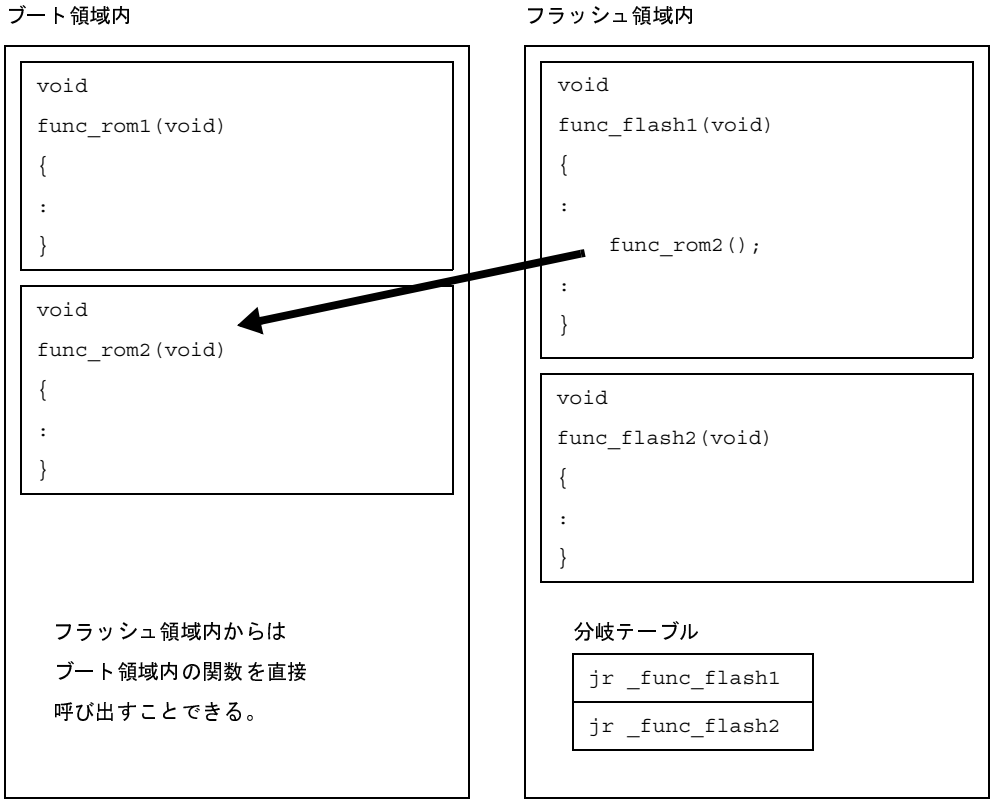
また，関数と同様に，外部変数の参照の可否にも関係します。

フラッシュ領域内に定義されているグローバル変数は，ブート領域内から参照することはできません。そのため，ブート領域内，フラッシュ領域内それぞれで同じ名前の外部変数を定義することができます。その外部変数に対する参照は，それぞれの領域内からの参照のみになります。

（d）フラッシュ領域内からブート領域内の関数を呼び出すとき

フラッシュ領域内からブート領域内にある関数を呼び出すとき，ブート領域内の内容は変わらないので，フラッシュ領域内からはブート領域内にある関数を直接呼び出すことができます。

図 B―18　フラッシュ領域内からブート領域内



また，関数と同様に，外部変数の参照の可否にも関係します。ブート領域内に定義されているグローバル変数は，フラッシュ領域内から参照することができます。

（3）再リンク機能の実現方法

再リンク機能を実現する具体的な方法について説明します。

（a）CubeSuite+ のプロジェクト

再リンク機能を実現する場合，“ブート領域側”と“フラッシュ領域側”を別々に作成することになります。つまり，一度ブート領域側を作成したあと（ROM に書き込んだのち）は，フラッシュ領域側だけを変更することになります。そのため，CubeSuite+ でプロジェクトを作成するときは，次のように分けて作成してください。

- ブート領域側に配置するプロジェクト
- フラッシュ領域側に配置するプロジェクト（今後変更することがあるプロジェクト）

また，“スタート・アップ・ルーチン”，および“リンク・ディレクティブ・ファイル”も，それぞれのプロジェクト用に別々に用意します。

（b）.ext_func 疑似命令

ブート領域側から，フラッシュ領域側の関数を呼び出したい場合，まず，ブート領域側に .ext_func 疑似命令を使って“呼び出す関数名（ラベル名）と ID 番号”を付けます。.ext_func 疑似命令の書式は次のようになります。

```
.ext_func 関数名 , ID 番号
```

ID 番号は正数で指定します。また，“同じ関数名で異なる ID 番号を指定”したり“異なる関数名に同じ ID 番号を指定”したりすることはできません。

ブート領域側に，.ext_func 疑似命令を使ってフラッシュ領域側にある関数名を指定すると，分岐テーブル（ext_table）が作成されます。この ext_table のアドレスはユーザで指定します。

指定方法は“ブート領域側のロード・モジュール”を作成するとき，および“フラッシュ領域側のロード・モジュール”を作成するとき，それぞれのリンク・オプション“-ext_table”で次のように指定します。

```
-ext_table 指定するアドレス
```

関数本体へ分岐するときは，作成した分岐テーブルの先頭から，ID 番号によるオフセット参照をすることによって実際の関数アドレスを取得し，そして分岐することになります。

以下に，例を示します。

```
func_flash0()
func_flash1()
func_flash2()
```

上記 3 つの C 関数がフラッシュ領域に配置されていて，これらをブート領域側から呼び出したい場合，ブート領域側にアセンブラで次のように記述します。

```
.ext_func _func_flash0, 0
.ext_func _func_flash1, 1
.ext_func _func_flash2, 2
```

C ソース・ファイル内に書くときは，#pragma asm ～ #pragma endasm 指令，または __asm() 命令を使って記述します。#pragma asm ～ #pragma endasm 指令を使った例は，次のようになります。

```
#pragma asm
        .ext_func _func_flash0, 0
        .ext_func _func_flash1, 1
        .ext_func _func_flash2, 2
#pragma endasm
```

なお，これらの .ext_func 疑似命令群の記述は，記述漏れやソース間の矛盾が生じることを防ぐため，つまり，“同じ関数名で異なる ID 番号を指定”したり“異なる関数名に同じ ID 番号を指定”というような間違いを防ぐため，1 つのファイルにまとめて，すべてのソースに .include 疑似命令で（C 言語で記述するときは，#include 命令で）インクルードすることを推奨します。

上記のように #pragma asm ～ #pragma endasm 指令を使ったファイルをインクルードすると，コンパイル時に次のメッセージが出ますが，無視してください（または，“個別の警告”で出力しないように設定してください）。

```
W2244: '#pragma asm' used out of function is not supported completely.
```

再リンク機能のイメージは，次のようになります。

ユーザが記述するアセンブラ・ソース	リンク後のアセンブラ・イメージ
[ext_table.inc] .ext_func _func_flash0, 0 .ext_func _func_flash1, 1 .ext_func _func_flash2, 2	
[rom.s] .include "ext_table.inc" .extern _func_flash0 .extern _func_flash1 .extern _func_flash2 jarl _func_flash0, lp jarl _func_flash1, lp jarl _func_flash2, lp	[rom.out] .extern __ext_table_head jarl __ext_table_head+0x4*0,lp jarl __ext_table_head+0x4*1,lp jarl __ext_table_head+0x4*2,lp

<table>
<tr><th>ユーザが記述するアセンブラ・ソース</th><th>リンク後のアセンブラ・イメージ</th></tr>
<tr><td rowspan="2">[flash.s]
include "ext_table.inc"
.globl _func_flash0
.globl _func_flash2
_func_flash0:
:
jmp [lp]

.globl _func_flash1
_func_flash1:
:
jmp [lp]

_func_flash2:
:
jmp [lp]</td><td>[flash.o]
#（分岐テーブル）
.section ".ext_table", text
.globl __ext_table_head
.extern _func_flash0
.extern _func_flash1
.extern _func_flash2
__ext_table_head:
jr _func_flash0
jr _func_flash1
jr _func_flash2</td></tr>
<tr><td>#（関数本体）
.globl _func_flash0
_func_flash0:
:
jmp [lp]

.globl _func_flash1
_func_flash1:
:
jmp [lp]

.globl _func_flash2
_func_flash2:
:
jmp [lp]</td></tr>
</table>

このように .ext_func 疑似命令を指定すると，ext_table というシンボルでテーブルが生成され，その先頭シンボルが“__ext_table_head”になります。

ブート領域側の“jarl _func_flash0, lp”というコードは，__ext_table_head からのオフセットで _func_flash0 のアドレスを取得し，jarl 命令で関数本体へジャンプします。

（c）スタート・アップ・ルーチン

ブート領域側のスタート・アップ・ルーチンと，フラッシュ領域側のスタート・アップ・ルーチンは，それぞれに用意します。それぞれのスタート・アップ・ルーチンでしなくてはならない処理は，次のようになります。

- ブート領域側で tp，gp，ep 値をセットする
- ブート領域側で使用する RAM 領域を初期化するため，_rcopy 関数を呼び出す
- ブート領域側からフラッシュ領域側のスタート・アップ・ルーチンへ分岐する
- フラッシュ領域側で使用する RAM 領域を初期化するため，_rcopy 関数を呼び出す
- フラッシュ領域側の処理へ移行

tp，gp，ep をブート領域内で使用しない場合は，値のセットをフラッシュ領域で行っても問題ありません。また，_rcopy 関数を使用して初期値データのコピーを行う際は，ロード・モジュールに対し，ROM 化プロセッサによる“ROM 化”を行っている必要があります。rompsec セクションの先頭シンボルを持った rompcrt.o を用意し，リンク・オプション“-lr”を指定してリンクします。これによって作成されたパッキング・セクションを用いて，_rcopy 関数で初期値ありデータをコピーします（「B. 4 ROM 化プロセッサ」参照）。

tp，gp，ep 値ですが，これらはブート領域側，およびフラッシュ領域側で同じアドレス値を使用することを推奨します。異なる値にすることも可能ですが，その場合はブート領域側とフラッシュ領域側の命令コードを行き来するたびに，値を設定しなおす必要が出てきます。

ブート領域側	フラッシュ領域側

`__start:`
 	
`mov #__tp_TEXT, tp`
`mov #__gp_DATA, gp`
`mov #__ep_DATA, ep`
 	
`:`
 	
`# ブート領域 側の main 関数へ`
`# main 関数という名前にこだわる必要はない`
`jarl _main, lp`
 	
`.ext_func _flash_start 3`
`jr __flash_start`
	`.ext_func _flash_start 3`
	`jr __flash_start`
	
	`__flash_start:`
	
	`:`
	
	`# フラッシュ領域 側の main 関数へ`
	`jarl _main, lp`

`extern unsigned long _S_romp;`
 	
`void main(void)`
`{`
`_rcopy(&_S_romp, -1);`
`:`
`}`
	`extern unsigned long _S_romp;`
	
	`void main(void)`
	`{`
	`_rcopy(&_S_romp, -1);`
	`:`
	`}`

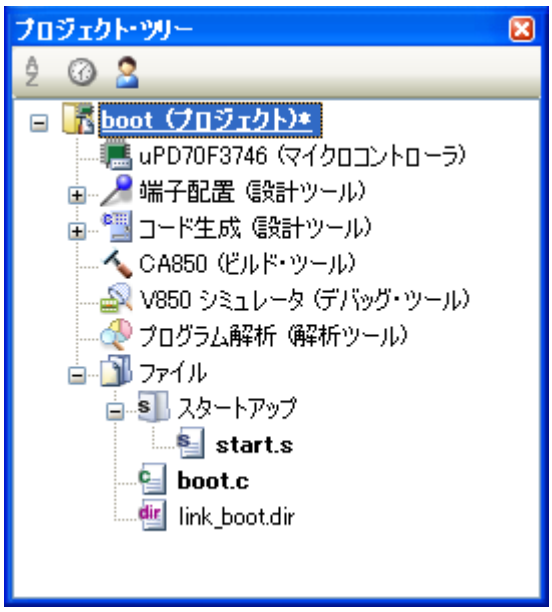
（d）プロジェクトの具体的な作成方法

- ブート領域側のプロジェクトの作成

ブート領域側のプロジェクトを作成し，ビルド対象ファイルをプロジェクトに追加します。

スタートアップ・ルーチンは，スタートアップ・ノードに追加します。

図 B—19 ブート領域側のプロジェクト

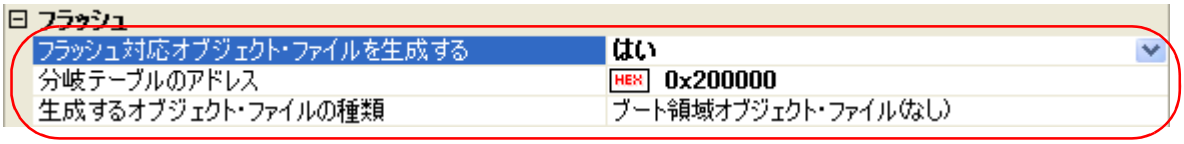


- ブート領域側のプロジェクトのビルド・オプションの設定

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。ビルド・オプションの設定は，［フラッシュ］カテゴリで行います。

［フラッシュ対応オブジェクト・ファイルを生成する］プロパティで［はい］を選択すると，［分岐テーブルのアドレス］プロパティと［生成するオブジェクト・ファイルの種類］プロパティが表示されます。

図 B—20 ブート領域側の［フラッシュ］カテゴリ



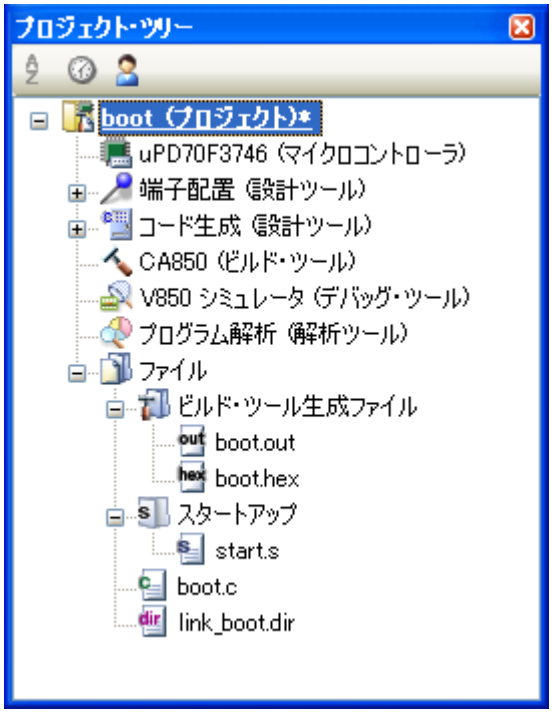
［分岐テーブルのアドレス］プロパティで，分岐テーブルの先頭アドレス（フラッシュ領域内のアドレス）を指定します。指定可能な値の範囲は，0x0 ～ 0xffffffff（16 進数）です。デフォルトでは，“0x0”が設定されています。

また，［生成するオブジェクト・ファイルの種類］プロパティで，［ブート領域オブジェクト・ファイル (なし)］を選択します（デフォルト）。

- ブート領域側のプロジェクトのビルドの実行

  ブート領域側のプロジェクトのビルドを実行すると，ロード・モジュール・ファイルが生成されます。

図 B—21　ブート領域側の生成ファイル



- フラッシュ領域側のプロジェクトの作成

  フラッシュ領域側のプロジェクトを作成し，ビルド対象ファイルをプロジェクトに追加します。

  スタートアップ・ルーチンは，スタートアップ・ノードに追加します。

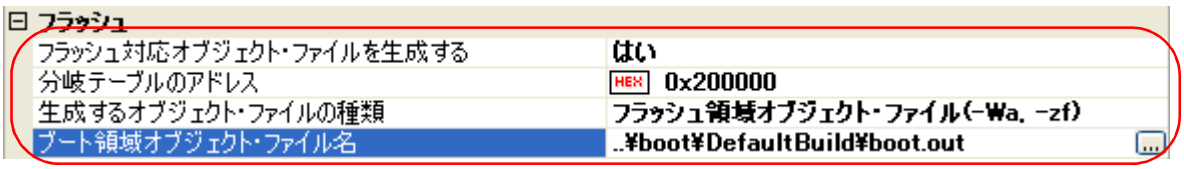
図 B—22　フラッシュ領域側のプロジェクト

- フラッシュ領域側のプロジェクトのビルド・オプションの設定

プロジェクト・ツリーでビルド・ツール・ノードを選択し，プロパティ パネルの［共通オプション］タブを選択します。ビルド・オプションの設定は，［フラッシュ］カテゴリで行います。

［フラッシュ対応オブジェクト・ファイルを生成する］プロパティで［はい］を選択すると，［分岐テーブルのアドレス］プロパティと［生成するオブジェクト・ファイルの種類］プロパティが表示されます。

図 B―23　フラッシュ領域側の［フラッシュ］カテゴリ



［分岐テーブルのアドレス］プロパティで，分岐テーブルの先頭アドレス（ブート領域側と同じアドレス）を指定します。

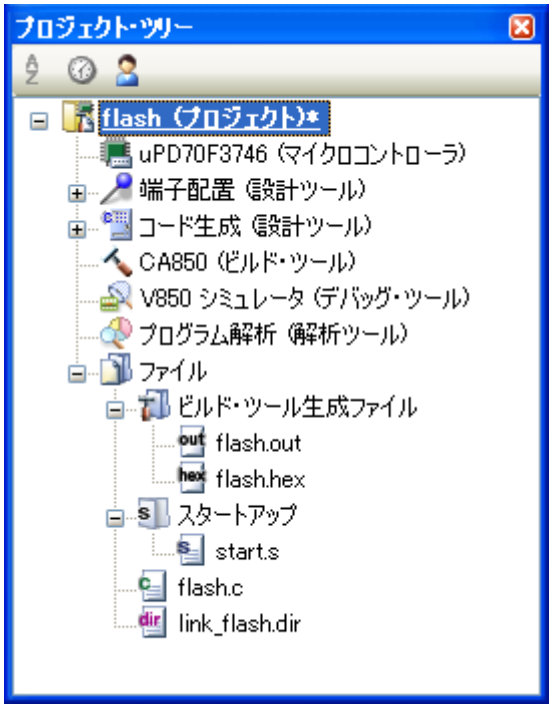
また，［生成するオブジェクト・ファイルの種類］プロパティで［フラッシュ領域オブジェクト・ファイル (-Wa, -zf)］を選択すると，［ブート領域オブジェクト・ファイル名］プロパティが表示されます。ここで，ブート領域側のオブジェクト・ファイルを指定します。

注意　ここで指定するものは，リンカが出力したオブジェクト・ファイルです。ROM 化プロセッサが出力したオブジェクト・ファイルを指定すると，エラーとなるので注意してください。

- フラッシュ領域側のプロジェクトのビルドの実行

フラッシュ領域側のプロジェクトのビルドを実行することにより，再リンク機能を実現したロード・モジュール・ファイルが生成されます。

図 B―24　フラッシュ領域側の生成ファイル



(e) リンク・ディレクティブ・ファイルの記述

ブート領域側のプロジェクトと，フラッシュ領域側のプロジェクトで，それぞれリンク・ディレクティブを持ちます。リンク・ディレクティブ・ファイルを記述する際の注意事項は次のとおりです。

- RAM 領域に置くセクションのアドレスは，ブート領域側とフラッシュ領域側でオーバラップしていても，プロジェクトが異なるので，リンカ等でエラーを出力することができません。つまり，オーバラップさせることが可能です。ブート領域側とフラッシュ領域側で同時に参照する必要のある RAM 領域は，オーバラップさせないようにアドレス指定する必要があります。
- tp，gp，ep 値ですが，これらはブート領域側，およびフラッシュ領域側で同じアドレス値を使用することを推奨します。異なる値にすることも可能ですが，その場合はブート領域側とフラッシュ領域側の命令コードを行き来するたびに，値を設定しなおす必要が出てきます。
- 分岐テーブル（ext_table）に関するリンク・ディレクティブの記述は必要ありません。リンク・オプション“-ext_table”で指定されたアドレスに自動的に配置されます。

  ただし，次の点に注意が必要です。

  - -ext_table で指定されたアドレスに，分岐テーブルのサイズ分の空き領域があった場合，そのまま配置されます。他のセグメントへの影響はありません。このようにするのが最も理想的です。
  - -ext_table で指定されたアドレスに，分岐テーブルのサイズ分の空き領域がなかった場合，エラーになります。たとえば -ext_table で指定したアドレスは“アドレス指定された TEXT セグメント内”で，すでにコードが配置されている場合などです。これに該当する例は，次のようになります。

分岐テーブルのアドレス指定

```
-ext_table 0x500
```

リンク・ディレクティブ・ファイル（の一部）

```
TEXT : !LOAD ?RX V0x400 {
        .text = $PROGBITS ?AX .text;
};
```

（TEXT セグメントのサイズが 0x100 バイト以上あるとします）

このとき，分岐テーブルは 0x500 番地に割り付けることができないため，リンク時にエラーになります。-ext_table で指定する値を変更してください。

- 再リンク機能を使う前は，-ext_table で指定したアドレスには，他のセグメントが割り当てられていたが，リンク・ディレクティブには，そのセグメントのアドレス指定がなかった場合，-ext_table で指定されたアドレスには分岐テーブルが配置され，元あったセグメントは，分岐テーブルの後ろに移動します。
  ただし，セグメントがずれたことにより，さらに後ろの“アドレス指定されたセグメント”に重なった場合は，エラーになります。

分岐テーブルのアドレス指定

```
-ext_table 0x500
```

リンク・ディレクティブ・ファイル（の一部）

```
TEXT : !LOAD ?RX {
        .text = $PROGBITS ?AX .text;
};
```

（TEXT セグメントよりも前のセグメントから続きで，0x500 番地から TEXT セグメントが割り当てられていたとします）

このとき，TEXT セグメントにはアドレス指定がないため，分岐テーブルは 0x500 番地に割り付けられ，TEXT セグメントは分岐テーブルの後ろに割り付けられます。

(f) .ext_ent_size 疑似命令

フラッシュ領域内にある分岐テーブルから，実際の関数を呼び出すとき，デフォルトでは，次のようにjr 命令による分岐命令が出力されます。

```
__ext_table_head:
        jr      _func_flash0
        jr      _func_flash1
        jr      _func_flash2
```

しかし，アーキテクチャ上，jr 命令では 22 ビット範囲内（± 1M バイト範囲内）にしか分岐することができません。それ以上のアドレス，つまり，32 ビット空間すべてに分岐できるようにしたい場合，.ext_ent_size 疑似命令を追加で指定します。.ext_ent_size 疑似命令は書式は，次のようになります。

```
.ext_ent_size テーブルのエントリ・サイズ
```

エントリ・サイズで指定できる値は“4”“8”“10”のいずれかになります。ここでの“テーブルのエントリ・サイズ”とは“1 つの分岐処理に必要な命令サイズ”ということになります。

デフォルトは“4”で，この場合は，次のように 4 バイト命令が配置されます。

```
jr      _flash_func0    -- 4 バイト命令
```

“8”を指定すると，次のように合計で 8 バイトの命令が配置されます。

```
mov     #_flash_func0, r1       -- 6 バイト命令
jmp     [r1]                    -- 2 バイト命令
```

“10”を指定すると，次のように合計で 10 バイトの命令が配置されます。

```
movhi   hi1(#_flash_func0), r0, r1      -- 4 バイト命令
movea   lo(#_flash_func0), r1, r1       -- 4 バイト命令
jmp     [r1]                            -- 2 バイト命令
```

8 バイト命令が使用できる（この命令セットに対応している）のは“V850Ex ／ V850E2 コアの場合のみ”です。

V850 で使用する場合は“10”を設定してください。また，V850 ／ V850Ex ／ V850E2 コア共通のオブジェクトを作成する場合（-cn オプション指定する場合）は，“10”で統一してください。

(g) ライブラリについて

ブート領域やフラッシュ領域からライブラリ関数を呼び出していた場合，ライブラリは呼び出した側のオブジェクトにリンクされます。たとえば，フラッシュ領域側にライブラリがリンクされていても，ブート領域からも同じライブラリ関数を呼び出していた場合は，ブート領域にも同じライブラリがリンクされます。つまり，ライブラリ関数を呼び出す場合は，ブート領域とフラッシュ領域との間で分岐は起こらないため，ライブラリ関数に対して .ext_func 疑似命令で関数指定する必要はありません。

ただし，“ブート領域側にリンクされたライブラリが，フラッシュ領域側の関数へ分岐する”というような特殊な場合に関しては，.ext_func 疑似命令で関数指定する必要があります。

CA850 にパッケージされている“標準ライブラリ”や“数学ライブラリ”に関しては，.ext_func 疑似命令で関数指定する必要はありません。

（h）割り込みハンドラについて

割り込みハンドラの呼び出し部分は，割り込みハンドラ・アドレスのある領域側に記述してください。

次の場合，割り込みハンドラ関数名に対しても，.ext_func 疑似命令で関数指定する必要があります。

- 割り込みハンドラ・アドレスは“ブート領域側”
- 割り込みハンドラ本体は“フラッシュ領域側”

<table>
<tr><th>ユーザが記述するアセンブラ・ソース</th><th>リンク後のアセンブラ・イメージ</th></tr>
<tr><td>[ext_table.inc]
.ext_func _int_flash0, 0</td><td></td></tr>
<tr><td>[rom.s]
.include "ext_table.inc"
.extern _int_flash0
.section "INT00", text
jr _int_flash0</td><td>[rom.out]
.section "INT00", text
jr __ext_table_head+0x4*0,lp</td></tr>
<tr><td rowspan="2">[flash.s]
.include "ext_table.inc"
.globl _int_flash0
_int_flash0:
:
reti</td><td>[flash.o]
#（分岐テーブル）
.section ".ext_table", text
.globl __ext_table_head
.extern _int_flash0
__ext_table_head:
jr _int_flash0</td></tr>
<tr><td>#（ハンドラ本体）
.globl _int_flash0
_int_flash0:
:
reti</td></tr>
</table>

B. 3. 4　補足事項

ここでは，リンカに関するその他の補足的な事項について説明します。

（1）-A オプションの使い方

ここでは，-A オプションの使い方について説明します。

CubeSuite+ の場合，プロパティ パネルの［リンク・オプション］タブをオープンし，［その他］カテゴリの［GP 情報を表示する］プロパティで［はい (-A)］を選択します。

（a）機能

ソース・ファイルのコンパイル時，およびアセンブル時に，ca850，および as850 に対して指定できる -G*num* オプションに対し，*num* に設定する値の目安となる情報を，コマンド・ラインで -A オプションを指定して起動した場合は標準出力に表示します。CubeSuite+ で［GP 情報を表示する］プロパティで［はい (-A)］を選択した場合は，出力 パネルに表示します。

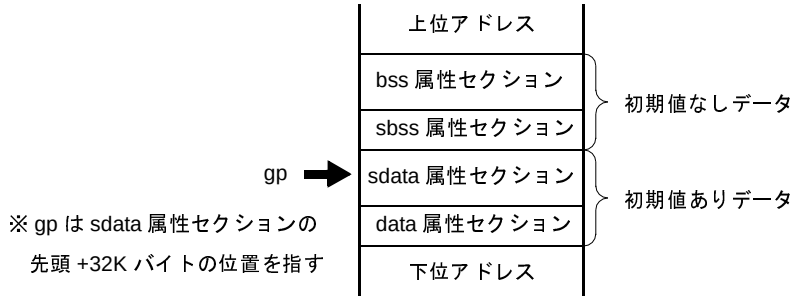
-G*num* オプションは「*num* バイト以下のデータを .sdata セクション，または .sbss セクションに配置する」オプションです。

ca850 および as850 は，sdata，sbss，data，bss 領域に配置するデータに対し，次のような規則に従ってコードを出力します。

まず gp レジスタから 1 命令でアクセスできる領域である“sdata 属性セクション”，“sbss 属性セクション”へ配置しようとします（初期値ありデータが sdata 属性セクションへ，初期値なしデータが sbss 属性セクションへ配置されます）。

これらの領域は gp と 16 ビットのディスプレイスメントを用いてアクセスするコードになるため，配置できる範囲は gp から± 32K バイト内に限られます。これらの領域に収まりきらなかった場合，gp レジスタから 2 命令でアクセスできる領域である“data 属性セクション”，“bss 属性セクション”へ配置しようとします（初期値ありデータが data 属性セクションへ，初期値なしデータが bss 属性セクションへ配置されます）。これらの領域は，ますアクセス領域のアドレス生成を行い，gp と 32 ビットのディスプレイスメントを用いてアクセスするコードを生成します。そのため，4G バイトすべての空間へアクセスが可能になります。

図 B―25　gp オフセット参照セクションのメモリ配置イメージ



したがって，1 命令でアクセス可能な sdata 属性セクション，sbss 属性セクションに，より多くのデータを割り付けた方が，実行効率やオブジェクト効率が良くなります。

データの割り付けは，C ソースの場合は #pragma section 指令，アセンブラ言語ソースの場合は .section 疑似命令によって，ユーザが意図的に配置場所を指定する方法があります。

この方法の他に，sdata 属性セクション，sbss 属性セクションへ割り付けるデータのサイズの閾値を設け，そのサイズ以下のデータを sdata 属性セクション，sbss 属性セクションに配置する指定ができれば，ソース・プログラムに手を加えることなく，より多くのデータを配置することが可能になります。この指定をするのが ca850，as850 で指定する -G*num* オプションです。ここで *num* に指定する値は，データ・サイズになりますが，目安となる値がわかると便利です。

この情報を出力するのが，-A オプションです。

-A オプションがリンカに指定された場合，-G*num* オプションの *num* の設定において目安として用いることのできる情報を出力します。

（b）出力情報の説明

（-r オプションを指定しない）実行可能なオブジェクト・ファイルの生成時にこのオプションを指定した場合の出力情報の例と，（-r オプションを指定した）リロケータブルなオブジェクト・ファイルの生成時にこのオプションを指定した場合の出力情報の例を，次に示します。

例 1. 実行可能なオブジェクト・ファイルに対する出力情報

```
        ******** LINK EDITOR GP INFORMATION ********
   (1)        (2)           (3)           (4)            (5)         (6)
GP SYMBOL    SECTION      SECTION      SECTION          GP
NAME         NAME         SIZE(REAL)   SIZE(ASSUMED)    NUMBER
_gp_DATA
             .sdata       0x000af10
                                       0x00002000       4           *OK*
                                       0x00003450       8           *OK*
                                       0x00004430       12          *OK*
                                       0x000050a8       16          *OK*
                                       0x00007b40       20          *OK*
                                       0x0000a010       24
                                       0x0000af10       32
             .sbss        0x00012050
                                       0x00000050       4           *OK*
                                       0x00002050       16          *OK*
                                       0x00007050       512         *OK*
                                       0x00010050       1024
```

2. リロケータブルなオブジェクト・ファイルに対する出力情報

```
        ******** LINK EDITOR GP INFORMATION ********
   (1)        (2)           (3)           (4)            (5)         (6)
GP SYMBOL    SECTION      SECTION      SECTION          GP
NAME         NAME         SIZE(REAL)   SIZE(ASSUMED)    NUMBER
*(NOT AVAILABLE)
             .sdata       0x000af10
                                       0x00002000       4           *OK*
                                       0x00003450       8           *OK*
                                       0x00004430       12          *OK*
                                       0x000050a8       16          *OK*
                                       0x00007b40       20          *OK*
                                       0x0000a010       24
                                       0x0000af10       32
             .sbss        0x00012050
                                       0x00000050       4           *OK*
                                       0x00002050       16          *OK*
             *GpCommon*   0x00010000
                                       0x00005000       512         *OK*
                                       0x00010000       1024
```

項番	説明
(1)	グローバル・ポインタ・シンボルの名前 リンク時に用いられたグローバル・ポインタ・シンボルの名前。リロケータブルなオブジェクト・ファイルの場合，“*(NOT AVAILABLE)*”が表示されます。
(2)	セクション名 データが割り付けられた sdata ／ sbss 属性セクションの名前。リロケータブルなオブジェクト・ファイルでは未定義外部シンボルのセクションへの割り付けを確定することはできないため，リンカは，仮想的なセクション“*GpCommon*”を内部的に生成し，このセクションに一時的に割り付けます。
(3)	セクションの実サイズ データの整列によって生じるホールの領域などが考慮されている，“セクションの実サイズ”。
(4)	想定されるセクションのサイズ この欄の右の欄で示される値を *num* として -G*num* オプションを指定して ca850 を起動した場合に想定されるセクションのサイズ。このサイズの計算では，各データに対し実際の整列条件は考慮せず一律に 4 バイト以上の整列条件を想定しているため，必ずしも実際に生成されるセクションの実サイズに一致するとはかぎりません。
(5)	想定された -G*num* オプションの *num* の値 この欄の左の欄で示される，“想定されるセクションのサイズ”の計算において想定された，ca850，および as850 起動時における -G*num* オプションの *num* の値。
(6)	判定結果 この欄の左の欄で示される値を *num* として -G*num* オプションを指定し，ca850 を起動した場合にセクションのサイズが 15 ビット（0x0 ～ 0x7fff）の範囲に入るかどうかの判断結果[注]。入る場合“*OK*”が表示され，入らない場合は何も表示されません。

注 C コンパイラでは，通常，データの割り付けられるセクションは data ／ sdata ／ sbss ／ bss 属性セクションの順に下位のアドレスから割り付けられ，グローバル・ポインタ（gp）は sdata 属性セクションの先頭アドレス＋ 32K バイトを指すようスタート・アップ・モジュールなどにおいて設定されることが想定されているため，この判定で OK が出た場合 sdata ／ sbss 属性セクションは 16 ビットのディスプレースメントを用いて参照することのできるメモリの範囲に割り付けられていると考えることができます。

（c）注意事項

このオプションで出力される情報は，あくまで目安であり，たとえば，次のような場合，判定結果が正しくなくなる可能性があります。

- リンク・ディレクティブなどにおいて，ホールを生成するようなセクションの配置を指定した
- グローバル・ポインタ・シンボルに対し直接アドレスを指定した
- #pragma section 指令で，.sdata ／ .sbss セクションにデータを割り当てた

（d）使用例

```
C:¥>ld850 -A file1.o file2.o
```

file1.o と file2.o をリンクし，コンパイル時，およびアセンブル時に ca850，および as850 に対して指定することのできる -G*num* オプションの *num* の設定において目安として用いることのできる情報を標準出力に出力します。

（2）アーカイブ・ファイル

アーカイブ・ファイルは，アーカイバにおいて複数のオブジェクト・ファイルを結合することによって生成されます。

リンカは，アーカイブ・ファイルが指定された時点において未解決な外部参照についてアーカイブ・ファイルを検索し[注 1]，必要とされるオブジェクト・ファイルのみをリンクします。

アーカイブ・ファイルは，リンク・ディレクティブのマッピング・ディレクティブでも指定できます。マッピング・ディレクティブにおいて指定された場合も，その指定された時点において未解決な外部参照について検索され，必要とされるオブジェクト・ファイル[注 2]のみがリンクされます。

注 1. アーカイブ・ファイルは，それが含んでいる各オブジェクト・ファイルに属すシンボルのシンボル・テーブルを持っており，そのアーカイブ・ファイルにより未解決な外部参照が解決されなくなるまで繰り返し検索されます。

2. 参照されているシンボルが定義されているオブジェクト・ファイルです。

（3）予約シンボル

リンカは，リンクの処理において，各出力セクションの先頭アドレス，各出力セクションの終端を越える最初のアドレス，および生成された実行可能なオブジェクト・ファイルの終端を越える最初のアドレス値を値として持つ予約シンボルを生成します。

ユーザがこれらの予約シンボルと同名のシンボルを定義した場合，リンカは定義されたシンボルを用い，独自に生成することはしません。

セクションの先頭アドレス値を値として持つ予約シンボルとしては，その出力セクションの名前の頭に“__s”を付けることによって構成される名前のシンボルが用いられます。

ただし，そのセクション名が“.”で始まっている場合，その“.”を取った後ろの名前の頭に“__s”を付けることによって構成される名前のシンボルが用いられます。セクションの終端を越える最初のアドレス値を値として持つ予約シンボルとしては，その出力セクションの名前の頭に“__e”を付けることによって構成される名前のシンボルが用いられます。

ただし，そのセクション名が“.”で始まっている場合，その“.”を取った後ろの名前の頭に“__e”を付けることによって構成される名前のシンボルが用いられます。生成された実行可能なオブジェクト・ファイルの終端を越える最初のアドレス値を値として持つ予約シンボルとしては，__end が用いられます。

リンカの用いるデフォルトのリンク・ディレクティブでは出力セクションとして次の予約セクションが用いられています。

表 B―12 予約セクション

.text，.pro_epi_runtime，.data，.sdata，.sbss，.bss，.sconst，.const，.sedata，.sebss，.sidata，.sibss，.tidata，.tibss，.tidata.byte，.tibss.byte，.tidata.word，.tibss.word

このため，リンカは通常，次に示した予約シンボルを生成することになります。

表 B―13 通常のオブジェクト・ファイルにおける特殊シンボル

__end，__ebss，__econst，__edata，__epro_epi_runtime，__esbss，__esconst，__esdata，__esebss，__esedata，__esibss，__esidata，__etext，__etibss，__etibss.byte，__etibss.word，__etidata，__etidata.byte，__etidata.word，__sbss，__sconst，__sdata，__spro_epi_runtime，__ssbss，__ssconst，__ssdata，__ssebss，__ssedata，__ssibss，__ssidata，__stibss，__stibss.byte，__stibss.word，__stidata，__stidata.byte，__stidata.word

注意 生成するシンボルは，上記のうち，リンク処理後の実行形式ファイルにセクションが存在するもののみとなります。リンカでは，リンク・ディレクティブ・ファイルに対してマッピング・ディレクティブを記述しても，実際に割り付けられるセクションが存在しなければ，セクションは存在しないものとして扱います。

（4）期待したセクションに割り付けられない場合

リンク・ディレクティブ・ファイル内で，セクションに割り付けるオブジェクト・ファイルやアーカイブ・ファイルを指定しても，ファイル名の記述の仕方によってはそれらが期待したセクションに割り付けられないことがあります。その場合，リンク・マップ（-m）を参照しながら，リンク・マップで表示されているファイル名と，パス名も含めてまったく同じ名前でリンク・ディレクティブ・ファイルに指定し，再リンクしてください。

（5）V850 コアと V850Ex コア

V850Ex は，他の V850 コア・マイクロプロセッサに対して上位互換です。V850 コアで使用していたソース・プログラムを，V850Ex で使用できます。その際，V850 コア・オブジェクトは，as850 のオプションにより，コア内共通のオブジェクト・ファイルとして作成してください。

なお，“V850Ex 内共通”として作成したオブジェクト・ファイルは，V850Ex，V850E2 以外のオブジェクト・ファイルとリンクすることはできません。

詳細は，「（1）マジック・ナンバ」を参照してください。

（6）V850 コアと V850E2 コア

V850E2 は，他の V850 コア・マイクロプロセッサに対して上位互換です。V850 コアで使用していたソース・プログラムを，V850E2 で使用できます。その際，V850 コア・オブジェクトは，as850 のオプションにより，コア内共通のオブジェクト・ファイルとして作成してください。

なお，“V850E2 内共通”として作成したオブジェクト・ファイルは，V850E2 以外のオブジェクト・ファイルとリンクすることはできません。

詳細は，「（1）マジック・ナンバ」を参照してください。

（7）数学ライブラリ

数学ライブラリ関数をプログラム中に使用し，リンク時に数学ライブラリ（libm.a）をリンクしても，undefined symbol などのエラーが出ることがあります。これは標準ライブラリなどとのリンク順番に関係していることがあります。ANSI 準拠の順番でリンクする必要がありますので，標準ライブラリを最後にリンクしてください。特にコマンド・ラインからリンカを起動するときには注意してください。具体的には -lm，-lc の順番にオプションを記述してください。

（8）main 関数

main 関数を作らずにリンクをした場合，_main シンボルが undefined symbol エラーとして出力されることがあります。特にスタート・アップ・ルーチンを独自に指定せず，デフォルトのスタート・アップ・ルーチン（crtN.o，crtE.o【V850E】）がリンクされた場合，またはパッケージに用意されている crtN.s，crtE.s をそのままアセンブル，リンクして使用した場合に発生します。これは crtN.s や crtE.s の最後の方に書かれてある「jarl _main, lp」というコードが原因です。main 関数が必要ない場合，ここを独自に書き換え，再アセンブルして作成したオブジェクトをスタート・アップ・ルーチンとしてください。また，リアルタイム OS を使用したアプリケーションの場合，通常 main 関数はありません。リアルタイム OS のサンプルとして提供されているスタート・アップ・ルーチンを使用してください。

（9）プロローグ／エピローグ・ランタイム・ライブラリ

プロローグ／エピローグ・ランタイム・ライブラリは，専用の .pro_epi_runtime セクションに配置する必要があります。配置していない場合，次のメッセージを出力し，リンクを中止します。

```
F4286 : section ".pro_epi_runtime" must be specified in link directive.
```

リンク・ディレクティブ・ファイルを指定している場合には，.text セクションの手前にマッピング・ディレクティブを記述してください。

```
.pro_epi_runtime = $PROGBITS ?AX .pro_epi_runtime;
.text = $PROGBITS ?AX;
```

.pro_epi_runtime セクションを .text セクションの後ろに配置すると，ROM 化の際，パッキングされたセクションのデフォルト動作での配置位置と重なります。.text セクションの手前に配置することを推奨します。リンク・ディレクティブ・ファイルを指定しない場合には，.text セクションの手前にリンクします。

（a）注意事項

- プロローグ／エピローグ・ランタイム・ライブラリは，標準ライブラリ libc.a に含まれています。
- .pro_epi_runtime セクションは通常のセクションと異なり，入力セクション名が固定されており，専用のセクションだけが配置されます。
- .pro_epi_runtime セクションを .text セクションの後ろに配置すると，ROM 化する場合にパッキングされたセクションのデフォルト動作での配置位置と重なります。.text セクションの手前に配置してください。
- V850Ex ／ V850E2 コアのデバイス指定時にはプロローグ／エピローグ・ランタイム・ライブラリは，callt 命令を使用しています。スタート・アップ・ルーチンで CTBP の設定を行ってください。

（10）ROM 化のためのリンク

ROM 化を行う場合は，パッキング・セクション領域を考慮したリンク・ディレクティブを記述する必要があります。詳細については，「B. 4 ROM 化プロセッサ」を参照してください。

デフォルトのリンク・ディレクティブを使用し，CONST セグメントを利用する場合，ROM 化がうまくできません。それは，デフォルトのリンク・ディレクティブでは，CONST セグメントを TEXT セグメントの直後に配置しているため，ROM 化プロセッサがデフォルトの動作によって，パッキング・セクション（rompsec セクション）と CONST セグメントが重なってしまうためです。パッケージに添付されているサンプル・ディレクティブ[注]を参考にし，次のいずれかの対応を行ってください。

注 “インストール・フォルダ¥ CubeSuite+ ¥ CA850 ¥ V*x.xx* ¥ smp850 ¥ ca850”に格納されている“v850def.dir ／ v850def2.dir ／ v850def3.dir” です。

v850def.dir	内蔵 ROM ／ RAM，外部 RAM を使ったサンプル
v850def2.dir	内蔵 ROM ／ RAM だけを使ったサンプル
v850def3.dir	内蔵 ROM ／ RAM，外部 RAM，内蔵命令 RAM を使ったサンプル（V850E/ME2 など）

また，使用するマイクロプロセッサにあわせたメモリ配置を行う必要があります。CONST セグメントを TEXT セグメントの手前に配置します。

```
CONST : !LOAD ?R{
        .const = $PROGBITS ?A .const;
};

TEXT : !LOAD ?RX{
        .text = $PROGBITS ?AX;
};
```

TEXT セグメントの後ろにパッキング・セクションの領域（「B. 4　ROM 化プロセッサ」参照）を確保し，その後ろに CONST セグメントを配置します。

```
TEXT : !LOAD ?RX{
        .text = $PROGBITS ?AX;
};

        [ パッキング・セクションの領域 ]

CONST : !LOAD ?R V0x200000{ ←パッキング・セクションを考慮したアドレスを指定
        .const = $PROGBITS ?A .const;
};
```

（11）プログラマブル周辺 I/O レジスタ

プログラマブル周辺 I/O レジスタ機能を使用するアプリケーション・プログラムの場合，アセンブル時に予約セクションの .bpc セクションが出力されます。リンカは，リンクの入力オブジェクト・ファイルに .bpc セクションが存在する場合，BPC 値として指定されている値のチェックを行います。入力オブジェクト・ファイル間で値が統一されていない場合，リンカは次のようなエラー・メッセージを出力し，リンク処理を中断します。

```
F4457: input files have different BPC value.
0x00001234      file1.o
0x00001234      file2.o
0x00001235      file3.o
*(none)*        file4.o
```

上記の場合，file3.o で設定されている値が異なるため，エラーとなっています。

なお，プログラマブル周辺 I/O レジスタを参照していないオブジェクトは，チェックの対象とはなりません。上記の file4.o のように“*(none)*”と表示されます。

BPC 値のチェックでエラーがなかった場合，セクション・タイプ SHT_PROGBITS，セクション属性 none，セクション・サイズ 0x4 の .bpc セクションが生成されます。.bpc セクションには，BPC 値を既定ビット数分シフトして得られる，プログラマブル I/O レジスタ領域の先頭アドレスが格納されます。

例 V850E/IA1 使用時に BPC 値を“0x1234”と指定した場合，プログラマブル周辺 I/O レジスタ領域の先頭アドレスは，この値を 14 ビット左シフトした“0x48d0000”となります。この際，.bpc セクション内の情報は次のようになります。

```
.bpc
  Address    00  01  02  03  04  05  06  07  -  08  09  0A  0B  0C  0D  0E  0F
0x00000000 : 00  00  8d  04                  -                              ...
```

- 以上の処理は，リロケータブルなオブジェクト・ファイル作成時，実行可能オブジェクト・ファイル作成時を問わず行われます。
- .bpc セクションは，情報用の特殊な予約セクションであり，メモリにロードされることはありません。したがって，通常のセクションのように，リンク・ディレクティブに記述する必要はありません。

（12）オプション・バイト

オプション・バイト機能を利用するにはアセンブラ・ソースに次のような 6 バイトのデータを記述します。

```
.section "OPTION_BYTES"
.byte   0b00000001      -- 0x7a
.byte   0b00000000      -- 0x7b
.byte   0b00000000      -- 0x7c
.byte   0b00000000      -- 0x7d
.byte   0b00000000      -- 0x7e
.byte   0b00000000      -- 0x7f
```

- オプション・バイト機能を持たないデバイスを指定した場合は，通常の入力セクションとして扱います。
- オプション・バイト機能を持つデバイスを指定し，本セクションの記述を省略した場合は，デバイス・ファイルに設定された初期値を使用します。
- 本セクションは必ず 6 バイト分を記述してください。6 バイト以下の場合は，次のメッセージを出力し，リンクを中止します。

```
F4112: illegal "section" section size.
```

- 設定不可能となっているビットは，初期値からの変更ができません。変更された場合は，次のメッセージを出力します。

```
W4613: illegal flash mask option access (file:"file" address:num1 bit:num2)
```

B. 4　ROM 化プロセッサ

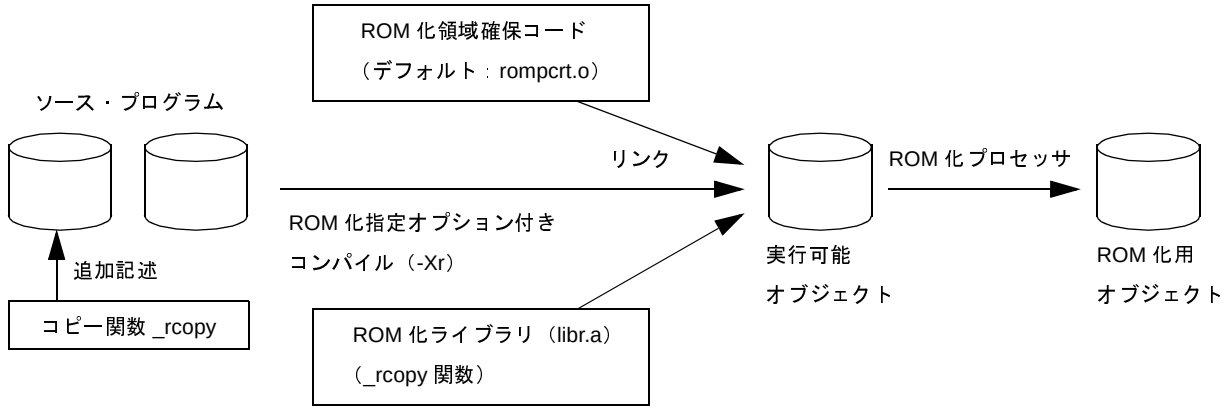
プログラム中で，グローバルに変数を宣言すると，初期値を持つ変数ならば data 属性のセクションへ，初期値を持たない変数ならば bss 属性のセクションというように，RAM 上のセクションに配置されます。特に初期値を持つ変数ならば，その初期値自体が RAM 上に配置されます。その他，アプリケーションの高速化のために，プログラム・コードを内蔵 RAM 領域へ配置する場合もあります。

組み込みシステムの場合，デバッグ時にインサーキット・エミュレータなどを使用する場合，実行可能なモジュールを配置イメージのままダウンロードして実行できます。しかし実際にプログラムをターゲット・システムの ROM 領域に書き込んで実行する場合，data 属性のセクションにある初期値情報や，RAM 領域に配置するプログラム・コードを，実行前に RAM 上に展開されていなければなりません。つまり RAM に展開するデータを ROM 上に持たせておき，それをアプリケーション実行前に ROM から RAM へコピーする作業が必要になります。

ROM 化プロセッサ（romp850）は，data 属性セクションの変数の初期値情報や，RAM 上に配置するプログラムを，1 つのセクションにパッキングするツールです。このセクションを ROM 上に配置し，CA850 で用意されているコピー関数を呼び出すことによって，初期値情報やプログラムを容易に RAM 上へ展開することができます。

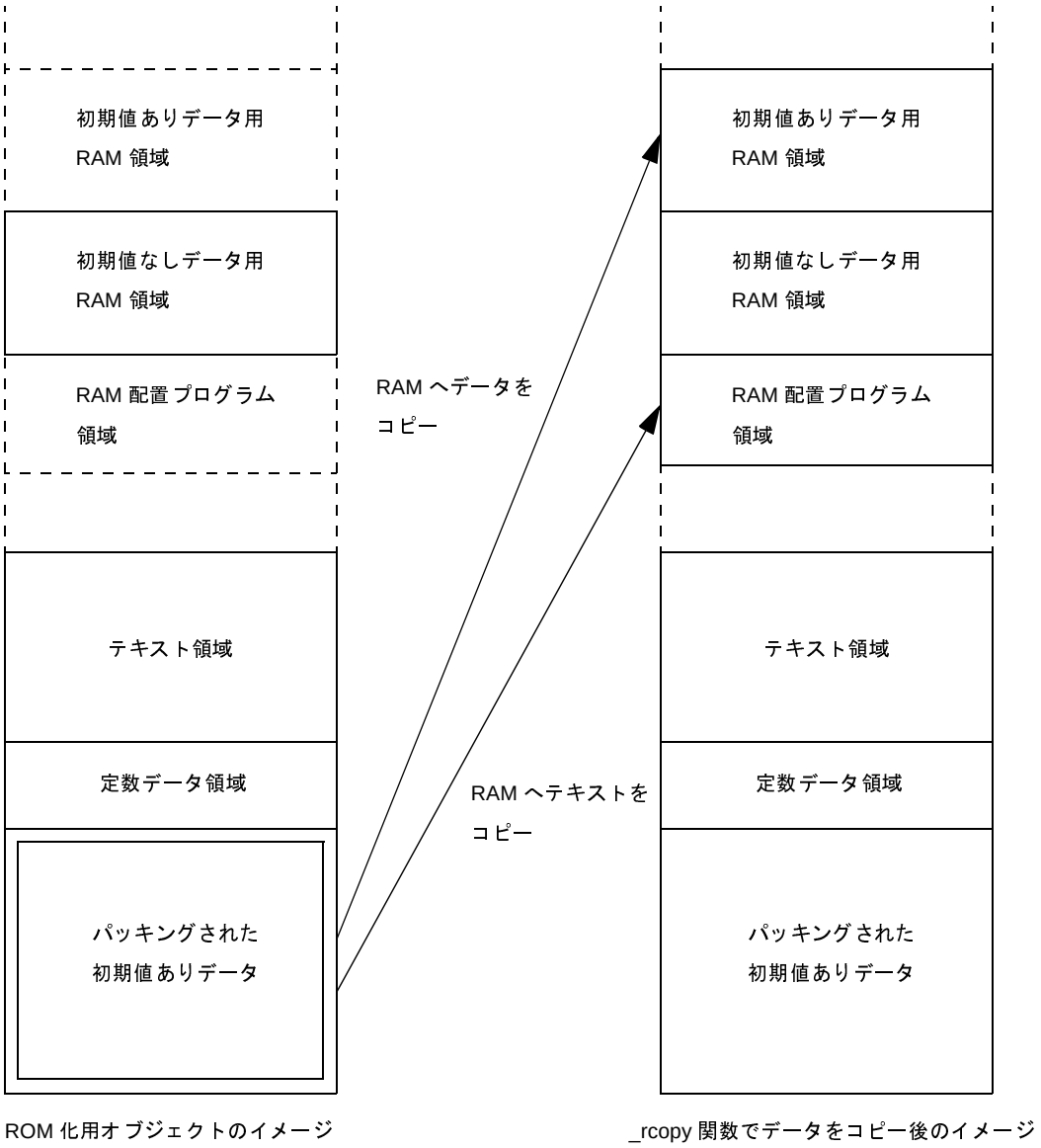
ROM 化用オブジェクトを作成する流れの概要は，次図のようになります。

図 B―26　ROM 化用オブジェクトの作成



図のように ROM 化用オブジェクトを作成すると，_rcopy 関数を実行することによって，RAM に配置するデータを，パッキングされた ROM からコピーします。イメージは次のようになります。

図 B—27 _rcopy 関数呼び出し前後のイメージ



ここで，ROM 化用オブジェクトに必要となるセクション名，およびそのセクションの先頭アドレス（ラベル名）は，デフォルトでは次のようになっています。

- パッキングしたセクション名→ rompsec セクション
- rompsec セクションの先頭アドレス（ラベル名）→ __S_romp

そして rompsec セクションから，RAM 領域へコピーする関数は次のとおりです。

- コピー関数→ _rcopy，_rcopy1，_rcopy2，_rcopy4 関数

この関数は lib850¥r** にあるライブラリ“libr.a”に格納されています。

__S_romp は lib850¥r** にある“rompcrt.o”で定義されているラベルです（このソース・ファイルは rompcrt.s）。rompcrt.o をそのまま使用することにより，ROM 化プロセッサによって自動的に .text 属性の直後（4 バイトでアライ

ンしたところ）に，rompsec セクションを作成します。そして __S_romp が rompsec セクションの先頭アドレスを指すラベルになります。

このように自動的に rompsec セクションを作成する方法のほかに，rompcrt.s に相当するプログラムを独自に作成して配置することもできます。詳しくは，「（2）作成手順（カスタマイズ）」を参照してください。

実際に ROM 化するには，この ROM 化用オブジェクトを作成してから，ヘキサ・ファイルに変換し，ROM 上に書き込むことになります。

なお，パッキングの必要なデータがアプリケーションに存在しなかった場合は，この ROM 化用オブジェクトを生成する必要はありません。ld850 で作成したオブジェクトを，そのままヘキサ・ファイルに変換してください。

また，ROM 化プロセッサは，リロケーション解決したオブジェクト・ファイルにシンボル情報，デバッグ情報が含まれる場合，それらを削除することなく ROM 化用のオブジェクト・ファイルを生成します。そのため，ROM 化後のオブジェクト・ファイルでもデバッガによるソース・デバッグができます。

B. 4. 1　入出力ファイル

ROM 化プロセッサでは，次のファイルを入力ファイルとして扱うことができます。

file1.out	ld850 によって出力された実行可能オブジェクト

出力されるファイルは，次のファイルです。

file2.out	ROM 化用実行可能オブジェクト

入出力ファイル名は，ld850，ROM 化プロセッサにてそれぞれで指定できます。出力ファイルのデフォルトは romp.out です。

B. 4. 2　rompsec セクション

（1）パッキングするセクションの種類

rompsec セクションとしてパッキングする対象となるものは，デフォルトでは「書き込み可能な属性を持つセクションに割り当てられたデータ」です。また，V850/V850E1 コアを持つデバイス指定時には，内蔵命令 RAM に配置するセクションもパッキング対象となります（V850E2 コアを持つデバイス指定時は対象となりません）。その他，オプション（-t オプション）を指定することによって「text 属性，const 属性を持つ任意のセクション」をパッキングすることも可能です。

具体的には次に示すようになります。

- 「表 B—14　ROM 化プロセッサによりパッキングされる予約セクション」に示す予約セクション
- アセンブラ・プログラムにおいて .section 疑似命令により sdata 属性，または data 属性を指定し，任意の名前で生成したセクション，および内蔵命令 RAM に配置するセクション

表 B—14　ROM 化プロセッサによりパッキングされる予約セクション

.data，.sdata，.sedata，.sidata，.tidata，.tidata.byte，.tidata.word

ただし，ユーザが指定する「text 属性，const 属性を持つ任意のセクション」をパッキングせず，さらに上記のセクションが実行可能モジュールに存在しない場合は，ROM 化オブジェクトを生成する必要はありません。「表 B—14　ROM 化プロセッサによりパッキングされる予約セクション」のセクションが存在するかしないかは，リンク・マップ・ファイルを参照してください。

なお，ROM 化プロセッサによって作成されたオブジェクト・ファイルをダンプ・ツールで参照することにより，.data セクションや .sdata セクション，内蔵 RAM に配置したセクション（割り込みハンドラのセクションも含む）などの代わりに rompsec セクションが作成されていることを確認できます。

（2）rompsec セクションのサイズ

rompsec セクションとして確保されるサイズについて説明します。

ROM 化用モジュールを作成するときは，rompsec セクションのサイズと，使用している CPU の内蔵 ROM 領域，ターゲット・システムの ROM 領域のアドレスやサイズに注意します。rompsec セクションが，他のセクションとオーバラップしないようにリンク・ディレクティブ・ファイルを記述してください。具体的な記述例は，「B. 4. 3　ROM 化用オブジェクトの作成」を参照してください。

次に rompsec セクションのサイズを求める計算式を示します。

8 ＋ 16 ×（sdata ／ data 属性セクションの数）＋ sdata ／ data 属性セクションのサイズ
＋パディング・サイズ[注]

たとえば，.sdata，.data セクションが存在し，それぞれのサイズが 1002 バイト，1000 バイトで，それぞれのセクションの整列条件が 4 バイトの場合，rompsec セクションのサイズは次のようになります。

8 ＋ 16 × 2 ＋ 1002 ＋ 1000 ＋ 2 ＝ 2044 バイト

注　ROM 化対象のセクションの整列条件により 1 セクションあたり 0 ～ 3 バイトになります。

（3）rompsec セクションとリンク・ディレクティブ

ROM 化時には，.text セクションの直後に rompsec セクションが追加されます。よって，.text セクションを ROM の最後に配置するか，rompsec セクションを明示的に ROM の終端に指定することで，ROM の終端までの rompsec セクションを配置できます。

- ROM 化処理を考慮したリンク・ディレクティブ

```
# 内蔵 ROM に SCONST ／ CONST ／ TEXT を配置
SCONST : !LOAD ?R {
        .sconst = $PROGBITS ?A .sconst;
};

CONST : !LOAD ?R {
        .const = $PROGBITS ?A .const;
};

# 内蔵 ROM の最後に .text を配置
TEXT : !LOAD ?RX {
        .pro_epi_runtime = $PROGBITS ?AX .pro_epi_runtime;
```

```
        .text = $PROGBITS ?AX .text;
        rompsec = $PROGBITS ?AX .text { rompcrt.o };
};

# 外部 RAM に DATA を配置
DATA : !LOAD ?RW V0x100000 {
        .data = $PROGBITS ?AW;
        .sdata = $PROGBITS ?AWG;
        .sbss = $NOBIT ?AWG;
        .bss = $NOBIT ?AW;
};

# 内蔵 RAM に SIDATA を配置
SIDATA : !LOAD ?RW V0xffe000 {
        .sidata = $PROGBITS ?AW .sidata;
        .sibss = $NOBIT ?AWG .sibss;
};

__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA};
__ep_DATA @ %EP_SYMBOL;
```

rompsec セクションが内蔵 ROM 領域を越えた場合には，次のメッセージを出力して処理を中止します。

```
F8425: rompsec section overflowed highest address of target machine.
```

-rom_less オプションを指定することで，内蔵 ROM 領域を無視することができます。

また，-Ximem_overflow=warning オプションを指定することで，エラー・メッセージを警告メッセージにすることができます。

rompsec セクションを外部 ROM 領域の終端に配置する場合には，これらのチェックは行われません。メモリマップ情報を参照して，ROM に収まっているかの判断を行ってください。

なお，ROM の途中に rompsec セクションを配置する必要がある場合には，次のように rompsec セクションのサイズと配置アドレスから，rompsec セクションの配置される領域を認識した上で，rompsec セクション直後のセグメントに対して，適切なアドレス指定をしてください。

- ROM 化処理を考慮したリンク・ディレクティブ（サイズ考慮）

```
# 内蔵 ROM に SCONST ／ CONST ／ TEXT を配置
SCONST : !LOAD ?R {
        .sconst = $PROGBITS ?A .sconst;
};

# 内蔵 ROM の途中に .text を配置
TEXT : !LOAD ?RX {
```

```
        .pro_epi_runtime = $PROGBITS ?AX .pro_epi_runtime;
        .text = $PROGBITS ?AX .text;
        rompsec = $PROGBITS ?AX .text { rompcrt.o };
};

# TEXT と CONST の間に rompsec

# 内蔵 ROM の最後に rompsec のサイズを考慮したアドレス指定を行って CONST を配置
CONST : !LOAD ?R Vx3f800 {
        .const = $PROGBITS ?A .const;
};

# 外部 RAM に DATA を配置
DATA : !LOAD ?RW V0x100000 {
        .data = $PROGBITS ?AW;
        .sdata = $PROGBITS ?AWG;
        .sbss = $NOBIT ?AWG;
        .bss = $NOBIT ?AW;
};

# 内蔵 RAM に SIDATA を配置
SIDATA : !LOAD ?RW V0xffe000 {
        .sidata = $PROGBITS ?AW .sidata;
        .sibss = $NOBIT ?AWG .sibss;
};
__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA};
__ep_DATA @ %EP_SYMBOL;
```

B. 4. 3　ROM 化用オブジェクトの作成

（1）作成手順（デフォルト）

ここでは，デフォルトで用意されている ROM 化用領域確保コード（rompcrt.o）を使用した方法を示します。

（a）アプリケーションの中で，コピー関数を呼び出します。

コピー関数はスタート・アップ・ルーチン内や main 関数の先頭など，なるべく始めの方で起動するようにします。コピー関数には _rcopy，_rcopy1，_rcopy2，_rcopy4 があり，それぞれ転送サイズに違いがあります（_rcopy と _rcopy1 は同じ）。これらについての詳細は「B. 4. 4　コピー関数」を参照してください。

以下の例では，main 関数の先頭で _rcopy 関数を起動しています。

例　コピー関数 _rcopy の使用例

```
#define ALL_COPY(-1)

int _rcopy(unsigned long *, long);
extern unsigned long _S_romp;

void main(void)
{
        int     ret;

ret = _rcopy(&_S_romp, ALL_COPY);

                :
}
```

（b）ROM 化時には，.text セクションの直後に rompsec セクションが追加されます。

.text セクションを ROM の最後に配置することで，ROM の終端までの rompsec セクションを配置できます（「（3）rompsec セクションとリンク・ディレクティブ」参照）。

（c）コンパイル・オプションで“ROM 化用オブジェクトの生成”を指定します。

- コマンド・ラインからの場合

コンパイル・オプション“-Xr”を追加指定します。

- CubeSuite+ からの場合

プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択します。

図 B―28　［ROM 化用オブジェクト・ファイルを出力する］プロパティ

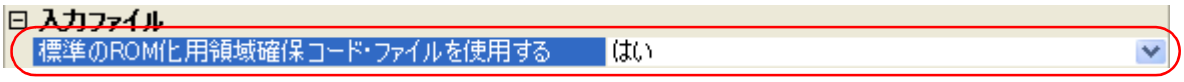
これにより，__S_romp というラベルが，オブジェクト内の .text セクションの終端を越える最初のアドレスを指すコードが生成されます。

（d）ROM 化プロセス・オプションを指定します。

- CubeSuite+ からの場合

プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［入力ファイル］カテゴリの［標準の ROM 化用領域確保コード・ファイルを使用する］プロパティで［はい］（デフォルト）を選択します。

図 B―29　［標準の ROM 化用領域確保コード・ファイルを使用する］プロパティ



（e）コンパイル，リンクを行います。

ca850 に対して ROM 化用オブジェクトの生成を指示することにより，ROM 化用領域確保コードである“rompcrt.o”（lib850￥r** 内に存在）と，_rcopy 関数が格納されている“libr.a”が自動的にリンクされます。この際，リンクする順番が関係します。“rompcrt.o”は，TEXT 属性群の最後にリンクする必要がありますので，コマンド・ラインから起動している場合，-l オプションでリンク指定するライブラリ群よりも後にリンクしてください。CubeSuite+ を使用している場合は，自動的に TEXT 属性群の最後にリンクするため，特に意識する必要はありません。

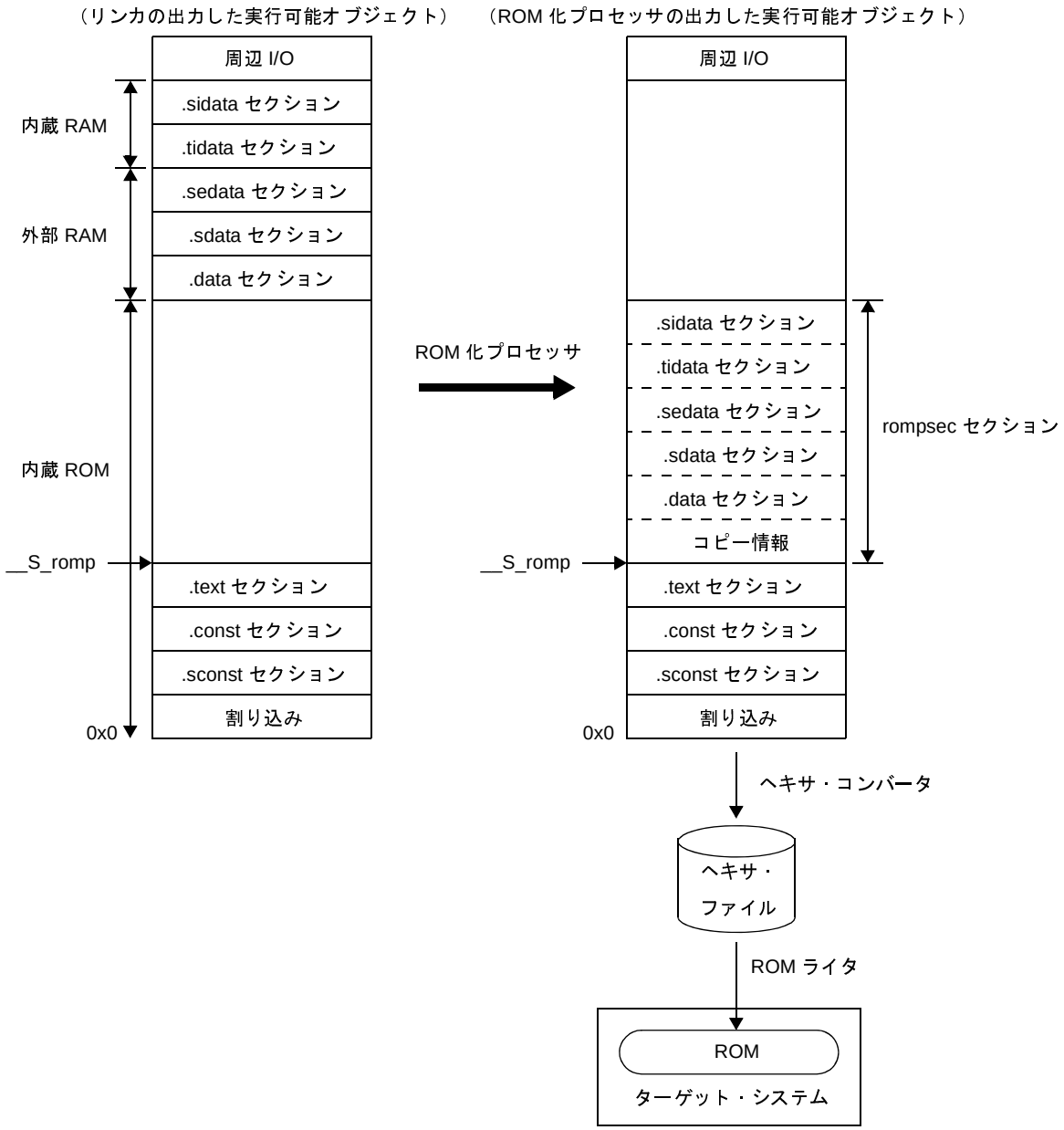
注意　リンカの -rescan オプションを指定した場合，rompcrt.o の後にライブラリがリンクされ，ROM 化プロセッサにて F8426 エラーとなることがあります。その場合は，rompsec セクションの領域を明示的に確保してください（「(3) rompsec セクションとリンク・ディレクティブ」参考）。

（f）ROM 化プロセッサを起動します。

（d）で完成した実行可能モジュールから，ROM 化プロセッサを使用して ROM 化用モジュールを生成します。

なお，CubeSuite+ で ROM 化用オブジェクトの生成を指定している場合は，（d）から（e）まで自動的に行われ，ヘキサ・ファイルまで生成されます。コマンド・ラインから起動している場合は，ca850 から ld850 の起動まで行って実行可能モジュールを生成したあと，ROM 化プロセッサを起動して ROM 化用オブジェクトを生成します。マップのイメージを図にすると，次のようになります。

図 B―30　ROM 化のイメージ 1



（2）作成手順（カスタマイズ）

ここでは，ROM 化用領域確保コードにあたる“rompcrt.o”を独自に作成し，rompcrt セクションの先頭アドレスや配置場所を自分で決定する方法を示します。

（a）デフォルトの ROM 化用領域確保コード“rompcrt.s”に相当するコードを記述します。

ここではファイル名を“rompack.s”，ROM 化用領域の先頭を指すシンボル名を“__rompack”とします。また，このとき，このシンボルの存在するセクションを“rompack セクション”とします。この場合，rompack.s は，次のようなコードになります。

例 rompack.s

```
        .file           "rompack.s"
        .section        ".rompack",text
        .align          4
        .globl          __rompack, 4
__rompack:
```

（b）アプリケーションの中で，コピー関数を呼び出します。

コピー関数はスタート・アップ・ルーチン内や main 関数の先頭など，なるべく始めの方で起動するようにします。コピー関数には _rcopy，_rcopy1，_rcopy2，_rcopy4 があり，それぞれ転送サイズに違いがあります（_rcopy と _rcopy1 は同じ）。これらについての詳細は「B. 4. 4 コピー関数」を参照してください。

以下の例では，main 関数の先頭で _rcopy 関数を起動しています。

例 コピー関数 _rcopy の使用例

```
#define ALL_COPY (-1)

int _rcopy(unsigned long *, long);
extern unsigned long _rompack;

void main(void)
{
        int     ret;

        ret = _rcopy(&_rompack, ALL_COPY);
                :
}
```

（c）リンク・ディレクティブで，作成した rompack セクションを定義します。

これと同時にアドレスを指定すると，rompack セクションの配置場所を任意に決定することもできます。

rompack セクションを含むセグメントを ROMPACK とし，このセグメントを 0x3000 番地に配置するとした場合，リンク・ディレクティブは次のようになります。

例　リンク・ディレクティブの指定例

```
TEXT:    !LOAD ?RX V0x1000 {
        .text = $PROGBITS ?AX .text;
};

ROMPACK: !LOAD ?RX V0x3000 {
        .rompack = $PROGBITS ?AX .rompack;
};
                :
```

このとき，ROMPACK セグメントの配置アドレスが，前後のセグメントと重ならないように，rompack セクションのサイズを，「（2）rompsec セクションのサイズ」にしたがって見積もり，リンク・ディレクティブ・ファイルに反映します。

（d）コンパイル・オプションで“ROM 化用オブジェクトの生成”を指定します。

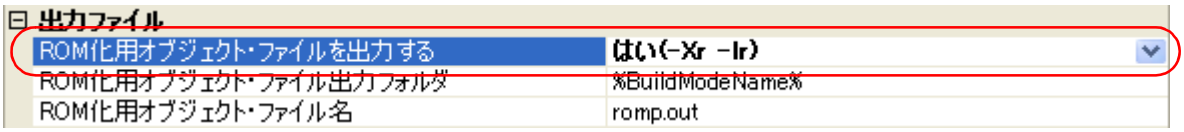
- コマンド・ラインからの場合

コンパイル・オプション“-Xr”を追加指定します。

- CubeSuite+ からの場合

プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［出力ファイル］カテゴリの［ROM 化用オブジェクト・ファイルを出力する］プロパティで［はい (-Xr -lr)］を選択します。

図 B―31　［ROM 化用オブジェクト・ファイルを出力する］プロパティ



これにより，ラベル“rompack”が rompsec と同じアドレスを指すコードを生成します。

（e）ROM 化プロセス・オプションを指定します。

- コマンド・ラインからの場合

ROM 化プロセッサのオプションで，ROM 化用領域確保コードのエントリ・シンボルを指定する“-b オプション”で“__rompack”を指定します。

- CubeSuite+ からの場合

プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［入力ファイル］カテゴリの［標準の ROM 化用領域確保コード・ファイルを使用する］プロパティで［いいえ］を選択し，［ROM 化用領域確保コードのファイル名］プロパティに“rompack.s”，または“rompack.o”を追加します。

図 B―32 ［標準の ROM 化用領域確保コード・ファイルを使用する］，および［ROM 化用領域確保コードのファイル名］プロパティ

［その他］カテゴリの［エントリ・ラベル］プロパティに rompack セクションの先頭ラベルである“__rompack”を指定します。

図 B―33 ［エントリ・ラベル］プロパティ

（f）コンパイル，リンクを行います。

ca850 に対して「ROM 化用オブジェクトの生成」を指示することにより，_rcopy 関数が格納されている“libr.a”が自動的にリンクされます。

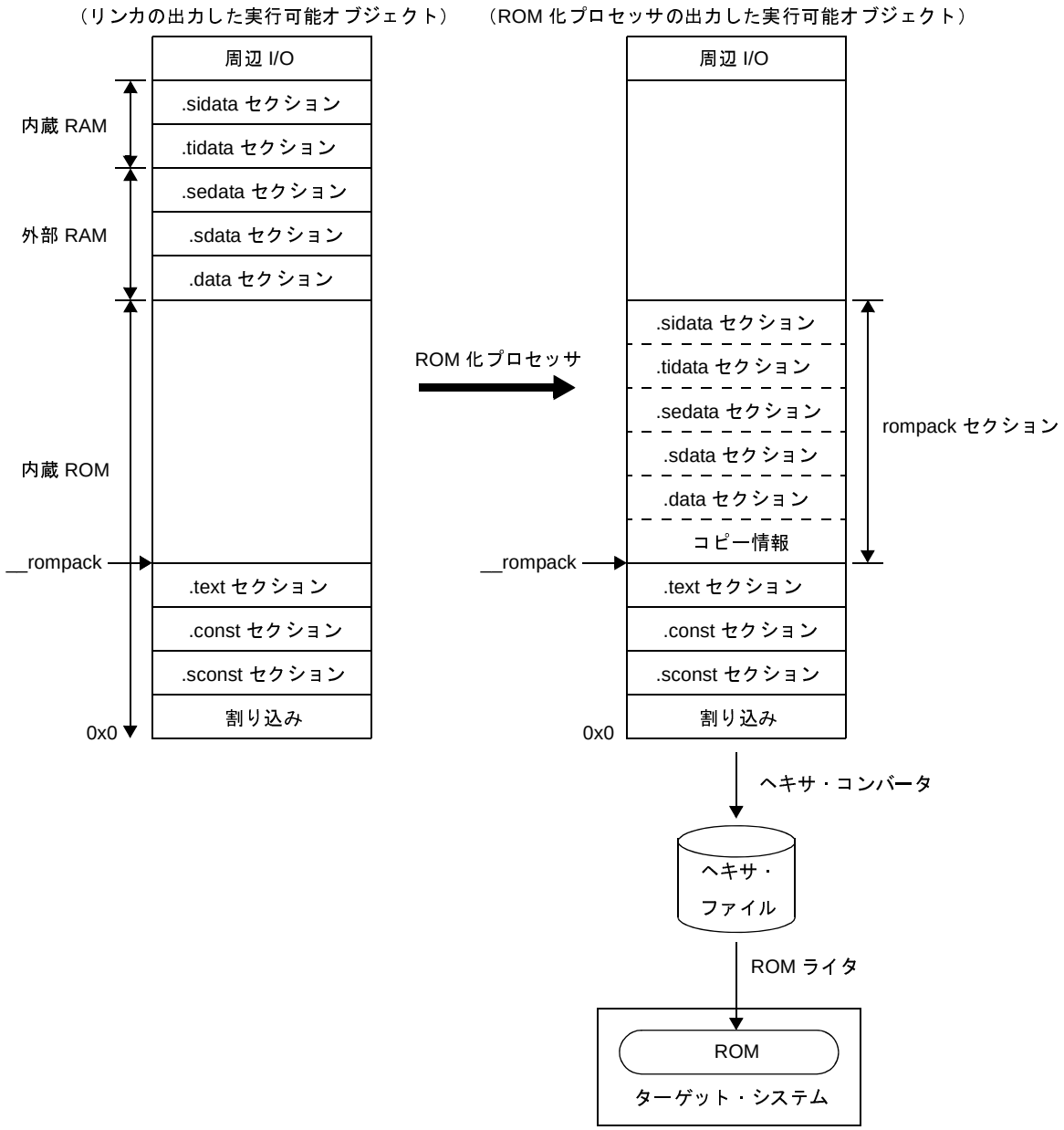
注意 リンカの -rescan オプションを指定した場合，rompcrt.o の後にライブラリがリンクされ，ROM 化プロセッサにて F8426 エラーとなることがあります。その場合は，rompsec セクションの領域を明示的に確保してください（「(3) rompsec セクションとリンク・ディレクティブ」参考）。

（g）ROM 化プロセッサを起動します。

（f）で完成した実行可能モジュールから，ROM 化プロセッサを使用して ROM 化用モジュールを生成します。

なお，CubeSuite+ で ROM 化用オブジェクトの生成を指定している場合は，（f）から（g）まで自動的に行われ，ヘキサ・ファイルまで生成されます。コマンド・ラインから起動している場合は，ca850 から ld850 の起動まで行って実行可能モジュールを生成したあと，ROM 化プロセッサを起動して ROM 化用オブジェクトを生成します。マップのイメージを図にすると，次のようになります。

図 B―34　ROM 化のイメージ 2



B. 4. 4　コピー関数

ここでは，ROM 化するプログラムに必要なコピー・ルーチン（_rcopy）について説明します。

表 B―15　コピー・ルーチン一覧

関数名	機能
_rcopy	ROM 化セクションのコピー（1 バイト転送）
_rcopy1	ROM 化セクションのコピー（1 バイト転送）
_rcopy2	ROM 化セクションのコピー（2 バイト転送）
_rcopy4	ROM 化セクションのコピー（4 バイト転送）

転送先の RAM の仕様に応じて，1 バイト転送，2 バイト転送，4 バイト転送を使い分けてください。各関数の仕様は次のようになります。

_rcopy

[概要]

- 初期値データ／RAM テキスト注のコピー（1 バイト）

注　RAM に配置する初期値ありデータ・セクション，および内蔵 RAM 用テキスト・セクションです。

[形式]

```
int             _rcopy(&label, number)
unsigned long   label;
long            number;
```

[説明]

- _rcopy(&label, number) は，label の示すアドレス以降に存在する rompsec セクション内の情報を元に，コピーしたいセクション番号 number の初期値データ，または RAM に配置するテキストを，RAM 領域に 1 バイトずつコピーします。number に -1 を指定した場合，rompsec セクション内のすべてのセクションをコピーします。セクション番号 number は，1 から始まる正数です。
- デフォルトは，セクションが入力ファイル中に出現した順番に割り当てられます。ROM 化プロセッサのオプション“-p オプション”，“-t オプション”で rompsec セクションに配置するセクションを指定した場合は，指定した順番に割り当てられます。
- CubeSuite+ において，プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［セクション・リスト］カテゴリの［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択すると，#define による“番号”と“ラベル”の対応付けされた C ソース・ヘッダ・ファイルが生成され，ラベル名によって，number を指定することができます。
- 具体的な使用例については，「B. 4. 5　コピー関数の使用例」を参照してください。

[戻り値]

0	正常終了（正しくコピーされた場合）
-1	異常終了（正しくコピーされなかった場合）

[注意事項]

- label の示すアドレスが rompsec セクションの先頭でなかった場合はコピーを行いません。
- _rcopy は ROM 化プロセッサで生成された情報にしたがってコピーを行います。
  _rcopy 実行時にコピー先のアドレスにオフセットを加えるような処理はできません。
- コピーを行うとオーバライトが生じる場合，コピーを行いません。

- _rcopy の第一引数 label には，絶対値を持つグローバルなラベル，または絶対アドレスを指定してください。これら以外のものを指定した場合，その結果は保証されません。
- _rcopy 関数は _rcopy1 関数と同じ機能です。旧版からの互換性のために _rcopy を用意してあります。

_rcopy1

[概要]

- 初期値データ／RAM テキスト注のコピー（1 バイト）

注　RAM に配置する初期値ありデータ・セクション，および内蔵 RAM 用テキスト・セクションです。

[形式]

```
int             _rcopy1(&label, number)
unsigned long   label;
long            number;
```

[説明]

- _rcopy1(&label, number) は，label の示すアドレス以降に存在する rompsec セクション内の情報を元に，コピーしたいセクション番号 number の初期値データ，または RAM に配置するテキストを，RAM 領域に 1 バイトずつコピーします。number に -1 を指定した場合，rompsec セクション内のすべてのセクションをコピーします。セクション番号 number は，1 から始まる正数です。
- デフォルトは，セクションが入力ファイル中に出現した順番に割り当てられます。ROM 化プロセッサのオプション“-p オプション”，“-t オプション”で rompsec セクションに配置するセクションを指定した場合は，指定した順番に割り当てられます。
- CubeSuite+ において，プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［セクション・リスト］カテゴリの［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択すると，#define による“番号”と“ラベル”の対応付けされた C ソース・ヘッダ・ファイルが生成され，ラベル名によって，number を指定することができます。
- 具体的な使用例については，「B. 4. 5　コピー関数の使用例」を参照してください。

[戻り値]

0	正常終了（正しくコピーされた場合）
-1	異常終了（正しくコピーされなかった場合）

[注意事項]

- label の示すアドレスが rompsec セクションの先頭でなかった場合はコピーを行いません。
- _rcopy1 は ROM 化プロセッサで生成された情報にしたがってコピーを行います。
  _rcopy1 実行時にコピー先のアドレスにオフセットを加えるような処理はできません。
- コピーを行うとオーバライトが生じる場合，コピーを行いません。

- _rcopy1 の第一引数 label には，絶対値を持つグローバルなラベル，または絶対アドレスを指定してください。これら以外のものを指定した場合，その結果は保証されません。
- _rcopy1 関数は _rcopy 関数と同じ機能です。旧版からの互換性のために _rcopy を用意してあります。

_rcopy2

[概要]

- 初期値データ／RAM テキスト注のコピー（2 バイト）

注　RAM に配置する初期値ありデータ・セクション，および内蔵 RAM 用テキスト・セクションです。

[形式]

```
int             _rcopy2(&label, number)
unsigned long   label;
long            number;
```

[説明]

- _rcopy2(&label, number) は，label の示すアドレス以降に存在する rompsec セクション内の情報を元に，コピーしたいセクション番号 number の初期値データ，または RAM に配置するテキストを，RAM 領域に 2 バイトずつコピーします。number に -1 を指定した場合，rompsec セクション内のすべてのセクションをコピーします。セクション番号 number は，1 から始まる正数です。
- デフォルトは，セクションが入力ファイル中に出現した順番に割り当てられます。ROM 化プロセッサのオプション“-p オプション”，“-t オプション”で rompsec セクションに配置するセクションを指定した場合は，指定した順番に割り当てられます。
- CubeSuite+ において，プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［セクション・リスト］カテゴリの［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択すると，#define による“番号”と“ラベル”の対応付けされた C ソース・ヘッダ・ファイルが生成され，ラベル名によって，number を指定することができます。
- 具体的な使用例については，「B. 4. 5　コピー関数の使用例」を参照してください。

[戻り値]

0	正常終了（正しくコピーされた場合）
-1	異常終了（正しくコピーされなかった場合）

[注意事項]

- label の示すアドレスが rompsec セクションの先頭でなかった場合はコピーを行いません。
- _rcopy2 は ROM 化プロセッサで生成された情報にしたがってコピーを行います。
  _rcopy2 実行時にコピー先のアドレスにオフセットを加えるような処理はできません。
- コピーを行うとオーバライトが生じる場合，コピーを行いません。

- _rcopy2 の第一引数 label には，絶対値を持つグローバルなラベル，または絶対アドレスを指定してください。これら以外のものを指定した場合，その結果は保証されません。

_rcopy4

[概要]

- 初期値データ／RAM テキスト注のコピー（4 バイト）

注 RAM に配置する初期値ありデータ・セクション，および内蔵 RAM 用テキスト・セクションです。

[形式]

```
int             _rcopy4(&label, number)
unsigned long   label;
long            number;
```

[説明]

- _rcopy4(&label, number) は，label の示すアドレス以降に存在する rompsec セクション内の情報を元に，コピーしたいセクション番号 number の初期値データ，または RAM に配置するテキストを，RAM 領域に 4 バイトずつコピーします。number に -1 を指定した場合，rompsec セクション内のすべてのセクションをコピーします。セクション番号 number は，1 から始まる正数です。
- デフォルトは，セクションが入力ファイル中に出現した順番に割り当てられます。ROM 化プロセッサのオプション“-p オプション”，“-t オプション”で rompsec セクションに配置するセクションを指定した場合は，指定した順番に割り当てられます。
- CubeSuite+ において，プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［セクション・リスト］カテゴリの［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択すると，#define による“番号”と“ラベル”の対応付けされた C ソース・ヘッダ・ファイルが生成され，ラベル名によって，number を指定することができます。
- 具体的な使用例については，「B. 4. 5 コピー関数の使用例」を参照してください。

[戻り値]

0	正常終了（正しくコピーされた場合）
-1	異常終了（正しくコピーされなかった場合）

[注意事項]

- label の示すアドレスが rompsec セクションの先頭でなかった場合はコピーを行いません。
- _rcopy4 は ROM 化プロセッサで生成された情報にしたがってコピーを行います。
  _rcopy4 実行時にコピー先のアドレスにオフセットを加えるような処理はできません。
- コピーを行うとオーバライトが生じる場合，コピーを行いません。

- _rcopy4 の第一引数 label には，絶対値を持つグローバルなラベル，または絶対アドレスを指定してください。これら以外のものを指定した場合，その結果は保証されません。

B. 4. 5 コピー関数の使用例

(1) すべてのセクションを 1 バイト転送する場合

```
extern unsigned long _S_romp;

main()
{
        int     ret;3

        ret = _rcopy(&_S_romp, -1);
        /* -Xr 指定により，絶対値を持つグローバルなラベルを指定 */
}
```

このように，ca850 に ROM 化用オプションを指定することにより，ラベルは絶対アドレス参照となります。したがって，_rcopy() をアセンブラ・ソース・プログラムで呼び出す場合は，次のようにしてください。

```
.extern __S_romp, 4     -- 外部ラベルとして宣言

-- __S_romp の絶対アドレスを第一引数，-1 を第二引数として _rcopy を呼び出し
mov     #__S_romp, r6
mov     -1, r7
jarl    __rcopy, lp
```

(2) 1 ～ 6 のセクションを 4 バイト転送，7 ～ 11 のセクションを 1 バイト転送する場合

```
extern unsigned long _S_romp;
main()
{
        int     ret, num;

        for(num = 1; num<=6; num++) {
                ret = _rcopy4(&_S_romp, num);
                if(ret == -1) {
                        /* エラー処理 */
                }
        }

        for(num = 7; num <= 11; num++) {
                ret = _rcopy1(&_S_romp, num);
                if(ret == -1) {
                        /* エラー処理 */
                }
        }
}
```

（3）誤った指定の例1

```
extern unsigned long _S_romp;
char *cp;

func()
{
        int     ret;

        cp = &_S_romp;          /* 変数に入れたため，第一引数が gp 相対値となる */
        ret = _rcopy(cp, -1);
}
```

（4）誤った指定の例2

```
extern unsigned long _S_romp;
int     i;

func()
{
        int     ret;

        i = 0x100;              /* 変数に入れたため，第一引数が gp 相対値となる */
        ret = _rcopy(i, -1);
}
```

- number に指定するセクション番号は 1 から始まる正の整数です。
  セクション名とセクション番号の関連は，メモリ・マップから参照できますが，CubeSuite+ を使用した場合，プロパティ パネルの［ROM 化プロセス・オプション］タブをオープンし，［セクション・リスト］カテゴリの［ROM 化用セクション・ファイルを出力する］プロパティで［はい］を選択することにより，セクション番号とラベルの対応付けがされた C 言語ヘッダ・ファイルを作成できます。つまり，number にラベルを使用することができます。
- number にセクション番号，または -1 以外の指定を行った場合，コピーを行いません。
- 複数の RAM が存在し，複数のコピー・ルーチンを使いわける場合，number に -1 の指定を行うと，すべてのセクション整列等の問題により，正常にコピーされません。
  number には -1 の指定をせず，セクション番号を指定してください。
- number に -1 を指定した場合，セクション番号順にコピーを行います。
  途中で上記の各問題によりコピーが行われないセクションが発生した場合，戻り値として -1 を返却します。問題となったセクションよりも後のセクションは，コピーされません。

B. 4. 6　操作方法

ここでは，ROM 化プロセッサの操作方法について説明します。

（1）コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>romp850[オプション]... ファイル名
        [ ]：    [ ] 内は省略できます。
        ...：    直前の [ ] 内のパターンの繰り返しができます。
```

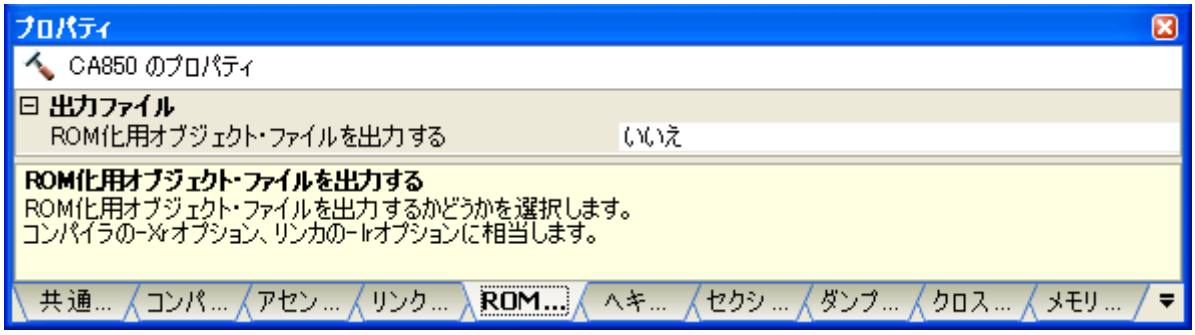
（2）CubeSuite+ でのオプション設定

CubeSuite+ から ROM 化プロセス・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネルにおいて，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［ROM 化プロセス・オプション］タブを選択します。

タブ上で各プロパティを設定することにより，対応する ROM 化プロセス・オプションを設定することができます。

図 B―35　プロパティ パネル：［ROM 化プロセス・オプション］タブ



B. 4. 7　オプション

ここでは，ROM 化プロセス・オプションについて説明します。

ROM 化プロセス・オプションの分類と説明を示します。

表 B—16　ROM 化プロセス・オプション

分類	オプション	説明
ファイル	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。
	-o	生成するオブジェクト・ファイル名を指定します。
ROM 化プロセッサ	-Ximem_overflow=warning	内蔵 ROM ／ RAM をオーバーフローした際のチェックの制御を行います。
	-b	指定ラベル値を，生成される rompsec セクションの先頭アドレスとします。
	-d	rompsec セクションのみを持つオブジェクト・ファイルを生成します。
	-i	入力ファイル，および出力ファイルのアドレスの重複チェックを行いません。
	-m	生成するオブジェクト・ファイルのメモリ・マップを出力します。
	-p	data 属性，または sdata 属性を持つセクションの内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。
	-rom_less	rompsec セクションに対して，内蔵 ROM 周辺の配置エラー・チェックを行いません。
	-t	text 属性，または const 属性を持つセクションの内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。
その他	-F	デバイス・ファイルを，指定フォルダから探します。
	-V	バージョン情報を標準エラー出力に出力します。
	-help	オプションの説明を標準エラー出力に出力します。
	@	指定ファイルをコマンド・ファイルとして扱います。

ファイル

ファイルに関するオプションには，次のものがあります。

- +err_file
- -err_file
- -o

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>romp850 +err_file=err a.out
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>romp850 -err_file=err a.out
```

-o

[記述形式]

```
-o ofile
```

- 省略時解釈

  生成されるオブジェクト・ファイル名を romp.out とします。

[機能説明]

- 生成されるオブジェクト・ファイル名を *ofile* とします。

[使用例]

- 生成するオブジェクト・ファイル名を test.out とします。

```
C:¥>romp850 -o test.out a.out
```

ROM 化プロセッサ

ROM 化プロセッサのオプションには，次のものがあります。

- -Ximem_overflow=warning
- -b
- -d
- -i
- -m
- -p
- -rom_less
- -t

-Ximem_overflow=warning

[記述形式]

```
-Ximem_overflow=warning
```

- 省略時解釈

  オーバーフロー時にはエラー・メッセージを出力し，処理を中止します。

[機能説明]

- 内蔵 ROM ／ RAM をオーバーフローした際のチェックの制御を行います。
- オーバーフロー時には警告メッセージを出力し，処理を継続します。

[使用例]

- 内蔵 ROM ／ RAM をオーバーフローした際のチェックの制御を行います。

```
C:¥>romp850 -Ximem_overflow=warning a.out
```

-b

[記述形式]

```
-b label
```

- 省略時解釈

  ラベル __S_romp を，生成される rompsec セクションの先頭アドレスとします。

[機能説明]

- ラベル *label* 値を，生成される rompsec セクションの先頭アドレスとします。
- 指定したラベルがオブジェクト・ファイル中に存在しない場合，またはこのオプションを複数回指定した場合，メッセージを出力し処理を中止します。

[使用例]

- ラベル __rompack 値を，生成される rompsec セクションの先頭アドレスとします。

```
C:￥>romp850 -b __rompack a.out
```

-d

[記述形式]

```
-d
```

- 省略時解釈

  生成するファイル中に text 属性を持つセクションも入れます。

[機能説明]

- 生成するファイル中に text 属性を持つセクションを入れずに，rompsec セクションのみを持つオブジェクト・ファイルを生成します。

[使用例]

- 生成するファイル中に text 属性を持つセクションを入れずに，rompsec セクションのみを持つオブジェクト・ファイルを生成します。

```
C:￥>romp850 -d a.out
```

-i

[記述形式]

```
-i
```

- 省略時解釈

  入力ファイル，および出力ファイルのアドレスの重複チェックを行い，不正箇所があった場合にメッセージを出力し，リンクの処理を中止します。

[機能説明]

- 入力ファイル，および出力ファイルのアドレスの重複チェックを行いません。

[使用例]

- 入力ファイル，および出力ファイルのアドレスの重複チェックを行いません。

```
C:￥>romp850 -i a.out
```

-m

[記述形式]

```
-m[=mapfile]
```

- 省略時解釈

  リンク・マップを出力しません。

[機能説明]

- 生成するオブジェクト・ファイルのメモリ・マップを *mapfile* に出力します。
- *mapfile* を省略した場合，標準出力に出力します。

[使用例]

- 生成するオブジェクト・ファイルのメモリ・マップを map に出力します。

```
C:¥>romp850 -m=map a.out
```

-p

[記述形式]

```
-p section
```

- 省略時解釈

  data 属性，または sdata 属性を持つすべてのセクション，および内蔵命令 RAM に配置されるセクションの内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。

[機能説明]

- セクション名 *section* の内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。
- このオプションは，data 属性，または sdata 属性を持つセクションに関するオプションです。
- 複数回指定した場合，指定した順に rompsec セクションに入れます。
- 指定したセクションがオブジェクト・ファイル中に存在しない場合，メッセージを出力し，処理を中止します。
- セクション名に空白は使用できません。

[使用例]

- セクション名 .sdata の内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。

```
C:￥>romp850 -p .sdata a.out
```

-rom_less

[記述形式]

```
-rom_less
```

- 省略時解釈

  rompsec セクションに対して，内蔵 ROM 周辺の配置エラー・チェックを行います。

[機能説明]

- rompsec セクションに対して，内蔵 ROM 周辺の配置エラー・チェックを行いません。
- ROM レス・モード使用時に指定することを推奨します。
- シングルチップ・モード選択時の内蔵 ROM オーバのチェックには対応していません。
- このオプションを指定して内蔵 ROM オーバーフローのチェックを無効とし，ダンプ・ツールで確認してください。

[使用例]

- rompsec セクションに対して，内蔵 ROM 周辺の配置エラー・チェックを行いません。

```
C:¥>romp850 -rom_less a.out
```

-t

[記述形式]

```
-t section
```

- 省略時解釈

  内蔵命令 RAM に配置される各セクションの内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。

[機能説明]

- セクション名 *section* の内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。
- このオプションは text 属性，または const 属性を持つセクションに関するオプションです。
- 複数回指定した場合，指定した順に rompsec セクションに入れます。
- 指定したセクションがオブジェクト・ファイル中に存在しない場合，メッセージを出力し，処理を中止します。
- このオプションで指定できるセクションは，text 属性，または const 属性を持つセクションで，これ以外の属性のセクションを指定した場合は，メッセージを出力し，処理を中止します。
- セクション名に空白は使用できません。
- 内蔵命令 RAM 搭載のデバイス・ファイルを指定してリンクされた入力ファイルに対して，このオプションで特定のセクションを指定した場合，指定されなかった内蔵命令 RAM に配置されたセクションは，rompsec セクションに入らないだけでなく，出力ファイル中からも削除されます。

[使用例]

- セクション名 .text の内容とそのアドレス，およびサイズの情報を rompsec セクションに入れます。

```
C:￥>romp850 -t .text a.out
```

その他

その他のオプションには，次のものがあります。

- -F
- -V
- -help
- @

-F

［記述形式］

```
-F devpath
```

- 省略時解釈

  デバイス・ファイルを，標準フォルダから探します。

［機能説明］

- デバイス・ファイルをフォルダ *devpath* から探します。

［使用例］

- デバイス・ファイルを，フォルダ D:￥dev から探します。

```
C:￥>romp850 -F D:￥dev a.out
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- ROM 化プロセッサのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- ROM 化プロセッサのバージョン情報を標準エラー出力に出力します。

```
C:¥>romp850 -V
```

-help

[記述形式]

```
-help
```

- 省略時解釈

  なし

[機能説明]

- ROM 化プロセッサのオプションの説明を標準エラー出力に出力します。

[使用例]

- オプションの説明を標準エラー出力に出力します。

```
C:¥>romp850 -help
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

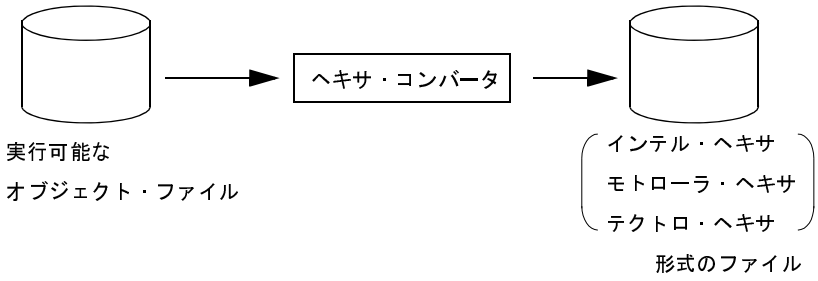
```
C:￥>romp850 @command
```

B. 5　ヘキサ・コンバータ

ヘキサ・コンバータ（hx850）は，ROM 化プロセッサが出力する実行可能なオブジェクト・ファイルを入力し，ヘキサ・フォーマットに変換します。

なお，アプリケーション中に初期値ありデータがないなど，ROM 化プロセッサを使用する必要のないアプリケーションの場合は，リンカが出力する実行可能なオブジェクト・ファイルを入力します。

図 B—36　ヘキサ・コンバータにおける動作の流れ



B. 5. 1　入出力ファイル

ヘキサ・コンバータでは，次のファイルを入力ファイルとして扱うことができます。

file1.out	ld850，または romp850 によって出力された実行可能オブジェクト

ヘキサ・フォーマットの出力として，次に示す形式を指定できます。

（1）インテル・ヘキサ・フォーマット

- インテル拡張ヘキサ・フォーマット

（2）テクトロ・ヘキサ・フォーマット

- 拡張テクトロニクス・ヘキサ・フォーマット

（3）モトローラ・ヘキサ・フォーマット

- S タイプ・フォーマット（スタンダード・アドレス）
- S タイプ・フォーマット（32 ビット・アドレス）

注　ヘキサ・フォーマットの各行のアドレスは，昇順で出力されます。

出力リストについての詳細は，「3. 3　ヘキサ・コンバータ」を参照してください。

B. 5. 2　操作方法

ここでは，ヘキサ・コンバータの操作方法について説明します。

(1) コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>hx850 [オプション]... ファイル名
        [ ]:    [ ]内は省略できます。
        ...:    直前の[ ]内のパターンの繰り返しができます。
```

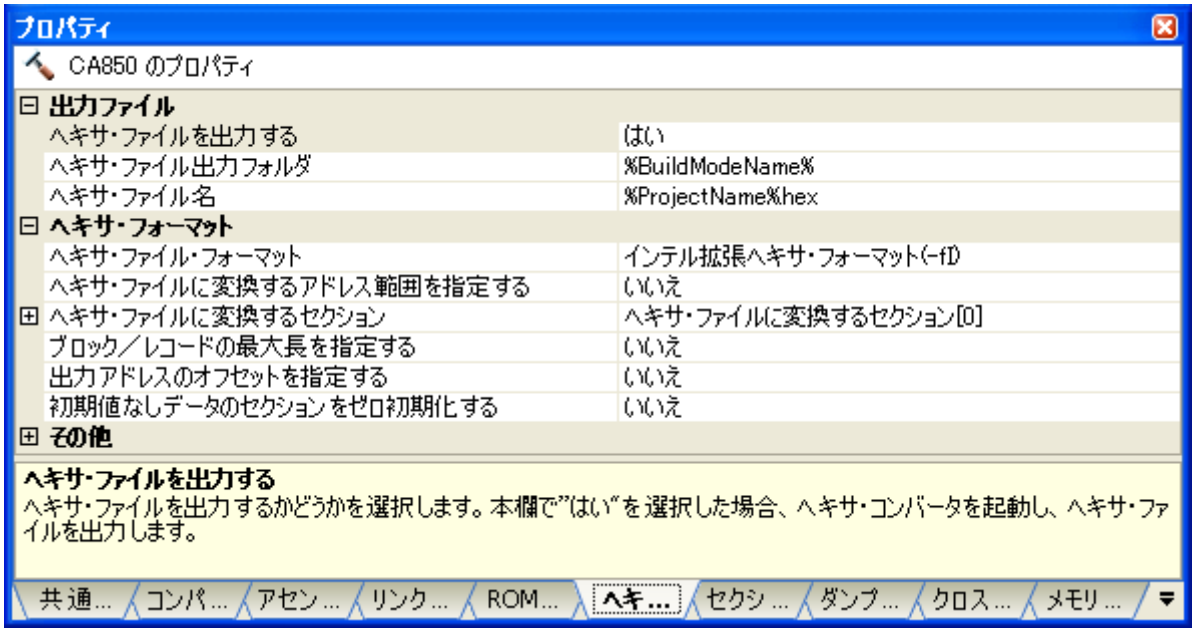
(2) CubeSuite+ でのオプション設定

CubeSuite+ からヘキサ・コンバート・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネルにおいて，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［ヘキサ・コンバート・オプション］タブを選択します。

タブ上で各プロパティを設定することにより，対応するヘキサ・コンバート・オプションを設定することができます。

図 B―37　プロパティ パネル：［ヘキサ・コンバート・オプション］タブ



B. 5. 3　オプション

ここでは，ヘキサ・コンバート・オプションについて説明します。

ヘキサ・コンバート・オプションの分類と説明を示します。

表 B―17　ヘキサ・コンバート・オプション

分類	オプション	説明
ファイル	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。
	-o	指定ファイルにヘキサ変換した結果を出力します。
フォーマット	-b	指定された値をブロック長の最大値とします。
	-d	出力するアドレスのオフセットを指定します。
	-f	ヘキサ・フォーマットを指定します。
	-I	指定されるセクションのコードを変換し，出力します。
	-S	シンボル・テーブルを変換し，出力します。
	-U	指定アドレスから指定されたサイズ領域のすべてのコードをヘキサ変換し，出力します。
	-x	シンボル・テーブルを変換して出力する際，ローカル・シンボルも対象とします。
	-rom_less	デバイス・ファイルに定義された内蔵 ROM 領域の情報を使用しないようにします。
	-z	セクション・タイプ NOBITS とセクション属性 A を持つセクションに対し，セクションのサイズ分だけ null 文字（¥0）を生成します。
その他	-F	デバイス・ファイルを，指定フォルダから探します。
	-V	バージョン情報を標準エラー出力に出力します。
	@	指定ファイルをコマンド・ファイルとして扱います。

ファイル

ファイルに関するオプションには，次のものがあります。

- +err_file
- -err_file
- -o

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>hx850 +err_file=err -o a.hex a.out
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>hx850 -err_file=err -o a.hex a.out
```

-o

[記述形式]

```
-o ofile
```

- 省略時解釈

  ヘキサ変換した結果を標準出力に出力します。

[機能説明]

- *ofile* という名前のファイルにヘキサ変換した結果を出力します。

[使用例]

- ヘキサ変換した結果をファイル test に出力します。

```
C:¥>hx850 -o test a.out
```

フォーマット

フォーマットに関するオプションには，次のものがあります。

- -b
- -d
- -f
- -I
- -S
- -U
- -x
- -rom_less
- -z

-b

[記述形式]

```
-bnum
```

- 省略時解釈

  ブロック長に各ヘキサ・フォーマットごとに定められたデフォルトの値を用います。

[機能説明]

- *num* で指定された値をブロック長（インテル拡張ヘキサ・フォーマット，およびモトローラ S タイプ・ヘキサ・フォーマットの場合，1 データ・レコードで示されるコードのバイト数）の最大値とします。
- *num* は 10 進数，あるいは，0x，または 0X で始まる 16 進数で指定します。

表 B―18　ヘキサ・フォーマットのブロック／レコード

ヘキサ・フォーマット	指定できる範囲	デフォルト
インテル拡張	1 ～ 255（0x01 ～ 0xff）	31（0x1f）
モトローラ S タイプ	1 ～ 251（0x01 ～ 0xfb）	80（0x50）
モトローラ S タイプ（32 ビット・アドレス）	1 ～ 250（0x01 ～ 0xfa）	80（0x50）
拡張テクトロニクス	16 ～ 255（0x10 ～ 0xff）	255（0xff）

［使用例］

- インテル拡張ヘキサ・フォーマットの 1 データ・レコードで示されるコードのバイト数の最大値を 255 とします。

```
C:￥>hx850 -b255 -o a.hex a.out
```

-d

[記述形式]

```
-dnum
```

- 省略時解釈

  出力するアドレスをオフセットとして計算しません。

[機能説明]

- 出力するアドレスを *num* からのオフセットとします。
- *num* は 10 進数，あるいは，0x，または 0X で始まる 16 進数で指定します。
- 指定可能な値は，0 ～ 0xfffffffe の範囲です。
- 出力するアドレスは，指定した値からのオフセット値となります。
- デフォルトは 0 です。

[使用例]

- 出力するアドレスを 0x10000 からのオフセットとします。

```
C:¥>hx850 -d0x10000 -o a.hex a.out
```

-f

[記述形式]

```
-fc
```

- 省略時解釈

  インテル拡張ヘキサ・フォーマットになります。

[機能説明]

- 文字 *c* で指定されるヘキサ・フォーマットを用います。
- 文字 *c* の意味は次のようになっています。

I	インテル拡張
S	モトローラＳタイプ
s	モトローラＳタイプ（32 ビット・アドレス）
T	拡張テクトロニクス

- -fT オプションと -U オプションを同時に指定した場合，-U オプションの指定は無視されます。

[使用例]

- モトローラＳタイプ（32 ビット・アドレス）フォーマットを用います。

```
C:¥>hx850 -fs -o a.hex a.out
```

-I

[記述形式]

```
-Iname
```

- 省略時解釈

  NOBITS 以外のセクション・タイプとセクション属性 A を持つすべてのセクションを変換します。

[機能説明]

- セクション名 *name* で指定されるセクションのコードを変換し，出力します。つまり，ヘキサ・コンバータは，セグメント単位ではなくセクション単位に変換を行います。
- 初期値の指定されていないデータに対するセクション（セクション・タイプ NOBITS とセクション属性 A を持つセクション）が指定された場合，セクションのサイズ分だけ null 文字（¥0）を生成します。
- ヘキサ・コンバータは，セグメント単位ではなくセクション単位に変換します。
- セクション名に空白は使用できません。
- このオプションと -U オプションを同時に指定した場合，このオプションの指定は無視されます。

[使用例]

- .text セクションのコードを変換し，出力します。

```
C:¥>hx850 -I.text -o a.hex a.out
```

-S

[記述形式]

```
-S
```

- 省略時解釈

  シンボル・テーブルを変換し，出力しません。

[機能説明]

- シンボル・テーブルを変換し，出力します。
- 拡張テクトロニクス・ヘキサ・フォーマットが指定された（-fT が指定された）場合にのみ有効です。
- このオプションと -U オプションを同時に指定した場合，このオプションの指定は無視されます。

[使用例]

- シンボル・テーブルを変換し，出力します。

```
C:￥>hx850 -fT -S -o a.hex a.out
```

-U

[記述形式]

```
-U
-Unum
-Unum,start,size
-Ustart,size
```

- 省略時解釈

  NOBITS 以外のセクション・タイプとセクション属性 A を持つすべてのセクションのみ変換します。

[機能説明]

- アドレス *start* からサイズ *size* で指定された領域のすべてのコードをヘキサ変換し，出力します。
- *start*，*size* の指定を省略した場合，デバイス・ファイルで定義された内蔵 ROM 領域のすべてのコードをヘキサ変換し，出力します。
- 指定された領域のうち未使用領域は *num* で満たします。*num* は，1 バイト，または 2 バイト指定ができます。*num* が 2 桁，または 4 桁に満たない場合，満たない分の 0 が頭に指定されたものとみなします。
- *num* として指定した 2 バイトの下位 1 バイトは上位アドレスに，上位 1 バイトは下位アドレスに配置されます。
- *num* を省略した場合，0xff で満たします。
- このオプションは，拡張テクトロニクス・ヘキサ・フォーマット指定時に使用できません。
- このオプションを指定した場合，-I，-S，-x，-Z オプションの指定は無視されます。

[使用例]

- デバイス・ファイルで定義された内蔵 ROM 領域のすべてのコードをヘキサ変換し，出力します。未使用領域は，0xff で満たします。

```
C:¥>hx850 -U -o a.hex a.out
```

-x

[記述形式]

```
-x
```

- 省略時解釈

  シンボル・テーブルを変換して出力する際，グローバル・シンボルのみを対象とします。

[機能説明]

- シンボル・テーブルを変換して出力する際，ローカル・シンボルも対象とします。
- -S オプションとともに指定された場合にのみ有効です。
- このオプションと -U オプションを同時に指定した場合，このオプションの指定は無視されます。

[使用例]

- シンボル・テーブルを変換して出力する際，ローカル・シンボルも対象とします。

```
C:￥>hx850 -fT -S -x -o a.hex a.out
```

-rom_less

[記述形式]

```
-rom_less
```

- 省略時解釈

  デバイス・ファイルに定義された内蔵 ROM 領域の情報を使用します。

[機能説明]

- -U オプション指定時に，デバイス・ファイルに定義された内蔵 ROM 領域の情報を使用しないようにします。

  また，ヘキサ変換する領域が内蔵 ROM 領域をはみ出した場合，警告メッセージの出力を行いません。
- このオプションと -U オプションを同時に指定する場合は，-U オプションの *start*，*size* の指定が必要になります。
- このオプションと -U オプションの *start*，*size* を省略した場合，デバイス・ファイルに定義された内蔵 ROM 領域が変換対象になります。

  また，ヘキサ変換する領域が内蔵 ROM 領域をはみだした場合は，警告メッセージを出力します。

[使用例]

- デバイス・ファイルに定義された内蔵 ROM 領域の情報を使用せずに，-U オプションを指定します。

```
C:￥>hx850 -rom_less -U0xff,0x0,1000 -o a.hex a.out
```

-z

[記述形式]

```
-z
```

- 省略時解釈

  NOBITS 以外のセクション・タイプとセクション属性 A を持つすべてのセクションのみ変換します。

[機能説明]

- セクション・タイプ NOBITS とセクション属性 A を持つセクション（初期値の指定されていないデータに対するセクション，たとえば .bss セクション，および .sbss セクション）に対し，セクションのサイズ分だけ null 文字（¥0）を生成します。
- このオプションと -U オプションを同時に指定した場合，このオプションの指定は無視されます。

[使用例]

- セクション・タイプ NOBITS とセクション属性 A を持つセクションに対し，セクションのサイズ分だけ null 文字（¥0）を生成します。

```
C:¥>hx850 -z -o a.hex a.out
```

その他

その他のオプションには，次のものがあります。

- -F
- -V
- @

-F

[記述形式]

```
-F devpath
```

- 省略時解釈

  デバイス・ファイルを，標準フォルダから探します。

[機能説明]

- デバイス・ファイルをフォルダ *devpath* から探します。

[使用例]

- デバイス・ファイルを，フォルダ D:￥dev から探します。

```
C:￥>hx850 -F D:￥dev -o a.hex a.out
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- ヘキサ・コンバータのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- ヘキサ・コンバータのバージョン情報を標準エラー出力に出力します。

```
C:¥>hx850 -V
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

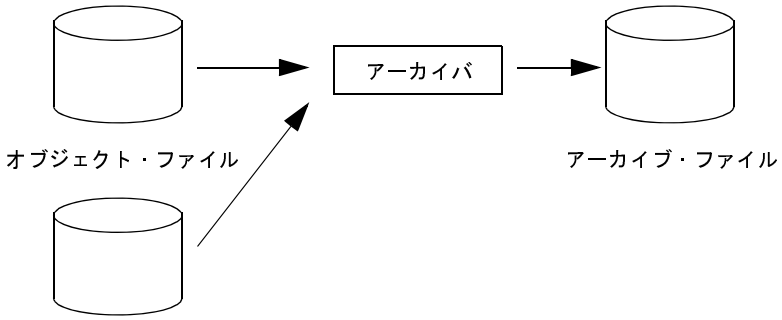
```
C:￥>hx850 @command
```

B. 6　アーカイバ

アーカイバとは，指定されたリロケータブルなオブジェクト・ファイルを結合し，1 つのアーカイブ・ファイルを生成するユーティリティです。つまり，ユーザが複数のオブジェクトをまとめ，“ライブラリ”として作成する場合に使用します。

CA850 に入っている“ar850”がアーカイバです。

図 B―38　アーカイバにおける動作の流れ



アーカイバによって生成されるアーカイブ・ファイルは，リンク時の入力ファイルとして指定できます。アーカイブ・ファイルが指定された場合，ld850 は指定されたアーカイブ・ファイル内から必要とされるオブジェクトを検索し，見つかったオブジェクトのみをリンクします。

B. 6. 1　操作方法

ここでは，アーカイバの操作方法について説明します。

(1) コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>ar850 [エラー出力指定オプション] キー [オプション][メンバ名注] アーカイブ・ファイル名 [メンバ名，またはファイル名]...
```

[ ]：　[ ] 内は省略できます。

...：　直前の [ ] 内のパターンの繰り返しができます。

注　ファイルは，アーカイブ・ファイル内に連結されるとメンバと呼ばれます。メンバは，連結される前のファイルのときと同じ名前を持ちます。

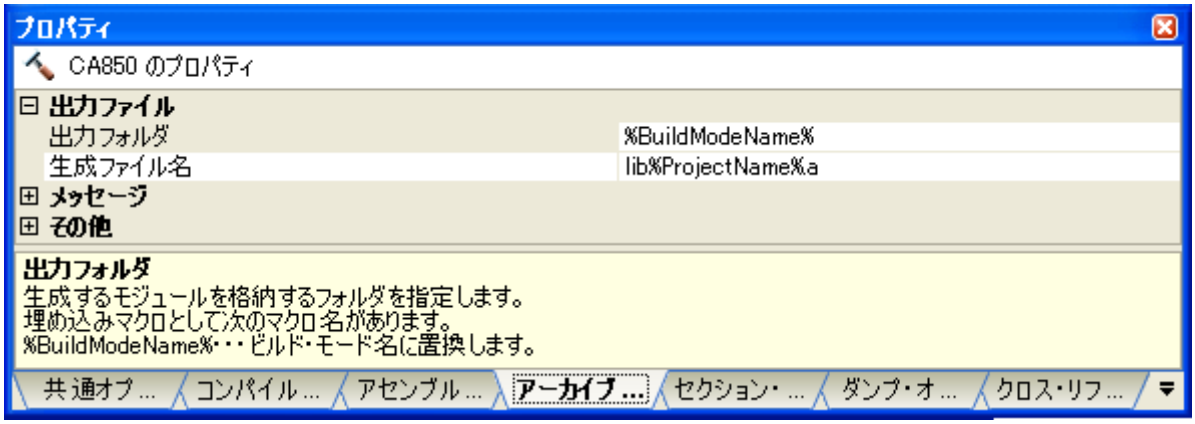
(2) CubeSuite+ でのオプション設定

CubeSuite+ からアーカイブ・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネル上において，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［アーカイブ・オプション］タブを選択します。

タブ上で必要なプロパティを設定することにより，各アーカイブ・オプションを設定することができます。

図 B—39　プロパティ パネル：［アーカイブ・オプション］タブ



コマンド・ライン上からアーカイバを起動する場合，オブジェクト・ファイル群をまとめて，アーカイブ・ファイルを作成します。また，アーカイブ・ファイル内のオブジェクトの操作など，細かい作業を行うことができます。

一方，CubeSuite+ を使用してアーカイブ・ファイルを生成する場合，ソース・ファイルをコンパイル，アセンブルし，作られたオブジェクトをアーカイブ・ファイルにまとめるという作業になります。CubeSuite+ 上からは完成されたアーカイブ・ファイルに対して操作することはできません。そのため，コマンド・ラインと CubeSuite+ を使い分けていく必要があります。

B. 6. 2　キー／オプション

ここでは，アーカイバのキーとオプションの種類と機能について説明します。

キーとは，起動時に必ずどれかが指定されていなければならないもので，オプションとは省略することができるものです。

アーカイバのキー／オプションの分類と説明を示します。

表 B―19　アーカイバ・キー

分類	キー	説明
キー	V	アーカイバのバージョン情報を標準エラー出力に出力します。
	d	指定されたメンバを指定されたアーカイブ・ファイルから削除します。
	m	指定されたメンバを指定されたアーカイブ・ファイルの最後に移動します。
	ma	指定されたメンバを指定されたアーカイブ・ファイル内のメンバの直後に移動します。
	mb	指定されたメンバを指定されたアーカイブ・ファイル内のメンバの直前に移動します。
	q	指定されたファイルを指定されたアーカイブ・ファイルの最後に追加します。
	r	指定されたファイルを指定されたアーカイブ・ファイル内の同名のメンバと入れ替えます。
	ra	指定されたファイルを指定されたアーカイブ・ファイル内の同名のメンバと入れ替え，指定メンバの直後に移動します。
	ru	指定されたファイルが指定されたアーカイブ・ファイル内の同名のメンバより最近に更新されている場合，入れ替えを行います。
	t	指定されたアーカイブ・ファイル内に存在しているメンバのメンバ名を出力します。
	x	指定されたアーカイブ・ファイル内のメンバを取り出し，同名のファイルを生成します。

表 B―20　アーカイブ・オプション

分類	オプション	説明
アーカイバ	c	メッセージを出力しません。
	v	アーカイバの実行状況を出力します。
	@	指定ファイルをコマンド・ファイルとして扱います。
出力ファイル	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。

キー

アーカイバのキーには，次のものがあります。

- V
- d
- m
- ma
- mb
- q
- r
- ra
- ru
- t
- x

V

[記述形式]

```
V
```

- 省略時解釈

  なし

[機能説明]

- アーカイバのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- アーカイバのバージョン情報を標準エラー出力に出力します。

```
C:¥>ar850 V
```

d

[記述形式]

```
d
```

- 省略時解釈

  なし

[機能説明]

- 指定されたメンバを指定されたアーカイブ・ファイルから削除します。

[使用例]

- アーカイブ・ファイル libarc.a からメンバ sub.o を削除します。

```
C:¥>ar850 d libarc.a sub.o
```

m

[記述形式]

```
m
```

- 省略時解釈

  なし

[機能説明]

- 指定されたメンバを指定されたアーカイブ・ファイルの最後に移動します。

[使用例]

- アーカイブ・ファイル libarc.a 中のメンバ sub.o をアーカイブ・ファイルの最後に移動します。

```
C:¥>ar850 m libarc.a sub.o
```

ma

[記述形式]

```
ma member
```

- 省略時解釈

  なし

[機能説明]

- 指定されたメンバを指定されたアーカイブ・ファイル内のメンバ *member* の直後に移動します。
- *member* を省略した場合，処理を中止します。

[使用例]

- アーカイブ・ファイル libarc.a 中のメンバ sub.o をアーカイブ・ファイルのメンバ main.o の直後に移動します。

```
C:￥>ar850 ma main.o libarc.a sub.o
```

mb

[記述形式]

```
mb member
```

- 省略時解釈

  なし

[機能説明]

- 指定されたメンバを指定されたアーカイブ・ファイル内のメンバ *member* の直前に移動します。
- *member* を省略した場合，処理を中止します。

[使用例]

- アーカイブ・ファイル libarc.a 中のメンバ sub.o をアーカイブ・ファイルのメンバ main.o の直前に移動します。

```
C:￥>ar850 mb main.o libarc.a sub.o
```

q

[記述形式]

```
q
```

- 省略時解釈

  なし

[機能説明]

- 指定されたファイルを指定されたアーカイブ・ファイルの最後に追加します。

  指定されたファイルと同名のメンバが存在しているかどうかのチェックは行いません。
- 指定されたアーカイブ・ファイルが存在しない場合，指定されたファイルを含んだアーカイブ・ファイルを新規に生成します。

  指定されたファイルと同名のメンバのチェックは行いません。
- 同名のメンバが存在した場合，アーカイブ・ファイルに複数の同名メンバが含まれ，リンク時には最も古いメンバが選択されます。
- 新規作成を行う場合は，古いアーカイブ・ファイルを必ず削除してください。
- 同名のメンバを入れ替える場合には，r キーを使用してください。

[使用例]

- アーカイブ・ファイル libarc.a 中の最後にメンバ sub.o を追加します。

```
C:¥>ar850 q libarc.a sub.o
```

r

[記述形式]

```
r
```

- 省略時解釈

  なし

[機能説明]

- 指定されたファイルを指定されたアーカイブ・ファイル内の同名のメンバと入れ替えます。
- 指定されたアーカイブ・ファイル内に同名のメンバが存在しない場合，指定されたファイルを指定されたアーカイブ・ファイルの最後に追加します。
- 指定されたアーカイブ・ファイルが存在しない場合，指定されたファイルを含んだアーカイブ・ファイルを新規に生成します。

[使用例]

- アーカイブ・ファイル libarc.a 中のメンバ sub.o を入れ替えます。

```
C:￥>ar850 r libarc.a sub.o
```

ra

[記述形式]

```
ra member
```

- 省略時解釈

  なし

[機能説明]

- 指定されたファイルを指定されたアーカイブ・ファイル内の同名のメンバと入れ替え，メンバ *member* の直後に移動します。
- 指定されたアーカイブ・ファイル内に同名のメンバが存在しない場合，指定されたファイルを指定されたアーカイブ・ファイルの最後に追加します。
- *member* を省略した場合，処理を中止します。

[使用例]

- アーカイブ・ファイル libarc.a 中のメンバ sub.o を入れ替え，アーカイブ・ファイルのメンバ main.o の直後に移動します。

```
C:￥>ar850 ra main.o libarc.a sub.o
```

ru

[記述形式]

```
ru
```

- 省略時解釈

  なし

[機能説明]

- 指定されたファイルが指定されたアーカイブ・ファイル内の同名のメンバより最近に更新されている場合，入れ替えを行います。
- 指定されたアーカイブ・ファイル内に同名のメンバが存在しない場合，指定されたファイルを指定されたアーカイブ・ファイルの最後に追加します。
- 指定されたアーカイブ・ファイルが存在しない場合，指定されたファイルを含んだアーカイブ・ファイルを新規に生成します。

[使用例]

- sub.o がアーカイブ・ファイル libarc.a 中の sub.o より最近に更新されている場合に入れ替えます。

```
C:￥>ar850 ru libarc.a sub.o
```

t

[記述形式]

```
t
```

- 省略時解釈

  なし

[機能説明]

- メンバ名が指定された場合，指定されたアーカイブ・ファイル内に存在しているメンバのメンバ名のみを出力します。
- メンバ名が指定されなかった場合，指定されたアーカイブ・ファイル内に存在しているすべてのメンバのメンバ名を標準出力に出力します。

[使用例]

- アーカイブ・ファイル libarc.a 中に存在しているすべてメンバのメンバ名のみを出力します。

```
C:¥>ar850 t libarc.a
```

x

[記述形式]

```
x
```

- 省略時解釈

  なし

[機能説明]

- メンバ名が指定された場合，指定されたメンバが指定されたアーカイブ・ファイル内に存在していればそのメンバを取り出し，同名のファイルを生成します。
- メンバ名が指定されなかった場合，指定されたアーカイブ・ファイル内に存在しているすべてのメンバを取り出し，同名のファイルを生成します。アーカイブ・ファイルの内容は変更されません。

[使用例]

- アーカイブ・ファイル libarc.a 中に存在しているメンバ sub.o を取り出しファイルを生成します。

```
C:¥>ar850 x libarc.a sub.o
```

アーカイバ

アーカイバに関するオプションには，次のものがあります。

- c
- v
- @

c

[記述形式]

```
c
```

- 省略時解釈

  なし

[機能説明]

- メッセージを出力しません。

[使用例]

- メッセージを出力しません。

```
C:￥>ar850 tc libarc.a
```

v

[記述形式]

```
v
```

- 省略時解釈

  なし

[機能説明]

- アーカイバの実行状況を“[a|d|q|m|r|x] - *file*”の形式で出力します。

a - *file*	追加
d - *file*	削除
q - *file*	新規作成，または追加
m - *file*	移動
r - *file*	置換
x - *file*	抽出

[使用例]

- 実行状況を表示します。

```
C:¥>ar850 dv libarc.a sub.o
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>ar850 @command
```

出力ファイル

出力ファイルに関するオプションには，次のものがあります。

- +err_file
- -err_file

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>ar850 +err_file=err ar850 d libarc.a sub.o
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>ar850 -err_file=err ar850 d libarc.a sub.o
```

B. 7　セクション・ファイル・ジェネレータ

ここでは，セクション・ファイル，およびセクション・ファイル・ジェネレータについて説明します。

B. 7. 1　セクション・ファイル

セクション・ファイルとは，C ソース・ファイル内で宣言されている外部変数（グローバル変数），静的変数（スタティック変数）を配置するセクションを定義するファイルです。コンパイル時にセクション・ファイルを参照することにより，これらの変数を配置するセクションを決定できます。デフォルトでは，V850 マイクロコントローラの内蔵 RAM 領域に配置するセクションである .tidata 属性，.tidata.word 属性，.tidata.byte 属性，.sidata 属性，.sedata 属性，.sdata 属性のセクションに，アクセス頻度の高い変数をできるだけ多く割り当てるようにしています。

C コンパイラでは，C ソース・ファイルにて外部変数を宣言し，それを意図したセクションに配置する方法には，次の 3 種類があります。

（1）コンパイル・オプション（-G*num*）により，.sdata セクション／ .sbss セクションへデータ・サイズを限定して配置する

（2）#pragma section 指令により，変数ごとに配置するセクションを決定する

（3）セクション・ファイルにより，コンパイラ起動時に指定した変数を配置する

（1）の方法は，あるサイズ以下の外部変数を .sdata か .sbss のどちらかに配置できればよい場合に適した方法です。コンパイル・オプションにて指定するため，C ソース・ファイルに変更を加える必要はありません。

配置セクションをもっと自由に設定したい場合は，（2）の方法のように C ソース・ファイルにて“#pragma section”指令を使用し，明示的に配置セクションを指定します。しかし，この場合は C ソース・ファイルに変更を加える必要があります。

配置セクションを自由に設定したいが，たとえば，ANSI に厳密に沿った記述をするために，#pragma section 指令を使用したくないときや，以前 CA850 以外でコンパイルした C ソース・ファイルを，ソースにあまり変更を加えずに CA850 用に移植したいときなどは，（1）や（2）の方法はあまり使用できません。

これを解決するのが，（3）の「セクション・ファイル」を使用する方法です。

セクション・ファイルにて，

- 静的変数の場合，その変数が宣言されている C ソース・ファイル名
- 外部変数名，静的変数名とそれらを配置したいセクション名

をすべての外部変数，静的変数に対して定義します。そして，CA850 にセクション・ファイルを参照させることにより，C ソース・ファイルに修正を加えることなく，これらの変数を意図した場所に配置できます。

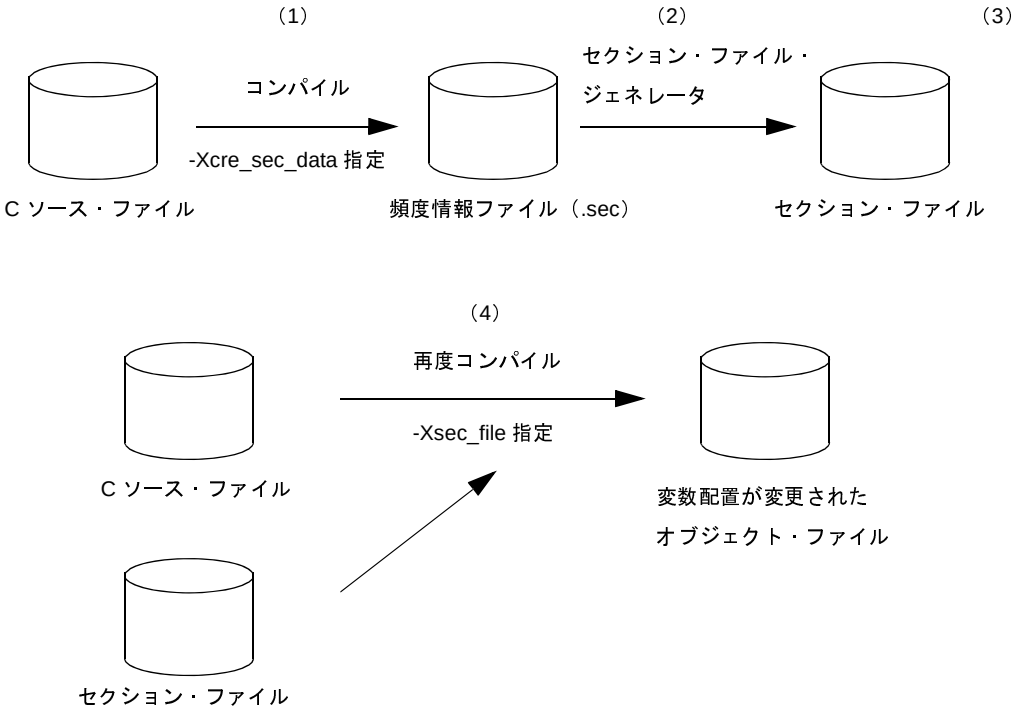
CA850 では，コンパイルのオプション指定（-Xcre_sec_data，または -Xcre_sec_data_only オプション）により，頻度情報ファイルが生成されますが，このファイルを「セクション・ファイル・ジェネレータ」に入力することによってセクション・ファイルを生成します。

ただし，セクション・ファイル・ジェネレータは V850 マイクロコントローラの内蔵 RAM への配置を目的とした“tidata 属性”，“tidata.word 属性”，“tidata.byte 属性”，“sidata 属性”，“sedata 属性”，“sdata 属性”のセクションにデータを配置するための情報を出力する仕様になっています。

なお，セクション・ファイルはテキスト形式のファイルなので，エディタなどで編集して変更することができます。つまり，セクション・ファイル・ジェネレータによって出力されたセクション・ファイルに変更を加え，最終的なセクション・ファイルを作成していきます。

完成したセクション・ファイルを使用（-Xsec_file オプション指定）して，もう一度コンパイルを行うと，外部変数，静的変数が指定したセクションに配置されたオブジェクトが完成します。

図 B—40　セクション・ファイル指定によるコンパイルのイメージ



（1）セクション・ファイルを作成するために，-Xcre_sec_data オプション指定で一度コンパイルします。

（2）セクション・ファイル・ジェネレータで頻度情報ファイルをセクション・ファイルに変換します。

（3）必要ならばセクション・ファイルを編集します。

（4）セクション・ファイルを入力するため -Xsec_file オプション指定で再度コンパイルします。

セクション・ファイルの書式については，「3. 4　セクション・ファイル・ジェネレータ」を参照してください。

セクション・ファイルにて配置指定ができる変数は，外部変数（グローバル変数），ファイル内静的変数（ファイル内で宣言されたスタティック変数），関数内静的変数（関数内で宣言されたスタティック変数）です。文字列定数（“abc”など）を配置指定することはできません。

複数の C ソース・ファイルを個々にコンパイルし，リンクしてオブジェクト・ファイルを生成する場合，それぞれに頻度情報出力を指定してコンパイルし，複数の .sec ファイルを生成することになります。ただし，セクション・ファイル生成時は，すべての .sec ファイルを一度にセクション・ファイル・ジェネレータに入力し，統合する

必要があります。そうしないと，外部変数に対して変数情報が統合されず，有効なセクション・ファイルを生成できません。

セクション・ファイルで指定した変数は“#pragma section”指令でセクション配置指定したことと同じになります。したがって，外部変数の仮定義は“定義”として扱われるため，複数ファイルで仮定義した場合，リンク時にエラーとなります。そのような場合，外部変数を参照するファイルでは，必ず extern 宣言する必要があります。

なお，セクション・ファイルで配置指定した変数が C ソース・ファイル内で #pragma section 指令によって違うセクションへの配置指定がされていた場合，セクション・ファイルによる指定が優先されます。

また，コンパイル・オプションで“-G*num*”が指定されていても，セクション・ファイルでその変数が .sdata ／ .sbss セクションへの配置を指定されていれば，*num* の大きさに関係なく，.sdata ／ .sbss セクションへ配置されます。つまり，「セクション・ファイル指定」「#pragma section 指定」「-G*num* 指定」の優先度は次のようになります。

（優先度高）セクション・ファイル指定 ＞ #pragma section 指定 ＞ -G*num* 指定（優先度低）

B. 7. 2　操作方法

ここでは，セクション・ファイル・ジェネレータの操作方法について説明します。

(1) コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>sf850 [オプション]... ファイル名 [ファイル名]...
      [ ]:    [ ]内は省略できます。
      ...:    直前の[ ]内のパターンの繰り返しができます。
```

(2) コマンド入力による利用

ここでは，コマンド・ライン上からセクション・ファイルを利用する方法について説明します。

(a) 始めに頻度情報ファイルを生成します。C コンパイラのオプション“-Xcre_sec_data_only”を指定して，C ソース・ファイルをコンパイルすると，その C ソース・ファイル中の外部変数，静的変数に関する頻度情報ファイルが生成されます。ファイル名はデフォルトで“C ソース・ファイル名 .sec”です。-Xcre_sec_data_only オプションと同時にファイル名を指定した場合は，指定したファイル名が頻度情報ファイル名になります。

例　func1.c に関する頻度情報をファイル“secsrc”に出力します。

```
C:￥>ca850 -cpu 3201 -Xcre_sec_data_only=secsrc func1.c
```

(b) 生成した頻度情報ファイルをセクション・ファイル・ジェネレータに入力し，セクション・ファイルを出力します。このとき，変数を tidata 属性，tidata.word 属性，tidata.byte 属性，sidata 属性，sedata 属性，sdata 属性のセクションへ配置指定するセクション・ファイルが生成されます。

例 頻度情報ファイル func1.sec，func2.sec，func3.sec をセクション・ファイルとして 1 つにまとめ，ファイル“secfile”に出力します。

```
C:¥>sf850 func1.sec func2.sec func3.sec -o secfile
```

ファイル数が多い場合は，コマンド・ファイルを作っておくと便利です。コマンド・ファイルについては，「(2) コマンド・ファイル」を参照してください。

(c) 出力されたセクション・ファイルは，デフォルトではすべての変数を .tidata 属性のセクションへ配置する指定になっているため，必要に応じてセクション・ファイルを修正します。
なお，セクション・ファイル・ジェネレータ起動時“-O オプション”を指定すると，参照頻度の高い順に .tidata 属性のセクションのメモリ範囲に収まる分の変数を自動的に選択できます。

(d) C コンパイラのオプション“-Xsec_file”を指定して，C ソース・ファイルを再コンパイルします。コンパイルの結果，入力したセクション・ファイルに従ったセクション配置のオブジェクト・ファイルが生成されます。

例 セクション・ファイルとして secfile を入力し，func1.c，func2.c，func3.c をコンパイルします。

```
C:¥>ca850 -cpu 3201 -Xsec_file secfile func1.c func2.c func3.c
```

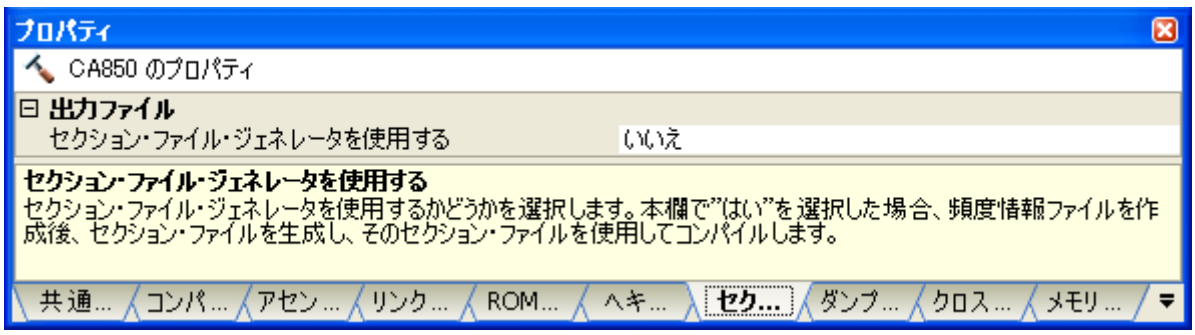
(3) CubeSuite+ でのオプション設定

CubeSuite+ からセクション・ファイル・ジェネレート・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネルにおいて，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［セクション・ファイル・ジェネレート・オプション］タブを選択します。

タブ上で各プロパティを設定することにより，対応するセクション・ファイル・ジェネレート・オプションを設定することができます。

図 B—41 プロパティ パネル：［セクション・ファイル・ジェネレート］タブ



B. 7. 3　オプション

ここでは，セクション・ファイル・ジェネレート・オプションについて説明します。

セクション・ファイル・ジェネレート・オプションの分類と説明を示します。

表 B—21　セクション・ファイル・ジェネレート・オプション

分類	オプション	説明
セクション・ファイル・ジェネレータ	-O	利用頻度の高い順に，最適化対象のセクションに配置可能な分だけの変数を判断して出力します。
	-V	セクション・ファイル・ジェネレータのバージョン情報を標準エラー出力に出力します。
	-Xcs	-O オプション指定時に指定されたセクションに割り付けられている変数を最適化の対象にしません。
	-Xcv	-O オプション指定時に指定された変数を最適化の対象にしません。
	-cl	出力するセクション・ファイルのコメント・レベルを指定します。
	+err_file	エラー・メッセージをファイルに追加保存します。
	-err_file	エラー・メッセージをファイルに上書き保存します。
	-h	セクション・ファイル・ジェネレータのオプションの説明を標準エラー出力します。
	-help	
	-ns	出力するセクション・ファイル中の変数名を出現した順番でソートします。
	-o	出力するセクション・ファイル名を指定します。
	-size_tidata	.tidata.word ／ .tidata.byte セクションに変数を配置するサイズの上限を制限します。
	-size_tidata_byte	.tidata.byte セクションに変数を配置するサイズの上限を制限します。
	-size_sidata	.sidata セクションに変数を配置するサイズの上限を制限します。
	-size_sedata	.sedata セクションに変数を配置するサイズの上限を制限します。
	-size_sdata	.sdata セクションに変数を配置するサイズの上限を制限します。
	-sname	出力するセクション・ファイル中の変数名を変数名の辞書順にソートします。
	-ssection	出力するセクション・ファイル中の変数名を割り当てられるセクション名の辞書順にソートします。
	-ssize	出力するセクション・ファイル中の変数名をサイズの小さい順にソートします。
	-v	セクション・ファイル・ジェネレータの実行過程を表示します。
	@	指定ファイルをコマンド・ファイルとして扱います。

セクション・ファイル・ジェネレータ

セクション・ファイル・ジェネレータのオプションには，次のものがあります。

- -O
- -V
- -Xcs
- -Xcv
- -cl
- +err_file
- -err_file
- -h/-help
- -ns
- -o
- -size_tidata
- -size_tidata_byte
- -size_sidata
- -size_sedata
- -size_sdata
- -sname
- -ssection
- -ssize
- -v
- @

-O

[記述形式]

```
-Oc
```

- 省略時解釈

  出現したすべての変数をセクション・ファイルに出力します。

[機能説明]

- *c* に何も指定しない場合は，利用頻度の高い順に，最適化対象のセクションに配置可能な分だけの変数を判断して出力します。
- .tidata セクションに配置可能なサイズは 256 バイトであり，内部的にはバイト・データの .tidata.byte（128 バイト）とワード・データの .tidata.word に分かれています。このオプションの指定では，その合計が 256 バイトになるまで変数を選択し，セクション・ファイルに出力します。

  ただし，バイト・データは 128 バイトに達したところで選択を終了します。
- *c* に 2 を指定した場合は，変数のサイズあたりの利用頻度の高い順に，.tidata，.sidata，.sedata，.sdata セクションに順番で配置できるように選択し，配置可能な分だけの変数を判断して出力します。

[使用例]

- 利用頻度の高い順に，最適化対象のセクションに配置可能な分だけの変数を判断して出力します。

```
C:¥>sf850 -O main.sec
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- セクション・ファイル・ジェネレータのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- セクション・ファイル・ジェネレータのバージョン情報を標準エラー出力に出力します。

```
C:￥>sf850 -V
```

-Xcs

[記述形式]

```
-Xcs[=name]
```

- 省略時解釈

  なし

[機能説明]

- -O，または -O2 オプション指定時に *name* で指定されたセクションに割り付けられている変数を最適化の対象にしません。
- *name* はリンク・ディレクティブ・ファイルに指定するセクション名を指定します。
- bss 属性のセクションは .bss ／ .sbss の部分を .data ／ .sdata に置き換えてください。
- *name* を省略した場合は，すべてのセクション名が指定されたものとみなします。
- *name* に .tidata が指定された場合は .tidata.word，.tidata.byte の 2 つが指定されたものとみなします。

[使用例]

- .const セクションに割り付けられている変数を最適化の対象にしません。

```
C:￥>sf850 -O -Xcs=.const main.sec
```

-Xcv

[記述形式]

```
-Xcv=name
```

- 省略時解釈

  なし

[機能説明]

- -O，または -O2 オプション指定時に *name* で指定された変数を最適化の対象にしません。
- *name* は「表 3—1　変数の種類と表示」の表示と同じ書式で指定します

[使用例]

- 変数 val を最適化の対象にしません。

```
C:￥>sf850 -O -Xcv=val main.sec
```

-cl

[記述形式]

```
-cl num
```

- 省略時解釈

  -cl 1

[機能説明]

- 出力するセクション・ファイルのコメント・レベルを指定します。
- *num* には次のものが指定できます。

0	コメントを出力しない。
1	日付などのファイル生成情報，および変数情報とその説明を出力する変数情報は，セクション名，サイズ，利用頻度である外部変数でセクション名が決定されていない場合，‘-’を出力する。
2	レベル 1 に加え，書式ガイドを出力する -O 指定時は，.tidata セクションに入らないと判断された変数もコメントとして出力する。

[使用例]

- 出力するセクション・ファイルのコメント・レベルを 2 とします。

```
C:￥>sf850 -cl 2 main.sec
```

+err_file

[記述形式]

```
+err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に追加保存します。

[使用例]

- エラー・メッセージをファイル err に追加保存します。

```
C:¥>sf850 +err_file=err main.sec
```

-err_file

[記述形式]

```
-err_file=file
```

- 省略時解釈

  なし

[機能説明]

- エラー・メッセージをファイル *file* に上書き保存します。

[使用例]

- エラー・メッセージをファイル err に上書き保存します。

```
C:¥>sf850 -err_file=err main.sec
```

-h/-help

[記述形式]

```
-h
-help
```

- 省略時解釈

  なし

[機能説明]

- セクション・ファイル・ジェネレータのオプションの説明を標準エラー出力し，終了します。

[使用例]

- セクション・ファイル・ジェネレータのオプションの説明を標準エラー出力します。

```
C:￥>sf850 -help
```

-ns

[記述形式]

```
-ns
```

- 省略時解釈

  出力するセクション・ファイル中の変数名を利用頻度順でソートします。

[機能説明]

- 出力するセクション・ファイル中の変数名をソートせず，出現した順番でソートします。

[使用例]

- 出力するセクション・ファイル中の変数名をソートせず，出現した順番でソートします。

```
C:￥>sf850 -ns main.sec
```

-o

[記述形式]

```
-o name
```

- 省略時解釈

  標準出力に出力します。

[機能説明]

- *name* を出力するセクション・ファイル名とします。

[使用例]

- secfile を出力するセクション・ファイル名とします。

```
C:¥>sf850 -o secfile main.sec
```

-size_tidata

［記述形式］

```
-size_tidata=num
```

- 省略時解釈

  -size_tidata=256

［機能説明］

- -O，または -O2 オプション指定時に .tidata.word/.tidata.byte セクションに変数を配置するサイズの上限を *num* バイトに制限します。
- *num* には 10 進で 0 ～ 2147483647 までの整数値が指定可能です。

［使用例］

- .tidata.word/.tidata.byte セクションに変数を配置するサイズの上限を 128 バイトに制限します。

```
C:￥>sf850 -O -size_tidata=128 main.sec
```

-size_tidata_byte

[記述形式]

```
-size_tidata_byte=num
```

- 省略時解釈

  -size_tidata_byte=128

[機能説明]

- -O，または -O2 オプション指定時に .tidata.byte セクションに変数を配置するサイズの上限を *num* バイトに制限します。
- *num* には 10 進で 0 ～ 2147483647 までの整数値が指定可能です。

[使用例]

- .tidata.byte セクションに変数を配置するサイズの上限を 64 バイトに制限します。

```
C:￥>sf850 -size_tidata_byte=64 main.sec
```

-size_sidata

[記述形式]

```
-size_sidata=num
```

- 省略時解釈

  -size_sidata=32512

[機能説明]

- -O2 オプション指定時に .sidata セクションに変数を配置するサイズの上限を *num* バイトに制限します。
- *num* には 10 進で 0 ～ 2147483647 までの整数値が指定可能です。

[使用例]

- .sidata セクションに変数を配置するサイズの上限を 32000 バイトに制限します。

```
C:￥>sf850 -size_sidata=32000 main.sec
```

-size_sedata

[記述形式]

```
-size_sedata=num
```

- 省略時解釈

  -size_sidata=32768

[機能説明]

- -O2 オプション指定時に .sedata セクションに変数を配置するサイズの上限を *num* バイトに制限します。
- *num* には 10 進で 0 ～ 2147483647 までの整数値が指定可能です。

[使用例]

- .sedata セクションに変数を配置するサイズの上限を 16384 バイトに制限します。

```
C:￥>sf850 -size_sedata=16384 main.sec
```

-size_sdata

[記述形式]

```
-size_sdata=num
```

- 省略時解釈

  -size_sdata=65536

[機能説明]

- -O2 オプション指定時に .sdata セクションに変数を配置するサイズの上限を *num* バイトに制限します。
- *num* には 10 進で 0 ～ 2147483647 までの整数値が指定可能です。

[使用例]

- .sdata セクションに変数を配置するサイズの上限を 32768 バイトに制限します。

```
C:￥>sf850 -size_sdata=32768 main.sec
```

-sname

[記述形式]

```
-sname
```

- 省略時解釈

  なし

[機能説明]

- 出力するセクション・ファイル中の変数名を変数名の辞書順にソートします。
- 変数名が同じ場合，ファイル名，関数名の辞書順にソートします。

[使用例]

- 出力するセクション・ファイル中の変数名を変数名の辞書順にソートします。

```
C:￥>sf850 -sname func.sec
```

-ssection

[記述形式]

```
-ssection
```

- 省略時解釈

  なし

[機能説明]

- 出力するセクション・ファイル中の変数名を割り当てられるセクション名の辞書順にソートします。
- セクション名が同じ場合，利用頻度の高い順にソートします。

[使用例]

- 出力するセクション・ファイル中の変数名を割り当てられるセクション名の辞書順にソートします。

```
C:￥>sf850 -ssection main.sec
```

-ssize

[記述形式]

```
-ssize
```

- 省略時解釈

  なし

[機能説明]

- 出力するセクション・ファイル中の変数名をサイズの小さい順にソートします。
- サイズが同じ場合，利用頻度の高い順にソートします。

[使用例]

- 出力するセクション・ファイル中の変数名をサイズの小さい順にソートします。

```
C:￥>sf850 -ssize main.sec
```

-v

[記述形式]

```
-v
```

- 省略時解釈

  なし

[機能説明]

- セクション・ファイル・ジェネレータの実行過程を表示します。

[使用例]

- セクション・ファイル・ジェネレータの実行過程を表示します。

```
C:¥>sf850 -v main.sec
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>sf850 @command
```

B. 7. 4　注意事項

セクション・ファイル・ジェネレータのオプションの中には，他のオプションと同時に指定された場合，無効になるものがあります。

- ソートに関するオプション，-o，および -cl は，複数指定された場合，あとから指定されたものを有効とし，その他を無効とします。
- -V，-h，-help が同時に指定された場合，先に指定されたものを有効とし，その他を無効とします。
- -O とソートに関するオプションが同時に指定された場合，-O を有効とし，ソートに関するオプションは無効とします。
- セクション・ファイル・ジェネレータに入力する頻度情報ファイルはＣコンパイラが出力したそのままの状態のものを使用してください。その内容を修正した頻度情報ファイルを入力した場合，動作は保証されません。

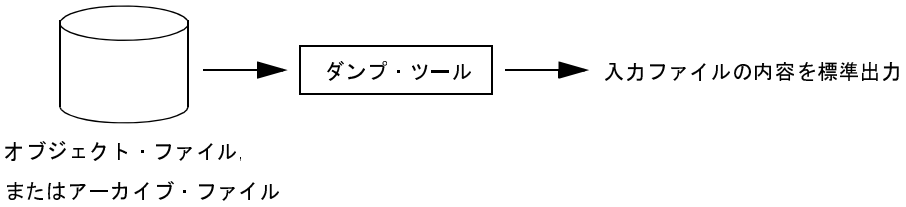
セクション・ファイル・ジェネレータによって出力されるセクション・ファイルの内容については「3. 4　セクション・ファイル・ジェネレータ」を参照してください。

B. 8　ダンプ・ツール

ダンプ・ツールとは，指定されたオブジェクト・ファイルやアーカイブ・ファイルの内容や情報を表示します。作成したオブジェクト・ファイルやアーカイブ・ファイル中の，セクション／セグメントのアドレスや属性，シンボル名などの情報を確認する場合などに利用します。

CA850 に入っている“dump850”がダンプ・ツールです。

図 B―42　ダンプ・ツールにおける動作の流れ



なお，アーカイブ・ファイルをダンプ・ツールに入力した場合，アーカイブ・ファイル内に“オブジェクト・ファイルではないメンバ”が存在した場合，警告メッセージを出力して，次のメンバの処理を行います。ただし，-e オプションを指定されている場合を除きます。

オプションについての詳細は，「B. 8. 2　オプション」を参照してください。

B. 8. 1　操作方法

ここでは，ダンプ・ツールの操作方法について説明します。

（1）コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>dump850 [オプション]... ファイル名 [ファイル名]...
      [ ]:    [ ]内は省略できます。
      ...:    直前の[ ]内のパターンの繰り返しができます。
```

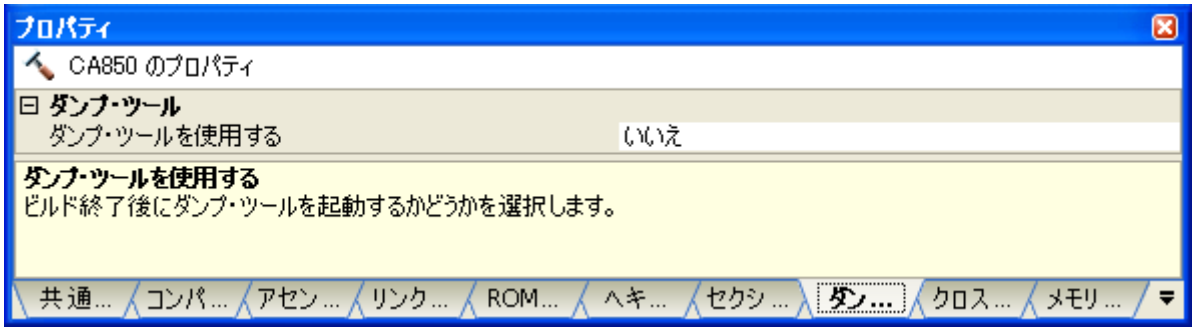
（2）CubeSuite+ でのオプション設定

CubeSuite+ からダンプ・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネル上において，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［ダンプ・オプション］タブを選択します。

タブ上で必要なプロパティを設定することにより，各ダンプ・オプションを設定することができます。

図 B—43　プロパティ パネル：［ダンプ・オプション］タブ



B. 8. 2　オプション

ここでは，ダンプ・オプションについて説明します。

ダンプ・オプションの分類と説明を示します。

表 B―22　ダンプ・オプション

分類	オプション	説明
ダンプ・ツール	-A	指定されたオブジェクト・ファイル，またはアーカイブ・ファイルのすべての内容を表示します。
	-T	アーカイブ・ヘッダの内容の表示においてメンバの更新年月日を表示しません。
	-V	ダンプ・ツールのバージョン情報を標準エラー出力に出力します。
	-a	指定されたファイル内に存在する，すべてのメンバのアーカイブ・ヘッダの内容を表示します。
	-b	デバッグ情報の内容を表示します。
	-c	ストリング・テーブルの内容を表示します。
	-d	セクション・ヘッダ・テーブル・インデックスで示されるセクション“から”表示します。
	+d	セクション・ヘッダ・テーブル・インデックスで示されるセクション“まで”表示します。
	-e	指定されたアーカイブ・ファイル内に存在するメンバの内容を表示します。
	-f	指定されたオブジェクト・ファイル，またはアーカイブ・ファイル内に存在するすべてのメンバの ELF ヘッダの内容を表示します。
	-g	指定されたアーカイブ・ファイルのアーカイブ・シンボル・テーブル内に存在する外部シンボルの内容を表示します。
	-h	指定されたオブジェクト・ファイル，またはアーカイブ・ファイル内に存在するすべてのセクション・ヘッダの内容を表示します。
	-i	指定されたオブジェクト・ファイル，またはアーカイブ・ファイル内に存在するすべてのプログラム・ヘッダの内容を表示します。
	-k	グローバル・ポインタ・テーブルの内容を表示します。
	-l	ライン・ナンバ情報の内容を表示します。
	-m	指定されたファイルのアーカイブ・ストリング・テーブル内に存在するストリングの内容を表示します。
	-n	指定されたセクションの内容を表示します。
	-p	タイトルを表示しません。
	-r	リロケーション情報の内容を表示します。
	-s	セクションの内容を表示します。
	-t	シンボル・テーブルの内容を指定箇所のシンボル・テーブル・エントリから表示します。
	+t	シンボル・テーブルの内容を指定箇所のシンボル・テーブル・エントリまで表示します。
	-v	セクション属性などの値を，その値の意味を示す文字列で表示します。
	-z	関数のライン・ナンバ情報の内容を指定箇所のライン・ナンバ・エントリから表示します。
	+z	ライン・ナンバ情報の内容を指定箇所のライン・ナンバ・エントリまで表示します。
	@	指定ファイルをコマンド・ファイルとして扱います。

ダンプ・ツール

ダンプ・ツールのオプションには，次のものがあります。

- -A
- -T
- -V
- -a
- -b
- -c
- -d
- +d
- -e
- -f
- -g
- -h
- -i
- -k
- -l
- -m
- -n
- -p
- -r
- -s
- -t
- +t
- -v
- -z
- +z
- @

-A

[記述形式]

```
-A
```

- 省略時解釈

  -A

[機能説明]

- 指定されたオブジェクト・ファイル，またはアーカイブ・ファイルのすべての内容を表示します。
- このオプションは“-a b c f g h i k l m r s t”を指定したことに等しくなります。どのオプションも指定されなかった場合，-A オプションが指定されたものとみなします。

[使用例]

- a.out のすべての内容を表示します。

```
C:￥>dump850 -A a.out
```

-T

[記述形式]

```
-T
```

- 省略時解釈

  なし

[機能説明]

- アーカイブ・ヘッダの内容の表示においてメンバの更新年月日を表示しません。

[使用例]

- アーカイブ・ヘッダの内容の表示においてメンバの更新年月日を表示しません。

```
C:¥>dump850 -T libarc.a
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- ダンプ・ツールのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- ダンプ・ツールのバージョン情報を標準エラー出力に出力します。

```
C:¥>dump850 -V
```

-a

[記述形式]

```
-a
```

- 省略時解釈

  -a

[機能説明]

- 指定されたアーカイブ・ファイル内に存在する，すべてのメンバのアーカイブ・ヘッダの内容を表示します。

[使用例]

- libarc.a 内に存在する，すべてのメンバのアーカイブ・ヘッダの内容を表示します。

```
C:¥>dump850 -a libarc.a
```

-b

[記述形式]

```
-b
```

- 省略時解釈

  -b

[機能説明]

- デバッグ情報の内容を表示します。

[使用例]

- デバッグ情報の内容を表示します。

```
C:¥>dump850 -b a.out
```

-c

[記述形式]

```
-c
```

- 省略時解釈

  -c

[機能説明]

- ストリング・テーブルの内容を表示します。

[使用例]

- ストリング・テーブルの内容を表示します。

```
C:¥>dump850 -c a.out
```

-d

[記述形式]

```
-d num
```

- 省略時解釈

  すべてのセクションを表示します。

[機能説明]

- セクション・ヘッダ・テーブル・インデックス *num* で示されるセクション“から”表示します。

[使用例]

- セクション・ヘッダ・テーブル・インデックスの 2 で示されるセクションから表示します。

```
C:¥>dump850 -d 2 a.out
```

+d

[記述形式]

```
+d num
```

- 省略時解釈

  すべてのセクションを表示します。

[機能説明]

- セクション・ヘッダ・テーブル・インデックス *num* で示されるセクション“まで”表示します。

[使用例]

- セクション・ヘッダ・テーブル・インデックスの 9 で示されるセクションまで表示します。

```
C:¥>dump850 +d 9 a.out
```

-e

[記述形式]

```
-e
```

- 省略時解釈

  なし

[機能説明]

- 指定されたアーカイブ・ファイル内に存在する（アーカイブ・シンボル・テーブル，アーカイブ・ストリング・テーブル，およびオブジェクト・ファイル以外の）メンバの内容を表示します。

[使用例]

- libarc.a 内に存在する（アーカイブ・シンボル・テーブル，アーカイブ・ストリング・テーブル，およびオブジェクト・ファイル以外の）メンバの内容を表示します。

```
C:￥>dump850 -e libarc.a
```

-f

[記述形式]

```
-f
```

- 省略時解釈

  -f

[機能説明]

- 指定されたオブジェクト・ファイル，またはアーカイブ・ファイル内に存在するすべてのメンバの ELF ヘッダの内容を表示します。

[使用例]

- a.out 内に存在するすべてのメンバの ELF ヘッダの内容を表示します。

```
C:￥>dump850 -f a.out
```

-g

[記述形式]

```
-g
```

- 省略時解釈

  -g

[機能説明]

- 指定されたアーカイブ・ファイルのアーカイブ・シンボル・テーブル内に存在する外部シンボルの内容を表示します。

[使用例]

- libarc.a のアーカイブ・シンボル・テーブル内に存在する外部シンボルの内容を表示します。

```
C:￥>dump850 -g libarc.a
```

-h

[記述形式]

```
-h
```

- 省略時解釈

  -h

[機能説明]

- 指定されたオブジェクト・ファイル，またはアーカイブ・ファイル内に存在するすべてのセクション・ヘッダの内容を表示します。

[使用例]

- a.out 内に存在するすべてのセクション・ヘッダの内容を表示します。

```
C:￥>dump850 -h a.out
```

-i

[記述形式]

```
-i
```

- 省略時解釈

  -i

[機能説明]

- 指定されたオブジェクト・ファイル，またはアーカイブ・ファイル内に存在するすべてのプログラム・ヘッダの内容を表示します。

[使用例]

- a.out 内に存在するすべてのプログラム・ヘッダの内容を表示します。

```
C:￥>dump850 -i a.out
```

-k

[記述形式]

```
-k
```

- 省略時解釈

  -k

[機能説明]

- グローバル・ポインタ・テーブルの内容を表示します。

[使用例]

- グローバル・ポインタ・テーブルの内容を表示します。

```
C:¥>dump850 -k a.out
```

-l

[記述形式]

```
-l
```

- 省略時解釈

  -l

[機能説明]

- ライン・ナンバ情報の内容を表示します。

[使用例]

- ライン・ナンバ情報の内容を表示します。

```
C:¥>dump850 -l a.out
```

-m

[記述形式]

```
-m
```

- 省略時解釈

  -m

[機能説明]

- 指定されたアーカイブ・ファイルのアーカイブ・ストリング・テーブル内に存在するストリングの内容を表示します。

[使用例]

- libarc.a のアーカイブ・ストリング・テーブル内に存在するストリングの内容を表示します。

```
C:￥>dump850 -m libarc.a
```

-n

[記述形式]

```
-n name
```

- 省略時解釈

  すべてのセクションの内容を表示します。

[機能説明]

- セクション名 *name* で示されるセクションの内容を表示します。

[使用例]

- .text セクションの内容を表示します。

```
C:¥>dump850 -n .text a.out
```

-p

[記述形式]

```
-p
```

- 省略時解釈

  なし

[機能説明]

- タイトルを表示しません。

[使用例]

- タイトルを表示しません。

```
C:¥>dump850 -p a.out
```

-r

[記述形式]

```
-r
```

- 省略時解釈

  リロケーション情報の内容を表示します。

[機能説明]

- リロケーション情報の内容を表示します。

[使用例]

- リロケーション情報の内容を表示します。

```
C:¥>dump850 -r main.o
```

-S

[記述形式]

```
-s
```

- 省略時解釈

  -s

[機能説明]

- セクションの内容を表示します。

[使用例]

- セクションの内容を表示します。

```
C:¥>dump850 -s a.out
```

-t

[記述形式]

```
-t [num]
```

- 省略時解釈

  すべてのシンボル・テーブルの内容を表示します。

[機能説明]

- シンボル・テーブルの内容を *num* 番目のシンボル・テーブル・エントリから表示します。
- *num* を省略した場合，1 番目のシンボル・テーブル・エントリから表示します。

[使用例]

- シンボル・テーブルの内容を 5 番目のシンボル・テーブル・エントリから表示します。

```
C:￥>dump850 -t 5 a.out
```

+t

[記述形式]

```
+t num
```

- 省略時解釈

  すべてのシンボル・テーブルの内容を表示します。

[機能説明]

- シンボル・テーブルの内容を *num* 番目のシンボル・テーブル・エントリまで表示します。

[使用例]

- シンボル・テーブルの内容を 10 番目のシンボル・テーブル・エントリまで表示します。

```
C:¥>dump850 +t 10 arc.a
```

-v

[記述形式]

```
-v
```

- 省略時解釈

  セクション属性などの値を数字で表示します。

[機能説明]

- セクション属性などの値を数字ではなく，その値の意味を示す文字列で表示します（「3. 5. 2　要素の値と意味」参照）。

[使用例]

- セクション属性などの値を数字ではなく，その値の意味を示す文字列で表示します。

```
C:￥>dump850 -v a.out
```

-z

[記述形式]

```
-z name [num]
```

- 省略時解釈

  すべての関数のライン・ナンバ情報の内容を表示します。

[機能説明]

- 関数 *name* のライン・ナンバ情報の内容を *num* 番目のライン・ナンバ・エントリから表示します。
- *num* を省略した場合，1 番目のライン・ナンバ・エントリから表示します。

[使用例]

- 関数 func のライン・ナンバ情報の内容を 1 番目のライン・ナンバ・エントリから表示します。

```
C:￥>dump850 -z func a.out
```

+z

[記述形式]

```
+z num
```

- 省略時解釈

  すべての関数のライン・ナンバ情報の内容を表示します。

[機能説明]

- ライン・ナンバ情報の内容を *num* 番目のライン・ナンバ・エントリまで表示します。

[使用例]

- 関数 func のライン・ナンバ情報の内容を 10 番目のライン・ナンバ・エントリまで表示します。

```
C:¥>dump850 -z func +z 10 a.out
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

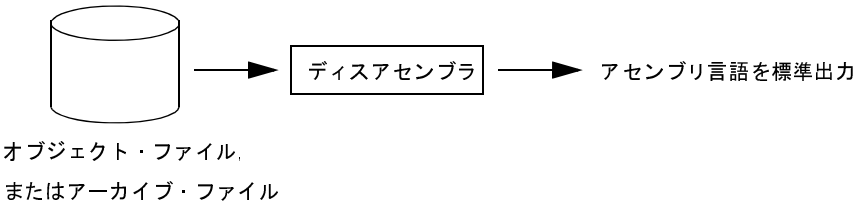
```
C:￥>dump850 @command
```

B. 9　ディスアセンブラ

ディスアセンブラとは，コンパイル／アセンブル後のオブジェクト・ファイルや，アーカイバで作成されたアーカイブ・ファイルなどから，プログラム・コードをアセンブリ言語に変換し，出力するユーティリティです。オブジェクトがどのようなコードになっているかを検証したい場合などに使用します。

CA850 に入っている“dis850”がディスアセンブラです。

図 B—44　ディスアセンブラにおける動作の流れ



B. 9. 1　操作方法

ここでは，ディスアセンブラの操作方法について説明します。

(1) コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>dis850 [オプション]... ファイル名 [ファイル名]...
      [ ]:    [ ]内は省略できます。
      ...:    直前の[ ]内のパターンの繰り返しができます。
```

B. 9. 2　オプション

ここでは，ディスアセンブル・オプションついて説明します。

注意　どのオプションも指定されなかった場合，-o オプションが指定されたものとみなします。

ディスアセンブル・オプションの分類と説明を示します。

表 B―23　ディスアセンブル・オプション

分類	オプション	説明
ディスアセンブラ	-A	オプション -aoptr を指定したものとみなします。
	-F	デバイス・ファイルを指定フォルダから検索します。
	-V	ディスアセンブラのバージョン情報を標準エラー出力に出力します。
	-a	アドレスを表示します。
	-c	コード（アセンブラ命令，データ）を表示します。
	-e	エンド・アドレスを指定します。
	-l	表示するサイズを指定します。
	-m	アセンブラ・ソースの形式で表示します。
	-o	シンボルからのオフセットを表示します。
	-p	プロセッサの命令フォーマットに従って並べられたコードを表示します。
	-r	レジスタの r0，r2，r3，r4，r5，r30，r31 を，zero，hp，sp，gp，tp，ep，lp として表示します。
	-s	スタート・アドレスを指定します。
	-t	表示内容を示すタイトルを表示します。
	-v	コメントなどを表示します。
	@	指定ファイルをコマンド・ファイルとして扱います。

ディスアセンブラ

ディスアセンブラのオプションには，次のものがあります。

-A

[記述形式]

```
-A
```

- 省略時解釈

  -o

[機能説明]

- オプション -aoptr を指定したものとみなします。

[使用例]

- a.out のアドレス，シンボルからのオフセット，コード，表示内容を示すタイトルを表示します。また，レジスタの r0，r2，r3，r4，r5，r30，r31 を，zero，hp，sp，gp，tp，ep，lp として表示します。

```
C:￥>dis850 -A a.out
```

-F

[記述形式]

```
-F devpath
```

- 省略時解釈

  デバイス・ファイルを，標準フォルダから検索します。

[機能説明]

- デバイス・ファイルをフォルダ *devpath* から検索します。

[使用例]

- デバイス・ファイルを，フォルダ D:￥dev から探します。

```
C:￥>dis850 -F D:￥dev a.out
```

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- ディスアセンブラのバージョン情報を標準エラー出力に出力し，終了します。

[使用例]

- ディスアセンブラのバージョン情報を標準エラー出力に出力します。

```
C:¥>dis850 -V
```

-a

[記述形式]

```
-a
```

- 省略時解釈

  なし

[機能説明]

- オブジェクト・ファイル，またはアーカイブ・ファイルに含まれている情報のうち，アドレスを表示します。

[使用例]

- a.out に含まれている情報のうち，アドレスを表示します。

```
C:￥>dis850 -a a.out
```

-c

[記述形式]

```
-c
```

- 省略時解釈

  なし

[機能説明]

- オブジェクト・ファイル，またはアーカイブ・ファイルのコード（アセンブラ命令，データ）を表示します。

[使用例]

- a.out のコード（アセンブラ命令，データ）を表示します。

```
C:¥>dis850 -c a.out
```

-e

[記述形式]

```
-e address
```

- 省略時解釈

  -e 0xffffffff

[機能説明]

- エンド・アドレスを指定します。
- *address* は 10 進数，または 0x を先頭に付けた 16 進数で指定します。

[使用例]

- エンド・アドレスを 0xffff とします。

```
C:￥>dis850 -e 0xffff a.out
```

-l

[記述形式]

```
-l size
```

- 省略時解釈

  -l 0xffffffff

[機能説明]

- 表示するサイズを指定します。
- *size* は 10 進数，または 0x を先頭に付けた 16 進数で指定します。

[使用例]

- 表示するサイズを 0xffff とします。

```
C:¥>dis850 -l 0xffff a.out
```

-m

[記述形式]

```
-m
```

- 省略時解釈

  シンボル・オフセットなどとともにアセンブラ・ソースを表示します。

[機能説明]

- アセンブラ・ソースの形式で表示します。

[使用例]

- アセンブラ・ソースの形式で表示します。

```
C:¥>dis850 -m a.out
```

-o

[記述形式]

```
-o
```

- 省略時解釈

  -a オプション，または -m オプションが指定されていなければシンボルからのオフセットを表示します。

[機能説明]

- オブジェクト・ファイル，またはアーカイブ・ファイルに含まれている情報のうち，シンボルからのオフセットを表示します。

[使用例]

- a.out に含まれている情報のうち，シンボルからのオフセットを表示します。

```
C:￥>dis850 -o a.out
```

-p

[記述形式]

```
-p
```

- 省略時解釈

  なし

[機能説明]

- オブジェクト・ファイル，またはアーカイブ・ファイルに含まれている情報のうち，プロセッサの命令フォーマットに従って並べられたコードを表示します。
- -c オプションが指定された場合は -c を優先します。

[使用例]

- a.out に含まれている情報のうち，プロセッサの命令フォーマットに従って並べられたコードを表示します。

```
C:￥>dis850 -p a.out
```

-r

[記述形式]

```
-r
```

- 省略時解釈

  レジスタはすべて rnum の形で表示します。num は 0 から 31 の数値です。

[機能説明]

- レジスタの r0，r2，r3，r4，r5，r30，r31 を，zero，hp，sp，gp，tp，ep，lp として表示します。

[使用例]

- レジスタの r0，r2，r3，r4，r5，r30，r31 を，zero，hp，sp，gp，tp，ep，lp として表示します。

```
C:¥>dis850 -r a.out
```

-s

［記述形式］

```
-s address
```

- 省略時解釈

  -s 0x0

［機能説明］

- スタート・アドレスを指定します。
- *address* は 10 進数，または 0x を先頭に付けた 16 進数で指定します。
- *address* の数値が 0xfffffffe より大きい場合は無視します。

［使用例］

- スタート・アドレスを 0x1000 とします。

```
C:￥>dis850 -s 0x1000 a.out
```

-t

[記述形式]

```
-t
```

- 省略時解釈

  なし

[機能説明]

- オブジェクト・ファイル，またはアーカイブ・ファイルに含まれている情報のうち，表示内容を示すタイトルを表示します。

[使用例]

- a.out に含まれている情報のうち，表示内容を示すタイトルを表示します。

```
C:￥>dis850 -t a.out
```

-v

[記述形式]

```
-v
```

- 省略時解釈

  なし

[機能説明]

- コメントなどを表示します。

[使用例]

- コメントなどを表示します。

```
C:¥>dis850 -v a.out
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>dis850 @command
```

B. 9. 3　注意事項

注意事項を以下に示します。

- オブジェクト・ファイルに同じアドレスのラベルが存在する場合，シンボル・テーブル内で後ろに出現した方が優先されます。
- プログラムが 0 番地から始まっていて，かつ 0 番地を示すシンボルが存在しないオブジェクトのため，出力時に 0 番地のシンボル出力が必要とされるような場合，“__dummy”を 0 番地のシンボルとして出力することがあります。

B. 10　クロス・リファレンス・ツール

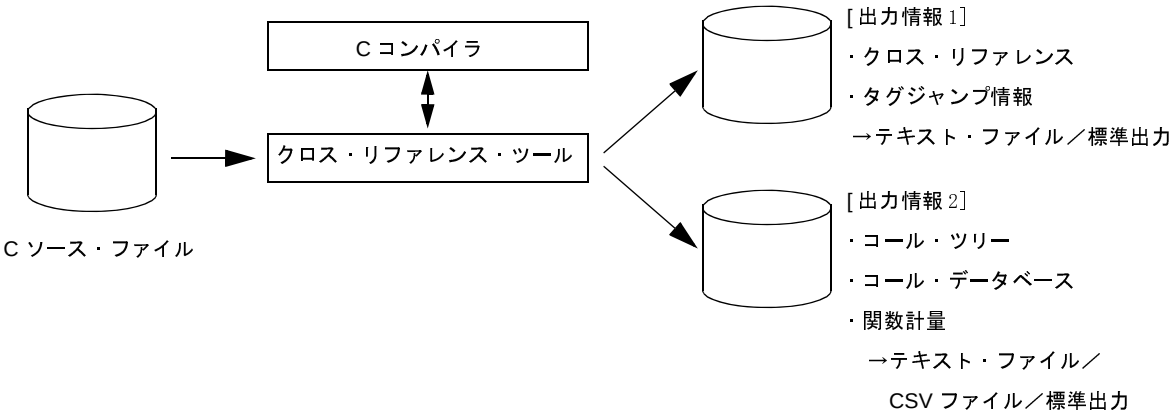
クロス・リファレンス・ツール“cxref”は，C ソース・ファイルを基に，識別子の参照と定義位置を検出するツールです。対象となる識別子は，関数，および変数（auto 変数以外）であり，その記憶クラスも識別します。検出結果として，クロス・リファレンス情報と，タグ・ジャンプ情報を出力します。また，関数単位の解析を行い，コール・ツリー，関数計量，コール・データベースを出力することもできます。

クロス・リファレンス・ツールの処理において，“参照”とは，式の中にその識別子が現れることをいいます。“定義”とは，宣言文の中にその識別子が現れることをいいます。式か宣言文か判断ができないものについては，クロス・リファレンスは“不明”として扱います。

クロス・リファレンス・ツールが出力するコール・ツリー，関数計量，コール・データベースは，次のような特長があります。

- ターゲット，および ca850 の最適化に依存しない
- オプションにより，標準出力が可能

図 B―45　クロス・リファレンス・ツールにおける動作の流れ



B. 10. 1　入 出 力

（1）入力ファイル

クロス・リファレンス・ツールの入力ファイルは，C ソース・ファイルです。クロス・リファレンス・ツールの起動時に，“-cpp850”オプションを指定すると，指定した C ソース・ファイルをプリプロセッサに通したあと，クロス・リファレンス・ツールが処理を行います。

- クロス・リファレンス・ツールは，入力する C ソース・ファイルに構文エラーが含まれないことを前提に処理を行います。
  C ソース・ファイルは一度コンパイルを実行し，構文エラーがないことを確認してください。
- 文字セットは，シフト JIS とします。
- クロス・リファレンス・ツールは，C ソース・ファイルに含まれるプリプロセス指令をエラー扱いせず，単に無視して解析を行います。したがって，次のものを含まない C ソース・ファイルであれば，ca850 を通っていないファイルであっても，“-cpp850”オプション指定せず，そのまま処理することができます。これは，ヘッダ・ファイルを無視したい場合，偽条件ブロック内も解析の対象としたい場合，およびマクロ名をクロス・リファレンスの対象としたい場合に有効です。

- “{}”のバランスを乱す条件ブロック
- 制御構造のマクロ化
- 宣言文のマクロ化

- 入力ファイルには行番号情報とコメント情報を含めることができます。

（2）出力情報

クロス・リファレンス・ツールが出力する情報は，次のとおりです。

（a）クロス・リファレンス

ファイルごとに，そのファイル内で使用されている変数，および関数のクロス・リファレンス情報を出力します。

（b）タグ情報

変数，および関数について，その定義ファイル名と行番号の情報（タグ・ジャンプ情報）を出力します。

（c）コール・ツリー

ある関数が，どの関数を呼び出しているかをツリー状に出力します。

（d）関数計量

関数の“ライン数”，“呼び出される頻度”の情報を出力します。

（e）コール・データベース

ある関数が，どの関数を何回呼び出しているかを出力します。

これらの情報についての詳細は，「3. 7　クロス・リファレンス・ツール」を参照してください。

B. 10. 2　操作方法

ここでは，クロス・リファレンス・ツールの操作方法について説明します。

（1）コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>cxref ［オプション］... ［ファイル名］...
      [ ]：    [ ]内は省略できます。
      ...：    直前の[ ]内のパターンの繰り返しができます。
```

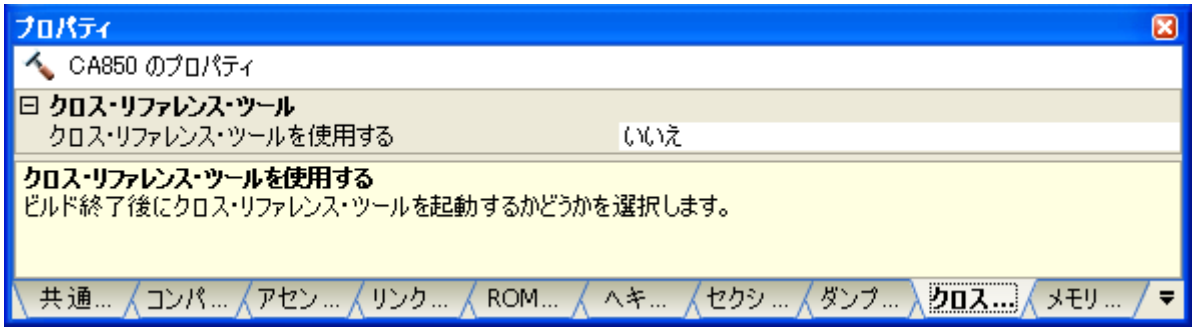
（2）CubeSuite+ でのオプション設定

CubeSuite+ からクロス・リファレンス・オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネル上において，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［クロス・リファレンス・オプション］タブを選択します。

タブ上で必要なプロパティを設定することにより，各クロス・リファレンス・オプションを設定することができます。

図 B—46　プロパティ パネル：［クロス・リファレンス・オプション］タブ



B. 10. 3　オプション

ここでは，クロス・リファレンス・ツールのオプションについて説明します。

クロス・リファレンス・オプションの分類と説明を示します。

表 B—24　クロス・リファレンス・オプション

分類	オプション	説明
共通オプション	-V	クロス・リファレンス・ツールのバージョン情報を標準出力に出力します。
	-all	すべての情報を，テキスト形式，および CSV 形式の各ファイルに出力します。
	-cpp850	C ソース・ファイルを ca850（プリプロセッサ）に通してから処理します。
	-d	型名として扱う識別子，または識別子を記述したファイル名を指定します。
	-file	情報を記述したファイル名を指定します。
	-h	オプションの説明を出力します。
	-help	
	-i	実行結果に表示しない識別子を指定します。
	-ni	インクルード・ファイルの情報を表示しません。 または，実行結果に表示しないファイル名を指定します。
	-o	出力ファイルのパスを指定します。
	@	指定ファイルをコマンド・ファイルとして扱います。
クロス・リファレンス	-x	クロス・リファレンスをテキスト形式で，指定したファイルに出力します。
	-xstd	クロス・リファレンスを標準出力に出力します。
タグ情報	-t	タグ情報をテキスト形式で，指定したファイルに出力します。
	-tstd	タグ情報を標準出力に出力します。

分類	オプション	説明
コール・ツリー	-c	コール・ツリーをテキスト形式で，指定したファイルに出力します。
	-cc	コール・ツリーを CSV 形式で，指定したファイルに出力します。
	-call	コール・ツリーをテキスト形式と CSV 形式で，指定したファイルに出力します。
	-ce	出力の省略方法を指定します。
	-cf	コール・ツリーを出力する関数名，またはコール・ツリーを出力する関数名を記述したテキスト・ファイルを指定します。
	-cl	出力レベルを指定します。
	-cp	引数，戻り値を含めて出力します。
	-cr	参照情報を含めて出力します。
	-cs	ソース・ファイル名，記述開始行を含めて出力します。
	-cstd	テキスト形式のコール・ツリーを標準出力に出力します。
	-ct	先頭のツリーのみ出力します。
関数計量	-m	関数計量をテキスト形式で，指定したファイルに出力します。
	-mc	関数計量を CSV 形式で，指定したファイルに出力します。
	-mall	関数計量をテキスト形式と CSV 形式で，指定したファイルに出力します。
	-ms	出力順序を指定します。
	-mstd	テキスト形式の関数計量を標準出力に出力します。
コール・データベース	-b	コール・データベースをテキスト形式で，指定したファイルに出力します。
	-bc	コール・データベースを CSV 形式で，指定したファイルに出力します。
	-ball	コール・データベースをテキスト形式と CSV 形式で，指定したファイルに出力します。
	-mstd	テキスト形式のコール・データベースを標準出力に出力します。

共通オプション

クロス・リファレンス・ツールの共通オプションには，次のものがあります。

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- クロス・リファレンス・ツールの版番号を出力し，終了します。

[使用例]

- クロス・リファレンス・ツールの版番号を出力します。

```
C:¥>cxref -V
```

-all

[記述形式]

```
-all
```

- 省略時解釈

  クロス・リファレンスを標準出力に出力します。

[機能説明]

- すべての情報を，テキスト形式，および CSV 形式の各ファイルに出力します。
- “-x -t -c -cc -m -mc -b -bc”を指定したのと同じです。

[使用例]

- すべての情報を，テキスト形式，および CSV 形式の各ファイルに出力します。

```
C:￥>cxref -all main.c
```

-cpp850

[記述形式]

```
-cpp850
```

- 省略時解釈

  ca850（プリプロセッサ）を実行しません。

[機能説明]

- C ソース・ファイルを ca850（プリプロセッサ）に通してから処理します。
- このオプション以降のオプションは，すべて ca850 のオプションとして渡されます。したがって，このオプションは，クロス・リファレンス・オプションとしては，最後に指定する必要があります。
- 行番号をより正確に出力するために，プリプロセッサで，ソース・プログラムのコメントを含める“-C”オプションを指定することを推奨します。

[使用例]

- C ソース・ファイルを ca850（プリプロセッサ）に通してから処理します。

```
C:¥>cxref -cpp850 main.c
```

-d

[記述形式]

```
-dident
-d=file
```

- 省略時解釈

  なし

[機能説明]

- 型名として扱う識別子 *ident* を指定します。
- 型名として扱う識別子を記述したファイル名 *file* を指定します。

[使用例]

- 識別子 U16 を型名として扱います。

```
C:¥>cxref -dU16 main.c
```

-file

[記述形式]

```
-file=file
```

- 省略時解釈

  なし

[機能説明]

- 次の情報を記述したファイル名 *file* を指定します。
  - 実行結果に表示しないファイル名
  - 実行結果に表示しない識別子名
  - 型名として扱う識別子名
- -file=*file* と -ni を同時に指定した場合，前に指定した -file=*file* の *file* における“NoIncludeFile”の内容は無効となります。
- -ni ／ -i ／ -d ／ -file オプションに指定するファイルの形式

  -ni ／ -i ／ -d でそれぞれのオプションで該当するセクション情報を読み出し，-file オプションによりすべてのセクション情報を読み出します。

  次の 3 つのセクションを記述できます。
  - NoIncludeFile セクション
  - IgnoreIdent セクション
  - DefinitionType セクション

  なお，行頭が“//”で始まる行は，コメントとして扱います。

（1）NoIncludeFile セクション

解析結果として表示しない情報をファイル単位で指定します。主にインクルード・ファイルを記述します。

ここに記述したファイル名は -ni オプションに続いて指定する場合と同じ効果があります。

ファイル名は 1 行に 1 つ記述します。

ワイルド・カードが使用可能です。

```
[NoIncludeFile]
// *.h のファイルすべて
*.h
// 共通の定義ファイル
common.def
```

（2）IgnoreIdent セクション

解析結果として表示しない情報を識別子単位で指定します。

ここに記述したファイル名は -i オプションに続いて指定する場合と同じ効果があります。

識別子は 1 行に 1 つ記述します。

```
[IgnoreIdent]
// 各処理で一時的に使用する共通エリア
tmp
buf
work
```

（3）DefinitionType セクション

型名として扱う識別子を指定します。

ここに記述したファイル名は -d オプションに続いて指定する場合と同じ効果があります。

識別子は 1 行に 1 つ記述します。

```
[DefinitionType]
// 1 バイトの型
BYTE
UBYTE
// 2 バイトの型
WORD
UWORD
```

［使用例］

- ファイル noresult で指定したファイル名，識別子の情報を解析結果として表示しません。
  また，noresult で指定した識別子を型名として扱います。

```
C:￥>cxref -file=noresult main.c
```

-h/-help

[記述形式]

```
-h
-help
```

- 省略時解釈

  なし

[機能説明]

- オプションの説明を出力し，終了します。

[使用例]

- cxref のオプションの説明を出力します。

```
C:¥>cxref -help
```

-i

[記述形式]

```
-iident
```

- 省略時解釈

  なし

[機能説明]

- 実行結果に表示しない識別子を指定します。

[使用例]

- 識別子 data を実行結果に表示しません。

```
C:¥>cxref -idata main.c
```

-ni

[記述形式]

```
-ni
-nifile
-ni=file
```

- 省略時解釈

  なし

[機能説明]

- -ni の場合，インクルード・ファイルの情報を表示しません。
- -ni*file* の場合，実行結果に表示しないファイル名 *file* を指定します。

  file には，次のワイルド・カードを使用することができます。

?	任意の 1 文字
*	0 文字以上の任意の文字列

- -ni=*file* の場合，実行結果に表示しないファイル名を記述したファイル名 *file* を指定します。

[使用例]

- インクルード・ファイルの情報を表示しません。

```
C:¥>cxref -ni main.c
```

- ファイル名に“ r ”を含むファイルの情報を表示しません。

```
C:¥>cxref -ni*r* main.c
```

- ファイル名に“ e ”を含み，その後に 2 文字以上の文字があるファイルの情報を表示しません。

```
C:¥>cxref -ni*e??* main.c
```

- ファイル名が“ w ”で始まる 2 文字以上で末尾が“.h”のファイルの情報を表示しません。

```
C:¥>cxref -niw?*.h main.c
```

- ファイル noresult に記述されたファイル名の情報を表示しません。

```
C:¥>cxref -ni=noresult main.c
```

-o

[記述形式]

```
-o path
```

- 省略時解釈

  出力ファイルはカレント・パスに出力されます。

[機能説明]

- 出力ファイルのパス *path* を指定します。

[使用例]

- ファイルをフォルダ D:￥sample に出力します。

```
C:￥>cxref -o D:￥sample main.c
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>cxref @command
```

クロス・リファレンス

クロス・リファレンス用オプションには，次のものがあります。

- -x
- -xstd

-x

[記述形式]

```
-x[=file]
```

- 省略時解釈

  標準出力に出力します。

[機能説明]

- クロス・リファレンスをテキスト形式で，指定したファイルに出力します。
- “*=file*”を省略した場合，ファイル名は cxref になります。

[使用例]

- クロス・リファレンスをテキスト形式で，ファイル cxfile に出力します。

```
C:¥>cxref -x=cxfile main.c
```

-xstd

[記述形式]

```
-xstd
```

- 省略時解釈

  -xstd

[機能説明]

- クロス・リファレンスを標準出力に出力します（デフォルト）。

[使用例]

- クロス・リファレンスを標準出力に出力します。

```
C:¥>cxref -xstd main.c
```

タグ情報

タグ情報用オプションには，次のものがあります。

- -t
- -tstd

-t

[記述形式]

```
-t[=file]
```

- 省略時解釈

  なし

[機能説明]

- タグ情報をテキスト形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は ctags になります。

[使用例]

- タグ情報をテキスト形式で，ファイル tagfile に出力します。

```
C:¥>cxref -t=tagfile main.c
```

-tstd

[記述形式]

```
-tstd
```

- 省略時解釈

  なし

[機能説明]

- タグ情報を標準出力に出力します。

[使用例]

- タグ情報を標準出力に出力します。

```
C:¥>cxref -tstd main.c
```

コール・ツリー

コール・ツリー用オプションには，次のものがあります。

-c

[記述形式]

```
-c[=file]
```

- 省略時解釈

  なし

[機能説明]

- コール・ツリーをテキスト形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は ccalltre.lst になります。

[使用例]

- コール・ツリーをテキスト形式で，ファイル callfile.lst に出力します。

```
C:￥>cxref -c=callfile.lst main.c
```

-cc

[記述形式]

```
-cc[=file]
```

- 省略時解釈

  なし

[機能説明]

- コール・ツリーを CSV 形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は ccalltre.csv になります。

[使用例]

- コール・ツリーを CSV 形式で，ファイル callfile.csv に出力します。

```
C:￥>cxref -cc=callfile.csv main.c
```

-call

[記述形式]

```
-call[=file]
```

- 省略時解釈

  なし

[機能説明]

- コール・ツリーをテキスト形式と CSV 形式で，指定したファイルに出力します。
- ファイル名は *file*.lst，および *file*.csv になります。
- *file* に拡張子を付けて指定した場合，その拡張子は無視されます。
- “*=file*”を省略した場合，ファイル名は ccalltre.lst，および ccalltre.csv になります。

[使用例]

- コール・ツリーをテキスト形式と CSV 形式で，ファイル callfile.lst と callfile.csv に出力します。

```
C:¥>cxref -call=callfile main.c
```

-ce

[記述形式]

```
-cenum
```

- 省略時解釈

  -ce3

[機能説明]

- 出力の省略方法を指定します。
- *num* には，次のいずれかが指定可能です。

1	すべて出力する。
2	同一レベルのコール・ツリーの場合，省略する。
3	一度出力したら省略する。

[使用例]

- 同一レベルのコール・ツリーの場合，出力を省略します。

```
C:¥>cxref -call -ce2 main.c
```

-cf

[記述形式]

```
-cfstring
-cf=file
```

- 省略時解釈

  なし

[機能説明]

- コール・ツリーを出力する関数名を *string* に指定します。
- コール・ツリーを出力する関数名を記述した，テキスト・ファイル *file* を指定します。

[使用例]

- コール・ツリーを出力する関数名を func とします。

```
C:¥>cxref -call -cffunc main.c
```

-cl

[記述形式]

```
-cl num
```

- 省略時解釈

  -cl255

[機能説明]

- 出力レベルを指定します。*num* には，1 ～ 255 が指定可能です。

[使用例]

- 出力レベル 128 とします。

```
C:¥>cxref -call -cl128 main.c
```

-cp

[記述形式]

```
-cp
```

- 省略時解釈

  なし

[機能説明]

- 引数，戻り値を含めて出力します。

[使用例]

- 引数，戻り値を含めて出力します。

```
C:¥>cxref -call -cp main.c
```

-cr

[記述形式]

```
-cr
```

- 省略時解釈

  なし

[機能説明]

- 参照情報を含めて出力します。

[使用例]

- 参照情報を含めて出力します。

```
C:¥>cxref -call -cr main.c
```

-cs

[記述形式]

```
-cs
```

- 省略時解釈

  なし

[機能説明]

- ソース・ファイル名，記述開始行を含めて出力します。

[使用例]

- ソース・ファイル名，記述開始行を含めて出力します。

```
C:¥>cxref -call -cs main.c
```

-cstd

[記述形式]

```
-cstd
```

- 省略時解釈

  なし

[機能説明]

- テキスト形式のコール・ツリーを標準出力に出力します。

[使用例]

- テキスト形式のコール・ツリーを標準出力に出力します。

```
C:¥>cxref -cstd main.c
```

-ct

[記述形式]

```
-ct
```

- 省略時解釈

  なし

[機能説明]

- 先頭のツリーのみ出力します。

[使用例]

- 先頭のツリーのみ出力します。

```
C:¥>cxref -call -ct main.c
```

関数計量

関数計量用オプションには，次のものがあります。

- -m
- -mc
- -mall
- -ms
- -mstd

-m

[記述形式]

```
-m[=file]
```

- 省略時解釈

  なし

[機能説明]

- 関数計量をテキスト形式で，指定したファイル *file* に出力します。
- “=*file*”を省略した場合，ファイル名は cmeasure.lst になります。

[使用例]

- 関数計量をテキスト形式で，ファイル measurefile.lst に出力します。

```
C:¥>cxref -m=measurefile.lst main.c
```

-mc

[記述形式]

```
-mc[=file]
```

- 省略時解釈

  なし

[機能説明]

- 関数計量を CSV 形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は cmeasure.csv になります。

[使用例]

- 関数計量を CSV 形式で，ファイル measurefile.csv に出力します。

```
C:¥>cxref -mc=measurefile.csv main.c
```

-mall

[記述形式]

```
-mall[=file]
```

- 省略時解釈

  なし

[機能説明]

- 関数計量をテキスト形式と CSV 形式で，指定したファイルに出力します。
- ファイル名は *file*.lst，および *file*.csv になります。
- *file* に拡張子を付けて指定した場合，その拡張子は無視されます。
- “*=file*”を省略した場合，ファイル名は cmeasure.lst，および cmeasure.csv になります。

[使用例]

- 関数計量をテキスト形式と CSV 形式で，ファイル measurefile.lst と measurefile.csv に出力します。

```
C:¥>cxref -mall=measurefile main.c
```

-ms

[記述形式]

```
-ms[+|-]num
```

- 省略時解釈

  ソートされず，関数の出現順に出力されます。

[機能説明]

- 出力順序を指定します。*num* には，次のいずれかが指定可能です。

1	関数名のアルファベット順にソートして出力する。
2	ファイル名，関数名のアルファベット順にソートして出力する。
3	ソートしない。

- ‘+’ を指定した場合，昇順に出力します。‘-’ を指定した場合，降順に出力します。デフォルトは，降順です。

[使用例]

- 関数名のアルファベット順に降順でソートして出力します。

```
C:¥>cxref -ms1 main.c
```

-mstd

[記述形式]

```
-mstd
```

- 省略時解釈

  なし

[機能説明]

- テキスト形式の関数計量を標準出力に出力します。

[使用例]

- テキスト形式の関数計量を標準出力に出力します。

```
C:¥>cxref -mstd main.c
```

コール・データベース

コール・データベース用オプションには，次のものがあります。

- -b
- -bc
- -ball
- -bstd

-b

[記述形式]

```
-b[=file]
```

- 省略時解釈

  なし

[機能説明]

- コール・データベースをテキスト形式で，指定したファイル *file* に出力します。
- “=*file*”を省略した場合，ファイル名は cprofile.dat になります。

[使用例]

- コール・データベースをテキスト形式で，ファイル calldbfile.dat に出力します。

```
C:¥>cxref -b=calldbfile.dat main.c
```

-bc

[記述形式]

```
-bc[=file]
```

- 省略時解釈

  なし

[機能説明]

- コール・データベースを CSV 形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は cprofile.csv になります。

[使用例]

- コール・データベースを CSV 形式で，ファイル calldbfile.csv に出力します。

```
C:￥>cxref -bc=calldbfile.csv main.c
```

-ball

[記述形式]

```
-ball[=file]
```

- 省略時解釈

  なし

[機能説明]

- コール・データベースをテキスト形式と CSV 形式で，指定したファイルに出力します。
- ファイル名は *file*.dat，および *file*.csv になります。
- *file* に拡張子を付けて指定した場合，その拡張子は無視されます。
- “*=file*”を省略した場合，ファイル名は cprofile.dat，および cprofile.csv になります。

[使用例]

- コール・データベースをテキスト形式と CSV 形式で，ファイル calldbfile.dat と calldbfile.csv に出力します。

```
C:¥>cxref -ball=calldbfile main.c
```

-bstd

[記述形式]

```
-bstd
```

- 省略時解釈

  なし

[機能説明]

- テキスト形式のコール・データベースを標準出力に出力します。

[使用例]

- テキスト形式のコール・データベースを標準出力に出力します。

```
C:¥>cxref -bstd main.c
```

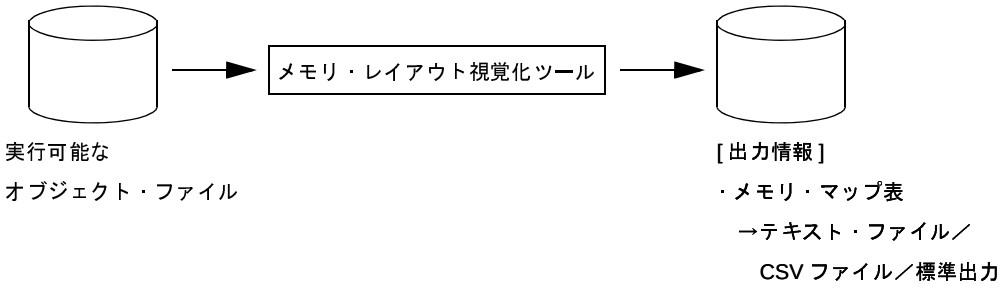
B. 11　メモリ・レイアウト視覚化ツール

メモリ・レイアウト視覚化ツールとは，作成したロード・モジュールから，変数のメモリ配置情報を読み出し，表示するユーティリティです。

CA850 に入っている“rammap”がメモリ・レイアウト視覚化ツールです。

変数のメモリ配置情報を，テキスト・ファイルや CSV ファイルとして出力します。

図 B—47　メモリ・レイアウト視覚化ツールにおける動作の流れ



B. 11. 1　入 出 力

（1）入力ファイル

メモリ・レイアウト視覚化ツールの入力ファイルは，ld850 が出力した実行可能なオブジェクト・ファイル注（.out ファイル）です。

注　再リンク可能なオブジェクト・ファイル，または romp850 が出力したファイル（.out ファイル）は除く。

（2）出力情報

メモリ・レイアウト視覚化ツールが出力する情報は，変数名，サイズ，メモリ配置を示すメモリ・マップです。

（a）メモリ・マップ表

変数名，サイズ，メモリ配置を示すメモリ・マップ表を出力します。

この情報についての詳細は，「3. 8　メモリ・レイアウト視覚化ツール」を参照してください。

B. 11. 2　操作方法

ここは，メモリ・レイアウト視覚化ツールの操作方法について説明します。

（1）コマンド入力による方法

コマンドは，コマンド・プロンプトで次のように入力します。

```
C:￥>rammap ［オプション］［ファイル名］
        ［ ］：　［ ］内は省略できます。
```

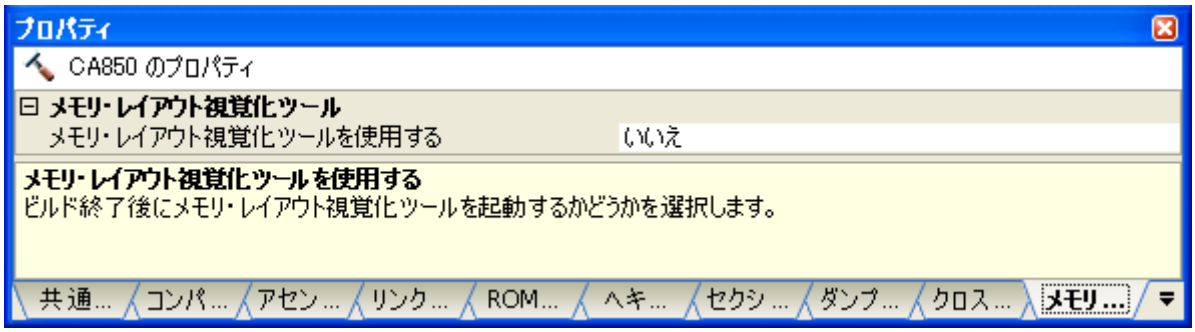
（2）CubeSuite+ でのオプション設定

CubeSuite+ からメモリ・レイアウト視覚化オプションを設定する方法について説明します。

CubeSuite+ のプロジェクト・ツリー パネルにおいて，ビルド・ツール・ノードを選択したのち，［表示］メニュー→［プロパティ］を選択すると，プロパティ パネルがオープンします。次に，［メモリ・レイアウト視覚化オプション］タブを選択します。

タブ上で各プロパティを設定することにより，対応するメモリ・レイアウト視覚化オプションを設定することができます。

図 B―48　プロパティ パネル：［メモリ・レイアウト視覚化オプション］タブ



B. 11. 3　オプション

ここでは，メモリ・レイアウト視覚化オプションについて説明します。

メモリ・レイアウト視覚化ツールの分類と説明を示します。

表 B—25　メモリ・レイアウト視覚化オプション

分類	オプション	説明
メモリ・レイアウト視覚化ツール	-V	メモリ・レイアウト視覚化ツールの版番号を標準出力に出力します。
	-all	すべての情報を，テキスト形式，および CSV 形式の各ファイルに出力します。
	-h	オプションの説明を出力します。
	-help	
	-m	メモリ・マップ表をテキスト形式で，指定したファイルに出力します。
	-mall	メモリ・マップ表をテキスト形式と CSV 形式で，指定したファイルに出力します。
	-mc	メモリ・マップ表を CSV 形式で，指定したファイルに出力します。
	-mr	メモリ・マップ表を出力する範囲を指定します。
	-mstd	テキスト形式のメモリ・マップ表を標準出力に出力します。
	-o	出力ファイルのパスを指定します。
	@	指定ファイルをコマンド・ファイルとして扱います。

メモリ・レイアウト視覚化ツール

メモリ・レイアウト視覚化ツールのオプションには，次のものがあります。

-V

[記述形式]

```
-V
```

- 省略時解釈

  なし

[機能説明]

- メモリ・レイアウト視覚化ツールの版番号を出力し，終了します。

[使用例]

- メモリ・レイアウト視覚化ツールの版番号を出力します。

```
C:¥>rammap -V
```

-all

[記述形式]

```
-all
```

- 省略時解釈

  テキスト形式のメモリ・マップ表を標準出力に出力します。

[機能説明]

- すべての情報を，テキスト形式，および CSV 形式の各ファイルに出力します。
- “-mall”を指定したのと同じです。

[使用例]

- すべての情報を，テキスト形式，および CSV 形式の各ファイルに出力します。

```
C:￥>rammap -all a.out
```

-h/-help

[記述形式]

```
-h
-help
```

- 省略時解釈

  なし

[機能説明]

- オプションの説明を出力し，終了します。

[使用例]

- メモリ・レイアウト視覚化ツールのオプションの説明を出力します。

```
C:¥>rammap -help
```

-m

[記述形式]

```
-m[=file]
```

- 省略時解釈

  -m

[機能説明]

- メモリ・マップ表をテキスト形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は rammap.txt になります。

[使用例]

- メモリ・マップ表をテキスト形式で，ファイル memmapfile.txt に出力します。

```
C:¥>rammap -m=memmapfile.txt a.out
```

-mall

[記述形式]

```
-mall[=file]
```

- 省略時解釈

  テキスト形式のメモリ・マップ表を標準出力に出力します。

[機能説明]

- メモリ・マップ表をテキスト形式と CSV 形式で，指定したファイルに出力します。
- ファイル名は *file*.txt，および *file*.csv になります。
- *file* に拡張子を付けて指定した場合，その拡張子は無視されます。
- “=*file*”を省略した場合，ファイル名は rammap.txt，および rammap.csv になります。

[使用例]

- メモリ・マップ表をテキスト形式と CSV 形式で，ファイル memmapfile.txt と memmapfile.csv に出力します。

```
C:¥>rammap -mall=memmapfile a.out
```

-mc

[記述形式]

```
-mc[=file]
```

- 省略時解釈

  テキスト形式のメモリ・マップ表を標準出力に出力します。

[機能説明]

- メモリ・マップ表を CSV 形式で，指定したファイル *file* に出力します。
- “*=file*”を省略した場合，ファイル名は rammap.csv になります。

[使用例]

- メモリ・マップ表を CSV 形式で，ファイル memmapfile.csv に出力します。

```
C:￥>rammap -mc=memmapfile.csv a.out
```

-mr

[記述形式]

```
-mrrange
```

- 省略時解釈

  オブジェクト中すべての範囲のメモリ・マップ表の対象とします。

[機能説明]

- メモリ・マップ表を出力する範囲を指定します。
- “-mr”と範囲の間に空白を入れないでください。
- アドレスには，8 進数，10 進数，16 進数が指定可能です。

8 進数指定形式	-mr0200000-0400000
10 進数指定形式	-mr65536-131072
16 進数指定形式	-mr0x10000-0x20000

- 複数の範囲を指定することができます。
- 複数指定には -mr オプションを複数指定する方法と，範囲ごとにカンマ（ , ）で区切る方法があります。
- 指定した範囲の重なりによって，次のようにみなされます。

例 1. a ～ b，c ～ d の 2 つの範囲を指定したものとみなされます。

a ---------- b　c ---------- d

2. a ～ d の 1 つの範囲を指定したものとみなされます。

a ---------- b
　　c ---------- d

3. a ～ d の 1 つの範囲を指定したものとみなされます。

a ---------- b
　　c ---------- d

4. a ～ b の 1 つの範囲を指定したものとみなされます。

a ---------- b
　　c ---------- d

注意 1. 実際のアドレス範囲は，16 バイトに整列されます。
開始アドレスは，指定した値を 16 バイトに丸めた値（0xfffffff0 との論理積）となります。終了アドレスは，指定した値を 16 バイトに丸め，0xF を加算した値となります。

-mr0x10000-0x20000	0x10000 ～ 0x2000f
-mr0x10004-	0x10000 ～ 0xffffffff
-mr-0x20005	0x0 ～ 0x2000f

2. 範囲指定が不正な場合，エラー・メッセージが出力され，処理が中断されます。

［使用例］

- メモリ・マップ表を出力する範囲を 0x10000 ～ 0x20000 とします。

```
C:¥>rammap a.out -mr0x10000-0x20000
```

- メモリ・マップ表を出力する開始アドレスを 0x10000 とします。この場合，終了アドレスは 0xffffffff となります。

```
C:¥>rammap a.out -mr0x10000-
```

- メモリ・マップ表を出力する終了アドレスを 0x20000 とします。この場合，開始アドレスは 0x0 となります。

```
C:¥>rammap a.out -mr-0x20000
```

- メモリ・マップ表を出力する範囲を 0x10000 ～ 0x20000 と 0x30000 ～ 0x40000 とします。

```
C:¥>rammap a.out -mr0x10000-0x20000 -mr0x30000-0x40000
```

または，

```
C:¥>rammap a.out -mr0x10000-0x20000,0x30000-0x40000
```

-mstd

[記述形式]

```
-mstd
```

- 省略時解釈

  -mstd

[機能説明]

- テキスト形式のメモリ・マップ表を標準出力に出力します。

[使用例]

- テキスト形式のメモリ・マップ表を標準出力に出力します。

```
C:¥>rammap -mstd a.out
```

-o

[記述形式]

```
-o path
```

- 省略時解釈

  カレント・パスに出力されます。

[機能説明]

- 出力ファイルのパス *path* を指定します。

[使用例]

- ファイルをフォルダ D:￥sample に出力します。

```
C:￥>rammap -mc -o D:￥sample a.out
```

@

[記述形式]

```
@cfile
```

- 省略時解釈

  コマンド・ファイルがないものとみなします。

[機能説明]

- *cfile* をコマンド・ファイルとして扱います。
- コマンド・ファイルとは，コマンドに対するオプションやファイル名をコマンド・ラインの引数として指定するのではなくファイルに記述して指定するものです。
- Windows 上では，コマンドに対するオプション指定の文字列の長さに制限があります。数多くのオプションを設定し，オプションを認識しきれない場合などに，コマンド・ファイルを作成し，このオプションを指定してください。
- コマンド・ファイルについての詳細は「(2) コマンド・ファイル」を参照してください。

[使用例]

- command をコマンド・ファイルとして扱います。

```
C:￥>rammap @command
```

付録C 索 引

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.04.01	―	初版発行

CubeSuite+ V1.00.00 ユーザーズマニュアル
V850 ビルド編

発行年月日　2011 年 4 月 1 日　Rev.1.00
発行　ルネサス エレクトロニクス株式会社
〒211-8668 神奈川県川崎市中原区下沼部 1753

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）　(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

CubeSuite+ V1.00.00

ルネサス エレクトロニクス株式会社

R20UT0557JJ0100