



**IAR Embedded  
Workbench**

# IDE プロジェクト管理 およびビルドガイド

Arm Limited

**Arm® コア**

UIDEARM-14-J

 **IAR**  
SYSTEMS

## 著作権事項

© 1999–2019 IAR Systems AB.

本書のいかなる部分も、IAR システムズの書面による事前の同意なく複製することを禁止します。本書で解説するソフトウェアは使用許諾契約に基づき提供され、その条項に従う場合に限り使用または複製できるものとします。

## 免責事項

本書の内容は予告なく変更されることがあります。また、IAR システムズは、その内容についていかなる責任を負うものではありません。本書の内容については正確を期していますが、IAR システムズは誤りや記載漏れについて一切の責任を負わないものとします。

IAR システムズおよびその従業員、契約業者、本書の執筆者は、いかなる場合でも、特殊、直接、間接、または結果的な損害、損失、費用、負担、請求、要求、およびその性質を問わず利益損失、費用、支出の補填要求について、一切の責任を負わないものとします。

## 商標

IAR Systems、IAR Embedded Workbench、Embedded Trust、C-Trust、IAR Connect、C-SPY、C-RUN、C-STAT、IAR Visual State、IAR KickStart Kit、I-jet、I-jet Trace、I-scope、IAR Academy、IAR、および IAR Systems のロゴタイプは、IAR Systems AB が所有権を有する商標または登録商標です。

Microsoft および Windows は、Microsoft Corporation の登録商標です。

Arm、Cortex、Thumb、and TrustZone は、Arm Limited の登録商標です。EmbeddedICE は Arm Limited の商標です。uC/OS-II および uC/OS-III は Micrium, Inc の商標です。CMX-RTX は CMX Systems, Inc の商標です。ThreadX は Express Logic の商標です。RTXC は、Quadros Systems の商標です。Fusion は、Unicoi Systems の商標です。

Renesas Synergy は、Renesas Electronics Corporation の商標です。

Adobe および Acrobat Reader は、Adobe Systems Incorporated の登録商標です。

その他のすべての製品名は、その所有者の商標または登録商標です。

## 改版情報

第 14 版 : 2019 年 5 月

部品番号 : UIDEARM-14-J

本ガイドは、IAR Embedded Workbench® for Arm のバージョン 8.40.x に適用します。

内部参照 : BB5、Mym8.3、INIT。

# 目次（章）

目次（章） .....	3
表 .....	13
はじめに .....	15
<b>パート 1. プロジェクト管理およびビルド</b> .....	21
開発環境 .....	23
プロジェクト管理 .....	97
プロジェクトのビルド .....	123
編集 .....	139
<b>パート 2. リファレンス情報</b> .....	185
製品ファイル .....	187
メニューリファレンス .....	195
一般オプション .....	217
コンパイラオプション .....	227
アセンブラオプション .....	245
出力コンバータオプション .....	253
カスタムビルドオプション .....	255
ビルドアクションオプション .....	257
リンカオプション .....	259
ライブラリビルダオプション .....	275
用語集 .....	277
索引 .....	293



# 目次

目次（章） .....	3
表 .....	13
はじめに .....	15
<b>本ガイドの対象者</b> .....	15
必要な知識 .....	15
<b>本ガイドの使用方法</b> .....	15
<b>本ガイドの内容</b> .....	16
パート 1. プロジェクト管理およびビルド .....	16
パート 2. リファレンス情報 .....	16
<b>その他のドキュメント</b> .....	17
ユーザガイドおよびリファレンスガイド .....	17
オンラインヘルプシステムを参照 .....	18
Web サイト .....	18
<b>表記規則</b> .....	19
表記規則 .....	19
命名規約 .....	20
 <b>パート 1. プロジェクト管理およびビルド</b> .....	21
<b>開発環境</b> .....	23
<b>IAR Embedded Workbench IDE の概要</b> .....	23
IDE およびビルドツールチェーンの概要について .....	23
アプリケーションを解析してチェックするツール .....	24
拡張可能なモジュール化構造の環境 .....	25
画面のウィンドウのレイアウト .....	25
<b>IDE の使用およびカスタマイズ</b> .....	26
IDE の実行 .....	26
サンプルプロジェクトの使用 .....	27
画面上のウィンドウの編成 .....	33
ツールオプションの指定 .....	34

ツールバーへのボタンの追加 .....	35
ツールバーからボタンを削除 .....	36
ツールバーボタンを表示 / 非表示 .....	36
ファイル名拡張子 .....	37
外部のアナライザを使用するにあたって .....	37
[ツール] メニューからの外部ツールの呼出し .....	40
[ツール] メニューへのコマンドラインコマンドの追加 .....	41
外部エディタの連携 .....	41

## **IDE についてのリファレンス情報** ..... 43

[IAR Embedded Workbench IDE] ウィンドウ .....	45
[カスタマイズ] ダイアログボックス .....	50
ボタンの外観ダイアログボックス .....	53
[ツール出力] ウィンドウ .....	54
[共通フォント] オプション .....	55
[キーカスタマイズ] オプション .....	56
[言語] オプション .....	58
[エディタ] オプション .....	59
[自動インデントの設定] ダイアログボックス .....	63
[外部エディタ] のオプション .....	64
[セットアップファイル] オプション .....	66
[色とフォント] オプション .....	67
[メッセージ] オプション .....	68
[プロジェクト] オプション .....	69
[外部アナライザ] のオプション .....	72
[外部アナライザ] ダイアログボックス .....	73
ソースコード管理オプション（廃止） .....	75
[デバッガ] オプション .....	77
[スタック] オプション .....	79
[ターミナル I/O] オプション .....	81
[ツールの設定] ダイアログボックス .....	83
[ビューアの設定] ダイアログボックス .....	85
[ビューア拡張子の編集] ダイアログボックス .....	87
[ファイル名拡張子] ダイアログボックス .....	88
[ファイル名拡張子のオーバーライド] ダイアログボックス .....	89

[ファイル名拡張子の編集] ダイアログボックス .....	90
製品情報ダイアログボックス .....	90
引数変数 .....	91
[カスタムの引数変数の設定] ダイアログボックス .....	93
CMSIS Manager ダイアログボックス .....	96
<b>プロジェクト管理</b> .....	97
<b>プロジェクト管理の概要</b> .....	97
プロジェクト管理の概要について .....	97
プロジェクトの作成方法 .....	99
バージョン管理システムの IDE 操作 .....	102
<b>プロジェクト管理</b> .....	103
ワークスペースやそのプロジェクトの作成及び管理 .....	103
ワークスペースおよびそのプロジェクトの表示 .....	105
Subversion の操作 .....	106
CMSIS-Pack ソフトウェアパックのインストール .....	107
IAR Embedded Workbench でサポートする CMSIS-Pack の 使用 .....	108
<b>プロジェクト管理のリファレンス情報</b> .....	110
[ワークスペース] ウィンドウ .....	111
[新規プロジェクトの作成] ダイアログボックス .....	116
[プロジェクトの構成] ダイアログボックス .....	117
[新規ビルド構成] ダイアログボックス .....	118
[プロジェクトコネクションの追加] ダイアログボックス ....	119
Subversion のバージョン管理システムメニュー .....	120
Subversion の状態 .....	121
<b>プロジェクトのビルド</b> .....	123
<b>プロジェクトのビルドの概要</b> .....	123
プロジェクトのビルドの概要について .....	123
ツールチェーンの拡張 .....	123
<b>プロジェクトのビルド</b> .....	124
[オプション] ダイアログボックスを使用したプロジェ クトオプションの設定 .....	125
プロジェクトのビルド .....	128

ビルド中に検出されたエラーの修正 .....	129
ビルド前およびビルド後のアクションの使用 .....	129
バッチによる複数構成のビルド .....	130
コマンドラインからのビルド .....	131
外部ツールの追加 .....	132
<b>ビルドに関するリファレンス情報</b> .....	133
[オプション] ダイアログボックス .....	134
[ビルド] ウィンドウ .....	135
[バッチビルド] ダイアログボックス .....	137
[バッチビルドの編集] ダイアログボックス .....	138
<b>編集</b> .....	139
<b>IAR Embedded Workbench エディタの概要</b> .....	139
エディタの概要について .....	139
ソースブラウザ情報の概要について .....	140
エディタ環境のカスタマイズ .....	140
<b>ファイルの編集</b> .....	140
テキストの自動インデント .....	141
中括弧と括弧のマッチング .....	142
エディタウィンドウをペインに分割 .....	142
テキストのドラッグ .....	142
コードの折りたたみ .....	142
語句の入力補完 .....	143
コードの入力補完 .....	143
パラメータのヒント .....	144
コードテンプレートの使用と追加 .....	144
構文カラー表示 .....	146
ブックマークの追加 .....	147
エディタコマンドとショートカットキーの使用と カスタマイズ .....	147
ステータス情報の表示 .....	148
<b>プログラミングのサポート</b> .....	148
挿入ポイント履歴に移動します .....	148
関数への移動 .....	148



シンボルの定義または宣言の検索 .....	149
シンボルへの参照の検索 .....	149
選択した関数についての関数の呼出しの検索 .....	149
ソースファイルとヘッダファイル間の切替え .....	149
ブラウズ情報の表示 .....	150
テキスト検索 .....	150
オンラインヘルプのリファレンス情報へのアクセス .....	151
<b>エディタについてのリファレンス情報 .....</b>	<b>151</b>
[エディタ] ウィンドウ .....	152
[検索] ダイアログボックス .....	161
[ファイルで検索] ウィンドウ .....	163
[置換] ダイアログボックス .....	164
[ファイルから検索] ダイアログボックス .....	165
[ファイル内で置換] ダイアログボックス .....	167
[インクリメンタル検索] ダイアログボックス .....	170
[宣言] ウィンドウ .....	171
[曖昧な定義] ウィンドウ .....	172
[参照] ウィンドウ .....	173
[ソースブラウザ] ウィンドウ .....	174
[ソース参照ログ] ウィンドウ .....	177
[ファイルの曖昧さの解決] ダイアログボックス .....	179
[コールグラフ] ウィンドウ .....	179
[テンプレート] ダイアログボックス .....	180
エディタのショートカットキー操作のまとめ .....	181

## パート 2. リファレンス情報 .....

### 製品ファイル .....

<b>インストール先ディレクトリ構成 .....</b>	<b>187</b>
ルートディレクトリ .....	187
arm ディレクトリ .....	188
common ディレクトリ .....	189
install-info ディレクトリ .....	189

製品ディレクトリ構成 .....	190
さまざまな設定ファイル .....	190
グローバル設定のファイル .....	190
ローカル設定のファイル .....	191
ファイルタイプ .....	192
メニューリファレンス .....	195
メニュー .....	195
[ファイル] メニュー .....	195
[編集] メニュー .....	198
[表示] メニュー .....	202
[プロジェクト] メニュー .....	206
[メモリ消去] ダイアログボックス .....	211
[ツール] メニュー .....	212
[ウィンドウ] メニュー .....	215
[ヘルプ] メニュー .....	216
一般オプション .....	217
一般オプションの説明 .....	217
ターゲット .....	217
出力 .....	220
ライブラリ構成 .....	222
ライブラリオプション 1 .....	224
ライブラリオプション 2 .....	225
MISRA-C .....	226
コンパイラオプション .....	227
コンパイラオプションの説明 .....	227
複数ファイルのコンパイル .....	228
言語 1 .....	229
言語 2 .....	231
コード .....	232
最適化 .....	233
出力 .....	235
リスト .....	236

プリプロセッサ .....	237
診断 .....	239
MISRA-C .....	240
エンコード .....	241
追加オプション .....	242
[インクルードディレクトリの編集] ダイアログボックス ....	243
<b>アセンブラオプション .....</b>	<b>245</b>
<b>アセンブラオプションの概要 .....</b>	<b>245</b>
言語 .....	245
出力 .....	247
リスト .....	247
プリプロセッサ .....	249
診断 .....	250
追加オプション .....	252
<b>出力コンバータオプション .....</b>	<b>253</b>
<b>出力コンバータオプションの説明 .....</b>	<b>253</b>
出力 .....	253
<b>カスタムビルドオプション .....</b>	<b>255</b>
<b>カスタムビルドオプションの説明 .....</b>	<b>255</b>
カスタムツール構成 .....	255
<b>ビルドアクションオプション .....</b>	<b>257</b>
<b>ビルドアクションのオプションの説明 .....</b>	<b>257</b>
ビルドアクションの構成 .....	257
<b>リンカオプション .....</b>	<b>259</b>
<b>リンカオプションの説明 .....</b>	<b>259</b>
設定 .....	260
ライブラリ .....	261
入力 .....	262
最適化 .....	263
アドバンスト .....	264
出力 .....	265

リスト .....	266
#define .....	267
診断 .....	268
チェックサム .....	270
エンコード .....	272
追加オプション .....	273
[追加ライブラリの編集] ダイアログボックス .....	274
<b>ライブラリビルダオプション</b> .....	275
<b>ライブラリビルダオプションの説明</b> .....	275
出力 .....	276
<b>用語集</b> .....	277
<b>索引</b> .....	293

# 表

1: 本ガイドで使用されている表記規則 .....	19
2: このガイドで使用されている命名規約 .....	20
3: 引数変数 .....	91
4: <code>iarbuild.exe</code> コマンドラインオプション .....	131
5: エディタで挿入ポイントを移動するショートカットキー .....	181
6: エディタでテキストを選択するためのショートカットキー .....	182
7: エディタでスクロールするためのショートカットキー .....	182
8: その他のエディタのショートカットキー .....	182
9: Scintilla の追加ショートカットキー .....	183
10: <code>arm</code> ディレクトリ .....	188
11: <code>common</code> ディレクトリ .....	189
12: ファイルタイプ .....	192



# はじめに

- 本ガイドの対象者
- 本ガイドの使用方法
- 本ガイドの内容
- その他のドキュメント
- 表記規則

---

## 本ガイドの対象者

本ガイドは、IAR Embedded Workbench を使用してアプリケーションを開発し、IDE で利用可能なすべての機能およびツールを活用する場合に利用してください。

### 必要な知識

IAR Embedded Workbench のツールを使用するには、以下について十分な知識が必要です。

- 使用する Arm core コアの命令セットとアーキテクチャ（チップメーカーのドキュメントを参照）
- C/C++ プログラミング言語
- 組み込みシステム用アプリケーションの開発
- ホストコンピュータのオペレーティングシステム

IDE に組み込まれている他の開発ツールについて詳しくは、「17 ページの *その他のドキュメント*」を参照してください。

---

## 本ガイドの使用方法

本ガイドの各章は、特定のトピックを解説します。数多くの章が、*情報のタイプ*に応じて異なるセクションに分けられています：

- *概念*では、トピックについて説明するとともに、そのトピックに関連する機能の概要についてふれます。要件や制限の一覧も含まれます。このセクションを読んで、トピックエリアについて理解してください。

- タスクは、トピックエリアに関連する役立つタスクの一覧です。タスクの大半は、手順ごとの説明にも登場します。必要なタスクについての情報や、特定タスクの実行方法については、このセクションを参照してください。
- **参照情報**ではトピックエリアに関連した参照情報が提供されます。特定のGUI コンポーネントについて詳しくは、このセクションを参照してください。IDE の特定のコンポーネントに関するこの種の情報は、F1 を押すと簡単に入手できます。

IAR Embedded Workbench を初めてお使いになられる場合は、IAR インフォメーションセンターにあるチュートリアルが、IAR Embedded Workbench の使用方法の習得に役立ちます。

最後に、IAR システムズのユーザドキュメントでわからない用語がある場合は、**用語集**を参照してください。

---

## 本ガイドの内容

本ガイドの構成および各章の概要を以下に示します。

### パート 1. プロジェクト管理およびビルド

アプリケーションの編集、ビルドの手順について説明します。

- 「**開発環境**」では、IAR Embedded Workbench 開発環境の概要を説明します。また、必要に応じて環境をカスタマイズする機能についても説明します。
- 「**プロジェクト管理**」では、ワークスペースを作成し、複数のオブジェクト、ビルド構成、グループ、ソースファイル、オプションを指定して、バージョンの異なるアプリケーションを管理する方法を説明します。
- 「**プロジェクトのビルド**」では、アプリケーションのビルド手順について説明します。
- 「**編集**」では、IAR Embedded Workbench エディタの詳細、使用方法、関連機能について説明します。また、任意の外部エディタとの連携方法についても説明します。

### パート 2. リファレンス情報

- 「**製品ファイル**」では、ディレクトリ構成および各ディレクトリに含まれるファイルの種類について説明します。
- 「**メニューリファレンス**」には、メニューやメニューコマンドに関する詳しいリファレンス情報が含まれています。
- 「**一般オプション**」は、ターゲット、出力、ライブラリ、MISRA-C オプションについて説明します。



- 「コンパイラオプション」では、言語、最適化、コード、出力、リストファイル、プリプロセッサ、診断、MISRA-C のコンパイラオプションを指定します。
- 「アセンブラオプション」では、言語、出力、リスト、プリプロセッサ、診断用のアセンブラオプションについて説明します。
- 「出力コンバータオプション」では、ELF 形式からリンカ出力ファイルの変換に使用できるオプションについて説明します。
- 「カスタムビルドオプション」では、ツールのカスタム設定用オプションについて説明します。
- 「ビルドアクションオプション」では、ビルド前とビルド後のアクション用オプションについて説明します。
- 「リンカオプション」では、リンクを設定するオプションについて説明します。
- 「ライブラリビルダオプション」では、ライブラリをビルドするためのオプションについて説明します。

---

## その他のドキュメント

ユーザドキュメントは、ハイパーテキスト PDF 形式、およびコンテキスト依存のオンラインヘルプシステム（HTML フォーマット）があります。ドキュメンテーションには、インフォメーションセンタあるいは IAR Embedded Workbench IDE の [ヘルプ] メニューからアクセスできます。オンラインヘルプシステムは、F1 キーを押しても使用できます。

### ユーザガイドおよびリファレンスガイド

IAR システムズの各開発ツールについては、一連のガイドで説明しています。以下はツールとガイドの一覧です。

- IAR システムズの製品のインストールおよび登録の要件と詳細については、同梱されているインストールとライセンス・クイックリファレンスおよびライセンスガイドをご覧ください。
- プロジェクト管理および構築のために IDE の使用は、『*Arm 用 IDE プロジェクト管理およびビルドガイド*』を参照してください。
- IAR C-SPY® デバッガの使用および C-RUN ランタイムエラー解析は、『*ARM 用 C-SPY® デバッガガイド*』を参照してください。
- Arm 用 IAR C/C++ コンパイラのプログラミングおよび IAR ILINK リンカを使用したリンクについては、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

- Arm 用 IAR アセンブラを使用したプログラミングについては、『*ARM 用 IAR アセンブラリファレンスガイド*』を参照してください。
- C-STAT と必要なチェックを使用した静的解析の実行については、『*C-STAT® Static Analysis Guide*』を参照してください。
- MISRA-C ガイドラインを使用して、安全性を最重要視したアプリケーションを開発する方法については、『*IAR Embedded Workbench® MISRA-C:2004 Reference Guide*』または『*IAR Embedded Workbench® MISRA-C:1998 Reference Guide*』を参照してください。
- I-jet の使用法については、『*I-jet®, I-jet Trace, I-scope 用 IAR デバッグプローブガイド*』を参照してください。
- IAR J-Link および IAR J-Trace を使用については、『*J-Link/J-Trace ユーザーガイド*』を参照してください。
- IAR Embedded Workbench for Arm の旧バージョンで開発したアプリケーションコードやプロジェクトの移行については、『*IAR Embedded Workbench® 移行ガイド*』を参照してください。

注：製品のインストール内容によっては、他のドキュメントも提供される場合があります。

## オンラインヘルプシステムを参照

コンテキスト依存のオンラインヘルプの内容は以下のとおりです。

- IDE プロジェクト管理およびビルド
- IAR C-SPY® デバッガを使用したデバッグ
- IAR C/C++ コンパイラ
- IAR アセンブラ
- DLIB ライブラリ関数のキーワードリファレンス情報 関数のリファレンス情報を確認するには、エディタウィンドウで関数名を選択し、F1 キーを押します。
- C-STAT
- MISRA-C

## Web サイト

推奨 Web サイト：

- チップ製造元のウェブサイト。
- Arm limited Web サイト ([www.arm.com](http://www.arm.com)) には、Arm core に関する情報とニュースが記載されています。
- IAR システムズの Web サイト ([www.iar.com](http://www.iar.com)) では、アプリケーションノートおよびその他の製品情報を公開しています。

- C 標準化作業グループの Web サイト、[www.open-std.org/jtc1/sc22/wg14](http://www.open-std.org/jtc1/sc22/wg14)。
- C++ Standards Committee の Web サイト、[www.open-std.org/jtc1/sc22/wg21](http://www.open-std.org/jtc1/sc22/wg21)。
- C++ プログラミング言語の Web サイト、[isocpp.org](http://isocpp.org) このウェブサイトには C++ プログラミングの推奨の本の一覧が掲載されています。
- C および C++ 参照のウェブサイト、[en.cppreference.com](http://en.cppreference.com)

表記規則

IAR システムズのドキュメントでプログラミング言語 C と記述されている場合、特に記述がない限り C++ も含まれます。

製品のインストールでディレクトリを参照するとき、たとえば `arm¥doc`、場所のフルパスを前提とします。例えば、`c:¥Program Files¥IAR Systems¥Embedded Workbench N.n¥arm¥doc` のようになります。ここで、バージョン番号の最初の数字は、IAR Embedded Workbench 共有コンポーネントのバージョン番号の最初の数字を反映しています。

表記規則

IAR システムズのドキュメントでは、以下の表記規則を使用します。

スタイル	用途
computer	・ソースコードの例、ファイルパス。 ・コマンドライン上のテキスト。 ・2 進数、16 進数、8 進数。
parameter	パラメータとして使用される実際の値を表すプレースホルダ。たとえば、 <code>filename.h</code> の場合、 <code>filename</code> はファイルの名前を表します。
[option]	リンカまたは stack usage control ディレクティブのオプション部分。 【と】は実際のディレクティブの一部ではありませんが、[、]、{、または } はいずれもディレクティブ構文の一部です。
{option}	リンカまたは stack usage control ディレクティブの必須部分、{ と } は実際のディレクティブの一部ではありませんが、[、]、{、または } はディレクティブ構文の一部です。
[option]	コマンドラインオプションまたは pragma ディレクティブのオプション部分。
[a b c]	代替の選択肢を持つコマンドラインオプションまたは pragma ディレクティブのオプション部分。
{a b c}	コマンドラインオプションまたは pragma ディレクティブの必須部分。
太字	画面で表示されるメニュー、メニューコマンド、ボタン、ダイアログボックス の名前を示します。

表 1: 本ガイドで使用されている表記規則





スタイル	用途
斜体	<ul style="list-style-type: none"><li>• 本ガイドや他のガイドへのクロスリファレンスを示します。</li><li>• 強調。</li></ul>
...	3 点リーダーは、その前の項目を任意の回数繰り返せることを示します。
	IAR Embedded Workbench® IDE 固有の内容を示します。
	コマンドライン インタフェース固有の内容を示します。
	開発やプログラミングについてのヒントを示します。
	ワーニングを示します。

表 1: 本ガイドで使用されている表記規則 (続き)

命名規約

以下の命名規約は、このガイドに記述されている IAR システムズの製品およびツールで使用されています。

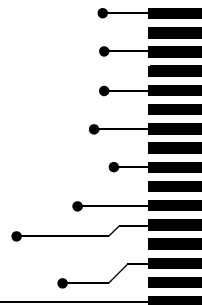
ブランド名	一般名称
IAR Embedded Workbench® for Arm	IAR Embedded Workbench®
IAR Embedded Workbench® IDE for Arm	IDE
IAR C-SPY® デバッガ for Arm	C-SPY、デバッガ
IAR C-SPY® シミュレータ for Arm	シミュレータ
Arm 用 IAR C/C++ コンパイラ ™	コンパイラ
Arm 用 IAR アセンブラ ™	アセンブラ
IAR ILINK リンカ ™	ILINK、リンカ
IAR DLIB ランタイム環境 ™	DLIB ランタイム環境

表 2: このガイドで使用されている命名規約

# パート I. プロジェクト管理およびビルド

このパートは以下の章で構成されます。

- 開発環境
- プロジェクト管理
- プロジェクトのビルド
- 編集





# 開発環境

- IAR Embedded Workbench IDE の概要
- IDE の使用およびカスタマイズ
- IDE についてのリファレンス情報

---

## IAR Embedded Workbench IDE の概要

以下のトピックを解説します：

- IDE およびビルドツールチェーンの概要について
- アプリケーションを解析してチェックするツール
- 拡張可能なモジュール化構造の環境
- 画面のウィンドウのレイアウト

### IDE およびビルドツールチェーンの概要について

IDE は、アプリケーションのビルドに必要なすべてのツール（ビルドツールチェーン）が統合されている環境です。C/C++ コンパイラ、C/C++ ライブラリ、アセンブラ、リンカ、ライブラリツール、エディタ、Make ユーティリティ付きプロジェクトマネージャ、および IAR C-SPY® デバッガが含まれます。ソースコードのビルドに使用されるツールは、ビルドツールと呼ばれます。

製品パッケージに付属のツールチェーンは、特定のマイクロコントローラをサポートしています。IDE では、さまざまなマイクロコントローラに対する複数のツールチェーンを同時に格納できます。つまり、いくつかのマイクロコントローラ用に IAR Embedded Workbench をインストールしている場合、どのマイクロコントローラ向けに開発するかを選択できます。

**注：**既に構築されているプロジェクト環境で外部ツールとして利用する場合は、コンパイラ、アセンブラ、リンカ、ライブラリツールをコマンドライン環境で実行することもできます。

## アプリケーションを解析してチェックするツール

IAR Embedded Workbench には、以下のようなアプリケーションのエラーを解析して検出するための各種サポートが付属しています。

- コンパイラおよびリンカのエラー、ワーニング、リマーク  
診断メッセージはすべて、説明を要しない完結型のメッセージとして出力されます。エラーは構文や動作のエラーを、ワーニングは潜在的な問題を、リマーク（デフォルトではオフ）は標準規格からの逸脱をそれぞれ示します。メッセージをダブルクリックすると、対応するソースコードの構造がエディタウィンドウで強調表示されます。詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。
- リンク時のスタック使用量解析  
適切な状況下では、cstartup や割込み関数、RTOS タスクなど、各呼出しツリーについて最大スタック使用量を正確に算出できます。詳細は『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。
- 静的解析用の C-STAT  
C-STAT は、特定のルールセットからの逸脱を検出する静的解析ツールです。各ルールで、安全でないソース構造の検出対象を指定をします。ルールは MISRA (MISRA-C:2004、MISRA-C++:2008、MISRA-C:2012)、CWE、CERT など様々な団体が提唱するものです。C-STAT およびその規則の使用法について詳しくは、『*C-STAT® Static Analysis Guide*』を参照してください。
- MISRA-C:1998 および 2004  
C-STAT での MISRA チェックに加えて、IDE には MISRA-C:1998 および 2004 のコンパイラチェックが用意されています。詳細は『*IAR Embedded Workbench® MISRA-C:2004 Reference Guide*』または『*IAR Embedded Workbench® MISRA-C:1998 Reference Guide*』を参照してください。
- C-SPY のデバッグ機能（プロファイリング、コードカバレッジ、トレース、Power デバッグなど）詳細については、『*ARM 用 C-SPY® デバッグガイド*』を参照してください。
- ランタイムエラーチェック用の C-RUN  
ランタイムエラー解析は、アプリケーションの実行中に不正なコードの構造を検出する方法です。これは、アプリケーション内にコードを実装するか、C/C++ ライブラリの機能をランタイムエラーチェックをサポートする専用のライブラリに置き換えることにより実行します。C-RUN は、算術チェック、境界チェック、チェック済みヒープを使用したヒープチェックという 3 種類のランタイムエラーチェックをサポートしています。詳細については、『*ARM 用 C-SPY® デバッグガイド*』を参照してください。



## 拡張可能なモジュール化構造の環境

IDE にはプロジェクトに必要なあらゆる機能が備わっていますが、他のツールを統合することも可能です。たとえば、以下のことが可能です。

- カスタムビルドというメカニズムを使用して、他のツールをツールチェーンに追加します (123 ページのツールチェーンの拡張を参照)。
- IAR visualSTATE をツールチェーンに追加して、IDE でステートマシンダイアグラムをプロジェクトに直接追加することも可能です。
- Subversion のバージョン管理システムを使用して、異なるバージョンのソースコードを追跡します。また、IDE では Subversion の作業用コピーのファイルにアタッチできます。
- lint ツールなど外部のアナライザを追加して、プロジェクト全体、複数のファイル、またはプロジェクトの 1 つのファイルで使用することができます。通常は、コンパイルのときと同じ設定およびソースコードファイルソースのセットを使用して、コードについて静的なコード解析を実行します。37 ページの外部のアナライザを使用するにあたってを参照してください。
- IDE 内部から簡単にアクセスできるように、外部のツールを [ツール] メニューに追加します。そのため、メニューコマンドとしてメニューに表示されるように事前に設定したツールに応じて、表示されるメニューが異なる場合があります。
- カスタムの引数変数を設定します。これは通常、サードパーティ製品をインストールしてそのインクルードディレクトリを指定する場合などに役立ちます。カスタムの引数変数を使用して、プロジェクトに含めるファイルへの参照を簡略化することも可能です。

## 画面のウィンドウのレイアウト

IDE では、開くウィンドウにそれぞれデフォルトの位置があり、それは現在開かれている他のウィンドウによって変わります。ウィンドウの位置やレイアウトの調整を任意に設定できます。各ウィンドウは、ドッキングまたはフローティングのどちらかの状態で使用できます。

各ウィンドウは特定の位置にドッキングして、タブグループとして編成できます。ドッキングされたウィンドウの 1 つをサイズ変更すると、ドッキングされた他のウィンドウのサイズがそれに従って変更されます。また、ウィンドウをフローティング化することができます。フローティングウィンドウは、常に他のウィンドウよりも前に表示されます。フローティング化されたウィンドウの位置とサイズは、現在開いている他のウィンドウには影響しません。フローティングウィンドウは画面上の任意の位置に移動することができ、IAR Embedded Workbench IDE のメインウィンドウの外側など、画面上の任意の場所に移動できます。

一度保存したワークスペースを開くと、保存したときと同じウィンドウが同じサイズで同じ位置に開きます。

C-SPY 環境で実行されるプロジェクトのレイアウトはすべて個別に保存されます。ワークスペースに関する情報の他に、開いているすべてのデバッガ固有のウィンドウに関する情報も保存されます。

**注:** エディタウィンドウは常にドッキングされています。エディタウィンドウを開くと、その位置は現在開いている他のウィンドウに応じて自動的に決まります。エディタウィンドウの操作方法の詳細については、139 ページの *IAR Embedded Workbench* エディタの概要を参照してください。

## IDE の使用およびカスタマイズ

以下のタスクについて解説します。

- IDE の実行
- サンプルプロジェクトの使用
- 画面上のウィンドウの編成
- ツールオプションの指定
- ツールバーへのボタンの追加
- ツールバーからボタンを削除
- ツールバーボタンを表示 / 非表示
- ファイル名拡張子
- 外部のアナライザを使用するにあたって
- [ツール] メニューからの外部ツールの呼出し
- [ツール] メニューへのコマンドラインコマンドの追加
- 外部エディタの連携

123 ページの ツールチェーンの拡張を参照してください。

C-SPY 関連のカスタマイズの詳細については、『*ARM 用 C-SPY® デバッガガイド*』を参照してください。

### IDE の実行

Windows タスクバーの [スタート] ボタンをクリックして、[すべてのプログラム] > [IAR Systems] > [IAR Embedded Workbench for Arm] > [IAR Embedded Workbench] を選択します。

コマンドラインまたは Windows エクスプローラからプログラムを起動するには、IAR システムズのインストール先の common¥bin ディレクトリにある IarIdePm.exe ファイルを実行します。

## ワークスペースファイル名のダブルクリック

ワークスペースファイル名には、拡張子 `eww` が付いています。ワークスペースのファイル名をダブルクリックすると、IDE が起動します。

複数バージョンの IAR Embedded Workbench がインストールされている場合、どのバージョンでプロジェクトファイルが作成されたに関係なく、ワークスペースファイルは、そのファイルタイプを使用する最後に使用されたバージョンの IAR Embedded Workbench によって開かれます。

## サンプルプロジェクトの使用

アプリケーションのサンプルは IAR Embedded Workbench に同梱されています。これらのサンプルを使用して、IAR システムズの開発ツールを使用する準備を行えます。また、これらのサンプルを基にして、アプリケーションプロジェクトを開始することもできます。

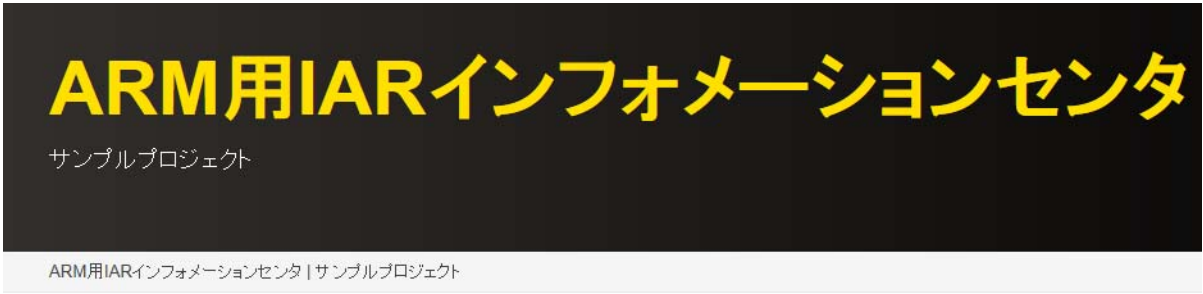
IAR Systems により提供されるサンプルプロジェクトに加えて、IAR Embedded Workbench から多くの CMSIS-Pack のサンプルプロジェクトにもアクセスできます。

サンプルは現状のまま提供されます。既製のワークスペースファイルが、ソースコードファイルおよび関連する他のすべてのファイルとともに提供されます。

### サンプルプロジェクトをダウンロードするには：

- 1 デフォルトでは、ダウンロードしたサンプルが、容量に制限がある可能性のあるお使いのシステム ディスクにインストールされます。場所を変更する場合は、グローバルカスタム引数変数 `$EXAMPLES_DIR$` を設定し、サンプルをダウンロードしたいパスにその値を設定します。93 ページの *[カスタムの引数変数の設定]* ダイアログボックスを参照してください。
- 2 **[ヘルプ] > [インフォメーションセンタ]** を選択して、**[サンプルプロジェクト]** をクリックします。

- 3 [ダウンロード可能なサンプルプロジェクト] で、使用デバイスと同じチップの製造元のダウンロードボタンをクリックします。



サンプルプロジェクト

特定のデバイスおよび評価ボード用のハードウェア周辺機器のデモを行うサンプルアプリケーション。

Example projects that can be downloaded

All examples	
Aiji	
AmbiqMicro	

- 4 表示されるダイアログボックスで、サンプルの取得元を選択します。以下から選択します。
- IAR システムズからダウンロード
  - インストール用 DVD からコピー。この場合、[参照] ボタンを使用して必要なサンプルの自己解凍アーカイブを探します。アーカイブは DVD の %examples-archive ディレクトリにあります

選択したデバイスのベンダーのサンプルが、お使いのコンピュータに展開されます。グローバルカスタム引数変数 \$EXAMPLES\_DIR\$ を定義して場所を変更していない限り、サンプルはプログラムディレクトリ、またはお使いの Windows オペレーティングシステムにより関連のディレクトリに展開されます。

- 5 ダウンロードしたサンプルは、インフォメーションセンタの **Installed example projects** のリストに表示されます。

サンプルプロジェクトを実行するには：

- 1 [ヘルプ] > [インフォメーションセンタ] を選択して、[サンプルプロジェクト] をクリックします。
- 2 [installed example projects] で、使用している評価ボードまたはスターターキットと一致するサンプルを参照するか、IAR Systems ウェブサイトからサンプルをダウンロードする場合は、[サンプルプロジェクトをダウンロードするには] の手順に従ってください。



[プロジェクトを開く] ボタンをクリックします。

- 3 表示されたダイアログボックスで、プロジェクトの配置先フォルダを選択します。
- 4 [ワークスペース] ウィンドウに、使用可能なサンプルプロジェクトが表示されます。プロジェクトを1つ選択します。アクティブなプロジェクト（太字で強調表示）でない場合は、そのプロジェクトを右クリックして、コンテキストメニューから [アクティブに設定] を選択します。

- 5 プロジェクト設定を表示するには、[プロジェクト] > [オプション] を選択します。[一般オプション] > [ターゲット] > [プロセッサ選択] および [デバッグ] > [設定] > [ドライバ] の設定を確認します。プロジェクトのその他の設定については、選択したターゲットシステムに合わせて設定されます。

C-SPY オプションの詳細と、ターゲットボード操作に C-SPY を設定する方法については、『ARM 用 C-SPY® デバッグガイド』を参照してください。

[OK] をクリックして、プロジェクトの [オプション] ダイアログボックスを閉じます。



- 6 アプリケーションをコンパイルしてリンクするには、[プロジェクト] > [メイク] を選択するか、[メイク] ボタンをクリックします。

- 7 C-SPY を起動するには、[プロジェクト] > [デバッグ] を選択するか、[ダウンロードしてデバッグ] ボタンをクリックします。C-SPY がターゲットシステムとの接続を確立できない場合は、『ARM 用 C-SPY® デバッグガイド』を参照してください。



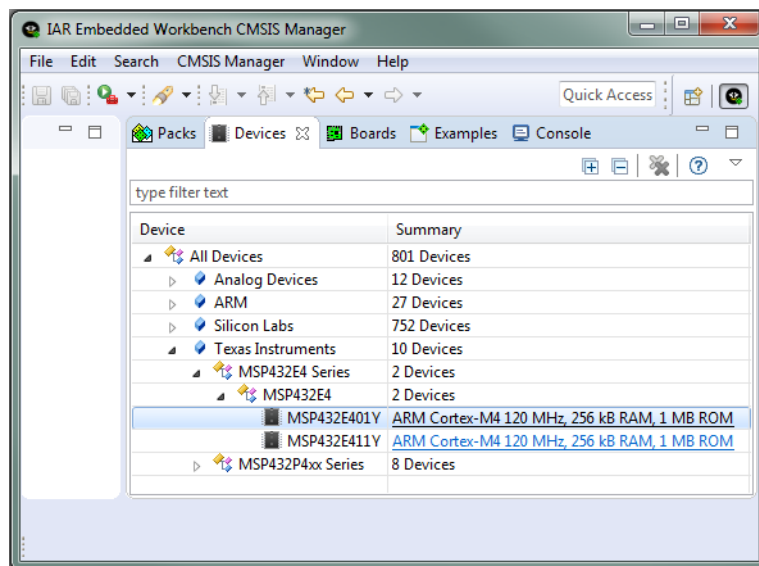
- 8 アプリケーションを起動するには、[デバッグ] > [実行] を選択するか [実行] ボタンをクリックします。

実行を停止するには、[ブレーク] ボタンをクリックします。

**CMSIS-Pack サンプルプロジェクトを使用するには：**



- 1 IAR Embedded Workbench ワークスペースで、[プロジェクト] > [CMSIS-Manager] を選択するか、**CMSIS-Manager** ツールバーボタンをクリックします。
- 2 [ワークスペースを名前を付けて保存] ダイアログボックスを使用してワークスペースを保存します。
- 3 表示される **CMSIS Manager** ダイアログボックスで、[デバイス] ビューに移動して、使用しているデバイスを選択します。



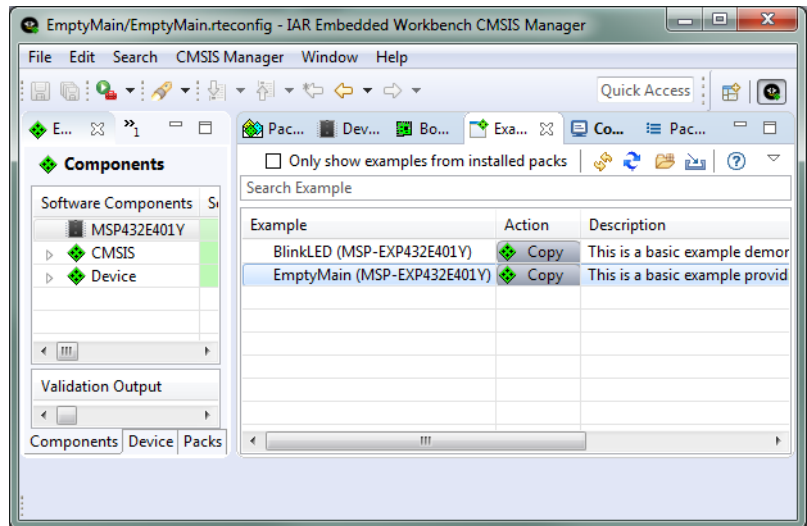
**CMSIS Manager** ダイアログボックスの詳細については、96 ページの *CMSIS Manager* ダイアログボックスを参照してください。

- 4 *CMSIS-Pack* ソフトウェアパックのインストールで示されたように *CMSIS-Pack* ソフトウェアパックをまだインストールしていない場合は、**[パック]** タブをクリックし必要なパックを選択し **[インストール]** ボタンをクリックします。

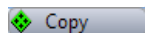
インストールプロセスを示すステータスメッセージをプリントする **[コンソール]** ビューの切り替えをフォーカスします。

- 5 **[デバイス]** タブをクリックしデバイスがまだ選択されていることを確認します。

- 6 **Examples** タブをクリックします。**Example** ビューには、選択したデバイスの利用可能なサンプルプロジェクトがリストされます。

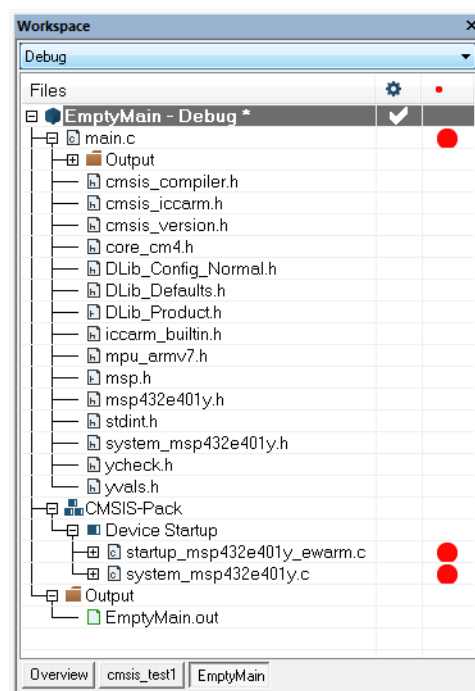






Copy

- 7 サンプルプロジェクトを選択して関連のアクションボタン [コピー] をクリックして、IAR Embedded Workbench ワークスペースに CMSIS サンプルプロジェクトをコピーします。



- 8 [コンポーネント] ビューで、他のソフトウェアパックに解決していない依存関係があるか確認します。依存関係を解決については、107 ページの *CMSIS-Pack* ソフトウェアパックのインストールの特にステップ 6 を参照してください。
- 9 [プロジェクト] > [オプション] を選択し、プロジェクトオプションの設定を検証します。これで CMSIS サンプルプロジェクトを IAR Embedded Workbench で開始できる準備ができました。

## 画面上のウィンドウの編成

以下の方法を使用して、画面上のウィンドウを編集します。

- タブグループからタブされたウィンドウを切断して、別ウィンドウとして配置するには、タブグループからタブをドラッグして取り出します。

- ウィンドウまたはタブをフローティング化するには、ウィンドウのタイトルバーをダブルクリックします。
- ウィンドウをドラッグして動かす場合、**Ctrl** を押すことでドッキングを防ぐことができます。
- ウィンドウを開いている別のウィンドウと同じタブグループに配置するには、ドラッグして、他のウィンドウにドロップします。オーガナイザーコントロールのいずれかの矢印ボタンにそれをドロップして、どのようにドッキングするかを制御します。

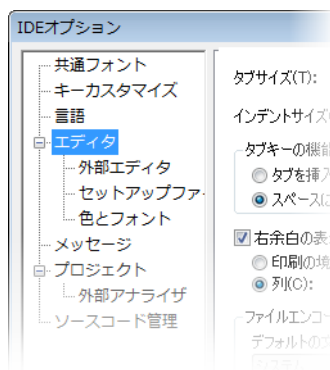


25 ページの *画面* のウィンドウのレイアウトを参照してください。

## ツールオプションの指定

IDE をカスタマイズするコマンドは **[ツール]** メニューにあります。

- 1 **[IDE オプション]** ダイアログボックスを表示するには、**[ツール]** > **[オプション]** を選び、幅広いオプションにアクセスすることができます。



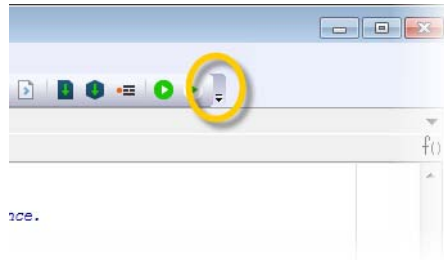
- 2 ダイアログボックスの右のオプションにアクセスするには、左のカテゴリを選択します。

IDE をカスタマイズするさまざまなオプションの詳細については、212 ページの **[ツール]** メニューを参照してください。

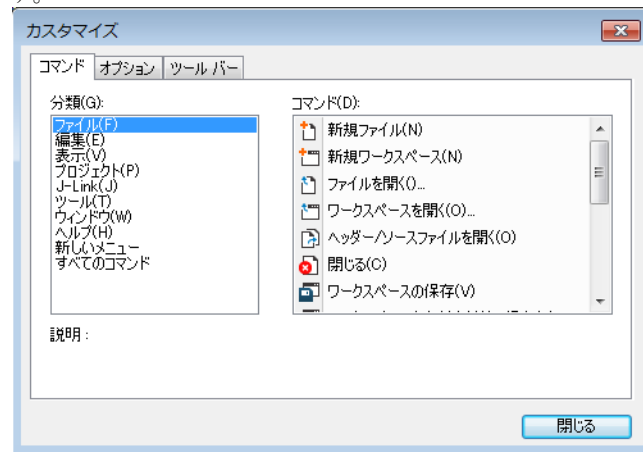
## ツールバーへのボタンの追加

ツールバーのボタンを使用すると、IDE メニューのコマンドにショートカットできます。

- 1 メイン IDE ウィンドウのツールバーに新しいボタンを追加するには、[ツールバーオプション] ボタンをクリックし、[ボタンの表示 / 非表示] > [カスタマイズ] を選択します。



- 2 [カスタマイズ] ダイアログボックスで [コマンド] ページで開きます。  
[分類] リストで、ツールバーに追加したいコマンドのメニューを選択します。



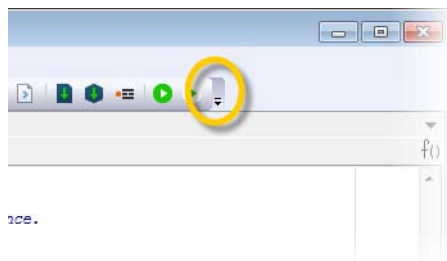
- 3 [コマンド] リストからコマンドを、ボタンとしてコマンドを挿入したいツールバーにドラッグします。

既存のボタンを新しい場所にドラッグして、再構成できます。

**注 :** ボタンを追加する代わりに、一時的に非表示になっているボタンを表示するには、36 ページの ツールバーボタンを表示 / 非表示を参照してください。

### ツールバーからボタンを削除

- 1 メイン IDE ウィンドウのいずれかのツールバーからボタンを削除するには、[ツールバーオプション] ボタンをクリックし、[ボタンの表示 / 非表示] > [カスタマイズ] を選択します。[カスタマイズ] ダイアログボックスが開きますが無視します。



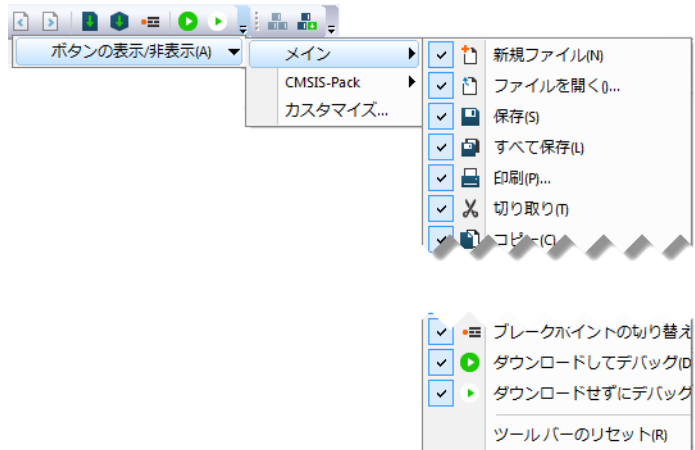
- 2 削除したいツールバーのボタンを右クリックし、コンテキストメニューから [削除] を選びます。

**注 :** ボタンを削除する代わりに、一時的に非表示にしたい場合は 36 ページの ツールバーボタンを表示 / 非表示を参照してください。

### ツールバーボタンを表示 / 非表示

IDE ツールバーからボタンを削除する代わりに、その表示をオン / オフに切り替えることができます。

- 1 メイン IDE ウィンドウのいずれかのツールバーから、一時的にボタンを表示するには、[ツールバーオプション] ボタンをクリックし、[ボタンの表示 / 非表示] > [ツールバー] を選択します。



- 2 表示 / 非表示にしたいコマンドのボタンを選択または選択解除します。

**注：** ツールバーからボタンをすべて削除するには、36 ページのツールバーからボタンを削除を参照してください。

## ファイル名拡張子

IDE で、認識するファイル名拡張子の数を増やすことができます。デフォルトでは、ビルドツールチェーンの各ツールは、標準的なファイル名拡張子に対応します。また、それ以外のファイル名拡張子を持つソースファイルを使用する場合は、使用可能なファイル名拡張子を変更できます。

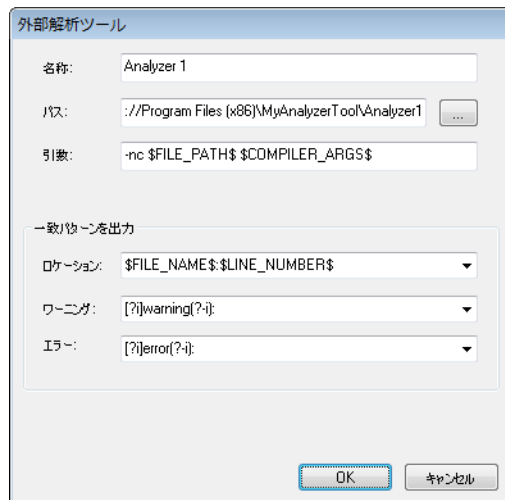
必要なコマンドにアクセスするには、[ツール] > [ファイル名拡張子] を選択します。

88 ページの [ファイル名拡張子] ダイアログボックスを参照してください。

デフォルトのファイル名拡張子をコマンドラインからオーバーライドするには、ファイル名の指定時に拡張子を明示的に指定します。

## 外部のアナライザを使用するにあたって

- 1 外部のアナライザを [プロジェクト] メニューに追加するには、[ツール] > [オプション] を選択して [IDE オプション] ダイアログボックスを開き、[プロジェクト] > [外部アナライザ] ページを選択します。
- 2 呼出しを設定するには、[追加] をクリックして [外部アナライザ] ダイアログボックスを開きます。



呼び出したいアナライザに必要な詳細を指定します。

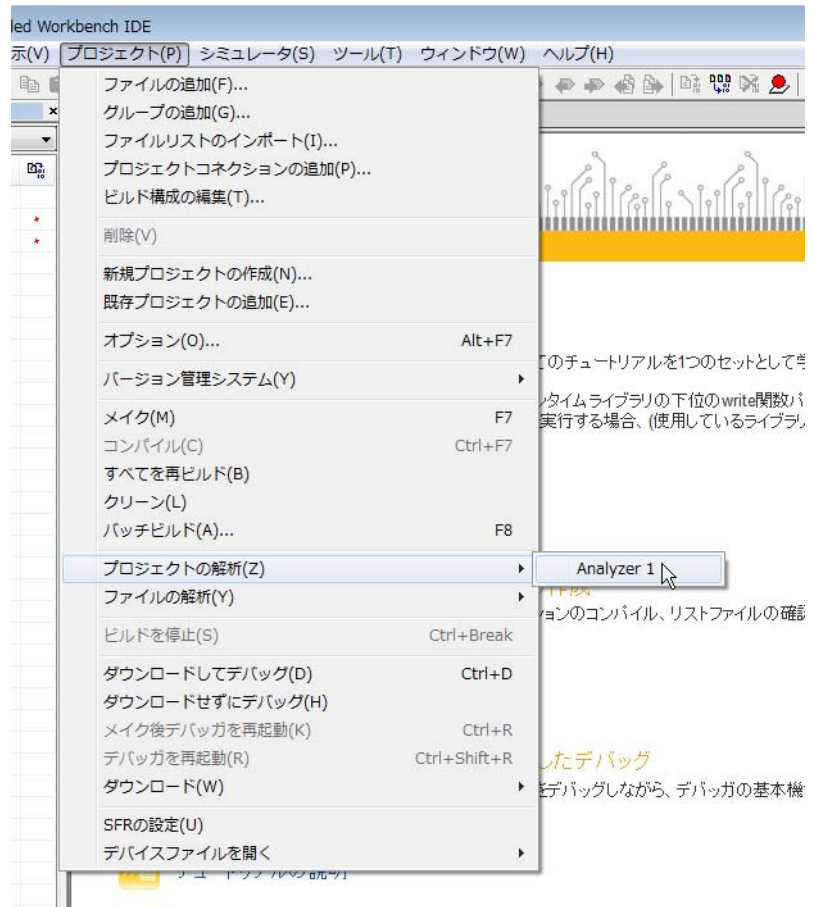
ソースファイルの位置への参照を検索するためのワーニングおよびエラーメッセージを識別する 3 つの正規表現を指定（またはリストから選択）するには、**[一致パターンを出力]** を使用します。

終了したら、**[OK]** をクリックします。

このダイアログボックスの詳細な情報については、73 ページの *[外部アナライザ] ダイアログボックス* を参照してください。

- 3 **[IDE オプション]** ダイアログボックスで、**[OK]** をクリックします。

- 4 [プロジェクト] > [プロジェクトの解析] を選択して、実行するアナライザを選択します。または、[ファイルの解析] を選択して、個々のファイルに対してアナライザを実行します。

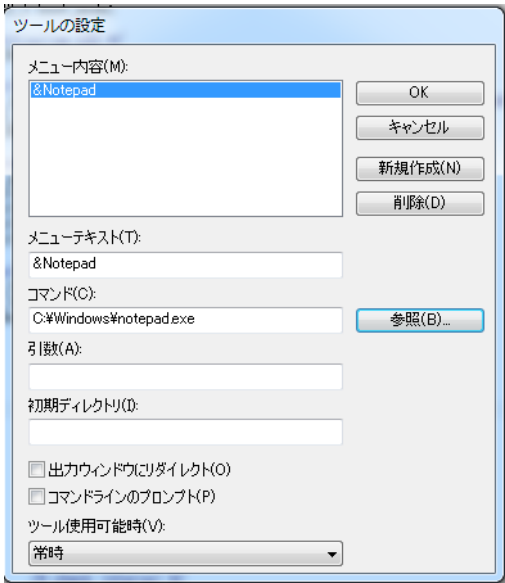


指定したそれぞれの正規表現が、外部アナライザの出力の各行に適用されます。アナライザの出力は【ビルドログ】ウィンドウに一覧表示されます。【外部アナライザ】ダイアログボックスで指定した【位置】正規表現に一致する行をダブルクリックすると、エディタウィンドウで対応する位置にジャンプすることができます。

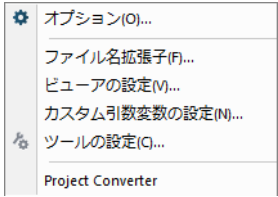
注：解析を途中で停止するには、【ビルドを停止】ボタンをクリックします。

【ツール】メニューからの外部ツールの呼出し

- 1 メニューにメモ帳などの外部ツールを追加するには、【ツール】>【ツールの設定】を選択して、【ツールの設定】ダイアログボックスを開きます。



- 2 スクリーンショットに従ってテキストフィールドに情報を入力します。このダイアログボックスの詳細な情報については、83 ページの 【ツールの設定】ダイアログボックスを参照してください。
- 3 適切な情報を入力して [OK] をクリックすると、指定したメニューコマンドが【ツール】メニューに表示されます。



注：IDE のツールチェーンの拡張に【ツールの設定】ダイアログボックスを使用することはできません。標準ビルドツールチェーンに外部ツールを追加する場合は、「123 ページの ツールチェーンの拡張」を参照してください。



## 【ツール】メニューへのコマンドラインコマンドの追加

コマンドラインコマンドとバッチファイル呼出しは、コマンドシェルから実行する必要があります。コマンドラインコマンドを【ツール】メニューに追加すると、そのメニューからコマンドラインコマンドを実行できます。

バックアップなどのコマンドを【ツール】メニューを追加して、ネットワークドライブに project ディレクトリ全体のコピーを作成するには、次の手順に従います。

- 1 【ツール】 > 【ツールの設定】を選択して、【ツールの設定】ダイアログボックスを開きます。
- 2 【コマンド】テキストボックスで、**cmd.exe** コマンドシェルを入力または検索します。
- 3 【引数】テキストボックスで、コマンドラインコマンドかバッチファイル名を指定します。例：

```
/C copy c:\project¥*. * F:
```

別の方法として、引数変数を使用して再配置可能なパスを許可することもできます。

```
/C copy $PROJ_DIR¥*. * F:
```

引数のテキストは、以下に示すように指定する必要があります。

```
/C name
```

ここで、*name* は、実行するコマンドかバッチファイルの名前です。

/c オプションは、実行後にシェルを終了するように指定し、ツールの終了をIDEが検出できるようにします。

## 外部エディタの連携

【外部エディタ】オプション（【ツール】 > 【オプション】 > 【エディタ】 > 【外部エディタ】を選択して表示）では、任意の外部エディタを指定できます。

注：C-SPY を使用したデバッグ中には、現在のデバッグ状態の表示に外部エディタは使用されません。内蔵のエディタが使用されます。

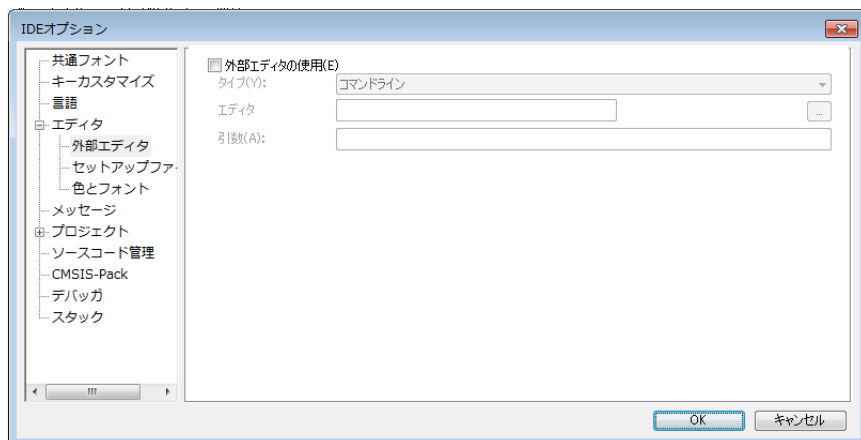
任意の外部エディタを指定するには、次の手順に従います。

- 1 【外部エディタの使用】オプションを選択します。
- 2 外部エディタを呼び出すには、【タイプ】ドロップダウンメニューで以下の2つの方法のどちらかを選択します。

- **[コマンドライン]** は、外部エディタを呼び出して、コマンドラインパラメータを渡します。
  - **[DDE]** は、DDE (Windows Dynamic Data Exchange) を使用して、外部エディタを呼び出します。
- 3** コマンドラインを使用する場合は、エディタに渡すコマンド、すなわちエディタの名前とそのパスを指定します。以下に例を示します。

C:\Windows\NOTEPAD.EXE

引数を外部エディタに送信するには、**[引数]** フィールドに引数を入力します。たとえば、「\$FILE\_PATH\$」と入力するとエディタが起動して、アクティブなファイルが開きます（エディタ、プロジェクト、[メッセージ] ウィンドウ）。



**注：**[ターミナル I/O] のオプションは、C-SPY デバッガの実行中のみ使用できます。

- 4** DDE を使用する場合、**[サービス]** フィールドでエディタの DDE サービス名を指定します。**[コマンド]** フィールドで、エディタに送信するコマンドシーケンスを表す文字列を指定します。

サービス名とコマンド文字列は、使用する外部エディタに応じて指定します。外部エディタのドキュメントを参照して、適切に設定してください。

コマンド文字列は、以下の形式で入力する必要があります。

```
DDE-Topic CommandString1
DDE-Topic CommandString2
```

以下に例を示します。この例は、Codewright® に適用されます。

☒ 外部エディタの使用(E)

タイプ(Y): DDE

エディタ(D): C:\CW32\cw32.exe

サービス(S): Codewright

コマンド(C):  
System BufEditFile \$FILE\_PATH\$  
\$FILE\_PATH\$ MovToLine \$CUR\_LINE\$

この例で指定したコマンド文字列で、外部エディタが開いて、指定したファイルがアクティブになります。カーソルは、たとえばファイル内の文字列を検索している場合や [メッセージ] ウィンドウでエラーメッセージをダブルクリックした場合のように、ファイルを開いたコンテキストの定義に従って、現在の行に置かれます。

## 5 OK をクリックします。

[ワークスペース] ウィンドウでファイル名をダブルクリックすると、そのファイルは外部エディタで開かれます。

引数に変数を使用できます。引数変数の詳細については、91 ページの *引数変数* を参照してください。

## IDE についてのリファレンス情報

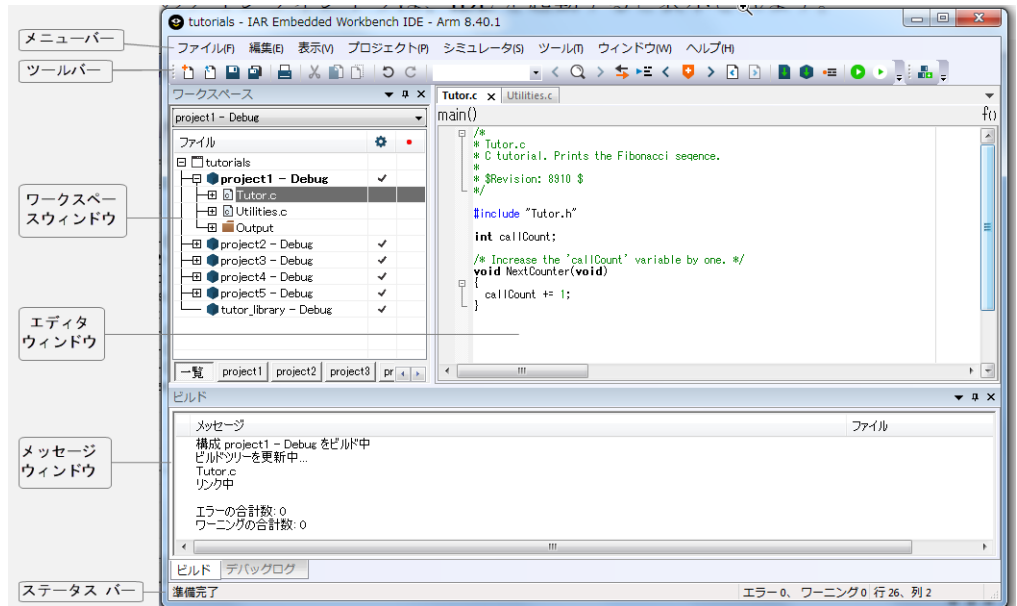
リファレンス情報：

- 45 ページの *[IAR Embedded Workbench IDE]* ウィンドウ
- 50 ページの *[カスタマイズ]* ダイアログボックス
- 53 ページの *ボタンの外観* ダイアログボックス
- 54 ページの *[ツール出力]* ウィンドウ
- 55 ページの *[共通フォント]* オプション
- 56 ページの *[キーカスタマイズ]* オプション
- 58 ページの *[言語]* オプション
- 59 ページの *[エディタ]* オプション

- 63 ページの [自動インデントの設定] ダイアログボックス
- 64 ページの [外部エディタ] のオプション
- 66 ページの [セットアップファイル] オプション
- 67 ページの [色とフォント] オプション
- 68 ページの [メッセージ] オプション
- 69 ページの [プロジェクト] オプション
- 72 ページの [外部アナライザ] のオプション
- 73 ページの [外部アナライザ] ダイアログボックス
- 75 ページの ソースコード管理オプション (廃止)
- 77 ページの [デバッガ] オプション
- 79 ページの [スタック] オプション
- 81 ページの [ターミナル I/O] オプション
- 83 ページの [ツールの設定] ダイアログボックス
- 85 ページの [ビューアの設定] ダイアログボックス
- 87 ページの [ビューア拡張子の編集] ダイアログボックス
- 88 ページの [ファイル名拡張子] ダイアログボックス
- 89 ページの [ファイル名拡張子のオーバーライド] ダイアログボックス
- 90 ページの [ファイル名拡張子の編集] ダイアログボックス
- 90 ページの 製品情報ダイアログボックス
- 91 ページの 引数変数
- 93 ページの [カスタムの引数変数の設定] ダイアログボックス
- 96 ページの CMSIS Manager ダイアログボックス

## [IAR Embedded Workbench IDE] ウィンドウ

IDE のメインウィンドウは、IDE を起動すると表示されます。



以下の図は、ウィンドウとそのデフォルトのレイアウトを示します。

### メニューバー

メニューバーには以下が含まれます。

#### ファイル

ソースファイルおよびプロジェクトファイルのオープン、保存、出力、IDE の終了を実行するためのコマンド。

#### 編集

エディタウィンドウでの編集 / 検索用コマンドと、C-SPY でのブレークポイントの設定 / 解除用コマンド。

#### 表示

ウィンドウを開いたり、表示するツールバーを制御するためのコマンド。

## プロジェクト

プロジェクトへのファイルの追加、グループの作成、現在のプロジェクトでの IAR システムズツールの実行のためのコマンド。

## シミュレータ

C-SPY シミュレータに固有のコマンド。このメニューは、**【オプション】** ダイアログボックスでシミュレータドライバを選択した場合のみ使用できます。

## C-SPY ハードウェアドライバ

使用する C-SPY ハードウェアデバッグドライバに固有のコマンド、つまり **【オプション】** ダイアログボックスで選択した C-SPY ドライバ。一部の IAR Embedded Workbench 製品では、メニュー名に使用する C-SPY ドライバ名が反映され、その他についてはメニュー名は **【エミュレータ】** となります。

## ツール

ユーザが設定可能なメニューで、IDE と使用するツールをこれに追加できます。

## ウィンドウ

IDE ウィンドウの操作や画面上での配置変更のコマンド。

## ヘルプ

IDE に関するヘルプを提供するコマンド。

各メニューの詳細については、195 ページのメニューを参照してください。

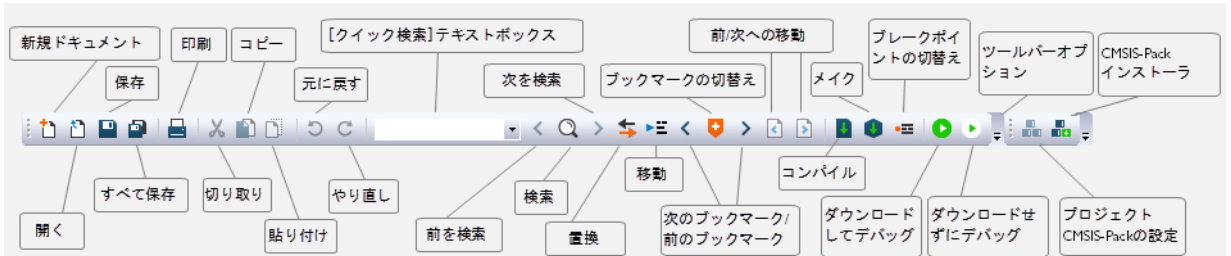
## ツールバー



IDE ツールバーには、IDE のメニューで最も便利なコマンドを実行するためのボタンと、文字列を入力してすばやく検索するためのテキストボックスがあります。ツールバーへのボタンの追加および削除の方法については、26 ページの *IDE の使用およびカスタマイズ* を参照してください。

マウスポインタでボタンをポイントすると、そのボタンの説明が表示されます。コマンドが使用できない場合は、対応するツールバーボタンは灰色表示され、クリックできないようになっています。

ツールバーはドッキング可能 ; アレンジするにはドラッグアンドドロップします。

下図に、各ツールバーボタンに対応するメニューコマンドを示します。



注：C-SPY の起動時、**Download and Debug** ボタンは [メイク後デバッグを再起動] ボタン  に、[ダウンロードせずにデバッグ] は [デバッグを再起動] ボタン  にそれぞれ変わります。

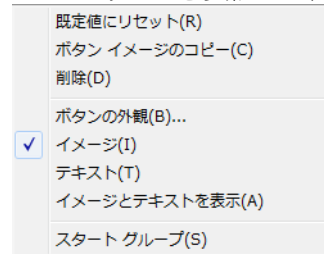
#### ツールバーオプション



[ツールバーオプション] ボタンをクリックして [ツールバーオプション] メニューを開きます。

#### コンテキストメニュー

[カスタマイズ] ダイアログ ボックスが開いているときは、ツールバーのボタンを右クリックすると、このコンテキストメニューを使用できます。このダイアログボックスを開く方法については、50 ページの [カスタマイズ] ダイアログボックスを参照してください。



以下のコマンドがあります。

#### 規定値にリセット

ボタンアイコンを非表示にして代わりにボタン名を表示します。

#### ボタンイメージのコピー

ボタンアイコンをコピーして、クリップボードに画像を保存します。

#### 削除

ツールバーからボタンを削除します。

### ボタンの外観

【ボタンの外観】ダイアログボックスを表示します（53 ページのボタンの外観ダイアログボックスを参照）。

### イメージ

アイコンとしてだけボタンを表示します。

### テキスト

テキストとしてだけボタンを表示します。

### イメージとテキストを表示

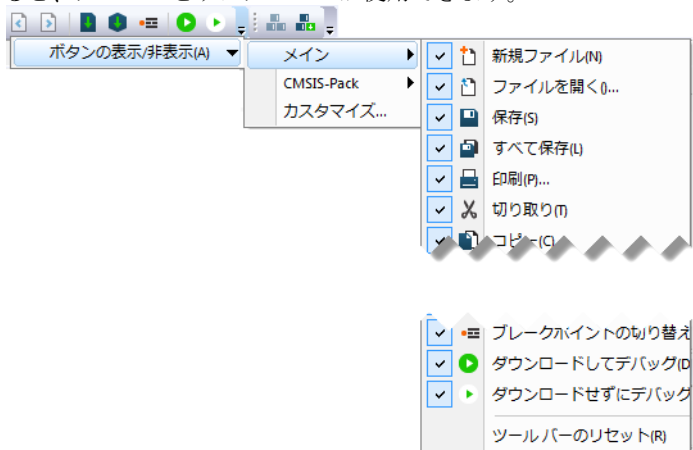
ボタンのアイコンとテキストの両方を表示します。

### スタートグループ

ボタンの左側に区切り文字を挿入します。

## ツールバーオプションメニュー

ツールバーの一番右側にある【ツールバーオプション】ボタンをクリックすると、メニューとサブメニューが使用できます。



以下のコマンドがあります。

### ボタンの表示 / 非表示

サブメニューを開きます。



### ツールバー

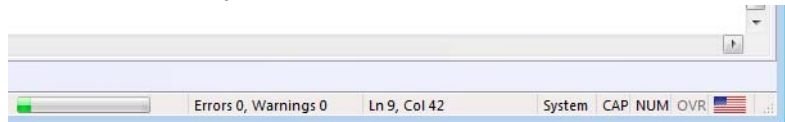
ツールバーにあるすべてのコマンドボタンのリストのサブメニューを開きます。ツールバーに表示委 / 非表示するボタンのチェックボックスを選択または選択解除します。[ツールバーのリセット] を選択してデフォルトで表示されたようにツールバーを戻します。

### カスタマイズ

[カスタマイズ] ダイアログボックスを表示します (50 ページの [カスタマイズ] ダイアログボックスを参照)。

### ステータスバー

ウィンドウの下にあるステータスバーは、[ウィンドウ] メニューから有効にすることができます。

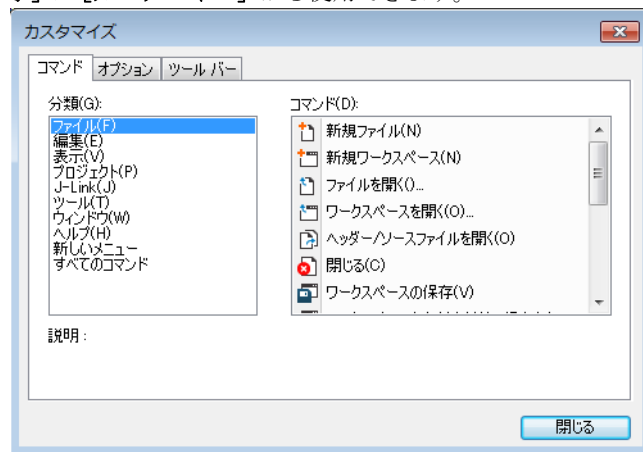


ステータスバーには以下のものが表示されます。

- ソースブラウザの進行状況
- ビルド中に発生したエラーとワーニングの数
- エディタウィンドウの挿入ポイントの位置。編集の際は、ステータスバーに挿入ポイントを含む現在の行と列番号が表示されます
- 文字エンコーディング
- 修飾キー Caps Lock、Num Lock、上書きの状態
- 製品パッケージが英語以外の言語でも入手できる場合、隅に表示されるフラグが、使用中の言語バージョンを示します。言語を変更するには、このフラグをクリックします。変更は次回 IDE を起動したときに反映されます

## 「カスタマイズ」 ダイアログボックス

「カスタマイズ」 ダイアログボックスは、メイン IDE ウィンドウの一番右端にある「ツールバーオプション」 ボタンをクリックし、「ボタンの表示 / 非表示」 > 「カスタマイズ」 から使用できます。



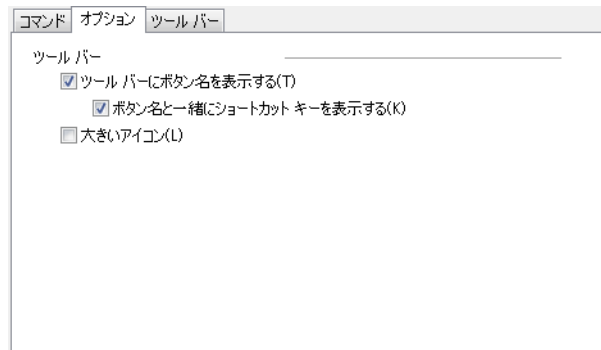
「カスタマイズ」 ダイアログ ボックスの「コマンド」 ページにオプションがあります。

### 分類

IDE のメニューをリストします。メニュー名を選択すると、ツールバーにボタンとして追加するために、コマンドをメニューで使用できるようになります。「新しいメニュー」を選択すると、ツールバーにカスタムドロップダウンメニューを追加できます。

### コマンド

ツールバーの 1 つにドラッグでき、ボタンとして追加できるメニューコマンドをリストします。「新しいメニュー」で「カテゴリ」を選択すると、コマンド「新しいメニュー」をツールバーにドラッグすることができ、カスタムドロップダウンメニューがツールバーに追加されます。「コマンド」リストのコマンドはカスタムメニューを読み込む為にドラッグできます。



[カスタマイズ] ダイアログ ボックスの [オプション] ページにオプションがあります。

#### ツールバーにボタン名を表示する

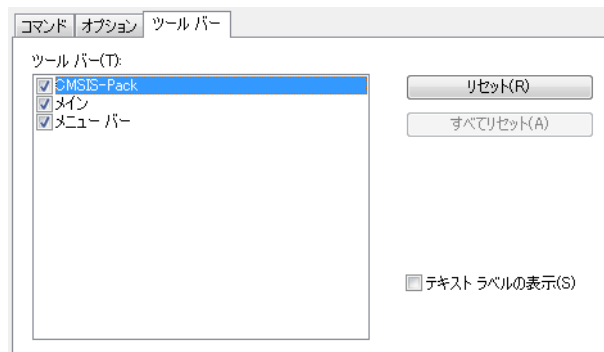
ツールバーのボタンのツールチップを有効にします。ツールチップにはボタンの名前が表示されます。

#### ボタン名と一緒にショートカットキーを表示する

ツールバーにあるボタンのツールチップテキストにはキーボードのショートカットが含まれます。

#### 大きいアイコン

ツールバーのボタンのサイズを大きくします。



【カスタマイズ】ダイアログ ボックスの【ツールバー】ページにオプションがあります。

### ツールバー

ツールバーを選択 / 選択解除してメイン IDE ウィンドウでそれを表示 / 非表示します。メニューバーは非表示にできません。

### リセット

選択したツールバーをデフォルト表示に戻します。

### すべてリセット

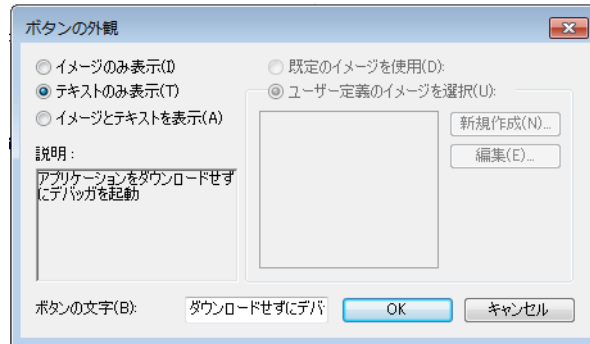
このボタンは無効です。

### テキストラベルの表示

選択したツールバーのボタン名を表示します。

## ボタンの外観ダイアログボックス

[カスタマイズ] ダイアログ ボックスが開いているときに、ツールバーを右クリックし、コンテキストメニューから [ボタンの外観] を選択すると、[ボタンの外観] ダイアログボックスを使用できます。



このダイアログ ボックスを使用してツールバーのボタンの表示名を変更します。

### イメージのみ表示

このオプションは影響しません。

### テキストのみ表示

テキストボックス [ボタンの文字] を有効にします。

### イメージとテキストを表示

テキストボックス [ボタンの文字] を有効にします。

### 既定のイメージを使用

このオプションは無効です。

### ユーザー定義のイメージを選択

このオプションは無効です。

### 新規作成

このボタンは無効です。

### 編集

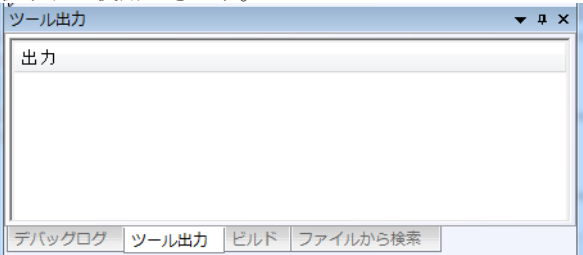
このボタンは無効です。

ボタンの文字

ツールバーのボタンの表示名。テキストを編集して名前を変更します。

[ツール出力] ウィンドウ

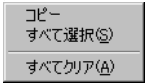
[ツール出力] ウィンドウは、[表示] > [メッセージ] > [ツール出力] を選択すれば使用できます。



このウィンドウには、[ツール] メニューのユーザ定義ツールによるすべてのメッセージ出力が表示されます。ただし、[ツールの設定] ダイアログボックスで [出力ウィンドウにリダイレクト] オプションを選択している必要があります (83 ページの [ツールの設定] ダイアログボックスを参照)。デフォルトでは、このウィンドウは他のメッセージウィンドウとグループ化されて表示されます。

コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

コピー

ウィンドウの内容をコピーします。

すべて選択

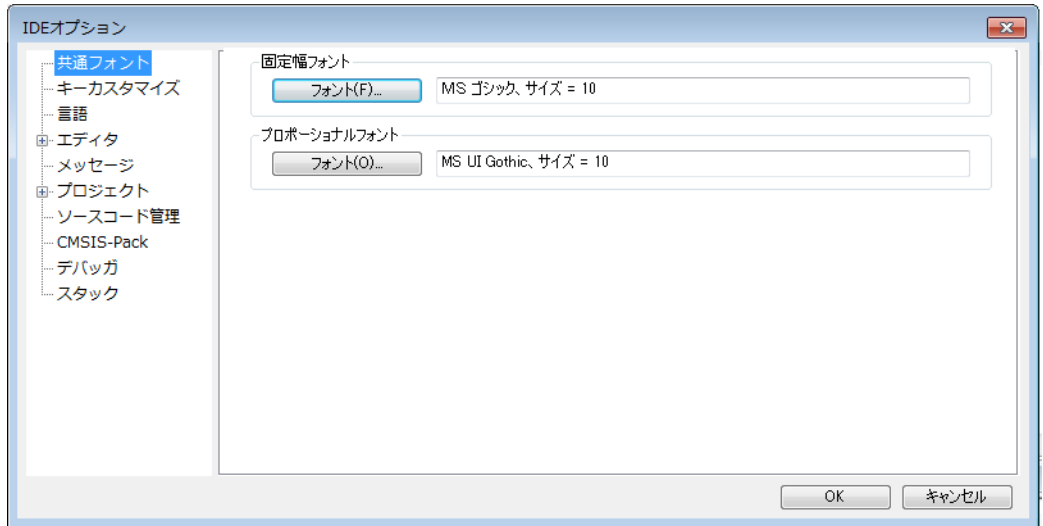
ウィンドウの内容を選択します。

すべてをクリア

ウィンドウの内容を削除します。

## 【共通フォント】 オプション

【共通フォント】 オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、エディタウィンドウを除くすべてのプロジェクトウィンドウで使用するフォントを設定します。

エディタウィンドウのフォントの変更方法については、67 ページの [色とフォント] オプションを参照してください。

### 固定幅フォント

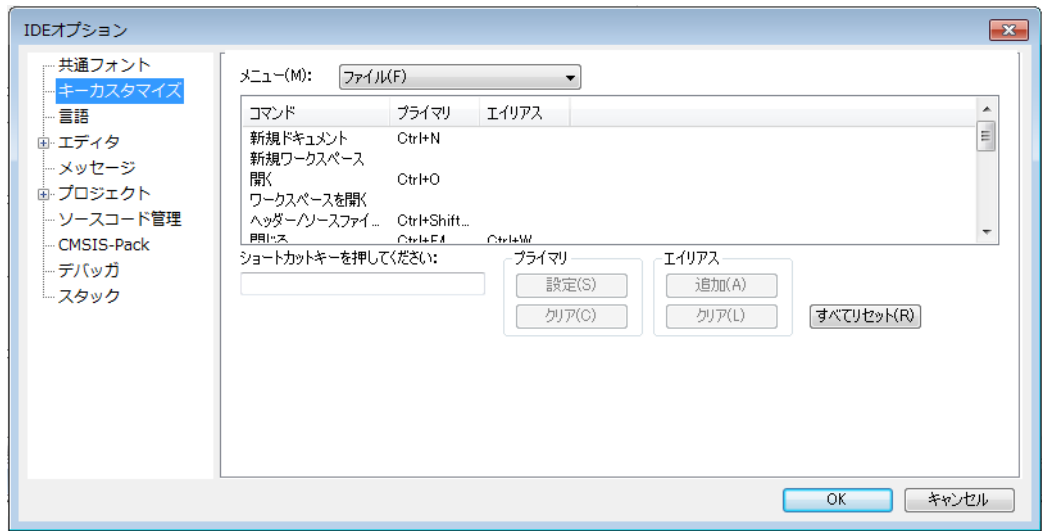
[逆アセンブリ]、[レジスタ]、[メモリ] の各ウィンドウで使用するフォントを選択します。

### プロポーショナルフォント

[逆アセンブリ]、[レジスタ]、[メモリ] およびエディタウィンドウを除く、すべてのウィンドウで使用するフォントを選択します。

## 「キーカスタマイズ」オプション

「キー カスタマイズ」 オプションは、「ツール」 > 「オプション」 を選択すると使用できます。



このページを使用して、IDE のメニューコマンドで使用されるショートカットキーをカスタマイズします。

### メニュー

編集するメニューを選択します。選択したメニューについて現在定義されているすべてのショートカットキーが、「メニュー」 ドロップダウンリストの下に一覧表示されます。

### コマンドのリスト

独自のショートカットキーを設定するメニューコマンドを、選択したメニューで使用可能なすべてのコマンドのリストから選択します。

### ショートカットキーを押してください

選択したコマンドのショートカットキーとして使用するキーの組合せを入力します。他のコマンドで使用されているショートカットの設定や追加はできません。



## プライマリ

以下から選択します。

### セット

キーの組合せを、リストで選択したコマンドのショートカットとして  
[ショートカットキーを押してください] フィールドに保存します。

### クリア

リストで選択したコマンドのショートカットとして表示されたプライ  
マリキーの組合せを削除します。

メニューコマンド名の横に新しいショートカットキーが表示されます。

## エイリアス

以下から選択します。

### 追加

キーの組合せを、リストで選択したコマンドのエイリアス（表示され  
ないショートカット）として [ショートカットキーを押してください]  
フィールドに保存します。

### クリア

リストで選択したコマンドのショートカットとして表示されたエイリ  
アスのキーの組合せを削除します。

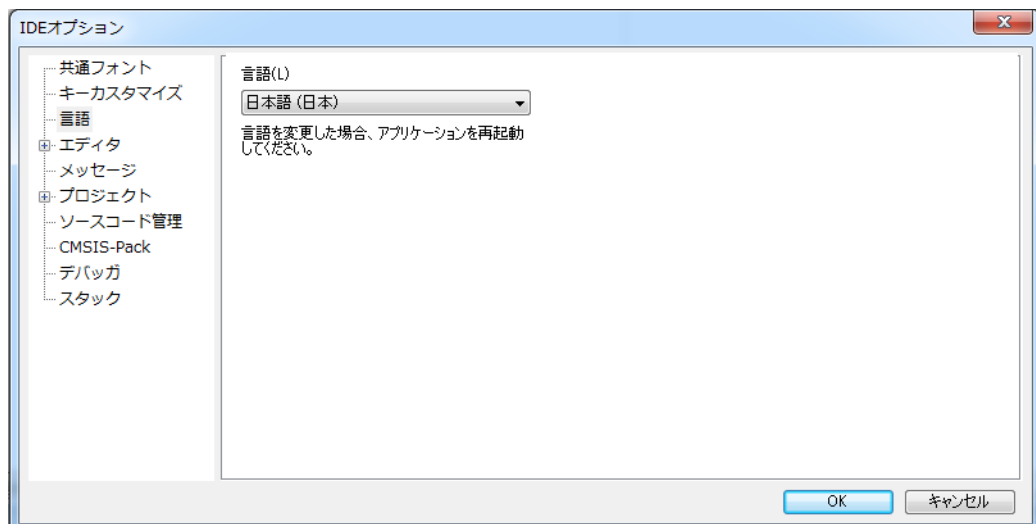
メニューコマンド名の横に新しいショートカットキーは表示されません。

## すべてリセット

すべてのコマンドショートカットキーを出荷時設定に戻します。

## 【言語】 オプション

【言語】 オプションは 【ツール】 > 【オプション】 を選択すると使用できます。



このページを使用して、ウィンドウやメニュー、ダイアログボックスなどで使用する言語を指定します。

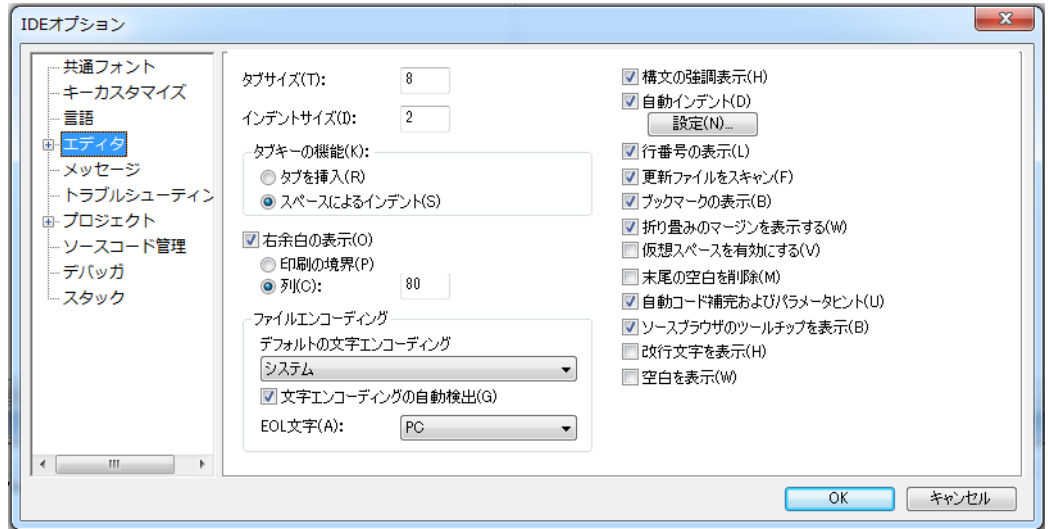
### 言語

使用する言語を指定します。使用可能な言語は製品パッケージに応じて、**英語（米国）** および **日本語（日本）** となります。

**注：** 同一ディレクトリの複数の異なるツールチェーンに対して IAR Embedded Workbench がインストールされ、これらのツールチェーンで異なる言語が使用可能である場合、IDE で言語が混在することがあります。

## 【エディタ】オプション

【エディタ】オプションは、【ツール】>【オプション】を選択すると使用できます。



このページを使用して、エディタを設定します。エディタについて詳しくは、139 ページの **編集** を参照してください。

### タブサイズ

タブ文字の幅を文字間隔で指定します。

### インデントサイズ

インデント付きで表を作成するとき使用するスペースの数を指定します。

### タブキーの機能

【タブ】キーを押したときの動作を制御します。以下から選択します。

#### タブを挿入

タブキーを押すと、タブ文字を 1 つ挿入します。

#### スペースによるインデント

タブキーを押したときに、インデント（スペース文字）を 1 回分挿入します。

## 右余白の表示

エディタウィンドウの右側の余白部分の外側領域が薄い灰色で表示されます。このオプションを選択すると、左右の余白の間にあるテキストエリアの幅を設定できます。以下を基準に幅を選択して設定します。

### 印刷の境界

印刷可能な領域（プリンタの一般設定から読み込まれます）を基準に幅を決定します。

### 列

列数を基準に幅を決定します。

## ファイルエンコーディング

ファイルのエンコーディングを制御します。以下から選択します。

### デフォルトの文字エンコーディング

新しいファイルでデフォルトとして使用される文字エンコーディングを選択します。以下から選択します。

システム（Windows の設定を使用）

西ヨーロッパ言語

UTF-8

日本語 (Shift-JIS)

簡体中国語 (GB2312)

繁体中国語 (Big5)

韓国語 (Unified Hangul Code)

アラビア語

西ヨーロッパ言語

ギリシャ語

ヘブライ語

タイ語

バルト言語

ロシア語

ベトナム語

エディタウィンドウのコンテキストメニューから文字エンコーディングを指定した場合、そのエンコーディングによって、特定のドキュメントについてこの設定がオーバーライドされます。

### 文字エンコーディングの自動検出

既存のドキュメントを開くときに使用する文字エンコーディングが自動的に検出されます。

## EOL 文字

エディタの文書を保存したときに使用する改行文字を選択します。以下から選択します。

**PC** (デフォルト)。Windows と DOS 形式の改行文字。

**UNIX**、UNIX 形式の改行文字。

**元ファイルに従う**。開かれたときファイルに設定されているのと同じ、PC または UNIX のどちらかの形式の改行文字。開かれたファイルに両方の形式が存在する場合や、どちらも存在しない場合には、PC 形式の改行文字が使用されます。

## 構文の強調表示

C/C++ アプリケーションの構文をさまざまなテキスト形式でエディタに表示します。

構文強調表示の詳細については、67 ページの **「色とフォント」オプション**、146 ページの **「構文カラー表示」** を参照してください。

## 自動インデント

Return キーを押すと、新しい行が自動的にインデントされます。C/C++ ソースファイルの場合、**「設定」** ボタンをクリックして自動インデント機能を設定します (63 ページの **「自動インデントの設定」** ダイアログボックスを参照)。他のテキストファイルの場合は、新しい行のインデントは前の行と同一に設定されます。

## 行番号の表示

エディタウィンドウに行番号を表示します。

## 更新ファイルをスキャン

他のツールで修正されたファイルをエディタで再ロードします。

ファイルが IDE で開かれていて、同じファイルが同時に別のツールで修正されている場合、そのファイルが自動的に IDE で再ロードされます。ただし、ファイルの編集をすでに開始している場合、ファイルを再ロードする前にプロンプトが表示されます。

## ブックマークの表示

エディタウィンドウの左側に列を表示します。この列には、コンパイラのエラーとワーニング、**「ファイルから検索」** の結果、ユーザのブックマーク、ブレイクポイントのアイコンが表示されます。

### 折り目の余白を表示

エディタで、エディタウィンドウの左側に折り目の余白を表示します。詳細については、142 ページの *コードの折りたたみ* を参照してください。

### 仮想スペースを有効にする

挿入ポイントをテキストエリアの外側に動かせるようにします。

### 末尾の空白を削除

ファイルをディスクに保存するときに、末尾の空白を削除します。最後の空白とは、空白以外の最後の文字と行末文字の間の空白文字です。

### コードの自動補完およびパラメータのヒント

コードの補完およびパラメータのヒントを有効にします。詳細については、140 ページの *ファイルの編集* を参照してください。

### ソースブラウザのツールチップを表示

現在カーソルの位置にある識別子に関する詳細情報の表示を切り替えます。

### 改行文字を表示

エディタウィンドウで、キャリッジリターンと改行文字の表示を切り替えます。

### 空白を表示

エディタウィンドウでは、半角の空白スペースにはピリオド (.)、タブには (>) の表示に切り替えます。

## 〔自動インデントの設定〕 ダイアログボックス

〔自動インデントの設定〕 ダイアログボックスは、〔IDE オプション〕 ダイアログボックスの 〔エディタ〕 カテゴリから使用できます。



このダイアログボックスを使用して、エディタによる C/C++ ソースコードの自動インデントを設定します。

インデントの詳細は、141 ページのテキストの *自動インデント* を参照してください。

### 左括弧 (a)

左括弧をインデントするときの空白文字数を指定します。

### 本文 (b)

左括弧の後または次の行にまたがる文の後のコードのインデントに使用する追加空白文字数を指定します。

### ラベル (c)

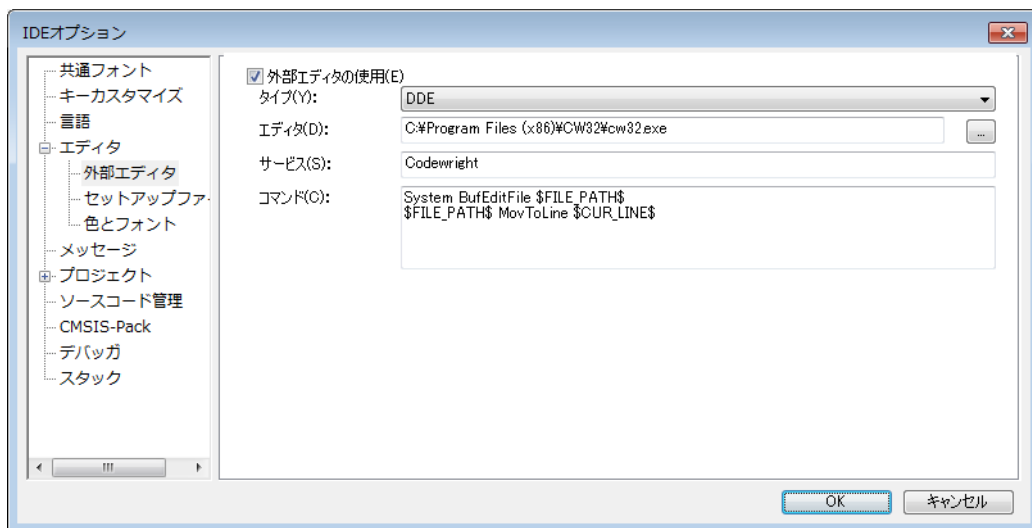
ケースラベルを含むラベルをインデントする際に使用する追加空白文字数を指定します。

### サンプルコード

このエリアに、テキストボックスで設定したインデント用設定が反映されます。すべてのインデントは、前の行、文、その他の文法構造と相対的に設定されます。

## 【外部エディタ】のオプション

【外部エディタ】オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、外部エディタを指定します。

**注：**このダイアログボックスの内容は、[タイプ] オプションの設定によって異なります。

41 ページの *外部エディタの連携* を参照してください。

### 外部エディタの使用

外部エディタを使用を有効にします。

### タイプ

インタフェースのタイプを選択します。以下から選択します。

- コマンドライン
- DDE (Windows Dynamic Data Exchange)

### エディタ

外部エディタのファイル名とパスを指定します。参照ボタンを使用して選択することもできます。



## 引数

エディタに引き渡す引数を指定します。インタフェースのタイプとして [コマンドライン] を選択した場合に限り、適用できます。

## サービス

エディタで使用する DDE サービス名を指定します。インタフェースのタイプとして [DDE] を選択した場合に限り適用できます。

サービス名は、使用する外部エディタに応じて指定します。外部エディタのドキュメントを参照して、適切に設定してください。

## コマンド

エディタに引き渡すコマンド文字列のシーケンスを指定します。コマンド文字列は、以下の形式で入力する必要があります。

```
DDE-Topic CommandString1  
DDE-Topic CommandString2
```

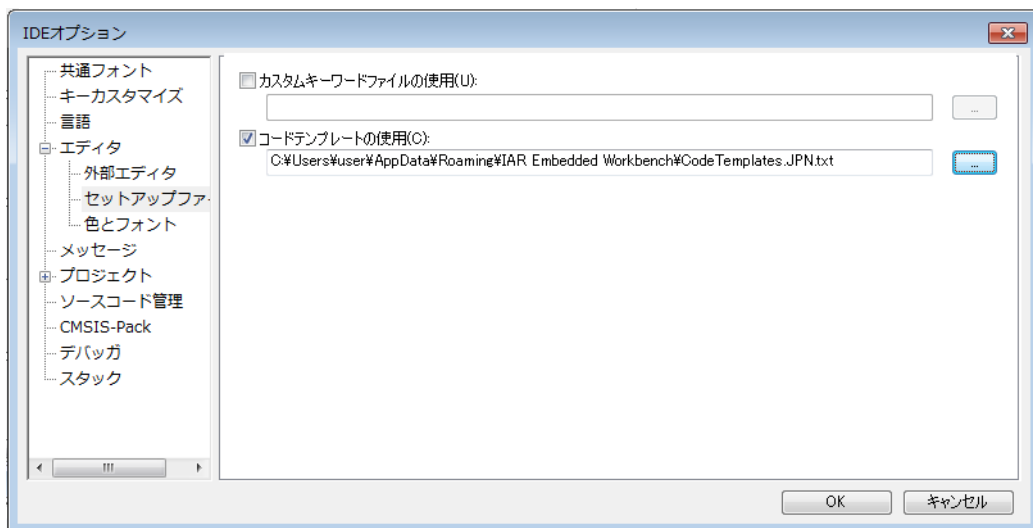
インタフェースのタイプとして [DDE] を選択した場合に限り適用できます。

コマンド文字列は、使用する外部エディタに応じて指定します。外部エディタのドキュメントを参照して、適切に設定してください。

**注：**引数に変数を使用できます (91 ページの *引数変数* を参照)。

## [セットアップファイル] オプション

[エディタセットアップファイル] オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、エディタのセットアップファイルを指定します。

### カスタムキーワードファイルの使用

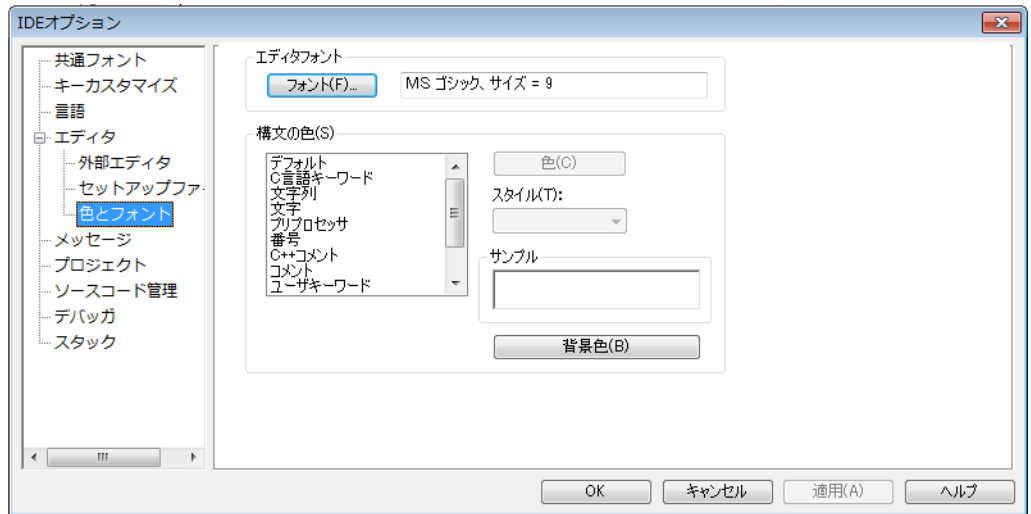
エディタで強調表示するキーワードを含むテキストファイルを指定します。構文カラー表示については、146 ページの [構文カラー表示](#) を参照してください。

### コードテンプレートの使用

頻繁に使用するコードをソースファイルに挿入するためのコードテンプレートを記述したテキストファイルを指定します。コードテンプレートの使用方法については、144 ページの [コードテンプレートの使用と追加](#) を参照してください。

## 【色とフォント】オプション

【色とフォント】オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、エディタウィンドウのテキストに使用する色とフォントを指定します。アセンブラおよび C/C++ ソースコードの構文強調表示を制御するキーワードは、それぞれ `syntax_icc.cfg` と `syntax_asm.cfg` のファイルに記述されています。これらのファイルは、`arm%config` ディレクトリにあります。

### エディタフォント

[フォント] ボタンをクリックして、標準 [フォント] ダイアログを開き、エディタウィンドウで使用するフォントとそのサイズを選択できます。

### 構文の色

構文の要素をリストで選択して、色とスタイルを設定します。

#### 色

選択可能な色が一覧表示されます。リストから [カスタム] を選択して、自分の色を定義します。

#### スタイル

選択した要素について、[通常]、[太字]、[斜体] のスタイルを選択します。

### サンプル

選択した要素の現在の外観を表示します。

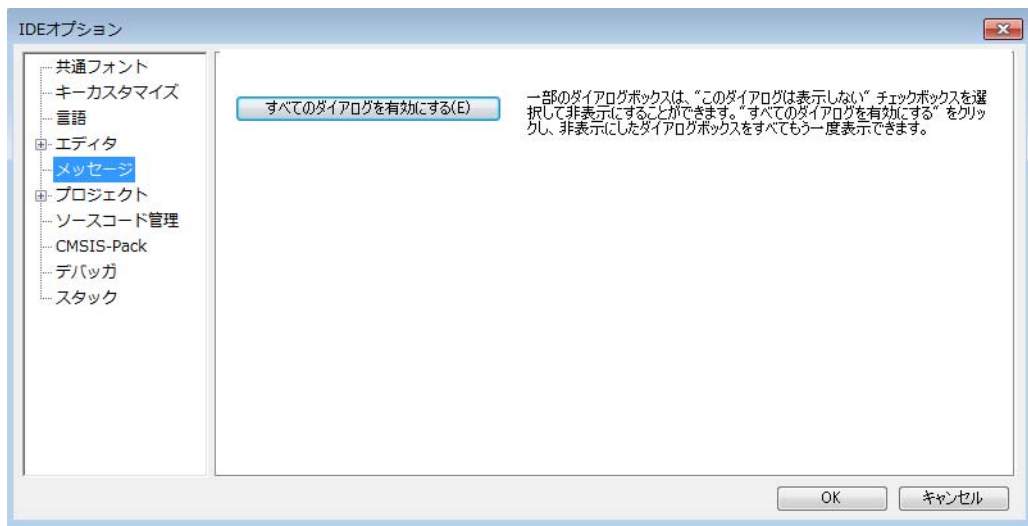
### 背景色

エディタウィンドウの背景色をクリックして設定します。

**注:** [ユーザーキーワード] 構文要素は、カスタムキーワードファイルにリストしたキーワードを参照します (66 ページの [セットアップファイル] オプションを参照)。

## [メッセージ] オプション

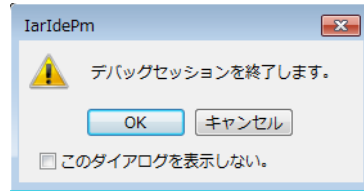
[メッセージ] オプションは [ツール] > [オプション] を選択すると使用できます。



このページを使用して以前に無効にしたすべてのダイアログボックスを再び有効にできます。

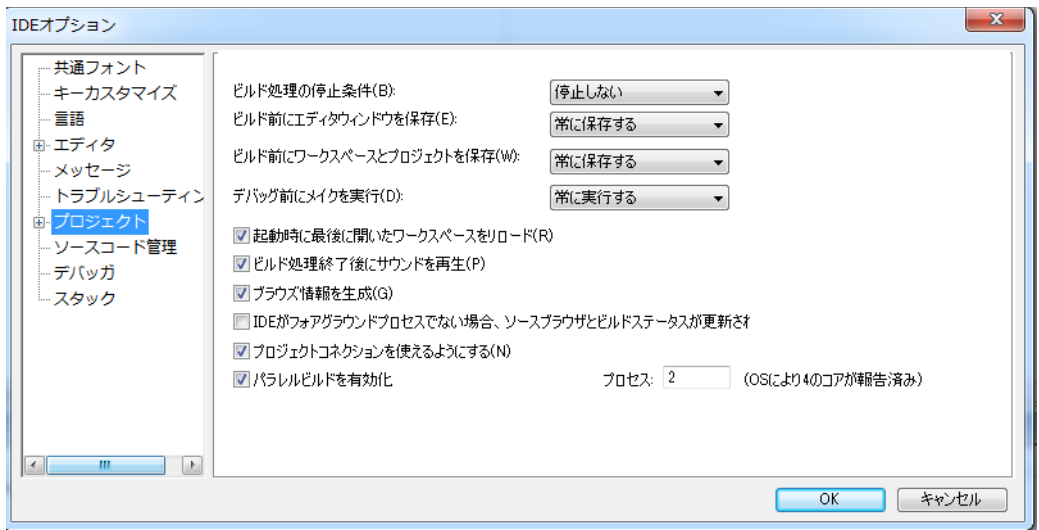
## すべてのダイアログを有効にする

たとえば、[次回からこのダイアログを表示しない] チェックボックスを選択して非表示にしたすべてのダイアログボックスを有効にできます。



## 【プロジェクト】 オプション

【プロジェクト】 オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、[メイク] と [ビルド] コマンドにオプションを設定します。

## ビルド処理の停止条件

ビルド処理の停止条件を選択します。以下から選択します。

### 停止しない

停止しません。

### ワーニング

ワーニングやエラーで停止します。

### エラー

エラーで停止します。

## ビルド前にエディタウィンドウを保存

ビルド処理の前にエディタウィンドウを保存するタイミングを選択します。以下から選択します。

### 保存しない

保存しません。

### 保存前に確認する

保存前に確認します。

### 常に保存する

[メイク] / [ビルド] の実行前に常に保存します。

## ビルド前にワークスペースとプロジェクトを保存

ビルド処理の前に、ワークスペースとインクルードされたプロジェクトをいつ保存するかを選択します。以下から選択します。

### 保存しない

保存しません。

### 保存前に確認する

保存前に確認します。

### 常に保存する

[メイク] / [ビルド] の実行前に常に保存します。

## デバッグ前にメイクを実行

デバッガセッションを開始するにあたってメイク処理を実行するタイミングを選択します。以下から選択します。

### 実行しない

デバッグセッションの前にメイク処理を実行しません。

### 実行前に確認する

メイク処理を実行する前に確認します。

### 常に実行する

デバッグセッションの前にメイク処理を常に実行します。

### 起動時に最後に開いたワークスペースをリロード

次に IAR Embedded Workbench IDE を起動するときに、前回アクティブだったワークスペースを自動的にロードします。

### ビルド処理終了後にサウンドを再生

ビルド処理の完了時に音を再生します。

### ブラウザ情報を生成

[ソースブラウザ] ウィンドウに表示させるソースブラウザ情報の生成を有効にします (174 ページの [ソースブラウザ] ウィンドウ参照)。

### IDE がフォアグラウンドプロセスでない場合、ソースブラウザとビルドステータスが更新されません

IDE がフォアグラウンドプロセスでない場合、ソースブラウザを停止します。これは、ビルドステータスが [ワークスペース] ウィンドウで更新されないということでもあります。このオプションは、ラップトップコンピュータの使用時に消費電力を抑えたい場合に役立ちます。

### プロジェクト接続を有効化

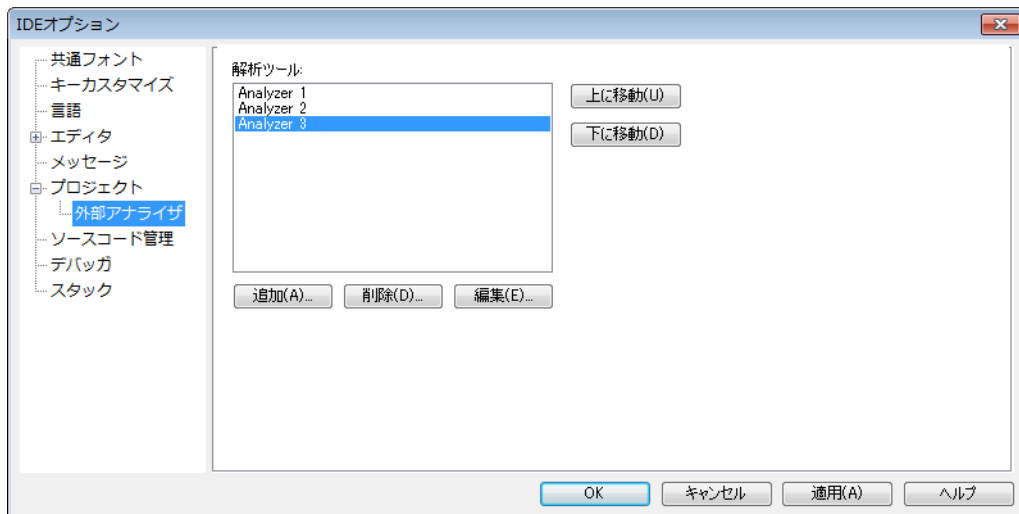
ライブのプロジェクト接続の設定サポートを有効化します (119 ページの [プロジェクトコネクションの追加] ダイアログボックスを参照)。

### パラレルビルドを有効化

パラレルビルドのサポートを有効化します。コンパイラは CPU で使用可能なコアを有効利用するため、いくつかの並列処理で実行されます。[プロセス] テキストボックスで、使用するプロセスの数を指定します。使用可能なコアをすべて用いると、IDE の反応が鈍くなることがあります。

## 【外部アナライザ】のオプション

【外部アナライザ】オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、標準のビルドツールチェーンに外部アナライザを追加します。外部アナライザは、ユーザープロジェクトの C/C++ ソースコードで動作します。ヘッダファイルまたはアセンブラ ソースコードファイルは、解析されません。

詳細については、37 ページの *外部のアナライザを使用するにあたって* を参照してください。

### 解析ツール

標準のビルドツールチェーンに追加した外部アナライザを一覧表示します。



**上に移動**

リストで選択したアナライザを上 に 1 つ移動します。この順序は [プロジェクト] メニューに反映されます。

**下に移動**

リストで選択したアナライザを下 に 1 つ移動します。この順序は [プロジェクト] メニューに反映されます。

**追加**

[外部アナライザ] ダイアログボックスが表示され、ここで新しいアナライザをツールチェーンに追加して、アナライザの呼出しを設定することができます。

**削除**

選択したアナライザをアナライザのリストから削除します。

**編集**

[外部アナライザ] ダイアログボックスが表示され、ここで選択したアナライザの呼出しに関する詳細を編集できます。

**[外部アナライザ] ダイアログボックス**

[外部アナライザ] ダイアログボックスは、[ツール] > [オプション] > [プロジェクト] > [外部アナライザ] を選択すると使用できます。

外部解析ツール

名称: Analyzer 1

パス: C:\Program Files (x86)\MyAnalyzerTool\Analyzer1

引数: -nc \$FILE\_PATH\$ \$COMPILER\_ARGS\$

一致パターンを出力

ロケーション: \$FILE\_NAME\$: \$LINE\_NUMBER\$

ワーニング: [?]\warning[?-:]

エラー: [?]\error[?-:]

OK キャンセル

このダイアログボックスを使用して、標準のビルドツールチェーンに追加する外部アナライザの呼出しを設定します。

外部アナライザは、ユーザープロジェクトの C/C++ ソースコードで動作します。ヘッダファイルまたはアセンブラ ソースコードファイルは、解析されません。

詳細については、37 ページの *外部のアナライザを使用するにあたって* を参照してください。

## 名前

外部アナライザの名前を指定します。名前は一意でなければなりません。

## パス

アナライザの実行可能ファイルのパスを指定します。参照ボタンを使用して選択することもできます。

## 引数

アナライザに渡す引数を指定します。

引数の指定に引数変数を使用できる点に注意してください (91 ページの *引数変数* を参照)。

## ロケーション

ソースファイルの場所を検索するための正規表現を指定します。この正規表現は各出力行に適用され、**[ビルドログ]** ウィンドウにテキストとして表示されます。指定する正規表現に一致する行をダブルクリックすることができます。

引数変数 `$FILE_NAME$`、`$LINE_NUMBER$`、`$COLUMN_NUMBER$` を使用して、ファイル名や行番号、列番号をそれぞれ識別することができます。事前に定義された以下の表現のいずれかを選択します。

**`¥"?$FILE_NAME$¥"?:$LINE_NUMBER$`**

たとえば、フォーム `file.c:17` の位置を一致させます。

**`¥"?$FILE_NAME$¥"? +$LINE_NUMBER$`**

たとえば、フォーム `file.c17` の位置を一致させます。

**`¥"?$FILE_NAME$¥"?`**

たとえば、フォーム `file.c` の位置を一致させます。

または、独自の表現を指定できます。たとえば、正規表現 `Msg: $FILE_NAME$ @ $LINE_NUMBER$` を出力文字列 `Msg:MySourceFile.c @ 32` に適用すると、こ

のファイルは `MySourceFile.c` として、行番号は 32 としてそれぞれ識別されます。

## ワーニング

この表現に一致する出力行にはすべて、ワーニングシンボルが付きます。

たとえば、`(?i)warning(?-i):` という表現は、`warning:` という文字列（大文字小文字の別を問わず）を含むすべての行をワーニングとして識別します。

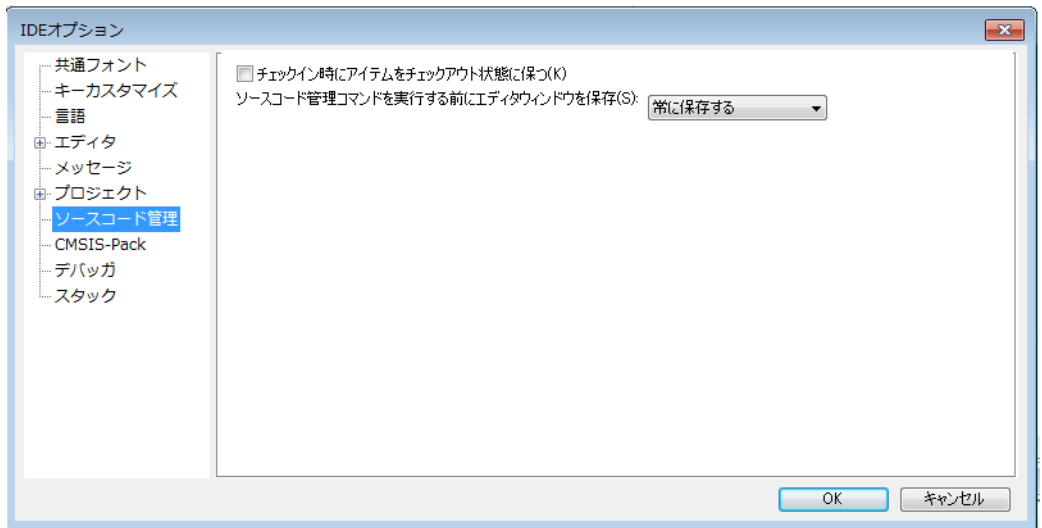
## エラー

この表現に一致する出力行にはすべて、エラーシンボルが付きます。エラーはワーニングに優先します。

たとえば、`(?i)error(?-i):` という表現は、`error:` という文字列（大文字小文字の別を問わず）を含むすべての行をエラーとして識別します。

## ソースコード管理オプション（廃止）

[ソースコード管理] オプションは、[ツール] > [オプション] を選択すると使用できます。



このページを使用して、IAR Embedded Workbench プロジェクトと SCC プロジェクト間の相互作用を設定します。

**注：**これは廃止されたオプションで新しいプロジェクトではサポートされません。

### **チェックイン時にアイテムをチェックアウト状態に保つ**

[ファイルのチェックイン] ダイアログボックスの [チェックアウトを維持] オプションのデフォルト設定を指定します。

### **ソースコード管理コマンドを実行する前にエディタウィンドウを保存**

ソースコード管理コマンドの実行前にエディタウィンドウを保存するかどうかを決定します。以下から選択します。

#### **保存しない**

ソースコード管理コマンドを実行する前にエディタウィンドウを保存しません。

#### **保存前に確認する**

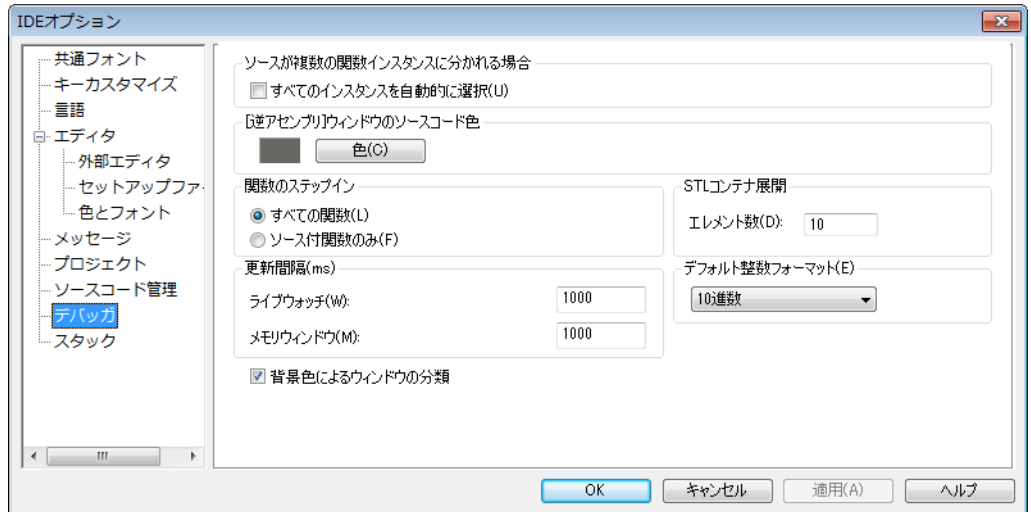
ソースコード管理コマンドを実行する前に確認します。

#### **常に保存する**

ソースコード管理コマンドを実行する前に常にエディタウィンドウを保存します。

## 【デバッガ】 オプション

【デバッガ】 オプションは 【ツール】 > 【オプション】 を選択すると使用できます。



このページを使用して、デバッガ環境を設定します。

### ソースが複数の関数インスタンスに分かれる場合

一部のソースコード（テンプレートコードなど）は、複数のコードインスタンスに対応しています。このようなコードでソース位置を指定する場合（ソースのブレークポイント設定時など）は、C-SPYですべてのインスタンスを選択するか、その一部だけを選択するかを指定できます。【すべてのインスタンス】オプションを使用して、C-SPYで最初に確認しないですべてのインスタンスを処理します。

### 【逆アセンブリ】 ウィンドウのソースコード色

【色】 ボタンをクリックして、【逆アセンブリ】 ウィンドウのソースコードの色を選択します。独自の色を定義するには、リストから【カスタム】を選択します。

### 関数のステップイン

【ステップイン】 コマンドの動作を制御します。以下から選択します。

#### すべての関数

デバッガですべての関数にステップインします。

### ソース付関数

デバッガは、ソースコードを認識する関数だけにステップインします。これにより、ライブラリ関数内のコードのステップ実行や、逆アセンブリモードのデバッグを回避することができます。

### STL コンテナ展開

コンテナの値を **［ウォッチ］** ウィンドウなどで展開したときに、最初に表示されるエレメント数を指定します。

### 更新間隔（ミリ秒）

**［ライブウォッチ］** および **［メモリ］** の各ウィンドウの内容を更新する間隔をミリ秒で指定します。

これらのテキストボックスは、使用している **C-SPY** ドライバがアプリケーションの実行中にターゲットのシステムメモリにアクセス可能な場合にのみ使用できます。

### デフォルト整数フォーマット

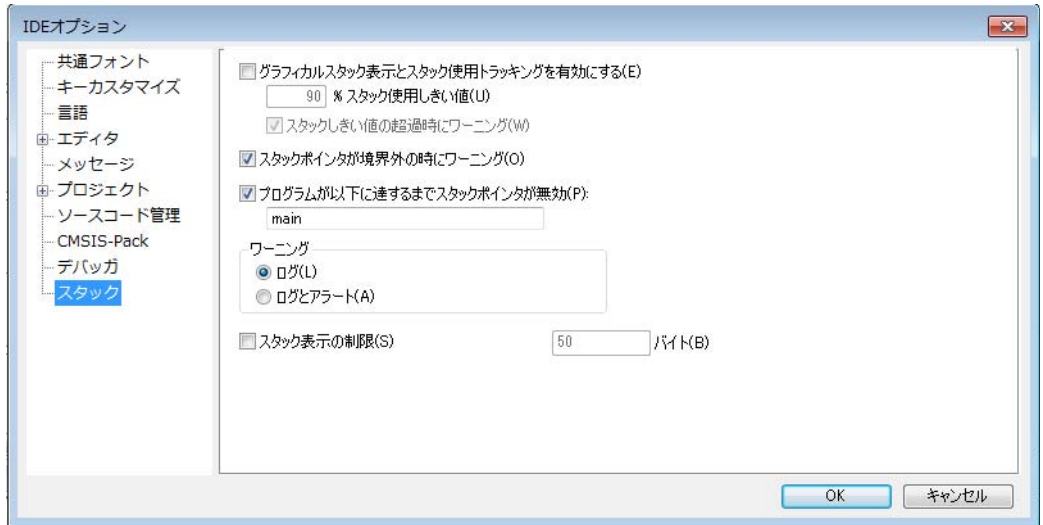
**［ウォッチ］**、**［ローカル］** ウィンドウおよび関連するウィンドウでのデフォルトの整数フォーマットを選択します。

### 背景色によるウィンドウの分類

一部の **C-SPY** ウィンドウで背景色のオンとオフを切り替えます。ウィンドウのタイプを色分けして区別します。たとえば、割込みに関するウィンドウの背景色がある色で、ウォッチ関連のウィンドウはすべて別の色というようにです。

## 【スタック】 オプション

【スタック】 オプションは、【ツール】 > 【オプション】 または 【メモリ】 ウィンドウのコンテキストメニューから選択すると使用できます。



このページを使用して、【スタック】 ウィンドウに固有のオプションを設定します。

### グラフィカルスタック表示とスタック使用トラッキングを有効にする

【スタック】 ウィンドウ上部のグラフィカルスタックバーを有効にします。スタックオーバーフローの検出も有効になります。スタックバーとそこで提供される情報については、『ARM 用 C-SPY® デバッグガイド』を参照してください。

### スタック使用しきい値

C-SPY がスタックオーバーフローについてワーニングを表示するスタック使用量を指定します。

### スタックしきい値の超過時にワーニング

スタック使用量が【スタック使用しきい値】 オプションに指定したしきい値を超えた場合に、C-SPY からワーニングを発生します。

### スタックポインタが境界外の時にワーニング

スタックポインタがスタックメモリ範囲外を指したときに、C-SPY からワーニングを発生します。

## プログラムが以下に達するまでスタックポインタが無効

アプリケーションコード中でスタックの表示と検証を行う位置を指定します。**[スタック]** ウィンドウでは、実行がこの位置に到達するまでは、スタック使用量情報が表示されません。

デフォルトでは、main 関数に到達するまではスタック使用量が表示されません。main 関数がないアプリケーション（たとえば、アセンブラのみのプロジェクト）の場合、独自の開始ラベルを指定する必要があります。このオプションを選択すると、C-SPY は、毎回のリセット後、指定された位置のブレイクポイントに到達するまでこのブレイクポイントを保持します。

通常は、スタックポインタはシステム初期化コード `cstartup` に設定しますが、最初の命令からスタック使用量をトレースする必要はありません。このオプションを使用することで、アプリケーションのこの部分について誤ったワーニングや誤解を招くスタック表示を回避できます。

## ワーニング

ワーニングを出力する場所を選択します。以下から選択します。

### ログ

**[デバッグログ]** ウィンドウにワーニングを表示します。

### ログとアラート

**[デバッグログ]** ウィンドウ、アラートのダイアログボックスにワーニングを表示します。

## スタック表示の制限

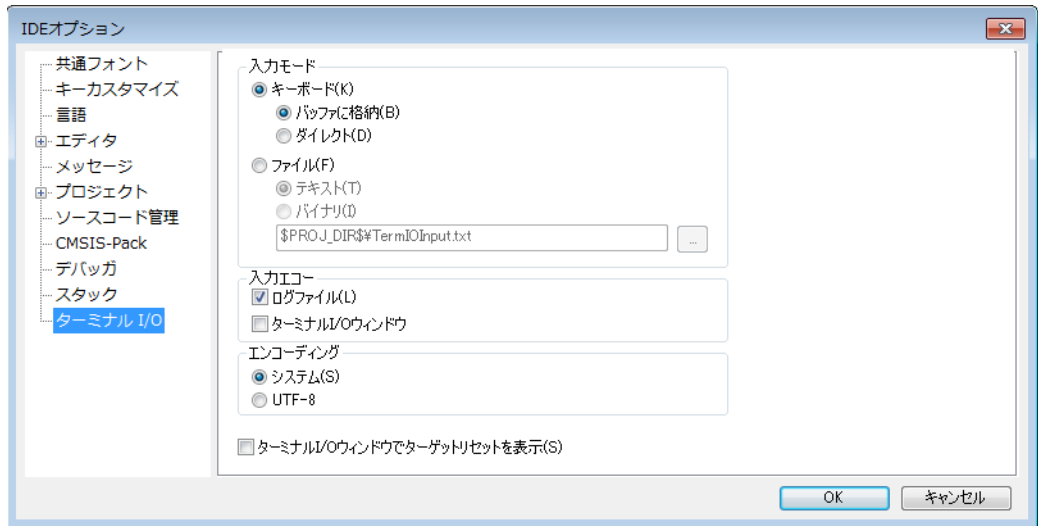
**[スタック]** ウィンドウで表示されるメモリ容量を、スタックポインタからのバイト数で指定します。このオプションは、スタックが大きい場合や、スタックの最初の部分の表示だけが必要な場合に便利です。このオプションを使用すると、特にターゲットシステムからのメモリリード速度が遅い場合に、**[スタック]** ウィンドウの表示速度を向上することができます。デフォルトでは、**[スタック]** ウィンドウにはスタック全体、つまりスタックポインタからスタックの最後までが表示されます。デバッガがスタックのメモリ範囲を特定できない場合は、このオプションを選択していない場合でも、表示されるバイト範囲が制限されます。

**注:** **[スタック]** ウィンドウは、アプリケーションの実行速度には影響しませんが、実行停止時に表示される情報を更新するために、大量のデータをリードする場合があります。



## 【ターミナル I/O】 オプション

【ターミナル I/O】 オプションは、C-SPY を実行しているときは、【ツール】 > 【オプション】 を選択すると使用できます。



このページを使用して、C-SPY のターミナル I/O 機能を設定します。

### 入力モード

ターミナル I/O の入力を読み取る方法を制御します。

**キーボード** キーボードからの入力文字を読み込みます。以下から選択します。

**バッファに格納**：入力された文字をバッファに格納します。

**ダイレクト**：入力された文字をバッファに格納しません。

**ファイル** ファイルからの入力文字を読み込みます。以下から選択します。

**テキスト**：テキストファイルから入力文字を読み込みます。

**バイナリ**：バイナリファイルから入力文字を読み込みます。

参照ボタンを使用して入力ファイルを選択することもできます。

## 入力エコー

入力文字をエコーするかどうか、およびどこにエコーするかを指定します。以下から選択します。

### ログファイル

ターミナル I/O ログファイルの入力文字をエコーします。オプション **[デバッグ]** > **[ログ]** > **[ログの有効化]** を有効にしておく必要があります。

### [ターミナル I/O] ウィンドウ

**[ターミナル I/O]** ウィンドウの入力文字をエコーします。

## エンコーディング

ターミナル入力と出力に使用するエンコードを決定します。以下から選択します。

### システム

Windows の設定を使用します。

### UTF-8

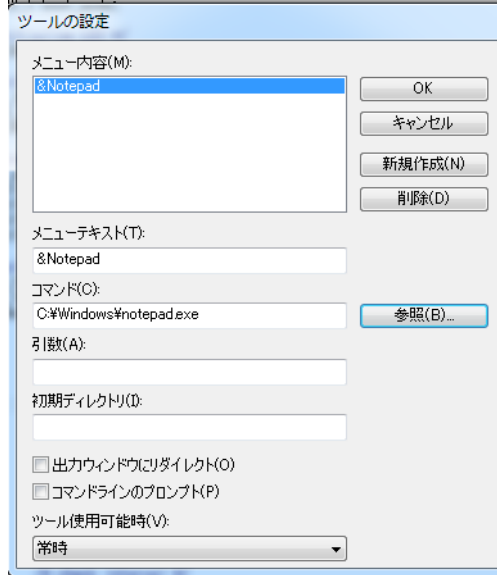
UTF-8 エンコードを使用します。

## ターミナル I/O ウィンドウでターゲットリセットを表示

ターゲットのリセット時に、C-SPY の **[ターミナル I/O]** ウィンドウでメッセージを表示します。

## 【ツールの設定】 ダイアログボックス

【ツールの設定】 ダイアログボックスは、【ツール】 メニューから表示します。



このダイアログボックスを使用して、メモ帳など【ツール】メニューに追加するツールを指定します。



**注：**標準ビルドツールチェーンに外部ツールを追加する場合は、「123 ページのツールチェーンの拡張」を参照してください。

引数で変数を使用することによって、コマンドラインのレビジョン管理システムとのインターフェースや、選択したファイルに対して外部ツールを実行するなどの便利なツールを設定できます。

コマンドラインのコマンドやバッチファイルを【ツール】メニューに追加するする：

- I 【コマンド】 テキストボックスで、cmd.exe コマンドシェルを入力または検索します。

- 2 [引数] テキストボックスで、以下のようにコマンドラインコマンドかバッチファイル名を入力します。

```
/C name
```

ここで、*name* は、実行するコマンドかバッチファイルの名前です。

/c オプションは、実行後にシェルを終了するように指定し、ツールの終了を IDE が検出できるようにします。

例については、41 ページの [ツール] メニューへのコマンドラインコマンドの追加を参照してください。

## 新規作成

このダイアログボックスを使用して設定する新しいメニューコマンドにスタブを作成します。

## 削除

[メニュー内容] リストで選択されたコマンドを削除します。

## メニュー内容

定義したすべてのメニューコマンドを一覧表示します。

## メニューテキスト

メニューコマンドの名前を指定します。& という記号を名前のどこかに追加すると、その後の文字（この例では *n*）がこのコマンドのニーモニックキーとして表示されます。指定したテキストが、[メニュー内容] リストに反映されます。

## コマンド

メニューからコマンドを選択したときに実行されるツールとそのパスを指定します。参照ボタンを使用して選択することもできます。

## 引数

オプション: コマンドの引数を指定します。

## 初期ディレクトリ

ツールの初期作業ディレクトリを指定します。

## 出力ウィンドウにリダイレクト

〔ツール 出力〕メッセージウィンドウの〔ツール出力〕ページにコンソール出力をすべて送信するよう指定します。このオプションを指定して起動したツールは、キーボードなどによるユーザ入力を受け付けることはできません。

入力が必要なツールや、実行するコンソールに特別な条件があるツールは、このオプションを指定すると動作しません。

## コマンドラインのプロンプト

コマンドを〔ツール〕メニューから選択したときに、コマンドライン引数の指定を指示するプロンプトを表示します。

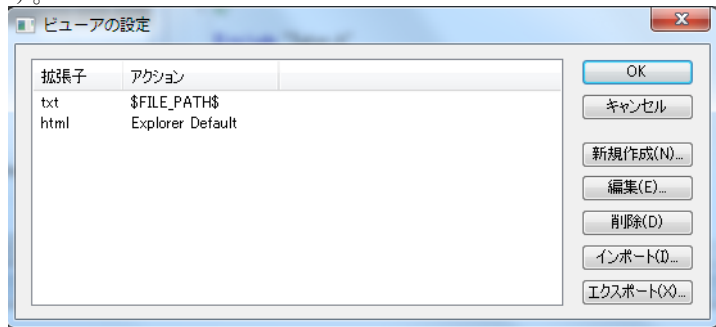
## ツール使用可能時

ツールが使用可能なコンテキストを指定します。以下から選択します。

- 常時
- デバッグ時
- デバッグ時以外

## 〔ビューアの設定〕ダイアログボックス

〔ビューアの設定〕ダイアログボックスは、〔ツール〕メニューから表示します。



このダイアログボックスには、IAR Embedded Workbench が処理できる文書フォーマットとビューアアプリケーション間のデフォルトの関連に対するオーバーライドが一覧表示されます。

## 表示エリア

このエリアには以下の列が含まれます。

### 拡張子

IAR Embedded Workbench が処理できる、明示的に定義された文書フォーマットのファイル名拡張子。

### アクション

文書タイプを開くときに使用されるビューアアプリケーション。[デフォルトエクスプローラ] は、Windows Explorer の指定タイプに関連するデフォルトのアプリケーションが使用されるということです。

## 新規作成

[ビューア拡張子の編集] ダイアログボックスが表示されます (87 ページの [ビューア拡張子の編集] ダイアログボックスを参照)。

## 編集

[ビューア拡張子の編集] ダイアログボックスが表示されます (87 ページの [ビューア拡張子の編集] ダイアログボックスを参照)。

## 削除

選択したファイル名拡張子とビューアアプリケーション間の関連を削除します。

## インポート

XML 形式の File Viewer Association ファイルを配置してインポートできるファイルブラウザを開きます。このファイルにはドキュメント形式と viewer アプリケーション間の関係が含まれています。製品インストールの arm¥src¥ide¥ディレクトリにあるサンプルファイル

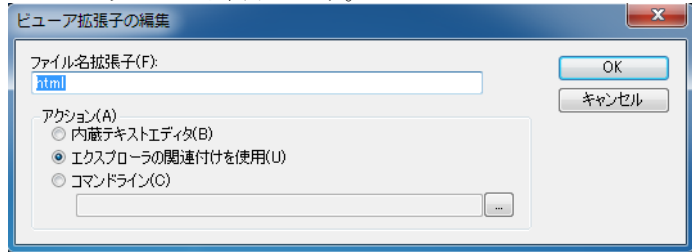
FileViewerAssociationsExample.xml で、このファイルの XML 構造は説明されています。

## エクスポート

標準の [名前を付けて保存] ダイアログボックスを表示して、[ビューアの設定] ダイアログボックスのドキュメント形式とビューアアプリケーション間の現在の関係を XML フォーマットのファイルに保存できます。

## 【ビューア拡張子の編集】 ダイアログボックス

【ファイル名拡張子の編集】ダイアログボックスは、【ビューアの設定】ダイアログボックスから表示します。



このダイアログボックスを使用して、新規文書タイプを開いたり、既存の文書タイプの設定を編集する方法を指定します。

### ファイル名拡張子

区切り文字のピリオド (.) も含めて、文書タイプのファイル名拡張子を指定します。

### アクション

【ファイル名の拡張子】テキストボックスで指定したファイル名拡張子を持つ文書を開く方法を選択します。以下から選択します。

#### 内蔵テキストエディタ

指定したタイプのすべての文書を IAR Embedded Workbench のテキストエディタで開きます。

#### エクスプローラの関連付けを使用

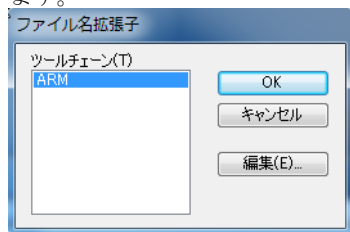
指定したタイプのすべての文書を、Windows Explorer の指定タイプに関連付けられたデフォルトのアプリケーションで開きます。

#### コマンドライン

指定したタイプのすべての文書を、入力または指定したビューアアプリケーションで開きます。ツールに使用するコマンドラインオプションを提供できます。例えば、ビューアアプリケーションのパスの後に \$FILE\_PATH\$ を入力し、アクティブなファイル（エディタ、プロジェクト、メッセージウィンドウなど）でビューアを開始します。

## 【ファイル名拡張子】ダイアログボックス

【ファイル名拡張子】ダイアログボックスは、【ツール】メニューから表示します。



このダイアログボックスを使用して、ビルドツールで認識されるファイル名拡張子をカスタマイズします。これは、ファイル名の拡張子が異なるソースファイルが多数ある場合に便利です。

### ツールチェーン

ホストコンピュータ上に IAR Embedded Workbench がインストールされたツールチェーンを一覧表示します。ファイル名拡張子をカスタマイズするツールチェーンを選択します。

\* 文字は、ユーザ定義のオーバーライドを示します。\* 文字がない場合は、出荷時設定が使用されます。

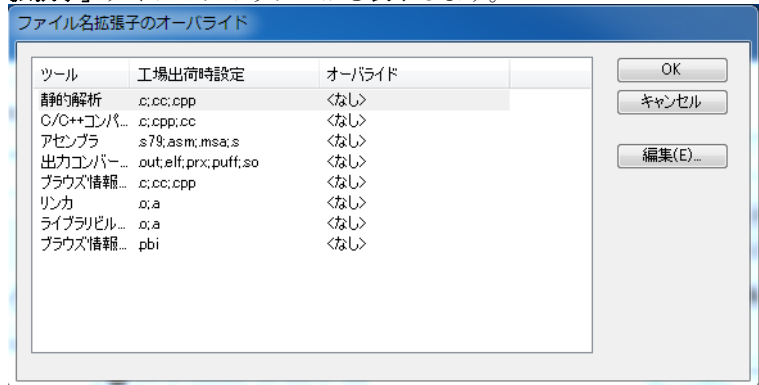
### 編集

【ファイル名拡張子のオーバーライド】ダイアログボックスを表示します (89 ページの 【ファイル名拡張子のオーバーライド】ダイアログボックスを参照)。



## 【ファイル名拡張子のオーバーライド】ダイアログボックス

【ファイル名拡張子のオーバーライド】ダイアログボックスは、【ファイル名の拡張子】ダイアログボックスから表示します。



このダイアログボックスには、ビルドツールで認識されるファイル名拡張子が一覧表示されます。

### 表示エリア

このエリアには以下の列が含まれます。

#### ツール

ビルドチェーンで使用可能なツール。

#### 工場出荷時設定

ビルドツールによりデフォルトで認識されるファイル名拡張子。

#### オーバーライド

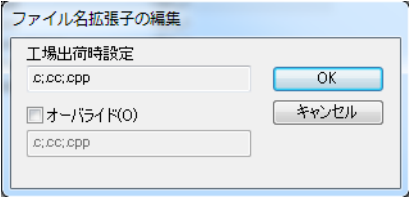
デフォルトへのオーバーライドがあった場合に、ビルドツールにより認識されるファイル名拡張子。

### 編集

選択したツールについて【ファイル名拡張子の編集】ダイアログボックスが表示されます。

## 【ファイル名拡張子の編集】ダイアログボックス

【ファイル名拡張子の編集】ダイアログボックスは、【ファイル名拡張子のオーバーライド】ダイアログボックスから表示します。



このダイアログボックスには、IDE で認識されるファイル名拡張子が一覧表示され、新しいファイル名拡張子を追加することができます。

### 工場出荷時設定

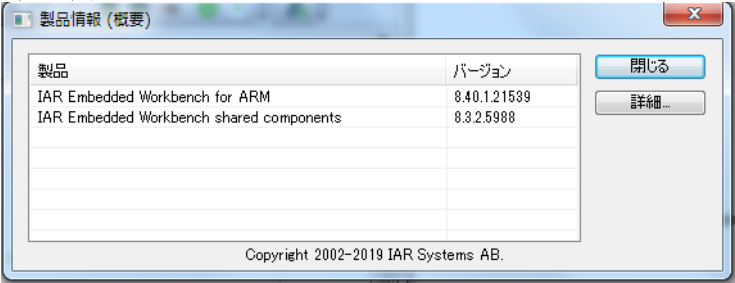
デフォルトで認識されるファイル名拡張子を一覧表示します。

### オーバーライド

認識させたいファイル名拡張子を指定します。拡張子が複数の場合はコンマかセミコロンで区切ります。また、最初のピリオドも含めて入力する必要があります。

## 製品情報ダイアログボックス

【製品情報】ダイアログボックスは【ヘルプ】メニューから使用できます。



このダイアログボックスには、お使いの IAR Embedded Workbench 製品のインストールおよび共有コンポーネントのバージョン番号がリストされます。

**注：**共有コンポーネントのバージョン番号の最初のけた（このスクリーンショットでは 8）は、デフォルトのインストールディレクトリ `x:\Program Files\IAR Systems\Embedded Workbench 8.n\` になります。

## 詳細

インストールした製品の様々なコンポーネントのバージョン番号がリストされるダイアログボックスを開きます。

## 引数変数

引数変数は、パスや引数に使用することができます。たとえば、**[オプション]** ダイアログボックスでインクルードパスを指定するときや、マクロのような現在のコンテキストに依存する拡張が必要な場合（引数におけるツールへの拡張など）です。広範な定義済みの引数変数を使用できるほか、独自に作成することも可能です（93 ページの **[カスタムの引数変数の設定]** ダイアログボックスを参照）。定義済みの引数変数は以下のとおりです。

変数	説明
\$COMPILER_ARGS\$	コンパイラを使用してコンパイルする際に使用されるファイル名以外のすべてのコンパイラオプション。この引数変数は、 <b>[外部アナライザ]</b> ダイアログボックスの <b>[引数]</b> テキストボックスに限られます
\$CONFIG_NAME\$	現在のビルド構成の名前（Debug、Release など）
\$CUR_DIR\$	現在のディレクトリ
\$CUR_LINE\$	現在の行
\$DATE\$	今日の日付、現在の場所による形式。これにより、ファイルパスでの使用に適していない変数を作成することがあります
\$EW_DIR\$	IAR Embedded Workbench のトップディレクトリ（例、c:\Program Files\IAR Systems\Embedded Workbench N.n）
\$EXE_DIR\$	実行可能ファイル出力用ディレクトリ
\$FILE_BNAME\$	ファイル名（拡張子を除く）
\$FILE_BPATH\$	フルパス（拡張子を除く）
\$FILE_DIR\$	アクティブなファイルのディレクトリ（ファイル名を除く）
\$FILE_FNAME\$	アクティブなファイルのファイル名（パスを除く）
\$FILE_PATH\$	アクティブなファイルのフルパス（エディタ、プロジェクト、メッセージウィンドウ）
\$LIST_DIR\$	リスト出力用ディレクトリ
\$OBJ_DIR\$	オブジェクト出力用ディレクトリ
\$PROJ_DIR\$	プロジェクトディレクトリ
\$PROJ_FNAME\$	プロジェクトファイル名（パスなし）

表 3: 引数変数

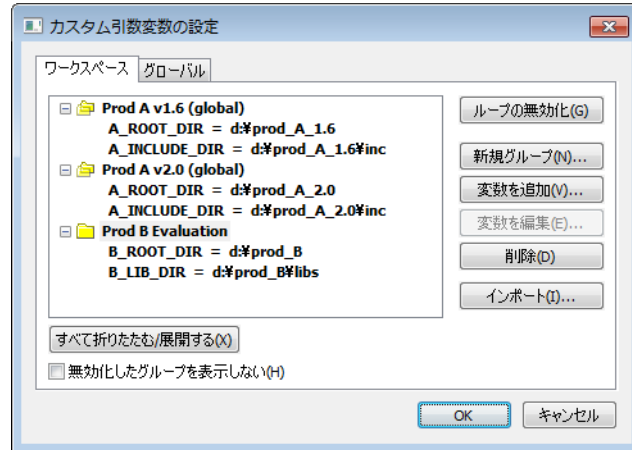
変数	説明
\$PROJ_PATH\$	プロジェクトファイルのフルパス
\$TARGET_DIR\$	主要出力ファイル用ディレクトリ
\$TARGET_BNAME\$	主要出力ファイルのファイル名（パス、拡張子を除く）
\$TARGET_BPATH\$	主要出力ファイルのフルパス（拡張子を除く）
\$TARGET_FNAME\$	主要出力ファイルのファイル名（パスを除く）
\$TARGET_PATH\$	主要出力ファイルのフルパス
\$TOOLKIT_DIR\$	アクティブな製品のディレクトリ（例 c:\Program Files\IAR Systems\Embedded Workbench N.n\arm
\$USER_NAME\$	ホストログイン名
\$WS_DIR\$	アクティブなワークスペースのディレクトリ (iarbuild.exe や cspybat.exe を使用しているときは IDE でのみ使用可能)
\$_ENVVAR_\$	Windows の環境変数 ENVVAR。\$_ と _\$ に囲まれた名前はす べて、そのシステム環境変数に展開されます
\$MY_CUSTOM_VAR\$	独自の引数変数については、93 ページの [カスタムの引数変 数の設定] ダイアログボックスを参照してください。\$ およ び \$ 内の名前はすべて、定義した値まで展開されます

表 3: 引数変数 (続き)

引数変数は、[IDE オプション] ダイアログボックスの一部のページでも使用  
できます (212 ページの [ツール] メニューを参照)。

## 〔カスタムの引数変数の設定〕 ダイアログボックス

〔カスタムの引数変数の設定〕 ダイアログボックスは [ツール] メニューから使用できます。



このダイアログボックスを使用して、カスタムの引数変数を定義および編集します。これは通常、サードパーティ製品をインストールし、引数変数を使用してそのインクルードディレクトリを指定する場合などに役立ちます。カスタムの引数変数を使用して、プロジェクトに含めるファイルへの参照を簡略化することも可能です。

カスタムの引数変数は、2つの異なる範囲のいずれかを持ちます。

- **ワークスペースローカルな変数。**これは特定のワークスペースに関連付けられており、変数が作成されたときにロードされたワークスペースからでないと見ることができません。
- **グローバル変数。**これはすべてのワークスペースで使用可能です。

変数を指定グループ内で編成することができます。

### ワークスペースとグローバルタブ

希望する変数の範囲を持ったタブをクリックします。

#### ワークスペース

- グローバルとワークスペースローカルの変数は、どちらも表示エリアで見ることができます。
- ワークスペースローカル変数のみ編集や削除が可能です。
- 変数のグループおよび個々の変数を、ローカルのレベルで追加またはインポートすることができます。

- ワークスペースのローカル変数は特定のディレクトリにある `Workspace.custom_argvars` ファイルに格納されます (191 ページの *ローカル設定のファイル* 参照)。

### グローバル

- グローバルとして定義された変数のみが表示エリアに表示されます。すべての変数を編集または削除できます。
- 変数のグループおよび個々の変数を、グローバルレベルで追加またはインポートすることができます。
- グローバル変数は特定のディレクトリにある `global.custom_argvars` ファイルに格納されます (190 ページの *グローバル設定のファイル* 参照)。



ビルドツールの設定でカスタムの引数変数に依存する場合、プロジェクトがビルドするために必要な情報の一部が `.custom_argvars` ファイルに含まれている可能性がある点に注意してください。このため、カスタムの引数ファイル (ローカルとグローバルのワークスペース) のバージョンを管理して、これらの変数の必要性を文書化すべきかどうかを検討してください。

### 展開 / すべて折りたたむ

変数のビューを展開または折りたたみます。

### 無効化されたグループを隠す

以前に無効にしたすべての変数グループを非表示にします。

### グループの有効化 / グループの無効化

選択した変数グループを有効化または無効化します。開いているタブに応じて結果は異なります。

- **[ワークスペース]** タブ: グループを有効化または無効化すると、現在のワークスペースのみに反映されます。
- **[グローバル]** タブ: 有効化すると、新規作成したワークスペースのみに影響します。これらは、現在のグローバルの状態をワークスペースのデフォルトとして継承します。

**注:** 無効化されたグループの一部となっている変数は使用できません。

### 新規グループ

**[新規グループ]** ダイアログボックスが開き、新しいグループ名を指定できます。[OK] をクリックすると、グループが作成されてカスタム引数変数の一覧に表示されます。

## 変数の追加

**[変数の追加]** ダイアログボックスが開き、選択したグループに対して新しい変数名と値を指定することができます。**[OK]** をクリックすると、変数が作成されてカスタム引数変数の一覧に表示されます。

また、以前に定義した変数をインポートしても変数を追加できます。以下の**[インポート]** を参照してください。

## 変数の編集

**[変数の編集]** ダイアログボックスが開き、選択した変数の名称と値を編集することができます。**[OK]** をクリックすると、変数が作成されてカスタム引数変数の一覧に表示されます。

## 削除

選択したグループまたは変数を削除します。

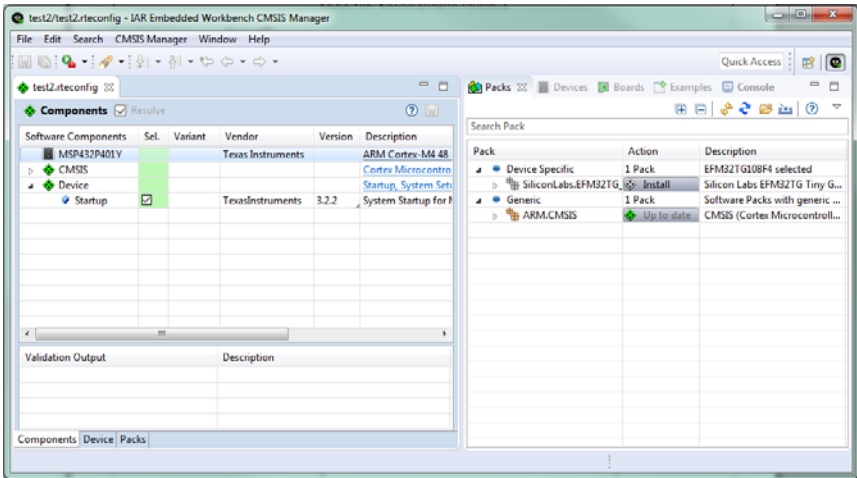
## インポート

ワークスペース `.custom_argvars` ファイルを配置できるファイルブラウザを開きます。このファイルはサードパーティ製品をインストールしたときに作成されたファイルか、または別のワークスペースに関連する定義済みの変数を含んでいます。

## CMSIS Manager ダイアログボックス



CMSIS Manager ダイアログボックスは、[プロジェクト] > [CMSIS-Manager] を選択するか、CMSIS-Manager ツールバーボタンをクリックして利用できます。



このダイアログボックスを使用して、CMSIS ソフトウェアパックとサンプルプロジェクトを管理します。

詳細については、以下を参照してください。

- 107 ページの CMSIS-Pack ソフトウェアパックのインストール
- 108 ページの IAR Embedded Workbench でサポートする CMSIS-Pack の使用
- 特に手順 27 ページのサンプルプロジェクトの使用CMSIS-Pack サンプルプロジェクトを使用するには



このダイアログボックスで利用可能なビュー、ボタン、およびメニューコマンドについては、F1 を押すか、クエスチョンマークアイコンをクリックしてオンラインヘルプを表示します。オンラインヘルプシステムは、コンテキストに依存します。つまり、選択しているビューによって、異なるヘルプトピックが表示されます。



# プロジェクト管理

- プロジェクト管理の概要
- プロジェクト管理
- プロジェクト管理のリファレンス情報

---

## プロジェクト管理の概要

以下のトピックを解説します：

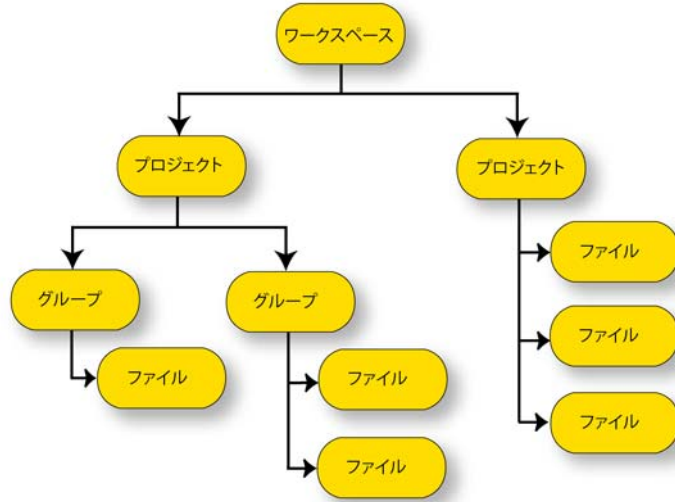
- プロジェクト管理の概要について
- プロジェクトの作成方法
- バージョン管理システムの IDE 操作

### プロジェクト管理の概要について

数百ものファイルを扱う大規模な開発プロジェクトでは、簡単にアクセスでき、数人のエンジニアによる保守が可能な構造に、ファイルを編成する必要があります。

IDE は、C/C++ ソースコードファイル、アセンブラファイル、インクルードファイル、その他の関連モジュールなど、すべてのプロジェクトモジュールを管理するための機能を装備しています。ワークスペースを作成し、1 つまたは複数のプロジェクトを追加できます。ファイルはファイルグループごと

に編成が可能で、プロジェクト、グループ、ファイルのすべてのレベルでオプションを設定できます。



リビルド実行時に必要なモジュールが再変換されるように、変更が記録されます。そのため、古いモジュールが含まれる実行可能ファイルが作成されることがありません。

以下は IDE の追加機能です。

- スムーズに開発が開始できるように、ビルドおよび実行がすぐに可能なプロジェクトテンプレートが付属
- プロジェクトを階層構造で表示
- 階層構造でシンボルを表示できるソースブラウザ
- オプションを全体、ソースファイルのグループ単位、個々のソースファイル単位に設定可能
- [メイク] コマンドでは自動的に変更を検出して、必要な操作のみを実行します
- IAR Embedded Workbench と外部のツール間の接続を設定するプロジェクト接続
- テキストベースのプロジェクトファイル
- カスタムビルドユーティリティにより、標準ツールチェーンを簡単に拡張可能
- プロジェクトファイルを入力としてコマンドラインビルド可能

## プロジェクトファイルのナビゲート

プロジェクトファイルをナビゲートするには、[ワークスペース] ウィンドウまたは[ソースブラウザ] ウィンドウを使用する、主に2種類の方法があります。[ワークスペース] ウィンドウには、論理的にグループ化されたソースファイル、依存ファイル、出力ファイルが、階層構造で表示されます。一方、[ソースブラウザ] ウィンドウには、現在[ワークスペース] ウィンドウでアクティブなビルド構成の情報が表示されます。ビルド構成については、変数、関数、型定義など、グローバルに定義されているすべてのシンボルが階層的に[ソースブラウザ]に表示されます。クラスについては、基底クラスの情報も表示されます。

ソースのブラウザの詳細については、140 ページの *ソースブラウザ情報の概要*についてを参照してください。

## プロジェクトの作成方法

IDE を使用すると、論理構造が一目でわかるような階層ツリー構造にプロジェクトを編成できます。

IDE は、ソフトウェア開発プロジェクトで通常行われる作成方法に合わせて設計されています。たとえば、バージョンの異なるターゲットハードウェアに対応して関連するバージョンのアプリケーションを開発する、初期のバージョンにはデバッグルーチンを組み込み最終アプリケーションには組み込まないようにする、などの編成が考えられます。

異なるターゲットハードウェアに応じ、複数バージョンのアプリケーションを開発する場合でも、ソースファイルは共通であることが多いので、それらのファイルのコピーを1つだけ保持するようにして、修正が自動的にアプリケーションの各バージョンに反映されるように作成することができます。また、ハードウェア依存部分を処理アプリケーションのように、複数のバージョンでソースファイルが異なる場合もあります。

以降のセクションでは、階層のさまざまなレベルについて説明します。

## プロジェクトとワークスペース

通常は、1つまたは複数のプロジェクトを作成し、それぞれが次のいずれかを含むようにします。

- ソースコードファイル。組み込みアプリケーションまたはライブラリを生成するときに使用できます。ライブラリプロジェクトとアプリケーションプロジェクトを組み合わせる例については、チュートリアル「ライブラリの作成と利用」の例を参照してください。
- C-SPY でロードする外部でビルドされた実行可能ファイル。IDE の外でビルドされた実行可能ファイルをロードする方法については、『*ARM 用 C-SPY® デバッグガイド*』を参照してください。

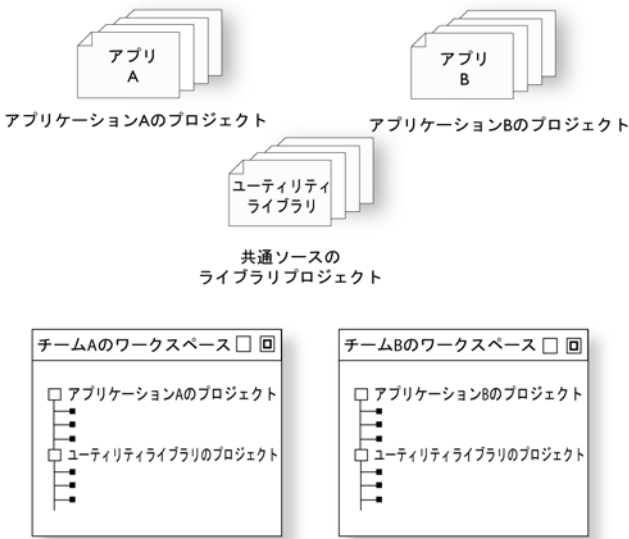
関連プロジェクトが複数存在する場合は、それらに同時にアクセスして操作できます。そのために、関連するプロジェクトをワークスペースに編成する機能があります。

1つのワークスペースには、1つ以上のプロジェクトを追加できます。どのプロジェクトも、少なくとも1つのワークスペースに属している必要があります。

1つ例を示します。2つの関連するアプリケーション、たとえばAとBを開発します。開発チームAはアプリAを、開発チームBはアプリBを開発します。2つのアプリケーションは関連性があるので、ソースコードの一部は両アプリケーション間で共有できます。この場合、以下のプロジェクトモデルを適用できます。

- 3つのプロジェクト各アプリケーション用に1つずつのプロジェクト、共通ソースコード用にもう1つのプロジェクト
- 2つのワークスペースチームAのワークスペースとチームBのワークスペース

共通のソースをライブラリプロジェクト（コンパイル済みだがリンクはされていないオブジェクトコード）にまとめる方法は、不要なコンパイルを避けることができるので、使いやすさと効率の両面で優れています。次の図に例を示します。



## プロジェクトとビルド構成

いくつか異なるバージョンのプロジェクトをビルドしなければならないことがよくあります。たとえば、さまざまなリンカおよびデバッガの設定を必要とする、異なるデバッグソリューションの場合などです。別の例として、実行のトレース用に特別なデバッグ出力を持つ、別にビルドされた実行可能ファイルが必要な場合などがあります。IAR Embedded Workbench では、プロジェクトごとに複数のビルド構成を定義できます。たとえば、デバッグとリリースの2つだけを必要とする単純なケースがあります。この2つのビルド構成は、最適化、デバッグ情報、出力形式に使用するオプションだけが異なります。リリース構成では、プリプロセッサシンボル NDEBUG が定義され、アプリケーションにはアサートが含まれません。

ビルド構成を追加すると、複数のターゲットデバイス上でアプリケーションを使用する場合などに便利です。つまり、アプリケーションは同じで、コードのハードウェア関連の部分が異なる場合です。したがって、ビルドするターゲットデバイスに応じて、ビルド構成からいくつかのソースファイルを除外できます。プロジェクト A では、以下のビルド構成によって要件が満たされます。

- プロジェクト A — デバイス 1: リリース
- プロジェクト A — デバイス 1: デバッグ
- プロジェクト A — デバイス 2: リリース
- プロジェクト A — デバイス 2: デバッグ

## グループ

通常は、プロジェクトには論理的に関連する数百のファイルが含まれます。そこで、関連するソースファイルをまとめてグループを定義して、そのような複数のグループを各プロジェクトに定義することができます。また、複数レベルのサブグループを定義して、論理階層を表現することもできます。デフォルトでは、グループはすべてのビルド構成に存在しますが、グループを特定のビルド構成から排除するように指定することもできます。

## ソースファイルとそのパス

ソースファイルは、プロジェクトノードかグループ階層の直下に置くことができます。プロジェクトのファイルの数が多く、扱いにくい場合は、グループ階層を使用すると便利です。デフォルトでは、各ファイルはプロジェクトのすべてのビルド構成に存在しますが、ファイルを特定のビルド構成から排除するように指定することもできます。

実際にビルドされ、出力コードにリンクされるのは、ビルド構成に含まれるファイルだけです。

プロジェクトを正常にビルドすると、ソースファイルの下に、そこにインクルードされているファイルとそこから生成された出力ファイルが構造化されて表示されます。

**注：**ビルド構成の設定によって、ソースファイルのコンパイル時に使用するインクルードファイルを選択できます。これは、コンパイル後にソースファイルに関連付けられているインクルードファイルのセットは、ビルド構成によって異なる場合があることを意味します。

IDE は、ある程度までのソースファイルの相対パスをサポートします。

- プロジェクトファイル

プロジェクトファイルのファイル部分のパスは、同じドライブにある場合は相対パスです。パスは、\$PROJ\_DIR\$ または \$EW\_DIR\$ のいずれかに対して相対的です。引数変数 \$EW\_DIR\$ が使用されるのは、パスが \$EW\_DIR\$ へのサブディレクトリにあるファイルを参照し、\$EW\_DIR\$ からの距離が \$PROJ\_DIR\$ からの距離より短い場合だけです。

プロジェクトファイルの一部であるファイルのパスは、ファイルが異なるドライブにある場合は絶対パスです。

- ワークスペースファイル

ワークスペースファイルと同じドライブにあるファイルについては、パスは \$PROJ\_DIR\$ に対して相対的です。

ワークスペースファイルと異なるドライブにあるファイルについては、パスは絶対パスです。

- デバッグファイル

デバッグイメージファイルにデバッグ情報が含まれる場合、ソースファイルを参照するファイル内のすべてのパスは絶対パスです。

## ドラッグアンドドロップ

Windows Explorer から、個々のソースファイルやプロジェクトファイルを [ワークスペース] ウィンドウにドラッグすることができます。ソースファイルをグループにドロップすると、そのグループに追加されます。プロジェクトツリーの外 ([ワークスペース] ウィンドウの背景) にドロップされたソースファイルは、アクティブプロジェクトに追加されます。

## バージョン管理システムの IDE 操作

IAR Embedded Workbench IDE は Subversion (SVN) 作業用コピーにあるどのファイルも識別してアクセスできます (106 ページの *Subversion の操作* を参照)。

IDE 内から IAR Embedded Workbench プロジェクトを外部 SVN プロジェクトに接続すると、通常使用する操作を一部実行することができます。

IAR Embedded Workbench プロジェクトをバージョン管理システムに接続する場合、使用しているバージョン管理のクライアントアプリケーションの操作に慣れている必要があります。

**注：**IDE からバージョン管理システムを操作する際に表示される一部のウィンドウやダイアログボックスは、そのバージョン管理システムによって表示されたものであり、IAR システムズが提供するドキュメントでは説明されていません。SCC システムのクライアントアプリケーションの詳細については、そのアプリケーションに付属するドキュメントを参照してください。

**注：**異なるバージョン管理システムでは、最も基本的な概念に関する部分であっても、使用されている用語が異なります。IDE とバージョン管理システム間で操作を行う場合の説明を読む際は、この点に注意する必要があります。

---

## プロジェクト管理

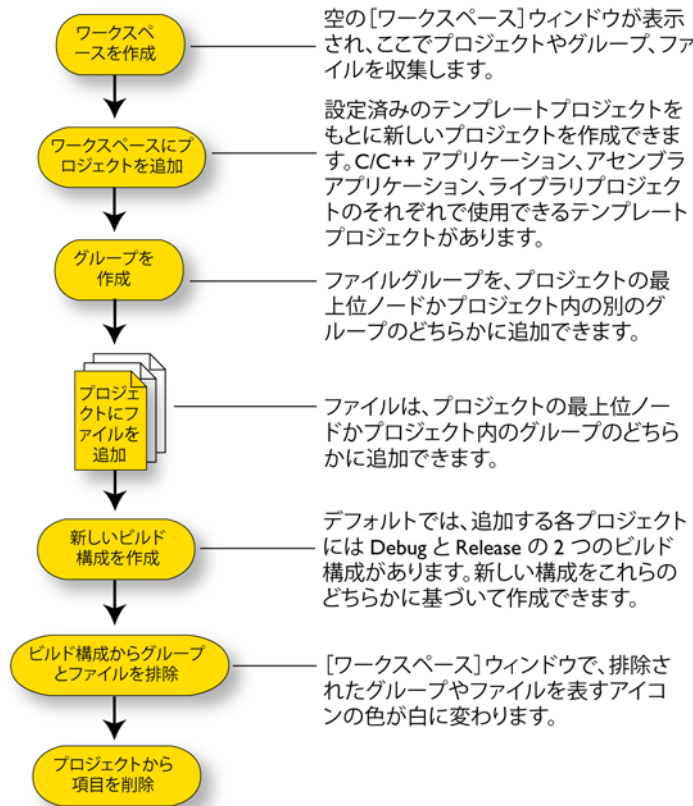
以下のタスクについて解説します。

- ワークスペースやそのプロジェクトの作成及び管理
- ワークスペースおよびそのプロジェクトの表示
- Subversion の操作
- CMSIS-Pack ソフトウェアパックのインストール
- IAR Embedded Workbench でサポートする CMSIS-Pack の使用

### ワークスペースやそのプロジェクトの作成及び管理

ここでは、ワークスペース、プロジェクト、グループ、ファイル、ビルド構成を作成する手順の概要について説明します。手順ごとの詳しい例については、チュートリアル「*アプリケーションプロジェクトの作成*」を参照してください。

ワークスペースの作成および管理の手順とその内容を次に示します。



**注:** 新しいビルド構成には、同じプロジェクト内の他のビルド構成と同じツールチェーンを使う必要はありません。また、これらの手順をすべてこの順序で実行しなくてもよい場合があります。

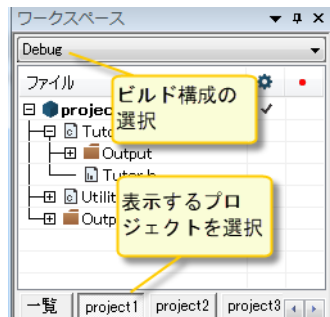
[ファイル] メニューには、ワークスペースを作成するコマンドがあります。  
[プロジェクト] メニューには、プロジェクトの作成、プロジェクトへのファイルの追加、グループの作成、プロジェクトオプションの指定、現在のプロジェクトに対する IAR システムズ製開発ツールの実行を行うコマンドがあります。



## ワークスペースおよびそのプロジェクトの表示

【ワークスペース】ウィンドウは、アプリケーションの開発中にプロジェクトやファイルにアクセスするインタフェースです。

- 1 【ワークスペース】ウィンドウの下端にあるタブをクリックすると、そのプロジェクトが表示されます。



ビルドされたファイルごとに出力フォルダアイコンが表示されます。このフォルダには、オブジェクトファイルやリストファイルなどの生成されたファイルがあります。リストファイルは、リストファイルオプションが有効な場合にのみ生成されます。プロジェクトノードに関連付けられた出力フォルダには、実行可能ファイルやリンカマップファイル（リストファイルオプションが有効な場合）など、プロジェクト全体に関連して生成されたファイルが含まれます。

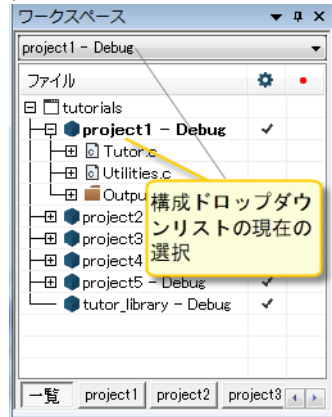
また、インクルードされているヘッダファイルも表示され、依存関係をわかりやすく示しています。

- 2 別のビルド構成のプロジェクトを表示するには、【ワークスペース】ウィンドウ上端のドロップダウンリストからそのビルド構成を選択します。

選択したプロジェクトとビルド構成は、【ワークスペース】ウィンドウで強調表示されます。アプリケーションをビルドすると、ドロップダウンリストで選択したこのプロジェクトとビルド構成がビルドされます。

- 3 ワークスペースですべてのプロジェクトの概要を表示するには、【ワークスペース】ウィンドウの下端にある【一覧】タブをクリックします。

すべてのプロジェクトメンバの概要が表示されます。



〔ビルド構成〕ドロップダウンリストで現在選択されている項目は、ワークスペースの概要が表示されるときの強調表示されます。

## SUBVERSIONの 操作

IAR Embedded Workbench のバージョン管理統合では、クライアントアプリケーション `svn.exe` と `TortoiseProc.exe` を使用して、最も一般的なサブバージョン操作のいくつかを IDE から直接実行できます。

**IAR Embedded Workbench プロジェクトを Subversion システムに接続するには、以下を行います。**

- 1 Subversion クライアントアプリケーションで、Subversion の作業用コピーを設定します。
- 2 IDE で、アプリケーションプロジェクトを Subversion の作業用コピーに接続します。

**Subversion の作業用コピーを設定するには：**

- 1 IDEでSubversionの統合を使用するには、`svn.exe` と `TortoiseProc.exe` がパスにあることを確認します。
- 2 Subversion リポジトリから作業用コピーをチェックアウトします。

プロジェクトを構成するファイルは、同じ作業用コピーからでなくても問題ありません。プロジェクトのすべてのファイルは個別に処理されます。ただし、`TortoiseProc.exe` では、異なるリポジトリからのファイルを同時にチェックインすることはできません。

アプリケーションプロジェクトを Subversion の作業用コピーに接続するには：

- 1 [ワークスペース] ウィンドウで、Subversion の作業用コピーを作成したプロジェクトを選択します。
- 2 [プロジェクト] メニューから、[バージョン管理システム] > [プロジェクトを Subversion に接続] を選択します。このコマンドは、[ワークスペース] ウィンドウを右クリックして表示されるコンテキストメニューでも選択できます。

Subversion の作業用コピーにアクセスするコマンドの詳しい情報については、120 ページの *Subversion* のバージョン管理システムメニューを参照してください。

### Subversion の状態の表示

IAR Embedded Workbench プロジェクトが Subversion の作業用コピーに接続されると、バージョン管理のステータス情報を表す列が [ワークスペース] ウィンドウに表示されます。さまざまなアイコンが表示され、それぞれの Subversion の状態を表します (121 ページの *Subversion* の状態を参照)。

### CMSIS-PACK ソフトウェアパックのインストール

CMSIS-Pack はソフトウェアコンポーネント、デバイスのサポート、評価ボードのサポート、プロジェクト例を提供します。

手順 *IAR Embedded Workbench* でサポートする *CMSIS-Pack* の使用を行うには、まず必要な CMSIS-Pack ソフトウェアパックをインストールすることが推奨されています。

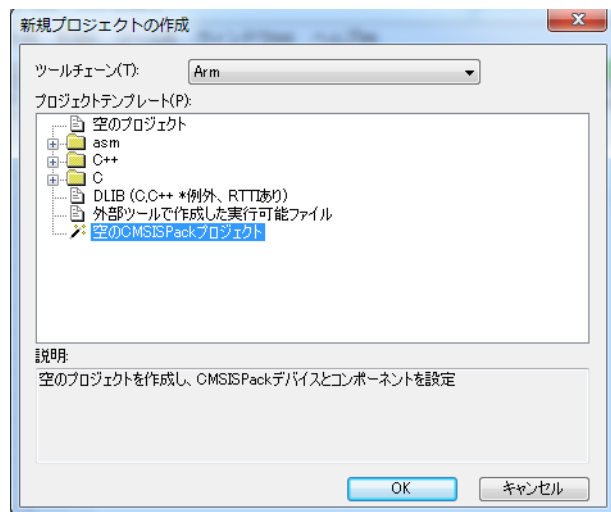
#### CMSIS-Pack ソフトウェアパックのインストール：

- 1 IAR Embedded Workbench プロジェクトで、[プロジェクト] > [CMSIS-Manager] を選択します。
- 2 [デバイス] タブをクリックし、ツリー構造でデバイスに移動し選択します。
- 3 [パック] タブをクリックし、インストールするソフトウェアパックを選択します。
- 4 [インストール] ボタンをクリックし、インストールを開始します。

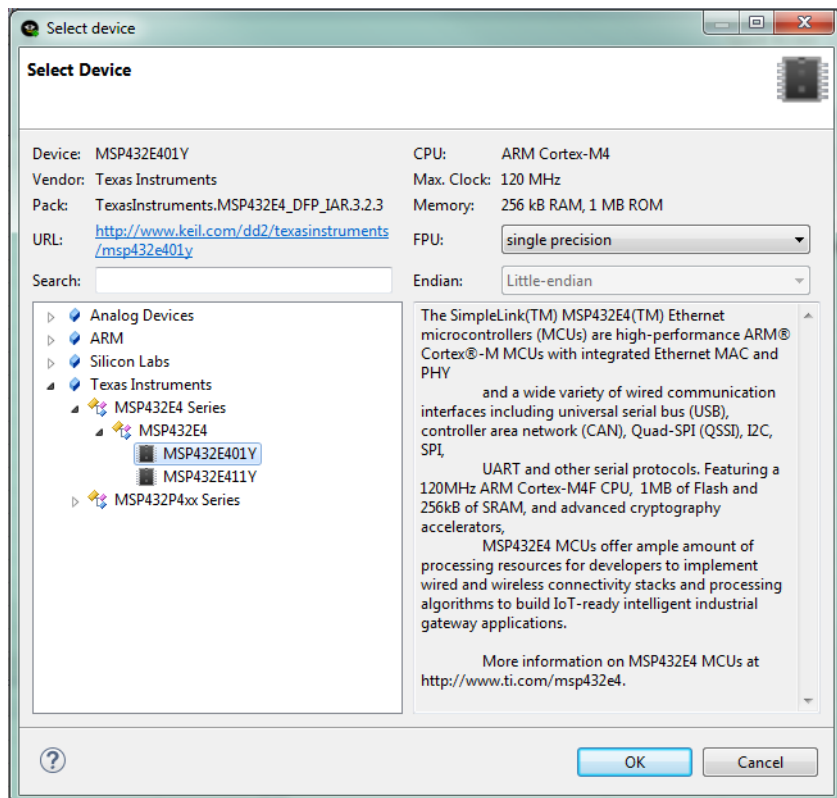
インストールプロセスを示すステータスメッセージをプリントする [コンソール] ビューの切り替えをフォーカスしますが完了。

## IAR EMBEDDED WORKBENCHで サポートする CMSIS-PACKの 使用

- 1 IAR Embedded Workbench ワークスペースで、[プロジェクト] > [新規プロジェクトの作成] を選択します。
- 2 [新規プロジェクトの作成] ダイアログボックスで [Empty CMSISPack プロジェクト] を選択して **OK** をクリックします。

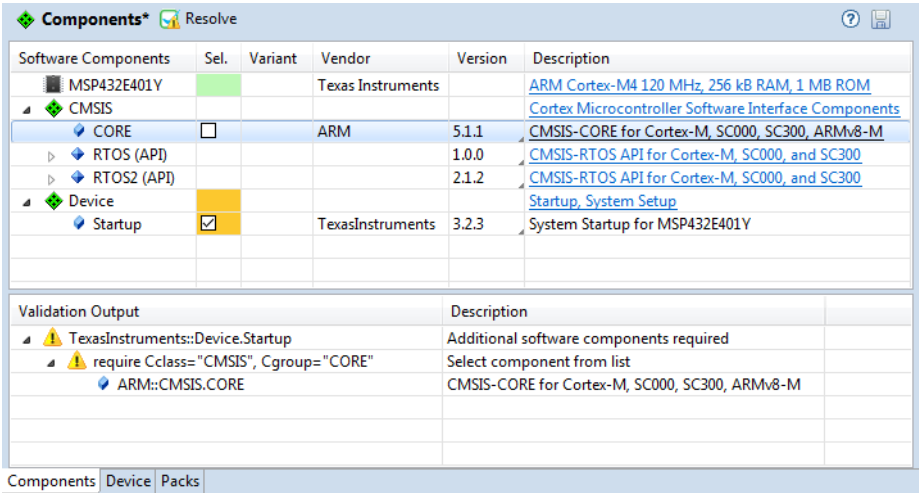


- 3 表示される [名前をつけて保存] ダイアログボックスでプロジェクトを保存します。プロジェクトは、使用しているワークスペースとは異なるディレクトリに保存されるので、気をつけてください。
- 4 表示される [デバイスの選択] ダイアログボックスで、デバイスを選択して **OK** をクリックします。



- 5 **CMSIS Manager** ダイアログボックスが表示されます。このダイアログボックスを使用して、ビルド済みの CMSIS-Pack ソフトウェアコンポーネントとデバイスで使用可能なプロジェクト例を選択します。詳細については、96 ページの *CMSIS Manager* ダイアログボックスを参照してください。27 ページの *サンプルプロジェクトの使用* を参照してください。

6 [コンポーネント] ビューで、必要なソフトウェアコンポーネントを選択します。



7 解決されていない依存関係はオレンジでハイライトされます。解決されていない依存関係を解決するには、それを選択しツールバーの[解決] ボタンをクリックします。

依存関係が赤でハイライトされている場合は、**CMSIS Manager** ダイアログボックスでは、その解像度は見つかりません。通常、足りないパックをインストールする必要があります。

8 デバイスを変更するには、[デバイス] ビューで、[変更] ボタンをクリックして、使用するデバイスを選択します。

9 選択したものを有効にするには、[ファイル] > [保存] を選択します。IAR Embedded Workbench プロジェクトは、**CMSIS Manager** ダイアログボックスで選択したものに関連するファイルに追加されます。

## プロジェクト管理のリファレンス情報

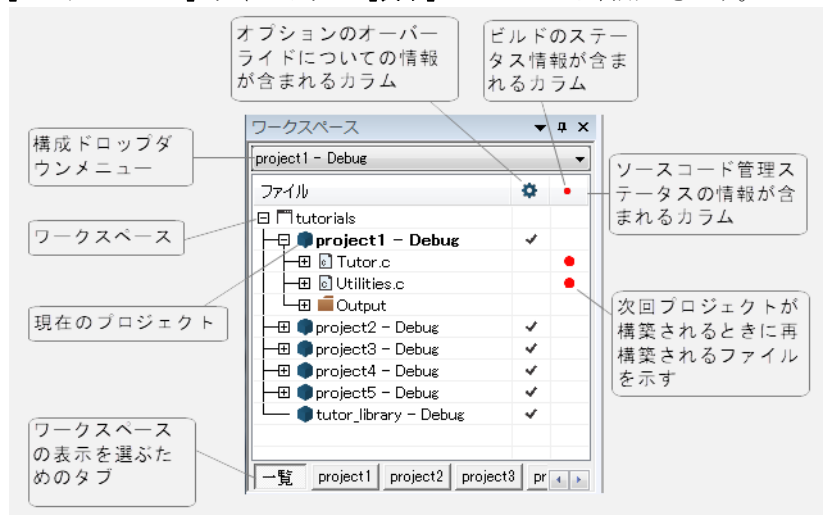
リファレンス情報:

- 111 ページの [ワークスペース] ウィンドウ
- 116 ページの [新規プロジェクトの作成] ダイアログボックス
- 117 ページの [プロジェクトの構成] ダイアログボックス

- 118 ページの [新規ビルド構成] ダイアログボックス
- 119 ページの [プロジェクトコネクションの追加] ダイアログボックス
- 120 ページの *Subversion* のバージョン管理システムメニュー
- 121 ページの *Subversion* の状態

## 【ワークスペース】ウィンドウ

【ワークスペース】ウィンドウは【表示】メニューから利用できます。



このウィンドウを使用して、アプリケーション開発中にプロジェクトやファイルにアクセスします。















### ドロップダウンリスト

ウィンドウ上部にあるドロップダウンリストでは、特定プロジェクト用のウィンドウで表示するビルド構成を選択できます。

## 表示エリア

このエリアには 4 つの列が含まれます。

[ファイル] 列には、現在のワークスペースの名前、ワークスペースに含まれるプロジェクト、グループ、ファイルのツリー表現が表示されます。以下から 1 つまたは複数のアイコンが表示されます。

	ワークスペース
	プロジェクト
	複数ファイルのコンパイルを伴うプロジェクト
	ファイルグループ
	ビルドから除外されたグループ
	ファイルグループ、複数ファイルコンパイルの一部
	ファイルグループ、複数ファイルコンパイルの一部だがビルドから除外される
	オブジェクトファイルまたはライブラリ
	アセンブラソースファイル
	C ソースファイル
	C++ ソースファイル
	ビルドから除外されたソースファイル
	ヘッダファイル
	テキストファイル
	HTML テキストファイル
	制御ファイル、たとえばリンカ設定ファイルなど
	IDE 内部ファイル
	その他のファイル





オプションのオーバーライドに関するステータス情報を示す列には、プロジェクトのレベルごとに 3 つのアイコンのいずれかが表示されます。

空白	このファイル / グループの設定 / オーバライドはありません。
黒のチェックマーク	このファイル / グループのローカル設定 / オーバライドがあります。
赤のチェックマーク	このファイル / グループのローカル設定 / オーバライドがありますが、これらは、継承した設定と同一であるか、複数ファイルのコンパイルが使用されるため無視されます。すなわち、オーバーライドは不要です。



ビルドステータス情報を示す列には、プロジェクトのファイルごとに 3 つのアイコンのいずれかが表示されます。

空白	次回のプロジェクトのビルド時、このファイルはリビルドされません。
赤色い点	次回のプロジェクトのビルド時、このファイルはリビルドされます。
小さい赤色の点	このファイルはリビルド中です。



この列にはバージョン管理のステータス情報が含まれます。さまざまなアイコンについては、121 ページの *Subversion* の状態を参照してください。

ウィンドウ下部のタブを使用して、表示するプロジェクトを選択できます。また、ワークスペース全体の概要を表示することもできます。

プロジェクト管理および【ワークスペース】ウィンドウの使用について詳しくは、97 ページのプロジェクト管理の概要を参照してください。

## コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

### オプション

【ワークスペース】ウィンドウで選択した項目に対して、各ビルドツールのオプションを設定できるダイアログボックスを表示します。ビルドからそれを除外する例。プロジェクト全体、ファイルのグループ、個々のファイルのオプションを設定できます。125 ページの *「オプション」ダイアログボックスを使用したプロジェクトオプションの設定* を参照してください。

### メイク

最後のビルド以降に変更されたファイルだけをコンパイル、アセンブル、リンクして、現在のターゲットを最新状態に更新します。

### コンパイル

選択されているファイルを必要に応じてコンパイル/アセンブルします。【ワークスペース】ウィンドウで選択するか、コンパイル対象ファイルが開かれたエディタウィンドウを選択することで、ファイルを選択できます。

### すべてを再ビルド

選択したビルド構成のすべてのファイルを再コンパイルし、再リンクします。

## クリーン

中間ファイルを削除します。

## C-STAT 静的解析 > プロジェクトの解析

C-STAT で選択したプロジェクトを解析します。C-STAT の詳細は、『C-STAT® Static Analysis Guide』を参照してください。

## C-STAT 静的解析 > ファイルの解析

C-STAT で選択したファイルを解析します。C-STAT の詳細は、『C-STAT® Static Analysis Guide』を参照してください。

## C-STAT 静的解析 > 分析結果をクリア

C-STAT で以前に実行した解析の情報を消去します。C-STAT の詳細は、『C-STAT® Static Analysis Guide』を参照してください。

## C-STAT 静的解析 > HTML サマリの生成

標準の [名前を付けて保存] ダイアログボックスでは、HTML のレポートの概要の保存先を選択し、それを作成できます。C-STAT の詳細は、『C-STAT® Static Analysis Guide』を参照してください。

## C-STAT 静的解析 > 詳細な HTML レポートの生成

標準の [名前を付けて保存] ダイアログボックスを表示し、HTML 形式の詳細なレポートの保存先を選択して、レポートを作成します。C-STAT の詳細は、『C-STAT® Static Analysis Guide』を参照してください。

## ビルドを停止

現在のビルド処理を停止します。

## 追加 > ファイルの追加

プロジェクトにファイルを追加するためのダイアログボックスを表示します。

## 追加 > ファイル名の追加

指定したファイルをプロジェクトに追加します。このコマンドは、エディタで開かれているファイルがある場合にだけ使用できます。

## 追加 > グループの追加

[グループの追加] ダイアログボックスを表示して、新規グループをプロジェクトに追加できます。グループの詳細は、101 ページの **グループを参照してください。**

## 削除

選択した項目を [ワークスペース] ウィンドウから削除します。

### 名前の変更

【グループの名称変更】ダイアログボックスが表示され、グループの名前を変更できます。グループの詳細は、101 ページの [グループを参照](#) してください。

### バージョン管理システム

ソースコード管理用コマンドのサブメニューを表示します (120 ページの *Subversion* のバージョン管理システムメニューを参照)。

### ファイルの場所を開く

選択したファイルが存在するディレクトリを表示するファイルエクスプローラを開きます。

### ファイルのプロパティ

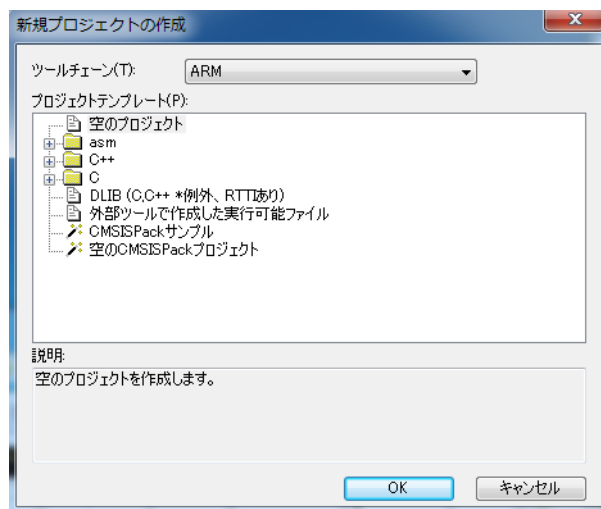
選択したファイルについて、標準の【ファイルプロパティ】ダイアログボックスを表示します。

### アクティブに設定

概要ウィンドウで選択したプロジェクトをアクティブプロジェクトに設定します。【メイク】コマンド実行時にビルドされるのがアクティブプロジェクトです。

## 【新規プロジェクトの作成】ダイアログボックス

【新規プロジェクトの作成】ダイアログボックスは、【プロジェクト】メニューから使用できます。



このダイアログボックスを使用して、テンプレートプロジェクトに基づいて新しいプロジェクトを作成します。テンプレートプロジェクトは、C/C++ アプリケーション、アセンブラアプリケーション、ライブラリプロジェクトに使用できます。また、自分でテンプレートプロジェクトを作成することもできます。

### ツールチェーン

ビルド対象ターゲットを選択します。異なるターゲット用に複数のバージョンの IAR Embedded Workbench がホストコンピュータにインストールされている場合は、ドロップダウンリストからターゲットのすべてまたは一部を選択できることがあります。

### プロジェクトテンプレート

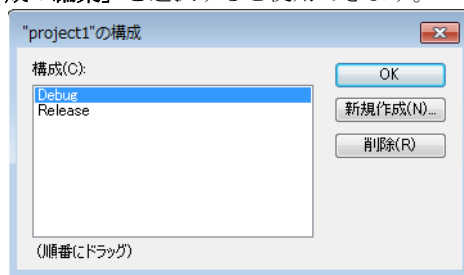
この使用可能なテンプレートプロジェクトのリストから、新規プロジェクトの基となるテンプレートを選択します。

### 説明

現在選択されているテンプレートの説明。

## [プロジェクトの構成] ダイアログボックス

[プロジェクトの構成] ダイアログボックスは、[プロジェクト] > [ビルド構成の編集] を選択すると使用できます。



このダイアログボックスを使用して、選択したプロジェクトに新しいビルド構成を定義します。完全に新規か、前のプロジェクトに基づくものかどちらかです。

### 構成

既存の構成を表示します。ここに表示された構成を、新しい構成のテンプレートとして使用できます。

## 新規作成

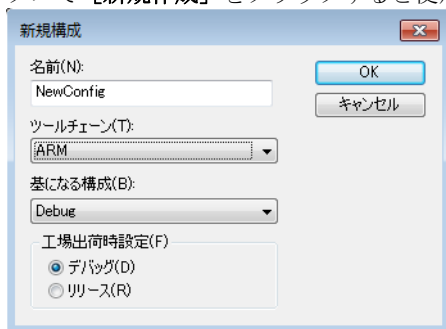
新しいビルド構成を定義するためのダイアログボックスを表示します (118 ページの [\[新規ビルド構成\] ダイアログボックス](#) を参照)。

## 削除

**[構成]** リストで選択した構成を削除します。

## [新規ビルド構成] ダイアログボックス

**[新規構成]** ダイアログボックスは、**[プロジェクトの構成]** ダイアログボックスで **[新規作成]** をクリックすると使用できます。



このダイアログボックスを使用して、新しいビルド構成を定義します。全く新しい構成か、現在定義された構成に基づくかどうかです。

## 名前

ビルド構成名を入力します。

## ツールチェーン

ビルド対象ターゲットを指定します。異なるターゲット用に複数のバージョンの IAR Embedded Workbench がホストコンピュータにインストールされている場合は、ドロップダウンリストからターゲットのすべてまたは一部を選択できることがあります。

## 基になる構成

新しい構成の基になる、現在定義済みのビルド構成を選択します。新しい構成は、プロジェクトの設定と出荷時設定情報を既存の構成から継承します。**[なし]** を選択すると、新しい構成は工場出荷時の設定だけに基づくようになります。

## 工場出荷時設定

新規のビルド構成に適用するデフォルトの工場出荷時設定を選択します。[オプション] ダイアログボックスの [工場出荷時設定] ボタンをクリックすると、ここで指定した出荷時設定がプロジェクトで使用されます。

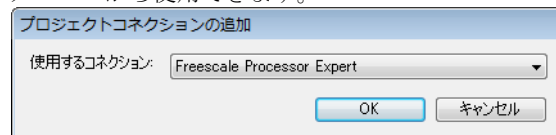
以下から選択します。

[デバッグ] のビルド構成に適した工場出荷時設定。

[リリース] のビルド構成に適した工場出荷時設定。

## [プロジェクトコネクションの追加] ダイアログボックス

[プロジェクトコネクションの追加] ダイアログボックスは、[プロジェクト] メニューから使用できます。



このダイアログボックスを使用して、IAR Embedded Workbench と外部ツール間のプロジェクト接続を設定します。たとえば、これは IAR Embedded Workbench で外部ツールから提供されたソースコードファイルをビルドする場合などに便利です。ソースファイルが自動的にプロジェクトに追加されます。ファイルのセットが変更になった場合、IAR Embedded Workbench でプロジェクトがビルドされると、新しいファイルのセットが自動的に使用されます。

このサポートを無効にする方法は、69 ページの [プロジェクト] オプションを参照してください。

## 使用するコネクション

接続を設定する外部ツールを選択します。

## OK

接続を指定するダイアログボックスが表示されます。

## Subversion のバージョン管理システムメニュー

[バージョン管理システム] サブメニューは、[プロジェクト] メニューか、[ワークスペース] ウィンドウのコンテキストメニューから選択できます。

以下のコマンドがあります。

コミット...
追加
戻す...
更新...
差分...
ログ...
プロパティ(P)...
更新(E)
プロジェクトをSCCプロジェクトに接続(N)...
プロジェクトをSCCプロジェクトから切断(D)...
プロジェクトをSubversionに接続...
プロジェクトをSubversionから切断...

外部バージョン管理システムの操作の詳細は、102 ページのバージョン管理システムの IDE 操作を参照してください。

### メニューのコマンド

以下のコマンドが Subversion で使用できます。

#### コミット

選択したファイルについて Tortoise の [コミット] ダイアログボックスを表示します。

#### 追加

選択したファイルについて Tortoise の [追加] ダイアログボックスを表示します。

#### 戻す

選択したファイルについて Tortoise の [戻す] ダイアログボックスを表示します。

#### 更新

選択したファイルについて Tortoise の [更新] ウィンドウを開きます。

#### 差分

選択したファイルについて Tortoise の [差分] ウィンドウを開きます。

#### ログ

選択したファイルについて Tortoise の [ログ] ウィンドウを開きます。



## プロパティ

選択したファイルについてバージョン管理システムで使用可能な情報を表示します。

## 更新

プロジェクトに含まれるすべてのファイルのバージョン管理システムの表示ステータスを更新します。このコマンドは、バージョン管理システムで管理するすべてのプロジェクトで常に使用できます。

## プロジェクトを Subversion に接続










svn.exe と TortoiseProc.exe がパスにあるかどうかを確認し、IAR Embedded Workbench プロジェクトと既存のチェックアウトされた作業用コピーとの接続を有効にします。この接続を作成すると、ステータス情報を示す特別な列 **【ワークスペース】** ウィンドウに表示されます。ソースファイルは IDE の外部からチェックアウトしなくてはならない点に注意してください。

## プロジェクトを Subversion から切断

選択した IAR Embedded Workbench プロジェクトと Subversion の接続を削除します。**【ワークスペース】** ウィンドウで SVN ステータス情報を示す列が、そのプロジェクトについては表示されなくなります。

# Subversion の状態

Subversion により管理される各ファイルは、複数の状態のいずれかになります。

	(青の A)	追加済。
	(赤の C)	衝突あり。
	(赤の D)	削除済。
	(赤の I)	無視されました。
	(ブランク)	変更なし。
	(赤の M)	変更済。
	(赤の R)	置換済。
	(灰色の X)	外部定義によって作成されたバージョンがつけられていないディレクトリ。
	(灰色の疑問符)	アイテムがバージョン管理の下にありません。



(黒の感嘆符) アイテムが存在しないか (SVN 以外のコマンドにより削除済)、不完全です。



(赤の波形符号) アイテムが別の型のアイテムによって妨害されました。

**注 :** IAR Embedded Workbench IDE のバージョン管理システムでは、**Subversion** が提供する情報を使用します。**Subversion** が状態について誤った、あるいは不完全な情報を提供すると、IDE で誤ったアイコンが表示されることがあります。

# プロジェクトのビルド

- プロジェクトのビルドの概要
- プロジェクトのビルド
- ビルドに関するリファレンス情報

---

## プロジェクトのビルドの概要

以下のトピックを解説します：

- プロジェクトのビルドの概要について
- ツールチェーンの拡張

### プロジェクトのビルドの概要について

ビルド処理は、以下の手順で構成されます。

- プロジェクトオプションの設定
- アプリケーションプロジェクトまたはライブラリプロジェクトのビルド
- ビルドで検出されたエラーの修正

**【バッチビルド】** コマンドを使用すると、ビルド処理の効率を上げることができます。このコマンドを使用すると、1回の操作で複数のビルドを実行できます。必要に応じて、ビルド前とビルド後のアクションを指定することも可能です。

プロジェクトをビルドするには、IAR Embedded Workbench IDE を使用する以外に、コマンドラインユーティリティ `iarbuild.exe` を使用方法もあります。

アプリケーションおよびライブラリオブジェクトのビルドの例については、インフォメーションセンタのチュートリアルを参照してください。ライブラリプロジェクトのビルドの詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

### ツールチェーンの拡張

IAR Embedded Workbench では、標準のツールチェーンを拡張するためのツール (Custom Build) が提供されています。この機能を使用して、外部ツール (IAR システムズ以外のベンダが提供するツール) を実行します。プロジェク

ト内の特定のファイルが変更されるたびに、外部ツールを実行させることができます。

**[カスタムツール設定]** ページでカスタムビルドオプションを指定すると、ビルドコマンドは、IAR Embedded Workbench IDE とその関連ファイルを処理するのと同じ方法で、外部ツールとその関連ファイルを処理します。外部ツールとその入力ファイルと生成される出力ファイルの関係は、C/C++ コンパイラ、c ファイル、h ファイル、o ファイルの間の関係に似ています。カスタムビルドオプションについて詳しくは、255 ページの *カスタムビルドオプション* を参照してください。

外部ツールの入力として使用するファイルのファイル名拡張子を指定します。プロジェクトを最後にビルドした後で入力ファイルが変更された場合は、c ファイルが変更されたときにコンパイラが実行されるのと同じように、外部ツールが実行されます。同様に、他の入力ファイル（インクルードファイルなど）への変更も検出されます。

外部ツールの名前を指定する必要があります。同時に、外部ツールが必要とするコマンドラインオプションや、外部ツールが生成する出力ファイルの名前も指定できます。ファイル情報の一部を表すのに、引数変数を使用できます。

カスタムビルドオプションは、プロジェクトツリーの任意のレベルに対して指定できます。指定したオプションは、プロジェクトツリーの下位レベルに継承されます。

### ツールチェーンに追加可能なツール

IAR Embedded Workbench ツールチェーンに追加できる外部ツールか、ツールの種類の例を以下に示します。

- 言語仕様に基づいてファイルを生成するツール（Lex、YACC など）
- バイナリファイル、たとえばビットマップイメージやオーディオデータを含むファイルを、アセンブラか C ソースファイルのデータテーブルに変換するツールです。（このデータは、コンパイルして、アプリケーションの他のファイルとリンク可能）

詳細については、132 ページの *外部ツールの追加* を参照してください。

---

## プロジェクトのビルド

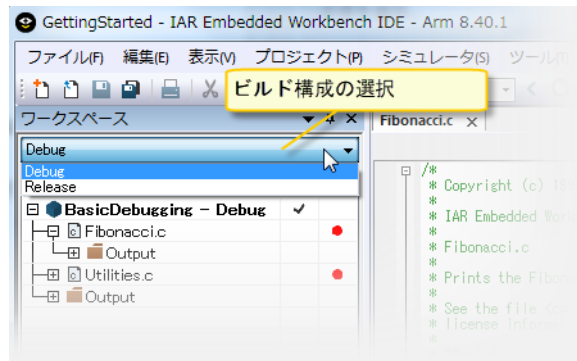
以下のタスクについて解説します。

- [オプション] ダイアログボックスを使用したプロジェクトオプションの設定
- プロジェクトのビルド

- ビルド中に検出されたエラーの修正
- ビルド前およびビルド後のアクションの使用
- バッチによる複数構成のビルド
- コマンドラインからのビルド
- 外部ツールの追加

### 【オプション】ダイアログボックスを使用したプロジェクトオプションの設定

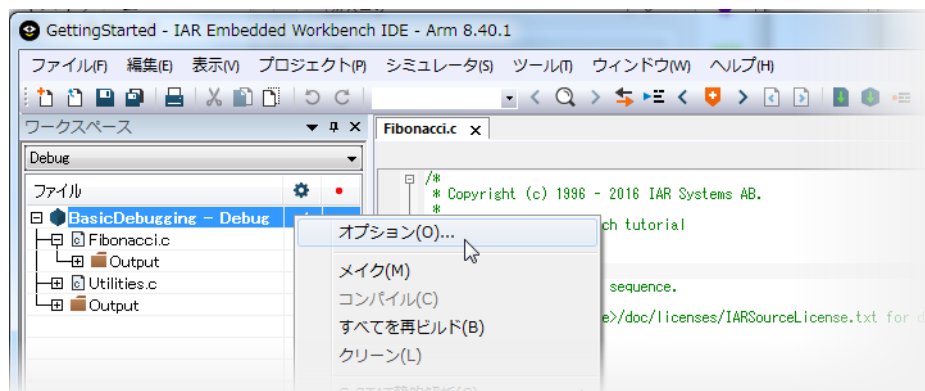
- 1 プロジェクトオプションを設定する前に、ビルド構成を選択します。



デフォルトでは、プロジェクトの作成時に IDE は [デバッグ] と [リリース] という 2 つのビルド構成を作成します。ビルド構成にはそれぞれ独自のプロジェクト設定があり、他の設定には依存しません。

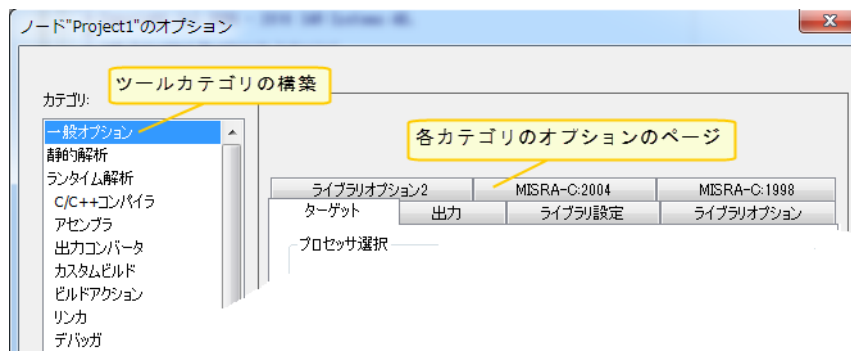
たとえば、デバッグに使用する設定は、最適化の程度は低く、デバッグに適した出力を生成します。逆に、最終アプリケーションのビルド構成は、高度に最適化され、フラッシュ / PROM プログラマに適した出力を生成します。

- プロジェクト全体、ファイルのグループ、個々のファイルなど、オプションを設定するレベルを決定してください。[ワークスペース] ウィンドウでそのレベルを選択（この例ではプロジェクトレベル）し、コンテキストメニューで[オプション]を選択して[オプション]ダイアログボックスを表示します。



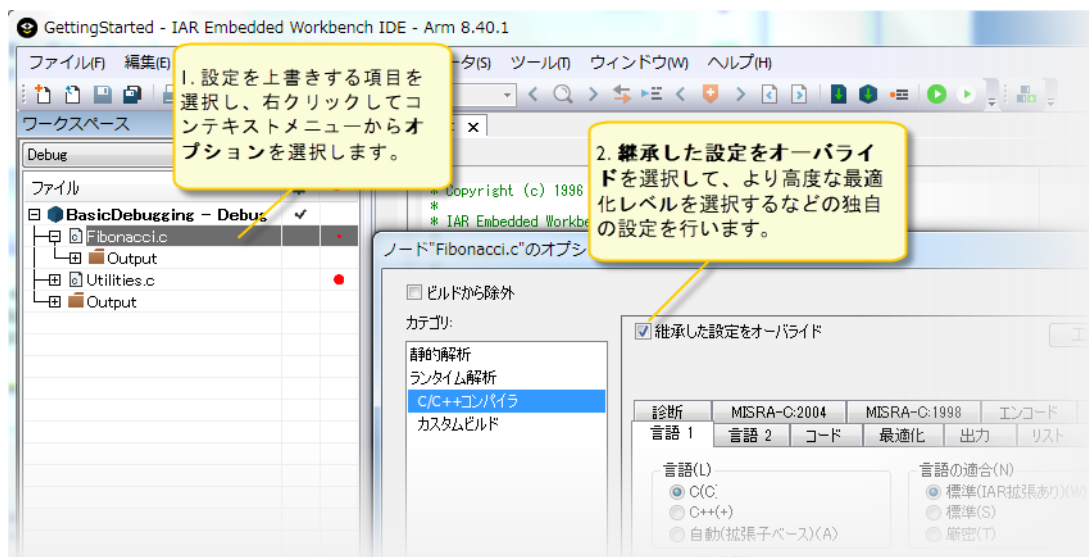
注：オプションの設定について、1つ重要な制限があります。グループまたはファイルレベルでオプションを設定（グループまたはファイルレベルでオーバーライド）すると、ファイルを適用対象とする上位レベルのオプションは、一切このグループまたはファイルに適用されなくなります。

- [オプション] ダイアログボックスでは、ビルドツールごとにビルドツールのカテゴリに対するオプションを指定します。



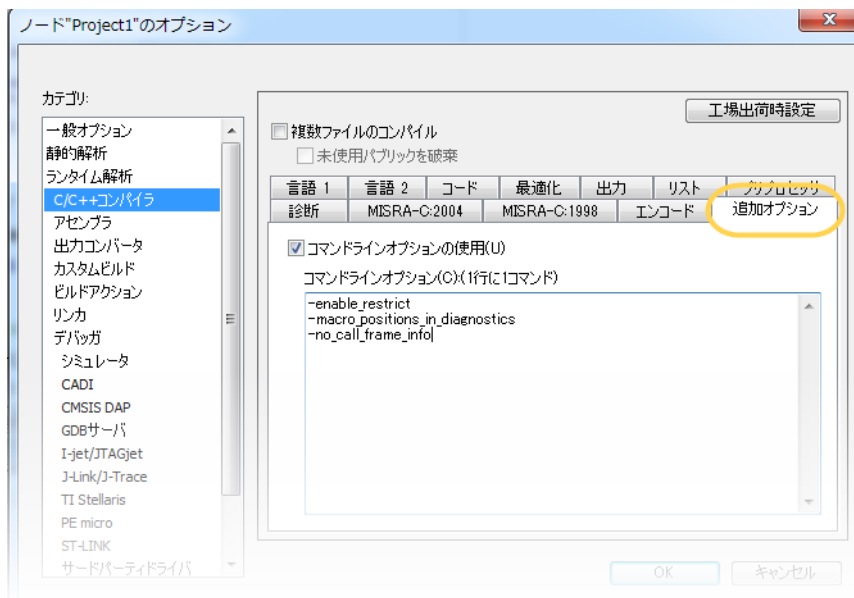
[一般オプション]、[リンカ]、[デバッガ] の各カテゴリのオプションは、プロジェクトレベルでのみ設定できます。これらはビルド構成全体に影響し、個々のグループやファイルには設定できないためです。ただし、他のカテゴリのオプションは、プロジェクトやファイルグループ、個別のファイルに対しても設定できます。

- 4 オプションを設定するビルドツールを選択するには、[カテゴリ] リストからカテゴリを選択します。[カテゴリ] リストで使用できるツールは、製品に含まれるツールによって異なります。カテゴリを選択すると、そのコンポーネントのオプションを含むページが表示されます。このページは複数ページにわたる場合もあります。
- 5 表示/変更するオプションのタイプに対応するタブをクリックします。適切に設定を行います。以下は参考情報です。
  - プロジェクトレベルの設定をオーバライドするには、必要な項目（たとえば特定のファイルグループまたは個々のファイル）を選択して、オプション [継承した設定をオーバライド] を選択します。



新しい設定は、選択されたグループのすべてのメンバ、すなわちファイルとファイルグループに影響を与えます。ローカルのオーバライドは、[ワークスペース] ウィンドウの別の列にチェックマークが付きます。

- IDE では使用できず、コマンドラインオプションとしてのみ使用可能なオプションを指定するには、[追加オプション] ページを使用します。



- すべての設定をデフォルトの出荷時設定に戻すには、[工場出荷時設定] ボタンをクリックします。このボタンは、[一般オプション] と [カスタムビルド] を除くすべてのカテゴリで使用できます。使用可能な出荷時設定は2つあります。Debug と Release です。どちらを使用するかは、ビルド構成に応じて異なります（118 ページの [新規ビルド構成] ダイアログボックス参照）。
- 認識されないファイル名拡張子を持つソースファイルをプロジェクトに追加した場合、そのソースファイルに対してオプションを設定することはできません。ただし、そのファイル名拡張子に対するサポートを追加できます。詳細については、88 ページの [ファイル名拡張子] ダイアログボックスを参照してください。

## プロジェクトのビルド

プロジェクトは、アプリケーションプロジェクトまたはライブラリプロジェクトとしてビルドできます。



ビルドコマンドにアクセスするには、[プロジェクト] メニューか [ワークスペース] ウィンドウで項目を右クリックして表示されるコンテキストメニューを使用します。

アプリケーションプロジェクトとしてプロジェクトをビルドするには、[メイク]、[コンパイル]、[すべてを再ビルド] の3つのビルドコマンドのいずれかを選択します。これらはバックグラウンドで実行されるため、プロジェクトのビルド中にも IDE で編集や作業をそのまま続けられます。

ライブラリプロジェクトとしてプロジェクトをビルドするには、プロジェクトをビルドする前に [プロジェクト] > [オプション] > [一般オプション] > [出力] > [出力ファイル] > [ライブラリ] を選択します。続いて、オプションのダイアログボックスの [カテゴリ] リストで [リンク] が [ライブラリビルダ] に置き換わり、ビルドの結果がライブラリになります。例については、チュートリアルを参照してください。

詳細については、206 ページの [プロジェクト] メニューを参照してください。

## ビルド中に検出されたエラーの修正

エラーメッセージは [ビルド] メッセージウィンドウに表示されます。

[ビルド] メッセージウィンドウへの出力レベルを指定するには、次の手順に従います。

- 1 [ビルド] メッセージで右クリックし、コンテキストメニューを開きます。
- 2 コンテキストメニューから、出力レベルを選択します。[すべて] は、コンパイルおよびリンク情報を含むすべてのメッセージを表示し、[エラー] は、エラーだけを表示しますが、ワーニングやその他のメッセージは表示しません。

ソースコードにエラーが含まれる場合、[ビルド] ウィンドウでエラーリストのエラーメッセージをダブルクリックするか、エラーを選択して **Enter** キーを押すことによって、該当するソースファイルの選択したエラーの位置に直接移動できます。

ビルド中に検出された問題をすべて解決して、プロジェクトをリビルドしたら、生成されたコードをソースレベルで直接デバッグできます。

[ビルド] メッセージウィンドウの情報については、「135 ページの [ビルド] ウィンドウ」を参照してください。

## ビルド前およびビルド後のアクションの使用

必要に応じて、ビルド前とビルド後に実行するアクションを指定することが可能です。[オプション] ダイアログボックスのビルド前およびビルド後のア

クションのオプション（[プロジェクト] メニューからアクセス）では、必要なアクションを指定します。

ビルドアクションのオプションについては、「257 ページの *ビルドアクションオプション*」を参照してください。



### ビルド前アクションの使用によるタイムスタンプ

ビルド前アクションを使用して、ビルドに関するタイムスタンプを結果のバイナリファイルに埋め込むことができます。以下の手順を実行します。

- 1 専用のタイムスタンプファイル（timestamp.c など）を作成し、プロジェクトに追加します。
- 2 このソースファイルで、プリプロセッサマクロ `__TIME__` と `__DATE__` を使用して文字列変数を初期化します。
- 3 [プロジェクト] > [オプション] > [ビルドアクション] を選択して、[ビルドアクション] ダイアログボックスを開きます。
- 4 [プリビルドコマンドライン] テキストフィールドにビルド前アクションを指定します。たとえば、以下のように指定します。

```
cmd /c "del "$OBJ_DIR$¥timestamp.o"
```

このコマンドは timestamp.o オブジェクトファイルを削除します。

別の方法として、オープンソースコマンドラインユーティリティ `touch` をこの目的に使用できます。また、他にもソースファイルの変更時刻を更新する適当なユーティリティを使用できます。以下に例を示します。

```
"touch $PROJ_DIR$¥timestamp.c"
```

- 5 プロジェクトが完全には最新状態でない場合は、次の [メイク] コマンドの使用時に、通常のビルドプロセスの前にビルド前アクションが呼び出されます。そして、通常のビルドプロセスで常に timestamp.c が再コンパイルされ、最終的には正しいタイムスタンプがバイナリファイルに埋め込まれます。

すでにプロジェクトが最新状態の場合には、ビルド前アクションは呼び出されません。すなわち、ビルドは実行されず、バイナリファイルには最後にビルドされたときのタイムスタンプが引き続き使用されます。

### バッチによる複数構成のビルド

バッチビルド機能を使用すると、複数の構成を同時にビルドできます。バッチは、ビルド構成が順序付けて記述されているリストです。[バッチビルド] ダイアログボックスは、[プロジェクト] メニューからアクセスでき、複数の構成バッチを作成、変更、ビルドできます。

複数の構成を含むワークスペースの場合は、複数のバッチを定義すると便利です。ワークスペース全体をビルドするのではなく、リリース設定とデバッグ設定のように特定のビルド構成だけをビルドできます。

[**バッチビルド**] ダイアログボックスの詳細については、137 ページの [**バッチビルド**] ダイアログボックスを参照してください。

コマンドラインからのビルド

コマンドラインからプロジェクトをビルドするには、common¥bin ディレクトリにある IAR コマンドラインビルドユーティリティ (iarbuild.exe) を使用します。これは通常、連続した統合のテストを自動化する際に役立ちます。

入力としてプロジェクトファイルを使用して、以下の構文で呼び出します。

```
iarbuild project.ewp [ -clean | -build | -make | -cstat_analyze |  
-cstat_clean] config[,config1,config2,...] [*[-log  
errors|warnings|info|all] [-parallel number] [-varfile filename]
```

使用可能なパラメータは以下のとおりです。

パラメータ	説明
project.ewp	IAR Embedded Workbench プロジェクトファイル。
-clean	すべての中間ファイルおよび出力ファイルを削除します。
-build	指定したビルド構成のすべてのファイルをリビルド / 再リンク。
-make	最後のビルド以降に変更されたファイルだけをコンパイル、アセンブル、リンクして、指定したビルド構成を最新状態に更新。
-cstat_analyze	C-STAT を使用してプロジェクトを分析し、メッセージについての情報を生成します。詳細については、『C-STAT® Static Analysis Guide』を参照してください。
-cstat_clean	プロジェクトの C-STAT メッセージデータベースを消去します。詳細については、『C-STAT® Static Analysis Guide』を参照してください。
config *	config、ビルドする構成の名前（定義済の構成 [Debug] または [Release] か、ユーザが独自に定義した名前）を指定。（ビルド構成の詳細については、101 ページの <i>プロジェクトとビルド構成</i> を参照）。 *（ワイルドカード文字）、-clean、-build、-make の各コマンドでは、プロジェクトに定義されたすべての構成が処理されます。

表 4: iarbuild.exe コマンドラインオプション

パラメータ	説明
-log errors	ビルドのエラーメッセージを表示。
-log warnings	ビルドのワーニング、エラーメッセージを表示。
-log info	ビルドのワーニングメッセージ、エラーメッセージ、および #pragma message プリプロセッサディレクティブによって出力されるメッセージを表示します。
-log all	ビルドで出力されるすべてのメッセージを表示（コンパイラのサインオン情報やフルコマンドラインなど）。
-parallel number	CPU でコアを有効利用するため、コンパイラを実行するための並列処理の数を指定します。
-varfile filename	ワークスペースのスコープにカスタム定義した引数変数を、使用するファイルを指定することによりビルドエンジンで使用可能にします。93 ページの <i>「カスタムの引数変数の設定」ダイアログボックス</i> を参照してください。

表 4: iarbuild.exe コマンドラインオプション (続き)

プロジェクトファイルを指定しないでコマンドシェルからアプリケーションを実行すると、使用できるパラメータとその構文を示すサインオンメッセージが表示されます。

ビルドプロセスが成功した場合は、IAR コマンドラインユーティリティは 0 を返します。それ以外は、ゼロ以外の数字と診断メッセージを返します。

外部ツールの追加

ツール *Flex* をツールチェーンに追加する例を以下に示します。他のツールも同じ手順で追加できます。

この例では、Flex はファイル myFile.lex を入力として受け取ります。出力として、myFile.c と myFile.h の 2 つのファイルが生成されます。

- 1 使用するファイル、ここでは myFile.lex をプロジェクトに追加します。
- 2 **「ワークスペース」** ウィンドウでこのファイルを選択して、**「プロジェクト」** > **「オプション」** を選択します。カテゴリリストで **「カスタムビルド」** を選択します。
- 3 **「ファイル名の拡張子」** フィールドにファイル名拡張子「.lex」を入力します。先頭にピリオド(.)を忘れずに指定してください。
- 4 **「コマンドライン」** フィールドに、外部ツールを実行するコマンドラインを入力します。以下に例を示します。

```
flex $FILE_PATH$ -o$FILE_BNAME$.c
```

ビルド処理中に、このコマンドラインは以下のように展開されます。

```
flex myFile.lex -omyFile.c
```

引数変数および特に `$FILE_BNAME$` の使用方法には注意してください。これは入力ファイルのベース名を出力します。この例では `c` 拡張子が追加されて、入力ファイル `foo.lex` と同じディレクトリに C ソースファイルが提供されます。これらの変数の詳細については、91 ページの *引数変数* を参照してください。

- 5** **【出力ファイル】** フィールドに、ビルドに関連して生成される出力ファイルを記述します。この例では、ツール **Flex** がソースファイルとヘッダファイルを1つずつ生成します。**【出力ファイル】** テキストボックスでこれら2つのファイルを表すテキストは以下のようになります。

```
$FILE_BPATH$.c  
$FILE_BPATH$.h
```

- 6** 外部ツールがビルド中に使用するファイルが他にもある場合は、それらのファイルをたとえば次のように **【追加入力ファイル】** フィールドに追加する必要があります。

```
$TOOLKIT_DIR$%inc%stdio.h
```

使用するファイルを追加する必要があるのは、依存ファイルが変更された場合、条件が変わるのでリビルドする必要があるためです。

- 7** **OK** をクリックします。
- 8** アプリケーションをビルドするには、**【プロジェクト】** > **【メイク】** を選択します。

---

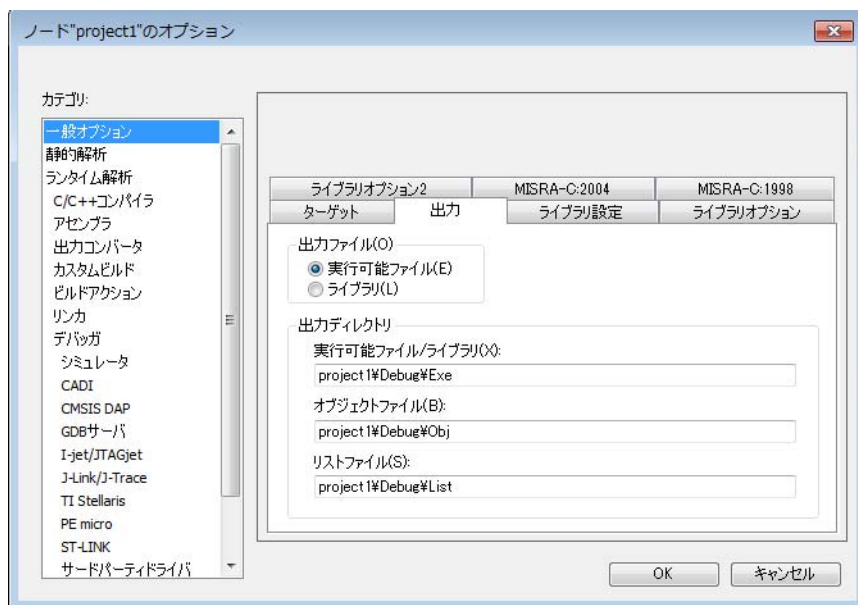
## ビルドに関するリファレンス情報

リファレンス情報：

- 134 ページの **【オプション】** ダイアログボックス
- 135 ページの **【ビルド】** ウィンドウ
- 137 ページの **【バッチビルド】** ダイアログボックス
- 138 ページの **【バッチビルドの編集】** ダイアログボックス

## 【オプション】 ダイアログボックス

【オプション】 ダイアログボックスは、【プロジェクト】メニューから表示します。



このダイアログボックスを使用して、プロジェクト設定を指定します。

125 ページの 【オプション】 ダイアログボックスを使用したプロジェクトオプションの設定を参照してください。

### カテゴリ

オプションを設定する対象のビルドツールを選択します。使用可能なカテゴリは、IAR Embedded Workbench IDE にインストールされているツールによって異なり、通常は以下が含まれます。

- 一般オプション
- 静的解析。これらのオプションについては、『C-STAT® Static Analysis Guide』を参照してください
- ランタイム解析。これらのオプションについて詳しくは、『ARM 用 C-SPY® デバuggガイド』を参照してください
- C/C++ コンパイラ
- アセンブラ

- 出力コンバータ、ELF 出力を Motorola、Intel 標準、その他の簡易フォーマットに変換するためのオプション（253 ページの出力コンバータオプションを参照）
- カスタムビルド、ツールチェーンの拡張オプション
- ビルドアクション、ビルド前 / ビルド後のアクション用オプション
- リンカ。アプリケーションプロジェクトで使用可能。ライブラリプロジェクトでは使用不可
- ライブラリビルダ。ライブラリプロジェクトでは使用可。アプリケーションプロジェクトでは使用不可
- デバッグ
- シミュレータ
- C-SPY ハードウェアドライバ、その他のハードウェアデバッグに固有のオプション

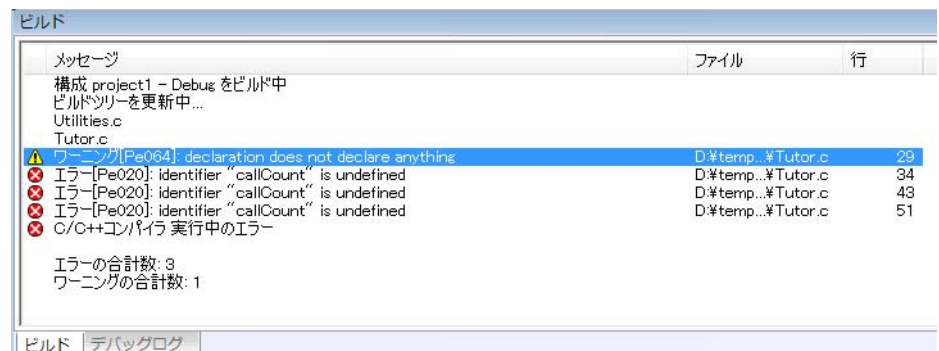
カテゴリを選択すると、IDE のコンポーネントに対するオプションのページが表示されます。

## 工場出荷時設定

すべての設定をデフォルトの工場出荷時設定に戻します。なお、このオプションはすべてのカテゴリに使用できるわけではありません。

## 【ビルド】 ウィンドウ

【ビルド】 ウィンドウは、【表示】 > 【メッセージ】 を選択すれば使用できます。

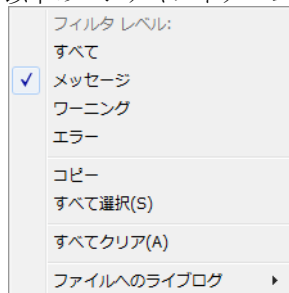


このウィンドウには、ビルド構成をビルドする際に生成されたメッセージが表示されます。デフォルトでは、このウィンドウは他のメッセージウィンドウとグループ化されて表示されます。【ビルド】 ウィンドウでメッセージをダ

ブルクリックすると、該当ファイルが編集用に開かれ、挿入ポイントが正しい箇所に表示されます。

## コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

### すべて

コンパイラとリンカの情報を含むすべてのメッセージを表示します。

### メッセージ

すべてのメッセージを表示。

### ワーニング

ワーニングやエラーを表示します。

### エラー

エラーのみ表示します。

### コピー

ウィンドウの内容をコピーします。

### すべて選択

ウィンドウの内容を選択します。

### すべてをクリア

ウィンドウの内容を削除します。

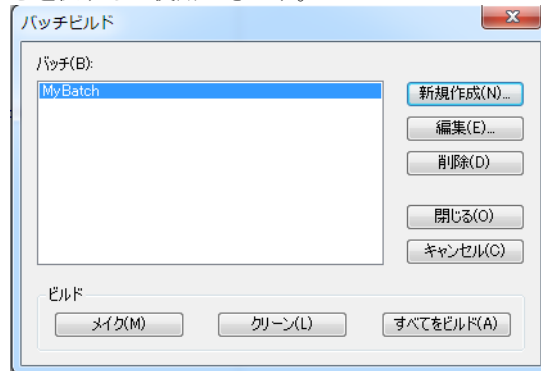
### ファイルへのライブログ

ログファイルにビルドメッセージを書き込んで、ログにフィルターレベルを設定するためのコマンドで、サブメニューを表示します。



## [バッチビルド] ダイアログボックス

[バッチビルド] ダイアログボックスは [プロジェクト] > [バッチビルド] を選択すると使用できます。



このダイアログボックスには、ビルド構成の定義済みバッチがすべて一覧表示されます。詳細については、130 ページのバッチによる複数構成のビルドを参照してください。

### バッチ

現在定義されたビルド構成のバッチのこのリストから、ビルドするバッチを選択します。

### ビルド

実行するビルドコマンドを指定します。

- メイク
- クリーン
- すべてを再ビルド

### 新規作成

新しいビルド構成のバッチを定義するための [バッチビルドの編集] ダイアログボックスを表示します (138 ページの [バッチビルドの編集] ダイアログボックスを参照)。

### 削除

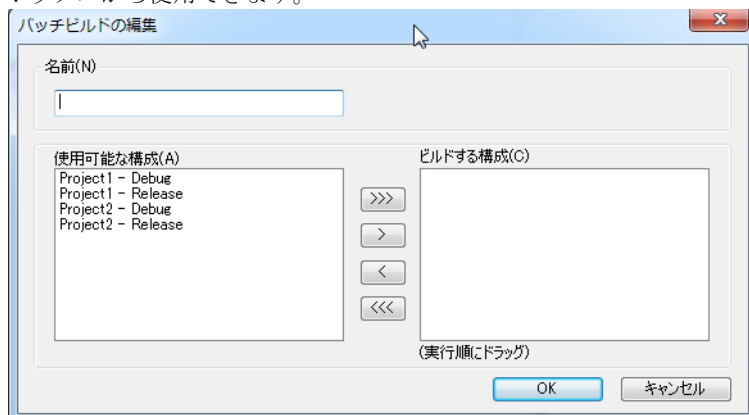
選択したバッチを削除します。

## 編集

既存のビルド構成のバッチを編集するための **「バッチビルドの編集」** ダイアログボックスを表示します。

### 「バッチビルドの編集」ダイアログボックス

**「バッチビルドの編集」** ダイアログボックスは、**「バッチビルド」** ダイアログボックスから使用できます。



このダイアログボックスを使用して、ビルド構成の新しいバッチを作成し、既存のバッチを編集します。

## 名前

作成しているバッチ名を入力するか、編集中のバッチの既存の名前を変更します（希望する場合）。

## 使用可能な構成

ワークスペースに属するすべてのビルド構成のこのリストから、作成または編集しているバッチに含める構成を選択します。

ビルド構成を **「使用可能な構成」** リストから **「ビルドする構成」** リストに移動するには、矢印ボタンを使用します。

## ビルドする構成

作成または編集中のバッチに含めるビルド構成の一覧を表示します。ビルド構成を上下にドラッグして、構成の順序を設定します。

# 編集

- IAR Embedded Workbench エディタの概要
- ファイルの編集
- プログラミングのサポート
- エディタについてのリファレンス情報

---

## IAR Embedded Workbench エディタの概要

以下のトピックを解説します：

- エディタの概要について
- ソースブラウザ情報の概要について
- エディタ環境のカスタマイズ

IAR Embedded Workbench IDE で外部エディタを使用する方法については 41 ページの [外部エディタの連携](#) を参照してください。

### エディタの概要について

統合されたテキストエディタでは、複数のファイルを同時に編集できるほか、基本的な編集機能のほかに以下のようなソフトウェア開発に固有の機能が提供されます。

- 語句とコードの自動入力
- 新しい行およびブロックの自動インデント
- 丸括弧と角括弧のマッチング
- ソースファイル内での関数ナビゲーション
- DLIB ライブラリ関数および言語拡張のリファレンス情報を表示できる文脈依存ヘルプシステム
- C/C++ プログラムおよびアセンブラディレクティブの構文を識別するテキストスタイルと色
- 複数ファイル検索などの強力な検索 / 置換コマンド
- エラーリストからコンテキストを直接表示
- マルチバイトサポート
- パラメータのヒント

- ブックマーク
- 各ウィンドウで無制限にアンドゥ/リドゥ可能

## ソースブラウズ情報の概要について

ソースブラウズ情報は、オプションでバックグラウンドで継続的に生成されます。この情報は、以下のようなプログラミングの支援として役立つ数多くの機能で使用されます。

- [ソースブラウザ] ウィンドウ
- 定義に移動、または宣言に移動
- すべての参照を検索
- 関数への呼出しまたは関数からの呼出しをすべて検索。結果はコールグラフとして表示されます

プロジェクトのファイルを保存する際に、ソースブラウズの情報が更新されます。編集済みのソースファイルを保存したり、新しいプロジェクトを開くときは、最新の情報が表示されるまで少し時間がかかります。更新中は、処理情報がステータスバーに表示されます。



**注:** IAR Embedded Workbench IDE から別のプログラムに切り替える際にソースブラウズ情報の生成を停止するには、**IDE がフォアグラウンドプロセスでない場合、ソースブラウザとビルドステータスが更新されません**オプションを有効にしてください。

## エディタ環境のカスタマイズ

IDE エディタは、[IDE オプション] の [エディタ] ページと [エディタ] > [色とフォント] ページで構成できます。これらのページにアクセスするには、[ツール] > [オプション] を選択します。

これらのページの詳細については、212 ページの [ツール] メニューを参照してください。

---

## ファイルの編集

エディタウィンドウでは、ソースコードの記述、表示、変更を行います。

以下のタスクについて解説します。

- テキストの自動インデント
- 中括弧と括弧のマッチング

- エディタウィンドウをペインに分割
- テキストのドラッグ
- コードの折りたたみ
- 語句の入力補完
- コードの入力補完
- パラメータのヒント
- コードテンプレートの使用と追加
- 構文カラー表示
- ブックマークの追加
- エディタコマンドとショートカットキーの使用とカスタマイズ
- ステータス情報の表示

関連項目：

- 148 ページの *プログラミングのサポート*
- 41 ページの *外部エディタの連携*

## テキストの自動インデント

テキストエディタには、さまざまな種類のインデントがあります。アセンブラソースファイルと通常のテキストファイルは、エディタによって、行の先頭が前の行の先頭に一致するように自動インデントされます。

複数の行をインデントする場合は、該当する行を選択して、**Tab** キーを押します。

選択した行をまとめて再び左に移動するには、**Shift+Tab** キーを押します。

C/C++ ソースファイルの場合、エディタは C/C++ ソースコードの構文に従って行をインデントします。インデントは以下のタイミングで実行されます。

- Enter キーを押したとき
- {、}、:、# のいずれかの特殊文字が入力されたとき
- 1 行または複数行を選択して **【編集】 > 【自動インデント】** コマンドを選択したとき

インデントを有効/無効にするには、以下の手順を実行します。

- 1 **【ツール】 > 【オプション】** を選択して、**【エディタ】** を選択します。
- 2 **【自動インデント】** オプションを選択 / 選択解除します。

C/C++ の自動インデントをカスタマイズするには、**【設定】** ボタンをクリックします。

詳細については、63 ページの *〔自動インデントの設定〕* ダイアログボックスを参照してください。

## 中括弧と括弧のマッチング

対応する括弧を淡い灰色で強調表示するには、括弧の横に挿入ポイントを置きます。

```
void NextCounter(void)
{
    callCount += 1;
}
```

挿入ポイントが括弧の近くにある間は、対応する括弧は強調表示されたままです。

挿入ポイントを含む中括弧で囲まれたテキストをすべて選択するには、**〔編集〕 > 〔括弧のマッチング〕** を選択します。その後は、**〔括弧のマッチング (拡張)〕** または **〔括弧のマッチング (縮小)〕** を選択するたびに、選択される範囲が次の階層の中括弧で囲まれた範囲まで拡大または縮小します。

**注：**これらの機能（自動検出、括弧で囲まれたテキストの選択）はどちらも、`()`、`[]`、`{}`、`<>`（**〔すべての括弧のマッチング〕** が必要です）に適用されます。

## エディタウィンドウをペインに分割

エディタウィンドウを水平または垂直に複数のペインに分割して、同一ソースファイルの異なる部分を同時に表示したり、2 つの異なるペイン間でテキストを移動することができます。

ウィンドウをペインに分割（縦または横方向）するには、**〔ウィンドウ〕 > 〔分割〕** コマンドを使用します。

1 つのペインに戻すには、分割コントロールをダブルクリックする、またはウィンドウの端にドラッグします。

## テキストのドラッグ

エディタウィンドウ内でテキストを動かしたり、エディタウィンドウ間でコピーするには、テキストを選択して新しい場所にドラッグしてください。

## コードの折りたたみ

コードの折りたたみを使用すると、コードのセクションを非表示にしたり表示することができます。

複数の行を折りたたんだり展開するには、折り目の余白にある折りたたみポイントをクリックします。



折りたたみポイントの位置は、ドキュメントの内容の階層構造に依存します。たとえば、C/C++ の括弧の文字や XML ファイルの要素階層などです。**【折り畳みを切り替え】** コマンド (Ctrl+Alt+F) を使用して、現在のドキュメントにあるすべての折り目を展開する（または折りたたむ）ことができます。このコマンドは、エディタウィンドウの**【編集】**メニューから使用できます。折り目の余白は、**【ツール】 > 【オプション】 > 【エディタ】** から有効または無効にすることができます。

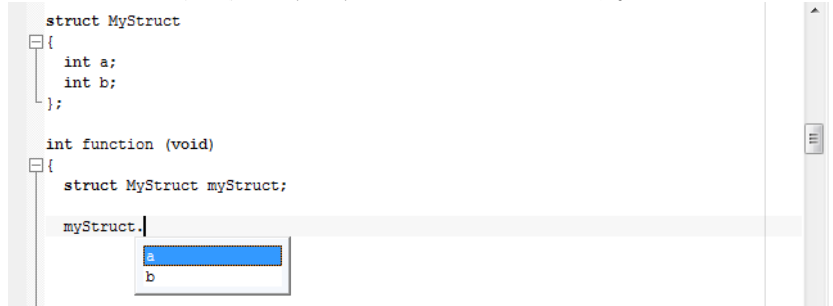
## 語句の入力補完

語句の入力補完機能では、ドキュメントの他の部分の内容から入力語を推測して補完します。

入力した語句の続きをエディタに補完させるには、Ctrl+Alt+ スペースキーを押すか、コンテキストメニューから**【語句の入力補完】**を選択します。候補が正しくない場合は、コマンドを繰り返して新しい候補を表示します。

## コードの入力補完

クラスで使用可能なシンボルの一覧をエディタで表示するには、クラスまたはオブジェクト名の後に .、->、または :: を入力します。



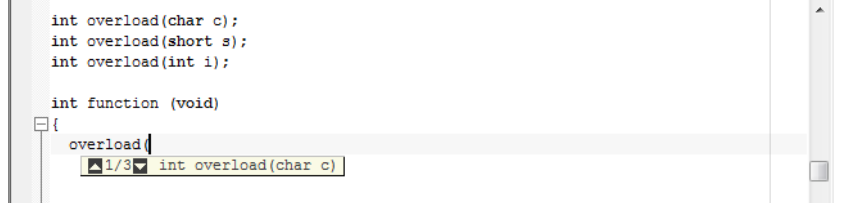
., ->, :: の後ろ以外にカーソルを配置すると、コンテキストメニューにアクティブな翻訳単位で使用可能なすべてのシンボルが一覧表示されます。

リストのシンボル名をクリックするか、矢印キーでシンボル名を選択して Return を押すと現在の挿入ポイントにそれが挿入されます。

## パラメータのヒント

関数のパラメータをツールチップ情報としてエディタで表示するには、関数名の後に最初の括弧を入力します。

ある関数のオーバーロードされたバージョンがいくつかある場合、ツールチップの矢印 (Ctrl+Up/Down) をクリックすると使用するバージョンを選択できます。パラメータをテキストとして挿入するには、Ctrl+Enter キーを押します。



## コードテンプレートの使用と追加

コードテンプレートは、たとえば for ループや if 文のように、頻繁に使用されるソースコードシーケンスを簡単に挿入するための方法です。コードテンプレートは、通常のテキストファイルで定義します。デフォルトで、いくつかのサンプルテンプレートが提供されています。それ以外に、簡単に独自のコードテンプレートを追加できます。

コードテンプレートの用法を設定するには、以下の手順に従います。

- 1 [ツール] > [オプション] > [エディタ] > [セットアップファイル] を選択します。
- 2 [コードテンプレートの使用] オプションを選択 / 選択解除します。デフォルトで、コードテンプレートは有効になっています。
- 3 テキストフィールドで、使用するテンプレートファイルを指定します。

- デフォルトのテンプレートファイル

オリジナルのテンプレートファイル CodeTemplates.txt (IAR Embedded Workbench の英語版と日本語版の両方をお使いの場合は CodeTemplates.ENU.txt や CodeTemplates.JPN.txt) は、別のディレクトリに格納されます (190 ページの [グローバル設定のファイル](#) 参照)。これはローカルにコピーされたファイルで、編集しても安全です。

- 独自のテンプレートファイル

独自のテンプレートファイルを選択するには、先にファイルが作成済みでなければならない点に注意してください。自分のテンプレートファイルを作成する場合は、[編集] > [コードテンプレート] > [テンプレートの編集] を選択し、新しいファイル名でコードテンプレートを追加します。テ



ンプレートを定義するための構文は、デフォルトテンプレートファイルに記述されています。

参照ボタンを使用して選択することもできます。

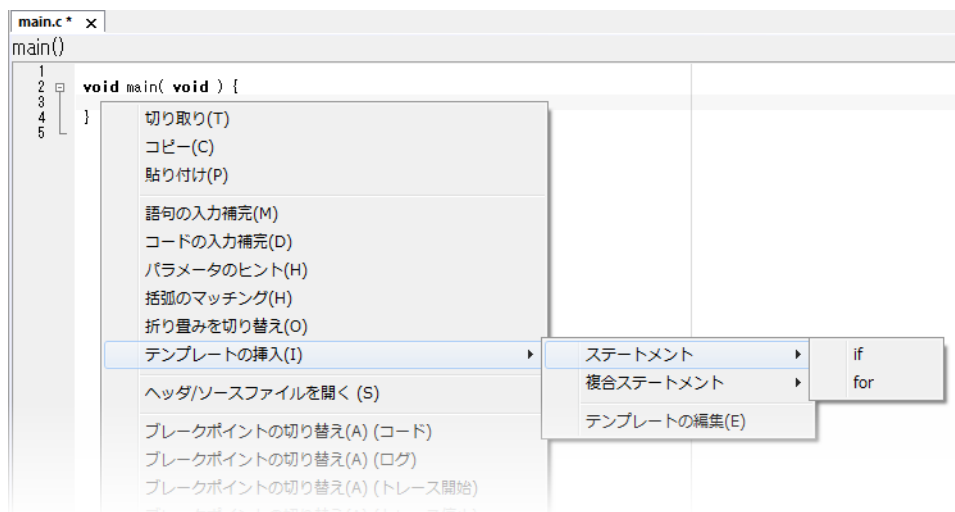
- 4 自分のテンプレートファイルで新しいテンプレートを使用するには、つぎのことを行う必要があります。
  - そのファイル名を [コードテンプレートの使用] テキストボックスから削除します。
  - [コードテンプレートの使用] オプションの選択を解除し、[OK] をクリックします。
  - IAR Embedded Workbench IDE を再起動します。
  - [ツール] > [オプション] > [エディタ] > [セットアップファイル] を再び選択します。

今度は、選択した言語のバージョンの IDE に対応したデフォルトのコードテンプレートファイルが [コードテンプレートの使用] テキストボックスに表示されるはずです。チェックボックスをオンにすると、テンプレートが有効になります。

ソースコードにコードテンプレートを挿入するには、以下の手順に従います。

- I エディタウィンドウで、テンプレートを挿入する場所を右クリックし、[テンプレートの挿入] (Ctrl+Alt+V) を選択します。

2 表示されるメニューからコードテンプレートを選択します。



終値とカウンタ変数を必要とする for ループのように、コードテンプレートがフィールドへの入力が必要とする場合は、入力ダイアログボックスが表示されます。

## 構文カラー表示

[ツール] > [オプション] > [エディタ] > [構文の強調表示] オプションを有効にすると、IAR Embedded Workbench エディタは、自動的に以下のソースコードのさまざまな部分の構文を認識します。

- C と C++ のキーワード
- C と C++ のコメント
- アセンブラディレクティブとコメント
- プリプロセッサのディレクティブ
- 文字列

ソースコードは、部分ごとに異なるテキストスタイルで表示されます。

これらのスタイルを変更するには、[ツール] > [オプション] を選択して、[エディタ] > [色とフォント] オプションを使用します。詳細については、67 ページの [色とフォント] オプションを参照してください。

自動的に構文カラー表示されるキーワードのセットを定義するには、次の手順に従います。

- 1 テキストファイルに、自動的に構文カラー表示の対象にするすべてのキーワードを記述します。各キーワードは、スペースや改行で区切ります。
- 2 [ツール] > [オプション] を選択して、[エディタ] > [セットアップファイル] を選択します。
- 3 [カスタムキーワードファイルの使用] オプションを選択して、新しく作成したテキストファイルを指定します。参照ボタンを使用して選択することもできます。
- 4 [エディタ] > [色とフォント] を選択して、[構文の色] リストから [ユーザキーワード] を選択します。フォント、色、タイプスタイルを指定します。詳細については、67 ページの [色とフォント] オプションを参照してください。

エディタウィンドウで、キーワードファイルに記述したキーワードを入力して、指定どおりにそのキーワードがカラー表示されていることを確認します。

### ブックマークの追加

[編集] > [移動] > [ブックマークの切替え] コマンドを使用すると、ブックマークを追加 / 削除できます。ブックマークされた位置間を移動するには、[編集] > [移動] > [次のブックマークへ移動] または [前のブックマークへ移動] を選択します。

### エディタコマンドとショートカットキーの使用とカスタマイズ

[編集] メニューには、無制限のアンドゥ / リドゥといった、エディタウィンドウでの編集と検索のためのコマンドがあります。これらのコマンドの一部は、エディタウィンドウを右クリックして表示されるコンテキストメニューからも選択できます。各コマンドの詳細については、198 ページの [編集] メニューを参照してください。

以下の操作を行うエディタショートカットキーもあります。

- 挿入ポイントの移動
- テキストのスクロール
- テキストの選択

ショートカットキーの詳細については、181 ページの *エディタのショートカットキー操作のまとめ* を参照してください。

デフォルトのショートカットキーバインディングを変更するには、[ツール] > [オプション] を選択して、[キーカスタマイズ] タブをクリックしま

す。詳細については、56 ページの **「キーカスタマイズ」** オプションを参照してください。

## ステータス情報の表示

ステータスバーは、**「表示」** > **「ステータスバー」** を選択すれば使用できます。詳細については、45 ページの **「IAR Embedded Workbench IDE」** ウィンドウを参照してください。



# プログラミングのサポート

ソフトウェア開発で役立つ機能がいくつかエディタに備わっています。このセクションでは、エディタの使用に関するさまざまなタスクについて説明します。

以下のタスクについて解説します。

- 挿入ポイント履歴に移動します
- 関数への移動
- シンボルの定義または宣言の検索
- シンボルへの参照の検索
- 選択した関数についての関数の呼出しの検索
- ソースファイルとヘッダファイル間の切替え
- ブラウズ情報の表示
- テキスト検索
- オンラインヘルプのリファレンス情報へのアクセス

## 挿入ポイント履歴に移動します

挿入ポイントの現在の位置は、**「定義に移動」** コマンドの実行時や、**「ファイルから検索」** コマンドの結果をクリックしたときに、挿入ポイントの履歴に追加されます。**「次へ移動」**  と **「前へ移動」**  ボタン（あるいは Alt + → または Alt + ←）を使用して、履歴内を前後に移動できます。

## 関数への移動



エディタウィンドウの右上隅にある **「関数に移動」** ボタンをクリックすると、ウィンドウに表示されているソースファイルで定義されているすべての関数がリスト表示されます。リストで関数をクリックすると、その関数の位置に直接移動できます。ファイルを保存すると、リストが更新されます。

## シンボルの定義または宣言の検索

グローバルシンボルや関数の定義または宣言を表示するには、以下に示す3つの方法があります。

- エディタウィンドウでシンボルを右クリックして、表示されるコンテキストメニューで **【定義に移動】** または **【宣言に移動】** コマンドを選択。複数の宣言が見つかった場合、それらは**宣言**ウィンドウに一覧表示され、そこから特定の宣言に移動することができます
- **【ソースブラウザ】** ウィンドウで、シンボルをダブルクリックして定義を表示
- **【ソースブラウザ】** ウィンドウでシンボルか関数を右クリックして、表示されるコンテキストメニューで **【定義に移動】** コマンドを選択

シンボルや関数の定義は、エディタウィンドウに表示されます。

## シンボルへの参照の検索

特定のシンボルへのすべての参照を検索するには、エディタウィンドウでシンボルを選択して右クリックし、コンテキストメニューから **【すべての参照を検索】** を選択します。見つかったすべての参照が **【参照】** ウィンドウに表示されます。

参照間を移動できるようになりました。

## 選択した関数についての関数の呼出しの検索

関数からの呼出しおよび関数の呼出しをすべて検索するには、エディタウィンドウまたは **【ソースブラウザ】** ウィンドウで関数を選択して右クリックし、コンテキストメニューから **【すべてのコール元を検索】** または **【すべてのコール先を検索】** を選択します。結果は **【コールグラフ】** ウィンドウに表示されます。

関数呼出し間を移動できるようになりました。

## ソースファイルとヘッダファイル間の切替え

挿入ポイントが `#include` 行にある場合、コンテキストメニューで **["header.h" を開く]** コマンドを選択して、ヘッダファイルをエディタウィンドウで開くことができます。また、コマンド **【ヘッダ/ソースファイルを開く】** を選択すると、現在のファイルに対応するヘッダファイルやソースファイルを開いたり、すでに開いている場合はアクティブにしたりできます。このコマンドは、挿入ポイントが `#include` 行の近くにあるときに選択できます。

## ブラウザ情報の表示

- 1 [ソースブラウザ] ウィンドウを開くには、[表示] > [ソースブラウザ] > [ソースブラウザ] を選択します。表示されるのは、アクティブなビルド構成のソースブラウズ情報です。

ウィンドウで右クリックして表示されるコンテキストメニューで、ファイルフィルタとタイプフィルタを選択できます。

- 2 [ソースブラウザ] ウィンドウでブラウザ情報を表示するには、[ツール] > [オプション] > [プロジェクト] を選択して、オプション [ブラウザ情報を生成] を選択します。

## テキスト検索

エディタには、以下に示すさまざまな標準的検索機能が用意されています。

- [クイックサーチ] テキストボックス
- [検索] ダイアログボックス
- [置換] ダイアログボックス
- [ファイルから検索] ダイアログボックス
- [ファイル内で置換] ダイアログボックス
- [インクリメンタル検索] ダイアログボックス

ツールバーの [クイックサーチ] テキストボックスを使用するには、次の手順に従います。

- 1 検索する文字を入力して Enter キーを押します。
- 2 Esc キーを押すと検索をキャンセルします。アクティブなエディタウィンドウでテキストを検索する場合は、この方法が最も簡単です。

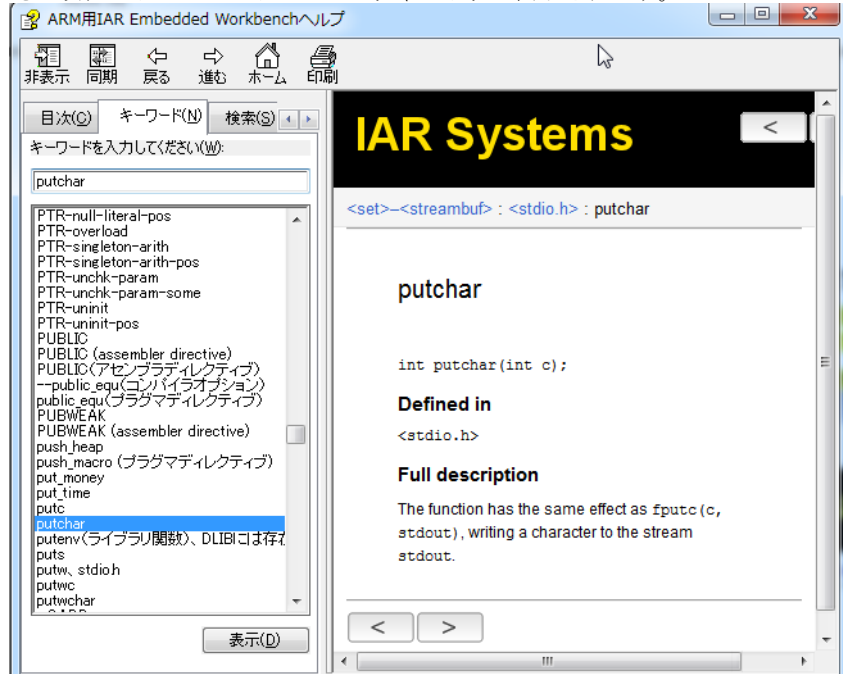
[検索]、[置換]、[ファイルから検索]、[ファイル内で置換]、[インクリメンタル検索] 機能を使用するには、次の手順に従います。

- 1 検索コマンドを使用する前に、[ツール] > [オプション] > [エディタ] を選択して、[ブックマークの表示] オプションが選択されているか確認します。
- 2 [編集] メニューから適切な検索コマンドを選択します。各検索機能の詳しい情報については、198 ページの [編集] メニューを参照してください。
- 3 左端に表示される青い旗のアイコンを削除するには、[ファイルで検索] ウィンドウで右クリックしてコンテキストメニューから [すべてをクリア] を選択します。

## オンラインヘルプのリファレンス情報へのアクセス

ライブラリ関数、拡張キーワード、組み込み関数などの構文を知る必要がある場合は、エディタウィンドウで関数名を選択して F1 を押します。

その項目のドキュメントがヘルプウィンドウに表示されます。



## エディタについてのリファレンス情報

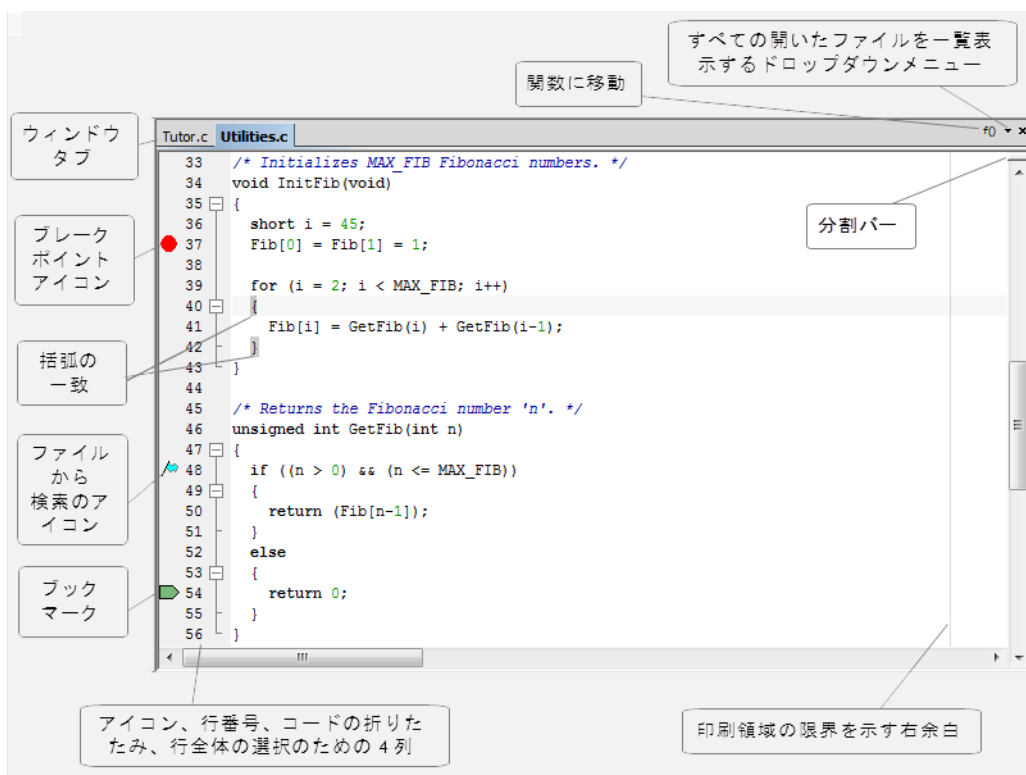
リファレンス情報：

- 152 ページの [エディタ] ウィンドウ
- 161 ページの [検索] ダイアログボックス
- 163 ページの [ファイルで検索] ウィンドウ
- 164 ページの [置換] ダイアログボックス
- 165 ページの [ファイルから検索] ダイアログボックス
- 167 ページの [ファイル内で置換] ダイアログボックス
- 170 ページの [インクリメンタル検索] ダイアログボックス
- 171 ページの [宣言] ウィンドウ

- 172 ページの [曖昧な定義] ウィンドウ
- 173 ページの [参照] ウィンドウ
- 174 ページの [ソースブラウザ] ウィンドウ
- 177 ページの [ソース参照ログ] ウィンドウ
- 179 ページの [ファイルの曖昧さの解決] ダイアログボックス
- 179 ページの [コールグラフ] ウィンドウ
- 180 ページの [テンプレート] ダイアログボックス
- 181 ページのエディタのショートカットキー操作のまとめ

## [エディタ] ウィンドウ

エディタウィンドウは、IDE でテキストファイルを開いたり作成すると表示されます。





[ファイル] メニューからファイルを選択するか、[ワークスペース] ウィンドウでファイルをダブルクリックすることによって、1 つ以上のテキストファイルを開くことができます。エディタウィンドウの右上にある ドロップダウンメニューから、開いているすべてのファイルを選択できます。複数のエディタウィンドウを同時に開いておくことができます。

ソースコードファイルと HTML ファイルがエディタウィンドウに表示されます。開いている HTML 文書では、HTML ファイルへのハイパーリンクは通常の Web ブラウザと同じように機能します。eww ワークスペースファイルへのリンクは、IDE でワークスペースを開くことや、現在開いているワークスペースおよび HTML ドキュメントを閉じることができます。

ソースファイルを印刷する場合には、[ツール] > [オプション] > [エディタ] を選択し、[行番号の表示] オプションを有効にしておくくと便利です。

エディタウィンドウは常にドッキングされていて、サイズと位置は他の開いているウィンドウに応じて変化します。

エディタの使用について詳しくは、140 ページの *ファイルの編集* と 148 ページの *プログラミングのサポート* を参照してください。

## 相対ソースファイルパス

IDE は相対ソースファイルパスを部分的にサポートしています。

ソースファイルがプロジェクトファイルディレクトリかプロジェクトファイルディレクトリ内のサブディレクトリにある場合、IDE はプロジェクトファイルとの相対パスを使用してソースファイルにアクセスします。

## ドキュメントのコメント

// (C++) または /\* (C および C++) から始まる通常のコメントに加えて、エディタは、/\*\*, /\*!, /// or //! で始まる *ドキュメントのコメント* もサポートします。エディタは通常のコメントからドキュメントのコメントを識別できます。デフォルトでは、エディタはコメントに 2 つの種類のカラーを割り当てます。

ドキュメントのコメント内で、エディタは ¥ または @ で始まるキーワードを識別できます。キーワードが **doxygen** キーワードとして識別される場合、デフォルトでは残りのコメントに異なるカラーが使用されます。キーワードが識別されない場合（存在しないまたはスペルが間違っている）は、デフォルトでは、正しい **doxygen** キーワードとは異なるカラーになります。

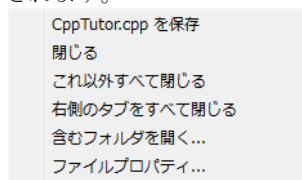
ドキュメントのコメントは、変数と関数のツールチップとパラメータのヒントに表示されます。

## ウィンドウのタブ、タブグループ、タブのコンテキストメニュー

開いているファイルの名前がタブに表示されます。複数のファイルを開いている場合、それらのファイルはタブグループとして編成されます。表示するファイルに対応するタブをクリックします。ファイルが最後に保存された後に変更された場合、たとえば「Utilities.c \*」のように、タブのファイル名の末尾にアスタリスクが表示されます。ファイルがリードオンリーの場合、錠前のアイコンがタブに表示されます。

ファイルがアクティブなプロジェクトのメンバでない場合、タブのツールチップにはフルパスと備考が表示されます。

エディタウィンドウのタブを右クリックするとコンテキストメニューが表示されます。



以下のコマンドがあります。

### ファイルの保存

ファイルを保存します。

### 閉じる

ファイルを閉じます。

### これ以外すべて閉じる

現在のタブ以外のすべてのタブを閉じます。

### 右側のタブをすべて閉じる

現在のタブに右側にあるタブをすべて閉じます。

### ファイルの場所を開く

選択したファイルが存在するディレクトリを表示するファイルエクスプローラを開きます。

### ファイルのプロパティ

標準のファイルプロパティダイアログボックスを表示します。

## 複数のエディタウィンドウと分割バー

同時に複数のエディタウィンドウを表示できます。[ウィンドウ] メニューのコマンドを使用すると、エディタウィンドウをペインに分割して、複数のエ

エディタウィンドウを開くことができます。エディタウィンドウ間でファイルを移動するコマンドもあります。

【ウィンドウ】メニューの各コマンドについて詳しくは、215 ページの 【ウィンドウ】メニューを参照してください。

## 関数に移動



エディタウィンドウの右上隅にある【関数に移動】ボタンをクリックして、C/C++ エディタウィンドウで使用されているすべての関数を一覧表示することができます。

```
Tutor.h
#include "Tutor.h"
NextCounter
void NextCounter()
DoForegroundProcess
void DoForegroundProcess()
main
void main()
```

検索する関数の名前を入力してリストをフィルタします。エディタウィンドウに表示する関数の名前をクリックします。

コンテキストメニュー

以下のコンテキストメニューがあります。

切り取り(T)
コピー(C)
貼り付け(P)
語句の入力補完(M)
コードの入力補完(D)
パラメータのヒント(H)
括弧のマッチング(H)
折り畳みを切り替え(O)
テンプレートの挿入(I) ▶
ヘッダ/ソースファイルを開く (S)
'main'の定義に移動(D)
'main'の宣言に移動(D)
'main'へのすべての参照を検索(R)
mainへのすべてのコール元を検索
mainからのすべてのコール先を検索
トレースを検索
ブレークポイントの切り替え(A) (コード)
ブレークポイントの切り替え(A) (ログ)
ブレークポイントの切り替え(A) (トレース開始)
ブレークポイントの切り替え(A) (トレース停止)
ブレークポイントの有効化/無効化(N)
'main'のデータブレークポイントを設定
'main'のデータログブレークポイントを設定
カーソル位置にPCを移動
クイックウォッチ(Q): 'main'
ウォッチに追加(W): 'main'
ライブウォッチへ追加(L): 'main'
PCへ移動(V)
カーソルまで実行(R)
文字エンコード ▶
オプション(O)...

このメニューの内容は、デバッガが起動中かどうか、および使用している C-SPY ドライバによって異なります。通常は、このメニューで他のブレークポイントタイプが使用できることがあります。利用できるブレークポイントの詳細については、『*ARM 用 C-SPY® デバッガガイド*』を参照してください。

以下のコマンドがあります。

### 切り取り、コピー、貼り付け

Windows 標準のコマンド。

### 語句の入力補完

入力内容に応じて、エディタドキュメントの他の部分の内容から入力語を推測して補完します。

### コードの入力補完

..`->`、または..`::`の後に挿入ポイントを配置したり、これらの文字の前にクラスやオブジェクト名があるときに、クラスで使用可能なシンボルの一覧を表示します。詳細については、143 ページの *コードの入力補完* を参照してください。

### パラメータのヒント

パラメータを、入力した関数のパラメータリストのツールチップ情報として提示します。ある関数のオーバーロードされたバージョンがいくつかある場合、ツールチップの矢印をクリックすると使用するバージョンを選択できます。詳細については、144 ページの *パラメータのヒント* を参照してください。

### 括弧のマッチング

挿入ポイントの直近の括弧内のテキストをすべて選択します。すでに選択されている場合は、その外側の次の括弧まで選択範囲を拡大します。外側に括弧がない場合は、ビープ音を再生します。

### 折り畳みを切り替え

アクティブなプロジェクトのすべてのコードの折り目を展開したり折りたたみます。

### テンプレートの挿入

挿入ポイントの位置に挿入するコードテンプレートを選択できるリストを、エディタウィンドウで表示します。選択したコードテンプレートでフィールドへの入力が必要な場合は、**[テンプレート]** ダイアログボックスが表示されます。このダイアログボックスの詳細な情報については、180 ページの *[テンプレート] ダイアログボックス* を参照してください。コードテンプレートの使用方法については、144 ページの *コードテンプレートの使用と追加* を参照してください。

### "header.h" を開く

"header.h" という名前のヘッダファイルをエディタウィンドウで開きます。同じ名前を持つヘッダファイルが複数あって、IDE が依存情報にアクセスできない場合、**【ファイルの曖昧さの解決】** ダイアログボックスが表示されます（179 ページの **【ファイルの曖昧さの解決】** ダイアログボックスを参照）。このメニューコマンドは、コンテキストメニューを表示したときに挿入ポイントが `#include` 行にある場合にだけ使用できます。

### ヘッダ / ソースファイルを開く

現在のファイルと同じベース名を持つヘッダまたはソースコードファイルを開きます。コマンド実行時に対象ファイルが開かれていない場合は、そのファイルを開きます。このメニューコマンドは、コンテキストメニューを表示したときに挿入ポイントが `#include` 行を除く任意の行にある場合に使用できます。このコマンドは、**【ファイル】 > 【開く】** メニューから選択することもできます。

### シンボル定義に移動

シンボルの定義にカーソルを置きます。ソースコードに定義がない場合、代わりに最初の宣言が使用されます。複数の定義が見つかった場合、それらは **【曖昧な定義】** ウィンドウに一覧表示されます。172 ページの **【曖昧な定義】** ウィンドウを参照してください。

### シンボル宣言に移動

宣言が 1 つだけ見つかった場合、このコマンドはカーソルをシンボルの宣言に置きます。複数の宣言が見つかった場合、これらは **【宣言】** ウィンドウに一覧表示されます。

### シンボルへのすべての参照の検索

参照は **【参照】** ウィンドウに一覧表示されます。

### シンボルへのすべてのコールの検索

**【コールグラフ】** ウィンドウが開きます。ここでは、選択した関数を呼び出すプロジェクト内のあらゆる関数が表示されます（179 ページの **【コールグラフ】** ウィンドウを参照）。このコマンドが無効になっている場合、エディタウィンドウで関数を必ず選択してください。

### シンボルからのすべてのコールの検索

**【コールグラフ】** ウィンドウを開きます。ここでは、選択した関数から呼び出されるプロジェクト内のすべての関数が表示されます（179 ページの **【コールグラフ】** ウィンドウを参照）。このコマンドが無効になっている場合、エディタウィンドウで関数を必ず選択してください。

### トレースを検索

指定の場所（ソースコードのカーソルの位置）に該当する箇所があるか【トレース】ウィンドウの内容を検索し、結果を【トレースを検索】ウィンドウに表示します。このメニューコマンドでは、使用する C-SPY ドライバでトレースがサポートされている必要があります（『ARM 用 C-SPY® デバッグガイド』参照）。

### ブレークポイントの切替え（コード）

ソースウィンドウで、カーソルを含む、または直近の文か命令で、コードブレークポイントを設定 / 解除します。コードブレークポイントについては、『ARM 用 C-SPY® デバッグガイド』を参照してください。

### ブレークポイントの切替え（ログ）

ソースウィンドウで、カーソルを含む、または直近の文か命令で、ログブレークポイントを設定 / 解除します。ログブレークポイントについては、『ARM 用 C-SPY® デバッグガイド』を参照してください。

### ブレークポイントの切替え（トレース開始）

トレース開始ブレークポイントを切替えます。ブレークポイントがトリガされると、トレースデータの収集が始まります。トレース開始ブレークポイントの詳細については、『ARM 用 C-SPY® デバッグガイド』を参照してください。このメニューコマンドは、使用している C-SPY ドライバでトレースがサポートされている場合にのみ使用できます。

### ブレークポイントの切替え（トレース停止）

トレース停止ブレークポイントを切替えます。ブレークポイントがトリガされると、トレースデータの収集が停止します。トレース停止ブレークポイントについては、『ARM 用 C-SPY® デバッグガイド』を参照してください。このメニューコマンドは、使用している C-SPY ドライバでトレースがサポートされている場合にのみ使用できます。

### ブレークポイントの有効化 / 無効化

ブレークポイントの有効（実際には削除せず、後で再度使用できる状態にする）と有効を切り替えます。

### '変数'に対するデータブレークポイントの設定

静的変数のデータログブレークポイントを切り替えます。使用している C-SPY ドライバでサポートされていることが必要です。データブレークポイントについては、『ARM 用 C-SPY® デバッグガイド』を参照してください。

### '変数'のデータログブレイクポイントの設定

静的変数のデータログブレイクポイントを切り替えます。使用している C-SPY ドライバでサポートされていることが必要です。このウィンドウで設定するブレイクポイントは、リードとライトの両方のアクセスでトリガされます。これを変更するには、**[ブレイクポイント]** ウィンドウを使用します。データログおよびデータログブレイクポイントについては、『*ARM 用 C-SPY® デバッグガイド*』を参照してください。

### ブレイクポイントの編集

**[ブレイクポイントの編集]** ダイアログボックスが表示され、ソースコード行の挿入ポイントがある場所で使用可能なブレイクポイントを編集できます。複数のブレイクポイントが行にある場合、使用可能なすべてのブレイクポイントの一覧を示すサブメニューがその行に表示されます。

### 次の実行文の設定

コードを実行せずに、選択した文か命令の位置にプログラムカウンタを設定します。このコマンドは、デバッグ使用時にだけ使用できます。詳細については、『*ARM 用 C-SPY® デバッグガイド*』を参照してください。

### クイック ウォッチに追加：シンボル

**[クイック ウォッチ]** ウィンドウを開いて、シンボルを追加します (『*ARM 用 C-SPY® デバッグガイド*』参照)。このコマンドは、デバッグ使用時にだけ使用できます。

### ウォッチへ追加：シンボル

**[ウォッチ]** ウィンドウを開いて、シンボルを追加します。このコマンドは、デバッグ使用時にだけ使用できます。

### ライブウォッチに追加：シンボル

**[ライブ ウォッチ]** ウィンドウを開いて、シンボルを追加します (『*ARM 用 C-SPY® デバッグガイド*』参照)。このコマンドは、デバッグ使用時にだけ使用できます。

### PC へ移動

カーソルを、エディタウィンドウで現在の PC 位置に移動します。このコマンドは、デバッグ使用時にだけ使用できます。

### カーソルまで実行

現在の文や命令から、カーソルのある文または命令まで実行します。このコマンドは、デバッグ使用時にだけ使用できます。



## 文字エンコーディング

指定された文字エンコーディングに従ってソースファイルを解釈します。以下から選択します。

システム (Windows の設定を使用)

西ヨーロッパ言語

UTF-8

日本語 (Shift-JIS)

簡体字中国語 (GB2312)

繁体字中国語 (Big5)

韓国語 (統一ハングルコード)

アラビア語

バルト言語

中央ヨーロッパ言語

ギリシャ語

ヘブライ語

ロシア語

タイ語

ベトナム語

UTF-8 に変換 (ドキュメントを UTF-8 に変換)

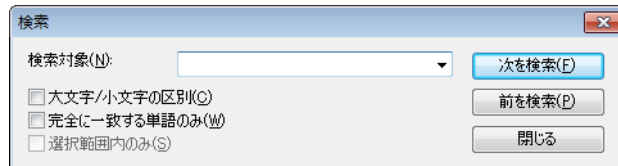
[文字エンコーディングの自動検出] オプションで正しいエンコーディングが判別できなかったり、オプションが選択解除されている場合は、これらの設定のいずれかを使用します。ファイルのエンコーディングについて詳しくは、59 ページの [エディタ] オプションを参照してください。

## オプション

[IDE オプション] ダイアログボックスを表示して、212 ページの [ツール] メニューを参照してください。

## [検索] ダイアログボックス

[検索] ダイアログボックスは [編集] メニューからアクセスできます。



エディタウィンドウで検索した場合は、[メモリ] ウィンドウで検索した場合に比べてダイアログボックスの内容が異なることがあります。このスクリーンショットは、エディタウィンドウで検索した時のダイアログボックスを示しています。

## 検索テキスト

検索するテキストを指定します。古い検索文字列を使用するには、ドロップダウンリストを使用します。

[メモリ] ウィンドウで検索するとき、検索する値は、複数のユニットサイズを表示する必要があります。たとえば、[メモリ] ウィンドウで [2つのユニット] サイズを使用しているときは、検索値を複数の2バイトにする必要があります。

## 大文字 / 小文字の区別

指定されたテキストの大文字と小文字が完全に一致するものだけを検索します。このオプションを指定しない場合は、int を検索すると、INT、Int も検索されます。このオプションは、エディタウィンドウでの検索の実行時にだけ使用できます。

## 完全に一致する単語のみ

単語として一致する箇所だけを検索します。このオプションを指定しない場合は、int を検索すると、print、sprintf も検索されます。このオプションは、エディタウィンドウでの検索の実行時にだけ使用できます。

## 16 進数値を検索

指定した16進数値を検索します。このオプションは、[メモリ] ウィンドウでの検索時にだけ使用できます。

## 選択範囲内のみ

検索処理を選択した行（エディタウィンドウで検索する場合）または選択したメモリエリア（[メモリ] ウィンドウで検索する場合）に限定します。このオプションは、ダイアログボックスを開く前に選択を行った場合にのみ有効になります。

## 次を検索



指定したテキストの次の一致箇所を検索します。

## 前を検索

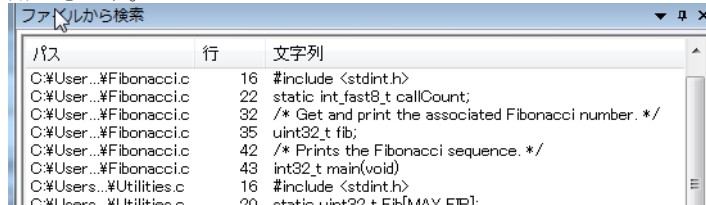
指定したテキストの前の一致箇所を検索します。

## 停止

実行中の検索を停止します。このボタンは、[メモリ] ウィンドウでの検索時にだけ使用できます。

## 【ファイルで検索】 ウィンドウ

【ファイルから検索】 ウィンドウは、【表示】 > 【メッセージ】 を選択すれば使用できます。

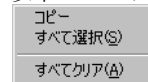


このウィンドウには、【編集】 > 【検索と置換】 > 【ファイルから検索】 コマンドの出力が表示されます。デフォルトでは、このウィンドウは他のメッセージウィンドウとグループ化されて表示されます。

このウィンドウでメッセージをダブルクリックすると、対応するファイルが開き、挿入ポイントが正しい箇所に表示されます。ソースの位置は、青い旗のアイコンで強調表示されます。【編集】 > 【次のエラー/タグ】 を選択するか、F4 を押してシーケンスの次の項目にジャンプします。

### コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

#### コピー

ウィンドウの選択された内容をコピーします。

#### すべて選択

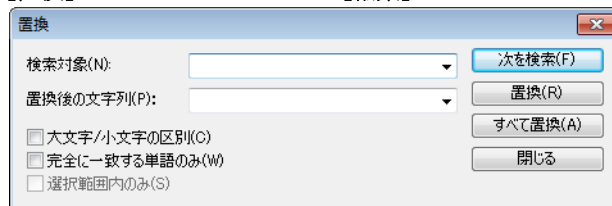
ウィンドウの内容を選択します。

#### すべてをクリア

ウィンドウの内容とエディタウィンドウの左端にある青い旗のアイコンをすべてを削除します。

## 〔置換〕 ダイアログボックス

〔置換〕 ダイアログボックスは〔編集〕メニューからアクセスできます。



エディタウィンドウで検索した場合は、〔メモリ〕ウィンドウで検索した場合に比べてダイアログボックスの内容が異なります。

### 検索テキスト

検索するテキストを指定します。古い検索文字列を使用するには、ドロップダウンリストを使用します。

### 置換後の文字列

一致する箇所と置換するテキストを指定します。古い検索文字列を使用するには、ドロップダウンリストを使用します。

### 大文字 / 小文字の区別

指定されたテキストの大文字と小文字が完全に一致するものだけを検索します。このオプションを指定しない場合は、`int` を検索すると、`INT`、`Int` も検索されます。このオプションは、エディタウィンドウでの検索の実行時にだけ使用できます。

### 完全に一致する単語のみ

単語として一致する箇所だけを検索します。このオプションを指定しない場合は、`int` を検索すると、`print`、`sprintf` も検索されます。このオプションは、エディタウィンドウ内で検索するときだけに使用できます。

### 16 進数値を検索

指定した 16 進数値を検索します。このオプションは、〔メモリ〕ウィンドウでの検索時にだけ使用できます。

### 選択範囲内のみ

検索処理を選択した行（エディタウィンドウで検索する場合）または選択したメモリエリア（〔メモリ〕ウィンドウで検索する場合）に限定します。このオプションは、ダイアログボックスを開く前に選択を行った場合にのみ有効になります。

**次を検索**

指定したテキストの次の一致箇所を検索します。

**置換**

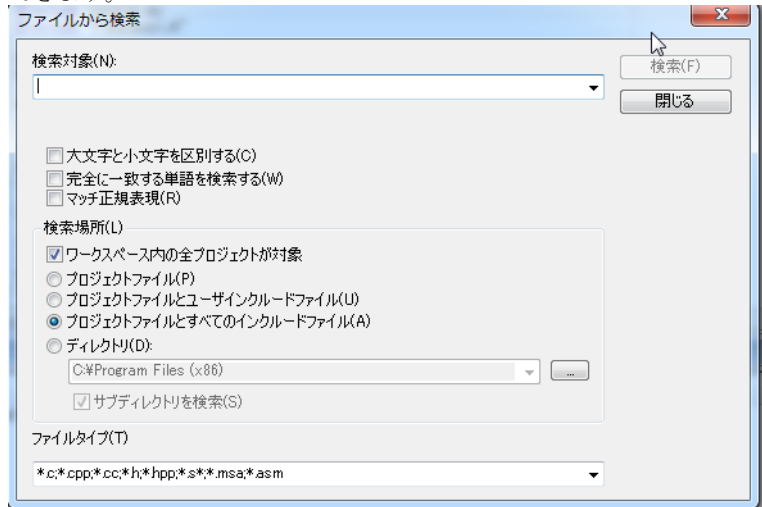
一致箇所のテキストを指定テキストに置換します。

**すべて置換**

現在のエディタウィンドウで検索テキストに一致する箇所をすべて置換します。

**【ファイルから検索】ダイアログボックス**

【ファイルから検索】ダイアログボックスは、【編集】メニューからアクセスできます。



このダイアログボックスを使用して、ファイル内で文字列を検索します。

検索結果は、【ファイルから検索】メッセージウィンドウ（【表示】メニューからアクセス）で表示されます。【編集】>【次のエラー/タグ】コマンドを選択するか、【ファイルから検索】メッセージウィンドウでメッセージをダブルクリックして、一致箇所に移動することができます。対応するファイルがエディタウィンドウで表示され、一致箇所の最初の位置に挿入ポイントが設定されます。左端に表示される青い旗は、検索した文字列を含む行を示します。

## 検索テキスト

検索する文字列または正規表現を指定します。古い検索文字列 / 式を使用するには、ドロップダウンリストを使用します。以下の条件を必要なだけ使用して、検索を絞り込むことができます。

### 大文字 / 小文字の区別

指定されたテキストの大文字と小文字が完全に一致するものだけを検索します。このオプションを指定しない場合は、`int` を検索すると、`INT`、`Int` も検索されます。

### 完全に一致する単語のみ

単語として独立した文字列のみを検索します（ニーモニック）。このオプションを指定しない場合は、`int` を検索すると、`print`、`sprintf` なども検索されます。

### マッチ正規表現

検索文字列を正規表現として解釈します。つまり、Perl プログラミング言語の標準に従う必要があります。

## 検索場所

検索するファイルを以下から選択します。

### ワークスペース内の全プロジェクトが対象

アクティブなプロジェクトだけでなく、ワークスペースの全プロジェクトが検索されます。

### プロジェクトファイル

明示的にプロジェクトに追加した全ファイルが検索されます。

### プロジェクトファイルとユーザインクルードファイル

プロジェクトに明示的に追加したすべてのファイルと、それらのファイルに含まれるすべてのファイル（IAR Embedded Workbench のインストールディレクトリ内のインクルードファイルを除く）に対して検索が実行されます。

### プロジェクトファイルとすべてのインクルードファイル

プロジェクトに明示的に追加したすべてのファイルと、それらに含まれるすべてのファイルが検索されます。

### ディレクトリ

指定したディレクトリが検索されます。最近検索した場所が、ドロップダウンリストに保存されます。参照ボタンを使用して選択することもできます。

### サブディレクトリを検索

指定したディレクトリと、そのサブディレクトリがすべて検索されます。

### ファイルタイプ

検索するファイルのタイプを選択するフィルタ。フィルタはすべての【検索場所】設定に適用されます。ドロップダウンリストからフィルタを選択します。テキストフィールドは編集可能で、自分のフィルタを追加することができます。フィルタのゼロ文字以上の任意の文字を示すには \* を、任意の 1 文字を示すには ? を使用します。

### 停止

実行中の検索を停止します。このボタンは、検索中にだけ使用できます。

## 【ファイル内で置換】ダイアログボックス

【ファイル内で置換】ダイアログボックスは、【編集】メニューから使用できます。



このダイアログボックスを使用して、指定した文字列を複数のテキストファイル内で検索し、別の文字列と置換します。

置換の結果は、【ファイルから検索】メッセージウィンドウ（【表示】メニューからアクセス）で表示されます。【編集】>【次のエラー/タグ】コマ

ンドを選択するか、**[ファイルから検索]** メッセージウィンドウでメッセージをダブルクリックして、一致箇所に移動することができます。対応するファイルがエディタウィンドウで表示され、一致箇所の最初の位置に挿入ポイントが設定されます。左端に表示される青い旗は、検索した文字列を含む行を示します。

## 検索テキスト

検索して置換する文字列または正規表現を指定します。古い検索文字列 / 式を使用するには、ドロップダウンリストを使用します。以下の条件を必要に応じて使用して、検索を絞り込むことができます。

### 大文字 / 小文字の区別

指定されたテキストの大文字と小文字が完全に一致するものだけを検索します。このオプションを指定しない場合は、`int` を検索すると、`INT`、`Int` も検索されます。

### 完全に一致する単語のみ

単語として独立した文字列のみを検索します (ニーモニック)。このオプションを指定した場合、`int` を検索すると、`print`、`sprintf` などは検索されません。

### マッチ正規表現

検索文字列を正規表現として解釈します。つまり、Perl プログラミング言語の標準に従う必要があります。

## 置換後の文字列

元の文字列と置換する文字列を指定します。以前の置換文字列を使用するには、ドロップダウンリストを使用します。

## 検索場所

検索するファイルを以下から選択します。

### ワークスペース内の全プロジェクトが対象

アクティブなプロジェクトだけでなく、ワークスペースの全プロジェクトが検索されます。

### プロジェクトファイル

明示的にプロジェクトに追加した全ファイルが検索されます。

### プロジェクトファイルとユーザインクルードファイル

プロジェクトに明示的に追加したすべてのファイルと、それらのファイルに含まれるすべてのファイル (IAR Embedded Workbench のインストールディレクトリ内のインクルードファイルを除く) に対して検索が実行されます。



**プロジェクトファイルとすべてのインクルードファイル**

プロジェクトに明示的に追加したすべてのファイルと、それらに含まれるすべてのファイルが検索されます。

**ディレクトリ**

指定したディレクトリが検索されます。最近検索した場所が、ドロップダウンリストに保存されます。参照ボタンを使用して選択することもできます。

**サブディレクトリを検索**

指定したディレクトリと、そのサブディレクトリがすべて検索されます。

**ファイルタイプ**

検索するファイルのタイプを選択するフィルタ。フィルタはすべての**【検索場所】**設定に適用されます。ドロップダウンリストからフィルタを選択します。テキストフィールドは編集可能で、自分のフィルタを追加することができます。フィルタのゼロ文字以上の任意の文字を示すには\*を、任意の1文字を示すには?を使用します。

**停止**

実行中の検索を停止します。このボタンは、検索中にだけ使用できます。

**閉じる**

ダイアログボックスを閉じます。継続中の検索を先に停止する必要があります。

**次を検索**

指定した検索文字列に一致する次の箇所を検索します。

**置換**

見つかった文字列を置換して、指定した検索文字列の次の一致箇所を検索します。

**すべて置換**

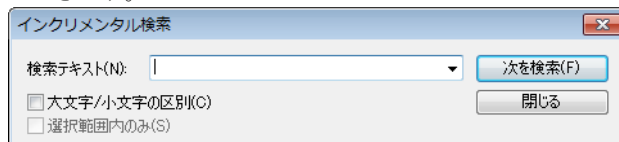
すべてのファイルを保存して、検索文字列に一致するすべての文字列を置換します。

**ファイルをスキップ**

現在のファイル内の一致箇所をスキップします。

## 【インクリメンタル検索】ダイアログボックス

【インクリメンタル検索】ダイアログボックスは【編集】メニューからアクセスできます。



このダイアログボックスを使用して、検索文字列を徐々に調整または拡張します。

### 検索テキスト

検索する文字列を入力します。検索は、挿入ポイントの位置（開始位置）から実行されます。検索文字列に文字を追加したり削除するたびに、検索結果がすぐ変わります。文字を1つ削除すると、開始地点から検索がもう一度スタートします。

【インクリメンタル検索】ダイアログボックスを表示したときにエディタウィンドウで文字列を選択している場合は、その文字列が【検索テキスト】に表示されます。

古い検索文字列を使用するには、ドロップダウンリストを使用します。

### 大文字 / 小文字の区別

指定されたテキストの大文字と小文字が完全に一致するものを検索します。このオプションを指定しない場合は、int を検索すると、INT、Int も検索されます。

### 次を検索

現在の検索文字列に一致する次の箇所を検索します。【次を検索】ボタンをクリックしたときに【検索テキスト】テキストボックスが空白の場合は、検索文字列がドロップダウンリストから自動的に選択されます。この文字列を検索するには、【次を検索】をクリックします。

### 閉じる

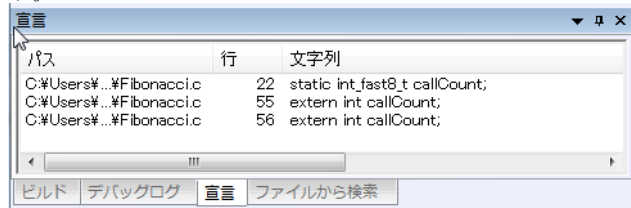
ダイアログボックスを閉じます。

### 選択範囲内のみ

検索処理を選択した行に限定します。このオプションは、ダイアログボックスを開く前に複数の行が選択された場合にのみ使用可能です。

## 【宣言】 ウィンドウ

【宣言】 ウィンドウは、[表示] > [ソースブラウザ] を選択すると使用できます。



このウィンドウには、エディタウィンドウのコンテキストメニューにある**【宣言に移動】** コマンドの結果が表示されます。

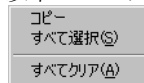
デフォルトでは、このウィンドウは他のメッセージウィンドウとグループ化されて表示されます。

特定のシンボルへの参照を検索して一覧表示するには、エディタウィンドウでシンボルを選択して右クリックし、コンテキストメニューから**【宣言に移動】** を選択します。すべての宣言が**【宣言】** ウィンドウに一覧表示されます。

このウィンドウでメッセージをダブルクリックすると、対応するファイルが開き、挿入ポイントが正しい箇所に表示されます。**【編集】** > **【次のエラー/タグ】** を選択するか、F4 を押してシーケンスの次の項目にジャンプします。

### コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

#### コピー

ウィンドウの内容をコピーします。

#### すべて選択

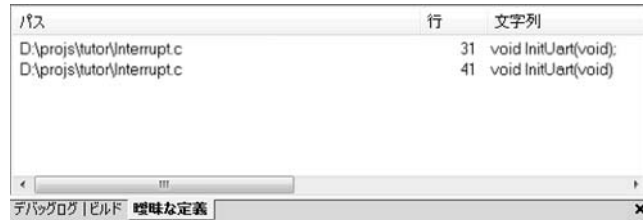
ウィンドウの内容を選択します。

#### すべてをクリア

ウィンドウの内容を削除します。

## 「曖昧な定義」ウィンドウ

「曖昧な定義」ウィンドウは「表示」>「ソースブラウザ」を選択すると使用できます。



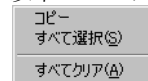
このウィンドウには、エディタウィンドウのコンテキストメニューで「定義に移動」コマンドを実行した結果が表示されます（ソースブラウザで複数の定義が見つかった場合）。

デフォルトでは、このウィンドウは他のメッセージウィンドウとグループ化されて表示されます。

このウィンドウでメッセージをダブルクリックすると、対応するファイルが開き、挿入ポイントが正しい箇所に表示されます。「編集」>「次のエラー/タグ」を選択するか、F4を押してシーケンスの次の項目にジャンプします。

### コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

#### コピー

ウィンドウの内容をコピーします。

#### すべて選択

ウィンドウの内容を選択します。

#### すべてをクリア

ウィンドウの内容を削除します。

## 【参照】 ウィンドウ

【参照】 ウィンドウは、【表示】 > 【ソースブラウザ】 を選択すると使用できます。



このウィンドウには、エディタウィンドウのコンテキストメニューにある【すべての参照を検索】 コマンドの結果が表示されます。

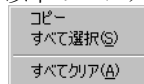
デフォルトでは、このウィンドウは他のメッセージウィンドウとグループ化されて表示されます。

特定のシンボルへの参照を検索して一覧表示するには、エディタウィンドウでシンボルを選択して右クリックし、コンテキストメニューから【すべての参照を検索】を選択します。すべての参照は【参照】 ウィンドウに一覧表示されます。

このウィンドウでメッセージをダブルクリックすると、対応するファイルが開き、挿入ポイントが正しい箇所に表示されます。【編集】 > 【次のエラー/タグ】 を選択するか、F4 を押してシーケンスの次の項目にジャンプします。

### コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

#### コピー

ウィンドウの内容をコピーします。

#### すべて選択

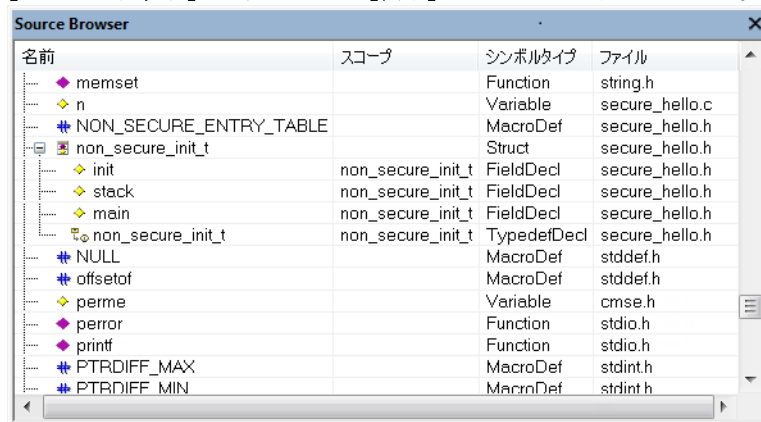
ウィンドウの内容を選択します。

#### すべてをクリア

ウィンドウの内容を削除します。

## 【ソースブラウザ】 ウィンドウ

【ソースブラウザ】 ウィンドウは【表示】メニューから利用できます。



このウィンドウには、アクティブなビルド構成で定義されているすべてのシンボルが、アルファベット順で階層表示されます。すなわち、ソースファイル内のシンボルについてソースブラウザの情報が利用でき、その構成のファイル部分が情報に含まれます。ソースブラウザの情報は、リンクされたライブラリ内のシンボルについては利用できません。

このウィンドウの使用方法について詳しくは、150 ページの [ブラウズ情報の表示](#) を参照してください。

### 表示エリア

表示エリアには、4 つの列があります。

**名前** プロジェクトで定義されているグローバルシンボルや関数の名前が表示されます。名前のない struct や union のような無名型の場合、定義されたファイル名および行番号に基づいて名前が決まります。これらの擬似名は角括弧で囲まれています。

**スコープ** 項目が属する範囲（名前空間およびクラス / 構造体）。















**シンボルタイプ** 各エレメントのシンボルの種類を表示します。

**ファイル** 項目の定義を含むファイル名（パスを除く）。

各列をソートするには、ヘッダをクリックします。

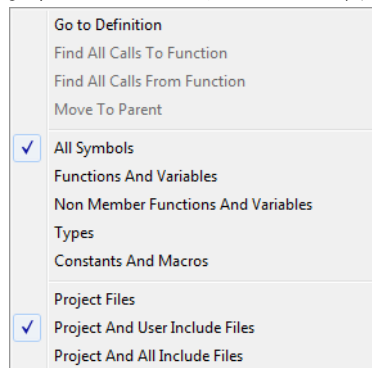
## シンボルの種類に使用されるアイコン

使用されるアイコンは次のとおりです。

	基底クラス
	クラス
	設定
	列挙型
	列挙定数
 (黄色のひし形)	構造体のフィールド
 (紫のひし形)	関数
	マクロ
	名前空間
	テンプレートクラス
	テンプレート関数
	タイプの定義
	共用体
 (黄色のひし形)	変数

## コンテキストメニュー

以下のコンテキストメニューが表示エリアで使用できます。



以下のコマンドがあります。

### 定義に移動

エディタウィンドウで、選択した項目の定義を表示します。

### すべてのコールを検索

[**コールグラフ**] ウィンドウが開きます。ここでは、選択した関数を呼び出すプロジェクト内のあらゆる関数が表示されます (179 ページの [**コールグラフ**] ウィンドウを参照)。子のコマンドを無効にした場合、[**ソースブラウザ**] ウィンドウで関数を必ず選択してください。

### すべてのコールを検索

[**コールグラフ**] ウィンドウを開きます。ここでは、選択した関数から呼び出されるプロジェクト内のすべての関数が表示されます (179 ページの [**コールグラフ**] ウィンドウを参照)。子のコマンドを無効にした場合、[**ソースブラウザ**] ウィンドウで関数を必ず選択してください。

### 親エレメントに移動

選択したエレメントがクラス、構造体、共用体、列挙型、名前空間のメンバーである場合に、このメニューコマンドを使用して、挿入ポイントをそのエレメントの親エレメントに移動することができます。

### すべてのシンボル

タイプフィルタ。プロジェクト内で定義されたすべてのグローバルシンボルや関数が表示されます。

### 関数と変数

タイプフィルタ。プロジェクト内で定義されたすべての関数や変数が表示されます。

### 非メンバ関数と変数

タイプフィルタ。クラスのメンバではない関数および変数がすべて表示されます。

### 型

タイプフィルタ。プロジェクト内で定義されたすべての型（構造体、クラスなど）が表示されます。

### 定数およびマクロ

タイプフィルタ。プロジェクト内で定義されたすべての定数やマクロが表示されます。



### プロジェクトファイル

ファイルフィルタ。プロジェクトに明示的に追加したすべてのファイル（それらのファイルでインクルードされるファイルを除く）内のシンボルが表示されます。

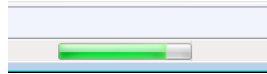
### プロジェクトとユーザインクルードファイル

ファイルフィルタ。プロジェクトに明示的に追加したすべてのファイルと、それらのファイルでインクルードされるすべてのファイル内（IAR Embedded Workbench のインストールディレクトリ内のインクルードファイルを除く）のシンボルが表示されます。

### プロジェクトとすべてのインクルードファイル

ファイルフィルタ。プロジェクトに明示的に追加したすべてのファイルと、それらのファイルでインクルードされるすべてのファイル内のシンボルが表示されます。

## プログレスバー

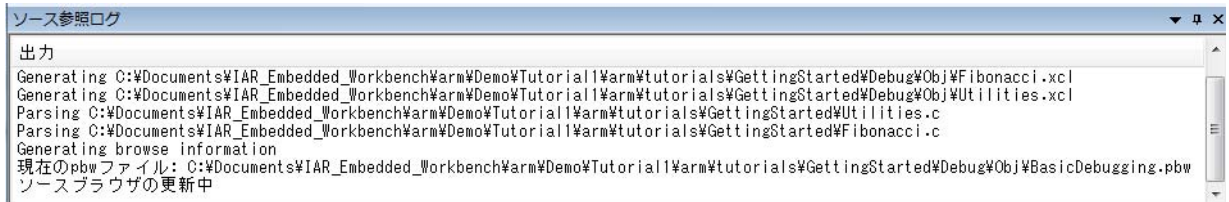


プロジェクトのソースブラウザ情報を作成中は、IDE ウィンドウのステータスバーに緑のプログレスバーが表示されます。このプログレスバーをクリックして、コメントのあるコンテキストメニューを開き、[ソース参照ログ] ウィンドウを開きます（177 ページの [ソース参照ログ] ウィンドウを参照）。

ソースブラウザで既知のエラーが発生すると、プログレスバーは赤色になります。

## [ソース参照ログ] ウィンドウ

[ソース参照ログ] ウィンドウは、[表示] > [メッセージ] を選択すると使用できます。



このウィンドウには、ソースブラウザの操作からの出力が表示されます。

## コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

### すべて

ソースブラウザから送信されたすべてのメッセージを表示します。これは主に IAR システムズの技術サポートへ提供するために役立ちます。

### メッセージ

ソースブラウザが行っていることまた解析中に発生したエラーについての情報を提供します。

### エラー

ソース参照中に受信したエラーのみを表示します。

### コピー

ウィンドウの内容をコピーします。

### すべて選択

ウィンドウの内容を選択します。

### すべてをクリア

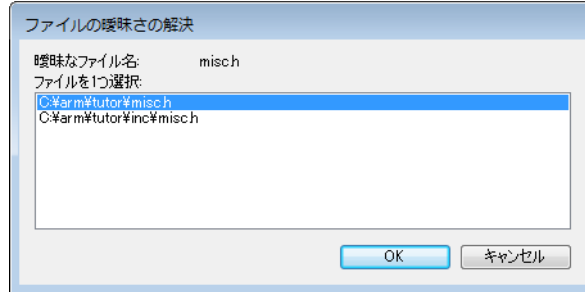
ウィンドウの内容をクリアします。

### ファイルへのライブログ

ログファイルにソース参照メッセージを書き込んで、ログにフィルターレベルを設定するためのコマンドで、サブメニューを表示します。

## 【ファイルの曖昧さの解決】 ダイアログボックス

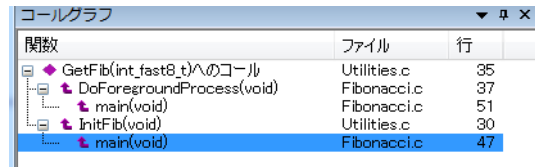
【ファイルの曖昧さの解決】 ダイアログボックスは、エディタが同じ名前のヘッダファイルを 1 つ以上見つけた時に表示されます。



このダイアログボックスには、エディタウィンドウのコンテキストメニューで **["header.h"] を開く** コマンドを選択して、IDE が依存情報にアクセスできないときに、複数のヘッダファイルが見つかった場合にそれらが一覧表示されます。

## 【コールグラフ】 ウィンドウ

【コールグラフ】 ウィンドウは、**[表示] > [ソースブラウザ] > [コールグラフ]** を選択すると使用できます。



このウィンドウには関数への呼出しまたは関数からの呼出しが表示されます。関数の呼出し間の移動に便利です。

コールグラフを表示するには、エディタウィンドウまたは **[ソースブラウザ]** ウィンドウで関数名を選択し、右クリックして コンテキストメニューから **[すべてのコール先を検索]** または **[すべてのコール元を検索]** を選択します。

このウィンドウでエントリをダブルクリックして、関数呼出し（呼出しがエントリに適切でない場合は定義）の位置に挿入ポイントを配置します。必要に応じて、エディタで呼出しを含むファイルが開きます。

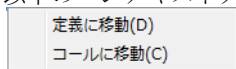
## 表示エリア

表示エリアには、選択した関数のコールグラフが表示され、各行に関数が1つ示されます。以下の列が使用できます。

関数	選択した関数のコールグラフが表示されます。最初に選択した関数、続いて呼び出される関数または呼び出す関数がすべて一覧表示されます。選択した関数を呼び出す関数には左矢印が、選択した関数によって呼び出される関数には右矢印がそれぞれ付きます。
ファイル	ソースファイル名。
行	呼出しの行番号。

## コンテキストメニュー

以下のコンテキストメニューがあります。



以下のコマンドがあります。

### 定義に移動

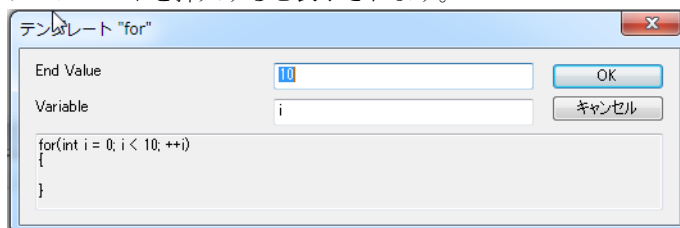
関数定義の位置に挿入ポイントを配置します。

### コールに移動

関数呼出しの位置に挿入ポイントを配置します。

## [テンプレート] ダイアログボックス

[テンプレート] ダイアログボックスは、フィールドへの入力が必要なコードテンプレートを挿入すると表示されます。



挿入したソースコードテンプレートで必要なフィールド入力を指定するには、このダイアログボックスを使用します。

**注：**この図は、for ループ用コードを自動挿入するためのデフォルトコードテンプレートの表示です。

## テキストフィールド

テキストフィールドの必要な入力内容を指定します。コードテンプレートの定義に応じたフィールドが表示されます。

## 表示エリア

表示エリアには、入力した値を使用してコードテンプレートから生成されるコードが表示されます。コードテンプレートの使用について詳しくは、144 ページの *コードテンプレートの使用と追加* を参照してください。

## エディタのショートカットキー操作のまとめ

エディタでは以下の 3 種類のショートカットキーを使用できます。

- 定義済みのショートカットキー。[IDE オプション] ダイアログボックスを使用して edit できます
- Scintilla エディタで提供されるショートカットキー
- カスタムのショートカットキー。[IDE オプション] ダイアログボックスを使用して追加できます

下表に、エディタの定義済みショートカットキーをまとめています。

### 挿入ポイントの移動

挿入ポイントの移動操作	キー
左側の   文字	←
右側の   文字	→
左側の   語	Ctrl + ←
右側の   語	Ctrl + →
左側の語句   字。mixedCaseName など、大文字小文字が混在する場合	Ctrl + Alt + ←
右側の語句   字。mixedCaseName など、大文字小文字が混在する場合	Ctrl + Alt + →
上に   行分移動	↑
下に   行分移動	↓
前のパラグラフに移動	Ctrl + Alt + ↑
次のパラグラフに移動	Ctrl + Alt + ↓
行の先頭まで移動	Home
行の最後まで移動	End

表 5: エディタで挿入ポイントを移動するショートカットキー

挿入ポイントの移動操作	キー
ファイルの先頭まで移動	Ctrl + Home
ファイルの最後まで移動	Ctrl + End

表 5: エディタで挿入ポイントを移動するショートカットキー (続き)

テキストの選択

テキストを選択するには、Shift を押して、挿入ポイントを移動するための対応するコマンドを押します。また、次のコマンドが使用できます。

選択操作	キー
列ベースのブロック	Shift + Alt + →

表 6: エディタでテキストを選択するためのショートカットキー

テキストのスクロール

スクロール操作	キー
上に 1 行 パラメータのヒントのテキストボックスで 使用すると、このショートカットによって 選択肢のひとつ上の行に移動します	Ctrl + ↑
下に 1 行 パラメータのヒントのテキストボックスで 使用すると、このショートカットによって 選択肢のひとつ下の行に移動します	Ctrl + ↓
上に 1 ページ	Page Up
下に 1 ページ	Page Down

表 7: エディタでスクロールするためのショートカットキー

その他のショートカットキー

説明	キー
パラメータのヒントのテキストボックスで 使用すると、このショートカットによって パラメータがテキストとしてソースコード に挿入されます	Ctrl + Enter
括弧のマッチング: 選択内容を、{}、[]、() の次のレベルのマッチングまで拡張します	Ctrl + B

表 8: その他のエディタのショートカットキー

説明	キー
括弧のマッチング: 選択内容を、{}、[]、() ()、<> の次のレベルのマッチングまで拡張します	Ctrl + Alt + B
括弧のマッチング: 選択内容を、{}、[]、() の次のレベルのマッチングまで縮小します	Ctrl + Shift + B
括弧のマッチング: 選択内容を、{}、[]、() ()、<> の次のレベルまで縮小します	Ctrl + Alt + Shift + B
選択したテキストを小文字に変更	Ctrl + u
選択したテキストを大文字に変更	Ctrl + U
コードの入力補完	Ctrl + スペース
語句の入力補完	Ctrl + Alt + スペース
テンプレートの挿入	Ctrl + Alt + V
パラメータのヒント	Ctrl + Shift + スペース
ズーム	マウスのホイール
ズームイン	Ctrl + テンキーの '+'
ズームアウト	Ctrl + テンキーの '-'
ノーマルズーム	Ctrl + テンキーの '/'

表 8: その他のエディタのショートカットキー (続き)

### Scintilla の追加ショートカットキー

説明	キー
ウィンドウの行を上または下にスクロール	Ctrl + ↑ Ctrl + ↓
長方形のブロックを選択して、そのサイズ を上または下に 1 行分、あるいは左または 右に 1 列分変更します	Shift + Alt + 矢印キー
挿入ポイントを 1 パラグラフ分、上または 下に移動	Ctrl + Alt + ↑ Ctrl + Alt + ↓
選択内容を 1 パラグラフ分、上または下に 拡張	Ctrl + Shift + Alt + ↑ Ctrl + Shift + Alt + ↓
挿入ポイントを 1 字分、左または右に移動	Ctrl + ← Ctrl + →
選択内容を 1 字分、左または右に拡張	Ctrl + Shift + ← Ctrl + Shift + →

表 9: Scintilla の追加ショートカットキー

説明	キー
選択内容を語句の次の最初または最後まで 拡張	Ctrl + Shift + Alt + ← Ctrl + Shift + Alt + →
その行にある最初の空白以外の文字に移動	Home
行頭に移動	Alt + Home
行頭まで選択	Shift + Alt + Home
長方形のブロックをページの最初または最 後まで選択	Shift + Alt + Page Up Shift + Alt + Page Down
次の語句の最初までを削除	Ctrl + Delete
前の語句の最初までを削除	Ctrl + Backspace
行末まで前方に削除	Ctrl + Shift + Delete
行末まで後方に削除	Ctrl + Shift + Backspace
ズームイン	Ctrl + Add (テンキーの +)
ズームアウト	Ctrl + Subtract (テンキーの -)
ズームの倍率を 100% に戻す	Ctrl + Divide (テンキーの /)
現在の行を切り取る	Ctrl + L
現在の行をコピー	Ctrl + Shift + T
現在の行を削除	Ctrl + Shift + L
選択内容を小文字に変換	Ctrl + U
選択内容を大文字に変換	Ctrl + Shift + U

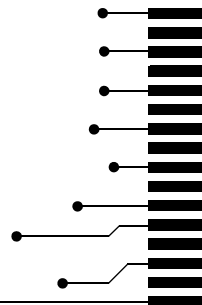
表 9: Scintilla の追加ショートカットキー (続き)



## パート 2. リファレンス情報

このパートは以下の章で構成されます。

- 製品ファイル
- メニューリファレンス
- 一般オプション
- コンパイラオプション
- アセンブラオプション
- 出力コンバータオプション
- カスタムビルドオプション
- ビルドアクションオプション
- リンカオプション
- ライブラリビルダオプション





# 製品ファイル

- インストール先ディレクトリ構成
- 製品ディレクトリ構成
- さまざまな設定ファイル
- ファイルタイプ

---

## インストール先ディレクトリ構成

以下のトピックを解説します：

- ルートディレクトリ
- arm ディレクトリ
- common ディレクトリ
- install-info ディレクトリ

インストール手順を実行すると、IAR システムズの開発ツールで使用する各種ファイルを含む複数のディレクトリが作成されます。以下では、各ディレクトリにデフォルトで含まれるファイルについて説明します。

### ルートディレクトリ

デフォルトのインストールルートルートディレクトリは通常 `x:\Program Files\IAR Systems\Embedded Workbench N.n\` で、`x` は Microsoft Windows がインストールされているドライブです。`N.n` の最初の数字は、IAR Embedded Workbench 共有コンポーネントのバージョン番号です。

このバージョン番号はお使いの IAR Embedded Workbench 製品のバージョン番号とは異なります。IDE と製品のバージョン番号を検索するには、90 ページの製品情報ダイアログボックスを参照してください。

arm ディレクトリ

arm ディレクトリには、製品固有のサブディレクトリがすべて含まれています。

ディレクトリ	説明
arm¥bin	コンパイラ、アセンブラ、リンカ、ライブラリツール、C-SPY®ドライバなどの、Arm 固有のコンポーネント用の実行可能ファイルが含まれます。
arm¥config	開発環境やプロジェクトの設定に使用する以下のようなファイルが含まれます。 <ul style="list-style-type: none"><li>• 設定ファイル (*.icf)</li><li>• 特殊機能レジスタの記述ファイル (*.sfr)</li><li>• C-SPY デバイス記述ファイル (*.ddf)</li><li>• デバイス選択ファイル (*.i79, *.menu)</li><li>• 各種デバイス向けフラッシュローダアプリケーション (*.out)</li><li>• 構文カラー表示用設定ファイル (*.cfg)</li><li>• アプリケーション/ライブラリプロジェクト用のプロジェクトテンプレート (*.ewp)、ライブラリプロジェクト用のライブラリ設定ファイル</li></ul>
arm¥cstat	C-STAT に関連したファイルを含めます。
arm¥doc	このユーザーガイドのハイパーテキスト形式の PDF のオンラインバージョン、Arm リファレンスガイド、オンラインヘルプガイド (*.chm) も含めます。ディレクトリには、Arm ツールに関する最近追加された情報があるリリースノートも含まれます。
arm¥drivers	C-SPY ドライバで必要な低レベルのデバイスドライバ（特に USB ドライブ）が格納されています。
arm¥examples	サンプルプロジェクトの関連ファイルが含まれており、[インフォメーションセンタ] から開くことができます。
arm¥inc	標準 C/C++ ライブラリのヘッダファイルなど、インクルードファイルが含まれます。また、特殊機能レジスタ (SFR) を定義するヘッダファイルも含まれています。これらのファイルは、コンパイラとアセンブラの両方で使用されます。
arm¥lib	コンパイラが使用するビルド済みライブラリおよび対応するライブラリ設定ファイルが含まれます。
arm¥plugins	プラグインモジュールとしてロード可能なコンポーネント用の実行可能ファイルおよび説明ファイルが含まれます。

表 10: arm ディレクトリ

ディレクトリ	説明
arm¥rtos	IAR Embedded Workbench に統合されたサードパーティ製 RTOS およびミドルウェアソリューションの製品情報、評価バージョン、サンプルプロジェクトを含みます。
arm¥src	一部の設定可能なライブラリ関数およびライブラリソースコードのソースファイルが含まれます。 ILINK リンカの場合、ディレクトリには ELF ユーティリティのソースコードも含まれます。
arm¥tutorials	インフォメーションセンタのチュートリアルで使用されるファイルが含まれます。

表 10: arm ディレクトリ (続き)

## common ディレクトリ

common ディレクトリには、すべての IAR Embedded Workbench 製品で共有するコンポーネント用のサブディレクトリが含まれています。

ディレクトリ	説明
common¥bin	エディタ、グラフィカルユーザインタフェースのコンポーネントなど、すべての IAR Embedded Workbench 製品に共通のコンポーネント用実行可能ファイルが含まれます。IDE 用の実行可能ファイルもここにあります。
common¥config	IDE で開発環境の設定に使用されるファイルが含まれます。
common¥doc	すべての IAR Embedded Workbench 製品に共通のコンポーネントに関する最新の追加情報とリリースノートが含まれます。これらのファイルの内容を確認することをお勧めします。ディレクトリには、インストールやライセンスに関するドキュメントもあります。
common¥plugins	プラグインモジュールとしてロード可能なコンポーネント用の実行可能ファイルや記述ファイル（コードカバレッジ用のサンプルモジュールなど）が格納されています。

表 11: common ディレクトリ

## install-info ディレクトリ

install-info ディレクトリには、インストールされている製品コンポーネントのメタデータ（バージョン番号、名前など）が含まれています。これらのファイルは変更しないでください。

---

## 製品ディレクトリ構成

プロジェクトを構築するとき、IDE はプロジェクトのディレクトリに新しいディレクトリを作成します。サブディレクトリが作成され、ディレクトリの名前は、ビルドの構成で使用しているものが使用され、通常 `Debug` や `Release` です。ディレクトリにはこれらのサブディレクトリが含まれます。

List	さまざまなリストファイルの保存先ディレクトリ。
Obj	コンパイラとアセンブラが生成したオブジェクトファイルの保存先ディレクトリ。オブジェクトファイルの拡張子は <code>o</code> であり、リンカの入力として使用されます。
Exe	保存先ディレクトリ： <ul style="list-style-type: none"><li>● 実行可能ファイルの拡張子 <code>out</code> であり、IAR C-SPY® デバッガの入力として使用されます。</li><li>● ライブラリオブジェクトファイルの拡張子 <code>a</code>。</li><li>● ファイルの拡張子 <code>o</code> の <code>TrustZone</code> インポートライブラリファイル。</li></ul>

---

## さまざまな設定ファイル

IDE で作業するとき、IDE はさまざまなタイプの設定のファイルを作成します。これらのファイルは、グローバル設定またはローカル設定が含まれているかどうかで、異なるディレクトリに格納されます。

### グローバル設定のファイル

グローバル設定のファイルは、`C:\Users\%User%\AppData\Local\IAR Embedded Workbench` に格納されます。これらはグローバル設定ファイルです。

CodeTemplates.txt CodeTemplates.ENU.txt CodeTemplates.JPN.txt	ファイルは仮定義されたコードテンプレートを維持します。  英語以外の言語で利用可能な IDE を使用している場合は、IAR Embedded Workbench を初めて起動したときに言語バージョンを選択するように求められます。この場合、ファイル名の拡張子は <code>ENU</code> や <code>JPN</code> で、選択する言語により異なります（英語または日本語）。  144 ページのコードテンプレートの使用と追加を参照してください。
---------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>global.custom_argvars</code>	カスタム引数変数があるファイルは、グローバルスコープに定義されます。
	93 ページの [カスタムの引数変数の設定] ダイアログボックスを参照してください。
<code>IarIde.xml</code>	インストールされた IAR Embedded Workbench 製品の IDE とプロジェクトの設定グローバルがあるファイル。

## ローカル設定のファイル

ローカル設定のファイルは、プロジェクトのディレクトリに作成される設定ディレクトリに格納されます。これらはローカル設定ファイルです。

<code>Project.dbgdt</code>	デバッガデスクトップ設定のファイル。
<code>Project.Buildconfig.cspy.bat</code>	C-SPY が呼び出されるといつも作成されるバッチファイル。
<code>Project.Buildconfig.driver.xcl</code>	C-SPY が呼び出されるといつも作成されるファイル。これには、お使いの C-SPY ドライバ専用で使用されるコマンドラインオプションが含まれます。
<code>Project.Buildconfig.general.xcl</code>	C-SPY が呼び出されるといつも作成されるファイル。これには、cspybat 専用で使用されるコマンドラインオプションが含まれます。
<code>Project.dnx</code>	デバッガ初期化情報のファイル。
<code>Workspace.wsdtd</code>	ワークスペースデスクトップ設定のファイル。
<code>Workspace.wspos</code>	メインの IDE ウィンドウに置く情報があるファイル。
<code>Workspace.custom_argvars</code>	ワークスペースのローカルスペース用に定義されるカスタムの引数変数のファイル。93 ページの [カスタムの引数変数の設定] ダイアログボックスを参照してください。

# ファイルタイプ

IAR システムズの開発ツールは、以下のデフォルトのファイル名拡張子を使用して、生成されるファイルやその他の認識されるファイルタイプを識別します：

拡張子	ファイルタイプ	出力元	入力先
a	ライブラリ	iarchive	ILINK
asm	アセンブラソースコード	テキストエディタ	アセンブラ
bat	Windows コマンドバッチファイル	C-SPY	ウィンドウ
board	フラッシュローダの設定	テキストエディタ	C-SPY
c	C ソースコード	テキストエディタ	コンパイラ
cfg	構文カラー表示設定	テキストエディタ	IDE
cgx	呼出しグラフファイル	ILINK	－
chm	オンラインヘルプシステムファイル	--	IDE
cpp	C++ ソースコード	テキストエディタ	コンパイラ
crun	C-RUN のフィルタ設定	IDE	IDE
cspy.bat	cspybat の呼出しファイル	C-SPY	－
dat	STL コンテナのフォーマット用マクロ	IDE	IDE
dbgdt	デバッガのデスクトップ設定	C-SPY	C-SPY
ddf	デバイス記述ファイル	テキストエディタ	C-SPY
dep	依存関係情報	IDE	IDE
dnx	デバッガ初期化ファイル	C-SPY	C-SPY
ewd	C-SPY のプロジェクト設定	IDE	IDE
ewp	IAR Embedded Workbench プロジェクト（現行バージョン）	IDE	IDE
ewplugin	プラグインモジュール用 IDE 記述ファイル	--	IDE
ewt	C-STAT および C-RUN のプロジェクト設定	IDE	IDE
eww	ワークスペースファイル	IDE	IDE

表 12: ファイルタイプ



拡張子	ファイルタイプ	出力元	入力先
flash	フラッシュローダの設定	テキストエディタ	C-SPY
flashdict	フラッシュローダのリダイレクトの指定	テキストエディタ	C-SPY
fmt	<b>【ローカル】</b> ウィンドウ、 <b>【ウォッチ】</b> ウィンドウでの表示フォーマット設定	IDE	IDE
h	C/C++、またはアセンブラのヘッダソース	テキストエディタ	コンパイラ、アセンブラの#include
helpfiles	[ヘルプ] メニュー構成ファイル	テキストエディタ	IDE
html、htm	HTML ドキュメント	テキストエディタ	IDE
i	プリプロセス済みソース デバイス選択ファイル	コンパイラ テキストエディタ	コンパイラ IDE
icf	リンカ設定ファイル	テキストエディタ	ILINK
inc	アセンブラのヘッダソース	テキストエディタ	アセンブラの#include
ini	プロジェクト設定	IDE	—
log	ログ情報	IDE	—
lst	リスト出力	コンパイラ、アセンブラ	—
mac	C-SPY マクロ定義	テキストエディタ	C-SPY
menu	デバイス選択ファイル	テキストエディタ	IDE
o	オブジェクトモジュール	コンパイラ、アセンブラ	ILINK
out	ターゲットアプリケーション	ILINK	EPROM、C-SPY など
out	ターゲットアプリケーション (デバッグ情報を含む)	ILINK	C-SPY、その他のシンボリックデバッグ

表 12: ファイルタイプ (続き)

拡張子	ファイルタイプ	出力元	入力先
pack	CMSIS-Pack パッケージファイル	ソフトウェアベンダー	CMSIS-Pack パッケージ マネージャ
pbd	ソースブラウザ情報	IDE	IDE
pbi	ソースブラウザ情報	IDE	IDE
pew	IAR Embedded Workbench プロジェクト (旧プロジェクトフォーマット)	IDE	IDE
prj	IAR Embedded Workbench プロジェクト (旧プロジェクトフォーマット)	IDE	IDE
reggroups	ユーザー定義の登録グループの設定	IDE	IDE
s	アセンブラソースコード	テキストエディタ	アセンブラ
sfr	特殊機能レジスタの定義	テキストエディタ	C-SPY
sim	フラッシュローダの入力用にフォーマットされた簡易コード	C-SPY	C-SPY
suc	スタック使用量制御ファイル	テキストエディタ	ILINK
svd	システムビューの説明	テキストエディタ	C-SPY
vsp	visualSTATE プロジェクトファイル	IAR visualSTATE Designer	IAR visualSTATE Designer および IAR Embedded Workbench IDE
wsdt	ワークスペースのデスクトップ設定	IDE	IDE
wspos	メインの IDE ウィンドウの配置情報	IDE	IDE
xcl	拡張コマンドライン	テキストエディタ	アセンブラ、コンパイラ、リンカ、cspybat、ソースブラウザ

表 12: ファイルタイプ (続き)

IDE を実行すると、いくつかのファイルが作成され、プロジェクトディレクトリの専用ディレクトリに格納されます。デフォルトでは \$PROJ\_DIR\$¥Debug、\$PROJ\_DIR\$¥Release、\$PROJ\_DIR\$¥settings に格納されます。これらのディレクトリやファイルはどれも IDE の実行には影響を与えないため、必要に応じて問題なくこれらのファイルを削除できます。

# メニューリファレンス

- メニュー

---

## メニュー

リファレンス情報：

- [ファイル] メニュー
- [編集] メニュー
- [表示] メニュー
- [プロジェクト] メニュー
- [ツール] メニュー
- [ウィンドウ] メニュー
- [ヘルプ] メニュー

また、デバッガを起動すると、C-SPY 専用のメニューが使用可能になります。これらのメニューについては、『*ARM 用 C-SPY® デバッガガイド*』を参照してください。

## [ファイル] メニュー

[ファイル] メニューには、ワークスペースおよびソースファイルのオープン、保存、出力、IDE の終了を実行するためのコマンドが表示されます。

また、最近開いたファイルやワークスペースの番号付きリストも表示されます。メニューから選択することにより、これらを開くことができます。

	新規ファイル(N)	Ctrl+N
	新規ワークスペース(N)	
	ファイルを開く(O)...	Ctrl+O
	ワークスペースを開く(O)...	
	ヘッダー/ソースファイルを開く(O)	Ctrl+Shift+H
	閉じる(C)	Ctrl+F4
	ワークスペースの保存(V)	
	ワークスペースを名前を付けて保存(K)...	
	ワークスペースを閉じる(E)	
	保存(S)	Ctrl+S
	名前を付けて保存(A)...	
	すべて保存(U)	
	ページ設定(U)...	
	印刷(P)...	Ctrl+P
	最近使用したファイル(F)	▶
	最近使用したワークスペース(R)	▶
	終了(X)	

メニューのコマンド

以下のコマンドがあります。



**新規ファイル (&N)**  
新規テキストファイルを作成します。



**新規ワークスペース**  
新しいワークスペースを作成します。



**ファイルを開く (Ctrl+O)**  
テキストファイルや HTML ドキュメントを選択するために、標準のオープンダイアログボックスを開きます。152 ページの [エディタ] ウィンドウを参照してください。



**ワークスペースを開く**  
開くワークスペースファイルを選択するために、標準のオープンダイアログボックスを開きます。新しいワークスペースを開く前に、現在開かれているワークスペースを保存して閉じるかどうかを確認するメッセージが表示されます。

**ヘッダ / ソースファイルを開く (Ctrl+Shift+H)**

現在のファイルに対応するヘッダファイルやソースファイルを開き、現在のファイルから新しく開かれたファイルにフォーカスを移します。このコマンドは、エディタウィンドウのコンテキストメニューからも実行できます。

**閉じる**

アクティブなウィンドウを閉じます。修正されているファイルを、閉じる前に保存するかどうかを確認するメッセージが表示されます。

**名前を付けてワークスペースを保存**

現在のワークスペースファイルを保存します。

**ワークスペースに名前を付けて保存**

新しい名前でワークスペースを保存するために、標準の保存ダイアログボックスを開きます。

**ワークスペースを閉じる**

現在のワークスペースファイルを閉じます。

**保存 (Ctrl+S)**

現在のテキストファイルやワークスペースファイルを保存します。

**名前を付けて保存**

現在のファイルを別名で保存するためのダイアログボックスを表示します。

**すべて保存**

開かれているすべてのテキストドキュメントとワークスペースファイルを保存します。

**ページ設定**

印刷オプションを設定するためのダイアログボックスを表示します。

**印刷 (Ctrl+P)**

テキストドキュメントを印刷するためのダイアログボックスを表示します。

**最近使用したファイル**

最近開いたテキストドキュメントをすばやく開くためのサブメニューを表示します。

**最近使用したワークスペース**

最近開いたワークスペースファイルをすばやく開くためのサブメニューを表示します。



終了

IDEを終了します。テキストファイルを閉じる前に、変更内容を保存するかどうかを確認するメッセージが表示されます。プロジェクトの変更は自動的に保存されます。

[編集] メニュー

[編集] メニューから、編集 / 検索用のコマンドを実行できます。

	元に戻す(U)	Ctrl+Z
	やり直し(R)	Ctrl+Y
	切り取り(T)	Ctrl+X
	コピー(C)	Ctrl+C
	貼り付け(P)	Ctrl+V
	すべて選択(L)	Ctrl+A
	検索と置換(F)	▶
	移動(G)	▶
	コードテンプレート(O)	▶
	次のエラー/タグ(X)	F4
	前のエラー/タグ(V)	Shift+F4
	語句の入力補完(M)	Ctrl+Alt+Space
	コードの入力補完(D)	Ctrl+Space
	パラメータのヒント(H)	Ctrl+Shift+Space
	括弧のマッチング(H)	▶
	折り返みを切り替え(T)	Ctrl+Alt+F
	自動インデント(I)	Ctrl+T
	ブロックコメント(K)	Ctrl+K
	ブロックコメントの解除(B)	Ctrl+Shift+K
	ブレークポイントの切り替え(A)	F9
	ブレークポイントの有効化/無効化(N)	Ctrl+F9

メニューのコマンド

以下のコマンドがあります。



元に戻す (Ctrl+Z)

現在のエディタウィンドウで最後に行った変更を取り消します。



やり直し (Ctrl+Y)

現在のエディタウィンドウで [元に戻す] により取り消した変更を再実行します。エディタウィンドウごとに、編集の元に戻す / やり直しを無制限に実行できます。

**切り取り (Ctrl+X)**

エディタウィンドウとテキストボックスでテキストを切り取るための Windows 標準のコマンド。

**コピー (Ctrl+C)**

エディタウィンドウとテキストボックスでテキストをコピーするための Windows 標準のコマンド。

**貼り付け (Ctrl+V)**

エディタウィンドウとテキストボックスでテキストを貼り付けるための Windows 標準のコマンド。

**すべて選択 (Ctrl+A)**

アクティブなエディタウィンドウで、すべてのテキストを選択します。

**検索と置換 > 検索 (Ctrl+F)**

現在のエディタウィンドウでテキストを検索するための **[検索]** ダイアログボックスを表示します (161 ページの **[検索]** ダイアログボックスを参照)。**[検索]** コマンドの選択時に **[メモリ]** ウィンドウに挿入ポイントがある場合は、ダイアログボックスに表示されるオプションが変化します。**[検索]** コマンドを選択したときに挿入ポイントが **[トレース]** ウィンドにある場合、**[トレースを検索]** ダイアログボックスが開きます。このダイアログボックスの内容は、使用する C-SPY ドライバによって異なります (詳しくは『**ARM 用 C-SPY® デバッグガイド**』を参照)。

**検索と置換 > 次を検索 (F3)**

指定した文字列に一致する次の箇所を検索します。

**検索と置換 > 前を検索 (Shift+F3)**

指定した文字列に一致する前の箇所を検索します。

**検索と置換 > 次を検索 (指定文字列) (Ctrl+F3)**

現在選択されている文字列または現在挿入ポイントを囲んでいる単語に一致する次の箇所を検索します。

**検索と置換 > 前を検索 (指定文字列) (Ctrl+Shift+F3)**

現在選択されている文字列または現在挿入ポイントを囲んでいる単語に一致する前の箇所を検索します。

**検索と置換 > 置換 (Ctrl+H)**

指定した文字列を検索し、一致箇所を別の文字列に置換するためのダイアログボックスを表示します (164 ページの **[置換]** ダイアログボックスを参照)。

【置換】コマンドの選択時に【メモリ】ウィンドウに挿入ポイントがある場合は、ダイアログボックスに表示されるオプションが変化します。



#### 検索と置換 > ファイルから検索

指定した文字列を複数のテキストファイルで検索するためのダイアログボックスを表示します（163 ページの【ファイルで検索】ウィンドウを参照）。



#### 検索と置換 > ファイル内で置換

複数のテキストファイル内で指定した文字列を検索し、それらを別の文字列に置換するためのダイアログボックスを表示します（167 ページの【ファイル内で置換】ダイアログボックスを参照）。



#### 検索と置換 > インクリメンタル検索 (Ctrl+I)

検索文字列を少しずつ変更し、検索の絞込みや拡大を行うことができるダイアログボックスを表示します（170 ページの【インクリメンタル検索】ダイアログボックスを参照）。



#### 移動 > 移動 (Ctrl+G)

【行へ移動】ダイアログボックスを表示します。このダイアログボックスを使用して、現在のエディタウィンドウで指定されている行や列に挿入ポイントを移動できます。



#### 移動 > ブックマークの切替え (Ctrl+F2)

アクティブなエディタウィンドウの挿入ポイントのある行で、ブックマークを設定 / 解除します。



#### 移動 > 前のブックマーク (Shift+F2)

挿入ポイントを、【ブックマークの切替え】コマンドで定義した前のブックマークに移動します。



#### 移動 > 次のブックマーク (F2)

挿入ポイントを、【ブックマークの切替え】コマンドで定義した次のブックマークに移動します。



#### 移動 > 前へ移動 (Alt+ ←)

挿入ポイント履歴で前の項目に移動します。挿入ポイントの現在の位置は、【定義に移動】コマンドの実行時や、【ファイルから検索】コマンドの結果をクリックしたときに、履歴に追加されます。



#### 移動 > 次へ移動 (Alt+ →)

挿入ポイント履歴で次の項目に移動します。挿入ポイントの現在の位置は、【定義に移動】コマンドの実行時や、【ファイルから検索】コマンドの結果をクリックしたときに、履歴に追加されます。



**移動 > 定義に移動 (F12)**

選択されたシンボルや挿入ポイントが置かれているシンボルの定義を表示します。ブラウズ情報が有効な場合に、このメニューコマンドを使用できます (69 ページの [プロジェクト] オプションを参照)。

**コードテンプレート > テンプレートの挿入 (Ctrl+Alt+V)**

挿入ポイントの位置に挿入するコードテンプレートを選択できるリストを、エディタウィンドウで表示します。選択したコードテンプレートでフィールドへの入力が必要な場合は、[テンプレート] ダイアログボックスが表示されます (180 ページの [テンプレート] ダイアログボックスを参照)。コードテンプレートの使用方法については、144 ページのコードテンプレートの使用と追加を参照してください。

**コードテンプレート > テンプレートの編集**

現在のコードテンプレートファイルを開き、既存のコードテンプレートの修正やユーザ定義コードテンプレートの追加を行います。コードテンプレートの使用方法については、144 ページのコードテンプレートの使用と追加を参照してください。

**次のエラー / タグ (F4)**

メッセージウィンドウにエラーメッセージのリストや [ファイルで検索] による検索の結果が含まれる場合、このコマンドによってそのリストの次の項目がエディタウィンドウに表示されます。

**前のエラー / タグ (Shift+F4)**

メッセージウィンドウにエラーメッセージのリストや [ファイルで検索] による検索の結果が含まれる場合、このコマンドによってそのリストの前の項目がエディタウィンドウに表示されます。

**語句の入力補完 (Ctrl+Alt+Space)**

入力内容に応じて、エディタドキュメントの他の部分の内容から入力語を推測して補完します。

**コードの入力補完 (Ctrl+Space)**

..、->、:: の後に挿入ポイントを配置したり、これらの文字の前にクラスやオブジェクト名があるときに、クラスで使用可能なシンボルの一覧を表示します。詳細については、143 ページのコードの入力補完を参照してください。

**パラメータのヒント (Ctrl+Shift+Space)**

パラメータを、入力した関数のパラメータリストのツールチップ情報として提示します。ある関数のオーバーロードされたバージョンがいくつかある場合、ツールチップの矢印をクリックすると使用するバージョンを選択できます。詳細については、144 ページのパラメータのヒントを参照してください。

括弧のマッチング

挿入ポイントの直近の括弧内のテキストをすべて選択します。すでに選択されている場合は、その外側の次の括弧まで選択範囲を拡大します。外側に括弧がない場合は、ビープ音を再生します。



折り畳みを切り替え (Ctrl+Alt+F)

アクティブなプロジェクトのすべてのコードの折り目を展開したり折りたたみます。



自動インデント (Ctrl+T)

C/C++ ソースファイルで選択した行にインデントを設定します。インデントの設定については、63 ページの [自動インデントの設定] ダイアログボックスを参照してください。



ブロックコメント (Ctrl+K)

C++ のコメント文字列 // を、選択した行の最初に追加します。



ブロックコメントの解除 (Ctrl+Shift+K)

C++ のコメント文字列 // を、選択した行の最初から削除します。



ブレークポイントの切替え (F9)

ソースウィンドウで、カーソルを含むかまたはカーソルの近くの文か命令で、ブレークポイントを設定 / 解除します。このコマンドは、デバッグツールバーのアイコンボタンから実行できます。

ブレークポイントの有効化 / 無効化 (Ctrl+F9)

ブレークポイントの無効（実際には削除せず、後で再度使用できる状態にする）と有効を切り替えます。

[表示] メニュー

[表示] メニューは、IDE でウィンドウを開くためのいくつかのコマンドを提供します。C-SPY の実行中は、このメニューからデバッガ固有のウィンドウも開くことができます。以下については、『ARM 用 C-SPY® デバッガガイド』を参照してください。



## メニューのコマンド

以下のコマンドがあります。

### メッセージ

メッセージウィンドウへのアクセスを付与するサブメニューを表示します。[ビルド]、[ファイルで検索]、[ソース表示ログ]、[ツールオプション]、[CMSIS-Pack ログ]、[デバッグログ] は IAR Embedded Workbench コマンドからメッセージとテキスト出力を表示します。メニューから選択したウィンドウがすでに開いている場合は、そのウィンドウがアクティブになります。



### ワークスペース

現在の [ワークスペース] ウィンドウを開きます (111 ページの [ワークスペース] ウィンドウを参照)。



### [ソースブラウザ] > [ソースブラウザ]

[ソースブラウザ] ウィンドウを表示します (174 ページの [ソースブラウザ] ウィンドウを参照)。



### [ソースブラウザ] > [参照]

[リファレンス] ウィンドウを表示します (173 ページの [参照] ウィンドウを参照)。



### [ソースブラウザ] > [宣言]

[宣言] ウィンドウを表示します (171 ページの [宣言] ウィンドウを参照)。



### ソースブラウザ > 曖昧な定義

[曖昧な定義] ウィンドウを開きます。 (172 ページの [曖昧な定義] ウィンドウを参照)。



### [ソースブラウザ] > [コールグラフ]

[コールグラフ] ウィンドウを表示します (179 ページの [コールグラフ] ウィンドウを参照)。



### C-STAT>C-STAT メッセージ

[C-STAT メッセージ] ウィンドウを開きます。 (C-STAT® Static Analysis Guide を参照)。

### C-RUN> メッセージ

[C-RUN メッセージ] ウィンドウを開きます。 (『ARM 用 C-SPY® デバッグガイド』を参照)。

### C-RUN> メッセージルール

[C-RUN メッセージルール] ウィンドウを開きます。 (『ARM 用 C-SPY® デバッグガイド』を参照)。



### ブレイクポイント

【ブレイクポイント】 ウィンドウを表示します (ARM 用 C-SPY® デバッガガイドを参照)。



### コールスタック

【コールスタック】 ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### ウォッチ

サブメニューから 【ウォッチ】 ウィンドウのインスタンスを開きます。C-SPY の実行中のみ使用できます。



### ライブウォッチ

【ライブウォッチ】 ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### クイックウォッチ

【クイックウォッチ】 ウィンドウを表示します。C-SPY の実行中のみ使用できます。

### 自動

【自動変数】 ウィンドウを表示します。C-SPY の実行中のみ使用できます。

### ローカル

【ローカル】 ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### 静的変数

【静的変数】 ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### メモリ

サブメニューから 【メモリ】 ウィンドウのインスタンスを開きます。C-SPY の実行中のみ使用できます。

### レジスタ

レジスタウィンドウ 【レジスタ】 および 【レジスタユーザーグループのセットアップ】 にアクセス可能なサブメニューを表示します。C-SPY の実行中のみ使用できます。



### 逆アセンブリ

【逆アセンブリ】 ウィンドウを開きます。C-SPY の実行中のみ使用できます。



### スタック

サブメニューから **[スタック]** ウィンドウのインスタンスを開きます。C-SPY の実行中のみ使用できます。



### シンボルメモリ

**[シンボルメモリ]** ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### ターミナル I/O

**[ターミナル I/O]** ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### マクロ > マクロクイック起動

**[マクロクイック起動]** ウィンドウが開きます。C-SPY の実行中のみ使用できます。



### マクロ > マクロ登録

**[マクロ登録]** ウィンドウが開きます。C-SPY の実行中のみ使用できます。



### マクロ > デバッグマクロ

**[デバッグマクロ]** ウィンドウが開きます。C-SPY の実行中のみ使用できます。



### シンボル

**[シンボル]** ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### コードカバレッジ

**[コードカバレッジ]** ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### イメージ

**[イメージ]** ウィンドウを表示します。C-SPY の実行中のみ使用できます。



### コア

**[コア]** ウィンドウを開きます。C-SPY の実行中のみ使用できます。

### 障害例外ビューア

**[Fault exception viewer]** ウィンドウを開きます。『ARM 用 C-SPY® デバッグガイド』を参照してください。このメニューコマンドは、C-SPY を実行しているときのみ使用できます。

## 【プロジェクト】メニュー

【プロジェクト】メニューには、ワークスペース、プロジェクト、グループ、ファイルの操作用コマンド、ビルドツールのオプションの指定用コマンド、現在のプロジェクトでツールを実行するためのコマンドが表示されます。



### メニューのコマンド

以下のコマンドがあります。



#### ファイルの追加

現在のプロジェクトに追加するファイルを選択するためのダイアログボックスを表示します。



### グループの追加

新しいグループを作成するためのダイアログボックスを表示します。  
**[グループ名]** テキストボックスには、新しいグループ名を入力します。  
 グループの詳細は、101 ページの **グループ** を参照してください。



### ファイルリストのインポート

通常の **[開く]** ダイアログボックスを表示します。このダイアログボックスを使用して、IAR システムズの別のツールチェーンで作成したプロジェクトからファイルやグループに関する情報をインポートできます。

ファイル拡張子が古い形式の `pew`、`prj` のいずれかであるプロジェクトファイルから情報をインポートするには、現在の IAR Embedded Workbench のコンテキストメニューにある **[ファイルリストのエクスポート]** を使用して、先に情報をエクスポートする必要があります。

### プロジェクトコネクションの追加

**[プロジェクトコネクションの追加]** ダイアログボックスを表示します (119 ページの **[プロジェクトコネクションの追加]** ダイアログボックスを参照)。

### ビルド構成の編集

新しいビルド構成の定義や既存のビルド構成の削除を行うための **[プロジェクトの構成]** ダイアログボックスを表示します。117 ページの **[プロジェクトの構成]** ダイアログボックスを参照してください。



### 削除

**[ワークスペース]** ウィンドウで、選択した項目をワークスペースから削除します。



### 新規プロジェクトの作成

**[新規プロジェクトの作成]** ダイアログボックスを表示します。ここでは、新規プロジェクトを作成してそれをワークスペースに追加できます (116 ページの **[新規プロジェクトの作成]** ダイアログボックスを参照)。



### 既存プロジェクトの追加

既存のプロジェクトをワークスペースに追加するための標準の **[開く]** ダイアログボックスを表示します。

**オプション (Alt+F7)**

【ワークスペース】ウィンドウで選択した項目に対して、各ビルドツールのオプションを設定できる【オプション】ダイアログボックスを表示します（134 ページの【オプション】ダイアログボックスを参照）。プロジェクト全体、ファイルのグループ、個々のファイルのオプションを設定できます。

**バージョン管理システム**

バージョン管理用コマンドのサブメニューを表示します（120 ページの *Subversion* のバージョン管理システムメニューを参照）。

**メイク (F7)**

最後のビルド以降に変更されたファイルだけをコンパイル、アセンブル、リンクして、現在のビルド構成を最新状態に更新。

**コンパイル (Ctrl+F7)**

選択されているファイルやグループをコンパイル/アセンブルします。

【ワークスペース】ウィンドウで、1 つまたは複数のファイルを選択できます。グループが異なる場合も含め、同一プロジェクト内のすべてのファイルを選択できます。コンパイルするファイルが表示されたエディタウィンドウを選択することもできます。【コンパイル】コマンドは、選択したすべてのファイルがコンパイルまたはアセンブル可能な場合にのみ有効です。

グループを選択することもできます。その場合、コンパイルできないファイル（ヘッダファイルなど）がグループに含まれている場合でも、そのグループ内（およびネストされたグループ内）のコンパイル可能なファイルごとにコマンドが実行されます。

選択したファイルが複数ファイルコンパイルグループの一部である場合にも、コマンドは選択したファイルのみに作用します。

**すべてを再ビルド**

現在のターゲットのすべてのファイルをリビルドし再リンクします。

**クリーン**

すべての中間ファイルを削除します。

**バッチビルド (F8)**

【バッチビルド】ダイアログボックスを表示します。ここでは、指定したバッチビルド構成を作成し、指定したバッチをビルドできます。137 ページの【バッチビルド】ダイアログボックスを参照してください。



**C-STAT 静的解析 > プロジェクトの解析**

C-STAT で選択したプロジェクトを解析します。C-STAT の詳細については、『*C-STAT® Static Analysis Guide*』を参照してください。

**C-STAT 静的解析 > ファイルの解析**

C-STAT で選択したファイルを解析します。C-STAT の詳細については、『*C-STAT® Static Analysis Guide*』を参照してください。

**C-STAT 静的解析 > 分析結果をクリア**

C-STAT で以前に実行した解析の情報を消去します。C-STAT の詳細については、『*C-STAT® Static Analysis Guide*』を参照してください。

**C-STAT 静的解析 > HTML サマリの生成**

標準の保存用ダイアログボックスを表示し、HTML 形式のレポートサマリの保存先を選択して、サマリを作成します。C-STAT の詳細については、『*C-STAT® Static Analysis Guide*』を参照してください。

**C-STAT 静的解析 > 詳細な HTML レポートの生成**

標準の保存用ダイアログボックスを表示し、HTML 形式の詳細なレポートの保存先を選択して、レポートを作成します。C-STAT の詳細については、『*C-STAT® Static Analysis Guide*』を参照してください。

**プロジェクトの解析**

選択した外部のアナライザを実行して、プロジェクトのすべてのソースコードについて解析を実行します。アナライザの一覧は、[IDE オプション] ダイアログボックスの [外部アナライザ] ページで指定したアナライザから読み込まれます。

このメニューコマンドは、外部アナライザを追加した時だけ使用できることに気を付けてください。詳細については、37 ページの *外部のアナライザを使用するにあたって* を参照してください。

**ファイルの解析**

選択した外部アナライザを実行し、ファイルグループまたは個々のファイルに対して解析を実行します。アナライザの一覧は、[IDE オプション] ダイアログボックスの [外部アナライザ] ページで指定したアナライザから読み込まれます。

このメニューコマンドは、外部アナライザを追加した時だけ使用できることに気を付けてください。詳細については、37 ページの *外部のアナライザを使用するにあたって* を参照してください。

**ビルドを停止 (Ctrl+Break)**

現在のビルド処理を停止します。

**ダウンロードしてデバッグ (Ctrl+D)**

プロジェクトのオブジェクトファイルのデバッグができるように、アプリケーションをダウンロードし、C-SPY を起動します。必要であれば、C-SPY の実行前に **make** が実行され、プロジェクトが更新されます。このコマンドは、デバッグセッション中は使用できません。

**ダウンロードせずにデバッグ**

プロジェクトのオブジェクトファイルのデバッグができるように、C-SPY を起動します。このメニューコマンドは、**[ダウンロード]** ページの **[ダウンロードを中止する]** オプションのショートカットです。**[ダウンロードせずにデバッグ]** コマンドは、デバッグセッション中は使用できません。

**実行しているターゲットにアタッチ**

ターゲットシステムをリセットせずに、実行中アプリケーションの現在の場所にデバッガをアタッチします。プロジェクト内にブレークポイントを定義した場合、C-SPY ドライバは、アタッチ中にそれらを設定します。C-SPY ドライバが、ターゲットシステムを停止することなく、それらを設定することができない場合、ブレークポイントは無効になります。また、このオプションは、ダウンロードと **[指定位置まで実行]** オプションを抑制します。

オプションを使用できない場合は、お使いの C-SPY ドライバとデバイスの組み合わせにより、サポートされていないことがあります。

**メイク後デバッガを再起動**

C-SPY の停止、アクティブなビルド構成の作成、デバッガ再開を実行します。すべてを 1 つのコマンドで実行します。このコマンドは、デバッグセッション中しか使用できません。

**デバッガを再起動**

C-SPY の停止とデバッガ再開を実行します。すべてを 1 つのコマンドで実行します。このコマンドは、デバッグセッション中しか使用できません。

**ダウンロード**

フラッシュダウンロードおよび消去のためのコマンド。以下のコマンドから選択します。

**[アクティブなアプリケーションのダウンロード]** は、すべてのデバッグセッションを開始せずに、アクティブなアプリケーションをターゲットにダウンロードします。この結果は、デバッグセッションを開始して実行が発生する前に終了したときとほぼ同じになります。

【ファイルのダウンロード】は、標準の【開く】ダイアログボックスを開きます。ここでは、完全なデバッグセッションを開始せずにターゲットシステムにダウンロードするファイルを指定できます。

【メモリ消去】は、フラッシュメモリのすべてを消去します。

.board ファイルでフラッシュメモリが 1 つだけ指定されている場合、消去を確認する単純な確認のダイアログボックスが表示されます。ただし、.board ファイルで 2 つ以上のフラッシュメモリが指定されている場合、【メモリ消去】ダイアログボックスが表示されます。このダイアログボックスの詳細については、『ARM 用 C-SPY® デバッグガイド』を参照してください。

### SFR の設定

【SFR の設定】ウィンドウが開いて、C-SPY が情報を持っている現在定義された SFR が表示されます。このウィンドウの情報については、『ARM 用 C-SPY® デバッグガイド』を参照してください。

### CMSIS-Manager

**CMSIS Manager** ダイアログボックスを表示します (96 ページの *CMSIS Manager* ダイアログボックス参照)。

このメニューコマンドは、ターゲットが CMSIS-Pack をサポートしている場合にのみ利用可能です。

### デバイス記述ファイルを開く

使用中のすべてのデバイスファイルおよび SFR 定義ファイルの一覧からファイルを選択することができるサブメニューを開きます。

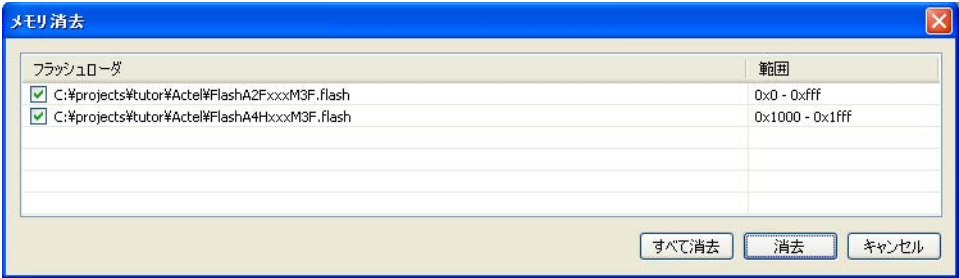
### レジスタリストを保存

すべての定義されたレジスタのリストを生成します。それには各レジスタのサイズ、場所、アクセスタイプの情報がある SFR も含まれます。デバッグセッション中は、リストには現在のレジスタの値も含まれます。このメニューコマンドは、プロジェクトを IDE に読み込まれているときだけ使用できます。

## 【メモリ消去】ダイアログボックス

【メモリ消去】ダイアログボックスは、【プロジェクト】>【ダウンロード】>【メモリの消去】を選んだ時、およびお使いのフラッシュメモリシステム構成

ファイル（ファイルの拡張子 .board）で 2 つ以上のフラッシュメモリを指定しているときに表示されます。



このダイアログボックスを使用して、必要な数のフラッシュメモリを消去します。

表示エリア

各行にフラッシュメモリのデバイス設定ファイル（ファイル名の拡張子 .flash）のパスと関連のメモリ範囲が一覧表示されます。消去するメモリを選択してください。

ボタン

以下のボタンを選択できます。

すべて消去

個別に選択した行に関係なく、このダイアログボックスに表示されたすべてのメモリが消去されます。

消去

選択したメモリが消去されます。

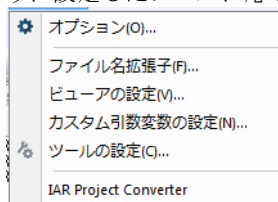
キャンセル

ダイアログボックスを閉じます。

[ツール] メニュー

[ツール] メニューには、共通フォントの変更、ショートカットキーの変更などの、環境のカスタマイズ用コマンドが表示されます。

ユーザが定義可能なメニューで、IAR Embedded Workbench で使用するツールを追加することができます。したがって、メニュー項目として表示されるように設定したツールに応じて、表示が異なる場合があります。



## メニューコマンド

以下のコマンドがあります。



### オプション

**[IDE オプション]** ダイアログボックスを表示します。このダイアログボックスを使用して、IDE のカスタマイズができます。以下を参照してください。

55 ページの [共通フォント] オプション

56 ページの [キーカスタマイズ] オプション

58 ページの [言語] オプション

59 ページの [エディタ] オプション

63 ページの [自動インデントの設定] ダイアログボックス

64 ページの [外部エディタ] のオプション

66 ページの [セットアップファイル] オプション

67 ページの [色とフォント] オプション

68 ページの [メッセージ] オプション

69 ページの [プロジェクト] オプション

75 ページの ソースコード管理オプション (廃止)

77 ページの [デバッガ] オプション

79 ページの [スタック] オプション

81 ページの [ターミナル I/O] オプション

### ファイル名拡張子

ビルドツールで指定可能なファイル名の拡張子を定義するための  
[ファイル名拡張子] ダイアログボックスを表示します (88 ページの  
[ファイル名拡張子] ダイアログボックスを参照)。

### ビューアの設定

ドキュメント表示用のビューアアプリケーションを設定するための  
[ビューアの設定] ダイアログボックスを表示します (85 ページの  
[ビューアの設定] ダイアログボックスを参照)。

### カスタム引数変数の設定

カスタム引数変数を定義および編集できる [カスタム引数変数の設定]  
ダイアログボックスを表示します (93 ページの [カスタムの引数変数  
の設定] ダイアログボックスを参照)。



### ツールの設定

外部ツールを利用するためのインタフェースを設定できる [ツールの  
設定] ダイアログボックスを表示します (83 ページの [ツールの設  
定] ダイアログボックスを参照)。

### IAR プロジェクトコンバーター

別のツールベンダーからのプロジェクトファイルを IAR Embedded  
Workbench のプロジェクトファイルに変換できる [IAR プロジェクト  
コンバーター] ダイアログボックスを表示します。arm¥doc ディレク  
トリにあるプロジェクトコンバーターガイドを参照してください。

### Notepad

ユーザ設定項目。ユーザが [ツール] メニューに追加した項目が表示  
されます。

## [ウィンドウ] メニュー

[ウィンドウ] メニューでは、IDE ウィンドウの操作や画面上での配置変更のコマンドを提供します。



[ウィンドウ] メニューの最後のセクションには、画面で開かれているウィンドウのリストが一覧表示されます。リストからウィンドウを選択すると、そのウィンドウに切り替えます。

### メニューのコマンド

以下のコマンドがあります。



#### ドキュメントを閉じる (Ctrl+W)

アクティブなエディタドキュメントを閉じます。



#### ウィンドウを閉じる

アクティブな IDE ウィンドウを閉じます。



#### 分割

ウィンドウを縦または横方向に 2 つか 4 つに分割し、同一ファイルの異なる部分を同時に表示します。

#### タブを新規エディタウィンドウとして縦に並べる

現在のエディタウィンドウのとなり新しい空白のウィンドウを開いて、アクティブなドキュメントを次のウィンドウに移動します。

#### タブを新規エディタウィンドウとして横に並べる

現在のエディタウィンドウの下に新しい空白のウィンドウを開いて、アクティブなドキュメントを次のウィンドウに移動します。

**次のウィンドウへタブを移動**

現在のウィンドウのアクティブなドキュメントを次のウィンドウに移動します。

**前のウィンドウへタブを移動**

現在のウィンドウのアクティブなドキュメントを前のウィンドウに移動します。

**すべてのアクティブでないタブを閉じる**

現在のタブ以外のすべてのタブを閉じます。

**アクティブなタブの右側にあるすべてのタブを閉じる**

現在のタブに右側にあるタブをすべて閉じます。

**すべてのエディタタブを閉じる**

エディタウィンドウで表示されているすべてのタブを閉じます。

**ツールバー**

【メイン】 / 【デバッグ】 オプションは、2 つのツールバーの表示 / 非表示を切り替えます。

**ステータスバー**

ステータスバーの表示 / 非表示を切り替えます。

## 【ヘルプ】メニュー

【ヘルプ】メニューでは、IAR Embedded Workbench のヘルプが提供されます。このメニューから、ユーザーインターフェースのバージョン番号と IDE のバージョン番号が検索できます（90 ページの製品情報ダイアログボックス参照）。

インフォメーションセンタには【ヘルプ】メニューからもアクセスできます。インフォメーションセンタは、チュートリアルやプロジェクトのサンプル、ユーザガイド、サポート情報、リリースノートなど、プロジェクト開発の開始時や作業中に必要な情報リソースに簡単にアクセス可能にするナビゲーションシステムです。また、IAR Systems の Web サイトで役に立つセクションへのショートカットも提供します。



# 一般オプション

- 一般オプションの説明

## 一般オプションの説明

リファレンス情報：

- ターゲット
- 出力
- ライブラリ構成
- ライブラリオプション 1
- ライブラリオプション 2
- MISRA-C

IDE の一般オプションを設定するには、以下の手順に従います。

- 1 [プロジェクト] > [オプション] を選択して、[オプション] ダイアログボックスを開きます。
- 2 [カテゴリ] リストで [一般オプション] を選択します。
- 3 すべての設定をデフォルトの出荷時の設定に戻すには、[出荷時設定] ボタンをクリックします。

## ターゲット

[ターゲット] オプションでは、IAR C/C++ コンパイラおよびアセンブラのターゲット固有の機能を指定します。

ターゲット

プロセッサ選択

☒ コア(O) Cortex-M23

☐ デバイス(D) Nuvoton M2351

☐ CMSIS-Pack None

エンディアンモード

☒ リトル(L)

☐ ビッグ(B)

☐ BE32(3)

☐ BE16(8)

浮動小数点演算の設定

FPU(F) 無し

Dレジスタ(R) -

☐ DSP拡張(E)

☐ Advanced SIMD (NEON)(A)

☒ TrustZone

Mode Non-secure

## プロセッサ選択

プロセッサを選択します。

### コア

使用しているプロセッサコアです。派生品の詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

### デバイス

使用しているデバイスです。デバイスを選択すると、デフォルトのリンカ設定ファイルおよび C-SPY® デバイス記述ファイルが自動的に決定されます。デフォルトファイルのオーバーライド方法については、『*ARM 用 C-SPY® デバッグガイド*』を参照してください。

### CMSIS-Pack

**CMSIS Manager** ダイアログボックスで選択したデバイス。詳細については、96 ページの *CMSIS Manager* ダイアログボックスを参照してください。

## エンディアンモード

プロジェクトのバイトオーダーを選択します。

### リトル

最も低いバイトはメモリの最も低いアドレスに格納されます。最も高いバイトは一番重要です。最も高いアドレスに格納されます。

### ビッグ

一番低いアドレスに一番重要なバイトが保持されます。一方、最も高いアドレスには重要度の低いバイトが保持されます。ビッグエンディアンモードには2つのうちどちらからを選択します。

**BE8** を選択すると、データはビッグエンディアン、コードはリトルエンディアンになります。

**BE32** を選択すると、データとコードの両方がビッグエンディアンコードになります。

## FPU

浮動小数点ユニットを選択します。

### なし (デフォルト)

ソフトウェア浮動小数点ライブラリが使用されます。

### VFPv2

アーキテクチャ VFPv2 に準拠した VFP ユニット。

**VFPv3**

アーキテクチャ VFPv3 に準拠した VFP ユニット。

**VFPv4**

アーキテクチャ VFPv4 に準拠した VFP ユニット。

**VFPv4 単精度**

アーキテクチャ VFPv4、単精度に準拠した VFP ユニット。

**VFPv5 単精度**

アーキテクチャ VFPv5、単精度に準拠した VFP ユニット。

**VFPv5 倍精度**

アーキテクチャ VFPv5、倍精度に準拠した VFP ユニット。

**VFP9-S**

CPU コアの ARM9E ファミリで使用可能な VFPv2 アーキテクチャ。そのため、このコプロセッサを選択することは、VFPv2 アーキテクチャを選択することと同じです。

VFP コプロセッサを選択することで、ソフトウェア浮動小数点ライブラリの使用を、サポートされたすべての浮動小数点演算にオーバーライドします。

**D レジスタ**

コンパイラで使用する D レジスタの数を選択します。

**Advanced SIMD (NEON)**

デバイスで使用できる場合は、プロジェクトに NEON アーキテクチャを選択します。

**DSP 拡張**

デバイスで使用可能な場合は、このオプションを選択してコンパイラに DSP 命令を使用させます。

**TrustZone**

デバイスで使用できる場合は、プロジェクトで TrustZone を有効にします。

[コア] オプションを Cortex-M23 または Cortex-M33 に設定している場合は、このオプションは自動的に選択されます。デバイスに TrustZone がない場合は、このオプションは選択されません。

その他のコアには、デバイスに TrustZone がない、または TrustZone がいつもあるコアを選択している限り、このオプションは自動的に選択解除されます。

**TrustZone** オプションが選択されている場合は、コンパイラとアセンブラオプション `--cmse` が使用されます。

TrustZone の詳細は、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## モード

現在のプロジェクトがセキュアかセキュアでないモードかを指定します。

**TrustZone** オプションが選択されている場合は、このオプションは、自動的に選択されます。

### secure

プロジェクトがセキュアモードでビルドされているかを示します。

セキュアモードを選択している場合は、コンパイラおよびアセンブラオプション `--cmse` は自動的に設定されます。

### non-secure

プロジェクトがセキュアでないモードでビルドされているかを示します。

## 出力

【出力】 オプションによって、出力ファイルのタイプが決まります。また、実行可能ファイル、オブジェクトファイル、リストファイルの保存先も指定できます。

The screenshot shows a dialog box titled "出力" (Output). It contains the following elements:

- 出力ファイル(O):** A section with two radio buttons:
  - ☒ 実行可能ファイル(E) (Executable File)
  - ☐ ライブラリ(L) (Library)
- 出力ディレクトリ** (Output Directories): A section with three text input fields:
  - 実行可能ファイル/ライブラリ(X):** The text "Debug\Exe" is entered.
  - オブジェクトファイル(O):** The text "Debug\Obj" is entered.
  - リストファイル(S):** The text "Debug\List" is entered.

## 出力ファイル

出力ファイルのタイプを選択します。

### 実行可能ファイル（デフォルト）

ビルドプロセスの結果、リンカがアプリケーション（実行可能出力ファイル）を作成します。この設定を使用した場合は、リンカのオプションを【オプション】ダイアログボックスで設定できます。出力を作成する前に、該当するリンカオプションを設定する必要があります。

### ライブラリ

ビルドプロセスの結果、ライブラリビルダがライブラリ出力ファイルを作成します。この設定を使用した場合は、ライブラリビルダオプションを【オプション】ダイアログボックスで設定でき、【リンカ】はカテゴリリストから削除されます。ライブラリを作成する前に、オプションを設定する必要があります。

## 出力ディレクトリ

目的のディレクトリのパスを指定します。プロジェクトディレクトリとの相対パスを指定します。以下を指定できます。

### 実行可能ファイル/ライブラリ

実行可能ファイルやライブラリファイルのデフォルトディレクトリをオーバーライドします。プロジェクトの実行可能ファイルを保存するディレクトリの名前を入力します。

### オブジェクトファイル

オブジェクトファイルのデフォルトのディレクトリをオーバーライドします。プロジェクトのオブジェクトファイルを保存するディレクトリの名前を入力します。

### リストファイル

リストファイルのデフォルトのディレクトリをオーバーライドします。プロジェクトのリストファイルを保存するディレクトリの名前を入力します。

## ライブラリ構成

【ライブラリ構成】オプションによって、使用するライブラリが決まります。

ランタイムライブラリ、ライブラリ構成、これらのライブラリ構成が提供するランタイム環境、可能なカスタマイズについては、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## ライブラリ

使用するランタイムライブラリを選択します。使用可能なライブラリについては、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

実際に使用されるライブラリオブジェクトファイルとライブラリ設定ファイルの名前は、それぞれ【**ライブラリファイル**】テキストボックスと【**設定ファイル**】テキストボックスに表示されます。

## 設定ファイル

使用されるライブラリ構成を表示します。ライブラリ設定ファイルは、プロジェクトの設定に応じて自動的に選択されます。【**ライブラリ**】ドロップダウンリストから【**カスタム DLIB**】を選択した場合は、ライブラリ設定ファイルを指定する必要があります。

## ライブラリでのスレッドのサポートを有効にする

ランタイムライブラリをスレッドとともに使用できるように自動的に設定します。

## ライブラリ低レベルインタフェースの実装

ライブラリに追加する I/O の低レベルインタフェースの種類を制御します。

Cortex-M の場合、以下から選択します。

なし

ライブラリで利用可能な I/O の低レベルサポートはなし。独自の `__write` 関数を用意して、ライブラリの一部の I/O 関数を使用する必要があります。

[セミホスティング] および [セミホスティングを経由した stdout/stderr]

BKPT 命令を使用するセミホスティング I/O。

[セミホスティング] および [SWO 経由の stdout/stderr]

SWO インタフェースが使用されている stdout および stderr 出力以外のすべての関数に BKPT 命令を使用する Semihosted I/O。これは、アプリケーションがデータ転送の実行を停止する必要のない、きわめて高速のメカニズムを意味します。

**IAR ブレークポイント**

使用できません。

他のコアの場合、以下から選択してください。

なし

ライブラリで利用可能な I/O の低レベルサポートはなし。独自の `__write` 関数を用意して、ライブラリの一部の I/O 関数を使用する必要があります。

**セミホスティング**

svc 命令（以前の SWI）を使用するセミホスティング I/O。

**IAR ブレークポイント**

IAR 独自の派生セミホスティング。svc 命令を使用しないため svc ベクタ上にブレークポイントを設定する必要がありません。RTOS など、自身のために svc ベクタを必要とするアプリケーションをデバッグするときに便利です。この方法は、パフォーマンスの向上にも有効です。ただし、他のベンダ製ツールを使用してビルドされたアプリケーション、ライブラリ、オブジェクトファイルでは動作しません。

## CMSIS

CMSIS サポートを有効にするには、以下のオプションを使用します。

**CMSIS を使用する**

CMSIS ヘッダファイルをコンパイラのインクルードパスに追加します。

アプリケーションのソースコードに CMSIS ヘッダファイルが明示的に含まれる場合、このオプションは使用しないでください。このオプションは、Cortex-M デバイスでのみ使用できます。

### DSP ライブラリ

アプリケーションを CMSIS DSP ライブラリにリンクします。このオプションは、Cortex-M デバイスでのみ使用できます。

## ライブラリオプション I

「ライブラリオプション」では、printf と scanf のフォーマットを選択します。

The screenshot shows the 'Library Options' dialog box. It has four tabs: 'Target', 'Output', 'Library Settings', and 'Library Options'. The 'Library Options' tab is active. Under 'Printf Format', there is a dropdown menu set to 'Automatic', a checkbox for 'Enable multi-byte character support' (unchecked), and a text box containing 'Format automatically selected, multi-byte support disabled.' The same structure is repeated for 'Scanf Format'. At the bottom, there is a checkbox for 'Buffered terminal output (B)' which is also unchecked.

フォーマットの機能の詳細については、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

### printf フォーマット

「自動」が選択されている場合、リンカはコンパイラからの情報に基づいて、printf 関連の機能に適切なフォーマットを自動的に選択します。

すべての printf 関連の機能に対するデフォルトのフォーマットをオーバーライドする（wprintf の派生型を除く）には、以下から選択します。

- IAR DLIB ライブラリの Printf フォーマット：フル、大、小、極小

アプリケーションの要件に合ったフォーマットを選択してください。

「マルチバイト文字サポートを有効にする」を選択して、printf フォーマットがマルチバイトをサポートするようにします。

### scanf のフォーマット

「自動」が選択されている場合、リンカはコンパイラからの情報に基づいて、scanf 関連の機能に適切なフォーマットを自動的に選択します。



すべての scanf 関連の機能に対するデフォルトのフォーマットをオーバーライドする（wscanf の派生型を除く）には、以下から選択します。

- IAR DLIB ライブラリの Scanf フォーマット：フル、大、小

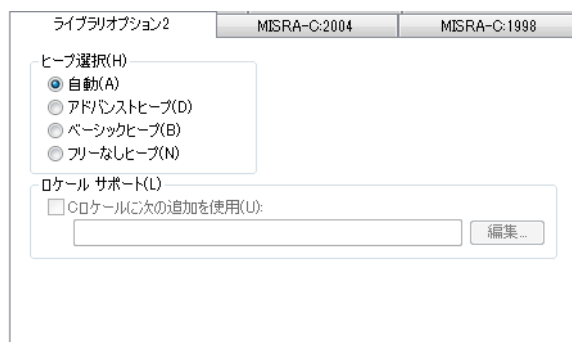
アプリケーションの要件に合ったフォーマットを選択してください。

〔マルチバイト文字サポートを有効にする〕を選択して、scanf フォーマットがマルチバイトをサポートするようにします。

## バッファターミナル出力

プログラムの実行中に、それぞれの新しい文字をすぐに C-SPY の〔ターミナル I/O〕ウィンドウに出力するのではなく、ターミナル出力をバッファに格納します。このオプションは、通信速度が遅いデバッグシステムを使用する場合に便利です。

## ライブラリオプション 2



### ヒープの選択

使用するヒープを選択します。ヒープの詳細は、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

#### 自動

アプリケーションで使用するヒープを自動的に選択します。

コードに free または realloc の呼び出しが含まれてない場合は、フリーなしヒープが選択されます。コードでメモリ割り当てルーティンを読み出す場合は、アドバンストヒープが選択されます。それ以外はベーシックヒープが選択されます。

### アドバンストヒープ

アドバンストヒープを選択します。

### ベーシックヒープ

ベーシックヒープを選択します。

### フリーなしヒープ

最も小さい可能性のヒープの実行を使用します。このヒープは `free` または `realloc` をサポートしてないので、スタートアップの段階で、さまざまなバッファなどにヒープメモリを割り当てるアプリケーションだけに適しています。このヒープメモリが割り当てを解除されることはありません。

## ロケール サポート

C ロケールに加えてリンカが使用するロケールを選択します。(C ロケールが含まれているライブラリ設定を選択した場合に必要です。)

## MISRA-C

[**MISRA-C:1998**] と [**MISRA-C:2004**] オプションは、ソースコードの MISRA-C 規則からの逸脱を IDE が確認する方法を制御します。この設定は、コンパイラとリンカの両方に使用されます。

特定のオプションについて詳しくは、[ヘルプ] メニューから『*IAR Embedded Workbench® MISRA-C:2004 Reference Guide*』または『*IAR Embedded Workbench® MISRA-C:1998 Reference Guide*』を参照してください。

# コンパイラオプション

- コンパイラオプションの説明

---

## コンパイラオプションの説明

リファレンス情報：

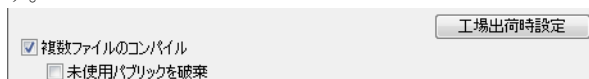
- 複数ファイルのコンパイル
- 言語 1
- 言語 2
- コード
- 最適化
- 出力
- リスト
- プリプロセッサ
- 診断
- MISRA-C
- エンコード
- 追加オプション
- [インクルードディレクトリの編集] ダイアログボックス

**IDE のコンパイラオプションを設定するには、以下の手順に従います。**

- 1** **【プロジェクト】 > 【オプション】** を選択して、**【オプション】** ダイアログボックスを開きます。
- 2** **【カテゴリ】** リストで **【C/C++ コンパイラ】** を選択します。
- 3** すべての設定をデフォルトの出荷時の設定に戻すには、**【出荷時設定】** ボタンをクリックします。

## 複数ファイルのコンパイル

特定のコンパイラオプションを設定する前に、複数ファイルのコンパイルを使用するかどうかの指定ができます。これは、最適化のテクニックの1つです。



### 複数ファイルのコンパイル

［ワークスペース］ウィンドウで選択したプロジェクトファイルのグループに対して、複数ファイルのコンパイルを有効にします。

このオプションは、プロジェクト全体で使用するほか、ファイルのグループごとに使用できます。このようなグループ内のすべての C/C++ ソースファイルが、1 回のコンパイラの呼出しで一緒にコンパイルされます。

つまり、選択したグループに含まれるファイルはすべて、そのグループまたは任意のオプションが設定された直近の親ノードに設定されているコンパイラオプションを使用してコンパイルされます。1 つまたは複数のファイルに対してオーバーライドするコンパイラオプションはすべて、ビルド時に無視されます。これは、グループコンパイルでオプションのセットを 1 つだけ使用する必要があるためです。

複数ファイルのコンパイルがどのように［ワークスペース］ウィンドウに表示されるかについては、「111 ページの［ワークスペース］ウィンドウ」を参照してください。

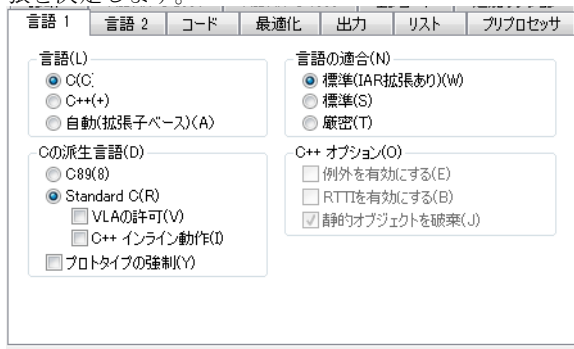
### 未使用パブリックを破棄

コンパイルユニットからの未使用のパブリック関数および変数をすべて破棄します。

複数ファイルのコンパイルおよび未使用パブリック関数の破棄の詳細については、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

## 言語 1

[言語 1] オプションによって、使用するプログラミング言語と有効にする拡張を決定します。



サポートされている言語、派生言語、言語拡張の詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## 言語

C または C++ のコンパイラサポートを決定します。

### C (デフォルト)

コンパイラでソースコードを C として扱います。つまり、C++ 固有の機能は使用できません。

### C++

コンパイラでソースコードを C++ として扱います。

### 自動

コンパイルするファイルのファイル名拡張子に応じて、言語サポートが自動的に決定されます。

c: このファイル名拡張子を持つファイルは C ソースファイルとして扱われます。

cpp: このファイル名拡張子を持つファイルは C++ ソースファイルとして扱われます。

## 言語の適合

標準の C/C++ 言語にどれくらい厳密に準拠するかを制御します。

### 標準 (IAR 拡張あり)

Arm 固有のキーワードを標準の C/C++ 言語への拡張として受け入れます。IDE では、この設定はデフォルトで有効です。

### 標準

IAR システムズの拡張を無効にしますが、選択した C/C++ の派生言語に厳密に準拠するわけではありません。非常に役立つ C/C++ への緩和対応もそのまま利用できます。

### 厳密

選択した C/C++ の派生言語に厳密に準拠します。この設定は C/C++ に役立つ拡張や緩和措置の数多くを無効にします。

## C の派生言語

サポートされている言語が C の場合に、派生言語を選択します。

### C89

C 規格ではなく C89 規格を有効にします。この設定は、MISRA C チェックが有効になっている場合は必須です。

### 標準 C

C18 規格 (C 規格) を有効にします。これはコンパイラで使用される標準規格で、C89 よりも厳密です。C89 に固有の機能は使用できません。このほかに、以下を選択してください。

**VLA の許可** : C11 可変長配列の使用を許可します。

**C++ インライン動作** : C 規格のソースコードファイルをコンパイルする際に C++ インライン動作を有効にします。

### プロトタイプ強制

コンパイラが、すべての関数に正しいプロトタイプがあるかどうかを強制的に検証するようにします。すなわち、ソースコードに以下のいずれかが含まれていると、エラーが出力されます。

- 宣言のない関数、Kernighan & Ritchie C 形式で宣言された関数呼出し
- 先にプロトタイプが宣言されていない **public** 関数の関数定義
- プロトタイプを含まない型の関数ポインタによる間接的な関数呼び出し

## C++ オプション

C++ 言語オプションの選択：

### 例外を有効にする

C++ 言語の例外サポートを有効にします。

### RTTI を有効にする

C++ 言語でランタイム型情報 (RTTI) のサポートを有効にします。

### 静的オブジェクトを破棄

コンパイラはコードを出力して、プログラム終了時に破棄が必要な C++ 静的変数を破壊します。

## 言語 2

[言語 2] オプションは、一部の言語拡張の使用を制御します。

言語 1	言語 2	コード	最適化	出力	リスト	プリプロセッサ
<p>‘char’ の型(P)</p> <p><input type="radio"/> 符号付き</p> <p><input checked="" type="radio"/> 符号なし</p>						
<p>浮動小数点数動作(F)</p> <p><input checked="" type="radio"/> 厳密な適合</p> <p><input type="radio"/> 緩和(サイズ縮小/高速化)</p>						

### ‘char’ の型

通常は、コンパイラは char 型を unsigned char として解釈します。**‘CHAR’ の型**で **[符号付き]**を選択すると、コンパイラは別のコンパイラとの互換目的などで char 型を signed char として認識します。

**注：**ランタイムライブラリは、符号なしの単純な文字型を使用してコンパイルされています。**[符号あり]** オプションを選択すると、unsigned 単純文字を使用するライブラリ機能への参照は機能しなくなります。

### 浮動小数点数動作

浮動小数点数動作を制御します。以下から選択します。

#### 厳密な適合

コンパイラで、浮動小数点式について C および浮動小数点の標準に厳密に準拠します。

## 緩和

コンパイラで、言語規則を緩和して、浮動小数点式をより積極的に最適化します。このオプションは、以下の条件を満たす浮動小数点式のパフォーマンスを向上させます。

- 式に単精度および倍精度の値が両方含まれている。
- 倍精度の値が精度を失わずに単精度に変換できる。
- 式の結果は単精度に変換されます。

倍制度の代わりに単精度で計算を実行すると、精度が失われることがあります。

## コード

[コード] オプションは、コンパイラのコード生成を制御します。

The screenshot shows the 'Code' tab of a compiler options dialog. It contains two main sections: 'Processor Mode' and 'Position Independent Code/Data'. In 'Processor Mode', 'Thumb(T)' is selected with a radio button. In 'Position Independent Code/Data', there are three unchecked checkboxes: 'Code and read-only data (ropi)(C)', 'Read/Write data (rwpi)(R)', and 'Dynamic read/write initialization (N)'. Below these is another unchecked checkbox: 'Code memory data read generation (N)'.

これらのコンパイラオプションについては、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## プロセッサモード

プロジェクトのプロセッサモードを選択します。

### Arm

完全な 32 ビット命令セットを使用するコードを生成します。

### Thumb

縮小 16 ビット命令セットを使用するコードを生成します。Thumb コードではメモリの使用量を最小限に抑え、8/16 ビットパス環境でのパフォーマンスを向上させます。



## 位置独立コード/データ

コンパイラで位置独立コードとデータをどう扱うかを決定します。

### コードおよびリードオンのデータ (ropi)

アドレスコードおよびリードオンのデータへの PC 関連の参照を使用するコードを生成。

### リード/ライトデータ (rwpi)

静的ベースレジスタからアドレス書き込み可能なデータへのオフセットを使用するコードを生成。

### 動的なリード/ライト初期化なし

静的 C 変数のランタイムの初期化を無効化。

## コードメモリ内のデータ Read 不生成

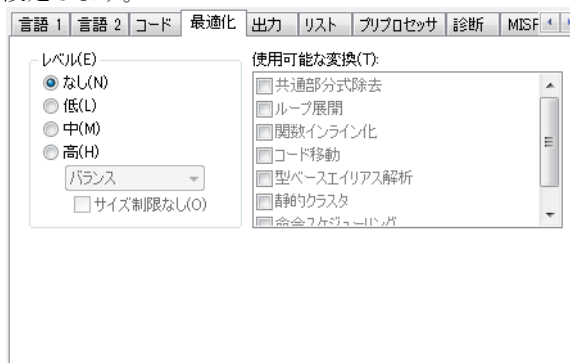
このオプションを使用して、データの読取りが許可されておらず、コードの実行のみが可能なメモリ領域から実行されるコードを生成します。

このオプションは、リンカが実行するライブラリの自動選択にも影響します。IAR 固有の ELF 属性を使用して、このオプションによりコンパイルされたライブラリを使用するかどうかが決まります。

このオプションは ARMv6-M コアおよび ARMv7 コアでのみ使用できます。また、ARMv7 コアの場合に限り、オプション `--ropi` や `--rwpi` と組み合わせることも可能です。詳しくは、『*ARM 用 IAR C/C++ 開発ガイド*』のコンパイラオプション `--no_literal_pool` の項を参照してください。

## 最適化

【最適化】 オプションは、オブジェクトコード生成の最適化の種類とレベルを設定します。



## レベル

最適化レベルを選択します。

### なし

最適化なし。最も充実したデバッグサポートを提供します。

### 低

最も低い最適化レベルです。

### 中

中くらいの最適化レベルです。

### 高（バランス）

最も高い最適化レベルで、速度とサイズのバランスをとります。

### 高（サイズ）

最も高い最適化レベルで、サイズを重視します。

### 高（速度）

最も高い最適化レベルで、速度を重視します。

### サイズ制約なし

速度を重視して最適化しますが、コードサイズの拡張のために通常の制限を緩和します。このオプションは、**[高（速度）]** のレベルでのみ使用できます。

デフォルトでは、デバッグプロジェクトのサイズの最適化は、完全にデバッグが可能なレベルに設定されます。一方リリースプロジェクトでは、速度を損なうことなく小さいコードを生成する、バランスの取れた最適化レベルに設定されます。

各最適化レベルで実行される最適化リストについては、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## 使用可能な変換

さまざまな最適化レベルで、どの変換が使用可能かを選択します。変換が使用可能な場合、チェックボックスを使用して、各変換を有効か無効にすることができます。以下から選択します。

- 共通部分式除去
- ループ展開
- 関数インライン化
- コード移動
- 型ベースエイリアス解析
- 静的クラスタ

- 命令スケジューリング
- ベクトル化

デバッグプロジェクトでは、デフォルトで変換が無効になっています。リリースプロジェクトでは、デフォルトで変換が有効になっています。

個別に無効にできる変換の説明は、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## 出力

[出力] オプションは、生成されるコンパイラ出力を決定します。

出力

☒ デバッグ情報の生成(D)

Codeセクション名(C):

.text

### デバッグ情報の生成

C-SPY® や他のシンボリックデバッガで必要なオブジェクトモジュールに追加情報を含めるようにコンパイラを設定します。

[デバッグ情報の生成] は、デフォルトでは選択されています。コンパイラでデバッグ情報を生成しないように設定する場合は、このオプションの選択を解除します。

**注:** デバッグ情報を含めると、オブジェクトファイルのサイズが増加します。

### コードセクション名

コンパイラは、IAR ILINK リンカが参照する指定セクションに関数を配置します。[コードセクション名] を使用してデフォルト名とは異なる名前を指定し、アプリケーションソースコードの任意の部分を、デフォルトでない別のセクションに配置します。異なるアドレス範囲のコードの配置を管理し、@ 表記または #pragma location ディレクティブでは不十分な場合に、このオプションが有益です。

**注:** デフォルトで使用するセクション以外の定義済みセクションに関数を明示的に配置する場合は注意してください。状況によっては有益なオプション

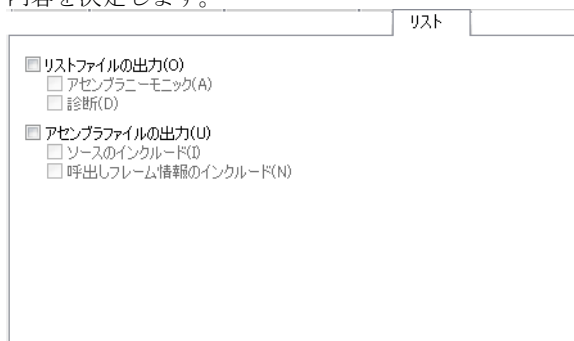
ですが、配置を間違えると、コンパイル時やリンク時のエラーメッセージやアプリケーションの誤動作が発生することがあります。状況を慎重に考慮し、宣言および関数や変数の使用に関する要件に、厳密に従ってください。

セクション名の変更時には、対応するリンカ設定ファイルも変更する必要があります。ことに、注意してください。

セグメントの詳細およびコードの配置を制御するための各種方法の詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## リスト

【リスト】 オプションによって、コンパイラでリストファイルを生成し、その内容を決定します。



デフォルトでは、コンパイラはリストファイルを生成しません。リストファイルかアセンブラファイルを生成する場合は、以下のオプションを選択します。リストファイルは、リストディレクトリに保存されます。ファイル名は、ソースファイル名に拡張子 `lst` を付けて設定されます。

このリストファイルをデフォルトのリストファイル用ディレクトリ以外のディレクトリに保存する場合、【一般オプション】カテゴリの【出力ディレクトリ】オプションを使用します（220 ページの出力を参照）。

【ワークスペース】ウィンドウの【出力】フォルダから出力ファイルを直接開くことができます。

### リストファイルの出力

コンパイラにリストファイルを生成するよう指示します。【ワークスペース】ウィンドウの【出力】フォルダから出力ファイルを直接開くことができます。

デフォルトでは、コンパイラはリストファイルを生成しません。リストファイルの内容を以下から選択します。

### アセンブラニーモニック

アセンブラニーモニックをリストファイルに含めます。

### 診断

診断情報をリストファイルに含めます。

## アセンブラ出力ファイル

コンパイラにアセンブラリストファイルを生成するよう指示します。リストファイルの内容を以下から選択します。

### ソースのインクルード

ソースコードをアセンブラファイルに含めます。

### 呼出しフレーム情報のインクルード

コンパイラが生成したランタイムモデル属性情報、呼出しフレーム情報、フレームサイズ情報を含めます。

## プリプロセッサ

[プリプロセッサ] オプションを使用して、コンパイラおよびアセンブラで使用するシンボルとインクルードパスを定義できます。

### 標準のインクルードディレクトリを無視

通常はコンパイラとアセンブリは、標準のインクルードディレクトリ内にあるインクルードファイルを自動的に検索します。このオプションを使用して、この動作を無効にします。

## 追加インクルードディレクトリ

インクルードファイルを検索するディレクトリのフルパスを1行に1つずつ指定します。標準のインクルードディレクトリより先に、ここで指定するすべてのディレクトリが指定した順序で検索されます。

参照ボタンを使用して **【インクルードディレクトリの編集】** ダイアログボックスを表示します。ここでは、ファイルブラウザを使用してディレクトリを指定できます。詳細については、243 ページの **【インクルードディレクトリの編集】** ダイアログボックスを参照してください。

絶対パスへの依存を避け、異なるマシンおよびファイルシステムの場所でプロジェクトをより簡単に移植できるようにするには、`$TOOLKIT_DIR$` および `$PROJ_DIR$` のような引数変数を使用することができます (91 ページの *引数変数を参照*)。

## プリインクルードファイル

ソースファイルの最初の行より先にインクルードするファイルを指定します。

## シンボル定義

値も含めてマクロシンボルを値を定義します (1 行に 1 つ)。たとえば以下のようになります。

```
TESTVER=1
```

この例では、このような行がソースファイルの先頭より前に現れた場合と同じ効果があります。

```
#define TESTVER 1
```

値を持たない行は `=1` を指定したときと同じ効果があります。

## ファイルへのプリプロセッサ出力

コンパイラおよびアセンブラで、プリプロセッサの結果をファイル名拡張子 `i` のファイルに出力します。これは `1st` ディレクトリにあります。以下から選択します。

### コメントの保持

出力にコメントを含めます。通常はコメントは空白として扱われ、その内容はプリプロセッサ出力には含まれません。

### #line ディレクティブ生成

出力に `#line` ディレクティブを生成し、各行の出所を示します。

## 診断

**[診断]** オプションは、診断メッセージの分類 / 表示方法を設定します。デフォルト以外の分類を指定する場合に使用します。

診断
<input checked="" type="checkbox"/> リマークを有効化(N) 診断を無効化(S): <input type="text"/> リマークとして処理(R): <input type="text"/> ワーニングとして処理(W): <input type="text"/> エラーとして処理(E): <input type="text"/> <input type="checkbox"/> すべてのワーニングをエラーとして処理(T)

**注:** 致命的なエラーの診断メッセージを無効にしたり、あるいは致命的なエラーの分類を変更することはできません。

### リマークを有効化

リマークの生成を有効にします。デフォルトでは、リマークは出力されません。

最も軽度の診断メッセージを、リマークと呼びます。リマークは、ソースコード中で、生成したコードで異常な動作の原因となる可能性がある部分を示します。

### 診断を無効化

指定したタグの診断メッセージの出力を無効にします。

たとえば、ワーニング `Xx117` と `Xx177` を無効にするには、次のように入力します。

```
Xx117, Xx177
```

### リマークとして処理

診断メッセージをリマークとして分類します。リマークは、最も軽度の診断メッセージです。リマークは、ソースコード中で、生成したコードで異常な動作の原因となる可能性がある部分を示します。

たとえば、`Xx177` のワーニングをリマークとして分類するには、次のように入力します。

```
Xx177
```

### ワーニングとして処理

診断メッセージをワーニングとして分類します。ワーニングは、問題はあるが、コンパイルの途中終了の原因にはならないエラーや脱落を示します。

たとえば、Xx826 のリマークをワーニングとして分類するには、次のように入力します。

Xx826

### エラーとして処理

診断メッセージをエラーとして分類します。エラーは、言語の規則違反のうち、オブジェクトコードが生成されず、終了コードがゼロ以外になるものを示します。

たとえば、Xx117 のワーニングをエラーとして分類するには、次のように入力します。

Xx117

### すべてのワーニングをエラーとして処理

すべてのワーニングをエラーとして分類します。コンパイラがエラーを検出した場合は、オブジェクトコードは生成されません。

## MISRA-C

[MISRA-C:1998] と [MISRA-C:2004] オプションは、[一般オプション] カテゴリの対応するオプションをオーバーライドします。

特定のオプションについて詳しくは、[ヘルプ] メニューから『*IAR Embedded Workbench® MISRA-C:2004 Reference Guide*』または『*IAR Embedded Workbench® MISRA-C:1998 Reference Guide*』を参照してください。



## エンコード

[エンコード] オプションは、ソースファイル、出力ファイル、入力ファイルのエンコードを決定します。

### デフォルトのソースファイルのエンコード

バイト オーダーマーク (BOM) のないソースファイルを読み込むとき、コンパイラが使用するエンコードを指定します。

#### Raw (C ロケール)

Raw エンコード (C ロケール) をデフォルトのソースファイルのエンコードとして設定します。

#### システム ロケール

システム ロケールをデフォルトのソースファイルのエンコードとして設定します。

#### UTF-8

UTF-8 エンコードをデフォルトのソースファイルのエンコードとして設定します。

### テキスト出力ファイルのエンコード

テキスト出力ファイルを生成するときに使用するのエンコードを指定します。

#### ソースエンコードと同じ

ソースファイルのエンコードと同じエンコードを使用します。

#### システム ロケール

システムのロケールエンコードを使用します。

#### UTF-8

UTF-8 エンコードを使用します。

### UTF-16 リトルエンディアン

UTF-16 リトルエンディアンエンコードを使用します。

### UTF-16 ビッグエンディアン

UTF-16 ビッグエンディアンエンコードを使用します。

### BOM 付き

バイトオーダーマーク (BOM) を出力ファイルに追加します。

出力ファイルに UTF エンコードを選択した時だけこのオプションは使用できます。

## デフォルトの入力ファイルのエンコード

バイト オーダーマーク (BOM) のないテキスト入力ファイルを読み込むとき、コンパイラが使用するエンコードを指定します。

### システム ロケール

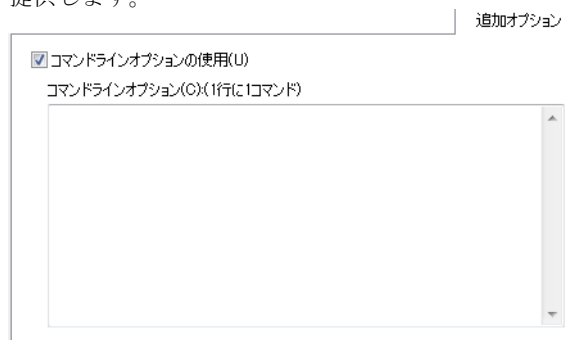
システム ロケールエンコードをデフォルトのエンコードとして設定します。

### UTF-8

UTF-8 エンコードをデフォルトのエンコードとして設定します。

## 追加オプション

[追加オプション] ページは、ツールへのコマンドラインインターフェースを提供します。

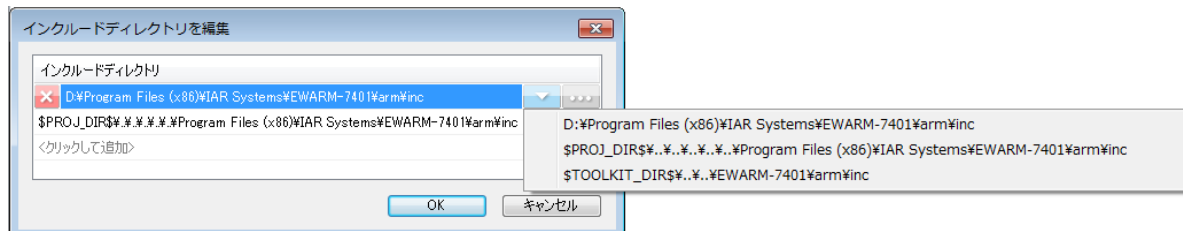


### コマンドラインオプションの使用

ツールに引き渡される追加のコマンドライン引数を指定します (GUI でサポートされていません)。

## 【インクルードディレクトリの編集】ダイアログボックス

【インクルードディレクトリの編集】ダイアログボックスは、コンパイラおよびアセンブラのカテゴリの【オプション】ダイアログボックスで【プリプロセッサ】ページから使用できます。



このダイアログボックスを使用して、インクルードパスを指定または削除したり、パスを相対または絶対にすることができます。

インクルードディレクトリにパスを追加するには、次の手順に従います。

- 1 <クリックして追加>という文字部分をクリックします。参照ダイアログボックスが表示されます。
- 2 適切なインクルードディレクトリを探して、[選択]をクリックします。インクルードパスが表示されます。他のライブラリを追加するには、<クリックして追加>をクリックします。

パスを相対または絶対にするには、次の手順に従います。

- 1 ドロップダウンの矢印をクリックします。コンテキストメニューに、引数変数 \$PROJ\_DIR\$ と \$TOOLKIT\_DIR\$ への絶対パスおよび相対パスが表示されます（該当する場合）。
- 2 どちらかを選択します。

パスの順序を変更するには、次の手順に従います。

- 1 ショートカットキーの組合せ Ctrl+Up/Down を使用します。
- 2 選択した組合せに応じて一覧がソートされます。

インクルードパスを削除するには、次の手順に従います。

- 1 インクルードパスを選択して、行頭の赤い十字マークをクリックするか、[削除] キーを押します。
- 2 選択したパスが表示されなくなります。



# アセンブラオプション

- アセンブラオプションの概要

## アセンブラオプションの概要

リファレンス情報：

- 言語
- 出力
- リスト
- プリプロセッサ
- 診断
- 追加オプション

IDE のアセンブラオプションを設定するには、以下の手順に従います。

- 1 [プロジェクト] > [オプション] を選択して、[オプション] ダイアログボックスを開きます。
- 2 [カテゴリ] リストで [アセンブラ] を選択します。
- 3 すべての設定をデフォルトの出荷時の設定に戻すには、[出荷時設定] ボタンをクリックします。

## 言語

[言語] オプションは、アセンブラ言語の特定の動作を制御します。

言語

☒ ユーザシボルで大文字と小文字を区別する(U)

マクロの引|用符(Q):

<>

☐ 代替コモニック、オペランド、レジスタ名を許可(A)

☐ コードメモリ内のデータRead不生成(N)

### ユーザシンボルで大文字と小文字を区別する

大文字 / 小文字の区別を切り替えます。デフォルトでは、大文字と小文字が区別されます。つまり、`LABEL` と `label` は異なるシンボルを示します。大文字 / 小文字の区別をオフにする場合、`LABEL` と `label` は同じシンボルを指します。

### マクロの引用符

各マクロ引数の左右の引用符に使用する文字を選択します。デフォルトでは、これらの引用符は `<` と `>` です。

[マクロ引用符の文字] によって、他の表記法に合わせて引用符を変更したり、あるいはマクロ引数に `<` や `>` を含めることができます。

マクロの引用符(Q):



### 代替ニモニック、オペランド、レジスタ名を許可

既存のアプリケーションから IAR Assembler for Arm へ移行するために、代替レジスタ名、ニモニック、およびオペランドを使用可能にすることができます。この操作には、アセンブラコマンドラインの `-j` オプションを使用します。このオプションを、Arm ADS/RVCT アセンブラ用に書かれたアセンブラソースコードに使用します。詳細については、『*ARM 用 IAR アセンブラリファレンスガイド*』を参照してください。

### コードメモリ内のデータ Read 不生成

このオプションを使用して、データの読取りが許可されておらず、コードの実行のみが可能なメモリ領域から実行されるコードを生成します。

このオプションは、リンカが実行するライブラリの自動選択にも影響します。IAR 固有の ELF 属性を使用して、このオプションによりコンパイルされたライブラリを使用するかどうかが決まります。

このオプションは Arm v7-M コアでのみ使用できます。位置独立でコンパイルされたコードとは使用できません。

## 出力

[出力] オプションによって、生成されるアセンブラ出力が決まります。

出力

☒ デバッグ情報の生成(D)

### デバッグ情報の生成

アセンブラでデバッグ情報を生成します。アプリケーションとともにデバッガを使用する場合に、このオプションを使用します。デフォルトでは、このオプションは、デバッグプロジェクトでは選択されていて、リリースプロジェクトでは選択が解除されています。

## リスト

[リスト] オプションによって、アセンブラでリストファイルを生成し、その内容を決定します。

言語 出力 リスト プリプロセッサ 診断 追加オプション

☒ リストファイルの出力(O)

<input checked="" type="checkbox"/> ヘッダを含む(H)	<input type="checkbox"/> クロスリファレンスを含む(C)
<input checked="" type="checkbox"/> リストを含む(L)	<input type="checkbox"/> #define
<input type="checkbox"/> #include されたテキスト(T)	<input type="checkbox"/> 内部シンボル(Y)
<input type="checkbox"/> マクロ定義(D)	<input type="checkbox"/> 2行間隔(P)
<input checked="" type="checkbox"/> マクロ拡張子(X)	<input type="checkbox"/> 行数/ページ(G): 80
<input type="checkbox"/> マクロ実行情報(I)	タブ間隔(B): 8
<input type="checkbox"/> アセンブラ行のみ(S)	
<input type="checkbox"/> 複数行コード(U)	

### リストファイルの出力

アセンブラでリストファイルを生成して、それをファイル `sourcename.lst` に送信します。デフォルトでは、アセンブラはリストファイルを生成しません。

このリストファイルをデフォルトのリストファイル用ディレクトリ以外のディレクトリに保存する場合、**[一般オプション]** カテゴリの **[出力ディレクトリ]** オプションを使用します (220 ページの *出力* を参照)。**[ワークスペース]** ウィンドウの **[出力]** フォルダから出力ファイルを直接開くことができます。

## ヘッダを含む

ヘッダを含めます。アセンブラリストファイルのヘッダには、製品バージョン、アセンブリの日付と時刻の情報、および使用されたアセンブラオプションと同等のコマンドラインを含みます。

## リストを含む

リストファイルにインクルードする情報のタイプを選択します。

### #include されたテキスト

リストファイルに #include ファイルを含みます。

### マクロ定義

マクロ定義をリストファイルに含みます。

### マクロ拡張子

マクロ拡張をリストファイルから除外します。

### マクロ実行情報

マクロ実行情報をすべてのマクロ呼出しに印刷します。

### アセンブラ行のみ

リストファイルから偽の条件付きアセンブラセクションの行を除外します。

### 複数行コード

必要に応じて、ディレクティブで生成したコードを複数行にリストにします。

## クロスリファレンスを含む

リストファイルの末尾にクロスリファレンステーブルをインクルードします。

### #define

プリプロセッサ #defines をインクルードします。

### 内部シンボル

ユーザ定義およびアセンブラ内部のすべてのシンボルをインクルードします。



## 2 行間隔

2 行間隔を許可します。

## 行数 / ページ

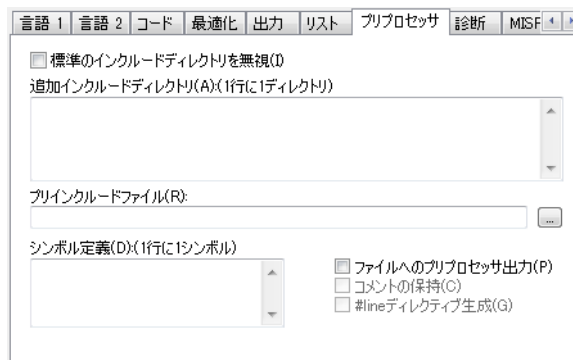
ページあたりの行数を 10 ～ 150 の範囲で指定します。デフォルトのページあたりの行数は、アセンブラのリストファイルの場合 80 です。

## タブ間隔

タブストップあたりの文字位置数を、2 ～ 9 の範囲で変更します。デフォルトでは、アセンブラでタブストップが 8 文字ごとに設定されます。

## プリプロセッサ

[プリプロセッサ] オプションを使用して、コンパイラおよびアセンブラで使用するシンボルとインクルードパスを定義できます。



### 標準のインクルードディレクトリを無視

通常はコンパイラとアセンブリは、標準のインクルードディレクトリ内にあるインクルードファイルを自動的に検索します。このオプションを使用して、この動作を無効にします。

### 追加インクルードディレクトリ

インクルードファイルを検索するディレクトリのフルパスを 1 行に 1 つずつ指定します。標準のインクルードディレクトリより先に、ここで指定するすべてのディレクトリが指定した順序で検索されます。

参照ボタンを使用して [インクルードディレクトリの編集] ダイアログボックスを表示します。ここでは、ファイルブラウザを使用してディレクトリを

指定できます。詳細については、243 ページの **「インクルードディレクトリの編集」** ダイアログボックスを参照してください。

絶対パスへの依存を避け、異なるマシンおよびファイルシステムの場所でプロジェクトをより簡単に移植できるようにするには、\$TOOLKIT\_DIR\$ および \$PROJ\_DIR\$ のような引数変数を使用することができます (91 ページの **引数変数** を参照)。

## シンボル定義

値も含めてマクロシンボルを値を定義します (1 行に 1 つ)。たとえば以下のようになります。

```
TESTVER=1
```

この例では、このような行がソースファイルの先頭より前に現れた場合と同じ効果があります。

```
#define TESTVER 1
```

値を持たない行は =1 を指定したときと同じ効果があります。

## 診断

**【診断】** オプションでは、個々のワーニングやワーニングの範囲を制御します。

診断

ワーニング

☒ 有効(E) ☒ すべてのワーニング(A)

☐ 無効(D) ☐ 特定ワーニング(J):

☐ ワーニング From(F):  To(T):

☐ 最大エラー数(M):

## ワーニング

アセンブラのワーニングを制御します。プログラミングエラーに起因するなどの理由で正当なソースコードの要素を検出した場合、アセンブラはワーニングメッセージを表示します。デフォルトでは、すべてのワーニングが有効になっています。ワーニングの生成を制御するには、以下のいずれかを選択します。

**有効化**

ワーニングを有効にします。

**無効**

ワーニングを無効にします。

**すべてのワーニング**

すべての警告を有効 / 無効にします。

**特定ワーニング**

指定するワーニングを有効 / 無効にします。

**ワーニング from to**

指定した範囲のすべてのワーニングを有効 / 無効にします。

アセンブラワーニングの詳細については、『*ARM 用 IAR アセンブラリファレンスガイド*』を参照してください。

**すべてのワーニングを無効にする**

すべてのワーニングを無効にします。

**ワーニングまたはワーニングの範囲を無効にする**

ワーニングまたは指定した範囲のワーニングを無効にします。

**ワーニングまたはワーニングの範囲を有効にする**

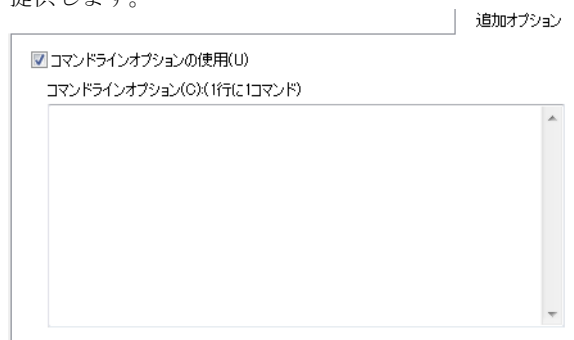
ワーニングまたは指定した範囲のワーニングを無効にします。

**最大エラー数**

エラーの最大数を指定します。つまり、たとえば 1 つのアセンブリでより多くのエラーを確認するために、報告されるエラーの数を増やしたり減らすことができます。デフォルトでは、アセンブラで報告されるエラーの最大数は 100 です。

## 追加オプション

[追加オプション] ページは、ツールへのコマンドラインインターフェースを提供します。



### コマンドラインオプションの使用

ツールに引き渡される追加のコマンドライン引数を指定します（GUI でサポートされていません）。

# 出力コンバータオプション

- 出力コンバータオプションの説明

## 出力コンバータオプションの説明

リファレンス情報：

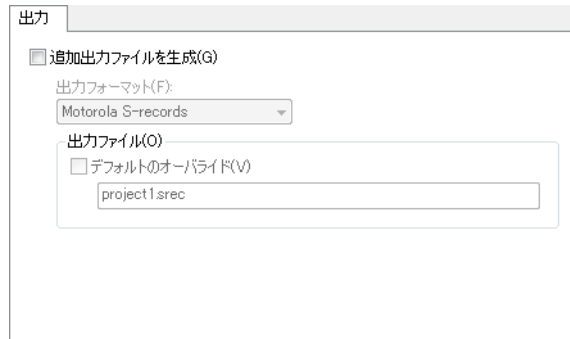
- 出力

IDE のコンバータのオプションを設定するには、以下の手順に従います。

- 1 [プロジェクト] > [オプション] を選択して、[オプション] ダイアログボックスを開きます。
- 2 [カテゴリ] リストで [出力コンバータ] を選択します。

## 出力

[出力コンバータ] オプションによって、プログラム可能な出力フォーマットの詳細を決定します。



## 追加出力の生成

ILINK リンカでは、出力として ELF を生成します（オプションとしてデバッグ情報用の DWARF を含む）。[追加出力の生成] を使用すると、コンバータ ielftool で ELF 出力を Motorola や Intel 拡張など指定した形式に変換できます。コンバータの詳細については、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

**注：**リンカ出力のファイル名拡張子を変更し、出力コンバータ ielftool を使用して出力を変換する場合、ielftool が新しいファイル名拡張子を認識するようにしてください。これを実行するには、[ツール] > [ファイル名拡張子] を選び、ツールチェーンを選択して [編集] をクリックします。[ファイル名拡張子のオーバーライド] ダイアログボックスで、[出力コンバータ] を選択して [編集] をクリックします。[ファイル名拡張子の編集] ダイアログボックスで、[オーバーライド] を選択して新しいファイル名拡張子を入力し、[OK] をクリックします。ielftool が新しいファイル名拡張子を認識するようになります。

### 出力フォーマット

ielftool からの出力形式を選択します。以下から選択します。**Motorola S-records**、**Intel 拡張 hex**、**Texas Instruments TI-TXT**、**Raw バイナリ**、および**シンプルコード**。コンバータの詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

### 出力ファイル

ielftool により変換された出力ファイルの名前を指定します。デフォルトでは、リンカはファイル名の拡張子を持つプロジェクト名を使用します。ファイル名の拡張子は、選択した出力フォーマット (srec、hex など) によって異なります。デフォルト名をオーバーライドするには、[デフォルトのオーバーライド] を選択して、代替のファイル名またはファイル名拡張子を指定します。

# カスタムビルドオプション

- カスタムビルドオプションの説明

## カスタムビルドオプションの説明

リファレンス情報：

- カスタムツール構成

IDE のカスタムビルドオプションを設定するには、以下の手順に従います。

- 1 [プロジェクト] > [オプション] を選択して、[オプション] ダイアログボックスを開きます。
- 2 [カテゴリ] リストで [カスタムビルド] を選択します。

## カスタムツール構成

[カスタムツール設定] オプションは、ツールチェーンに追加するツールの呼出しを制御します。

カスタムツール設定

ファイル名拡張子(F):

コマンドライン(C):

出力ファイル(1行に1ファイル)(O):

追加入力ファイル(1行に1ファイル)(A):

☐ 他のすべてのツールより先にこのツールを実行する

例については、123 ページのツールチェーンの拡張を参照してください。

### ファイル名拡張子

カスタムツールで処理するファイルタイプのファイル名拡張子を指定します。複数のファイル名拡張子を入力できます。区切り文字には、コンマ、セミコロン、空白文字を使用します。以下に例を示します。

.htm; .html

## コマンドライン

外部ツールを実行するためのコマンドラインを指定します。

## 出力ファイル

外部ツールからの出力ファイルの名前を指定します。

## 追加入力ファイル

ビルド処理中に外部ツールが使用する追加ファイルがあれば指定します。これらの追加入力ファイル（依存ファイル）を修正した場合は、リビルドの必要性が検出されます。

## 他のすべてのツールより先にこのツールを実行する

指定したカスタムビルドツールを、他のどのツールよりも先に実行するようにします。これは、**clean** コマンドの実行後やツールを最初に実行するとき、一部のツールに便利です。通常は不明なビルドの依存関係によって起こるエラーの解決に使用されます。たとえば、ツールによってヘッダファイル (h) が生成され、このオプションを使用しない場合、ソースファイルには、生成されていないこのヘッダファイルを含めることができません。



# ビルドアクションオプション

- ビルドアクションのオプションの説明

---

## ビルドアクションのオプションの説明

リファレンス情報：

- ビルドアクションの構成

IDE のビルドアクションのオプションを設定するには、以下の手順に従います。

- 1 [プロジェクト] > [オプション] を選択して、[オプション] ダイアログボックスを開きます。
- 2 [カテゴリ] リストで [ビルドアクション] を選択します。

## ビルドアクションの構成

【ビルドアクションの構成】 オプションでは、IDE でのビルド前およびビルド後のアクションを指定します。これらのオプションは、ビルド構成全体に適用されます。グループやファイル単位で設定することはできません。

ビルド前アクションやビルド後アクションでゼロ以外のエラーコードが返された場合、[ビルド] や [メイク] コマンド全体が中止されます。

### プリビルドコマンドライン

ビルドの前に直接実行されるコマンドラインを指定します。参照ボタンを使用して、実行するツールを検索します。構成が更新済みの場合は、コマンドは実行されません。

### ポストビルドコマンドライン

ビルドが成功した後に直接実行されるコマンドラインを指定します。参照ボタンを使用して、実行するツールを検索します。構成がすでに更新されていた場合は、コマンドは実行されません。このオプションは、出力ファイルのコピーや後処理に便利です。

# リンカオプション

- リンカオプションの説明

---

## リンカオプションの説明

リファレンス情報：

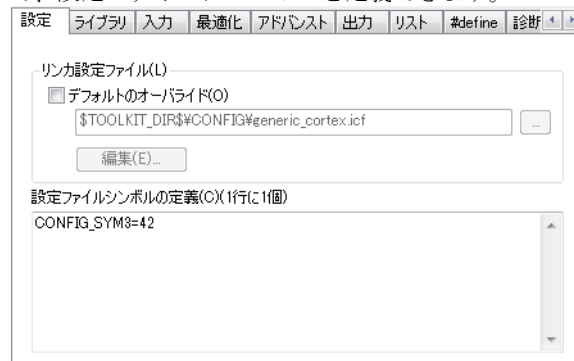
- 設定
- ライブラリ
- 入力
- 最適化
- アドバンスド
- 出力
- リスト
- #define
- 診断
- チェックサム
- エンコード
- 追加オプション
- [追加ライブラリの編集] ダイアログボックス

**IDE のリンカオプションを設定するには、以下の手順に従います。**

- 1** [プロジェクト] > [オプション] を選択して、[オプション] ダイアログボックスを開きます。
- 2** [カテゴリ] リストで [リンカ] を選択します。
- 3** すべての設定をデフォルトの出荷時の設定に戻すには、[出荷時設定] ボタンをクリックします。

## 設定

[設定] オプションを使用すると、リンカ設定ファイルのパスと名前を指定して、設定ファイルにシンボルを定義できます。



### リンカ設定ファイル

デフォルトのリンカ設定ファイルは、使用するプロジェクト設定に応じて自動的に選択されます。デフォルトのファイルをオーバーライドするには、**[デフォルトのオーバーライド]** を選択し、他のファイルを指定します。

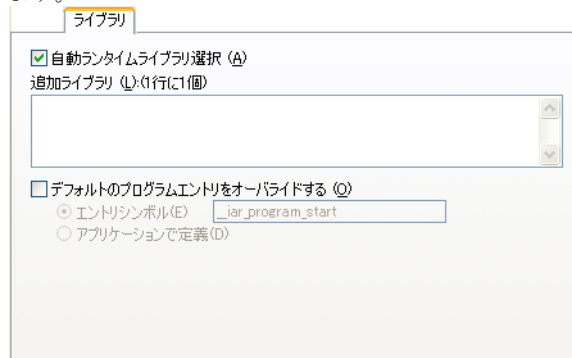
引数変数 `$TOOLKIT_DIR$` または `$PROJ_DIR$` を使用して、プロジェクト固有あるいは定義済みの設定ファイルを指定することができます。

### 設定ファイルシンボルの定義

設定ファイルで使用する常時設定シンボルを定義します。このようなシンボルは、リンカ設定ファイルの `define symbol` ディレクティブを使用して定義したシンボルと同じ効果があります。

## ライブラリ

【ライブラリ】オプションでは、使用されているライブラリのセットを選択します。



使用可能なライブラリについては、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

### 自動ランタイムライブラリ選択

プロジェクト設定に基づいて、リンカで適切なライブラリを自動的に選択します。

### 追加ライブラリ

リンク処理中にリンカが含める追加ライブラリを指定します。ライブラリは 1 行に 1 つしか指定できず、ライブラリへのフルパスを指定する必要があります。

参照ボタンを使用して【追加ライブラリの編集】ダイアログボックスを表示します。ここでは、ファイルブラウザを使用してライブラリを指定できます。詳細については、274 ページの【追加ライブラリの編集】ダイアログボックスを参照してください。

引数変数 \$PROJ\_DIR\$ と \$TOOLKIT\_DIR\$ が使用できます (91 ページの引数変数を参照)。

または、[ワークスペース] ウィンドウで補足のライブラリをプロジェクトに直接追加することができます。この例は、ライブラリの作成および使用のチュートリアルにあります。

### デフォルトのプログラムエントリをオーバーライドする

デフォルトでは、プログラムエントリには \_\_iar\_program\_start というラベルが設定されています。リンカは、プログラムエントリラベルを含むモ

ジュールが含まれていて、そのラベルを含むセクションが破棄されていないことを確認します。

**[デフォルトプログラムエントリのオーバーライド]** は、デフォルトのエントリラベルをオーバーライドします。以下から選択してください。

**エントリシンボル**

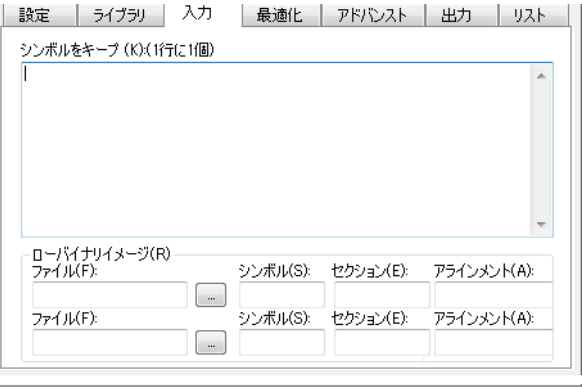
デフォルト以外のエントリシンボルを指定します。

**アプリケーションで定義**

リンクされたオブジェクトコードに定義されたエントリシンボルを使用します。リンカは、通常の場合と同様に、すべてのプログラムモジュールと、すべてのシンボル参照に必要なライブラリモジュールを含め、root 属性が設定されたすべてのセクション、またはそのようなセクションから直接的 / 間接的に参照されるすべてのセクションを保持します。

**入力**

**[入力]** オプションは、リンカへの入力を処理する方法を指定します。



**シンボルをキープ**

最終のアプリケーションに常に含まれるべきシンボル、または 1 行につき複数のシンボルを定義します。

デフォルトでは、リンカはアプリケーションで必要なシンボルのみを保存します。

## ローバイナリイメージ

通常の入力ファイルに加えて、ピュアバイナリファイルをリンクします。以下のパラメータを指定します。

### ファイル

リンクするバイナリファイルを入力します。

### シンボル

バイナリデータが配置されるセクションにより定義されるシンボルを入力します。

### セクション

バイナリデータを配置するセクションを入力します。

### アラインメント

バイナリデータが配置されるセクションのアラインメントを入力します。

ファイルの内容全体が、指定したセクションに配置されます。つまり、このセクションはロウバイナリ出力フォーマットなどのピュアバイナリデータだけを含むことができます。指定したファイルの内容が配置されるセクションは、指定したシンボルがアプリケーションで要求される場合にだけ含まれます。シンボルを強制的に参照するには、[シンボルをキープ]を使用します。単一出力ファイルおよび `--keep` オプションについては、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

## 最適化

[最適化] オプションは、リンカの最適化を制御します。

最適化

☐ 小さいルーチンのインライン化

☐ 重複セクションのマージ(M)

☒ C++仮想関数除去を実行(P)

☐ VFE情報を持たないモジュールがある場合(V)

これらのオプションの詳細については、『ARM 用 IAR C/C++ 開発ガイド』を参照してください。

## 小さいルーチンのインライン化

可能な場合にはリンカがルーチンの呼出しをルーチン本体と置き換えるようにします。

## 重複セクションのマージ

リンカで、リードオンリーのセクションのコピーを 1 つだけ保持します。

これによって異なる関数や定数が同じアドレスを持つことがあるため、このオプションを選択すると、異なるアドレスに依存するアプリケーションが正しく機能しなくなるため注意してください。

## C++ 仮想関数除去を実行

仮想関数除去の最適化を有効にします。

仮想関数の除去を強制的に使用するには、**[VFE 情報を持たないモジュールがある場合]** オプションを有効にします。これは、必要な情報を持たない一部のモジュールが仮想関数の呼出しを実行したり、動的ランタイム型情報を使用すると安全でなくなる可能性があります。

## アドバンスト

**[アドバンスト]** オプションは、その他のリンカの機能を制御します。

アドバンスト

☒ C++例外を許可(A)  
☐ 常に含める  
☒ スタックの使用量解析を有効化(E)  
 制御ファイル  ...  
 呼出しグラフ出力(L) (XML):  ...

これらのオプションの詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

## C++ 例外を許可

このオプションを使用しない場合、インクルードされたコードに `throw` がある場合にリンカがエラーを生成します。

アプリケーションで例外が間違って使用されないようにリンカでチェックする場合は、このオプションを使用しないでください。



## C++ 例外を常に含める

不要と思われる場合でも、リンカは例外処理コードとテーブルをインクルードします。

インクルードされるコードに `rethrow` ではない `throw` 式がある場合、リンカは使用する例外を考慮します。コードのその他の部分にそうした `throw` 式がなければ、リンカは `operator new`、`dynamic_cast`、`typeid` を用意して、失敗したときに例外をスローするのではなく `abort` を呼び出します。コードに他のスローが含まれておらず、これらのコンストラクトからの例外を検出しなければならない場合、このオプションを使用しなければならないことがあります。

アプリケーションで例外が間違って使用されないようにリンカでチェックする場合は、このオプションを使用しないでください。

## スタックの使用量解析を有効にする

スタックの使用量解析を有効にします。リンカマップファイルを生成する場合、スタック使用量の章がマップファイルに含まれます。また、以下のことを実行できます。

### 制御ファイル

使用するスタック使用量制御ファイルを指定して、スタック使用量の解析を制御したり、モジュールや関数のより詳しいスタック使用量情報を提供します。拡張子を指定しない場合は、`suc` が使用されます。

### 呼出しグラフ出力

リンカで生成する呼出しグラフの名前を指定します。拡張子を指定しない場合は、`cgx` が使用されます。

## 出力

[出力] オプションによって、生成されるリンカ出力が決まります。

出力

出力ファイル名(O):

☒ 出力ファイルにデバッグ情報を含める (O)

TrustZone import library

## 出力ファイル名

ILINK 出力ファイル名を設定します。デフォルトでは、リンカはファイル名拡張子を持つプロジェクト名を使用します。out デフォルト名をオーバーライドするには、出力ファイルの別名を指定します。

**注：**リンカ出力のファイル名拡張子を変更し、出力コンバータ `ielftool` を使用して出力を変換する場合、`ielftool` が新しいファイル名拡張子を認識するようにしてください。これを実行するには、**[ツール] > [ファイル名拡張子]**を選び、ツールチェーンを選択して**[編集]**をクリックします。**[ファイル名拡張子のオーバーライド]**ダイアログボックスで、**[出力コンバータ]**を選択して**[編集]**をクリックします。**[ファイル名拡張子の編集]**ダイアログボックスで、**[オーバーライド]**を選択して新しいファイル名拡張子を入力し、**[OK]**をクリックします。`ielftool` が新しいファイル名拡張子を認識ようになります。

## 出力ファイルにデバッグ情報を含める

リンカでデバッグ情報の DWARF も含めた ELF 出力ファイルを生成します。

## TrustZone インポートライブラリ

セキュアプロジェクトがビルドされると、リンカは、セキュアでない部分からコールされるセキュアな部分の関数のリファレンスだけが含まれている、ライブラリファイルを自動的に生成します。**[TrustZone インポートライブラリ]** オプションを使用して、このファイルの名前を指定します。TrustZone インポートライブラリファイルは、プロジェクト実行ファイルと同じディレクトリに格納されます。

## リスト

**[リスト]** オプションは、リンカリストの生成を制御します。

リスト

- ☒ リンカマップファイルの表示 (G)
- ☒ ログファイルの生成 (E)
  - ☐ ライブラリ自動選択 (A)
  - ☐ 初期化決定 (I)
  - ☐ モジュール選択 (M)
  - ☐ リダイレクトされたシンボル (R)
  - ☐ セクション選択 (S)
  - ☐ スタック使用量のコールグラフ (T)
  - ☐ 未使用のセクションフラグメント (U)
  - ☐ ベニア統計 (V)

## リンカマップファイルの表示

リンカでリンカメモリマップを生成し、それを `list` ディレクトリにある `projectname.map` ファイルに送信します。マップファイルとその内容に関する詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

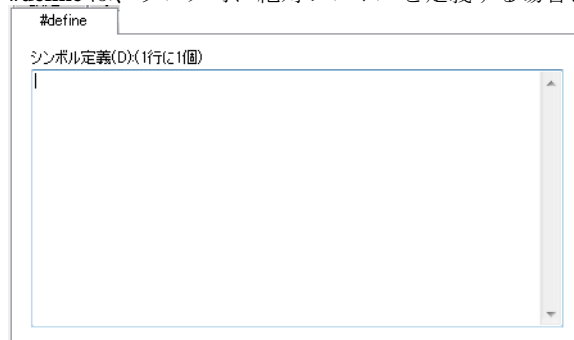
## ログファイルの生成

リンカで、ログ情報を `list` ディレクトリにある `projectname.log` ファイルに保存します。ログ情報は、実行可能なイメージが現在の状態になった原因を把握するために利用できる場合があります。以下を保存できます。

- ライブラリ自動選択
- 初期化決定
- モジュール選択
- リダイレクトされたシンボル
- セクション選択
- スタック使用量呼び出しグラフ
- 未使用のセクションフラグメント
- ベニア統計

## #define

**#define** は、リンク時に絶対シンボルを定義する場合に使用します。



## シンボル定義

リンク時に使用する絶対シンボルを定義します。これは、設定目的の場合に特に便利です。プロジェクトに定義するシンボルを 1 行ごとに 1 つ入力して、その値を指定します。以下に例を示します。

```
TESTVER=1
```

等 (=) 号の前後に空白文字を挿入しないでください。

リンカ設定ファイルに定義できるシンボルの数に制限はありません。この方法で定義したシンボルは、リンカが生成する `?ABS_ENTRY_MOD` という特別なモジュールに含まれます。

既存のシンボルを再定義しようとすると、エラーメッセージが表示されます。

診断

**[診断]** オプションは、診断メッセージの分類 / 表示方法を設定します。デフォルト以外の分類を指定する場合に使用します。

診断

☐ リマークを有効化(N)

診断を無効化(S):

リマークとして処理(R):

ワーニングとして処理(W):

エラーとして処理(E):

☐ すべてのワーニングをエラーとして処理(T)

**注:** 致命的なエラーの診断メッセージを無効にしたり、あるいは致命的なエラーの分類を変更することはできません。

リマークを有効化

リマークの生成を有効にします。デフォルトでは、リマークは出力されません。

最も軽度の診断メッセージを、リマークと呼びます。リマークは、ソースコード中で、生成したコードで異常な動作の原因となる可能性がある部分を示します。

診断を無効化

指定したタグの診断メッセージの出力を無効にします。

たとえば、ワーニング `Xx117` と `Xx177` を無効にするには、次のように入力します。

`Xx117, Xx177`

### リマークとして処理

診断メッセージをリマークとして分類します。リマークは、最も軽度の診断メッセージです。リマークは、ソースコード中で、生成したコードで異常な動作の原因となる可能性がある部分を示します。

たとえば、Xx177 のワーニングをリマークとして分類するには、次のように入力します。

```
Xx177
```

### ワーニングとして処理

診断メッセージをワーニングとして分類します。ワーニングは、問題はあるが、リンク処理の終了前にリンカが終了する原因にはならないエラーや脱落を示します。

たとえば、Xx826 のリマークをワーニングとして分類するには、次のように入力します。

```
Xx826
```

### エラーとして処理

診断メッセージをエラーとして分類します。エラーは、リンクの規則違反のうち、実行可能なイメージが生成されず、終了コードがゼロ以外になるものを示します。

たとえば、Xx117 のワーニングをエラーとして分類するには、次のように入力します。

```
Xx117
```

### すべてのワーニングをエラーとして処理

すべてのワーニングをエラーとして分類します。リンカがエラーを検出した場合は、実行可能イメージは生成されません。

## チェックサム

[チェックサム] オプションは、フィルとチェックサムを制御します。

チェックサム

☒ 未使用コードメモリをフィルする(F)

フィルパターン:

開始アドレス(T): 
 終了アドレス(E):

☒ チェックサム生成(G)

サイズ(Z): 
 アライメント(A):

アルゴリズム:

☐ フルサイズでの結果(U)

補数(C):

ビット順(B):

☐ 語句内でバイトオーダーを逆順にする(R)

チェックサム単位サイズ(H):

初期値(I):

☒ 入力として使用(N)

チェックサム計算の詳細については、『*ARM 用 IAR C/C++ 開発ガイド*』を参照してください。

### 未使用コードメモリをフィルする

指定範囲の未使用メモリをフィルします。

#### フィルパターン

セグメントパート間のギャップに使用するフィルのパターンを 16 進数表記で指定します。

#### 開始アドレス

フィルする範囲の開始アドレスを指定します。

#### 終了アドレス

フィルする範囲の終了アドレスを指定します。

### チェックサム生成

指定した範囲でチェックサムを生成します。

以下から選択します。

#### チェックサムのサイズ

チェックサムのサイズ (1、2、4 バイト) を選択します。

#### アライメント

チェックサムのオプションのアライメントを指定します。通常、アライメントしていないデータにプロセッサがアクセスするときに役立ちます。アライメントを明示的に指定しない場合は、1 のアライメントが使用されます。

## アルゴリズム

チェックサムの計算時に使用するアルゴリズムを選択します。以下から選択します。

**算術合計**。単純な算術合計のアルゴリズム。結果は 8 ビットに切り詰められます。

**CRC16** (デフォルト)、CRC16 アルゴリズム (生成多項式 0x1021)。

**CRC32**。CRC32 アルゴリズム (生成多項式 0x4C11DB7)。

**CRC 多項式**。CRC 多項式アルゴリズムで、指定した値の生成多項式です。

**CRC64ISO**。CRC64ISO アルゴリズム (生成多項式 0x1B)。

**CRC64ECMA**、CRC64ECMA アルゴリズム (生成多項式 0x42F0E1EBA9EA3693)。

## フルサイズでの結果

算術合計アルゴリズムの結果を、1 バイトに切り詰めるのではなく、指定したサイズで生成します。

## 補数

派生した補数 (1 の補数または 2 の補数) を選択します。

## ビット順

各バイトの処理されるビット順を選択します。以下から選択します。

**MSB 優先** : 各バイトで最重要のビットを最初に出力します。

**LSB 優先** : 各バイトのビット順を逆にして、最も重要でないビットを先に出力します。

## 語句内でバイトオーダを逆順にする

[サイズ] で指定したサイズの各語句内で、入力データのビット順を逆にします。

## 初期値

チェックサムの初期値を指定します。これは、使用する core に専用のチェックサム計算があり、その計算をリンカが実行する計算に一致させる場合に使用します。

## 入力として使用

入力データの先頭に、[初期値] で指定した値を含む [サイズ] の語句を 1 字付けます。

### チェックサムユニットサイズ

チェックサムを計算するユニットのサイズを選択します。これは通常、1 回の繰返しにつき 8 ビットを超えるチェックサムを計算するハードウェア CRC 実装と同じチェックサムを生成するようリンカに指示する場合に便利です。以下から選択します。

**8 ビット** : 8 ビットのチェックサムを計算します。

**16 ビット** : 16 ビットのチェックサムを計算します。

**32 ビット** : 32 ビットのチェックサムを計算します。

## エンコード

[エンコード] オプションは、入力ファイルからとリンカからの出力ファイルの文字のエンコードを制御します。

### デフォルトの入力ファイルのエンコード

バイト オーダーマーク (BOM) のないテキスト入力ファイルを読み込むとき、リンカが使用するデフォルトのエンコードを指定します。

#### システム ロケール

システム ロケールをデフォルトのエンコードとして設定します。

#### UTF-8

UTF-8 エンコードをデフォルトとして設定します。

### テキスト出力ファイルのエンコード

テキスト出力ファイルを生成するときに使用するリンカのエンコードを指定します。

#### システム ロケール

システムのロケールエンコードを使用します。



**UTF-8**

UTF-8 エンコードを使用します。

**UTF-16 リトルエンディアン**

UTF-16 リトルエンディアンエンコードを使用します。

**UTF-16 ビッグエンディアン**

UTF-16 ビッグエンディアンエンコードを使用します。

**BOM 付き**

バイトオーダーマークを出力ファイルに追加します。

出力ファイルに UTF エンコードを選択した時だけこのオプションは使用できます。

## 追加オプション

[追加オプション] ページは、ツールへのコマンドラインインターフェースを提供します。

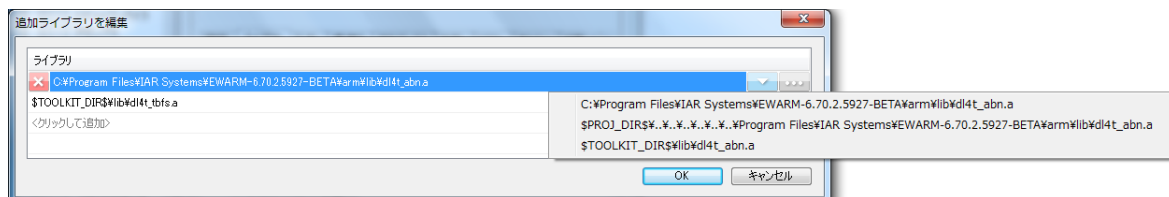
The screenshot shows a dialog box titled "追加オプション" (Additional Options). Inside, there is a checkbox labeled "コマンドラインオプションの使用(U)" (Use command-line options) which is checked. Below it is a text area labeled "コマンドラインオプション(O)(1行に1コマンド)" (Command-line options, 1 command per line). The text area is currently empty and has a vertical scrollbar on the right side.

**コマンドラインオプションの使用**

ツールに引き渡される追加のコマンドライン引数を指定します（GUI でサポートされていません）。

## 【追加ライブラリの編集】 ダイアログボックス

【追加ライブラリの編集】 ダイアログボックスは、【オプション】 ダイアログボックスの 【ライブラリ】 ページから使用できます。



このダイアログボックスを使用して、追加のライブラリを指定したり、ライブラリへのパスを相対あるいは絶対にしします。

追加のライブラリを指定するには、次の手順に従います。

- 1 <クリックして追加> という文字部分をクリックします。参照ダイアログボックスが表示されます。
- 2 適切なインクルードディレクトリを探して、【開く】をクリックします。ライブラリ一覧が表示されます。

他のライブラリを追加するには、<クリックして追加>をクリックします。

パスを相対または絶対にするには、次の手順に従います。

- 1 ドロップダウンの矢印をクリックします。コンテキストメニューに、引数変数 \$PROJ\_DIR\$ と \$TOOLKIT\_DIR\$ への絶対パスおよび相対パスが表示されます（該当する場合）。
- 2 どちらかを選択します。

ライブラリの順序を変更するには、次の手順に従います。

- 1 ショートカットキーの組合せ Ctrl+Up/Down を使用します。
- 2 選択した組合せに応じて一覧がソートされます。

一覧からライブラリを削除するには、次の手順に従います。

- 1 ライブラリを選択して、行頭の赤い十字マークをクリックするか、【削除】キーを押します。
- 2 選択したライブラリが表示されなくなります。

# ライブラリビルダオプション

- ライブラリビルダオプションの説明

---

## ライブラリビルダオプションの説明

リファレンス情報：

- 出力

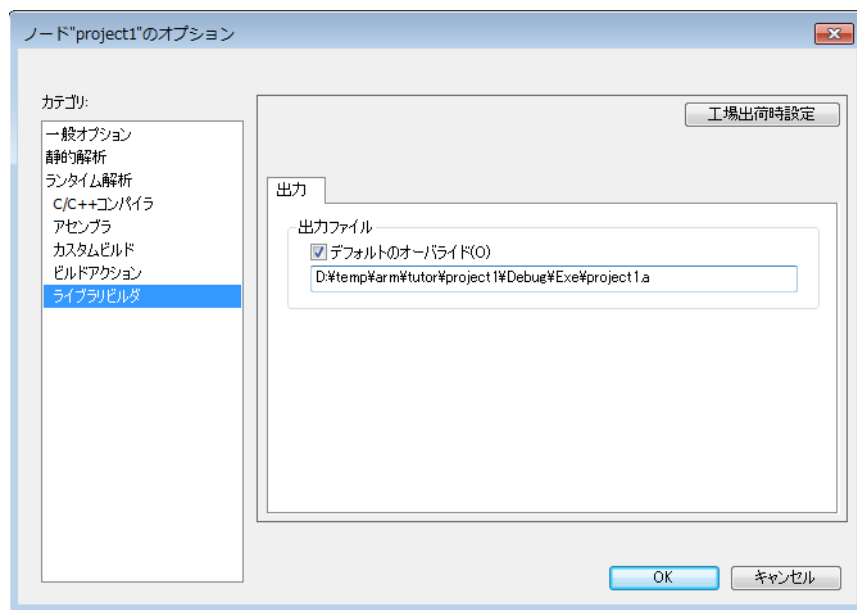
ライブラリビルダのオプションはデフォルトでは使用できません。これらのオプションを IDE で設定するには、先にライブラリビルダツールをカテゴリリストに追加する必要があります。

IDE のライブラリビルドオプションを設定するには、以下の手順に従います。

- 1 [プロジェクト] > [オプション] > [一般オプション] > [出力] を選択します。
- 2 [ライブラリ] オプションを選択します。[ライブラリビルダ] がカテゴリとして [オプション] ダイアログボックスに表示されます。
- 3 [カテゴリ] リストで [ライブラリビルダ] を選択します。

## 出力

【出力】 オプションはライブラリビルダを制御し、ビルド処理の結果として、ライブラリビルダはライブラリ出力ファイルを作成します。



### 出力ファイル

ライブラリビルダからの出力ファイルの名前を指定します。デフォルトでは、リンクはファイル名の拡張子を持つプロジェクト名を使用します。デフォルト名をオーバーライドするには、[デフォルトのオーバーライド]を選択して出力ファイルの別名を指定します。

# 用語集

この用語集は、組み込みシステムのプログラミングに関連した一般的な用語を対象しています。用語によっては、ご使用の IAR Embedded Workbench® のバージョンに該当しないこともあります。

## A

### 絶対アドレス

リンクが割り当てるアドレスではなく、ソースコードで指定したオブジェクトの特定メモリアドレス。

### アドレス式

値がアドレスになっている式。

### AEABI

Arm Limited が提唱する Arm Embedded Application Binary Interface。

### アプリケーション

IAR システムズのツールキットのユーザが開発し、ターゲットプロセッサで組み込みアプリケーションとして実行されるプログラム。

### Ar

アーカイブ、つまりライブラリから作成、修正、抽出するための GNU バイナリユーティリティ。 *larchive* を参照してください。

### アーキテクチャ

コンピュータ設計者の間で使用される、複雑な情報処理システムの構造を示す用語。使用される命令やデータの種類、メモリ構成やアドレッシング方法、システムの実装方法などを示します。プロセッサ設計で使用される主流アーキテクチャとして、ハーバードアーキテクチャとノイマンアーキテクチャの 2 つがあります。

### アーカイブ

ライブラリを参照してください。

### アセンブラディレクティブ

アセンブラの処理を制御するコマンドセット。

### アセンブラ言語

ターゲットプロセッサと、入出力レジスタやデータエリアに対する処理を指定するために使用する、個々のマシン固有のニーモニックセット。メモリ使用量の節約や、アプリケーションの実行速度の向上には、C/C++ よりもアセンブラ言語の方が適している場合があります。

### アセンブラオプション

アセンブラのデフォルトの動作を変更するためのパラメータ。

### 属性

*セクション属性*を参照してください。

### 自動変数

変数が宣言されている関数が呼び出されるごとに、変数の新しいインスタンスが自動的に作成されることを指します。静的オーバーレイを使用するシステム（関数が再帰的に呼び出された場合でも、ローカル変数が 1 つのインスタンスにだけ存在）でのローカル変数の処理と比較できます。ローカル変数と呼ばれることもあります。レジスタ変数と比較してください。

## B

### バックトレース

IAR C-SPY® デバッガが関数から正常に戻るように、呼出しフレーム情報を最新に保つための情報。 *呼出しフレーム情報*を参照してください。

### バンク

メモリバンクを参照してください。

### バンク切替え

異なるメモリバンクの切替え。このソフトウェア技術を使用することで、メモリの異なる部分が同一のアドレス空間を占有できるため、コンピュータの使用可能メモリが増加します。

## バンクコード

複数のメモリバンクに分散したコード。各関数はそれぞれ1つのバンクだけにしか常駐できません。

## バンクデータ

複数のメモリバンクに分散したデータ。各データオブジェクトがそれぞれ1つのメモリバンク内に収まる必要があります。

## バンクメモリ

同一アドレス用に複数の格納場所があるメモリ。メモリバンクを参照してください。

## バンク切替ルーチン

メモリバンクを選択するコード。

## パッチファイル

コマンドラインインタプリタで実行されるオペレーティングシステムコマンドを記述したテキストファイル。UNIXでは、コマンドラインインタプリタがUNIXシェルに含まれているため、「シェルスクリプト」と呼びます。パッチファイルを使用して、既存のコマンドを組み合わせ、新しいコマンドとして実行することができます。

## ビットフィールド

1単位として見なされるビットのグループ。

## ブロック、リンカ設定ファイル

連続するコードまたはデータ。ブロック、オーバーレイ、セクションのいずれかで構成され、空の場合もあります。ブロックには名前があり、ブロックの開始および終了アドレスはアプリケーションから参照できます。また、ブロックには、最大サイズ、特定のサイズ、または最小アラインメントなどの属性を指定できます。内容の順序は固定または任意です。

## ブレークポイント

- 1 コードブレークポイント: プログラム中、そこに到達するとデバッグ用の特殊な処理が実行される地点。通常は、プログラムの実行の停止や、プログラムの変数の一部または全部のダンプを行う箇所に、ブレークポイントを使用します。ブレークポイントは、

は、プログラムの実行を詳細に検証する場合にプログラムそのものの一部として、またはプログラマがデバッグツールでの対話セッションの一部として設定します。

- 2 データブレークポイント: メモリ中で、そこにアクセスするとデバッグ用の特殊な処理が実行される地点。通常は、リード/ライト処理のいずれかでアドレス位置がアクセスされてプログラムの実行を停止する場合に、データブレークポイントを使用します。
- 3 イミディエイトブレークポイント: メモリ中で、そこにアクセスするとデバッグ用の特殊な処理が実行される地点。通常は、メモリアクセス命令の実行中(アクセスの種類に応じて、実際のメモリアクセスの前後)に、プログラム実行を一時停止してユーザが指定したアクションを実行する場合に、イミディエイトブレークポイントを使用します。実行はその後再開されます。この機能は、C-SPYのシミュレータバージョンでのみ使用できます。

# C

## 呼出しフレーム情報

C関数をコンパイルしたコードで、完全な関数の呼出しスタック(コールスタック)を、プログラムカウンタの位置に関わらず、また実行に影響を及ぼすことなく、IAR C-SPY®デバッガで表示できるようにするための情報。バックトレースを参照してください。

## 呼び出し規約

プログラム内の関数が別の関数を呼び出す方法を規定したもの。レジスタパラメータの処理方法、値を返す方法、呼出し先関数が保持するレジスタなどが規定されています。C/C++関数では、すべてコンパイラが自動的に処理します。アセンブラ言語で記述したコードの場合は、C/C++関数からの呼出しや、C/C++関数の呼出しを実行できるように、呼出し規約のルールに従う必要があります。Cの呼出し規約およびC++の呼出し規約は、同一でない場合があります。

## 安価

安価なメモリアクセスのように使用します。安価なメモリアクセスでは、実行にかかるサイクル数や、実装に必要なコードバイト数が少なくなります。メモリアクセスが安価なことを、低コストと言います。メモリアクセスコストを参照してください。

## チェックサム

通信または保管の際に発生した恐れのあるエラーを削除する目的で大きなデータブロックから計算された小さいデータ。CRC（巡回冗長検査）と比較してください。

## コードバンキング

バンクコードを参照してください。

## コードモデル

コードモデルは、アプリケーション用コードの生成方法を制御します。通常は、コードモデルは、関数の呼出し方法や関数が配置されるコード segment などの挙動を制御します。アプリケーションのすべてのオブジェクトファイルは、同一のコードモデルを使用してコンパイルする必要があります。

## コードポインタ

コードポインタとは、関数ポインタを意味します。多くのマイクロコントローラでは複数の異なる方法で関数を呼び出せるため、組込みシステム用のコンパイラでは通常はこれらの方法をすべて使用できます。

コードポインタとデータポインタを混同しないでください。

## コード segment

コードを含むリードオンリー segment。セクションを参照してください。

## コンパイル単位

翻訳単位を参照してください。

## コンパイラオプション

コンパイラのデフォルトの動作を変更するためのパラメータ。

## コンテキストメニュー

コンテキストメニューはユーザーインターフェースで右クリックすると表示され、文脈固有のメニューコマンドを提供します。

## コスト

メモリアクセスコストを参照してください。

## CRC（巡回冗長検査）

バイナリ多項式および初期値に基づいたチェックサムアルゴリズム。CRC アルゴリズムは、単純な算術チェックサムアルゴリズムよりも複雑で、より優れたエラー検出の能力があります。現在幅広く使用されているほとんどのチェックサム算出アルゴリズムは、CRC に基づいています。チェックサムと比較してください。

## C-SPY オプション

IAR C-SPY デバッガのデフォルトの動作を変更するためのパラメータ。

## Cstartup

アプリケーションの実行開始前にシステムを設定するコード。

## C 形式のプリプロセッサ

プリプロセッサは、実際のコンパイル前に入力ストリームを前処理するスタンドアロンアプリケーションかコンパイラ内蔵機能です。C 形式のプリプロセッサは、標準の C で設定された規則に従い、#define、#if、#include などのテキストマクロ置換、条件付きコンパイル、他のファイルのインクルードなどを処理するためのコマンドを実装します。

# D

## データバンキング

バンクデータを参照してください。

## データモデル

データモデルは、デフォルトのメモリタイプを指定します。静的 / グローバル変数、動的に割り当てられたデータ、ランタイムスタックに、アクセスするために使用される方法および生成されるコードを、制御しま

す。また、デフォルトのポインタタイプと、静的 / グローバル変数が配置されるデータ section も制御します。1つのプロジェクトで同時に使用できるデータモデルは1つだけです。また、プロジェクト内のすべてのユーザモジュールとライブラリモジュールで同一のモデルを使用する必要があります。

## データポインタ

多くの core では、異なるメモリタイプやアドレス空間にアクセスするため、複数のアドレッシングモードがあります。通常は、組込みシステム用コンパイラでは、空きメモリに効率的にアクセスできるように、複数のデータポインタタイプセットに対応しています。

## データ表現

データタイプのメモリでの配置方法、データタイプが表現する値の範囲。

## 宣言

オブジェクト（変数、関数）が存在することをコンパイラに対して明示することを指します。オブジェクトそのものは、1つの翻訳単位（ソースファイル）だけで定義する必要があります。オブジェクトは、使用前に宣言し、定義しておく必要があります。通常は、多くのファイルで使用するオブジェクトを1つのソースファイルで定義します。オブジェクトの宣言はヘッダファイルに記述し、そのオブジェクトを使用するファイルでそのヘッダファイルをインクルードします。

以下に例を示します。

```
/* 変数 "a" がどこかに存在。関数  
"b" は2つの int パラメータを取得して1つの  
int. を返します。*/
```

```
extern int a;  
int b(int, int);
```

## 定義

変数か関数そのものを指します。プリケーションの各変数 / 関数につき1つだけ、定義を記述できます。仮定義を参照してください。

以下に例を示します。

```
int a;  
int b(int x, int y)  
{  
    return x + y;  
}
```

## デマングル

マングル化された名前をより一般的な C/C++ 名に復元すること。マングル化を参照してください。

## デバイス記述ファイル

入出力レジスタ (SFR) 定義、割込みベクタ、制御レジスタ定義などのデバイス固有の情報を含む、C-SPY で使用されるファイル。

## デバイスドライバ

高水準のプログラミングインタフェースを周辺デバイスに提供するソフトウェア。

## デジタル信号プロセッサ (DSP)

マイクロプロセッサに類似するデバイスで、内部 CPU が離散時間信号処理用に最適化されています。デジタル信号プロセッサは、マイクロプロセッサの標準命令に加えて、一般的な信号処理計算を高速に実行するための複雑な命令セットもサポートしています。

## 逆アセンブリウィンドウ

メモリの内容を逆アセンブルしてマシン命令に変換し、可能であれば、対応する C ソースコードを挿入して表示する C-SPY ウィンドウ。

## DWARF

ソースレベルデバッグをサポートする業界標準デバッグフォーマット。これは、オブジェクトでデバッグ情報を表すときに IAR ILINK リンカで使用されるフォーマットです。

## 動的初期化

C で記述されたプログラム内の変数は、実行の初期段階で（main 関数が呼び出される前に）初期化されます。これらの変数は、コンパイル時やリンク時に決定される静的な値で初期化されます。これを静的初期化と呼びます。C++ では、グローバルオブジェクトのこ



ンストラクタや、動的メモリ割当てなどのコードを実行することで、変数の初期化が必要な場合があります。

### 動的メモリ割当て

変数の保存には、リンク時に静的に行う方法と、実行時に動的に行う方法の2つがあります。動的メモリ割当ては、多くの場合はヒープから実行されます。ヒープのサイズにより、動的オブジェクトや変数に使用可能なメモリ量が決定されます。動的メモリ割当てには、同時に使用されない複数の変数やオブジェクトを同一メモリに格納することで、アプリケーションに必要なメモリ量を削減できるという利点があります。ヒープメモリを参照してください。

### 動的オブジェクト

実行時に割当て、作成、破棄、解放が行われるオブジェクト。動的オブジェクトは、ほとんどの場合、動的に割り当てられたメモリに格納されます。*静的オブジェクト*と比較してください。

## E

### EEPROM

Electrically Erasable, Programmable Read-Only Memory（電氣的消去可能プログラマブルリードオンリーメモリ）の略。電子的に消去して書き換えることが可能なROM。

### ELF

Executable and Linking Format、業界標準オブジェクトファイルフォーマット。これは、IAR ILINK リンカにより使用されるフォーマットです。デバッグ情報はDWARFを使用してフォーマット化されます。

### Embedded C++

組込みシステムのプログラミング用に設計された、C++ プログラミング言語のサブセット。言語の設計時に、組込みシステム開発で性能と移植性が特に重要であることが考慮されています。

### 組込みシステム

特定用途向けに設計されたハードウェアとソフトウェアの組合せ。組込みシステムがより大規模なシステムや製品の一部となっている場合も多数あります。

### エミュレータ

プロセッサファミリの派生品のエミュレーションを実行するハードウェアデバイス。エミュレータは、しばしば実際の core の代わりに使用し、プリント基板（実際の用途では core を接続）に接続デバイス経由で接続します。エミュレータは、常にターゲットプロセッサと完全に同様に動作し、デバッグですべてのシステムアクチュエータが必要な場合や、デバイスドライバをデバッグする場合に使用します。

### Enea OSE Load モジュールフォーマット

OSE オペレーティングシステムでロード可能な特別な ELF フォーマット。*ELF* を参照してください。

### 列挙型

その型の変数で可能なすべての値のリストを定義に含む型。一般的な例として、[true, false] のリスト中のいずれかの値を取るブール値や、[Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday] のいずれかの値を取る曜日などがあります。列挙型は、C や Ada などの型付き言語の機能です。

文字、整数（サイズ固定）、浮動小数点数などの型も、大きな意味では列挙型と見なされる場合があります（通常は列挙型には属さない）。

### EPROM

Erasable, Programmable Read-Only Memory（消去可能プログラマブルリードオンリーメモリ）の略。紫外線の照射により消去した後に、書き換えることが可能なROM。

### 実行可能イメージ

実行可能なイメージが含まれます。いくつかの再配置可能オブジェクトファイルのリンク結果とライブラリで構成されます。オブジェクトファイルに使用されるファイルフォーマットは、デバッグ情報用の組込みDWARFを含む ELF です。

### 例外

プロセッサハードウェア、メモリ管理ユニット (MMU) などの、プロセッサと緻密に結合されたハードウェアが開始する割込み。アーキテクチャルールの違反（保護メモリへのアクセス）や、極端なエラー状態（ゼロによる除算）を示します。

この用語と、C++ 言語（Embedded C++ を除く）で使用される **例外** という用語とを混同しないでください。

## 高価

**高価**なメモリアクセスのように使用します。高価なメモリアクセスでは、実行にかかるサイクル数か、実装に必要なコードバイト数が多くなります。メモリアクセスが高価なことを、高コストであると言います。メモリアクセスコストを参照してください。

## 拡張キーワード

C/C++ での非標準キーワード。通常は、オブジェクト、定義、宣言（データ、関数）を制御します。キーワードを参照してください。

# F

## フィル

実行可能イメージの **section** 間に存在するバイト（特定のフィルパターンを使用）を埋めること。これらのバイトは、**section** のアラインメント要求のために存在します。

## フォーマット指定子

`printf` などのライブラリ関数で文字列のフォーマットを指定するために使用します。次の例では、関数呼出しでフォーマット指定子 `%c` を 1 つ含むフォーマット文字列を 1 つ指定しており、`a` の値を 1 つの ASCII 文字として出力します。

```
printf("a = %c", a);
```

# G

## 一般オプション

IDE に含まれる全ツールのデフォルトの動作を変更するためのパラメータ。

## 汎用ポインタ

ハーバードアーキテクチャベースの **core** など、すべてのメモリタイプを示すことができるポインタ。

# H

## ハーバードアーキテクチャ

ハーバードアーキテクチャベースの **core** は、独立したデータバスと命令バスを備えています。これにより、並列実行が可能となっています。命令のフェッチ中に、現在の命令がデータバスで実行されます。現在の命令が完了すると、次の命令をすぐに実行できます。これにより、理論上はノイマンアーキテクチャよりも大幅に高速な実行が可能です。ただし、回路は複雑になります。ノイマンアーキテクチャと比較してください。

## ヒープメモリ

ヒープとは、システムで動的メモリ割当て用に確保されたメモリプールです。ヒープの一部をアプリケーション専用に使用することができます。ヒープから割り当てたメモリは、アプリケーションが明示的に解放してヒープに戻すまで有効です。このタイプのメモリは、アプリケーションを実行するまで必要なオブジェクト量がわからない場合に便利です。

このタイプのメモリは、メモリ容量が限られているシステムや、長期間実行するシステムで使用すると問題が生じることがあります。

## ヒープサイズ

動的に割当て可能なメモリの合計サイズ。

## Host

ターゲットプロセッサと通信するコンピュータ。この用語は、デバッガを実行するコンピュータと、開発した組込みアプリケーションを実行する **core** とを区別するために使用します。

# I

## Iarchive

アーカイブ、つまりライブラリを作成する IAR システムズのユーティリティ。Iarchive は、IAR Embedded Workbench に付属しています。

**IDE（統合開発環境）**

必要なすべてのツールを1つのアプリケーションに統合したプログラミング環境。

**lelfdumparm**

IAR システムズのユーティリティ。ELF 再配置可能イメージまたは実行可能イメージの内容のテキスト表示を作成するために使用します。

**lelftool**

IAR システムズのユーティリティ。ELF 実行可能イメージ上でさまざまな変換（フィル、チェックサム、フォーマット変換など）を実行するために使用します。

**ILINK**

ELF/DWARF フォーマットで絶対出力を生成する IAR ILINK リンカ。

**ILINK 設定**

使用可能な物理メモリの定義およびこれらのメモリに対するセクション（コードやデータ）の配置。ILINK では、実行可能イメージを構築する設定が必要です。

**イメージ**

実行可能イメージを参照してください。

**インクルードファイル**

ソースファイルにインクルードされるテキストファイル。この処理は、多くの場合はプリプロセッサが実行します。

**リンカ設定ファイルの初期化設定**

イニシャライザで RAM セクションを初期化する方法を定義します。通常、定数でなく、`noinit` 以外の変数のみが初期化されます。たとえば、コードの一部も初期化できます。

**初期化された section**

起動時に特定の値で初期化されるリード/ライト section。セクションを参照してください。

**インラインアセンブラ**

C 言語の文の間に直接挿入するアセンブラ言語。

**インライン化**

呼出し先関数の本体を関数呼出しに置き換える最適化処理。この最適化により実行速度が向上し、場合によっては生成コードのサイズも削減できます。

**命令ニーモニック**

アセンブラ言語で、マシン命令を表現するために使用される語や頭辞語。プロセッサによって命令セットが異なるため、ADD、BR（分岐）、BLT（値が小さい場合に分岐）、MOVE、LDR（レジスタのロード）などの命令を表現するニーモニックセットも異なります。

**割込みベクタ**

割込み発生時に実行されるコードの一部か、そのコードを示すポインタ。

**割込みベクタテーブル**

割込みベクタをタイプ別にインデックス化して格納したテーブル。このテーブルには、プロセッサでの割込みと割込みサービスルーチンのマッピングが格納され、プログラマが初期化する必要があります。

**割込み**

組込みシステムでは、割込みを使用して、タイマオーバフローやボタンが押されたときなどの外部イベントを即座に検出します。

割込みは、通常処理を一時停止し、制御フローを「割込みハンドラ」ルーチンに一時的に渡す非同期イベントです。割込みは、ハードウェア（入出力、タイマ、マシンチェック）とソフトウェア（モニタプログラム、システム呼出し、トラップ命令）の両方により発生します。トラップと比較してください。

**組込み**

ネイティブのコンパイラオブジェクト、プロパティ、イベント、メソッドを意味する形容詞。

**組み込み関数**

1. 特定のマシンコードシーケンスに直接展開される関数呼出し。2. コンパイラが内部的用途（浮動小数点演算など）で呼び出す関数。

## lobjmanip

ELF オブジェクトファイルの低レベルの操作に使用する IAR システムズのユーティリティ。

# K

## キーカスタマイズ

IDE で使用するメニューコマンド用キーショートカット。

## キーワード

プログラミング言語の構文で定義されているシンボルセット。言語で使用されるすべてのキーワードは予約済みで、識別子（変数やプロシージャなどのユーザ定義オブジェクト）として使用することはできません。  
*拡張キーワード*を参照してください。

# L

## L 値

代入文の左辺の変更可能な値。単純な変数、逆参照されたポインタがこれに該当します。(x + 10) のような式には新しい値を代入できないため、L 値にはなりません。

## 言語拡張

ターゲット固有の C 言語拡張。

## ライブラリ

*ランタイムライブラリ*を参照してください。

## ライブラリ設定ファイル

ランタイムライブラリの設定が記述されたファイル。このファイルでは、ランタイムライブラリに含まれる機能が定義されています。ランタイムライブラリのビルドを調整するために使用されます。*ランタイムライブラリ*を参照してください。

## リンカ設定ファイル

実行可能イメージを構築するときに IAR ILINK リンカにより使用される設定を含むファイル。*ILINK 設定*を参照してください。

## ローカル変数

*自動変数*を参照してください。

## ロケーションカウンタ

*プログラムロケーションカウンタ (PLC)* を参照してください。

## 論理アドレス

*仮想アドレス (論理アドレス)* を参照してください。

# M

## MAC (積和演算)

乗算を加算と共に実行する特殊な命令、オンチップデバイス。次の形式のフィルタや変換を多数使って信号処理を実行する場合に多用されます。

$$y_j = \sum_{i=0}^N c_i \cdot x_{i+j}$$

MAC のアキュムレータは、通常のレジスタより高精度（ビット数が多い）です。*デジタル信号プロセッサ (DSP)* を参照してください。

## マクロ

- 1 アセンブラマクロは、ユーザ定義のアセンブラ行セットであり、後で指定のマクロ名を参照することにより、ソースファイルに展開されます。参照時には、パラメータの置換が行われます。
- 2 C マクロは、ソースファイルの前処理中に使用されるテキスト置換の仕組みです。マクロは、`#define` プリプロセッサディレクティブを使用して定義します。それ以降の翻訳単位でマクロ名が記述された箇所が、各マクロに対応する置換用テキストに置換されます。

- 3 C-SPY マクロは、C-SPY の機能を拡張するためにユーザが記述できるプログラムです。C-SPY マクロは、典型的な例として、ブレークポイントに対応付けて使用します。ブレークポイントに到達したときにそのマクロを実行し、周辺デバイスのシミュレーション、複雑な条件の評価、トレースの出力などを行うことができます。

C-SPY マクロ言語は、C の簡易版ですが、C ほど厳密なデータ型がありません。

### メールボックス

RTOS でのメールボックスとは、複数のタスク間の通信拠点です。タスクは、別のタスクのメールボックスにメッセージを保存することで、そのタスクにメッセージを送信できます。メールボックスは、メッセージキュー、メッセージポートとも呼びます。

### マングル化

マングル化とは、複雑な C/C++ 名を簡単な名前にマッピングするときに使用される技術です。ILINK メッセージの C/C++ シンボルに対して、マングル化した名前とデマングル化した名前の両方を生成できます。

### メモリ、リンカ設定ファイル

物理メモリ。物理メモリに含まれるユニット数および 1 つのユニットを構成するビット数。リンカ設定ファイルで定義されます。メモリは、常に 0x0 ~ (サイズ-1) からアドレスできます。

### メモリアクセスコスト

メモリアクセスコストは、アクセス実行に必要なクロックサイクル数かコードのバイト数で示されます。サイズの大きな命令や多数の命令が必要なメモリは、よりサイズが小さい命令や少ない命令でアクセスできるメモリよりもアクセスコストが高い、というように使用します。

### メモリエリア

メモリの領域を意味します。

### メモリバンク

バンクメモリ内のシーケンシャルメモリの最小単位。core の物理アドレス空間で一度に認識できるメモリバンクは 1 つです。

### メモリマップ

core で使用可能なさまざまなメモリエリアのマップ。

### メモリモデル

メモリ階層やシステムが処理できるメモリ容量を示します。アプリケーションで同時に使用できるメモリモデルは 1 つだけです。また、すべてのユーザモジュールやライブラリモジュールで同一のモデルを使用する必要があります。

### マイクロコントローラ

組込みシステムとして動作する 1 つの集積回路上のマイクロプロセッサ。マイクロコントローラは、CPU に加え、小容量の RAM、PROM、タイマ、入出力ポートを内蔵しています。

### マイクロプロセッサ

1 つ（または少数の）集積回路に内蔵された CPU。シングルチップマイクロプロセッサには、メモリ、メモリ管理、キャッシュ、浮動小数点演算ユニット、入出力ポート、タイマなどのコンポーネントを内蔵できます。このようなデバイスを、マイクロコントローラとも呼びます。

### モジュール

オブジェクト。オブジェクトファイルはモジュールを含み、ライブラリは 1 つ以上のオブジェクトを含みます。リンクの基本単位。モジュールには、シンボル定義（エクスポート）や外部シンボルへの参照（インポート）が含まれます。C/C++ のコンパイル時には、翻訳単位ごとに 1 つモジュールが生成されます。

### 複数ファイルのコンパイル

コンパイラで複数のソースファイルを 1 つのコンパイルユニットとしてコンパイルするテクニック。これにより、コンパイルユニット内の複数のソースファイルでのインライン化、クロスコール、クロスジャンプなど、プロシージャ間の最適化が可能になります。

## N

### ネスト割込み

割込みに対して別の割込みを実行できるシステムを、ネスト割込み機能を持つと言います。

## 非バンクメモリ

core の物理アドレス空間で、各メモリアドレスにつき 1 つ格納場所があること。

## 非初期化メモリ

リセット時に任意の値を持つことができる、またはソフトリセット時にリセット前の値を保持できるメモリ。

## 非初期化 section

起動時に初期化されないリード/ライト section。セクションを参照してください。

## 不揮発性ストレージ

バッテリバックアップ RAM、ROM、磁気テープ、磁気ディスクなどの、電源を切断してもデータを保持できるメモリデバイス。揮発性ストレージと比較してください。

## NOP

No operation（無動作命令）の略。何の処理も実行せず、遅延を発生させるために使用する命令。パイプラインアーキテクチャでは、NOP 命令を使用して、パイプラインを同期させることができます。パイプラインを参照してください。



## Objcopy

ELF フォーマットの絶対オブジェクトファイルを、たとえば、フォーマット Motorola-std や Intel-std などの絶対オブジェクトファイルに変換する GNU バイナリユーティリティ。elftool を参照してください。

## オブジェクト

ライブラリメンバのオブジェクトファイル。

## オブジェクトファイル、絶対

実行可能イメージを参照してください。

## オブジェクトファイル、再配置可能

ソースファイルをコンパイルまたはアセンブルした結果。オブジェクトファイルに使用されるファイルフォーマットは、デバッグ情報用の組込み DWARF を含む ELF です。

## 演算子

関数として使用されるシンボルで、引数が 2 つある場合は中置記法（+ など）、引数が 1 つだけの場合は前置記法（ビット単位の否定を示す ~ など）で使われます。多くの言語では、算術演算や論理演算などの組込み関数に演算子を使います。

## 演算子の優先順位

それぞれの演算子には優先順位が設定され、演算子とオペランドが評価される順番はそれによって決定されます。優先順位が一番高い演算子が最初に評価されます。演算子およびオペランドをグループ化し、式の評価順序を変更するには、括弧を使います。

## オプション

コンパイラやリンカなどツールの動作を制御するコマンドのセット。オプションは、コマンドラインや IDE によって指定できます。

## 出力イメージ

実行可能イメージを参照してください。

## オーバーレイ、リンカ設定ファイル

ブロックと似ているが、それぞれがブロック、オーバーレイ、セクションで構成されるいくつかの重複エンティティを含む。オーバーレイのサイズは、その最大要素で決定されます。オーバーレイメモリ領域のコードは、C-SPY デバッガでデバッグできません。



## パラメータの受渡し

呼び出し規約を参照してください。

## 周辺ユニット

メモリや入出力デバイスなど、プロセッサ以外のハードウェアコンポーネント。

## パイプライン

計算が流れる一連のステージで構成される構造。他の処理がパイプライン経由で実行中でも、パイプラインの開始地点で新しい処理を開始できます。

## 配置、リンカ設定ファイル

ブロック、オーバレイおよびセクションを領域に配置する方法。コードおよびデータが、使用可能な物理メモリに実際にどのように配置されるかを決定します。

## ポインタ

指定した型の他のオブジェクトのアドレスを格納するオブジェクト。

## #pragma

C/C++ プログラムのコンパイル中に、`#pragma` プリプロセッサディレクティブが検出されると、コンパイラを処理系定義に従って動作させます。これには、コンソールでの出力生成、それ以降のオブジェクトの宣言の変更、最適化レベルの変更、言語拡張の有効/無効の切替えなどがあります。

## プリエンブティブマルチタスク

RTOS のタスクは、より優先順位の高いプロセスが有効になるまでの間、実行を許可されます。割込みの結果、優先順位の高いタスクが有効になる場合があります。プリエンブティブとは、タスクが一定の実行時間（タイムスライス）を割り当てられている場合でも、プロセスの使用権を失うことがあることを意味します。割込みが発生すると、タスクスケジューラはそのときに有効なタスクの中で優先順位が最高のものを特定し、そのタスクに処理を切り替えます。特定されたタスクが割込み前に実行されていたタスクと異なる場合は、前のタスクは割込み時点の状態で一時停止します。ラウンドロビンと比較してください。

## プリプロセッサディレクティブ

実際のコードの解析を開始する前に実行されるディレクティブ。

## プリプロセッサ

C 形式のプリプロセッサを参照してください。

## プロセッサ選択

コンパイラがサポートする別のチップ構成。

## プログラムカウンタ (PC)

命令のアドレッシングに使用する特殊なプロセッサレジスタ。プログラムロケーションカウンタ (PLC) と比較してください。

## プログラムロケーションカウンタ (PLC)

IAR アセンブラで、現在の命令のコードアドレスを指定する際に使用します。PLC は、算術式で利用できる特別なシンボル（通常は \$）で表現されます。ロケーションカウンタ (LC) と呼ばれます。

## プロジェクト

ユーザアプリケーション開発プロジェクト。

## プロジェクトオプション

アプリケーションを実行するターゲットプロセッサなどの、プロジェクト全体に適用される一般オプション。

## PROM

Programmable Read-Only Memory（プログラマブルリードオンリーメモリ）の略。1 回だけプログラム可能な ROM。

# Q

## 修飾子

型修飾子を参照してください。

# R

## 範囲、リンカ設定ファイル

メモリ内での連続するアドレスの範囲。領域は、範囲で構成されます。

## リードオンリー section

コードまたは定数を含むセクション。セクションを参照してください。

## リアルタイムオペレーティングシステム (RTOS)

割込みが発生してから割込みハンドラが開始されるまでの遅延と、タスクのスケジューリング方法を保証するオペレーティングシステム。一般的に RTOS は、通常のデスクトップ用オペレーティングシステムよりも大幅に小さなサイズとなっています。リアルタイムシステムと比較してください。

## リアルタイムシステム

プロセスが時間に依存するコンピュータシステム。*リアルタイムオペレーティングシステム (RTOS)* と比較してください。

## 領域、リンカ設定ファイル

重複しない範囲のセット。範囲は、1つ以上のメモリに存在できます。ILINK の場合はブロック、オーバーレイ、セクションは、リンカ設定ファイルの領域に配置されます。

## 領域式、リンカ設定ファイル

領域リテラル、領域、リンカ設定ファイルで使用できる共通セット操作で構成される領域。

## 領域リテラル、リンカ設定ファイル

メモリ内で重複しない1つ以上の範囲セットを定義するリテラル。

## レジスタ

特にアクセス速度が高速で、プログラム実行時の一時記憶エリアとして確保されている小型オンチップメモリユニット。通常の容量は数バイトです。

## レジスタ定数

システム初期化の際に、プロセッサの専用レジスタにロードされる値。コンパイラは、定数が専用レジスタに格納されていることを前提に、コードを生成することができます。

## レジスタロック

通常のコード生成時に、コンパイラで一部のプロセッサレジスタの使用を禁止することを指します。多くの状況で使用します。たとえば、高速化のため、システムの一部をアセンブラ言語で記述する場合があります。この部分に、専用のプロセッサレジスタを割り当てる場合もあります。また、オペレーティングシステムやサードパーティ製ソフトウェアでレジスタが使用される場合もあります。

## レジスタ変数

通常、レジスタ変数は、関数の（スタック）フレームの代わりにレジスタに格納されるローカル変数を指します。レジスタ変数は、メモリアクセスが不要で、コンパイラでレジスタ変数を使用することで命令の実行

時間を短縮できるため、他の変数よりも大幅に高速です。*自動変数*を参照してください。

## リレー

ベニアの同義語。ベニアを参照。

## 再配置可能 section

リンク前にメモリ位置を固定していないセクション。

## リセット

システムの初期状態から再起動することを指します。リセットは、ハードウェア（ハードリセット）またはソフトウェア（ソフトリセット）から実行できます。ハードリセットは通常は電源投入と区別できませんが、ソフトリセットは区別できます。

## ROM モニタ

デバッグツールでの使用に特化した組込みソフトウェア。評価ボードチップの ROM に格納されていて、シリアルポートかネットワーク接続経由でデバッガと通信します。ROM モニタは、メモリアドレス（ロケーション）やレジスタの表示と修正、ブレークポイントの作成と削除、アプリケーションの実行などの基本コマンドセットを提供します。デバッガは、これらの基本コマンドを組み合わせて、プログラムのダウンロードやステップ実行など、より高度な機能を実現できます。

## ラウンドロビン

オペレーティングシステムでのタスクスケジュール。ここでは、すべてのタスクの優先順位レベルが同じであり、1つずつ順番に実行されます。プリエンプティブマルチタスクと比較してください。

## RTOS

*リアルタイムオペレーティングシステム (RTOS)* を参照してください。

## ランタイムライブラリ

オブジェクトファイルから参照される場合のみ、つまり条件付きでリンクされる場合のみ、実行可能イメージに含まれる再配置可能オブジェクトファイルの集合。

## ランタイムモデル属性

相互に互換性のない複数のモジュールがアプリケーションにリンクされないようにする仕組み。ランタイ



ム属性は、名前付きのキーと対応する値のペアで構成されます。

ILINK は、ライブラリを自動的に選択するときに、ランタイムモデル属性を使用して、正しいライブラリが使用されているか確認します。

## R 値

代入文の右辺に指定可能な値。単純に値だけがこれに該当します。L 値を参照してください。

# S

## 飽和演算

ほとんどの C/C++ の実装では、 $\text{mod-}2^N$  の補数ベースの演算を使用します。オーバーフロー時には、定義域で値がラップされます。つまり、 $(127 + 1) = -128$  となります。一方、飽和演算では、定義域でのラップが許可されていません。たとえば、定義域の上限値が 127 の場合、 $(127 + 1) = 127$  となります。飽和演算は、ラップが許可されているとオーバーフロー状態が致命的な問題になる信号処理で、よく使用されます。

## スケジューラ

RTOS でタスク切替えを担当する部分。また、実行を許可するタスクの選択も担当します。スケジューリングアルゴリズムには多種ありますが、ほとんどは静的スケジューリング（コンパイル時に実行）または動的スケジューリング（次に実行するタスクを、タスク切替え時のシステムの状態に応じて実行時に選択）のいずれかです。ほとんどのリアルタイムシステムでは、システムのリアルタイム要件違反を排除するため、静的スケジューリングが使用されています。

## スコープ

アプリケーションコード内で、関数や変数を名前で見ることができる部分。ある項目のスコープは、ファイル、関数、ブロックのいずれかに制限されることがあります。

## セクション

データまたはテキストのいずれかを含むエンティティ。通常は 1 つ以上の変数または関数です。セクションは、最小のリンク可能ユニットです。

## セクション属性

各セクションは名前と属性を持つ。属性は、セクションの内容、つまり、セクションの内容がリードオンリー、リード/ライト、コード、データなどを定義します。

## セクションフラグメント

セクションの一部。通常は変数または関数です。

## セクションの選択

リンカ設定ファイルにおいて、セクションセクタを使用してセクションのセットを定義します。セクションは、複数の選択の一部となる可能性がある場合、最も制限の厳しいセクションセクタに属します。セクタには、セクション属性（セクションの内容で選択）、セクション名（セクション名で選択）、オブジェクト名（特定のオブジェクトから選択）の 3 種類があり、これらは個別に使用したり、組み合わせて使用してセクションのセットを選択したりできます。

## セマフォ

リソースへの排他的アクセスを保証するために使用するフラグの一種。リソースとしては、ハードウェアポート、構成メモリ、変数などがあります。複数のタスクが同一リソースにアクセスする必要がある場合は、リソースにアクセスする部分のコード（クリティカルセクション）をすべてのタスクに対して排他的にする必要があります。これには、そのリソースを保護するセマフォを取得し、他のタスクからそのリソースを遮断します。他のタスクがそのリソースを使用する場合は、そのタスクもセマフォを取得する必要があります。セマフォが使用中の場合は、セマフォが解放されるまで待機する必要があります。セマフォが解放された後は、2 番目のタスクが実行を許可され、セマフォを取得してリソースへの排他的アクセスを実行できます。

## 重要度

何らかの問題を検出したときにアセンブラ、コンパイラ、デバッガから返される診断応答の重要度。通常、重要度は、リマーク、ワーニング、エラー、致命的なエラーの 4 段階です。リマークは問題の可能性を示すだけですが、致命的なエラーの場合はプログラミングツールが処理の完了前に終了したことを示します。

## 共有

いくつかの方法でアドレスが可能な物理メモリ。ILINK の場合は、リンク設定ファイルで定義します。

## ショートアドレッシング

多くの core では、内部 RAM、メモリマップド I/O へのアクセスを効率的に行うため、特別なアドレッシングモードがあります。データポインタを参照してください。

## 副作用

C/C++ の式がシステムの状態を変更することを、副作用があると言います。例として、変数への代入や、変数に後置インクリメント演算子を使用する場合などがあります。C/C++ の規格では、副作用のある変数を式で複数回使用しないように規定されています。たとえば、次の文はこのルールに違反します。

```
*d++ = *d;
```

## 信号

シグナルは、イベントベースのタスク間通信を提供します。1 つのタスクは、他の 1 つ以上のタスクからのシグナルを待つことがあります。待っているシグナルをタスクが受信すると、実行が続行されます。RTOS では、シグナルを待つタスクは処理時間を費やさないため、他のタスクを実行できます。

## シミュレータ

ホスト上で実行し、ターゲットプロセッサと可能な限り同一に動作するデバッグツール。シミュレータは、ハードウェアが使用できないときか、ハードウェアをデバッグに使わないときに、アプリケーションのデバッグのために使用します。物理的な周辺デバイスには通常接続しません。シミュレーションされたプロセッサは、多くの場合は実際のハードウェアよりも（場合によっては大幅に）低速になります。

## ステップ実行

デバッガで一度に 1 つずつ命令や C 言語の文を実行することを指します。

## スケルトンコード

ユーザがコードを特定用途化できる、未完成のコードフレームワーク。

## 特殊機能レジスタ (SFR)

core のハードウェアコンポーネントに対するリード/ライトに使用するレジスタ。

## スタックフレーム

データオブジェクト（保持レジスタ、ローカル変数、特定のスコープ用に一時的に保持する必要のある他のデータオブジェクト）を含むデータ構造（通常は関数）。

以前のコンパイラでは、関数全体でスタックフレームのサイズとレイアウトが固定されていましたが、最近のコンパイラでは、関数内の任意の箇所/時間で、動的にレイアウトとサイズを変更できる場合があります。

## スタック section

スタック用エリアを確保する section または section。ほとんどのプロセッサは呼出しとパラメータで同一のスタックを使用しますが、一部のプロセッサでは個別のスタックを使用します。

## 標準ライブラリ

C/C++ 標準で定義されている C/C++ ライブラリ関数、および浮動小数点ルーチンなどのコンパイラのサポートルーチン。

## 静的オブジェクト

リンク時にメモリが割り当てられ、システム起動時（または最初の使用時）に作成されるオブジェクト。*動的オブジェクト*と比較してください。

## 静的オーバーレイ

パラメータや自動変数に動的配置方式を使用する代わりに、リンク時にパラメータや自動変数にエリアを割り当てます。この方法ではスタックの使用効率は最悪になりますが、スタックアクセスが高価な、またはスタックアクセスがまったくない小型チップには、適している場合があります。

## 静的割当てメモリ

この種のメモリは、リンク時に 1 度だけ割り当てられ、アプリケーションの実行終了まで有効です。global または static として宣言された変数が、この方法で割り当てられます。

## 構造体値

構造体および共用体の集合名。構造体は、メモリに連続的に配置されたデータオブジェクトの集合です（データオブジェクト間にパッドバイトが挿入されていることもある）。共用体は、同一メモリアドレス（ロケーション）を共有するデータの集合です。

## シンボル位置

正確なアドレスがわからないためにシンボル名を使用している位置。

# T

## ターゲット

- 1 アーキテクチャ。
- 2 ハードウェア。アプリケーション開発対象の組み込みシステムを指します。この用語は、通常はシステムとホストシステムの区別に使用します。

## タスク（スレッド）

タスクは、システムでの実行スレッドです。多くの並列で実行されるタスクを含むシステムを、マルチタスクシステムと呼びます。プロセッサは一度に1つの命令ストリームだけを実行するため、ほとんどのシステムは何らかのタスク切替えメカニズム（多くの場合コンテキスト切替えと呼ぶ）を実装し、処理時間をすべてのタスクに配分します。次に実行を許可するタスクの決定プロセスを、スケジューリングと呼びます。一般的なスケジューリング方法として、プリエンプティブマルチタスクとラウンドロビンがあります。

## 仮定義

定義が同一で、絶対アドレスである場合に、複数のファイルで定義可能な変数。

## ターミナル I/O

C-SPY の端末シミュレーションウィンドウ。

## タイマ

プログラム実行とは無関係にカウントを実行する周辺デバイス。

## タイムスライス

RTOS で、タスクスケジューリングアルゴリズムを実行せずに1つのタスクを実行可能な（最長）時間。タスク切替えまでに、複数の連続したタイムスライスにわたって1つのタスクが実行されることがあります。また、プリエンプティブシステムで、より優先順位の高いタスクが割り込みにより実行された場合のように、タスクが自身に割り当てられたタイムスライス全体を使用できないこともあります。

## 翻訳単位

ソースファイルと、プリプロセッサディレクティブ `#include` でインクルードされるすべてのヘッダファイルやソースファイル（`#if` や `#ifdef` などの条件プリプロセッサディレクティブで省略された行を除く）を合せたもの。

## トラップ

命令ストリームに特殊な命令を挿入することで実行される割り込み。多くのシステムでは、トラップを使用して、オペレーティングシステム関数を呼び出します。ソフトウェア割り込みとも呼びます。

## 型修飾子

標準 C/C++ では `const`、`volatile`。IAR システムズのコンパイラは、通常はメモリや他の型属性用にターゲット固有の型修飾子を追加します。

# U

## UBROF (Universal Binary Relocatable Object Format)

使用する製品パッケージに XLINK リンカが含まれる場合に、一部の IAR システムズのプログラミングツールにより生成されるファイルフォーマット。

# V

## 値式、リンカ設定ファイル

C 式と同様の構文を使用する式で構成できる定数値。

## ベニア

Arm と Thumb などモードで不一致がある場合、または呼び出し命令がそのアドレスに達しない場合、呼出し側と呼出し先の間のスプリングボードとして挿入される小さなコード。

## 仮想アドレス（論理アドレス）

コンパイラ、リンカ、ランタイムシステムによって、使用前に物理メモリアドレスに変換する必要があるアドレス。仮想アドレスはアプリケーションで認識されるアドレスであり、システムの他の部分で認識されるアドレスとは異なる場合があります。

## 仮想空間

IAR Embedded Workbench のエディタの機能で、実際の文字のある領域外に挿入ポイントを移動できます。

## 揮発性ストレージ

揮発性記憶デバイスに保存したデータは、そのデバイスの電源を切った場合は保持されません。電源を切った後もデータを保持するには、不揮発性ストレージに保存する必要があります。C 言語のキーワードである `volatile` と混同しないでください。不揮発性ストレージと比較してください。

## ノイマンアーキテクチャ

命令とデータの両方が共通のデータチャネルで転送されるコンピュータアーキテクチャ。ハーバードアーキテクチャと比較してください。

# W

## ウォッチポイント

C 言語の変数や式の値を、アプリケーション実行中に C-SPY の [ウォッチ] ウィンドウでトレースします。

# X

## XAR

UBROF 形式でアーカイブ（ライブラリ）を作成する IAR ツール。XAR は IAR Embedded Workbench に付属しています。

## XLIB

UBROF フォーマットでアーカイブ（ライブラリ）を作成したり、オブジェクトコードのリスト化、絶対オブジェクトファイルを別のフォーマットの絶対オブジェクトファイルに変換するための IAR ツール。XLIB は IAR Embedded Workbench に付属しています。

## XLINK

UBROF 出力フォーマットを使用する IAR XLINK リンカ。

# Z

## ゼロ初期化済み section

起動時にゼロに初期化されるセクション。セクションを参照してください。

## ゼロオーバーヘッドループ

ループ条件（ループ開始地点へ戻る分岐を含む）の処理に時間がまったくかからないループ。通常はプロセッサの特別なハードウェア機能として実装されるため、利用できないアーキテクチャがあります。

## ゾーン

プロセッサによって、メモリアーキテクチャは大幅に異なります。ゾーンとは、C-SPY で名前付きメモリエリアを示す用語です。たとえば、個別にアドレッシング可能なコードおよびデータメモリを持つプロセッサでは、少なくとも 2 つゾーンがあります。複雑なバンクメモリ方式を採用したプロセッサの場合は、ゾーンが複数存在する場合があります。

## あ

アイコン	
本ガイド	20
SVN の状態	121
[ワークスペース] ウィンドウ	112
アクセラレータキー。ショートカットキーを参照	
アクティブに設定 ([ワークスペース] ウィンドウのコンテキストメニュー)	116
アサーション、ビルドアプリケーション	101
アセンブラオプション	245
プリプロセッサ	249
リスト	247
言語	245
出力	247
診断	250
アセンブラオプション、定義	277
アセンブラソースファイル ([ワークスペース] ウィンドウアイコン)	112
アセンブラディレクティブ	
エディタのテキストスタイル	146
定義	277
アセンブラニーモニック (出力リストファイルの 設定)	237
アセンブラのコメント、エディタのテキスト スタイル	146
アセンブラの出力、デバッグ情報を含める	247
アセンブラファイルの出力 (コンパイラオ プション)	237
アセンブラプリプロセッサ	249
アセンブラリストファイル	
クロスリファレンス、生成	248
コンパイラ呼出しフレーム情報	237
タブによる移動量、指定	249
ヘッダ、含む	248
ページあたりの行数、指定	249
条件付き情報、指定	247-248
生成	247
アセンブラ言語、定義	277

アセンブラ行のみ (インクルードリスト化の 設定)	248
アセンブラ、コマンドラインバージョン	23
アドバンスド (リンカオプション)	264
アドレス式、定義	277
アプリケーションで定義 (デフォルトプログ ラムエントリのオーバーライド設定)	262
アプリケーション、定義	277
アライン (ロウバイナリイメージの設定)	263
アラインメント (チェックサム生成の設定)	270
アルゴリズム (チェックサム生成の設定)	271
アーカイブ、定義	277
アーキテクチャ、定義	277

## い

インクルードファイル	188
コンパイラ、パスを指定	237, 249
パスの指定	238, 249
定義	283
インストールされるファイル	187
インクルード	188
ドキュメント	188
ライブラリ	188
実行可能	189
インストールパス、デフォルト	187
インストール先ディレクトリ	19
インデントサイズ (エディタのオプション)	59
インデント、エディタ	141
インラインアセンブラ、定義	283
インライン化、定義	283

## う

ウィンドウ	
画面上の編成	25
画面上の編成方法	33
ウォッチポイント、定義	292
エイリアス ([キーカスタマイズ] オプション)	57

エディタ	
インデント	141
コマンド	147
コードテンプレート	144
コードの折りたたみ	142
コードの入力補完	143
ショートカットキー	181
ステータスバー、使用	148
パラメータのヒント	144
外部	41
括弧と中括弧のマッチング	142
環境のカスタマイズ	140
関数のショートカット	148, 155
語句の入力補完	143
使用	139
分割バー	154
options	59
エディタ ([外部エディタ] オプション)	64
エディタウィンドウ	152
エディタウィンドウでの検索	150
エディタセットアップファイル ([IDE オプション] ダイアログボックス)	66
エディタセットアップファイル、オプション	66
エディタフォント ([エディタ色とフォント] オプション)	67
エミュレータ (C-SPY ドライバ)、定義	281
エラーとして処理 (コンパイラオプション)	240
エラーとして処理 (リンカオプション)	269
エラーメッセージ	
コンパイラ	240
リンカ	269
エラー解析 (C-RUN)、対象ドキュメント	17
エラー、修正	129
エンコーディング、エディタオプション	60
エンコード (コンパイラオプション)	241
エンディアンモード (一般オプション)	218
エントリシンボル (デフォルトプログラムエントリのオーバーライド設定)	262

## お

オブジェクトファイル (出力ディレクトリの設定)	221
オブジェクトファイルまたはライブラリ ([ワークスペース] ウィンドウアイコン)	112
オブジェクトファイル (再配置可能)、定義	286
オブジェクトファイル (絶対)、定義	286
オブジェクト、定義	286
オプション ([ワークスペース] ウィンドウのコンテキストメニュー)	114
オプションの指定	228
オプション、定義	286
オンラインドキュメント	
ターゲット固有、ディレクトリ	188
[ヘルプ] メニューから使用可能	216
オーバーレイ、定義	286

## か

ガイドラインの確認	15
カスタムキーワードファイルの使用 (エディタのオプション)	66
カスタムツール設定 (カスタムビルドオプション)	255
カスタムビルド	123
使用	132
カスタムビルド構成	124
カスタム変数、引数変数として	92
カテゴリ、[オプション] ダイアログボックス	127, 134

## き

キーカスタマイズ、定義	284
キーカスタマイズ ([IDE オプション] ダイアログボックス)	56

キーボードのショートカット。ショートカットキーを参照	
キーワード	
エディタでの構文カラーの指定	147
コメント	153
言語拡張の有効化	230
定義	284
キー操作のまとめ、エディタ	181

## く

グラフィカルスタック表示とスタック使用トラッキングを有効にする ([スタック] オプション)	79
-clean (iarbuild コマンドラインオプション)	131
クリーン ([ワークスペース] ウィンドウのコンテキストメニュー)	115
グループの追加 ([ワークスペース] ウィンドウのコンテキストメニュー)	115
グループ、定義	101
クロスリファレンス (アセンブラオプション)	248

## こ

コア (プロセッサ選択の設定)	218
コスト。メモリアクセスコストを参照。	
このガイドで使用されている規則	19
コマンド ([外部エディタ] オプション)	65
コマンドプロンプトアイコン、本ガイド	20
コマンドライン (カスタムビルドオプション)	256
コマンドラインオプション	
表記規則	19
[ツール] メニューから指定	41
コマンドラインオプションの使用 (コンパイラのオプション)	242, 252, 273
コミット (Subversion 管理メニュー)	120
コメントの保持 (ファイルへのプリプロセッサ出力の設定)	238
コメントの doxygen キーワード	153
コメント、ドキュメントのコメントタイプ	153
コンテキストメニュー、定義	279

コンパイラオプション	227
エンコード	241
コード	232
コードおよびリードオンリーのデータ	233
リスト	236
リード/ライトデータ	233
言語 1	229
言語 2	231
最適化	233
出力	235
診断	239, 268
定義	279
MISRA-C	240
コンパイラのリストファイル	
アセンブラニーモニック、含める	237
ソースコード、含める	237
生成	236
コンパイラの呼出しフレーム情報のインクルード (アセンブラ出力ファイル設定)	237
コンパイラの診断	237
コンパイラ出力	
デバッグ情報を含める	235
デフォルトディレクトリのオーバーライド	221
コンパイラ、コマンドラインバージョン	23
コンパイル ([ワークスペース] ウィンドウのコンテキストメニュー)	114
コンバータオプション	253
コンピュータスタイル、表記規則	19
コード	
スケルトン、定義	290
テスト	129
バンク、定義	278
コードおよびリードオンリーのデータ (コンパイラオプション)	233
コードセクション名 (コンパイラオプション)	235
コードセクション、定義	279
コードテンプレートの使用 (エディタのオプション)	66
コードテンプレート、エディタで使用	144
コードのテンプレート、使用	144

コードの自動補完およびパラメータのヒント (エディタオプション) .....	62
コードの折りたたみ、エディタ .....	142
コードの入力補完、エディタ .....	143
コードポインタ、定義 .....	279
コードメモリ内のデータ Read 不生成 (アセンブラオプション) .....	246
コードメモリ内のデータ Read 不生成 (コンパイラオプション) .....	233
コードメモリ、使用部分のフィル .....	270
コードモデル、定義 .....	279
コールフレーム情報 定義 .....	278

## さ

サイズ (チェックサム生成の設定) .....	270
サイズの最適化 .....	234
サイズ制約なし (レベル設定) .....	234
サンプルプロジェクト .....	27
ダウンロード中 .....	27
実行 .....	29
サービス ([外部エディタ] のオプション) .....	65

## し

シグナル、定義 .....	290
シミュレータ、定義 .....	290
ショートアドレッシング、定義 .....	290
ショートカットキー .....	147
カスタマイズ .....	56
ショートカットキーを押してください ([キーカスタマイズ] オプション) .....	56
ショートカットメニュー。コンテキストメニューを 参照	
シンボル リンカでの定義 .....	267
定義 .....	291
ユーザシンボルも参照	
シンボル (ロウバイナリイメージの設定) .....	263

シンボルをキープ (リンカオプション) .....	262
シンボル位置、定義 .....	291
シンボル定義オプション .....	238, 250
シンボル定義 (リンカオプション) .....	267
シンボル、定義 .....	238, 250

## す

スクロール、～のためのショートカットキー .....	147
スケジューラ (RTOS)、定義 .....	289
スケルトンコード、定義 .....	290
スコープ、定義 .....	289
スタック ([IDE オプション] ダイアログボック ス) .....	79
スタックしきい値の超過時にワーニング ([スタック] オプション) .....	79
スタックセグメント、定義 .....	290
スタックの使用量解析を有効にする (リンカオプション) .....	265
スタックフレーム、定義 .....	290
スタックポインタが境界外の時にワーニング ([スタック] オプション) .....	79
ステップ実行、定義 .....	290
ステータスバー .....	49
すべてのワーニング (ワーニング設定) .....	251
すべてのワーニングをエラーとして処理 (コンパイラオプション) .....	240
すべてのワーニングをエラーとして処理 (リンカオプション) .....	269
すべてのワーニングを無効にする (アセンブ ラオプション) .....	251
すべてリセット ([キーカスタマイズ] オプション) .....	57
すべてを再ビルド ([ワークスペース] ウィンドウのコンテキストメニュー) .....	114
すべて保存 ([ファイル] メニュー) .....	197
スペースによるインデント (タブキーの機能設定) ..	59
スレッド、定義 .....	291



# せ

セクション (ロウバイナリイメージの設定) .....	263
セクションの属性、定義 .....	289
セクションフラグメント、定義 .....	289
セクション選択、定義 .....	289
セマフォ、定義 .....	289
ゼロオーバーヘッドループ、定義 .....	292
ゼロ初期化されたセクション、定義 .....	292

# そ

その他のファイル ([ワークスペース] ウィンドウアイコン) .....	112
ソースが複数の関数インスタンスに分かれる場合 ...	77
ソースコード	
コンパイラリストファイルに含める .....	237
テンプレート .....	144
ソースコード管理 ([IDE オプション] ダイアログボックス) .....	75
ソースのインクルード (アセンブラ出力ファイル 設定) .....	237
ソースファイル	
パス .....	101, 153
プロジェクト管理 .....	101
編集 .....	140
ソースファイルの編集 .....	140
ソース参照ログ (表示メニュー) .....	177
ゾーン、定義 .....	292

# た

タイプ ([外部エディタ] のオプション) .....	64
タイマ、定義 .....	291
タイムスライス、定義 .....	291
タスク、定義 .....	291
タブキーの機能 (エディタのオプション) .....	59
タブサイズ (エディタのオプション) .....	59
タブを挿入 (タブキーの機能設定) .....	59-60

タブ間隔 (アセンブラオプション) .....	249
ターゲット、定義 .....	291
ターゲット (一般オプション) .....	217
ターミナル I/O ([IDE オプション] ダイアロ グボックス) .....	81
ターミナル I/O ウィンドウ、定義 .....	291

# ち

チェックサム	
生成 .....	270
生成ツール .....	283
定義 .....	279
CRC .....	279
チェックサム (リンカオプション) .....	270
チェックサムユニットサイズ (チェックサム 生成の設定) .....	272
チェックサム生成 (リンカオプション) .....	270

# つ

ツールアイコン、本ガイド .....	20
ツールチェーン	
概要 .....	23
拡張 .....	123
ツールの設定 ([ツール] メニュー) .....	83
ツールバー、IDE .....	46
カスタマイズ .....	35
ツール、ユーザ設定 .....	83

# て

ディレクトリ	
コンパイラ、標準のインクルードを無視 ...	237, 249
ルート .....	187
common .....	189
ディレクトリ構成 .....	187
テキストの選択、~のためのショートカットキー ..	147

テキストファイル ([ワークスペース] ウィンドウアイコン) .....	112
テキスト出力ファイルのエンコード (コンパイ ラ オプション) .....	241
テキスト出力ファイルのエンコード (リンカ オ プション) .....	272
デジタル信号プロセッサ、定義 .....	280
テスト、コード .....	129
デバイス (プロセッサ選択の設定) .....	218
デバイスドライバ、定義 .....	280
デバイス記述ファイル .....	188
定義 .....	280
デバイス選択ファイル .....	188
デバッグ ([IDE オプション] ダイアログボッ クス) .....	77
デバッグ情報	
アセンブラで生成 .....	247
コンパイラ、生成 .....	235
デバッグ情報の生成 (アセンブラオプション) ....	247
デバッグ情報の生成 (コンパイラオプション) ....	235
デバッグ前にメイクを実行 ([IDE プロジェクト] オプション) .....	70
デフォルトのインストールパス .....	187
デフォルトのソースファイルのエンコード (コンパイラ オプション) .....	241
デフォルトのプログラムエントリをオーバーライド (リンカオプション) .....	261
デフォルトの出荷時設定の復元 .....	128
デフォルトの入力ファイルのエンコード (コンパイラ オプション) .....	242
デフォルトの入力ファイルのエンコード (リンカ オプション) .....	272
デフォルト整数フォーマット (IDE オプション) ....	78
デマングル、定義 .....	280
データポインタ、定義 .....	280
データモデル、定義 .....	280
データ表現、定義 .....	280

## と

ドキュメント .....	187
オンライン .....	188
ガイドの概要 .....	17
本ガイド .....	15
本ガイドの概要 .....	16
ドキュメントのコメント タイプ .....	153
ドッキング可能なウィンドウ .....	25
ドラッグアンドドロップ	
エディタウィンドウのテキスト .....	142
[ワークスペース] ウィンドウのファイル .....	102
トラップ、定義 .....	291

## な

なし (レベル設定) .....	234
------------------	-----

## ね

ネスト割込み、定義 .....	285
-----------------	-----

## の

ノイマンアーキテクチャ、定義 .....	292
----------------------	-----

## は

バイトオーダ、設定 .....	218
パイプライン、定義 .....	286
パス	
インクルードファイル .....	238, 249
コンパイラのインクルードファイル .....	237, 249
ソースファイル .....	153
相対、Embedded Workbench .....	102, 153
バックトレース情報、定義 .....	277
バッチビルド .....	130
バッチファイル	
定義 .....	278

[ツール] メニューから指定 .....	41
バッファした書込み (リンクオプション) .....	225
パラメータ	
コマンドラインからのビルド時 .....	131
表記規則 .....	19
パラメータのヒント、エディタ .....	144
バンクコード、定義 .....	278
バンクデータ、定義 .....	278
バンクメモリ、定義 .....	278
バンク切替えルーチン、定義 .....	278
バンク切替え、定義 .....	277
バージョン管理システムメニュー .....	120
バージョン管理システム ([ワークスペース] ウィンドウコンテキストメニュー) .....	116
バージョン番号	
本ガイド .....	2
Embedded Workbench .....	216
ハーバードアーキテクチャ、定義 .....	282

## ひ

ビッグ (エンディアンモードの設定) .....	218
ビットフィールド、定義 .....	278
ビット順 (チェックサム生成の設定) .....	271
ビューア拡張子の編集 ([ツール] メニュー) .....	87
ビルド	
コマンド .....	128
コマンドラインから .....	131
ファイルの除外 .....	114
処理 .....	123
前後のアクション .....	130
options .....	69
-build (iarbuild コマンドラインオプション) .....	131
ビルドアクション .....	130
ビルドアクションの構成 (ビルドアクション オプション) .....	257
ビルドウィンドウ ([表示] メニュー) .....	135
ビルドからファイルを除外 .....	114
ビルドから除外されたグループ ([ワークスペース] ウィンドウアイコン) .....	112

ビルドから除外されたソースファイル ([ワークスペース] ウィンドウアイコン) .....	112
ビルドを停止 ([ワークスペース] ウィンドウの コンテキストメニュー) .....	115
ビルド構成	
作成 .....	104
定義 .....	101
ビルド処理終了後にサウンドを再生 ([IDE プロジェクト] オプション) .....	71
ビルド前にエディタウィンドウを保存 ([IDE プロジェクト] オプション) .....	70
ビルド前にワークスペースとプロジェクトを保存 ([IDE プロジェクト] オプション) .....	70
ビルド中に検出されたエラーの修正 .....	129
ヒープサイズ、定義 .....	282
ヒープメモリ、定義 .....	282

## ふ

ファイル	
間のナビゲート .....	99
編集 .....	140
ファイル (ロウバイナリイメージの設定) .....	263
ファイルエンコーディング (エディタオ プション) .....	60
ファイルグループ ([ワークスペース] ウィンドウアイコン) .....	112
ファイルタイプ	
インクルード .....	188
コマンドライン拡張 .....	194
デバイスの選択 .....	188
デバイス記述 .....	188
ドキュメント .....	188
フラッシュローダアプリケーション .....	188
プロジェクトテンプレート .....	188
ヘッダ .....	188
ライブラリ .....	188
リンク設定ファイル .....	188
構文カラー表示設定 .....	188
特殊機能レジスタの記述ファイル .....	188

C-STAT .....	188
drivers .....	188
readme .....	188
ファイルの追加 ([ワークスペース] ウィンドウのコンテキストメニュー) .....	115
ファイルプロパティ ([ワークスペース] ウィンドウのコンテキストメニュー) .....	116
ファイルへのプリプロセッサ出力 (コンパイラオ プション) .....	238
ファイル拡張子。ファイル名拡張子を参照	
ファイル名拡張子 .....	192
デフォルト以外 .....	37
cfg、構文強調表示 .....	67
eww、ワークスペースファイル .....	27
ファイル名拡張子 (カスタムビルドオプション) ..	255
フィルパターン (フィルの設定) .....	270
フィル、定義 .....	282
フォント	
エディタ .....	67
プロポーショナル .....	55
固定幅 .....	55
フォーマット指定子、定義 .....	282
ブックマーク	
エディタで表示 .....	61
追加 .....	147
ブックマークの表示 (エディタのオプション) .....	61
プライマリ ([キーカスタマイズ] オプション) .....	57
ブラウザ情報を生成 ([IDE プロジェクト] オプション) .....	71
プラグイン	
arm (サブディレクトリ) .....	188
common (サブディレクトリ) .....	189
#pragma ディレクティブ、定義 .....	287
フラッシュローダアプリケーション .....	188
プリインクルードファイル (コンパイラオ プション) .....	238
プリエンブティブマルチタスク、定義 .....	287
プリビルドコマンドライン (ビルドアクシ ョンオプション) .....	258

プリプロセッサ	
定義。C 形式プリプロセッサを参照	
文字列変数を初期化するマクロ .....	130
NDEBUG シンボル .....	101
プリプロセッサオプション .....	237
プリプロセッサディレクティブ	
エディタのテキストスタイル .....	146
定義 .....	287
プリプロセッサ (アセンブラオプション) .....	249
フルサイズでの結果 (チェックサム生成の設定) ..	271
ブレークポイント、定義 .....	278
プログラミング経験 .....	15
プログラムカウンタ、定義 .....	287
プログラムロケーションカウンタ、定義 .....	287
プログラム可能な出力フォーマット (コンバー タオプション) .....	254
プログラム、アプリケーションも参照	
プロジェクト	
グループとファイルの排除 .....	104
グループ、作成 .....	104
ビルド .....	128
バッチで .....	130
ビルド構成、作成 .....	104
ファイルの追加 .....	104
ワークスペース、作成 .....	104
管理 .....	97
作成 .....	99, 104
定義 .....	99, 287
例 .....	27
ダウンロード中 .....	27
実行 .....	29
プロジェクト IDE ([IDE プロジェクト] オプション) .....	69
プロジェクト ([ワークスペース] ウィンドウアイコン) .....	112
プロジェクトオプション、定義 .....	287
プロジェクトメイク、オプション .....	69
プロジェクトモデル .....	97
プロジェクトを Subversion に接続 (Subversion 管理メニュー) .....	121

プロジェクトを Subversion プロジェクトから切断 (Subversion 管理メニュー) .....	121
プロジェクト接続を有効化 ([IDE プロジェクト] オプション) .....	71
プロセッサモード (コードオプション) .....	232
プロセッサ選択、定義 .....	287
プロセッサ選択 (一般オプション) .....	218
ブロック、定義 .....	278
プロトタイプの強制 (C 派生言語の設定) .....	230
プロトタイプ、存在の検証 .....	230
プロパティ (Subversion 管理メニュー) .....	121
プロポーショナルフォント (IDE オプション) .....	55
フローティングウィンドウ .....	25

## へ

ヘッダファイル .....	188
迅速なアクセス .....	149
ヘッダファイル ([ワークスペース] インドウアイコン) .....	112
ヘッダを含む (アセンブラオプション) .....	248
ベニア、定義 .....	292

## ほ

ポインタ	
スタックポインタが範囲外の時にワーニング .....	79
定義 .....	287
ポストビルドコマンドライン (ビルドアクションオプション) .....	258
ホスト、定義 .....	282
ボタンの外観ダイアログボックス .....	53
ポップアップメニュー。コンテキストメニューを参照	

## ま

マイクロコントローラ、定義 .....	285
マイクロプロセッサ、定義 .....	285
マクロテキスト (インクルードリスト化の設定) ...	248

マクロの引用符 (アセンブラオプション) .....	246
マクロ拡張子 (インクルードリスト化の設定) .....	248
マクロ実行情報 (インクルードリスト化の設定) ...	248
マクロ、定義 .....	284
マップファイル、リンカからの生成 .....	267
マルチタスク、定義 .....	287
マルチバイト文字サポートを有効にする (一般オプション) .....	224-225
マングル化、定義 .....	285

## め

メイク ([ワークスペース] ウィンドウのコンテキストメニュー) .....	114
メッセージ ([IDE オプション] ダイアログボックス) .....	68
メニュー .....	195
メニューバー .....	45
メニュー ([キーカスタマイズ] オプション) .....	56
メモリ	
定義 .....	285
メモリアクセスコスト、定義 .....	285
メモリエリア、定義 .....	285
メモリバンク、定義 .....	285
メモリマップ、定義 .....	285
メモリモデル、定義 .....	285
メモリ、使用部分のフィル .....	270
メールボックス (RTOS)、定義 .....	285

## も

モジュール、定義 .....	285
モード (一般オプション) .....	220

## ゆ

ユーザシンボルで大文字と小文字を区別する (アセンブラオプション) .....	246
-----------------------------------------	-----

# ら

ライブラリ (出力ファイル設定).....	221
ライブラリオプション 1 (一般オプション) .....	224
ライブラリビルダ、出力オプション.....	276
ライブラリファイル .....	188
ライブラリ関数	
オンラインヘルプ .....	18
ステップイン情報の回避 (ソースを持つ関数のみ) .....	78
設定可能 .....	189
ライブラリ設定ファイル	
定義 .....	284
IDE からの指定 .....	222
ライブラリ設定 (一般オプション).....	222
ライブラリ低レベルインタフェースの実装 (一般オプション) .....	222
ライブラリ、定義 .....	288
ライブラリ (リンカオプション).....	261
ライブラリ (一般オプション).....	222
ラウンドロビン、定義 .....	288
ラベル (c) ([自動インデントの設定] オプション) .....	63
ランタイムエラー解析、対象ドキュメント .....	17
ランタイムモデル属性、定義.....	288
ランタイムライブラリ	
指定.....	222
定義.....	288

# り

リアルタイムオペレーティングシステム、定義....	287
リアルタイムシステム、定義.....	288
リスト (リンカオプション).....	266
リストファイル	
アセンブラ	
クロスリファレンス、生成.....	248
コンパイラランタイム情報.....	237
タブによる移動量、指定.....	249

ヘッダ、含む .....	248
ページあたりの行数、指定.....	249
条件付き情報、指定.....	247-248
コンパイラ	
アセンブラニーモニック、含める.....	237
ソースコード、含める.....	237
生成 .....	236
リストファイル (出力ディレクトリの設定).....	221
リストファイルの出力 (アセンブラオプション) ...	247
リストファイルの出力 (コンパイラオプション) ...	236
リストを含む (アセンブラオプション).....	248
リスト (アセンブラオプション).....	247
リスト (コンパイラオプション).....	236
リセット、定義 .....	288
リトルエンディアン (エンディアンモードの設定) .....	218
リマークとして処理 (コンパイラオプション) .....	239, 269
リマークを有効化 (コンパイラオプション) ...	239, 268
リマーク、診断の分類 .....	239, 269
リリースノート .....	188
リレー、定義 .....	288
リンカ	
オプションの設定 .....	259
コマンドラインバージョン .....	23
リンカオプション .....	259
アドバンスト.....	264
ライブラリ.....	261
リスト.....	266
最適化.....	263
出力.....	265
入力.....	262
表記規則.....	19
Checksum .....	270
Config (設定) .....	260
define .....	267
リンカコマンドファイル。リンカ設定ファイルを参照	
リンカシンボル、定義 .....	267
リンカの設定ファイル、定義.....	283

リンカマップファイルの表示（リンカオプション）	267
リンカ設定ファイル	
ディレクトリ	188
リンカで指定	260
定義	284
リンカ設定ファイル（リンカオプション）	260
リードオンリーセクション、定義	287
リード/ライトデータ（コンパイラオプション）	233

## る

ルートディレクトリ	187
-----------	-----

## れ

レイアウト、Embedded Workbench の	26
レジスタ	
定義	288
inc ディレクトリのヘッダファイル	188
レジスタロック、定義	288
レジスタ定数、定義	288
レジスタ変数、定義	288
レベル（コンパイラオプション）	234

## ろ

ロウバイナリイメージ（リンカオプション）	263
-log（iarbuild コマンドラインオプション）	131–132
ログ（Subversion 管理メニュー）	120
ログファイルの生成（リンカオプション）	267
ログファイル、リンカからの生成	267
ロケーションカウンタ、定義	287

## わ

ワークスペース	
作成	104
使用	103

ワークスペース（[ワークスペース]	
ウィンドウアイコン	112
ワークスペースを開く（[ファイル] メニュー）	196
ワークスペースを閉じる（[ファイル] メニュー）	197
ワーニング	
アセンブラ	250–251
コンパイラ	240
リンカ	269
ワーニング（アセンブラオプション）	250
ワーニングとして処理（コンパイラオプション）	240
ワーニングとして処理（リンカオプション）	269
ワーニングまたはワーニングの範囲を無効にする（アセンブラオプション）	251
ワーニングまたはワーニングの範囲を有効にする（アセンブラオプション）	251
ワーニング範囲指定（ワーニング設定）	251

## A

a（ファイル名拡張子）	192
アセンブラオプション、定義	277
Advanced SIMD（NEON）（一般オプション）	219
AEABI、定義	277
ANSI C. C89 を参照	
Arm（プロセッサモード設定）	232
ar、定義	277
asm（ファイル名拡張子）	192

## B

bat（ファイル名拡張子）	192
bin、arm（サブディレクトリ）	188
bin、common（サブディレクトリ）	189
board（ファイル名拡張子）	192

## C

c（ファイル名拡張子）	192
C（言語設定）	229

C ソースファイル ([ワークスペース] ウィンドウアイコン) .....	112
C のキーワード、エディタのテキストスタイル .....	146
C のコメント、エディタのテキストスタイル .....	146
C の派生言語 (コンパイラオプション) .....	230
C 形式プリプロセッサ、定義 .....	279
cfg (ファイル名拡張子) .....	192
cgx (ファイル名拡張子) .....	192
CHAR の型 (コンパイラオプション) .....	231
chm (ファイル名拡張子) .....	192
CMSIS バックログ .....	203
CMSIS Manager ダイアログボックス .....	96
CMSIS-Pack .....	96
サンプルプロジェクト .....	30
ソフトウェアパックのインストール .....	107
新規プロジェクトの作成 .....	108
common (ディレクトリ) .....	189
\$CONFIG_NAME\$ (引数変数) .....	91
config, arm (サブディレクトリ) .....	188
config, common (サブディレクトリ) .....	189
cpp (ファイル名拡張子) .....	192
CRC 多項式 (チェックサムアルゴリズム) .....	271
CRC、定義 .....	279
CRC16 (チェックサムアルゴリズム) .....	271
CRC32 (チェックサムアルゴリズム) .....	271
CRC64ECMA (チェックサムアルゴリズム) .....	271
CRC64ISO (チェックサムアルゴリズム) .....	271
cstartup (システム起動コード) プログラムが以下に達するまでスタック ポインタが無効 .....	80
定義 .....	279
cstat, arm (サブディレクトリ) .....	188
\$CUR_DIR\$ (引数変数) .....	91
\$CUR_LINES\$ (引数変数) .....	91
C-RUN ランタイムエラー解析、対象ドキュメント ..	17
C-SPY ウィンドウの色、切替え .....	78
C-SPY オプション 定義 .....	279
C-STAT 静的分析、ドキュメント .....	18

C/C++ 構文 コンパイラで有効化 .....	229
スタイルのオプション .....	67
C++ オプション (コンパイラオプション) .....	231
C++ 例外を許可 (リンカオプション) .....	264
C++ inline semantics (C dialect setting) .....	230
C++ (言語設定) .....	229
C++ ソースファイル ([ワークスペース] ウィンドウアイコン) .....	112
C++ のキーワード、エディタのテキストスタイル ..	146
C++ のコメント、エディタのテキストスタイル .....	146
C++ 仮想関数除去を実行 (リンカオプション) .....	264
C++ 用語 .....	19
C89 (C 派生言語の設定) .....	230

## D

D レジスタ (一般オプション) .....	219
dat (ファイル名拡張子) .....	192, 194
\$DATE\$ (引数変数) .....	91
dbgdt (ファイル名拡張子) .....	192
ddf (ファイル名拡張子) .....	192
#define オプション (リンカオプション) .....	267
define (リンカオプション) .....	267
dep (ファイル名拡張子) .....	192
DLIB 命名規約 .....	20
dnx (ファイル名拡張子) .....	192
doc, arm (サブディレクトリ) .....	188
doc, common (サブディレクトリ) .....	189
drivers, arm (サブディレクトリ) .....	188
DSP デジタル信号プロセッサを参照。	
DSP 拡張 (一般オプション) .....	219
DWARF、定義 .....	280
Dynamic Data Exchange (DDE) .....	42
外部エディタの呼び出し .....	64



# E

EEPROM、定義	281
ELF、変換	254
Embedded C++	
定義	281
Embedded Workbench	
エディタ	139
バージョン番号、表示	216
メインウィンドウ	45
リファレンス情報	195
レイアウト	26
実行	26
Enea OSE Load モジュールフォーマット、定義	281
EOL 文字 (エディタのオプション)	61
EPROM、定義	281
ewd (ファイル名拡張子)	192
ewp (ファイル名拡張子)	192
ewplugin (ファイル名拡張子)	192
eww (ファイル名拡張子)	192
ワークスペースファイル	27
\$EW_DIR\$ (引数変数)	91
\$EXAMPLES_DIR\$ (カスタム引数変数)	27
examples、arm (サブディレクトリ)	188
\$EXE_DIR\$ (引数変数)	91

# F

\$FILE_DIR\$ (引数変数)	91
\$FILE_FNAME\$ (引数変数)	91
\$FILE_PATH\$ (引数変数)	91
flash (ファイル名拡張子)	193
flashdict (ファイル名の拡張子)	193
fnt (ファイル名拡張子)	193
FPU (一般オプション)	218

# H

h (ファイル名拡張子)	193
helpfiles (ファイル名拡張子)	193
htm (ファイル名拡張子)	193
html (ファイル名拡張子)	193
HTML テキストファイル ([ワークスペース] ウィンドウアイコン)	112

# I

i (ファイル名拡張子)	193
iarbuild、コマンドラインからのビルド	131
IarIdePm.exe	26
icf (ファイル名拡張子)	193
IDE	
概要	23
定義	283
IDE がフォアグラウンドプロセスでない場合、 ソースブラウザとビルドステータスが更新され ません (IDE プロジェクトオプション)	71
IDE 内部ファイル ([ワークスペース] ウィンドウアイコン)	112
ieldump、定義	283
ielftool、定義	283
ILINK	
options	259
ILINK 設定ファイルでの初期化、定義	283
ILINK、定義	283
inc (ファイル名拡張子)	193
inc、arm (サブディレクトリ)	188
ini (ファイル名拡張子)	193
iobjmanip、定義	284
I/O レジスタ。SFR を参照	

# L

L 値、定義	284
lib、arm (サブディレクトリ)	188

lightbulb アイコン、本ガイドの .....	20
#line ディレクティブ、生成 コンパイラ .....	238
\$LIST_DIR\$ (引数変数) .....	91
log (ファイル名拡張子) .....	193–194
lst (ファイル名拡張子) .....	193

## M

mac (ファイル名拡張子) .....	193
MAC、定義 .....	284
menu (ファイル名拡張子) .....	193
metadata (サブディレクトリ) .....	189
MISRA-C コンパイラオプション .....	240
ドキュメント .....	18
一般オプション .....	226

## N

NDEBUG、プリプロセッサシンボル .....	101
NOP (アセンブラ命令)、定義 .....	286

## O

o (ファイル名の拡張子) .....	193
objcopy、定義 .....	286
\$OBJ_DIR\$ (引数変数) .....	91
options アセンブラ .....	245
エディタ .....	59
エディタのセットアップファイル .....	66
カスタムビルド .....	255
コンパイラ .....	227
コンバータ .....	253
ビルドアクション .....	257
ライブラリビルダ .....	275
リンカ .....	259
out (ファイル名拡張子) .....	193

## P

pbd (ファイル名拡張子) .....	194
pbi (ファイル名拡張子) .....	194
pew (ファイル名拡張子) .....	194
printf フォーマッタ (一般オプション) .....	224
prj (ファイル名拡張子) .....	194
\$PROJ_DIR\$ (引数変数) .....	91
\$PROJ_FNAMES\$ (引数変数) .....	91
\$PROJ_PATHS\$ (引数変数) .....	92
PROM、定義 .....	287

## R

R 値、定義 .....	289
readme ファイル、リリースノートを参照	
ROM モニタ、定義 .....	288
ropi、位置に依存しない .....	233
rtos arm (サブディレクトリ) .....	189
RTOS、定義 .....	287
RTTI コンパイラで有効化 .....	231
RTTI を有効にする (C++ オプション設定) .....	231
rwpi、位置に依存しない .....	233

## S

s (ファイル名拡張子) .....	194
scanf のフォーマッタ (一般オプション) .....	224
section バイナリデータ .....	263
定義 .....	289
SFR ヘッダファイル .....	188
定義 .....	290
sfr (ファイル名拡張子) .....	194
sim (ファイル名拡張子) .....	194
src、arm (サブディレクトリ) .....	189

STL コンテナ展開 (IDE オプション) .....	78
Subversion の状態と対応するアイコン .....	121
suc (ファイル名拡張子) .....	194
svd (ファイル名拡張子) .....	194

## T

\$TARGET_BNAME\$ (引数変数) .....	92
\$TARGET_BPATH\$ (引数変数) .....	92
\$TARGET_DIRS (引数変数) .....	92
\$TARGET_FNAME\$ (引数変数) .....	92
\$TARGET_PATH\$ (引数変数) .....	92
Thumb (プロセッサモード設定) .....	232
\$TOOLKIT_DIR\$ (引数変数) .....	92
TrustZone (一般オプション) .....	219
TrustZone インポートライブラリ (リンカオプション) .....	266
tutorials, arm (サブディレクトリ) .....	189

## U

UBROF	
生成ツール .....	292
定義 .....	291
\$USER_NAME\$ (引数変数) .....	92

## V

VFPv2 (FPU 設定) .....	218
VFPv3 (FPU 設定) .....	219
VFPv4 (FPU 設定) .....	219
VFPv4 単精度 (FPU 設定) .....	219
VFPv5 単精度 (FPU 設定) .....	219
VFPv5 倍精度 (FPU 設定) .....	219
VFP9-S (FPU 設定) .....	219
visualSTATE	
ツールチェーンの一部 .....	25
プロジェクトファイル .....	194

VLA の許可 (C 派生言語の設定) .....	230
vsp (ファイル名拡張子) .....	194

## W

Web サイト、推奨 .....	18
wsdt (ファイル名拡張子) .....	194

## X

XAR、定義 .....	292
xcl (ファイル名拡張子) .....	194
XLIB、定義 .....	292
XLINK、定義 .....	292

## 記号

[インクリメンタル検索] ダイアログボックス ([編集] メニュー) .....	170
[インクルードディレクトリの編集] ダイアロ グボックス (プリプロセッサオプション) .....	243
[ウィンドウ] メニュー .....	215
[オプション] ダイアログボックス ([プロジェクト] メニュー) .....	134
使用 .....	125
[カスタマイズ] ダイアログボックス .....	50
[カスタムの引数変数の設定] ダイアログボックス ..	93
[グループの名称変更] ダイアログボックス .....	116
[コード] ページ (コンパイラオプション) .....	232
[すべての参照を検索] ウィンドウ ([表示] メニュー) .....	179
[ソースブラウザ] ウィンドウ .....	174
使用 .....	150
[ツール出力] ウィンドウ .....	54
[ツール] メニュー .....	212
[テンプレート] ダイアログボックス ([編集] メニュー) .....	180
[バッチビルドの編集] ダイアログボックス ([プロジェクト] メニュー) .....	138

[バッチビルド] ダイアログボックス ([プロジェクト] メニュー).....	137
[ビューアの設定] ダイアログボックス ([ツール] メニュー).....	85
[ファイルから検索] ウィンドウ ([表示] メニュー).....	163
[ファイルから検索] ダイアログボックス ([編集] メニュー).....	165
[ファイル内で置換] ダイアログボックス ([編集] メニュー).....	167
[ファイル名拡張子のオーバーライド] ダイアロ グボックス ([ツール] メニュー).....	89
[ファイル名拡張子の編集] ダイアログボックス ([ツール] メニュー).....	90
[ファイル名拡張子] ダイアログボックス ([ツール] メニュー).....	88
[ファイル] メニュー.....	195
[プロジェクトコネクションの追加] ダイアロ グボックス ([プロジェクト] メニュー).....	119
[プロジェクトの構成] ダイアログボックス ([プロジェクト] メニュー).....	117
[プロジェクト] ページ ([IDE オプション] ダイアログボックス).....	69
[プロジェクト] メニュー.....	206
[ヘルプ] メニュー.....	216
[メッセージ] ウィンドウ、出力内容.....	136, 178
[メモリ消去] ダイアログボックス.....	211
[ワークスペース] ウィンドウ.....	111
ファイルのドラッグアンドドロップ.....	102
[ワークスペース] ウィンドウアイコン.....	112
[逆アセンブリ] ウィンドウのソースコード色 (IDE オプション).....	77
[逆アセンブリ] ウィンドウ、定義.....	280
[検索] ダイアログボックス ([編集] メニュー)...	161
[言語] ([IDE オプション] ダイアログボックス)...	58
[行へ移動] ダイアログボックス.....	200
[最適化] ページ (コンパイラオプション).....	233
[参照] ウィンドウ ([表示] メニュー).....	173
[新規プロジェクトの作成] ダイアログボックス ([プロジェクト] メニュー).....	116

[新規構成] ダイアログボックス ([プロジェクト] メニュー).....	118
[宣言] ウィンドウ ([表示] メニュー).....	171
[置換] ダイアログボックス ([編集] メニュー)...	164
[追加ライブラリの編集] ダイアログボックス (リンカオプション).....	274
[表示] メニュー.....	202
[編集] メニュー.....	198
[IDE オプション] ダイアログボックス.....	59
#define (インクルードクロスリファレンスの 設定).....	248
#define オプション (リンカオプション).....	267
#included テキスト (インクルードリスト化の 設定).....	248
#line ディレクティブ生成 (ファイルへの プリプロセッサ出力の設定).....	238
#pragma ディレクティブ、定義.....	287
% スタック使用しきい値 ([スタック] オプション).....	79
\$CONFIG_NAMES (引数変数).....	91
\$CUR_DIR\$ (引数変数).....	91
\$CUR_LINE\$ (引数変数).....	91
\$DATES\$ (引数変数).....	91
\$EW_DIR\$ (引数変数).....	91
\$EXAMPLES_DIR\$ (カスタム引数変数).....	27
\$EXE_DIR\$ (引数変数).....	91
\$FILE_DIR\$ (引数変数).....	91
\$FILE_FNAME\$ (引数変数).....	91
\$FILE_PATH\$ (引数変数).....	91
\$LIST_DIR\$ (引数変数).....	91
\$OBJ_DIR\$ (引数変数).....	91
\$PROJ_DIR\$ (引数変数).....	91
\$PROJ_FNAME\$ (引数変数).....	91
\$PROJ_PATH\$ (引数変数).....	92
\$TARGET_BNAME\$ (引数変数).....	92
\$TARGET_BPATH\$ (引数変数).....	92
\$TARGET_DIR\$ (引数変数).....	92
\$TARGET_FNAME\$ (引数変数).....	92
\$TARGET_PATH\$ (引数変数).....	92

\$TOOLKIT\_DIR\$ (引数変数).....92

\$USER\_NAME\$ (引数変数).....92

数字

2 行間隔 (インクルードクロスリファレンスの  
設定) .....249