User's Manual

CubeSuite+ V1.00.00

統合開発環境

ユーザーズマニュアル　RL78,78K0R ビルド編

対象デバイス

RL78 ファミリ

78K0R 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，RL78 ファミリ，78K0R 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，RL78 ファミリ，78K0R 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。

第 1 章　概　　説
第 2 章　機　　能
第 3 章　ビルドの出力リスト
第 4 章　サンプル・プログラム
第 5 章　注意事項
付録 A　ウインドウ・リファレンス
付録 B　コマンド・リファレンス
付録 C　索　　引

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	このマニュアル	R20UT0556E
	V850 ビルド編	R20UT0557J	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。

この資料に記載されている会社名，製品名などは，各社の商標または登録商標です。

〔メ　　モ〕

〔メ　　モ〕

〔メ　　モ〕

目　　次

第１章　概　　説　…　12

1. 1　概　　要　…　12
1. 2　特　　長　…　14

第２章　機　　能　…　15

2. 1　概　　要　…　15
　2. 1. 1　ロード・モジュールを作成する　…　15
　2. 1. 2　ユーザ・ライブラリを作成する　…　17
2. 2　ビルド・ツールのバージョンを変更する　…　19
2. 3　ビルド対象ファイルを設定する　…　20
　2. 3. 1　スタートアップ・ルーチンを設定する　…　20
　2. 3. 2　プロジェクトにファイルを追加する　…　22
　2. 3. 3　プロジェクトからファイルを外す　…　27
　2. 3. 4　ファイルをビルド対象から外す　…　27
　2. 3. 5　ファイルをカテゴリに分類する　…　28
　2. 3. 6　ファイルの表示順を変更する　…　29
　2. 3. 7　ファイルの依存関係を更新する　…　30
2. 4　出力ファイルの種類を設定する　…　33
　2. 4. 1　出力ファイル名を変更する　…　33
　2. 4. 2　アセンブル・リストを出力する　…　35
　2. 4. 3　マップ情報を出力する　…　35
　2. 4. 4　シンボル情報を出力する　…　36
2. 5　コンパイル・オプションを設定する　…　38
　2. 5. 1　コード・サイズを優先した最適化を行う　…　39
　2. 5. 2　実行速度を優先した最適化を行う　…　39
　2. 5. 3　インクルード・パスを追加する　…　40
　2. 5. 4　定義マクロを設定する　…　41
　2. 5. 5　C++ のコメントを有効にする　…　42
　2. 5. 6　浮動小数点対応の標準入出力関数を使用する　…　43
　2. 5. 7　演算器の使用設定を変更する　…　43
2. 6　アセンブル・オプションを設定する　…　44
　2. 6. 1　インクルード・パスを追加する　…　44
　2. 6. 2　定義マクロを設定する　…　46
2. 7　リンク・オプションを設定する　…　47
　2. 7. 1　ユーザ・ライブラリを追加する　…　48
2. 8　ROM 化プロセス・オプションを設定する　…　50
2. 9　オブジェクト・コンバート・オプションを設定する　…　51
　2. 9. 1　ヘキサ・ファイルの出力を設定する　…　52
2. 10　ライブラリ生成オプションを設定する　…　53
　2. 10. 1　ライブラリ・ファイルの出力を設定する　…　53
2. 11　変数／関数配置オプションを設定する　…　54

2. 11. 1　変数や関数を効率よく配置する　…　54
2. 11. 2　ROM/RAM 使用量を表示する　…　59
2. 12　個別にビルド・オプションを設定する　…　60
2. 12. 1　プロジェクト単位でビルド・オプションを設定する　…　60
2. 12. 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　…　60
2. 13　オンチップ・デバッガを使用するための準備をする　…　63
2. 14　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　…　65
2. 14. 1　ビルド対象ファイルを準備する　…　65
2. 14. 2　ブート領域側のプロジェクトを設定する　…　65
2. 14. 3　フラッシュ領域側のプロジェクトを設定する　…　69
2. 15　ビルドの設定をする　…　73
2. 15. 1　ファイルのリンク順を設定する　…　73
2. 15. 2　サブプロジェクトのビルド順を変更する　…　74
2. 15. 3　ビルド・オプションを一覧表示する　…　74
2. 15. 4　ビルド対象プロジェクトを変更する　…　75
2. 15. 5　ビルド・モードを追加する　…　76
2. 15. 6　ビルド・モードを変更する　…　78
2. 15. 7　ビルド・モードを削除する　…　80
2. 15. 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　…　81
2. 16　ビルドを実行する　…　82
2. 16. 1　更新ファイルのビルドを実行する　…　85
2. 16. 2　すべてのファイルのビルドを実行する　…　86
2. 16. 3　他の処理と平行してビルドを実行する　…　86
2. 16. 4　ビルド・モードを一括してビルドを実行する　…　88
2. 16. 5　ファイル単位でコンパイル／アセンブルする　…　89
2. 16. 6　ビルドの実行を中止する　…　90
2. 16. 7　ビルド結果をファイルに保存する　…　91
2. 16. 8　中間ファイル，生成ファイルを削除する　…　91
2. 17　スタックを見積もる　…　93
2. 17. 1　起動と終了　…　93
2. 17. 2　呼び出し関係を確認する　…　94
2. 17. 3　スタック情報を確認する　…　95
2. 17. 4　不明関数を確認する　…　96
2. 17. 5　スタック・サイズを変更する　…　97

第 3 章　ビルドの出力リスト　…　99

3. 1　C コンパイラ　…　99
3. 1. 1　アセンブラ・ソース・ファイル　…　99
3. 1. 2　エラー・リスト・ファイル　…　103
3. 1. 3　プリプロセス・リスト・ファイル　…　105
3. 1. 4　クロスリファレンス・リスト・ファイル　…　107
3. 2　アセンブラ　…　110
3. 2. 1　アセンブル・リスト・ファイルのヘッダ　…　110
3. 2. 2　アセンブル・リスト　…　111
3. 2. 3　シンボル・リスト　…　113
3. 2. 4　クロスリファレンス・リスト　…　114
3. 2. 5　エラー・リスト　…　116
3. 3　リ ン カ　…　117

3. 3. 1　リンク・リスト・ファイルのヘッダ　…　117
3. 3. 2　マップ・リスト　…　118
3. 3. 3　パブリック・シンボル・リスト　…　120
3. 3. 4　ローカル・シンボル・リスト　…　121
3. 3. 5　エラー・リスト　…　122
3. 4　ROM 化プロセッサ　…　123
3. 4. 1　リンク・マップ・ファイルのヘッダ　…　123
3. 4. 2　マップ・リスト　…　124
3. 4. 3　パブリック・シンボル・リスト　…　126
3. 4. 4　ローカル・シンボル・リスト　…　127
3. 4. 5　エラー・リスト　…　129
3. 5　オブジェクト・コンバータ　…　130
3. 5. 1　エラー・リスト　…　130
3. 6　ライブラリアン　…　131
3. 6. 1　ライブラリ情報出力リスト　…　131
3. 7　リスト・コンバータ　…　132
3. 7. 1　アブソリュート・アセンブル・リスト　…　132
3. 7. 2　エラー・リスト　…　132
3. 8　変数／関数情報ファイル生成ツール　…　133
3. 8. 1　変数／関数情報ファイル　…　133

第 4 章　サンプル・プログラム　…　136

4. 1　C コンパイラ　…　136
4. 1. 1　C ソース・ファイル　…　136
4. 2　アセンブラ　…　138
4. 2. 1　k0rmain.asm　…　138
4. 2. 2　k0rsub.asm　…　139

第 5 章　注意事項　…　140

付録 A　ウインドウ・リファレンス　…　147

A. 1　説　　明　…　147

付録 B　コマンド・リファレンス　…　342

B. 1　C コンパイラ　…　342
B. 1. 1　入出力ファイル　…　343
B. 1. 2　機　　能　…　344
B. 1. 3　操作方法　…　347
B. 1. 4　オプション　…　350
B. 2　アセンブラ　…　408
B. 2. 1　入出力ファイル　…　408
B. 2. 2　機　　能　…　409
B. 2. 3　操作方法　…　409
B. 2. 4　オプション　…　413
B. 3　リ ン カ　…　460

B. 3. 1　入出力ファイル　…　460
B. 3. 2　機　　能　…　461
B. 3. 3　操作方法　…　462
B. 3. 4　オプション　…　465
B. 3. 5　ブートーフラッシュ再リンク機能　…　517
B. 4　ROM 化プロセッサ　…　532
B. 4. 1　入出力ファイル　…　532
B. 4. 2　機　　能　…　533
B. 4. 3　操作方法　…　534
B. 4. 4　オプション　…　539
B. 5　オブジェクト・コンバータ　…　565
B. 5. 1　入出力ファイル　…　565
B. 5. 2　機　　能　…　566
B. 5. 3　操作方法　…　579
B. 5. 4　オプション　…　583
B. 6　ライブラリアン　…　603
B. 6. 1　入出力ファイル　…　603
B. 6. 2　機　　能　…　604
B. 6. 3　操作方法　…　605
B. 6. 4　オプション　…　609
B. 6. 5　サブコマンド　…　617
B. 7　リスト・コンバータ　…　628
B. 7. 1　入出力ファイル　…　629
B. 7. 2　機　　能　…　629
B. 7. 3　操作方法　…　632
B. 7. 4　オプション　…　634
B. 8　変数／関数情報ファイル生成ツール　…　644
B. 8. 1　入出力ファイル　…　644
B. 8. 2　機　　能　…　645
B. 8. 3　変数／関数情報　…　646
B. 8. 4　操作方法　…　650
B. 8. 5　オプション　…　653

付録 C　索　　引　…　661

第 1 章　概　　説

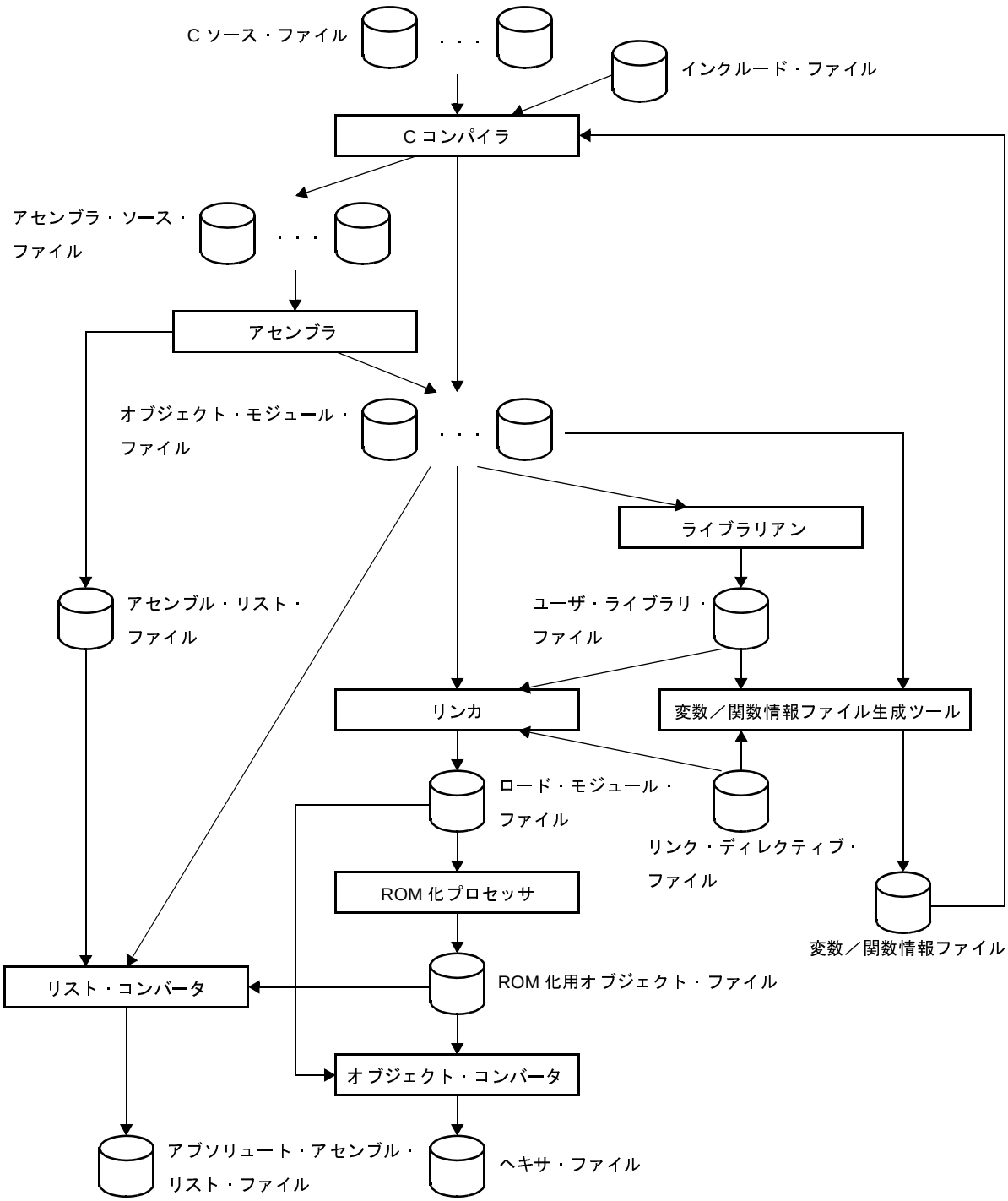
この章では，ビルド・ツールの概要について説明します。

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・ファイルからロード・モジュール・ファイル，ヘキサ・ファイル，ライブラリ・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



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。

- ROM 化機能

ROM 化とは，初期値あり外部変数などの初期値を ROM に配置しておき，システム実行時に RAM にコピーする処理です。

CA78K0R は，プログラムの ROM 化処理付きのスタートアップ・ルーチンを提供しているので，スタートアップ時の ROM 化処理などを記述する手間を省くことができます。

備考 ROM 化機能についての詳細は，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

- マクロ機能

アセンブラ・ソース・ファイル中で同じ命令群を何回も記述する場合，該当命令群を 1 つのマクロ名で定義することができます。

備考 マクロ機能についての詳細は，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

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. 15　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. 15. 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. 15. 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. 16 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. 16. 1 更新ファイルのビルドを実行する」参照）
- リビルド（「2. 16. 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. 16. 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. 16. 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。

注意 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・ファイルについて，以下のような制御を行います。
なお，品種指定が異なる入力ファイル（デバイス共通オブジェクトを出力する場合を除く）に対しても，リンク不可（エラー）とします。

		入力			
		78K0R/RL78 拡張命令非搭載品	RL78 拡張命令搭載品	78K0R/RL78 拡張命令非搭載品 デバイス共通オブジェクト出力	RL78 拡張命令搭載品 デバイス共通オブジェクト出力
出力	78K0R/RL78 拡張命令非搭載品	リンク可	リンク不可（エラー）	リンク可	リンク不可（エラー）
	RL78 拡張命令搭載品	リンク可	リンク可	リンク可	リンク可

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. 15 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. 15. 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，ライブラリアン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 ライブラリ生成オプションを設定する」参照）。

(5) ビルドの実行

ビルドを実行します（「2. 16 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. 16. 1 更新ファイルのビルドを実行する」参照）
- リビルド（「2. 16. 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. 16. 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. 16. 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。

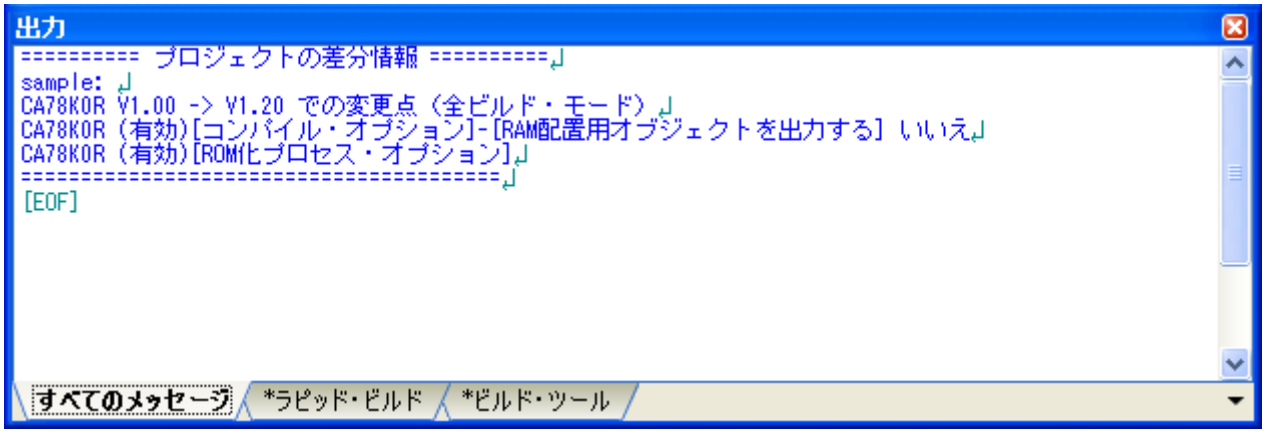
3.　コンパイラ・パッケージによってオプションに変更がある場合は，選択したバージョンにあわせて，ビルド・ツールの各プロパティの表示を切り替えます。

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。

図 2—2　オプションの変更情報の出力イメージ



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［コンパイル・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，［スタートアップ］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（通常）］／［はい（ブート領域用）］／［はい（フラッシュ領域用）］を選択してください。

図 2—3　［標準のスタートアップを使用する］プロパティ

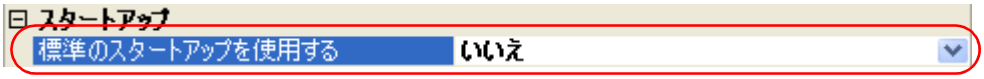
［使用する標準スタートアップ・ルーチン］プロパティに，使用する標準スタートアップ・ルーチンのオブジェクト・ファイル名が表示されます。

(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［コンパイル・オプション］タブを選択します。

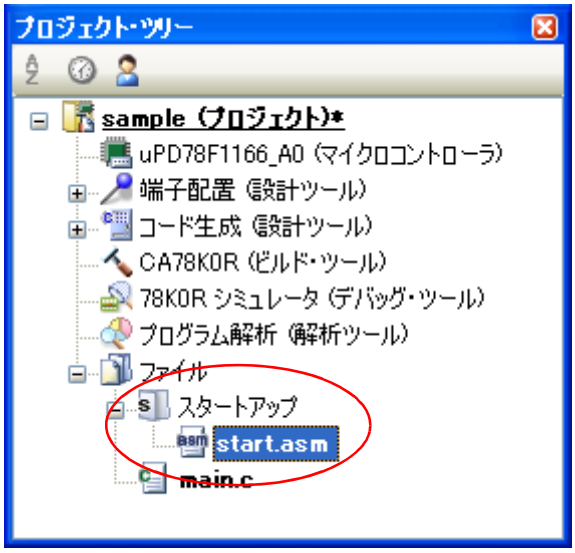
標準以外のスタートアップ・ルーチンを使用するには，［スタートアップ］カテゴリの［標準のスタートアップを使用する］プロパティで［いいえ］を選択してください（デフォルトでは，［はい（通常）］が選択されています）。

図 2—4　［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. 2　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—5 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・ルーチンの作成方法については，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

2. 3. 2　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

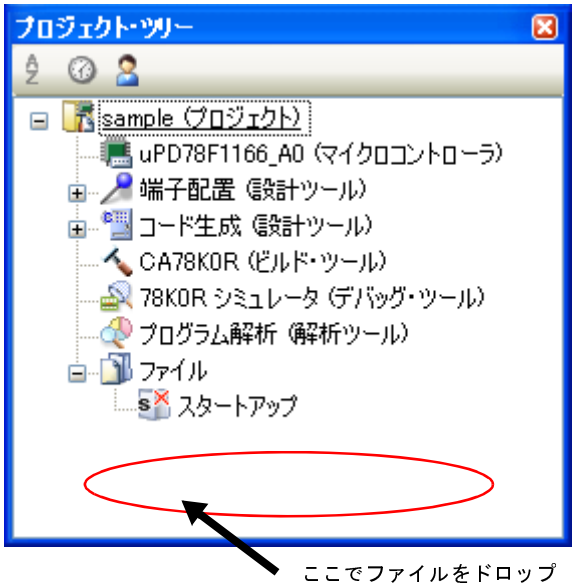
(1) 既存のファイルを追加する場合

(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—6　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。

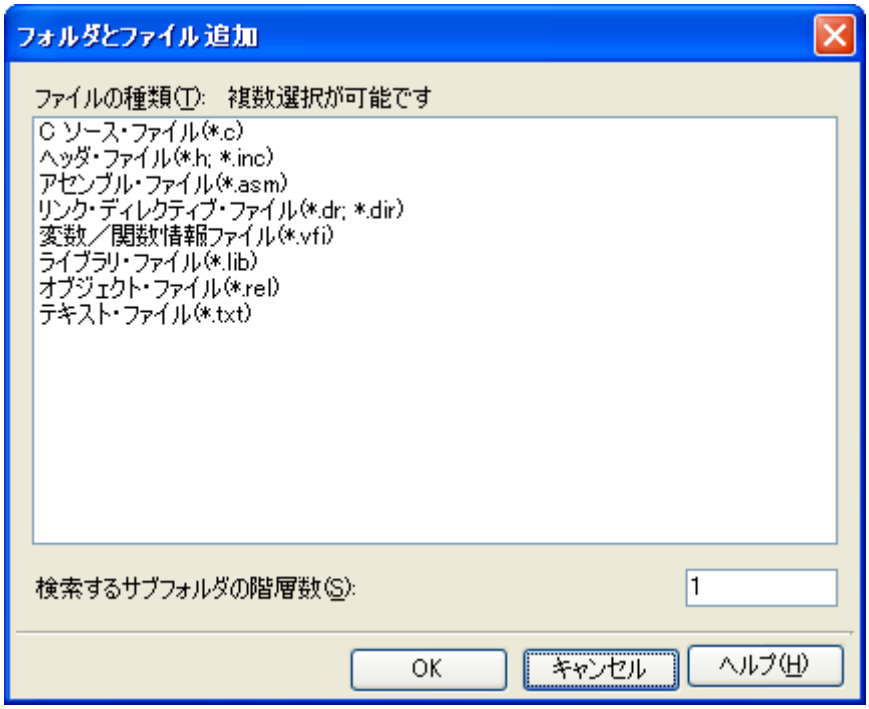
(b) フォルダ単位で追加する

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—7 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。

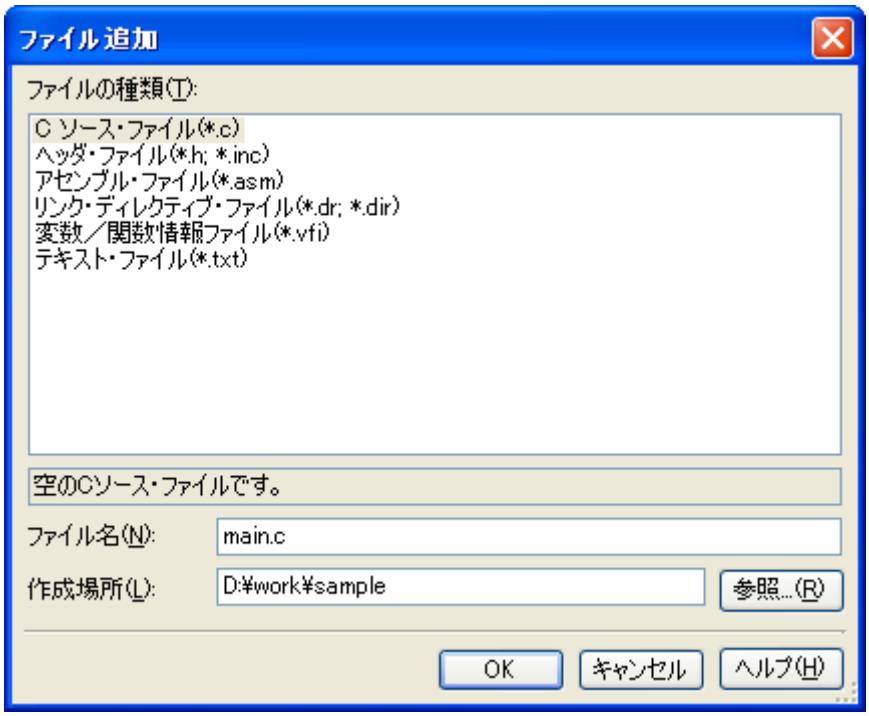
フォルダの追加先はファイル・ノード以下となります。
なお，フォルダはプロジェクト・ツリーではカテゴリとなります。

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. 5 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—8 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—9 プロジェクト・ツリー パネル（ファイル main.c 追加後）

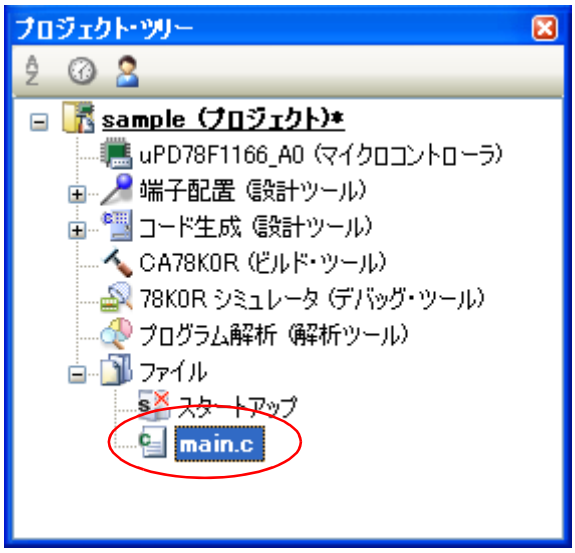
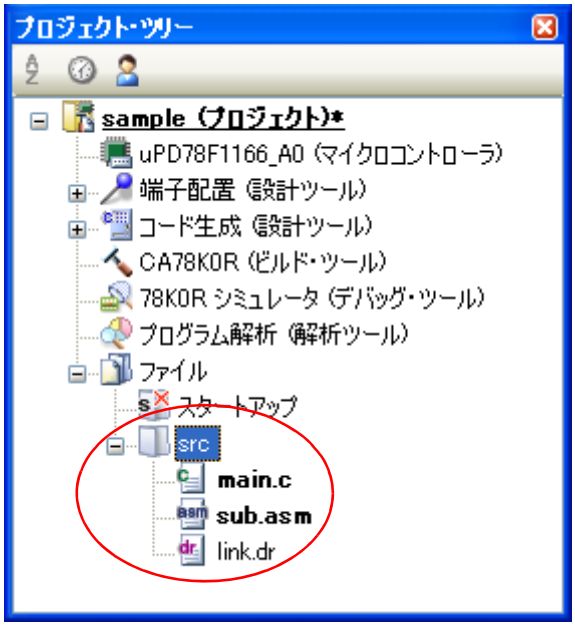


図 2—10　プロジェクト・ツリー パネル（フォルダ src 追加後）



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. 6　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.rel 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. 12. 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。
ただし，ソース・ファイルは 1000 個まで追加可能です。

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（.asm）
- 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）
- 変数／関数情報ファイル（.vfi）
- マップ・ファイル（.map）
- シンボル・テーブル・ファイル（.sym）
- ヘキサ・ファイル（.hex，.hxb，.hxf）
- テキスト・ファイル（.txt）

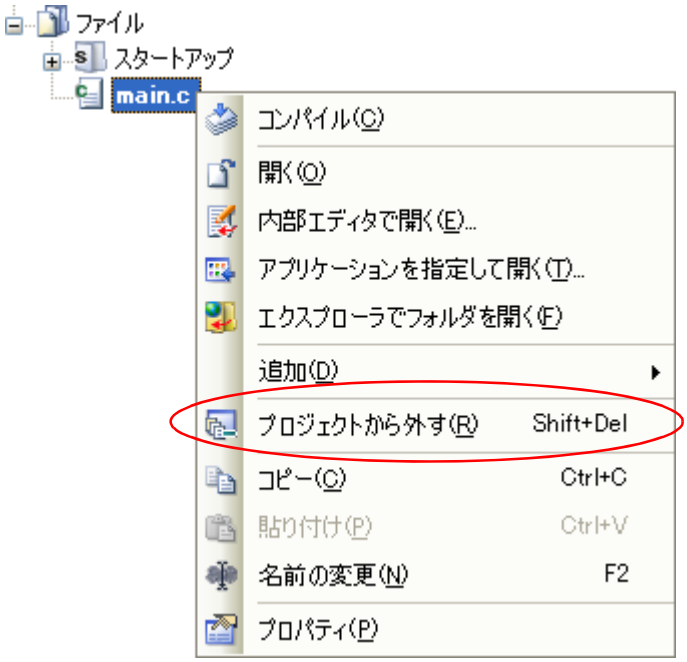
備考 1. 以下のいずれかの方法により，上記以外のファイルもエディタ パネルでオープンすることができます。
- ファイルをドラッグし，エディタ パネルにドロップする。
- ファイルを選択し，コンテキスト・メニューの［内部エディタで開く ...］を選択する。

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. 3 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—11 ［プロジェクトから外す］項目



2. 3. 4 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—12 ［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，変数／関数情報ファイル，オブジェクト・ファイル，ライブラリ・ファイルです。

2. 3. 5 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—13 ［新しいカテゴリを追加］項目（ファイル・ノードの場合）

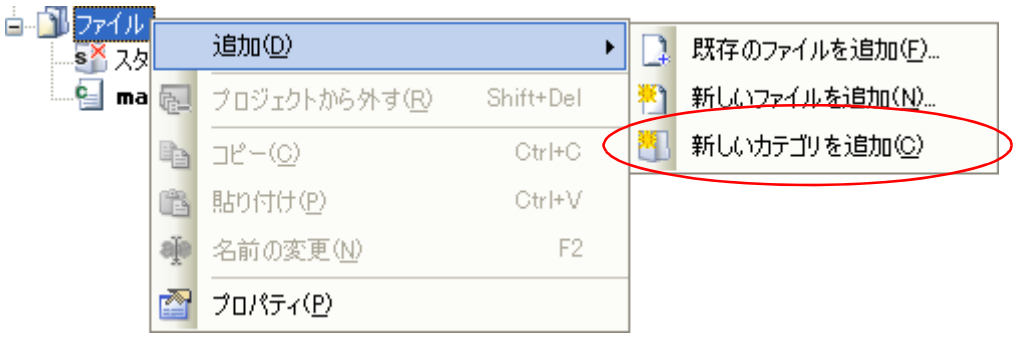
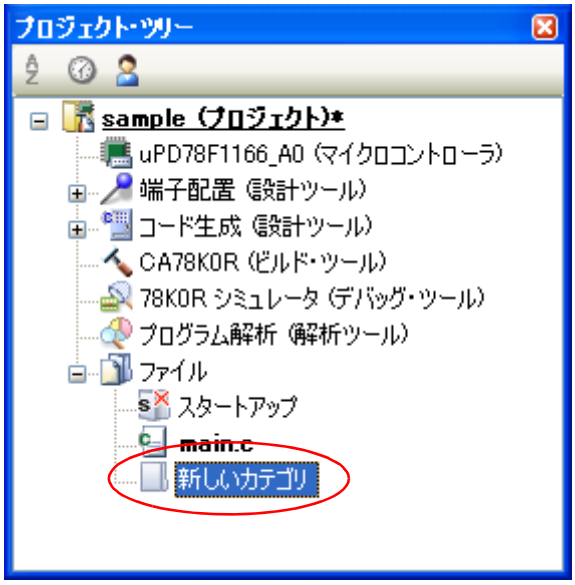


図 2—14 プロジェクト・ツリー パネル（カテゴリ・ノード追加後）



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。

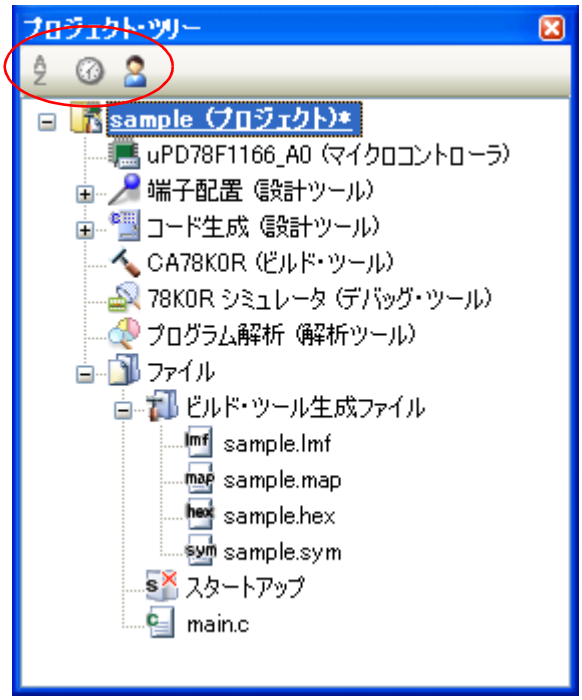
3. カテゴリのネスト数の上限は 20 です。

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. 6 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—15 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. 7 ファイルの依存関係を更新する

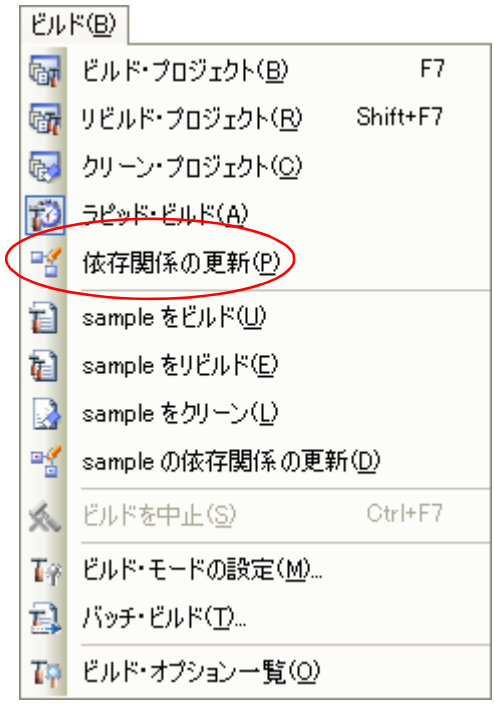
コンパイル・オプションの設定，アセンブル・オプションの設定で，ファイルの依存関係に影響する変更（インクルード・ファイルのパスの変更，C ソース・ファイル，およびアセンブラ・ソース・ファイル中にヘッダ・ファイルのインクルード文を追加など）を行った場合は，該当ファイルの依存関係を更新する必要があります。

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。

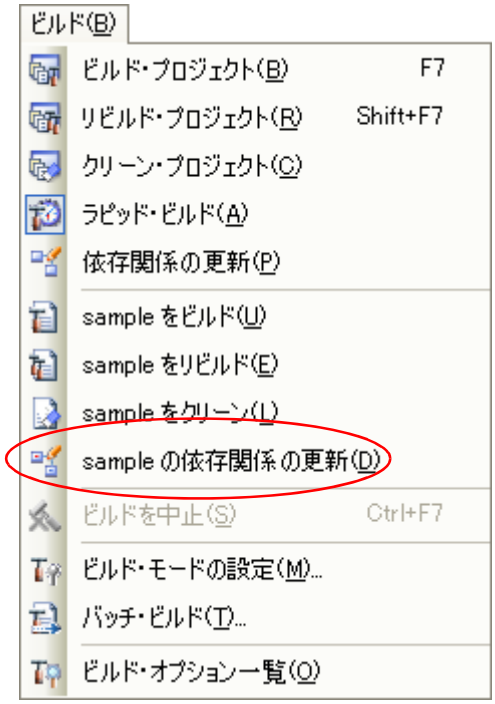
図 2—16 ［依存関係の更新］項目



(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—17 ［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）。

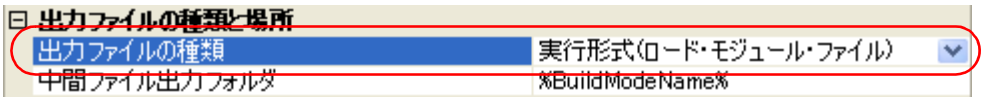
```
/* comment */   #include    "header6.h"     /* 依存関係なしと判断する */

/*
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—18　［出力ファイルの種類］プロパティ



(1)［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。

(2)［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。

［オブジェクト・コンバート・オプション］タブの［ヘキサ・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・ツールが出力するロード・モジュール・ファイル，ヘキサ・ファイル，ライブラリ・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。

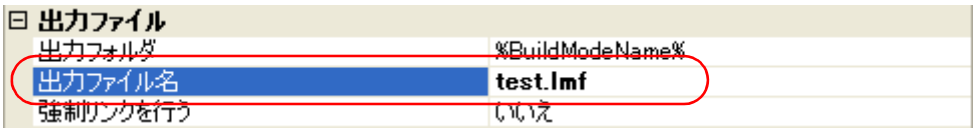
ロード・モジュール・ファイル名　：%ProjectName%.lmf
ヘキサ・ファイル名　：%ProjectName%.hex
ライブラリ・ファイル名　：%ProjectName%.lib

これらのファイル名の変更方法を，以下に示します。

(1) 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—19　［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）

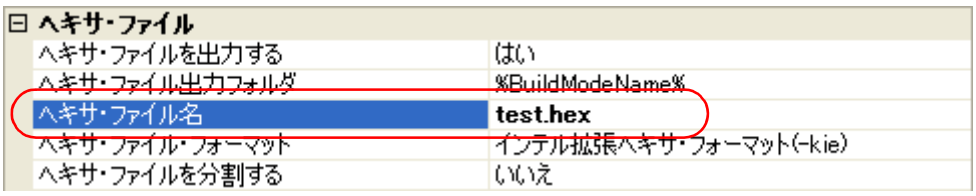


備考　［共通オプション］タブの［よく使うオプション（リンカ）］カテゴリの［出力ファイル名］プロパティでも，同様に変更することができます。

(2) 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［オブジェクト・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］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—20　［ヘキサ・ファイル名］プロパティ



注意　［ヘキサ・ファイルを分割する］プロパティで［はい (-zf)］を選択した場合，ヘキサ・ファイルは .hxb，および .hxf に分割して出力されます。
詳細については，「B. 5. 2　機　　能」を参照してください。

備考　［共通オプション］タブの［よく使うオプション（オブジェクト・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。

(3) ライブラリ・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［ライブラリ生成オプション］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。

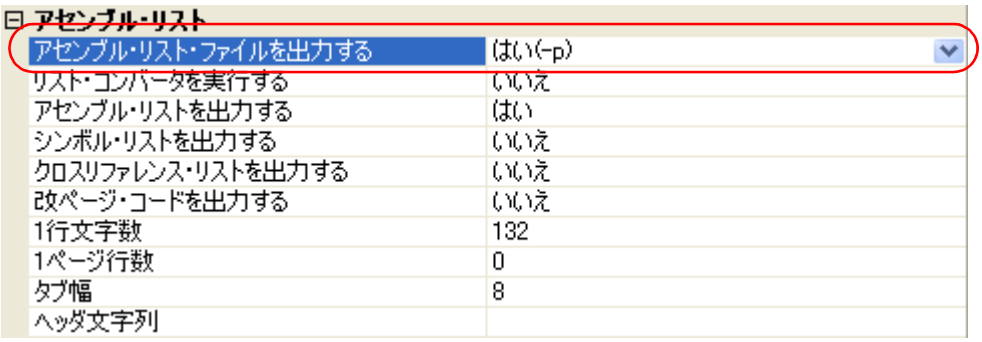
図 2—21　［出力ファイル名］プロパティ（ライブラリ・ファイルの場合）

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 (-p)］（デフォルト）を選択してください。

図 2—22 ［アセンブル・リスト・ファイルを出力する］プロパティ



備考 1. アセンブル・リストについては，「3. 2. 2 アセンブル・リスト」を参照してください。

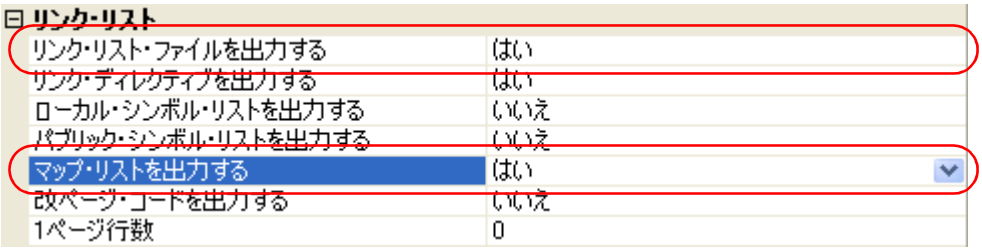
2. オブジェクト・モジュール・ファイルの出力のみが目的でアセンブルする場合などに［アセンブル・リスト・ファイルを出力する］プロパティで［いいえ (-np)］を選択すると，アセンブル時間を短縮することができます。

2. 4. 3 マップ情報を出力する

マップ情報（セグメントの配置に関する情報）はリンク・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［リンク・オプション］タブを選択します。リンク・リスト・ファイルの出力の設定は，［リンク・リスト］カテゴリで行います。

図 2—23 ［リンク・リスト・ファイルを出力する］，および［マップ・リストを出力する］プロパティ



［リンク・リスト・ファイルを出力する］プロパティで［はい］（デフォルト）を選択すると，［マップ・リストを出力する］プロパティが表示されます。リンク・リスト・ファイルにマップ情報を出力するには，［はい］を選択してください（デフォルト）。

備考　マップ情報については，「3. 4. 2　マップ・リスト」を参照してください。

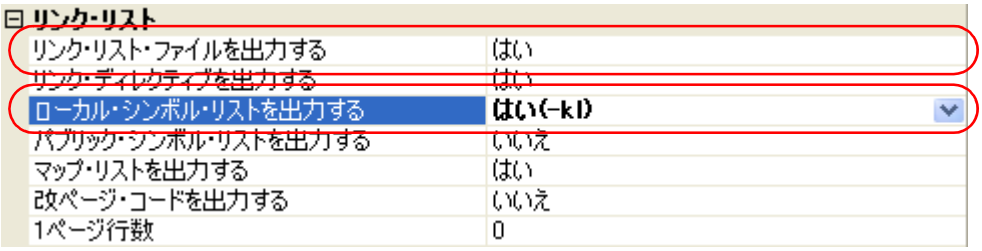
2. 4. 4　シンボル情報を出力する

入力モジュール内で定義されているシンボル情報（ローカル・シンボル，パブリック・シンボル）はリンク・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［リンク・オプション］タブを選択します。

シンボル情報の出力の設定は，［リンク・リスト］カテゴリで行います。

(1) ローカル・シンボル・リストを出力する場合

図 2—24　［リンク・リスト・ファイルを出力する］，および［ローカル・シンボル・リストを出力する］プロパティ

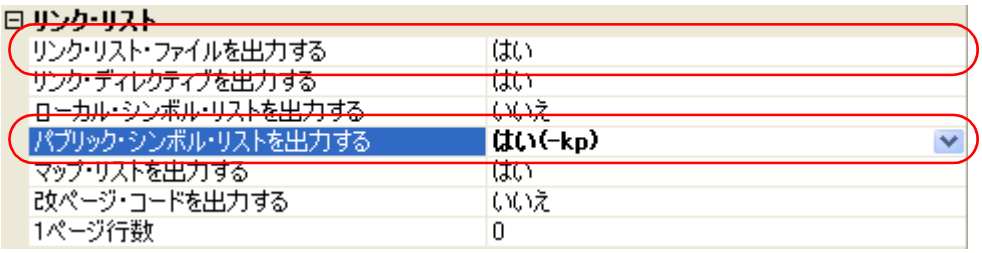


［リンク・リスト・ファイルを出力する］プロパティで［はい］（デフォルト）を選択すると，［ローカル・シンボル・リストを出力する］プロパティが表示されます。リンク・リスト・ファイルにローカル・シンボル・リストを出力するには，［はい (-kl)］を選択してください（デフォルトでは，［いいえ］が選択されています）。

備考　ローカル・シンボル・リストについては，「3. 4. 4　ローカル・シンボル・リスト」を参照してください。

(2) パブリック・シンボル・リストを出力する場合

図 2—25　［リンク・リスト・ファイルを出力する］，および［パブリック・シンボル・リストを出力する］プロパティ



［リンク・リスト・ファイルを出力する］プロパティで［はい］（デフォルト）を選択すると，［パブリック・シンボル・リストを出力する］プロパティが表示されます。リンク・リスト・ファイルにパブリック・シンボル・リストを出力するには，［はい (-kp)］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）。

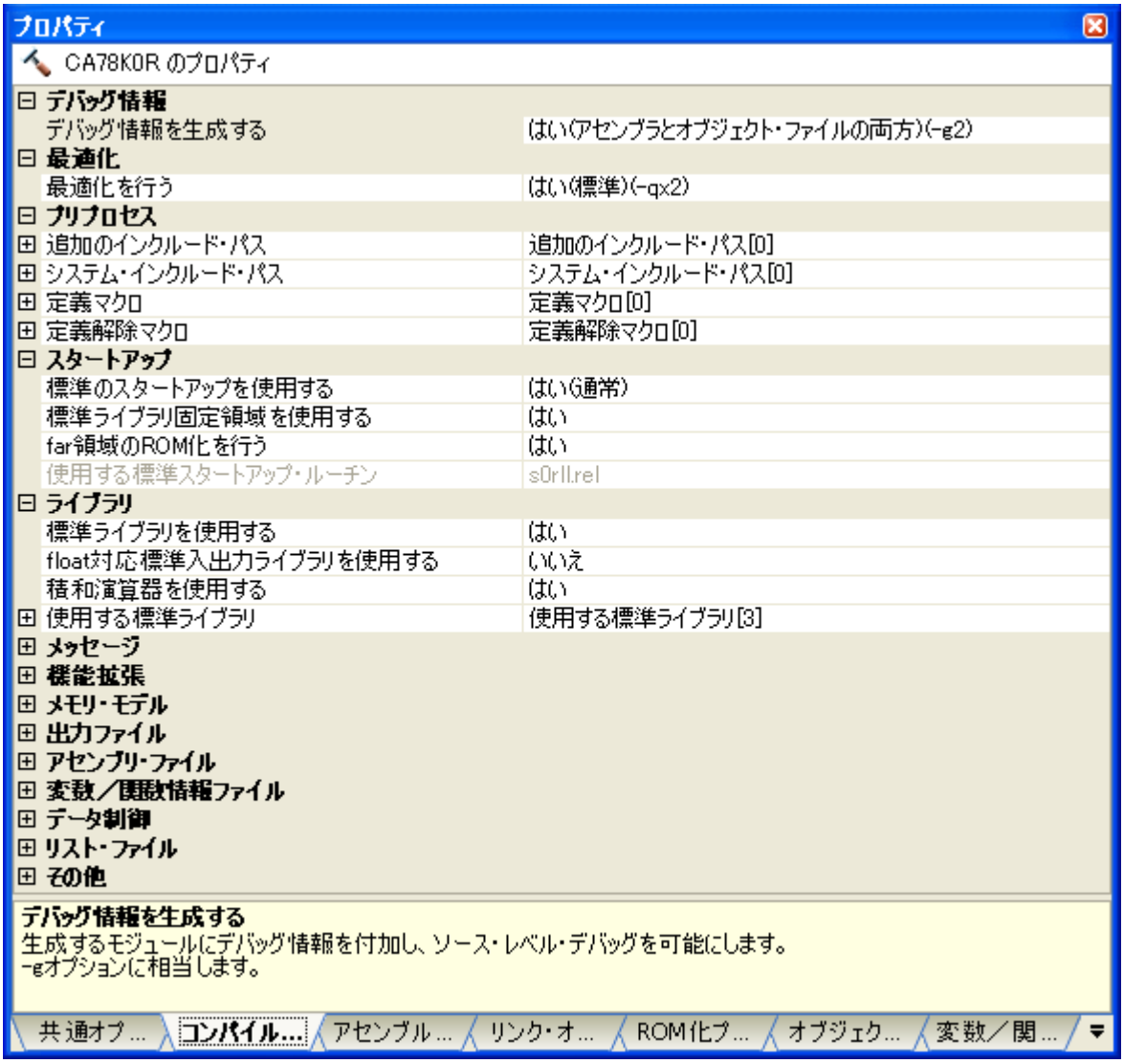
備考 パブリック・シンボル・リストについては,「3. 4. 3 パブリック・シンボル・リスト」を参照してください。

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—26　プロパティ パネル：［コンパイル・オプション］タブ



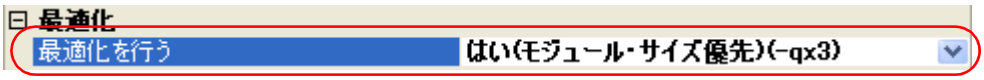
備考　よく使うオプションについては，［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリにまとめられています。

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［最適化を行う］プロパティで［はい (モジュール・サイズ優先)(-qx3)］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—27 ［最適化を行う］プロパティ（コード・サイズ優先の場合）



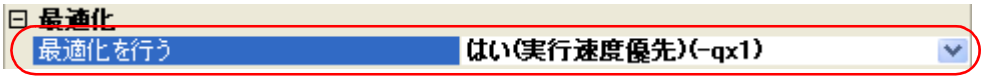
備考 ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［最適化を行う］プロパティでも，同様に設定することができます。

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［最適化を行う］プロパティで［はい (実行速度優先)(-qx1)］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—28 ［最適化を行う］プロパティ（実行速度優先の場合）



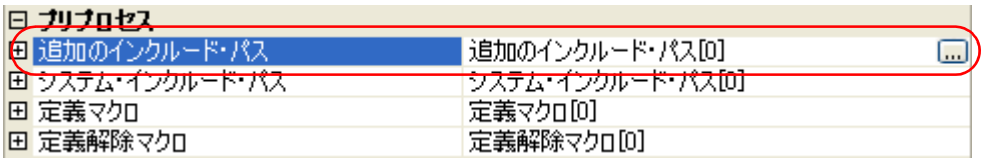
備考 ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［最適化を行う］プロパティでも，同様に設定することができます。

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。

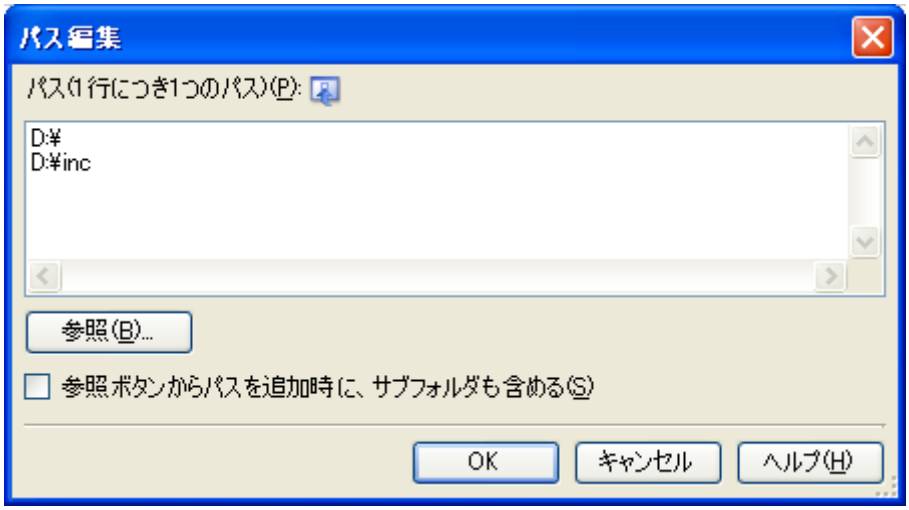
インクルード・パスの設定は，［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで行います。

図 2—29 ［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—30 パス編集 ダイアログ

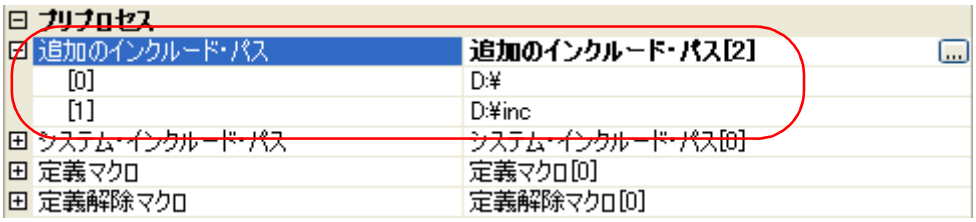


［パス (1 行につき 1 つのパス)］にインクルード・パスを 1 行に 1 つずつ入力します。1 行に 259 文字まで，64 行まで指定可能です。

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—31 ［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。

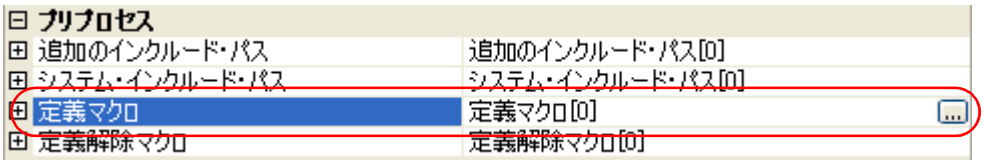
備考 ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［追加のインクルード・パス］プロパティでも，同様に設定することができます。

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。

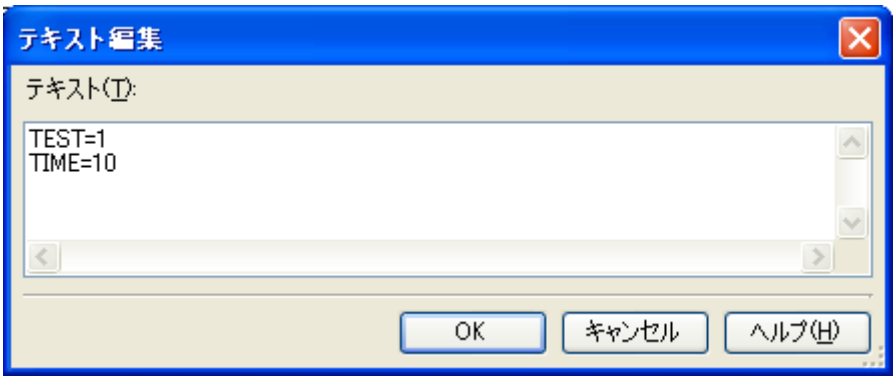
定義マクロの設定は，［プリプロセス］カテゴリの［定義マクロ］プロパティで行います。

図 2—32 ［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。

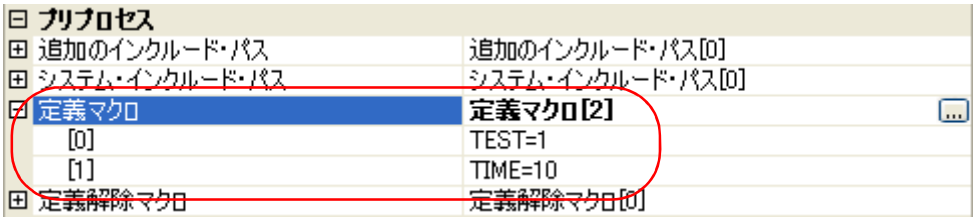
図 2—33 テキスト編集 ダイアログ



［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—34 ［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。

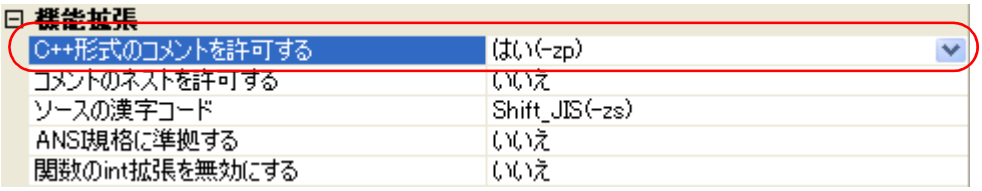
備考 ［共通オプション］タブの［よく使うオプション（コンパイラ）］カテゴリの［定義マクロ］プロパティでも，同様に設定することができます。

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，［機能拡張］カテゴリの［C++ 形式のコメントを許可する］プロパティで［はい (-zp)］を選択してください（デフォルト）。

図 2—35 ［C++ 形式のコメントを許可する］プロパティ



2. 5. 6 浮動小数点対応の標準入出力関数を使用する

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。

［ライブラリ］カテゴリの［標準ライブラリを使用する］プロパティで［はい］を選択すると，［float 対応標準入出力ライブラリを使用する］プロパティが表示されます。浮動小数点対応の標準入出力関数（sprintf，sscanf，printf，vprintf，vsprintf）を使用するには，［はい］を選択してください。

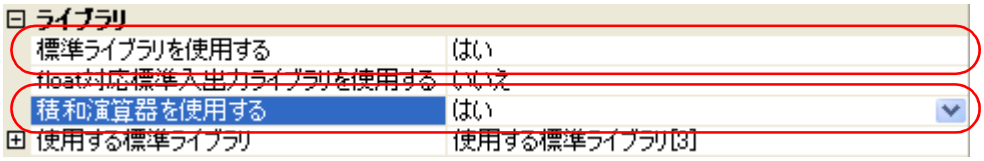
図 2—36 ［標準ライブラリを使用する］，および［float 対応標準入出力ライブラリを使用する］プロパティ

2. 5. 7 演算器の使用設定を変更する

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［コンパイル・オプション］タブを選択します。

［ライブラリ］カテゴリの［標準ライブラリを使用する］プロパティで［はい］を選択すると，［乗除算器を使用する］プロパティ／［乗算器を使用する］プロパティ／［積和演算器を使用する］プロパティが表示されます。乗除算器／乗算器／積和演算器に対応した標準ライブラリを使用する場合は［はい］（デフォルト），使用しない場合は［いいえ］を選択してください。

図 2—37 ［標準ライブラリを使用する］，および［積和演算器を使用する］プロパティ

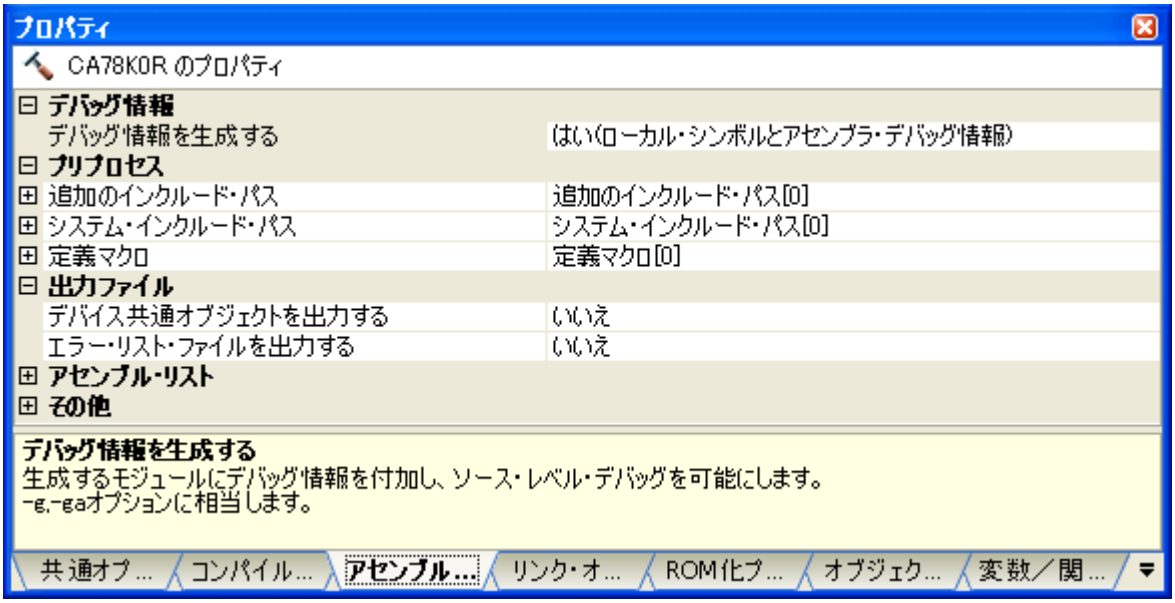


2. 6　アセンブル・オプションを設定する

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—38　プロパティ パネル：［アセンブル・オプション］タブ



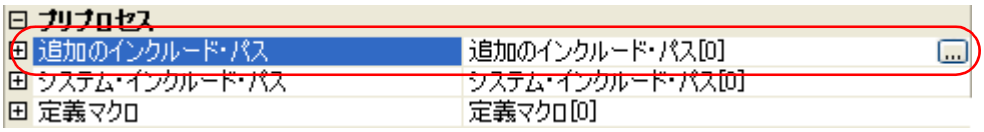
備考　よく使うオプションについては，［共通オプション］タブの［よく使うオプション（アセンブラ）］カテゴリにまとめられています。

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。

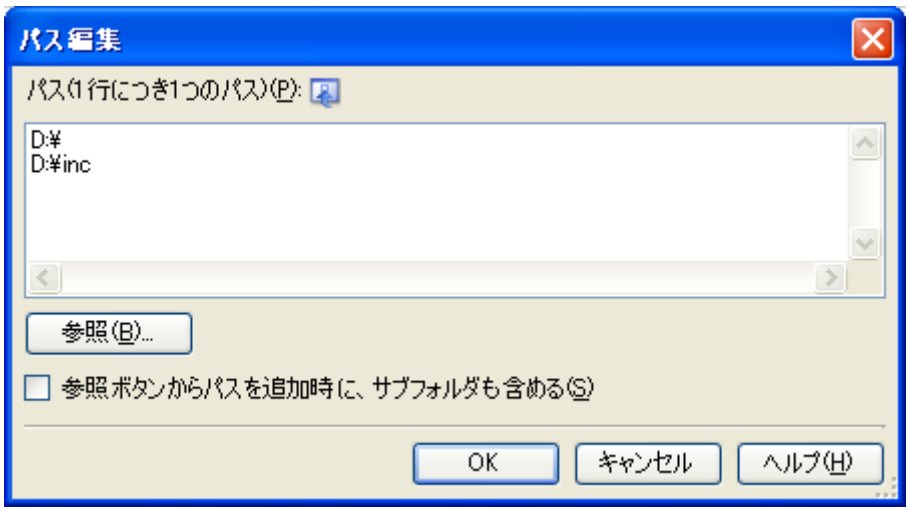
インクルード・パスの設定は，［プリプロセス］カテゴリの［追加のインクルード・パス］プロパティで行います。

図 2—39　［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—40 パス編集 ダイアログ

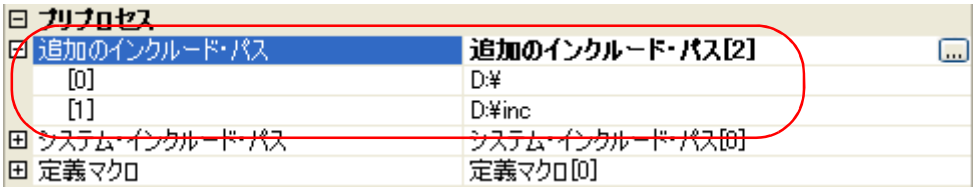


［パス (1 行につき 1 つのパス)］にインクルード・パスを 1 行に 1 つずつ入力します。1 行に 259 文字まで，64 行まで指定可能です。

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—41 ［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。

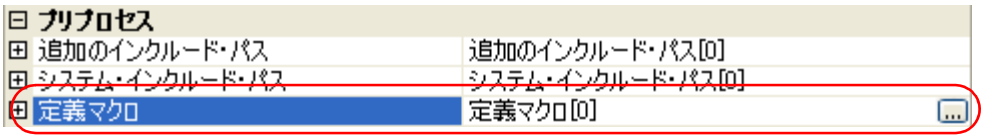
備考 ［共通オプション］タブの［よく使うオプション（アセンブラ）］カテゴリの［追加のインクルード・パス］プロパティでも，同様に設定することができます。

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。

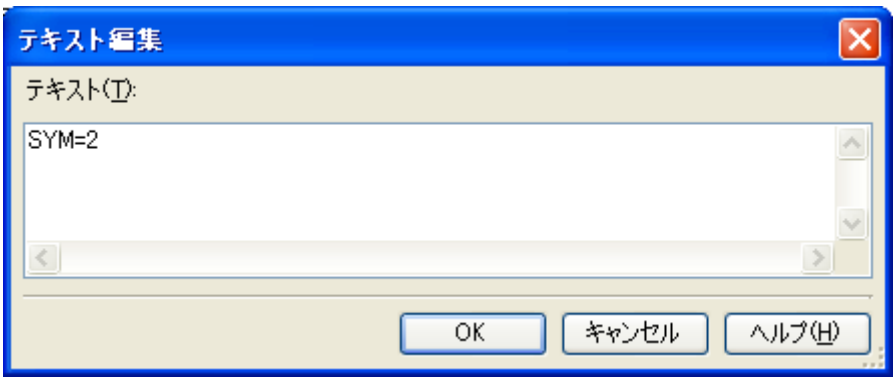
定義マクロの設定は，［プリプロセス］カテゴリの［定義マクロ］プロパティで行います。

図 2—42 ［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。

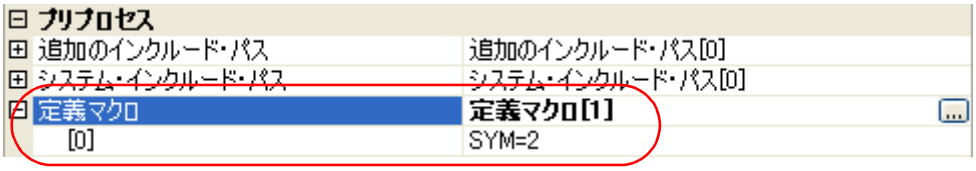
図 2—43 テキスト編集 ダイアログ



［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—44 ［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。

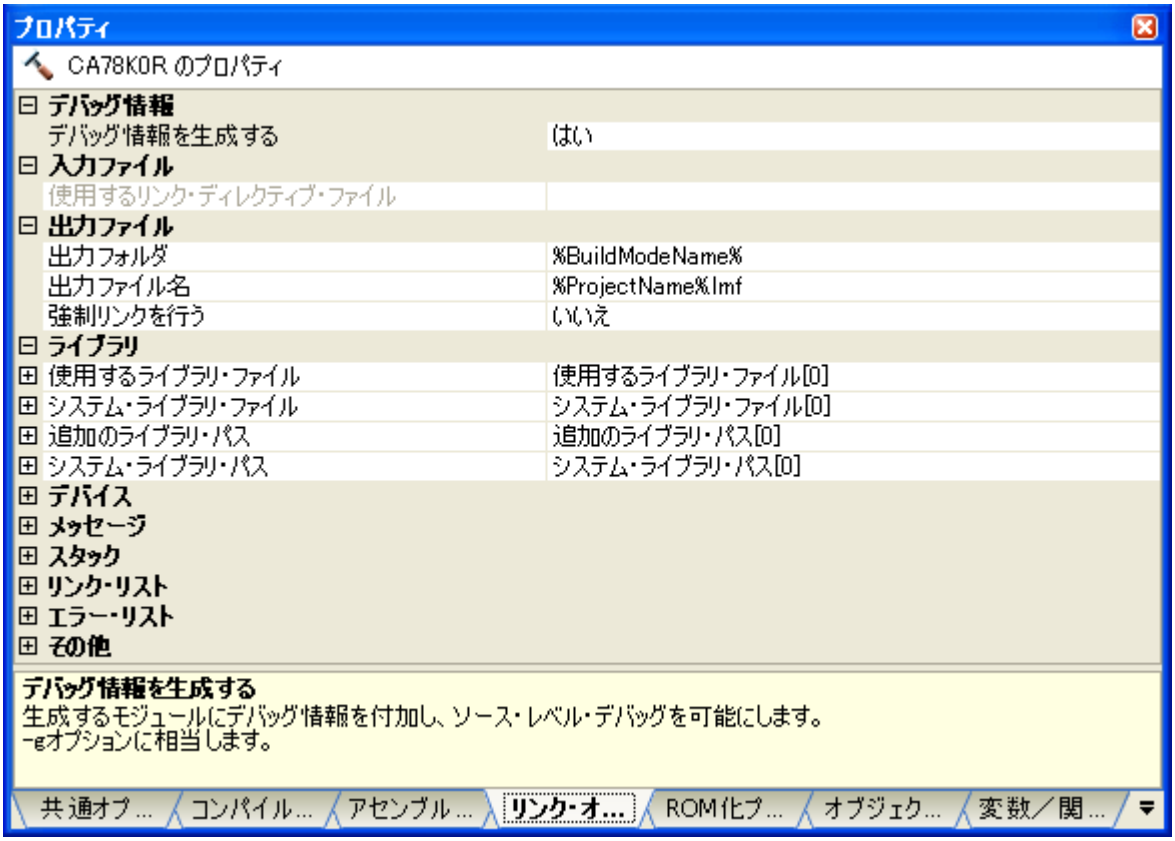
2.7　リンク・オプションを設定する

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—45　プロパティ パネル：［リンク・オプション］タブ



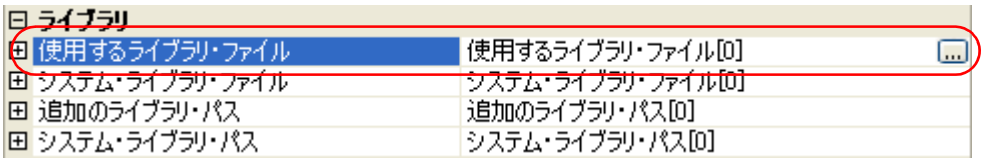
備考　よく使うオプションについては，［共通オプション］タブの［よく使うオプション（リンカ）］カテゴリにまとめられています。

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。

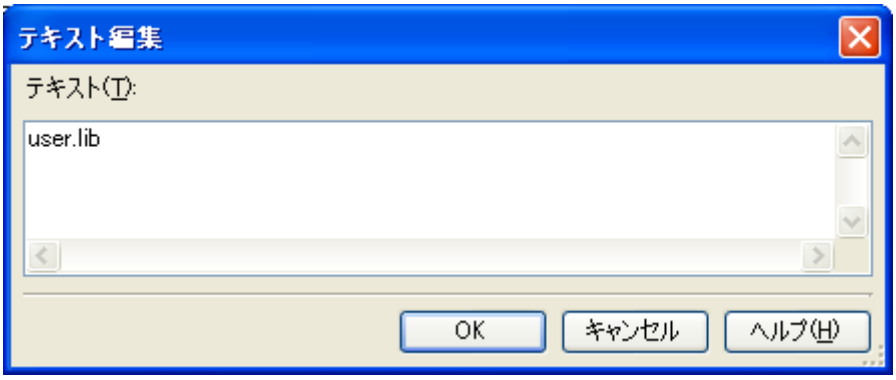
ユーザ・ライブラリの追加は，［ライブラリ］カテゴリの［使用するライブラリ・ファイル］プロパティで行います。

図 2—46 ［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。

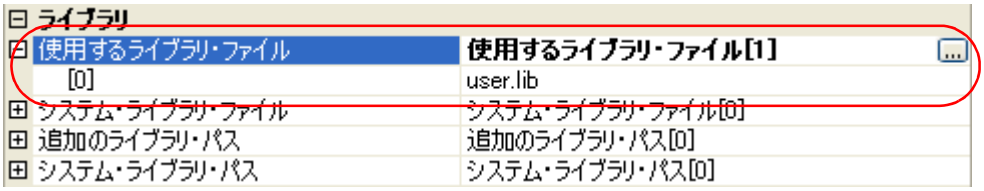
図 2—47 テキスト編集 ダイアログ



［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。

［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—48 ［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。

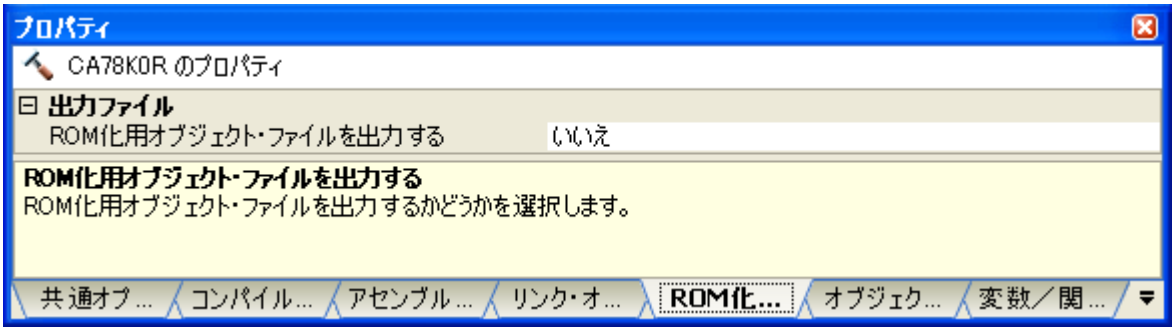
2. 8 ROM 化プロセス・オプションを設定する

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—49 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。

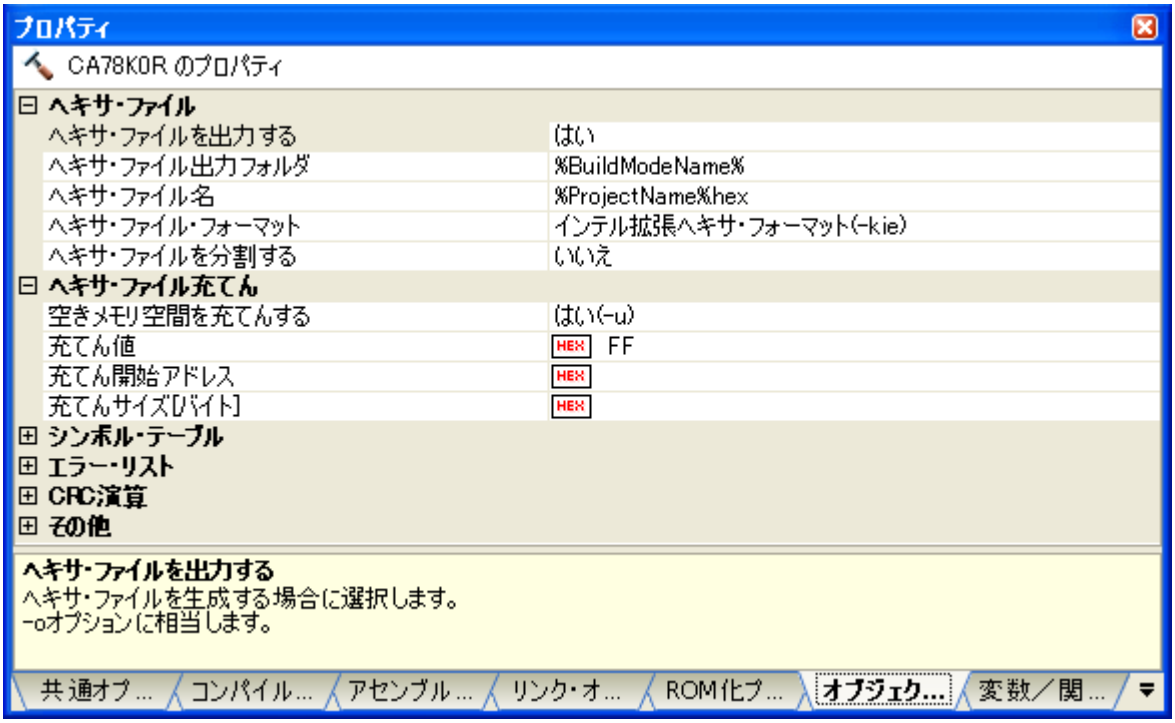
2.9　オブジェクト・コンバート・オプションを設定する

オブジェクト・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［オブジェクト・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，対応するオブジェクト・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—50　プロパティ パネル：［オブジェクト・コンバート・オプション］タブ



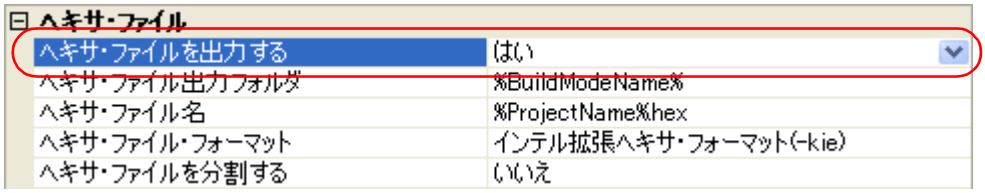
備考　よく使うオプションについては，［共通オプション］タブの［よく使うオプション（オブジェクト・コンバータ）］カテゴリにまとめられています。

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［オブジェクト・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，［ヘキサ・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 (-no)］を選択してください。

図 2—51 ［ヘキサ・ファイルを出力する］プロパティ



備考 シンボル・テーブル・ファイルの出力のみが目的でオブジェクト・コンバートする場合などに［ヘキサ・ファイルを出力する］プロパティで［いいえ (-no)］を選択すると，オブジェクト・コンバート時間を短縮することができます。

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 259 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。

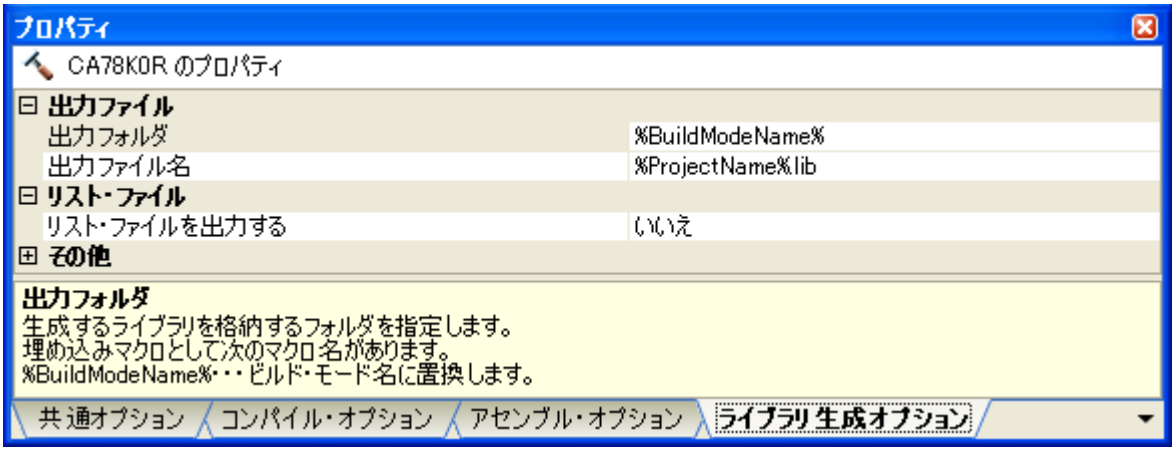
2. 10　ライブラリ生成オプションを設定する

ライブラリアン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［ライブラリ生成オプション］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，対応するライブラリ生成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—52　プロパティ パネル：［ライブラリ生成オプション］タブ

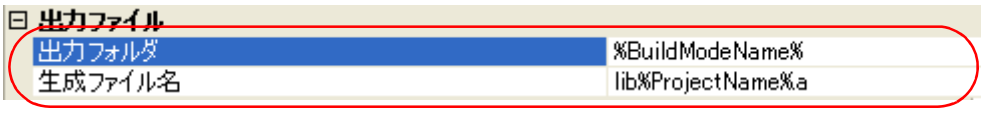


2. 10. 1　ライブラリ・ファイルの出力を設定する

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［ライブラリ生成オプション］タブを選択します。

ライブラリ・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—53　［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 259 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

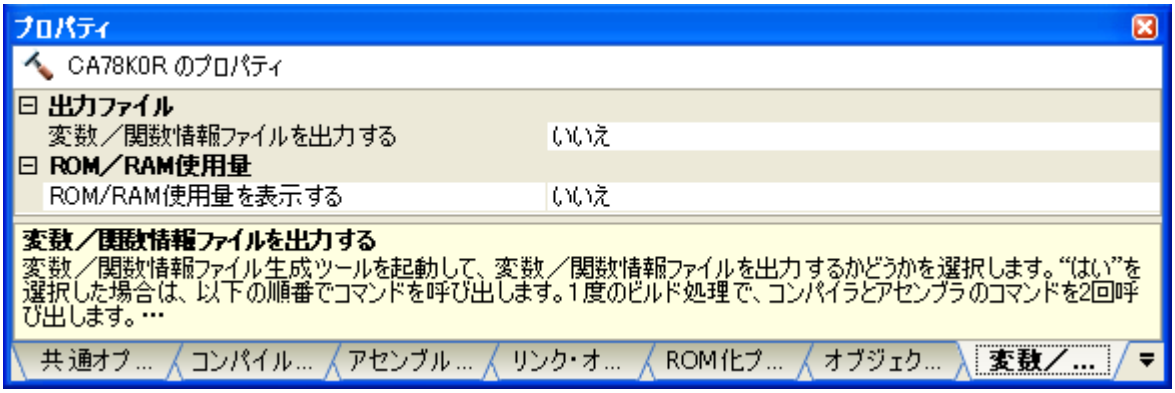
［出力ファイル名］プロパティにおいて，テキスト・ボックスへの直接入力により行います。テキスト・ボックスには 259 文字まで指定可能です。デフォルトでは，“%ProjectName%.lib”が設定されています。“%ProjectName%”は，埋め込みマクロで，プロジェクト名に置換します。

2. 11 変数／関数配置オプションを設定する

変数／関数情報ファイル生成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［変数／関数配置オプション］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，対応する変数／関数配置オプションを設定することができます。

図 2—54 プロパティ パネル：［変数／関数配置オプション］タブ



2. 11. 1 変数や関数を効率よく配置する

変数や関数を効率よく配置するには，変数／関数情報ファイル生成ツールを使用します。変数／関数情報ファイル生成ツールにより，変数／関数情報ファイル（参照されるすべての変数や関数の配置情報を記述したファイル）を生成し，その変数／関数情報ファイルを使用してコンパイルを行うことで，変数は saddr 領域，関数は callt 領域に配置されます。

以下に，操作手順を示します。

- 変数／関数情報ファイルを自動生成して変数や関数の配置を行う場合
- 自動生成した変数／関数情報ファイルを編集して使用する場合

なお，本機能を使用する前に，ビルドが正常に終了してロード・モジュール・ファイルが生成されていることを確認してください。

(1) 変数／関数情報ファイルを自動生成して変数や関数の配置を行う場合

1 回のビルドにより，変数／関数情報ファイルを自動生成し，そのファイルを使用して変数や関数の配置までを行う場合の手順を示します。

(a) 変数／関数情報ファイルの生成の設定

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［変数／関数配置オプション］タブを選択します。

［変数／関数情報ファイルを出力する］プロパティで［はい］を選択すると，空の変数／関数情報ファイルを生成し，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，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティで設定されているものとなります。

備考 すでに同名の変数／関数情報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—55 ［変数／関数情報ファイルを出力する］プロパティ

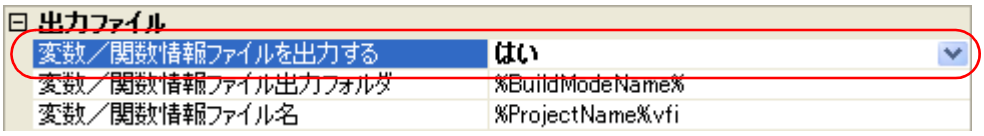
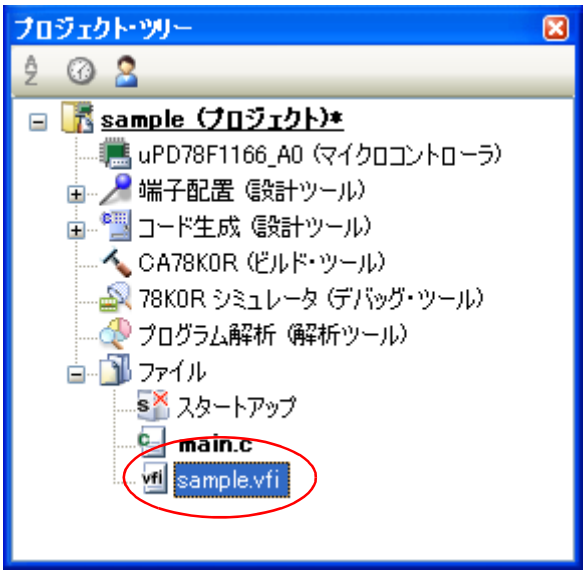


図 2—56 プロジェクト・ツリー パネル（変数／関数情報ファイル生成後）



変数／関数情報ファイルの出力フォルダ，およびファイル名の設定は，変更することもできます。

- 出力フォルダの設定

［変数／関数情報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 259 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。

本プロパティを変更すると，空の変数／関数情報ファイルを生成し，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）。

- 出力ファイル名の設定

［変数／関数情報ファイル名］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%.vfi”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，空の変数／関数情報ファイルを生成し，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）。

(b) プロジェクトのビルドの実行

プロジェクトのビルドを実行してください。

変数／関数情報ファイルが生成され，自動的にそれをコンパイラに入力して再度リビルドが実行されます。

備考 ビルドの実行により，「(a) 変数／関数情報ファイルの生成の設定」で生成した変数／関数情報ファイルが上書きされます。

ビルドが正常に終了すると，変数や関数の配置を行ったロード・モジュール・ファイルが生成されます。

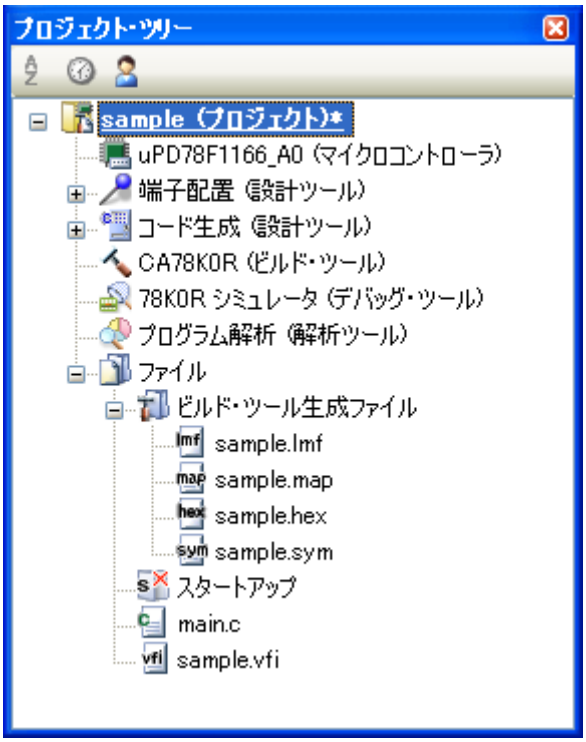
この時，メッセージ「E7001 : The link error was found.」が表示された場合，リンク時にエラーが発生しています。

そこで，以下の対処を行い，変数／関数情報ファイルを無効にしてください。

- ［変数／関数配置オプション］タブの［変数／関数情報ファイルを出力する］プロパティで［いいえ］を選択します。

- プロジェクト・ツリーに表示している変数／関数情報ファイル（*.vfi）のプロパティで，［ビルドの対象とする］を［いいえ］にします。
  または，変数／関数情報ファイルを選択したのち，コンテキスト・メニュー→［プロジェクトから外す］を選択します。

図 2—57 プロジェクト・ツリー パネル（ロード・モジュール・ファイル生成後）



(2) 自動生成した変数／関数情報ファイルを編集して使用する場合

変数／関数情報ファイルは，ユーザが編集することも可能です。

「(1) 変数／関数情報ファイルを自動生成して変数や関数の配置を行う場合」で生成した変数／関数情報ファイルをユーザが編集し，そのファイルを使用して変数や関数の配置を行う場合の手順を示します。

(a) 変数／関数情報ファイルの編集

「(1) 変数／関数情報ファイルを自動生成して変数や関数の配置を行う場合」で自動生成した変数／関数情報ファイルを編集します。

備考 自動生成した変数／関数情報ファイルのフォーマットについては，「3. 8. 1 変数／関数情報ファイル」を参照してください。

変数／関数情報ファイルの記述内容は，以下のフォーマットに従ってください。

```
; 変数情報
変数名，参照回数，サイズ，参照タイプ," ファイル名 ",const   ;static 変数，かつ const 変数
変数名，参照回数，サイズ，参照タイプ,,const                ;global 変数，かつ const 変数
変数名，参照回数，サイズ，参照タイプ," ファイル名 "          ;static 変数
変数名，参照回数，サイズ，参照タイプ                         ;global 変数
変数名，参照回数，サイズ，参照タイプ,,const,boot   ; ブート領域側の global 変数，かつ const 変数
変数名，参照回数，サイズ，参照タイプ,,,boot        ; ブート領域側の global 変数
;
; 関数情報
関数名，参照回数，サイズ," ファイル名 "   ;static 関数
関数名，参照回数，サイズ                  ;global 関数
関数名，参照回数，サイズ,,boot            ; ブート領域側の global 関数
```

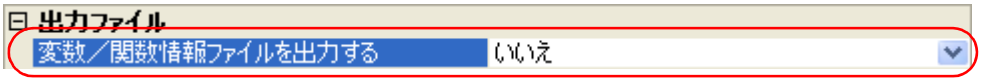
備考 変数，および関数は，優先順位の高いものから順に記述してください。
配置を行わない変数，および関数については，行頭にセミコロンを付加してコメントにしてください。

(b) 変数／関数情報ファイルの生成の設定

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［変数／関数配置オプション］タブを選択します。

［変数／関数情報ファイルを出力する］プロパティで［いいえ］を選択します。

図 2—58 ［変数／関数情報ファイルを出力する］プロパティ



(c) プロジェクトのビルドの実行

プロジェクトのビルドを実行してください。

変数／関数情報ファイルで指定した内容で変数や関数の配置を行ったロード・モジュール・ファイルが生成されます。

注意 拡張子が“vfi”のファイルをプロジェクトに追加した場合，そのファイルは変数／関数情報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変数／関数情報ファイルとみなされます。
変数／関数情報をプロジェクトに追加する際，すでに変数／関数情報ファイルを追加している場合は，追加する最新の変数／関数情報ファイルのみがビルド対象となり，追加済みの変数／関数情報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変数／関数情報ファイルをビルド対象に設定する際，ほかにも変数／関数情報ファイルを追加している場合は，ビルド対象に設定した変数／関数情報ファイルのみがビルド対象となり，それ以外の変数／関数情報ファイルはビルド対象外となります。

2. 11. 2 ROM/RAM 使用量を表示する

変数／関数情報ファイル生成ツールを使用すると，リンク後の ROM/RAM 使用量を出力 パネルに表示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［変数／関数配置オプション］タブを選択します。

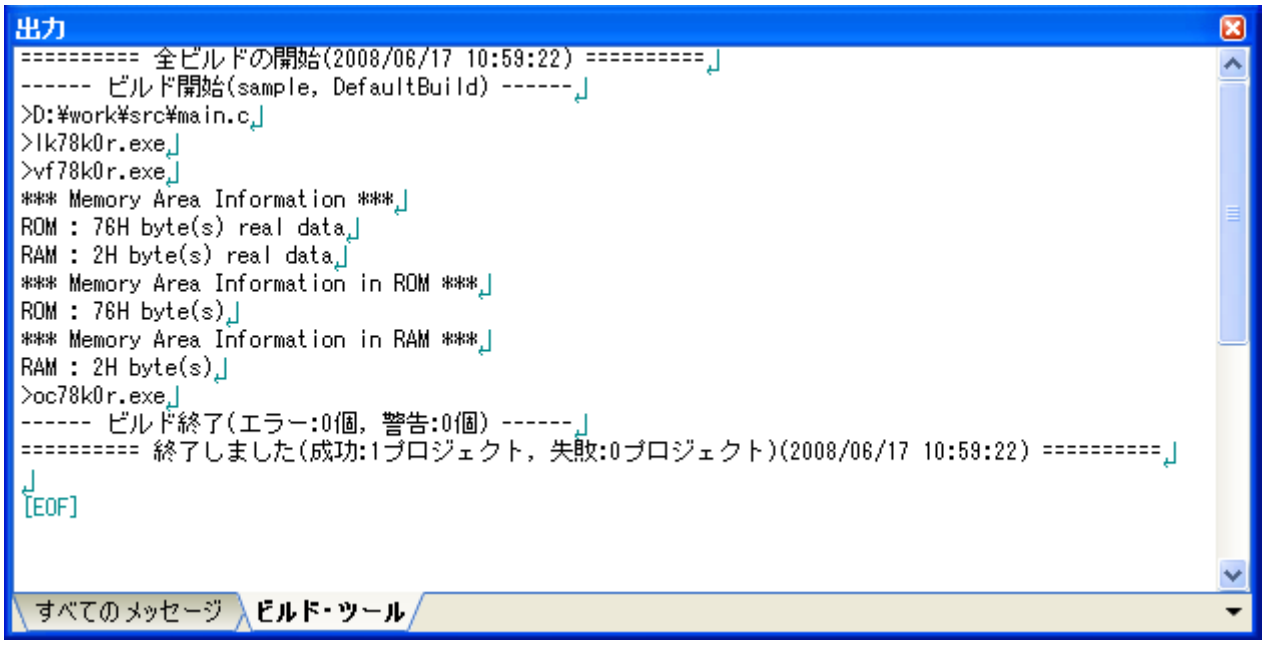
ROM/RAM 使用量を表示するには，［ROM/RAM 使用量］カテゴリの［ROM/RAM 使用量を表示する］プロパティで［はい］を選択してください（デフォルトでは，［いいえ］が選択されています）。

図 2—59 ［ROM/RAM 使用量を表示する］プロパティ

ビルドを実行すると，出力 パネルにおいて，ROM/RAM 使用量がビルド結果に続けて出力されます。

最初に使用量の合計値が出力され，次にメモリ領域ごとに使用量が出力されます。

図 2—60 ROM/RAM 使用量の表示結果イメージ



2. 12　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. 12. 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. 12. 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. 12. 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，リスト・コンバータ	→	［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
オブジェクト・コンバータ	→	［オブジェクト・コンバート・オプション］タブ
ライブラリアン	→	［ライブラリ生成オプション］タブ
変数／関数情報ファイル生成ツール	→	［変数／関数配置オプション］タブ

2. 12. 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

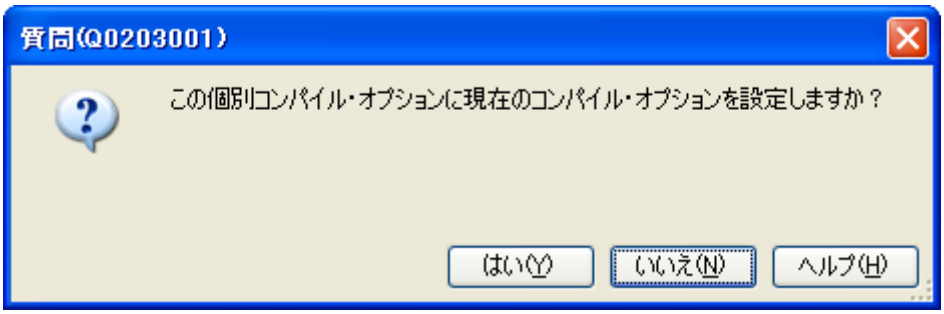
プロジェクトに追加している各ソース・ファイルに対して，コンパイル・オプション，またはアセンブル・オプションを個別に設定することができます。

(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—62　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。

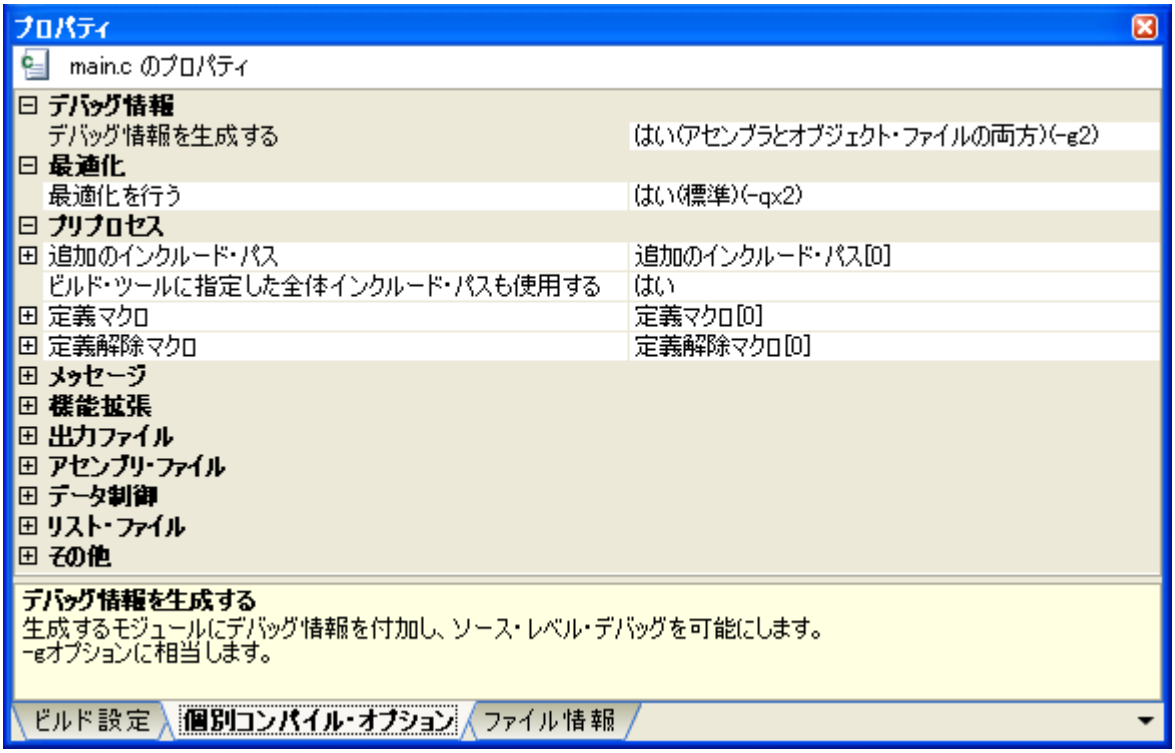
図 2—61　［個別コンパイル・オプションを設定する］プロパティ

図 2—62　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—63　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—65　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—64 ［個別アセンブル・オプションを設定する］プロパティ

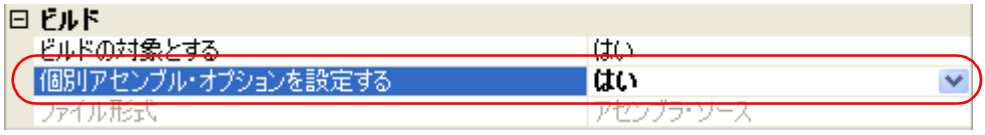
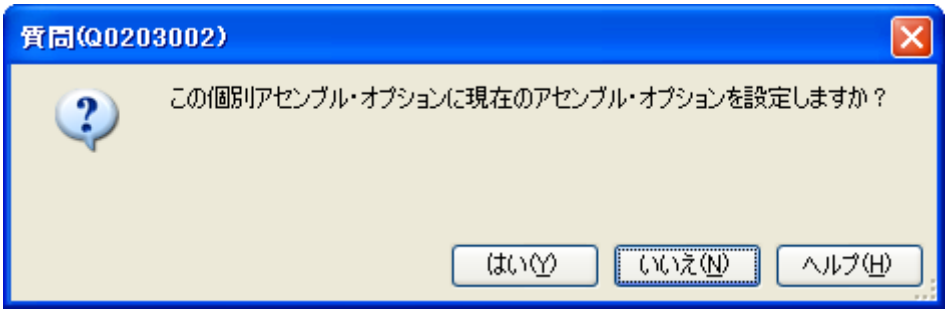
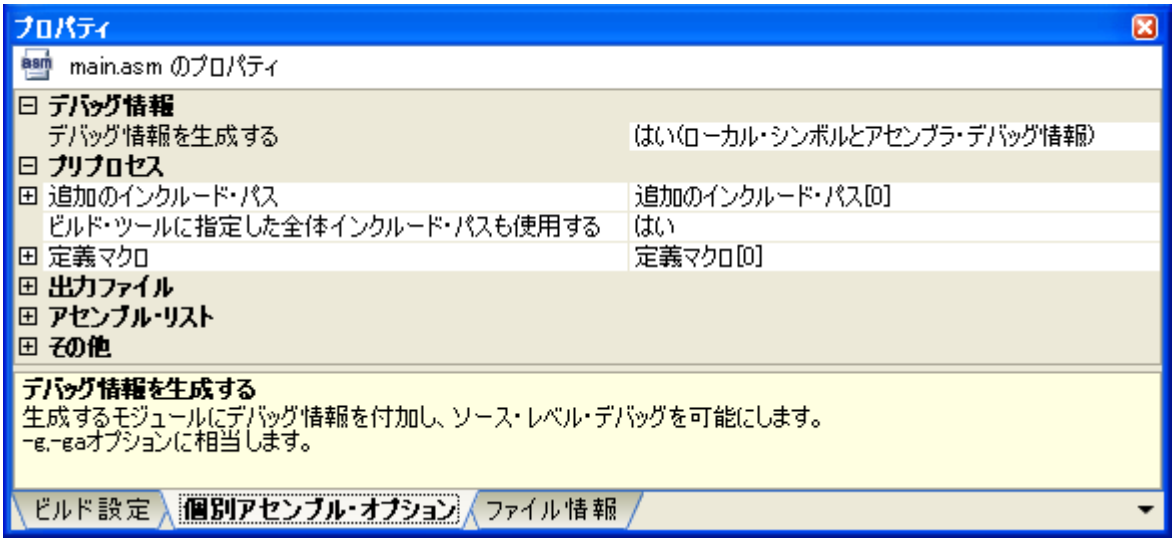


図 2—65 メッセージ ダイアログ



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—66 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。［アセンブリ・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］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. 13　オンチップ・デバッガを使用するための準備をする

オンチップ・デバッガを使用するためには，オンチップ・デバッグ，ユーザ・オプション・バイト，セキュリティ ID の設定が必要です。

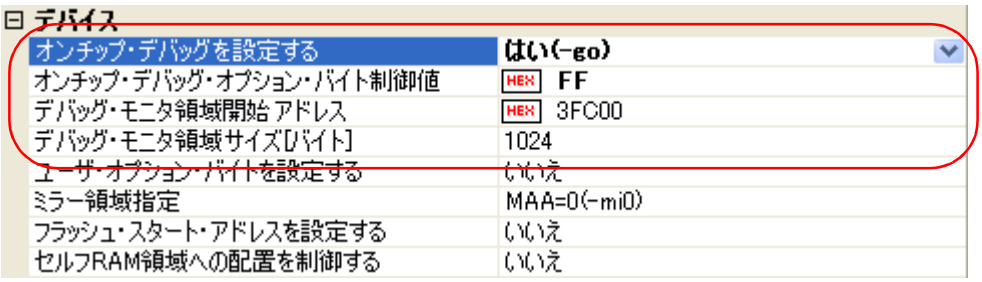
(1) オンチップ・デバッグの設定

オンチップ・デバッグを設定することにより，マイコンの持つオンチップ・デバッグ機能を有効にします。

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［リンク・オプション］タブを選択します。オンチップ・デバッグの設定は，［デバイス］カテゴリで行います。

［オンチップ・デバッグを設定する］プロパティで［はい (-go)］を選択すると，［オンチップ・デバッグ・オプション・バイト制御値］プロパティ，［デバッグ・モニタ領域開始アドレス］プロパティ，［デバッグ・モニタ領域サイズ [バイト]］プロパティが表示されます。

図 2—67　［オンチップ・デバッグを設定する］，［オンチップ・デバッグ・オプション・バイト制御値］，［デバッグ・モニタ領域開始アドレス］，および［デバッグ・モニタ領域サイズ [バイト]］プロパティ



［オンチップ・デバッグ・オプション・バイト制御値］プロパティで，オンチップ・デバッグ・オプション・バイトの制御値を 0x なしの 16 進数で指定します。指定可能な値の範囲は，0 ～ FF です。

［デバッグ・モニタ領域開始アドレス］プロパティで，デバッグ・モニタ領域の開始アドレスを 0x なしの 16 進数で指定します。指定可能な値の範囲は，0 ～ FFFFF です（デフォルト：内部 ROM のエンド・アドレス－1024 ＋ 1）。

［デバッグ・モニタ領域サイズ [バイト]］プロパティで，デバッグ・モニタ領域のサイズを 10 進数で指定します。指定可能な値の範囲は，0 ～ 1024【RL78】／ 88 ～ 1024【78K0R】です（デフォルト：512【RL78】／ 1024【78K0R】）。

(2) ユーザ・オプション・バイトの設定

ユーザ・オプション・バイトを設定することにより，ウォッチドッグ・タイマの設定，電圧検出回路／低電圧検出回路の設定，システム予約領域の設定を行います。

ユーザ・オプション・バイトの設定も，［リンク・オプション］タブの［デバイス］カテゴリで行います。

［ユーザ・オプション・バイトを設定する］プロパティで［はい (-gb)］を選択すると，［ユーザ・オプション・バイト値］プロパティが表示されます。

図 2—68 ［ユーザ・オプション・バイトを設定する］，および［ユーザ・オプション・バイト値］プロパティ

［ユーザ・オプション・バイト値］プロパティで，ユーザ・オプション・バイト値を 0x なしの 16 進数で指定します。指定可能な値の範囲は，0 ～ FFFFFF です。

上記のように指定した場合は，0xC0 番地に 0xFD，0xC1 番地に 0xFE，0xC2 番地に 0xFF が設定されます。

(3) セキュリティ ID の設定

セキュリティ ID は，デバッガ起動時の認証を行うために使用します。

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［共通オプション］タブを選択します。

［デバイス］カテゴリの［セキュリティ ID］プロパティで，セキュリティ ID を 20 桁の 16 進数で指定します。指定可能な値は，0 ～ FFFFFFFFFFFFFFFFFFFE【RL78】／0 ～ FFFFFFFFFFFFFFFFFFFF【78K0R】です。

図 2—69 ［セキュリティ ID］プロパティ

上記のように指定した場合は，0xC4 番地に 0x11，0xC5 番地に 0x22，0xC6 番地に 0x33，0xC7 番地に 0x44，0xC8 番地に 0x55，0xC9 番地に 0x66，0xCA 番地に 0x77，0xCB 番地に 0x88，0xCC 番地に 0x99，0xCD 番地に 0xAA が設定されます。

備考 デバッグ・ツールとの接続については，「CubeSuite+ RL78,78K0R デバッグ編」を参照してください。

2. 14 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. 5 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. 14. 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) #pragma ext_func 指令の記述

C ソース・ファイルに，#pragma ext_func 指令を記述します。

#pragma ext_func 指令で，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。

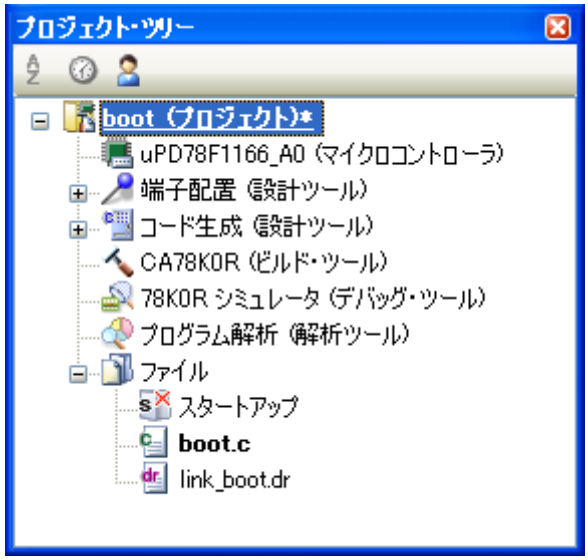
備考 記述漏れやソース間の矛盾が生じることを防ぐため，#pragma ext_func 指令の記述は 1 つのファイルにまとめ，ブート領域側，フラッシュ領域側を問わず，全 C ソース・ファイルにインクルードすることを推奨します。

2. 14. 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。

図 2—70 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 パネルで各ビルド・オプションを設定します。

(a) 変数／関数配置オプションの設定

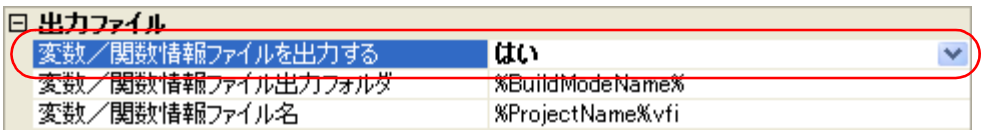
変数／関数情報ファイルを生成して変数や関数の配置を行う場合は，変数／関数配置オプションを設定します。

［変数／関数配置オプション］タブを選択します。

［出力ファイル］カテゴリの［変数／関数情報ファイルを出力する］プロパティで［はい］を選択すると，空の変数／関数情報ファイルを生成し，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，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティで設定されているものとなります。

備考 すでに同名の変数／関数情報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—71 ブート領域側の［変数／関数情報ファイルを出力する］プロパティ



変数／関数情報ファイルの出力フォルダ，およびファイル名を変更する場合は，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティを設定してください。［変数／関数情報ファイル名］プロパティを変更すると，空の変数／関数情報ファイルを生成し，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）。

(b) コンパイル・オプションの設定

［コンパイル・オプション］タブを選択します。

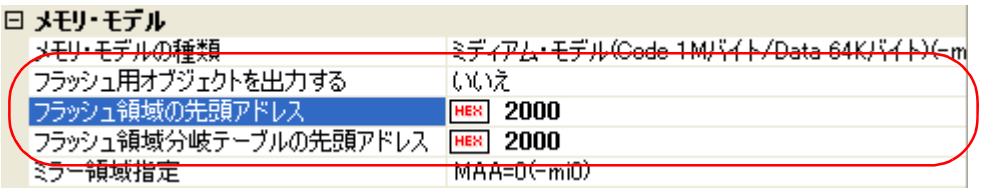
［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］プロパティで［いいえ］を選択します（デフォルト）。

また，［フラッシュ領域の先頭アドレス］プロパティと［フラッシュ領域分岐テーブルの先頭アドレス］プロパティを設定します。

指定可能な値の範囲は，どちらも 0C0 ～ 0EDFFF です。

備考　［フラッシュ領域分岐テーブルの先頭アドレス］プロパティに指定するアドレスは，フラッシュ領域内のアドレスとします。

図 2—72　ブート領域側の［フラッシュ用オブジェクトを出力する］，［フラッシュ領域の先頭アドレス］，および［フラッシュ領域分岐テーブルの先頭アドレス］プロパティ



次に，［スタートアップ］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（ブート領域用）］を選択します。

図 2—73　ブート領域側の［標準のスタートアップを使用する］プロパティ

(c) リンク・オプションの設定

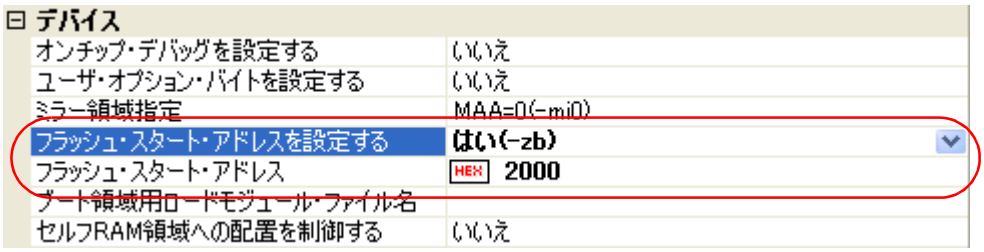
［リンク・オプション］タブを選択します。

［デバイス］カテゴリの［フラッシュ・スタート・アドレスを設定する］プロパティで［はい (-zb)］を選択すると，［フラッシュ・スタート・アドレス］プロパティが表示されます。

ここで，フラッシュ・メモリ領域の先頭アドレスを指定します。指定可能な値の範囲は，選択したデバイスに依存します。

備考　本プロパティには，［コンパイル・オプション］タブの［メモリ・モデル］カテゴリの［フラッシュ領域の先頭アドレス］プロパティと同じ値が設定されます。
本プロパティを変更すると，［フラッシュ領域の先頭アドレス］プロパティに同じ値を設定します。

図 2—74 ブート領域側の［フラッシュ・スタート・アドレスを設定する］，および［フラッシュ・スタート・アドレス］プロパティ

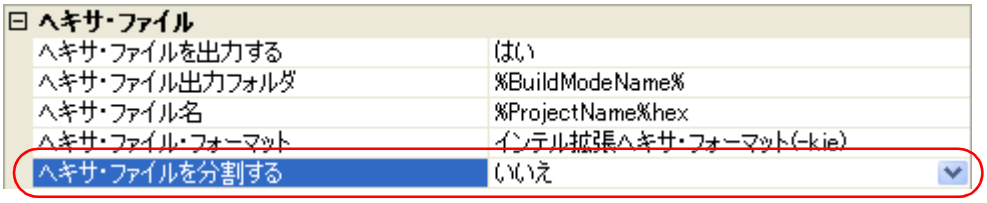


(d) オブジェクト・コンバート・オプションの設定

［オブジェクト・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］カテゴリの［ヘキサ・ファイルを分割する］プロパティで［いいえ］を選択します（デフォルト）。

図 2—75 ブート領域側の［ヘキサ・ファイルを分割する］プロパティ



(3) ブート領域側のプロジェクトのビルドの実行

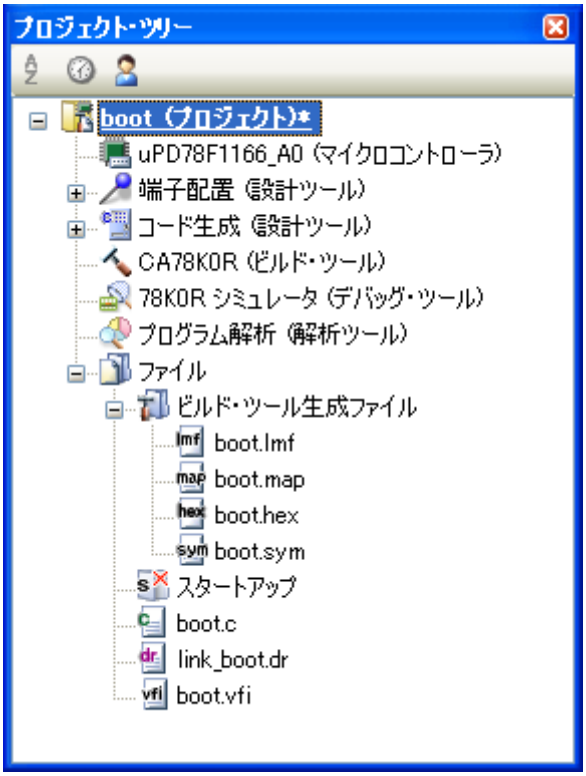
ブート領域側のプロジェクトのビルドを実行すると，ロード・モジュール・ファイルが生成されます。

また，ヘキサ・ファイルが生成されます。

変数／関数情報ファイルを生成した場合は，自動的にそれをコンパイラに入力して再度リビルドが実行されます。

備考 ビルドの実行により，「(a) 変数／関数配置オプションの設定」で生成した変数／関数情報ファイルを上書きします。

図 2—76 ブート領域側の生成ファイル

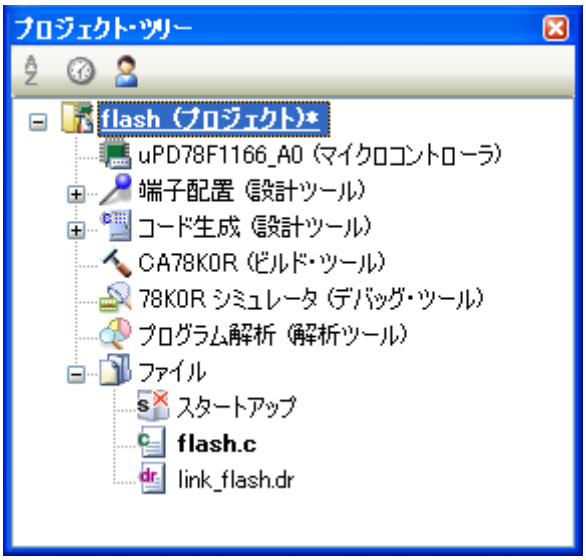


2. 14. 3 フラッシュ領域側のプロジェクトを設定する

(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。

図 2—77 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 パネルで各ビルド・オプションを設定します。

(a) 変数／関数配置オプションの設定

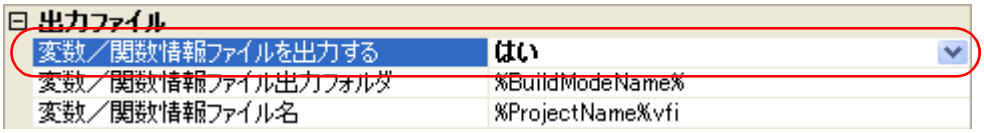
変数／関数情報ファイルを生成して変数や関数の配置を行う場合は，変数／関数配置オプションを設定します。

［変数／関数配置オプション］タブを選択します。

［出力ファイル］カテゴリの［変数／関数情報ファイルを出力する］プロパティで［はい］を選択すると，空の変数／関数情報ファイルを生成し，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，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティで設定されているものとなります。

備考 すでに同名の変数／関数情報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—78 フラッシュ領域側の［変数／関数情報ファイルを出力する］プロパティ



変数／関数情報ファイルの出力フォルダ，およびファイル名を変更する場合は，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティを設定してください。［変数／関数情報ファイル名］プロパティを変更すると，空の変数／関数情報ファイルを生成し，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）。

(b) コンパイル・オプションの設定

［コンパイル・オプション］タブを選択します。

［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］プロパティで［はい (-zf)］を選択します。

また，［フラッシュ領域の先頭アドレス］プロパティと［フラッシュ領域分岐テーブルの先頭アドレス］プロパティを設定します。

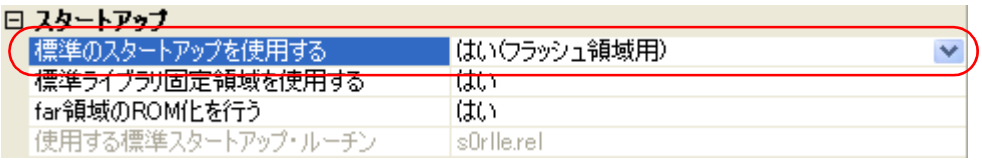
指定可能な値の範囲は，0C0 ～ 0EDFFF です。

備考 ［フラッシュ領域分岐テーブルの先頭アドレス］プロパティに指定するアドレスは，ブート領域側のプロジェクトで指定したアドレスと同じものとします。

図 2—79　フラッシュ領域側の［フラッシュ用オブジェクトを出力する］，［フラッシュ領域の先頭アドレス］，および［フラッシュ領域分岐テーブルの先頭アドレス］プロパティ

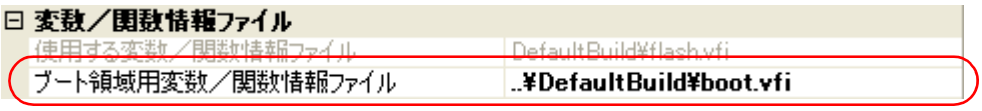
次に，［スタートアップ］カテゴリの［標準のスタートアップを使用する］プロパティで［はい（フラッシュ領域用）］を選択します。

図 2—80　フラッシュ領域側の［標準のスタートアップを使用する］プロパティ



次に，「2. 14. 2　ブート領域側のプロジェクトを設定する」で作成したブート領域側の変数／関数情報ファイルをフラッシュ領域側のプロジェクトに追加します。［変数／関数情報ファイル］カテゴリの［ブート領域用変数／関数情報ファイル］プロパティでブート領域側の変数／関数情報ファイルを指定します。

図 2—81　フラッシュ領域側の［ブート領域用変数／関数情報ファイル］プロパティ

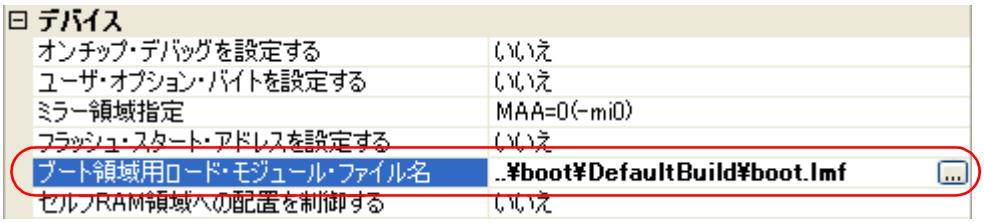


（c）リンク・オプションの設定

「2. 14. 2　ブート領域側のプロジェクトを設定する」で作成したブート領域側のロード・モジュール・ファイルをフラッシュ領域側のプロジェクトに追加します。［リンク・オプション］タブを選択します。

［デバイス］カテゴリの［ブート領域用ロード・モジュール・ファイル名］プロパティに，ブート領域側のロード・モジュール・ファイルを指定します。

図 2—82　フラッシュ領域側の［ブート領域用ロード・モジュール・ファイル名］プロパティ

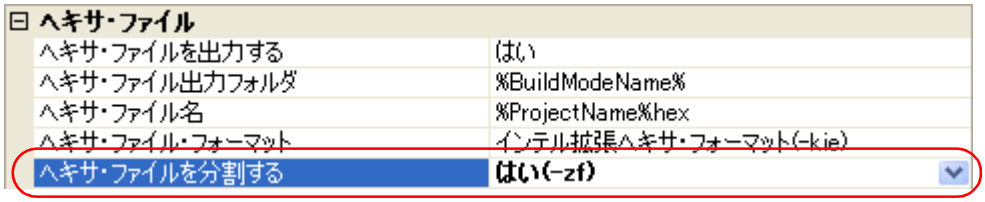


(d) オブジェクト・コンバート・オプションの設定

［オブジェクト・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］カテゴリの［ヘキサ・ファイルを分割する］プロパティで［はい (-zf)］を選択します（デフォルト）。

図 2—83 フラッシュ領域側の［ヘキサ・ファイルを分割する］プロパティ

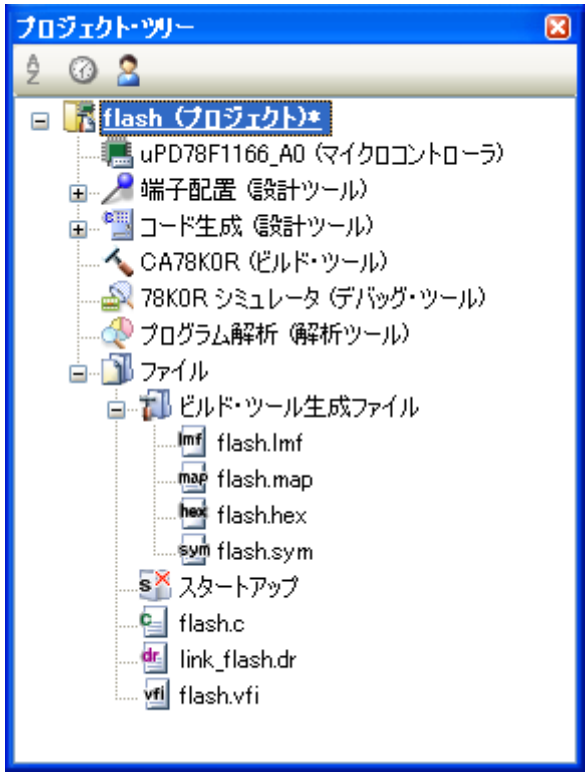


(3) フラッシュ領域側のプロジェクトのビルドの実行

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。

また，ブート領域用のヘキサ・ファイル（「2. 14. 2 ブート領域側のプロジェクトを設定する」で生成されたファイルと同じ内容となります）とフラッシュ領域用のヘキサ・ファイルが生成されます。

図 2—84 フラッシュ領域側の生成ファイル



2. 15 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。

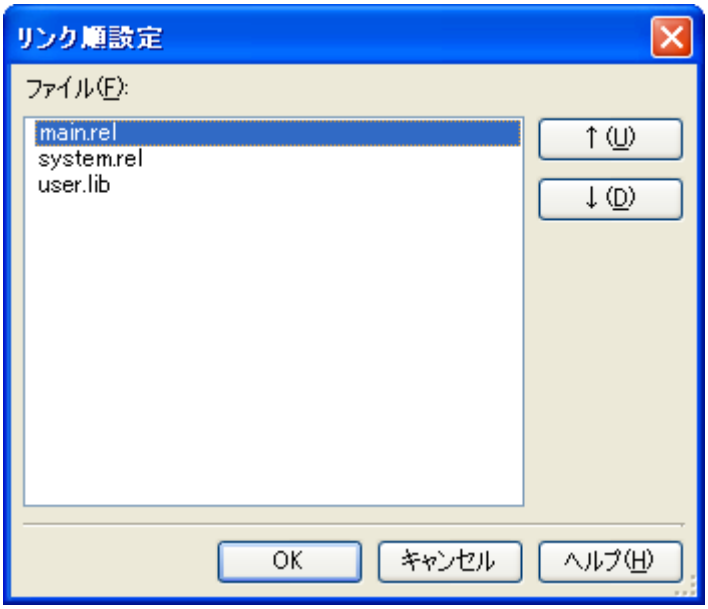
- ファイルのリンク順を設定する
- サブプロジェクトのビルド順を変更する
- ビルド・オプションを一覧表示する
- ビルド対象プロジェクトを変更する
- ビルド・モードを追加する
- ビルド・モードを変更する
- ビルド・モードを削除する
- 現在のビルド・オプションをプロジェクトの標準に設定する

2. 15. 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—85 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. 15. 2　サブプロジェクトのビルド順を変更する

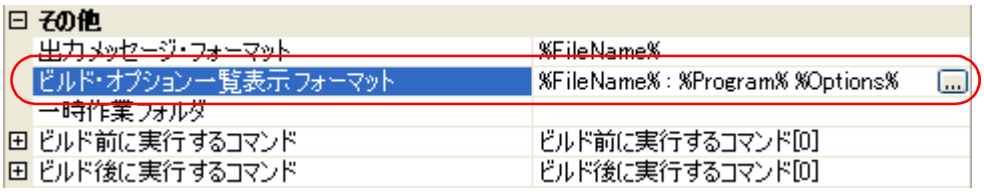
ビルドの実行は，サブプロジェクト，メイン・プロジェクトの順で行いますが，複数のサブプロジェクトを追加している場合，サブプロジェクトのビルド順はプロジェクト・ツリーでの表示順となります。
プロジェクト・ツリーでのサブプロジェクトの表示順を変更するには，移動するサブプロジェクトをドラッグし，移動先でドロップしてください。

2. 15. 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—86　［ビルド・オプション一覧表示フォーマット］プロパティ



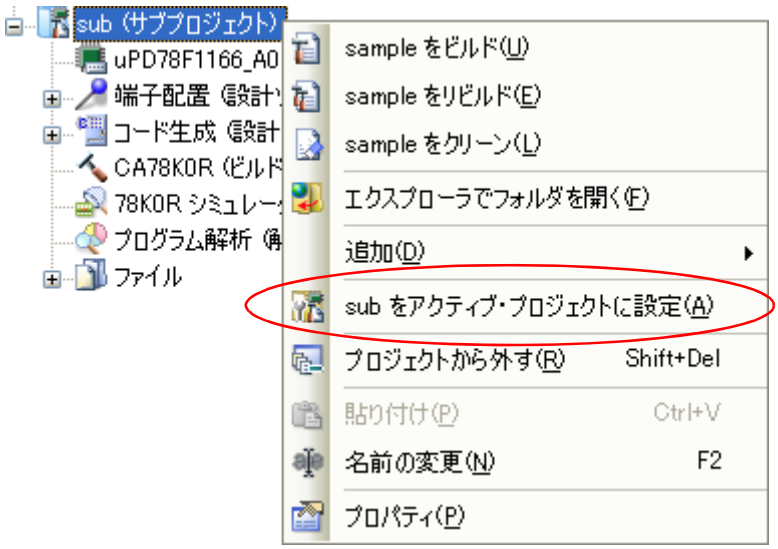
デフォルトでは，“%FileName% : %Program% %Options%”が設定されています。
“%FileName%”，“%Program%”，“%Options%”は，埋め込みマクロで，それぞれビルド中のファイル名，実行中のプログラム名，ビルド時のコマンド・ライン・オプションに置換します。

2. 15. 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。

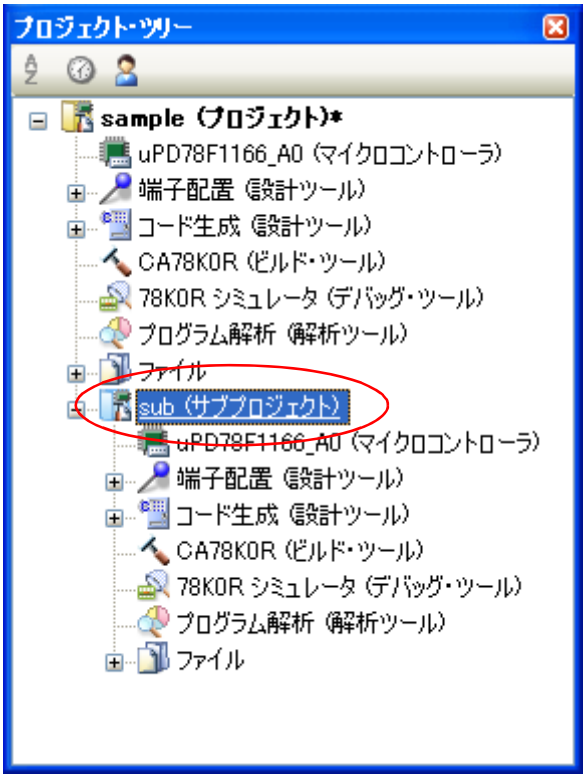
アクティブ・プロジェクトを設定するには，プロジェクト・ツリーでアクティブ・プロジェクトに設定するメイン・プロジェクト・ノード，またはサブプロジェクト・ノード選択し，コンテキスト・メニューの［*選択しているプロジェクト*をアクティブ・プロジェクトに設定］を選択してください。

図 2—87 ［*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—88 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. 15. 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

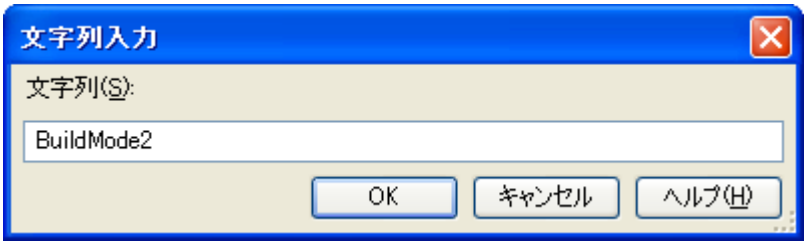
新規のビルド・モードの作成は，既存のビルド・モードの複製により行います。

［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—89 ビルド・モード設定 ダイアログ

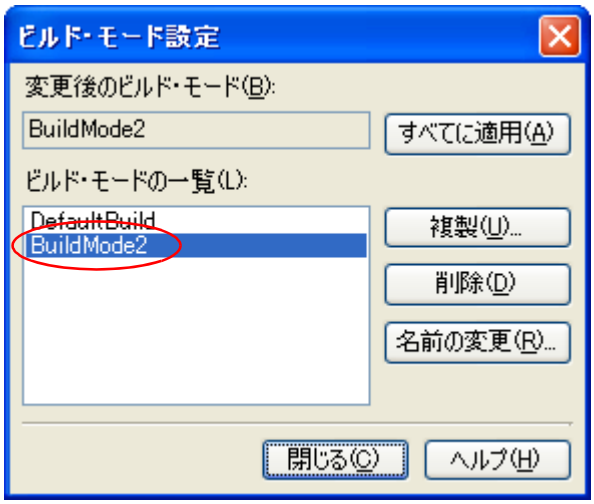
ビルド・モードの一覧から複製元のビルド・モードを選択したのち，［複製 ...］ボタンをクリックすると，文字列入力 ダイアログがオープンします。

図 2—90 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—91 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. 15. 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。

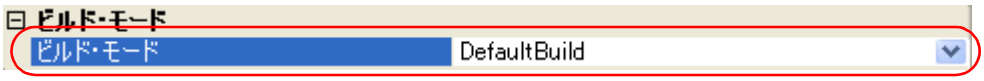
2. 15. 6 ビルド・モードを変更する

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。

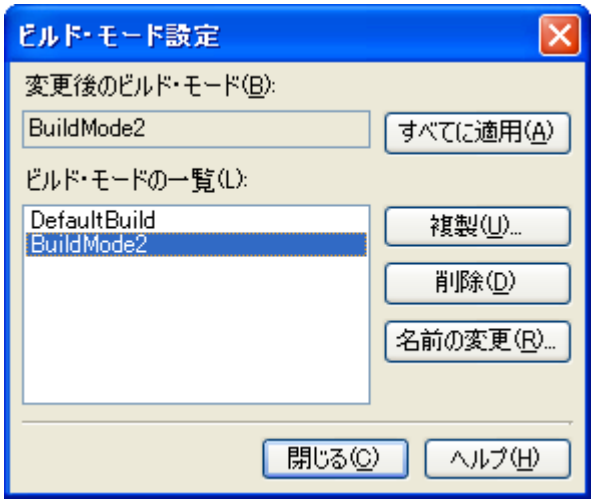
図 2—92 ［ビルド・モード］プロパティ



(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—93 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. 15. 5 ビルド・モードを追加する」を参照してください。

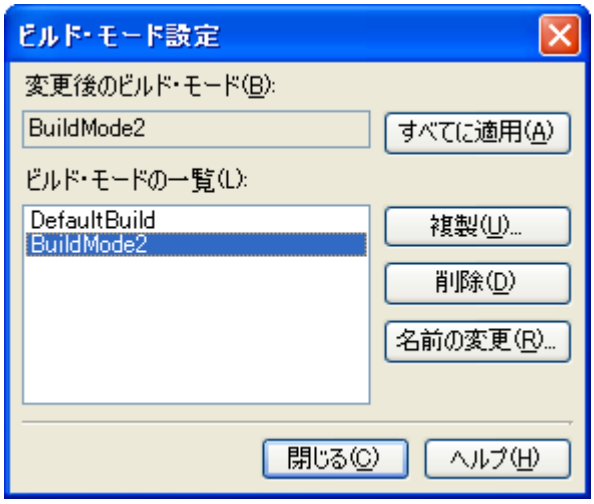
2. ビルド・モードの一覧でビルド・モードを選択したのち，［名前の変更］ボタンをクリックすることにより，ビルド・モードの名前を変更することができます。ただし，“DefaultBuild”は名前を変更することができません。

2. 15. 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。

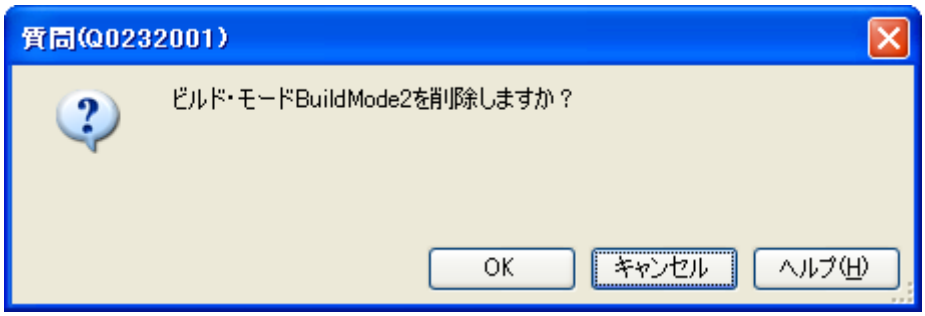
［ビルド］メニュー→［ビルド・モードの設定 ...］を選択すると，ダイアログがオープンします。

図 2—94 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—95 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。

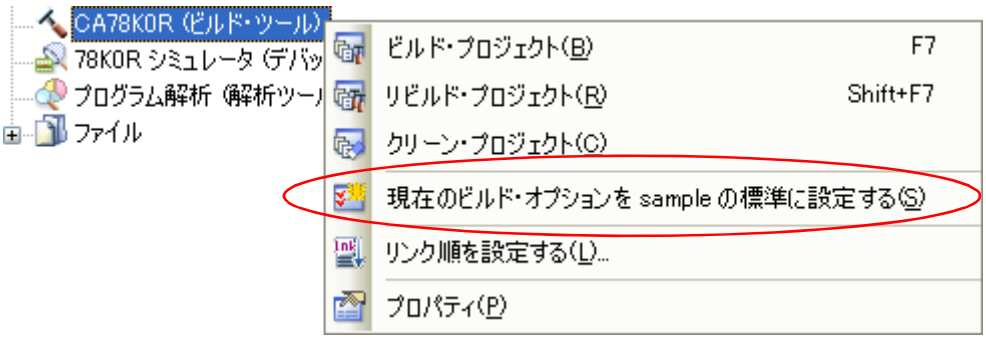
2. 15. 8 現在のビルド・オプションをプロジェクトの標準に設定する

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—96 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—97 ［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—98 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. 16　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. 16. 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. 16. 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. 16. 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. 16. 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. 15. 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—99　ビルドの実行結果（ビルド，リビルド，バッチ・ビルドの場合）

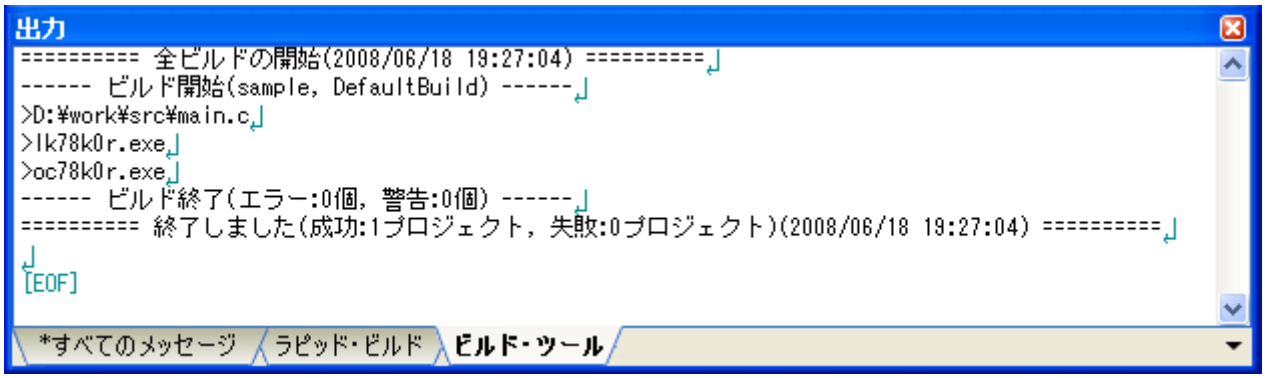
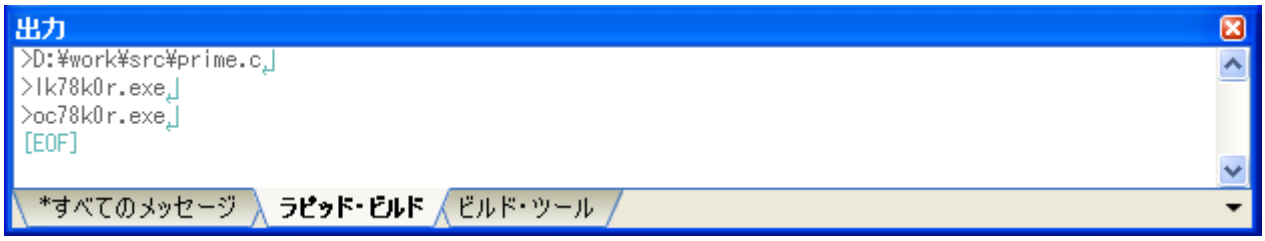


図 2—100 ビルドの実行結果（ラピッド・ビルドの場合）



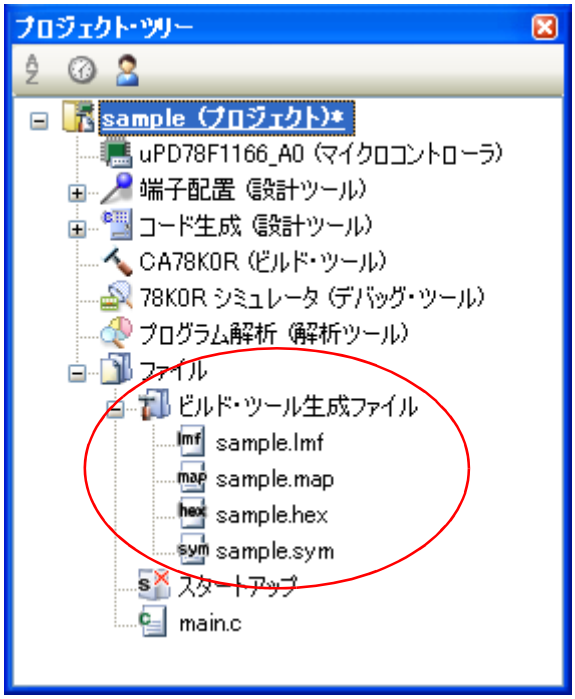
備考 1. ［ラピッド・ビルド］タブの表示文字列は，淡色表示になります。

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—101 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（*.lmf）

リンク・リスト・ファイル（*.map）

エラー・リスト・ファイル（*.elk）

ヘキサ・ファイル（*.hex，*.hxb，*.hxf）

シンボル・テーブル・ファイル（*.sym）

エラー・リスト・ファイル（*.eoc）

- ライブラリ用のプロジェクトの場合

ライブラリ・ファイル (*.lib)

リスト・ファイル (*.lst)

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. 16. 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. 15. 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。

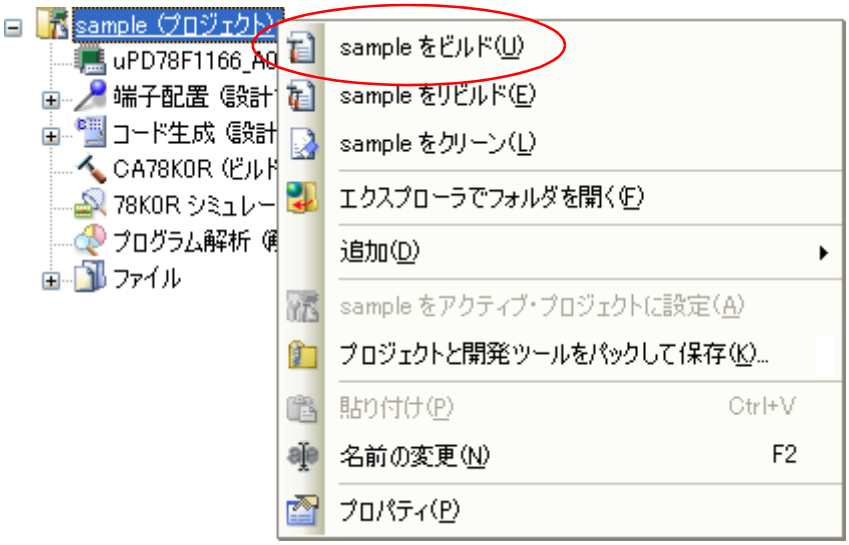
(1) プロジェクト全体のビルドを実行する場合

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—102　［*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. 7　ファイルの依存関係を更新する」参照）。

2. 16. 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. 15. 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。

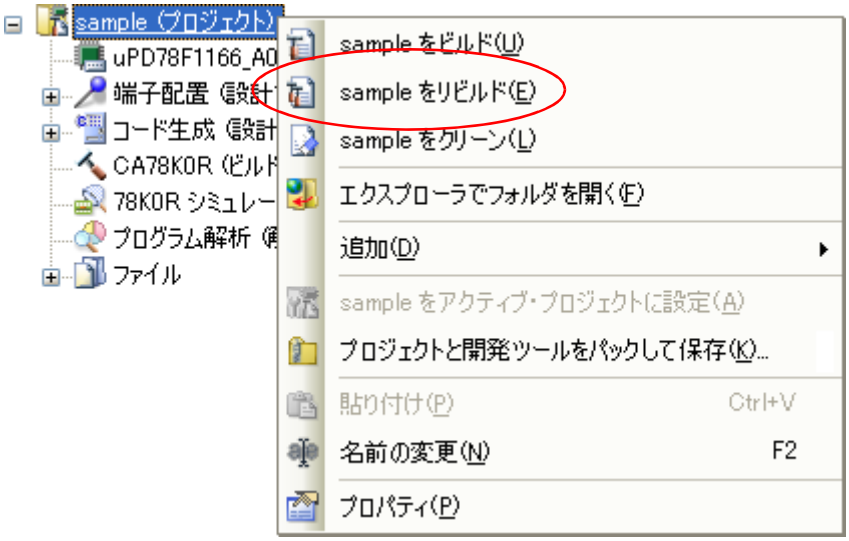
(1) プロジェクト全体のリビルドを実行する場合

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—103　［*アクティブ・プロジェクト*をリビルド］項目



2. 16. 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，変数／関数情報ファイル，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

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—104 ［ラピッド・ビルド］項目（ラピッド・ビルドが有効の場合）

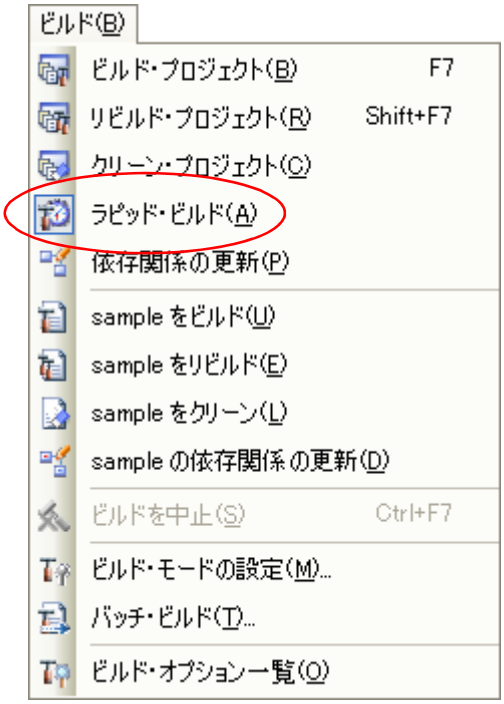
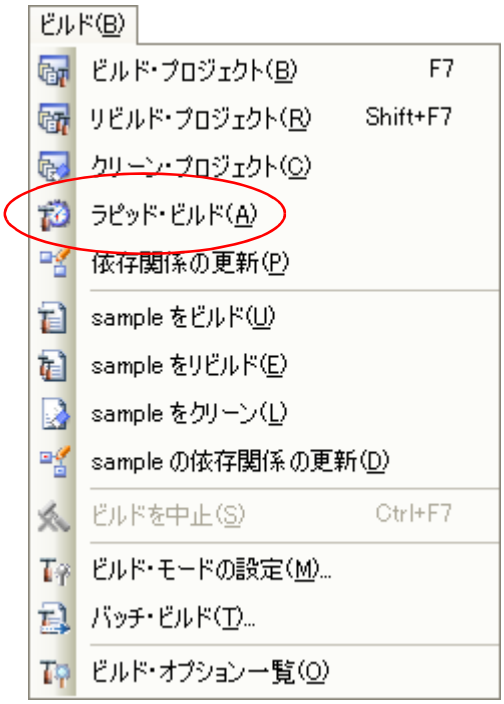


図 2—105 ［ラピッド・ビルド］項目（ラピッド・ビルドが無効の場合）



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. 16. 4 ビルド・モードを一括してビルドを実行する

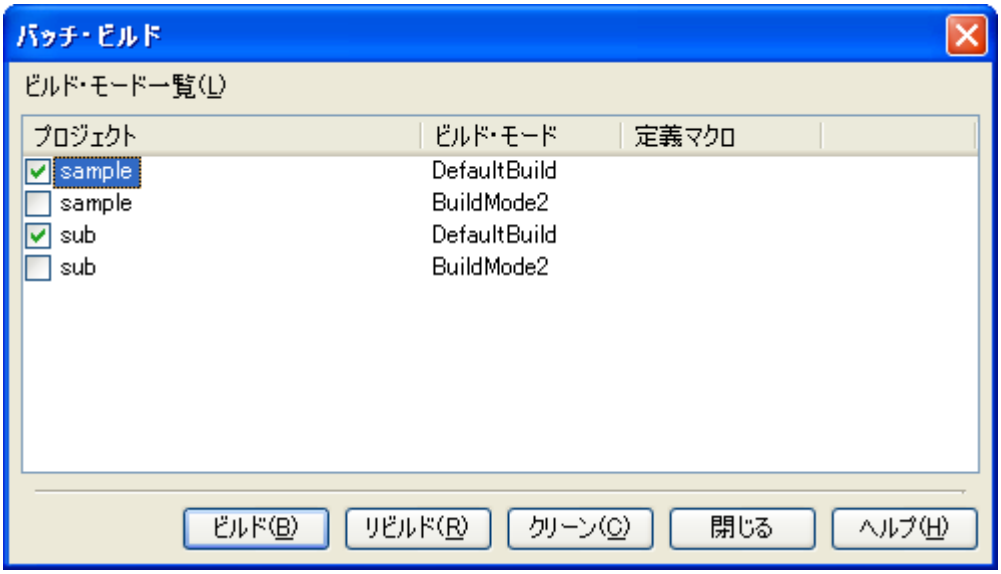
プロジェクト（メイン・プロジェクト，およびサブプロジェクト）が持つビルド・モードを一括して，ビルド／リビルド／クリーンを行うことができます（以降，“バッチ・ビルド”と呼びます）。

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. 16. 1 更新ファイルのビルドを実行する」参照
- リビルド → 「2. 16. 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. 16. 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—106 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。

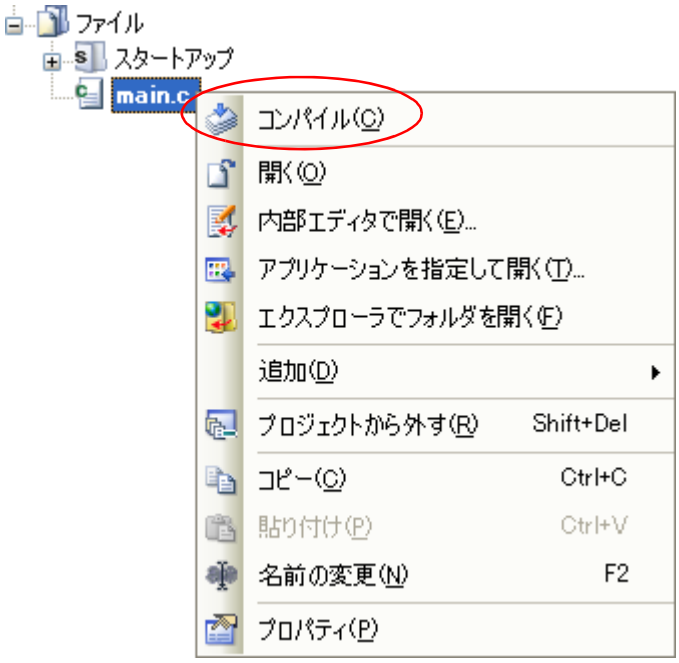
2. 16. 5 ファイル単位でコンパイル／アセンブルする

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。

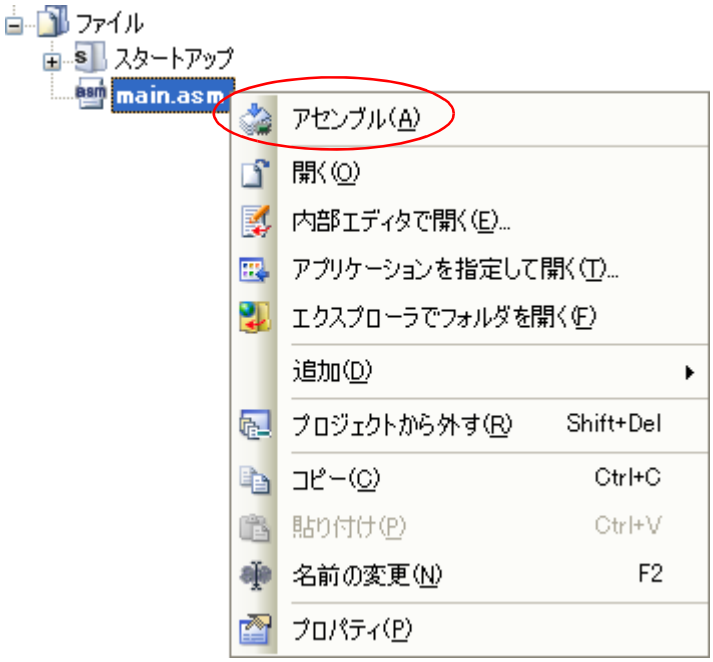
図 2—107 ［コンパイル］項目



(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—108 ［アセンブル］項目



2. 16. 6 ビルドの実行を中止する

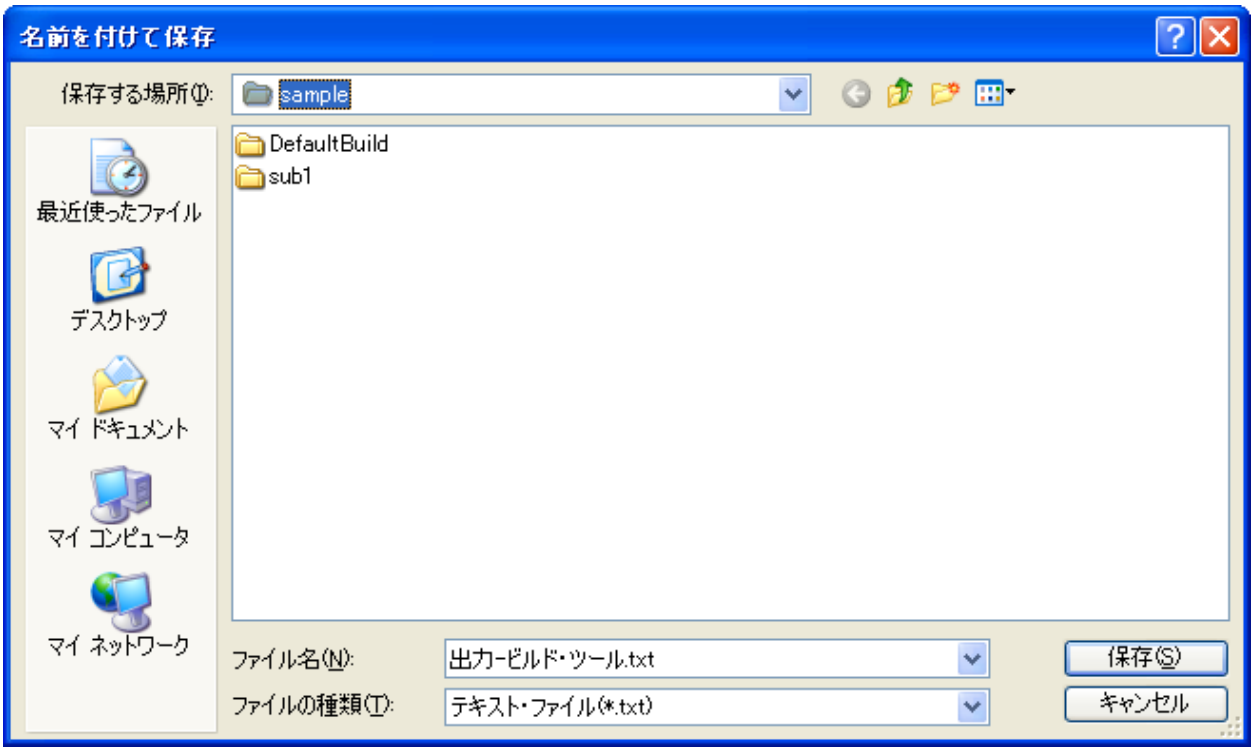
実行中のビルド，リビルド，バッチ・ビルドを中止するには，ツールバーの ボタンをクリックしてください。

2. 16. 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—109 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. 16. 8 中間ファイル，生成ファイルを削除する

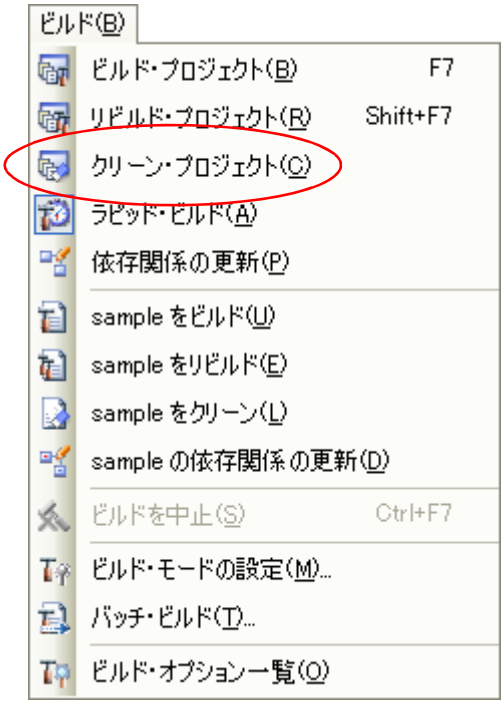
ビルドの実行により出力された中間ファイル，生成ファイルをすべて削除することができます（以降，“クリーン”と呼びます）。

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. 15. 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。

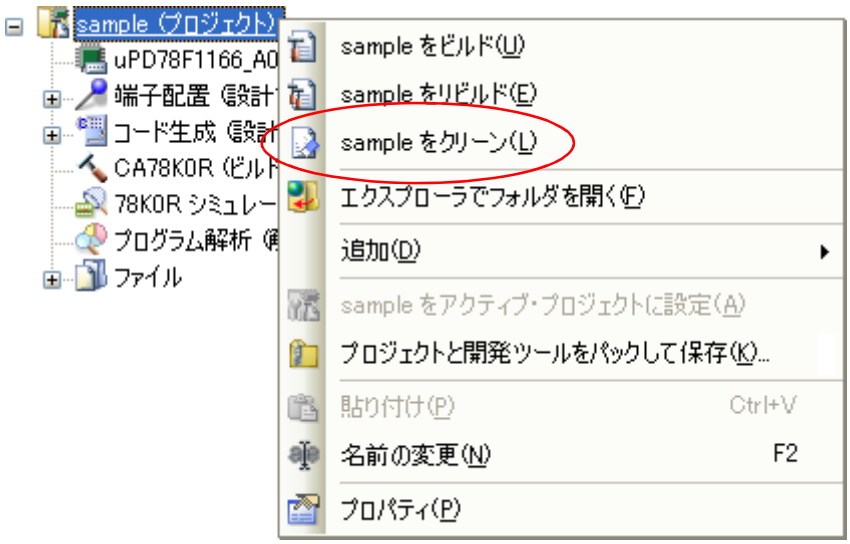
図 2—110 ［クリーン・プロジェクト］項目



(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—111 ［*アクティブ・プロジェクトをクリーン*］項目



2. 17　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. 17. 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。

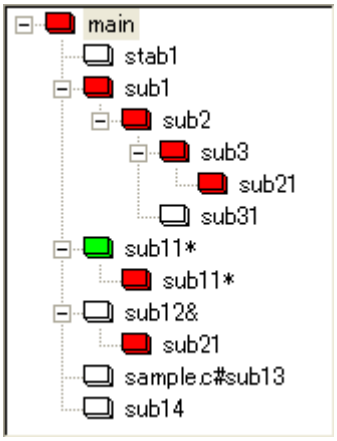
図 2—112　スタック見積もりツールの起動イメージ

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→［sk78k0r の終了］を選択することにより行います。

2. 17. 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—113 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. 17. 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—114　リスト表示エリア

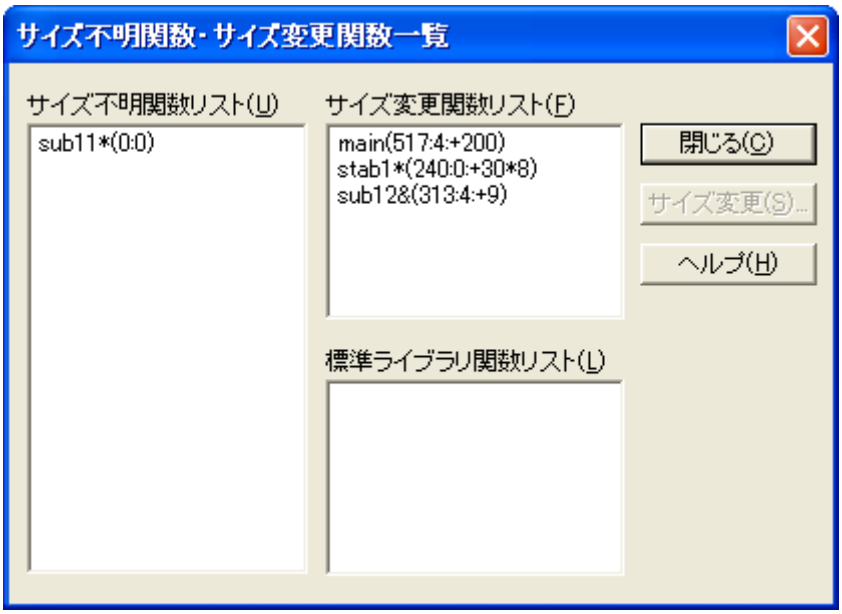
関数	スタックサイズ	単体スタックサイズ	加算サイズ	ファイル名
main	304	4		D:¥proj¥STK¥...
stab1	0	0		D:¥proj¥STK¥...
sub1	300	216		D:¥proj¥STK¥...
sub11*	0	0		D:¥proj¥STK¥...
sub12&	4	4		D:¥proj¥STK¥...
sample.c...	0	0		D:¥proj¥STK¥...
sub14	0	0		D:¥proj¥STK¥...

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. 17. 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—115 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. 17. 5 スタック・サイズを変更する

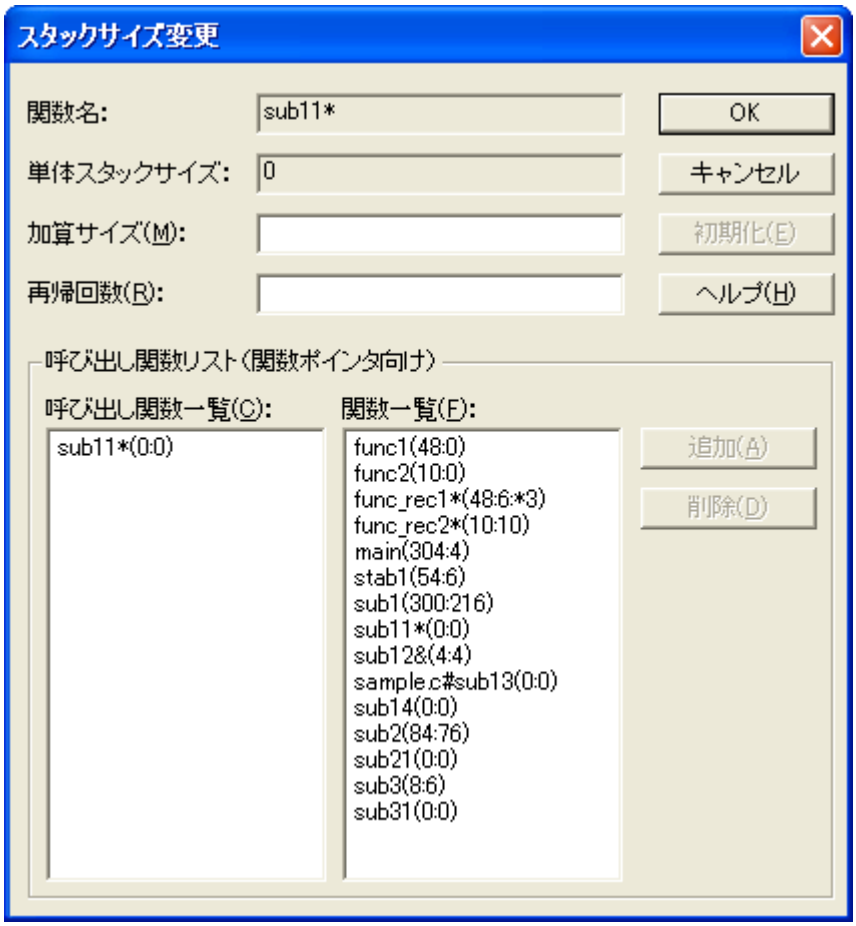
スタック見積もりツールがスタック情報を取得できていない関数，および意図的に情報を変更したい関数については，スタックサイズ変更 ダイアログ，またはスタック・サイズ指定ファイルを用いることにより，該当値を動的に設定することができます。

(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—116 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—117 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　C コンパイラ

C コンパイラは，次のファイルを出力します。

- アセンブラ・ソース・ファイル
- エラー・リスト・ファイル
- プリプロセス・リスト・ファイル
- クロスリファレンス・リスト・ファイル

備考　C コンパイラの入出力ファイルについては，「B. 1. 1　入出力ファイル」を参照してください。

3. 1. 1　アセンブラ・ソース・ファイル

アセンブラ・ソース・ファイルは，C ソースのコンパイル結果の ASCII イメージのリストで，C ソースに対応したアセンブリ言語のソース・ファイルです。

また，アセンブラ・ソース・ファイル作成指定オプション -sa により，コメントとして C ソースを含めることができます。

CubeSuite+ におけるアセンブラ・ソース・ファイルの出力設定は，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［コンパイル・オプション］タブで行います。［アセンブリ・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］を選択してください。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。

```
; 78K0R C Compiler V(1)x.xx Assembler Source       Date:(2)xx xxx xxxx Time:(3)xx:xx:xx

; Command   : (4)-cf1166a0 prime.c -sa
; In-file   : (5)prime.c
; Asm-file  : (6)prime.asm
; Para-file : (7)

    $PROCESSOR((8)F1166A0)
(9)  $DEBUG
(10)$NODEBUGA
(11)$KANJICODE SJIS
(12)$RAM_ALLOCATE(@@CNSTR,@@CNSTLR,@@CODER)
(13)$TOL_INF        03FH, 0xxxH, 00H, 04000H, 00H, 00H, 00H

(14)$DGS    FIL_NAM, .file,         03BH, 0FFFEH, 03FH, 067H, 01H, 00H
```

```
                :

(15)        EXTRN   _@RTARG0

                :

    ; line  (16)1 : (17)#define TRUE    1
    ; line  (16)2 : (17)#define FALSE   0
    ; line  (16)3 : (17)#define SIZE    20

                :

(15)_main :
(18)$DGL    1, 39
(15)        push    hl                          ;(22)[INF] 1, 1
(15)        subw    sp,#08H                     ;(22)[INF] 2, 1
(15)        movw    hl,sp                       ;(22)[INF] 3, 1

                :

(19)??bf_main :

                :

    ; (23)*** Code Information ***
    ;
    ; (24)$FILE C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥Vx.xx¥
Smp78k0r¥CC78K0R¥prime.c
    ;
    ; (25)$FUNC printf(8)
    ; (26)      void=(pointer s:ax, int i:[sp+4])
    ; (27)      CODE SIZE= 18 bytes, CLOCK_SIZE= 16 clocks, STACK_SIZE= 8 bytes
    ;
    ; (25)$FUNC putchar(17)
    ; (26)      void=(char c:x)
    ; (27)      CODE SIZE= 4 bytes, CLOCK_SIZE= 9 clocks, STACK_SIZE= 2 bytes
    ;
    ; (25)$FUNC main(23)
    ; (26)      void=(void)
    ; (27)      CODE SIZE= 148 bytes, CLOCK_SIZE= 107 clocks, STACK_SIZE= 16 bytes
    ;
    ; (28)$CALL printf(33)
    ; (29)      void=(pointer:ax, int:[sp+4])
    ;
    ; (28)$CALL putchar(35)
```

```
; (29)       void=(int:ax)
; (28)$CALL printf(40)
; (29)       void=(pointer:ax, int:[sp+4])

; Target chip : (20)uPD78F1166_A0
; Device file : (21)Vx.xx
```

項番	内容	形式
(1)	バージョン番号	“*x.yz*”の形式で表します。
(2)	日付	システム日付。 “*DD Mmm YYYY*”の形式で表します。
(3)	時間	システム時間。 “*HH* : *MM* : *SS*”の形式で表します。
(4)	コマンド行	コマンド行の“CC78K0R”以降を出力します。 80 カラム目からは，次行の 15 カラム目から出力します。ただし，1 カラム目に“ ; ”を出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(5)	C ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.c”を付加します。80 カラム目からは，次行の 15 カラム目から出力します。ただし，1 カラム目に“ ; ”を出力します。
(6)	アセンブラ・ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.asm”を付加します。80 カラム目からは，次行の 15 カラム目から出力します。ただし，1 カラム目に“ ; ”を出力します。
(7)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。 80 カラム目からは，次行の 15 カラム目から出力します。ただし，1 カラム目に“ ; ”を出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(8)	デバイス種別	-c オプションで指定された文字列です。
(9)	デバッグ情報	DEBUG コントロールを出力します。 $DEBUG，あるいは $NODEBUG のいずれかです。
(10)	アセンブラのデバッグ情報の制御	NODEBUGA コントロールを出力します。 $NODEBUGA を出力します。
(11)	漢字種別情報	漢字種別を出力します。 $KANJICODE SJIS，$KANJICODE EUC，あるいは $KANJICODE NONE を出力します。
(12)	RAM 領域配置指定制御命令	指定したセグメント名を持つセグメントを RAM に配置します。 本項目は，-zx オプションを指定した場合のみ出力します。
(13)	ツール情報	ツール情報，バージョン番号，エラー情報，オプションの有無などを出力します（$TOL_INF で始まる情報）。
(14)	シンボル情報	シンボル情報を出力します（$DGS で始まる情報）。 デバッグ情報出力指定オプションを指定した場合にのみ出力します。ただし，-g1 が指定された場合は出力しません。
(15)	アセンブラ・ソース本体	コンパイル結果のアセンブラ・ソースを出力します。

項番	内容	形式
(16)	行番号	C ソース・ファイルの行番号（右詰めゼロ・サプレスの 10 進数）です。
(17)	C ソース	入力 C ソース・イメージです。 80 カラム目からは，次行の 16 カラム目から出力します。ただし，1 カラム目に“ ; ”を出力します。
(18)	ライン・ナンバ情報	ライン・ナンバ・エントリ用の行番号（$DGL で始まる情報）です。 デバッグ情報出力指定オプションを指定した場合のみ出力します。ただし，-g1 が指定された場合は出力しません。
(19)	シンボル情報作成用ラベル	関数のラベル情報です（?? で始まる情報）。 デバッグ情報出力指定オプションを指定した場合にのみ出力します。
(20)	コンパイルの対象品種名	コマンド行オプション -c，またはソース・ファイルにおいて指定された対象デバイス名を表示します。
(21)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。
(22)	サイズ，クロック	出力命令に対する，サイズ，クロック数を出力します（; [INF] で始まる情報）。 出力命令のクロック数を確定できない場合，クロック 1 ／クロック 2 の形式で出力します。 クロック数は，内部 RAM 領域，SFR 領域をアクセスした場合，またはデータ・アクセスをしない場合のクロック数を出力します。 また，条件分岐命令の場合，条件成立時のクロック数を出力します。 ハザードは，考慮していません。 したがって，実際のクロック数と異なりますので注意してください。 参考の値になります。
(23)	関数情報（開始）	関数情報の開始を示します。
(24)	関数情報（ファイル名）	対象ソース・ファイル名をフルパスで出力します（; $FILE で始まる情報）。
(25)	関数情報（定義関数）	関数名，および定義行番号を 10 進で出力します（; $FUNC で始まる情報）。
(26)	関数情報（定義関数の返り値，引数）	定義関数の返り値レジスタ，引数情報（レジスタ，またはスタック位置）を出力します。
(27)	関数情報（定義関数のサイズ，クロック，スタック）	定義関数に対する静的に計算したサイズ，クロック，最大消費スタックを出力します。 ここで表示されるスタック量は，関数本体で使用するスタック量のみです。 例えば，関数内で別の関数を呼び出している場合，呼び出し先の関数で使用したスタック分は，呼び出し元の関数のスタック量には加算されません。 また，CLOCK_SIZE は項番 (22) のクロックを加算したものです。
(28)	関数情報（呼び出し関数）	関数名と関数呼び出し行番号を 10 進で出力します（; $CALL で始まる情報）。
(29)	関数情報（呼び出し関数の返り値，引数）	関数呼び出し時の，返り値レジスタと引数情報（レジスタ，またはスタック位置）を出力します。

3. 1. 2　エラー・リスト・ファイル

エラー・リスト・ファイルは，コンパイル中に発生したワーニングやエラー・メッセージの集まりでできています。

コンパイル・オプションを指定することにより，エラー・リスト中に C ソースを付加することができます。C ソースを含むエラー・リスト・ファイルは，C ソースを修正し，リスト・ヘッダなどのコメントを削除することにより，C ソース・ファイルとして使用することができます。

CubeSuite+ におけるエラー・リスト・ファイルの出力設定は，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［コンパイル・オプション］タブで行います。［リスト・ファイル］カテゴリの［エラー・リスト・ファイルを出力する］プロパティで［はい］を選択してください。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。

- C ソース・ファイル付きのエラー・リスト・ファイル

```
/*
78K0R C Compiler V(1)x.xx Error List            Date:(2)xx xxx xxxx Time:(3)xx:xx:xx

Command   : (4)-cf1166a0 prime.c -se
C-file    : (5)prime.c
Err-file  : (6)prime.cer
Para-file : (7)
*/

(8) #define TRUE    1
(8) #define FALSE   0
(8) #define SIZE    20

(8) char    mark[SIZE+1];

(8) void main(){
(8)     int i, prime, k, count;
(8)     cont = 0;
    *** CC78K0R error (9)E0711: (10)Undeclared 'cont' ; function 'main'
(8)     for (i = 0; i <= SIZE; i++)
(8)             mark[i] = TRUE;
(8)     for (i = 0; i <= SIZE; i++) {
(8)             if (mark[i]) {
(8)                     prime = i + i + 3;
(8)                     printf("%6d", prime );
    *** CC78K0R warning (9)W0745: (10)Expected function prototype
            :
/*
(11)Target chip : uPD78F1166_A0
(12)Device file : Vx.xx
Compilation complete, (13)1 error(s) and (14)5 warning(s) found.
*/
```

項番	内容	形式
(1)	バージョン番号	“*x.yz*”の形式で表します。
(2)	日付	システム日付。 “*DD Mmm YYYY*”の形式で表します。
(3)	時間	システム時間。 “*HH* : *MM* : *SS*”の形式で表します。
(4)	コマンド行	コマンド行の“CC78K0R”以降を出力します。 80 カラム目からは，次行の 13 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(5)	C ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.c”を付加します。80 カラム目からは，次行の 13 カラム目から出力します。
(6)	エラー・リスト・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.cer”を付加します。80 カラム目からは，次行の 13 カラム目から出力します。
(7)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。 80 カラム目からは，次行の 13 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(8)	C ソース	入力 C ソース・イメージです。 80 カラム目以降も折り返しを行わず出力します。
(9)	エラー・メッセージ番号	エラー番号を“*#nnnn*”の形式で出力します。 # は，内部エラーの場合は C，フェイタル・エラーの場合は E，アボート・エラーの場合は F，ワーニングの場合は W となります。 *nnnn* はエラー番号で，10 進数 4 桁で表します（ゼロ・サプレスしません）。
(10)	エラー・メッセージ	エラー・メッセージを出力します。 80 カラム以降も折り返しを行わず出力します。
(11)	コンパイルの対象品種名	コマンド行オプション -c，またはソース・ファイルにおいて指定された対象デバイス名を表示します。
(12)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。
(13)	エラー個数	右詰めゼロ・サプレスの 10 進数。
(14)	ワーニング個数	右詰めゼロ・サプレスの 10 進数。

- エラー・メッセージのみのエラー・リスト・ファイル

```
(1) prime.c ((2)18) : CC78K0R warning (3)W0745: (4)Expected function prototype
(1) prime.c ((2)20) : CC78K0R warning (3)W0745: (4)Expected function prototype
(1) prime.c ((2)26) : CC78K0R warning (3)W0622: (4)No return value
(1) prime.c ((2)37) : CC78K0R warning (3)W0622: (4)No return value
(1) prime.c ((2)44) : CC78K0R warning (3)W0622: (4)No return value

     Target chip : (5)uPD78F1166_A0
     Device file : (6)Vx.xx

Compilation complete, (7)0 error(s) and (8)5 warning(s) found.
```

項番	内容	形式
(1)	C ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.c”を付加します。
(2)	行番号	右詰めゼロ・サプレスの 10 進数。
(3)	エラー・メッセージ番号	エラー番号を“*#nnnn*”の形式で出力します。 # は，内部エラーの場合は C，フェイタル・エラーの場合は E，アボート・エラーの場合は F，ワーニングの場合は W となります。 *nnnn* はエラー番号で，10 進数 4 桁で表します（ゼロ・サプレスしません）。
(4)	エラー・メッセージ	エラー・メッセージを出力します。
(5)	コンパイルの対象品種名	コマンド行オプション -c，またはソース・ファイル中において指定された対象品種名を表示します。
(6)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。
(7)	エラー個数	右詰めゼロ・サプレスの 10 進数。
(8)	ワーニング個数	右詰めゼロ・サプレスの 10 進数。

3. 1. 3　プリプロセス・リスト・ファイル

プリプロセス・リスト・ファイルは，C ソース・ファイルのプリプロセス処理のみを行った結果の ASCII イメージ・ファイルです。

-k オプションを指定する際，処理種別として“n”を指定しなければ，C ソース・ファイルとして使用することができます。-kd を指定すると，#define の展開を行ったリストを出力します。

CubeSuite+ におけるプリプロセス・リスト・ファイルの出力設定は，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［コンパイル・オプション］タブで行います。［プリプロセス・リスト・ファイル］カテゴリの［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］を選択してください。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。

PAGEWIDTH = 80 の場合は，以下のようになります。

```
/*
78K0R C Compiler V(1)x.xx Preprocess List                Date:(2)xx xxx xxxx  Page: (3)xxxx

Command   : (4)-cf1166a0 prime.c -p -lw80
In-file   : (5)prime.c
PPL-file  : (6)prime.ppl
Para-file : (7)
*/

    (8) 1 : (9)#define TRUE    1
    (8) 2 : (9)#define FALSE   0
    (8) 3 : (9)#define SIZE    20
    (8) 4 : (9)
    (8) 5 : (9)char    mark [SIZE+1];
    (8) 6 : (9)
/*
 (10)Target chip : uPD78F1166_A0
 (11)Device file : Vx.xx
*/
```

項番	内容	形式
(1)	バージョン番号	“*x.yz*”の形式で表します。
(2)	日付	システム日付。 “*DD Mmm YYYY*”の形式で表します。
(3)	ページ数	右詰めゼロ・サプレスの 10 進数。
(4)	コマンド行	コマンド行の“CC78K0R”以降を出力します。 1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(5)	C ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.c”を付加します。1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。
(6)	プリプロセス・リスト・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.ppl”を付加します。1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。
(7)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。 1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。ただし，1 カラム目に“;”を出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(8)	行番号	右詰めゼロ・サプレスの 10 進数。
(9)	C ソース	入力 C ソースです。 1 行に入らない場合は，超過分を次行の 9 カラム目から出力します。
(10)	コンパイルの対象品種名	コマンド行オプション -c，またはソース・ファイルにおいて指定された対象品種名を表示します。

項番	内容	形式
(11)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。

3. 1. 4　クロスリファレンス・リスト・ファイル

クロスリファレンス・リスト・ファイルは，C ソース・ファイル中で宣言，定義，参照されている関数名，変数名などの識別子の一覧表です。属性やその行番号などの情報も含んでいます。これらを出現順に出力します。

CubeSuite+ におけるクロスリファレンス・リスト・ファイルの出力設定は，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［コンパイル・オプション］タブで行います。［リスト・ファイル］カテゴリの［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］を選択してください。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。

PAGEWIDTH = 80 の場合は，以下のようになります。

```
78K0R C Compiler V(1)x.xx Cross reference List          Date:(2)xx xxx xxxx  Page: (3)xxxx

Command   : (4)-cf1166a0 prime.c -x -lw80
In-file   : (5)prime.c
Xref-file : (6)prime.xrf
Para-file : (7)
Inc-file  : (8)[n]

(9)ATTRIB (10)MODIFY (11)TYPE (12)SYMBOL (13)DEFINE   (14)REFERENCE

   EXTERN     NEAR       array    mark          5            14      16      22
   EXTERN     FAR        func     main          7
   AUTO1                 int      i             9            13      13      13      14
                                                             15      15      15      16
                                                             17      17      21
   AUTO1                 int      prime         9            17      18      21      21
   AUTO1                 int      k             9            21      21      21      22
   AUTO1                 int      count         9            11      19      20      25
  :
/*
 (15)Target chip : uPD78F1166_A0
 (16)Device file : Vx.xx
*/
```

項番	内容	形式
(1)	バージョン番号	“*x.yz*”の形式で表します。
(2)	日付	システム日付。 “*DD Mmm YYYY*”の形式で表します。

項番	内容	形式
(3)	ページ数	右詰めゼロ・サプレスの 10 進数。
(4)	コマンド行	コマンド行の“CC78K0R”以降を出力します。 1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(5)	C ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.c”を付加します。1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。
(6)	クロスリファレンス・リスト・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.xrf”を付加します。1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。
(7)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。 1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(8)	インクルード・ファイル	C ソース中で指定されたファイル名を出力します。 n は 1 で始まる数字でインクルード・ファイル番号を示します。1 行に入らない場合は，超過分を次行の 13 カラム目から出力します。インクルード・ファイルがない場合は，この行は出力されません。
(9)	シンボル属性	シンボル属性を表します。 外部変数は EXTERN，static 変数は外部が EXSTC で内部が INSTC，auto 変数は AUTOnn，レジスタ変数は REGnn（nn は 1 で始まる数字でスコープを示します），typedef 宣言は外部が EXTYP で内部が INTYP，ラベルは LABEL，構造体・共用体のタグは TAG，メンバは MEMBER，関数のパラメータは PARAM です。
(10)	シンボル修飾属性	シンボル修飾属性を表します。 左詰めです。 CONST（const 変数），VLT（volatile 変数），CALLT（callt 関数），SREG（sreg・bit 変数），RWSFR（sfr 変数），ROSFR（リード・オンリーの sfr 変数），WOSFR（ライト・オンリーの sfr 変数），VECT（割り込み関数），NEAR（near 領域配置の変数，関数），FAR（far 領域配置の変数，関数）の属性があります。
(11)	シンボル・タイプ	シンボルのタイプを表します。 char，int，short，long，field となります。unsigned タイプは，それぞれ先頭に u が付加されます。 他に void，float，double，ldouble（longdouble），func，array，pointer，struct，union，enum，bit，inter，#define があります。
(12)	シンボル名	15 文字を越えた場合は，1 行に収まるときはシンボル名をそのまま出力し，1 行に収まらないときは超過分を次行の 23 カラム目から出力し，（13），（14）の項目を次行の 39 カラム目から出力します。
(13)	シンボル定義行番号	シンボルの定義された行番号とファイル名で，行番号（5 桁）：インクルード・ファイル番号（2 桁）の形式で表します。
(14)	シンボル参照行番号	シンボルを参照している行番号とファイル名で行番号（5 桁）：インクルード・ファイル番号（2 桁）の形式で表します。1 行に入らないときは，超過分を次行の 48 カラム目から出力します。

項番	内容	形式
(15)	コンパイルの対象品種名	コマンド行オプション -c、またはソース・ファイルにおいて指定された対象品種名を表示します。
(16)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。

3. 2　アセンブラ

アセンブラは，次のリストを出力します。

出力リスト・ファイル名	出力リスト名
アセンブル・リスト・ファイル	アセンブル・リスト・ファイルのヘッダ
	アセンブル・リスト
	シンボル・リスト
	クロスリファレンス・リスト
エラー・リスト・ファイル	エラー・リスト

CubeSuite+ におけるアセンブル・リスト・ファイルの出力設定は，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［アセンブル・オプション］タブで行います。［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 (-p)］を選択してください。エラー・リスト・ファイルについては，［出力］カテゴリの［エラー・リスト・ファイルを出力する］プロパティで［はい (-e)］を選択してください。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。

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. 2. 1　アセンブル・リスト・ファイルのヘッダ

ヘッダ部は，常にアセンブル・リスト・ファイルの先頭に出力されます。

```
78K0R Assembler (1)Vx.xx (2)SAMPLE_TITLE     Date:(3)xx xxx xxxx  Page: (4)xxxx
(5)SAMPLE_SUBTITLE
Command:    (6) k0rmain.asm -cf1166a0
Para-file:  (7) -ks -kx
In-fine:    (8) k0rmain.asm
Obj-file:   (9) k0rmain.rel
Prn-file:   (10)k0rmain.prn
```

項番	内容	形式
(1)	アセンブラのバージョン番号	“*x.yz*”の形式で表します。
(2)	タイトル文字列	-lh オプション，または TITLE 制御命令によって指定された文字列です。
(3)	アセンブル・リストの作成年月日	アセンブル・リストの作成年月日。 “*DD Mmm YYYY*”の形式で表します。
(4)	ページ番号	右詰めゼロ・サプレスの 10 進数。
(5)	サブタイトル文字列	SUBTITLE 制御命令によって指定された文字列を表示します。
(6)	コマンド行	コマンド行を出力します。 1 行に入らない場合は，超過分を次行の 11 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。

項番	内容	形式
(7)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。 1 行に入らない場合は，超過分を次行の 11 カラム目から出力します。1 個以上の空白タブは，1 個の空白で置き換えます。
(8)	入力ソース・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.asm”を付加します。1 行に入らない場合は，超過分を次行の 11 カラム目から出力します。
(9)	出力オブジェクト・モジュール・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.rel”を付加します。1 行に入らない場合は，超過分を次行の 11 カラム目から出力します。
(10)	プリント・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.prn”を付加します。1 行に入らない場合は，超過分を次行の 11 カラム目から出力します。

3. 2. 2　アセンブル・リスト

アセンブル・リストは，アセンブル結果をエラー・メッセージ（エラーがある場合のみ）とともに出力します。

CubeSuite+ におけるアセンブル・リストの出力設定は，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［アセンブル・オプション］タブで行います。［アセンブル・リスト］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］を選択してください。

```
        Assemble list

(1)ALNO (2)STNO (6)ADRS   (8)OBJECT    (3)M (4)I   (5)SOURCE STATEMENT
      1       1
      2       2                                     NAME    SAMPM
      :
     31      31    0000A   RFD0000                  CALL  !CONVAH
                                                    ; convert ASCII <- HEX
     32      32                                     ; output BC-register <- ASCII code
     33      33    0000D    00000000                MOV   DE, #LOWW (STASC)
                                                    ; set DE <- store ASCII code table
                   00011    00
(7)*** ERROR E2202, STNO 33 ( 33) Illegal operand
     34      34    00012    63                      MOV   A, B
     35      35    00013    99                      MOV   [DE], A
      :
Segment informations :

(9)ADRS (10)LEN   (11)NAME

   FFE20    00003H    DATA
   00000    00002H    CODE
   00000    00019H    ?CSEG
```

```
 Target chip : (12)uPD78xxx
 Device file : (13)Vx.xx
Assembly complete, (14)1 error(s) and (15)0 warning(s) found. ( (16)33)
```

項番	内容	形式
(1)	ソースのイメージの行番号	右詰めゼロ・サプレスの 10 進数。
(2)	行番号	右詰めゼロ・サプレスの 10 進数。 INCLUDE ファイルの展開，マクロ展開も含みます。
(3)	マクロ表示	マクロを表示します。 - M：マクロ定義行です。 - #n：マクロ展開行です。n はネスト・レベルです。 - 空白：マクロ定義行／マクロ展開行ではありません。
(4)	INCLUDE 表示	INCLUDE を表示します。 - In：INCLUDE ファイル中です。n はネスト・レベルです。 - 空白：INCLUDE ファイル未使用
(5)	ソース・ステートメント	入力ソース・ステートメントを表示します。 1 行に入らない場合は，超過分を次行に出力します。
(6)	ロケーション・カウンタ値	機械命令，DB，DW，DS，DBIT，ラベルに対して，その行の先頭アドレスが表示されます。 ゼロ・サプレスなしの 16 進数。
(7)	エラーの発生行	エラーの発生行です。必要な項目が表示されます。
(8)	リロケーション情報	リロケーション情報を表示します。 - R：リンカによってオブジェクト・コード，またはシンボル値が変更されます。 - 空白：オブジェクト・コード，またはシンボル値が変更されません。
(9)	セグメント・アドレス	セグメントの先頭アドレスを表示します。 ゼロ・サプレスなしの 16 進数。
(10)	セグメント・サイズ	セグメントのサイズを表示します。 ゼロ・サプレスなしの 16 進数。
(11)	セグメント名	セグメント名を表示します。
(12)	アセンブラの対象デバイス	コマンド行オプションまたはソース・ファイルにおいて指定された対象デバイスを表示します。
(13)	デバイス・ファイルのバージョン番号	入力したデバイス・ファイルのバージョン番号を表示します。
(14)	フェイタル・エラーの個数	右詰めゼロ・サプレスの 10 進数。
(15)	ワーニングの個数	右詰めゼロ・サプレスの 10 進数。
(16)	最終エラー行	右詰めゼロ・サプレスの 10 進数。

3. 2. 3　シンボル・リスト

ソース内で定義されているシンボル（ローカル・シンボルを含む）の情報を出力します。

CubeSuite+ におけるシンボル・リストの出力設定は，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［アセンブル・オプション］タブで行います。［アセンブル・リスト］カテゴリの［シンボル・リストを出力する］プロパティで［はい (-ks)］を選択してください。

```
          Symbol Table List

(1)VALUE   (2)ATTR   (3)RTYP  (4)NAME            (1)VALUE   (2)ATTR   (3)RTYP  (4)NAME
             CSEG              ?CSEG                          CSEG              CODE
   -----H              EXT     CONVAH                         DSEG              DATA
   FFE20H    ADDR              HDTSA                    0H    ADDR      PUB     MAIN
             MOD               SAMPM                    0H    ADDR      PUB     START
   FFE21H    ADDR              STASC                -----H              EXT     _@STBEG
```

項番	内容	形式
(1)	シンボル値	シンボルの持つ値を表示します。 右詰めゼロ・サプレスの 16 進数。
(2)	シンボル属性	シンボル属性を表します。 左詰めです。 - CSEG：コード・セグメント名 - DSEG：データ・セグメント名 - BSEG：ビット・セグメント名 - MAC：マクロ名 - MOD：モジュール名 - SET：SET 疑似命令によって定義されたシンボル - NUM：NUMBER 属性シンボル - ADDR：ADDRESS 属性シンボル - BIT：BIT 属性シンボル（addr.bit） - SABIT：BIT 属性シンボル（saddr.bit） - SFBIT：BIT 属性シンボル（sfr.bit） - RBIT：BIT 属性シンボル（A.bit，X.bit，PSW.bit） - SFR：SFR を EQU 疑似命令で定義したネーム - SFRP：SFRP を EQU 疑似命令で定義したネーム - 空白：EXTRN，または EXTBIT 宣言された外部参照シンボル - *****：未定義シンボル
(3)	シンボル参照形式	シンボル参照形式を表します。 左詰めです。 - EXT：EXTRN 宣言された外部参照シンボル（SADDR 属性） - EXTB：EXTBIT 宣言された外部参照シンボル（saddr.bit） - PUB：PUBLIC 宣言された外部定義シンボル - 空白：ローカル・シンボル，セグメント名，マクロ名，モジュール名 - *****：未定義シンボル
(4)	定義されたシンボル名	定義されたシンボル名を表示します。 左詰めです。

3. 2. 4　クロスリファレンス・リスト

ソース内で定義されたシンボルがソースのどこで（行番号）参照されているかという情報が出力されます。

CubeSuite+ におけるクロスリファレンス・リストの出力設定は，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［アセンブル・オプション］タブで行います。［アセンブル・リスト］カテゴリの［クロスリファレンス・リストを出力する］プロパティで［はい (-kx)］を選択してください。

```
          Cross-Reference List

(1)NAME      (2)VALUE    (3)R (4)ATTR  (5)RTYP (6)SEGNAME (7)XREFS
   ?CSEG                       CSEG             ?CSEG        22#
   CODE                        CSEG             CODE         19#
   CONVAH       -----H     E            EXT                  12@    31
   DATA                        DSEG             DATA         15#
   HDTSA        FFE20H         ADDR             DATA         16#    28    29
   MAIN             0H         ADDR     PUB     CODE         11@    20#
   SAMPM                       MOD                            2#
   START            0H     R   ADDR     PUB     ?CSEG        11@    20    23#
   STASC        FFE21H         ADDR             DATA         17#    33
   _@STBEG      -----H     E            EXT                  13@    26
```

項番	内容	形式
(1)	定義されたシンボル名	定義されたシンボル名を表示します。 左詰めです。 16 文字を越えた場合は，シンボル名をそのまま出力し，（2），（4），（5），（6），（7），（8）の項目を次行から出力します。
(2)	シンボル値	シンボルの持つ値を表示します。 右詰めゼロ・サプレスの 16 進数。
(3)	リロケーション属性	リロケーション属性を表示します。 - R：リロケータブルなシンボル - E：external なシンボル - 空白：アブソリュートなシンボル - *：未定義シンボル

項番	内容	形式
(4)	シンボル属性	シンボル属性を表します。 左詰めです。 - CSEG：コード・セグメント名 - DSEG：データ・セグメント名 - BSEG：ビット・セグメント名 - MAC：マクロ名 - MOD：モジュール名 - SET：SET 疑似命令によって定義されたシンボル - NUM：NUMBER 属性シンボル - ADDR：ADDRESS 属性シンボル - BIT：BIT 属性シンボル（addr.bit） - SABIT：BIT 属性シンボル（saddr.bit） - SFBIT：BIT 属性シンボル（sfr.bit） - RBIT：BIT 属性シンボル（A.bit，X.bit） - SFR：SFR を EQU 疑似命令で定義したネーム - SFRP：SFRP を EQU 疑似命令で定義したネーム - 空白：EXTRN，または EXTBIT 宣言された外部参照シンボル - ****：未定義シンボル
(5)	シンボル参照形式	シンボル参照形式を表します。 左詰めです。 - EXT：EXTRN 宣言された外部参照シンボル（SADDR 属性） - EXTB：EXTBIT 宣言された外部参照シンボル（saddr.bit） - PUB：PUBLIC 宣言された外部定義シンボル - 空白：ローカル・シンボル，セグメント名，マクロ名，モジュール名 - *****：未定義シンボル
(6)	定義されたセグメント名	シンボルが定義されたセグメント名を表示します。 左詰めです。
(7)	定義・参照行番号	定義・参照行番号を表示します。 - 定義行：xxxxx# - 参照行：xxxxx △　（△は空白 1 つ） - EXTRN 宣言，EXTBIT 宣言，PUBLIC 宣言：xxxxx@

3. 2. 5　エラー・リスト

アセンブラ起動時に出力されたエラー・メッセージが格納されています。

```
PASS1 Start
(1)ERROR.ASM((2)26) : RA78K0R (3)error (4)E2202: (5)Illegal operand
(1)ERROR.ASM((2)32) : RA78K0R (3)error (4)E2202: (5)Illegal operand
PASS2 Start
(1)ERROR.ASM((2)26) : RA78K0R (3)error (4)E2202: (5)Illegal operand
(1)ERROR.ASM((2)29) : RA78K0R (3)error (4)E2407: (5)Undefined symbol reference 'DTSA'
(1)ERROR.ASM((2)29) : RA78K0R (3)error (4)E2303: (5)Illegal expression
(1)ERROR.ASM((2)32) : RA78K0R (3)error (4)E2202: (5)Illegal operand
(1)ERROR.ASM((2)37) : RA78K0R (3)error (4)E2407: (5)Undefined symbol reference 'F'
(1)ERROR.ASM((2)37) : RA78K0R (3)error (4)E2303: (5)Illegal expression
```

項番	内容	形式
(1)	エラーの発生したソース・ファイル名	エラーの発生したソース・ファイル名を出力します。
(2)	エラーの発生行	左詰めゼロ・サプレス
(3)	エラーの種類	エラーの種類を表示します。
(4)	エラー番号	エラー番号を“#*mnnn*”の形式で出力します。 #は，内部エラーの場合はC，フェイタル・エラーの場合はE，アボート・エラーの場合はF，ワーニングの場合はWとなります。 *m*は，アセンブラの場合は2，リンカの場合は3，オブジェクト・コンバータの場合は4，ライブラリアンの場合は5，リスト・コンバータの場合は6となります。 *nnn*はエラー番号です。
(5)	エラー・メッセージ	エラー・メッセージを出力します。

備考　ファイル名，エラーの発生行は，表示されない場合もあります。

3.3　リンカ

リンカは，次のリストを出力します。

出力リスト・ファイル名	出力リスト名
リンク・リスト・ファイル	リンク・リスト・ファイルのヘッダ
	マップ・リスト
	パブリック・シンボル・リスト
	ローカル・シンボル・リスト
エラー・リスト・ファイル	エラー・リスト

CubeSuite+ におけるリンク・リスト・ファイルの出力設定は，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［リンク・オプション］タブで行います。［リンク・リスト］カテゴリの［リンク・リスト・ファイルを出力する］プロパティで［はい］を選択してください。エラー・リスト・ファイルについては，［エラー・リスト］カテゴリの［エラー・リストを出力する］プロパティで［はい (-e)］を選択してください。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　リンカの入出力ファイルについては，「B. 3. 1　入出力ファイル」を参照してください。

3.3.1　リンク・リスト・ファイルのヘッダ

ヘッダ部は，常にリンク・リスト・ファイルの先頭に出力されます。

```
78K0R Linker (1)Vx.xx                                    Date:(2)xx xxx xxxx  Page: (3)xxxx

Command:     (4)k0rmain.rel k0rsub.rel -s -ok0r.map -dk0r.dr
Para-file:   (5)
Out-file:    (6)k0rmain.lmf
Map-File:    (7)k0r.map
Direc-File:  (8)k0r.dr
Directive:   (9)MEMORY ROM  : (0H, 0ED800H)
             (9)MEMORY RAM1 : (0FCF00H, 1100H)
             (9)MEMORY RAM  : (0FE000H, 1F00H)

*** Link information ***

(10)      6 output segment(s)
(11)    9DH byte(s) real data
(12)     40 symbol(s) defined
```

項番	内容	形式
(1)	リンカのバージョン番号	“*x.yz*”の形式で表します。

項番	内容	形式
(2)	リンク・リスト・ファイルの作成年月日	リンク・リスト・ファイルの作成年月日。 “*DD Mmm YYYY*”の形式で表します。
(3)	ページ番号	右詰めゼロ・サプレスの 10 進数。
(4)	コマンド行のイメージ	起動行に指定されたオプションを表示します。
(5)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。
(6)	出力ロード・モジュール・ファイル名	リンカが生成するロード・モジュール・ファイル名を出力します。
(7)	リンク・リスト・ファイル名	リンカが生成するリンク・リスト・ファイル名を出力します。
(8)	リンク・ディレクティブ・ファイル名	リンカが入力するリンク・ディレクティブ・ファイル名を出力します。
(9)	リンク・ディレクティブ・ファイルの内容	リンク・ディレクティブ・ファイルの内容を表示します。
(10)	ロード・モジュール・ファイルに出力されるセグメント数	ロード・モジュール・ファイルに出力されるセグメント数を表示します。 右詰めゼロ・サプレスの 10 進数。
(11)	ロード・モジュール・ファイルに出力されるデータの大きさ	ロード・モジュール・ファイルに出力されるデータの大きさを表示します。 右詰めゼロ・サプレスの 10 進数。
(12)	ロード・モジュール・ファイルに出力されるシンボル数	ロード・モジュール・ファイルに出力されるシンボル数を表示します。 右詰めゼロ・サプレスの 10 進数。

3. 3. 2　マップ・リスト

セグメントの配置に関する情報を出力します。

CubeSuite+ におけるマップ・リストの出力設定は，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［リンク・オプション］タブで行います。［リンク・リスト］カテゴリの［マップ・リストを出力する］プロパティで［はい］を選択してください。

```
*** Memory map ***

  (1)SPACE=REGULAR

  MEMORY=(2)ROM
  BASE ADDRESS=(3)00000H    SIZE=(4)ED800H
        (6)OUTPUT    (7)INPUT     (8)INPUT     (9)BASE      (10)SIZE
           SEGMENT      SEGMENT      MODULE       ADDRESS
           CODE                                  00000H       00002H    (11)CSEG  AT
                        CODE         SAMPM        00000H       00002H
(5)  * gap *                                     00002H       000BEH
           ?CSEGOB0                              000C0H       00004H    (11)CSEG  OPT_BYTE
           ?CSEG                                 000C4H       00059H    (11)CSEG
                        ?CSEG        SAMPM        000C4H       00017H
```

```
                        ?CSEG        SAMPS        000DBH        00042H
(5)  * gap *                                      0011DH        ED6E3H


  MEMORY=RAM1
  BASE ADDRESS=(3)FCF00H    SIZE=(4)01100H
        (6)OUTPUT    (7)INPUT     (8)INPUT     (9)BASE      (10)SIZE
           SEGMENT      SEGMENT      MODULE       ADDRESS
(5)  * gap *                                      FCF00H        01100H

  MEMORY=RAM
  BASE ADDRESS=(3)FE000H    SIZE=(4)01F00H
        (6)OUTPUT    (7)INPUT     (8)INPUT     (9)BASE      (10)SIZE
           SEGMENT      SEGMENT      MODULE       ADDRESS
(5)  * gap *                                      FE000H        01E20H
           DATA                                   FFE20H        00003H    (11)DSEG  AT
                        DATA         SAMPM        FFE20H        00003H
(5)  * gap *                                      FFE23H        000DDH

 Target chip : (12)uPD78xxx
 Device File : (13)Vx.xx
```

項番	内容	形式
(1)	メモリ空間名	メモリ空間名を表示します。
(2)	メモリ領域名	メモリ領域名を表示します。
(3)	メモリ領域の先頭アドレス	メモリ領域の先頭アドレスを表示します。 右詰め 0 パティングの 16 進数。
(4)	メモリ領域のサイズ	メモリ領域のサイズを表示します。 右詰め 0 パティングの 16 進数。
(5)	出力グループ	何も配置されていないエリアがある場合，“gap”を表示します。
(6)	ロード・モジュール・ファイルに出力されるセグメント名	ロード・モジュール・ファイルに出力されるセグメント名を表示します。
(7)	オブジェクト・モジュール・ファイルから読み込まれたセグメント名	オブジェクト・モジュール・ファイルから読み込まれた入力セグメント名を表示します。
(8)	入力モジュール名	(7) で表示した入力セグメントが存在していた入力ファイルのモジュール名を表示します。 8 文字を越えた場合は，入力モジュール名をそのまま出力し，(9)，(10)，(11) の項目を次行の 39 カラム目から出力します。
(9)	セグメントの先頭アドレス	出力セグメントが配置された先頭アドレスを表示します。
(10)	出力セグメントのサイズ	出力セグメントのサイズを表示します。
(11)	セグメント・タイプ，再配置属性	出力セグメントのセグメント・タイプ，再配置属性を表示します。

項番	内容	形式
(12)	アセンブルの対象製品	コマンド行オプション -c，またはソース・ファイルにおいて指定された対象品種名を表示します。
(13)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。

3. 3. 3　パブリック・シンボル・リスト

入力モジュール内で定義されているパブリック・シンボルの情報を出力します。

CubeSuite+ におけるパブリック・シンボル・リストの出力設定は，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［リンク・オプション］タブで行います。［リンク・リスト］カテゴリの［パブリック・シンボル・リストを出力する］プロパティで［はい (-kp)］を選択してください。

```
*** Public symbol list ***

(1)MODULE   (2)ATTR   (3)VALUE     (4)NAME

   SAMPM
              ADDR      00000H      MAIN
              ADDR      000D2H      START
   SAMPS
              ADDR      000E9H      CONVAH

              NUM       FFE20H      _@STBEG

              NUM       FE000H      _@STEND
```

項番	内容	形式
(1)	パブリック・シンボルが定義されたモジュール名	パブリック・シンボルが定義された入力オブジェクト・モジュール名を表示します。

項番	内容	形式
(2)	シンボル属性	シンボル属性を表示します。 - CSEG：コード・セグメント名 - DSEG：データ・セグメント名 - BSEG：ビット・セグメント名 - MAC：マクロ名 - MOD：モジュール名 - SET：SET 疑似命令によって定義されたシンボル - NUM：NUMBER 属性シンボル - ADDR：ADDRESS 属性シンボル - BIT：BIT 属性シンボル（addr.bit） - SABIT：BIT 属性シンボル（saddr.bit） - SFBIT：BIT 属性シンボル（sfr.bit） - RBIT：BIT 属性シンボル（A.bit，X.bit，PSW.bit） - SFR：SFR を EQU 疑似命令で定義したネーム - SFRP：SFRP を EQU 疑似命令で定義したネーム - 空白：EXTRN，または EXTBIT 宣言された外部参照シンボル - *****：未定義シンボル
(3)	シンボル値	パブリック・シンボルの値を表示します。
(4)	パブリック・シンボル名	パブリック・シンボル名を表示します。

3. 3. 4　ローカル・シンボル・リスト

入力モジュール内で定義されているローカル・シンボルの情報を出力します。

CubeSuite+ におけるローカル・シンボル・リストの出力設定は，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［リンク・オプション］タブで行います。［リンク・リスト］カテゴリの［ローカル・シンボル・リストを出力する］プロパティで［はい (-kl)］を選択してください。

```
*** Local symbol list ***

(1)MODULE   (2)ATTR   (3)VALUE     (4)NAME

  SAMPM
              MOD                   SAMPM
              DSEG                  DATA
              ADDR      FFE20H      HDTSA
              ADDR      FFE21H      STASC
              CSEG                  CODE
              CSEG                  ?CSEG
  SAMPS
              MOD                   SAMPS
              CSEG                  ?CSEG
              ADDR      00114H      SASC
              ADDR      0011AH      SASC1
```

項番	内容	形式
(1)	ローカル・シンボルが定義されたモジュール名	ローカル・シンボルが定義された入力オブジェクト・モジュール名を表示します。
(2)	シンボル属性	シンボル属性を表示します。 - CSEG：コード・セグメント名 - DSEG：データ・セグメント名 - BSEG：ビット・セグメント名 - MAC：マクロ名 - MOD：モジュール名 - SET：SET 疑似命令によって定義されたシンボル - NUM：NUMBER 属性シンボル - ADDR：ADDRESS 属性シンボル - BIT：BIT 属性シンボル（addr.bit） - SABIT：BIT 属性シンボル（saddr.bit） - SFBIT：BIT 属性シンボル（sfr.bit） - RBIT：BIT 属性シンボル（A.bit，X.bit，PSW.bit） - SFR：SFR を EQU 疑似命令で定義したネーム - SFRP：SFRP を EQU 疑似命令で定義したネーム - 空白：EXTRN，または EXTBIT 宣言された外部参照シンボル - *****：未定義シンボル
(3)	シンボル値	ローカル・シンボルの値を表示します。
(4)	ローカル・シンボル名	ローカル・シンボル名を表示します。

3. 3. 5　エラー・リスト

リンカ起動時に出力されたエラー・メッセージが格納されています。

```
LK78K0R (1)error (2)E3405: (3)Undefined symbol 'CONVAH' in file 'k0rmain.rel'
```

項番	内容	形式
(1)	エラーの種類	エラーの種類を表示します。
(2)	エラー番号	エラー番号を“*#nnnn*”の形式で出力します。 # は，内部エラーの場合は C，フェイタル・エラーの場合は E，アボート・エラーの場合は F，ワーニングの場合は W となります。 *nnnn* はエラー番号で，10 進数 4 桁で表します（ゼロ・サプレスしません）。
(3)	エラー・メッセージ	エラー・メッセージを出力します。

3. 4　ROM 化プロセッサ

ROM 化プロセッサは，次のリストを出力します。

出力リスト・ファイル名	出力リスト名
リンク・マップ・ファイル	リンク・マップ・ファイルのヘッダ
	マップ・リスト
	パブリック・シンボル・リスト
	ローカル・シンボル・リスト
エラー・リスト・ファイル	エラー・リスト

CubeSuite+ におけるリンク・マップ・ファイルの出力設定は，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［ROM 化プロセス・オプション］タブで行います。［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］を選択してください。エラー・リスト・ファイルについては，［エラー・リスト］カテゴリの［エラー・リストを出力する］プロパティで［はい (-e)］を選択してください。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　ROM 化プロセッサの入出力ファイルについては，「B. 4. 1　入出力ファイル」を参照してください。

3. 4. 1　リンク・マップ・ファイルのヘッダ

ヘッダ部は，常にリンク・マップ・ファイルの先頭に出力されます。

```
78K0R ROM Processor (1)Vx.xx                          Date:(2)xx xxx xxxx  Page: (3)xxxx

Command:     (4)a.lmf -rc300h -km
Para-file:   (5)
Out-file:    (6)a.lmf
Map-File:    (7)a_romp.map


*** Link information ***

(8)      6 output segment(s)
(9)    F7H byte(s) real data
(10)     78 symbol(s) defined
```

項番	内容	形式
(1)	ROM 化プロセッサのバージョン番号	“*x.yz*”の形式で表します。
(2)	リンク・マップ・ファイルの作成年月日	リンク・マップ・ファイルの作成年月日。 “*DD Mmm YYYY*”の形式で表します。

項番	内容	形式
(3)	ページ番号	右詰めゼロ・サプレスの 10 進数。
(4)	コマンド行のイメージ	起動行に指定されたオプションを表示します。
(5)	パラメータ・ファイルの内容	パラメータ・ファイルの内容を出力します。
(6)	出力ロード・モジュール・ファイル名	ROM 化プロセッサが生成するロード・モジュール・ファイル名を出力します。
(7)	リンク・マップ・ファイル名	ROM 化プロセッサが生成するリンク・マップ・ファイル名を出力します。
(8)	ロード・モジュール・ファイルに出力されるセグメント数	ロード・モジュール・ファイルに出力されるセグメント数を表示します。 右詰めゼロ・サプレスの 10 進数。
(9)	ロード・モジュール・ファイルに出力されるデータの大きさ	ロード・モジュール・ファイルに出力されるデータの大きさを表示します。 右詰めゼロ・サプレスの 10 進数。
(10)	ロード・モジュール・ファイルに出力されるシンボル数	ロード・モジュール・ファイルに出力されるシンボル数を表示します。 右詰めゼロ・サプレスの 10 進数。

3. 4. 2　マップ・リスト

セグメントの配置に関する情報を出力します。

CubeSuite+ におけるマップ・リストの出力設定は，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［ROM 化プロセス・オプション］タブで行います。［リンク・マップ］カテゴリの［マップ・リストを出力する］プロパティで［はい］を選択してください。

```
*** Memory map ***



  (1)SPACE=REGULAR


        (3)OUTPUT    (4)INPUT     (5)INPUT     (6)BASE      (7)SIZE
           SEGMENT      SEGMENT      MODULE       ADDRESS
           CODE                                  00000H       00002H        (8)CSEG AT
                        CODE         a.lmf
                                                 00000H       00002H
(2)  * gap *                                     00002H       000BEH
           ?CSEGOB0                              000C0H       00004H        (8)CSEG AT
                        ?CSEGOB0     a.lmf
                                                 000C0H       00004H
           ?CSEG                                 000C4H       00061H        (8)CSEG AT
                        ?CSEG        a.lmf
                                                 000C4H       00061H
(2)* gap *                                       00125H       001DBH
           @@ROMP                                00300H       00090H        (8)CSEG AT
                        _@ROMP       _rcopy      00300H       0008EH
                        PRO_RAM      a.lmf
                                                 0038EH       00002H
```

```
(2)* gap *                                    00390H      3FC70H

        (3)OUTPUT    (4)INPUT     (5)INPUT     (6)BASE     (7)SIZE
           SEGMENT      SEGMENT      MODULE      ADDRESS
(2)* gap *                                    FCF00H      02F20H
           DATA                               FFE20H      00003H      (8)DSEG AT
                        DATA         a.lmf
                                              FFE20H      00003H
(2)* gap *                                    FFE23H      0000DH
           PRO_RAM                            FFE30H      00002H      (8)CSEG AT
                        PRO_RAM      a.lmf
                                              FFE30H      00002H
(2)* gap *                                    FFE32H      000CEH
(2)* gap (Not Free Area) *                    FFF00H      00100H


*** Segment number ***


        (9)SEGMENT NAME                                  (10)SEGMENT NUMBER
           PRO_RAM                                           1


 Target chip : (11)uPD78xxx
 Device file : (12)Vx.xx
```

項番	内容	形式
(1)	メモリ空間名	メモリ空間名を表示します。
(2)	出力グループ	何も配置されていないエリアがある場合，“gap”を表示します。
(3)	ロード・モジュール・ファイルに出力されるセグメント名	ロード・モジュール・ファイルに出力されるセグメント名を表示します。
(4)	入力ロード・モジュール・ファイルから読み込まれたセグメント名	入力ロード・モジュール・ファイルから読み込まれた入力セグメント名を表示します。
(5)	入力ロード・モジュール・モジュール名	(4) で表示した入力セグメントが存在していた入力ロード・モジュール・ファイルのモジュール名を表示します。 8 文字を越えた場合は，入力ロード・モジュール・モジュール名をそのまま出力し，(6)，(7)，(8) の項目を次行の 39 カラム目から出力します。
(6)	セグメントの先頭アドレス	出力セグメントが配置された先頭アドレスを表示します。
(7)	出力セグメントのサイズ	出力セグメントのサイズを表示します。
(8)	セグメント・タイプ，再配置属性	出力セグメントのセグメント・タイプ，再配置属性を表示します。
(9)	セグメント名	ROM 化したセグメント名を表示します。

項番	内容	形式
(10)	セグメント番号	展開時に指定するセグメント番号（_rcopy() の引数）を表示します。 セグメント番号は 1 から始まる整数で，セグメントが入力ファイル中に出現した順番に割り当てます。
(11)	アセンブルの対象製品	コマンド行オプション -c，またはソース・ファイルにおいて指定された対象品種名を表示します。
(12)	デバイス・ファイル・バージョン	入力したデバイス・ファイルのバージョン番号を表示します。

3. 4. 3　パブリック・シンボル・リスト

入力モジュール内で定義されているパブリック・シンボルの情報を出力します。

CubeSuite+ におけるパブリック・シンボル・リストの出力設定は，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［ROM 化プロセス・オプション］タブで行います。［リンク・マップ］カテゴリの［パブリック・シンボル・リストを出力する］プロパティで［はい (-kp)］を選択してください。

```
*** Public symbol list ***

(1)MODULE    (2)ATTR      (3)VALUE     (4)NAME

  k0rram
              ADDR         00000H       MAIN
              ADDR         000C4H       START
              ADDR         000E3H       CONVAH
              NUM          FFE20H       _@STBEG
              NUM          00300H       __rcopy
              ADDR         FFE30H       pro_ram
              NUM          FCF00H       _@STEND
              NUM          00000H       _@MAA
  _rcopy
              ADDR         00300H       _@RCP
              NUM          0038EH       _S_romp
```

項番	内容	形式
(1)	パブリック・シンボルが定義されたモジュール名	パブリック・シンボルが定義された入力オブジェクト・モジュール名を表示します。

項番	内容	形式
(2)	シンボル属性	シンボル属性を表示します。 - CSEG：コード・セグメント名 - DSEG：データ・セグメント名 - BSEG：ビット・セグメント名 - MAC：マクロ名 - MOD：モジュール名 - SET：SET 疑似命令によって定義されたシンボル - NUM：NUMBER 属性シンボル - ADDR：ADDRESS 属性シンボル - BIT：BIT 属性シンボル（addr.bit） - SABIT：BIT 属性シンボル（saddr.bit） - SFBIT：BIT 属性シンボル（sfr.bit） - RBIT：BIT 属性シンボル（A.bit，X.bit，PSW.bit） - SFR：SFR を EQU 疑似命令で定義したネーム - SFRP：SFRP を EQU 疑似命令で定義したネーム - 空白：EXTRN，または EXTBIT 宣言された外部参照シンボル - *****：未定義シンボル
(3)	シンボル値	パブリック・シンボルの値を表示します。
(4)	パブリック・シンボル名	パブリック・シンボル名を表示します。

3. 4. 4　ローカル・シンボル・リスト

入力モジュール内で定義されているローカル・シンボルの情報を出力します。

CubeSuite+ におけるローカル・シンボル・リストの出力設定は，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［ROM 化プロセス・オプション］タブで行います。［リンク・マップ］カテゴリの［ローカル・シンボル・リストを出力する］プロパティで［はい (-kl)］を選択してください。

```
*** Local symbol list ***

(1)MODULE    (2)ATTR      (3)VALUE     (4)NAME

  k0rram
              MOD                       SAMPM
              DSEG                      DATA
              ADDR         FFE20H       HDTSA
              ADDR         FFE21H       STASC
              CSEG                      CODE
              CSEG                      ?CSEG
  k0rram
              MOD                       SAMPS
              CSEG                      ?CSEG
              ADDR         0011CH       SASC
              ADDR         00122H       SASC1
```

```
k0rram
            MOD                     k0rram
            CSEG                    PRO_RAM
_rcopy
            MOD                     _rcopy
                        00300H      _@ROMP
            ADDR        00384H      ?L_RCPC
            ADDR        0037BH      ?L_RCP9
            ADDR        00386H      ?L_RCPA
            ADDR        00318H      ?L_RCP1
            ADDR        0032CH      ?L_RCP2
            ADDR        0033BH      ?L_RCP3
            ADDR        0034AH      ?L_RCP4
            ADDR        00351H      ?L_RCP5
            ADDR        00360H      ?L_RCP6
            ADDR        00372H      ?L_RCP7
            ADDR        00372H      ?L_RCP8
```

項番	内容	形式
(1)	ローカル・シンボルが定義されたモジュール名	ローカル・シンボルが定義された入力オブジェクト・モジュール名を表示します。
(2)	シンボル属性	シンボル属性を表示します。 - CSEG：コード・セグメント名 - DSEG：データ・セグメント名 - BSEG：ビット・セグメント名 - MAC：マクロ名 - MOD：モジュール名 - SET：SET 疑似命令によって定義されたシンボル - NUM：NUMBER 属性シンボル - ADDR：ADDRESS 属性シンボル - BIT：BIT 属性シンボル（addr.bit） - SABIT：BIT 属性シンボル（saddr.bit） - SFBIT：BIT 属性シンボル（sfr.bit） - RBIT：BIT 属性シンボル（A.bit，X.bit，PSW.bit） - SFR：SFR を EQU 疑似命令で定義したネーム - SFRP：SFRP を EQU 疑似命令で定義したネーム - 空白：EXTRN，または EXTBIT 宣言された外部参照シンボル - *****：未定義シンボル
(3)	シンボル値	ローカル・シンボルの値を表示します。
(4)	ローカル・シンボル名	ローカル・シンボル名を表示します。

3. 4. 5　エラー・リスト

ROM 化プロセッサ起動時に出力されたエラー・メッセージが格納されています。

```
RP78K0R (1)error (2)E8405: (3)Undefined symbol 'CONVAH' in file 'a.lmf'
```

項番	内容	形式
(1)	エラーの種類	エラーの種類を表示します。
(2)	エラー番号	エラー番号を“*#nnnn*”の形式で出力します。 # は，内部エラーの場合は C，フェイタル・エラーの場合は E，アボート・エラーの場合は F，ワーニングの場合は W となります。 *nnnn* はエラー番号で，10 進数 4 桁で表します（ゼロ・サプレスしません）。
(3)	エラー・メッセージ	エラー・メッセージを出力します。

3. 5　オブジェクト・コンバータ

オブジェクト・コンバータは，次のリストを出力します。

出力リスト・ファイル名	出力リスト名
エラー・リスト・ファイル	エラー・リスト

CubeSuite+ におけるエラー・リスト・ファイルの出力設定は，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［オブジェクト・コンバート・オプション］タブで行います。［エラー・リスト］カテゴリの［エラー・リスト・ファイルを出力する］プロパティで［はい (-e)］を選択してください。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　オブジェクト・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. 5. 1　エラー・リスト

オブジェクト・コンバータ起動時に出力されたエラー・メッセージが格納されています。

出力形式は，リンカの出力するエラー・リストと同一です。

3. 6　ライブラリアン

ライブラリアンは，次のリストを出力します。

出力リスト・ファイル名	出力リスト名
リスト・ファイル	ライブラリ情報出力リスト

CubeSuite+ におけるリスト・ファイルの出力設定は，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［ライブラリ生成オプション］タブで行います。［リスト・ファイル］カテゴリの［リスト・ファイルを出力する］プロパティで［はい］を選択してください。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　ライブラリアンの入出力ファイルについては，「B. 6. 1　入出力ファイル」を参照してください。

3. 6. 1　ライブラリ情報出力リスト

ライブラリ・ファイル内のモジュールに関する情報を出力します。

```
78K0R librarian (1)Vx.xx                                Date:(2)xx xxx xxxx   Page(3)xxxx

LIB-FILE NAME : (4)k0r.lib          ((5)xx xxx xxxx)

  (6)0001   (7)k0rmain.rel      ((8)xx xxx xxxx)

    (9)MAIN     (9)START

  NUMBER OF PUBLIC SYMBOLS : (10)2

  (6)0002   (7)k0rsub.rel       ((8)xx xxx xxxx)

    (9)CONVAH

  NUMBER OF PUBLIC SYMBOLS : (10)1
```

項番	内容	形式
(1)	ライブラリアンのバージョン番号	“*x.yz*”の形式で表します。
(2)	リストの作成年月日	リストの作成年月日。 “*DD Mmm YYYY*”の形式で表します。
(3)	ページ数	右詰めゼロ・サプレスの 10 進数。
(4)	ライブラリ・ファイル名	指定されたファイル名を出力します。 ファイル・タイプが省略されている場合は，“.lib”を付加します。
(5)	ライブラリ・ファイルの作成年月日	ライブラリ・ファイルの作成年月日。 “*DD Mmm YYYY*”の形式で表します。

項番	内容	形式
(6)	モジュールの通番	0001 から番号を付けます。
(7)	モジュール名	モジュール名を表示します。 ファイル・タイプが省略されている場合は，“.rel”を付加します。
(8)	モジュール作成年月日	モジュール作成年月日。 “*DD Mmm YYYY*”の形式で表します。
(9)	パブリック・シンボル名	パブリック・シンボル名を表示します。
(10)	モジュール内で定義されているパブリック・シンボル数	モジュール内で定義されているパブリック・シンボル数を表示します。 右詰めゼロ・サプレスの 10 進数。

3. 7　リスト・コンバータ

リスト・コンバータは，次のリストを出力します。

出力リスト・ファイル名	出力リスト名
アブソリュート・アセンブル・リスト・ファイル	アブソリュート・アセンブル・リスト
エラー・リスト・ファイル	エラー・リスト

CubeSuite+ におけるアブソリュート・アセンブル・リスト・ファイルの出力設定は，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［アセンブル・オプション］タブで行います。［アセンブル・リスト］カテゴリの［リスト・コンバータを実行する］プロパティで［はい］を選択してください。エラー・リスト・ファイルについては，［アセンブル・リスト］カテゴリの［リスト・コンバータのエラー・リスト・ファイルを出力する］プロパティで［はい (-e)］を選択してください。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。

備考　リスト・コンバータの入出力ファイルについては，「B. 7. 1　入出力ファイル」を参照してください。

3. 7. 1　アブソリュート・アセンブル・リスト

アセンブル・リストにアブソリュートな値を埋め込んで出力します。

出力形式は，アセンブラの出力するアセンブル・リストと同一です。

3. 7. 2　エラー・リスト

リスト・コンバータ起動時に出力されたエラー・メッセージが格納されています。

出力形式は，アセンブラの出力するエラー・リストと同一です。

3. 8　変数／関数情報ファイル生成ツール

変数／関数情報ファイル生成ツールは，次のファイルを出力します。

- 変数／関数情報ファイル

備考　変数／関数情報ファイル生成ツールの入出力ファイルについては，「B. 8. 1　入出力ファイル」を参照してください。

3. 8. 1　変数／関数情報ファイル

変数／関数情報ファイルは，変数や関数を効率よく配置するための情報をまとめたファイルです。

CubeSuite+ における変数／関数情報ファイルの出力設定は，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［変数／関数配置オプション］タブで行います。［出力ファイル］カテゴリの［変数／関数情報ファイルを出力する］プロパティで［はい］を選択してください。出力先は，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］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。

```
;VF78K0R (1)Vx.xx
; Attention:The semicolon at the head of line means the line is a comment.
;           Please refer to the "format information" for the item of each section.
;(2)*** format information ***
;[sreg]
;variable,count,size,type,"file",const  ;static-const
;variable,count,size,type,,const        ;global-const
;variable,count,size,type,"file"        ;static
;variable,count,size,type               ;global
;variable,count,size,type,,const,boot   ;global-const in boot
;variable,count,size,type,,,boot        ;global in boot
;;type : near=1 , far=2 , sreg=0
;
;[callt]
;variable,count,type,"file"     ;static
;variable,count,type            ;global
;variable,count,type,,boot      ;global in boot
;;type : near=1 , far=2 , callt=0
;
;(3)*** gap information ***
;[callt-gap]
;(4)START     (5)SIZE
;   00080H       00040H
;[base-gap]
;(4)START     (5)SIZE
;   00190H       00E70H
;   01004H       00FFCH
;   02004H       00008H
```

```
;    02018H        000ECH
;    0210CH        03EF4H
;    060F7H        00001H
;    06100H        09EFCH
;[saddr-gap]
;(4)START      (5)SIZE
;    FFE26H        000BAH
;
;(6)*** variable information ***
[sreg]
(7)f,(8)3,(9)1,(10)1
(7)flash_a,(8)2,(9)2,(10)1
(7)flash_b,(8)2,(9)2,(10)1
;(7)var1,(8)1,(9)2,(10)1,(11)"flash.c",(12)const
;(7)var2,(8)1,(9)2,(10)1,,const
(7)var3,(8)1,(9)4,(10)1,(11)"flash.c"
;(7)boot_a,(8)1,(9)2,(10)0,,,(13)boot
;(7)boot_b,(8)1,(9)2,(10)0,,,(13)boot
;
;(14)*** function information ***
[callt]
;(15)f1,(16)1,(17)1,(18)"flash.c"
;(15)f2,(16)1,(17)1
;(15)func,(16)1,(17)1,,(19)boot
```

項番	内容	形式
(1)	バージョン番号	“*x.yz*”の形式で表します。
(2)	フォーマット情報（開始）	変数情報，関数情報のフォーマット情報の開始を示します。
(3)	空き領域情報（開始）	saddr 領域，BASE 領域，callt 領域の空き領域情報の開始を示します。 行頭にセミコロンを付加し，コメントとします。
(4)	空き領域情報（スタート・アドレス）	空き領域のスタート・アドレスを出力します。
(5)	空き領域情報（サイズ）	空き領域のサイズを出力します。
(6)	変数情報（開始）	変数情報の開始を示します。 変数情報は，優先順位の高い変数から順に出力します。 const 変数，sreg 変数，static 変数，フラッシュ領域側で参照するブート領域側で定義した変数は saddr 領域への配置対象外であるため，行頭にセミコロンを付加し，コメントとします。
(7)	変数情報（変数名）	変数名を出力します。
(8)	変数情報（参照回数）	変数の参照回数を出力します。
(9)	変数情報（サイズ）	変数のサイズを出力します。

項番	内容	形式
(10)	変数情報（参照タイプ）	変数の参照タイプを出力します。 near：1（near 領域から saddr 領域へ変更） far：2（far 領域から saddr 領域へ変更） sreg：0（sreg 指定により，すでに saddr 領域に配置）
(11)	変数情報（ファイル名）	対象ソース・ファイル名を“　”で囲んで出力します。 static 変数の場合は出力しますが，global 変数の場合は出力しません。
(12)	変数情報（const 変数）	const 変数の場合，“const”を出力します。
(13)	変数情報（ブート領域側の変数）	フラッシュ領域側で参照するブート領域側で定義した変数の場合，“boot”を出力します。
(14)	関数情報（開始）	関数情報の開始を示します。 関数情報は，優先順位の高い変数から順に出力します。 フラッシュ領域側の関数，callt 関数，static 関数は saddr 領域への配置対象外であるため，行頭にセミコロンを付加し，コメントとします。
(15)	関数情報（関数名）	関数名を出力します。
(16)	関数情報（参照回数）	関数の参照回数を出力します。
(17)	関数情報（参照タイプ）	関数の参照タイプを出力します。 near：1（near 領域から callt 領域へ変更） far：2（far 領域から callt 領域へ変更） sreg：0（すでに callt 領域に配置）
(18)	関数情報（ファイル名）	対象ソース・ファイル名を“　”で囲んで出力します。 static 関数の場合は出力しますが，global 関数の場合は出力しません。
(19)	関数情報（ブート領域側の関数）	フラッシュ領域側で参照するブート領域側で定義した関数の場合，“boot”を出力します。

第 4 章　サンプル・プログラム

この章では，本製品の CA78K0R（ビルドツール）に添付されているサンプル・プログラムのリストを紹介します。

4. 1　C コンパイラ

ここでは，C コンパイラのサンプル・プログラムのリストを紹介します。

4. 1. 1　C ソース・ファイル

```
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark [ SIZE + 1 ] ;

void main ( void ) {
        int     i , prime , k , count ;

        count = 0 ;

        for ( i = 0 ; i <= SIZE ; i++ )
                mark [ i ] = TRUE ;
        for ( i = 0 ; i <= SIZE ; i++ ) {
                if ( mark [ i ] ) {
                        prime = i + i + 3 ;
                        printf ( "%6d" , prime ) ;
                        count++ ;
                        if ( ( count%8 ) == 0 ) putchar ( '\n' ) ;
                        for ( k = i + prime ; k <= SIZE ; k += prime )
                                mark [ k ] = FALSE ;
                }
        }
        printf ( "\n%d primes found." , count ) ;
}

printf ( char *s , int i ) {
        int     j ;
        char    *ss ;

        j = i ;
        ss = s ;
```

```
}

putchar ( char c ) {
        char    d ;
        d = c ;
}
```

備考　本ファイルをコンパイルすると，次のワーニングが出力されます。

```
prime.c (17) : CC78K0R warning W0745 : Expected function prototype
prime.c (19) : CC78K0R warning W0745 : Expected function prototype
prime.c (33) : CC78K0R warning W0622 : No return value
prime.c (38) : CC78K0R warning W0622 : No return value
```

4. 2　アセンブラ

ここでは，アセンブラのサンプル・プログラムのリストを紹介します。

4. 2. 1　k0rmain.asm

```
        NAME    SAMPM
; ***********************************************
;       HEX -> ASCII Conversion Program
;               main-routine
; ***********************************************

PUBLIC  MAIN , START
EXTRN   CONVAH
EXTRN   _@STBEG

DATA    DSEG    AT      0FFE20H
HDTSA : DS      1
STASC : DS      2

CODE    CSEG    AT      0H
MAIN :  DW      START

        CSEG
START :

        ; chip initialize
        MOVW    SP , #_@STBEG

        MOV     HDTSA , #1AH
        MOVW    HL , #LOWW ( HDTSA )    ; set hex 2-code data in HL registor

        CALL    !CONVAH                 ; convert ASCII <- HEX
                                        ; output BC-register <- ASCII code
        MOVW    DE , #LOWW ( STASC )    ; set DE <- store ASCII code table
        MOV     A , B
        MOV     [ DE ] , A
        INCW    DE
        MOV     A , C
        MOV     [ DE ] , A
        BR      $$

        END
```

4. 2. 2　k0rsub.asm

```
        NAME    SAMPS
; ************************************************************
;       HEX -> ASCII Conversion Program
;               sub-routine
;   input condition     : ( HL )        <- hex 2 code
;   output condition    : BC-register   <- ASCII 2 code
; ************************************************************

PUBLIC  CONVAH

        CSEG
CONVAH :
        XOR     A , A
        ROL4    [ HL ]          ; hex lower code load
        CALL    !SASC
        MOV     B , A           ; store result

        XOR     A , A
        ROL4    [ HL ]          ; hex lower code load
        CALL    !SASC
        MOV     C , A           ; store result
        RET

; ************************************************************
;       subroutine      convert ASCII code
;
;   input       Acc ( lower 4bits )     <- hex code
;   output      Acc                     <- ASCII code
; ************************************************************

SASC :
        CMP     A , #0AH        ; check hex code > 9
        BC      $SASC1
        ADD     A , #07H        ; bias ( +7H )
SASC1 :
        ADD     A , #30H        ; bias ( +30H )
        RET

        END
```

第 5 章　注意事項

この章では，CubeSuite+，CA78K0R の各コマンドを使用する際の注意事項を示します。

（1）漢字コード種別

EUC コードを含むソースを使用するときは，環境変数 LANG78K を euc に設定するか，-ze オプションを指定してください。

CubeSuite+ 使用時は，プロパティ パネルの［コンパイル・オプション］タブの［機能拡張］カテゴリの［ソースの漢字コード］プロパティ（C ソース・ファイルの場合），および［アセンブル・オプション］タブの［その他］カテゴリの［ソースの漢字コード］プロパティ（アセンブラ・ソース・ファイルの場合）で設定してください。

指定した漢字コードとソース中に含まれる漢字コードが異なる場合，ビルド時にエラーになる，またはソースの一部を誤ってコメントとみなして正しくビルドされない場合があります。

（2）コンパイル・オプションの指定

コンパイル・オプションの指定時，次の点に注意してください。

- 複数指定が可能でないオプションを複数指定した場合には，後に指定した方を優先します。
- -c に続けて指定する種別を省略することはできません。
- ヘルプ指定があった場合には，他のすべてのオプションは無効になります。

（3）ソース・ファイル名

C コンパイラでは，ソース・ファイル名の拡張子を除いた部分（プライマリ名）を，デフォルトでモジュール名として使用します。そのため，使用可能なソース・ファイル名には，若干の制限があります。

- ファイル名の長さは，ホスト OS の許す範囲内のプライマリ名と拡張子で構成し，プライマリ名と拡張子の間は，ドット（ . ）で区切る形式としてください。
- プライマリ名，拡張子ともに，使用可能な文字は，ホスト OS が許している文字から，括弧（ ( ) ），セミコロン（ ; ），コンマ（ , ）を除いた文字とします。ただし，ファイル名とパス名の先頭に，ハイフン（ - ）を使用することはできません。また，スペースや 2 バイト文字を含むファイル名を指定しないでください。
- パラメータ・ファイル内では，ファイル名とパス名にシャープ（#）を使用することはできません。

（4）アセンブラ・ソース・ファイルを出力して使用する場合

C ソース・ファイル中に，#asm ブロック，または __asm 文などのアセンブリ言語による記述がある場合，ロード・モジュール・ファイル作成手順は，コンパイル，アセンブル，リンクの順になります。

アセンブリ言語による記述がある場合などのように，C コンパイラで直接オブジェクトを出力せずに，いったんアセンブラ・ソース・ファイルを出力し，アセンブルして使用する場合には，次の点に注意してください。

- C ソース・ファイル中に，#asm ブロック，および __asm 文がある場合は，アセンブリ記述を有効にするためにコンパイル・オプション -a，または -sa オプションを指定して，出力されたアセンブラ・ソー

ス・ファイルをアセンブルしてください。

CubeSuite+ 使用時は，プロパティ パネルの［コンパイル・オプション］タブの［アセンブリ・ファイル］カテゴリの［アセンブリ・ファイルを出力する］プロパティでアセンブラ・ソース・ファイルの出力を指定するか，アセンブラ・ソース・ファイルしか出力しないソースに対し，［個別コンパイル・オプション］タブの［アセンブリ・ファイル］カテゴリ［アセンブリ・ファイルを出力する］プロパティでアセンブラ・ソース・ファイルの出力を指定してください。

- CubeSuite+ 使用時に，アセンブラ・ソース・ファイルの出力を指定した場合は，コンパイル・オプション -o，-no にかかわらずアセンブラを起動します。

(5) スタック解決用シンボル生成指定オプション（-s）

スタック領域を確保するためには，リンク時にリンク・オプション -s を指定してください。

CubeSuite+ 使用時は，プロパティ パネルの［リンク・オプション］タブの［スタック］カテゴリを設定してください。なお，CubeSuite+ 使用時に，ソース・ファイル指定に C ソース・ファイルが含まれる場合は，-s オプションが自動的に付加されます。

(6) オブジェクト・コンバータの使用

オブジェクト・コンバータは，-r（オブジェクトのアドレス・ソート），および -u（充てん値指定）オプションを指定して使用してください。

CubeSuite+ 使用時は，プロパティ パネルの［オブジェクト・コンバート・オプション］タブの［ヘキサ・ファイル］カテゴリの［ヘキサ・ファイル充てん］カテゴリを設定してください。

本オプションは，デフォルトでは指定されています。

オブジェクトがアドレス・ソートされていない場合，ROM コード発注（アクロス処理，テープ・アウトと呼ばれている作業です）を行うとエラーになりますので，-r は必ず指定してください（指定の解除をしないでください）。

(7) オブジェクト充てん値指定オプション（-u）

オブジェクト・コンバート・オプション -u で，スタート・アドレスを指定した場合，スタート・アドレス，またはコードが配置されたアドレスの小さい方のアドレスから充てんを開始します。内部 RAM 領域（ED800H - FFFFFH）には充てんを行いません。

記述形式を以下に示します。

```
-u 充てん値 [,[ スタート・アドレス ], サイズ ]
```

備考　[　] 内は省略可能です。

(8) インクルード・ファイルの依存関係

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てします。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 */
*/
```

また，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"     /* 依存関係がないと判断する */

/*
comment
*/  #include    "header7.h"                 /* 依存関係がないと判断する */
```

(9) ネットワークの使用

テンポラリ・ファイルを作成するフォルダをネットワーク上で共有されているファイル・システムに置くと，使用しているネットワーク・ソフトウエアの種類によっては，ファイルの競合が生じて異常動作を起こす場合があります。オプションや環境変数の設定によって，このような競合は避けてください。

CubeSuite+ 使用時は，ネットワーク環境でのテンポラリ・ファイルの使用は避けてください。

(10) セルフ・プログラミング使用時

セルフ・プログラミングのための各コマンドのオプションについて，次に示します。

(a) リンカ

ブート領域用ロード・モジュール・ファイルを作成する場合は，-zb オプションを使用します。

-zb オプションの後には，フラッシュ ROM 領域の先頭アドレスを指定してください。

CubeSuite+ 使用時は，プロパティ パネルの［リンク・オプション］タブの［デバイス］カテゴリの［フラッシュ・スタート・アドレスを設定する］プロパティ，および［フラッシュ・スタート・アドレス］プロパティを設定してください。

次に，フラッシュ領域用ロード・モジュール・ファイルを作成する場合は，ブート領域用ロード・モジュール・ファイルとフラッシュ領域用オブジェクト・モジュール・ファイルを入力し，再リンクします。

なお，フラッシュ領域用オブジェクト・モジュール・ファイルは，-zb オプションで指定したアドレス以降に配置してください。

ブート領域用ロード・モジュール・ファイルとフラッシュ領域用オブジェクト・モジュール・ファイルは，スモール・モデルとミディアム・モデルの混在を許可します。ただし，フラッシュ領域用オブジェクト・モジュール・ファイル内では統一されていなければなりません。

(b) オブジェクト・コンバータ

ブート領域用ロード・モジュール・ファイルとフラッシュ領域用オブジェクト・モジュール・ファイルを入力し，再リンクして出力されたロード・モジュール・ファイルを，オブジェクト・コンバータに入力します。

このとき，-zf オプションを指定することで，ブート領域用ヘキサ・ファイル（*.hxb）とフラッシュ領域用ヘキサ・ファイル（*.hxf）を分割して出力することができます。

CubeSuite+ 使用時は，プロパティ パネルの［オブジェクト・コンバート・オプション］タブの［ヘキサ・ファイル］カテゴリの［ヘキサ・ファイルを分割する］プロパティを設定してください。

(11) 変数／関数情報ファイル生成ツールの使用

(a) 関数呼び出しの記述

引数の型宣言がない関数を呼び出し，引数に「変数／関数情報ファイルで callt 配置を指定した関数のアドレス」を記述した場合，関数インタフェースが整合せずに不正動作となる場合があります。

- C ソース

```
int func_c()    /* callt in .vfi */
{
    return 0;
}

void func()
{
    func2(func_c);  /* W0553 */
}

int func2(int (*p)(void))
{
}
```

- 変数／関数情報ファイル

```
;*** variable information ***
[sreg]
;
;*** function information ***
[callt]
func_c,2,2
func2,1,2
```

上記の条件を満たす場合は，C コンパイラがワーニング W0553 を出力します。

関数呼び出しのプロトタイプ宣言を，型宣言を含めて記述してください。

または関数の引数に記述した関数に対して，変数／関数情報ファイルの callt 指定をコメントアウトしてください。

（b）#pragma section 指令を AT 開始アドレス付きで指定する場合

#pragma section 指令を AT 開始アドレス付きで指定したセクション中の関数や変数を，変数／関数情報ファイルで callt テーブル領域，saddr 領域に配置すると，不正動作となる場合があります。

- C ソース

```
#pragma section @@DATA @FCDATA AT 0FCF00H
#define dni1    (*(int *)0xfcf00)
int __near ni1; /* sreg in .vfi */
__sreg int x1, x2;

void func()
{
    x1 = ni1;
    x2 = dni1;
}

void main()
{
    ni1 = 0x10;
    func();
}
```

- 変数／関数情報ファイル

```
;*** variable information ***
[sreg]
;x2,1,2,0
;x1,1,2,0
;
;*** function information ***
[callt]
func,1,2
```

上記の C ソースは，変数 x1，x2 の値が両方とも 0x10 となることを期待しています。しかし，変数／関数情報ファイルによって，変数 ni1 が saddr 領域（0xffe20 ～）に割り当たると，変数 x1 の値は ni1 の値である 0x10，変数 x2 の値は 0xfcf00 番地の値となり，意図した動作になりません。

変数／関数情報ファイル生成ツールでは，#pragma section 指令を AT 開始アドレス付きで指定したセクション中の関数や変数は，sreg，callt 指定の対象外としています。

変数／関数情報ファイルを修正する際は，上記の関数や変数を callt テーブル領域，saddr 領域に配置する指定は行わないでください。

（c）ワーニングの扱い

変数／関数情報ファイル生成ツールによって callt テーブル領域に配置した関数のアドレスを扱う際，ワーニングが出力される場合があります。

- C ソース（ミディアム・モデル，またはラージ・モデルの場合）

```
void f1(void (*fp)()){}
void f2(void){}
void (*fp)(void);
void main(void)
{
    f1(f2);                 /* W0510: Pointer mismatch in function */
    f1((void (*)())f2);     /* キャストすれば OK */
    fp = f2;    /* W0416: Illegal type and size (far/near) pointer combination */
    fp = (void (*)())f2;    /* キャストすれば OK */
}
```

- 変数／関数情報ファイル

```
;*** variable information ***
[sreg]
fp,4,4,2
;
;*** function information ***
[callt]
f2,7,2
f1,2,2
```

動作上は問題ありませんが，ワーニングの出力を抑制したい場合は，関数ポインタを扱う処理でキャストしてください。

(d) 参照回数のカウント

関数のアドレス参照も関数呼び出し回数としてカウントするため，参照回数を適切にカウントできない場合があります。

(e) コンパイラが生成するローカル・シンボルの出力

コンパイラが生成するローカル・シンボルも変数／関数情報ファイルに出力されますが，コメントアウトしたままにしてください。

- 変数／関数情報ファイル

```
[sreg]
;L0003,2,1,2,"t08.c",const
;
;*** function information ***
[callt]
```

(f) ライブラリ・ファイル，およびロード・モジュール・ファイルの拡張子の変更

変数／関数情報ファイル生成ツールを使用する場合，ライブラリ・ファイルの拡張子（.lib），およびロード・モジュール・ファイルの拡張子（.lmf）は変更しないでください。

変更した場合，処理対象外の変数／関数が出力されることがあります。

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
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
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
ブート領域用変数／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ブート領域用変数／関数情報ファイルを選択

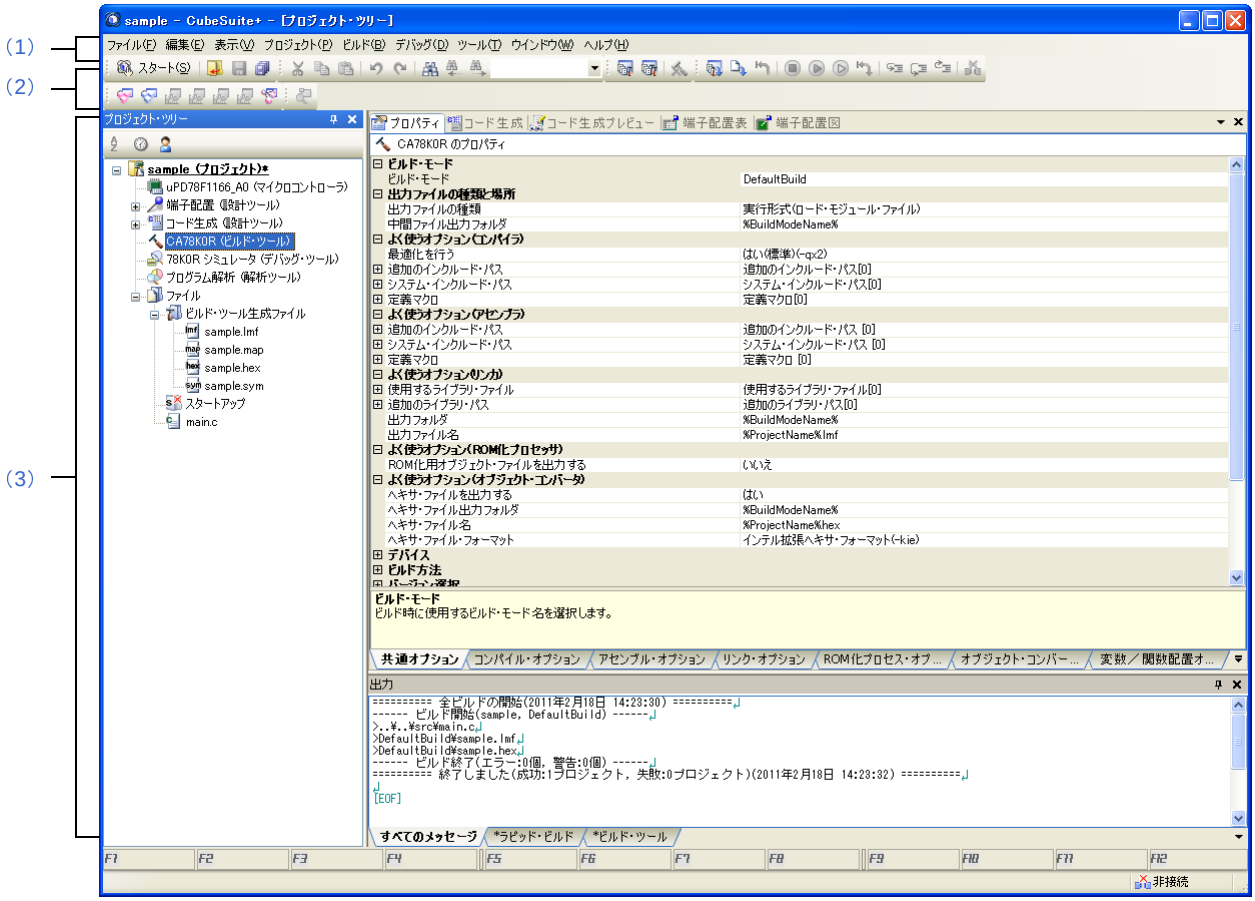
ウインドウ／パネル／ダイアログ名	機能概要
ブート領域用ロード・モジュール・ファイルを指定 ダイアログ	本ダイアログの呼び出し元に設定するブート領域用ロード・モジュール・ファイルを選択
名前を付けて保存 ダイアログ	編集中のファイル，または各パネルの内容をファイルに保存
プログラムから開く ダイアログ	ファイルを開くアプリケーションを選択
Stack Usage Tracer ウインドウ	スタック見積もりツールを起動した際，最初にオープンするウインドウ
サイズ不明関数・サイズ変更関数一覧 ダイアログ	スタック見積もりツールがスタック情報を取得できていない関数，意図的に情報の変更が行われた関数，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。

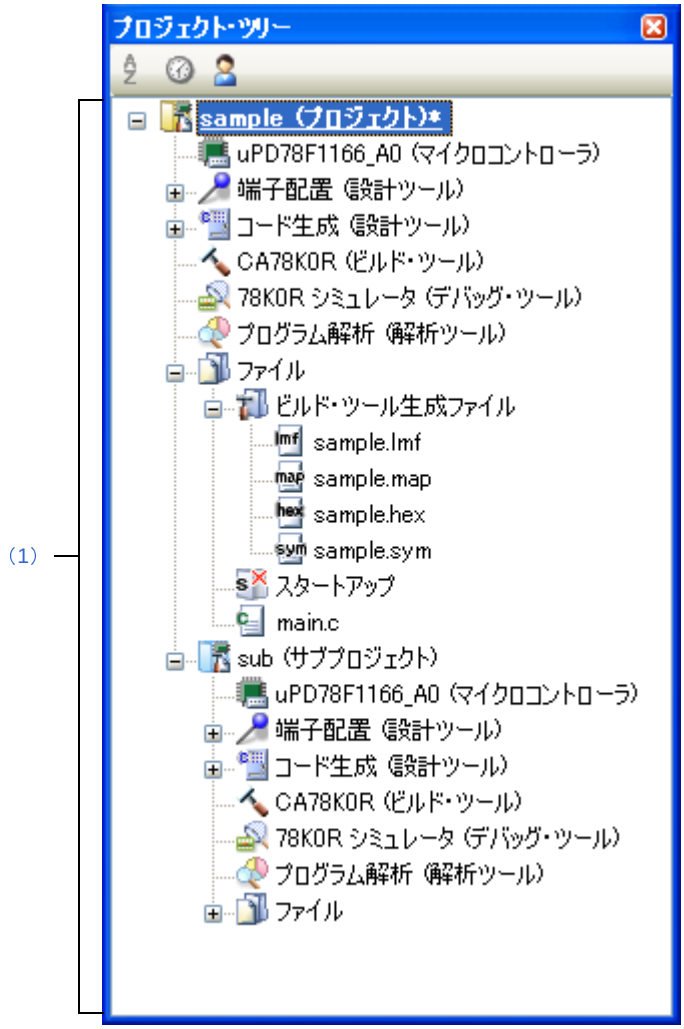
- プロジェクト・ツリー パネル
- プロパティ パネル
- エディタ パネル
- 出力 パネル

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（*.asm） - ヘッダ・ファイル（*.h，*.inc） - オブジェクト・ファイル（*.rel） - ライブラリ・ファイル（*.lib） - リンク・ディレクティブ・ファイル（*.dr，*dir） - 変数／関数情報ファイル（*.vfi） - その他のファイル（*.doc，*.xml 等）
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 (*.lmf) リンク・リスト・ファイル (*.map) エラー・リスト・ファイル（*.elk，*.erp，*.eoc） ヘキサ・ファイル (*.hex，*.hxb，*.hxf) シンボル・テーブル・ファイル (*.sym) - ライブラリ用のプロジェクトの場合 ライブラリ・ファイル (*.lib) リスト・ファイル (*.lst) 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. 5　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。

ノード	説明
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

ボタン	説明
	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（*.asm）
- 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）
- 変数／関数情報ファイル（*.vfi）
- マップ・ファイル（*.map）
- シンボル・テーブル・ファイル（*.sym）
- ヘキサ・ファイル（*.hex，*.hxb，*.hxf）
- 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。
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

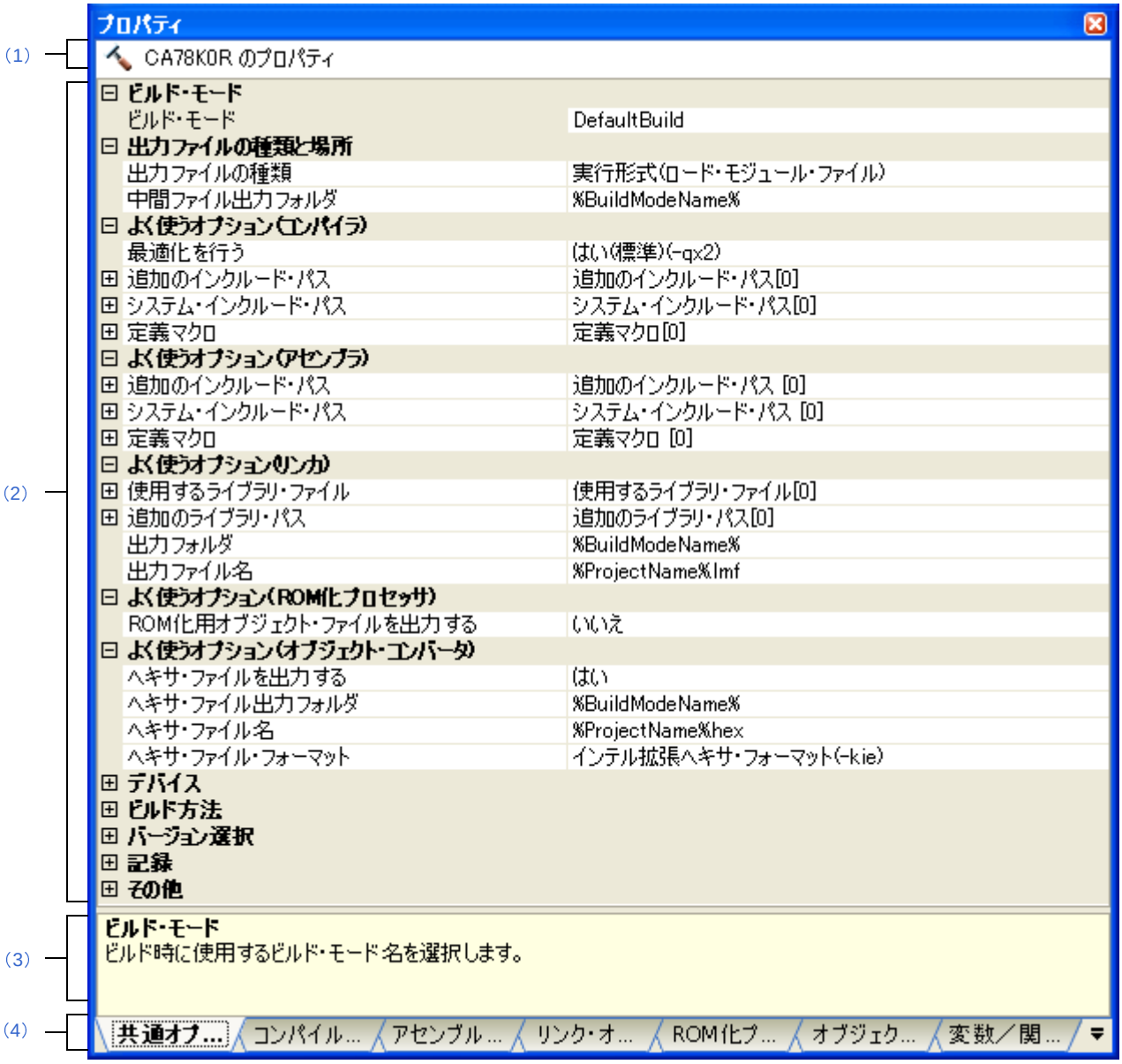
追加	プロジェクトにファイル，カテゴリ・ノードを追加するためのカスケード・メニューを表示します。

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
- ［オブジェクト・コンバート・オプション］タブ
- ［ライブラリ生成オプション］タブ
- ［変数／関数配置オプション］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，変数／関数情報ファイル，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）注
- ［ファイル情報］タブ

注　本タブは，［個別コンパイル・オプション］タブの［アセンブリ・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］を選択した場合も表示されます。

（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

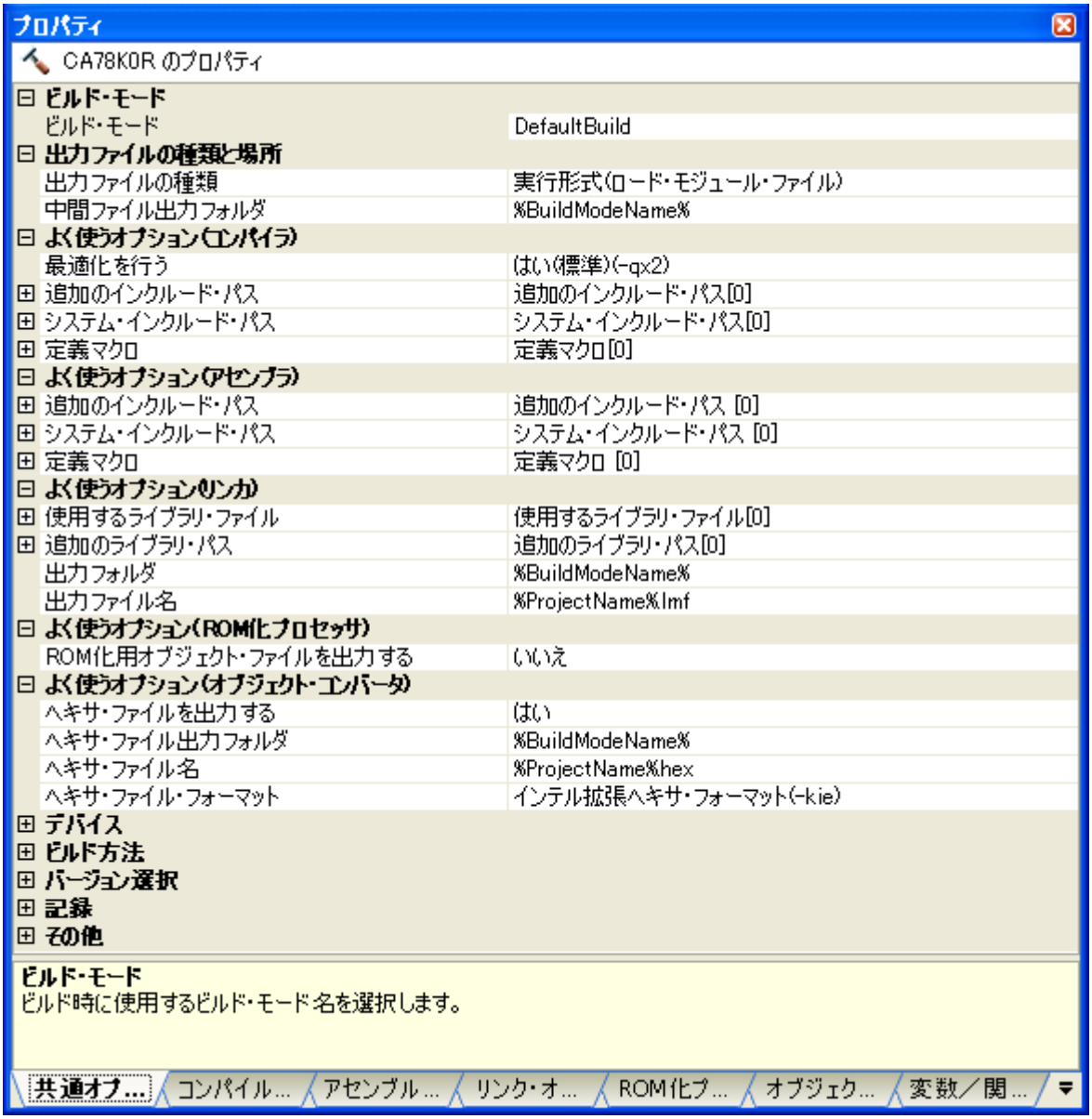
本タブでは，ビルド・ツールに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

（1）［ビルド・モード］
（2）［出力ファイルの種類と場所］
（3）［よく使うオプション（コンパイラ）］
（4）［よく使うオプション（アセンブラ）］
（5）［よく使うオプション（リンカ）］
（6）［よく使うオプション（ROM 化プロセッサ）］
（7）［よく使うオプション（オブジェクト・コンバータ）］
（8）［デバイス］
（9）［ビルド方法］
（10）［バージョン選択］
（11）［記録］
（12）［その他］

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（オブジェクト・コンバータ）］カテゴリ	［オブジェクト・コンバート・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，［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。また，［オブジェクト・コンバート・オプション］タブの［ヘキサ・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。 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。

中間ファイル出力フォルダ	中間ファイル（オブジェクト・モジュール・ファイル（*.rel），クロスリファレンス・リスト・ファイル（*.xrf）等）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。	
	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

(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。

最適化を行う	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 -qx に相当します。		
	デフォルト	はい (標準)(-qx2)	
	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 (実行速度優先)(-qx1)	実行速度を優先して最適化を行います。
		はい (標準)(-qx2)	実行速度，モジュール・サイズの両方を優先して最適化を行います。
		はい (モジュール・サイズ優先)(-qx3)	モジュール・サイズを優先して最適化を行います。
		はい (詳細設定)	［コンパイル・オプション］タブに［最適化（詳細）］カテゴリが表示され，そこで選択したオプションを優先して最適化を行います。 なお，［最適化（詳細）］カテゴリですべて［いいえ］を選択した場合は，最適化を行いません。
		いいえ (-nq)	最適化を行いません。

追加のインクルード・パス	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。 本プロパティを省略した場合，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します。 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一番最初に追加します。 インクルード・パスに大文字，小文字の区別はありません。	
	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。
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します。	
	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 30 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。

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。 本プロパティを省略した場合，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します。 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一番最初に追加します。 インクルード・パスに大文字，小文字の区別はありません。	
	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。
システム・インクルード・パス	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。	
	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。 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します。	
	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 30 個まで指定可能です。

(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。

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）を指定します。 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 -b 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します。	
	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	259 文字までの文字列 64 個まで指定可能です。
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 -i 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します。	
	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 64 個まで指定可能です。

出力フォルダ	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。	
	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	出力するロード・モジュールのファイル名を指定します。 拡張子は“.lmf”を指定してください。拡張子を省略した場合は，“.lmf”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%.lmf”を指定したものとみなします。	
	デフォルト	%ProjectName%.lmf
	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。		
	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	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］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	259 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。 “.lmf”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，“.lmf”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］を選択した場合のみ表示されます。	
	デフォルト	romp.lmf
	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（オブジェクト・コンバータ）］

オブジェクト・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。 オブジェクト・コンバータのオプション -o 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・ファイルを出力します。
		いいえ (-no)	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。 オブジェクト・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。 オブジェクト・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。 オブジェクト・コンバータのオプション -k 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［いいえ (-no)］を選択した場合は表示されません。		
	デフォルト	インテル拡張ヘキサ・フォーマット (-kie)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	インテル標準ヘキサ・フォーマット (-ki)	生成するヘキサ・ファイルをインテル標準ヘキサ・フォーマットとします。
		インテル拡張ヘキサ・フォーマット (-kie)	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)(-km)	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)(-kme)	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 (-kt)	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)［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。

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 -gi に相当します。 本プロパティは，［リンク・オプション］タブの［デバイス］カテゴリの［ブート領域用ロード・モジュール・ファイル名］プロパティを指定した場合は無効となります。 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	0x00000000000000000000【RL78】 0xffffffffffffffffffff【78K0R】	
	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 ～ 0xfffffffffffffffffffe【RL78】 0x00000000000000000000 ～ 0xffffffffffffffffffff【78K0R】 （20 桁（10 バイト）の 16 進数）	

(9)［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。

(10)［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	

（11）［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。

（12）［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。

出力メッセージ・フォーマット	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。		
	デフォルト	%FileName%	
	変更方法	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	
	指定可能値	%FileName%	出力メッセージにファイル名を表示します。
		%FileName%: %Options%	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。
		%Program% %Options%	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。
ビルド・オプション一覧表示フォーマット	ビルド・オプション一覧（「2. 15. 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。		
	デフォルト	%FileName% : %Program% %Options%	
	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	256 文字までの文字列	

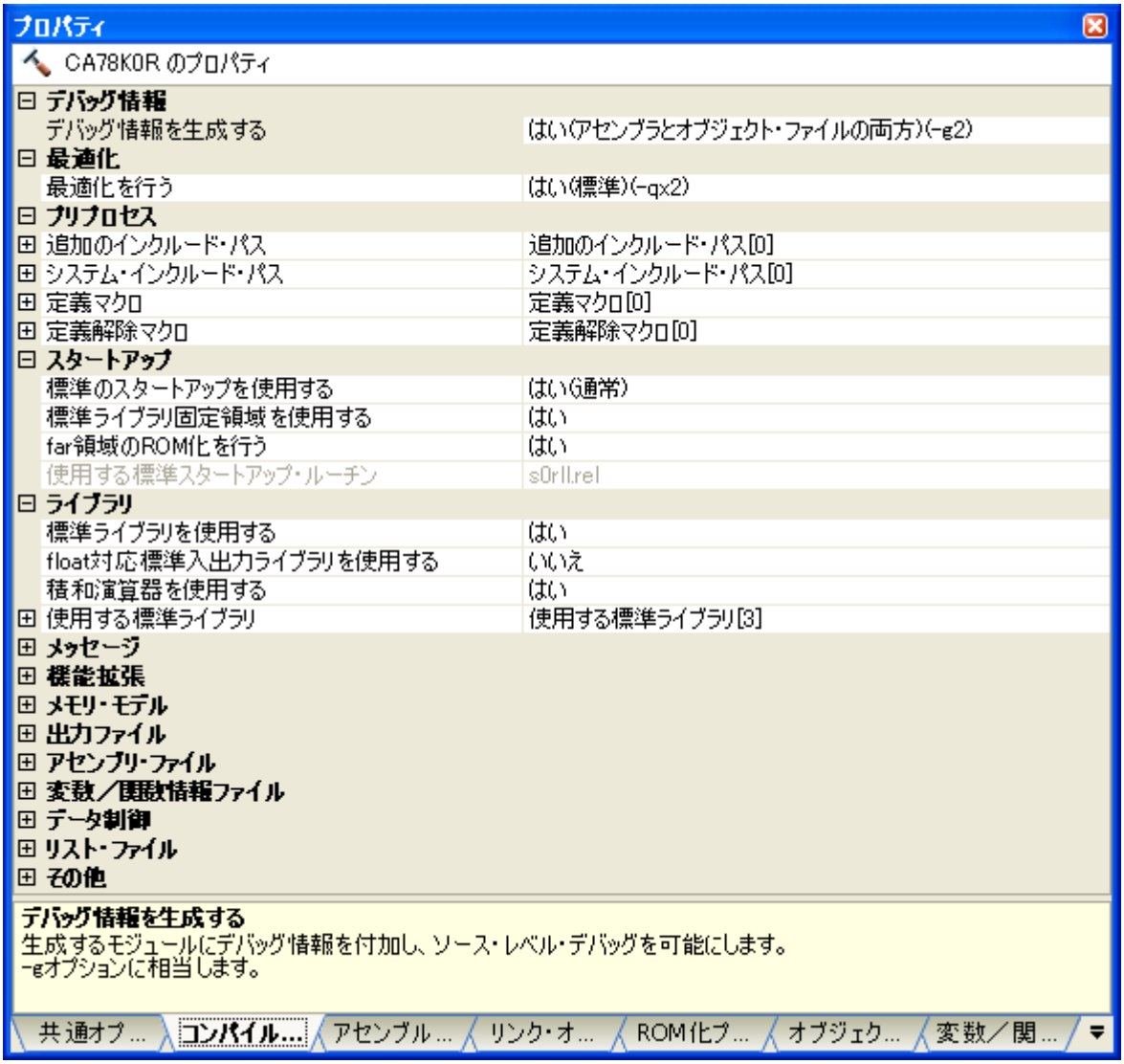
一時作業フォルダ	ビルド・ツールの各コマンドが実行中に生成する一時的なファイルの格納先フォルダを指定します。 各コマンドの -t オプションに相当します。 相対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とします。 絶対パスで指定した場合は，メイン・プロジェクト，またはサブプロジェクトのフォルダを基点とした相対パスに変換されます（ドライブが異なる場合を除く）。 空欄の場合は，プロジェクト・フォルダを指定したものとみなします。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，フォルダの参照 ダイアログによる編集
	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。

（1）［デバッグ情報］
（2）［最適化］
（3）［最適化（詳細）］
（4）［プリプロセス］
（5）［スタートアップ］
（6）［ライブラリ］
（7）［メッセージ］
（8）［機能拡張］
（9）［メモリ・モデル］
（10）［出力ファイル］
（11）［アセンブリ・ファイル］
（12）［変数／関数情報ファイル］
（13）［データ制御］
（14）［リスト・ファイル］
（15）［その他］

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	生成するモジュー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。 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 (アセンブラとオブジェクト・ファイルの両方)(-g2)	
	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 (オブジェクト・ファイルのみ)(-g1)	生成するオブジェクト・モジュール・ファイルにデバッグ情報を付加します。
		はい (アセンブラとオブジェクト・ファイルの両方)(-g2)	生成するオブジェクト・モジュール・ファイルとアセンブラ・ソース・ファイルにデバッグ情報を付加します。
		いいえ (-ng)	生成するオブジェクト・モジュール・ファイルにデバッグ情報を付加しません。

(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。

最適化を行う	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 -qx に相当します。		
	デフォルト	はい (標準)(-qx2)	
	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 (実行速度優先)(-qx1)	実行速度を優先して最適化を行います。
		はい (標準)(-qx2)	実行速度，モジュール・サイズの両方を優先して最適化を行います。
		はい (モジュール・サイズ優先)(-qx3)	モジュール・サイズを優先して最適化を行います。
		はい (詳細設定)	[最適化（詳細）] カテゴリが表示され，そこで選択したオプションを優先して最適化を行います。 なお，[最適化（詳細）] カテゴリですべて [いいえ] を選択した場合は，最適化を行いません。
		いいえ (-nq)	最適化を指定しません。

(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。

なお，本カテゴリは，[最適化] カテゴリの [最適化を行う] プロパティで [はい（詳細設定）] を選択した場合のみ表示されます。

演算順序の入れ替えを行う	演算式の実行順序入れ替えなどを行うことでレジスタの有効活用を図り，効率の良い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 -qw に相当します。		
	デフォルト	はい (式の演算順序を入れ替える)(-qw)	
	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 (式の演算順序を入れ替える)(-qw)	式の演算順序を入れ替えます。
		いいえ	式の演算順序の入れ替えを指定しません。
自動変数をレジスタ，または saddr 領域に割り当てる	自動変数をレジスタ，または saddr 領域に自動的に割り当てるかどうかを選択します。 コンパイラのオプション -qv に相当します。		
	デフォルト	はい (-qv)	
	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 (-qv)	自動変数をレジスタ，または saddr 領域に自動的に割り当てます。
		いいえ	自動変数のレジスタ，または saddr 領域への自動的に割り当てを指定しません。
レジスタ変数を saddr 領域にも割り当てる	レジスタ変数をレジスタに加えて saddr 領域にも割り当てるかどうかを選択します。 コンパイラのオプション -qr 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 (-qr)	レジスタ変数をレジスタに加えて saddr 領域にも割り当てます。
		いいえ	レジスタ変数の saddr 領域への割り当てを指定しません。
char 型演算を符号拡張しない	char に関する演算を汎整数拡張せずに行うかどうかを選択します。 コンパイラのオプション -qc に相当します。		
	デフォルト	はい (-qc)	
	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 (-qc)	char に関する演算を汎整数拡張せずに行います。注
		いいえ	char に関する演算を汎整数拡張して行います。
char 型を unsigned char とみなす	修飾子なしの char を unsigned char とみなすかどうかを選択します。 コンパイラのオプション -qu 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 (-qu)	修飾子なしの char を unsigned char とみなします。
		いいえ	修飾子なしの char を unsigned char とみなすことを指定しません。

分岐命令を最適化する	分岐命令の最適化を行うかどうかを選択します。 コンパイラのオプション -qj に相当します。		
	デフォルト	はい (-qj)	
	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 (-qj)	分岐命令の最適化を行います。
		いいえ	分岐命令の最適化を指定しません。
定型コードをライブラリに置き換える (サイズ優先最適化)	定型コードをライブラリに置き換えるかどうかを選択します。 コンパイラのオプション -ql に相当します。		
	デフォルト	はい (ライブラリに置き換えない)(-ql1)	
	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 (ライブラリに置き換えない)(-ql1)	定型コードをライブラリに置き換えません。モジュール・サイズを優先して最適化を行います。
		はい (関数の前後処理のみ)(-ql2)	関数の前後処理のみライブラリに置き換えます。
		はい (関数の前後処理，低レベル・ライブラリの使用，共通コードのサブルーチン化)(-ql3)	関数の前後処理のみライブラリに置き換えます。また，低レベル・ライブラリの使用，共通コードのサブルーチン化を行います。
		いいえ	定型コードのライブラリへの置き換えを指定しません。 実行速度を優先して最適化を行います。
相対分岐の switch 分岐テーブルを生成する	相対分岐の switch 分岐テーブルを生成するかどうかを選択します。 コンパイラのオプション -qt 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 (-qt)	相対分岐の switch 分岐テーブルを生成します。
		いいえ	相対分岐の switch 分岐テーブルの生成を指定しません。
デバッグに適した最適化を行う	デバッグに適した最適化を行うかどうかを選択します。 コンパイラのオプション -qg 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 (-qg)	デバッグに適した最適化を行います。
		いいえ	デバッグに適した最適化を指定しません。

注　-qc オプションを設定した場合の演算結果は，次のようになります。

演算対象	演算結果
unsigned char 型の変数と unsigned char 型の変数	unsigned char 型
unsigned char 型の変数と signed char 型の変数	unsigned char 型
signed char 型の変数と signed char 型の変数	signed char 型

演算対象	演算結果
-128 ～ 255 の定数と unsigned char 型の変数	unsigned char 型
-128 ～ 127 の定数と signed char 型の変数	signed char 型
0 ～ 255 で接尾語 U 付きの定数と signed char 型の変数	unsigned char 型

(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。
本プロパティを省略した場合，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します。
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一番最初に追加します。
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。
また，［システム・インクルード・パス］プロパティ，および［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］プロパティの指定数との合計が 64 個を越えた場合，ビルドの実行時にエラーとなります。</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>

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します。	
	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 30 個まで指定可能です。
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します。	
	デフォルト	定義解除マクロ [*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 30 個まで指定可能です。

(5) [スタートアップ]

スタートア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。

<table>
<tr><td rowspan="7">標準のスタートアップを使用する</td><td colspan="3">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・ファイルを登録していない場合は，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。
なお，標準ビルド・オプション（「2. 15. 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」参照）として記憶している本プロパティの値は，［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］プロパティを変更することにより，デフォルト値にリセットされます。</td></tr>
<tr><td>デフォルト</td><td colspan="2">- ［フラッシュ用オブジェクトを出力する］プロパティで［はい (-zf)］を選択した場合
［はい (フラッシュ領域用)］
- [フラッシュ用オブジェクトを出力する] プロパティで［いいえ］を選択した場合
［はい (通常)］</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（通常）</td><td>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。
なお，本項目は，[フラッシュ用オブジェクトを出力する] プロパティで［はい (-zf)］を選択した場合は表示されません。</td></tr>
<tr><td>はい（ブート領域用）</td><td>コンパイラ付属のブート領域用オブジェクト・モジュール・ファイルをリンクします。
なお，本項目は，[フラッシュ用オブジェクトを出力する] プロパティで［はい (-zf)］を選択した場合は表示されません。</td></tr>
<tr><td>はい（フラッシュ領域用）</td><td>コンパイラ付属のフラッシュ領域用オブジェクト・モジュール・ファイルをリンクします。
なお，本項目は，[フラッシュ用オブジェクトを出力する] 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>いいえ</td><td>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。</td></tr>
</table>

標準ライブラリ固定領域を使用する	標準ライブラリ brk，sbrk，malloc，calloc，realloc，free，exit，rand，srand，div，ldiv，strtok，atof，strtod，数学関数，浮動小数点ランタイム・ライブラリが使用する固定領域（RAM）を使用するかどうかを選択します。 これらの関数を使用しない場合は，［いいえ］を選択すると，RAM を節約することができます。 なお，本プロパティは，［標準のスタートアップを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	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	標準ライブラリが使用する固定領域を使用します。
		いいえ	標準ライブラリが使用する固定領域を使用しません。
far 領域の ROM 化を行う	far 領域の ROM 化を行うかどうかを選択します。 なお，本プロパティは，［標準のスタートアップを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	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	far 領域の ROM 化を行います。
		いいえ	far 領域の ROM 化を行いません。
使用する標準スタートアップ・ルーチン	現在の設定において，リンク時に使用する標準スタートアップ・ルーチン・オブジェクトのファイル名を表示します。**注** なお，本プロパティは，［標準のスタートアップを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	*使用するスタートアップ・ルーチン・ファイル名*	
	変更方法	変更不可	

注　スタートアップ・ルーチン・ファイル名の命名規則は，次のようになっています。

```
s0r<model><lib><flash>.rel
```

<model>

m	メモリ・モデルがスモール・モデルの場合，またはミディアム・モデル，かつ far 領域の ROM 化を行わない場合
l	メモリ・モデルがラージ・モデル，または far 領域の ROM 化を行う場合

<lib>

なし	標準ライブラリ固定領域を使用しない場合
l	標準ライブラリ固定領域を使用する場合

<flash>

なし	通常用オブジェクトを生成する場合
b	ブート領域用オブジェクトを生成する場合
e	フラッシュ領域用オブジェクトを生成する場合

(6)［ライブラリ］

ライブラ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。

標準ライブラリを使用する	リンク時に標準ライブラリを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・ファイルを登録していない場合は，ライブラリをリンクしません。		
	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	リンク時に標準ライブラリをリンクします。
		いいえ	リンク時に標準ライブラリをリンクしません。
float 対応標準入出力ライブラリを使用する	浮動小数点の入出力に対応した標準ライブラリ（sprintf，sscanf，printf，vprintf，vsprintf）を使用するかどうかを選択します。 なお，本プロパティは，［標準ライブラリを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	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	浮動小数点の入出力に対応した標準ライブラリを使用します。
		いいえ	浮動小数点の入出力に対応した標準ライブラリを使用しません。
乗除算器を使用する	乗除算器に対応した標準ライブラリをリンクするかどうかを選択します。 ただし，乗除算器の有無は，使用するマイクロコントローラによって異なります。 なお，本プロパティは，乗除算器がないマイクロコントローラの場合，乗除積和算命令を搭載したマイクロコントローラの場合，および［標準ライブラリを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	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	乗除算器に対応した標準ライブラリを使用します。
		いいえ	乗除算器に対応した標準ライブラリを使用しません。

乗算器を使用する	乗算器に対応した標準ライブラリをリンクするかどうかを選択します。 ただし，乗算器の有無は，使用するマイクロコントローラによって異なります。 なお，本プロパティは，乗算器がないマイクロコントローラの場合，乗除積和算命令を搭載したマイクロコントローラの場合，および［標準ライブラリを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	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	乗算器に対応した標準ライブラリを使用します。
		いいえ	乗算器に対応した標準ライブラリを使用しません。
積和演算器を使用する	積和演算器に対応した標準ライブラリをリンクするかどうかを選択します。 ただし，積和演算器の有無は，使用するマイクロコントローラによって異なります。 なお，本プロパティは，積和演算器がないマイクロコントローラの場合，乗除積和算命令を搭載したマイクロコントローラの場合，および［標準ライブラリを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	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	積和演算器に対応した標準ライブラリを使用します。
		いいえ	積和演算器に対応した標準ライブラリを使用しません。
使用する標準ライブラリ	現在の設定において，リンク時に使用する標準ライブラリの数とそのファイル名を表示します。 リンクするライブラリ・ファイル名はサブプロパティとして表示します。注 なお，本プロパティは，［標準ライブラリを使用する］プロパティで［いいえ］を選択した場合は表示されません。		
	デフォルト	使用する標準ライブラリ名 [*使用する標準ライブラリ数*]	
	変更方法	変更不可	

注　ライブラリ・ファイル名の命名規則は，次のようになっています。

```
cl<line><mul><model><float><flash>.lib
```

<line>

0r	RL78 拡張命令非搭載品，または 78K0R の場合
78	RL78 拡張命令搭載品の場合

<mul>

なし	乗算器／乗除算器を使用しない場合
x	乗算器を使用する場合
d	乗除算器を使用する場合
a	積和演算器を使用する場合

<model>

m	メモリ・モデルがスモール・モデル，またはミディアム・モデルの場合
l	メモリ・モデルがラージ・モデルの場合

<float>

なし	float 対応標準入出力ライブラリを使用しない場合
f	float 対応標準入出力ライブラリを使用する場合

<flash>

なし	通常用／ブート領域用オブジェクトを生成する場合
e	フラッシュ領域用オブジェクトを生成する場合

(7)［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	通常出力	
	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 (-w0)	ワーニング・メッセージを出力しません。
		通常出力	通常のワーニング・メッセージを出力します。
		詳細出力 (-w2)	詳細なワーニング・メッセージを出力します。

(8)［機能拡張］

機能拡張に関する詳細情報の表示，および設定の変更を行います。

C++ 形式のコメントを許可する	C++ 形式のコメント（“//”）使用を許可するかどうかを選択します。 コンパイラのオプション -zp に相当します。		
	デフォルト	はい (-zp)	
	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 (-zp)	C++ 形式のコメント使用を許可します。
		いいえ	C++ 形式のコメント使用を許可しません。

コメントのネストを許可する	コメント（“/* ... */”）のネスト使用を許可するかどうかを選択します。 コンパイラのオプション -zc 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 (-zc)	コメントのネスト使用を許可します。
		いいえ	コメントのネスト使用を許可しません。
ソースの漢字コード	ソースの漢字コードを選択します。 コンパイラのオプション -zs，-ze，-zn に相当します。		
	デフォルト	Shift_JIS(-zs)	
	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(-zs)	ソースの漢字コードを Shift_JIS と解釈します。
		EUC-JP(-ze)	ソースの漢字コードを EUC-JP と解釈します。
		なし (-zn)	ソースに漢字コードがないと解釈します。
ANSI 規格に準拠する	ANSI 規定外の機能を無効とし，ANSI 規定の一部の機能を有効とするかどうかを選択します。 コンパイラのオプション -za 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 (-za)	ANSI 規定外の機能を無効とし，ANSI 規定の一部の機能を有効とします。
		いいえ	ANSI 規定外の機能を有効とします。
関数の int 拡張を無効にする	関数の char/unsigned char 型引数および返り値を int 拡張しないかどうかを選択します。 コンパイラのオプション -zb 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 (-zb)	関数の char/unsigned char 型引数および返り値を int 拡張しません。
		いいえ	関数の char/unsigned char 型引数および返り値を int 拡張します。

RAM 配置用オブジェクトを出力する	コード，および ROM データを RAM 領域に配置するかどうかを選択します。 また，RAM 配置用ランタイム・ライブラリを配置する領域も指定します。 コンパイラのオプション -zx 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 (ランタイム・ライブラリを ROM 領域に配置)(-zx1)	コード，および ROM データを RAM 領域に配置し，ランタイム・ライブラリを ROM 領域に配置します。
		はい (ランタイム・ライブラリを RAM 領域に配置)(-zx2)	コード，および ROM データを RAM 領域に配置し，ランタイム・ライブラリを RAM 領域に配置します。 ［定型コードをライブラリに置き換える (サイズ優先最適化)］プロパティにおいて［はい (ライブラリに置き換えない)(-ql1)］を選択したものとします。
		いいえ	コード，および ROM データを RAM 領域に配置しません。

(9)［メモリ・モデル］

メモリ・モデ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・モデルの種類	メモリ・モデルの種類を指定します。 コンパイラのオプション -m に相当します。		
	デフォルト	ミディアム・モデル (Code 1M バイト /Data 64K バイト)(-mm)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	スモール・モデル (Code 64K バイト /Data 64K バイト)(-ms)	メモリ・モデルをスモール・モデルに指定します。
		ミディアム・モデル (Code 1M バイト /Data 64K バイト)(-mm)	メモリ・モデルをミディアム・モデルに指定します。
		ラージ・モデル (Code 1M バイト / Data 1M バイト)(-ml)	メモリ・モデルをラージ・モデルに指定します。
フラッシュ用オブジェクトを出力する	フラッシュ用オブジェクトを出力するかどうかを選択します。 コンパイラのオプション -zf 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 (-zf)	フラッシュ用オブジェクトを出力します。
		いいえ	フラッシュ用オブジェクトを出力しません。

フラッシュ領域の先頭アドレス	フラッシュ領域の先頭アドレスを 0x なしの 16 進数で指定します。 コンパイラのオプション -zz に相当します。 本プロパティを変更すると，［リンク・オプション］タブの［デバイス］カテゴリの［フラッシュ・スタート・アドレス］プロパティに同じ値を設定します。 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。		
	デフォルト	空欄	
	変更方法	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）	
フラッシュ領域分岐テーブルの先頭アドレス	フラッシュ領域分岐テーブルの先頭アドレスを 0x なしの 16 進数で指定します。 コンパイラのオプション -zt 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。		
	デフォルト	空欄	
	変更方法	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）	
ミラー領域指定	ミラー元領域を選択します。 MAA は，プロセッサ・モード・コントロール・レジスタ（PMC）のビット 0 です。 コンパイラのオプション -mi に相当します。 ［リンク・オプション］タブの［デバイス］カテゴリの［ミラー領域指定］プロパティを変更した場合，本プロパティも同じ値に変更されます。		
	デフォルト	MAA=0(-mi0)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	MAA=0(-mi0)	MAA に 0 を設定したものとします。
		MAA=1(-mi1)	MAA に 1 を設定したものとします。

（10）［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。

デバイス共通オブジェクトを出力する	デバイス共通オブジェクトを出力するかどうかを選択します。 コンパイラのオプション -common 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 (-common)	デバイス共通オブジェクトを出力します。
		いいえ	デバイス共通オブジェクトの出力を指定しません。

（11）［アセンブリ・ファイル］

アセンブリ・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	アセンブリ・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 -a，-sa，-li 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 (C ソース情報なし)(-a)	アセンブリ・ファイル（C ソース情報なし）を出力します。
		はい (C ソース情報あり (インクルード・ファイルを非展開))(-sa)	アセンブリ・ファイル（C ソース情報あり（インクルード・ファイルを非展開））を出力します。
		はい (C ソース情報あり (インクルード・ファイルを展開))(-sa,-li)	アセンブリ・ファイル（C ソース情報あり（インクルード・ファイルを展開））を出力します。
		いいえ	アセンブリ・ファイルを出力しません。

（12）［変数／関数情報ファイル］

変数／関数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。

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。

使用する変数／関数情報ファイル	saddr 領域への変数の配置と callt テーブル領域への関数の配置を指定する変数／関数情報ファイルです。 プロジェクトに登録している有効な変数／関数情報ファイルを検索し，そのファイル名を表示します。 コンパイラのオプション -ma に相当します。	
	デフォルト	*プロジェクトに登録している変数／関数情報ファイル名*
	変更方法	変更不可
ブート領域用変数／関数情報ファイル	ブート領域用のプロジェクトで使用する変数／関数情報ファイルを指定します。 コンパイラのオプション -ma 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，［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］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	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，ブート領域用変数／関数情報ファイルを指定 ダイアログによる編集
	指定可能値	259 文字までの文字列

（13）［データ制御］

データ制御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・フィールドを MSB から割り付ける	ビット・フィールド構造体のメンバを MSB から割り付けるかどうかを選択します。 コンパイラのオプション -rb 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 (-rb)	ビット・フィールド構造体のメンバを MSB から割り付けます。
		いいえ	ビット・フィールド構造体のメンバを LSB から割り付けます。
構造体メンバをパッキングする	構造体内の（2 バイト以上の）メンバを偶数番地に配置するためのアライン・データを挿入しないようにするかどうかを選択します。 コンパイラのオプション -rc 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 (-rc)	構造体内の（2 バイト以上の）メンバを偶数番地に配置するためのアライン・データを挿入しません。
		いいえ	構造体内の（2 バイト以上の）メンバを偶数番地に配置するためのアライン・データを挿入します。
間接参照を 1 バイト単位で行う	間接参照を 1 バイト単位で行うかどうかを選択します。 コンパイラのオプション -ra 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 (-ra)	間接参照を 1 バイト単位で行います。
		いいえ	間接参照を 1 バイト単位で行いません。

static 変数を saddr 領域に割り当てる	saddr 領域に配置させる static 変数の種類を選択します。 コンパイラのオプション -rs 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 (char サイズ)(-rs1)	char, unsigned char 型の自動変数を saddr 領域に配置します。
		はい (char,short,int サイズ)(-rs2)	char, unsigned char, short, unsigned short, int, unsigned int, enum, near ポインタ型の自動変数を saddr 領域に配置します。
		はい (char,short,int,long サイズ)(-rs4)	char, unsigned char, short, unsigned short, int, unsigned int, enum, long, unsigned long, ポインタ型の自動変数を saddr 領域に配置します。
		はい (構造体 , 共用体 , 配列)(-rsm)	構造体, 共用体, 配列型の自動変数を saddr 領域に配置します。
		はい (char サイズ , 構造体 , 共用体 , 配列)(-rs1m)	char, unsigned char, 構造体, 共用体, 配列型の自動変数を saddr 領域に配置します。
		はい (char,short,int サイズ , 構造体 , 共用体 , 配列)(-rs2m)	char, unsigned char, short, unsigned short, int, unsigned int, enum, near ポインタ, 構造体, 共用体, 配列型の自動変数を saddr 領域に配置します。
		はい (char,short,int,long サイズ , 構造体 , 共用体 , 配列)(-rs)	char, unsigned char, short, unsigned short, int, unsigned int, enum, long, unsigned long, ポインタ, 構造体, 共用体, 配列型の自動変数を saddr 領域に配置します。
		いいえ	static 変数を saddr 領域に配置しません。

外部変数を saddr 領域に割り当てる	saddr 領域に配置させる外部変数の種類を選択します。 コンパイラのオプション -rd に相当します。 なお，本プロパティは，［変数／関数情報ファイル］カテゴリの［使用する変数情報ファイル］プロパティにファイル名が設定されている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 (char サイズ)(-rd1)	char，unsigned char 型の自動変数を saddr 領域に配置します。
		はい (char,short,int サイズ)(-rd2)	char，unsigned char，short，unsigned short，int，unsigned int，enum，near ポインタ型の自動変数を saddr 領域に配置します。
		はい (char,short,int,long サイズ)(-rd4)	char，unsigned char，short，unsigned short，int，unsigned int，enum，long，unsigned long，ポインタ型の自動変数を saddr 領域に配置します。
		はい (構造体 , 共用体 , 配列)(-rdm)	構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		はい (char サイズ , 構造体 , 共用体 , 配列)(-rd1m)	char，unsigned char，構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		はい (char,short,int サイズ , 構造体 , 共用体 , 配列)(-rd2m)	char，unsigned char，short，unsigned short，int，unsigned int，enum，near ポインタ，構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		はい (char,short,int,long サイズ , 構造体 , 共用体 , 配列)(-rd)	char，unsigned char，short，unsigned short，int，unsigned int，enum，long，unsigned long，ポインタ，構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		いいえ	外部変数を saddr 領域に配置しません。
ROM データの配置先を指定する	ROM データの配置先を指定するかどうかを選択します。 コンパイラのオプション -rf，-rn 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 (far 領域)(-rf)	ROM データを far 領域に配置します。
		はい (near 領域)(-rn)	ROM データを near 領域に配置します。
		いいえ	ROM データの配置先を指定しません。

(14)［リスト・ファイル］

リスト・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。

プリプロセス・リスト・ファイルを出力する	プリプロセス・リスト・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 -p 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 (-p)	プリプロセス・リスト・ファイルを出力します。
		いいえ	プリプロセス・リスト・ファイルを出力しません。
コメントを出力しない	プリプロセス・リスト・ファイル中にコメントを出力しないかどうかを選択します。 コンパイラのオプション -kc に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kc)	プリプロセス・リスト・ファイル中にコメントを出力しません。
		いいえ	プリプロセス・リスト・ファイル中にコメントを出力します。
#define を展開する	プリプロセス・リスト・ファイル中に #define 文を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 -kd に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kd)	プリプロセス・リスト・ファイル中に #define 文を展開します。
		いいえ	プリプロセス・リスト・ファイル中に #define 文を展開しません。
#if，#ifdef，#ifndef を展開する	プリプロセス・リスト・ファイル中に #if，#ifdef，#ifndef 文を展開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 -kf に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	デフォルト	はい (-kf)	
	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 (-kf)	プリプロセス・リスト・ファイル中に #if，#ifdef，#ifndef 文を展開して出力します。
		いいえ	プリプロセス・リスト・ファイル中に #if，#ifdef，#ifndef 文を展開しません。

#include を展開する	プリプロセス・リスト・ファイル中に #include 文を展開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 -ki に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-ki)	プリプロセス・リスト・ファイル中に #include 文を展開して出力します。
		いいえ	プリプロセス・リスト・ファイル中に #include 文を展開しません。
#line を展開する	プリプロセス・リスト・ファイル中に #line 文を展開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 -kl に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	デフォルト	はい (-kl)	
	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 (-kl)	プリプロセス・リスト・ファイル中に #line 文を展開して出力します。
		いいえ	プリプロセス・リスト・ファイル中に #line 文を展開しません。
行番号を出力する	プリプロセス・リスト・ファイル中に行番号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 -kn に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	デフォルト	はい (-kn)	
	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 (-kn)	プリプロセス・リスト・ファイル中に行番号を出力します。
		いいえ	プリプロセス・リスト・ファイル中に行番号を出力しません。
エラー・リスト・ファイルを出力する	エラー・リスト・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 -e，-se 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 (C ソースなし)(-e)	エラー・リスト・ファイル（C ソースなし）を出力します。
		はい (C ソースあり)(-se)	エラー・リスト・ファイル（C ソースあり）を出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

クロスリファレンス・リスト・ファイルを出力する	クロスリファレンス・リスト・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 -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 (-x)	クロスリファレンス・リスト・ファイルを出力します。
		いいえ	クロスリファレンス・リスト・ファイルを出力しません。
改ページ・コードを出力する	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）の末尾に改ページ・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 -lf に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］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 (-lf)	リスト・ファイルの末尾に改ページ・コードを出力します。
		いいえ	リスト・ファイルの末尾に改ページ・コードを出力しません。
1 行文字数	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）の各行の文字数を指定します。 コンパイラのオプション -lw に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］を選択した場合のみ表示されます。		
	デフォルト	132	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	72 ～ 132（10 進数）	
1 ページ行数	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）の 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 コンパイラのオプション -ll に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］を選択した場合のみ表示されます。		
	デフォルト	0	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0，20 ～ 65535（10 進数）	

タブ幅	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）のタブ幅を指定します。 コンパイラのオプション -lt に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］を選択した場合のみ表示されます。	
	デフォルト	8
	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 ～ 8（10 進数）

（15）［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します。	
	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します。	
	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。

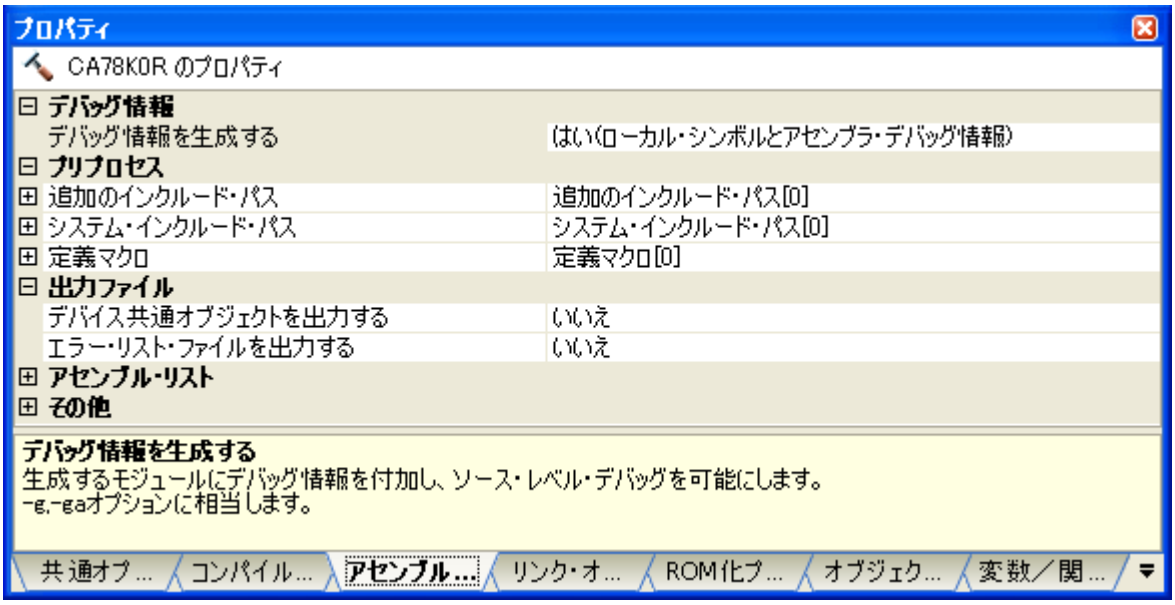
その他の追加オプション	その他に追加するコンパイラのオプションを入力します。 なお，ここで設定したオプションは，コンパイラのオプション群の最後に付加されます。	
	デフォルト	空欄
	変更方法	テキスト・ボックスによる直接入力，または［...］ボタンをクリックし，文字列入力 ダイアログによる編集
	指定可能値	259 文字までの文字列

［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。

（1）［デバッグ情報］
（2）［プリプロセス］
（3）［出力ファイル］
（4）［アセンブル・リスト］
（5）［その他］

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	生成するモジュー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，-ga に相当します。		
	デフォルト	はい (ローカル・シンボルとアセンブラ・デバッグ情報)	
	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 (アセンブラ・デバッグ情報)(-ng,-ga)	生成するオブジェクト・モジュール・ファイルにデバッグ情報 (アセンブラ・デバッグ情報) を付加します。
		はい (ローカル・シンボルとアセンブラ・デバッグ情報)	生成するオブジェクト・モジュール・ファイルにデバッグ情報 (ローカル・シンボルとアセンブラ・デバッグ情報) を付加します。
		いいえ (-ng,-nga)	生成するオブジェクト・モジュール・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。 本プロパティを省略した場合，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します。 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一番最初に追加します。 インクルード・パスに大文字，小文字の区別はありません。	
	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。 また，［システム・インクルード・パス］プロパティ，および［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］プロパティの指定数との合計が 64 個を越えた場合，ビルドの実行時にエラーとなります。

システム・インクルード・パス	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。		
	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。 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します。		
	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 30 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。

デバイス共通オブジェクトを出力する	各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。 アセンブラのオプション -common 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 (-common)	各デバイス共通のオブジェクトを出力します。
		いいえ	RL78，および 78K0R に対応したオブジェクトを出力します。
エラー・リスト・ファイルを出力する	エラー・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 -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)	エラー・リスト・ファイルを出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

(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 -p に相当します。		
	デフォルト	はい (-p)	
	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)	アセンブル・リスト・ファイルを出力します。
		いいえ (-np)	アセンブル・リスト・ファイルを出力しません。
リスト・コンバータを実行する	実行モジュールを生成した後にリスト・コンバータを実行するかどうかを選択します。 ただし，ライブラリ生成時には実行しません。 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［いいえ (-np)］を選択した場合は表示されません。		
	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	実行モジュールを生成した後にリスト・コンバータを実行します。
		いいえ	実行モジュールを生成した後にリスト・コンバータを実行しません。
リスト・コンバータのエラー・リスト・ファイルを出力する	リスト・コンバータを実行時に，エラー・リスト・ファイルを出力するかどうかを選択します。 リスト・コンバータのオプション -e に相当します。 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［いいえ (-np)］を選択した場合，［リスト・コンバータを実行する］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 (-e)	リスト・コンバータを実行時に，エラー・リスト・ファイルを出力します。
		いいえ	リスト・コンバータを実行時に，エラー・リスト・ファイルを出力しません。
アセンブル・リストを出力する	アセンブル・リスト・ファイル中にアセンブル・リスト情報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 -ka に相当します。 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［いいえ (-np)］を選択した場合は表示されません。		
	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	アセンブル・リスト・ファイル中にアセンブル・リスト情報を出力します。
		いいえ (-nka)	アセンブル・リスト・ファイル中にアセンブル・リスト情報を出力しません。

シンボル・リストを出力する	アセンブル・リスト・ファイル中にシンボル・リスト情報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 -ks に相当します。 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［いいえ (-np)］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 (-ks)	アセンブル・リスト・ファイル中にシンボル・リスト情報を出力します。
		いいえ	アセンブル・リスト・ファイル中にシンボル・リスト情報を出力しません。
クロスリファレンス・リストを出力する	アセンブル・リスト・ファイル中にクロスリファレンス・リスト情報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 -kx に相当します。 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［いいえ (-np)］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 (-kx)	アセンブル・リスト・ファイル中にクロスリファレンス・リスト情報を出力します。
		いいえ	アセンブル・リスト・ファイル中にクロスリファレンス・リスト情報を出力しません。
改ページ・コードを出力する	リスト・ファイルの末尾に改ページ・コード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 -lf に相当します。 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［いいえ (-np)］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 (-lf)	リスト・ファイルの末尾に改ページ・コードを出力します。
		いいえ	リスト・ファイルの末尾に改ページ・コードを出力しません。
1 行文字数	リスト・ファイルの各行の文字数を指定します。 アセンブラのオプション -lw に相当します。 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［いいえ (-np)］を選択した場合は表示されません。		
	デフォルト	132	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	72 ～ 2046（10 進数）	

1 ページ行数	リスト・ファイルの 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 アセンブラのオプション -ll に相当します。 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［いいえ (-np)］を選択した場合は表示されません。	
	デフォルト	0
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0，20 ～ 32767（10 進数）
タブ幅	リスト・ファイルのタブ幅を指定します。 アセンブラのオプション -lt に相当します。 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［いいえ (-np)］を選択した場合は表示されません。	
	デフォルト	8
	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 ～ 8（10 進数）
ヘッダ文字列	アセンブル・リスト・ファイルのヘッダを指定します。 全角文字，および半角スペースを含む文字列も指定可能です。 アセンブラのオプション -lh に相当します。 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［いいえ (-np)］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	半角 60 文字（全角 30 文字）までの文字列

(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。

ソースの漢字コード	ソースの漢字コードを選択します。 アセンブラのオプション -zs，-ze，-zn に相当します。		
	デフォルト	Shift_JIS(-zs)	
	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(-zs)	ソースの漢字コードを Shift_JIS と解釈します。
		EUC-JP(-ze)	ソースの漢字コードを EUC-JP と解釈します。
		なし (-zn)	ソースに漢字コードがないと解釈します。

78K0 互換マクロを使用する	78K0 マイコン用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル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 -compati 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 (-compati)	78K0 マイコン用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。
		いいえ	78K0 マイコン用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にしません。
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します。		
	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。	
アセンブル後に実行するコマンド	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します。		
	デフォルト	アセンブ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

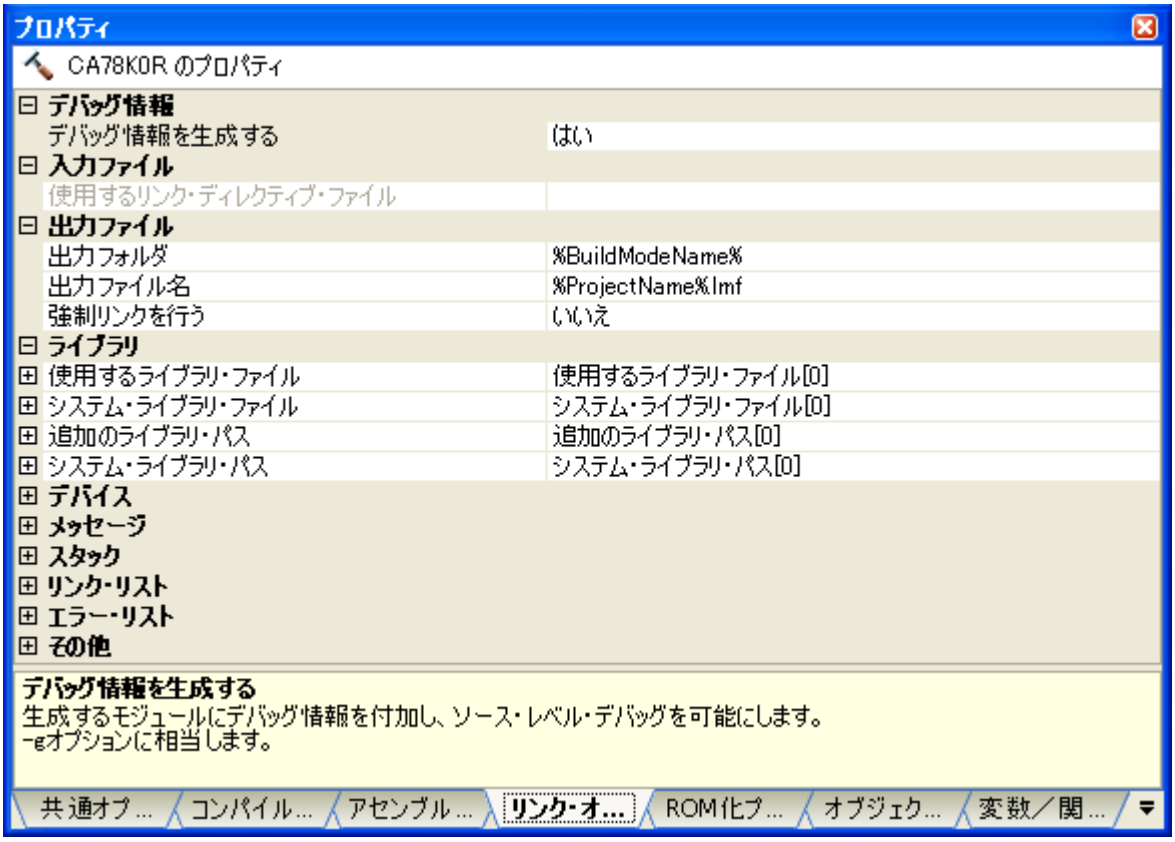
［リンク・オプション］タブ

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。

（1）［デバッグ情報］
（2）［入力ファイル］
（3）［出力ファイル］
（4）［ライブラリ］
（5）［デバイス］
（6）［メッセージ］
（7）［スタック］
（8）［リンク・リスト］
（9）［エラー・リスト］
（10）［その他］

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。

デバッグ情報を生成する	生成するモジュー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。 リンカのオプション -g 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	生成するオブジェクト・モジュール・ファイルにデバッグ情報を付加します。
		いいえ (-ng)	生成するオブジェクト・モジュール・ファイルにデバッグ情報を付加しません。

(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。

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	*プロジェクトに追加されたリンク・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。	
	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	出力するロード・モジュールのファイル名を指定します。 拡張子は“.lmf”を指定してください。拡張子を省略した場合は，“.lmf”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%.lmf”を指定したものとみなします。		
	デフォルト	%ProjectName%.lmf	
	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	
強制リンクを行う	リンク時にエラーがある場合でも，強制的にロード・モジュール・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 -j 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 (-j)	リンク時にエラーがある場合でも，強制的にロード・モジュール・ファイルを生成します。
		いいえ	リンク時にエラーがある場合，ロード・モジュール・ファイルを生成しません。

(4)［ライブラリ］

ライブラ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。

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）を指定します。 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 -b 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します。		
	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	259 文字までの文字列 64 個まで指定可能です。 また，［システム・ライブラリ・ファイル］プロパティ，および［コンパイル・オプション］タブの［ライブラリ］カテゴリの［使用する標準ライブラリ］プロパティの指定数との合計が 64 個を越えた場合，ビルドの実行時にエラーとなります。	
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 -i 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します。		
	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 64 個まで指定可能です。 また，［システム・ライブラリ・パス］プロパティ，および［コンパイル・オプション］タブの［ライブラリ］カテゴリの［使用する標準ライブラリ］プロパティの指定数との合計が 64 個を越えた場合，ビルドの実行時にエラーとなります。	
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 -i 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	

(5)［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。

オンチップ・デバッグを設定する	オンチップ・デバッグを設定するかどうかを選択します。 オンチップ・デバッグの制御値，デバッグ・モニタ領域の開始アドレス，およびサイズを変更します。 リンカのオプション -go に相当します。 なお，本プロパティは，オンチップ・デバッグ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	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 (-go)	オンチップ・デバッグを設定します。
		いいえ	オンチップ・デバッグを設定しません。

オンチップ・デバッグ・オプション・バイト制御値	オンチップ・デバッグ・オプション・バイトの制御値を 0x なしの 16 進数で指定します。 リンカのオプション -go 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，オンチップ・デバッグ機能を持たないデバイスの場合，および［オンチップ・デバッグを設定する］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	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）	
デバッグ・モニタ領域開始アドレス	デバッグ・モニタ領域の開始アドレスを 0x なしの 16 進数で指定します。 リンカのオプション -go に相当します。 空欄の場合は，“内部 ROM のエンド・アドレス－ 1024 ＋ 1”を指定したものとみなします。 なお，本プロパティは，［オンチップ・デバッグを設定する］プロパティが非表示の場合，または［いいえ］を選択した場合は表示されません。		
	デフォルト	内部 ROM のエンド・アドレス－ 1024 ＋ 1	
	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 ～ FFFFF（16 進数）	
デバッグ・モニタ領域サイズ [バイト]	デバッグ・モニタ領域のサイズを 10 進数で指定します。 リンカのオプション -go に相当します。 空欄の場合は，エラーとなります。 なお，本プロパティは，［オンチップ・デバッグを設定する］プロパティが非表示の場合，または［いいえ］を選択した場合は表示されません。		
	デフォルト	512【RL78】 1024【78K0R】	
	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 ～ 1024（10 進数）【RL78】 88 ～ 1024（10 進数）【78K0R】	
ユーザ・オプション・バイトを設定する	ユーザ・オプション・バイトを設定するかどうかを選択します。 リンカのオプション -gb 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 (-gb)	ユーザ・オプション・バイトを設定します。 ただし，［ユーザ・オプション・バイト値］プロパティが空欄の場合は，ユーザ・オプション・バイトを設定しません。
		いいえ	ユーザ・オプション・バイトを設定しません。

ユーザ・オプション・バイト値	ユーザ・オプション・バイト値を 0x なしの 16 進数で指定します。 リンカのオプション -gb 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，［ユーザ・オプション・バイトを設定する］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	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（指定可能な範囲は選択したデバイスに依存します）	
ミラー領域指定	RAM 空間にミラーされるセグメントを配置する領域を設定するかどうかを選択します。 リンカのオプション -mi に相当します。 ［コンパイル・オプション］タブの［出力ファイル］カテゴリの［ミラー領域指定］プロパティを変更した場合，本プロパティも同じ値に変更されます。		
	デフォルト	MAA=0(-mi0)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	MAA=0(-mi0)	MAA に 0 を設定したときにミラーされる領域にセグメントを配置します。
		MAA=1(-mi1)	MAA に 1 を設定したときにミラーされる領域にセグメントを配置します。
64K バイト境界配置指定	各 64K バイト領域の境界の最後の 1 バイトにセグメントの配置を行うかどうかを選択します。 リンカのオプション -ccza に相当します。 なお，本プロパティは，プロジェクトに C ソース・ファイルを登録し，［コンパイル・オプション］タブの［機能拡張］カテゴリの［ANSI 規格に準拠する］プロパティで［いいえ］を選択している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 (-ccza)	各 64K バイト領域の境界の最後の 1 バイトにセグメントの配置を行います。
		いいえ	各 64K バイト領域の境界の最後の 1 バイトにセグメントの配置を行いません。

フラッシュ・スタート・アドレスを設定する	フラッシュ ROM 内蔵製品用フラッシュ・スタート・アドレスを設定するかどうかを選択します。 リンカのオプション -zb に相当します。 フラッシュ ROM 領域セルフ書き換え機能を持たないデバイスでは，本プロパティは設定しないでください。 なお，本プロパティは，［コンパイル・オプション］タブの［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］プロパティで［はい (-zf)］を選択した場合は［いいえ］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 (-zb)	フラッシュ ROM 内蔵製品用フラッシュ・スタート・アドレスを設定します。
		いいえ	フラッシュ ROM 内蔵製品用フラッシュ・スタート・アドレスを設定しません。
フラッシュ・スタート・アドレス	フラッシュ ROM 内蔵製品用フラッシュ・スタート・アドレスを 0x なしの 16 進数で指定します。 リンカのオプション -zb に相当します。 フラッシュ ROM 領域セルフ書き換え機能を持たないデバイスでは，本プロパティは設定しないでください。 本プロパティを変更すると，［コンパイル・オプション］タブの［メモリ・モデル］カテゴリの［フラッシュ領域の先頭アドレス］プロパティに同じ値を設定します。 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，［フラッシュ・スタート・アドレスを設定する］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	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）	
ブート領域用ロード・モジュール・ファイル名	フラッシュ領域用のロード・モジュール・ファイルを作成する場合に，ブート領域ロード・モジュール・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。 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。 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）。 本プロパティを指定した場合，［共通オプション］タブの［デバイス］カテゴリの［セキュリティ ID］プロパティの設定は無効となります。		
	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，または［...］ボタンをクリックし，ブート領域用ロード・モジュール・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	

セルフ RAM 領域への配置を制御する	セルフ RAM 領域への配置を制御するかどうかを選択します。 リンカのオプション -self，-selfw に相当します。 フラッシュ ROM 領域セルフ書き換え機能を持たないデバイスでは，本プロパティは設定しないでください。		
	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 (エラーを表示)(-self)	セルフ RAM 領域への配置を禁止し，エラーを出力します。
		はい (警告を表示)(-selfw)	セルフ RAM 領域への配置時に，ワーニングを出力します。
		いいえ	セルフ RAM 領域を内部 RAM 領域として使用します。

(6)［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。

警告表示レベル	リンク時の警告表示レベルを選択します。 リンカのオプション -w に相当します。		
	デフォルト	通常出力	
	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 (-w0)	ワーニング・メッセージを出力しません。
		通常出力	通常のワーニング・メッセージを出力します。
		詳細出力 (-w2)	詳細なワーニング・メッセージを出力します。

(7)［スタック］

スタックに関する詳細情報の表示，および設定の変更を行います。

スタック解決用シンボルを生成する	スタック解決用シンボ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 -s に相当します。 なお，本プロパティは，プロジェクトに C ソース・ファイルを登録し，［コンパイル・オプション］タブの［スタートアップ］カテゴリの［標準のスタートアップ・ルーチンを使用する］プロパティで［はい］を選択した場合は常に［はい (-s)］となり，変更はできません。		
	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 (-s)	スタック解決用シンボルを生成します。
		いいえ	スタック解決用シンボルを生成しません。

領域名	スタック解決用シンボルを生成するメモリ領域名を指定します。 領域名を省略した場合，RAM が指定されたものとします。 リンカのオプション -s に相当します。 なお，本プロパティは，［スタック解決用シンボルを生成する］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	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	256 文字までの文字列	

(8)［リンク・リスト］

リンク・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。

リンク・リスト・ファイルを出力する	リンク・リスト・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 -p 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	リンク・リスト・ファイルを出力します。
		いいえ (-np)	リンク・リスト・ファイルを出力しません。
リンク・ディレクティブを出力する	リンク・リスト・ファイルにリンク・ディレクティブ情報を出力するかどうかを選択します。 リンカのオプション -kd に相当します。 なお，本プロパティは，［リンク・リスト・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。		
	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	リンク・リスト・ファイルにリンク・ディレクティブ情報を出力します。
		いいえ (-nkd)	リンク・リスト・ファイルにリンク・ディレクティブ情報を出力しません。
ローカル・シンボル・リストを出力する	リンク・リスト・ファイルにローカル・シンボル・リスト情報を出力するかどうかを選択します。 リンカのオプション -kl に相当します。 なお，本プロパティは，［リンク・リスト・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。		
	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 (-kl)	リンク・リスト・ファイルにローカル・シンボル・リスト情報を出力します。
		いいえ	リンク・リスト・ファイルにローカル・シンボル・リスト情報を出力しません。

パブリック・シンボル・リストを出力する	リンク・リスト・ファイルにパブリック・シンボル・リスト情報を出力するかどうかを選択します。 リンカのオプション -kp に相当します。 なお，本プロパティは，［リンク・リスト・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。		
	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 (-kp)	リンク・リスト・ファイルにパブリック・シンボル・リスト情報を出力します。
		いいえ	リンク・リスト・ファイルにパブリック・シンボル・リスト情報を出力しません。
マップ・リストを出力する	リンク・リスト・ファイルにマップ・リスト情報を出力するかどうかを選択します。 リンカのオプション -km に相当します。 なお，本プロパティは，［リンク・リスト・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。		
	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	リンク・リスト・ファイルにマップ・リスト情報を出力します。
		いいえ (-nkm)	リンク・リスト・ファイルにマップ・リスト情報を出力しません。
改ページ・コードを出力する	リンク・リスト・ファイルの末尾に改ページ・コードを出力するかどうかを選択します。 リンカのオプション -lf に相当します。 なお，本プロパティは，［リンク・リスト・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。		
	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 (-lf)	リンク・リスト・ファイルの末尾に改ページ・コードを出力します。
		いいえ	リンク・リスト・ファイルの末尾に改ページ・コードを出力しません。
1 ページ行数	リンク・リスト・ファイルの 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 リンカのオプション -ll に相当します。 なお，本プロパティは，［リンク・リスト・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。		
	デフォルト	0	
	変更方法	テキスト・ボックスによる直接入力	
	指定可能値	0，20 ～ 32767（10 進数）	

(9)［エラー・リスト］

エラー・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。

エラー・リストを出力する	エラー・リスト・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 -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)	エラー・リスト・ファイルを出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

（10）［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。

リンク前に実行するコマンド	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します。	
	デフォルト	リンク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。
リンク後に実行するコマンド	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します。	
	デフォルト	リンク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

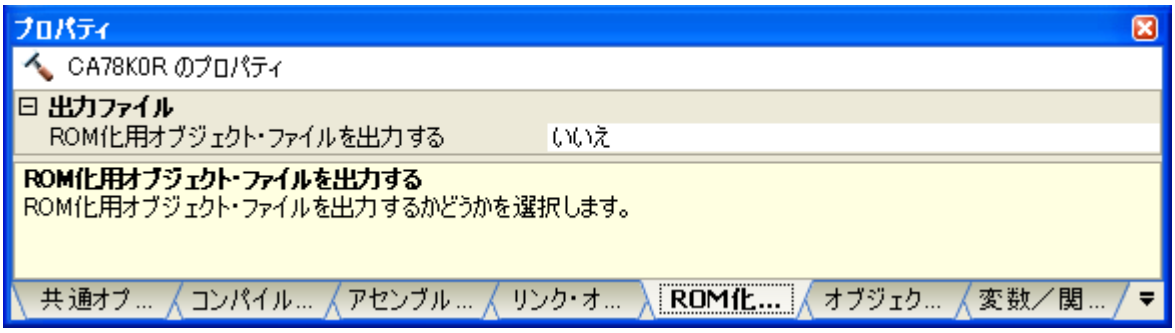
[ROM 化プロセス・オプション] タブ

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。

(1) [出力ファイル]
(2) [リンク・マップ]
(3) [エラー・リスト]
(4) [その他]

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。		
	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	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］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	259 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。 “.lmf”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，“.lmf”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］を選択した場合のみ表示されます。	
	デフォルト	romp.lmf
	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
コピー・ルーチンの先頭アドレス	コピー・ルーチンの先頭アドレスを 0x なしの 16 進数（例：100A0）で指定します。 ROM 化プロセッサのオプション -rc に相当します。 空欄の場合は，自動的に先頭アドレスを決定します。 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］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	0 ～ *プログラム空間の最大アドレス*

ROM 化領域開始アドレス	ROM 化の対象となる領域の先頭アドレスを 0x なしの 16 進数（例：100A0）で指定します。 本プロパティを指定した場合は，［ROM 化領域サイズ [バイト]］プロパティも指定してください。 ROM 化プロセッサのオプション -ra に相当します。 空欄の場合は，“0”を指定したものとみなします。 ［ROM 化領域サイズ [バイト]］プロパティも空欄の場合は，内部 RAM の範囲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］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	0 ～ *プログラム空間の最大アドレス*
ROM 化領域サイズ [バイト]	ROM 化の対象となる領域の先頭アドレスからのサイズを 16 進数（例：F00）で指定します。 ROM 化プロセッサのオプション -ra に相当します。 空欄の場合は，“0”を指定したものとみなします。 ［ROM 化領域サイズ [バイト]］プロパティを変更した結果，本プロパティの値が指定可能値の範囲に入らない場合は，本プロパティは空欄となります。 なお，本プロパティは，［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］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	1 ～ *プログラム空間の最大アドレス－充てん開始アドレス*＋ 0x1

(2)［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。

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。

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。 ROM 化プロセッサのオプション -p に相当します。 出力ファイル名は，［ROM 化用オブジェクト・ファイル名］プロパティで指定したファイル名の拡張子を“.map”で置き換えたファイル名となります。 その際，同名のファイルが存在する場合は削除します。		
	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	リンク・マップ・ファイルを出力します。
		いいえ (-np)	リンク・マップ・ファイルを出力しません。

ローカル・シンボル・リストを出力する	リンク・マップ・ファイルにローカル・シンボル・リスト情報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 -kl に相当します。 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［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 (-kl)	リンク・マップ・ファイルにローカル・シンボル・リスト情報を出力します。
		いいえ	リンク・マップ・ファイルにローカル・シンボル・リスト情報を出力しません。
パブリック・シンボル・リストを出力する	リンク・マップ・ファイルにパブリック・シンボル・リスト情報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 -kp に相当します。 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［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 (-kp)	リンク・マップ・ファイルにパブリック・シンボル・リスト情報を出力します。
		いいえ	リンク・マップ・ファイルにパブリック・シンボル・リスト情報を出力しません。
マップ・リストを出力する	リンク・マップ・ファイルにマップ・リスト情報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 -km に相当します。 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［いいえ］を選択した場合は表示されません。		
	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	リンク・マップ・ファイルにマップ・リスト情報を出力します。
		いいえ (-nkm)	リンク・マップ・ファイルにマップ・リスト情報を出力しません。
改ページ・コードを出力する	リンク・マップ・ファイルの末尾に改ページ・コード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 -lf に相当します。 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［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 (-lf)	リンク・マップ・ファイルの末尾に改ページ・コードを出力します。
		いいえ	リンク・マップ・ファイルの末尾に改ページ・コードを出力しません。

1 ページ行数	リンク・マップ・ファイルの 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 ROM 化プロセッサのオプション -ll に相当します。 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［いいえ］を選択した場合は表示されません。	
	デフォルト	0
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0，20 ～ 32767（10 進数）

(3)［エラー・リスト］

エラー・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。

エラー・リストを出力する	エラー・リスト・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 -e に相当します。 出力ファイル名は，［ROM 化用オブジェクト・ファイル名］プロパティで指定したファイル名の拡張子を“.erp”で置き換えたファイル名となります。		
	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 (-e)	エラー・リスト・ファイルを出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

(4)［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。

ROM 化前に実行するコマンド	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します。	
	デフォルト	ROM 化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。

ROM 化後に実行するコマンド	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します。	
	デフォルト	ROM 化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 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。	
	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

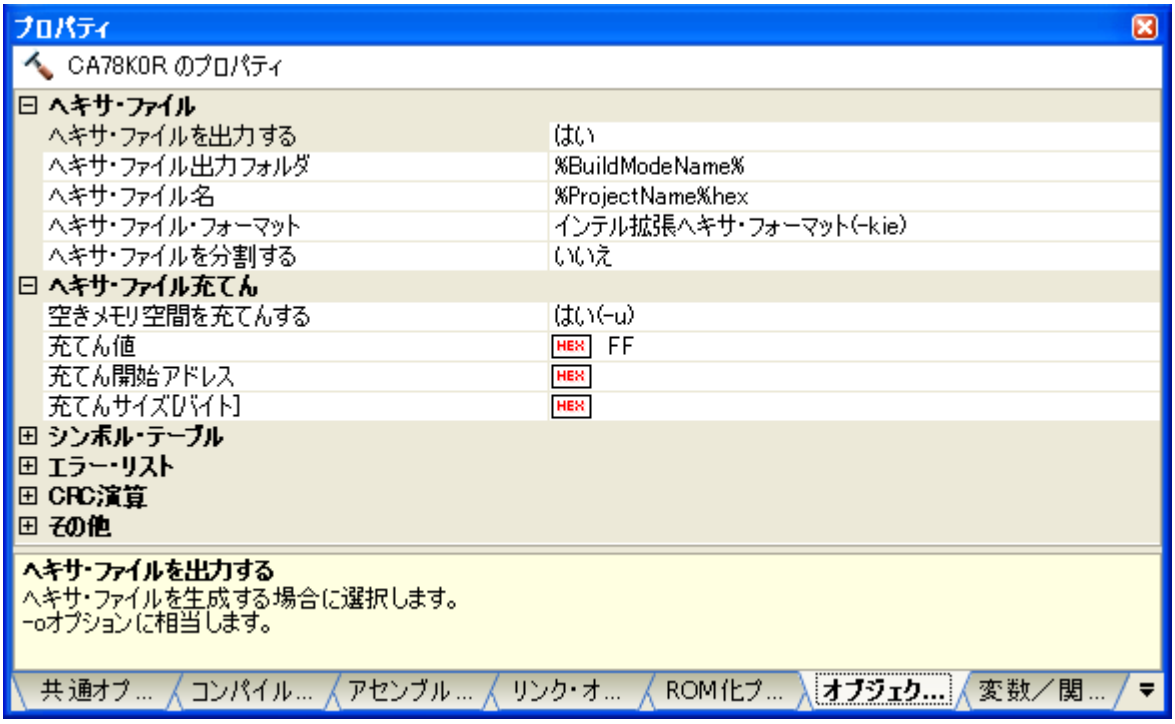
［オブジェクト・コンバート・オプション］タブ

本タブでは，オブジェクト・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。

（1）［ヘキサ・ファイル］
（2）［ヘキサ・ファイル充てん］
（3）［シンボル・テーブル］
（4）［エラー・リスト］
（5）［CRC 演算］
（6）［その他］

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：［オブジェクト・コンバート・オプション］タブ



[各カテゴリの説明]

(1) [ヘキサ・ファイル]

ヘキサ・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。 オブジェクト・コンバータのオプション -o 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・ファイルを出力します。
		いいえ (-no)	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。 オブジェクト・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 [いいえ (-no)] 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	
ヘキサ・ファイル名	ヘキサ・ファイル名を指定します。 オブジェクト・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 [いいえ (-no)] を選択した場合は表示されません。		
	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。 オブジェクト・コンバータのオプション -k 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［いいえ (-no)］を選択した場合は表示されません。		
	デフォルト	インテル拡張ヘキサ・フォーマット (-kie)	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	インテル標準ヘキサ・フォーマット (-ki)	生成するヘキサ・ファイルをインテル標準ヘキサ・フォーマットとします。
		インテル拡張ヘキサ・フォーマット (-kie)	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)(-km)	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)(-kme)	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 (-kt)	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・ファイルを分割する	フラッシュ ROM 内蔵製品のブート領域 ROM プログラムのリンク指定時において，ブート領域とそれ以外の領域を別のヘキサ・ファイルに分割出力するかどうかを選択します。 オブジェクト・コンバータのオプション -zf に相当します。 フラッシュ ROM 領域セルフ書き換え機能を持たないデバイスでは，本プロパティは設定しないでください。 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［いいえ (-no)］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 (-zf)	ブート領域とそれ以外の領域を別のヘキサ・ファイルに分割出力します。
		いいえ	ブート領域とそれ以外の領域を別のヘキサ・ファイルに分割出力しません。

(2)［ヘキサ・ファイル充てん］

ヘキサ・ファイル充てん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">空きメモリ空間を充てんする</td><td colspan="3">ヘキサ形式オブジェクトが出力されていないアドレスには，不要なコードが書き込まれることがあります。このようなアドレスにをアクセスした場合にプログラムが暴走するのを防ぐために，あらかじめコードを書き込むかどうかを選択します。
オブジェクト・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］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 (-no)］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">はい (-u)</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 (-u)</td><td>ヘキサ形式オブジェクトが出力されていないアドレスにあらかじめコードを書き込みます。</td></tr>
<tr><td>いいえ (-nu)</td><td>ヘキサ形式オブジェクトが出力されていないアドレスにあらかじめコードを書き込みません。</td></tr>
<tr><td rowspan="4">充てん値</td><td colspan="3">ヘキサ形式オブジェクトが出力されていないアドレスに書き込む数値を 0x なしの 16 進数（例：FF）で指定します。
空欄の場合，“FF”が指定されたものとします。
オブジェクト・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，［空きメモリ空間を充てんする］プロパティで［いいえ (-nu)］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">FF</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 ～ FF（16 進数）</td></tr>
<tr><td rowspan="4">充てん開始アドレス</td><td colspan="3">充てんの対象となる先頭アドレスを 0x なしの 16 進数（例：100A0）で指定します。
空欄の場合，0 が指定されたものとします。
本プロパティを指定した場合は，［充てんサイズ [バイト]］プロパティも指定してください。［充てんサイズ [バイト]］プロパティが空欄の場合，本プロパティの設定は無効となります。
オブジェクト・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，［空きメモリ空間を充てんする］プロパティで［いいえ (-nu)］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</td></tr>
<tr><td>指定可能値</td><td colspan="2">0 ～ プログラム空間の最大アドレス（16 進数）</td></tr>
</table>

充てんサイズ [バイト]	充てんの対象となる先頭アドレスからのサイズを 16 進数（例：F00）で指定します。 ［充てん開始アドレス］プロパティを変更した結果，本プロパティの値が指定可能値の範囲に入らない場合は，本プロパティは空欄となります。 オブジェクト・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，［空きメモリ空間を充てんする］プロパティで［いいえ (-nu)］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	1 ～ *プログラム空間の最大アドレス－充てん開始アドレス*＋ 0x1（16 進数） ただし，コンパイラでの範囲チェックはさらに厳格なため，ビルド時の実際の上限値はこの値より小さい場合があり，その場合はリンク時にエラーとなります。

(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。

シンボル・テーブル・ファイルを出力する	シンボル・テーブル・ファイルを出力するかどうかを選択します。 オブジェクト・コンバータのオプション -s に相当します。		
	デフォルト	はい (-s)	
	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)	シンボル・テーブル・ファイルを出力します。
		いいえ (-ns)	シンボル・テーブル・ファイルを出力しません。

(4)［エラー・リスト］

エラー・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。

エラー・リスト・ファイルを出力する	エラー・リスト・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 -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)	エラー・リスト・ファイルを出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

(5)［CRC 演算］

CRC 演算に関する詳細情報の表示，および設定の変更を行います。

CRC 演算を行う	CRC（Cyclic Redundancy Check）演算を行うかどうかを選択します。 オブジェクト・コンバータのオプション -crc 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 (-crc)	指定した範囲のヘキサ形式オブジェクトを，下位アドレスから上位アドレスの順で CRC 演算を行い，演算結果を指定したアドレスへ出力します。
		いいえ	CRC 演算，および演算結果の出力を行いません。
CRC 結果出力アドレス	CRC 演算の結果を出力するアドレスを 0x なしの 16 進数（例：FFF00）で指定します。 本プロパティは必ず指定してください。 オブジェクト・コンバータのオプション -crc に相当します。 なお，本プロパティは，［CRC 演算を行う］プロパティで［はい (-crc)］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	0 ～ FFF00（16 進数）	
CRC 演算の範囲	CRC 演算の対象となる範囲を，開始アドレス，終了アドレスの順で指定します（例：0h-0FFh）。 A ～ F で始まるアドレスの先頭には“0”を付けてください。 アドレスの末尾には“h”を付けてください。 カンマで区切ることにより，複数指定することができます（例：0h-0FFh,400h-4FFh）。 本プロパティは必ず指定してください。 オブジェクト・コンバータのオプション -crc に相当します。 なお，本プロパティは，［CRC 演算を行う］プロパティで［はい (-crc)］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	0h ～ 0FFF00h（16 進数）	
CRC 演算方法	CRC 演算方法を選択します。 各演算方法の詳細については，デバイスのユーザーズ・マニュアルを参照してください。 オブジェクト・コンバータのオプション -crc に相当します。 なお，本プロパティは，［CRC 演算を行う］プロパティで［はい (-crc)］を選択した場合のみ表示されます。		
	デフォルト	高速 CRC	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	高速 CRC	高速 CRC（high-speed CRC）用の演算結果を出力します。 CRC 演算の初期値は，0H となります。
		汎用 CRC	汎用 CRC（general-purpose CRC）用の演算結果を出力します。

CRC 演算の初期値	CRC 演算の初期値を 0x なしの 16 進数（例：FFFF）で指定します。 空欄の場合，0 が指定されたものとします。 オブジェクト・コンバータのオプション -crc に相当します。 なお，本プロパティは，［CRC 演算を行う］プロパティで［はい (-crc)］を選択した場合，および［CRC 演算方法］プロパティで［汎用 CRC］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	0 ～ FFFF（16 進数）

（6）［その他］

オブジェクト・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。

オブジェクト・コンバート前に実行するコマンド	オブジェクト・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%　：オブジェクト・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。 %ObjectConvertedFile%　：オブジェクト・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します。	
	デフォルト	オブジェクト・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。

オブジェクト・コンバート後に実行するコマンド	オブジェクト・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%　：オブジェクト・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。 %ObjectConvertedFile%　：オブジェクト・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します。	
	デフォルト	オブジェクト・コンバート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	その他に追加するオブジェクト・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，オブジェクト・コンバ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

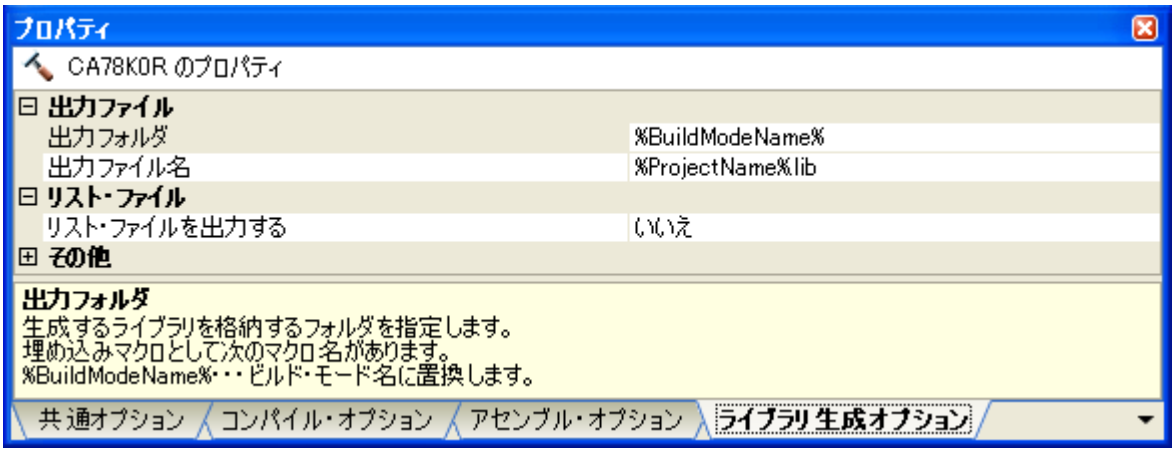
［ライブラリ生成オプション］タブ

本タブでは，ライブラリアンに対して，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）［出力ファイル］
（2）［リスト・ファイル］
（3）［その他］

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：［ライブラリ生成オプション］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	生成するライブラリ・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	259 文字までの文字列

出力ファイル名	出力するライブラリのファイル名を指定します。 拡張子は“.lib”を指定してください。拡張子を省略した場合は，“.lib”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	%ProjectName%.lib
	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)［リスト・ファイル］

リスト・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。

リスト・ファイルを出力する	ライブラリアンでリスト・ファイルを出力するかどうかを選択します。 list サブコマンドのオプション -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	リスト・ファイル（ライブラリ・ファイル内のモジュール情報）を出力します。
		いいえ	リスト・ファイル（ライブラリ・ファイル内のモジュール情報）を出力しません。
パブリック・シンボル情報を出力する	ライブラリアンでリスト・ファイルにパブリック・シンボル情報を出力するかどうかを選択します。 list サブコマンドのオプション -public に相当します。 なお，本プロパティは，［リスト・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。		
	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	リスト・ファイルにパブリック・シンボル情報を出力します。
		いいえ	リスト・ファイルにパブリック・シンボル情報を出力しません。
改ページ・コードを出力する	リスト・ファイルの末尾に改ページ・コードを出力するかどうかを選択します。 ライブラリアンのオプション -lf に相当します。 なお，本プロパティは，［リスト・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。		
	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 (-lf)	リスト・ファイルの末尾に改ページ・コードを出力します。
		いいえ	リスト・ファイルの末尾に改ページ・コードを出力しません。

1 行文字数	リスト・ファイルの各行の文字数を指定します。 ライブラリアンのオプション -lw に相当します。 なお，本プロパティは，［リスト・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。	
	デフォルト	132
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	72 ～ 260（10 進数）
1 ページ行数	リスト・ファイルの 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 ライブラリアンのオプション -ll に相当します。 なお，本プロパティは，［リスト・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。	
	デフォルト	0
	変更方法	テキスト・ボックスによる直接入力
	指定可能値	0，20 ～ 32767（10 進数）

（3）［その他］

ライブラリ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。

ライブラリ作成前に実行するコマンド	ライブラリ作成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。 %LibraryFile%　：ライブラリ作成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します。	
	デフォルト	ライブラリ作成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。

ライブラリ作成後に実行するコマンド	ライブラリ作成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。 %LibraryFile%　：ライブラリ作成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します。	
	デフォルト	ライブラリ作成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	その他に追加するライブラリアン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，ライブラリアン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

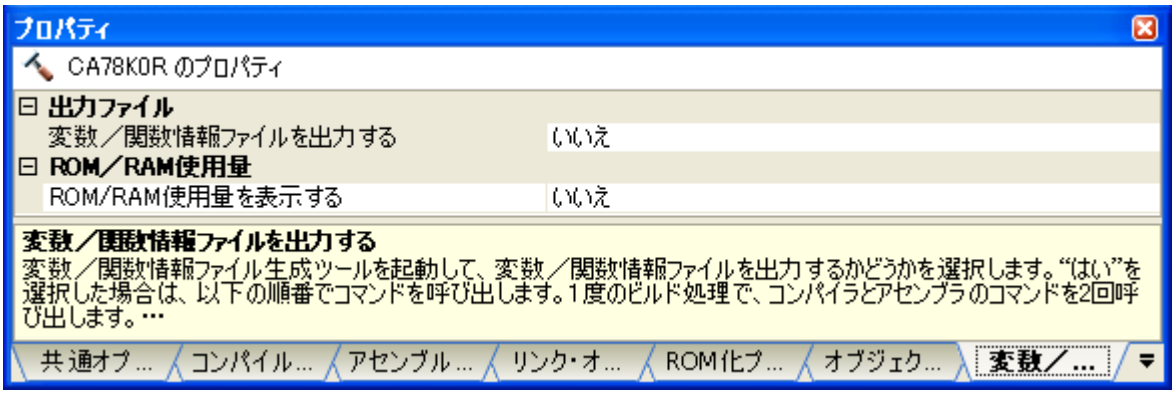
［変数／関数配置オプション］タブ

本タブでは，変数／関数情報ファイル生成ツールに対して，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）［出力ファイル］
（2）［マージン］
（3）［ROM/RAM 使用量］

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—11　プロパティ パネル：［変数／関数配置オプション］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。

変数／関数情報ファイルを出力する	変数／関数情報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	変数／関数情報ファイルを出力します。 なお，変数／関数情報ファイルは，ラピッド・ビルドの対象外となります。
		いいえ	変数／関数情報ファイルを出力しません。

変数／関数情報ファイル出力フォルダ	変数／関数情報ファイルを格納するフォルダを指定します。 変数／関数情報ファイル生成ツールのオプション -vo 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，［変数／関数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
変数／関数情報ファイル名	変数／関数情報ファイル名を指定します。 変数／関数情報ファイル生成ツールのオプション -vo に相当します。 拡張子は“.vfi”を指定してください。拡張子を省略した場合は，“.vfi”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。 なお，本プロパティは，［変数／関数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	%ProjectName%.vfi
	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)［マージン］

マージ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，［出力ファイル］カテゴリの［変数／関数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。

saddr 領域のマージン	saddr 領域のマージン・サイズを指定します。 変数／関数情報ファイル生成ツールで変数を saddr 領域に割り当てた後，コンパイル，リンクを行ったとき，処理の順番やアライメントの関係で配置エラーになる場合があります。このとき，saddr 領域にマージンを設定することにより，そのエラーを回避することができます。 変数／関数情報ファイル生成ツールのオプション -vs に相当します。	
	デフォルト	0
	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 ～ 192（10 進数）

（3）［ROM/RAM 使用量］

ROM/RAM 使用量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/RAM 使用量を表示する	ROM/RAM 使用量を出力 パネルに表示するかどうかを選択します。 変数／関数情報ファイル生成ツールのオプション -vx 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	ROM/RAM 使用量を表示します。
		いいえ	ROM/RAM 使用量を表示しません。

［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，変数／関数情報ファイル，オブジェクト・ファイル，ライブラリ・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—12　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）

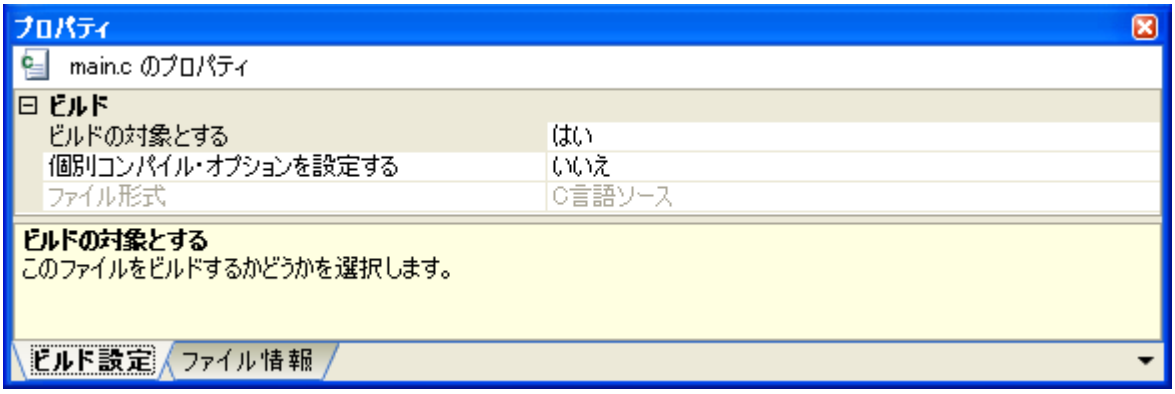


図 A—13　プロパティ パネル：［ビルド設定］タブ（アセンブラ・ソース・ファイルを選択した場合）

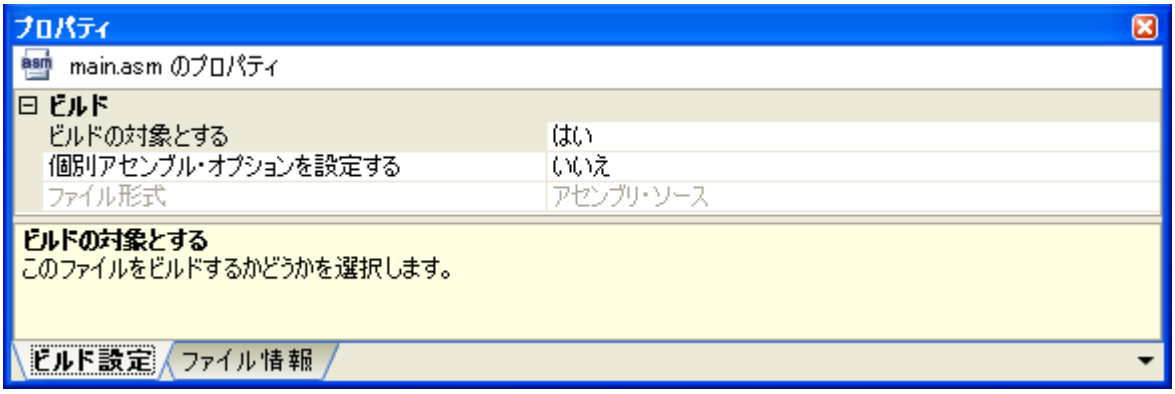


図 A—14　プロパティ パネル：［ビルド設定］タブ（リンク・ディレクティブ・ファイルを選択した場合）

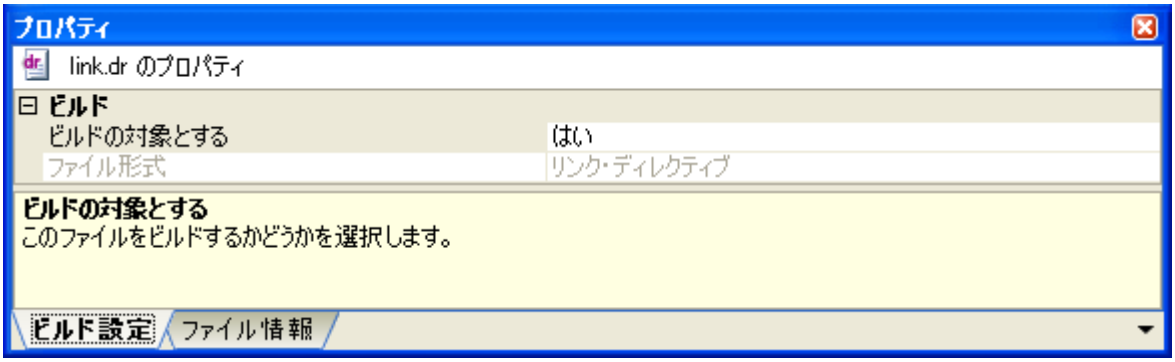


図 A—15　プロパティ パネル：［ビルド設定］タブ（変数／関数情報ファイルを選択した場合）

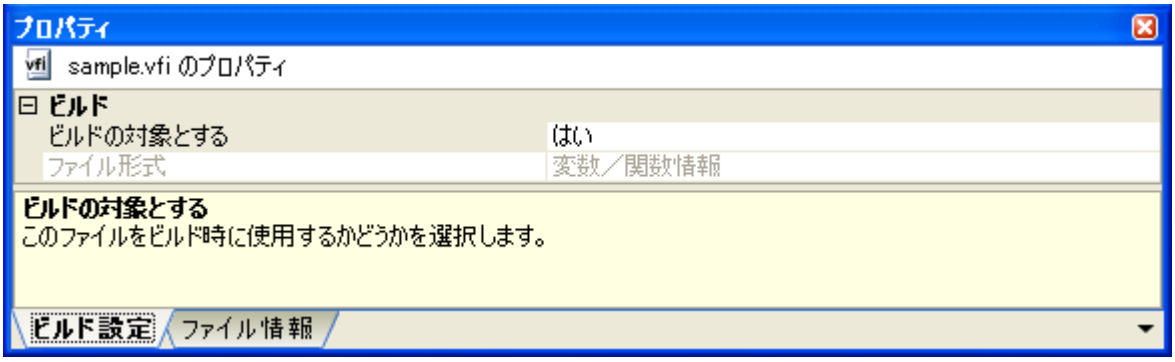


図 A—16　プロパティ パネル：［ビルド設定］タブ（オブジェクト・ファイルを選択した場合）

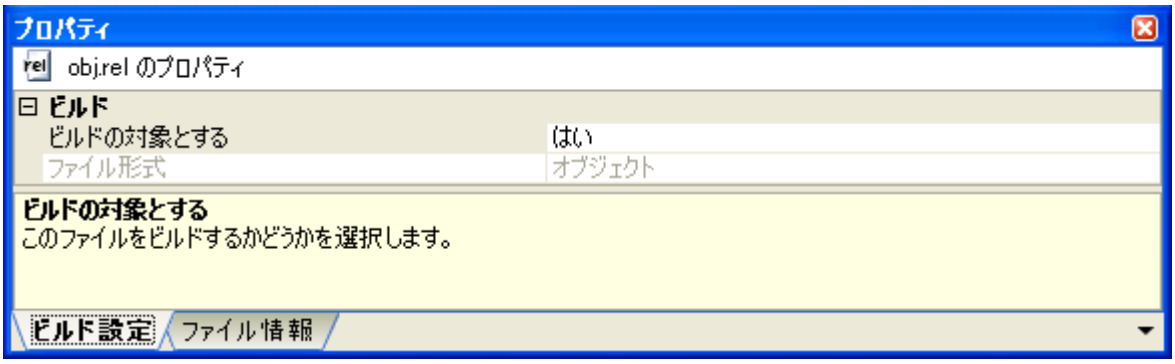
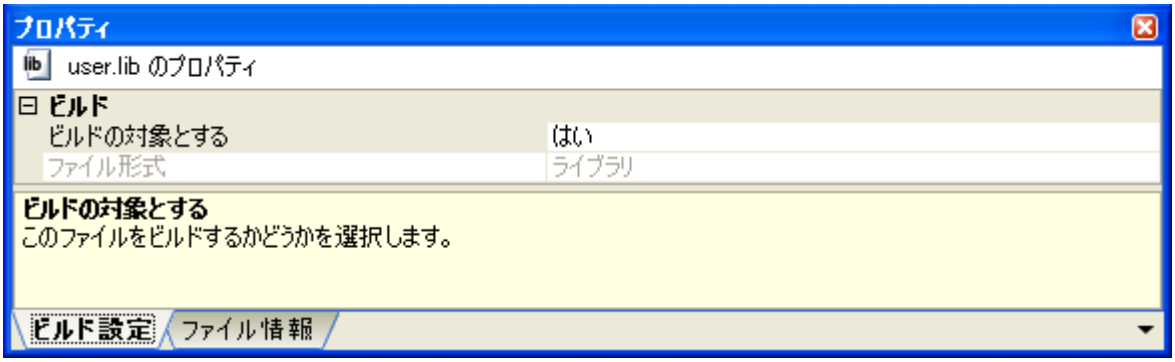


図 A—17　プロパティ パネル：［ビルド設定］タブ（ライブラリ・ファイルを選択した場合）



[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。

個別アセンブル・オプションを設定する	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。		
	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。
		いいえ	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。
ファイル形式	選択しているファイルの形式を表示します。		
	デフォルト	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） 変数／関数情報（変数／関数情報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（ライブラリ・ファイルを選択している場合）	
	変更方法	変更不可	

［個別コンパイル・オプション］タブ

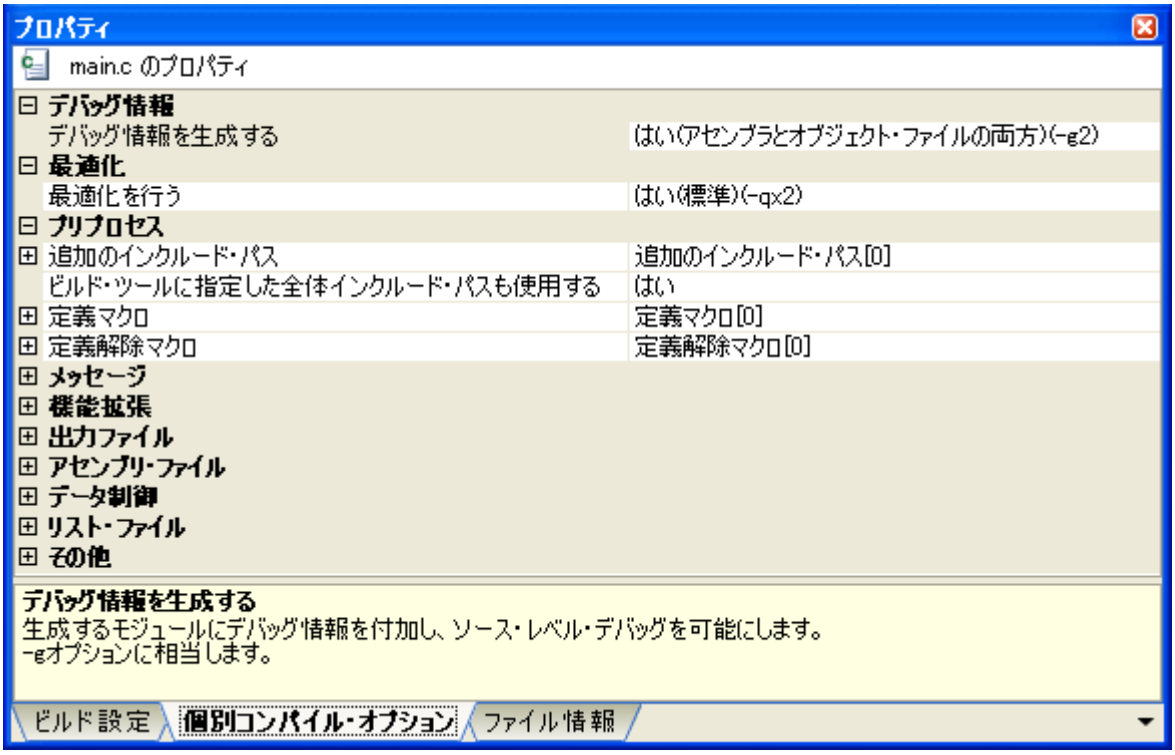
本タブでは，１つのＣソース・ファイルに対して，次に示すカテゴリごとに詳細情報の表示，および設定の変更を行います。

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。

（1）［デバッグ情報］
（2）［最適化］
（3）［最適化（詳細）］
（4）［プリプロセス］
（5）［メッセージ］
（6）［機能拡張］
（7）［出力ファイル］
（8）［アセンブリ・ファイル］
（9）［データ制御］
（10）［リスト・ファイル］
（11）［その他］

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—18　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	生成するモジュー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。 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 (オブジェクト・ファイルのみ)(-g1)	生成するオブジェクト・モジュール・ファイルにデバッグ情報を付加します。
		はい (アセンブラとオブジェクト・ファイルの両方)(-g2)	生成するオブジェクト・モジュール・ファイルとアセンブラ・ソース・ファイルにデバッグ情報を付加します。
		いいえ (-ng)	生成するオブジェクト・モジュール・ファイルにデバッグ情報を付加しません。

(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。

最適化を行う	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 -qx 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 (実行速度優先)(-qx1)	実行速度を優先して最適化を行います。
		はい (標準)(-qx2)	実行速度，モジュール・サイズの両方を優先して最適化を行います。
		はい (モジュール・サイズ優先)(-qx3)	モジュール・サイズを優先して最適化を行います。
		はい (詳細設定)	[最適化（詳細）] カテゴリを表示します。そこで選択したオプションを優先して最適化を行います。なお，[最適化（詳細）] カテゴリですべて [いいえ] を選択した場合は，最適化を行いません。
		いいえ (-nq)	最適化を指定しません。

(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。

なお，本カテゴリは，[最適化（詳細）] カテゴリの [最適化を行う] プロパティで [はい（詳細設定）] を選択した場合のみ表示されます。

演算順序の入れ替えを行う	演算式の実行順序入れ替えなどを行うことでレジスタの有効活用を図り，効率の良い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 -qw 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 (式の演算順序を入れ替える)(-qw)	式の演算順序を入れ替えます。
		いいえ	式の演算順序の入れ替えを指定しません。
自動変数をレジスタ，または saddr 領域に割り当てる	自動変数をレジスタ，または saddr 領域に自動的に割り当てるかどうかを選択します。 コンパイラのオプション -qv 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 (-qv)	自動変数をレジスタ，または saddr 領域に自動的に割り当てます。
		いいえ	自動変数のレジスタ，または saddr 領域への自動的に割り当てを指定しません。
レジスタ変数を saddr 領域にも割り当てる	レジスタ変数をレジスタに加えて saddr 領域にも割り当てるかどうかを選択します。 コンパイラのオプション -qr 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 (-qr)	レジスタ変数をレジスタに加えて saddr 領域にも割り当てます。
		いいえ	レジスタ変数の saddr 領域への割り当てを指定しません。
char 型演算を符号拡張しない	char に関する演算を汎整数拡張せずに行うかどうかを選択します。 コンパイラのオプション -qc 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 (-qc)	char に関する演算を汎整数拡張せずに行います。注
		いいえ	char に関する演算を汎整数拡張して行います。
char 型を unsigned char とみなす	修飾子なしの char を unsigned char とみなすかどうかを選択します。 コンパイラのオプション -qu 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 (-qu)	修飾子なしの char を unsigned char とみなします。
		いいえ	修飾子なしの char を unsigned char とみなすことを指定しません。

分岐命令を最適化する	分岐命令の最適化を行うかどうかを選択します。 コンパイラのオプション -qj 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 (-qj)	分岐命令の最適化を行います。
		いいえ	分岐命令の最適化を指定しません。
定型コードをライブラリに置き換える (サイズ優先最適化)	定型コードをライブラリに置き換えるかどうかを選択します。 コンパイラのオプション -ql 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 (ライブラリに置き換えない)(-ql1)	定型コードをライブラリに置き換えません。モジュール・サイズを優先して最適化を行います。
		はい (関数の前後処理のみ)(-ql2)	関数の前後処理のみライブラリに置き換えます。
		はい (関数の前後処理，低レベル・ライブラリの使用，共通コードのサブルーチン化)(-ql3)	関数の前後処理のみライブラリに置き換えます。また，低レベル・ライブラリの使用，共通コードのサブルーチン化を行います。
		いいえ	定型コードのライブラリへの置き換えを指定しません。 実行速度を優先して最適化を行います。
相対分岐の switch 分岐テーブルを生成する	相対分岐の switch 分岐テーブルを生成するかどうかを選択します。 コンパイラのオプション -qt 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 (-qt)	相対分岐の switch 分岐テーブルを生成します。
		いいえ	相対分岐の switch 分岐テーブルの生成を指定しません。
デバッグに適した最適化を行う	デバッグに適した最適化を行うかどうかを選択します。 コンパイラのオプション -qg 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 (-qg)	デバッグに適した最適化を行います。
		いいえ	デバッグに適した最適化を指定しません。

注　-qc オプションを設定した場合の演算結果は，次のようになります。

演算対象	演算結果
unsigned char 型の変数と unsigned char 型の変数	unsigned char 型
unsigned char 型の変数と signed char 型の変数	unsigned char 型
signed char 型の変数と signed char 型の変数	signed char 型

演算対象	演算結果
-128 ～ 255 の定数と unsigned char 型の変数	unsigned char 型
-128 ～ 127 の定数と signed char 型の変数	signed char 型
0 ～ 255 で接尾語 U 付きの定数と signed char 型の変数	unsigned char 型

(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。
本プロパティを省略した場合，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します。</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="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="2">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>
</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。 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します。	
	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 30 個まで指定可能です。
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します。	
	デフォルト	定義解除マクロ [*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 30 個まで指定可能です。

(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 (-w0)	ワーニング・メッセージを出力しません。
		通常出力	通常のワーニング・メッセージを出力します。
		詳細出力 (-w2)	詳細なワーニング・メッセージを出力します。

(6) [機能拡張]

機能拡張に関する詳細情報の表示，および設定の変更を行います。

C++ 形式のコメントを許可する	C++ 形式のコメント（“//”）使用を許可するかどうかを選択します。 コンパイラのオプション -zp 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 (-zp)	C++ 形式のコメント使用を許可します。
		いいえ	C++ 形式のコメント使用を許可しません。
コメントのネストを許可する	コメント（“/* ... */”）のネスト使用を許可するかどうかを選択します。 コンパイラのオプション -zc 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 (-zc)	コメントのネスト使用を許可します。
		いいえ	コメントのネスト使用を許可しません。
ソースの漢字コード	ソースの漢字コードを選択します。 コンパイラのオプション -zs，-ze，-zn 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(-zs)	ソースの漢字コードを Shift_JIS と解釈します。
		EUC-JP(-ze)	ソースの漢字コードを EUC-JP と解釈します。
		なし (-zn)	ソースに漢字コードがないと解釈します。
ANSI 規格に準拠する	ANSI 規定外の機能を無効とし，ANSI 規定の一部の機能を有効とするかどうかを選択します。 コンパイラのオプション -za 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 (-za)	ANSI 規定外の機能を無効とし，ANSI 規定の一部の機能を有効とします。
		いいえ	ANSI 規定外の機能を有効とします。
関数の int 拡張を無効にする	関数の char/unsigned char 型引数および返り値を int 拡張しないかどうかを選択します。 コンパイラのオプション -zb 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 (-zb)	関数の char/unsigned char 型引数および返り値を int 拡張しません。
		いいえ	関数の char/unsigned char 型引数および返り値を int 拡張します。

RAM 配置用オブジェクトを出力する	コード，および ROM データを RAM 領域に配置するかどうかを選択します。 また，RAM 配置用ランタイム・ライブラリを配置する領域も指定します。 コンパイラのオプション -zx 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 (ランタイム・ライブラリを ROM 領域に配置)(-zx1)	コード，および ROM データを RAM 領域に配置し，ランタイム・ライブラリを ROM 領域に配置します。
		はい (ランタイム・ライブラリを RAM 領域に配置)(-zx2)	コード，および ROM データを RAM 領域に配置し，ランタイム・ライブラリを RAM 領域に配置します。 ［定型コードをライブラリに置き換える (サイズ優先最適化)］プロパティにおいて［はい (ライブラリに置き換えない)(-ql1)］を選択したものとします。
		いいえ	コード，および ROM データを RAM 領域に配置しません。

(7)［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。 空欄の場合は，C ソース・ファイルの拡張子 .c を .rel 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	
デバイス共通オブジェクトを出力する	デバイス共通オブジェクトを出力するかどうかを選択します。 コンパイラのオプション -common 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 (-common)	デバイス共通オブジェクトを出力します。
		いいえ	デバイス共通オブジェクトの出力を指定しません。

(8)［アセンブリ・ファイル］

アセンブリ・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	アセンブリ・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 -a，-sa，-li 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 (C ソース情報なし)(-a)	アセンブリ・ファイル（C ソース情報なし）を出力します。
		はい (C ソース情報あり (インクルード・ファイルを非展開))(-sa)	アセンブリ・ファイル（C ソース情報あり（インクルード・ファイルを非展開））を出力します。
		はい (C ソース情報あり (インクルード・ファイルを展開))(-sa,-li)	アセンブリ・ファイル（C ソース情報あり（インクルード・ファイルを展開））を出力します。
		いいえ	アセンブリ・ファイルを出力しません。

(9)［データ制御］

データ制御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・フィールドを MSB から割り付ける	ビット・フィールド構造体のメンバを MSB から割り付けるかどうかを選択します。 コンパイラのオプション -rb 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 (-rb)	ビット・フィールド構造体のメンバを MSB から割り付けます。
		いいえ	ビット・フィールド構造体のメンバを LSB から割り付けます。
構造体メンバをパッキングする	構造体内の（2 バイト以上の）メンバを偶数番地に配置するためのアライン・データを挿入しないようにするかどうかを選択します。 コンパイラのオプション -rc 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 (-rc)	構造体内の（2 バイト以上の）メンバを偶数番地に配置するためのアライン・データを挿入しません。
		いいえ	構造体内の（2 バイト以上の）メンバを偶数番地に配置するためのアライン・データを挿入します。
間接参照を 1 バイト単位で行う	間接参照を 1 バイト単位で行うかどうかを選択します。 コンパイラのオプション -ra 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 (-ra)	間接参照を 1 バイト単位で行います。
		いいえ	間接参照を 1 バイト単位で行いません。

static 変数を saddr 領域に割り当てる	saddr 領域に配置させる static 変数の種類を選択します。 コンパイラのオプション -rs 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 (char サイズ)(-rs1)	char，unsigned char 型の自動変数を saddr 領域に配置します。
		はい (char,short,int サイズ)(-rs2)	char，unsigned char，short，unsigned short，int，unsigned int，enum，near ポインタ型の自動変数を saddr 領域に配置します。
		はい (char,short,int,long サイズ)(-rs4)	char，unsigned char，short，unsigned short，int，unsigned int，enum，long，unsigned long，ポインタ型の自動変数を saddr 領域に配置します。
		はい (構造体 , 共用体 , 配列)(-rsm)	構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		はい (char サイズ , 構造体 , 共用体 , 配列)(-rs1m)	char，unsigned char，構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		はい (char,short,int サイズ , 構造体 , 共用体 , 配列)(-rs2m)	char，unsigned char，short，unsigned short，int，unsigned int，enum，near ポインタ，構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		はい (char,short,int,long サイズ , 構造体 , 共用体 , 配列)(-rs)	char，unsigned char，short，unsigned short，int，unsigned int，enum，long，unsigned long，ポインタ，構造体，共用体，配列型の自動変数を saddr 領域に配置します。
		いいえ	static 変数を saddr 領域に配置しません。
ROM データの配置先を指定する	ROM データの配置先を指定するかどうかを選択します。 コンパイラのオプション -rf，-rn 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 (far 領域)(-rf)	ROM データを far 領域に配置します。
		はい (near 領域)(-rn)	ROM データを near 領域に配置します。
		いいえ	ROM データの配置先を指定しません。

(10)［リスト・ファイル］

リスト・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。

プリプロセス・リスト・ファイルを出力する	プリプロセス・リスト・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 -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 (-p)	プリプロセス・リスト・ファイルを出力します。
		いいえ	プリプロセス・リスト・ファイルを出力しません。
コメントを出力しない	プリプロセス・リスト・ファイル中にコメントを出力しないかどうかを選択します。 コンパイラのオプション -kc に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kc)	プリプロセス・リスト・ファイル中にコメントを出力しません。
		いいえ	プリプロセス・リスト・ファイル中にコメントを出力します。
#define を展開する	プリプロセス・リスト・ファイル中に #define 文を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 -kd に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kd)	プリプロセス・リスト・ファイル中に #define 文を展開します。
		いいえ	プリプロセス・リスト・ファイル中に #define 文を展開しません。
#if，#ifdef，#ifndef を展開する	プリプロセス・リスト・ファイル中に #if，#ifdef，#ifndef 文を展開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 -kf に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kf)	プリプロセス・リスト・ファイル中に #if，#ifdef，#ifndef 文を展開して出力します。
		いいえ	プリプロセス・リスト・ファイル中に #if，#ifdef，#ifndef 文を展開しません。

#include を展開する	プリプロセス・リスト・ファイル中に #include 文を展開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 -ki に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-ki)	プリプロセス・リスト・ファイル中に #include 文を展開して出力します。
		いいえ	プリプロセス・リスト・ファイル中に #include 文を展開しません。
#line を展開する	プリプロセス・リスト・ファイル中に #line 文を展開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 -kl に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kl)	プリプロセス・リスト・ファイル中に #line 文を展開して出力します。
		いいえ	プリプロセス・リスト・ファイル中に #line 文を展開しません。
行番号を出力する	プリプロセス・リスト・ファイル中に行番号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 -kn に相当します。 なお，本プロパティは，［プリプロセス・リスト・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。		
	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 (-kn)	プリプロセス・リスト・ファイル中に行番号を出力します。
		いいえ	プリプロセス・リスト・ファイル中に行番号を出力しません。
エラー・リスト・ファイルを出力する	エラー・リスト・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 -e，-se 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 (C ソースなし)(-e)	エラー・リスト・ファイル（C ソースなし）を出力します。
		はい (C ソースあり)(-se)	エラー・リスト・ファイル（C ソースあり）を出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

クロスリファレンス・リスト・ファイルを出力する	クロスリファレンス・リスト・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 -x 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 (-x)	クロスリファレンス・リスト・ファイルを出力します。
		いいえ	クロスリファレンス・リスト・ファイルを出力しません。
改ページ・コードを出力する	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）の末尾に改ページ・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 -lf に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］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 (-lf)	リスト・ファイル中に改ページ・コードを出力します。
		いいえ	リスト・ファイル中に改ページ・コードを出力しません。
1 行文字数	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）の各行の文字数を指定します。 コンパイラのオプション -lw に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］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	
	指定可能値	72 ～ 132（10 進数）	
1 ページ行数	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）の 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 コンパイラのオプション -ll に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］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	
	指定可能値	0，20 ～ 65535（10 進数）	

タブ幅	リスト・ファイル（プリプロセス・リスト・ファイル，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル）のタブ幅を指定します。 コンパイラのオプション -lt に相当します。 なお，本プロパティは，［エラー・リスト・ファイルを出力する］プロパティで［はい］，または［プリプロセス・リスト・ファイルを出力する］プロパティで［はい (-p)］，または［クロスリファレンス・リスト・ファイルを出力する］プロパティで［はい (-x)］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
	指定可能値	0 ～ 8（10 進数）

（11）［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。 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します。	
	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。 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します。	
	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。

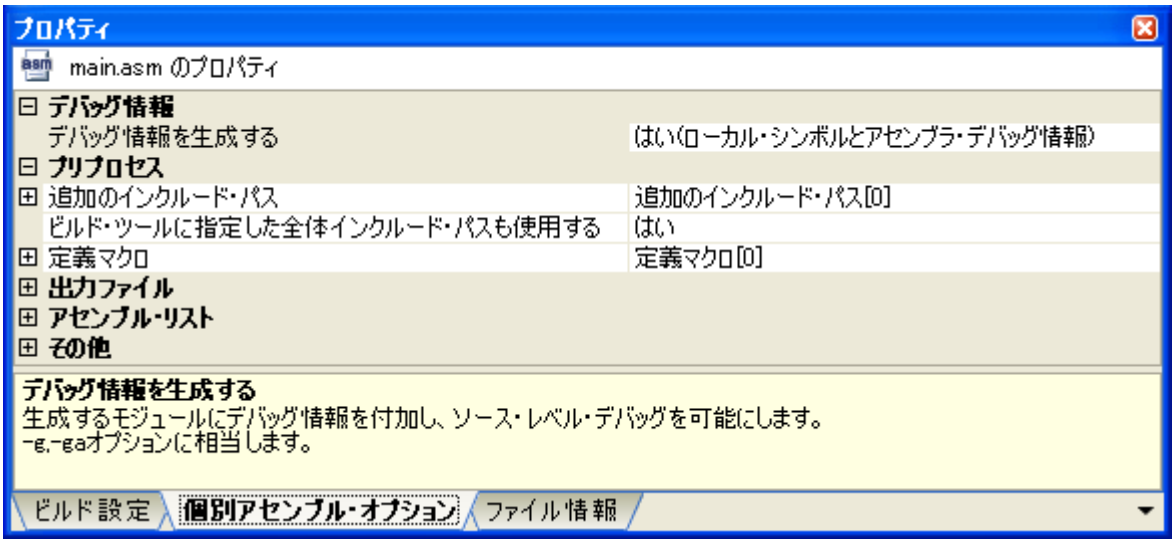
なお，本タブは，［アセンブル・オプション］タブの設定内容を継承します。［アセンブル・オプション］タブと異なる値を設定した場合は，プロパティが太字表示となります。

（1）［デバッグ情報］
（2）［プリプロセス］
（3）［出力ファイル］
（4）［アセンブル・リスト］
（5）［その他］

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［アセンブリ・ファイル］カテゴリの［アセンブル・ファイルを出力する］プロパティで［はい］を選択した場合にも表示されます。

図 A—19　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	生成するモジュー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，-ga 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 (アセンブラ・デバッグ情報)(-ng,-ga)	生成するオブジェクト・モジュール・ファイルにデバッグ情報 (アセンブラ・デバッグ情報) を付加します。
		はい (ローカル・シンボルとアセンブラ・デバッグ情報)	生成するオブジェクト・モジュール・ファイルにデバッグ情報 (ローカル・シンボルとアセンブラ・デバッグ情報) を付加します。
		いいえ (-ng,-nga)	生成するオブジェクト・モジュール・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。 本プロパティを省略した場合，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します。	
	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。 - ［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 - ［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 - ［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		
	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します。		
	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 30 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。 空欄の場合は，アセンブラ・ソース・ファイルの拡張子 .asm を .rel 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。		
	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	
デバイス共通オブジェクトを出力する	各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。 アセンブラのオプション -common 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 (-common)	各デバイス共通のオブジェクトを出力します。
		いいえ	RL78，および 78K0R に対応したオブジェクトを出力します。

エラー・リスト・ファイルを出力する	エラー・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 -e 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 (-e)	エラー・リスト・ファイルを出力します。
		いいえ	エラー・リスト・ファイルを出力しません。

(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 -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 (-p)	アセンブル・リスト・ファイルを出力します。
		いいえ (-np)	アセンブル・リスト・ファイルを出力しません。
リスト・コンバータを実行する	実行モジュールを生成した後にリスト・コンバータを実行するかどうかを選択します。 ただし，ライブラリ生成時には実行しません。 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［いいえ (-np)］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	実行モジュールを生成した後にリスト・コンバータを実行します。
		いいえ	実行モジュールを生成した後にリスト・コンバータを実行しません。
リスト・コンバータのエラー・リスト・ファイルを出力する	リスト・コンバータを実行時に，エラー・リスト・ファイルを出力するかどうかを選択します。 リスト・コンバータのオプション -e に相当します。 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［いいえ (-np)］を選択した場合，［リスト・コンバータを実行する］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 (-e)	リスト・コンバータを実行時に，エラー・リスト・ファイルを出力します。
		いいえ	リスト・コンバータを実行時に，エラー・リスト・ファイルを出力しません。

アセンブル・リストを出力する	アセンブル・リスト・ファイル中にアセンブル・リスト情報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 -ka に相当します。 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［いいえ (-np)］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	アセンブル・リスト・ファイル中にアセンブル・リスト情報を出力します。
		いいえ (-nka)	アセンブル・リスト・ファイル中にアセンブル・リスト情報を出力しません。
シンボル・リストを出力する	アセンブル・リスト・ファイル中にシンボル・リスト情報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 -ks に相当します。 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［いいえ (-np)］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 (-ks)	アセンブル・リスト・ファイル中にシンボル・リスト情報を出力します。
		いいえ	アセンブル・リスト・ファイル中にシンボル・リスト情報を出力しません。
クロスリファレンス・リストを出力する	アセンブル・リスト・ファイル中にクロスリファレンス・リスト情報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 -kx に相当します。 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［いいえ (-np)］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 (-kx)	アセンブル・リスト・ファイル中にクロスリファレンス・リスト情報を出力します。
		いいえ	アセンブル・リスト・ファイル中にクロスリファレンス・リスト情報を出力しません。
改ページ・コードを出力する	リスト・ファイル中に改ページ・コード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 -lf に相当します。 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［いいえ (-np)］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 (-lf)	リスト・ファイル中に改ページ・コードを出力します。
		いいえ	リスト・ファイル中に改ページ・コードを出力しません。

1 行文字数	リスト・ファイルの各行の文字数を指定します。 アセンブラのオプション -lw に相当します。 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［いいえ (-np)］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	72 ～ 2046（10 進数）
1 ページ行数	リスト・ファイルの 1 ページの行数を指定します。 0 を指定した場合は改頁しません。 アセンブラのオプション -ll に相当します。 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［いいえ (-np)］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	0，20 ～ 32767（10 進数）
タブ幅	リスト・ファイルのタブ幅を指定します。 アセンブラのオプション -lt に相当します。 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［いいえ (-np)］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 ～ 8（10 進数）
ヘッダ文字列	アセンブル・リスト・ファイルのヘッダを指定します。 全角文字，および半角スペースを含む文字列も指定可能です。 アセンブラのオプション -lh に相当します。 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［いいえ (-np)］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	半角 60 文字（全角 30 文字）までの文字列

(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。

ソースの漢字コード	ソースの漢字コードを選択します。 アセンブラのオプション -zs，-ze，-zn 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(-zs)	ソースの漢字コードを Shift_JIS と解釈します。
		EUC-JP(-ze)	ソースの漢字コードを EUC-JP と解釈します。
		なし (-zn)	ソースに漢字コードがないと解釈します。

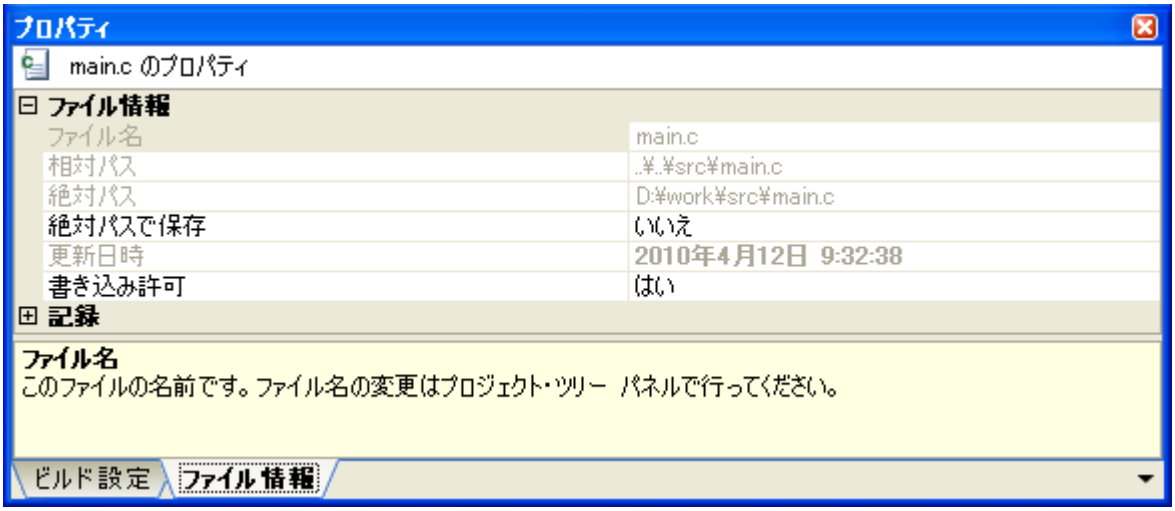
78K0 互換マクロを使用する	78K0 マイコン用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にするかどうかを選択します。 アセンブラのオプション -compati に相当します。		
	デフォルト	*全体オプションの設定値*	
	変更方法	ドロップダウン・リストによる選択	
	指定可能値	はい (-compati)	78K0 マイコン用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。
		いいえ	78K0 マイコン用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にしません。
アセンブル前に実行するコマンド	アセンブル処理前に実行するコマンドを指定します。 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します。		
	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。	
アセンブル後に実行するコマンド	アセンブル処理後に実行するコマンドを指定します。 埋め込みマクロとして次のマクロ名があります。 %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します。		
	デフォルト	アセンブ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	*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	

［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。

（1）［ファイル情報］
（2）［記録］

図 A—20　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。

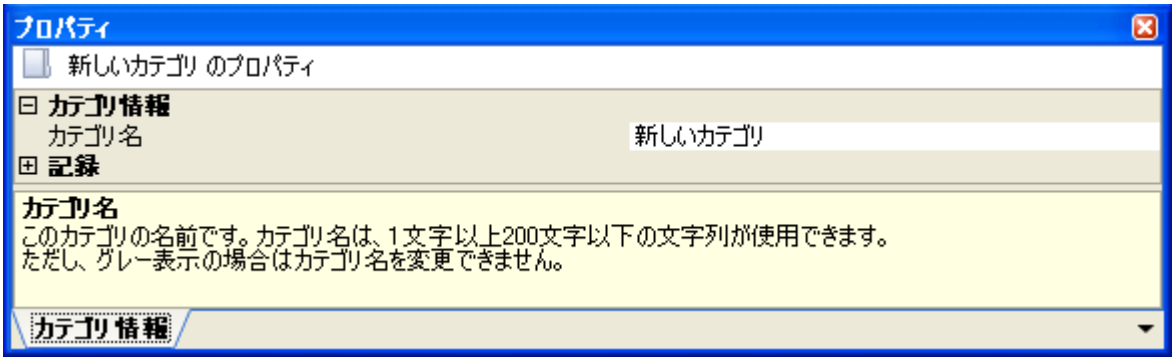
メモ	ファイルにメモを追加します。 1 行に 1 項目ずつ指定します。 追加したメモはサブプロパティとして表示します。	
	デフォルト	メモ [*項目数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

［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。

（1）［カテゴリ情報］
（2）［記録］

図 A—21　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。

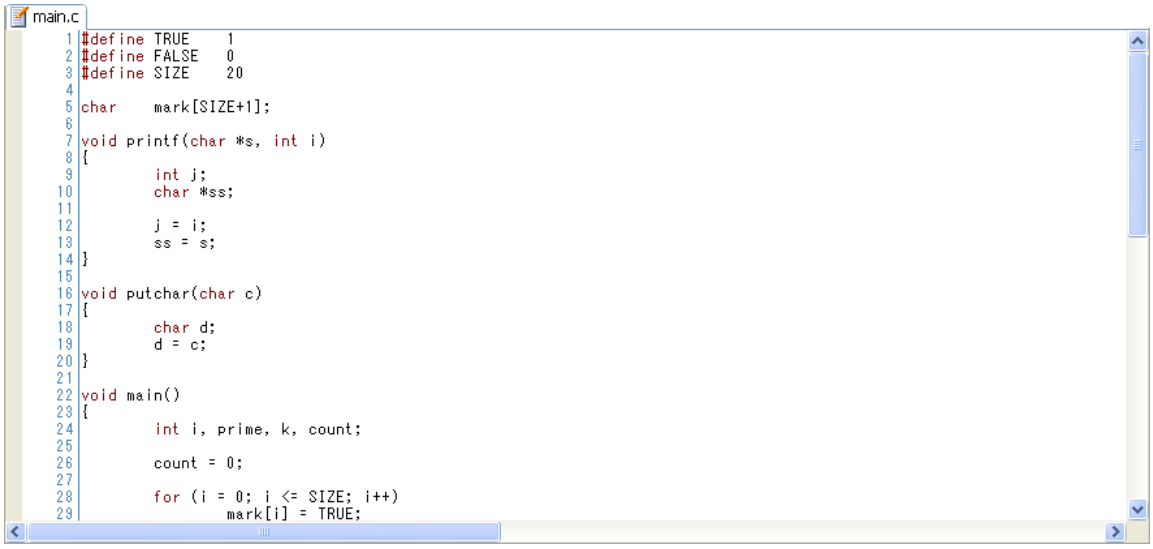
メモ	ファイルのカテゴリにメモを追加します。 1 行に 1 項目ずつ指定します。 追加したメモはサブプロパティとして表示します。	
	デフォルト	メモ [*項目数*]
	変更方法	[...] ボタンをクリックし，テキスト編集 ダイアログによる編集 サブプロパティはテキスト・ボックスによる直接入力も可能
	指定可能値	256 文字までの文字列 256 個まで指定可能です。

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+ RL78,78K0R コーディング編」を参照してください。

図 A—22　エディタ パネル

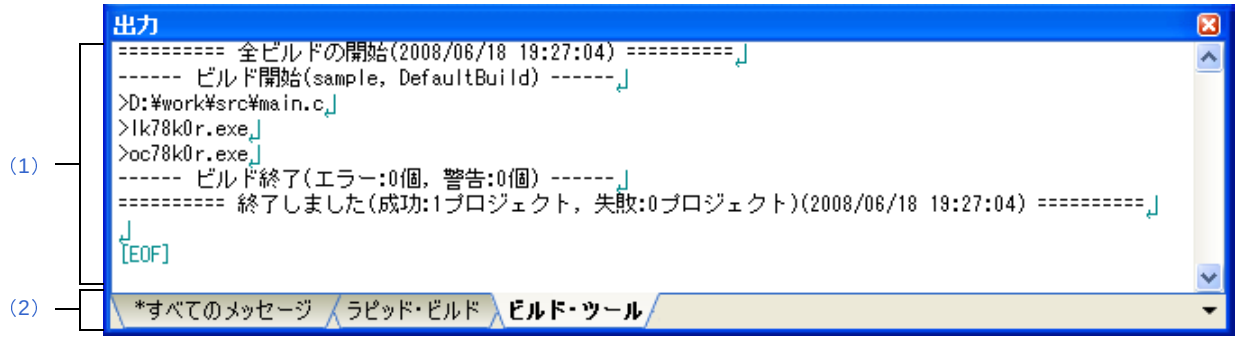


出力 パネル

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—23　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。

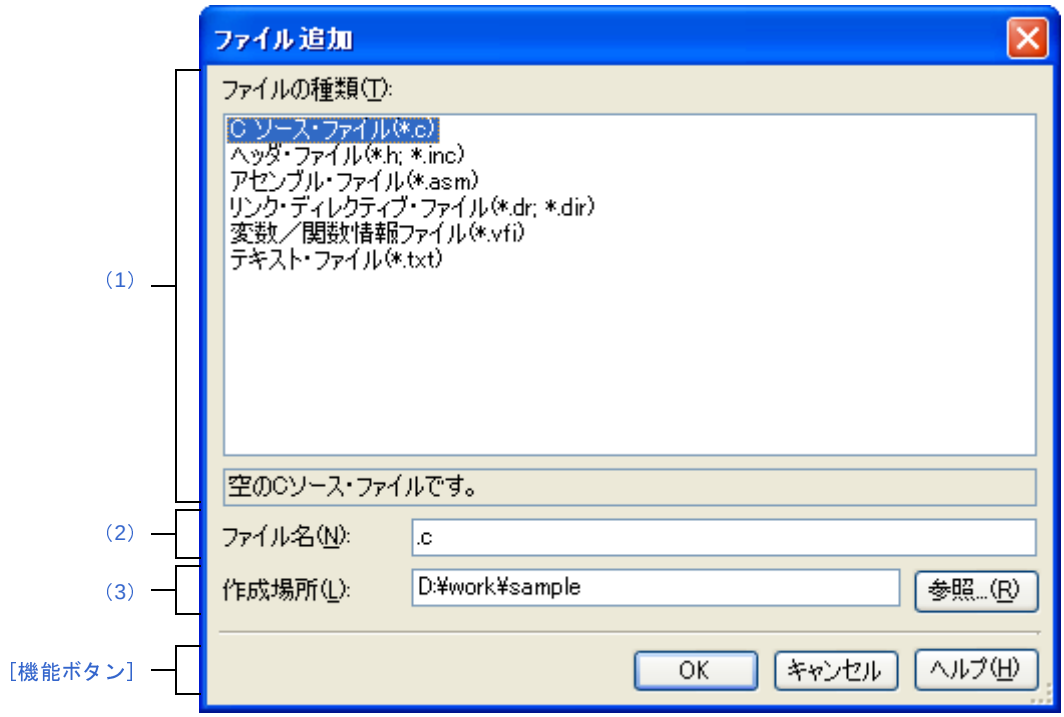
[コンテキスト・メニュー]

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—24　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（*.asm）
- リンク・ディレクティブ・ファイル（*.dr; .*dir）
- 変数／関数情報ファイル（*.vfi）
- 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]

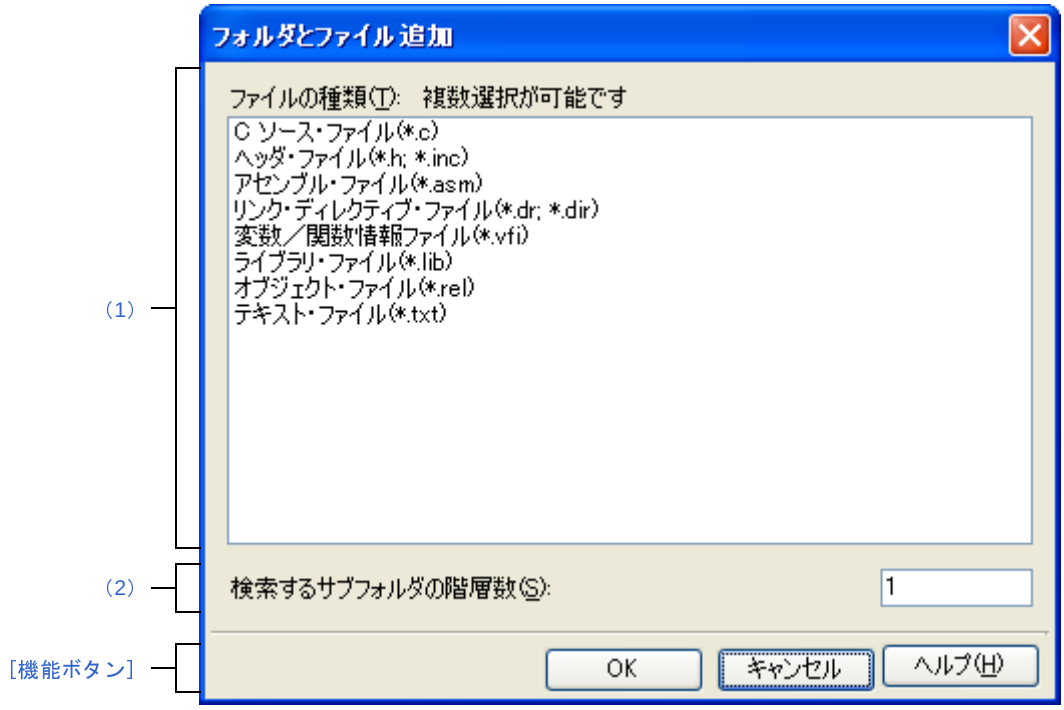
ボタン	機能
OK	入力したファイル名でファイルを作成して，プロジェクトに追加し，エディタ パネルでオープンします。そののち，本ダイアログをクローズします。
キャンセル	ファイルの生成を行わずに，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

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—25　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（*.asm）
- リンク・ディレクティブ・ファイル（*.dr; .*dir）
- 変数／関数情報ファイル（*.vfi）
- ライブラリ・ファイル（*.lib）
- オブジェクト・ファイル（*.rel）
- 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。

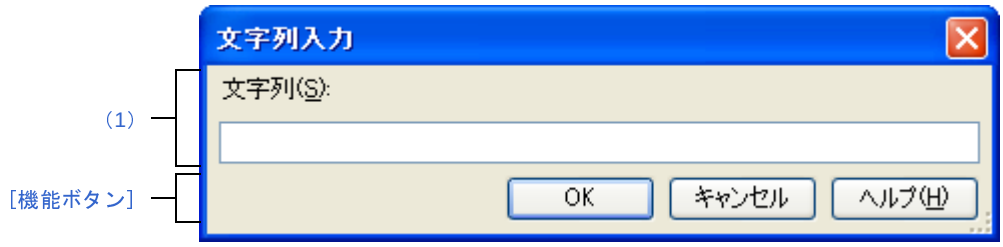
[機能ボタン]

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—26　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［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［スタック］カテゴリの［領域名］，［その他］カテゴリの［その他の追加オプション］
  - ［オブジェクト・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］
  - ［ライブラリ生成オプション］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］
- 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。

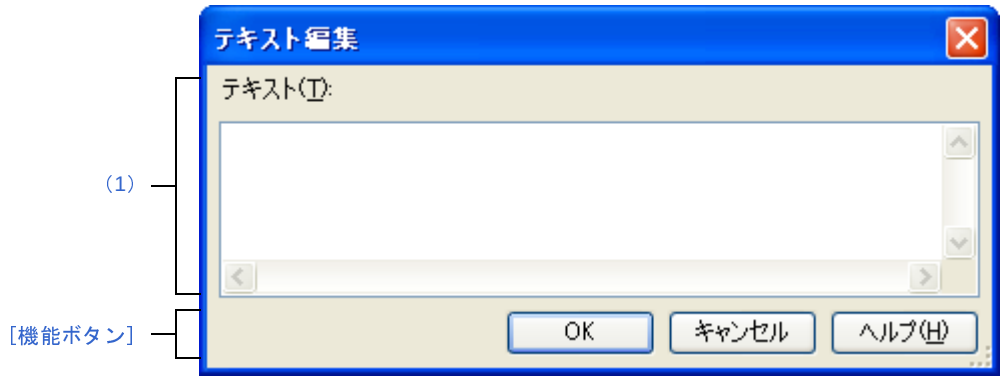
[機能ボタン]

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—27　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]
  - [オブジェクト・コンバート・オプション] タブの [その他] カテゴリの [オブジェクト・コンバート前に実行するコマンド]，[オブジェクト・コンバート後に実行するコマンド]
  - [ライブラリ生成オプション] タブの [その他] カテゴリの [ライブラリ作成前に実行するコマンド]，[ライブラリ作成後に実行するコマンド]
  - [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。

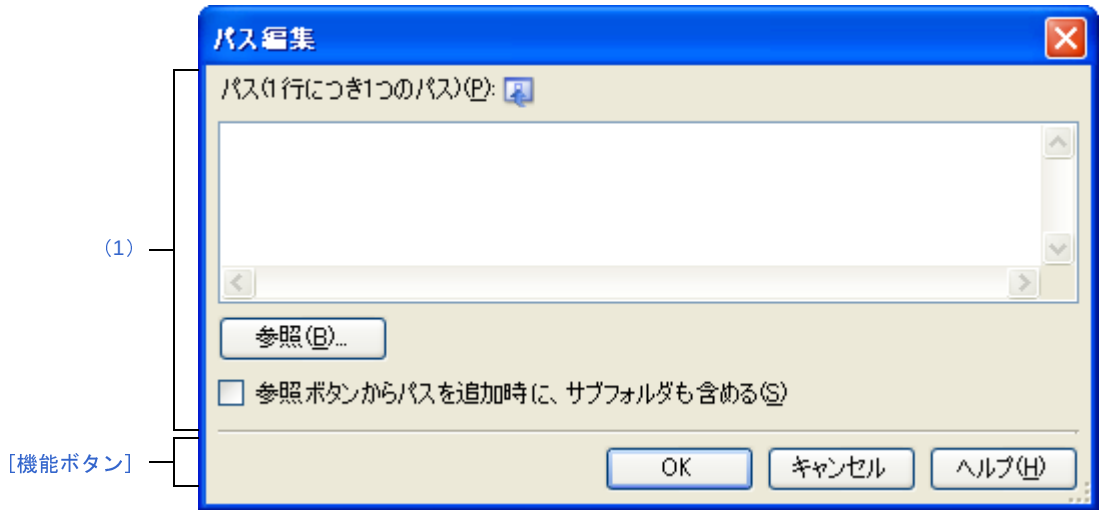
［機能ボタン］

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

パスの編集，追加を行います。

図 A—28　パス編集 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［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］
  - ［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，パスの編集，追加を行います。

パスは複数行指定可能です。1 行につき 1 つのパス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

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。
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 ...	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。

(c) [参照ボタンからパスを追加時に、サブフォルダも含める]

このチェック・ボックス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，サブフォルダも含めて [パス (1 行につき 1 つのパス)] にパスが追加されます（5 階層まで）。

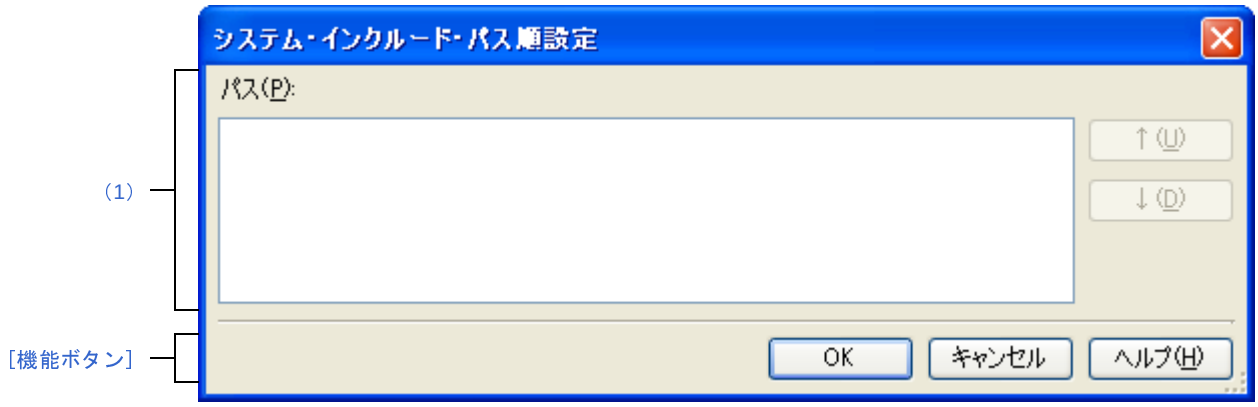
[機能ボタン]

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—29　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]

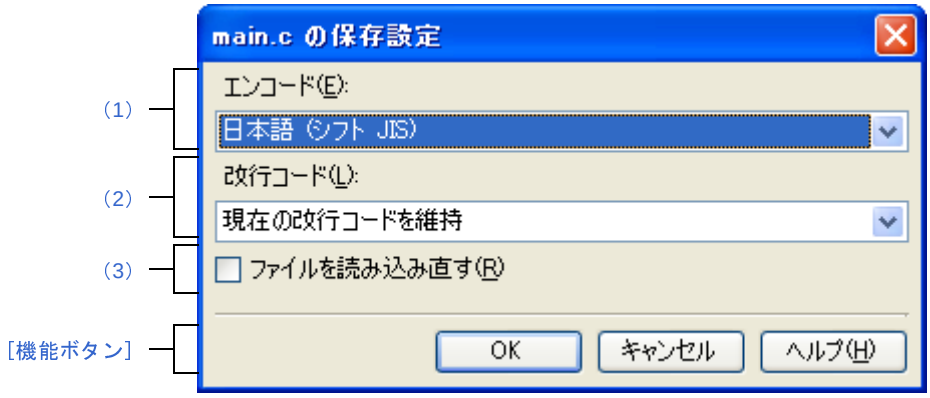
ボタン	機能
OK	コンパイラへのパスの指定順をパス一覧表示エリアの表示順に設定し，本ダイアログをクローズします。
キャンセル	パスの指定順の設定をキャンセルし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

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—30　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。

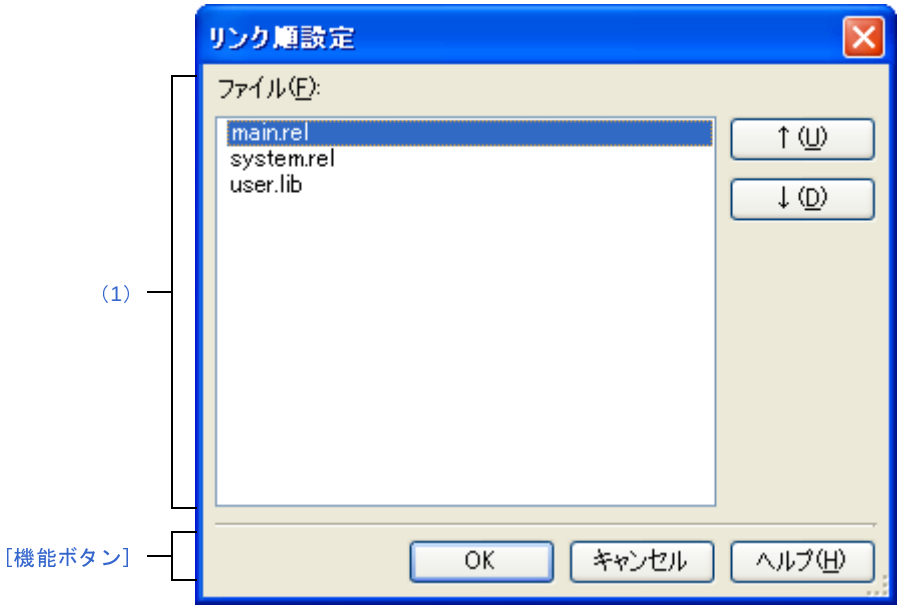
［機能ボタン］

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—31　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—32　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—33　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。

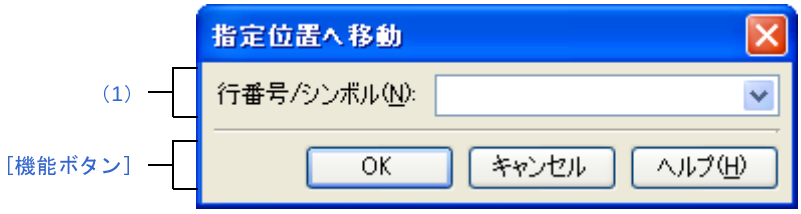
[機能ボタン]

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—34　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］

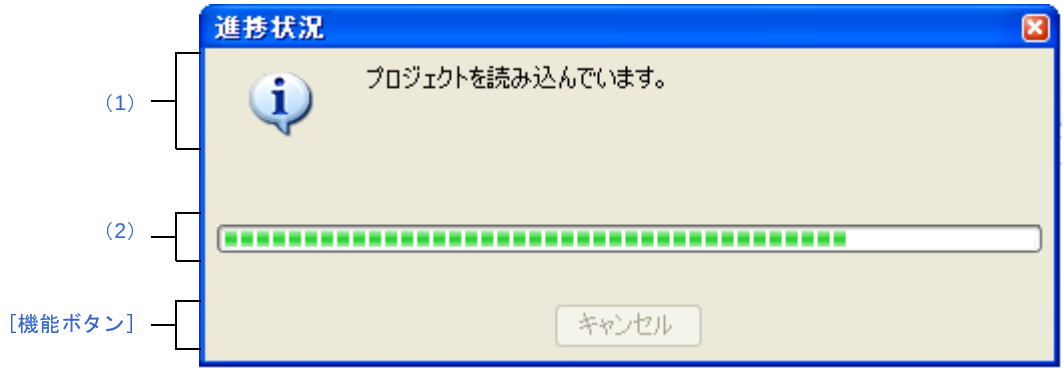
ボタン	機能
OK	指定した位置を対象パネルの表示の中央とし，キャレットを移動します。 なお，開いているファイルを登録しているプロジェクトがアクティブ，かつライブラリ用のプロジェクト以外の場合のみ有効です。
キャンセル	設定を無効とし，本ダイアログをクローズします。
ヘルプ	本ダイアログのヘルプを表示します。

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—35　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]

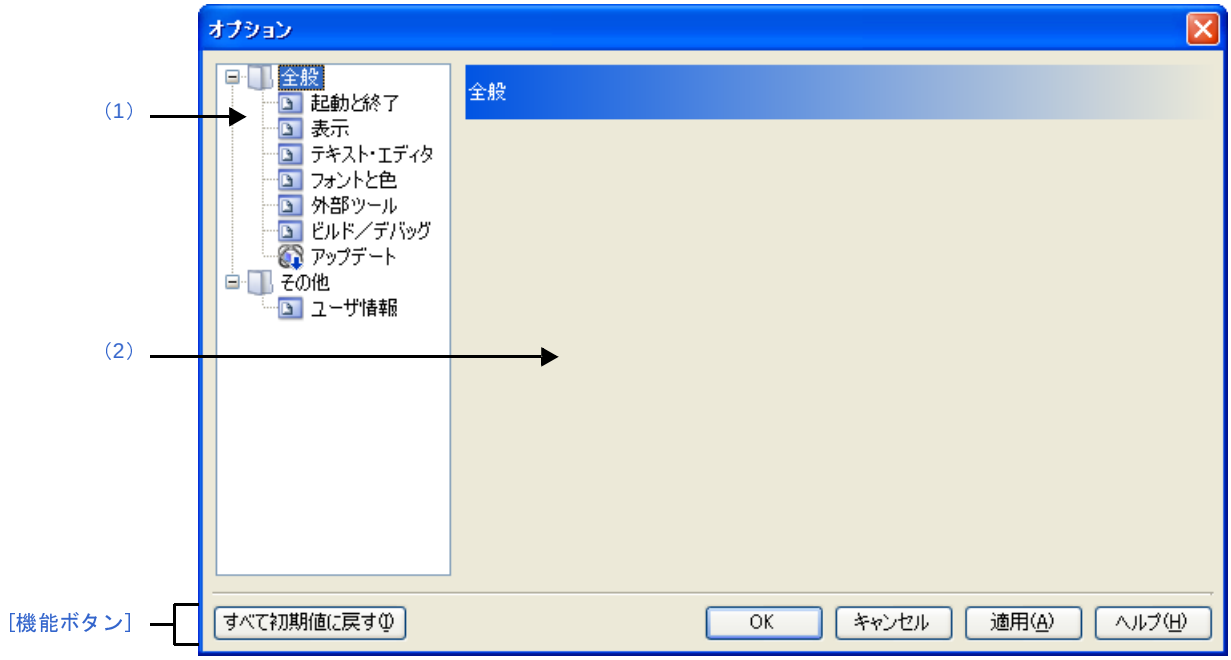
ボタン	機能
キャンセル	現在実行中の処理を中断し，本ダイアログをクローズします。 ただし，実行中の処理の中断が不可能な場合，本ボタンは無効となります。

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—36　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。

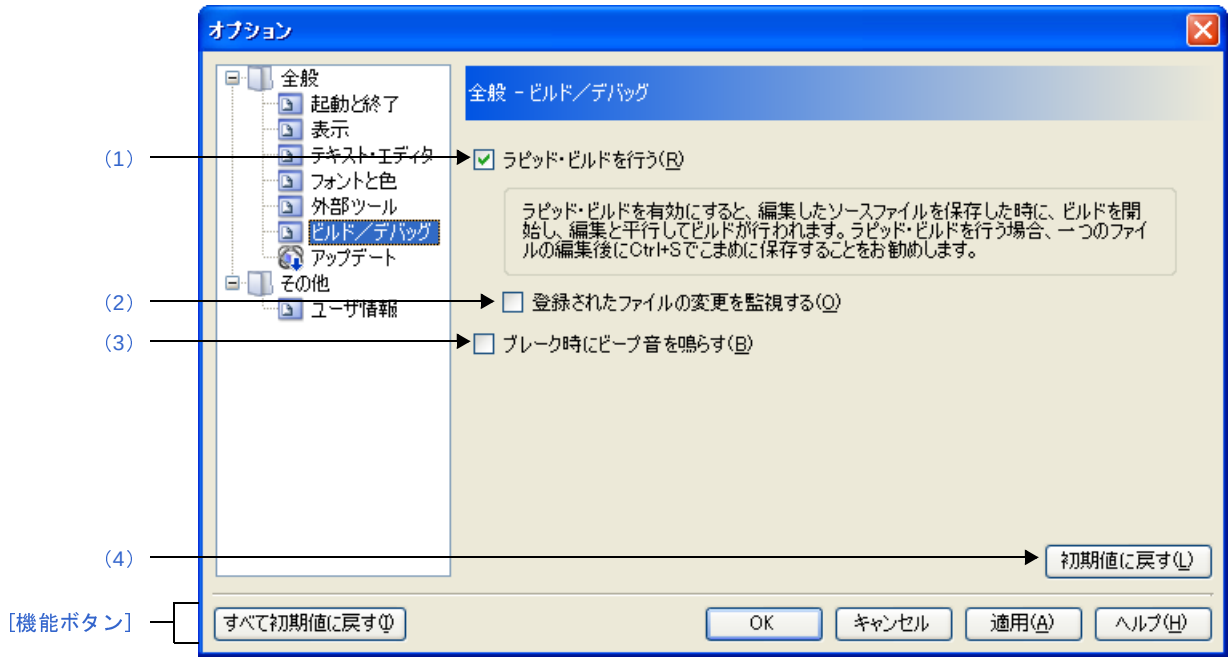
［機能ボタン］

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—37　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。

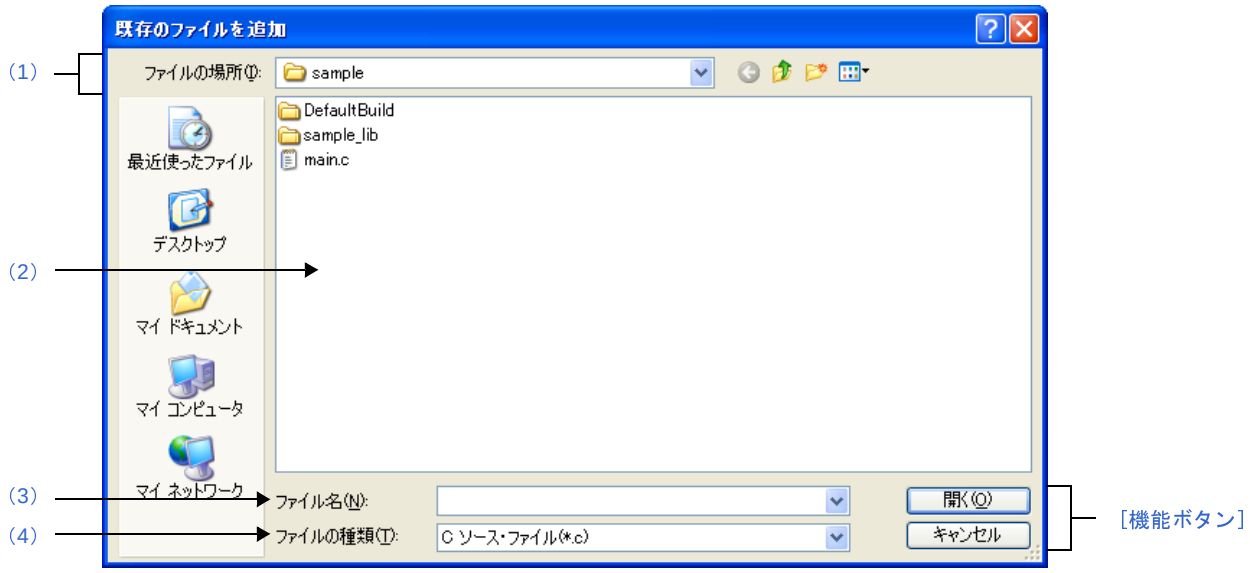
［機能ボタン］

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—38　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 (*.asm)	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 (*.dr; .*dir)	リンク・ディレクティブ・ファイル
変数／関数情報ファイル（*.vfi）	変数／関数情報ファイル
ライブラリ・ファイル (*.lib)	ライブラリ・ファイル
オブジェクト・ファイル (*.rel)	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）

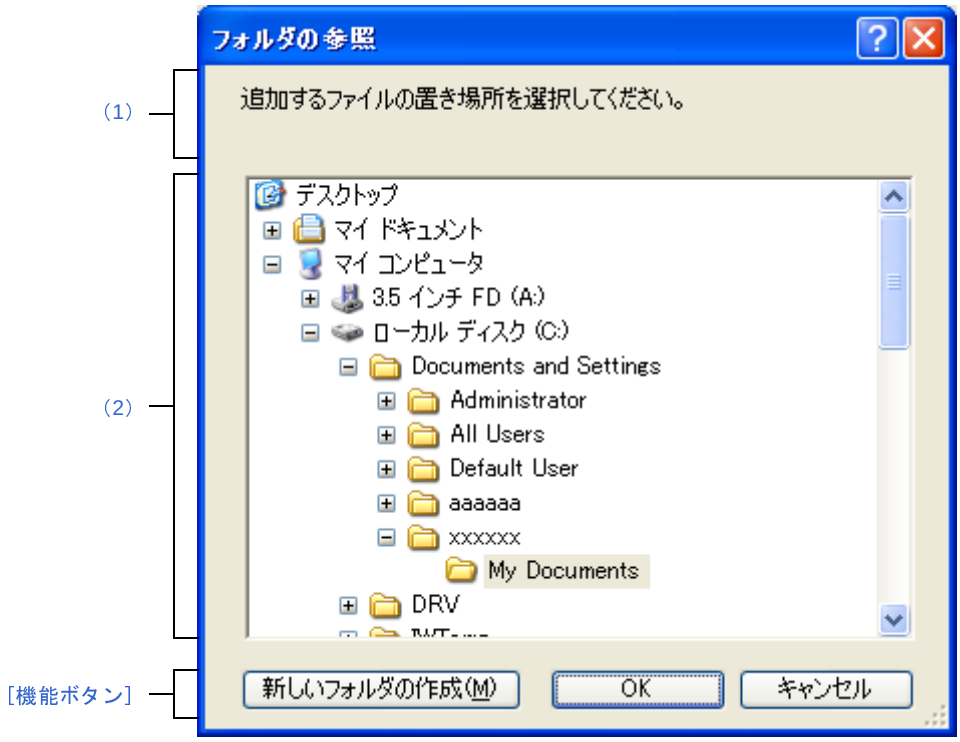
［機能ボタン］

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—39　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（オブジェクト・コンバータ）］カテゴリの［ヘキサ・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］
  - ［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］
  - ［オブジェクト・コンバート・オプション］タブの［ヘキサ・ファイル］カテゴリの［ヘキサ・ファイル出力フォルダ］
  - ［ライブラリ生成オプション］タブの［出力ファイル］カテゴリの［出力フォルダ］
  - ［変数／関数配置オプション］タブの［出力ファイル］カテゴリの［変数／関数情報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。

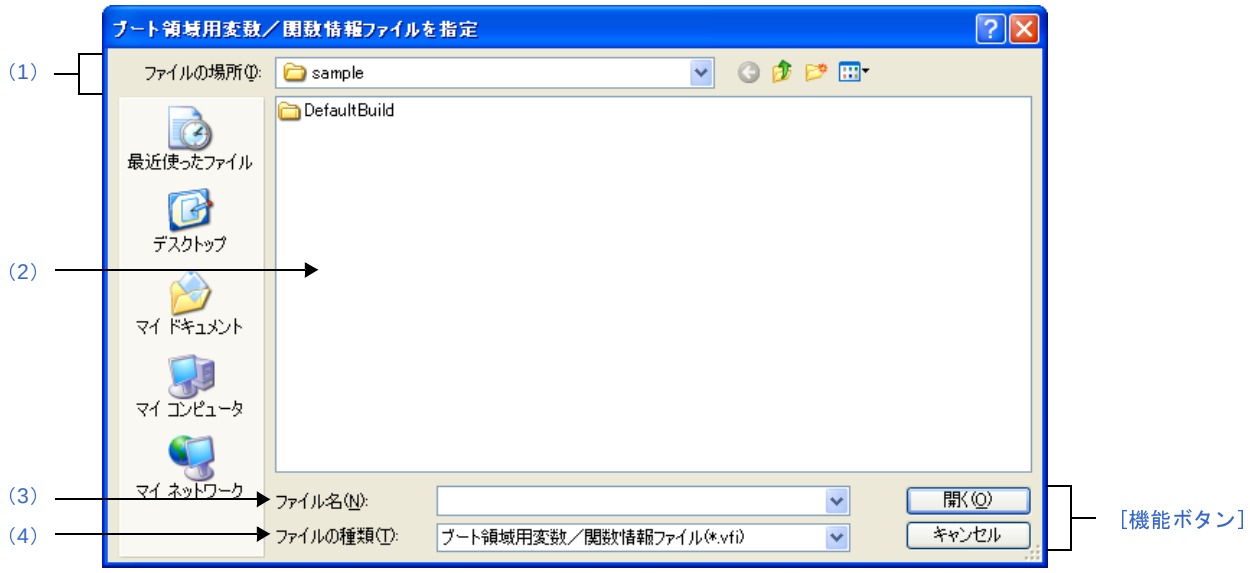
[機能ボタン]

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。

ブート領域用変数／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ブート領域用変数／関数情報ファイルの選択を行います。

図 A—40　ブート領域用変数／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［変数情報ファイル］カテゴリの［ブート領域用変数／関数情報ファイル］

［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。

ブート領域用変数／関数情報ファイル (*.vfi)	ブート領域用変数／関数情報ファイル

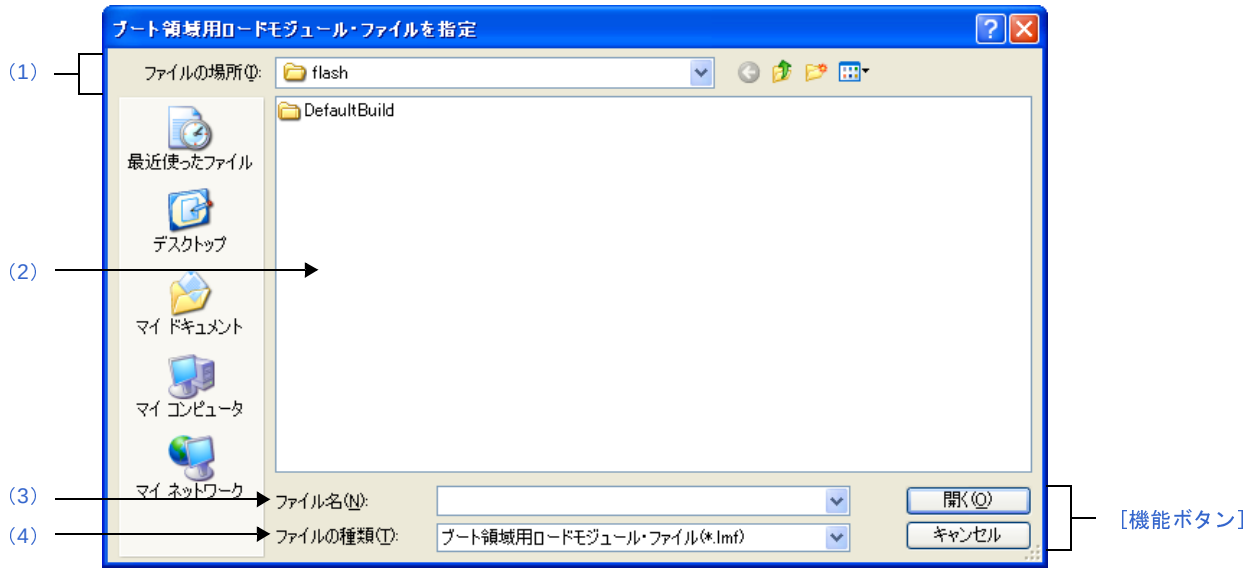
［機能ボタン］

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。

ブート領域用ロード・モジュール・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用ロード・モジュール・ファイルの選択を行います。

図 A—41　ブート領域用ロード・モジュール・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
  - ［リンク・オプション］タブの［デバイス］カテゴリの［ブート領域用ロード・モジュール・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。

ブート領域用ロード・モジュール・ファイル (*.lmf)	ブート領域用ロード・モジュール・ファイル（デフォルト）
すべてのファイル (*.*)	すべての形式

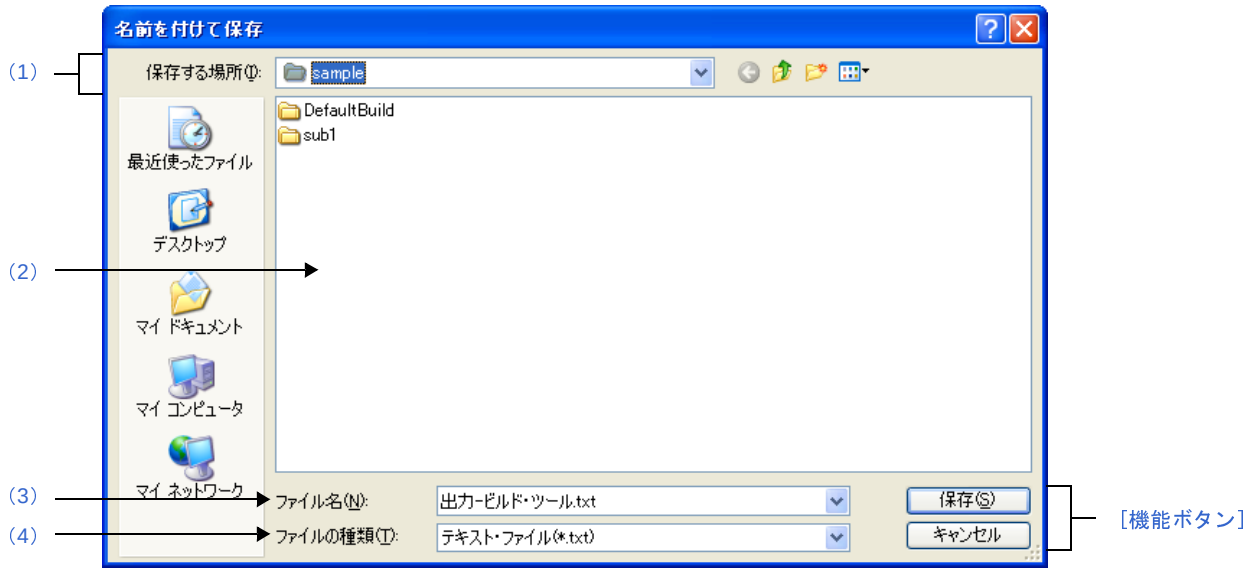
［機能ボタン］

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。

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—42　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 (*.asm)	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 (*.dr; .*dir)	リンク・ディレクティブ・ファイル
変数／関数情報ファイル（*.vfi）	変数／関数情報ファイル
マップ・ファイル (*.map)	マップ・ファイル
シンボル・テーブル・ファイル (*.sym)	シンボル・テーブル・ファイル
ヘキサ・ファイル (*.hex; *.hxb; *.hxf)	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

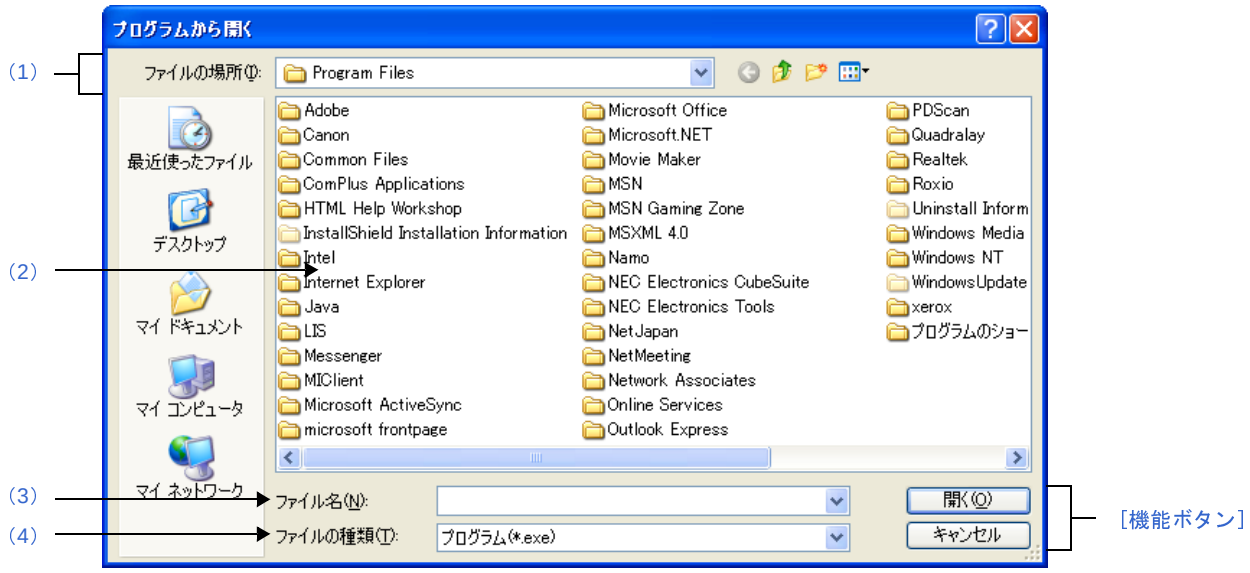
［機能ボタン］

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—43　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］

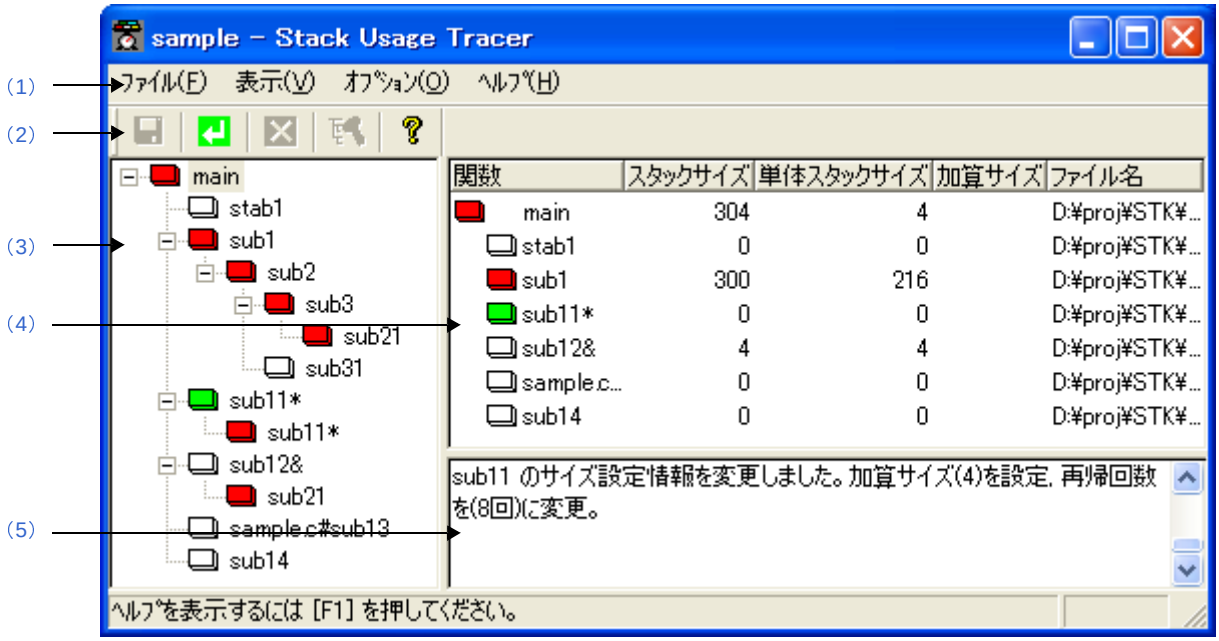
ボタン	機能
開く	指定したアプリケーションでファイルをオープンします。
キャンセル	本ダイアログをクローズします。

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—44　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。
sk78k0r 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

sk78k0r 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。
sk78k0r のバージョン情報	sk78k0r 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→［sk78k0r 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。

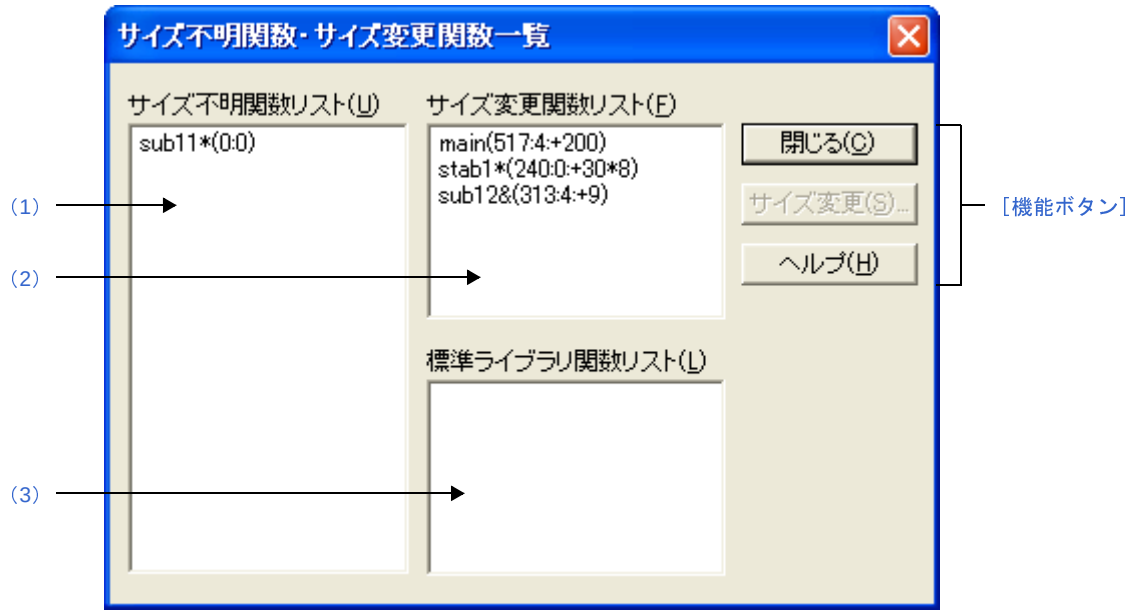
例　再帰関数 func_rec1 ／ func_rec2 の単体スタック・サイズを 8 バイトと仮定し，スタックサイズ変更 ダイアログにおいて，再帰関数 func_rec1 ／ func_rec2 の再帰回数を 3 回と設定した場合，func1 のスタック・サイズは“（8 ＋ 24）× 3”と正しく計算されますが，　func2 のスタック・サイズについては“8 × 3”と func_rec1 の呼び出しが無視された計算が行われます。

```
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—45　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—46　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。

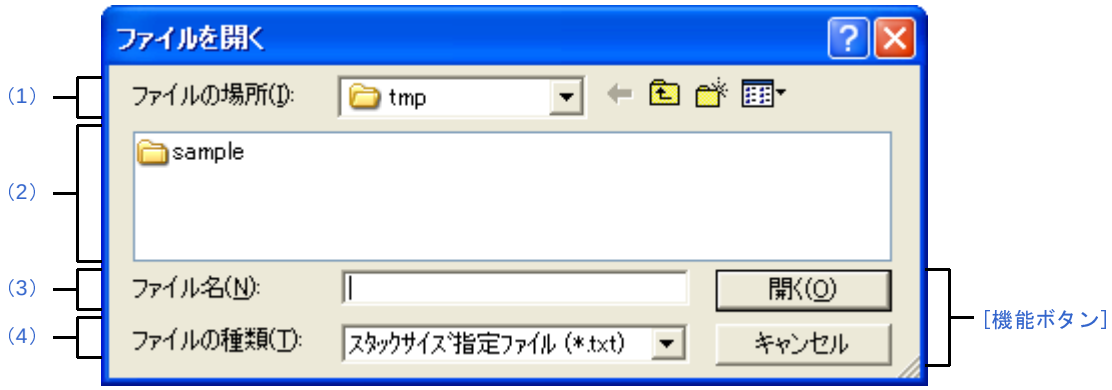
[機能ボタン]

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—47　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の詳細を説明します。

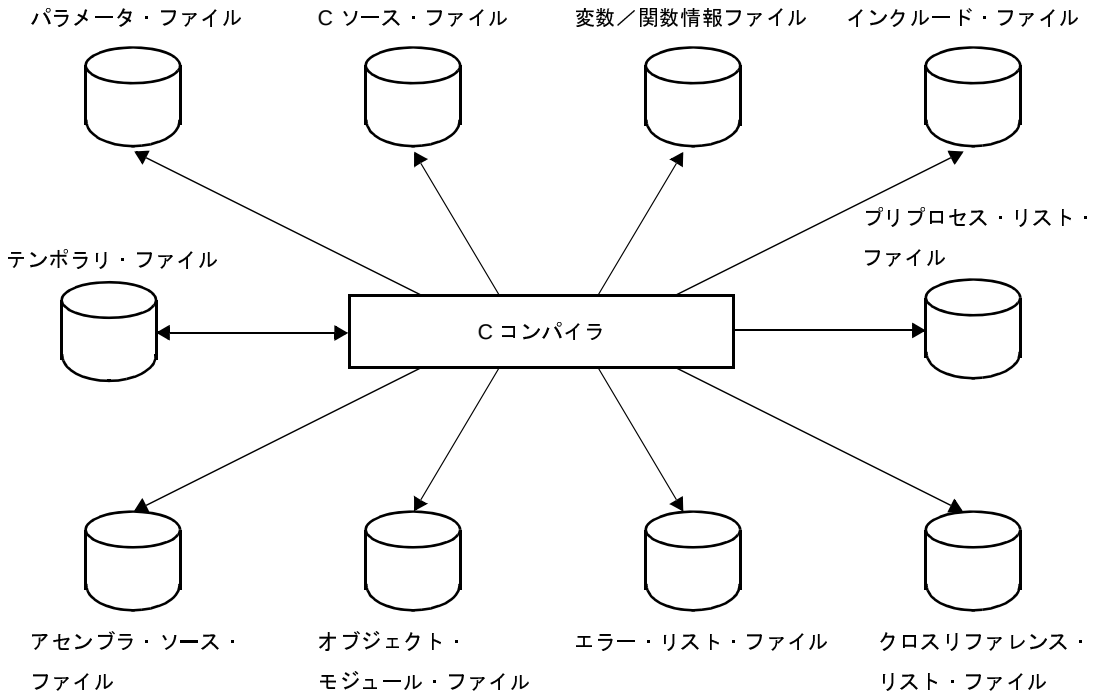
B. 1　C コンパイラ

C コンパイラは，C 言語で記述された C ソース・ファイルを入力し，それらを機械語に変換してオブジェクト・モジュール・ファイルとして出力します。また，コンパイル後にユーザがアセンブリ言語レベルで内容を確認，修正できるように，アセンブラ・ソース・ファイルを出力します。

さらに，コンパイル・オプションにより，プリプロセス・リスト，クロスリファレンス・リスト，エラー・リストなどのリスト・ファイルを出力します。

コンパイル・エラーがある場合は，エラー・メッセージをコンソール，エラー・リスト・ファイルなどに出力します。エラーがある場合は，エラー・リスト・ファイル以外の各種ファイルは出力されません。

図 B—1　C コンパイラの入出力ファイル



備考　コンパイル・エラーがある場合，エラー・リスト・ファイル，クロスリファレンス・リスト・ファイル以外の各種ファイルは出力されません。
テンポラリ・ファイルは，コンパイル正常終了時に正式な名前にリネームされます。また，異常終了時には削除されます。

B. 1. 1　入出力ファイル

C コンパイラの入出力ファイルを次に示します。

出力ファイルについての詳細は，「3. 1　C コンパイラ」を参照してください。

表 B—1　C コンパイラの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	C ソース・ファイル	- C 言語で記述されたソース・ファイル（ユーザ作成ファイル）	.c
	インクルード・ファイル	- C ソース・ファイルで参照するファイル - C 言語で記述されたファイル（ユーザ作成ファイル）	.h
	パラメータ・ファイル	- C コンパイラ実行の際にコマンド行で指定不可能な多数のコマンドを指定したいときにユーザがエディタで作成するファイル	.pcc
	変数／関数情報ファイル	- 変数，および関数の配置に関する情報ファイル	.vfi
出力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・イメージ・ファイル	.rel
	アセンブラ・ソース・ファイル	- コンパイル結果のオブジェクト・コードの ASCII イメージ・ファイル	.asm
	プリプロセス・リスト・ファイル	- #include などのプリプロセス命令を処理した結果のリスト・ファイル - ASCII イメージ・ファイル	.ppl
	クロスリファレンス・リスト・ファイル	- C ソース・ファイル中で使用されている関数名，変数名の情報が入ったリスト・ファイル	.xrf
	エラー・リスト・ファイル	- ソース・ファイルとコンパイル・エラー・メッセージを内容とするリスト・ファイル注	.cer .her .er .ecc
入出力ファイル	テンポラリ・ファイル	- コンパイルのための中間ファイル - コンパイル正常終了時には正式な名前にリネームされ，異常終了時には削除されます。	$nn （ファイル名固定）

注　エラー・リスト・ファイルには，次の 4 通りのファイル・タイプがあります。

ファイル・タイプ	説明
.cer	*.c' ファイルに対する，C ソース付きエラー・リスト・ファイル （-se オプション指定で出力）
.her	*.h' ファイルに対する，C ソース付きエラー・リスト・ファイル （-se オプション指定で出力）
.er	上記以外のファイルに対する，C ソース付きエラー・リスト・ファイル （-se オプション指定で出力）
.ecc	すべてのソース・ファイルに対する，C ソースなしエラー・リスト・ファイル （-e オプション指定で出力）

B. 1. 2　機　　能

（1）最適化手法

CA78K0R では，効率のよいオブジェクト・モジュール・ファイルを生成するために，最適化を行います。サポートする最適化手法を，次に示します。

表 B—2　最適化手法

フェーズ		内容	例
構文解析部	（a）	定数計算のコンパイル時実行	a＝3 * 5 ;　　→ a＝15 ;
	（b）	論理式の部分評価による真／偽の判定	0 && (a || b) → 0 1 || (a && b) → 1
	（c）	ポインタ，配列などのオフセット計算	オフセットをコンパイル時に計算します。
コード生成部	（d）	レジスタ管理	使用していないレジスタを有効に利用します。
	（e）	ターゲット CPU の持つ特殊な命令の利用	a＝a＋1；　→ inc 命令を使用します。 配列要素の代入に転送命令を使用します。
	（f）	短い命令の利用	同じ動作をする命令があった場合，バイト数の短い命令を利用します。 mov a , #0　→ clrb a
	（g）	ロング・ジャンプ命令のショート・ジャンプ命令への変更	出力された中間コードを再操作して行います。

フェーズ		内容	例
オプティマイザ	(h)	共通部分式の削除	a＝b＋c ;　→ a＝b＋c ; d＝b＋c＋e ;　d＝a＋e ;

	(i)	命令のループの外への移動	for (i＝0 ; i < 10 ; i++)
			{
			:
			a＝b＋c ;
			:
			}
			↓
			a＝b＋c ;
			for (i＝0 ; i < 10 ; i++)
			{
			:
			}

	(j)	無用命令の削除	a＝a ;　→ 削除 a＝b ; のあと，a が参照されない場合→ 削除（a はオートマティック変数）
	(k)	複写の削除	a＝b ;　→ c＝b＋d ; c＝a＋d ; これ以降，a が参照されない場合（a はオートマティック変数）。
	(l)	式の演算順序の変更	レジスタに演算結果を残しておいて，有効となる演算を先に実行します。
	(m)	記憶装置の割り付け（テンポラリ変数）	局所的に使われる変数をレジスタに割り付けます。
	(n)	のぞき穴式最適化	特殊パターンの置き換え 例 a＊1→a , a＋0→a
	(o)	演算の強さの軽減	例 a＊2→a＋a , a<<1
	(p)	記憶装置の割り付け（レジスタ変数）	アクセスの速いメモリへデータを割り付けます。 例 レジスタ，saddr（-qr 指定時のみ）
	(q)	ジャンプ最適化（-qj オプション）	連続するジャンプ命令を 1 つにまとめるなど。
	(r)	レジスタ割り付け（-qv/-qr/-rd/-rs オプション）	変数を自動的にレジスタに割り付けるなど。

備考　(a) ～ (g)，(n)，(o) は，最適化オプションの指定によらず行われます。

(h) ～ (m)，(q)，(r) の最適化は，最適化オプションが指定された場合に行われます。

(p) は，C ソース中に register 宣言がある場合に行います。ただし，saddr 領域には，-qr オプション指定時のみ割り付けます。

最適化オプションについては，「最適化指定」を参照してください。

（2）ROM 化機能

ROM 化とは，初期値あり外部変数などの初期値を ROM に配置しておき，システム実行時に RAM にコピーする処理です。

CA78K0R は，プログラムの ROM 化処理付きのスタートアップ・ルーチンを提供しているので，スタートアップ時の ROM 化処理などを記述する手間が省けます。

スタートアップ・ルーチンの内容については，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

（a）プログラムの ROM 化を行う方法

リンク時に，スタートアップ・ルーチン，オブジェクト・モジュール・ファイルとライブラリをリンクします。スタートアップ・ルーチンは，オブジェクト・プログラムの初期化を行います。

- s0r*.rel

スタートアップ・ルーチン（ROM 化対応）です。

初期化データのコピー・ルーチンを含み，初期化データの開始を示します。スタート・アドレスには，“_@cstart”というラベル（シンボル）が付けられます。

- cl0r*.lib

CA78K0R に添付されているライブラリです。

このライブラリ・ファイルの中には，次のものが含まれています。

- ランタイム・ライブラリ

  ランタイム・ライブラリ名は，シンボルの先頭に“@@”が付加されます。

- 標準ライブラリ

  標準ライブラリ名は，シンボルの先頭に“_”が付加されます。

- *.lib

ユーザ作成のライブラリです。

シンボルの先頭に“_”が付加されます。

注意　CA78K0R は，何種類かのスタートアップ・ルーチン，およびライブラリを提供しています。スタートアップ・ルーチン，およびライブラリについては，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

B. 1. 3　操作方法

(1) C コンパイラの起動方法

C コンパイラの起動には，2 つの方法があります。

(a) コマンド行での起動

```
X:[パス名]>cc78k0r[△オプション]...C ソース・ファイル名 [△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
cc78k0r	C コンパイラのコマンド名
オプション	C コンパイラに対して動作の詳細を指示します。 複数のコンパイル・オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。また，コンパイル・オプションに続くサブオプション，ファイル名などは，スペースなどの空白を入れずに続けて指定してください。なお，コンパイル・オプションに大文字，小文字の区別はありません。コンパイル・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。 空白を含むパスを設定する場合は，ダブルクオーテーション（” “）で囲んでください。
C ソース・ファイル名	コンパイルするソース・ファイル名 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（” “）で囲んでください。

例　アセンブラ・ソース・ファイル prime.asm を出力します。また，コード・サイズを優先して最適化を行います。

```
cc78k0r -cf1166a0 prime.c -aprime.asm -qx3
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，コンパイルするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>cc78k0r[△C ソース・ファイル名]△-f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	コンパイラの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[△] オプション [△オプション]...
```

- コマンド行で C ソース・ファイル名を省略した場合は，パラメータ・ファイル内で C ソース・ファイル名を 1 つだけ指定することができます。
- C ソース・ファイル名は，オプションのあとに記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定するすべてのコンパイル・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル k0rmain.pcc をエディタで作成し，コンパイラを起動します。

```
; parameter file
-cf1166a0 k0rmain.c -e -a
```

```
C:¥>cc78k0r -fk0rmain.pcc
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

C コンパイラが起動すると，次の実行開始メッセージが表示されます。

```
78K0R C Compiler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
```

(b) 実行終了メッセージ

コンパイルの結果，コンパイル・エラーが検出されなかった場合，C コンパイラは，次のメッセージを表示して制御をホスト OS に戻します。

```
Target chip : uPD78F1166_A0
Device file : Vx.xx

Compilation complete, 0 error(s) and 0 warning(s) found.
```

コンパイルの結果，コンパイル・エラーが検出された場合，C コンパイラはエラーとワーニングの数を表示して制御をホスト OS に戻します。

```
prime.c(18) : CC78K0R warning W0745 : Expected function prototype
prime.c(20) : CC78K0R warning W0745 : Expected function prototype
prime.c(26) : CC78K0R warning W0622 : No return value
prime.c(37) : CC78K0R warning W0622 : No return value
prime.c(44) : CC78K0R warning W0622 : No return value

Target chip : uPD78F1166_A0
Device file : Vx.xx

Compilation complete, 0 error(s) and 5 warning(s) found.
```

コンパイル中にコンパイル処理継続が不可能な致命的エラーが検出された場合，C コンパイラはメッセージを表示してコンパイルを中止し，制御をホスト OS に戻します。

例　存在しないコンパイル・オプションを指定した場合

```
C:¥>cc78k0r k0rmain.c -s
```

```
78K0R C Compiler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
CC78K0R error F0018 : Option is not recognized '-s'
Please enter 'CC78K0R--' , if you want help messages.
Program aborted.
```

この例では，存在しないコンパイル・オプションを指定したためにエラーとなり，コンパイルが中止されます。

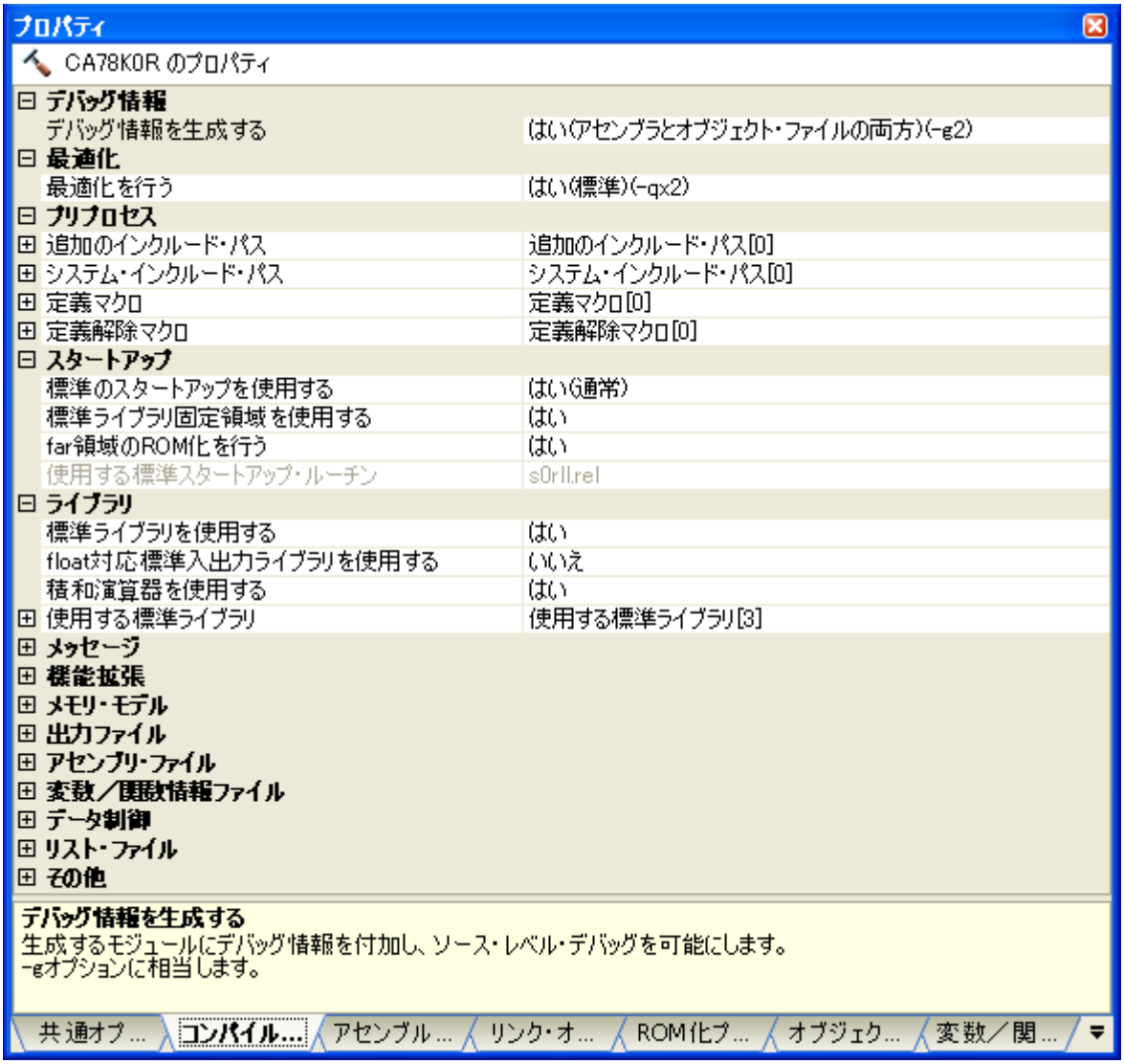
(3) CubeSuite+ でのオプション設定

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 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

（1）種類

コンパイル・オプションは，コンパイラの動作に細かい指示を与えるものです。

コンパイル・オプションの分類と説明を示します。

表 B—3　コンパイル・オプション

分類	オプション	説明
デバイス種別指定	-c	対象とするデバイス種別を指定します。
オブジェクト・モジュール・ファイル作成指定	-o	オブジェクト・モジュール・ファイルの出力を指定します。
	-no	

分類	オプション	説明
メモリ配置指定	-r	メモリの割り付け方法を指定します。
	-nr	
	-rd	外部変数／外部スタティック変数を自動的に saddr 領域に割り付けるように指定します。
	-nr	
	-rs	static auto 変数を自動的に saddr 領域に割り付けるように指定します。
	-nr	
最適化指定	-q	最適化種別を指定します。
	-nq	
デバッグ情報出力指定	-g	C ソース・レベルのデバッグ情報の出力を指定します。
	-ng	
プリプロセス・リスト・ファイル作成指定	-p	プリプロセス・リスト・ファイルの出力を指定します。
	-k	プリプロセス・リストに対する処理を指定します。
プリプロセス指定	-d	マクロ定義を行います。
	-u	マクロ定義を無効にします。
	-i	インクルード・ファイルを指定したフォルダから読み込みます。
アセンブラ・ソース・ファイル作成指定	-a	アセンブラ・ソース・ファイルの出力を指定します。
	-sa	
エラー・リスト・ファイル作成指定	-e	エラー・リスト・ファイルの出力を指定します。
	-se	
クロスリファレンス・リスト・ファイル作成指定	-x	クロスリファレンス・リスト・ファイルの出力を指定します。
リスト形式指定	-lw	各種リスト・ファイルの 1 行の文字数を指定します。
	-ll	各種リスト・ファイルの 1 ページの行数を指定します。
	-lt	各種リスト・ファイルのタブの展開文字数を変更します。
	-lf	各種リスト・ファイルの最後に改頁コードを付加します。
	-li	C ソース付きアセンブラ・ソース・ファイルにインクルード・ファイルの C ソースも付加します。
ワーニング出力指定	-w	ワーニング・メッセージをコンソールへ出力するか否かを指定します。
実行状態表示指定	-v	コンパイルの実行状態をコンソールに出力するか否かを指定します。
	-nv	
パラメータ・ファイル指定	-f	入力ファイル名，およびオプションを，指定したファイルより入力します。
テンポラリ・ファイル作成フォルダ指定	-t	テンポラリ・ファイルを指定したドライブ，フォルダに作成します。

分類	オプション	説明
機能拡張指定	-z	拡張機能に対する処理を有効にします。
	-nz	
デバイス・ファイル・サーチ・パス指定	-y	デバイス・ファイルをサーチするパスを指定します。
メモリ・モデル指定	-m	コンパイル時のメモリ・モデルの種類を指定します。
ミラー領域指定	-mi	ミラー元のセグメントの配置先を指定します。
共通オブジェクト指定	-common	RL78，および 78K0R 共通オブジェクトの出力を指定します。
変数／関数情報ファイル指定	-ma	変数／関数情報ファイルを指定します。
ヘルプ指定	--	コンソールにヘルプ・メッセージを出力します。
	-?	
	-h	

(2) 優先度

次の表に示すコンパイル・オプションのうち，縦軸のものと横軸のものを同時に 2 つ以上指定した場合の優先度について説明します。

表 B—4　コンパイル・オプションの優先度

	-no	-p	-np	-d	-u	-a	-e	-x	-sa
-r	×								
-q	×								
-g	×								
-k		△	×						
-d					○				
-u				○					
-sa						×			
-lw		△				△	△	△	
-ll		△				△	△	△	
-lt		△				△	△	△	
-lf		△				△	△	△	
-li									△

- ×で記した箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例　-rd，-g オプションは，無効となります。

```
C:¥>cc78k0r -cf1166a0 -e sample.c -no -rd -g
```

- △で記した箇所

横軸に示したオプションを指定しない場合，縦軸に示したオプションは無効となります。

例　-p オプションが指定されているので，-k オプションは有効です。

```
C:¥>cc78k0r -cf1166a0 -e sample.c -p -k
```

- ○で記した箇所

横軸のオプションと縦軸のオプションで，後ろに指定したものが優先となります。

例　-d オプションが後ろに指定されているので，-u オプションは無効となり，-d オプションが優先されます。

```
C:¥>cc78k0r -cf1166a0 -e sample.c -utest -dtest=1
```

- 空欄の箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは有効となります。

また，-o/-no オプションのように，オプション名の前に“n”を付加できるオプションを同時に指定した場合，あとで指定した方が有効となります。

例　-no オプションがあとに指定されているので，-o オプションは無効となり，-no オプションが有効となります。

```
C:¥>cc78k0r -cf1166a0 -e sample.c -o -no
```

「表 B—4　コンパイル・オプションの優先度」に記述されていないオプションは，ほかのオプションの影響を特に受けません。しかし，ヘルプ指定オプション（--/-?/-h）が指定された場合には，すべてのオプション指定が無効となります。

デバイス種別指定

デバイス種別指定オプションには，次のものがあります。

- -c

-c

[記述形式]

```
-c デバイス種別
```

- 省略時解釈
  省略することはできません。

[機能]

- -c オプションは，コンパイルの対象となるターゲット・デバイスを指示します。

[用途]

- 必ず指定してください。CA78K0R は，指定されたターゲット・デバイスに対してコンパイルを行い，それに対応したオブジェクト・コードを生成します。

[説明]

- -c オプションで指定可能なターゲット・デバイスと，それに対応するデバイス種別に関しては，「CubeSuite+ 使用上の留意点」を参照してください。
- CA78K0R 使用時には，デバイス・ファイルが必要となります。

[注意]

- -c オプションを省略することはできません。

[使用例]

- コマンド行でターゲット・デバイス uPD78F1166_A0 を指定します。

```
C:¥>cc78k0r -cf1166a0 prime.c
```

オブジェクト・モジュール・ファイル作成指定

オブジェクト・モジュール・ファイル作成指定オプションには，次のものがあります。

- -o/-no

-o/-no

[記述形式]

```
-o [ 出力ファイル名 ]
-no
```

- 省略時解釈

  -o *入力ファイル名* .rel

[機能]

- -o オプションは，オブジェクト・モジュール・ファイルの出力を指示します。また，その出力先や出力ファイル名を指示します。
- -no オプションは，オブジェクト・モジュール・ファイルを出力しません。

[用途]

- オブジェクト・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，-o オプションを指定します。
- アセンブラ・ソース・ファイルの出力のみが目的でコンパイルする場合などに，-no オプションを指定します。これにより，コンパイル時間が短縮されます。

[説明]

- -o オプションを指定する際に出力ファイル名を省略すると，オブジェクト・モジュール・ファイル名は，“*入力ファイル名* .rel”となります。
- -o オプションを指定する際に出力ファイル名の拡張子を省略すると，オブジェクト・モジュール・ファイル名は，“*出力ファイル名* .rel”となります。
- -o オプションを指定しても，コンパイル・エラーがある場合は，オブジェクト・モジュール・ファイルは出力されません。
- -o オプションを指定する際にドライブ名を省略すると，カレント・ドライブにオブジェクト・モジュール・ファイルが出力されます。
- -o と -no の両オプションが同時に指定された場合は，後ろに指定したものが優先となります。

[注意]

- CubeSuite+ 使用時に，出力先を変更する場合は，プロパティ パネルの［リンク・オプション］タブの［出力ファイル］カテゴリに，出力先を指定してください。
- 個別コンパイル・オプション設定時は，出力ファイル名の変更も可能です。
  ［個別コンパイル・オプション］タブの［出力ファイル］カテゴリに，ファイル名を指定してください。

[使用例]

- 先に指定した -no オプションは無効，後ろに指定した -o オプションは有効となり，オブジェクト・モジュール・ファイル prime.rel が出力されます。

```
C:¥>cc78k0r -cf1166a0 prime.c -no -o
```

メモリ配置指定

メモリ配置指定オプションには，次のものがあります。

- -r/-nr
- -rd/-nr
- -rs/-nr

-r/-nr

[記述形式]

```
-r 処理種別（複数指定可能）
-nr
```

- 省略時解釈

  -nr

[機能]

- -r オプションは，メモリへの割り付け方法を指示します。
- -nr オプションは，-r オプションを無効にします。

[用途]

- プログラムをメモリへ割り付けるときの，割り付け方法を指定します。

[説明]

- -r オプションで指定可能な処理種別を，次に示します。

  処理種別の指定を省略することはできません。省略した場合，致命的エラー（F0012）となります。

処理種別	内容
a	間接参照を 1 バイト単位で行います。
b	ビット・フィールドを MSB から割り付けます。
d[*n*][m] （*n* = 1, 2, 4）	外部変数／外部スタティック変数（const 型を除く）を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。 詳細については，「-rd/-nr」を参照してください。
s[*n*][m] （*n* = 1, 2, 4）	static auto 変数を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。 詳細については，「-rs/-nr」を参照してください。
c	間接参照を 1 バイト単位で行います。 さらに，構造体をパッキングし，構造体メンバを 1 バイト・アラインで配置します。

処理種別	内容
f	ROM データを far 領域に配置します。
n	ROM データを near 領域に配置します。

備考　処理種別は，複数指定することができます。

- -nr オプションが指定された場合は，次のように解釈されます。

処理種別	内容
a	間接参照を 1 バイト単位で行いません。
b	ビット・フィールドを LSB から割り付けます。
d	saddr 領域に自動的に割り付けません。
s	saddr 領域に自動的に割り付けません。
c	間接参照を 1 バイト単位で行いません。 さらに，構造体をパッキングせず，構造体メンバを 1 バイト・アラインで配置しません。

[使用例]

- 外部変数／外部スタティック変数，static auto 変数を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。

```
C:¥>cc78k0r -cf1166a0 -rds
```

-rd/-nr

[記述形式]

```
-rd[n][m] (n = 1, 2, 4)
-nr
```

- 省略時解釈
  -nr

[機能]

- -rd オプションは，外部変数／外部スタティック変数を自動的に saddr 領域に割り付けるように指示します。
- -nr オプションは，-rd オプションを無効にします。

[用途]

- 外部変数／外部スタティック変数（const 型を除く）を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。

[説明]

- *n* の値と m の指定によって，割り付ける変数の最大幅を次のように指定します。

n，m の指定	saddr 領域に割り当てる変数
n	- *n* = 1 の場合，char，unsigned char - *n* = 2 の場合，char，unsigned char，short，unsigned short，int，unsigned int，enum，near ポインタ - *n* = 4 の場合，char，unsigned char，short，unsigned short，int，unsigned int，enum，long，unsigned long，ポインタ
m	構造体，共用体，配列
省略した場合	すべての変数

- sreg 宣言された変数は，-rd オプションの指定にかかわらず，saddr 領域に割り付けられます。
- extern 宣言により参照する変数については，saddr 領域に割り付けられるものとして処理されます。
- 本オプションにより saddr 領域に割り付けられた変数は，sreg 変数と同様に扱います。

［使用例］

- char，unsigned char 型の外部変数／外部スタティック変数を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。

```
C:¥>cc78k0r -cf1166a0 -rd1
```

-rs/-nr

[記述形式]

```
-rs[n][m] (n = 1, 2, 4)
-nr
```

- 省略時解釈

  -nr

[機能]

- -rs オプションは，static auto 変数を自動的に saddr 領域に割り付けるように指示します。
- -nr オプションは，-rs オプションを無効にします。

[用途]

- static auto 変数を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。

[説明]

- *n* の値と m の指定によって，割り付ける変数の最大幅を次のように指定します。

n，m の指定	saddr 領域に割り当てる変数
n	- *n* = 1 の場合，char，unsigned char - *n* = 2 の場合，char，unsigned char，short，unsigned short，int，unsigned int，enum，near ポインタ - *n* = 4 の場合，char，unsigned char，short，unsigned short，int，unsigned int，enum，long，unsigned long，ポインタ
m	構造体，共用体，配列
省略した場合	すべての変数

- sreg 宣言された変数は，-rs オプションの指定にかかわらず，saddr 領域に割り付けられます。
- 本オプションにより saddr 領域に割り付けられた変数は，sreg 宣言された static auto 変数と同様に扱います。

[使用例]

- char，unsigned char 型の static auto 変数を sreg 宣言の有無にかかわらず，自動的に saddr 領域に割り付けます。

```
C:¥>cc78k0r -cf1166a0 -rs1
```

最適化指定

最適化指定オプションには，次のものがあります。

- -q/-nq

-q/-nq

[記述形式]

```
-q[最適化種別]（複数指定可能）
-nq
```

- 省略時解釈

  -qx2（-qcjlvw）

[機能]

- -q オプションは，最適化フェーズを呼び出し効率のよいオブジェクトを生成するよう指示します。
- -nq オプションは，-q オプションを無効にします。

[用途]

- オブジェクトの実行速度の向上，コード・サイズを削減したい場合に，-q オプションを指定します。

  -q オプション指定時に，複数の最適化を同時に有効にしたい場合は，最適化種別を続けて指定します。詳細については，［説明］を参照してください。

[説明]

- -q オプションで指定可能な最適化種別を，次に示します。

最適化種別	処理内容
省略	-qx2（-qcjlvw）が指定されたと見なします。
u	修飾子なしの char を unsigned char と見なすことにより，コード効率を良くします。

<table>
<tr><th>最適化種別</th><th>処理内容</th></tr>
<tr><td>c</td><td>char に関する演算を符号拡張せずに行います。
<table>
<tr><th>演算対象</th><th>演算結果</th></tr>
<tr><td>unsigned char 型の変数と unsigned char 型の変数</td><td>unsigned char 型</td></tr>
<tr><td>unsigned char 型の変数と signed char 型の変数</td><td>unsigned char 型</td></tr>
<tr><td>signed char 型の変数と signed char 型の変数</td><td>signed char 型</td></tr>
<tr><td>-128 ～ 255 の定数と unsigned char 型の変数</td><td>unsigned char 型</td></tr>
<tr><td>-128 ～ 127 の定数と signed char 型の変数</td><td>signed char 型</td></tr>
<tr><td>0 ～ 255 で接尾語 U 付きの定数と signed char 型の変数</td><td>unsigned char 型</td></tr>
</table></td></tr>
<tr><td>r</td><td>レジスタ変数をレジスタに加えて saddr 領域にも割り当てます。</td></tr>
<tr><td>j</td><td>分岐命令の最適化を行います。</td></tr>
<tr><td>x[n]
（n ＝ 1 - 3）</td><td>最適化オプションを，スピード／コード・サイズの優先順位により自動的に割り当てます。
n の値によって，割り当てるオプションが次のように異なります。n を省略した場合は，n ＝ 2 として解釈されます。
- n ＝ 1 の場合，スピード優先として，-qcjvw を指定したものと見なす
- n ＝ 2 の場合，デフォルトとして，-qcjlvw を指定したものと見なす
- n ＝ 3 の場合，コード・サイズ優先として，-qcjl3vw を指定したものと見なす</td></tr>
<tr><td>w</td><td>積極的な最適化を行います。
演算式の実行順序の入れ替えを行います。</td></tr>
<tr><td>v</td><td>引数，およびオートマティック変数を，レジスタ，または saddr 領域に自動的に割り当てます。</td></tr>
<tr><td>l[n]
（n ＝ 1 - 3）</td><td>コード・サイズを優先した最適化を行い，定型コード・パターンをライブラリに置き換えます。指定しない場合は，スピード優先の最適化を行います。
n の値によって，ライブラリに置き換える範囲が次のように異なります。n を省略した場合は，n ＝ 1 として解釈されます。
- n ＝ 1 の場合，ライブラリに置き換えない
- n ＝ 2 の場合，関数の前後処理のみライブラリに置き換える
- n ＝ 3 の場合，2 に加えて，低レベル・ライブラリの使用，および共通コードのサブルーチン化を行う</td></tr>
<tr><td>t</td><td>far 領域に配置された関数内の switch 文の分岐テーブルを相対分岐にします。
分岐距離が 64K バイトを越えた場合，エラー・メッセージ（F0924）が出力されます。</td></tr>
<tr><td>g</td><td>デバッグを優先します。</td></tr>
</table>

- 最適化種別は複数指定することができます。
- -q オプションを省略した場合，または最適化種別を省略した場合，-qcjlvw のオプションを指定した場合と同じ最適化を行います。
- デフォルト・オプションの一部を解除するには，解除したいオプション以外のオプションを指定することによって解除することができます（例：-qr を指定 → -qcjlvw を解除）。
- オブジェクト・モジュール・ファイル，アセンブラ・ソース・ファイルのいずれも出力されない場合，-qu 以外の -q オプションは無効となります。
- -q と -nq の両オプションが同時に指定された場合は，後ろに指定したものが有効になります。

- -q オプションが同時に複数指定された場合，最後に指定したものが有効となります。
- リアルタイム OS は，-qr オプションをサポートしていません。

［使用例］

- 修飾子なしの char を unsigned と見なすことにより，コード効率を良くします。

```
C:¥>cc78k0r -cf1166a0 prime.c -qu
```

- 先に指定した -qc オプションは無効，後ろに指定した -qr オプションは有効となり，norec の引数，auto 変数，および register 変数を saddr 領域に割り当てます。

```
C:¥>cc78k0r -cf1166a0 prime.c -qc -qr
```

- -qc と -qr の両方のオプションを有効にしたい場合は，次のようにコマンド入力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -qcr
```

デバッグ情報出力指定

デバッグ情報出力指定オプションには，次のものがあります。

- -g/-ng

-g/-ng

[記述形式]

```
-g[n] (n = 1, 2)
-ng
```

- 省略時解釈

  -g2

[機能]

- -g オプションは，オブジェクト・モジュール・ファイル中にデバッグ情報を付加するよう指示します。
- -ng オプションは，-g オプションを無効にします。

[用途]

- -g オプションを指定しない場合，デバッガへの入力となるオブジェクト・モジュール・ファイルに必要な行番号，シンボル情報が出力されません。したがって，ソース・レベル・デバッグを行うときには，リンクするすべてのモジュールを -g オプション指定でコンパイルします。

[説明]

- *n* の値による動作の違いを次に示します。

n の値	内容
省略	*n* = 2 が指定されたものと見なします。
1	オブジェクト・モジュール・ファイル中のみデバッグ情報（$DGS，$DGL で始まる情報）を付加し，アセンブラ・ソース・ファイル中には付加しません。 本オプションは，アセンブラ・ファイルを参照しやすくするためのものです。 オブジェクト・モジュール・ファイルには，デバッグ情報が付加されるため，ソース・デバッグも可能です。
2	オブジェクト・モジュール・ファイル中，アセンブラ・ソース・ファイル中にデバッグ情報を付加します

- -g と -ng が同時に指定された場合は，後ろに指定したものが有効となります。

- オブジェクト・モジュール・ファイル，アセンブラ・ソース・ファイルのいずれも出力されない場合，-g オプションは無効となります。

［使用例］

- オブジェクト・モジュール・ファイル prime.rel 中にデバッグ情報を付加します。

```
C:￥>cc78k0r -cf1166a0 prime.c -g
```

プリプロセス・リスト・ファイル作成指定

プリプロセス・リスト・ファイル作成指定オプションには，次のものがあります。

- -p
- -k

-p

[記述形式]

```
-p [ 出力ファイル名 ]
```

- 省略時解釈
  なし（ファイルを出力しません）

[機能]

- -p オプションは，プリプロセス・リスト・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。-p オプションが省略された場合，プリプロセス・リスト・ファイルを出力しません。

[用途]

- プリプロセス処理を -k オプションの処理種別に従って行ったあとのソース・ファイルを出力させたいとき，あるいは，プリプロセス・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-p オプションを指定します。

[説明]

- -p オプションを指定する際に出力ファイル名を省略すると，プリプロセス・リスト・ファイル名は，“*入力ファイル名* .ppl”となります。
- -p オプションを指定する際に出力ファイル名の拡張子を省略すると，プリプロセス・リスト・ファイル名は，“*出力ファイル名* .ppl”となります。
- -p オプションを指定する際にドライブ名を省略すると，カレント・ドライブにプリプロセス・リスト・ファイルが出力されます。

[注意]

- CubeSuite+ 使用時は，出力ファイル名を変更することはできません。

［使用例］

- プリプロセス・リスト・ファイル sample.ppl を出力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -psample.ppl
```

-k

[記述形式]

```
-k[処理種別]（複数指定可能）
```

- 省略時解釈
  -kfln

[機能]

- -k オプションは，プリプロセス・リストに対する処理を指示します。

[用途]

- プリプロセス・リスト・ファイルを出力する際にコメントを削除したり，定義の展開を参照するときに指定します。

[説明]

- -k オプションで指定可能な処理種別を次に示します。

処理種別	内容
省略	-kfln が指定されたものと見なします。
c	コメントの削除
d	#define の展開
f	#if，#ifdef，#ifndef の条件コンパイル
i	#include の展開
l	#line の処理
n	行番号とページング処理

備考　処理種別は，複数指定することができます。

- -p オプションが指定されない場合は，-k オプションは無効となります。
- -k オプションが同時にいくつか指定された場合は，最後に指定したものが有効となります。

［使用例］

- プリプロセス・リスト・ファイル prime.ppl を出力する際，コメントの削除，行番号とページング処理を行います。

```
C:¥>cc78k0r -cf1166a0 prime.c -p -kcn
```

出力例は，以下のようになります。

```
/*
78K0R C Compiler Vx.xx Preprocess List          Date:xx xxx xxxx    Page:   1

Command   : -cf1166a0 prime.c -p -kcn
In-file   : prime.c
PPL-file  : prime.ppl
Para-file :
*/

        1 : #define TRUE    1
        2 : #define FALSE   0
        3 : #define SIZE    200
        4 :
        5 : char    mark [ SIZE + 1 ] ;
        6 :
        7 : main ( )
        8 : {
         :
/*
 Target chip : uPD78F1166_A0
 Device file : Vx.xx
*/
```

プリプロセス指定

プリプロセス指定オプションには，次のものがあります。

- -d
- -u
- -i

-d

[記述形式]

```
-d マクロ名 [= 定義名 ] [, マクロ名 [= 定義名 ]] …（複数指定可能）
```

- 省略時解釈

  C ソース・ファイル中のマクロ定義のみを有効とします。

[機能]

- -d オプションは，C ソース・ファイル中の #define 文と同様のマクロ定義を行うよう指示します。

[用途]

- 特定のマクロ定義を有効にしたいときに指定します。

[説明]

- 各定義を“ , ”で区切ることにより，一度に 30 個までマクロ定義を行うことができます。
- “ = ”と“ , ”の前後には，空白を入れることはできません。
- 定義名が省略されると，“マクロ名 =1”として扱われます。
- -d と -u の両オプションで同じマクロ名が指定された場合，後ろに指定したものが有効となります。

[使用例]

- C ソース・ファイル prime.c 中に，

  #define TEST　1

  #define TIME　10

  の定義が行われていたものとします。

```
C:¥>cc78k0r -cf1166a0 prime.c -dTEST,TIME=10
```

-u

[記述形式]

```
-u マクロ名 [, マクロ名 ] …（複数指定可能）
```

- 省略時解釈

  -d で指定したマクロ定義を有効とします。

[機能]

- -u オプションは，C ソース・ファイル中の #undef 文と同様にマクロ定義を無効にします。

[用途]

- -d オプションで定義されたマクロ名を無効にするときに指定します。

[説明]

- 各マクロ名を“ , ”で区切ることにより，1 度に 30 個までマクロ定義を無効にすることができます。

  なお，“ , ”の前後には空白を入れることはできません。
- -u オプションで無効にすることができるマクロ定義は，-d オプションで定義されたものです。

  C ソース・ファイル中に #define によって定義されたマクロ名や，CA78K0R の持つシステム・マクロ名は，-u オプションで無効にすることはできません。
- -d と -u の両オプションで同じマクロ名が指定された場合は，後ろに指定したものが有効となります。

[使用例]

- 先に指定した -d オプションは無効，後ろに指定した -u オプションは有効となり，TEST のマクロ定義を無効にしています。

```
C:¥>cc78k0r -cf1166a0 prime.c -dTEST,TIME=10 -uTEST
```

-i

[記述形式]

```
-i フォルダ[, フォルダ] …（複数指定可能）
```

- 省略時解釈

下記フォルダが指定されたものとして扱われます。

（1）ソース・ファイルのあるフォルダ注1

（2）環境変数 INC78K0R により指定されたフォルダ

（3）C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥V*x.xx*¥inc78k0r 注2

注1. #include 文で，インクルード・ファイル名を" "（ダブル・コーテーション）で指定した場合は，ソース・ファイルのあるフォルダを最初に検索します。< > で指定した場合は，検索されません。

2. C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥V*x.xx*にインストールした場合の例です。

[機能]

- -i オプションは，C ソース・ファイル中の #include 文で指定されたインクルード・ファイルを指定されたフォルダから入力するよう指示します。

[用途]

- インクルード・ファイルをあるフォルダから検索したいときに指定します。

[説明]

- “ , ”で区切ることにより，一度に 64 個までフォルダを指定することができます。
  なお，“ , ”の前後には空白を入れることはできません。
- -i に続いてフォルダが複数指定されるか，あるいは -i オプションが複数指定された場合，指定された順番に #include で指定したファイルを検索します。
- 検索順序は次のようになります。

（1）ソース・ファイルのあるフォルダ注1

（2）-i オプションにより指定されたフォルダ

（3）環境変数 INC78K0R により指定されたフォルダ

（4）C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥V*x.xx*¥inc78k0r 注 2

注 1. #include 文で，インクルード・ファイル名を " "（ダブル・コーテーション）で指定した場合は，ソース・ファイルのあるフォルダを最初に検索します。< > で指定した場合は，検索されません。

2. C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥V*x.xx* にインストールした場合の例です。

［使用例］

- C ソース・ファイル prime.c 中の #include 文で指定されたインクルード・ファイルをフォルダ D:，および D:¥sample から入力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -iD:,D:¥sample
```

アセンブラ・ソース・ファイル作成指定

アセンブラ・ソース・ファイル作成指定オプションには，次のものがあります。

- -a
- -sa

-a

[記述形式]

```
-a [ 出力ファイル名 ]
```

- 省略時解釈
  アセンブラ・ソース・ファイルを出力しません。

[機能]

- -a オプションは，アセンブラ・ソース・ファイルの出力を指示します。また，その出力先や出力ファイル名を指示します。

[用途]

- アセンブラ・ソース・ファイルの出力先や出力ファイル名を変更したいときに，-a オプションを指定します。

[説明]

- -a オプションを指定する際に出力ファイル名を省略すると，アセンブラ・ソース・ファイル名は，“*入力ファイル名* .asm”となります。
- -a オプションを指定する際に出力ファイル名の拡張子を省略すると，アセンブラ・ソース・ファイル名は，“*出力ファイル名* .asm”となります。
- -a オプションを指定する際にドライブ名を省略すると，カレント・ドライブにアセンブラ・ソース・ファイルが出力されます。
- -a と -sa の両オプションが同時に指定された場合は，-sa オプションは無視されます。

[注意]

- CubeSuite+ 使用時は，出力ファイル名を変更することはできません。

［使用例］

- アセンブラ・ソース・ファイル sample.asm を出力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -asample.asm
```

-sa

[記述形式]

```
-sa[出力ファイル名]
```

- 省略時解釈
  アセンブラ・ソース・ファイルを出力しません。

[機能]

- -sa オプションは，アセンブラ・ソース・ファイルにコメントとして C ソースを付加します。
  また，その出力先や出力ファイル名を指示します。

[用途]

- アセンブラ・ソース・ファイルと C ソース・ファイルを一緒に出力したいときに，-sa オプションを指定します。

[説明]

- -sa オプションを指定する際に出力ファイル名を省略すると，アセンブラ・ソース・ファイル名は，“*入力ファイル名* .asm”となります。
- -sa オプションを指定する際に出力ファイル名の拡張子を省略すると，アセンブラ・ソース・ファイル名は，“*出力ファイル名* .asm”となります。
- -sa オプションを指定する際にドライブ名を省略すると，カレント・ドライブにアセンブラ・ソース・ファイルが出力されます。
- -sa と -a の両オプションが同時に指定された場合は，-sa オプションは無視されます。
- 出力アセンブラ・ソース・ファイルのコメントには，インクルード・ファイルの C ソースは付加しません。ただし，-li オプションを指定した場合は，コメントにインクルード・ファイルの C ソースも付加します。

[注意]

- CubeSuite+ 使用時は，出力ファイル名を変更することはできません。

[使用例]

- アセンブラ・ソース・ファイル prime.asm にコメントとして C ソース・ファイル prime.c を付加します。

```
C:¥>cc78k0r -cf1166a0 prime.c -sa
```

出力例は，以下のようになります。

```
; 78K0R C Compiler Vx.xx Assembler Source              Date:xx xxx xxxx  Time:xx:xx:xx

; Command   : -cf1166a0 prime.c -sa
; In-file   : prime.c
; Asm-file  : prime.asm
; Para-file :

$PROCESSOR ( f1166a0 )
$DEBUG
$NODEBUGA
$KANJICODE SJIS
$TOL_INF        03FH , 100H , 00H , 00H , 00H

$DGS    FIL_NAM , .file ,       037H , 0FFFEH , 03FH , 067H , 01H , 00H
$DGS    AUX_FIL , prime.c
$DGS    MOD_NAM , prime ,       00H , 0FFFEH , 00H , 077H , 00H , 00H
          :
        EXTRN   _@RTARG0
        EXTRN   @@isrem
        PUBLIC  _printf
        PUBLIC  _putchar
        PUBLIC  _mark
        PUBLIC  _main
          :
@@CODEL CSEG
_main :
$DGL    1 , 19
        push    hl                      ; [ INF ] 1 , 1
        subw    sp , #08H               ; [ INF ] 2 , 1
        movw    hl , sp                 ; [ INF ] 3 , 1
??bf_main :
; line   9 :    int i , prime , k , count ;
; line  10 :
; line  11 :    count = 0 ;
$DGL    0 , 4
        clrw    ax                      ; [ INF ] 1 , 1
        movw    [ hl ] , ax     ; count ; [ INF ] 1 , 1
; line  12 :
; line  13 :    for ( i = 0 ; i <= SIZE ; i++ )
$DGL    0 , 6
        movw    [ hl + 6 ] , ax ; i     ; [ INF ] 2 , 1
?L0003 :
        movw    ax , [ hl + 6 ] ; i     ; [ INF ] 2 , 1
        cmpw    ax , #0C8H      ; 200   ; [ INF ] 3 , 1
        or1     CY , a.7                ; [ INF ] 2 , 1
```

```
        skc                             ; [ INF ] 2 , 1
        bnz     $?L0004                 ; [ INF ] 2 , 4
          :


; *** Code Information ***
;
; $FILE C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥Vx.xx¥smp78k0r¥cc78k0r¥
prime.c
;
; $FUNC main ( 8 )
;       bc = ( void )
;       CODE SIZE = 117 bytes , CLOCK_SIZE = 86 clocks , STACK_SIZE = 16 bytes
;
; $CALL printf ( 18 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $CALL putchar ( 18 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $CALL putchar( 20 )
;       bc = ( int : ax )
;
; $CALL printf ( 25 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $FUNC printf ( 31 )
;       bc = ( pointer s : ax , int i : [ sp + 4 ] )
;       CODE SIZE = 22 bytes , CLOCK_SIZE = 20 clocks , STACK_SIZE = 14 bytes
;
; $FUNC putchar ( 41 )
;       bc = ( char c : x )
;       CODE SIZE = 16 bytes , CLOCK_SIZE = 16 clocks , STACK_SIZE = 6 bytes


; Target chip : uPD78F1166_A0
; Device file : Vx.xx
```

エラー・リスト・ファイル作成指定

エラー・リスト・ファイル作成指定オプションには，次のものがあります。

- -e
- -se

-e

[記述形式]

```
-e [ 出力ファイル名 ]
```

- 省略時解釈
  エラー・リスト・ファイルを出力しません。

[機能]

- -e オプションは，エラー・リスト・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。

[用途]

- エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-e オプションを指定します。

[説明]

- -e オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は，“*入力ファイル名*.ecc”となります。
- -e オプションを指定する際に出力ファイル名の拡張子を省略すると，エラー・リスト・ファイル名は，“*出力ファイル名*.ecc”となります。
- -e オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- -w0 オプションが指定された場合には，ワーニング・メッセージは出力されません。

[注意]

- CubeSuite+ 使用時は，出力ファイル名を変更することはできません。

[使用例]

- エラー・リスト・ファイル prime.ecc を出力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -e
```

出力例は，以下のようになります。

```
prime.c( 18 ) : CC78K0R warning W0745: Expected function prototype
prime.c( 20 ) : CC78K0R warning W0745: Expected function prototype
prime.c( 26 ) : CC78K0R warning W0622: No return value
prime.c( 37 ) : CC78K0R warning W0622: No return value
prime.c( 44 ) : CC78K0R warning W0622: No return value

 Target chip : uPD78F1166_A0
 Device file : Vx.xx
Compilation complete, 0 error(s) and 5 warning(s) found.
```

-se

[記述形式]

```
-se[出力ファイル名]
```

- 省略時解釈
  エラー・リスト・ファイルを出力しません。

[機能]

- -se オプションは，エラー・リスト・ファイルに C ソース・ファイルを付加します。また，その出力先や出力ファイル名を指示します。

[用途]

- エラー・リスト・ファイルと C ソース・ファイルを一緒に出力したいときに，-se オプションを指定します。

[説明]

- -se オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は，“*入力ファイル名*.cer”となります。
- -se オプションを指定する際に出力ファイル名の拡張子を省略すると，エラー・リスト・ファイル名は，“*出力ファイル名*.cer”となります。
- -se オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- インクルード・ファイルに対しては，フォルダ，およびファイル名の指定を行うことはできません。
  インクルード・ファイルのファイル・タイプが“H”の場合は“her”，“C”の場合は“cer”，それ以外の場合は“er”のファイル・タイプを持つエラー・リスト・ファイルをカレント・ドライブに出力します。
- エラーがなかった場合は，C ソースは付加されません。また，その場合は，インクルード・ファイルに対しては，エラー・リスト・ファイルは作成されません。
- -w0 オプションが指定された場合には，ワーニング・メッセージは出力されません。

[注意]

- CubeSuite+ 使用時は，出力ファイル名を変更することはできません。

[使用例]

- エラー・リスト・ファイル prime.cer に C ソース・ファイル prime.c を付加します。

```
C:¥>cc78k0r -cf1166a0 prime.c -se
```

出力例は，以下のようになります。

```
/*
78K0R C Compiler Vx.xx Error List              Date:xx xxx xxxx  Time:xx:xx:xx

Command   : -cf1166a0 prime.c -se
In-file   : prime.c
Err-file  : prime.cer
Para-file :
*/

#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark [ SIZE + 1 ] ;

void main ( void ) {
          :
        prime = i + i + 3 ;
        printf ( "%6d" , prime ) ;
*** CC78K0R warning W0745: Expected function prototype
        count++ ;
        if ( ( count%8 ) == 0 ) putchar ( '¥n' ) ;

*** CC78K0R warning W0745: Expected function prototype
        for ( k = i + prime ; k <= SIZE ; k += prime )
          :
}
```

クロスリファレンス・リスト・ファイル作成指定

クロスリファレンス・リスト・ファイル作成指定オプションには，次のものがあります。

- -x

-x

[記述形式]

```
-x [ 出力ファイル名 ]
```

- 省略時解釈
  クロスリファレンス・リスト・ファイルを出力しません。

[機能]

- -x オプションは，クロスリファレンス・リスト・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。クロスリファレンス・リスト・ファイルは，シンボルの参照頻度，シンボルの定義／参照箇所を調べるのに有効です。

[用途]

- クロスリファレンス・リスト・ファイルを出力したいとき，および出力先や出力ファイル名を変更したいときに -x オプションを指定します。

[説明]

- -x オプションを指定する際に出力ファイル名を省略すると，クロスリファレンス・リスト・ファイル名は，“*入力ファイル名* .xrf”となります。
- -x オプションを指定する際に出力ファイル名の拡張子を省略すると，クロスリファレンス・リスト・ファイル名は，“*出力ファイル名* .xrf”となります。
- C0101 以外の内部エラー，F0024，E から始まる番号のコンパイル・エラーが発生した場合でも，クロスリファレンス・リスト・ファイルを作成します。ただし，ファイルの内容は保証しません。

[注意]

- CubeSuite+ 使用時は，出力ファイル名を変更することはできません。

[使用例]

- クロスリファレンス・リスト・ファイル prime.xrf を出力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -x
```

出力例は，以下のようになります。

```
78K0R C Compiler Vx.xx Cross reference List                Date:xx xxx xxxx  Page:   1

Command   : -cf1166a0 prime.c -x
In-file   : prime.c
Xref-file : prime.xrf
Para-file :

ATTRIB MODIFY TYPE    SYMBOL          DEFINE   REFERENCE

EXTERN NEAR   array   mark                5       29      31      37
EXTERN FAR    func    printf              7       33      40
REG1          pointer s                   7       13
PARAM
REG1          int     i                   7       12
PARAM
REG1          int     j                   9       12
REG1          pointer ss                 10       13
EXTERN FAR    func    putchar            16       35
REG1          char    c                  16       19
PARAM
REG1          char    d                  18       19
EXTERN FAR    func    main               22
REG1          int     i                  24       28       28       28       29       30
                                                 30       30       31       32       32
                                                 36
REG1          int     prime              24       32       33       36       36
REG1          int     k                  24       36       36       36       37
REG1          int     count              24       26       34       35       40
              #define TRUE                1       29
              #define FALSE               2       37
              #define SIZE                3        5       28       30       36


 Target chip : uPD78F1166_A0
 Device file : Vx.xx
```

リスト形式指定

リスト形式指定オプションには，次のものがあります。

- -lw
- -ll
- -lt
- -lf
- -li

-lw

[記述形式]

```
-lw[文字数]
```

- 省略時解釈
  -lw132（コンソール出力の場合は 80 文字とします）

[機能]

- -lw オプションは，各種リスト・ファイルの 1 行の文字数を指示します。

[用途]

- 各種リスト・ファイルの 1 行の文字数を変更したいときに，-lw オプションを指定します。

[説明]

- -lw オプションで指定可能な文字数の範囲は，ターミネータ（CR，LF）は含めないで，72 ～ 132 です。
- 文字数を省略した場合は，1 行の文字数は 132 文字となります（コンソール出力の場合には，80 文字となります）。
- リスト・ファイルが何も指定されない場合，-lw オプションは無効となります。

[使用例]

- クロスリファレンス・リスト・ファイル prime.xrf の 1 行の文字数を 72 文字とします。

```
C:¥>cc78k0r -cf1166a0 prime.c -x -lw72
```

-ll

[記述形式]

```
-ll[行数]
```

- 省略時解釈

  改ページしません。

[機能]

- -ll オプションは，各種リスト・ファイルの 1 ページの行数を指示します。

[用途]

- 各種リスト・ファイルの 1 ページの行数を変更したいときに，-ll オプションを指定します。

[説明]

- -ll オプションで指定可能な行数の範囲は，20 ～ 65535 です。
- -ll0 を指定すると，改頁しません。
- 行数を省略した場合は，改ページしません。
- リスト・ファイルが何も指定されない場合，-ll オプションは無効となります。

[使用例]

- クロスリファレンス・リスト・ファイル prime.xrf の 1 ページの行数を 20 行とします。

```
C:¥>cc78k0r -cf1166a0 prime.c -x -ll20
```

-lt

[記述形式]

```
-lt[文字数]
```

- 省略時解釈
  -lt8

[機能]

- -lt オプションは，ソース中の HT（Horizontal Tabulation）コードを，各種リスト上でいくつかのブランク（空白）に置き換えて出力する（タビュレーション処理）ための，基本となる文字数を指示します。

[用途]

- -lw オプションで各種リストの 1 行の文字数を少なく指定した場合などに HT コードによるブランクを少なくし，文字数を節約するために -lt オプションを指定します。

[説明]

- -lt オプションで指定可能な文字数の範囲は，0 ～ 8 です。
- -lt0 を指定した場合，タビュレーション処理は行わず，タブ・コードを出力します。
- 文字数を省略した場合は，タブの展開文字数は 8 文字となります。
- リスト・ファイルが何も指定されない場合，-lt オプションは無効となります。

[使用例]

- -lt オプションの省略により，-lt8 オプションが指定されたものと見なされ，HT コードによるブランクを 8 とします。

```
C:¥>cc78k0r -cf1166a0 prime.c -p
```

- HT コードによるブランクを 1 とします。

```
C:¥>cc78k0r -cf1166a0 prime.c -p -lt1
```

-lf

[記述形式]

```
-lf
```

- 省略時解釈

  改頁コードを付加しません。

[機能]

- -lf オプションは，各種リスト・ファイルの最後に改頁コードを付加することを指示します。

[説明]

- リスト・ファイルが何も指定されない場合，-lf オプションは無効となります。

[使用例]

- アセンブラ・ソース・ファイル prime.asm の最後に改頁コードを付加します。

```
C:¥>cc78k0r -cf1166a0 prime.c -a -lf
```

-li

[記述形式]

```
-li
```

- 省略時解釈
  インクルード・ファイルの C ソースを付加しません。

[機能]

- -li オプションは，C ソース・コメント付きアセンブラ・ソース・ファイルに，インクルード・ファイルの C ソースも付加します。

[説明]

- -sa オプションを指定しない場合は，本オプションは無視されます。

[使用例]

- C ソース・コメント付きアセンブラ・ソース・ファイル prime.asm に，インクルード・ファイルの C ソースも付加します。

```
C:¥>cc78k0r -cf1166a0 prime.c -sa -li
```

ワーニング出力指定

ワーニング出力指定オプションには，次のものがあります。

- -w

-w

[記述形式]

```
-w[レベル]
```

- 省略時解釈

  -w1

[機能]

- -w オプションは，ワーニング・メッセージをコンソールに出力するか否かを指定します。

[用途]

- ワーニング・メッセージをコンソールに出力するか否かを指定します。

  また，詳細なメッセージを出力させることも可能です。

[説明]

- ワーニング・メッセージのレベルには，次のものがあります。

レベル	説明
0	ワーニング・メッセージを出力しません。
1	通常のワーニング・メッセージを出力します。
2	詳細なワーニング・メッセージを出力します。

- -e，-se オプションが指定された場合には，エラー・リスト・ファイルにもワーニング・メッセージが出力されます。
- レベル 0 を指定すると，ワーニング・メッセージをコンソール，エラー・リスト・ファイル（-e，-se 指定時）に出力しません。

［使用例］

- -w オプションの省略により，-w1 オプションが指定されたものと見なされ，通常のワーニング・メッセージを出力します。

```
C:¥>cc78k0r -cf1166a0 prime.c
```

実行状態表示指定

実行状態表示指定オプションには，次のものがあります。

- -v/-nv

-v/-nv

[記述形式]

```
-v
-nv
```

- 省略時解釈

  -nv

[機能]

- -v オプションは，現在のコンパイルの実行状態をコンソールに出力します。
- -nv オプションは，-v オプションを無効にします。

[用途]

- コンパイルの実行状況を確認したいときに指定します。

[説明]

- フェーズ名，および処理中の関数名を出力します。
- -v と -nv の両オプションが同時に指定された場合は，後ろに指定したものが優先となります。

[使用例]

- 現在のコンパイルの実行状態をコンソールに出力します。

```
C:¥>cc78k0r -cf1166a0 prime.c -v
```

パラメータ・ファイル指定

パラメータ・ファイル指定オプションには，次のものがあります。

- -f

-f

[記述形式]

```
-f ファイル名
```

- 省略時解釈
  コマンド行上からのみオプション，入力ファイル名の入力が可能

[機能]

- -f オプションは，オプション，あるいは入力ファイル名を指定のファイルから入力することを指示します。

[用途]

- コンパイル時に複数のオプションを入力するため，コマンド行では CA78K0R の起動に必要な情報を指定しきれないときに -f オプションを指定します。
- 繰り返し同じようにオプションを指定しコンパイルする際には，それらをパラメータ・ファイルに記述しておき，-f オプションを指定します。

[説明]

- パラメータ・ファイルのネストは許されません。
- パラメータ・ファイル中に記述可能な文字数に，制限はありません。
- 空白とタブをオプション，あるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，あるいは入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションの優先順位は，後ろに指定したものが優先となります。
- “ ; ”，および“#”以降に記述された文字は，行末まですべてコメントと解釈します。

[使用例]

- パラメータ・ファイル prime.pcc の内容

```
; parameter file
prime.c -cf1166a0 -aprime.asm -e -x
```

パラメータ・ファイル prime.pcc を使用してコンパイルします。

```
C:¥>cc78k0r -fprime.pcc
```

テンポラリ・ファイル作成フォルダ指定

テンポラリ・ファイル作成フォルダ指定オプションには，次のものがあります。

- -t

-t

[記述形式]

```
-t フォルダ
```

- 省略時解釈
  環境変数 TMP で指定したドライブ・フォルダに，テンポラリ・ファイルを作成します。環境変数 TMP による指定がない場合は，カレント・ドライブ，カレント・フォルダに，テンポラリ・ファイルを作成します。

[機能]

- -t オプションは，テンポラリ・ファイルを作成するドライブ，フォルダを指示します。

[用途]

- テンポラリ・ファイルの作成場所を指定することができます。

[説明]

- 以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ，次の作成時には上書きします。
- テンポラリ・ファイルは，必要とするメモリ・サイズがある場合は，メモリに展開します。
  必要とするメモリ・サイズがなくなった場合は，メモリの内容を指定されたフォルダ下にテンポラリ・ファイルを作成してファイルに書き出し，それ以降のテンポラリ・ファイルに対するアクセスは，メモリ上でなく，ファイルに対して行います。
- テンポラリ・ファイルは，コンパイル終了時に削除されます。また，［Ctrl］＋［C］キーを押すことによってコンパイルが中止されたときも，削除されます。

[使用例]

- テンポラリ・ファイルをフォルダ tmp に作成します。

```
C:¥>cc78k0r -cf1166a0 prime.c -ttmp
```

機能拡張指定

機能拡張指定オプションには，次のものがあります。

- -z/-nz

-z/-nz

[記述形式]

```
-z 種別（複数指定可能）
-nz
```

- 省略時解釈

  -nz

[機能]

- -z オプションは，拡張機能を有効とします。
- -nz オプションは，-z オプションを無効とします。
- 種別を省略することはできません。省略した場合は，致命的エラー（F0012）となります。

[用途]

- RL78，および 78K0R の拡張機能に対し，［説明］に示す機能を持たせます。

[説明]

- -z オプションの種別指定には，次のものがあります。

種別指定	説明
p	“//”以降改行までをコメントと解釈します。
c	“/* */”コメントのネストを許します。
s 注	コメント中の漢字コードを SJIS と解釈します。
e 注	コメント中の漢字コードを EUC と解釈します。
n 注	コメント中に漢字コードがないと解釈します。
b	char/unsigned char 型引数／返り値を int 拡張しません。

種別指定	説明
a	ANSI 規定外の機能を無効とし，ANSI 規定の一部の機能を有効とします。 具体的には次の動作を行います。 - 次のものは予約語ではなくなります。 callt/sreg/bit/boolean/#asm/#endasm - トライグラフ・シーケンス（3 文字表記）が有効となります。 - コンパイラ定義マクロ __STDC__ は 1 となります。 - far ポインタの関係演算を 3 バイトで行うことで，データを 64K バイト境界の最後の 1 バイトに配置可能とします。 - int 型ビット・フィールドに対し，次のワーニングを出力します。 （CC78K0R warning W0787 : Bit field type is not int） - -qc，-zp，-zc オプションに対し，-w2 指定時，次のワーニングを出力します。 （CC78K0R warning W0029 : ' -QC ' option is not portable） （CC78K0R warning W0031 : ' -ZP ' option is not portable） （CC78K0R warning W0032 : ' -ZC ' option is not portable） - 各種 #pragma 文に対し，-w2 指定時，次のワーニングを出力します。 （CC78K0R warning W0849 : #pragma statement is not portable） - __asm 文に対し，-w2 指定時，次のワーニングを出力し，アセンブル出力は行われます。 （CC78K0R warning W0850 : Asm statement is not portable） - #asm ～ #endasm ブロックに対し，-w2 指定時，次のエラーなどを出力します。 （CC78K0R error E0801 : Undefined control など）
f	フラッシュ用オブジェクトを出力します。
t *アドレス*	フラッシュ領域分岐テーブルの先頭アドレスを指定します。
z *アドレス*	フラッシュ領域の先頭アドレスを指定します。
x	RAM 配置用オブジェクトを出力します。 - callt 関数定義に対し，次のエラーを出力します。 （CC78K0R error E0791 : ‘-ZF’ / ‘-ZX’ option specified - cannot allocate a callt function ‘関数名’） - #pragma interrupt/rtos_interrupt 指令に対し，次のエラーを出力します。 （CC78K0R error E0873 : ‘-ZX’ option specified - cannot specify #pragma interrupt /rtos_interrupt） - RAM 配置用のセルフ・プログラミング機能がないデバイスでは，-zf 未指定時の割り込み関数定義に対し，次のエラーを出力します。 （CC78K0R error E0768 : Cannot allocate interrupt function in RAM area） - 次のワーニングを出力し，__near/__far 属性やメモリ・モデルにかかわらず，すべての関数に far 属性をつけます。 (CC78K0R warning W0070 : Functions are treated as far function) - *n* の値によって，呼び出すランタイム・ライブラリが異なります。*n* を省略した場合は，*n* ＝ 2 として解釈されます。 - *n* ＝ 1 の場合，ランタイム・ライブラリは ROM 配置用のライブラリを呼ぶ - *n* ＝ 2 の場合，ランタイム・ライブラリは RAM 配置用のライブラリを呼ぶ。 - -zx2 指定時に -ql を指定した場合は次のワーニングを出力し，-ql のレベルを強制的に -ql1 にします。 （CC78K0R warning W0046 : ‘-ZX2’ option specified - regarded as ‘-QL1’）

注　s，e，n は，同時に指定することはできません。

［使用例］

- C ソース・ファイル prime.c 中の“//”以降改行までをコメントと解釈します。また，“/* */”コメントのネストを許します。

```
C:¥>cc78k0r -cf1166a0 prime.c -zpc
```

デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定オプションには，次のものがあります。

- -y

-y

[記述形式]

```
-y フォルダ
```

- 省略時解釈
  通常のサーチ・パスのみ

備考　通常のサーチ・パスは，次のとおりです。

(1) < .. ¥ .. ¥ .. ¥ dev >（cc78k0r.exe の起動されたパスに対して）

(2) cc78k0r.exe の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

[機能]

- -y オプションは，デバイス・ファイル読み込みのためのサーチ・パスとして，指定されたパスを最初に探します。存在しなければ，通常のパスから探します。

[用途]

- デバイス・ファイルを通常のサーチ・パスにはない特定のフォルダにインストールする場合，そのパスを本オプションで指定します。

[注意]

- CubeSuite+ 使用時は，プロジェクト作成時に選択したマイクロコントローラにより，フォルダが決定されます。したがって，コンパイル・オプションで指定する必要はありません。

［使用例］

- デバイス・ファイル読み込みのため，“C:￥tmp￥dev”を最初にサーチします。

```
C:￥>cc78k0r -cf1166a0 -yC:￥tmp￥dev
```

メモリ・モデル指定

メモリ・モデル指定オプションには，次のものがあります。

- -m

-m

[記述形式]

```
-m 種別
```

- 省略時解釈
  -mm

[機能]

- -m オプションは，コンパイル時のメモリ・モデルの種類を指定します。
- 種別を同時に複数指定することはできません。
- 種別を省略することはできません。省略した場合は，致命的エラー（F0012）となります。

[用途]

- メモリ・モデルを指定することにより，関数，および変数の near/far 領域を指定します。
- C ソース・ファイル上で関数，および変数に __near/__far 修飾子を記述した場合は，__near/__far 修飾子による near/far 領域指定が優先されます。

[説明]

- -m オプションの種別指定には，次のものがあります。

種別指定	メモリ・モデル	説明
s	スモール・モデル	コード部最大 64K バイト，データ部最大 64K バイト，合計して 128K バイトとみなして，near/far 領域を指定します。
m	ミディアム・モデル	コード部最大 1M バイト，データ部最大 64K バイト，合計して 1M バイトとみなして，near/far 領域を指定します。
l	ラージ・モデル	コード部最大 1M バイト，データ部最大 1M バイト，合計して 1M バイトとみなして，near/far 領域を指定します。

注意　データ部，またはコード部が最大 64K バイトのメモリ・モデルを指定しても，__far 修飾子がある関数，および変数は，最大 1M バイトの空間に配置することができます。
メモリ・モデルの指定は，__near/__far 修飾子を持たない関数，および変数の配置を指定するものです。

［使用例］

- コンパイル時のメモリ・モデルは，スモール・モデルとなります。

```
C:¥>cc78k0r -cf1166a0 prime.c -ms
```

ミラー領域指定

ミラー領域指定オプションには，次のものがあります。

- -mi

-mi

[記述形式]

```
-mi0，または -mi1
```

- 省略時解釈

  -mi0

[機能]

- -mi オプションは，ミラー元のセグメントの配置先を指定します。

[用途]

- ミラー元のセグメントの配置先を指定したいときに，-mi オプションで指定します。

[説明]

- -mi0 指定時は，MAA に 0 を設定したときのミラー領域にセグメントを配置，-mi1 指定時は，MAA に 1 を設定したときのミラー領域にセグメントを配置します。

  ミラー領域の詳細については，デバイスのユーザーズ・マニュアルを参照してください。

[使用例]

- MAA に 1 を設定したときのミラー領域にセグメントを配置します。

```
C:¥>cc78k0r prime.c -mi1
```

共通オブジェクト指定

共通オブジェクト指定オプションには，次のものがあります。

- -common

-common

[記述形式]

```
-common
```

- 省略時解釈

  指定したデバイスに対応したオブジェクトを出力します。

[機能]

- -common オプションは，RL78，および 78K0R 共通オブジェクトの出力を指定します。

[用途]

- デバイス種別指定（-c）オプションに関係なく，RL78，および 78K0R 共通で使用することができるオブジェクトを生成します。

[説明]

- RL78，および 78K0R 共通で使用することができるオブジェクトを生成したい場合に指定してください。

[使用例]

- RL78，および 78K0R 共通で使用することができるオブジェクトを生成します。

```
C:¥>cc78k0r prime.c -cf1166a0 -common
```

変数／関数情報ファイル指定

変数／関数情報ファイル指定オプションには，次のものがあります。

- -ma

-ma

[記述形式]

```
-ma ファイル名 [ -ma ファイル名 ]
-ma ファイル名 [, ファイル名 ]
```

- 省略時解釈

  変数／関数情報ファイルを使用しません。

[機能]

- -ma オプションは，使用する変数／関数情報ファイルを指定します。

[用途]

- 変数／関数情報ファイルを使用して変数／関数を効率よく配置したいときに，-ma オプションを指定します。

[説明]

- ファイル名は，2 つまで指定可能です。
- 変数／関数情報ファイルにより，C ソースの記述とは別に，変数，および関数の属性を指定することができます。

  変数／関数情報ファイルの詳細については，「B. 8　変数／関数情報ファイル生成ツール」を参照してください。
- -ma と -rd を同時に指定した場合は，-rd を無視するというワーニング・メッセージを出力します。
- -ma と -rs を同時に指定した場合は，-rs のほうが優先度が高いので，-ma による配置が抑制されます。

[使用例]

- 変数／関数情報ファイル info.vfi を使用して，変数／関数を配置します。

```
C:¥>cc78k0r prime.c -cf1166a0 -mainfo.vfi
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --/-?/-h

--/-?/-h

[記述形式]

```
--/-?/-h
```

- 省略時解釈

  表示しません。

[機能]

- --/-?/-h オプションは，オプションの簡単な説明，デフォルト・オプションなどのヘルプ・メッセージをコンソールに表示します。

注意　本オプションは，CubeSuite+ 上では指定することはできません。

[用途]

- オプションとその説明が表示されます。C コンパイラ実行時に参照してください。

[説明]

- --/-?/-h オプションを指定すると，他のコンパイル・オプションはすべて無効となります。
- ヘルプ・メッセージの続きを参照する場合は [Enter] キーを，表示を途中で終了する場合には，[Enter] キー以外の文字を入力したあとに [Enter] キーを入力してください。

[使用例]

- ヘルプ・メッセージをコンソールに表示します。

```
C:¥>cc78k0r -h
```

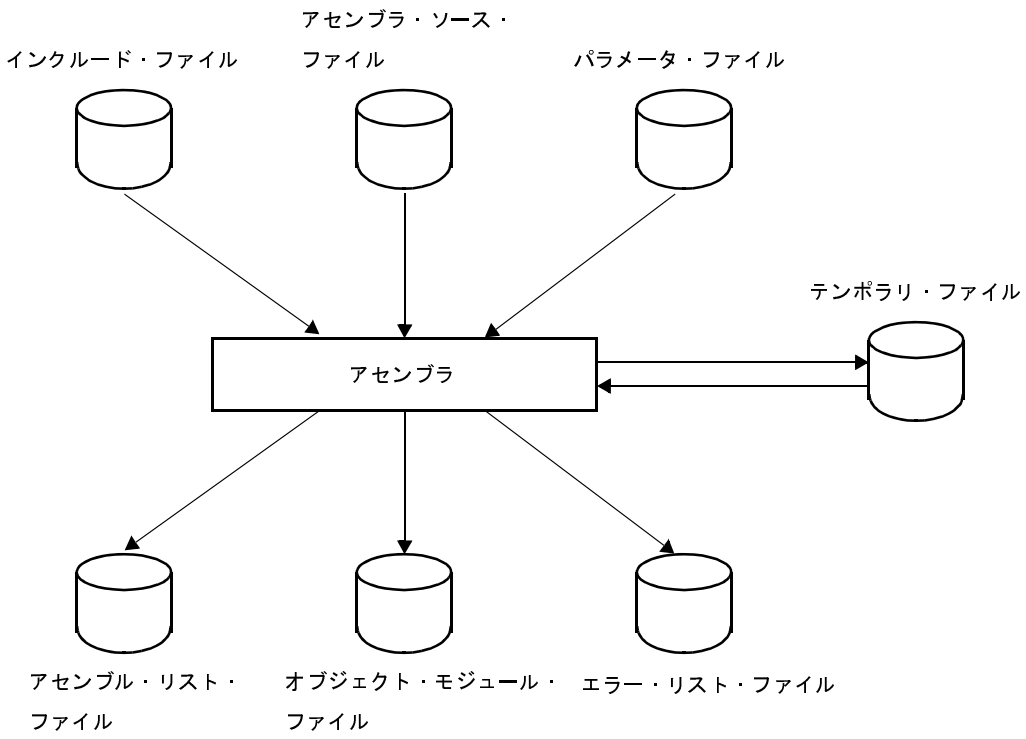
B. 2　アセンブラ

アセンブラは，RL78，および 78K0R のアセンブリ言語で記述されたソース・ファイルを入力し，それを機械語に変換してオブジェクト・モジュール・ファイルとして出力します。

さらに，アセンブル・リスト・ファイルやエラー・リスト・ファイルなどのリスト・ファイルを出力します。

アセンブル・エラーがある場合は，エラー・メッセージをアセンブル・リスト・ファイルやエラー・リスト・ファイルに出力し，エラーの原因を明示します。

図 B—3　アセンブラの入出力ファイル



B. 2. 1　入出力ファイル

アセンブラの入出力ファイルを次に示します。

出力リストについての詳細は，「3. 2　アセンブラ」を参照してください。

表 B—5　アセンブラの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	アセンブラ・ソース・ファイル	- RL78，および 78K0R 用のアセンブリ言語で記述されたソース・ファイル（ユーザ作成ファイル）	.asm
	インクルード・ファイル	- アセンブラ・ソース・ファイルで参照するファイル - RL78，および 78K0R 用のアセンブリ言語で記述されたファイル（ユーザ作成ファイル）	なし
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイル（ユーザ作成ファイル）	.pra
出力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイル	.rel
	アセンブル・リスト・ファイル	- アセンブル・リスト，クロスリファレンス・リストなどのアセンブル情報を持つファイル	.prn
	エラー・リスト・ファイル	- アセンブル時のエラー情報を持つファイル	.era
入出力ファイル	テンポラリ・ファイル	- アセンブルのためにアセンブラが自動生成するファイル アセンブル終了時には消去されます。	RA*xxxxx*.$$n （n＝1 - 4）

B. 2. 2　機　　能

（1）アセンブリ言語を機械語に変換

ソース・ファイルを読み込み，アセンブリ言語を機械語に変換します。

B. 2. 3　操作方法

（1）アセンブラの起動方法

アセンブラの起動には，2 つの方法があります。

（a）コマンド行での起動

```
X:[パス名]>ra78k0r[△オプション]... ソース・ファイル名[△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名

ra78k0r	アセンブラのコマンド名
オプション	アセンブラに対して動作の詳細を指示します。 複数のアセンブル・オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，アセンブル・オプションに大文字，小文字の区別はありません。アセンブル・オプションについての詳細は，「B. 2. 4　オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（” “）で囲んでください。
ソース・ファイル名	アセンブルするソース・ファイル名 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（” “）で囲んでください。

例　エラー・リスト・ファイル k0rmain.era を出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -e -np
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，アセンブルするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>ra78k0r[△ソース・ファイル]△-f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	アセンブラの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[△]オプション[△オプション]...
```

- コマンド行でソース・ファイル名を省略した場合は，パラメータ・ファイル内でソース・ファイル名を１つだけ指定することができます。
- ソース・ファイル名は，オプションのあとに記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定するすべてのアセンブル・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル k0rmain.pra をエディタで作成し，アセンブラを起動します。

```
; parameter file
k0rmain.asm -osample.rel
-psample.prn
```

```
C:¥>ra78k0r -fk0rmain.pra
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

アセンブラが起動すると，次の実行開始メッセージが表示されます。

```
78K0R Assembler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
```

(b) 実行終了メッセージ

アセンブルの結果，アセンブル・エラーが検出されなかった場合，アセンブラは次のメッセージを表示して制御をホスト OS に戻します。

```
PASS1 Start
PASS2 Start

 Target chip : uPD78xxx
 Device file : Vx.xx

Assembly complete, 0 error(s) and 0 warning(s) found.
```

アセンブルの結果，アセンブル・エラーが検出された場合，アセンブラはエラーとワーニングの数を表示して制御をホスト OS に戻します。

```
PASS1 Start
k0rmain.asm ( 12 ) : RA78K0R error E2201 : Syntax error
PASS2 Start
k0rmain.asm ( 12 ) : RA78K0R error E2201 : Syntax error
k0rmain.asm ( 29 ) : RA78K0R error E2407 : Undefined symbol reference 'CONVAH'
k0rmain.asm ( 29 ) : RA78K0R error E2303 : Illegal expression

 Target chip : uPD78xxx
 Device file : Vx.xx

Assembly complete, 3 error(s) and 0 warning(s) found.
```

アセンブル中にアセンブラ処理継続が不可能な致命的エラーが検出された場合，アセンブラはメッセージを表示してアセンブルを中止し，制御をホスト OS に戻します。

例 1. 存在しないソース・ファイルを指定した場合

```
C:¥>ra78k0r sample.asm
```

```
78K0R Assembler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RA78K0R error F2006 : File not found 'sample.asm'
Program aborted.
```

この例では，存在しないソース・ファイルを指定したためにエラーとなり，アセンブルが中止されます。

2. 存在しないアセンブル・オプションを指定した場合

```
C:¥>ra78k0r k0rmain.asm -z
```

```
78K0R Assembler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RA78K0R error F2012 : Missing parameter '-z'
Please enter 'RA78K0R--' , if you want help messages.
Program aborted.
```

この例では，存在しないアセンブル・オプションを指定したためにエラーとなり，アセンブルが中止されます。

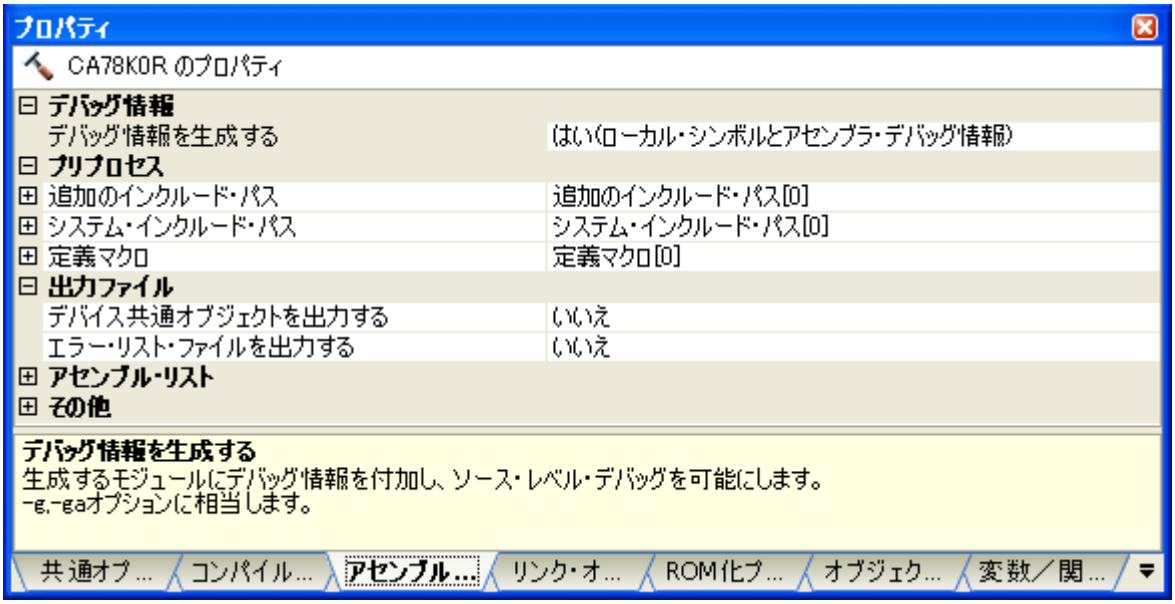
(3) CubeSuite+ でのオプション設定

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 B—4　プロパティ パネル：［アセンブル・オプション］タブ



B. 2. 4　オプション

（1）種類

アセンブル・オプションは，アセンブラの動作に細かい指示を与えるものです。

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
デバイス種別指定	-c	対象デバイスの種別を指定します。
オブジェクト・モジュール・ファイル出力指定	-o	オブジェクト・モジュール・ファイルの出力を指定します。
	-no	
オブジェクト・モジュール・ファイル強制出力指定	-j	強制的にオブジェクト・モジュール・ファイルを出力します。
	-nj	
デバッグ情報出力指定	-g	デバッグ情報（ローカル・シンボル情報）をオブジェクト・モジュール・ファイルへ出力します。
	-ng	
	-ga	アセンブラ・ソース・デバッグ情報をオブジェクト・モジュール・ファイルへ出力します。
	-nga	
インクルード・ファイル読み込みパス指定	-i	インクルード・ファイルを指定したパスから読み込みます。
アセンブル・リスト・ファイル出力指定	-p	アセンブル・リスト・ファイルの出力を指定します。
	-np	

分類	オプション	説明
アセンブル・リスト・ファイル情報指定	-ka	アセンブル・リスト・ファイル中にアセンブル・リストを出力します。
	-nka	
	-ks	アセンブル・リスト・ファイル中にシンボル・リストを出力します。
	-nks	
	-kx	アセンブル・リスト・ファイル中にクロスリファレンス・リストを出力します。
	-nkx	
アセンブル・リスト・ファイル形式指定	-lw	アセンブル・リスト・ファイルの 1 行に印字する文字数を変更します。
	-ll	アセンブル・リスト・ファイルの 1 頁に印字する行数を変更します。
	-lh	アセンブル・リスト・ファイルのヘッダに，指定された文字列を出力します。
	-lt	タブの展開文字数を変更します。
	-lf	アセンブル・リスト・ファイルの最後に，改頁コードを付加します。
	-nlf	
エラー・リスト・ファイル出力指定	-e	エラー・リスト・ファイルを出力します。
	-ne	
パラメータ・ファイル指定	-f	入力ファイル名，オプションを指定したファイルより入力します。
テンポラリ・ファイル作成パス指定	-t	テンポラリ・ファイルを指定したパスに作成します。
漢字コード指定	-zs	コメントに記述された漢字をシフト JIS コードとして解釈します。
	-ze	コメントに記述された漢字を EUC コードとして解釈します。
	-zn	コメントに記述された文字を漢字として解釈しません。
デバイス・ファイル・サーチ・パス指定	-y	デバイス・ファイルを指定されたパスから読み込みます。
シンボル定義指定	-d	シンボルの定義を行います。
共通オブジェクト指定	-common	RL78，および 78K0R 共通オブジェクト・モジュール・ファイルの出力を指定します。
78K0 互換マクロ	-compati	78K0 用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。
	-ncompati	
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

（2）優先度

次の表に示すアセンブル・オプションのうち，縦軸のものと横軸のものを同時に 2 つ以上指定した場合の優先度について説明します。

表 B—7　アセンブル・オプションの優先度

	-no	-np	-nka	-nks	-kx	-nkx	--
-j	×						×
-g	×						×
-p			△	△		△	×
-ka		×					×
-ks		×			×		×
-kx		×					×
-lw		×					×
-ll		×					×
-lh		×					×
-lt		×					×
-lf		×					×

- ×で記した箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例　-lw，-lf オプションは，無効となります。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -np -lw80 -lf
```

- △で記した箇所

横軸に示したオプション 3 つをすべて同時に指定した場合，縦軸に示したオプションは無効となります。

例　-nka，-nks，および -nkx を同時に指定したので，-p オプションは無効となります。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -p -nka -nks -nkx
```

- 空欄の箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは有効となります。

また，-o/-no オプションのように，オプション名の前に“n”を付加できるオプションを同時に指定した場合，あとで指定した方が有効となります。

例　-no オプションがあとに指定されているので，-o オプションは無効となり，-no オプションが有効となります。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -o -no
```

「表 B—7　アセンブル・オプションの優先度」に記述されていないオプションは，ほかのオプションの影響を特に受けません。しかし，ヘルプ指定オプション（ -- ）が指定された場合には，すべてのオプション指定が無効となります。

デバイス種別指定

デバイス種別指定オプションには，次のものがあります。

- -c

-c

[記述形式]

```
-c デバイス種別
```

- 省略時解釈

  省略することはできません。

[機能]

- -c オプションは，アセンブルの対象となるデバイスを指定します。

[用途]

- -c オプションは必ず指定してください。アセンブラは指定された対象デバイスに対してアセンブルを行い，それに対応したオブジェクト・コードを生成します。

[説明]

- -c オプションで指定可能な対象デバイスについては，「CubeSuite+　使用上の留意点」を参照してください。

[注意]

- -c オプションは省略できません。ただし，ソースの先頭で，-c オプションと同機能の制御命令（$PROCESSOR）を記述すれば，コマンド行での指定を省略することができます。

```
△$△PROCESSOR△(△デバイス種別△)
△$△PC△(△デバイス種別△)                ; 短縮形
```

[使用例]

- uPD78F1166_A0 を対象デバイスとして指定します。

```
C:¥>ra78k0r -cf1166a0 main.asm
```

オブジェクト・モジュール・ファイル出力指定

オブジェクト・モジュール・ファイル出力指定オプションには，次のものがあります。

- -o/-no

-o/-no

[記述形式]

```
-o [ 出力ファイル名 ]
-no
```

- 省略時解釈

  -o *入力ファイル名* .rel

[機能]

- -o オプションは，オブジェクト・モジュール・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -no オプションは，-o，-j，-g，-ga オプションを無効にします。

[用途]

- オブジェクト・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，-o オプションを指定します。

  アセンブル・リスト・ファイルの出力のみが目的でアセンブルする場合などは，-no オプションを指定します。これにより，アセンブル時間が短縮されます。

[説明]

- -o オプションを指定しても，フェイタル・エラーがある場合には，オブジェクト・モジュール・ファイルは出力されません。
- -o オブジェクトを指定する際にドライブ名を省略すると，カレント・ドライブにオブジェクト・モジュール・ファイルが出力されます。
- -o オプションを指定する際に出力ファイル名を省略すると，オブジェクト・モジュール・ファイル名は“*入力ファイル名* .rel”となります。
- -o と -no の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

［使用例］

- オブジェクト・モジュール・ファイル sample.rel を出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -osample.rel
```

オブジェクト・モジュール・ファイル強制出力指定

オブジェクト・モジュール・ファイル強制出力指定オプションには，次のものがあります。

- -j/-nj

-j/-nj

[記述形式]

```
-j
-nj
```

- 省略時解釈

  -nj

[機能]

- -j オプションは，フェイタル・エラーでもオブジェクト・モジュール・ファイルを出力するように指定します。
- -nj オプションは，-j オプションを無効にします。

[用途]

- 通常，フェイタル・エラーがある場合には，オブジェクト・モジュール・ファイルは出力されません。したがって，フェイタル・エラーがあるのを承知でプログラムを実行させたい場合には，-j オプションを指定してオブジェクト・モジュール・ファイルを出力します。

[説明]

- -j オプションを指定すると，フェイタル・エラーがある場合でもオブジェクト・モジュール・ファイルが出力されます。
- -j と -nj の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -no オプションを指定した場合は，-j オプションは無効となります。

[使用例]

- フェイタル・エラーの場合でもオブジェクト・モジュール・ファイル k0rmain.rel を出力するよう指定します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -j
```

デバッグ情報出力指定

デバッグ情報出力指定オプションには，次のものがあります。

- -g/-ng
- -ga/-nga

-g/-ng

[記述形式]

```
-g
-ng
```

- 省略時解釈

  -g

[機能]

- -g オプションは，オブジェクト・モジュール・ファイル中にデバッグ情報（ローカル・シンボル情報）を付加するよう指示します。
- -ng オプションは，-g オプションを無効にします。

[用途]

- -g オプションは，ローカル・シンボルも含めてシンボリック・デバッグを行うときに使用します。
- -ng オプションは，次の 3 種類の場合に使用します。

(1) グローバル・シンボルのみのシンボリック・デバッグ

(2) シンボルなしでのデバッグ

(3) オブジェクトのみを必要とするとき（PROM による評価時など）

[説明]

- -g と -ng の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -g/-ng オプションと -ga/-nga オプションを同時に指定した場合は，指定した位置にかかわらず -ga/-nga オプションが有効となります。
- -no オプションを指定した場合は，-g オプションは無効となります。

[注意]

- ソースの先頭で，-g/-ng オプションと同機能の制御命令（DEBUG/NODEBUG，または DG/NODG）を記述することができます。

  記述形式を次に示します。

```
△$△DEBUG
△$△DG          ; 短縮形
△$△NODEBUG
△$△NODG        ; 短縮形
```

[使用例]

- オブジェクト・モジュール・ファイル k0rmain.rel にデバッグ情報（ローカル・シンボル情報）を付加します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -g
```

-ga/-nga

[記述形式]

```
-ga
-nga
```

- 省略時解釈
  -ga

[機能]

- -ga オプションは，オブジェクト・モジュール・ファイル中にアセンブラ・ソース・デバッグ情報を付加するよう指示します。
- -nga オプションは，-g，-ga オプションを無効にします。

[用途]

- -ga オプションは，アセンブラのソース・レベルでデバッグするときに使用します。ソース・レベルでのデバッグには「統合デバッガ（別売）」が必要です。
- -nga オプションは，次の 3 種類の場合に使用します。

(1) アセンブラ・ソースなしでのデバッグ

(2) オブジェクトのみを必要とするとき（PROM による評価時など）

(3) C コンパイラ・ソース・レベルでのデバッグ

[説明]

- -ga と -nga の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -g/-ng オプションと -ga/-nga オプションを同時に指定した場合は，指定した位置にかかわらず -ga/-nga オプションが有効となります。
- -no オプションを指定した場合は，-ga オプションは無効となります。

[注意]

- ソースの先頭で，-ga，-nga オプションと同機能の制御命令（DEBUGA/NODEBUGA）を記述することができます。

  記述形式を次に示します。

```
△ $ △ DEBUGA
△ $ △ NODEBUGA
```

[使用例]

- オブジェクト・モジュール・ファイル k0rmain.rel にアセンブラ・ソース・デバッグ情報を付加します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -ga
```

インクルード・ファイル読み込みパス指定

インクルード・ファイル読み込みパス指定オプションには，次のものがあります。

- -i

-i

[記述形式]

```
-i パス名 [, パス名] …（複数指定可能）
```

- 省略時解釈

  インクルード・ファイルを次の順序で検索します。

（1）ソース・ファイルのあるパス

（2）環境変数 INC78K0R で指定したパス

[機能]

- -i オプションは，ソース中の“$include”で指定されたインクルード・ファイルを指定したパスから入力するよう指示します。

[用途]

- インクルード・ファイルを，あるパスから検索したいときに指定します。

[説明]

- “ , ”で区切ることにより，一度に複数のパス名を指定することができます。
- “ , ”の前後には，空白を入れることはできません。
- “$include”で指定したインクルード・ファイルの検索順序は，次のようになります。

（1）-i オプションに続いてパス名が複数指定された場合は，指定された順番でインクルード・ファイルを検索します。

（2）-i オプションが複数指定された場合は，後者の指定を優先する順番でインクルード・ファイルを検索します。

（3）-i オプションで指定したパスの検索後に，省略時解釈と同じ順序でインクルード・ファイルを検索します。

- -i に続けてパス名以外のものを指定した場合やパス名を省略した場合は，アボート・エラーとなります。
- -i を 65 個以上指定した場合は，アボート・エラーとなります。

[使用例]

- インクルード・ファイルをフォルダ C:￥sample1，C:￥sample2 の順で検索し，読み込みます。

```
C:￥>ra78k0r -cf1166a0 k0rmain.asm -iC:￥sample1,C:￥sample2
```

- インクルード・ファイルをフォルダ D:￥include files から読み込みます。

```
C:￥>ra78k0r -cf1166a0 k0rmain.asm -i"D:￥include files"
```

アセンブル・リスト・ファイル出力指定

アセンブル・リスト・ファイル出力指定オプションには，次のものがあります。

- -p/-np

-p/-np

[記述形式]

```
-p [ 出力ファイル名 ]
-np
```

- 省略時解釈

  -p *入力ファイル名* .prn

[機能]

- -p オプションは，アセンブル・リスト・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -np オプションは，-p，-ka，-ks，-kx，-lw，-ll，-lh，-lt，-lf のオプションを無効にします。

[用途]

- アセンブル・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-p オプションを指定します。
- オブジェクト・モジュール・ファイルの出力のみが目的でアセンブルする場合などに，-np オプションを指定します。これにより，アセンブル時間が短縮されます。

[説明]

- -p オプションを指定する際に出力ファイル名を省略すると，アセンブル・リスト・ファイル名は“*入力ファイル名* .prn”になります。
- -p オプションを指定する際にドライブ名を省略すると，カレント・ドライブにアセンブル・リスト・ファイルが出力されます。
- -p と -np の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- アセンブル・リスト・ファイル sample.prn を出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -psample.prn
```

アセンブル・リスト・ファイル情報指定

アセンブル・リスト・ファイル情報指定オプションには，次のものがあります。

- -ka/-nka
- -ks/-nks
- -kx/-nkx

-ka/-nka

[記述形式]

```
-ka
-nka
```

- 省略時解釈

  -ka

[機能]

- -ka オプションは，アセンブル・リスト・ファイル中にアセンブル・リストを出力します。
- -nka オプションは，-ka オプションを無効にします。

[用途]

- アセンブル・リストを出力したいときに，-ka オプションを指定します。

[説明]

- -ka と -nka の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -nka，-nks，および -nkx オプションをすべて指定した場合は，アセンブル・リスト・ファイルは出力されません。
- -np オプションを指定した場合は，-ka オプションは無効となります。

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn 中にアセンブル・リストを出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -ka
```

k0rmain.prn の内容は，以下のようになります。

```
      Assemble list

ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT
   1     1
   2     2                             NAME    SAMPM
   3     3                      ;****************************************
   4     4                      ;
   5     5                      ;     HEX -> ASCII Conversion Program
   6     6                      ;
   7     7                      ;              main-routine
   8     8                      ;
   9     9                      ;****************************************
  10    10
  11    11                      PUBLIC  MAIN , START
  12    12                      EXTRN   CONVAH
  13    13                      EXTRN   _@STBEG
  14    14
  15    15 -----                DATA    DSEG    AT 0FFE20H
  16    16 FFE20                HDTSA:  DS      1
  17    17 FFE21                STASC:  DS      2
  18    18
  19    19 -----                CODE    CSEG    AT 0H
  20    20 00000 R0000          MAIN:   DW      START
  21    21
  22    22 -----                        CSEG
  23    23 00000                START :
  24    24
  25    25                              ; chip initialize
  26    26 00000 RCBF80000              MOVW    SP , #_@STBEG
  27    27
  28    28 00004  CD201A                MOV     HDTSA , #1AH
  29    29 00007  3620FE                MOVW    HL , #LOWW ( HDTSA )  ; set hex 2-code
data in HL registor
        :
```

-ks/-nks

[記述形式]

```
-ks
-nks
```

- 省略時解釈
  -nks

[機能]

- -ks オプションは，アセンブル・リストに続いてシンボル・リストをアセンブル・リスト・ファイル中に出力します。
- -nks オプションは，-ks オプションを無効にします。

[用途]

- シンボル・リストを出力したいときに，-ks オプションを指定します。

[説明]

- -nka，-nks，および -nkx オプションがすべて指定された場合は，アセンブル・リスト・ファイルは出力されません。
- -ks と -kx を同時に指定した場合は，-ks は無視されます。
- -ks と -nks の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-ks オプションは無効となります。

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn 中に，アセンブル・リストに続いてシンボル・リストを出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -ks
```

k0rmain.prn の内容は，以下のようになります。

```
          Symbol Table List

VALUE  ATTR  RTYP NAME

       CSEG       ?CSEG
-----H       EXT  CONVAH
FFE20H ADDR       HDTSA
       MOD        SAMPM
FFE21H ADDR       STASC
```

```
VALUE  ATTR  RTYP NAME

       CSEG            CODE
       DSEG            DATA
    0H ADDR  PUB       MAIN
    0H ADDR  PUB       START
-----H       EXT       _@STBEG
```

-kx/-nkx

[記述形式]

```
-kx
-nkx
```

- 省略時解釈
  -nkx

[機能]

- -kx オプションは，アセンブル・リストに続いてクロスリファレンス・リストをアセンブル・リスト・ファイル中に出力します。
- -nkx オプションは，-kx オプションを無効にします。

[用途]

- ソース・ファイルで定義された各シンボルが，ソース中のどこでどれだけ参照されているか，また，アセンブル・リストの何行目の記述でそのシンボルを参照しているのかなどの情報を知りたいとき，クロスリファレンス・リストを出力します。

[説明]

- -nka，-nks，および -nkx オプションをすべて指定した場合，アセンブル・リスト・ファイルは出力されません。
- -ks と -kx を同時に指定した場合，-ks は無視されます。
- -kx と -nkx の両オプションを同時に指定した場合，あとで指定した方が有効となります。
- -np オプションを指定した場合，-kx オプションは無効となります。

[注意]

- ソースの先頭で，-kx/-nkx オプションと同機能の制御命令（XREF/NOXREF，または XR/NOXR）を記述することができます。
  記述形式を次に示します。

```
△$△XREF
△$△XR          ; 短縮形
△$△NOXREF
△$△NOXR        ; 短縮形
```

［使用例］

- アセンブル・リスト・ファイル k0rmain.prn 中に，アセンブル・リストに続いてクロスリファレンス・リストを出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -kx
```

k0rmain.prn の内容は，以下のようになります。

```
        Cross-Reference List


NAME     VALUE    R ATTR  RTYP SEGNAME XREFS


?CSEG               CSEG       ?CSEG      22#
CODE                CSEG       CODE       19#
CONVAH   -----H   E       EXT             12@    31
DATA                DSEG       DATA       15#
HDTSA    FFE20H     ADDR       DATA       16#    28     29
MAIN         0H     ADDR  PUB  CODE       11@    20#
SAMPM               MOD                    2#
START        0H   R ADDR  PUB  ?CSEG      11@    20     23#
STASC    FFE21H     ADDR       DATA       17#    33
_@STBEG  -----H   E       EXT             13@    26
```

アセンブル・リスト・ファイル形式指定

アセンブル・リスト・ファイル形式指定オプションには，次のものがあります。

- -lw
- -ll
- -lh
- -lt
- -lf/-nlf

-lw

[記述形式]

```
-lw[文字数]
```

- 省略時解釈
  -lw132（ディスプレイ出力の場合は 80 文字とします）

[機能]

- -lw オプションは，リスト・ファイルの 1 行の文字数を指定します。

[用途]

- 各種リスト・ファイルの 1 行の文字数を変更したいとき，-lw オプションを指定します。

[説明]

- -lw オプションで指定可能な文字数の範囲（ディスプレイ出力の場合は 80 文字まで）は，72 ～ 2046 です。
  範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。
- 文字数を省略した場合は，132 を指定したものとみなされます。
  ただし，アセンブル・リスト・ファイルの出力先がディスプレイの場合は，80 となります。
- 指定する文字数には，ターミネータ（CR，LF）は含みません。
- -np オプションを指定した場合，-lw オプションは無効となります。

[注意]

- ソースの先頭で，-lw オプションと同機能の制御命令（WIDTH）を記述することができます。
  記述形式を次に示します。

```
△$△WIDTH
```

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn の 1 行の文字数を 80 文字に指定します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -lw80
```

アセンブル・リスト・ファイル k0rmain.prn の内容は，以下のようになります。

```
      Assemble list

ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT

   1     1
   2     2                             NAME    SAMPM
   3     3                      ;****************************************
   4     4                      ;
   5     5                      ;     HEX -> ASCII Conversion Program
   6     6                      ;
   7     7                      ;              main-routine
   8     8                      ;
   9     9                      ;****************************************
  10    10
  11    11                      PUBLIC  MAIN , START
  12    12                      EXTRN   CONVAH
  13    13                      EXTRN   _@STBEG
  14    14
  15    15 -----                DATA    DSEG    AT 0FFE20H
  16    16 FFE20                HDTSA:  DS      1
  17    17 FFE21                STASC:  DS      2
  18    18
  19    19 -----                CODE    CSEG    AT 0H
  20    20 00000 R0000          MAIN:   DW      START
  21    21
  22    22 -----                        CSEG
  23    23 00000                START:
  24    24
  25    25                              ; chip initialize
  26    26 00000 RCBF80000      MOVW    SP , #_@STBEG
  27    27
  28    28 00004  CD201A        MOV     HDTSA , #1AH
  29    29 00007  3620FE        MOVW    HL , #LOWW ( HDTSA ) ; set hex 2-code data in
HL registor
        :
```

-ll

[記述形式]

```
-ll[行数]
```

- 省略時解釈

  -ll0（改頁しません）

[機能]

- -ll オプションは，アセンブル・リスト・ファイルの 1 頁の行数を指定します。

[用途]

- アセンブル・リスト・ファイルの 1 頁の行数を変更したいとき，-ll オプションを指定します。

[説明]

- -ll オプションで指定可能な行数の範囲は，20 ～ 32767 です。
- 範囲外の数値や数値以外のものが指定した場合，アボート・エラーとなります。
- 行数を省略した場合は，0 を指定したものとみなされます。
- 行数の 0 を指定した場合は，改頁されません。
- -np オプションを指定した場合は，-ll オプションは無効となります。

[注意]

- ソースの先頭で，-ll オプションと同機能の制御命令（LENGTH）を記述することができます。

  記述形式を次に示します。

```
△$△LENGTH
```

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn の 1 頁の行数を 20 行に指定します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -ll20
```

k0rmain.prn の内容は，以下のようになります。

```
78K0R Assembler Vx.xx                                    Date:xx xxx xxxx  Page:  1

Command:  -cf1166a0 k0rmain.asm -ll20
Para-file:
In-file:  k0rmain.asm
Obj-file: k0rmain.rel
Prn-file: k0rmain.prn

        Assemble list

--------------------------------------------------------------------------------



78K0R Assembler Vx.xx                                    Date:xx xxx xxxx  Page:  2

ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT

   1     1
   2     2                             NAME    SAMPM
   3     3                     ;****************************************
   4     4                     ;
   5     5                     ;     HEX -> ASCII Conversion Program
   6     6                     ;
   7     7                     ;              main-routine
   8     8                     ;

--------------------------------------------------------------------------------



78K0R Assembler Vx.xx                                    Date:xx xxx xxxx  Page:  3



ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT

   9     9                     ;****************************************
  10    10
  11    11                     PUBLIC  MAIN , START
  12    12                     EXTRN   CONVAH
  13    13                     EXTRN   _@STBEG
  14    14
  15    15 -----               DATA    DSEG    AT 0FFE20H
  16    16 FFE20               HDTSA:  DS      1
        :
```

-lh

[記述形式]

```
-lh 文字列
```

- 省略時解釈

  なし

[機能]

- -lh オプションは，アセンブル・リスト・ファイルのヘッダのタイトル欄に印字する文字列を指定します。

[用途]

- アセンブル・リスト・ファイルの内容を端的に表すようなタイトルを表示したいときに，-lh オプションを指定します。
- タイトルを各頁に印字することで，アセンブル・リスト・ファイルの内容がひと目でわかります。

[説明]

- 指定可能な文字列は，60 文字以内です。ただし，文字列内に空白を記述することはできません。
- 61 文字以上記述した場合には，先頭の 60 文字が有効となり，エラーは出力されません。

  なお，漢字，ひらがなは，1 文字を 2 文字として計算されます。

  1 行の最大文字数が 119 以下の場合，タイトルとしての文字列の有効長は次のとおりです。

  有効長 =（1 行の最大文字数）- 60
- 文字列を指定しなかった場合は，アボート・エラーとなります。
- -np オプションを指定した場合は，-lh オプションは無効となります。
- -lh オプションを省略した場合は，アセンブル・リスト・ファイルのタイトル欄は空白となります。
- 記述可能な文字セットを，次に示します。

文字	コマンド行	パラメータ・ファイル中
* ? > < |	” ” でくくることで記述可能	記述可能 ” ” でくくっても，コマンド行と同じように解釈されます
;	” ” でくくることで記述可能	記述不可 （コメントとみなされます）
#	記述可能	記述不可 （コメントとみなされます）
”（ダブル・クォーテーション）	有効文字としては記述不可	有効文字としては記述不可

文字	コマンド行	パラメータ・ファイル中
00H	記述不可	記述可能 ただし，文字列が切れたとみなされます
03H，06H，08H，0DH，0EH，10H，15H，17H，18H，1BH，7FH	記述不可	記述可能 ただし，アセンブル・リスト・ファイル中では“！”で表示 （単独の 0DH はリストには出力されません）
01H，02H，04H，05H，07H，0BH，0CH，0FH，11H，12H，13H，14H，16H，19H，1CH，1DH，1EH，1FH	記述可能 ただしアセンブル・リスト・ファイルでは“！”で表示	記述可能 ただし，アセンブル・リスト・ファイル中では“！”で表示
1AH	記述可能 ただしアセンブル・リスト・ファイルでは“！”で表示	記述不可 （ファイルの終わり）
英字	大／小文字がそのまま入力されます	大／小文字がそのまま入力されます
その他	記述可能	記述可能

備考　起動行上の * は，ワイルド・カード展開の対象とならない場合は” ”でくくらなくても記述可能です。

[注意]

- ソースの先頭で，-lh オプションと同機能の制御命令（TITLE，または TT）を記述することができます。
  記述形式を次に示します。

```
△$△TITLE△(△'文字列'△)
△$△TT△(△'文字列'△)          ; 短縮形
```

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn のヘッダにタイトル“RA78K0R_MAINROUTINE”を印字します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
```

k0rmain.prn の内容は，以下のようになります。

```
78K0R Assembler Vx.xx   RA78K0R_MAINROUTINE     Date:xx xxx xx  Page:1
                                ↑
                              タイトル

Command:  -cf1166a0 k0rmain.asm -lhRA78K0R_MAINROUTINE
Para-file:
In-file:  k0rmain.asm
Obj-file: k0rmain.rel
Prn-file: k0rmain.prn
```

```
        Assemble list

ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT
   1     1
   2     2                              NAME    SAMPM
   3     3                      ;****************************************
   4     4                      ;
   5     5                      ;     HEX -> ASCII Conversion Program
   6     6                      ;
   7     7                      ;              main-routine
        :
```

-lt

[記述形式]

```
-lt[文字数]
```

- 省略時解釈
  -lt8

[機能]

- -lt オプションは，ソース中の HT（Horizontal Tabulation）コードを，各種リスト上でいくつかのブランク（空白）に置き換えて出力する（タビュレーション処理）ための，基本となる文字数を指定します。

[用途]

- -lw オプションで，各種リストの 1 行の文字数を少なく指定した場合に，HT コードによるブランクを少なくし，文字数を節約するために，-lt オプションを指定します。

[説明]

- -lt オプションで指定可能な文字数の範囲は，0 ～ 8 です。
- 範囲外の数値や数値以外のものを指定した場合は，アボート・エラーとなります。
- 文字数を省略した場合は，8 を指定したものとみなされます。
- -lt0 を指定した場合は，タビュレーション処理は行わず，タブ・コードが出力されます。
- -np オプションを指定した場合は，-lt オプションは無効となります。

[注意]

- ソースの先頭で，-lt オプションと同機能の制御命令（TAB）を記述することができます。
  記述形式を次に示します。

```
△$△TAB△タブ数
```

[使用例]

- -lt オプションを省略した場合のアセンブル・リスト・ファイル sample.prn を参照します。

```
C:¥>ra78k0r -cf1166a0 sample.asm
```

sample.prn の内容は，以下のようになります。

```
      Assemble list

ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT

   1     1
   2     2                              NAME    SAMPM
   3     3                      ;****************************************
   4     4                      ;
   5     5                      ;     HEX -> ASCII Conversion Program
   6     6                      ;
   7     7                      ;              main-routine
   8     8                      ;
   9     9                      ;****************************************
  10    10
  11    11                      PUBLIC  MAIN , START
  12    12                      EXTRN   CONVAH
  13    13                      EXTRN   _@STBEG
  14    14
  15    15    -----             DATA    DSEG    AT 0FFE20H
  16    16    FFE20             HDTSA:  DS      1
  17    17    FFE21             STASC:  DS      2
  18    18
  19    19    -----             CODE    CSEG    AT 0H
  20    20    00000 R0000       MAIN:   DW      START
        :
```

- HT コードによるブランクを 1 に指定します。

```
C:¥>ra78k0r -cf1166a0 sample.asm -lt1
```

sample.prn の内容は，以下のようになります。

```
       Assemble list

ALNO  STNO ADRS   OBJECT   M I  SOURCE STATEMENT

   1     1
   2     2                              NAME    SAMPM
   3     3                      ; ****************************************
   4     4                      ;
   5     5                      ;       HEX -> ASCII Conversion Program
   6     6                      ;
   7     7                      ;             main-routine
   8     8                      ;
   9     9                      ; ****************************************
```

```
10    10
11    11                    PUBLIC  MAIN , START
12    12                    EXTRN   CONVAH
13    13                    EXTRN   _@STBEG
14    14
15    15 -----              DATA    DSEG    AT 0FFE20H
16    16 FFE20              HDTSA:  DS      1
17    17 FFE21              STASC:  DS      2
18    18
19    19 -----              CODE    CSEG    AT 0H
20    20 00000 R0000        MAIN:   DW      START
      :
```

備考　HT コードによるブランクは 1 つです。

-lf/-nlf

[記述形式]

```
-lf
-nlf
```

- 省略時解釈
  -nlf

[機能]

- -lf オプションは，アセンブル・リスト・ファイルの最後に，改頁コード（FF）の付加を指定します。
- -nlf オプションは，-lf オプションを無効にします。

[用途]

- アセンブル・リスト・ファイルの内容を印字したあとで改頁しておきたい場合に，-lf オプションを指定して改頁を付加します。

[説明]

- -np オプションを指定した場合は，-lf オプションは無効となります。
- -lf と -nlf の両オプションを同時に指定した場合は，あとで指定した方が優先されます。

[注意]

- ソースの先頭で，-lf/-nlf オプションと同機能の制御命令（FORMFEED/NOFORMFEED）を記述できます。
  記述形式を次に示します。

```
△ $ △ FORMFEED
△ $ △ NOFORMFEED
```

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn の最後に，改頁コードを付加します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -p -lf
```

エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定オプションには，次のものがあります。

- -e/-ne

-e/-ne

[記述形式]

```
-e[出力ファイル名]
-ne
```

- 省略時解釈

  -ne

[機能]

- -e オプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -ne オプションは，-e オプションを無効にします。

[用途]

- エラー・メッセージをファイルに保存しておきたい場合，-e オプションを指定します。
- エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-e オプションで指定します。

[説明]

- -e オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は“入力ファイル名.era”となります。
- -e オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- -e と -ne の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- エラー・リスト・ファイル k0rmain.era を出力します。

```
C:¥>ra78k0r -cf1166a0 k0rmain.asm -ek0rmain.era
```

```
78K0R Assembler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
PASS1 Start
k0rmain.asm(31) : RA78K0 error E2202: lllegal operand
PASS2 Start
k0rmain.asm(26) : RA78K0 error E2312: Operand out of range ( byte )
k0rmain.asm(31) : RA78K0 error E2202: lllegal operand

 Target chip : uPD78F1166_A0
 Device file : Vx.xx

Assembly complete, 3 error(s) and 0 warning(s) found.
```

k0rmain.era の内容は，以下のようになります。

```
PASS1 Start
k0rmain.asm(31) : RA78K0R error E2202: lllegal operand
PASS2 Start
k0rmain.asm(26) : RA78K0R error E2312: Operand out of range ( byte )
k0rmain.asm(31) : RA78K0R error E2202: lllegal operand
```

パラメータ・ファイル指定

パラメータ・ファイル指定オプションには，次のものがあります。

- -f

-f

[記述形式]

```
-f ファイル名
```

- 省略時解釈
  コマンド行上からのみオプション，入力ファイル名の入力が可能となります。

[機能]

- -f オプションは，オプション，あるいは入力ファイル名を指定のファイルから入力する指定を行います。

[用途]

- コマンド行では，アセンブラの起動に必要な情報を指定しきれないときに，-f オプションを指定します。
- アセンブルするたびに繰り返し同じようにオプションを指定する際には，それらをパラメータ・ファイルに記述しておき，-f オプションで指定します。

[説明]

- ファイル名を省略すると，アボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で -f オプションを指定すると，アボート・エラーとなります。
- パラメータ・ファイル中に記述可能な文字数に，制限はありません。
- 空白とタブ，および改行文字（LF）をオプション，あるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，あるいは入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定したものが優先されます。
- “；”，または“＃”以降に記述された文字は，改行文字（LF），または EOF の前まですべてコメントと解釈されます。
- -f オプションを複数指定すると，アボート・エラーとなります。

［使用例］

- パラメータ・ファイルを使用してアセンブルします。

パラメータ・ファイル k0rmain.pra の内容は，次のように設定します。

```
; parameter file

k0rmain.asm -osample.rel -g -cf1166a0
-psample.prn
```

コマンド行には，次のように入力します。

```
C:¥>ra78k0r -fk0rmain.pra
```

テンポラリ・ファイル作成パス指定

テンポラリ・ファイル作成パス指定オプションには，次のものがあります。

- -t

-t

[記述形式]

```
-t パス名
```

- 省略時解釈
  環境変数 TMP で指定したパス
  環境変数 TMP を指定していない場合は，カレント・パス

[機能]

- -t オプションは，テンポラリ・ファイルを作成するパスを指定します。

[用途]

- テンポラリ・ファイルの作成場所を指定することができます。

[説明]

- パス名として，パス以外のものは指定できません。
- パス名を省略することはできません。
- 以前に作成したテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ上書きされます。
- 必要とするメモリ・サイズがある間は，テンポラリ・ファイルはメモリに展開されます。
  メモリが足りなくなった時点で，メモリに展開されていたテンポラリ・ファイルの内容がディスクに書き出されます。
  以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行われます。
- テンポラリ・ファイルは，アセンブル終了時に削除されます。また，キー入力（[Ctrl] ＋ [C] キー）によってアセンブルが中止されたときにも削除されます。
- テンポラリ・ファイルの作成パスは，次の順番で決定されます。

(1) -t オプションで指定されたパス

(2) 環境変数 TMP で設定したパス（-t オプション省略の場合）

（3）カレント・パス（TMP を設定していない場合）

注意　（1），または（2）を指定した場合，指定したパスにテンポラリ・ファイルが作成できなければ，アボート・エラーとなります。

［使用例］

- テンポラリ・ファイルをフォルダ C:￥tmp に出力します。

```
C:￥>ra78k0r -cf1166a0 k0rmain.asm -tC:￥tmp
```

- テンポラリ・ファイルをフォルダ D:￥temporary files に出力します。

```
C:￥>ra78k0r -cf1166a0 k0rmain.asm -t"D:￥temporary files"
```

漢字コード指定

漢字コード指定オプションには，次のものがあります。

- -zs/-ze/-zn

-zs/-ze/-zn

[記述形式]

```
-zs
-ze
-zn
```

- 省略時解釈

  -zs

[機能]

- コメントに記述された漢字を，指定された漢字コードとして解釈します。
- オプションにより，漢字コードを次のように解釈します。

  -zs ： シフト JIS コード

  -ze ： EUC コード

  -zn ： 漢字として解釈しません。

[用途]

- コメント行の漢字コードの解釈を指定するときに使用します。

[説明]

- -zs/-ze/-zn オプションを同時に指定した場合は，あとで指定した方が有効となります。
- ソースの先頭で，-zs/-ze/-zn オプションと同機能の制御命令（KANJICODE）を記述することができます。

  記述形式を次に示します。

```
△$△KANJICODE△SJIS
△$△KANJICODE△EUC
△$△KANJICODE△NONE
```

- 環境変数（LANG78K）でも漢字コードを指定することができます。

［使用例］

- 漢字コードを EUC コードとして解釈します。

```
C:¥>ra78k0r k0rmain.asm -cf1166a0 -ze
```

デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定オプションには，次のものがあります。

- -y

-y

[記述形式]

```
-y パス名
```

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) ra78k0r.exe の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

[機能]

- -y オプションは，デバイス・ファイルを指定されたパスから読み込みます。

[用途]

- デバイス・ファイルが存在するパスを指定します。

[説明]

- -y オプションに続けてパス名以外を指定した場合，アボート・エラーとなります。
- -y オプションに続けて指定するパス名を省略した場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べて決定します。

(1) -y オプションで指定されたパス

(2) デバイス・ファイル・インストーラで登録されたパス

（3）RA78K0R の起動されたパス

（4）カレント・フォルダ

（5）環境変数 PATH

［使用例］

- デバイス・ファイルのパスをフォルダ C:￥78k0r￥dev に指定します。

```
C:￥>ra78k0r k0rmain.asm -cf1166a0 -yC:￥78k0r￥dev
```

- デバイス・ファイルのパスをフォルダ D:￥device files に指定します。

```
C:￥>ra78k0r k0rmain.asm -cf1166a0 -y"D:￥device files"
```

シンボル定義指定

シンボル定義指定オプションには，次のものがあります。

- -d

-d

[記述形式]

```
-d シンボル名 [= 数値 ] [, シンボル名 [= 数値 ] … ]
```

- 省略時解釈

  なし

[機能]

- -d オプションは，シンボルの定義を行います。

[用途]

- シンボルの定義を行いたい場合，-d オプションを指定します。

[説明]

- シンボルに与える数値は，2 進数，8 進数，10 進数，および 16 進数とします。数値指定を省略した場合は，1 が指定されたものとします。
- シンボルは，カンマで区切ることにより 30 個まで指定することができます。
- シンボル名は，31 文字まで記述することができます。
- 同名のシンボル名が重複した場合，あとで指定した方が有効となります。
- シンボル名の英字は，大文字，小文字を区別します。

  -d で定義したシンボルは，EQU/$SET/$RESET の代わりとなります。-d に指定したシンボル名がソースでも定義されていた場合，エラーとなります。

[使用例]

- シンボルの定義を 2 と指定します。

```
C:¥>ra78k0r k0rmain.asm -cf1166a0 -dSYM=2
```

共通オブジェクト指定

共通オブジェクト指定オプションには，次のものがあります。

- -common

-common

[記述形式]

```
-common
```

- 省略時解釈

  指定したデバイスに対応したオブジェクト・ファイルを出力します。

[機能]

- -common オプションは，RL78，および 78K0R 共通オブジェクト・モジュール・ファイルの出力を指定します。

[用途]

- デバイス種別指定（-c）オプションに関係なく，RL78，および 78K0R 共通で使用することができるオブジェクト・コードを生成します。

  RL78，および 78K0R の異なるデバイスを指定されたオブジェクト・ファイルとのリンクが可能になります。

[説明]

- RL78，および 78K0R 共通で使用することができるオブジェクト・コードを生成したい場合に指定してください。

[注意]

- -common オプションを指定した場合でも，デバイス種別指定（-c）オプション，または同機能の制御命令を省略することはできません。

  リンクの際に，入力したオブジェクト・モジュール・ファイルすべてに対して共通オブジェクト指定（-common）オプションが指定されているとエラーになります。

[使用例]

- RL78，および 78K0R 共通で使用することができるオブジェクト・コードを生成します。

```
C:¥>ra78k0r k0rsub.c -cf1166a0 -common
```

78K0 互換マクロ

78K0 互換マクロオプションには，次のものがあります。

- -compati/-ncompati

-compati/-ncompati

[記述形式]

```
-compati
-ncompati
```

- 省略時解釈
  -ncompati

[機能]

- -compati オプションは，78K0 用アセンブラで作成したアセンブラ・ソース・ファイルをアセンブル可能にします。
- -ncompati オプションは，-compati オプションを無効にします。

[用途]

- 通常，RL78，および 78K0R で使用できない 78K0 の命令が記述されたアセンブラ・ソース・ファイルをアセンブルした場合，フェイタル・エラー（E2337）となります。
  下記の RL78，および 78K0R で使用できない 78K0 の命令をソースの記述を変更せずにアセンブルしたい場合，-compati オプションを指定します。
  DIVUW/ROR4/ROL4/ADJBA/ADJBS/CALLF/DBNZ

[説明]

- -compati オプションを指定すると，「..￥inc78k0r￥compati.inc」（ra78k0r.exe の起動されたパスに対して）をインクルードして，RL78，および 78K0R で使用することができない 78K0 の命令をマクロ変換します。

[使用例]

- RL78，および 78K0R で削除された 78K0 の命令をマクロ変換します。

```
C:￥>ra78k0r k0rsub.asm -cf1166a0 -compati
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --

--

[記述形式]

```
--
```

- 省略時解釈

  表示しません。

[機能]

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

[用途]

- ヘルプ・メッセージは，アセンブル・オプションとその説明の一覧です。アセンブラを実行するときに参照してください。

[説明]

- -- オプションを指定すると，ほかのアセンブル・オプションは，すべて無効となります。
- ヘルプ・メッセージの続きを読む場合は，リターン・キーを押下してください。

  表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを押下してください。

注意　本オプションは，CubeSuite+ 上では指定することはできません。

[使用例]

- -- オプションを指定すると，ヘルプ・メッセージがディスプレイに出力されます。

```
C:¥>ra78k0r --
```

```
78K0R Assembler Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

usage : ra78k0r [option[...]] input-file [option[...]]
The option is as follows ([] means omissible).
-cx          :Select target chip. (x = f1166a0 etc.) * Must be specified.
-o[file]/-no :Create the object module file [with the specified name] / Not.
-e[file]/-ne :Create the error list file [with the specified name] / Not.
-p[file]/-np :Create the print file [with the specified name] / Not.
-ka/-nka     :Output the assemble list to print file / Not.
-ks/-nks     :Output the symbol table list to print file / Not.
-kx/-nkx     :Output the cross reference list to print file / Not.
-lw[width]   :Specify print file columns per line.
-ll[length]  :Specify print file lines per page.
-lf/-nlf     :Add Form Feed at end of print file / Not.
-lt[n]       :Expand TAB character for print file(n=1 to 8) / Not expand(n=0).
-lhstring    :Print list header with the specified string.
-g/-ng       :Output debug information to object file / Not.
-j/-nj       :Create object file if fatal error occurred / Not.
-idirectory[,directory ...] :Set include search path.
-tdirectory  :Set temporary directory.
-ydirectory  :Set device file search path.
-ffile       :Input option or source module file name from specified file.
-ga/-nga     :Output assembler source debug information to object file / Not.
-dname[=data][,name[=data][...]] :Define name [with data].
-common      :Create the common object module file for 78k0r.
-self        :Use Self-programming.
-zs/-ze/-zn  :Change source regulation.
               -zs:SJIS code usable in comment.
               -ze:EUC code usable in comment.
               -zn:no multibyte code in comment.
-compati/-nocompati :Use macro for DIVUW,ROR4,ROL4,ADJBA,ADJBS,CALLF,DBNZ / Not.
--           :Show this message.
DEFAULT ASSIGNMENT :
          -o -ne -p -ka -nks -nkx -lw132 -ll0 -nlf -lt8 -g -nj -ga
```

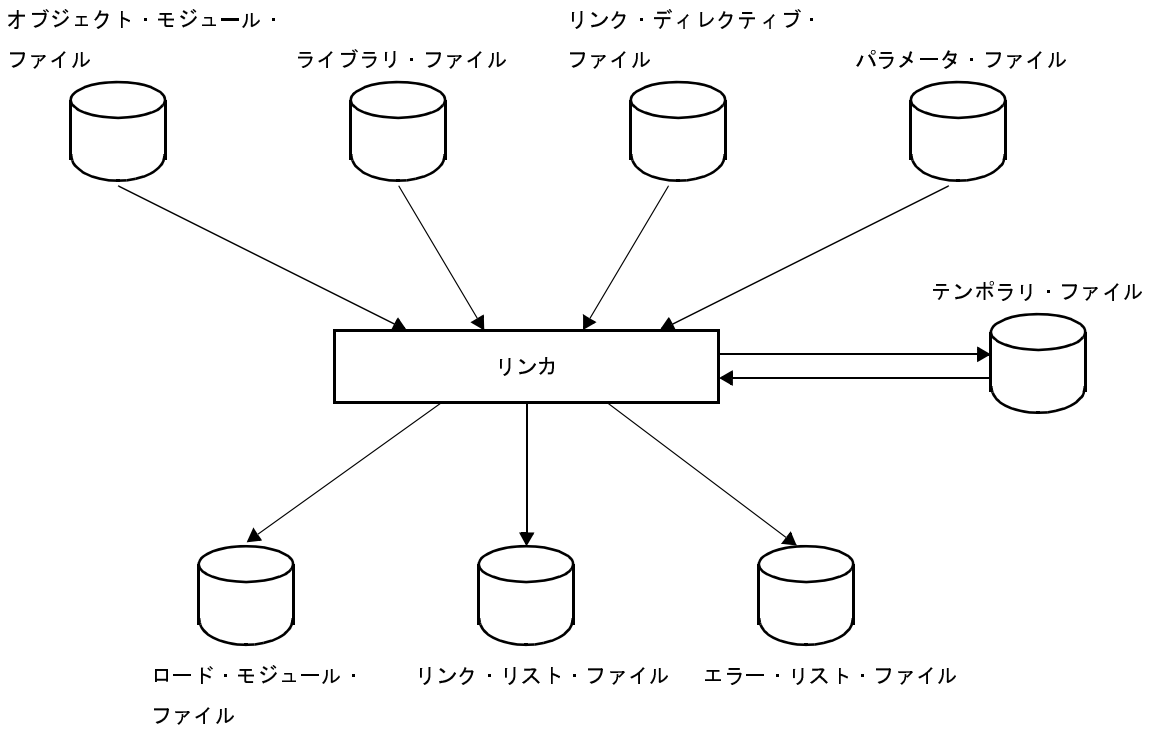
B. 3　リ ン カ

リンカは，RL78，および 78K0R 用のアセンブラが出力したいくつかのオブジェクト・モジュール・ファイルを入力し，配置アドレスを決定して 1 つにまとめたロード・モジュール・ファイルを出力します。

さらに，リンク・リスト・ファイルやエラー・リスト・ファイルなどのリスト・ファイルを出力します。

リンク・エラーがある場合は，エラー・メッセージをエラー・リスト・ファイルに出力し，エラーの原因を明示します。なお，エラーがある場合，ロード・モジュール・ファイルを出力しません。

図 B—5　リンカの入出力ファイル



B. 3. 1　入出力ファイル

リンカの入出力ファイルを次に示します。

出力リストについての詳細は，「3. 3　リ ン カ」を参照してください。

表 B—8　リンカの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイル - アセンブラの出力したファイル	.rel
	ライブラリ・ファイル	- 複数のオブジェクト・モジュール・ファイルが登録されたファイル - ライブラリアンの出力したファイル	.lib
	リンク・ディレクティブ・ファイル	- リンカに対するリンク指示を記述したファイル（ユーザ作成ファイル）	.dr
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイル（ユーザ作成ファイル）	.plk
出力ファイル	ロード・モジュール・ファイル	- リンク結果の全情報を持つバイナリ・イメージのファイル オブジェクト・コンバータの入力ファイルとなります。	.lmf
	リンク・リスト・ファイル	- リンク結果を表示するリスト・ファイル	.map
	エラー・リスト・ファイル	- リンク時のエラー情報を持つファイル	.elk
入出力ファイル	テンポラリ・ファイル	- リンクのためにリンカが自動生成するファイル リンク終了時には消去されます。	LK*xxxxx*.$$n （n＝1 - 3）

B. 3. 2　機　　能

(1) 入力セグメントの結合

リンカは，セグメントごとに配置するアドレスを決定し，管理します。

別々のオブジェクト・モジュール・ファイルにあっても，セグメントが同じであれば，そのセグメントを 1 つとみなして結合します。

(2) 入力モジュールの決定

入力としてライブラリ・ファイルが指定されていると，入力オブジェクト・モジュール・ファイルが参照しているモジュールをライブラリから探し出して，入力モジュールとして扱います。

(3) 入力セグメントの配置アドレス決定

入力モジュールの配置アドレスをセグメントごとに決定します。ソース・ファイル中で配置属性が指定されていれば，その属性に従って配置します。また，リンカの「リンク・ディレクティブ・ファイル」で配置属性を指定できます。

(4) オブジェクト・コードの修正

配置アドレスがオブジェクト・コードに埋め込まれるものは，(3) で決定した配置アドレスに従ってオブジェクト・コードを修正します。

B. 3. 3　操作方法

(1) リンカの起動方法

リンカの起動には，2 つの方法があります。

(a) コマンド行での起動

```
X:[パス名]>lk78k0r[△オプション]...オブジェクト・モジュール・ファイル名[△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
lk78k0r	リンカのコマンド名
オプション	リンカに対して動作の詳細を指示します。 複数のリンク・オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，リンク・オプションに大文字，小文字の区別はありません。リンク・オプションについての詳細は，「B. 3. 4　オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（”“）で囲んでください。
オブジェクト・モジュール・ファイル名	リンクするオブジェクト・モジュール・ファイル名 入力モジュールとして，最大 1024 個入力できます。 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（”“）で囲んでください。

例　デバッグ情報を付加したロード・モジュール・ファイル k0r.lmf を出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -ok0r.lmf -g
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，リンクするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>lk78k0r[△オブジェクト・モジュール・ファイル]△-f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	リンカの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[△] オプション [△オプション]...
```

- コマンド行でソース・ファイル名を省略した場合は，パラメータ・ファイル内でソース・ファイル名を１つだけ指定することができます。
- ソース・ファイル名は，オプションのあとに記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定するすべてのリンク・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル k0r.plk をエディタで作成し，リンカを起動します。

```
; parameter file
k0rmain.rel k0rsub.rel -ok0r.lmf -pk0r.map -e
-tC:¥tmp
```

```
C:¥>lk78k0r -fk0r.plk
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

リンカが起動すると，次の実行開始メッセージが表示されます。

```
78K0R Linker Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
```

(b) 実行終了メッセージ

リンクの結果，リンク・エラーが検出されなかった場合，リンカは次のメッセージを表示して制御をホスト OS に戻します。

```
Target chip : uPD78xxx
Device file : Vx.xx

Link complete, 0 error(s) and 0 warning(s) found.
```

リンクの結果，リンク・エラーが検出された場合，リンカはエラーとワーニングの数を表示して制御をホスト OS に戻します。

```
Target chip : uPD78xxx
Device file : Vx.xx

Link complete, 1 error(s) and 0 warning(s) found.
```

リンク中にリンカの処理継続が不可能な致命的エラーが検出された場合，リンカはメッセージを表示してリンクを中止し，制御をホスト OS に戻します。

- 存在しないオブジェクト・モジュール・ファイルを指定した場合

```
C:¥>lk78k0r samp1.rel samp2.rel
```

```
78K0R Linker Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

LK78K0R error F3006 : File not found 'samp1.rel'
LK78K0R error F3006 : File not found 'samp2.rel'
Program Aborted.
```

この例では，存在しないオブジェクト・モジュール・ファイルを指定したためにエラーとなり，リンクが中止されます。

- 存在しないリンク・オプションを指定した場合

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -z
```

```
78K0R Linker Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

LK78K0R error F3018 : Option is not recognized '-z'
Please enter 'LK78K0R --' , if you want help messages.
Program Aborted.
```

この例では，存在しないリンク・オプションを指定したためにエラーとなり，リンクが中止されます。

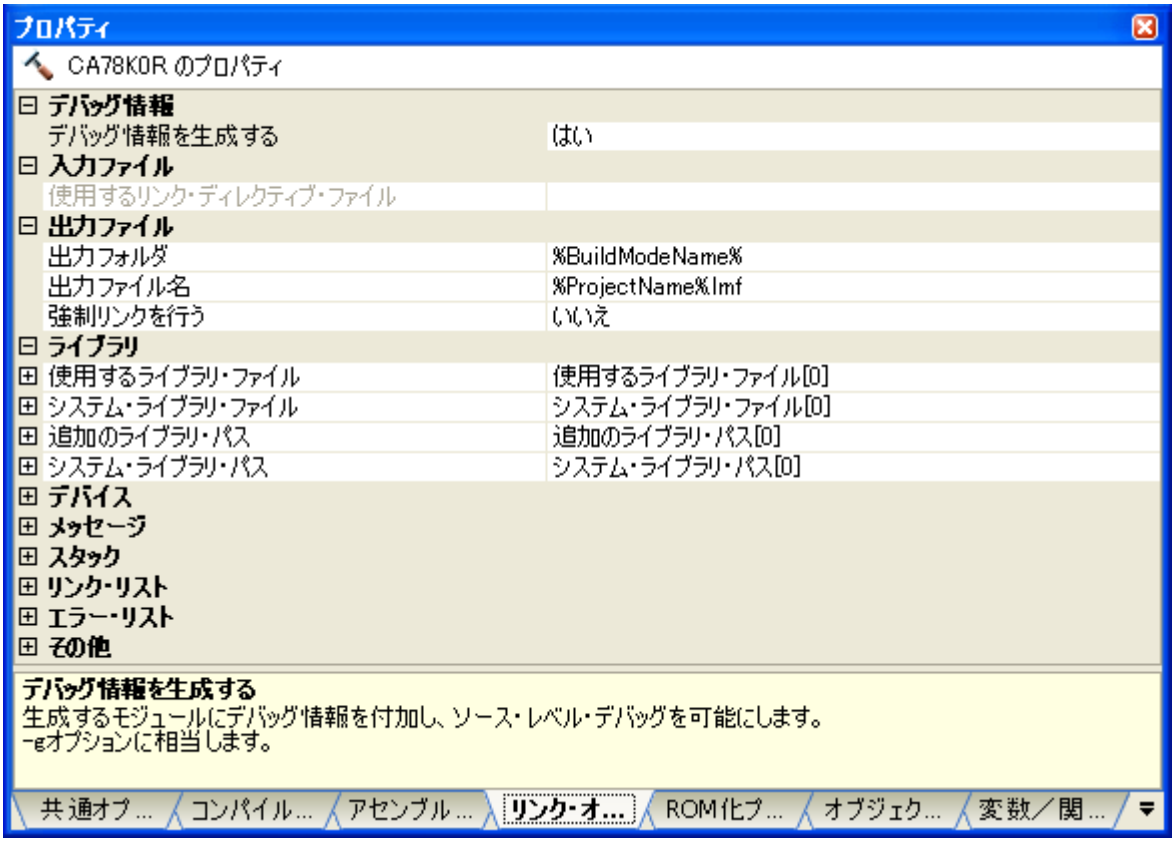
(3) CubeSuite+ でのオプション設定

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—6　プロパティ パネル：［リンク・オプション］タブ



B. 3. 4　オプション

（1）種類

リンク・オプションはリンカの動作に細かい指示を与えるものです。

リンク・オプションの分類と説明を示します。

表 B—9　リンク・オプション

分類	オプション	説明
ロード・モジュール・ファイル出力指定	-o	ロード・モジュール・ファイルの出力を指定します。
	-no	
ロード・モジュール・ファイル強制出力指定	-j	強制的にロード・モジュール・ファイルの出力をします。
	-nj	
デバッグ情報出力指定	-g	デバッグ情報をロード・モジュール・ファイルへ出力します。
	-ng	
スタック解決用シンボル生成指定	-s	スタック解決用のパブリック・シンボルを自動生成します。
	-ns	
リンク・ディレクティブ・ファイル指定	-d	指定のファイルをリンク・ディレクティブ・ファイルとして入力します。

分類	オプション	説明
リンク・リスト・ファイル出力指定	-p	リンク・リスト・ファイルの出力を指定します。
	-np	
リンク・リスト・ファイル情報指定	-km	リンク・リスト・ファイル中に，マップ・リストを出力します。
	-nkm	
	-kd	リンク・リスト・ファイル中に，リンク・ディレクティブ・ファイルを出力します。
	-nkd	
	-kp	リンク・リスト・ファイル中に，パブリック・シンボル・リストを出力します。
	-nkp	
	-kl	リンク・リスト・ファイル中に，ローカル・シンボル・リストを出力します。
	-nkl	
リンク・リスト・ファイル形式指定	-ll	リンク・リスト・ファイルの 1 頁に印字する行数を変更します。
	-lf	リンク・リスト・ファイルの最後に改頁コードを付加します。
	-nlf	
エラー・リスト・ファイル出力指定	-e	エラー・リスト・ファイルを出力します。
	-ne	
ライブラリ・ファイル指定	-b	指定のファイルをライブラリ・ファイルとして入力します。
ライブラリ・ファイル読み込みパス指定	-i	ライブラリ・ファイルを指定したパスから読み込みます。
パラメータ・ファイル指定	-f	入力ファイル名，オプションを指定したファイルより入力します。
テンポラリ・ファイル作成パス指定	-t	テンポラリ・ファイルを指定したパスに作成します。
デバイス・ファイル・サーチ・パス指定	-y	デバイス・ファイルを指定されたパスから読み込みます。
ワーニング・メッセージ出力指定	-w	ワーニング・メッセージをコンソールへ出力するか否かを指定します。
フラッシュ・メモリ内蔵製品のブート領域 ROM プログラムのリンク指定	-zb	フラッシュ・メモリ領域の先頭アドレスを指定します。
オンチップ・デバッグ指定	-go	オンチップ・デバッグを使用するか否かを指定します。
セキュリティ ID 指定	-gi	セキュリティ ID を指定します。
ユーザ・オプション・バイト指定	-gb	ユーザ・オプション・バイトに設定する値を指定します。
ミラー領域指定	-mi	ミラー元のセグメントの配置先を指定します。
64K バイト境界配置指定	-ccza	各 64K バイト領域の境界の最後の 1 バイトにセグメントの配置を行うかどうかを指定します。
	-nccza	

分類	オプション	説明
セルフ RAM 領域配置制御	-self	セルフ RAM 領域への配置制限を行うかどうかを指定します。
	-selfw	
コピー・ルーチン・アドレス指定	-rc	ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。
ROM 化領域指定	-ra	ROM 化対象領域を指定します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

(2) 優先度

次の表に示すリンク・オプションのうち，縦軸のものと横軸のものを同時に 2 つ以上指定した場合の優先度について説明します。

表 B—10　リンク・オプションの優先度

	-no	-ng	-np	-nkm	-nkp	-nkl	--
-j	×						×
-g	×						×
-p				△	△	△	×
-km			×				×
-kd			×	×			×
-kp		×	×				×
-kl		×	×				×
-ll			×				×
-lf			×				×

- ×で記した箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例　-km オプションは，無効となります。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -np -km
```

- △で記した箇所

横軸に示したオプション 3 つをすべて同時に指定した場合，縦軸に示したオプションは無効となります。

例　-nkm，-nkp，および -nkl を同時に指定したので，-p オプションは無効となります。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -p -nkm -nkp -nkl
```

- 空欄の箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは有効となります。

また，-o/-no オプションのように，オプション名の前に“n”を付加できるオプションを同時に指定した場合，あとで指定した方が有効となります。

例　-no オプションがあとに指定されているので，-o オプションは無効となり，-no オプションが有効となります。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -o -no
```

「表 B—10　リンク・オプションの優先度」に記述されていないオプションは，ほかのオプションの影響を特に受けません。しかし，ヘルプ指定オプション（ -- ）が指定された場合には，すべてのオプション指定が無効となります。

ロード・モジュール・ファイル出力指定

ロード・モジュール・ファイル出力指定オプションには，次のものがあります。

- -o/-no

-o/-no

[記述形式]

```
-o [ 出力ファイル名 ]
-no
```

- 省略時解釈

  -o *入力ファイル名* .lmf

[機能]

- -o オプションは，ロード・モジュール・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -no オプションは，-o，-j，-g オプションを無効にします。

[用途]

- ロード・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，-o オプションを指定します。
- リンク・リスト・ファイルの出力のみが目的でリンクする場合などに，-no オプションを指定します。これにより，リンク時間が短縮されます。

[説明]

- -o オプションを指定してもフェイタル・エラーがある場合は，ロード・モジュール・ファイルは出力されません。
- -o オプションを指定する際に“*出力ファイル名*”を省略すると，カレント・フォルダにロード・モジュール・ファイル“*入力ファイル名* .lmf ”が出力されます。
- “*出力ファイル名*”にパス名のみを指定すると，指定したパスに“*入力ファイル名* .lmf”が出力されます。
- -o と -no の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- ロード・モジュール・ファイル k0r.lmf を出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -ok0r.lmf
```

ロード・モジュール・ファイル強制出力指定

ロード・モジュール・ファイル強制出力指定オプションには，次のものがあります。

- -j/-nj

-j/-nj

[記述形式]

```
-j
-nj
```

- 省略時解釈

  -nj

[機能]

- -j オプションは，フェイタル・エラーの場合にでもロード・モジュール・ファイルを出力するよう指示します。
- -nj オプションは，-j オプションを無効にします。

[用途]

- 通常，フェイタル・エラーがある場合には，ロード・モジュール・ファイルは出力されません。
  したがって，フェイタル・エラーがあるのを承知でコマンドを実行させたい場合には，-j オプションを指定しロード・モジュール・ファイルを出力します。

[説明]

- -j オプションを指定すると，フェイタル・エラーがある場合でも，ロード・モジュール・ファイルが出力されます。
- -j と -nj の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -no オプションを指定した場合は，-j オプションは無効となります。

[使用例]

- ロード・モジュール・ファイル k0rsub.lmf を強制的に出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -j
```

デバッグ情報出力指定

デバッグ情報出力指定オプションには，次のものがあります。

- -g/-ng

-g/-ng

[記述形式]

```
-g
-ng
```

- 省略時解釈

  -g

[機能]

- -g オプションは，ロード・モジュール・ファイル中にデバッグ情報（ローカル・シンボル情報）を付加することを指定します。
- -ng オプションは，-g，-kp，-kl オプションを無効にします。

[用途]

- ソース・デバッガでシンボリック・デバッグを行うときには，必ず -g オプションを指定します。

[説明]

- -ng オプションが指定された場合は，パブリック・シンボル・リスト，およびローカル・シンボル・リストは出力されません。
- -g と -ng の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -no オプションを指定した場合は，-g オプションは無効となります。

[使用例]

- ロード・モジュール・ファイル k0rsub.lmf にデバッグ情報を付加します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -g
```

スタック解決用シンボル生成指定

スタック解決用シンボル生成指定オプションには，次のものがあります。

- -s/-ns

-s/-ns

[記述形式]

```
-s[領域名]
-ns
```

- 省略時解釈

  -ns

[機能]

- -s オプションは，スタック解決用のパブリック・シンボル“_@STBEG”と“_@STEND”を生成します。
- -ns オプションは，-s オプションを無効にします。

[用途]

- スタック領域を確保するために -s オプションを指定します。

[説明]

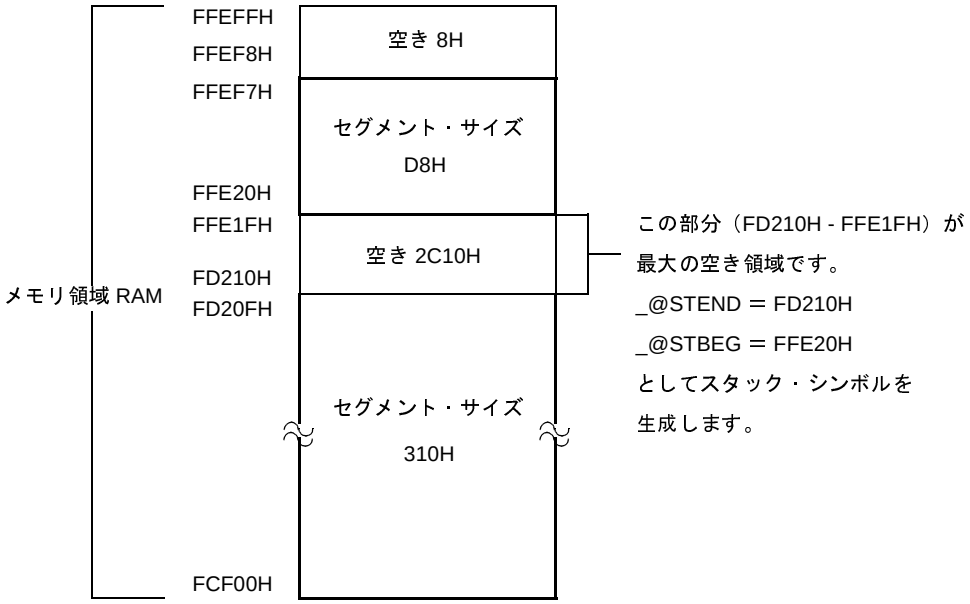
- 領域名には，利用者が定義したメモリ領域名，またはデフォルトで定義されているメモリ領域名を指定します。
- 領域名は，大文字と小文字を区別します。
- リンカは，-s オプションで指定したメモリ領域の中で，セグメントが配置されていない領域のうち，最大の空き領域を探します。そして，見つかった最大の空き領域の先頭アドレスを値として持つパブリック・シンボル“_@STEND”と，最終アドレス＋1 を値として持つパブリック・シンボル“_@STBEG”を生成します。
  これらのシンボルは，パブリック宣言された NUMBER 属性のシンボルとして扱われ，リンカのシンボル・テーブルの最後に登録されます。また，リンク・リスト・ファイルに出力される際，モジュール名欄は空白となります。
- 最大の空き領域のサイズが 10 バイト以下の場合は，ワーニング・メッセージが出力されます。
- 空き領域が存在しない場合は，ワーニング・メッセージが出力され，“_@STEND”と“_@STBEG”は指定したメモリ領域の最終アドレス＋1 を持ちます。
- 領域名を省略した場合は，“RAM”が指定されたものとします。
- -s と -ns の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- メモリ領域 RAM にスタック領域を確保します。

  ただし，RAM 領域にサイズ 310H のセグメントと saddr 領域に配置するサイズ D8H のセグメントを入力すると仮定します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -s
```



リンク・ディレクティブ・ファイル指定

リンク・ディレクティブ・ファイル指定オプションには，次のものがあります。

- -d

-d

[記述形式]

```
-d ファイル名
```

- 省略時解釈

  なし

[機能]

- -d オプションは，指定ファイルをリンク・ディレクティブ・ファイルとして入力することを指定します。

[用途]

- メモリ領域を新たに定義したり，デフォルトのメモリ領域を再定義する場合，またはセグメントを特定のアドレスやメモリ領域に配置したい場合，リンク・ディレクティブ・ファイルを作成します。このリンク・ディレクティブ・ファイルをリンカに入力するために，-d オプションを指定します。

[説明]

- ファイル名を省略すると，アボート・エラーとなります。
- リンク・ディレクティブ・ファイルのネストは許されません。
- リンク・ディレクティブ・ファイル中に記述可能な文字数に，制限はありません。
- -d オプションを複数指定したり，ファイル名を複数指定すると，アボート・エラーとなります。
- リンク・ディレクティブ・ファイルについての詳細は，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

[使用例]

- デフォルトのメモリ領域 ROM/RAM を再定義します。

  リンク・ディレクティブ・ファイル k0r.dr の内容は，下記のようになります。

```
MEMORY ROM : ( 0H , 40000H )
MEMORY RAM : ( 0FCF00H , 3000H )
```

リンク・ディレクティブ・ファイル k0r.dr をリンクします。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr
```

リンク・リスト・ファイル出力指定

リンク・リスト・ファイル出力指定オプションには，次のものがあります。

- -p/-np

-p/-np

[記述形式]

```
-p[出力ファイル名]
-np
```

- 省略時解釈

  -p *入力ファイル名*.map

[機能]

- -p オプションは，リンク・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -np オプションは，-p，-km，-kd，-kp，-kl，-ll，-lf オプションを無効にします。

[用途]

- リンク・リスト・ファイルの出力先や出力ファイル名を変更する場合は，-p オプションを指定します。
- ロード・モジュール・ファイルの出力だけを目的でリンクする場合などに，-np オプションを指定します。これにより，リンク時間が短縮されます。

[説明]

- -p オプションを指定する際に“*出力ファイル名*”を省略すると，カレント・フォルダにリンク・リスト・ファイル“*入力ファイル名*.map”を出力します。
- “出力ファイル名”にパス名のみを指定すると，指定したパスに“*入力ファイル名*.map”を出力します。
- -p と -np の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- リンク・リスト・ファイル k0r.map を作成します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -pk0r.map
```

リンク・リスト・ファイル情報指定

リンク・リスト・ファイル情報指定オプションには，次のものがあります。

- -km/-nkm
- -kd/-nkd
- -kp/-nkp
- -kl/-nkl

-km/-nkm

[記述形式]

```
-km
-nkm
```

- 省略時解釈

  -km

[機能]

- -km オプションは，リンク・リスト・ファイル中にマップ・リストを出力します。
- -nkm オプションは，-km，-kd オプションを無効にします。

[用途]

- リンク・リスト・ファイル中にマップ・リストを出力したいときに，-km オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションをすべて指定した場合は，リンク・リスト・ファイルは出力されません。
- -nkm オプションを指定した場合は，リンク・リスト・ファイル中にリンク・ディレクティブ・ファイルは出力されません。
- -km と -nkm の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-km オプションは無効となります。

[使用例]

- リンク・リスト・ファイル k0r.map にマップ・リストを出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -s -pk0r.map -km
```

k0r.map の内容は，以下のようになります。

```
78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   1

Command:    k0rmain.rel k0rsub.rel -s -pk0r.map -km
Para-file:
Out-file:   k0rmain.lmf
Map-file:   k0r.map
Direc-file:
Directive:

*** Link information ***

     4 output segment(s)
   5FH byte(s) real data
    41 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=00000H   SIZE=40000H
         OUTPUT   INPUT    INPUT    BASE     SIZE
         SEGMENT  SEGMENT  MODULE   ADDRESS
         CODE                       00000H   00002H   CSEG AT
                  CODE     SAMPM    00000H   00002H
* gap *                             00002H   000BEH
         ?CSEGOB0                   000C0H   00004H   CSEG OPT_BYTE
         ?CSEG                      000C4H   00059H   CSEG
                  ?CSEG    SAMPM    000C4H   00017H
                  ?CSEG    SAMPS    000DBH   00042H
* gap *                             0011DH   3FEE3H

  MEMORY=LRAM
  BASE ADDRESS=FCF00H   SIZE=03100H
         OUTPUT   INPUT    INPUT    BASE     SIZE
         SEGMENT  SEGMENT  MODULE   ADDRESS
* gap *                             FCF00H   02F20H
         DATA                       FFE20H   00003H   DSEG AT
                  DATA     SAMPM    FFE20H   00003H
* gap *                             FFE23H   000DDH
* gap (Not Free Area) *             FFF00H   00100H
```

-kd/-nkd

[記述形式]

```
-kd
-nkd
```

- 省略時解釈
  -kd

[機能]

- -kd オプションは，リンク・リスト・ファイル中にリンク・デイレクティブ・ファイルを出力します。
- -nkd オプションは，-kd オプションを無効にします。

[用途]

- リンク・リスト・ファイル中にリンク・ディレクティブ・ファイルを出力したい場合は，-kd オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションをすべて指定した場合は，リンク・リスト・ファイルは出力されません。
- -nkm オプションを指定した場合は，リンク・リスト・ファイル中にリンク・ディレクティブ・ファイルは出力されません。
- -kd と -nkd の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-kd オプションは無効となります。

[使用例]

- リンク・リスト・ファイル k0r.map にリンク・ディレクティブ・ファイルを出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -s -dk0r.dr -pk0r.map -kd
```

k0r.map の内容は，以下のようになります。

```
78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   1

Command:    k0rmain.rel k0rsub.rel -s -dk0r.dr -pk0r.map -kd
Para-file:
Out-file:   k0rmain.lmf
Map-file:   k0r.map
Direc-file: k0r.dr                          ←リンク・ディレクティブ・ファイル名
Directive:  MEMORY ROM : ( 0H , 0ED800H )   ←リンク・ディレクティブ・ファイルの内容
            MEMORY RAM : ( 0FCF00H , 1100H )
            MEMORY RAM : ( 0FE000H , 1F00H )

*** Link information ***

     6 output segment(s)
   9DH byte(s) real data
    40 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=00000H   SIZE=ED800H
         OUTPUT   INPUT    INPUT    BASE     SIZE
         SEGMENT  SEGMENT  MODULE   ADDRESS
         CODE                       00000H   00002H   CSEG AT
          :
```

-kp/-nkp

[記述形式]

```
-kp
-nkp
```

- 省略時解釈

  -nkp

[機能]

- -kp オプションは，リンク・リスト・ファイル中にパブリック・シンボル・リストを出力します。
- -nkp オプションは，-kp オプションを無効にします。

[用途]

- リンク・リスト・ファイル中にパブリック・シンボル・リストを出力する場合に，-kp オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションをすべて指定した場合は，リンク・リスト・ファイルは出力されません。
- -ng オプションを指定した場合は，パブリック・シンボル・リストは出力されません。
- -kp と -nkp の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-kp オプションは無効となります。

[使用例]

- リンク・リスト・ファイル k0r.map に，パブリック・シンボル・リストを出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -s -g -pk0r.map -kp
```

k0r.map の内容は，以下のようになります。

```
78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   1

Command:    k0rmain.rel k0rsub.rel -s -g -pk0r.map -kp
Para-file:
Out-file:   k0rmain.lmf
Map-file:   k0r.map
Direc-file:
Directive:

*** Link information ***

     6 output segment(s)
   9DH byte(s) real data
    40 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS= 00000H   SIZE=40000H
        :
-------------------------------------------------------------------------------------

78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   2

*** Public symbol list ***

MODULE  ATTR  VALUE    NAME

SAMPM
        ADDR  00000H   MAIN
        ADDR  000D2H   START
SAMPS
        ADDR  000E9H   CONVAH
        NUM   FFE20H   _@STBEG
        NUM   FCF00H   _@STEND

 Target chip : uPD78xxx
 Device file : Vx.xx
```

-kl/-nkl

[記述形式]

```
-kl
-nkl
```

- 省略時解釈

  -nkl

[機能]

- -kl オプションは，リンク・リスト・ファイル中にローカル・シンボル・リストを出力します。
- -nkl オプションは，-kl オプションを無効にします。

[用途]

- リンク・リスト・ファイル中にローカル・シンボル・リストを出力する場合に，-kl オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションがすべて指定された場合は，リンク・リスト・ファイルは出力されません。
- -ng オプションを指定した場合は，ローカル・シンボル・リストは出力されません。
- -kl と -nkl の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-kl オプションは無効となります。

[使用例]

- リンク・リスト・ファイル k0r.map に，ローカル・シンボル・リストを出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -s -g -pk0r.map -kl
```

k0r.map の内容は，以下のようになります。

```
78K0R Linker Vx.xx                                          Date:xx xxx xxxx  Page:   1

Command:    k0rmain.rel k0rsub.rel -s -g -pk0r.map -kl
Para-file:
Out-file:   k0rmain.lmf
Map-file:   k0r.map
Direc-file:
Directive:

*** Link information ***

     6 output segment(s)
   9DH byte(s) real data
    40 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR
        :
-------------------------------------------------------------------------------------

78K0R Linker Vx.xx                                          Date:xx xxx xxxx  Page:   2

*** Local symbol list ***

MODULE  ATTR  VALUE    NAME

SAMPM
        MOD            SAMPM
        DSEG           DATA
        ADDR  FFE20H   HDTSA
        ADDR  FFE21H   STASC
        CSEG           CODE
        CSEG           ?CSEG
SAMPS
        MOD            SAMPS
        CSEG           ?CSEG
        ADDR  0015CH   SASC
        ADDR  00162H   SASC1

 Target chip : uPD78xxx
 Device file : Vx.xx
```

リンク・リスト・ファイル形式指定

リンク・リスト・ファイル形式指定オプションには，次のものがあります。

- -ll
- -lf/-nlf

-ll

[記述形式]

```
-ll[行数]
```

- 省略時解釈

  -ll0（改頁しません）

[機能]

- -ll オプションは，リンク・リスト・ファイルの 1 頁の行数を指定します。

[用途]

- リンク・リスト・ファイルの 1 頁の行数を変更したいときに，-ll オプションで指定します。

[説明]

- -ll オプションで指定可能な行数の範囲は，20 ～ 32767 です。
- 範囲外の数値や数値以外のものが指定された場合には，アボート・エラーとなります。
- 行数を省略した場合，0 を指定したものとみなされます。
- 行数に 0 を指定した場合は，改頁しません。
- -np オプションを指定した場合は，-ll オプションは無効となります。

[使用例]

- リンク・リスト・ファイル k0r.map の 1 頁の行数を 20 行に指定します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -s -pk0r.map -ll20
```

k0r.map の内容は，以下のようになります。

```
78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   1

Command:    k0rmain.rel k0rsub.rel -s -pk0r.map -km -ll20
Para-file:
Out-file:   k0rmain.lmf
Map-file:   k0r.map
Direc-file:
Directive:

*** Link information ***

     4 output segment(s)
   5FH byte(s) real data
    41 symbol(s) defined

-------------------------------------------------------------------------------------

78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   2

*** Memory map ***

  SPACE=REGULAR

  MEMORY=ROM
  BASE ADDRESS=00000H  SIZE=40000H
         OUTPUT   INPUT    INPUT    BASE    SIZE
         SEGMENT  SEGMENT  MODULE   ADDRESS
         CODE                       00000H  00002H  CSEG AT
                  CODE     SAMPM    00000H  00002H
* gap *                             00002H  000BEH
         ?CSEGOB0                   000C0H  00004H  CSEG OPT_BYTE
         ?CSEG                      000C4H  00059H  CSEG
                  ?CSEG    SAMPM    000C4H  00017H
-------------------------------------------------------------------------------------

78K0R Linker Vx.xx                                        Date:xx xxx xxxx  Page:   3

                  ?CSEG    SAMPS    000DBH  00042H
* gap *                             0011DH  3FEE3H

  MEMORY=RAM
  BASE ADDRESS=FCF00H  SIZE=03100H
         OUTPUT   INPUT    INPUT    BASE    SIZE
         SEGMENT  SEGMENT  MODULE   ADDRESS
* gap *                             FCF00H  02F20H
```

```
        DATA                        FFE20H  00003H  DSEG AT
                 DATA     SAMPM     FFE20H  00003H
* gap *                             FFE23H  000DDH
* gap (Not Free Area) *             FFF00H  00100H

 Target chip : uPD78xxx
 Device file : Vx.xx
```

-lf/-nlf

[記述形式]

```
-lf
-nlf
```

- 省略時解釈
  -nlf

[機能]

- -lf オプションは，リンク・リスト・ファイルの最後に，改頁コード（FF）を付加することを指定します。
- -nlf オプションは，-lf オプションを無効にします。

[用途]

- リンク・リスト・ファイルの内容を印字したあとで改頁しておきたい場合，-lf オプションを指定して改頁コードを付加します。

[説明]

- -np オプションを指定した場合は，-lf オプションは無効となります。
- -lf と -nlf の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- リンク・リスト・ファイル k0r.map の最後に改頁コードを付加します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -pk0r.map -lf
```

エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定オプションには，次のものがあります。

- -e/-ne

-e/-ne

[記述形式]

```
-e[ ファイル名 ]
-ne
```

- 省略時解釈

  -ne

[機能]

- -e オプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -ne オプションは，-e オプションを無効にします。

[用途]

- エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-e オプションを指定します。

[説明]

- -e オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は“*入力ファイル名*.elk”となります。
- -e オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- -e と -ne の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- エラー・リスト・ファイル k0r.elk を作成します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -dk0r.dr -ek0r.elk
```

リンク・ディレクティブ・ファイル k0r.dr の内容に誤りがありました。

エラー・リスト・ファイル k0r.elk の内容は，以下のようになります。

```
k0r.dr(3) : RA78K0R error E3102: Directive syntax error
```

ライブラリ・ファイル指定

ライブラリ・ファイル指定オプションには，次のものがあります。

- -b

-b

[記述形式]

```
-b ファイル名
```

- 省略時解釈

  なし

[機能]

- -b オプションは，指定ファイルをライブラリ・ファイルとして入力することを指定します。

[用途]

- リンカは，入力モジュールが参照しているモジュールをライブラリ・ファイルから探し出し，そのモジュールだけを入力モジュールと結合します。
- ライブラリ・ファイルは，あらかじめ複数のモジュールを登録して，1 つのファイルにまとめておくためのものです。
- 共通的なモジュールをライブラリ・ファイルとして作成しておけば，ファイル管理の面でも操作性の面においても効率が良くなります。ライブラリ・ファイルをリンカに入力するには，-b オプションを指定します。

[説明]

- ファイル名を省略することはできません。
- ファイル名としてパス名を含めて指定した場合は，指定したパスからライブラリ・ファイルを入力します。指定したパスにライブラリ・ファイルが存在しない場合は，エラーとなります。
- ファイル名としてパス名を含まずに指定した場合は，-i オプションの指定，およびデフォルトのサーチ・パスからライブラリ・ファイルを入力します。
- -b オプションが複数指定された場合は，指定順にライブラリ・ファイルの入力を行います。-b オプションは最大 64 個まで指定することができます。
- ライブラリ・ファイルの作成方法についての詳細は，「B. 6　ライブラリアン」を参照してください。

［使用例］

- ライブラリ・ファイル k0r.lib を入力します。

  ライブラリ・ファイルには，k0rsub.rel が登録されています。

```
C:¥>lk78k0r k0rmain.rel -bk0r.lib
```

ライブラリ・ファイル読み込みパス指定

ライブラリ・ファイル読み込みパス指定オプションには，次のものがあります。

- -i

-i

[記述形式]

```
-i パス名 [, パス名] …（複数指定可能）
```

- 省略時解釈

  環境変数 LIB78K0R で指定したパス

  環境変数 LIB78K0R が指定されていない場合は，カレント・パス

[機能]

- -i オプションは，ライブラリ・ファイルを指定したパスから入力するよう指示します。

[用途]

- ライブラリ・ファイルをあるパスから検索したいときに指定します。

[説明]

- -i オプションは，-b オプションでパス名を含まないライブラリ・ファイル名が指定された場合にのみ有効です。
- -i は複数指定することができます。また，“ , ” で区切ることにより，一度に複数のパスを指定することができます。この際“ , ”の前後には，空白を入れることはできません。
- パス名は，1 回のリンクで最大 64 個まで指定可能です。パス名を複数指定した場合，リンカは指定順にライブラリ・ファイルの検索を行います。
- 指定したパスにライブラリ・ファイルが存在しない場合でも，エラーにはなりません。
- パス名を省略した場合は，アボート・エラーとなります。
- -b オプションでパス名を含まずにライブラリ・ファイルを指定した場合は，リンカは次に示す順序でパスを検索します。

（1）-i オプションで指定したパス

（2）環境変数 LIB78K0R で指定したパス

（3）カレント・パス

注意　いずれのパスにも指定された名前のライブラリ・ファイルが存在しない場合は，エラーとなります。

［使用例］

- ライブラリ・ファイルをフォルダ C:￥lib1，C:￥lib2 の順で検索し，読み込みます。

```
C:￥>lk78k0r k0rmain.rel k0rsub.rel -bk0r.lib -iC:￥lib1,C:￥lib2
```

- ライブラリ・ファイルをフォルダ D:￥library files から読み込みます。

```
C:￥>lk78k0r k0rmain.rel k0rsub.rel -bk0r.lib -i"D:￥library files"
```

パラメータ・ファイル指定

パラメータ・ファイル指定オプションには，次のものがあります。

- -f

-f

[記述形式]

```
-f ファイル名
```

- 省略時解釈
  コマンド行上からのみオプション，入力ファイル名の入力が可能となります。

[機能]

- -f オプションは，オプション，あるいは入力ファイル名を指定のファイルから入力することを指定します。

[用途]

- コマンド行では，リンカの起動に必要な情報を指定しきれないときに，-f オプションを指定します。
- リンクするたび繰り返し同じようにオプションを指定する場合には，それらをパラメータ・ファイルに記述しておき，-f オプションで指定します。

[説明]

- ファイル名を省略すると，アボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で -f オプションを指定すると，アボート・エラーとなります。
- パラメータ・ファイル中に記述可能な文字数に，制限はありません。
- 空白とタブ，および改行文字（LF）をオプション，あるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，あるいは入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- “；”，または“#”以降に記述された文字は，改行文字（LF），または EOF の前まですべてコメントと解釈します。
- -f オプションを複数指定すると，アボート・エラーとなります。

［使用例］

- パラメータ・ファイル k0r.plk を使用してリンクします。

パラメータ・ファイル k0r.plk の内容は，以下のようになります。

```
; parameter file
k0rmain.rel k0rsub.rel -ok0r.lmf -pk0r.map -e
-tC:¥tmp -g
```

コマンド行には，次のように入力します。

```
C:¥>lk78k0r -fk0r.plk
```

テンポラリ・ファイル作成パス指定

テンポラリ・ファイル作成パス指定オプションには，次のものがあります。

- -t

-t

[記述形式]

```
-t パス名
```

- 省略時解釈
  環境変数 TMP で指定したパス
  環境変数 TMP を指定していない場合は，カレント・パス

[機能]

- -t オプションは，テンポラリ・ファイルを作成するパスを指示します。

[用途]

- テンポラリ・ファイルの作成場所を指定することができます。

[説明]

- パス名として，パス以外のものは指定できません。
- パス名を省略することはできません。
- 以前に作成したテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ上書きされます。
- 必要とするメモリ・サイズがある間は，テンポラリ・ファイルはメモリに展開されます。
  メモリが足りなくなった時点で，メモリに展開されていたテンポラリ・ファイルの内容がディスクに書き出されます。
  以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行われます。
- テンポラリ・ファイルは，リンク終了時に削除されます。また，キー入力（［Ctrl］＋［C］キー）によってリンクが中止されたときにも削除されます。
- テンポラリ・ファイルの作成パスは，次の順番で決定されます。

(1) -t オプションで指定されたパス

(2) 環境変数 TMP で設定したパス（-t オプション省略の場合）

（3）カレント・パス（TMP を設定していない場合）

注意　（1），または（2）を指定した場合，指定されたパスにテンポラリ・ファイルが作成できなければ，アボート・エラーとなります。

［使用例］

- テンポラリ・ファイルをフォルダ C:￥tmp に出力します。

```
C:￥>lk78k0r k0rmain.rel k0rsub.rel -tC:￥tmp
```

- テンポラリ・ファイルをフォルダ D:￥temporary files に出力します。

```
C:￥>lk78k0r k0rmain.rel k0rsub.rel -t"D:￥temporary files"
```

デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定オプションには，次のものがあります。

- -y

-y

[記述形式]

```
-y パス名
```

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) lk78k0r.exe の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

[機能]

- -y オプションは，デバイス・ファイルを指定されたパスから読み込みます。

[用途]

- デバイス・ファイルが存在するパスを指定します。

[説明]

- -y オプションに続けてパス名以外が指定された場合，アボート・エラーとなります。
- -y オプションに続けて指定するパス名が省略された場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べて決定します。

(1) -y オプションで指定されたパス

(2) デバイス・ファイル・インストーラで登録されたパス

（3）LK78K0R の起動されたパス

（4）カレント・フォルダ

（5）環境変数 PATH

［使用例］

- デバイス・ファイルのパスをフォルダ C:￥78k0r￥dev に指定します。

```
C:￥>lk78k0r k0rmain.rel k0rsub.rel -yC:￥78k0r￥dev
```

- デバイス・ファイルのパスをフォルダ D:￥device files に指定します。

```
C:￥>lk78k0r k0rmain.rel k0rsub.rel -y"D:￥device files"
```

ワーニング・メッセージ出力指定

ワーニング・メッセージ出力指定オプションには，次のものがあります。

- -w

-w

[記述形式]

```
-w[レベル]
```

- 省略時解釈

  -w1

[機能]

- -w オプションは，ワーニング・メッセージをコンソールへ出力するか否かを指定します。

[用途]

- ワーニング・メッセージを出力させるレベルを指定することができます。

[説明]

- -w オプションに続けてレベル以外が指定された場合は，アボート・エラーとなります。
- レベルには，0，1，2 以外は指定できません。
- 出力させるレベルには，以下のものがあります。

  0：ワーニング・メッセージを出力しません。

  1：通常のワーニング・メッセージを出力します。

  2：詳細なワーニング・メッセージを出力します。

[使用例]

- 詳細なワーニング・メッセージを出力します。

```
C:¥>lk78k0r k0rmain.rel k0rsub.rel -w2
```

フラッシュ・メモリ内蔵製品のブート領域 ROM プログラムのリンク指定

フラッシュ・メモリ内蔵製品のブート領域 ROM プログラムのリンク指定オプションには，次のものがあります。

- -zb

-zb

[記述形式]

```
-zb アドレス
```

- 省略時解釈

  配置範囲の制限はありません。

[機能]

- -zb オプションは，フラッシュ・メモリ領域の先頭アドレスを指定します。

[説明]

- フラッシュ・メモリ内蔵製品のブート領域 ROM プログラムのリンク指定を行い，フラッシュ・メモリ領域の先頭アドレスを指定します。
- 指定可能な値の範囲は，0H ～ 0FFFFH です。
- アドレスを省略した場合は，エラーとなります。
- 設定したアドレス以降には，コードは配置できません。

注意　フラッシュ・メモリ領域セルフ書き換え機能を持たないデバイスでは，本オプションを使用しないでください。

[使用例]

- フラッシュ・メモリ領域の先頭アドレスとして 2000h を指定します。

```
C:¥>lk78k0r k0rmain.rel -zb2000H
```

オンチップ・デバッグ指定

オンチップ・デバッグ指定オプションには，次のものがあります。

- -go

-go

[記述形式]

-go *制御値*，*スタート・アドレス*［，*サイズ*］

- 省略時解釈

  オンチップ・デバッグを使用しません。

  C3H 番地はデバイス・ファイルの初期値

[機能]

- -go オプションは，オンチップ・デバッグを使用するか否かを指定します。

[用途]

- オンチップ・デバッグの制御値，デバッグ・モニタ領域のスタート・アドレス，およびサイズを変更したいときに，-go オプションで指定します。

[説明]

- 制御値には，オンチップ・デバッグ動作の制御値を設定してください。

  制御値として指定できない値を指定した場合は，アボート・エラーとなります。

  制御値についての詳細は，「QB-MINI2 プログラミング機能付きオンチップ・デバッグ・エミュレータ」のユーザーズ・マニュアル（U18371JJ）を参照してください。
- スタート・アドレスには，デバッグ・モニタ領域のスタート・アドレスを設定してください。

  スタート・アドレスとして指定可能な値の範囲は，0 ～ 0FFFFFH です。

  スタート・アドレスを省略した場合は，（内部 ROM のエンド・アドレス － 512）＋ 1【RL78】／（内部 ROM のエンド・アドレス － 1024）＋ 1【78K0R】を指定したものとみなされます。

  スタート・アドレスについての詳細は，「QB-MINI2 プログラミング機能付きオンチップ・デバッグ・エミュレータ」のユーザーズ・マニュアル（U18371JJ）を参照してください。
- サイズには，デバッグ・モニタ領域のサイズを設定してください。

  サイズとして指定可能な値の範囲は，0 ～ 1024【RL78】／ 88 ～ 1024【78K0R】です。

  サイズを省略した場合は，512 バイト【RL78】／ 1024 バイト【78K0R】を指定したものとみなされます。

  プログラム・サイズについての詳細は，「QB-MINI2 プログラミング機能付きオンチップ・デバッグ・エミュレータ」のユーザーズ・マニュアル（U18371JJ）を参照してください。

- 制御値，スタート・アドレス，サイズに数値以外が指定された場合には，アボート・エラーとなります。
  ただし，OCD プログラムの配置エンド・アドレス（スタート・アドレス＋サイズ）が内部 ROM のエンド・アドレス以下である必要があります。
- -go オプションが指定された場合，C3H 番地には，制御値が配置されます。
  2H，3H，CEH ～ D7H 番地と，-go オプションで指定したスタート・アドレスから指定されたサイズ分の領域は，FFH で充てんされるため，セグメントを配置することはできません。
  なお，C0 ～ C2H 番地は，-gb オプションにより，ユーザ・オプション・バイトの領域として確保されます。
- -go オプションが指定されなかった場合でも，C3H 番地は予約領域とするため，ユーザ・コードを配置することはできません。
- C3H 番地の制御値は，アセンブラ・ソース・ファイル中に以下の再配置属性のセグメントを定義することでも指定可能です。ただし，C0H 番地からのユーザ・オプション・バイトとあわせて 4 バイトで定義してください。

例　μPD78F1165 の場合

```
[ 任意のセグメント名 ]      CSEG    OPT_BYTE
                            DB      0FDH     ; 0xC0 番地
                            DB      0FEH     ; 0xC1 番地
                            DB      0FFH     ; 0xC2 番地
                            DB      04H      ; 0xC3 番地
```

アセンブラ・ソース・ファイルの指定と本オプションの指定が重なった場合には，本オプションが優先されます。
- オンチップ・デバッグ・オプション・バイト，ユーザ・オプション・バイトは，必ずデバイスのユーザーズ・マニュアルを参照して設定を行ってください。

[使用例]

- C3H 番地に，制御値として 04H を埋め込みます。
- スタート・アドレス（02FC00H 番地）から 1024 バイトをデバッグ・モニタ領域として確保します。

```
C:¥>lk78k0r k0rmain.rel -go04H,02FC00H,1024
```

セキュリティ ID 指定

セキュリティ ID 指定オプションには，次のものがあります。

- -gi

-gi

[記述形式]

```
-gi セキュリティ ID
```

- 省略時解釈

  セキュリティ ID を設定しません。

[機能]

- -gi オプションは，セキュリティ ID を指定します。

[用途]

- セキュリティ ID を設定したいときに，-gi オプションで指定します。

[説明]

- セキュリティ ID として指定可能な値の範囲は，0H ～ 0FFFFFFFFFFFFFFFFFFFEH【RL78】／ 0H ～ 0FFFFFFFFFFFFFFFFFFFFH【78K0R】です。
- “H”で終わる 16 進数の数値で指定してください。それ以外を指定した場合は，アボート・エラーとなります。
- 10 バイト以内で指定してください。10 バイトに満たない場合は，上位ビットに 0 を補てんします。
- セキュリティ ID は，C4H ～ CDH に設定されます。セキュリティ ID を設定した場合は，C4H ～ CDH へセグメントを配置することはできません。
- セキュリティ ID 機能を持たないデバイスに対して，本オプションを指定した場合は，エラーとなります。
- セキュリティ ID は，アセンブラ・ソース・ファイル中に以下の再配置属性のセグメントを定義することでも指定可能です。ただし，セグメントの再配置属性には，必ず SECUR_ID を指定してください。

```
[ 任意のセグメント名 ]      CSEG    SECUR_ID
                            DB      11H        ; 0xC4 番地
                            DB      22H        ; 0xC5 番地
                            DB      33H        ; 0xC6 番地
                            DB      44H        ; 0xC7 番地
                            DB      55H        ; 0xC8 番地
                            DB      66H        ; 0xC9 番地
                            DB      77H        ; 0xCA 番地
                            DB      88H        ; 0xCB 番地
                            DB      99H        ; 0xCC 番地
                            DB      0AAH       ; 0xCD 番地
```

アセンブラ・ソース・ファイルの指定と本オプションの指定が重なった場合には，本オプションが優先されます。

[注意]

- セキュリティ ID 機能を持ったデバイスに対して，本オプションが指定されていない場合は，任意のコードが配置されることがありますので，注意してください。

[使用例]

- 上記アセンブラ・ソース・ファイルでの指定と同じ“112233445566778899aah”を指定します。

```
C:¥>lk78k0r k0rmain.rel -gi112233445566778899aah
```

ユーザ・オプション・バイト指定

ユーザ・オプション・バイト指定オプションには，次のものがあります。

- -gb

-gb

[記述形式]

```
-gb ユーザ・オプション・バイト値
```

- 省略時解釈
  デバイス・ファイルの初期値

[機能]

- -gb オプションは，ユーザ・オプション・バイトに設定する値を指定します。

[用途]

- ユーザ・オプション・バイトの値を設定したいときに，-gb オプションで指定します。

[説明]

- ユーザ・オプション・バイト値として指定可能な値の範囲は，0 ～ 0FFFFFFH です。
- ユーザ・オプション・バイト値として指定することができない値を指定した場合は，アボート・エラーとなります。
- “H”で終わる 16 進数の数値で指定してください。それ以外を指定した場合は，アボート・エラーとなります。
- ユーザ・オプション・バイトは，C0H ～ C2H 番地に設定されます。
- -gb オプションを指定しなかった場合は，C0 ～ C2H 番地は予約領域とするため，ユーザ・コードを配置することはできません。
- 3 バイト以内で指定してください。3 バイトに満たない場合は，上位ビットに 0 を補てんします。
- C0H ～ C2H 番地のユーザ・オプション・バイト値は，アセンブラ・ソース・ファイル中に以下の再配置属性のセグメントを定義することでも指定可能です。
  ただし，C3H 番地の制御値とあわせて 4 バイトで定義してください。

```
[ 任意のセグメント名 ]      CSEG    OPT_BYTE
                            DB      0FDH    ; 0xC0 番地
                            DB      0FEH    ; 0xC1 番地
                            DB      0FFH    ; 0xC2 番地
                            DB      04H     ; 0xC3 番地
```

アセンブラ・ソース・ファイルの指定と本オプションの指定が重なった場合は，本オプションが優先されます。

- オンチップ・デバッグ・オプション・バイト，ユーザ・オプション・バイトは，必ずデバイスのユーザーズ・マニュアルを参照して設定を行ってください。

［使用例］

- ユーザ・オプション・バイト値として C0H 番地に FDH，C1H 番地に FEH，C2H 番地に FFH を指定します。

```
C:¥>lk78k0r k0rmain.rel -gb0FDFEFFH
```

ミラー領域指定

ミラー領域指定オプションには，次のものがあります。

- -mi

-mi

[記述形式]

```
-mi0，または -mi1
```

- 省略時解釈

  -mi0

[機能]

- -mi オプションは，ミラー元のセグメントの配置先を指定します。

[用途]

- ミラー元のセグメントの配置先を指定したいときに，-mi オプションで指定します。

[説明]

- CSEG MIRRORP の再配置属性のセグメントの配置先をリンカで指定します。
- -mi0 指定時は，MAA に 0 を設定したときのミラー領域にセグメントを配置，-mi1 指定時は，MAA に 1 を設定したときのミラー領域にセグメントを配置します。

  ミラー領域についての詳細は，デバイスのユーザーズ・マニュアルを参照してください。
- パブリック・シンボル“_@MAA”を生成します。このシンボルは，-mi0 指定時は“0”，-mi1 指定時は“1”を値として持つ NUMBER 属性のシンボルです。

[使用例]

- MAA に 1 を設定したときのミラー領域にセグメントを配置します。

```
C:¥>lk78k0r k0rmain.rel -mi1
```

ミラー領域が F1000H 番地～のデバイスの場合，CSEG MIRRORP のセグメントは 11000H 番地から配置されます。

コンパイラが提供するスタートアップ・ルーチンでの使用例は，以下のようになります。

```
MOVW    PMC , #_@MAA
```

この場合，PMC には“1”が格納されます。

64K バイト境界配置指定

64K バイト境界配置指定オプションには，次のものがあります。

- -ccza/-nccza

-ccza/-nccza

[記述形式]

```
-ccza
-nccza
```

- 省略時解釈
  -ccza（入力ファイルがアセンブラ出力ファイルのみの場合）
  -nccza（入力ファイルにコンパイラ出力ファイルがある場合）

[機能]

- -ccza オプションは，各 64K バイト領域の境界の最後の 1 バイト（xFFFFH 注）にセグメントの配置を行うかどうかを指定します。

注　x：0H - EH

[用途]

- 各 64K バイト領域の境界の最後の 1 バイトにセグメントの配置を行うかどうかを指定したいときに，-ccza オプションで指定します。

[説明]

- アセンブラでのみ開発する場合には，自動的に各 64K バイト領域の境界の最後の 1 バイトにも配置を行うため，本オプションを指定する必要はありません。
- コンパイラ出力のオブジェクト・モジュール・ファイルをリンカに入力した場合は，自動的に -nccza と指定されたものとみなし，各 64K バイト領域の境界の最後の 1 バイトに配置を行いません。
- コンパイラにて -za オプションを指定した場合には，各 64K バイト領域の境界の最後の 1 バイトへの配置が可能となるため，-ccza を指定してください。
- 各 64K バイト領域の境界の最後の 1 バイトへの配置についての詳細は，「CubeSuite+ RL78,78K0R コーディング編」を参照してください。

セルフ RAM 領域配置制御

セルフ RAM 領域配置制御オプションには，次のものがあります。

- -self/-selfw

-self/-selfw

[記述形式]

```
-self
-selfw
```

- 省略時解釈

  セルフ RAM 領域は内部 RAM として使用されます。

[機能]

- -self，-selfw オプションは，セルフ RAM 領域への配置制限を行うかどうかを指定します。

[用途]

- セルフ RAM 領域への配置制限を行いたいときに，-self，-selfw オプションで指定します。

[説明]

- -self を指定した場合，セルフ RAM 領域へのセグメント配置は禁止となり，エラーが出力されます。
- -selfw を指定した場合，セルフ RAM 領域へのセグメント配置は可能ですが，ワーニングが出力されます。
- -self，または -selfw オプションを指定した場合，スタック領域は saddr 領域以外に配置されます。

[使用例]

- セルフ RAM 領域へのセグメント配置を禁止し，エラーを出力します。

```
C:¥>lk78k0r k0rmain.rel -self
```

コピー・ルーチン・アドレス指定

コピー・ルーチン・アドレス指定オプションには，次のものがあります。

- -rc

-rc

[記述形式]

```
-rc アドレス
```

- 省略時解釈
  コピー・ルーチンを ROM 領域の空き領域へ配置します。

[機能]

- -rc オプションは，ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。

[用途]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定したいときに，-rc オプションで指定します。

[説明]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。

[使用例]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを 300H 番地に配置します。

```
C:¥>lk78k0r k0rmain.rel -rc300H
```

ROM 化領域指定

ROM 化領域指定オプションには，次のものがあります。

- -ra

-ra

[記述形式]

```
-ra 開始アドレス，終了アドレス
```

- 省略時解釈

  内部 RAM 領域を ROM 化対象とします。

[機能]

- -ra オプションは，ROM 化対象領域を指定します。

[用途]

- ROM 化対象領域を指定したいときに，-ra オプションで指定します。

[説明]

- ROM 化対象領域の開始アドレスと終了アドレスを指定します。

[使用例]

- 0FCF00H 番地から FFFFFH を ROM 化対象とします。

```
C:¥>lk78k0r k0rmain.rel -ra0FCF00H,0FFFFFH
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --

--

[記述形式]

```
--
```

- 省略時解釈

  表示しません。

[機能]

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

[用途]

- ヘルプ・メッセージは，リンク・オプションとその説明の一覧です。リンカを実行するときに参照してください。

[説明]

- -- オプションを指定すると，ほかのリンク・オプションはすべて無効となります。
- ヘルプ・メッセージの続きを読む場合は，リターン・キーを押下してください。表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを押下してください。

注意　本オプションは，CubeSuite+ 上では指定することはできません。

[使用例]

- -- オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。

```
C:¥>lk78k0r --
```

```
78K0R Linker Vx.xx [xx xxx xx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

usage : lk78K0r [option[...]] input-file [option[...]]
```

```
The option is as follows ([] means omissible).
-ffile           :Input option or input-file name from specified file.
-dfile           :Read directive file from specified file.
-bfile           :Read library file from specified file.
-idirectory[,directory...] :Set library file search path.
-o[file]/-no     :Create load module file [with specified name] / Not.
-p[file]/-np     :Create link map file [with specified name] / Not.
-e[file]/-ne     :Create error list file [with specified name] / Not.
-tdirectory      :Set temporary directory.
-km/-nkm         :Output map list to link map file/Not.
-kd/-nkd         :Output directive file image to link map file / Not.
-kp/-nkp         :Output public symbol list to link map file / Not.
-kl/-nkl         :Output local symbol list to link map file / Not.
-ll[page length] :Specify link map file lines per page.
-lf/-nlf         :Add Form Feed at end of the link map file / Not.
-s[memory area]/-ns : Create stack symbol [in specified memory area] / Not.
-g/-ng           :Output symbol information to load module file / Not.
-ydirectory      :Set device file search path.
-j/-nj           :Create load module file if fatal error occurred / Not.
-w[n]            :Change warning level(n=0 to 2).
-zbaddress       :Create Boot file (address:flash start address).
-godata,address[,size] :Change On-Chip Debug Option Bytes, start address, size(size=88 to
1024).
-giid            :Set Security ID.
-gbdata          :Set User Option Bytes.
-mi[0 or 1]      :Select allocation for MIRRORP segment.
-self[-selfw]    :Allocate user code to BRCROSS / Not.
-ccza/-nccza     :Allocate user code to nFFFFH / Not.
-rcaddress       :Specify _rcopy routine allocate address.
-rastart,end     :Specify area for ROM process the start and end.
--               :Show this message.

DEFAULT ASSIGNMENT: -o -p -ne -km -kd -nkp -nkl -ll0 -nlf -ns -g -nj -w1

directive file usage:
 MEMORY memory-area-name:(origin-value, size)[/memory-space-name]
 MERGE segment-name:[location-type-definition][merge-type-definition]
       [=memory-area-name][/memory-space-name]

 example: MEMORY ROM:(0H,4000H)
          MEMORY RAMA:(0FEF00H,100H)
          MERGE CSEG1:=OM
          MERGE DSEG1:AT(0FF000H)
```

B. 3. 5　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”して生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 ROM 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

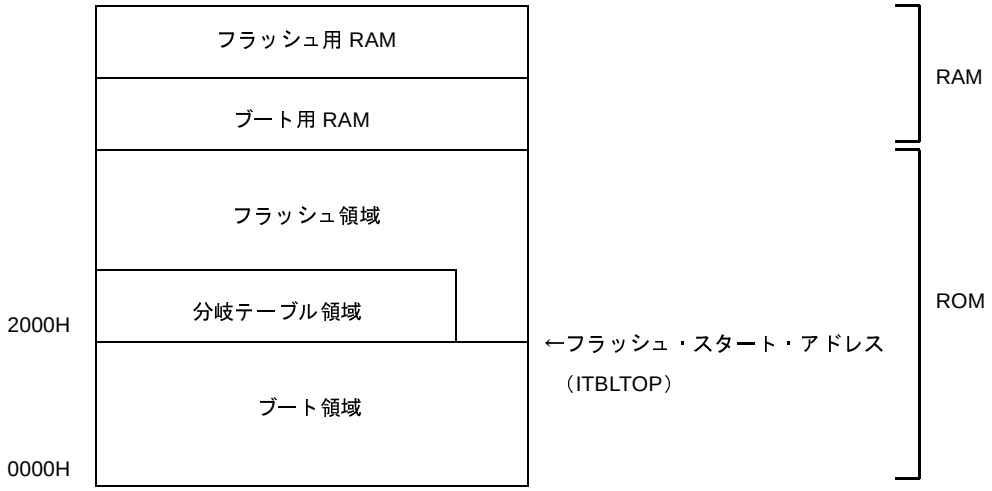
（b）ブート領域から，フラッシュ領域内の関数をコールするとき，いったんフラッシュ領域の分岐テーブルへジャンプし，その後，目的の関数への分岐命令を実行してジャンプする

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。

リセット時の動作は，次のようになります。

RESET 割り込みベクタ（ブート領域）
→ _@cstart（ブート領域）
→ _boot_main 関数（ブート領域）
→ ITBLTOP アドレス（フラッシュ領域）
→ _@cstarte（フラッシュ領域）
→ _main 関数（フラッシュ領域）



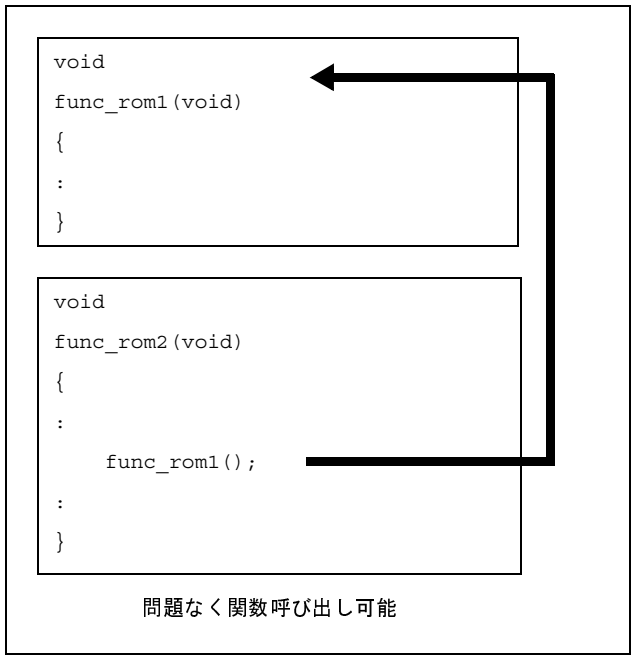
（2）再リンク機能のイメージ

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。

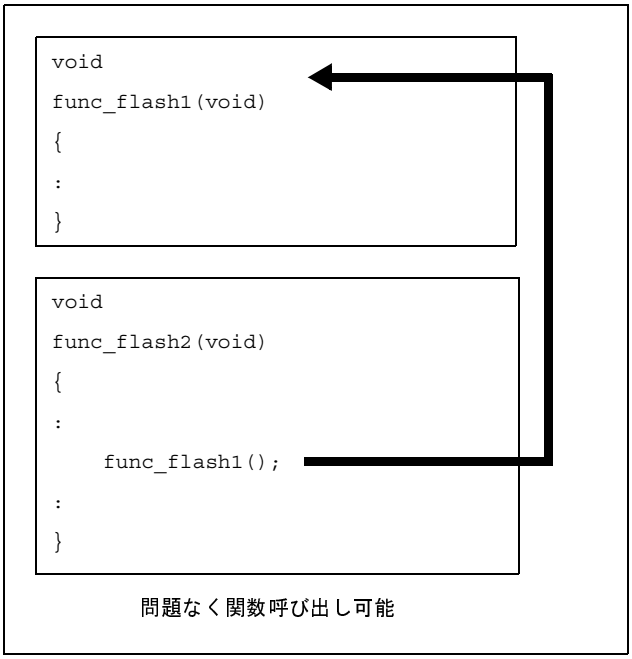
図 B—7　ブート領域内



(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—8　フラッシュ領域内

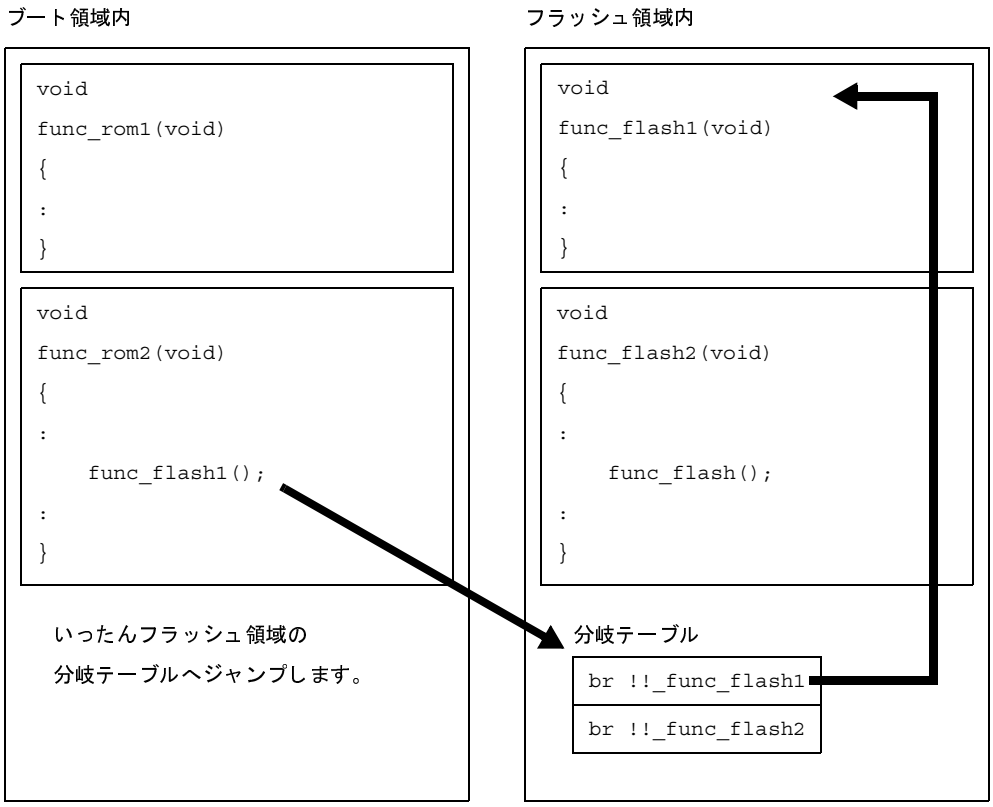


(c) ブート領域内からフラッシュ領域内の関数を呼び出すとき

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—9　ブート領域内からフラッシュ領域内



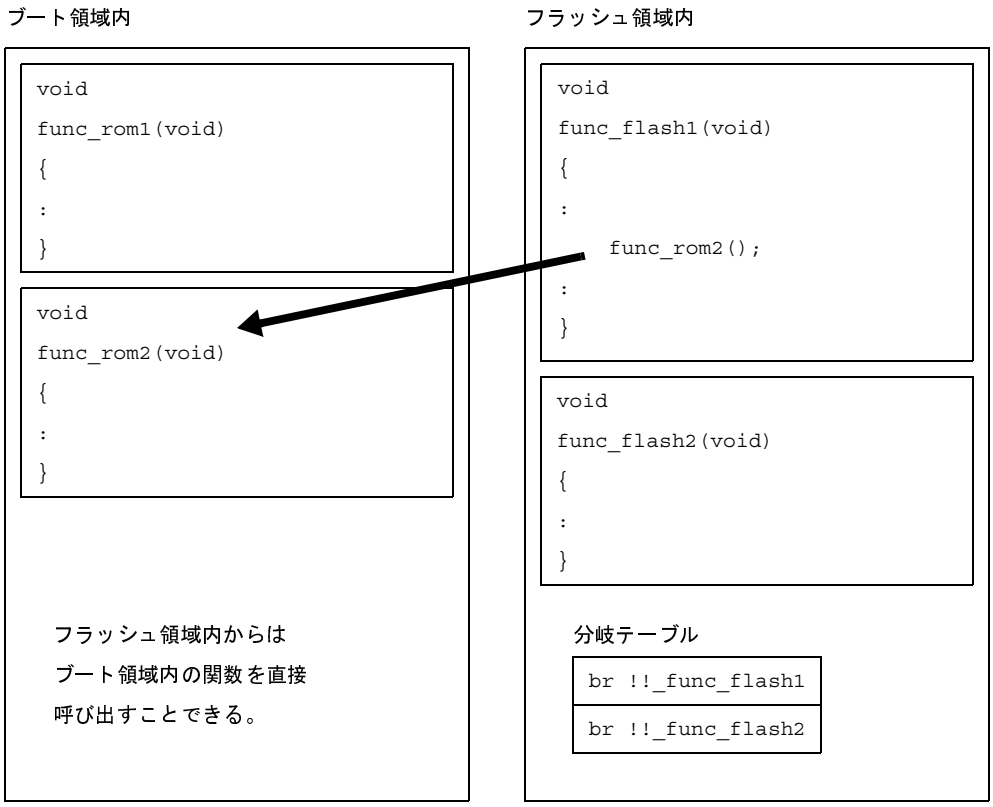
また，関数と同様に，外部変数の参照の可否にも関係します。

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—10　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) #pragma ext_func 指令

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 #pragma ext_func 指令を使って“呼び出す関数名（ラベル名）と ID 番号”を付けます。#pragma ext_func 指令の書式は次のようになります。

```
#pragma 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，#pragma ext_func 指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，分岐テーブルが作成されます。この分岐テーブル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，それぞれのコンパイル・オプション“-zt”で次のように指定します。

```
-zt 分岐テーブルの先頭アドレス
```

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()
```

上記 2 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，ブート領域側の C ソース・ファイルに次のように記述します。

```
#pragma ext_func func_flash0 1
#pragma ext_func func_flash1 2
```

なお，これらの #pragma ext_func 指令群の記述は，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 疑似命令でインクルードすることを推奨します。

再リンク機能のイメージは，次のようになります。

- ブート領域側の C ソース・ファイル

```
#include "ext_def.h"

int boot_a = 0x12;
int boot_b = 0x34;
extern  int func_flash1( int );
extern  int func_flash2( int );

void boot_main( )
{
```

```
        :
}

void func( void )
{
    int k;
    boot_a = func_flash1(boot_a);
    boot_b = func_flash2(boot_b);
}
```

- フラッシュ領域側の C ソース・ファイル

```
#include "ext_def.h"

extern  void func( void );

void main( void )
{
    func();
}
void func_flash1( )
{
        :
}
void func_flash2( )
{
        :
}
```

- ext_def.h

```
#pragma ext_func func_flash1 1
#pragma ext_func func_flash2 2
```

(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。CA78K0R では，ブート領域側，フラッシュ領域側のスタートアップ・ルーチンが提供されています。

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。

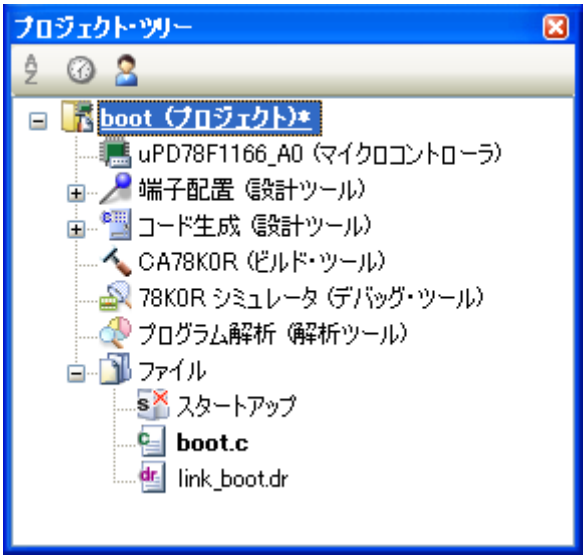
- ブート領域側で使用する RAM 領域を初期化するための処理を行う
- ブート領域側からフラッシュ領域側のスタートアップ・ルーチンへ分岐する
- フラッシュ領域側で使用する RAM 領域を初期化するための処理を行う
- フラッシュ領域側の処理へ移行

(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。

図 B—11　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 パネルで各ビルド・オプションを設定します。

- 変数／関数配置オプションの設定

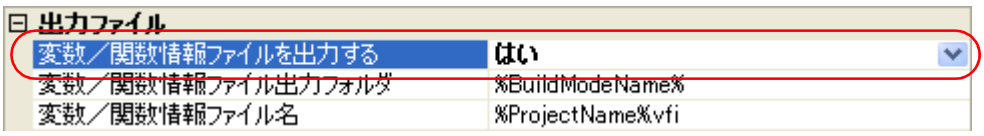
変数／関数情報ファイルを生成して変数や関数の配置を行う場合は，変数／関数配置オプションを設定します。

［変数／関数配置オプション］タブを選択します。

［出力ファイル］カテゴリの［変数／関数情報ファイルを出力する］プロパティで［はい］を選択すると，空の変数／関数情報ファイルを生成し，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，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティで設定されているものとなります。

備考　すでに同名の変数／関数情報ファイルが存在する場合は，ビルド対象に設定します。

図 B—12　ブート領域側の［変数／関数情報ファイルを出力する］プロパティ



変数／関数情報ファイルの出力フォルダ，およびファイル名を変更する場合は，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティを設定してください。［変数／関数情報ファイル名］プロパティを変更すると，空の変数／関数情報ファイルを生成し，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）。

- コンパイル・オプションの設定

［コンパイル・オプション］タブを選択します。

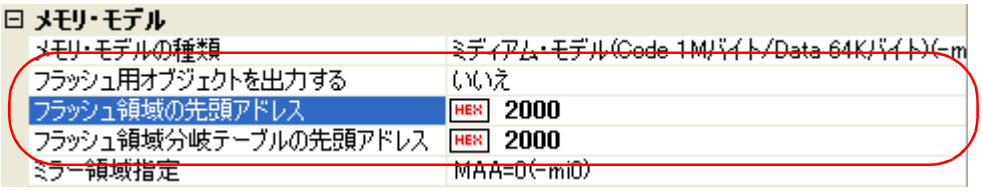
［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］プロパティで［いいえ］を選択します（デフォルト）。

また，［フラッシュ領域の先頭アドレス］プロパティと［フラッシュ領域分岐テーブルの先頭アドレス］プロパティを設定します。

指定可能な値の範囲は，どちらも 0C0 ～ 0EDFFF です。

備考　［フラッシュ領域分岐テーブルの先頭アドレス］プロパティに指定するアドレスは，フラッシュ領域内のアドレスとします。

図 B—13　ブート領域側の［メモリ・モデル］カテゴリ



次に，［スタートアップ］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（ブート領域用）］を選択します。

図 B—14　ブート領域側の［標準のスタートアップを使用する］プロパティ

- リンク・オプションの設定

［リンク・オプション］タブを選択します。

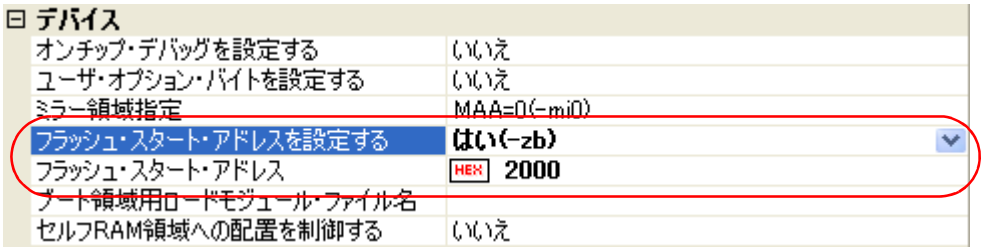
［デバイス］カテゴリの［フラッシュ・スタート・アドレスを設定する］プロパティで［はい (-zb)］を選択すると，［フラッシュ・スタート・アドレス］プロパティが表示されます。

ここで，フラッシュ・メモリ領域の先頭アドレスを指定します。指定可能な値の範囲は，選択したデバイスに依存します。

備考　本プロパティには，［コンパイル・オプション］タブの［メモリ・モデル］カテゴリの［フラッシュ領域の先頭アドレス］プロパティと同じ値が設定されます。

本プロパティを変更すると，［フラッシュ領域の先頭アドレス］プロパティに同じ値を設定します。

図 B—15　ブート領域側の［デバイス］カテゴリ

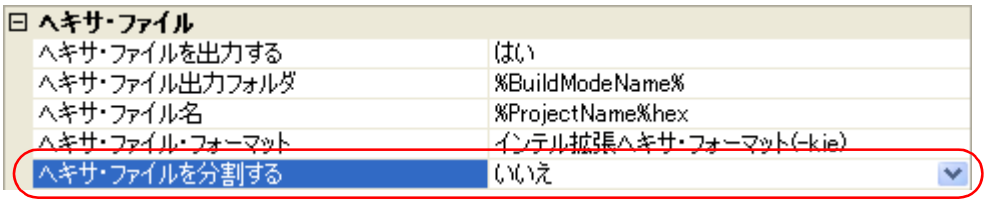


- オブジェクト・コンバート・オプションの設定

［オブジェクト・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］カテゴリの［ヘキサ・ファイルを分割する］プロパティで［いいえ］を選択します（デフォルト）。

図 B—16　ブート領域側の［ヘキサ・ファイルを分割する］プロパティ



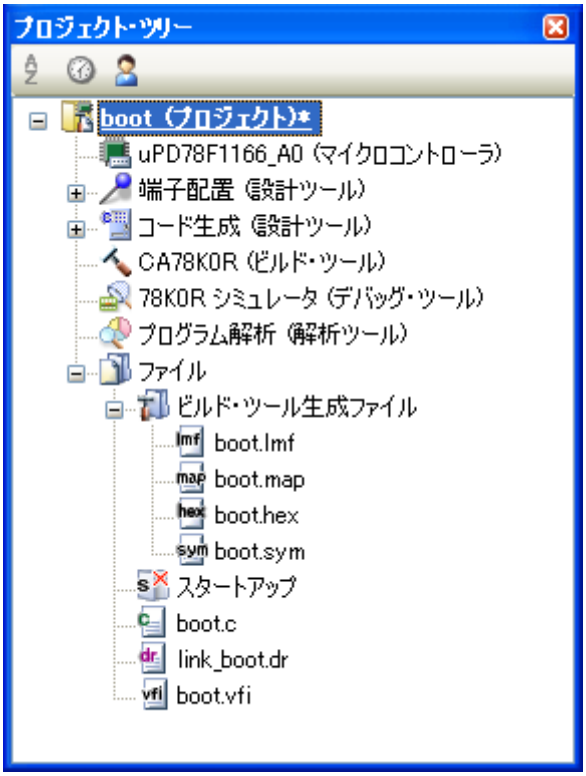
- ブート領域側のプロジェクトのビルドの実行

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。また，ヘキサ・ファイルが生成されます。

変数／関数情報ファイルを生成した場合は，自動的にそれをコンパイラに入力して再度リビルドが実行されます。

備考　ビルドの実行により，「変数／関数配置オプションの設定」で生成した変数／関数情報ファイルを上書きします。

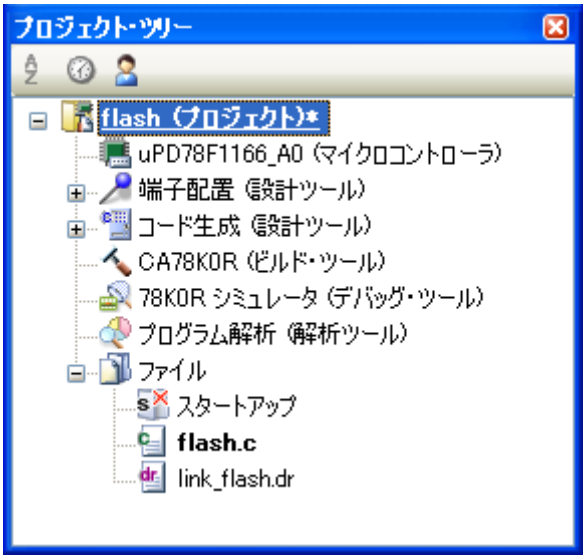
図 B—17　ブート領域側の生成ファイル



- 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。

図 B—18　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 パネルで各ビルド・オプションを設定します。

- 変数／関数配置オプションの設定

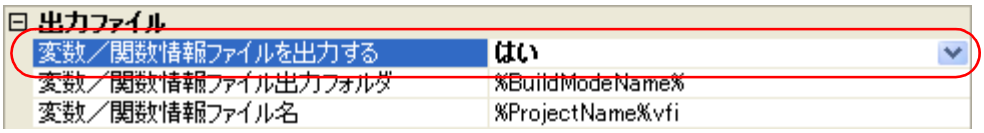
変数／関数情報ファイルを生成して変数や関数の配置を行う場合は，変数／関数配置オプションを設定します。

［変数／関数配置オプション］タブを選択します。

［出力ファイル］カテゴリの［変数／関数情報ファイルを出力する］プロパティで［はい］を選択すると，空の変数／関数情報ファイルを生成し，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，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティで設定されているものとなります。

備考　すでに同名の変数／関数情報ファイルが存在する場合は，ビルド対象に設定します。

図 B—19　フラッシュ領域側の［変数／関数情報ファイルを出力する］プロパティ



変数／関数情報ファイルの出力フォルダ，およびファイル名を変更する場合は，［変数／関数情報ファイル出力フォルダ］プロパティ，および［変数／関数情報ファイル名］プロパティを設定してください。［変数／関数情報ファイル名］プロパティを変更すると，空の変数／関数情報ファイルを生成し，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）。

- コンパイル・オプションの設定

［コンパイル・オプション］タブを選択します。

［メモリ・モデル］カテゴリの［フラッシュ用オブジェクトを出力する］プロパティで［はい (-zf)］を選択します。

また，［フラッシュ領域の先頭アドレス］プロパティと［フラッシュ領域分岐テーブルの先頭アドレス］プロパティを設定します。

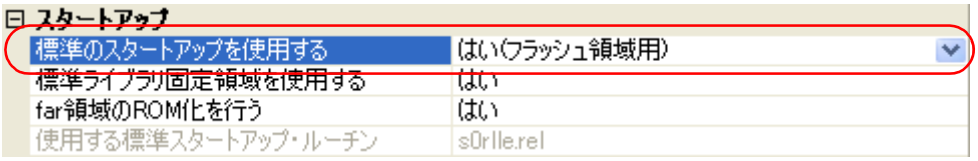
指定可能な値の範囲は，どちらも 0C0 ～ 0EDFFF です。

備考　［フラッシュ領域分岐テーブルの先頭アドレス］プロパティに指定するアドレスは，ブート領域側のプロジェクトで指定したアドレスと同じものとします。

図 B—20　フラッシュ領域側の［メモリ・モデル］カテゴリ

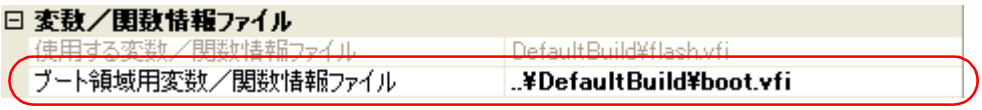
次に，［スタートアップ］カテゴリの［標準のスタートアップを使用する］プロパティで［はい（フラッシュ領域用）］を選択します。

図 B—21　フラッシュ領域側の［標準のスタートアップを使用する］プロパティ



次に，作成したブート領域側の変数／関数情報ファイルをフラッシュ領域側のプロジェクトに追加します。［変数／関数情報ファイル］カテゴリの［ブート領域用変数／関数情報ファイル］プロパティでブート領域側の変数／関数情報ファイルを指定します。

図 B—22　フラッシュ領域側の［ブート領域用変数／関数情報ファイル］プロパティ

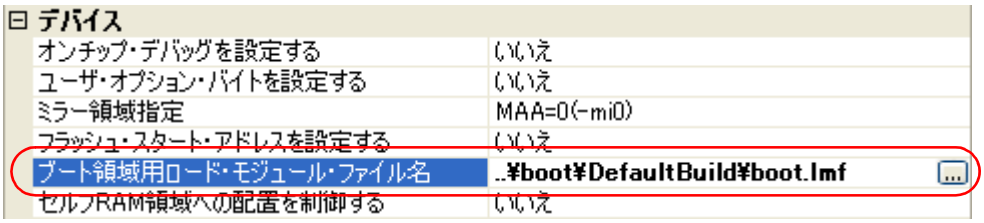


- リンク・オプションの設定

作成したブート領域側のロード・モジュール・ファイルをフラッシュ領域側のプロジェクトに追加します。［リンク・オプション］タブを選択します。

［デバイス］カテゴリの［ブート領域用ロード・モジュール・ファイル名］プロパティに，ブート領域側のロード・モジュール・ファイルを指定します。

図 B—23　フラッシュ領域側の［ブート領域用ロード・モジュール・ファイル名］プロパティ

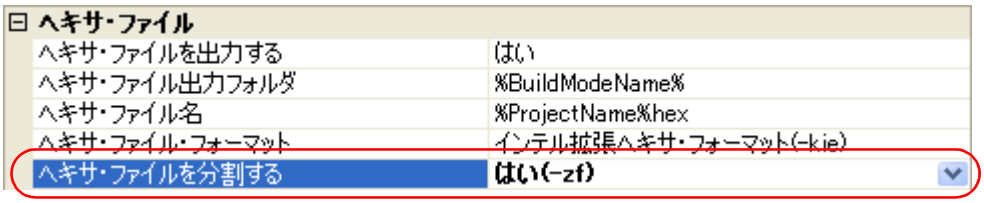


- オブジェクト・コンバート・オプションの設定

［オブジェクト・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］カテゴリの［ヘキサ・ファイルを分割する］プロパティで［はい (-zf)］を選択します。

図 B—24　フラッシュ領域側の［ヘキサ・ファイルを分割する］プロパティ

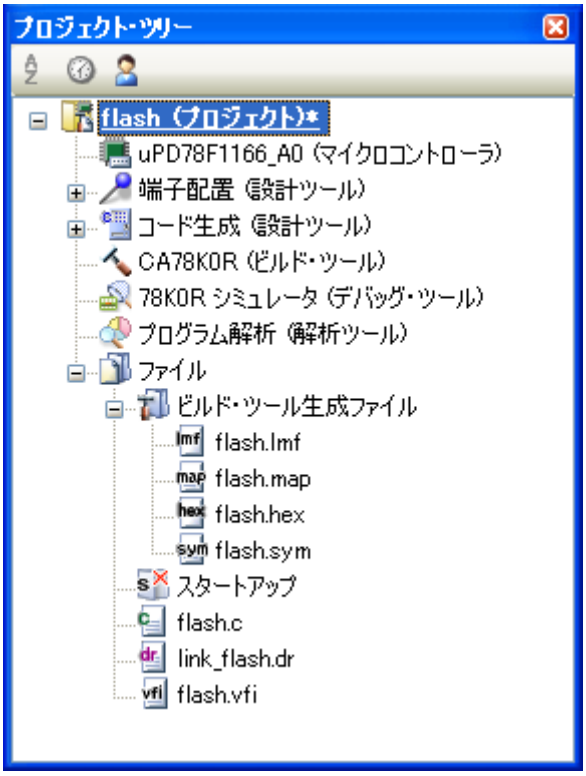


- フラッシュ領域側のプロジェクトのビルドの実行

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。

また，ブート領域用のヘキサ・ファイル（ブート領域側のプロジェクトのビルドの実行により生成されたファイルと同じ内容となります）とフラッシュ領域用のヘキサ・ファイルが生成されます。

図 B—25　フラッシュ領域側の生成ファイル



(e) 分岐テーブルのアドレスの変更方法

分岐テーブルの先頭アドレスを 2000H 以外に設定する場合は，次のようにして割込みベクタの処理も変更してください。

- vect.inc の“ITBLTOP EQU 2000H”のアドレス値を変更します。

  vect.inc のデフォルトのインストール先は，以下です。

  C:¥Program Files¥Renesas Electronics¥CubeSuite+¥CA78K0R¥Vx.xx¥src¥cc78k0r¥src

- ..¥bat¥repvect.bat

  ..¥bat¥mkstup.bat

  を DOS プロンプト上で起動してスタートアップ，ライブラリを更新し，..¥..¥..¥lib78k0r にコピーしてリンク用に使用します。

(f) リンク・ディレクティブ・ファイルの記述

リンク・ディレクティブ・ファイルを使用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すると，リンカ等でエラーを出力します。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。
- 分岐テーブルに関するリンク・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。

  - -zt 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。
  - -zt 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。

(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，#pragma interrupt 指令で関数指定する必要があります。

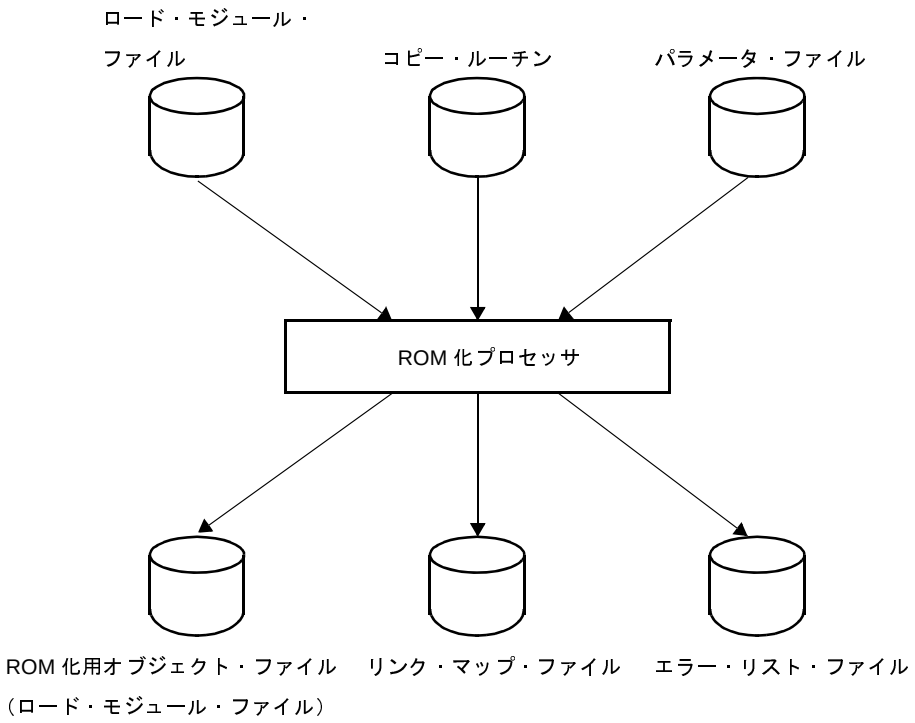
- 割り込みハンドラ・アドレスは“ブート領域側”
- 割り込みハンドラ本体は“フラッシュ領域側”

B. 4　ROM 化プロセッサ

ROM 化プロセッサは，リンクが正常に終了したロード・モジュール・ファイルを入力し，ROM 化を行ったロード・モジュール・ファイルを出力します。

C ソースを変更することなく，プログラムの一部を RAM 領域に配置して実行することができます。

図 B—26　ROM 化プロセッサの入出力ファイル



B. 4. 1　入出力ファイル

ROM 化プロセッサの入出力ファイルを次に示します。

表 B—11　ROM 化プロセッサの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	ロード・モジュール・ファイル	- リンク後の ROM 化対象ファイル リンクが正常に終了したファイルを入力します。	.lmf
	コピー・ルーチン （“_rcopy.rel”固定）	- コピー・ルーチン ROM 化プロセッサが自動的にリンクします。	—
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイル（ユーザ作成ファイル）	.ppr

種類	ファイル名	説明	デフォルト・ファイル・タイプ
出力ファイル	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）	- ROM 化を行ったファイル デフォルトのファイル名は“*入力ファイルのプライマリ・ネーム* _orig.lmf”です。 同名のファイル名が存在する場合は，上書きします。	.lmf
	リンク・マップ・ファイル	- ROM 化情報を含むマップ・ファイル デフォルトのファイル名は“*入力ファイルのプライマリ・ネーム* _romp.map”です。 同名のファイル名が存在する場合は，上書きします。	.map
	エラー・リスト・ファイル	- ROM 化時のエラー情報を持つファイル	.erp

B. 4. 2　機　　能

（1）コピー・ルーチン

_rcopy.rel からコピー・ルーチンを読み込んで，出力ファイルにリンクします。

配置アドレスはリンカにより決定しますが，オプションでアドレスを指定することも可能です。

（2）ROM 化対象領域

ROM 化対象領域は，デバイス・ファイルにて定義された内部 RAM 領域となります。

ただし，オプションで対象領域を指定することも可能です。

（3）生成情報

入力ファイルに，@@ROMP セグメントとして，コピー・ルーチン／ ROM 化対象セグメントのアドレス情報を追加し，ROM 化対象セグメントのロウデータの所属セグメントをリンク時のセグメントから @@ROMP セグメントへ移動を行ったものを，出力ファイルとして生成します。

備考　@@ROMP は予約語です。

なお，このセグメント名は変更できません。

B. 4. 3　操作方法

(1) ROM 化プロセッサの起動方法

ROM 化プロセッサの起動には，2 つの方法があります。

(a) コマンド行での起動

```
X:[パス名]>rp78k0r[△オプション]...ロード・モジュール・ファイル名
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
rp78k0r	ROM 化プロセッサのコマンド名
オプション	ROM 化プロセッサに対して動作の詳細を指示します。 複数の ROM 化プロセス・オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，ROM 化プロセス・オプションに大文字，小文字の区別はありません。ROM 化プロセス・オプションについての詳細は，「B. 4. 4 オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（”“）で囲んでください。
ロード・モジュール・ファイル名	ROM 化するロード・モジュール・ファイル名 入力モジュールとして，最大 1024 個入力できます。 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（”“）で囲んでください。

例　リンク・マップ・ファイル k0r.map を出力します。

```
C:¥>rp78k0r a.lmf -pk0r.map
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，ROM 化するたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>rp78k0r[△ロード・モジュール・ファイル]△-f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	ROM 化プロセッサの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[△] オプション [△オプション]...
```

- コマンド行でソース・ファイル名を省略した場合は，パラメータ・ファイル内でソース・ファイル名を１つだけ指定することができます。
- ソース・ファイル名は，オプションのあとに記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定するすべての ROM 化プロセス・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル k0r.ppr をエディタで作成し，ROM 化プロセッサを起動します。

```
; parameter file
a.lmf -ok0r.lmf -pk0r.map -e
-tC:¥tmp
```

```
C:¥>rp78k0r -fk0r.ppr
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

ROM 化プロセッサが起動すると，次の実行開始メッセージが表示されます。

```
78K0R ROM Processor Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
```

(b) 実行終了メッセージ

ROM 化の結果，ROM 化エラーが検出されなかった場合，ROM 化プロセッサは次のメッセージを表示して制御をホスト OS に戻します。

```
Target chip : uPD78xxx
Device file : Vx.xx

Link complete, 0 error(s) and 0 warning(s) found.
```

ROM 化の結果，ROM 化エラーが検出された場合，ROM 化プロセッサはエラーとワーニングの数を表示して制御をホスト OS に戻します。

```
Target chip : uPD78xxx
Device file : Vx.xx

Link complete, 1 error(s) and 0 warning(s) found.
```

ROM 化中に ROM 化プロセッサの処理継続が不可能な致命的エラーが検出された場合，ROM 化プロセッサはメッセージを表示して ROM 化を中止し，制御をホスト OS に戻します。

- 存在しないロード・モジュール・ファイルを指定した場合

```
C:¥>rp78k0r samp.lmf
```

```
78K0R ROM Processor Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RP78K0R error F8006 : File not found 'samp.lmf'
Program Aborted.
```

この例では，存在しないロード・モジュール・ファイルを指定したためにエラーとなり，ROM 化が中止されます。

- 存在しない ROM 化プロセス・オプションを指定した場合

```
C:¥>rp78k0r a.lmf -z
```

```
78K0R ROM Processor Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RP78K0R error F8018 : Option is not recognized '-z'
Please enter 'RP78K0R --' , if you want help messages.
Program Aborted.
```

この例では，存在しない ROM 化プロセス・オプションを指定したためにエラーとなり，ROM 化が中止されます。

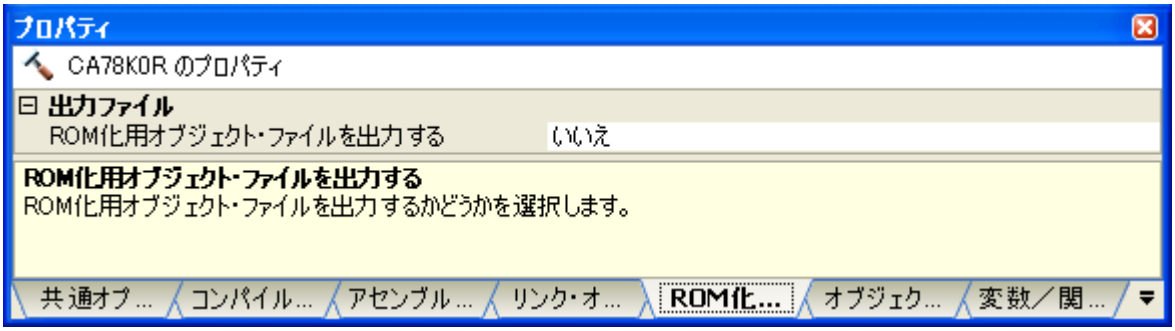
(3) CubeSuite でのオプション設定

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—27　プロパティ パネル：［ROM 化プロセス・オプション］タブ



（4）コピー関数の使用例

注意　78K0R と RL78 ではセルフ・プログラミングの仕様が異なりますので注意してください。

（a）以下のような C ソースとリンク・ディレクティブで，main 関数を ROM 領域に，self_init 関数を RAM_CODE 領域に配置するとします。
RAM 領域で実行する self_init 関数が呼ばれる前に _rcopy 関数を呼び出してください。

- main.c

```
int _rcopy(int);
void main()
{
    int ret;
    ret = _rcopy(1);
      :
    self_init();
      :
}
```

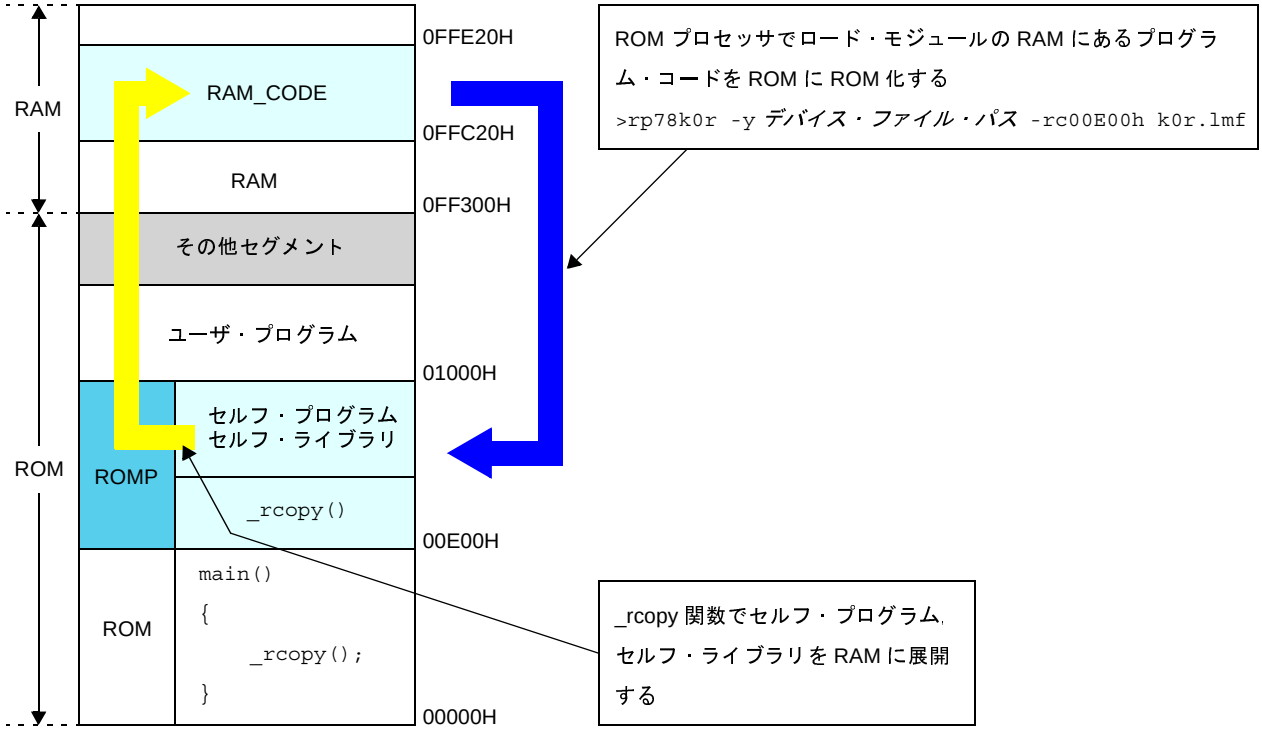
- self_tmp.c

```
#pragma section @@CODEL PROG_RAM
void self_init()
{
      :
}
```

- 78k0r.dr

```
memory ROM       :(000000H, 000E00H)
memory ROMP      :(000E00H, 000200H)
memory RAM       :(0FF300H, 00920H)
memory RAM_CODE  :(0FFC20H, 00200H)
;
merge PROG_RAM   := RAM_CODE
merge @@LCODE    := RAM_CODE
merge @@FSL_RCD  := RAM_CODE
```

図 B—28　マップ・イメージ図



(b)（a）のイメージでビルドして作成したロード・モジュール・ファイルの RAM 領域にあるプログラム・コード領域を ROM 化プロセッサで ROM 化します。

例では，_rcopy 関数と領域名“PROG_RAM”のプログラム・コードが 0E00H 番地に ROM 化されます。

コマンドを以下に示します。

```
>rp78k0r -y デバイス・ファイル・パス -rc00E00h k0r.lmf
```

B. 4. 4　オプション

（1）種類

ROM 化プロセス・オプションは ROM 化プロセッサの動作に細かい指示を与えるものです。

ROM 化プロセス・オプションの分類と説明を示します。

表 B—12　ROM 化プロセス・オプション

分類	オプション	説明
ロード・モジュール・ファイル出力指定	-o	ロード・モジュール・ファイルの出力を指定します。
	-no	
リンク・マップ・ファイル出力指定	-p	リンク・マップ・ファイルの出力を指定します。
	-np	
リンク・マップ・ファイル情報指定	-km	リンク・マップ・ファイル中に，マップ・リストを出力します。
	-nkm	
	-kp	リンク・マップ・ファイル中に，パブリック・シンボル・リストを出力します。
	-nkp	
	-kl	リンク・マップ・ファイル中に，ローカル・シンボル・リストを出力します。
	-nkl	
リンク・マップ・ファイル形式指定	-ll	リストの 1 頁に印字する行数を変更します。
	-lf	リンク・マップ・ファイルの最後に改頁コードを付加します。
	-nlf	
エラー・リスト・ファイル出力指定	-e	エラー・リスト・ファイルを出力します。
	-ne	
パラメータ・ファイル指定	-f	入力ファイル名，オプションを指定したファイルより入力します。
デバイス・ファイル・サーチ・パス指定	-y	デバイス・ファイルを指定されたパスから読み込みます。
コピー・ルーチン・アドレス指定	-rc	ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。
ROM 化領域指定	-ra	ROM 化対象領域を指定します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

（2）優先度

次の表に示す ROM 化プロセス・オプションのうち，縦軸のものと横軸のものを同時に 2 つ以上指定した場合の優先度について説明します。

表 B—13　ROM 化プロセス・オプションの優先度

	-no	-np	-nkm	-nkp	-nkl	--
-p			△	△	△	×
-km		×				×
-kp		×				×
-kl		×				×
-ll		×				×
-lf		×				×

- ×で記した箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例　-km オプションは，無効となります。

```
C:¥>rp78k0r a.lmf -np -km
```

- △で記した箇所

横軸に示したオプション 3 つをすべて同時に指定した場合，縦軸に示したオプションは無効となります。

例　-nkm，-nkp，および -nkl を同時に指定したので，-p オプションは無効となります。

```
C:¥>rp78k0r a.lmf -p -nkm -nkp -nkl
```

- 空欄の箇所

横軸に示したオプションを指定した場合，縦軸に示したオプションは有効となります。

また，-o/-no オプションのように，オプション名の前に“n”を付加できるオプションを同時に指定した場合，あとで指定した方が有効となります。

例　-no オプションがあとに指定されているので，-o オプションは無効となり，-no オプションが有効となります。

```
C:¥>rp78k0r a.lmf -o -no
```

「表 B—13　ROM 化プロセス・オプションの優先度」に記述されていないオプションは，ほかのオプションの影響を特に受けません。しかし，ヘルプ指定オプション（ -- ）が指定された場合には，すべてのオプション指定が無効となります。

ロード・モジュール・ファイル出力指定

ロード・モジュール・ファイル出力指定オプションには，次のものがあります。

- -o/-no

-o/-no

[記述形式]

```
-o [ 出力ファイル名 ]
-no
```

- 省略時解釈

  -o *入力ファイル名* .lmf

[機能]

- -o オプションは，ロード・モジュール・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -no オプションは，-o，-j，-g オプションを無効にします。

[用途]

- ロード・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，-o オプションを指定します。
- リンク・マップ・ファイルの出力のみが目的で ROM 化する場合などに，-no オプションを指定します。これにより，ROM 化時間が短縮されます。

[説明]

- -o オプションを指定してもフェイタル・エラーがある場合は，ロード・モジュール・ファイルは出力されません。
- -o オプションを指定する際に“*出力ファイル名*”を省略すると，カレント・フォルダにロード・モジュール・ファイル“*入力ファイル名* .lmf ”が出力されます。
  入力ファイル名は，“*入力ファイル名* _orig.lmf”に変更します。
- “*出力ファイル名*”にパス名のみを指定すると，指定したパスに“*入力ファイル名* .lmf”が出力されます。
- -o と -no の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

［使用例］

- ロード・モジュール・ファイル k0r.lmf を出力します。

```
C:¥>rp78k0r a.lmf -ok0r.lmf
```

リンク・マップ・ファイル出力指定

リンク・マップ・ファイル出力指定オプションには，次のものがあります。

- -p/-np

-p/-np

[記述形式]

```
-p[出力ファイル名]
-np
```

- 省略時解釈

  -p *入力ファイル名* _romp.map

[機能]

- -p オプションは，リンク・マップ・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -np オプションは，-p，-km，-kd，-kp，-kl，-ll，-lf オプションを無効にします。

[用途]

- リンク・マップ・ファイルの出力先や出力ファイル名を変更する場合は，-p オプションを指定します。
- ロード・モジュール・ファイルの出力だけを目的で ROM 化する場合などに，-np オプションを指定します。これにより，ROM 化時間が短縮されます。

[説明]

- -p オプションを指定する際に“*出力ファイル名*”を省略すると，カレント・フォルダにリンク・マップ・ファイル“*入力ファイル名* _romp.map”を出力します。
- “出力ファイル名”にパス名のみを指定すると，指定したパスに“*入力ファイル名* _romp.map”を出力します。
- -p と -np の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- リンク・マップ・ファイル k0r.map を作成します。

```
C:¥>rp78k0r a.lmf -pk0r.map
```

リンク・マップ・ファイル情報指定

リンク・マップ・ファイル情報指定オプションには，次のものがあります。

- -km/-nkm
- -kp/-nkp
- -kl/-nkl

-km/-nkm

[記述形式]

```
-km
-nkm
```

- 省略時解釈

  -km

[機能]

- -km オプションは，リンク・マップ・ファイル中にマップ・リストを出力します。
- -nkm オプションは，-km，-kd オプションを無効にします。

[用途]

- リンク・マップ・ファイル中にマップ・リストを出力したいときに，-km オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションをすべて指定した場合は，リンク・マップ・ファイルは出力されません。
- -nkm オプションを指定した場合は，リンク・マップ・ファイル中にリンク・ディレクティブ・ファイルは出力されません。
- -km と -nkm の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-km オプションは無効となります。

[使用例]

- リンク・マップ・ファイル a_romp.map にマップ・リストを出力します。

```
C:¥>rp78k0r a.lmf -km
```

a_romp.map の内容は，以下のようになります。

```
78K0R ROM Processor Vx.xx                                    Date:xx xxx xxxx  Page:   1

Command:    a.lmf -rc300h -km
Para-file:
Out-file:   a.lmf
Map-file:   a_romp.map


*** Link information ***

     6 output segment(s)
   F7H byte(s) real data
    78 symbol(s) defined


*** Memory map ***


  SPACE=REGULAR

         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         00000H    00002H  CSEG AT
                  CODE     a.lmf
                                       00000H    00002H
* gap *                               00002H    000BEH
         ?CSEGOB0                     000C0H    00004H  CSEG AT
                  ?CSEGOB0 a.lmf
                                       000C0H    00004H
         ?CSEG                        000C4H    00061H  CSEG AT
                  ?CSEG    a.lmf
                                       000C4H    00061H
* gap *                               00125H    001DBH
         @@ROMP                       00300H    00090H  CSEG AT
                  _@ROMP   _rcopy      00300H    0008EH
                  PRO_RAM  a.lmf
                                       0038EH    00002H
* gap *                               00390H    3FC70H

         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                               FCF00H    02F20H
         DATA                         FFE20H    00003H  DSEG AT
                  DATA     a.lmf
                                       FFE20H    00003H
```

```
* gap *                                FFE23H    0000DH
         PRO_RAM                       FFE30H    00002H  CSEG AT
                  PRO_RAM  a.lmf
                                       FFE30H    00002H
* gap *                                FFE32H    000CEH
* gap (Not Free Area) *                FFF00H    00100H


*** Segment number ***


         SEGMENT NAME                            SEGMENT NUMBER
         PRO_RAM                                 1


 Target chip : uPD78xxx
 Device file : Vx.xx
```

-kp/-nkp

[記述形式]

```
-kp
-nkp
```

- 省略時解釈
  -nkp

[機能]

- -kp オプションは，リンク・マップ・ファイル中にパブリック・シンボル・リストを出力します。
- -nkp オプションは，-kp オプションを無効にします。

[用途]

- リンク・マップ・ファイル中にパブリック・シンボル・リストを出力する場合に，-kp オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションをすべて指定した場合は，リンク・マップ・ファイルは出力されません。
- -ng オプションを指定した場合は，パブリック・シンボル・リストは出力されません。
- -kp と -nkp の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-kp オプションは無効となります。

[使用例]

- リンク・マップ・ファイル a_romp.map に，パブリック・シンボル・リストを出力します。

```
C:¥>rp78k0r a.lmf -kp
```

a_romp.map の内容は，以下のようになります。

```
78K0R ROM Processor Vx.xx                                    Date:xx xxx xxxx  Page:   1

Command:    a.lmf -rc300h -km
Para-file:
Out-file:   a.lmf
Map-file:   a_romp.map

*** Link information ***

     6 output segment(s)
   F7H byte(s) real data
    78 symbol(s) defined

*** Memory map ***

  SPACE=REGULAR

        :

*** Public symbol list ***

MODULE    ATTR       VALUE    NAME

k0rram
          ADDR       00000H   MAIN
          ADDR       000C4H   START
          ADDR       000E3H   CONVAH
          NUM        FFE20H   _@STBEG
          NUM        00300H   __rcopy
          ADDR       FFE30H   pro_ram
          NUM        FCF00H   _@STEND
          NUM        00000H   _@MAA
_rcopy
          ADDR       00300H   _@RCP
          NUM        0038EH   _S_romp


 Target chip : uPD78xxx
 Device file : Vx.xx
```

-kl/-nkl

[記述形式]

```
-kl
-nkl
```

- 省略時解釈
  -nkl

[機能]

- -kl オプションは，リンク・マップ・ファイル中にローカル・シンボル・リストを出力します。
- -nkl オプションは，-kl オプションを無効にします。

[用途]

- リンク・マップ・ファイル中にローカル・シンボル・リストを出力する場合に，-kl オプションを指定します。

[説明]

- -nkm，-nkp，および -nkl オプションがすべて指定された場合は，リンク・マップ・ファイルは出力されません。
- -ng オプションを指定した場合は，ローカル・シンボル・リストは出力されません。
- -kl と -nkl の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -np オプションを指定した場合は，-kl オプションは無効となります。

[使用例]

- リンク・マップ・ファイル a_romp.map に，ローカル・シンボル・リストを出力します。

```
C:¥>rp78k0r a.lmf -kl
```

a_romp.map の内容は，以下のようになります。

```
78K0R ROM Processor Vx.xx                                    Date:xx xxx xxxx  Page:   1

Command:    a.lmf -rc300h -km
Para-file:
Out-file:   a.lmf
Map-file:   a_romp.map


*** Link information ***

     6 output segment(s)
   F7H byte(s) real data
    78 symbol(s) defined


*** Memory map ***


  SPACE=REGULAR

        :

*** Local symbol list ***

MODULE    ATTR       VALUE    NAME

k0rram
          MOD                 SAMPM
          DSEG                DATA
          ADDR       FFE20H   HDTSA
          ADDR       FFE21H   STASC
          CSEG                CODE
          CSEG                ?CSEG
k0rram
          MOD                 SAMPS
          CSEG                ?CSEG
          ADDR       0011CH   SASC
          ADDR       00122H   SASC1
k0rram
          MOD                 k0rram
          CSEG                PRO_RAM
_rcopy
          MOD                 _rcopy
                     00300H   _@ROMP
          ADDR       00384H   ?L_RCPC
```

```
        ADDR        0037BH   ?L_RCP9
        ADDR        00386H   ?L_RCPA
        ADDR        00318H   ?L_RCP1
        ADDR        0032CH   ?L_RCP2
        ADDR        0033BH   ?L_RCP3
        ADDR        0034AH   ?L_RCP4
        ADDR        00351H   ?L_RCP5
        ADDR        00360H   ?L_RCP6
        ADDR        00372H   ?L_RCP7
        ADDR        00372H   ?L_RCP8


Target chip : uPD78xxx
Device file : Vx.xx
```

リンク・マップ・ファイル形式指定

リンク・マップ・ファイル形式指定オプションには，次のものがあります。

- -ll
- -lf/-nlf

-ll

[記述形式]

```
-ll[行数]
```

- 省略時解釈

  -ll0（改頁しません）

[機能]

- -ll オプションは，リンク・マップ・ファイルの 1 頁の行数を指定します。

[用途]

- リンク・マップ・ファイルの 1 頁の行数を変更したいときに，-ll オプションで指定します。

[説明]

- -ll オプションで指定可能な行数の範囲は，20 ～ 32767 です。
- 範囲外の数値や数値以外のものが指定された場合には，アボート・エラーとなります。
- 行数を省略した場合，0 を指定したものとみなされます。
- 行数に 0 を指定した場合は，改頁しません。
- -np オプションを指定した場合は，-ll オプションは無効となります。

[使用例]

- リンク・マップ・ファイル a_romp.map の 1 頁の行数を 20 行に指定します。

```
C:¥>rp78k0r a.lmf -ll20
```

a_romp.map の内容は，以下のようになります。

```
78K0R ROM Processor Vx.xx                                    Date:xx xxx xxxx  Page:   1

Command:    a.lmf -rc300h -km
Para-file:
Out-file:   a.lmf
Map-file:   a_romp.map


*** Link information ***

     6 output segment(s)
   F7H byte(s) real data
    78 symbol(s) defined


*** Memory map ***

-------------------------------------------------------------------------------------


78K0R ROM Processor Vx.xx                                    Date:xx xxx xxxx  Page:   2


  SPACE=REGULAR

         OUTPUT   INPUT    INPUT      BASE      SIZE
         SEGMENT  SEGMENT  MODULE     ADDRESS
         CODE                         00000H    00002H  CSEG AT
                  CODE     a.lmf
                                      00000H    00002H
* gap *                               00002H    000BEH
         ?CSEGOB0                     000C0H    00004H  CSEG AT
                  ?CSEGOB0 a.lmf
                                      000C0H    00004H
         ?CSEG                        000C4H    00061H  CSEG AT
                  ?CSEG    a.lmf
                                      000C4H    00061H
* gap *                               00125H    001DBH
         @@ROMP                       00300H    00090H  CSEG AT
                  _@ROMP   _rcopy     00300H    0008EH
-------------------------------------------------------------------------------------


78K0R ROM Processor Vx.xx                                    Date:xx xxx xxxx  Page:   3
```

```
                 PRO_RAM  a.lmf
                                     0038EH    00002H
* gap *                              00390H    3FC70H

        OUTPUT   INPUT    INPUT      BASE      SIZE
        SEGMENT  SEGMENT  MODULE     ADDRESS
* gap *                              FCF00H    02F20H
        DATA                        FFE20H    00003H  DSEG AT
                 DATA     a.lmf
                                     FFE20H    00003H
* gap *                              FFE23H    0000DH
        PRO_RAM                     FFE30H    00002H  CSEG AT
                 PRO_RAM  a.lmf
                                     FFE30H    00002H
* gap *                              FFE32H    000CEH
-------------------------------------------------------------------------------------


78K0R ROM Processor Vx.xx                                Date:xx xxx xxxx  Page:   4


* gap (Not Free Area) *              FFF00H    00100H


*** Segment number ***


        SEGMENT NAME                            SEGMENT NUMBER
        PRO_RAM                                 1


 Target chip : uPD78xxx
 Device file : Vx.xx
```

-lf/-nlf

[記述形式]

```
-lf
-nlf
```

- 省略時解釈
  -nlf

[機能]

- -lf オプションは，リンク・マップ・ファイルの最後に，改頁コード（FF）を付加することを指定します。
- -nlf オプションは，-lf オプションを無効にします。

[用途]

- リンク・マップ・ファイルの内容を印字したあとで改頁しておきたい場合，-lf オプションを指定して改頁コードを付加します。

[説明]

- -np オプションを指定した場合は，-lf オプションは無効となります。
- -lf と -nlf の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- リンク・マップ・ファイル k0r.map の最後に改頁コードを付加します。

```
C:¥>rp78k0r a.lmf -pk0r.map -lf
```

エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定オプションには，次のものがあります。

- -e/-ne

-e/-ne

[記述形式]

```
-e[ ファイル名 ]
-ne
```

- 省略時解釈

  -ne

[機能]

- -e オプションは，エラー・リスト・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -ne オプションは，-e オプションを無効にします。

[用途]

- エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-e オプションを指定します。

[説明]

- -e オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は“*入力ファイル名*.elk”となります。
- -e オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- -e と -ne の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- エラー・リスト・ファイル k0r.elk を作成します。

```
C:¥>rp78k0r a.lmf -ek0r.elk
```

リンク・ディレクティブ・ファイル k0r.dr の内容に誤りがありました。

エラー・リスト・ファイル k0r.elk の内容は，以下のようになります。

```
k0r.dr(3) : RA78K0R error E3102: Directive syntax error
```

パラメータ・ファイル指定

パラメータ・ファイル指定オプションには，次のものがあります。

- -f

-f

[記述形式]

```
-f ファイル名
```

- 省略時解釈
  コマンド行上からのみオプション，入力ファイル名の入力が可能となります。

[機能]

- -f オプションは，オプション，あるいは入力ファイル名を指定のファイルから入力することを指定します。

[用途]

- コマンド行では，ROM 化プロセッサの起動に必要な情報を指定しきれないときに，-f オプションを指定します。
- ROM 化するたび繰り返し同じようにオプションを指定する場合には，それらをパラメータ・ファイルに記述しておき，-f オプションで指定します。

[説明]

- ファイル名を省略すると，アボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で -f オプションを指定すると，アボート・エラーとなります。
- パラメータ・ファイル中に記述可能な文字数に，制限はありません。
- 空白とタブ，および改行文字（LF）をオプション，あるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，あるいは入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- “；”，または“#”以降に記述された文字は，改行文字（LF），または EOF の前まですべてコメントと解釈します。
- -f オプションを複数指定すると，アボート・エラーとなります。

［使用例］

- パラメータ・ファイル k0r.ppr を使用してリンクします。

パラメータ・ファイル k0r.ppr の内容は，以下のようになります。

```
; parameter file
a.lmf -ok0r.lmf -pk0r.map -e
-tC:¥tmp -g
```

コマンド行には，次のように入力します。

```
C:¥>rp78k0r -fk0r.ppr
```

デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定オプションには，次のものがあります。

- -y

-y

[記述形式]

```
-y パス名
```

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) rp78k0r.exe の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

[機能]

- -y オプションは，デバイス・ファイルを指定されたパスから読み込みます。

[用途]

- デバイス・ファイルが存在するパスを指定します。

[説明]

- -y オプションに続けてパス名以外が指定された場合，アボート・エラーとなります。
- -y オプションに続けて指定するパス名が省略された場合，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べて決定します。

(1) -y オプションで指定されたパス

(2) デバイス・ファイル・インストーラで登録されたパス

（3）RP78K0R の起動されたパス

（4）カレント・フォルダ

（5）環境変数 PATH

［使用例］

- デバイス・ファイルのパスをフォルダ C:￥78k0r￥dev に指定します。

```
C:￥>rp78k0r a.lmf -yC:￥78k0r￥dev
```

- デバイス・ファイルのパスをフォルダ D:￥device files に指定します。

```
C:￥>rp78k0r a.lmf -y"D:￥device files"
```

コピー・ルーチン・アドレス指定

コピー・ルーチン・アドレス指定オプションには，次のものがあります。

- -rc

-rc

[記述形式]

```
-rc アドレス
```

- 省略時解釈

  コピー・ルーチンを ROM 領域の空き領域へ配置します。

[機能]

- -rc オプションは，ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。

[用途]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定したいときに，-rc オプションで指定します。

[説明]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。

[使用例]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを 300H 番地に配置します。

```
C:¥>rp78k0r a.lmf -rc300H
```

ROM 化領域指定

ROM 化領域指定オプションには，次のものがあります。

- -ra

-ra

[記述形式]

```
-ra 開始アドレス，終了アドレス
```

- 省略時解釈

  内部 RAM 領域を ROM 化対象とします。

[機能]

- -ra オプションは，ROM 化対象領域を指定します。

[用途]

- ROM 化対象領域を指定したいときに，-ra オプションで指定します。

[説明]

- ROM 化対象領域の開始アドレスと終了アドレスを指定します。

[使用例]

- 0FCF00H 番地から FFFFFH 番地を ROM 化対象とします。

```
C:¥>rp78k0r a.lmf -ra0FCF00H,0FFFFFH
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --

--

[記述形式]

```
--
```

- 省略時解釈

  表示しません。

[機能]

- -- オプションは，ヘルプ・メッセージをディスプレイに出力します。

[用途]

- ヘルプ・メッセージは，リンク・オプションとその説明の一覧です。ROM 化プロセッサを実行するときに参照してください。

[説明]

- -- オプションを指定すると，ほかのリンク・オプションはすべて無効となります。
- ヘルプ・メッセージの続きを読む場合は，リターン・キーを押下してください。表示を途中で終了する場合は，リターン・キー以外の文字を入力したあとで，リターン・キーを押下してください。

注意　本オプションは，CubeSuite 上では指定することはできません。

[使用例]

- -- オプションを指定するとヘルプ・メッセージがディスプレイに出力されます。

```
C:¥>rp78k0r --
```

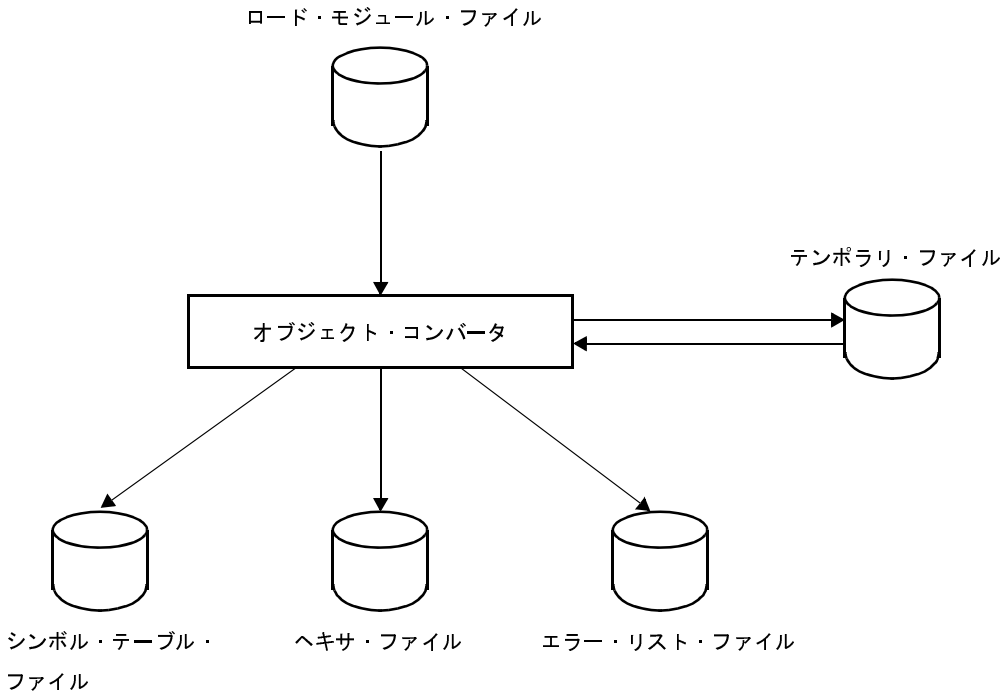
B. 5　オブジェクト・コンバータ

オブジェクト・コンバータは，CA78K0R のリンカが出力したロード・モジュール・ファイル（すべての参照アドレス情報が解決されていなければなりません）を入力し，それをヘキサ・ファイルとして出力します。

さらに，シンボリック・デバッグ時に使用するシンボル情報をシンボル・テーブル・ファイルとして出力します。

オブジェクト・コンバータ・エラーがある場合は，エラー・メッセージを出力し，エラーの原因を明示します。

図 B—29　オブジェクト・コンバータの入出力ファイル



B. 5. 1　入出力ファイル

オブジェクト・コンバータの入出力ファイルを次に示します。

表 B—14　オブジェクト・コンバータの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	ロード・モジュール・ファイル	- リンクの結果のオブジェクト・コードのバイナリ・イメージ・ファイル - リンカの出力したファイル	.lmf
	パラメータ・ファイル	- 実行コマンドのパラメータを内容とするファイル（ユーザ作成ファイル）	.poc

種類	ファイル名	説明	デフォルト・ファイル・タイプ
出力ファイル	ヘキサ・ファイル	- ロード・モジュール・ファイルを，ヘキサ・フォーマットで変換したファイル マスク ROM 発注時，または PROM プログラマ使用時に使います。	.hex
	シンボル・テーブル・ファイル	- 入力ファイルの各モジュールに含まれるシンボルの情報を持つファイル	.sym
	エラー・リスト・ファイル	- オブジェクト・コンバート時のエラー情報を持つファイル	.eoc

B. 5. 2　機　　能

(1) フラッシュ・メモリのセルフ書き換えモード対応

オブジェクト・コンバータは，フラッシュ・メモリのセルフ書き換えモード使用時に，フラッシュ・メモリに配置されたコードに対して，ブート領域とフラッシュ領域で別々のヘキサ・ファイルを生成することができます。別々のヘキサ・ファイルを出力するには，オブジェクト・コンバート・オプション -zf を指定します。ファイル・タイプは，次のようになります。

表 B—15　-zf オプション指定時のファイル・タイプ

ファイル	ファイル・タイプ
ブート領域 ROM プログラム側の出力ファイル	.hxb
ブート領域 ROM 以外のプログラム側の出力ファイル	.hxf

(2) ヘキサ・ファイル

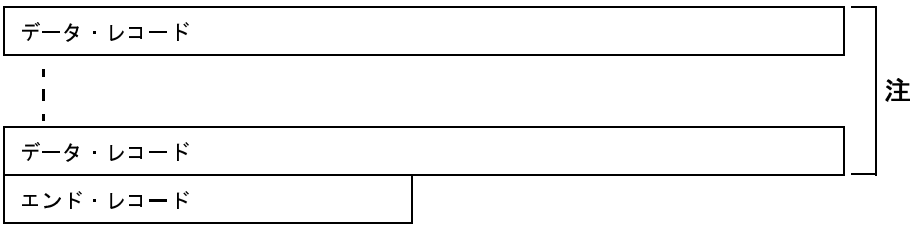
オブジェクト・コンバータの出力するヘキサ・ファイルは，PROM プログラマやデバッガなどの HEX ローダに入力可能です。

次に，サンプル・プログラムのヘキサ・ファイルを示します。

```
: 0200000080007E
: 1000800011201A1620FE9A93001421FE63958462B3
: 1000900095FAFE617131809AA40073617131809A82
: 0D00A000A40072AF4D8D020D070D30AFA8
: 00000001FF
```

(a) インテル標準ヘキサ・ファイルのフォーマット

図 B—30　インテル標準ヘキサ・フォーマット



注　データ・レコードは繰り返されます。

- データ・レコード

```
:       02      0000    00      8000    7E
(1)     (2)      (3)    (4)      (5)    (6)
```

項番	説明
(1)	レコード・マーク レコードの始まりを示します。
(2)	コード数（2 桁） レコードに納められるコードのバイト数です。最大 16 バイトになります。
(3)	ロケーション・アドレス（オフセット） そのレコードで表すコードの先頭アドレス（オフセット）を 16 進 4 桁で示します。
(4)	レコード・タイプ 00 で固定です。
(5)	コード（最大 32 桁） オブジェクト・コードを 1 バイトずつ上位 4 ビット，下位 4 ビットに分けて示します。 コードは，最大 16 バイトまで表現されます。
(6)	チェック・サム（2 桁） コード数からコードまでのデータを 0 から順に減算した値が入ります。

- エンド・レコード

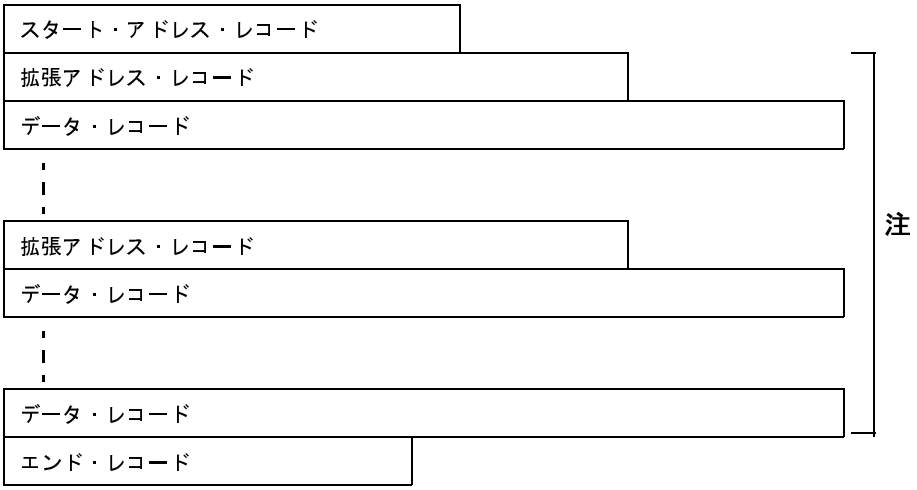
```
:       00      0000    01      FF
(1)     (2)      (3)    (4)      (5)
```

項番	説明
(1)	レコード・マーク レコードの始まりを示します。
(2)	コード数 00 で固定です。
(3)	0000 で固定です。

項番	説明
(4)	レコード・タイプ 01 で固定です。
(5)	チェック・サム FF で固定です。

(b) インテル拡張ヘキサ・ファイルのフォーマット

図 B—31　インテル拡張ヘキサ・フォーマット



注　拡張アドレス・レコード，データ・レコードは繰り返されます。

- 拡張アドレス・レコード

```
:       02      0000    02      XXXX    SS
(1)     (2)     (3)     (4)     (5)     (6)
```

項番	説明
(1)	レコード・マーク レコードの始まりを示します。
(2)	コード数 02 で固定です。
(3)	0000 で固定です。
(4)	レコード・タイプ 02 で固定です。
(5)	セグメントのパラグラフ値 セグメントのパラグラフ値を 16 進 4 桁で示します。
(6)	チェック・サム（2 桁） コード数からアドレスの上位 8 ビット値までのデータを 0 から順に減算した値が入ります。

- データ・レコード

```
:       02      0000    00      80000   7E
(1)     (2)     (3)     (4)     (5)     (6)
```

項番	説明
(1)	レコード・マーク レコードの始まりを示します。
(2)	コード数（2 桁） レコードに納められるコードのバイト数です。最大 16 バイトになります。
(3)	ロケーション・アドレス（オフセット） そのレコードで表すコードの先頭アドレス（オフセット）を 16 進 4 桁で示します。
(4)	レコード・タイプ 00H で固定です。
(5)	コード（最大 32 桁） オブジェクト・コードを 1 バイトずつ上位 4 ビット，下位 4 ビットに分けて示します。 コードは，最大 16 バイトまで表現されます。
(6)	チェック・サム（2 桁） コード数からコードまでのデータを 0 から順に減算した値が入ります。

- スタート・アドレス・レコード

```
:       04      0000    03      0000    0000    F9
(1)     (2)     (3)     (4)     (5)     (6)     (7)
```

項番	説明
(1)	レコード・マーク レコードの始まりを示します。
(2)	コード数 04 で固定です。
(3)	0000 で固定です。
(4)	レコード・タイプ 03 で固定です。
(5)	0000 で固定です。
(6)	0000 で固定です。
(7)	チェック・サム F9 で固定です。

- エンド・レコード

```
:       00      0000    01      FF
(1)     (2)     (3)     (4)     (5)
```

項番	説明
(1)	レコード・マーク レコードの始まりを示します。
(2)	コード数 00 で固定です。
(3)	0000 で固定です。
(4)	レコード・タイプ 01 で固定です。
(5)	FF で固定です。

(c) 拡張テクトロニクスヘキサ・ファイルのフォーマット

ヘキサ・ファイルは，次の 3 種類のブロックから構成されます。

- データ・ブロック
- シンボル・ブロック（未使用ブロックです。シンボル情報は，シンボル・テーブル・ファイルを用います。）
- ターミネーション・ブロック

各ブロックは，共通した 6 文字のヘッダ・フィールドで始まり，end-of-line で終了します。

ブロックの最大長は，先頭の文字%と end-of-line を含まずに 255 です。

次に，共通なヘッダ・フィールドの形式を示します。

表 B—16　拡張テクトロニクス・ヘッダ・フィールド

項目	ASCII キャラクタ数	説明
%	1	パーセント記号により，ブロックか拡張テクトロニクス・フォーマットであることが指定されます。
ブロック長	2	ブロック内のキャラクタ数を示す 2 桁の 16 進数です。このキャラクタ数には，先頭の%記号，および end-of-line は含まれません。
ブロックの種類	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
チェック・サム	2	先頭の %，チェック・サムの桁，および end-of-line を除くブロック内の全キャラクタの値[注]の合計を 256 で割った余りを表す 2 桁の 16 進数です。

注　チェック・サム評価のキャラクタの値

キャラクタ	値（10 進）
0 - 9	0 - 9

キャラクタ	値（10 進）
A - Z	10 - 35
$	36
%	37
.（ピリオド）	38
_（アンダースコア）	39
a - z	40 - 65

- データ・ブロック

データ・ブロックのフォーマットを，次に示します。

表 B—17　拡張テクトロニクスのデータ・ブロックのフォーマット

フィールド	ASCII キャラクタ数	説明
ヘッダ	6	標準ヘッダ・フィールド ブロックの種類＝6
ロード・アドレス	2 - 17	オブジェクト・コードがロードされるアドレスです。 キャラクタ数は可変です。
オブジェクト・コード	2n	2 桁の 16 進数として表されるバイト n 個を表します。

注意　拡張テクトロニクスでは，特定のフィールドで文字数 2 - 17（実際のデータとしては 1 - 16）まで変えることができます。可変フィールドの最初の文字は 16 進数で，残りのフィールド長を示します。数字の 0 は 16 文字の文字列が続けられることを示します。したがって，文字列は 1 - 16 文字となるため，文字列長を 1 文字加えて，可変長フィールドの長さは 2 - 17 となります。

```
%       15      6       1C      3       100     020202020202
(1)     (2)     (3)     (4)     (5)     (6)         (7)
```

項番	説明
(1)	ヘッダ・キャラクタ
(2)	ブロック長 15H ＝ 21
(3)	ブロックの種類 6
(4)	チェック・サム 1CH
(5)	ロード・アドレスの桁数
(6)	ロード・アドレス 100H

項番	説明
(7)	オブジェクト・コード 6 バイト

- ターミネーション・ブロック

ターミネーション・ブロックのフォーマットを，次に示します。

表 B—18　拡張テクトロニクスのターミネーション・ブロックのフォーマット

フィールド	ASCII キャラクタ数	説明
ヘッダ	6	標準ヘッダ・フィールド ブロックの種類＝8
ロード・アドレス	2 - 17	プログラム実行の開始アドレスです。 キャラクタ数は可変です。

```
%       08      8       1A      2       80
(1)     (2)     (3)     (4)     (5)     (6)
```

項番	説明
(1)	ヘッダ・キャラクタ
(2)	ブロック長 8H
(3)	ブロックの種類 8
(4)	チェック・サム 1AH
(5)	ロード・アドレスの桁数
(6)	ロード・アドレス 80H

- シンボル・ブロック（未使用）

拡張テクトロニクスのシンボル・ブロックは，シンボリック・デバッグ用に使用されるデータであり，次に示す属性を想定しています。

表 B—19　拡張テクトロニクスのシンボル・ブロックの属性

項目	属性
シンボル自体	1 - 16 個の英大小文字，数字，ピリオド，およびアンダースコア。 先頭文字に数字は許されません。
値	64 ビットまで（16 進数 16 桁）可能です。

項目	属性
種類	アドレス，またはスカラ（スカラはアドレスを除くすべての数値を示します）。 アドレスは，コード・アドレス（インストラクションのアドレス）とデータ・アドレス（データ項目のアドレス）に分けられます。
グローバル／ローカル指定	シンボルがグローバル（外部参照可能）かローカルかを示します。
セクション・メンバシップ	セクションは，メモリの名前の付いている範囲と考えることができます。 プログラムの各アドレスは，必ず 1 つにセクションに属しており，スカラはセクションに属していません。

シンボル・ブロックの形式を，次に示します。

表 B—20　拡張テクトロニクスのシンボル・ブロックのフォーマット

フィールド	ASCII キャラクタ数	説明
ヘッダ	6	標準ヘッダ・フィールド ブロックの種類＝3
セクション名	2 - 17	ブロック内で定義されるシンボルを含むセクション名です。キャラクタ数は可変です。
セクション定義	5 - 35	このフィールドは，各セクションのシンボル・ブロック 1 つによって表示されなければなりません。このフィールドは，任意の数のシンボル定義フィールドの前，または後に続けることができます。 フォーマットについては，「表 B—21　拡張テクトロニクスのシンボル・ブロック・セクション定義フィールド」を参照してください。
シンボル定義	各 5 - 35	ゼロ以上のシンボル定義フィールドです。 フォーマットについては，「表 B—22　拡張テクトロニクスのシンボル・ブロック・シンボル定義フィールド」を参照してください。

プログラム内のシンボルは，シンボル・ブロックとして転送されます。各シンボル・ブロックには，セクション名，およびこのセクションの属するシンボルのリストが含まれます（必要に応じて，どのセクションにもスカラを入れることができます）。
同じセクションのシンボルを 1 つ以上のブロックに入れることもできます。
シンボル・ブロック中のセクション定義フィールドとシンボル定義フィールドの形式を，次に示します。

表 B—21　拡張テクトロニクスのシンボル・ブロック・セクション定義フィールド

フィールド	ASCII キャラクタ数	説明
0	1	0 によりセクション定義フィールドであることが指定されます。
ベース・アドレス	2 - 17	セクション開始アドレスです。 キャラクタ数は可変です。
長さ	2 - 17	セクションの長さを示します。 キャラクタ数は可変で，次の式より算出されます。 1 －（上位アドレス－ベース・アドレス）

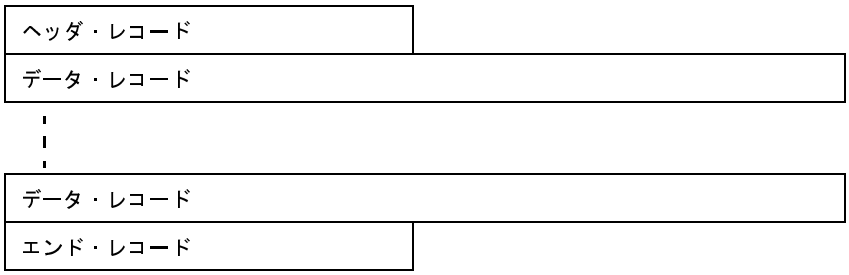
表 B—22　拡張テクトロニクスのシンボル・ブロック・シンボル定義フィールド

フィールド	ASCII キャラクタ数	説明
種類	1	シンボルのグローバル／ローカルの指定，およびシンボルの示す値の種類を示す 1 桁の 16 進数です。 1 ＝グローバル・アドレス 2 ＝グローバル・スカラ 3 ＝グローバル・コード・アドレス 4 ＝グローバル・データ・アドレス 5 ＝ローカル・アドレス 6 ＝ローカル・スカラ 7 ＝ローカル・コード・アドレス 8 ＝ローカル・データ・アドレス
シンボル	2 - 17	シンボル長，可変です。
数値	2 - 17	シンボルに対応する値で，キャラクタ数は可変です。

（d）モトローラ S タイプのフォーマット

変更生成されるヘキサ・ファイルは，3 種類，5 レコードからなり，ファイル全体の構成は次の図のようになります。

図 B—32　モトローラ S タイプ・フォーマット



レコードの種類を，次に示します。

表 B—23　モトローラ・ヘキサ・ファイルのレコードの種類

種類	レコード・タイプ
ヘッダ・レコード（オプショナル）	S0
データ・レコード	S2（スタンダード 24 ビット） S3（32 ビット）
エンド・レコード	S8（スタンダード 24 ビット） S7（32 ビット）

モトローラ・ヘキサ・フォーマットは，アドレスが 24 ビットのスタンダード・アドレスと 32 ビット・アドレスに分けられますが，スタンダード・アドレスの場合は，S0，S2，S8 レコード，32 ビット・アドレスの場合は，S0，S3，S7 レコードで構成されます。なお，ヘッダ・レコード S0 はオプションであり，出力されません。各 S レコードの末尾には，CR 文字が置かれます。

各レコードのフィールドの一般形式とその意味を，次に示します。

表 B—24　各レコードの一般形

レコード・タイプ	一般形
S0	S0XXYY … YYZZZZ
S2	S2XXWWWWWWDD … DDZZ
S3	S3XXWWWWWWWWDD … DDZZ
S7	S7XXWWWWWWWWZZ
S8	S8XXWWWWWWZZ

表 B—25　フィールドの意味

フィールド	意味
Sn	レコード・タイプ
XX	データ・レコード長 アドレスと 16 進データとチェック・サムのバイト数です。
YY … YY	ファイル名 入力ファイル名の ASCII コードを 16 進数で表現したものです。
WWWWWW [WW]	24 [32] ビット目アドレス
DD … DD	16 進データ データ 1 バイトを 2 桁の 16 進数で表現したものです。
ZZ	チェック・サム レコード長，アドレス，および 16 進データのバイトごとの和の 1 の補数の下位 1 バイトを 2 桁の 16 進数で表現したものです。

```
S2      08      00FF11      D4520A20    A0
(1)     (2)       (3)          (4)      (5)
```

項番	説明
(1)	レコード・タイプ
(2)	レコード長
(3)	ロード・アドレス（24 ビット・アドレス）
(4)	16 進データ
(5)	チェック・サム

- S0 レコード

```
S0      XX      YYYYYYYY    ZZ
(1)     (2)        (3)      (4)
```

項番	説明
(1)	レコード・タイプ
(2)	レコード長 (3) のバイト数＋(4) のバイト数です。
(3)	ファイル名
(4)	チェック・サム

- S2 レコード

```
S2      XX      WWWWWW      DD … DD     ZZ
(1)     (2)        (3)         (4)      (5)
```

項番	説明
(1)	レコード・タイプ
(2)	レコード長 (3) のバイト数＋(4) のバイト数＋(5) のバイト数です。
(3)	ロード・アドレス (4) のデータのロード・アドレスであり，24 ビットで 0H - FFFFFFH までの範囲です。
(4)	データ ロードされるデータそのものです。
(5)	チェック・サム

- S3 レコード

```
S3      XX      WWWWWWWW    DD … DD     ZZ
(1)     (2)        (3)         (4)      (5)
```

項番	説明
(1)	レコード・タイプ

項番	説明
(2)	レコード長 (3) のバイト数＋(4) のバイト数＋(5) のバイト数です。
(3)	ロード・アドレス (4) のデータのロード・アドレスであり，32 ビットで 0H - FFFFFFFFH までの範囲です。
(4)	データ ロードされるデータそのものです。
(5)	チェック・サム

- S7 レコード

```
S7      XX      WWWWWWWW    ZZ
(1)     (2)         (3)     (4)
```

項番	説明
(1)	レコード・タイプ
(2)	レコード長 (3) のバイト数＋(4) のバイト数です。
(3)	エントリ・アドレス エントリ・アドレスであり，32 ビットで 0H - FFFFFFFFH までの範囲です。
(4)	チェック・サム

- S8 レコード

```
S8      XX      WWWWWWWW    ZZ
(1)     (2)         (3)     (4)
```

項番	説明
(1)	レコード・タイプ
(2)	レコード長 (3) のバイト数＋(4) のバイト数です。
(3)	エントリ・アドレス エントリ・アドレスであり，24 ビットで 0H - FFFFFFH までの範囲です。
(4)	チェック・サム

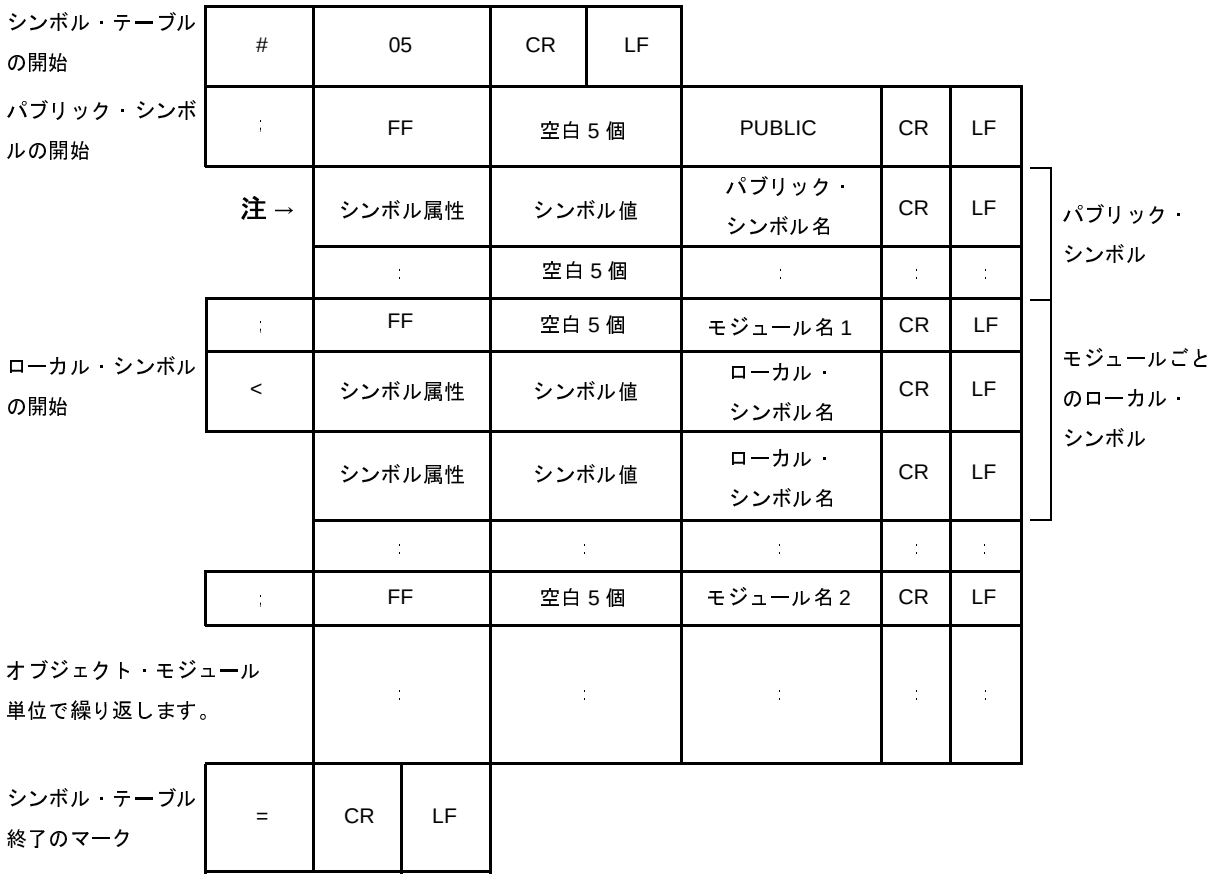
(3) シンボル・テーブル・ファイル

オブジェクト・コンバータの出力するシンボル・テーブル・ファイルは，デバッガへの入力となります。

次に，サンプル・プログラムのシンボル・テーブル・ファイルを示します。

```
#05
; FF     PUBLIC
01000E9CONVAH
0100000MAIN
01000D2START
00FFE20_@STBEG
00FCF00_@STEND
; FF     SAMPM
<02FFE20HDTSA
02FFE21STASC
; FF     SAMPS
<010015CSASC
0100162SASC1
=
```

図 B—33　シンボル・テーブル・ファイルのフォーマット



注　シンボル属性は，次の値をとります。

シンボル値のフォーマットについては，次の図を参照してください。

値	シンボル属性
00	EQU で定義した定数
01	コード・セグメント内のラベル
02	データ・セグメント内のラベル
03	ビット・シンボル
FF	モジュール名

図 B—34　シンボル値のフォーマット

・シンボル属性が NUMBER のとき

定数値　4 桁

・シンボル属性が LABEL のとき

アドレス値　4 桁

・シンボル属性がビット・シンボルのとき

上位 13 ビット	下位 3 ビット

上位 13 ビット ： 0FE00H からの相対アドレス

下位 3 ビット ： ビット位置（0 - 7）

B. 5. 3　操作方法

（1）オブジェクト・コンバータの起動方法

オブジェクト・コンバータの起動には，2 つの方法があります。

（a）コマンド行での起動

```
X:[パス名]>oc78k0r[△オプション]...ロード・モジュール・ファイル名[△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
oc78k0r	オブジェクト・コンバータのコマンド名
オプション	オブジェクト・コンバータに対して動作の詳細を指示します。 複数のオブジェクト・コンバート・オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，オブジェクト・コンバート・オプションに大文字，小文字の区別はありません。オブジェクト・コンバート・オプションについての詳細は，「B. 5. 4　オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（” “）で囲んでください。

ロード・モジュール・ファイル名	コンバートするロード・モジュール・ファイル名 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（” “）で囲んでください。

例　ヘキサ・ファイル samle.hex を出力します。

```
C:¥>oc78k0r k0r.lmf -osamle.hex
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，オブジェクト・コンバートするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>oc78k0r[△ロード・モジュール・ファイル]△-f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	オブジェクト・コンバータの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[△]オプション[△オプション]...
```

- コマンド行でロード・モジュール・ファイル名を省略した場合，パラメータ・ファイル内でロード・モジュール・ファイル名を 1 つだけ指定することができます。
- ロード・モジュール・ファイル名は，オプションのあとに記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定するすべてのオブジェクト・コンバート・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル k0r.poc をエディタで作成し，オブジェクト・コンバータを起動します。

```
; parameter file
k0r.lmf -osample.hex
-ssample.sym -r
```

```
C:¥>ra78k0r -fk0rmain.pra
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

オブジェクト・コンバータが起動すると，次の実行開始メッセージが表示されます。

```
78K0R Object Converter Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
```

(b) 実行終了メッセージ

オブジェクト・コンバートの結果，エラーが検出されなかった場合には，オブジェクト・コンバータは，次のメッセージを表示して制御をホスト OS に戻します。

```
Target chip : uPD78xxx
Device file : Vx.xx

Object Conversion Complete, 0 error(s) and 0 warning(s) found.
```

オブジェクト・コンバートの結果，エラーが検出された場合は，オブジェクト・コンバータはエラーとワーニングの数を表示して制御をホスト OS に戻します。

```
Target chip : uPD78xxx
Device file : Vx.xx

Object Conversion Complete, 3 error(s) and 0 warning(s) found.
```

オブジェクト・コンバート中に，オブジェクトコンバータの処理継続が不可能な致命的エラーが検出された場合，オブジェクト・コンバータはメッセージを表示してオブジェクト・コンバートを中止し，制御をホスト OS に戻します。

例 1. 存在しないロード・モジュール・ファイルを指定した場合

```
C:¥>oc78k0r sample.lmf
```

```
78K0R Object Converter Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RA78K0R error F4006 : File not found 'sample.lmf'
Program aborted.
```

この例では，存在しないロード・モジュール・ファイルを指定したためにエラーとなり，オブジェクト・コンバートが中止されました。

2. 存在しないオブジェクト・コンバート・オプションを指定した場合

```
C:¥>oc78k0r k0r.lmf -a
```

```
78K0R Object Converter Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RA78K0R error F4018 : Option is not recognized '-a'
Please enter 'OC78K0R--' , if you want help messages.
Program aborted.
```

この例では，存在しないオブジェクト・コンバート・オプションを指定したためにエラーとなり，オブジェクト・コンバートが中止されました。

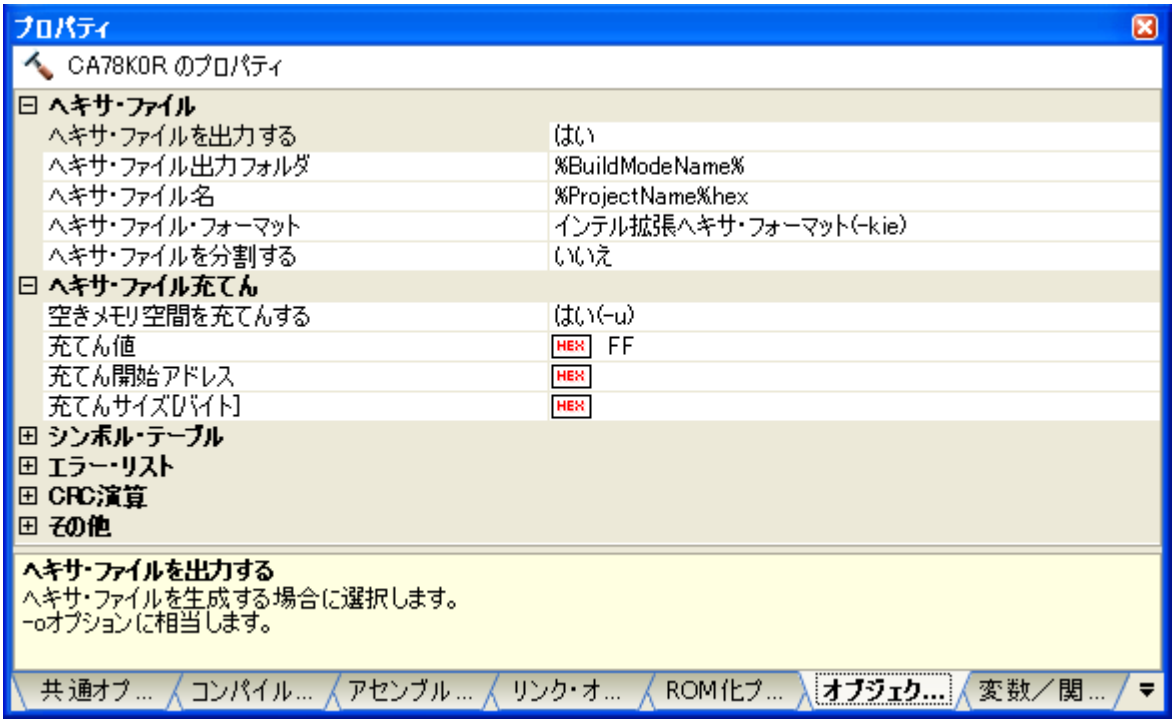
(3) CubeSuite+ でのオプション設定

CubeSuite+ からオブジェクト・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，［オブジェクト・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，対応するオブジェクト・コンバート・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：［オブジェクト・コンバート・オプション］タブ



B. 5. 4　オプション

(1) 種類

オブジェクト・コンバート・オプションは，オブジェクト・コンバータの動作に細かい指示を与えるものです。

オブジェクト・コンバート・オプションの分類と説明を示します。

表 B—26　オブジェクト・コンバート・オプション

分類	オプション	説明
ヘキサ・ファイル出力指定	-o	ヘキサ・ファイルの出力を指定します。
	-no	
シンボル・テーブル・ファイル出力指定	-s	シンボル・テーブル・ファイルの出力を指定します。
	-ns	
オブジェクト・アドレス順ソート指定	-r	ヘキサ形式オブジェクトをアドレス順にソートします。
	-nr	
オブジェクト充てん値指定	-u	ヘキサ形式オブジェクトが出力されないアドレス領域に対して，指定した充てん値をオブジェクト・コードとして出力します。
	-nu	
エラー・リスト・ファイル出力指定	-e	エラー・リスト・ファイルを出力します。
	-ne	
パラメータ・ファイル指定	-f	入力ファイル名，オプションを指定したファイルより入力します。
ヘキサ・フォーマット指定	-ki	インテル標準ヘキサ・フォーマット
	-kie	インテル拡張ヘキサ・フォーマット
	-kt	拡張テクトロニクス・フォーマット
	-km	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）
	-kme	モトローラ S タイプ・フォーマット（32 ビット・アドレス）
デバイス・ファイル・サーチ・パス指定	-y	デバイス・ファイルを指定されたパスから読み込みます。
フラッシュ・メモリ内蔵製品用ファイル分割出力指定	-zf	ブート領域とそれ以外の領域を別々のファイルに分割出力します。
CRC 演算指定	-crc	ヘキサ形式オブジェクトの CRC 演算を行います。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

ヘキサ・ファイル出力指定

ヘキサ・ファイル出力指定オプションには，次のものがあります。

- -o/-no

-o/-no

[記述形式]

```
-o [ 出力ファイル名 ]
-no
```

- 省略時解釈

  -o *入力ファイル名* .hex

[機能]

- -o オプションは，ヘキサ・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -no オプションは，ヘキサ・ファイルを出力しないことを指定します。

[用途]

- ヘキサ・ファイルの出力先や出力ファイル名を変更したいときに，-o オプションを指定します。
- シンボル・テーブル・ファイルの出力のみが目的でオブジェクト・コンバートする場合などに，-no オプションを指定します。これにより，オブジェクト・コンバート時間が短縮されます。

[説明]

- -o オプションを指定する際に出力ファイル名を省略すると，カレント・フォルダにヘキサ・ファイル（“*入力ファイル名* .hex”）が出力されます。
- 出力ファイル名にパス名のみを指定すると，指定したパスに“*入力ファイル名* .hex”が出力されます。
- -o と -no の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -zf オプションが指定された場合，次に示すファイル・タイプになります。

ファイル	ファイル・タイプ
ブート領域 ROM プログラム側の出力ファイル	.hxb
ブート領域 ROM 以外のプログラム側の出力ファイル	.hxf

［使用例］

- ヘキサ・ファイル sample.hex を出力します。

```
C:¥>oc78k0r k0r.lmf -osample.hex
```

シンボル・テーブル・ファイル出力指定

シンボル・テーブル・ファイル出力指定オプションには，次のものがあります。

- -s/-ns

-s/-ns

[記述形式]

```
-s[出力ファイル名]
-ns
```

- 省略時解釈

  -s *入力ファイル名*.sym

[機能]

- -s オプションは，シンボル・テーブル・ファイルの出力を指定します。また，その出力先や出力ファイル名を指定します。
- -ns オプションは，シンボル・テーブル・ファイルを出力しないことを指定します。

[用途]

- シンボル・テーブル・ファイルの出力先や出力ファイル名を変更したいときに，-s オプションを指定します。
- ヘキサ・ファイルの出力のみが目的でオブジェクト・コンバートする場合などに，-ns オプションを指定します。-ns オプションの指定により，オブジェクト・コンバート時間が短縮されます。

[説明]

- -s オプションを指定する際に出力ファイル名を省略すると，カレント・フォルダにシンボル・テーブル・ファイル（“*入力ファイル名*.sym”）が出力されます。
- 出力ファイル名にパス名のみを指定すると，指定したパスに“*入力ファイル名*.sym”が出力されます。
- -s と -ns の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- シンボル・テーブル・ファイル sample.sym を出力します。

```
C:¥>oc78k0r k0r.lmf -ssample.sym
```

オブジェクト・アドレス順ソート指定

オブジェクト・アドレス順ソート指定オプションには，次のものがあります。

- -r/-nr

-r/-nr

[記述形式]

```
-r
-nr
```

- 省略時解釈

  -r

[機能]

- -r オプションは，ヘキサ形式オブジェクトをアドレス順にソートして出力します。
- -nr オプションは，ヘキサ形式オブジェクトをロード・モジュール・ファイルに格納されている順に出力します。

[用途]

- ヘキサ形式オブジェクトがアドレス順にソートされる必要のない場合に，-nr オプションを指定します。

[説明]

- -r と -nr の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -no オプションを指定した場合，-r/-nr オプションは無効となります。

[使用例]

- ヘキサ形式オブジェクトをアドレス順にソートします。

```
C:¥>oc78k0r k0r.hex -r
```

オブジェクト充てん値指定

オブジェクト充てん値指定オプションには，次のものがあります。

- -u/-nu

-u/-nu

[記述形式]

```
-u 充てん値 [,[ スタート・アドレス],サイズ]
-nu
```

- 省略時解釈

  -u0FFH（0FFH を充てんします）

[機能]

- -u オプションは，ヘキサ形式オブジェクトが出力されないアドレス領域に対して，指定した充てん値をオブジェクト・コードとして出力します。
- -nu オプションは，-u オプションを無効にします。

[用途]

- ヘキサ形式オブジェクトが出力されていないアドレス領域には，不要なコードが書き込まれてしまうことがあります。このようなアドレスをプログラムが何らかの原因でアクセスした場合，プログラムの動作は予測できません。このため，あらかじめ -u オプションを指定して，ヘキサ形式オブジェクトが出力されていないアドレス領域にコードを書き込んでおきます。

[説明]

- 充てん値として指定可能な値の範囲は，0H ～ 0FFH です。

  2 進，8 進，10 進数字，または 16 進数字で指定します。範囲外の数値，または数値以外を指定した場合は，アボート・エラーとなります。
- スタート・アドレスには，充てんを行うアドレス領域の先頭アドレスを指定します。

  指定可能な値の範囲は，0H ～ 0FFEFFH です。

  2 進，8 進，10 進数字，または 16 進数字で指定します。範囲外の数値，または数値以外を指定した場合は，アボート・エラーとなります。また，スタート・アドレスを省略した場合は，0 を指定したものとみなされます。
- サイズには，充てんを行うアドレス領域のサイズを指定します。

  指定可能な値の範囲は，1H ～ 0FFF00H です。

2 進，8 進，10 進数字，または 16 進数字で指定します。範囲外の数値，または数値以外を指定した場合は，アボート・エラーとなります。また，スタート・アドレスを指定している場合は，サイズを省略することはできません。

- スタート・アドレスとサイズの両方の指定を省略した場合，オブジェクト・コンバータは内蔵 ROM の範囲が指定されたものとみなして処理を行います。
- -u と -nu の両オプションを同時に指定した場合は，あとで指定した方が有効となります。
- -no オプションを指定した場合は，-u/-nu オプションは無効になります。
- -u オプションで，複数のアドレス範囲を指定することはできません。
- -u オプションでのスタート・アドレス，サイズの指定形式とその解釈は，次のようになります。

（1）-u *充てん値*

対象デバイスに内蔵 ROM がある場合は，内蔵 ROM の範囲

（2）-u *充てん値*, *サイズ*

0 番地から，サイズ - 1 番地まで

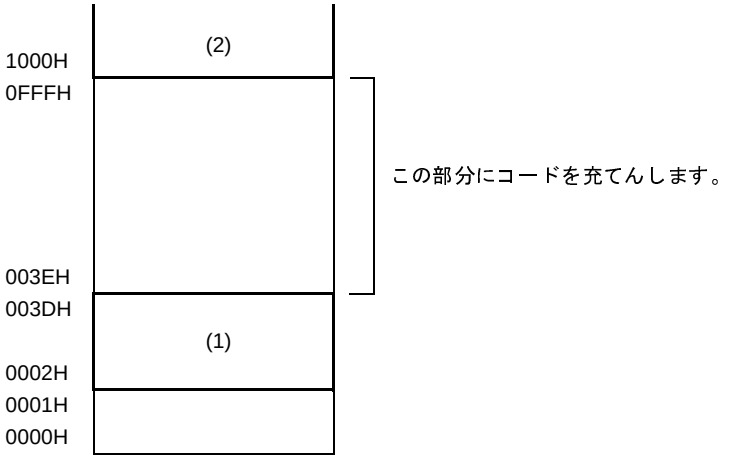
（3）-u *充てん値*, *スタート・アドレス*, *サイズ*

スタート・アドレスから，スタート・アドレス＋サイズ－ 1 番地まで

［使用例］

- ヘキサ形式オブジェクトが出力されていないアドレス領域にコードを充てんします。
  次のようなヘキサ・ファイルがあると仮定します。この場合，003EH - 0FFFH のアドレス領域にはコードが書き込まれていません。

```
: 020000000200FC
: 100002002B41000BFC80FE2B40000944F7083A20EC    ; (1)
: 100012001A6720FE2822006521FED350D25014FE1A    ; (1)
: 10002200B900059F2835002431B900059F28350005    ; (1)
: 0C003200242156AF0A8302A807A830560C
: 01000003B5D0d0026A3...                        ; (2)
: 1010100024A5F622B667...                       ; (2)
    :
: 00000001FF
```



003EH - 0FFFH のアドレス領域に，00H を充てんします。

```
C:¥>oc78k0r k0r.lmf -u00h,003eh,0fc2h
```

エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定オプションには，次のものがあります。

- -e/-ne

-e/-ne

[記述形式]

```
-e[出力ファイル名]
-ne
```

- 省略時解釈

  -ne

[機能]

- -e オプションは，エラー・リスト・ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。
- -ne オプションは，-e オプションを無効にします。

[用途]

- エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-e オプションを指定します。

[説明]

- -e オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は“*入力ファイル名*.eoc”となります。
- -e オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- -e と -ne の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- エラー・リスト・ファイル k0r.eoc を作成します。

```
C:¥>oc78k0r k0r.lmf -ek0r.eoc
```

パラメータ・ファイル指定

パラメータ・ファイル指定オプションには，次のものがあります。

- -f

-f

[記述形式]

```
-f ファイル名
```

- 省略時解釈
  コマンド行上からのみオプション，または入力ファイル名の入力が可能となります。

[機能]

- -f オプションは，オプション，または入力ファイル名を指定のファイルから入力することを指定します。

[用途]

- コマンド行では，オブジェクト・コンバータの起動に必要な情報を指定しきれないときに -f オプションを指定します。
- オブジェクト・コンバートするたびに繰り返し同じようにオプションを指定する場合には，それらをパラメータ・ファイルに記述しておいて，-f オプションで指定します。

[説明]

- ファイル名を省略すると，アボート・エラーとなります。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で -f オプションを指定すると，アボート・エラーとなります。
- パラメータ・ファイル中に記述可能な文字数に，制限はありません。
- 空白とタブ，および改行文字（LF）をオプション，あるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，または入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- “；”，または“#”以降に記述された文字は，改行文字（LF），または EOF の前まですべてコメントと解釈します。
- -f オプションを複数指定すると，アボート・エラーとなります。

［使用例］

- パラメータ・ファイル 78k0r.poc を使用してオブジェクト・コンバートします。

パラメータ・ファイル 78k0r.poc の内容は，以下のようになります。

```
; parameter file
k0r.lmf -osample.hex
-ssample.sym -r
```

コマンド行には，次のように入力します。

```
C:¥>oc78k0r k0r.lmf -f78k0r.poc
```

ヘキサ・フォーマット指定

ヘキサ・フォーマット指定オプションには，次のものがあります。

- -ki/-kie/-kt/-km/-kme

-ki/-kie/-kt/-km/-kme

[記述形式]

```
-ki
-kie
-kt
-km
-kme
```

- 省略時解釈

  -kie

[機能]

- 出力するヘキサ・ファイルのフォーマットを指定します。

[用途]

- 出力するヘキサ・ファイルのフォーマットを「インテル標準」，「インテル拡張」，「拡張テクトロニクス」，「モトローラＳタイプ（スタンダード・アドレス）」，「モトローラＳタイプ（32 ビット・アドレス）」の中から指定します。

[説明]

- 各オプションは，次のように指定します。

オプション	ヘキサ・フォーマット	範囲
-ki	インテル標準	0H - FFFFH（64K バイトまで）
-kie	インテル拡張	0H - FFFFFH（1M バイトまで）
-kt	拡張テクトロニクス	0H - FFFFFFFFH（4G バイトまで）
-km	モトローラＳタイプ（スタンダード・アドレス）	0H - FFFFFFH（16M バイトまで）
-kme	モトローラＳタイプ（32 ビット・アドレス）	0H - FFFFFFFFH（4G バイトまで）

［使用例］

- 出力するヘキサ・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）に指定します。

```
C:¥>oc78k0r k0r.lmf -km
```

デバイス・ファイル・サーチ・パス指定

デバイス・ファイル・サーチ・パス指定オプションには，次のものがあります。

- -y

-y

[記述形式]

```
-y パス名
```

- 省略時解釈

  デバイス・ファイルを読み込むパスは，次の順序で調べて決定されます。

(1) デバイス・ファイル・インストーラで登録されたパス

(2) oc78k0r.exe の起動されたパス

(3) カレント・フォルダ

(4) 環境変数 PATH

[機能]

- -y オプションは，デバイス・ファイルを指定されたパスから読み込みます。

[用途]

- デバイス・ファイルが存在するパスを指定します。

[説明]

- -y オプションに続けてパス名以外を指定した場合は，アボート・エラーとなります。
- -y オプションに続けて指定するパス名を省略した場合は，アボート・エラーとなります。
- デバイス・ファイルを読み込むパスは，次の順序で調べて決定します。

(1) -y オプションで指定されたパス

(2) デバイス・ファイル・インストーラで登録されたパス

（3）OC78K0R の起動されたパス

（4）カレント・フォルダ

（5）環境変数 PATH

［使用例］

- デバイス・ファイルのパスをフォルダ C:￥78k0r￥dev に指定します。

```
C:￥>oc78k0r k0r.lmf -yC:￥78k0r￥dev
```

- デバイス・ファイルのパスをフォルダ D:¥¥ device files に指定します。

```
C:￥>oc78k0r k0r.lmf -y"D:￥device files"
```

フラッシュ・メモリ内蔵製品用ファイル分割出力指定

フラッシュ・メモリ内蔵製品用ファイル分割出力指定オプションには，次のものがあります。

- -zf

-zf

[記述形式]

```
-zf
```

- 省略時解釈

  分割出力しません。

[機能]

- -zf オプションは，ブート領域とそれ以外の領域を別々のファイルに分割出力します。

[説明]

- フラッシュ・メモリ内蔵製品のブート領域 ROM プログラムのリンク指定時において，ブート領域とそれ以外の領域を別のヘキサ・ファイルに分割出力するオプションを追加します。
- -zf オプションを指定した場合，ブート領域 ROM プログラム側の出力ファイル・タイプは“hxb”，それ以外のプログラム側の出力ファイル・タイプは“hxf”になります。

注意　フラッシュ・メモリ領域セルフ書き換え機能を持たないデバイスでは，本オプションを使用しないでください。

[使用例]

- ブート領域とそれ以外の領域をヘキサ・ファイル k0r.hxb，k0r.hxf に分割出力します。

```
C:¥>oc78k0r k0r.lmf -zf
```

CRC 演算指定

CRC 演算指定オプションには，次のものがあります。

- -crc

-crc

[記述形式]

```
-crc 出力アドレス = 開始アドレス - 終了アドレス [, 開始アドレス - 終了アドレス] ... [/ 演算方法] [/ 初期値]
```

- 省略時解釈

  CRC 演算，および演算結果の出力を行いません。

[機能]

- -crc オプションは，CRC 演算を行うかどうかを指定します。

[用途]

- 指定した範囲のヘキサ形式オブジェクトを，下位アドレスから上位アドレスの順で CRC（Cyclic Redundancy Check）演算を行い，演算結果を出力アドレスへ出力します。

[説明]

- 出力アドレスとして指定可能な値の範囲は，0H ～ 0FFF00H です。

  2 進，8 進，10 進，または 16 進数字で指定します。

  範囲外の数値や数値以外を指定した場合，および省略した場合は，アボート・エラーとなります。
- 演算範囲（開始アドレス - 終了アドレス）は，カンマで区切ることにより，複数指定することができます。

  開始アドレス，終了アドレスとして指定可能な値の範囲は，0H ～ 0FFF00H です。

  範囲外の数値や数値以外を指定した場合，および省略した場合は，アボート・エラーとなります。
- 演算方法には HIGH，または GENERAL を指定します。

  HIGH を指定すると高速 CRC（high-speed CRC）用の演算結果を，GENERAL を指定すると汎用 CRC（general-purpose CRC）用の演算結果をそれぞれ得ることができます（「高速 CRC」，「汎用 CRC」の詳細については，デバイスのユーザーズ・マニュアルを参照してください）。

  なお，演算方法の指定を省略した場合は，HIGH を指定したものとして演算を行います。
- 初期値には演算用の初期値を指定します。

  初期値として指定可能な値の範囲は，0H ～ 0FFFFH です。

  ただし，演算方法に HIGH を指定した場合は初期値の指定を無視し，0H を指定したものとして演算を行います。

初期値を省略した場合は，0H を指定したものとして演算を行います。
範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。

- 演算範囲にある空き領域は，オブジェクト充てん値指定オプション（-u）に指定した値で演算を行います。
  オブジェクト充てん値指定オプションの指定がない場合は，0FFH を指定したものとして演算を行います。
  演算結果は，0H ～ 0FFFFH の 16 ビット・データとなります。
- 出力アドレスにすでにヘキサ形式オブジェクトが出力されているアドレスを含めることはできません。
  指定した場合は，アボート・エラーとなります。
- 演算範囲に出力アドレスを含めることはできません。
  指定した場合は，アボート・エラーとなります。
- 演算順は演算範囲の指定順ではありません。
  演算範囲として指定した下位アドレスから上位アドレスの順に演算を行います。
- -no オプションを指定した場合，-crc オプションは無効となります。

［使用例］

- 1000H 番地から 1FFDH 番地と 2000H 番地から 2FFFH 番地の領域に対して高速 CRC 演算を行い，その結果を 1FFEH 番地に出力します。

```
C:¥>oc78k0r -crc k0r.lmf -crc1FFEh=1000h-1FFDh,2000h-2FFFh/HIGH
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --

--

[記述形式]

```
--
```

- 省略時解釈

  表示しません。

[機能]

--- オプションは，ヘルプ・メッセージをディスプレイに出力します。

[用途]

- ヘルプ・メッセージは，オブジェクト・コンバート・オプションとその説明の一覧です。

  オブジェクト・コンバータを実行するときに参照してください。

[説明]

--- オプションを指定すると，ほかのオブジェクト・コンバート・オプションはすべて無効となります。

注意　本オプションは，CubeSuite+ 上では指定することはできません。

[使用例]

--- オプションを指定すると，ヘルプ・メッセージがディスプレイに出力されます。

```
C:¥>oc78k0r --
```

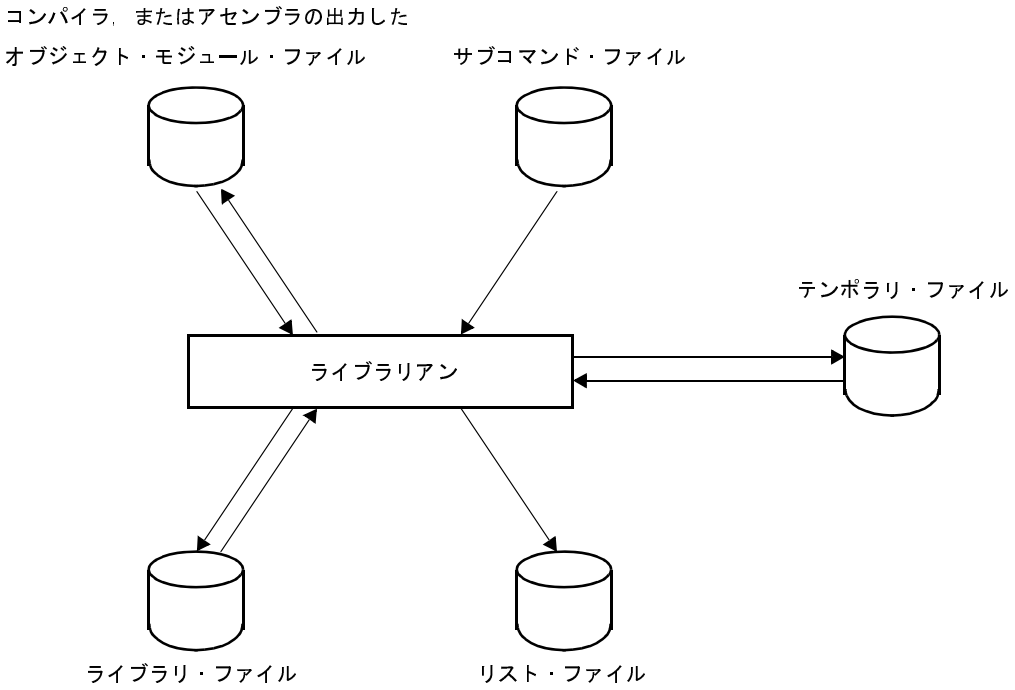
```
78K0R Object Converter Vx.xx [xx xxx xx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

usage : oc78k0r [option[...]] input-file [option[...]]
The option is as follows ([] means omissible).
-ffile           :Input option or input-file name from specified file.
-o[file]/-no     :Create HEX module file [with specified name] / Not.
-s[file]/-ns     :Create symbol table file [with specified name] / Not.
-e[file]/-ne     :Create the error list file [with the specified name] / Not.
-r/-nr           :Sort HEX object by address / Not.
-uvalue[,[start],size]/-nu :Fill up HEX object with specified value / Not.
-kkind           :Select hex format.  I;intel format  IE;intel extend format
                       T;tex format  M;s format  ME;s-32bit format
-ydirectory      :Set device file search path.
-zf              :Create boot hex module file (HXB), and flash hex module file(HXF).
-crcaddress=start-end[,start-end]...[/method][/init]
                 :Output CRC operation result value.
--               :Show this message.
DEFAULT ASSIGNMENT: -o -s -r -u0ffh
```

B. 6　ライブラリアン

ライブラリアンは，CA78K0R のオブジェクト・モジュール・ファイル，またはライブラリ・ファイルに対してモジュール単位で編集を行い，リスト・ファイルを出力します。

ライブラリアン・エラーがある場合は，エラー・メッセージを出力し，エラーの原因を明示します。

図 B—36　ライブラリアンの入出力ファイル



B. 6. 1　入出力ファイル

ライブラリアンの入出力ファイルを次に示します。

出力リストについての詳細は，「3. 6　ライブラリアン」を参照してください。

表 B—27　ライブラリアンの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	サブコマンド・ファイル	- 実行コマンドのサブコマンドとパラメータを内容とするファイル（ユーザ作成ファイル）	なし
出力ファイル	リスト・ファイル	- ライブラリ・ファイルの情報を出力した結果のファイル	.lst

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入出力ファイル	オブジェクト・モジュール・ファイル	- コンパイラ，またはアセンブラの出力したオブジェクト・モジュール・ファイル	.rel
	ライブラリ・ファイル	- ライブラリアンが出力したライブラリ・ファイルを入力し，内容を更新します。	.lib
	テンポラリ・ファイル	- ライブラリ化のためにライブラリアンが自動生成するファイル ライブラリアンの実行終了時には消去されます。	Lb*xxxxxx*.$$y （y＝1 - 6）

B. 6. 2　機　　能

(1) モジュールのライブラリ化

アセンブラ，およびリンカは，1 個の出力モジュールを 1 個のファイルに作成します。

したがって，モジュールの数が多い場合，ファイルの数も増加します。このため，複数のモジュールを 1 個のファイルにまとめる機能が用意されています。これをモジュールのライブラリ化と呼び，ライブラリ化されたファイルをライブラリ・ファイルと呼びます。

ライブラリ・ファイルは，リンカに入力することができます。したがって，モジュラ・プログラミングを行った場合，共通的なモジュールをライブラリ・ファイルとして作成しておけば，ファイル管理の面でも操作性の面においても効率が良くなります。

(2) ライブラリ・ファイルに対する編集

ライブラリアンには，ライブラリ・ファイルに対して次に示す編集機能があります。

- ライブラリ・ファイルへのモジュールの追加
- ライブラリ・ファイル内のモジュールの削除
- ライブラリ・ファイル内のモジュールの置換
- ライブラリ・ファイル内のモジュールの抽出

各機能についての詳細は，「B. 6. 5　サブコマンド」を参照してください。

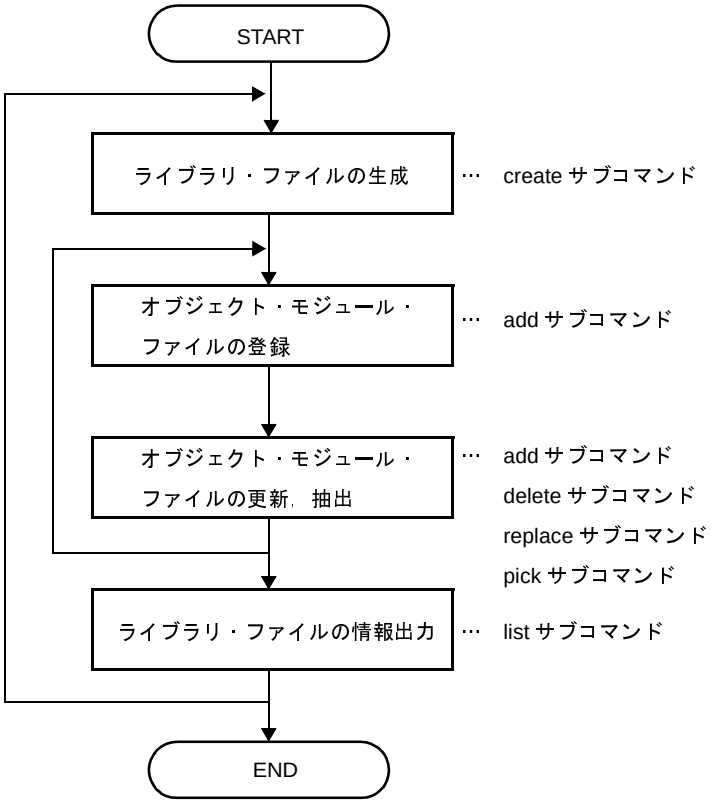
(3) ライブラリ・ファイル情報の出力

ライブラリアンには，ライブラリ・ファイルの中に格納されている情報のうち，次に示すものを編集し，出力する機能があります。

- モジュール名
- 作成プログラム
- 登録日付
- 更新日付
- PUBLIC シンボル情報

注意　ライブラリアンでは，(2)，(3) で説明した機能をそれぞれサブコマンドとして提供しています。ライブラリアンは，各サブコマンドを順次解説しながら処理を行います。サブコマンドの操作については，「B. 6. 5　サブコマンド」を参照してください。

一般的なライブラリ・ファイルの作成手順を，次に示します。



B. 6. 3　操作方法

(1) ライブラリアンの起動方法

ライブラリアンの起動には，2 つの方法があります。

(a) コマンド行での起動

```
X:[パス名]>lb78k0r[△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
lb78k0r	ライブラリアンのコマンド名

オプション	ライブラリアンに対して動作の詳細を指示します。 複数のライブラリ生成オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，ライブラリ生成オプションに大文字，小文字の区別はありません。ライブラリ生成オプションについての詳細は，「B. 6. 4　オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（”“）で囲んでください。

例　リスト・ファイルの 1 頁の行数を 20 行，1 行の文字数を 80 文字に指定します。

```
C:¥>lb78k0r -ll20 -lw80
```

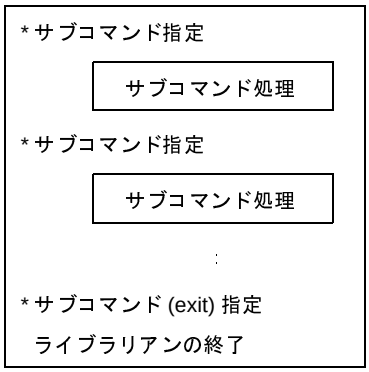
ライブラリアンが起動すると，次の起動メッセージが表示されます。

```
78K0R Librarian Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
*
```

“＊”に続いて，ライブラリアンのサブコマンドを指定します。

```
*create k0r.lib
*add k0r.lib k0rmain.rel k0rsub.rel
*exit
```

サブコマンドの入力を終了すると，各サブコマンドの処理を開始します。1 つのサブコマンドの処理が完了すると，再び“＊”が出力され，次のサブコマンドの入力待ち状態となります。終了サブコマンド（exit）が入力されるまで，この動作は繰り返されます。



1 行で指定可能な文字数は，128 文字までです。
1 行に必要なオペランド情報をすべて入力できない場合は，“＆”を用いて継続行の指定を行うことができます。なお，継続可能な行数は，15 行です。

(b) サブコマンド・ファイルによる起動

サブコマンド・ファイルとは，ライブラリアンへのコマンドを格納しているファイルです。

ライブラリアンの起動時にサブコマンド・ファイルを指定しない場合，“＊”のあとに複数のサブコマンドを入力しなければなりません。しかし，サブコマンド・ファイルを作成することにより，これら複数のサブコマンドの処理が一度にできます。

また，ライブラリ化のたびに同じサブコマンドを繰り返し指定することがあります。このような場合に，サブコマンド・ファイルを使用してライブラリ化を行います。

サブコマンド・ファイルを使用する場合には，ファイル名の前に“＜”を記述します。

サブコマンド・ファイルによる起動方法を次に示します。

```
X>lb78k0r △<サブコマンド・ファイル名 [△オプション] …
```

<	サブコマンド・ファイルを指定する場合は，必ず付加してください。
サブコマンド・ファイル名	サブコマンドが格納されているファイル

備考　サブコマンド・ファイルは，エディタなどで作成します。

サブコマンド・ファイル内での記述規則を次に示します。

```
サブコマンド名 オペランド情報
サブコマンド名 オペランド情報
  :
exit
```

- 1 つのサブコマンドが複数行にわたる場合，各行の行末に行の継続を示す“＆”を記述します。
- “;”（セミコロン）から行末までは，ライブラリアンのコマンドとしては解釈されず，コメントとしてみなされます。
- サブコマンド・ファイルの最後のサブコマンドが exit サブコマンドでない場合には，自動的に exit サブコマンドがあったものと解釈されます。
- ライブラリアンは，サブコマンドをサブコマンド・ファイルから読み込んで処理を行います。サブコマンド・ファイル内のすべてのサブコマンドの処理を終了すると，ライブラリアンは終了します。

例　サブコマンド・ファイル k0r.slb をエディタなどで作成し，ライブラリアンを起動します。

```
; library creation command
create k0r.lib
add k0r.lib k0rmain.rel &
k0rsub.rel
;
exit
```

```
C:¥>lb78k0r <k0r.slb
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

ライブラリアンが起動すると，次の実行開始メッセージが表示されます。

```
78K0R Librarian Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
*
```

(b) 実行終了メッセージ

ライブラリアンは，実行終了メッセージを出力しません。各処理を終了したあとにユーザが exit コマンドを入力することにより，ライブラリアンは制御をホスト OS に戻します。

```
*create k0r.lib
*add k0r.lib k0rmain.rel k0rsub.rel
*exit
```

ライブラリアンの処理継続が不可能な致命的エラーが検出された場合，ライブラリアンはメッセージを表示して，制御をホスト OS に戻します。

例　存在しないライブラリ生成オプションを指定した場合

```
C:¥>lb78k0r -a
```

```
78K0R Librarian Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
RA78K0R error F5018 : Option is not recognized '-z'
Usage : LB78K0R [options]
```

この例では，存在しないライブラリ生成オプションを指定したためにエラーとなり，ライブラリアンの実行が中止されます。

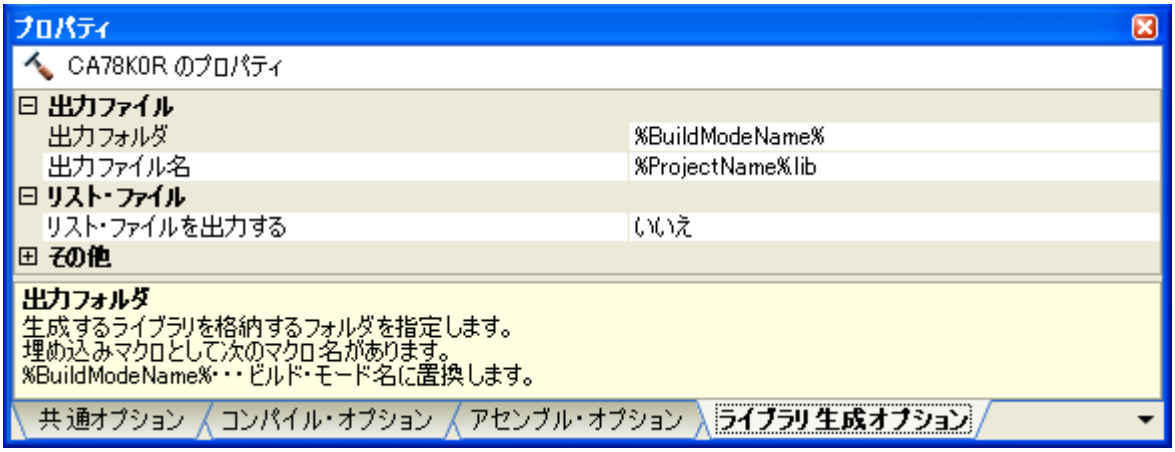
(3) CubeSuite+ でのオプション設定

CubeSuite+ からライブラリ生成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，［ライブラリ生成オプション］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，対応するライブラリ生成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：［ライブラリ生成オプション］タブ



B. 6. 4　オプション

(1) 種類

ライブラリ生成オプションは，ライブラリアンの動作に細かい指示を与えるものです。

ライブラリ生成オプションの分類と説明を示します。

表 B—28　ライブラリ生成オプション

分類	オプション	説明
リスト・ファイル形式指定	-lw	リスト・ファイルの 1 行に印字する文字数を変更します。
	-ll	リスト・ファイルの 1 頁に印字する行数を変更します。
	-lf	リスト・ファイルの最後に，改頁コードを付加します。
	-nlf	
テンポラリ・ファイル作成パス指定	-t	テンポラリ・ファイルを指定したパスに作成します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

リスト・ファイル形式指定

リスト・ファイル形式指定オプションには，次のものがあります。

- -lw
- -ll
- -lf/-nlf

-lw

[記述形式]

```
-lw[文字数]
```

- 省略時解釈

  -lw132（ディスプレイ出力の場合は 80 文字とします）

[機能]

- -lw オプションは，リスト・ファイルの 1 行の文字数を指示します。

[用途]

- リスト・ファイルの 1 行の文字数を変更したいときに，-lw オプションで指定します。

[説明]

- -lw オプションで指定可能な文字数の範囲（ディスプレイ出力の場合は，80 文字まで）は，72 ～ 260 です。
- 範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。
- 文字数を省略した場合は，132 を指定したものとみなされます。ただし，リスト・ファイルの出力先がディスプレイの場合は，80 となります。
- 指定する文字数には，ターミネータ（CR，LF）は含みません。
- list サブコマンドを指定していない場合は，-lw オプションは無視されます。
- -lw オプションを複数指定した場合は，あとで指定した方が有効となります。

[使用例]

- リスト・ファイルの 1 行の文字数を 80 文字に指定します。

```
C:¥>lb78k0r -lw80
```

-ll

[記述形式]

```
-ll[行数]
```

- 省略時解釈

  -ll0（改頁しません）

[機能]

- -ll オプションは，リスト・ファイルの 1 頁の行数を指定します。

[用途]

- リスト・ファイルの 1 頁の行数を変更したいときに，-ll オプションを指定します。

[説明]

- -ll オプションで指定可能な行数の範囲は，0，および 20 ～ 32767 です。
- 範囲外の数値や数値以外を指定した場合は，アボート・エラーとなります。
- 行数を省略した場合は，0 を指定したものとみなされます。
- 行数の 0 を指定した場合は，改頁されません。
- list サブコマンドを指定していない場合は，-ll オプションは無視されます。
- -ll オプションを複数指定した場合は，あとで指定した方が有効となります。

[使用例]

- リスト・ファイルの 1 頁の行数を 20 行に指定します。

```
C:¥>lb78k0r -ll20
```

-lf/-nlf

[記述形式]

```
-lf
-nlf
```

- 省略時解釈
  -nlf

[機能]

- -lf オプションは，リスト・ファイルの最後に改頁コード（FF）を付加することを指定します。
- -nlf オプションは，-lf オプションを無効にします。

[用途]

- リスト・ファイルの内容を印字したあとで改頁しておきたい場合に，-lf オプションを指定して改頁コードを付加します。

[説明]

- list サブコマンドを指定していない場合は，-lf オプションは無視されます。
- -lf と -nlf の両オプションを同時に指定した場合は，あとで指定した方が有効となります。

[使用例]

- リスト・ファイルの最後に改頁コードを付加します。

```
C:¥>lb78k0r -lf
```

テンポラリ・ファイル作成パス指定

テンポラリ・ファイル作成パス指定オプションには，次のものがあります。

- -t

-t

[記述形式]

```
-t パス名
```

- 省略時解釈
  環境変数 TMP により指定されたパス
  指定されていない場合は，カレント・パス

[機能]

- -t オプションは，テンポラリ・ファイルを作成するパスを指定します。

[用途]

- テンポラリ・ファイルの作成場所を指定することができます。

[説明]

- パス名として，パス以外のものは指定することはできません。
- パス名を省略することはできません。
- 以前に作成したテンポラリ・ファイルが存在している場合でも，ファイル保護をしていなければ上書きされます。
- 必要とするメモリ・サイズが残っている間は，テンポラリ・ファイルをメモリに展開します。
  なお，メモリが足りなくなった時点で，メモリに展開していたテンポラリ・ファイルの内容をディスクに書き出します。以降のテンポラリ・ファイルへのアクセスは，セーブしたディスク・ファイルに対して行います。
- テンポラリ・ファイルは，ライブラリ化終了時に削除されます。また，キー入力（[Ctrl] ＋ [C] キー）によってライブラリ化が中止されたときも，削除されます。
- テンポラリ・ファイルの作成パスは，次の順序で決定されます。

(1) -t オプションで指定したパス

(2) 環境変数 TMP で設定したパス（-t オプション省略の場合）

(3) カレント・パス（TMP を設定していない場合）

注意　(1)，または（2）を指定した場合，指定されたパスにテンポラリ・ファイルが作成できなければ，アボート・エラーとなります。

［使用例］

- テンポラリ・ファイルをフォルダ C:￥tmp に出力します。

```
C:￥>lb78k0r -tC:￥tmp
```

- テンポラリ・ファイルをフォルダ D:￥temporary files に出力します。

```
C:￥>lb78k0r -t"D:￥temporary files"
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --

--

[記述形式]

```
--
```

- 省略時解釈

  表示しません。

[機能]

--- オプションは，ヘルプ・メッセージをディスプレイに出力します。

[用途]

- ヘルプ・メッセージは，サブコマンドとその説明の一覧です。ライブラリアンを実行するときに参照してください。

[説明]

--- オプションを指定するとほかのライブラリ生成オプションは，すべて無効となります。

注意　本オプションは，CubeSuite+ 上では指定することはできません。

[使用例]

--- オプションを指定すると，ヘルプ・メッセージがディスプレイに出力されます。

```
C:¥>lb78k0r --
```

```
78K0R Librarian Vx.xx [xx xxx xx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
+-------------------------------------------------------------------------------+
| Subcommands : create,add,delete,replace,pick,list,help,exit                   |
|                                                                               |
|   Usage : subcommand[ option] masterLBF[ option] transaction[ option]         |
|                                                                               |
|               transaction :== OMFname                                         |
|                                  LBFname[(modulename[,...])]                  |
|                                                                               |
|   <create > : create masterLBF[ transaction]                                  |
|   <add    > : add masterLBF transaction                                       |
|   <delete > : delete masterLBF(modulename[,...])                              |
|   <replace> : replace masterLBF transaction                                   |
|   <pick   > : pick masterLBF(modulename[,...])                                |
|   <list   > : list[ option] masterLBF[(modulename[,...])                      |
|                  option : -p = output public symbol                           |
|                           -np = no output public symbol                       |
|                           -o filename = specify output file name              |
|   <help   > : help                                                            |
|   <exit   > : exit                                                            |
|                                                                               |
+-------------------------------------------------------------------------------+
```

B. 6. 5　サブコマンド

(1) 種類

サブコマンドは，ライブラリアンの動作に細かい指示を与えるものです。

サブコマンドの説明を示します。

表 B—29　サブコマンド

サブコマンド名	短縮名	説明
create	c	ライブラリ・ファイルを新規に作成します。
add	a	ライブラリ・ファイルにモジュールを追加します。
delete	d	ライブラリ・ファイル内のモジュールを削除します。
replace	r	ライブラリ・ファイル内のモジュールをほかのモジュールと置き換えます。
pick	p	ライブラリ・ファイル内のモジュールを取り出します。
list	l	ライブラリ・ファイル内のモジュール情報を出力します。
help	h	コンソールにヘルプ・メッセージを出力します。
exit	e	ライブラリアンを終了します。

(2) サブコマンド・ファイルの一般形式

＊サブコマンド［△*オプション*］△*ライブラリ・ファイル名*［△*オプション*］*トランザクション*［△*オプション*］

ライブラリ・ファイル名	直前に指定されたライブラリ・ファイル名は，“ . ”で置き換えることができます。
トランザクション	*トランザクション*= △*オブジェクト・モジュール・ファイル名* △*ライブラリ・ファイル名*［△(△*モジュール名*［△, …])］

備考　サブコマンド，オプションに大文字，小文字の区別はありません。

create

[記述形式]

```
create △ライブラリ・ファイル名 [ △トランザクション]
c △ライブラリ・ファイル名 [ △トランザクション]                    ; 短縮形
```

[機能]

- create サブコマンドは，ライブラリ・ファイルを新規に作成します。

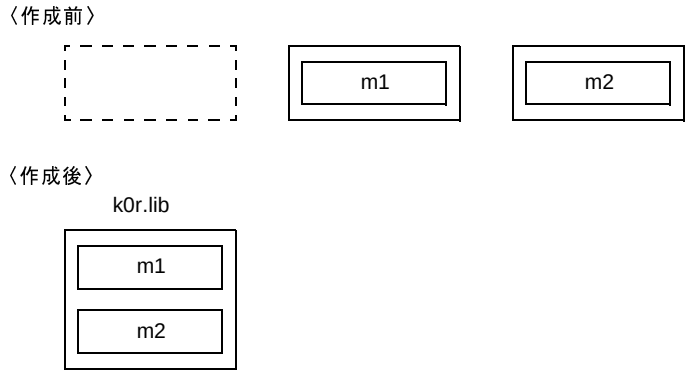
[説明]

- 作成されたライブラリ・ファイルのサイズは，0 となります。
- トランザクションを指定した場合は，ライブラリ・ファイルの作成と同時にモジュールを登録します。
- ライブラリ・ファイル名：
  指定したファイルがすでに存在している場合は，上書きします。
- トランザクション：
  ライブラリ・ファイル中に，パブリック・シンボルと同一のパブリック・シンボルを持つオブジェクト・モジュール・ファイルは，登録することができません。
  また，ライブラリ・ファイル中にあるモジュールと同一の名前のモジュールは，登録することができません。
- エラーが発生した場合は，処理を中断し，ライブラリ・ファイルは作成されません。

[使用例]

- ライブラリ・ファイル k0r.lib を作成し，同時にモジュール m1 と m2 を登録します。

```
*create k0r.lib m1.rel m2.rel
```



add

[記述形式]

```
add △ライブラリ・ファイル名△トランザクション
a △ライブラリ・ファイル名△トランザクション                    ; 短縮形
```

[機能]

- add サブコマンドは，既存のライブラリ・ファイルに対して，モジュールを追加します。

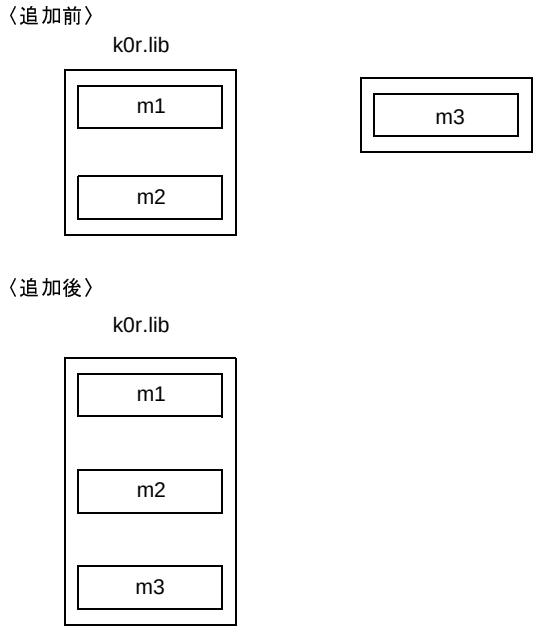
[説明]

- 追加するライブラリ・ファイル中には，モジュールが存在していなくてもかまいません。
- 追加するモジュールと同名のモジュールがライブラリ・ファイル内に存在する場合は，エラーとなります。
- 追加するモジュール中のパブリック・シンボルがライブラリ・ファイル内に存在する場合は，エラーとなります。

[使用例]

- ライブラリ・ファイル k0r.lib にモジュール m3 を追加します。

```
*add k0r.lib m3.rel
```



delete

[記述形式]

```
delete △ライブラリ・ファイル名△ ( △モジュール名 [ △ , … ] △ )
d △ライブラリ・ファイル名△ ( △モジュール名 [ △ , … ] △ )          ; 短縮形
```

[機能]

- delete サブコマンドは，既存のライブラリ・ファイルからモジュールを削除します。

[説明]

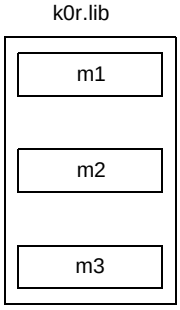
- 指定したモジュールがライブラリ・ファイルに存在しない場合は，エラーとなります。
- エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。

[使用例]

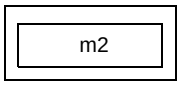
- ライブラリ・ファイル k0r.lib からモジュール m1，m3 を削除します。

```
*delete k0r.lib ( m1.rel , m3.rel )
```

〈削除前〉



〈削除後〉



replace

[記述形式]

```
replace △ライブラリ・ファイル名△トランザクション
r △ライブラリ・ファイル名△トランザクション                    ; 短縮形
```

[機能]

- replace サブコマンドは，既存のライブラリ・ファイルのモジュールを，ほかのオブジェクト・モジュール・ファイルのモジュールと置き換えます。

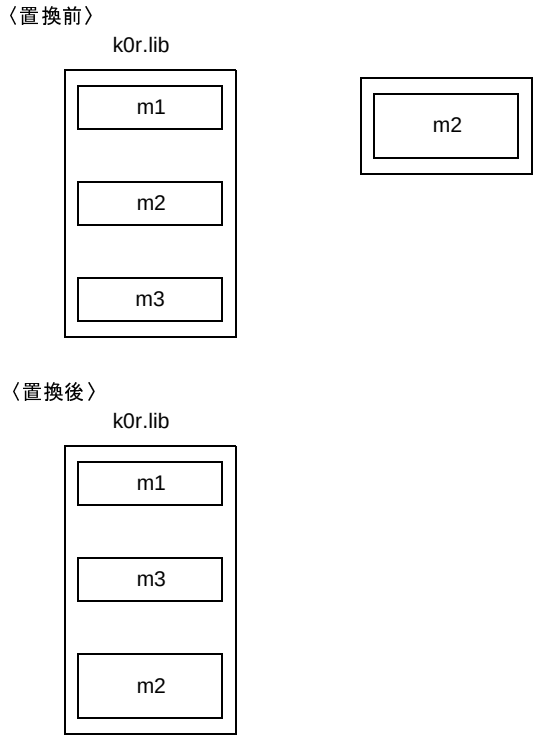
[説明]

- 置換するモジュール名と同名のモジュールが更新するライブラリ・ファイル内に存在しない場合は，エラーとなります。
- 置換するモジュール中のパブリック・シンボルがライブラリ・ファイル内に存在する場合は，エラーとなります。
- 置換するオブジェクト・モジュール・ファイル名は，登録時と同じファイル名でなければなりません。
- エラーが発生した場合は処理を中断し，ライブラリ・ファイルの状態は変化しません。

[使用例]

- ライブラリ・ファイル k0r.lib 中のモジュール m2 を置換します。

```
*replace k0r.lib m2.rel
```



ライブラリ・ファイル中のモジュール m2 を削除したあと，新たにモジュール m2 を登録するため，ライブラリ・ファイル中のモジュール m2 の順序は最後となります。

pick

[記述形式]

```
pick △ライブラリ・ファイル名△ ( △モジュール名 [ △ , … ] △ )
p △ライブラリ・ファイル名△ ( △モジュール名 [ △ , … ] △ )          ; 短縮形
```

[機能]

- pick サブコマンドは，既存のライブラリ・ファイルのモジュールから指定したモジュールを取り出します。

[説明]

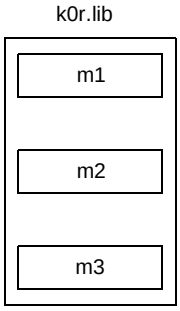
- 取り出したモジュールは，登録時のファイル名を持つオブジェクト・モジュール・ファイルとなります。
- 指定したモジュール名がライブラリ・ファイル内に存在しない場合は，エラーとなります。
- エラーの場合は，処理を中断します。ただし，複数のモジュールが指定されているときにエラーが発生した場合は，エラーとなったモジュールの直前までに取り出されたモジュールは有効となり，ディスク上に保存されます。

[使用例]

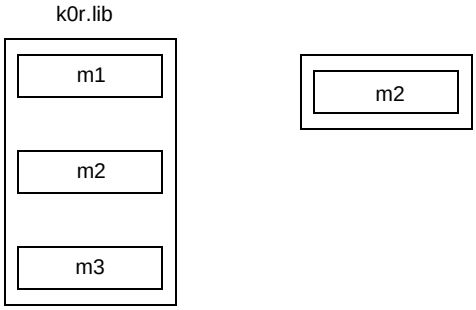
- ライブラリ・ファイル k0r.lib 中のモジュール m2 を取り出します。

```
*pick k0r.lib ( m2.rel )
```

〈抽出前〉



〈抽出後〉



list

[記述形式]

```
list[△オプション]△ライブラリ・ファイル名[△(△モジュール名[△, … ]△)]
l[△オプション]△ライブラリ・ファイル名[△(△モジュール名[△, … ]△)]          ; 短縮形
  オプション : -public/-nopublic
             : -o△ファイル名
```

[機能]

- list サブコマンドは，ライブラリ・ファイル内のモジュール情報を出力します。

[説明]

- オプションは，複数指定することができます。なお，オプションに大文字，小文字の区別はありません。
  - -o：

    出力ファイル名を省略した場合は，エラーとなります。

    ファイル・タイプを省略した場合は，“入力ファイル名 .lst”が入力されたものとします。
  - -public/-nopublic：

    -p/-np と指定することも可能です。

    -public は，パブリック・シンボル情報の出力を指示します。

    -nopublic は，-public を無効にします。

    -public と -nopublic の両方を指定した場合は，あとで指定した方が優先されます。

[使用例]

- ライブラリ・ファイル k0r.lib のモジュール情報をリスト・ファイル k0r.list に出力します。その際，パブリック・シンボル情報が出力されるように，-p オプションを指定します。

```
*list -p -ok0r.lst k0r.lib
```

リスト・ファイル k0r.lst の内容は，以下のようになります。

```
78K0R librarian Vx.xx                                        DATE : xx xxx xx   PAGE   1

LIB-FILE NAME : k0r.lib          (xx xxx xxxx)

  0001  k0rmain.rel      (xx xxx xxxx)

    MAIN     START

  NUMBER OF PUBLIC SYMBOLS :        2

  0002    k0rsub.rel     (xx xxx xxxx)

    CONVAH

  NUMBER OF PUBLIC SYMBOLS :        1
```

help

[記述形式]

```
help
h                    ; 短縮形
```

[機能]

- help サブコマンドは，ヘルプ・メッセージをディスプレイに出力します。

[説明]

- ヘルプ・メッセージは，サブコマンドとその説明の一覧です。help コマンド，または -- オプションを指定してライブラリアンを実行するときに参照してください。

[使用例]

- help コマンドを指定すると，ヘルプ・メッセージが出力されます。

```
*help
```

```
+-------------------------------------------------------------------------------+
| Subcommands : create,add,delete,replace,pick,list,help,exit                   |
|                                                                               |
|   Usage : subcommand[ option] masterLBF[ option] transaction[ option]         |
|                                                                               |
|               transaction :== OMFname                                         |
|                                  LBFname[(modulename[,...])]                  |
|                                                                               |
|   <create > : create masterLBF[ transaction]                                  |
|   <add    > : add masterLBF transaction                                       |
|   <delete > : delete masterLBF(modulename[,...])                              |
|   <replace> : replace masterLBF transaction                                   |
|   <pick   > : pick masterLBF(modulename[,...])                                |
|   <list   > : list[ option] masterLBF[(modulename[,...])                      |
|                   option : -p = output public symbol                          |
|                            -np = no output public symbol                      |
|                            -o filename = specify output file name             |
|   <help   > : help                                                            |
|   <exit   > : exit                                                            |
|                                                                               |
+-------------------------------------------------------------------------------+
```

exit

[記述形式]

```
exit
e                    ; 短縮形
```

[機能]

- exit サブコマンドは，ライブラリアンを終了します。

[説明]

- ライブラリアンを終了するときに使用します。

[使用例]

- ライブラリアンを終了します。

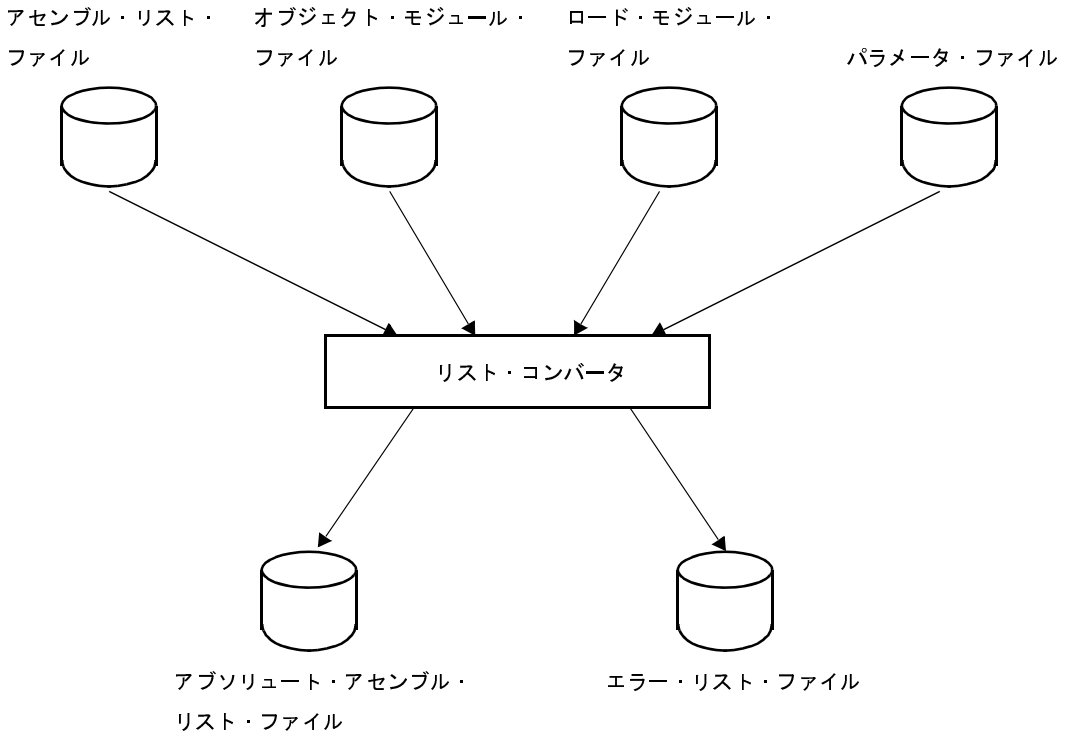
```
*exit
```

B. 7　リスト・コンバータ

リスト・コンバータは，アセンブラが出力するアセンブル・リスト・ファイル，オブジェクト・モジュール・ファイルと，リンカが出力するロード・モジュール・ファイルを入力します。

入力ファイル中のリロケータブルなアドレスやシンボルに実際のアドレスを埋め込み，アブソリュート・アセンブル・リスト・ファイルとして出力します。

図 B—38　リスト・コンバータの入出力ファイル



B. 7. 1　入出力ファイル

リスト・コンバータの入出力ファイルを次に示します。

表 B—30　リスト・コンバータの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイル	.rel
	アセンブル・リスト・ファイル	- アセンブル・リスト，クロスリファレンス・リストなどのアセンブル情報を持つファイル	.prn
	ロード・モジュール・ファイル	- リンク結果のオブジェクト・コードのバイナリ・イメージ・ファイル	.lmf
	パラメータ・ファイル	- 実行コマンドのパラメータを内容とするファイル（ユーザ作成ファイル）	.plv
出力ファイル	アブソリュート・アセンブル・リスト・ファイル	- 入力ファイル中のリロケータブルなアドレスやシンボルに実際のアドレスを埋め込んだリスト・ファイル	.p
	エラー・リスト・ファイル	- リスト・コンバート時のエラー情報を持つファイル	.elv

B. 7. 2　機　　能

(1) アセンブラ（リロケータブル・アセンブラ）の短所を解決

リスト・コンバータは，アセンブル・リスト・ファイルにロケーション，オブジェクト・コードを埋め込むことにより，リロケータブル・アセンブラの短所を解決します。

- リスト・コンバータの出力したアブソリュート・アセンブル・リストは，動作時のアドレスと完全に一致しています。
- 外部シンボルの実際の値がリスト上に埋め込まれます。
- リロケータブルな値がリスト上に実際の値として埋め込まれます。
- シンボル・テーブル，あるいはクロスリファレンス・リスト上のシンボル値に対しても，実際の値が埋め込まれます。

以下に，リスト・コンバータによって得られるアブソリュート・アセンブル・リストの例を示します。

例 1. ロケーションの埋め込みは，次のようになります。

- アセンブル・リスト

```
22  22  -----          CSEG
23  23  00000      START :
24  24
25  25                 ; chip initialize
26  26  00000 RCBF80000 MOVW    SP , #_@STBEG
27  27
28  28  00004  CD201A   MOV     HDTSA , #1AH
29  29  00007  3620FE   MOVW    HL , #LOWW ( HDTSA )  ; set hex 2-code data in HL registor
30  30
31  31  0000A RFD0000   CALL    !CONVAH               ; convert ASCII <- HEX
32  32                                                ; output BC-register <- ASCII code
33  33  0000D  3421FE   MOVW    DE , #LOWW ( STASC )  ; set DE <- store ASCII code table
34  34  00010  63       MOV     A , B
35  35  00011  99       MOV     [ DE ] , A
36  36  00012  A5       INCW    DE
37  37  00013  62       MOV     A , C
38  38  00014  99       MOV     [ DE ] , A
39  39
40  40  00015  EFFE     BR      $$
41  41
42  42                 END
```

- アブソリュート・アセンブル・リスト

```
22  22  -----          CSEG
23  23  000D2      START :
24  24
25  25                 ; chip initialize
26  26  000D2 RCBF820FE MOVW    SP , #_@STBEG
27  27
28  28  000D6  CD201A   MOV     HDTSA , #1AH
29  29  000D9  3620FE   MOVW    HL , #LOWW ( HDTSA )  ; set hex 2-code data in HL registor
30  30
31  31  000DC RFDE900   CALL    !CONVAH               ; convert ASCII <- HEX
32  32                                                ; output BC-register <- ASCII code
33  33  000DF  3421FE   MOVW    DE , #LOWW ( STASC )  ; set DE <- store ASCII code table
34  34  000E2  63       MOV     A , B
35  35  000E3  99       MOV     [ DE ] , A
36  36  000E4  A5       INCW    DE
37  37  000E5  62       MOV     A , C
38  38  000E6  99       MOV     [ DE ] , A
39  39
40  40  000E7  EFFE     BR      $$
41  41
42  42                 END
```

2. オブジェクト・コードの埋め込みは，次のようになります。

- アセンブル・リスト

```
22  22  -----           CSEG
23  23  00000       START :
24  24
25  25                  ; chip initialize
26  26  00000 RCBF80000 MOVW    SP , #_@STBEG
27  27
28  28  00004  CD201A   MOV     HDTSA , #1AH
29  29  00007  3620FE   MOVW    HL , #LOWW ( HDTSA )  ; set hex 2-code data in HL registor
30  30
31  31  0000A RFD0000   CALL    !CONVAH               ; convert ASCII <- HEX
32  32                                                ; output BC-register <- ASCII code
33  33  0000D  3421FE   MOVW    DE , #LOWW ( STASC )  ; set DE <- store ASCII code table
34  34  00010  63       MOV     A , B
35  35  00011  99       MOV     [ DE ] , A
36  36  00012  A5       INCW    DE
37  37  00013  62       MOV     A , C
38  38  00014  99       MOV     [ DE ] , A
39  39
40  40  00015  EFFE     BR      $$
41  41
42  42                  END
```

- アブソリュート・アセンブル・リスト

```
22  22  -----           CSEG
23  23  000D2       START :
24  24
25  25                  ; chip initialize
26  26  000D2 RCBF820FE MOVW    SP , #_@STBEG
27  27
28  28  000D6  CD201A   MOV     HDTSA , #1AH
29  29  000D9  3620FE   MOVW    HL , #LOWW ( HDTSA )  ; set hex 2-code data in HL registor
30  30
31  31  000DC RFDE900   CALL    !CONVAH               ; convert ASCII <- HEX
32  32                                                ; output BC-register <- ASCII code
33  33  000DF  3421FE   MOVW    DE , #LOWW ( STASC )  ; set DE <- store ASCII code table
34  34  000E2  63       MOV     A , B
35  35  000E3  99       MOV     [ DE ] , A
36  36  000E4  A5       INCW    DE
37  37  000E5  62       MOV     A , C
38  38  000E6  99       MOV     [ DE ] , A
39  39
40  40  000E7  EFFE     BR      $$
41  41
42  42                  END
```

B. 7. 3　操作方法

(1) リスト・コンバータの起動方法

リスト・コンバータの起動には，2 つの方法があります。

(a) コマンド行での起動

```
X:[パス名]>lc78k0r[△オプション]...入力ファイル名[△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
lc78k0r	リスト・コンバータのコマンド名
オプション	リスト・コンバータに対して動作の詳細を指示します。 複数のリスト・コンバート・オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，リスト・コンバート・オプションに大文字，小文字の区別はありません。 リスト・コンバート・オプションについての詳細は，「B. 7. 4　オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（”“）で囲んでください。
入力ファイル名	アセンブル・リストのプライマリ・ネーム 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（”“）で囲んでください。 ファイルの拡張子は“.prn”にしてください。

注意　コマンド行にアセンブル・リストのプライマリ・ネームのみを指定する場合には，オブジェクト・モジュール・ファイル，ロード・モジュール・ファイルのプライマリ・ネームは，アセンブル・リスト・ファイルのプライマリ・ネームと同一でなければなりません。
また，ファイル・タイプは次のようになっていなければなりません。

ファイル名	タイプ
オブジェクト・モジュール・タイプ	.rel
ロード・モジュール・ファイル	.lmf

例　アセンブル・リスト・ファイル k0rmain.prn とロード・モジュール・ファイル sample.lmf のプライマリ・ネームが異なる場合，ロード・モジュール・ファイル sample.lmf の入力を指定します。

```
C:¥>lc78k0r k0rmain.prn -lsample.lmf
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，リスト・コンバートするたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>lc78k0r[ △入力ファイル名 ] △ -f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	リスト・コンバータの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[ △ ] オプション [ △オプション ]...
```

- コマンド行で入力ファイル名を省略した場合，パラメータ・ファイル内に入力ファイル名を記述します。
- 入力ファイル名は，オプションのあとにも記述することができます。
- パラメータ・ファイルには，コマンド行で指定するすべてのリスト・コンバート・オプション，出力ファイル名を記述します。

例　パラメータ・ファイル k0r.plv をエディタで作成し，リスト・コンバータを起動します。

```
; parameter file
k0rmain -lk0r.lmf
-ek0r.elv
```

```
C:¥>ra78k0r -fk0rmain.pra
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

リスト・コンバータが起動すると，次の実行開始メッセージが表示されます。

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

Pass1 : start ...
Pass2 : start ...
```

(b) 実行終了メッセージ

リスト・コンバートの結果，リスト・コンバート・エラーが検出されなかった場合，リスト・コンバータは次のメッセージを表示して制御をホスト OS に戻します。

```
Conversion complete.
```

リスト・コンバート中に，リスト・コンバータ処理継続が不可能な致命的エラーが検出された場合，リスト・コンバータはメッセージを表示して処理を中止し，制御をホスト OS に戻します。

例　存在しないリスト・コンバート・オプションを指定した場合

```
List Conversion Program for RA78K0R Vx.xx [xx xxx xxxx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

RA78K0R error F6018 : Option is not recognized '-a'
Please enter 'LC78K0R --', if you want help messages.
Program aborted.
```

(3) CubeSuite+ でのオプション設定

CubeSuite+ では，リスト・コンバート・オプションは，アセンブル・オプションに含まれています。

アセンブル・オプションの設定については，プロパティ パネルの［アセンブル・オプション］タブを参照してください。

B. 7. 4　オプション

(1) 種類

リスト・コンバート・オプションは，リスト・コンバータの動作に細かい指示を与えるものです。

リスト・コンバート・オプションの分類と説明を示します。

表 B—31　リスト・コンバート・オプション

分類	オプション	説明
オブジェクト・モジュール・ファイル入力指定	-r	オブジェクト・モジュール・ファイルを入力します。
ロード・モジュール・ファイル入力指定	-l	ロード・モジュール・ファイルを入力します。
アブソリュート・アセンブル・リスト・ファイル出力指定	-o	アブソリュート・アセンブル・リスト・ファイルを出力します。
エラー・リスト・ファイル出力指定	-e	エラー・リスト・ファイルを出力します。
	-ne	
パラメータ・ファイル指定	-f	入力ファイル名，オプションを指定したファイルより入力します。
ヘルプ指定	--	ディスプレイにヘルプ・メッセージを出力します。

オブジェクト・モジュール・ファイル入力指定

オブジェクト・モジュール・ファイル入力指定オプションには，次のものがあります。

- -r

-r

[記述形式]

```
-r[入力ファイル名]
```

- 省略時解釈

  -r *アセンブル・リスト・ファイル名* .rel

[機能]

- -r オプションは，オブジェクト・モジュール・ファイルの入力を指示します。

[用途]

- オブジェクト・モジュール・ファイルのプライマリ・ネームが，アセンブル・リスト・ファイルのプライマリ・ネームと異なる場合，またはファイル・タイプが“.rel”でない場合は，-r オプションを指定します。

[説明]

- フェイタル・エラーがある場合は，アブソリュート・アセンブル・リスト・ファイルは出力されません。
- 入力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして“.rel”を付加してファイルを入力します。

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn とオブジェクト・モジュール・ファイル sample.rel のプライマリ・ネームが異なる場合，ロード・モジュール・ファイル sample.rel の入力を指定します。

```
C:¥>lc78k0r k0rmain.prn -lsample.rel
```

ロード・モジュール・ファイル入力指定

ロード・モジュール・ファイル入力指定オプションには，次のものがあります。

- -l

-l

[記述形式]

```
-l[入力ファイル名]
```

- 省略時解釈

  -l *アセンブル・リスト・ファイル名*.lmf

[機能]

- -l オプションは，ロード・モジュール・ファイルの入力を指定します。

[用途]

- ロード・モジュール・ファイルのプライマリ・ネームがアセンブル・リスト・ファイルのプライマリ・ネームと異なる場合，またはファイル・タイプが“.lmf”でない場合に，-l オプションを指定します。

[説明]

- フェイタル・エラーがある場合は，アブソリュート・アセンブル・リスト・ファイルは出力されません。
- 入力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして“.lmf”を付加してファイルを入力します。

[使用例]

- アセンブル・リスト・ファイル k0rmain.prn とロード・モジュール・ファイル sample.lmf のプライマリ・ネームが異なる場合，ロード・モジュール・ファイル sample.lmf の入力を指定します。

```
C:¥>lc78k0r k0rmain.prn -lsample.lmf
```

アブソリュート・アセンブル・リスト・ファイル出力指定

アブソリュート・アセンブル・リスト・ファイル出力指定オプションには，次のものがあります。

- -o

-o

[記述形式]

```
-o [ 出力ファイル名 ]
```

- 省略時解釈

  -o *アセンブル・リスト・ファイル名* .p

[機能]

- -o オプションは，アブソリュート・アセンブル・リスト・ファイルの出力を指定します。
  また，その出力先や出力ファイル名も指定することができます。

[用途]

- アブソリュート・アセンブル・リスト・ファイルの出力先や出力ファイル名を変更したいときに，-o オプションを指定します。

[説明]

- ファイル名にエラー・ファイルと同一のデバイスが指定された場合は，アボート・エラーとなります。
- -o オプションを指定する際に出力ファイル名を省略すると，アブソリュート・アセンブル・リスト・ファイル名は“*アセンブル・リスト・ファイル名* .p”となります。
- 出力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして“.p”を付加してファイルを出力します。
- -o オプションを指定する際にドライブ名を省略すると，カレント・ドライブにアブソリュート・アセンブル・リスト・ファイルが出力されます。

[使用例]

- アブソリュート・アセンブル・リスト・ファイル sample.p を作成します。

```
C:¥>lc78k0r k0rmain.prn -osample.p -lk0r.lmf
```

エラー・リスト・ファイル出力指定

エラー・リスト・ファイル出力指定オプションには，次のものがあります。

- -e/-ne

-e/-ne

[記述形式]

```
-e [ 出力ファイル名 ]
-ne
```

- 省略時解釈

  -ne

[機能]

- -e オプションは，エラー・リスト・ファイルの出力を指定します。
  また，その出力先や出力ファイル名も指定することができます。
- -ne オプションは，-e オプションを無効にします。

[用途]

- エラー・メッセージをファイルに保存しておきたい場合には，-e オプションを指定します。

[説明]

- ファイル名にアブソリュート・アセンブル・リスト・ファイルと同一のデバイスを指定した場合は，アボート・エラーとなります。
- -e オプションを指定する際に出力ファイル名を省略すると，エラー・リスト・ファイル名は“*アセンブル・リスト・ファイル名*.elv”となります。
- 出力ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして“.elv”を付加してファイルを出力します。
- -e オプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。
- -e と -ne オプションを同時に指定した場合は，あとで指定した方が有効となります。

［使用例］

- エラー・リスト・ファイル sample.elv を作成します。

```
C:¥>lc78k0r k0rmain.prn -esample.elv
```

エラー・リスト・ファイル sample.elv の内容は，以下のようになります。

```
RA78K0R warning W6701: Load module file is older than object module file 'k0rmain.lmf,
k0rmain.rel'

Pass1 : start


RA78K0R warning W6702: Load module file is older than assemble module file 'k0rmain.lmf,
k0rmain.prn'


Pass2 : start
```

パラメータ・ファイル指定

パラメータ・ファイル指定オプションには，次のものがあります。

- -f

-f

[記述形式]

```
-f ファイル名
```

- 省略時解釈
  コマンド行上からのみオプション，または入力ファイル名の入力が可能となります。

[機能]

- -f オプションは，オプション，あるいは入力ファイル名を指定のファイルから入力することを指定します。

[用途]

- コマンド行では，リスト・コンバータの起動に必要な情報を指定しきれないときに，-f オプションを指定します。
- リスト・コンバートするたびに繰り返し同じようにオプションを指定し，リスト・コンバートする場合には，それらをパラメータ・ファイルに記述しておき，-f オプションを指定します。

[説明]

- ファイル名を省略すると，アボート・エラーとなります。
- ファイル名のプライマリ・ネームのみを指定した場合は，ファイル・タイプとして“.plv”を付加してファイルをオープンします。
- パラメータ・ファイルのネストは許されません。パラメータ・ファイル中で -f オプションを指定すると，アボート・エラーとなります。
- パラメータ・ファイル中に記述可能な文字数に，制限はありません。
- 空白とタブ，および改行文字（LF）をオプション，あるいは入力ファイル名の区切りとします。
- パラメータ・ファイル中に記述したオプション，あるいは入力ファイル名は，コマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
- 展開されたオプションは，あとで指定した方が有効となります。
- -f オプションを複数指定すると，アボート・エラーとなります。
- “;”，または“#”以降に記述された文字は，改行文字（LF），または EOF の前まですべてコメントと解釈されます。

［使用例］

- パラメータ・ファイル k0r.plv を使用してリスト・コンバータを起動させます。

パラメータ・ファイル k0r.plv の内容は，以下のようになります。

```
: parameter file
k0rmain -lk0r.lmf
-ek0r.elv
```

コマンド行には，次のように入力します。

```
C:¥>lc78k0r -fk0r.plv
```

ヘルプ指定

ヘルプ指定オプションには，次のものがあります。

- --

--

[記述形式]

```
--
```

- 省略時解釈

  表示しません。

[機能]

--- オプションは，ヘルプ・メッセージをディスプレイに出力します。

[用途]

- ヘルプ・メッセージは，リスト・コンバート・オプションとその説明の一覧です。リスト・コンバータを実行するときに参照してください。

[説明]

--- オプションを指定すると，ほかのリスト・コンバート・オプションはすべて無効となります。

注意　本オプションは，CubeSuite+ 上では指定することはできません。

[使用例]

--- オプションを指定すると，ヘルプ・メッセージがディスプレイに出力されます。

```
C:¥>lc78k0r --
```

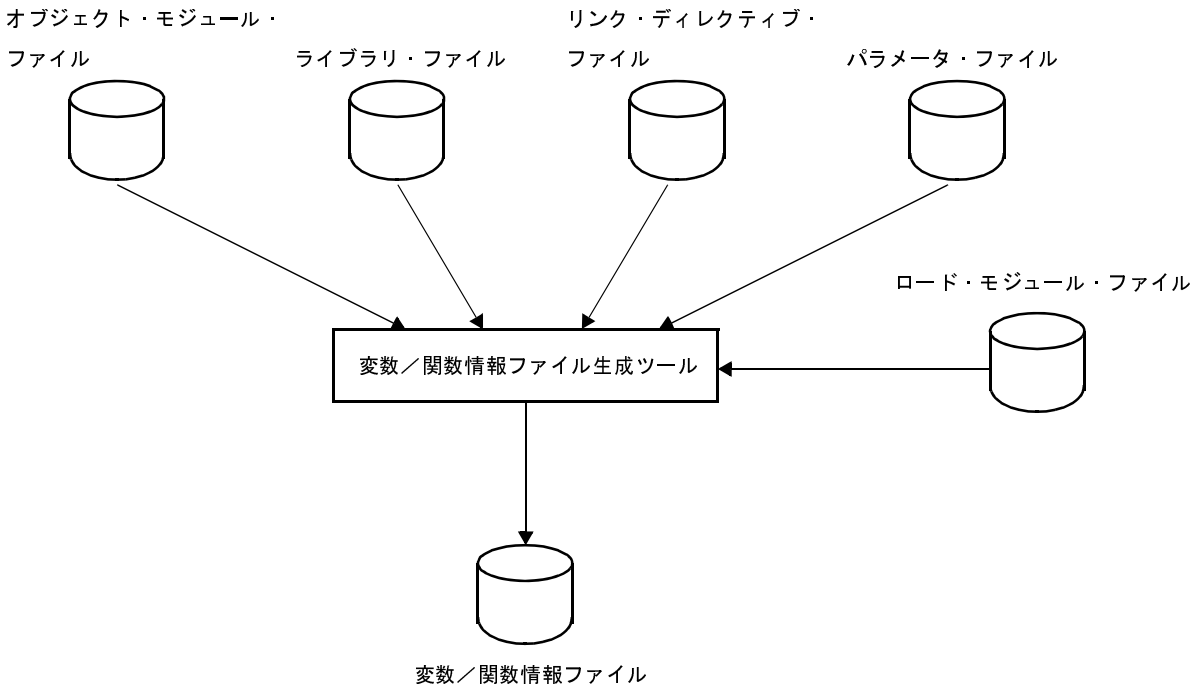
```
List Conversion Program for RA78K0R Vx.xx [xx xxx xx]
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

usage : LC78K0R [option[...]] input-file [option[...]]
The option is as follows ([] means omissible).
-r[file]: Specify object module file.
-l[file]: Specify load module file.
-o[file]: Specify output list file (absolute assemble list file).
-ffile  : Input option or input-file name from specified file.
-e[file]: Create error list file.
--      : Show this message.
```

B. 8　変数／関数情報ファイル生成ツール

変数／関数情報ファイル生成ツールは，C コンパイラ，アセンブラが出力したいくつかのオブジェクト・モジュール・ファイルを入力し，変数や関数を効率よく配置するための情報をまとめた変数／関数情報ファイルを出力します。

エラーがある場合は，エラー・メッセージを出力し，エラーの原因を明示します。なお，エラーがある場合，変数／関数情報ファイルを出力しません。

図 B—39　変数／関数情報ファイル生成ツールの入出力ファイル



B. 8. 1　入出力ファイル

変数／関数情報ファイル生成ツールの入出力ファイルを次に示します。

出力ファイルについての詳細は，「3. 8　変数／関数情報ファイル生成ツール」を参照してください。

表 B—32　変数／関数情報ファイル生成ツールの入出力ファイル

種類	ファイル名	説明	デフォルト・ファイル・タイプ
入力ファイル	オブジェクト・モジュール・ファイル	- 機械語情報と機械語の配置アドレスに関する再配置情報，およびシンボル情報を含んだバイナリ・ファイル - コンパイラ，またはアセンブラの出力したファイル	.rel
	ライブラリ・ファイル	- 複数のオブジェクト・モジュール・ファイルが登録されたファイル - ライブラリアンの出力したファイル	.lib
	リンク・ディレクティブ・ファイル	- リンカに対するリンク指示を記述したファイル（ユーザ作成ファイル）	.dr
	パラメータ・ファイル	- 実行プログラムのパラメータを内容とするファイル（ユーザ作成ファイル）	.plk
	ロード・モジュール・ファイル	- セルフ・プログラミング時に再入力するロード・モジュール・ファイル	.lmf
出力ファイル	変数／関数情報ファイル	- 参照されるすべての変数／関数の一覧で，saddr 領域／ callt テーブル領域への配置指示を記述したファイル	.vfi

B. 8. 2　機　　能

（1）変数／関数情報ファイルの生成

変数や関数のリロケーションを解決する際に参照回数をカウントし，効率よく配置するための情報をファイルに出力します。

この情報ファイルを利用することにより，C コンパイラは saddr 領域と callt テーブル領域に対して最適な配置指定を行い，コードの削減を図ることができるようになります。

（2）ROM/RAM 使用量の表示

リンク後の ROM/RAM 使用量を標準出力に表示します。

B. 8. 3　変数／関数情報

（1）領域について

（a）saddr 領域

RL78，および 78K0R には，saddr アドレッシング（ショート・ダイレクト・アドレッシング）というアドレスを 8 ビットで特定できる領域があります。

saddr アドレッシングは FFE20H からの 256 バイトの領域ですが，この中には汎用レジスタやポートが含まれており，ユーザ変数の配置には注意が必要です。そこで，変数／関数情報ファイル生成ツールが配置の対象とする saddr 領域は，FFE20H ～ FFEDFH の 192 バイトとします。

（b）CALLT テーブル領域

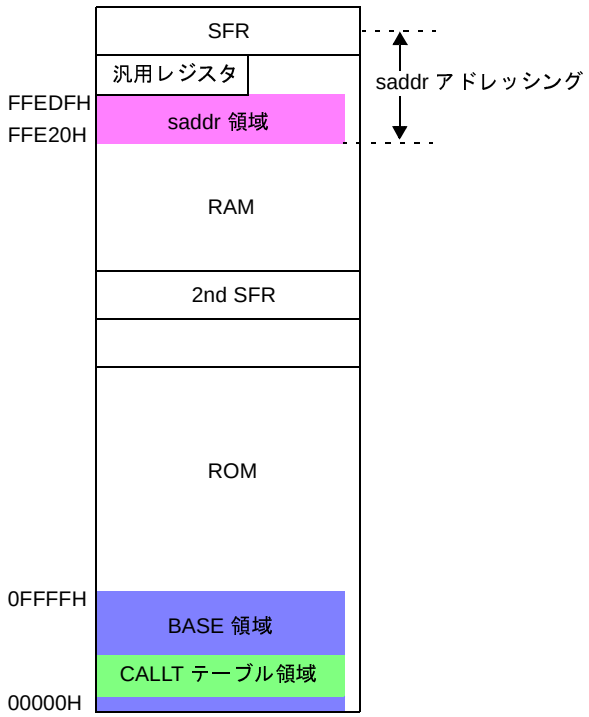
00080H ～ 000BFH の領域で，32 個の BASE 領域アドレスを分岐先として登録することができます。

（c）BASE 領域

00000H ～ 0FFFFH の領域で，2 バイト・コール命令（CALLT）の分岐先です。

アセンブラにより定義づけられた再配置属性 BASE により，指定することができます。

図 B—40　メモリ・マップ



(2) 変数情報

(a) 参照回数をカウント

リロケーション解決時に，参照シンボルの参照回数をカウントします。

(b) 空き領域の検出

通常配置後の saddr 領域の空き領域の開始アドレスとサイズを検出します。

(c) 優先順位の決定

1 バイトあたりのコード削減率を考慮し，以下の式により算出された値の高い順に優先順位を決定します。

参照回数÷シンボル・サイズ×参照タイプ[注]

注　参照タイプ

near　：1（near 領域から saddr 領域への変更は 1 バイトのコード削減）

far　：2（far 領域から saddr 領域への変更は 2 バイトのコード削減）

sreg　：0（sreg 指定により，すでに saddr 領域に配置されている変数については配置対象外）

例

変数	参照回数	シンボル・サイズ	参照タイプ	優先順位
sym1	10 回	2 バイト	near	10 ÷ 2 × 1 ＝ 5
sym2	6 回	1 バイト	far	6 ÷ 1 × 2 ＝ 12
sym3	6 回	1 バイト	sreg	6 ÷ 1 × 0 ＝ 0

参照回数が 10 回でも 2 バイト必要な変数で near である変数 sym1 よりも，参照回数が 6 回で far である変数 sym2 を saddr 領域に変更したほうが，コード効率が良くなります。

変数 sym3 はすでに saddr 領域に配置されているため，配置対象外です。

備考　優先順位付けの対象外である変数を以下に示します。

const 変数	const 変数はミラー元領域へ配置するので，saddr 領域への配置対象外とします。 ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。
sreg 変数	すでに sreg 指定してしている変数は配置対象外とします。 ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。
static 変数	static 変数は，ファイル内，関数内ともに配置対象外とします。 ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。
C ソース中に定義のない変数	C ソース中に定義のない変数は配置対象外とします（例：アセンブラ・ソース中で定義，ランタイム・ライブラリ内で定義）。 出力ファイルにも出力しません。

フラッシュ領域側で参照するブート領域側で定義した変数	フラッシュ領域側で参照するブート領域側で定義した変数は配置対象外とします。ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。
未参照変数	出力ファイルにも出力しません。

（d）アライメントの考慮

奇数アドレスに配置可能な変数を以下に示します。

- サイズが 1 バイトの変数（char，unsigned char，列挙型，構造体，共用体）
- サイズが 1 バイトの変数（char，unsigned char）の配列
- サイズが 1 バイトの列挙型，構造体，共用体の変数の配列で，かつ要素数が 1 の場合

（3）関数情報

（a）参照回数をカウント

リロケーション解決時に，参照シンボルの参照回数をカウントします。

（b）空き領域の検出

通常配置後の callt 領域，および BASE 領域の空き領域の開始アドレスとサイズを検出します。

（c）優先順位の決定

以下の式により算出された値の高い順に優先順位を決定します。

参照回数×参照タイプ[注]

注　参照タイプ

near　：1（near 領域から callt 領域への変更は 1 バイトのコード削減）

far　：2（far 領域から callt 領域への変更は 2 バイトのコード削減）

callt　：0（すでに callt 領域に配置されている関数については配置対象外）

例

関数	参照回数	参照タイプ	優先順位
func1	10 回	near	10 × 1 ＝ 10
func2	10 回	far	10 × 2 ＝ 20
func3	10 回	callt	10 × 0 ＝ 0

参照回数が 10 回で near である関数 func1 よりも，参照回数が 10 回で far である関数 func2 を callt 領域に変更したほうが，コード効率が良くなります。

関数 func3 はすでに callt 領域に配置されているため，配置対象外です。

備考　優先順位付けの対象外である関数を以下に示します。

フラッシュ領域側の関数	フラッシュ領域側の関数は callt 領域への登録ができないため，配置対象外とします。 ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。 また，フラッシュ領域側から参照しているブート領域側も配置対象外とし，コメントとして出力ファイルへ出力します。
callt 関数	すでに callt テーブルに登録されているため，配置対象外とします。 ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。
static 関数	static 関数は，ファイル内，関数内ともに配置対象外とします。 ただし，参照回数のカウントは行い，コメントとして出力ファイルへ出力します。
C ソース中に定義のない関数	C ソース中に定義のない関数は配置対象外とします（例：アセンブラ・ソース中で定義，ランタイム・ライブラリ内で定義）。 出力ファイルにも出力しません。
未参照関数	出力ファイルにも出力しません。

（d）関数の配置状態の考慮

near 参照の場合は，該当関数がすでに BASE 領域に配置されているので，BASE 領域の空き状況を考慮する必要はありません。far 参照の場合は，該当関数がどこに配置されているかはわかりません。

- near 参照の場合

  該当関数がすでに BASE 領域に配置されているので，callt 領域に登録します。

- far 参照で該当関数が BASE 領域外の場合

  BASE 領域に空きがあれば，該当関数を callt 領域に登録します。

  BASE 領域に空きがなければ，該当関数を callt 領域に登録しません。

- callt 参照の場合

  該当関数がすでに callt テーブルに登録されているので，除外します。

（4）変数／関数情報ファイルに出力しないシンボル

以下のシンボルは，変数／関数情報ファイルに出力しません。

- 未参照
- ライブラリ内で定義
- ロード・モジュール以外の EXTERN
- アセンブラ・ソース中で定義
- 所属セグメントの再配置属性が AT
- RTOS 用のタスク，RTOS 用の割り込みハンドラ
- ファーム ROM 関数
- ベクタ割り込み関数
- シンボル・タイプが T_NULL

B. 8. 4　操作方法

(1) 変数／関数情報ファイル生成ツールの起動方法

変数／関数情報ファイル生成ツールの起動には，2 つの方法があります。

(a) コマンド行での起動

```
X:[パス名]>vf78k0r[△オプション]...オブジェクト・モジュール・ファイル名[△オブジェクト・モジュール・ファイル名]...[△オプション]...
```

X	カレント・ドライブ名
パス名	カレント・フォルダ名
vf78k0r	変数／関数情報ファイル生成ツールのコマンド名
オプション	変数／関数情報ファイル生成ツールに対して動作の詳細を指示します。 複数の変数／関数配置オプションを指定する場合は，それぞれのオプション間を空白で区切ってください。なお，変数／関数配置オプションに大文字，小文字の区別はありません。変数／関数配置オプションについての詳細は，「B. 8. 5　オプション」を参照してください。 空白を含むパスを設定する場合は，ダブルクオーテーション（”“）で囲んでください。
オブジェクト・モジュール・ファイル名	変数／関数情報ファイルを生成するオブジェクト・モジュール・ファイル名 入力モジュールとして，最大 1024 個入力できます。 空白を含むパスのファイル名を指定する場合は，ダブルクオーテーション（”“）で囲んでください。

注意　オプション，およびオブジェクト・モジュール・ファイル名には，リンカと同じものを指定した上で，変数／関数情報ファイル生成ツール独自のオプションを付加してください。

例　変数／関数情報ファイル info.vfi を出力します。

```
C:¥>vf78k0r main.rel sub.rel -voinfo.vfi
```

(b) パラメータ・ファイルによる起動

パラメータ・ファイルは，起動に必要な情報がコマンド行に指定しきれない場合や，変数／関数情報ファイルを生成するたびに同じオプションを繰り返し指定するような場合に使用します。

パラメータ・ファイルを使用する場合には，コマンド行にパラメータ・ファイル指定オプション（-f）を指定します。

パラメータ・ファイルによる起動方法は，次のようになります。

```
X>vf78k0r[△オブジェクト・モジュール・ファイル]△-f パラメータ・ファイル名
```

-f	パラメータ・ファイル指定オプション
パラメータ・ファイル名	変数／関数情報ファイル生成ツールの起動に必要な情報を含んだファイル

備考　パラメータ・ファイルは，エディタなどで作成します。

パラメータ・ファイル内での記述規則を次に示します。

```
[△] オプション [△オプション]...
```

- コマンド行でソース・ファイル名を省略した場合は，パラメータ・ファイル内でソース・ファイル名を１つだけ指定することができます。
- ソース・ファイル名は，オプションのあとに記述することも可能です。
- パラメータ・ファイルには，コマンド行で指定するすべての変数／関数配置オプション，出力ファイル名を記述します。

例　パラメータ・ファイル sample.plk をエディタで作成し，変数／関数情報ファイル生成ツールを起動します。

```
; parameter file
main.rel sub.rel -osample.lmf -psample.map -e
-tC:¥tmp
```

```
C:¥>vf78k0r -fsample.plk -voinfo.vfi
```

(2) 実行開始メッセージ，終了メッセージ

(a) 実行開始メッセージ

変数／関数情報ファイル生成ツールが起動すると，次の実行開始メッセージが表示されます。

```
78K0R Var-Func-Inf Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation
```

(b) 実行終了メッセージ

変数／関数情報ファイル生成ツールの実行の結果，エラーが検出されなかった場合，変数／関数情報ファイル生成ツールは次のメッセージを表示して制御をホスト OS に戻します。

```
Target chip : uPD78F1166_A0
Device file : Vx.xx

VF check complete, 0 error(s) and 0 warning(s) found.
```

変数／関数情報ファイル生成ツールの実行中に処理継続が不可能な致命的エラーが検出された場合，変数／関数情報ファイル生成ツールはメッセージを表示して変数／関数情報ファイル生成ツールの実行を中止し，制御をホスト OS に戻します。

例を以下に示します。

- 存在しないオブジェクト・モジュール・ファイルを指定した場合

```
C:¥>vf78k0r samp1.rel samp2.rel -vosamp.vfi
```

```
78K0R Var-Func-Inf Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

VF78K0R error F0006 : File not found 'samp1.rel'
VF78K0R error F0006 : File not found 'samp2.rel'
Program Aborted.
```

この例では，存在しないオブジェクト・モジュール・ファイルを指定したためにエラーとなり，変数／関数情報ファイル生成ツールの実行が中止されます。

- 存在しない変数／関数配置オプションを指定した場合

```
C:¥>vf78k0r main.rel sub.rel -z
```

```
78K0R Var-Func-Inf Vx.xx [xx xxx xxxx]
 for RL78,78K0R Microcontroller
   Copyright(C) xxxx-xxxx Renesas Electronics Corporation

VF78K0R error F0018 : Option is not recognized '-z'
Program Aborted.
```

この例では，存在しない変数／関数配置オプションを指定したためにエラーとなり，リンクが中止されます。

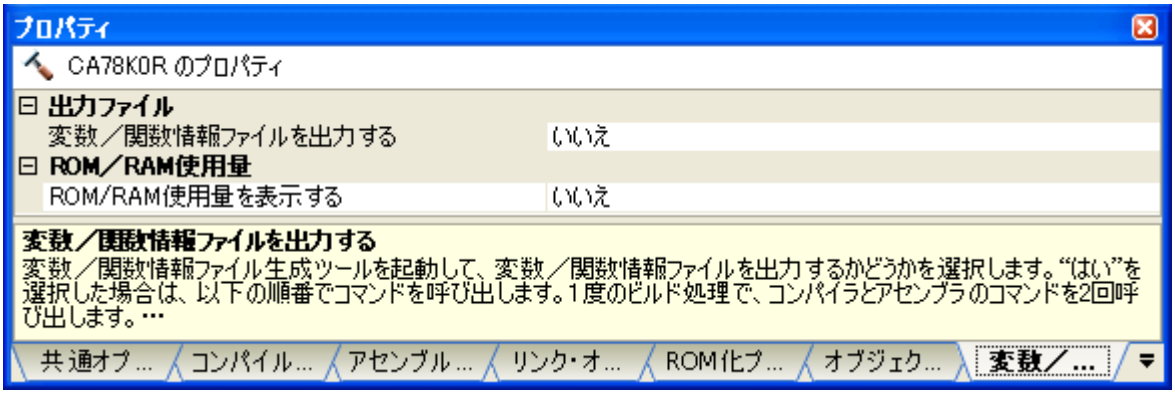
(3) CubeSuite+ でのオプション設定

CubeSuite+ から変数／関数配置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，［変数／関数配置オプション］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：［変数／関数配置オプション］タブ



B. 8. 5　オプション

（1）種類

変数／関数配置オプションは変数／関数情報ファイル生成ツールの動作に細かい指示を与えるものです。

変数／関数配置オプションの分類と説明を示します。

表 B—33　変数／関数配置オプション

分類	オプション	説明
変数／関数情報ファイル出力指定	-vo	変数／関数情報ファイルの出力を指定します。
saddr 空き領域指定	-vs	saddr 領域のマージン・サイズを指定します。
ROM/RAM 使用量表示指定	-vx	リンク後の ROM/RAM 使用量を標準出力に出力します。
コピー・ルーチン・アドレス指定	-rc	ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。
ROM 化領域指定	-ra	ROM 化対象領域を指定します。

変数／関数情報ファイル出力指定

変数／関数情報ファイル出力指定オプションには，次のものがあります。

- -vo

-vo

[記述形式]

```
-vo 出力ファイル名
```

- 省略時解釈
  省略することはできません（-vx オプション指定時を除く）。

[機能]

- -vo オプションは，変数／関数情報ファイルの出力を指定します。
  また，その出力先や出力ファイル名を指定します。

[用途]

- 変数／関数情報ファイルの出力を指定したいときに，-vo オプションを指定します。

[説明]

- デフォルト・ファイル・タイプは“.vfi”です。
- “*出力ファイル名*”には，パス名も含めて指定をすることができます。
- -vo オプションを指定してもリンクを行うまでにエラーがある場合は，変数／関数情報ファイルは出力されません。
- -vo オプションは，-vx オプションと同時に指定することはできません。

[使用例]

- 変数／関数情報ファイル info.vfi を出力します。

```
C:¥>vf78k0r main.rel sub.rel -voinfo.vfi
```

saddr 空き領域指定

saddr 空き領域指定オプションには，次のものがあります。

- -vs

-vs

[記述形式]

```
-vs[サイズ]
```

- 省略時解釈
  -vs0

[機能]

- 本ツールで変数を saddr 領域に割り当てた後，コンパイル，リンクを行ったとき，処理の順番やアライメントの関係で配置エラーになる場合があります。このとき，saddr 領域にマージンをもたせて配置することにより，そのエラーを回避することができます。
  -vs オプションは，saddr 領域のマージン・サイズを指定します。

[用途]

- 本ツールで変数を saddr 領域に割り当てた後，コンパイル，リンクを行った場合に発生する配置エラーを回避したいときに，-vs オプションを指定します。

[説明]

- “*サイズ*”には，saddr 領域のマージン・サイズ（バイト数）を指定します。
- 10 進，16 進，2 進での指定が可能です。
  指定可能な値は，10 進で 192 までです。193 以上を指定すると，エラーとなります。
- すべての空き領域に対して実際の空き領域よりも大きい値を指定すると，エラーとなります。
- -vs オプションは，-vo オプションを指定した場合のみ有効です。

[使用例]

- saddr 領域のマージン・サイズを 10 バイト（10 進数）に指定します。

```
C:¥>vf78k0r main.rel sub.rel -voinfo.vfi -vs10
```

- saddr 領域のマージン・サイズを 0AH バイト（16 進数）に指定します。

```
C:¥>vf78k0r main.rel sub.rel -voinfo.vfi -vs0AH
```

- saddr 領域のマージン・サイズを 1010B バイト（2 進数）に指定します。

```
C:¥>vf78k0r main.rel sub.rel -voinfo.vfi -vs1010B
```

ROM/RAM 使用量表示指定

ROM/RAM 使用量表示指定オプションには，次のものがあります。

- -vx

-vx

[記述形式]

```
-vx
```

- 省略時解釈

  ROM/RAM 使用量を標準出力に出力しません。

[機能]

- リンク後の ROM/RAM 使用量を標準出力に出力します。

[用途]

- リンク後の ROM/RAM 使用量を出力したいときに，-vx オプションを指定します。

[説明]

- -vx オプションは，-vo オプションと同時に指定することはできません。
- ROM/RAM 使用量の出力例を以下に示します。
  - デフォルトのメモリ領域名を使用した場合

```
*** Memory Area Information ***
ROM : xxxxxH byte(s) real data
RAM : xxxxxH byte(s) real data

*** Memory Area Information in ROM ***
ROM : xxxxxH byte(s)

*** Memory Area Information in RAM ***
RAM : xxxxxH byte(s)
```

- メモリ領域名をメモリ・ディレクティブにより定義した場合

```
*** Memory Area Information ***
ROM : xxxxxH byte(s) real data
RAM : xxxxxH byte(s) real data

*** Memory Area Information in ROM ***
ROM : xxxxxH byte(s)
ROM1 : xxxxxH byte(s)

*** Memory Area Information in RAM ***
RAM : xxxxxH byte(s)
RAM1 : xxxxxH byte(s)
```

最初に使用量の合計値が出力され，次に定義したメモリ領域ごとに使用量が出力されます。

[使用例]

- リンク後の ROM/RAM 使用量を標準出力に出力します。

```
C:¥>vf78k0r main.rel sub.rel -vx
```

コピー・ルーチン・アドレス指定

コピー・ルーチン・アドレス指定オプションには，次のものがあります。

- -rc

-rc

[記述形式]

```
-rc アドレス
```

- 省略時解釈

  コピー・ルーチンを ROM 領域の空き領域へ配置します。

[機能]

- -rc オプションは，ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。

[用途]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定したいときに，-rc オプションで指定します。

[説明]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを配置するアドレスを指定します。

[使用例]

- ROM 化したセグメントを RAM 領域へ展開するためのコピー・ルーチンを 300H 番地に配置します。

```
C:¥>vf78k0r main.rel -rc300H
```

ROM 化領域指定

ROM 化領域指定オプションには，次のものがあります。

- -ra

-ra

[記述形式]

```
-ra 開始アドレス，終了アドレス
```

- 省略時解釈

  内部 RAM 領域を ROM 化対象とします。

[機能]

- -ra オプションは，ROM 化対象領域を指定します。

[用途]

- ROM 化対象領域を指定したいときに，-ra オプションで指定します。

[説明]

- ROM 化対象領域の開始アドレスと終了アドレスを指定します。

[使用例]

- 0FCF00H 番地から FFFFFH を ROM 化対象とします。

```
C:¥>vf78k0r main.rel -ra0FCF00H,0FFFFFH
```

付録C 索 引

【記号】

--/-?/-h （CC78K0R） … 407

-- （LB78K0R） … 615

-- （LC78K0R） … 642

-- （LK78K0R） … 515

-- （OC78K0R） … 601

-- （RA78K0R） … 458

-- （RP78K0R） … 564

.asm … 409

.cer … 343

.dr … 461, 645

.ecc … 343

.elk … 461

.elv … 629

.eoc … 566

.er … 343

.era … 409

.erp … 533

.her … 343

.hex … 566

.lib … 461, 604, 645

.lmf … 461, 532, 533, 565, 629, 645

.lst … 603

.map … 461, 533

.p … 629

.plk … 461, 645

.plv … 629

.poc … 565

.ppr … 532

.pra … 409

.prn … 409, 629

.rel … 409, 461, 604, 629, 645

.sym … 566

.vfi … 645

【A】

-a （CC78K0R） … 375

add … 619

【B】

-b （LK78K0R） … 491

【C】

-c （CC78K0R） … 354

-c （RA78K0R） … 417

-ccza （LK78K0R） … 511

-common (CC78K0R) … 405

-common （RA78K0R） … 456

-compati（RA78K0R） … 457

-crc （OC78K0R） … 599

create … 618

C コンパイラ … 342

【D】

-d （CC78K0R） … 371

-d （LK78K0R） … 474

-d （RA78K0R） … 455

delete … 620

【E】

-e （CC78K0R） … 380

-e （LC78K0R） … 638

-e （LK78K0R） … 489

-e （OC78K0R） … 591

-e （RA78K0R） … 445

-e （RP78K0R） … 556

exit … 627

【F】

-f （CC78K0R） … 394

-f （LC78K0R） … 640

-f （LK78K0R） … 495

-f （OC78K0R） … 592

-f （RA78K0R） … 447

-f （RP78K0R） … 558

【G】

-g （CC78K0R） … 365

-g （LK78K0R） … 471
-g （RA78K0R） … 421
-ga （RA78K0R） … 423
-gb （LK78K0R） … 507
-gi （LK78K0R） … 505
-go （LK78K0R） … 503

【H】
help … 626

【I】
-i （CC78K0R） … 373
-i （LK78K0R） … 493
-i （RA78K0R） … 425
INC78K0R … 373

【J】
-j （LK78K0R） … 470
-j （RA78K0R） … 420

【K】
-k （CC78K0R） … 369
-ka （RA78K0R） … 428
-kd （LK78K0R） … 479
-ki （OC78K0R） … 594
-kie （OC78K0R） … 594
-kl （LK78K0R） … 483
-kl （RP78K0R） … 549
-km （LK78K0R） … 477
-km （OC78K0R） … 594
-km （RP78K0R） … 544
-kme （OC78K0R） … 594
-kp （LK78K0R） … 481
-kp （RP78K0R） … 547
-ks （RA78K0R） … 430
-kt （OC78K0R） … 594
-kx （RA78K0R） … 432

【L】
-l （LC78K0R） … 636
-lf （CC78K0R） … 389
-lf （LB78K0R） … 612
-lf （LK78K0R） … 488
-lf （RA78K0R） … 444
-lf （RP78K0R） … 555
-lh （RA78K0R） … 438
-li （CC78K0R） … 390
list … 624
-ll （CC78K0R） … 387
-ll （LB78K0R） … 611
-ll （LK78K0R） … 485
-ll （RA78K0R） … 436
-ll （RP78K0R） … 552
-lt （CC78K0R） … 388
-lt （RA78K0R） … 441
-lw （CC78K0R） … 386
-lw （LB78K0R） … 610
-lw （RA78K0R） … 434

【M】
-m （CC78K0R） … 402
-ma (CC78K0R) … 406
-mi （CC78K0R） … 404
-mi （LK78K0R） … 509

【N】
-nccza （LK78K0R） … 511
-ncompati （RA78K0R） … 457
-ne （LC78K0R） … 638
-ne （LK78K0R） … 489
-ne （OC78K0R） … 591
-ne （RA78K0R） … 445
-ne （RP78K0R） … 556
-ng （CC78K0R） … 365
-ng （LK78K0R） … 471
-ng （RA78K0R） … 421
-nga （RA78K0R） … 423
-nj （LK78K0R） … 470
-nj （RA78K0R） … 420
-nka （RA78K0R） … 428
-nkd （LK78K0R） … 479
-nkl （LK78K0R） … 483
-nkl （RP78K0R） … 549

-nkm （LK78K0R） … 477
-nkm （RP78K0R） … 544
-nkp （LK78K0R） … 481
-nkp （RP78K0R） … 547
-nks （RA78K0R） … 430
-nkx （RA78K0R） … 432
-nlf （LB78K0R） … 612
-nlf （LK78K0R） … 488
-nlf （RA78K0R） … 444
-nlf （RP78K0R） … 555
-no （CC78K0R） … 355
-no （LK78K0R） … 469
-no （OC78K0R） … 584
-no （RA78K0R） … 418
-no （RP78K0R） … 541
-np （LK78K0R） … 476
-np （RA78K0R） … 427
-np （RP78K0R） … 543
-nq （CC78K0R） … 362
-nr （CC78K0R） … 357, 359, 361
-nr （OC78K0R） … 587
-ns （LK78K0R） … 472
-ns （OC78K0R） … 586
-nu （OC78K0R） … 588
-nv （CC78K0R） … 393
-nz （CC78K0R） … 397

【O】
-o （CC78K0R） … 355
-o （LC78K0R） … 637
-o （LK78K0R） … 469
-o （OC78K0R） … 584
-o （RA78K0R） … 418
-o （RP78K0R） … 541

【P】
-p （CC78K0R） … 367
-p （LK78K0R） … 476
-p （RA78K0R） … 427
-p （RP78K0R） … 543
pick … 623

【Q】
-q （CC78K0R） … 362

【R】
-r （CC78K0R） … 357
-r （LC78K0R） … 635
-r （OC78K0R） … 587
-ra （LK78K0R） … 514
-ra （RP78K0R） … 563
-ra （VF78K0R） … 660
-rc （LK78K0R） … 513
-rc （RP78K0R） … 562
-rc （VF78K0R） … 659
-rd （CC78K0R） … 359
replace … 621
ROM 化プロセス・オプションの設定 … 50
ROM 化プロセッサ … 532
-rs （CC78K0R） … 361

【S】
-s （LK78K0R） … 472
-s （OC78K0R） … 586
-sa （CC78K0R） … 377
-se （CC78K0R） … 382
-self （LK78K0R） … 512
-selfw （LK78K0R） … 512

【T】
-t （CC78K0R） … 396
-t （LB78K0R） … 613
-t （LK78K0R） … 497
-t （RA78K0R） … 449

【U】
-u （CC78K0R） … 372
-u （OC78K0R） … 588

【V】
-v （CC78K0R） … 393
-vo （VF78K0R） … 654
-vs （VF78K0R） … 655
-vx （VF78K0R） … 657

【W】
-w （CC78K0R） … 391
-w （LK78K0R） … 501

【X】
-x （CC78K0R） … 384

【Y】
-y （CC78K0R） … 400
-y （LK78K0R） … 499
-y （OC78K0R） … 596
-y （RA78K0R） … 453
-y （RP78K0R） … 560

【Z】
-z （CC78K0R） … 397
-zb （LK78K0R） … 502
-ze （RA78K0R） … 451
-zf （OC78K0R） … 598
-zn （RA78K0R） … 451
-zs （RA78K0R） … 451

【あ行】
アクティブ・プロジェクト … 75
アセンブラ … 408
アセンブラ・ソース・ファイル … 99
アセンブル・オプションの設定 … 44
アセンブル・リスト … 111
アセンブル・リストの出力 … 35
アブソリュート・アセンブル・リスト … 132
インクルード・ファイル … 142
エディタ パネル … 281
エラー・リスト … 116, 122, 129, 130, 132
エラー・リスト・ファイル … 103
オブジェクト・コンバート・オプションの設定 … 51
オブジェクト・コンバータ … 565
オプション ダイアログ … 310
［全般 - ビルド／デバッグ］カテゴリ … 312

【か行】
カテゴリ … 28
既存のファイルを追加 ダイアログ … 314
クリーン … 91
クロスリファレンス・リスト・ファイル … 107
クロスリファレンス・リスト … 114
コピー・ルーチン … 532
コンパイル・オプションの設定 … 38

【さ行】
再リンク機能 … 517
サブコマンド … 617
システム・インクルード・パス順設定 ダイアログ … 297
指定位置へ移動 ダイアログ … 308
出力 パネル … 282
出力ファイル名の変更 … 33
処理中表示 ダイアログ … 309
シンボル情報の出力 … 36
シンボル・リスト … 113
［全般 - ビルド／デバッグ］カテゴリ … 312

【た行】
タグ・ジャンプ … 283
テキスト編集 ダイアログ … 292

【な行】
名前を付けて保存 ダイアログ … 322

【は行】
パス編集 ダイアログ … 294
バッチ・ビルド … 82, 88
バッチ・ビルド ダイアログ … 306
パブリック・シンボル・リスト … 120, 126
パラメータ･ファイル … 409, 461, 532, 565, 580, 629
標準ライブラリ … 346
ビルド … 82, 85
ビルド・ツールのバージョン … 19
ビルドの実行 … 82
ビルド・モード … 76, 78
ビルド・モード設定 ダイアログ … 303
ビルド・モードの削除 … 80
ビルド・モードの追加 … 76
ビルド・モードの変更 … 78
ファイル追加 ダイアログ … 285

ファイルの依存関係 … 30
ファイルの追加 … 22
ファイルの表示順 … 29
ファイルの保存設定 ダイアログ … 299
ブートーフラッシュ再リンク機能 … 517
ブート領域用変数／関数情報ファイルを指定 ダイアログ … 318
ブート領域用ロード・モジュール・ファイルを指定 ダイアログ … 320
フォルダとファイル追加 ダイアログ … 288
フォルダの参照 ダイアログ … 316
プリプロセス・リスト・ファイル … 105
プログラムから開く ダイアログ … 324
プロジェクト・ツリー パネル … 154
プロパティ パネル … 169
［ROM 化プロセス・オプション］タブ … 230
［アセンブル・オプション］タブ … 211
［オブジェクト・コンバート・オプション］タブ … 236
［カテゴリ情報］タブ … 279
［共通オプション］タブ … 173
［個別アセンブル・オプション］タブ … 270
［個別コンパイル・オプション］タブ … 255
［コンパイル・オプション］タブ … 187
［ビルド設定］タブ … 251
［ファイル情報］タブ … 277
［変数／関数配置オプション］タブ … 248
［ライブラリ生成オプション］タブ … 244
［リンク・オプション］タブ … 219
変数／関数配置オプションの設定 … 54

【ま行】

マップ情報の出力 … 35
マップ・リスト … 118, 124
メイン・ウインドウ … 149
文字列入力 ダイアログ … 290

【ら行】

ライブラリアン … 603
ライブラリ生成オプションの設定 … 53
ラピッド・ビルド … 82, 86
ランタイム・ライブラリ … 346
リスト・コンバータ … 628
リビルド … 82, 86
リンカ … 460
リンク・オプションの設定 … 47
リンク順設定 ダイアログ … 301
リンク・リスト・ファイル … 117
リンク・マップ・ファイル … 533
ローカル・シンボル・リスト … 121, 127
ロード・モジュール・ファイル … 461, 532, 533, 565, 629

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
RL78,78K0R 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

© 2011 Renesas Electronics Corporation. All rights reserved.
Colophon 1.0

CubeSuite+ V1.00.00

RENESAS

ルネサス エレクトロニクス株式会社

R20UT0556JJ0100