お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

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

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

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

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

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

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

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ（フラッシュメモリ・SRAM等）を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ（http://www.renesas.com）

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

User's Manual

SuperH™ RISC engine High-performance Embedded Workshop, Debugging Interface

チュートリアル

ルネサスマイクロコンピュータ開発環境システム

HS0700EWIW1SJ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2000.12

ご注意

1 本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合，または国外に持ち出す場合は日本国政府の許可が必要です。

2 本書に記載された情報の使用に際して，弊社もしくは第三者の特許権，著作権，商標権，その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合，弊社はその責を負いませんので予めご了承ください。

3 製品及び製品仕様は予告無く変更する場合がありますので，最終的な設計，ご購入，ご使用に際しましては，事前に最新の製品規格または仕様書をお求めになりご確認ください。

4 弊社は品質・信頼性の向上に努めておりますが，宇宙，航空，原子力，燃焼制御，運輸，交通，各種安全装置， ライフサポート関連の医療機器等のように，特別な品質・信頼性が要求され，その故障や誤動作が直接人命を脅かしたり，人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は，事前に弊社営業担当迄ご相談をお願い致します。

5 設計に際しては，特に最大定格，動作電源電圧範囲，放熱特性，実装条件及びその他諸条件につきましては，弊社保証範囲内でご使用いただきますようお願い致します。
保証値を越えてご使用された場合の故障及び事故につきましては，弊社はその責を負いません。
また保証値内のご使用であっても半導体製品について通常予測される故障発生率，故障モードをご考慮の上，弊社製品の動作が原因でご使用機器が人身事故，火災事故，その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。

6 本製品は耐放射線設計をしておりません。

7 本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。

8 本書をはじめ弊社半導体についてのお問い合わせ，ご相談は弊社営業担当迄お願い致します。

商標

Windows® および Windows®95、Windows®98、WindowsNT®、Windows®2000 は、米国マイクロソフトコーポレーションの米国およびその他の国における登録商標です。

IBM PC は、米国 IBM 社により管理されている計算機の名称です。

ELF/DWARF は、the Tool Interface Standards Committee で開発された、オブジェクトフォーマットの名称です。

本チュートリアルで使用されているすべての製品名またはブランド名は、それぞれの会社の商標または登録商標です。

まえがき

本チュートリアルは、Hitachi Embedded Workshop（以下、HEW と略します）および Hitachi Debugging Interface（以下、HDI と略します）の簡単な使用方法を述べたものです。HEW チュートリアルでは HEW の起動、プロジェクトの作成・編集方法から、アプリケーションの作成および HDI との連動にいたる一連のプログラム作成作業について説明します。また、HDI チュートリアルではシミュレータ・デバッガを用いて、サンプルプログラムのデバッグ方法について説明します。各ツールについては、下記のマニュアルを参照してください。

- Hitachi Embedded Workshop　ユーザーズマニュアル
- SuperH™ RISC engine C/C++コンパイラ、アセンブラ、最適化リンケージエディタ ユーザーズマニュアル
- SuperH™ RISC engine シミュレータ・デバッガ　ユーザーズマニュアル

SuperH™ RISC engine マイコンの詳細については、該当品種のプログラミングマニュアルおよびハードウェアマニュアルを参照してください。

なお、説明は SuperH™ RISC engine C/C++コンパイラパッケージに含まれる全てのツールをインストールした状態で進めます。

チュートリアルの規約

本チュートリアルに記載している画面は英語版 Windows®上で取得したものです。日本語版 Windows®では一部日本語表示します。また、本チュートリアルではディレクトリ区切り子としてバックスラッシュを使用していますが、日本語版 Windows®上ではバックスラッシュの代わりに円記号を使用してください。

表記上の規約

本書には、以下の表記上の規約があります。

表 1　表記上の規約

表　記	意　　味
[Menu->Menu Option]	**'->'の付いた太字の表記は、メニューオプションを指定する場合に使用します** **(例：** [File->Save As...])。

目次

1. HEW チュートリアル

1.1 HEW の起動

HEW のインストーラは、インストール正常終了時、Windows®のスタートメニューのプログラムフォルダの下に Hitachi Embedded Workshop という名称のフォルダを作成し、そのフォルダ内に HEW の実行プログラムである Hitachi Embedded Workshop の他、各種サポート情報のショートカットを登録します（**図** 1.1）。

なお、スタートメニューの表示内容は、ツールのインストール状況により異なる場合があります。

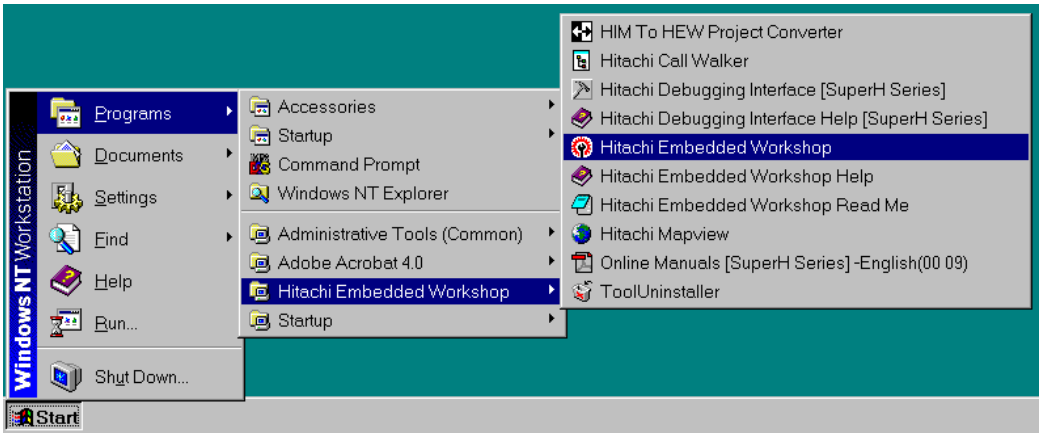

図 1.1 **スタートメニューによる** HEW **の起動**

このスタートメニューで、Hitachi Embedded Workshop をクリックすると起動メッセージを表示し、引き続き Welcome!ダイアログボックス（**図** 1.2）を表示します。なお、前の作業で開いていたプロジェクトを直接開きたい場合などは、[Tools->Options]で作業環境の設定変更が可能です。詳しくは、**「**Hitachi Embedded Workshop **ユーザーズマニュアル」**の**「第６章　環境のカスタマイズ」**を参照してください。

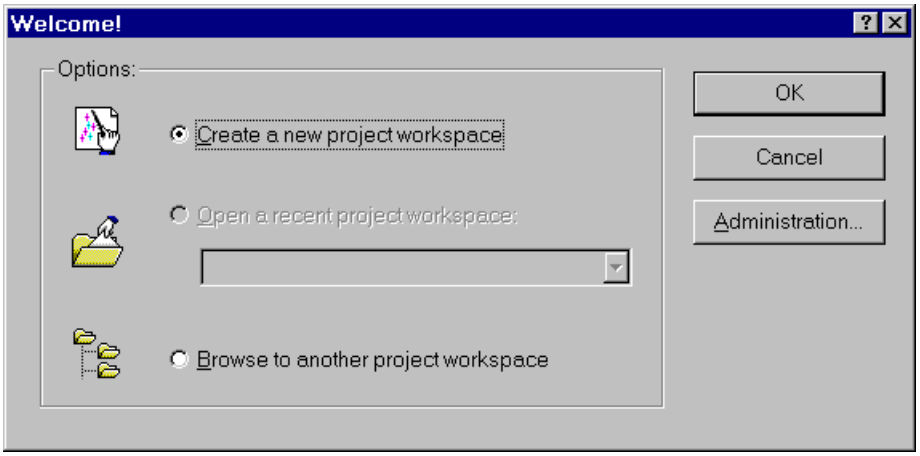

図 1.2　Welcome!**ダイアログボックス**

HEW を初めて使用する場合や、新たにプロジェクトを作成して作業を開始する場合は、[Create a new project workspace]を選択して[OK]をクリックしてください。既に作成したプロジェクトで作業する場合は、[Open a recent project workspace]または[Browse to another project workspace]を選択して[OK]をクリックしてください。

ここでは、[Create a new project workspace]を選択して[OK]をクリックします。

1.2 プロジェクトの作成

Welcome!ダイアログボックスで[Create a new project workspace]を選択して[OK]をクリックすると、新しいワークスペースとプロジェクト作成用の New Project Workspace ダイアログボックス（**図** 1.3）を表示します（HEW の起動後は、[File -> New Workspace…]でも表示できます）。このダイアログボックスでワークスペース名（新規作成時はプロジェクト名も同名です）や CPU の種類、プロジェクトタイプなどを設定します。

HEW が標準サポートするプロジェクトジェネレータでは**表** 1.1のプロジェクトタイプを扱っています。

ここでは、ワークスペース名として“ tutorial ”を入力し、Project type として Application を選択します。[Name]に“ tutorial ”と入力すると、[Directory]も“ c:¥tutorial ”となり、このディレクトリ以下にプロジェクトが生成されます。

なお、ワークスペースとして使用するディレクトリを変更する場合は、[Browse...]をクリックしてディレクトリを選択するか、直接[Directory]に入力してください。

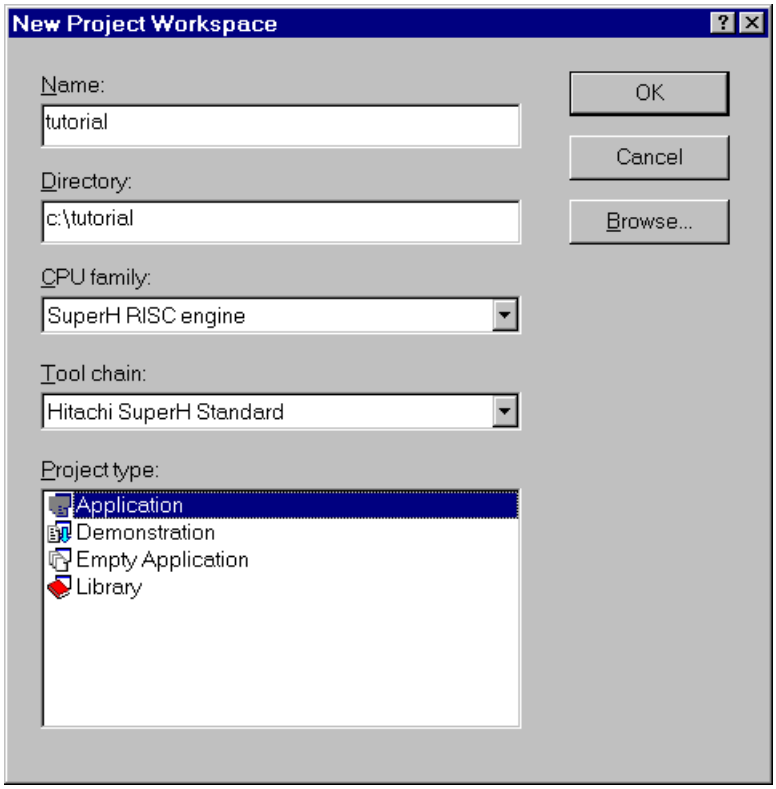

図 1.3 New Project Workspace **ダイアログボックス**

表 1.1 プロジェクトタイプ

プロジェクトタイプ	説明
Application	C/C++言語またはアセンブリ言語で記述した初期ルーチンファイルを含むプログラム開発用プロジェクト
Demonstration	C/C++言語またはアセンブリ言語で記述したデモンストレーションプログラム作成用プロジェクト（「1.5.2 Demonstration プロジェクト」）
Empty Application	プログラム開発用プロジェクト（生成ファイルはありません。ツールのオプション設定のみ行います。）
Library	ライブラリファイル作成用プロジェクト（生成ファイルはありません。ツールのオプション設定のみ行います。）

1.2.1 CPU の選択

New Project Workspace ダイアログボックスで[OK]をクリックすると、プロジェクトジェネレータを起動し、Step1 画面（CPU の設定）を表示します（図 1.4）。この画面では、開発の対象となる CPU を選択します。

ここでは、[CPU Series]で“ SH-1 ”を選択し、[CPU Type]から“ SH7020 ”を選択し、[Next >]をクリックしてください。クリックすると Step2 画面（CPU 固有のオプション設定）を表示します。

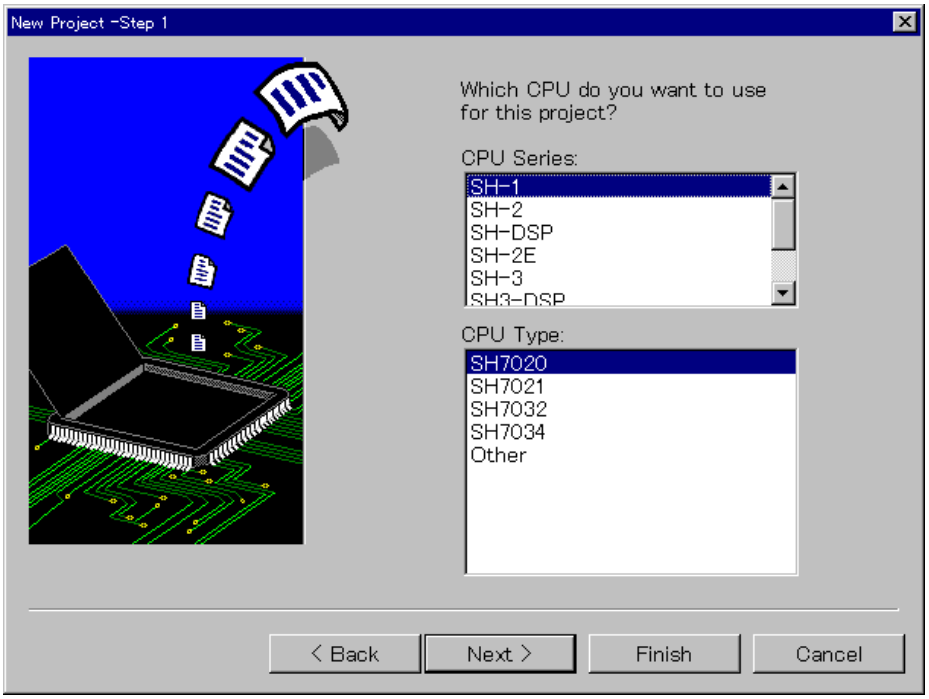

図 1.4 プロジェクトジェネレータの CPU 設定画面（Step1）

[CPU Type:]は、[CPU Series:]で選択した CPU シリーズに含まれる製品（以下、CPU タイプと呼びます）です。

[CPU Series:]の選択により、標準ライブラリ構築ツール、C/C++コンパイラおよびアセンブラの CPU 種別オプションを設定します。また、[CPU Series:]および[CPU Type:]の選択により、生成するファイルが異なります。開発するプログラムの対象となる CPU タイプを選択してください。選択したい CPU タイプがない場合は、ハードウェア仕様の近い CPU タイプまたは“ Other ”を選択してください。

[Next >]をクリックすると、次の画面を表示します。

[< Back]をクリックすると、この画面を表示する前の画面またはダイアログボックスに戻ります。

[Finish]をクリックすると、Summary ダイアログボックスが開きます。

[Cancel]をクリックすると、New Project Workspace ダイアログボックスに戻ります。

[< Back]、[Next >]、[Finish]および[Cancel]の機能は、プロジェクトジェネレータで共通の機能です。

1.2.2 CPU 固有のオプションの設定

Step1 画面で[Next >]をクリックすると、**図 1.5**に示す画面を表示します。

この画面で、CPU 固有で、かつ全プロジェクトファイルで共通のオプションを設定します。

ここでは、設定を変更しないで[Next >]をクリックします。クリックすると、Step3 画面（生成ファイルの設定）を表示します。

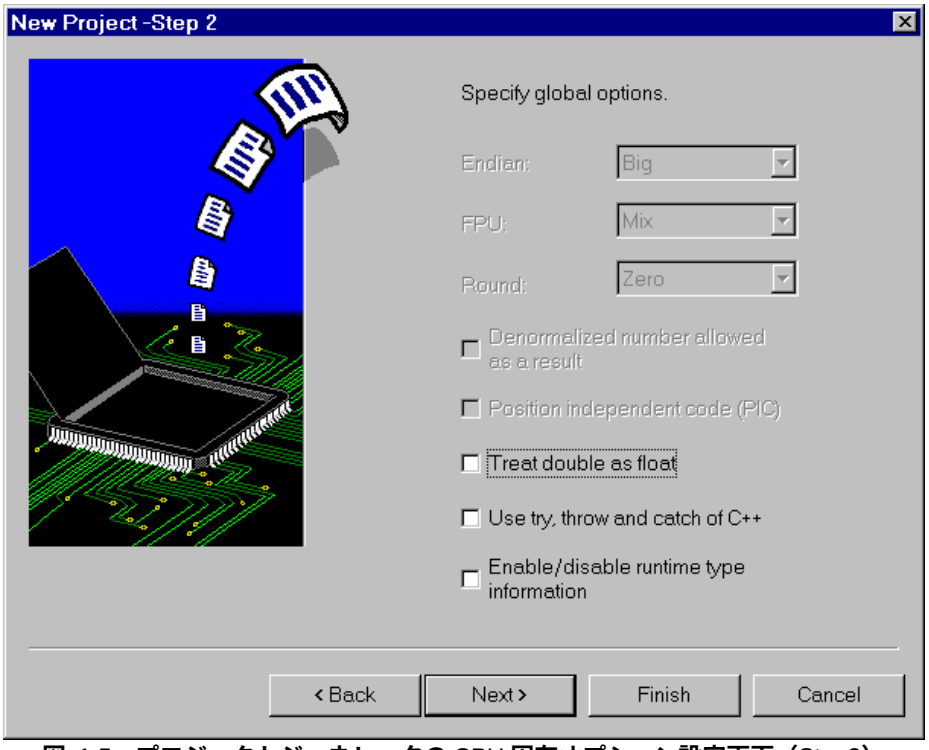

図 1.5 プロジェクトジェネレータの CPU 固有オプション設定画面（Step2）

Step1 および Step2 の画面設定により、以下のオプションを設定します。
プロジェクト作成後に設定を変更する場合は、下記ツールのオプション設定ダイアログ（HEW の [Options ->]）で変更してください。

＜C/C++ コンパイラ＞（＜標準ライブラリ構築ツール＞も同様）
CPU
除算の方式
メモリのバイト並び（エンディアン種別）
FPU
丸め方向
非正規化数の扱い
プログラムセクションポジションインディペンデント
double -> float 変換
例外処理機能
実行時型情報

＜アセンブラ＞
CPU
エンディアン種別
丸め方向
非正規化数の扱い

＜最適化リンケージエディタ＞
出力形式（プロジェクトタイプにより決定）
ライブラリファイルの指定（DSP ライブラリのみ）

Step1 画面で選択した CPU シリーズによりサポートしていないオプションのコントロールは、設定変更ができないようになっています。本チュートリアルでは Step1 画面で “ SH-1 ” を選択したため、[Treat double as float]、[Use try, throw and catch of C++]および[Enable/disable runtime type information]以外の表示がグレーになっており、これらのオプションを設定することはできません。
プロジェクトタイプが Library の場合は、他に設定する項目はありません。[Finish]を選択してください。
プロジェクトタイプが Demonstration の場合は、[Next >]をクリックすると、**「1.2.7 生成ファイル名の変更」**に示す Step6 画面を表示します（Demonstration の場合は、ダイアログボックスのタイトルが Step3 になります）。

1.2.3 生成ファイルの設定

Step2 画面で[Next >]をクリックすると、**図** 1.6に示す画面を表示します。

図 1.6 **プロジェクトジェネレータの生成ファイル設定画面**（Step3）

この画面で、生成するファイルを決定します。

ここでは、設定を変更しないで[Next >]をクリックします。クリックすると、Step4（標準ライブラリの設定）画面を表示します。

[Use I/O Library]のチェックを付けると、標準入出力ライブラリを活用できます。このとき、[Number of I/O Streams:]は 20 で固定です。

[Use Heap Memory]のチェックを付けると、ヒープ領域の管理用の関数 sbrk()を活用できます。このとき、[Heap Size:]で、管理するヒープ領域のサイズを設定できます。
[Heap Size:]は、16 進数または 10 進数で入力することができます。以下のように、入力してください。

H'ABCDE：16 進数
0xABCDE：16 進数
1234　　：10 進数

プロジェクト作成後に、[Heap Size]を変更する場合は、sbrk.h の以下の箇所の数値を変更してください。

```
#define    HEAPSIZE    0x400
```

[Generate main() Function]では、main 関数の生成方法（C、C++、生成なし）を設定します。
[I/O Register Definition Files]のチェックを付けると、C 言語で記述した I/O レジスタの定義ファイルを生成します。このとき、[Generate Hardware Setup Function]の設定が可能になります。

この画面の設定およびプロジェクトタイプにより生成するファイルが異なります。各コントロールの設定により生成するファイルを**表** 1.2に示します。

表 1.2　**生成ファイル画面の設定内容と生成ファイル**

コントロール	状態	生成ファイル
[Use I/O Library]	**チェック**	lowlvl.src lowsrc.c
[Number of I/O Streams:]	**入力不可能**	**－**
[Use Heap Memory]	**チェック**	sbrk.c
[Heap Size:]	**入力可能**	sbrk.h
[Generate main() Function]	C source file C++ source file None	tutorial.c tutorial.cpp **－** **（生成時、プロジェクト名がファイル名になります）**
[I/O Register Definition Files]	**チェック**	iodefine.h
[Generate Hardware Setup Function]	None Assembly source file C/C++ source file	**－** hwsetup.src hwsetup.c (**または、**cpp) **（**main **関数ファイルと同じ拡張子）**

注意　既に作成した関数 main**（**_main**）を使用する場合は、**[Generate main() Function]**のチェックをはずしてプロジェクトを作成し、該当するファイルをプロジェクトに追加してください。なお、使用する関数名が異なる場合、**resetprg.src **の関数呼び出し部分を修正する必要があります。**

1.2.4 標準ライブラリの設定

Step3 画面で[Next >]をクリックすると、**図** 1.7に示す画面を表示します。

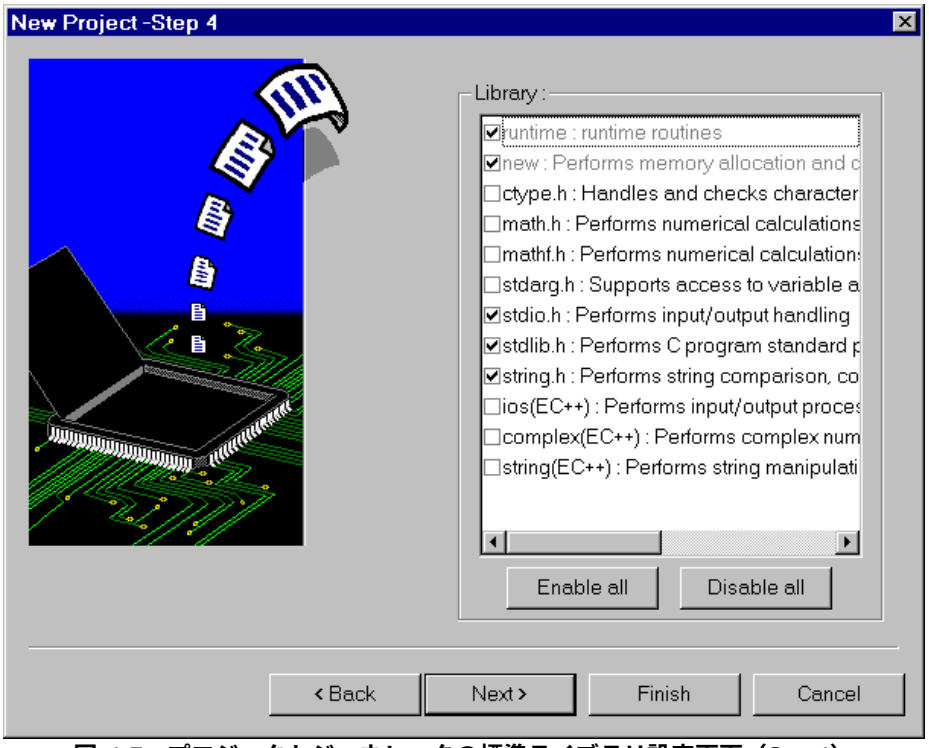

図 1.7 **プロジェクトジェネレータの標準ライブラリ設定画面**（Step4）

この画面で、プロジェクトで使用する標準ライブラリを設定します。ビルド実行時、[Library]でチェックした項目をもとに標準ライブラリが構築されます。

プロジェクト作成後に設定を変更する場合は、標準ライブラリ構築ツールのオプション設定ダイアログ（HEW の[Options -> SH Library Generator…]）で変更してください。

ここでは、設定を変更しないで[Next >]をクリックします。クリックすると、Step5（スタック領域の設定）画面を表示します。

1.2.5 スタック領域の設定

Step4 画面で[Next >]をクリックすると、**図** 1.8に示す画面を表示します。

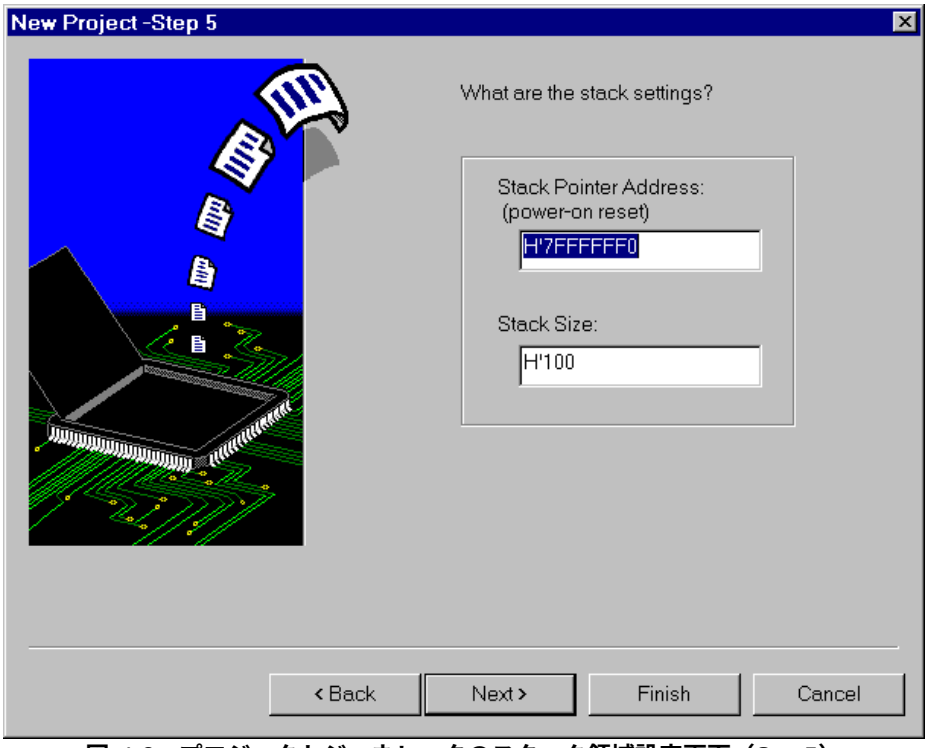

図 1.8 **プロジェクトジェネレータのスタック領域設定画面**（Step5）

この画面で、スタック領域を設定します。
ここでは、設定を変更しないで[Next >]をクリックします。クリックすると、Step6（ベクタ定義の設定）画面を表示します。

[Stack Pointer Address:]および[Stack Size:]には、16 進数または 10 進数で入力することができます。以下のように、入力してください。

H’ABCDE：16 進数
0xABCDE：16 進数
1234　　：10 進数

[Stack Size:]の設定は、stacksct.src に反映されます。また、[Stack Pointer Address:]および[Stack Size:]の設定により、最適化リンケージエディタのスタートオプションで、スタックのセクションのアドレスを決定します。スタックセクションの開始アドレスは、以下のように決定します。

スタックセクション名：Stack
スタックセクションアドレス ＝ [Stack Pointer Address:]の値 － [Stack Size:]の値

プロジェクト作成後にスタックサイズを変更する場合は、stacksct.src のスタック領域サイズを変更し、最適化リンケージエディタのオプションダイアログ（HEW の[Options -> Optlinker…]）のスタートオプションで、スタックセクションのアドレスを変更してください。

stacksct.src

```
        .section    Stack,STACK
        .export     _PowerON_Reset_SP
        .export     _Manual_Reset_SP
StackEND:           .res.b  H'100          -----> スタック領域サイズ
_PowerON_Reset_SP:  .equ $
_Manual_Reset_SP:   .equ $
        .end
```

1.2.6 ベクタの設定

Step5 画面で[Next >]をクリックすると、**図** 1.9に示す画面を表示します。

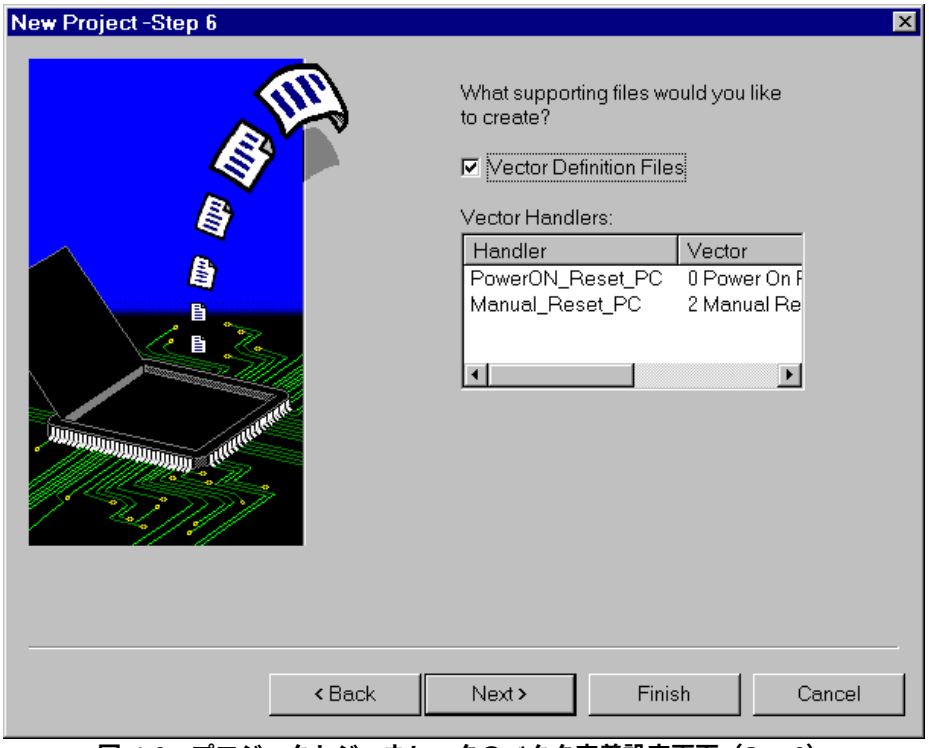

図 1.9 **プロジェクトジェネレータのベクタ定義設定画面**（Step6）

この画面では、ベクタを設定します。

ここでは、設定を変更しないで[Next >]をクリックします。クリックすると、Step7（生成ファイル名の確認）画面を表示します。

[Vector Definition Files]のチェックを付けると、ベクタテーブル定義ファイルを生成します。このとき、[Vector Handlers:]でリセットベクタを変更することができます。

[Vector Handlers:]の[Handler]列はリセットベクタのハンドラプログラム名を、[Vector]列にはベクタの説明を表示しています。自作のハンドラプログラムを使用する場合は、変更するハンドラプログラム名を選択してクリック後、自作のハンドラプログラム名を入力してください。

なお、[Vector Handlers:]のハンドラ名を変更すると、リセットプログラム用ファイル（resetprg.src）は生成しません。

[Vector Definition Files]のチェックを付けると、以下のファイルを生成します。

vecttbl.c（SH3,4系は、vecttbl.src）
intprg.c（SH3,4系は、intprg.src）
vect.h（SH3,4系は、vect.inc）
resetprg.src
vhander.src（SH3,4系のみ）

プロジェクト作成後に、割込みプログラムを編集または他のプログラムに変更する場合は、以下の手順でソースファイルを変更してください。

（1） 割込みプログラムを編集

- リセットプログラムは resetprg.src にアセンブリ言語で記述しています。セクションの初期化やスタックレジスタの設定（SH3,4 系のみ）等の簡単な初期設定のみですので、必要に応じて処理を追加してください。
- 割込みプログラムは、SH1,2 系は intprg.c に C 言語で、また、SH3,4 系は intprg.src にアセンブリ言語で記述しています。どちらも、プログラム名のみで処理内容は記述していませんので、必要な処理を追加してください。

（2） 割込みプログラムを他のプログラムと変更

- SH1,2 系の割込みプログラムを変更するには、定義ファイル（vect.h）内で変更対象である割込みベクタ用の関数を変更します。さらに、割込み関数ファイル（intprg.c）から同関数を削除し、新たに割込みプログラムとして使う関数を追加します。

例：ベクタ 4 のプログラム（INT_Illegal_code）を void UserIntProg4(void)に変更

vect.h

```
      .
      .
// 4 Illegal code
#pragma interrupt INT_Illegal_code          --->    #pragma interrupt UserIntProg4    に変更。
extern void INT_Illegal_code(void);          --->    extern void UserIntProg4 (void);    に変更。
      .
      .
```

intprg.c

```
      .
      .
// 4 Illegal code
void INT_Illegal_code(void){/* sleep(); */}    --->    削除し、void UserIntProg4(void)を記述。
      .                                                 void UserIntProg4(void)は、別のファイル
      .                                                 への記述も可。
```

- SH3,4 系の割込みプログラムを変更するには、定義ファイル（vect.inc）内で変更対象である割込み要因に対応するラベル（アセンブリモジュール名）を変更します。さらに、割込み関数ファイル（intprg.src）から同関数名を削除し、新たに割込みプログラムとして使う関数を追加します。

例：割込み要因コード H’040 のプログラム（_INT_TLBMiss_Load）を _UserIntProg040 に変更

vect.inc

```
        .
        .
    ;H'040 TLB miss/invalid (load)
    .global     _INT_TLBMiss_Load       --->    .global     _UserIntProg040   に変更。
        .
        .
```

intprg.src

```
        .
        .
;H'040 TLB miss/invalid (load)
_INT_TLBMiss_Load                       ---> 削除し、_UserIntProg040 を記述したファイル
        .                                    をプロジェクトに追加。
        .
```

vecttbl.src

```
        .
        .
    ;H'040 TLB miss/invalid (load)
    .data.l   _INT_TLBMiss_Load         --->    .data.l   _UserIntProg040   に変更。
        .
        .
```

1.2.7 生成ファイル名の変更

Step6 画面で[Next >]をクリックすると、**図** 1.10に示す画面を表示します。

図 1.10 **プロジェクトジェネレータの生成ファイル名確認画面**（Step7）

この画面が New Project ウィザードダイアログボックスの最終画面になります。これまでの画面の設定により生成されるファイルをリストに表示しています。リストの[File Name]列はファイル名を、[Extension]列は拡張子を、[Description]列はファイルの説明を表示しています。ファイル名は、変更することができます。変更する場合は、ファイル名を選択してクリック後、入力してください。

ここでは、設定を変更しないで[Finish]をクリックします。クリックすると、Summary ダイアログボックスを表示します。

注意 [Extension]**が“** h **”または“** inc **”のファイルは、インクルードファイルです。これらのファイル名を変更した場合、インクルードしているファイルの** include **文も変更する必要があります。**

1.2.8 設定確認

Step7 画面で[Finish]をクリックすると、**図** 1.11に示すダイアログボックスを表示します。

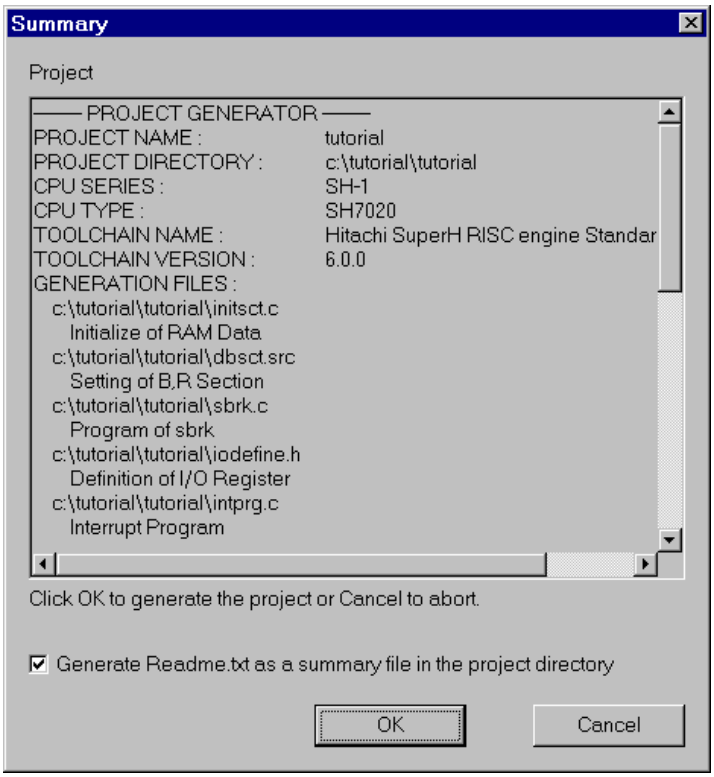

図 1.11 **プロジェクトジェネレータの** Summary **ダイアログボックス**

[Project Summary:]に、これまでの設定内容を表示しています。内容を確認して[OK]をクリックしてください。設定を変更したい場合は、[Cancel]をクリックしてください。

表示している内容は、テキストファイル（Readme.txt）として出力します。出力しない場合は、[Generate Readme.txt as a summary file in the project directory]のチェックをはずしてください。

[OK]をクリックすると、プロジェクトを作成します。

HEW はプロジェクトジェネレータが生成したファイルを組み込んだプロジェクトを開きます。

HEW の初期ウィンドウ状態は、**図** 1.12に示すように、プロジェクトの内容を示すワークスペースウィンドウ（図中の左側）と、ビルドやファイル間の文字列検索などの結果を表示するアウトプットウィンドウ（図中の下側）と、テキストファイル編集用のエディタウィンドウ（図中の右側）に分割されています。HEW のウィンドウの状態を変更する場合や、エディタを含めた各種ウィンドウの機能については、**「**Hitachi Embedded Workshop **ユーザーズマニュアル」**を参照してください。

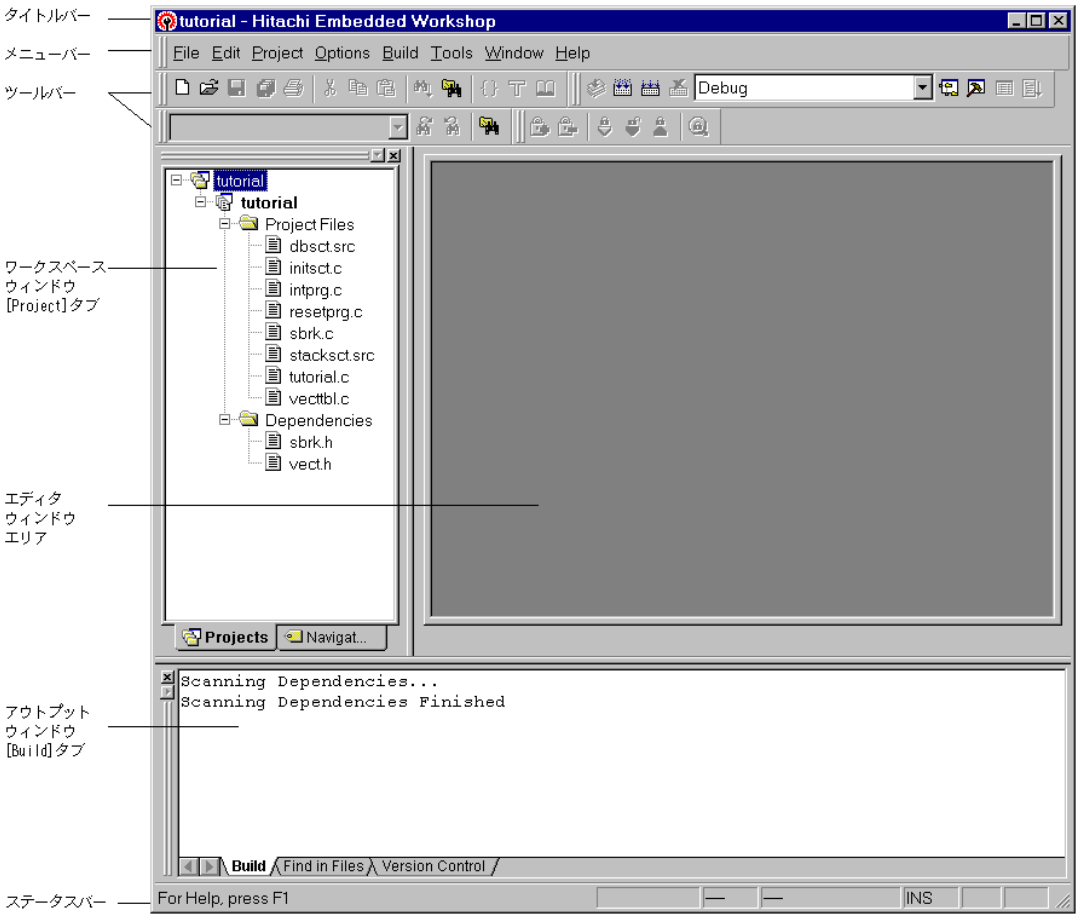

図 1.12　HEW **のウィンドウ構成**

また、プロジェクトジェネレータにより作成したプロジェクトは、ライブラリ構築ツール、C/C++コンパイラ、アセンブラ、最適化リンケージエディタなどのビルドに使うツールに対して基本的なオプションを設定しています。そのため、プロジェクト作成直後に[Build->Build]によりビルドすることも可能です（**図** 1.13）。

なお、ビルドはツールバー上の

ボタンをクリックしても実行できます。

また、これらの基本的なオプションは、各ツールのデフォルト設定として定義していますので、以後プロジェクトにファイルを追加してもそのファイルに個別に設定するオプションを除いて、新たにオプションを設定する必要はありません。

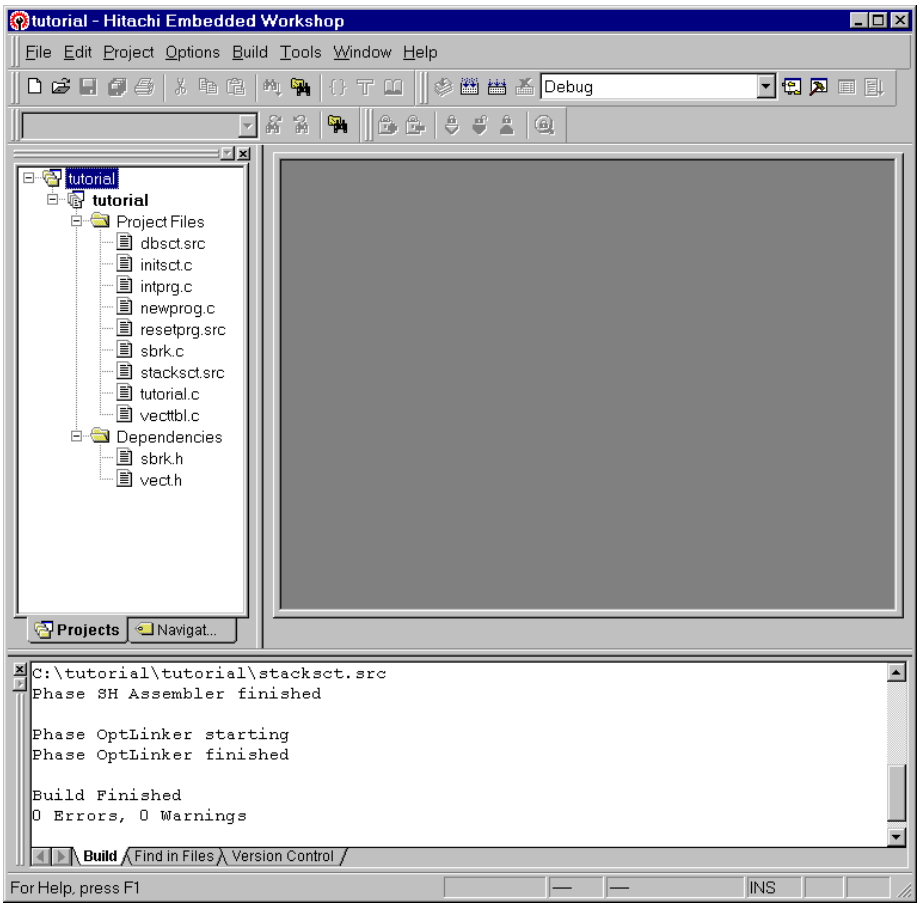

図 1.13　**ビルド実行時のウィンドウ**

1.3 プロジェクトの編集

プロジェクトジェネレータを使用してプロジェクトを作成するとリセットルーチンやRAMの初期化ルーチンなどの基本的なプログラムが生成されプロジェクトに組み込まれます。ここでは、生成されたファイルの修正や、新たなファイルのプロジェクトへの登録方法などについて説明します。

1.3.1 ソースファイルの編集と作成

プロジェクトに組み込まれたテキストファイルをワークスペースウィンドウの[Projects]タブで選択してダブルクリックすると、エディタウィンドウで編集することができます。ここでは、“tutorial.c”を開きます（**図** 1.14）。

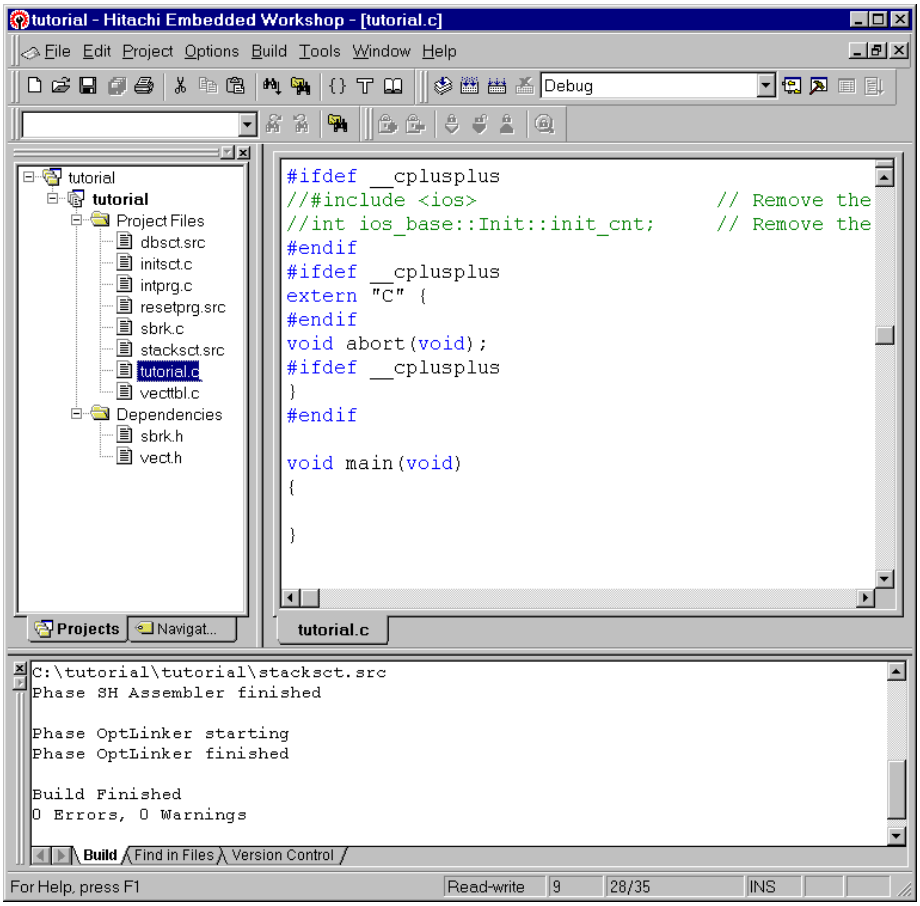

図 1.14 **プロジェクトファイルのオープン**

エディタウィンドウに開いたファイルは編集可能となり、編集するとウィンドウタイトルに表示するファイル名の右にアスタリスク（‘＊’）が付きます(**図** 1.15)。編集を終了し、ファイルを保存するとアスタリスクが消えます。

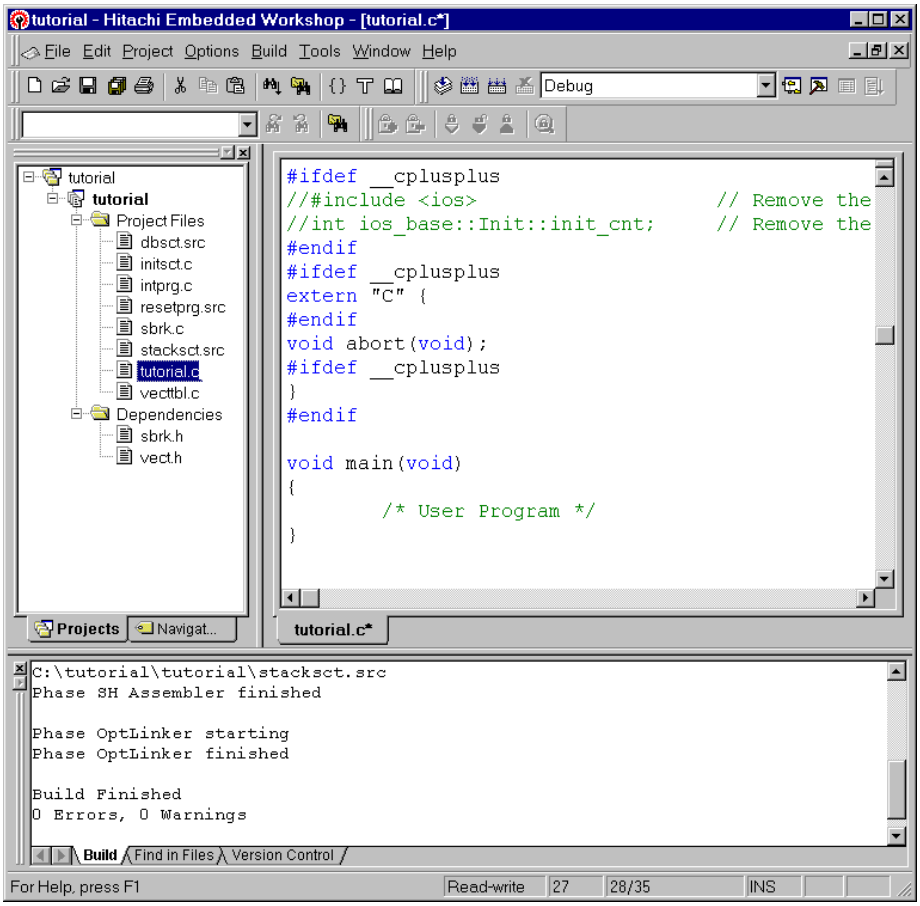

図 1.15　**プロジェクトファイルの編集**

また、既存のファイルを開く場合は、[File->Open]で、新規にファイルを作成する場合は、[File->New]でエディタウィンドウにテキストファイルを開くことができます。

ここでは、[File->New]で新規ファイルを開き、次の 3 行を入力し、[File->Save As...]により“ newprog.c ”の名称でファイルを保存します。

```
void NewProgram(void)
{
}
```

1.3.2 プロジェクトへのファイル追加と削除

次に既存のファイルをプロジェクトに追加する方法について説明します。ここでは、例として前節で新しく作成した“ newprog.c ”をプロジェクトに追加します。

[Project->Add Files...]により、Add File(s)ダイアログボックス（**図** 1.16）を表示します。追加するファイル（ここでは、“ newprog.c ”）を選択し、[Add]をクリックするとプロジェクトに追加されます。

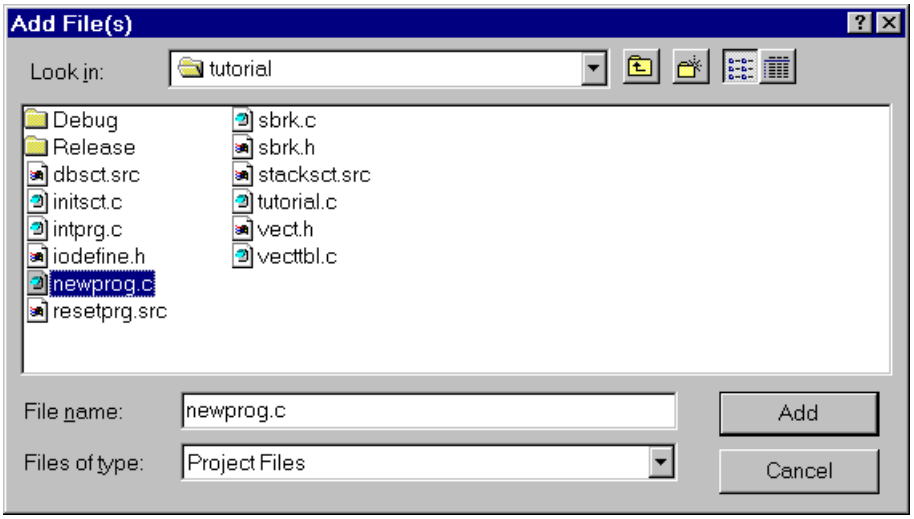

図 1.16 **プロジェクトへのファイルの追加**

プロジェクトに追加したファイルは、**図** 1.17に示すようにワークスペースウィンドウの[Projects]タブに表示されます。

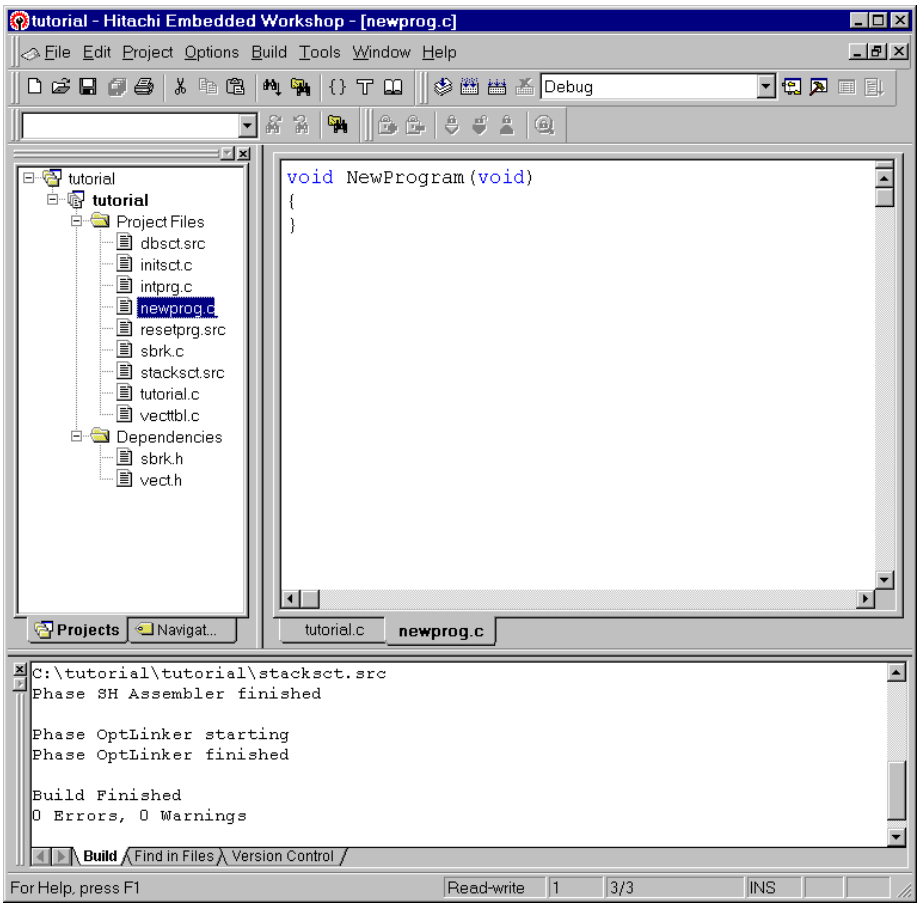

図 1.17　**ファイルを追加したプロジェクト**

次に、プロジェクトで不要となったファイルの削除方法を説明します。[Project->Remove File...]により、Remove Project Files ダイアログボックス（**図** 1.18）を表示します。ここで、削除したいファイルを選択（複数も可能）して[Remove]をクリックすると[Project files]のリストからファイル名が削除されます。[OK]をクリックすると実際にプロジェクトからファイルが削除されます。[Cancel]をクリックすると削除は無効になります。

また、ワークスペースウィンドウの[Projects]タブでファイルを選択して[Delete]キーを押下してもプロジェクトからファイルは削除されます。ただし、この場合は削除の取り消しができません。

なお、ファイルをプロジェクトから削除するのではなく、一時的にビルドから外したい場合は、**「**1.4.4 **プロジェクトファイルの除外」**を参照してください。

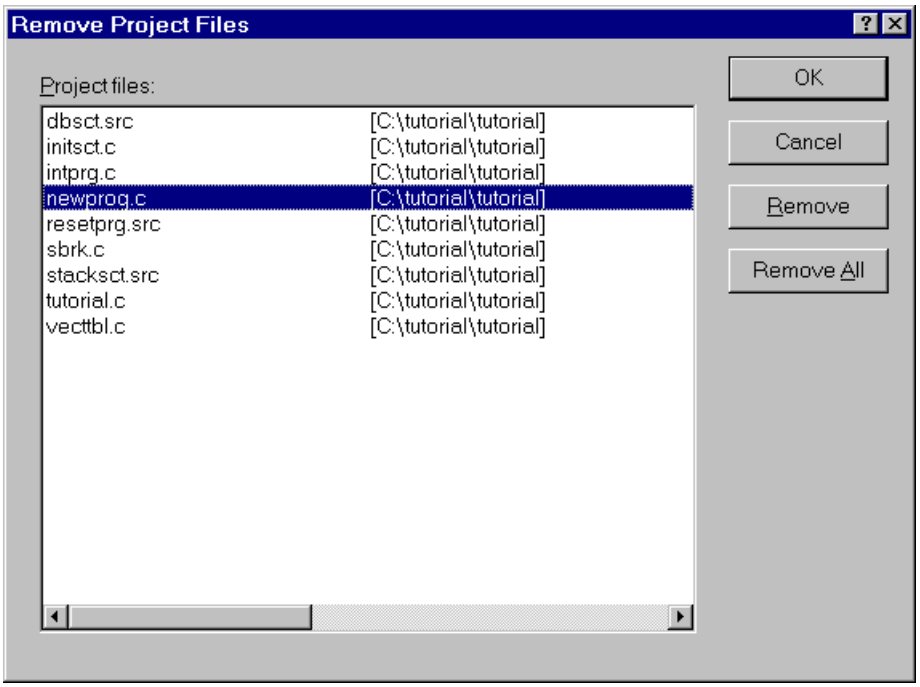

図 1.18　Remove Project Files **ダイアログボックス**

1.3.3 プロジェクトで使えるファイルの種類

プロジェクトジェネレータによりC言語とアセンブリ言語で記述したファイルをプロジェクトに登録しました。例えば、“ *.c ”はC言語ソースファイルに属し、“ *.src ”はアセンブリソースファイルに属します。

次に、プロジェクトに追加できるファイルの種類について説明します。

HEW は、ファイルの種類をその内容ではなく拡張子で識別します。[Project->File Extensions...]により、File Extensions ダイアログボックス（**図** 1.19）を表示します。ここで、プロジェクトで使用できるファイルの拡張子を確認できます。すべての拡張子はファイルグループにより分類され、このファイルグループ単位でビルド時に使うツールを決めることができます。

ユーザが独自に使用している拡張子を HEW で使用したい場合は、[Add...]をクリックして新たな拡張子を定義する必要があります。

また、[Extension]列で表示されているアイコンはファイルを開くためのツールを示しています。

ここで、アイコンが

で表示されている拡張子のファイルは HEW のエディタで開けるテキストファイルを表わします。

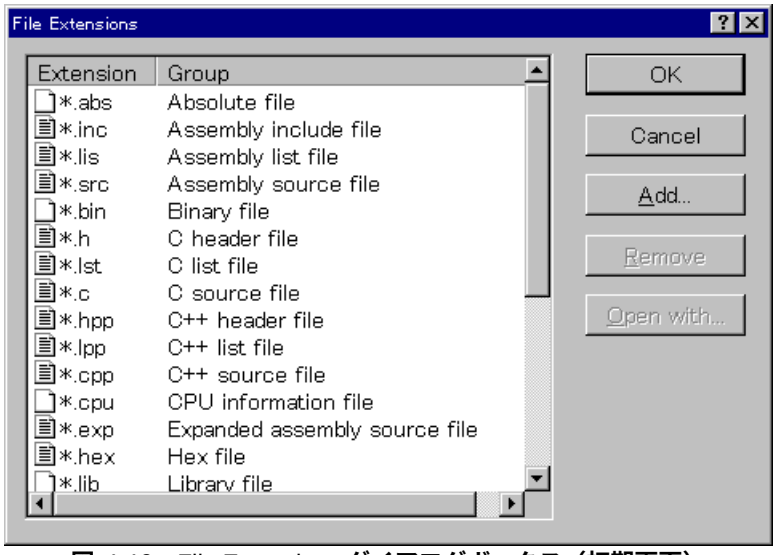

図 1.19　File Extensions **ダイアログボックス（初期画面）**

File Extensions ダイアログボックスで[Add]をクリックすると、Define File Extension ダイアログボックス（**図** 1.20）を表示します。このダイアログボックスの[File extension]に新たに定義する拡張子名を入力し、[File group]を設定します。既存のグループへの追加であれば[Existing group]を選択し、下のドロップダウンリストボックスから該当するグループを選択します。

また、新しいグループを作成する場合は、[New group]を選択し、下の入力エリアにグループ名を入力します。

ここでは、拡張子“ asm ”を[File extension]に指定し、ドロップダウンリストから“ Assembly source file ”を選択します。[OK]をクリックすると、“ *.asm ”ファイルも“ Assembly source file ”グループとしてプロジェクトに追加することができるようになります（**図** 1.21）。

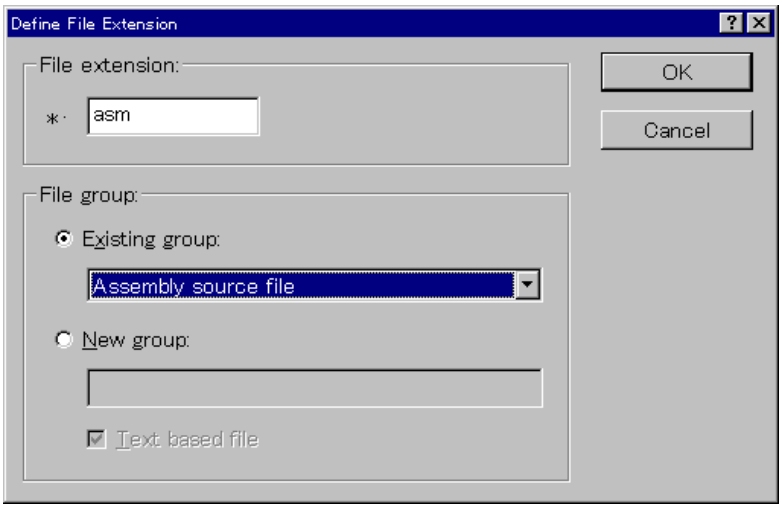

図 1.20　Define File Extension **ダイアログボックス**

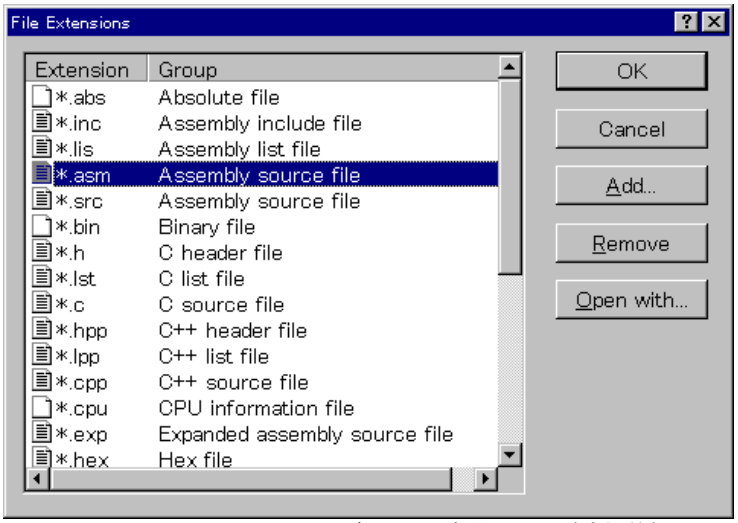

図 1.21　File Extensions **ダイアログボックス（追加後）**

1.3.4 ワークスペースウィンドウのカスタマイズ

ワークスペースウィンドウの[Projects]タブ上でマウスの右ボタンをクリックし、表示されるポップアップメニューで[Configure View...]を選択すると、Configure View ダイアログボックス（**図** 1.22）を表示します。ここでの設定で、ファイルごとのインクルード関係やファイルグループ単位での表示（**図** 1.23）などが可能になります。

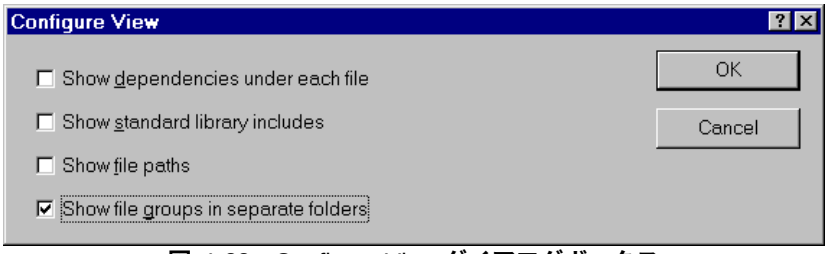

図 1.22 Configure View **ダイアログボックス**

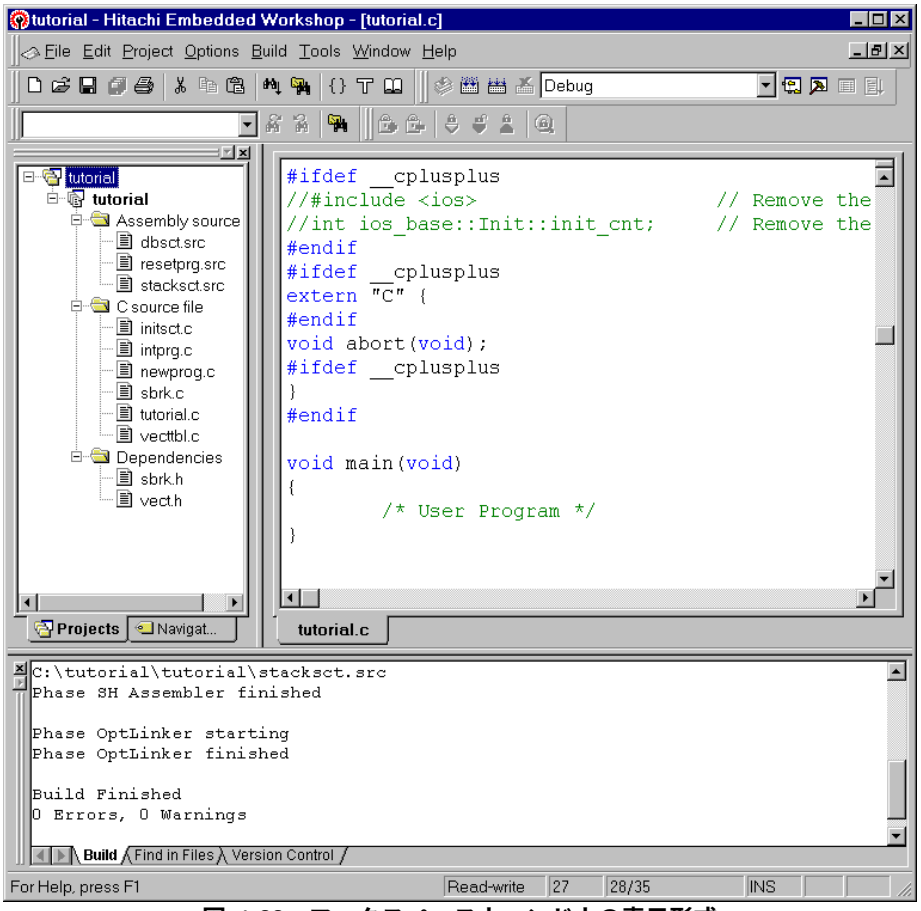

図 1.23 **ワークスペースウィンドウの表示形式**

1.4 オブジェクトの作成

1.4.1 ビルドの機能

オブジェクトの作成には、２種類の方法があります。コンパイル、アセンブル、リンクなどの一連のオブジェクト作成処理を、すべてのソースファイルに対して実施する全ビルドと、前回のオブジェクト作成以降に修正したソースファイル（または、インクルードファイルを修正したソースファイル）だけに対して実施するビルドの２種類です。

（ビルドは、ソースファイルなどの修正だけでなく、オプションの設定変更やプロジェクトファイルの構成を変更することにより影響を受けるファイルも対象になります）

ビルドは、[Build->Build]で実行を開始し、全ビルドは[Build->Build All]で実行を開始します。

なお、単一ファイルをコンパイル、または、アセンブルする場合は、ワークスペースウィンドウでファイルを選択し、[Build->Build File]を選択します。

注意 1. **ビルド実行開始時、エディタウィンドウの環境設定により、編集中のファイルはすべて保存されることがあります。編集したファイルで保存しないものについては、あらかじめファイルを閉じておいてください。**
エディタ機能およびエディタウィンドウのカスタマイズについては、「Hitachi Embedded Workshop **ユーザーズマニュアル」を参照してください。**

2. **プロジェクトに関連するファイルはビルド処理と競合する可能性があります。このようなファイルは、ビルド中に保存しないでください。**

1.4.2 オプション設定

次にオプションの変更方法を説明します。

プロジェクトジェネレータでプロジェクトを作成することにより、ビルドで必要となる基本的なオプションは設定されています。新たにオプションを変更する場合は、[Options-> <ツール名>]により表示する各ツールごとのオプション設定ダイアログボックスで設定します。

図 1.24に C/C++コンパイラのオプション設定ダイアログボックスを示します。ダイアログボックスは、左側のプロジェクトファイルリストと右側のカテゴリごとのオプション設定部分で構成されています。左側のファイルリストでファイルを選択すればそのファイル固有のオプションが設定でき、ファイルグループのフォルダを選択すればファイルグループに登録されているプロジェクトファイルで共通のオプションを設定できます。複数ファイルのオプションを同時に設定する場合は、[Ctrl]キーまたは[Shift]キーを押しながらマウスで該当するファイルを選択してください。

また、“ Default Options ”を選択すれば、ファイルグループのオプションの初期設定が変更できます。“ Default Options ”はファイルグループごとに設定できます。“ Default Options ”に設定したオプションは、そのファイルグループの拡張子を持つファイルをプロジェクトに追加した時に自動的に設定されます。

例えば、拡張子‘ C ’を持つファイル（“ *.C ”）をプロジェクトに追加すると、そのファイルには“ C source file ”の“ Default Options ”が設定されます。

なお、各オプション内容の詳細については、クロスソフトのユーザーズマニュアルを参照してください。

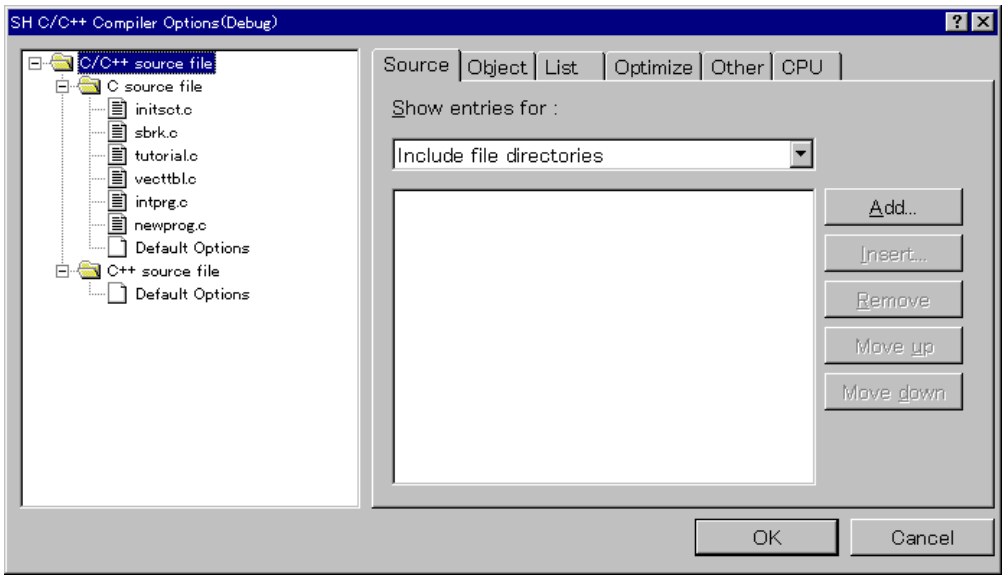

図 1.24 C/C++**コンパイラのオプション設定ダイアログボックス**

1.4.3 コンフィグレーションのカスタマイズ

前項で、プロジェクトファイルのオプション設定について説明しましたが、HEW では、同じプロジェクトファイルに対して複数の組み合わせでオプションを設定することが可能です。

プロジェクトジェネレータは、[Debug]と[Release]の２種類のコンフィグレーション（デバッグオプションの有無が異なる）を作成します。コンフィグレーションを変更する場合は、[Options->Build Configurations...]により Build Configurations ダイアログボックス（**図** 1.25）を表示し、[Current configuration]ドロップダウンリストから使用するコンフィグレーションを選択し[OK]をクリックします。HEW のツールバーにあるドロップダウンリストからもコンフィグレーションの選択は可能です。

また、Build Configurations ダイアログボックスでは、コンフィグレーションの新規作成や削除もできます。詳しくは、**「**Hitachi Embedded Workshop **ユーザーズマニュアル」**の**「第２章　ビルドの基本」**を参照してください。

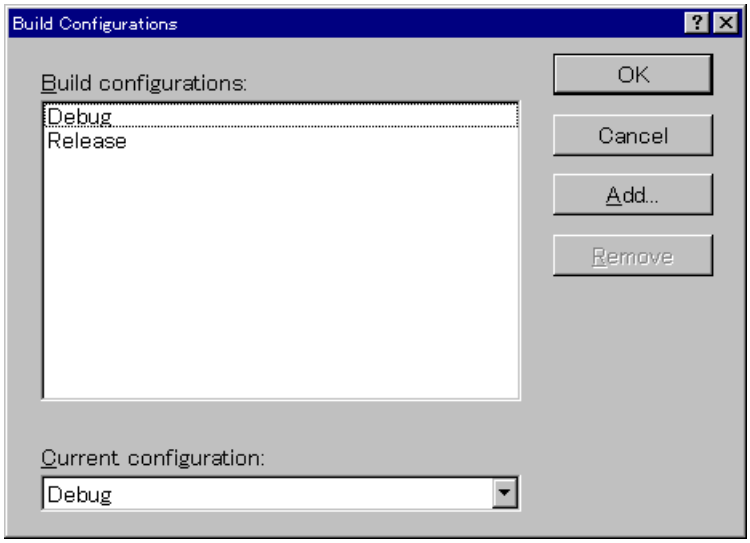

図 1.25　Build Configurations **ダイアログボックス**

1.4.4 プロジェクトファイルの除外

次にプロジェクトからファイルを除外する方法を説明します。これはプロジェクトからファイルを削除するのではなく、コンフィグレーション単位でファイルをビルドから外す場合に使います。

ワークスペースウィンドウ上で除外したいファイルを選択後、マウスの右ボタンをクリックし表示されるポップアップメニューから[Exclude Build <ファイル名>]を選択します。**図** 1.26に示すように、ファイルのアイコンに赤いバツ印が付けられ、ビルドから除外されます。

なお、除外したファイルを再びビルドに入れる場合は、削除時と同様にポップアップメニューから[Include Build <ファイル名>]を選択します。

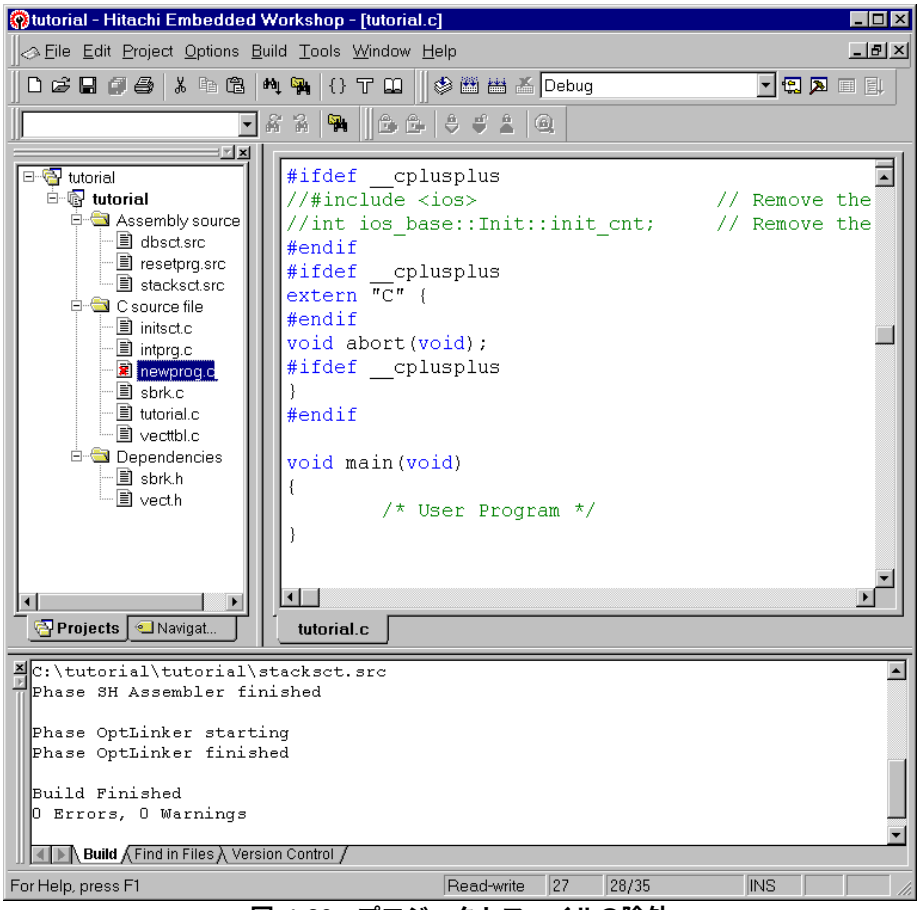

図 1.26 **プロジェクトファイルの除外**

1.4.5 ビルド時のエラー修正

次に、オブジェクトの作成処理中にコンパイラなどでエラーが発生した場合の対処について説明します。

図 1.27にエラー行を含むファイルをビルドした例を示します。ビルド実行により、アウトプットウィンドウにビルド結果（ここではコンパイル結果）が表示され、エラーが発生したことが分かります。ここで、アウトプットウィンドウの[Build]タブ上のエラーメッセージ表示行をダブルクリックすると、エディタウィンドウ上にエラーが発生したファイルを開き、エラー行にカーソルが移動します。

ただし、エラー原因の対策でファイルの行数が変わった場合、カーソルの移動先がずれることがあります。

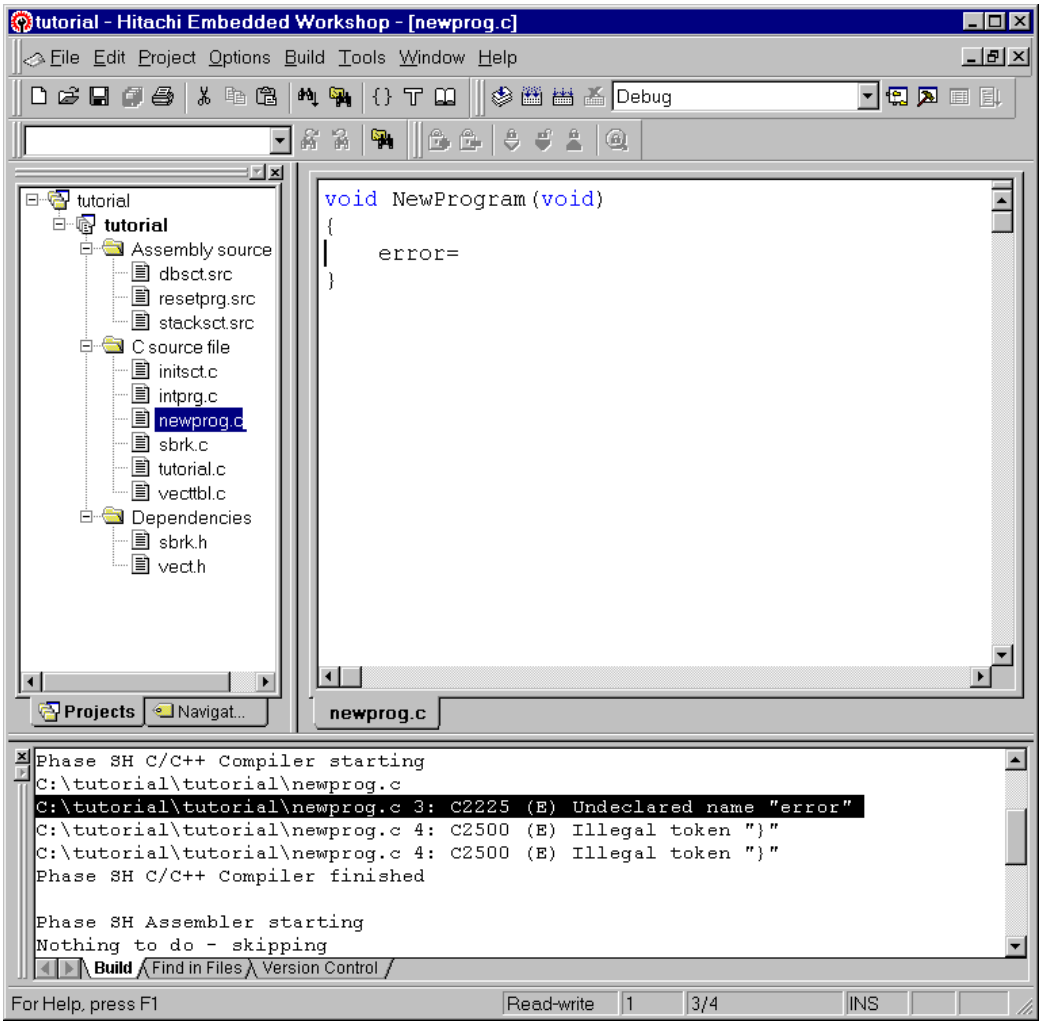

図 1.27 **ビルド時のエラー修正**

1.5 HDI インタフェース

1.5.1 HDI との連動

HEW は HDI（バージョン 4.0 以降）と連動して使うことができます。[Tools->Customize...]より Tools Customize ダイアログボックス（**図** 1.28）を開き、[Debugger]タブで設定します。[HDI location(V4.0 or greater)]で HDI の実行ファイルを指定し、[Session file]に HDI で生成したセッションファイルを指定します。セッションファイルの生成方法は、**「SuperH RISC engine シミュレータデバッガユーザーズマニュアル」**を参照してください。

この設定により HEW のツールバーボタン

で、HDI の起動とセッションファイルの実行まで可能となります。

また、[Download module]に HDI にダウンロードするモジュール（プログラム）を指定後、ビルドによりダウンロードモジュールを更新すると、HEW から HDI に自動的に制御が移るようになります。

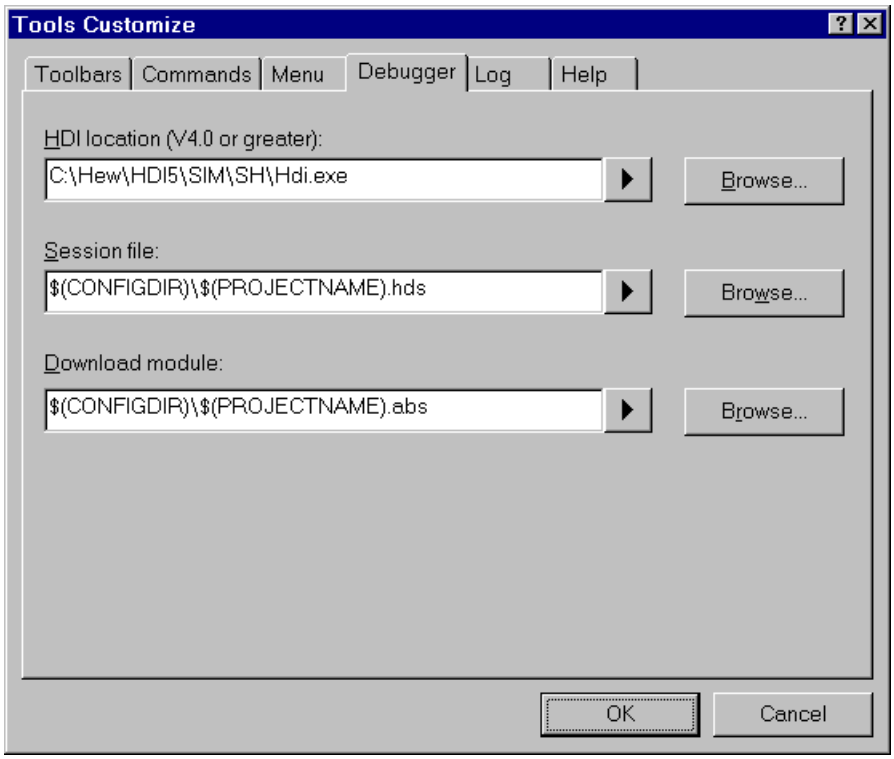

図 1.28　Tools Customize **ダイアログボックス**（Debugger **タブ**）

HDI のソースウィンドウ（図 1.29）上の行をダブルクリックすると、HEW のエディタウィンドウでソースファイルを開き、ダブルクリックした行にカーソルを移動します。

図 1.30には、HDI のソースウィンドウ上で“ tutorial.c ”の「void main(void)」の行をダブルクリックした時の HEW の表示結果を示します。

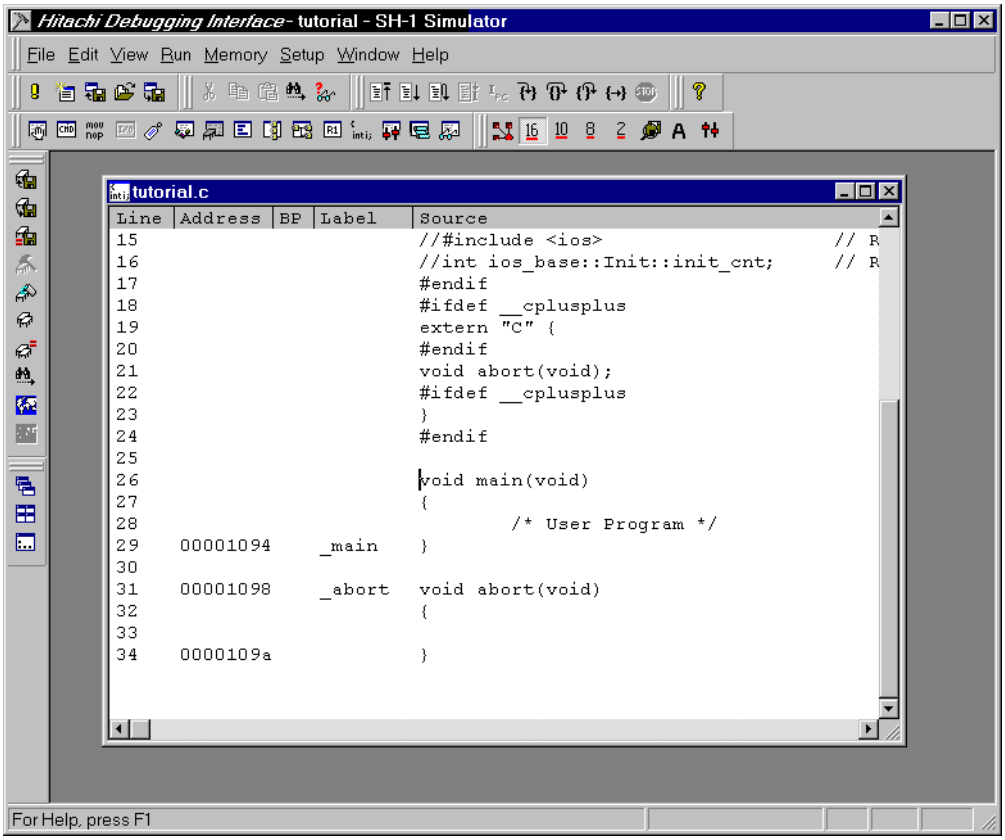

図 1.29　HDI ソースウィンドウ

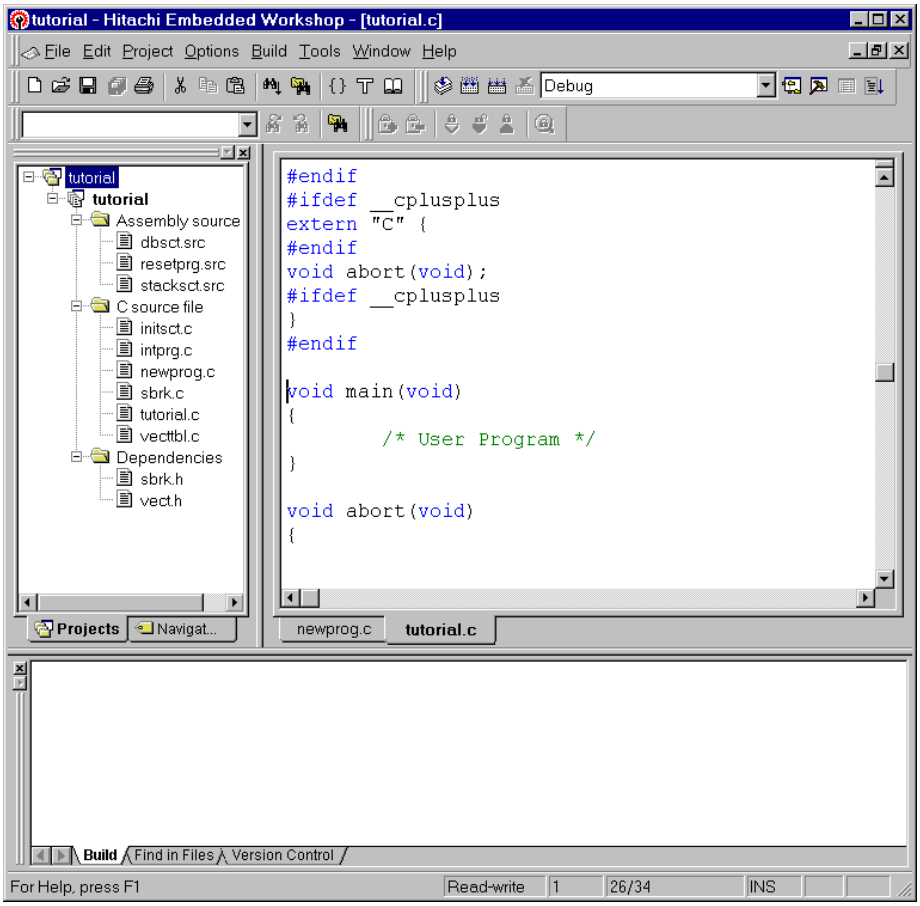

図 1.30　HEW と HDI の連動

1.5.2 Demonstration プロジェクト

プロジェクト作成時に New Project Workspace ダイアログボックス（**図** 1.31）でプロジェクトタイプとして Demonstration を選択すると、HDI シミュレータデバッガの Simulated I/O 機能を組込んだ、サンプルプロジェクトを作成することができます。このプロジェクトでは、main 関数にプログラムの進行状況やデータを表示するために、標準出力関数の一つである printf を使用しています。

Simulated I/O 機能を使う場合は、HDI でデバッグ環境を整えた後に**図** 1.31で示す System Configuration ダイアログボックス（[Setup -> Configure Platform…]で表示）の設定が必要です。

このダイアログボックスの System Call Address の[Enable]をチェックし、HEW が生成した lowlvl.src で定義している SIM_IO の値をアドレスとして入力します（CPU の種類によりアドレス値が異なる場合があります）。

```
SIM_IO:   .EQU        H'0000        （この場合は、アドレスに 0 を入力）
```

HDI の Simulated I/O Window を開き（[View -> Simulated I/O]で開く）、プログラムを実行すれば Simulated I/O 機能を用いたデバッグが可能となります。

詳細は、**「**SuperH RISC engine **シミュレータデバッガユーザーズマニュアル」**を参照してください。

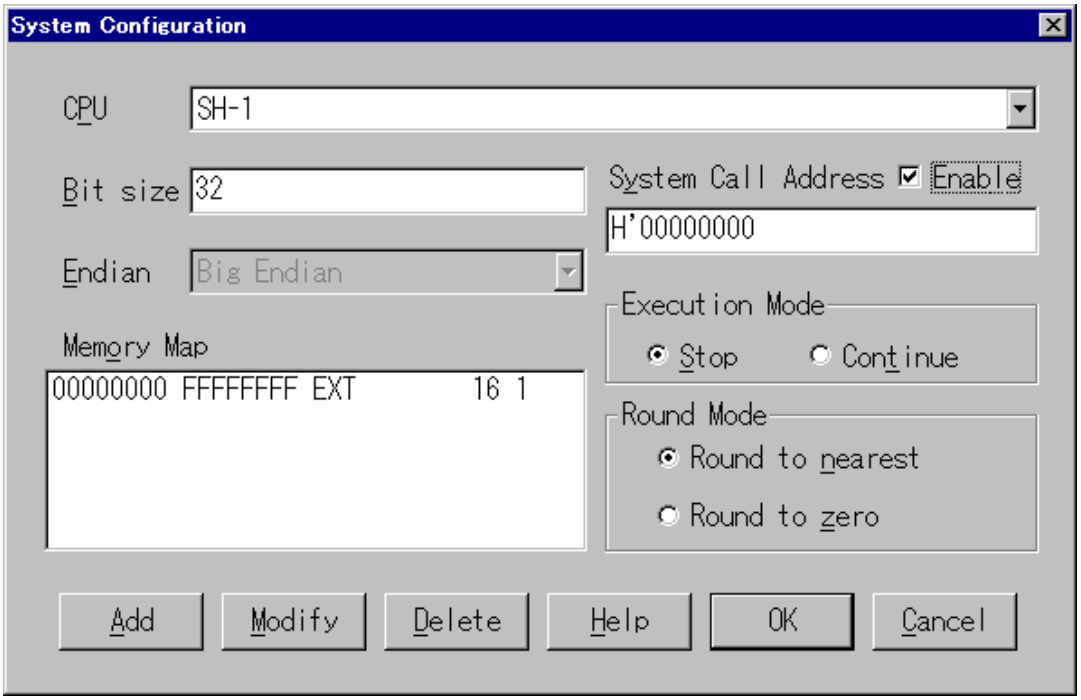

図 1.31 HDI **の** System Configuration **ダイアログボックス**

1.6 HEW の終了

通常、HEW は[File->Exit]で終了します。終了時、使用していたワークスペースの内容を保存するか確認するメッセージボックス（図 1.32）が表示されます。この終了時の設定や HEW 再起動時の設定は[Tools->Options...]で変更できます。詳しくは、「Hitachi Embedded Workshop ユーザーズマニュアル」を参照してください。

なお、HDI や他のユーザ登録ツールは、HEW が終了しても連動して終了しませんので個別に終了させてください。

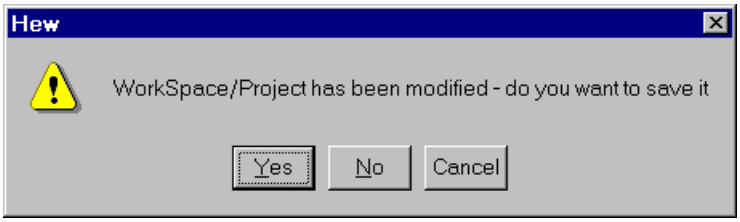

図 1.32 HEW 終了時の確認

2. HDI チュートリアル

2.1 HDI を使ったデバッグ

HDI と共に使われるシミュレータ・デバッガの主な特徴を紹介するために、サンプルプログラムを用いて説明します。

2.1.1 サンプルプログラム

このサンプルプログラムは、10 個のランダムデータを昇順にソートした後、降順にソートする C プログラムに基づいています。

サンプルプログラムでは、以下の処理を行います。

- main 関数でソートするランダムデータを生成します。
- sort 関数では main 関数で生成したランダムデータを格納した配列を入力し、昇順にソートします。
- change 関数では、sort 関数で生成した配列を入力し、降順にソートします。

サンプルプログラムは、インストールディスクの sort.c ファイルで提供しています。コンパイルしたバージョンのチュートリアルは、sort.abs ファイルで提供しています。

2.1.2 HDI の実行

HDI の機能をサンプルプログラムを用いて説明するため、HEW からではなく、HDI 単独で起動します。

- HDI を実行するには、**[HDI for SH series simulator]**アイコンをダブルクリックしてください。

図 2.1 HDI for SH series simulator **アイコン**

2.1.3 ターゲットの選択

HDI は複数のターゲットプラットフォームをサポートしています。現在のセッションのプラットフォームを選択するように指示します。

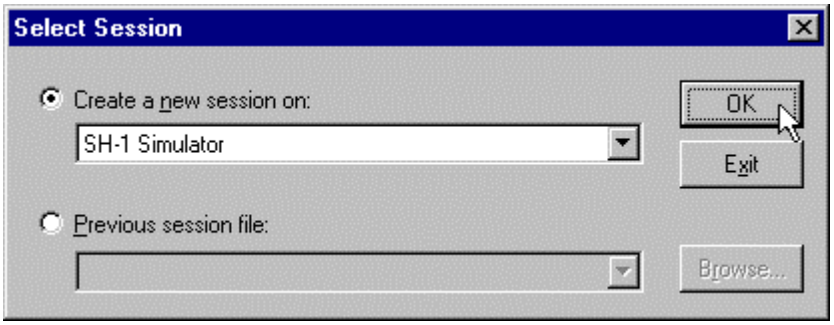

図 2.2 Select Session **ダイアログボックス**

- このチュートリアルでは、SH1 Simulator を選択し、**[OK]**ボタンをクリックしてください。

[File]メニューから **[New Session…]**を選択すれば、いつでもターゲットプラットフォームを変更できます。ただし、プラットフォームを１つしかインストールしていなければ、このメニューオプションは使えません。

シミュレータ・デバッガを正しく設定していれば、ステータスバーの"Link up"メッセージと共に、Hitachi Debugging Interface ウィンドウを表示します。次ページにウィンドウの機能を示します。

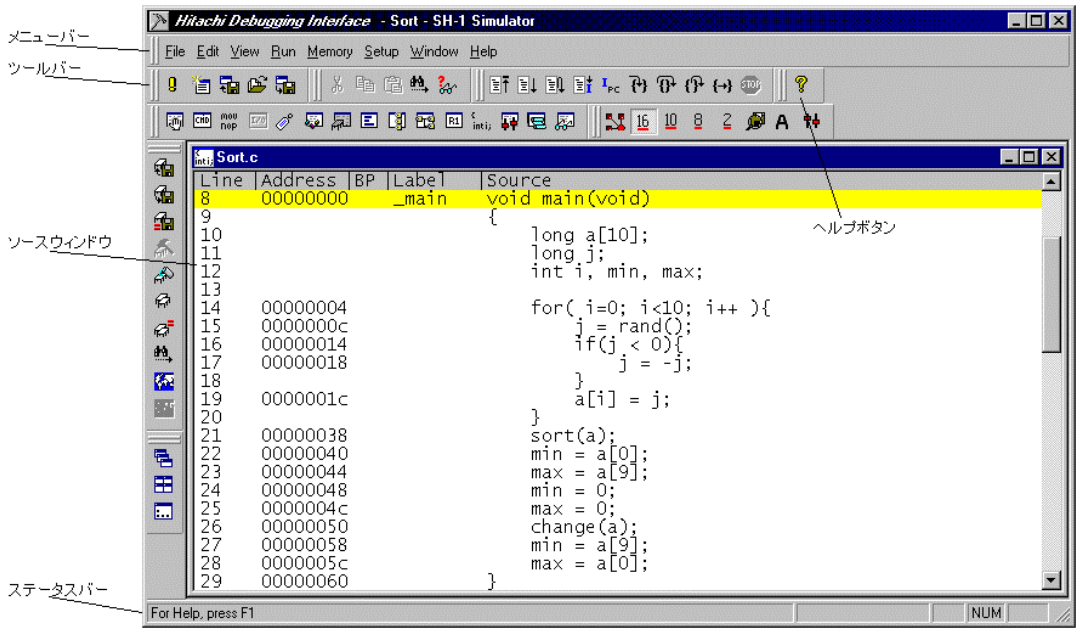

図 2.3 Hitachi Debugging Interface **ウィンドウ**

メニューバー	HDI **デバッガを使うための** HDI **コマンドへのアクセスを示します。**
ツールバー	**最もよく使う** HDI **コマンドのショートカットとして便利なボタンです。ツールボタンは、各メニューごとにブロック化しています。ツールボタンの配置はクリック・ドラッグ操作により設定することができます。また、表示/非表示は [Setup]メニューの [Customize]サブメニューのうち [Toolbar]オプションで設定することができます。**
ソースウィンドウ	**デバッグしているプログラムのソースを表示します。**
ステータスバー	**シミュレータ・デバッガの状態やダウンロードの進捗状況を表示します。**
ヘルプボタン	HDI **ユーザインタフェースの特徴に関する文脈依存ヘルプを起動します。**

2.1.4 メモリリソースの確保

次のステップでは、開発しているアプリケーションを動作させるためにメモリリソースの確保を行います。

- **[Memory]**メニューから **[Configure Map...]**を選択し、現在のメモリマップを表示してください。

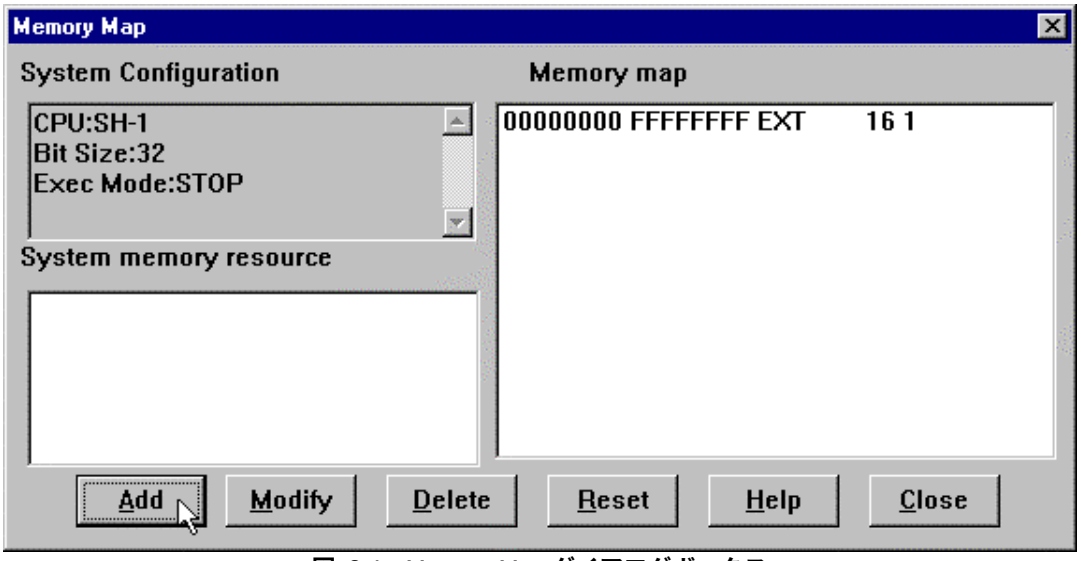

図 2.4 Memory Map **ダイアログボックス**

[Add]ボタンをクリックすると、System Memory Resource Modify ダイアログボックスを表示します。

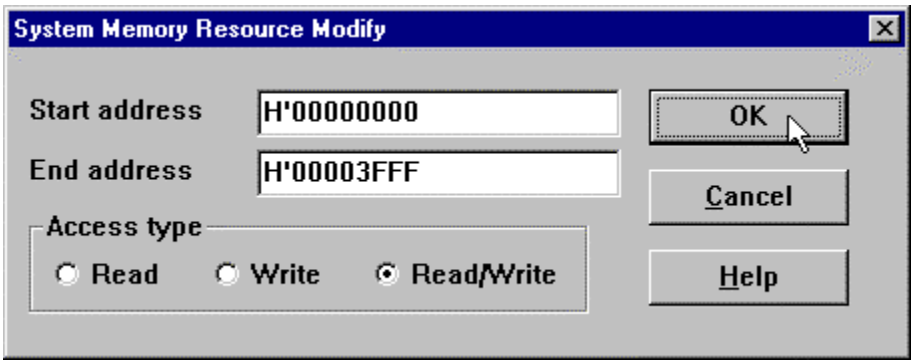

図 2.5 **System Memory Resource Modify ダイアログボックス**

[Access type]には、以下の３つのアクセス種別の１つを指定できます。

アクセス種別	説明
Read	**読み出しのみ可能**
Write	**書き込みのみ可能**
Read/Write	**読み出し/書き込み可能**

本チュートリアルでは、H'00000000 から H'00003FFF のメモリリソースを読み出し／書き込み可能領域として確保してください。

- [Start address]フィールドに H'00000000、[End address]フィールドに H'00003FFF、[Access type]を [Read/Write]に設定し、[OK]ボタンをクリックしてください。

Memory Map ダイアログボックスは変更した範囲を表示します。

- **[Close]**ボタンをクリックしてダイアログボックスを閉じてください。

2.1.5 チュートリアルプログラムのダウンロード

- **[File]**メニューから **[Load Program...]**を選択して、Load Program ダイアログボックスを開いてください。

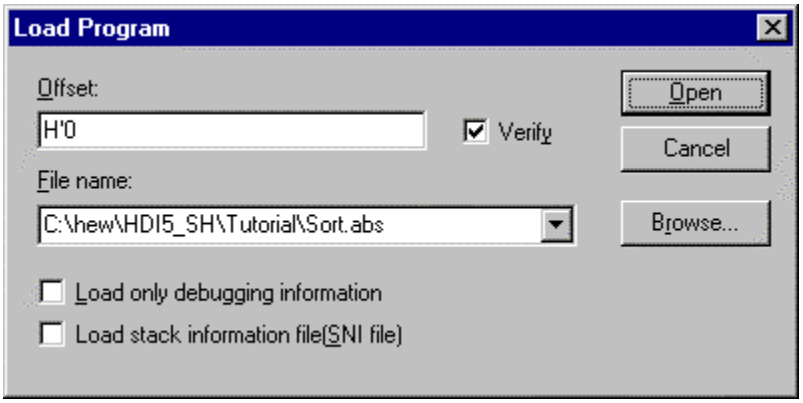

図 2.6 Load Program **ダイアログボックス**

- **[File name]**に sort.abs ファイルの存在するディレクトリのパス名を入力するか、**[Browse...]**ボタンをクリックしてください。**[Browse...]**ボタンをクリックすると Open ダイアログボックスが開くので、sort.abs を選択後 **[Open]**ボタンをクリックしてください。

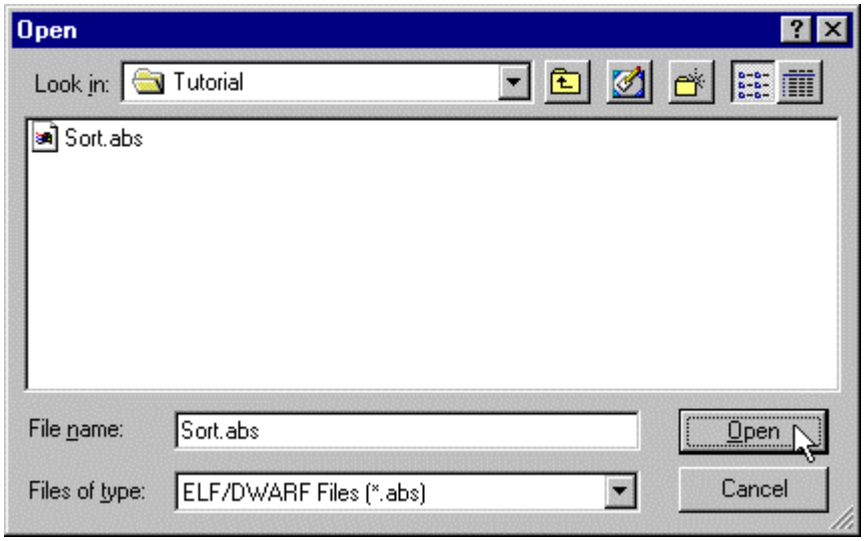

図 2.7 Open **ダイアログボックス**（[Load Program...]）

- Load Program ダイアログボックスの **[Open]**ボタンをクリックしてください。

ファイルをロードすると、以下のダイアログボックスに、プログラムコードが書き込まれたメモリエリアに関する情報を表示します。

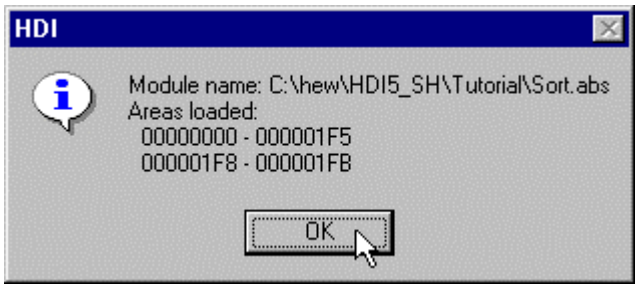

図 2.8 HDI **ダイアログボックス**

- **[OK]**ボタンをクリックしてください。

2.1.6 ソースプログラムの表示

HDI では、ソースレベルでプログラムをデバッグできます。したがって、デバッグするときに、C プログラムのリストをマシンコードと並べて見ることができます。そのためには、オブジェクトファイルに対応する C ソースファイルを読み込ませてください。

- **[View]**メニューから **[Source…]**を選択してください。
- ロードしたオブジェクトファイルに対応する C ソースファイルを選択してください。

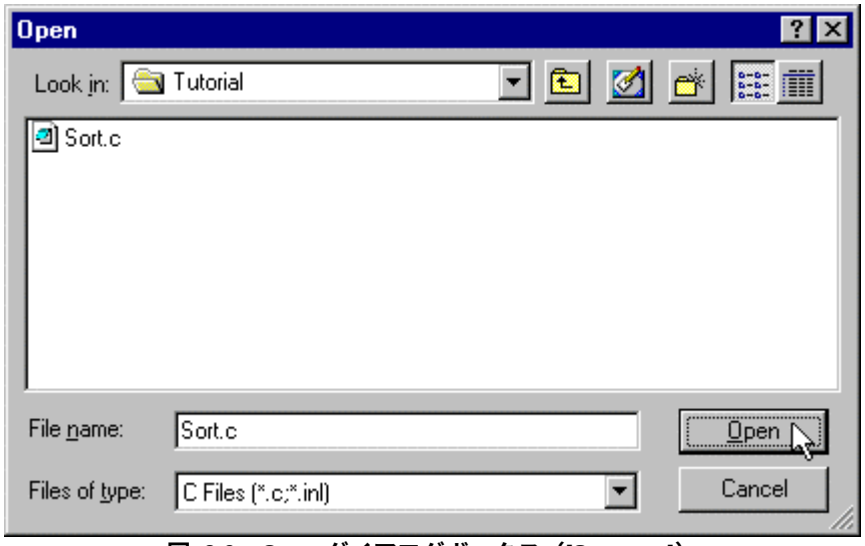

図 2.9 Open **ダイアログボックス**（[Source...]）

- sort.c ファイルを選択し、**[Open]**ボタンをクリックして Source ウィンドウを表示してください。

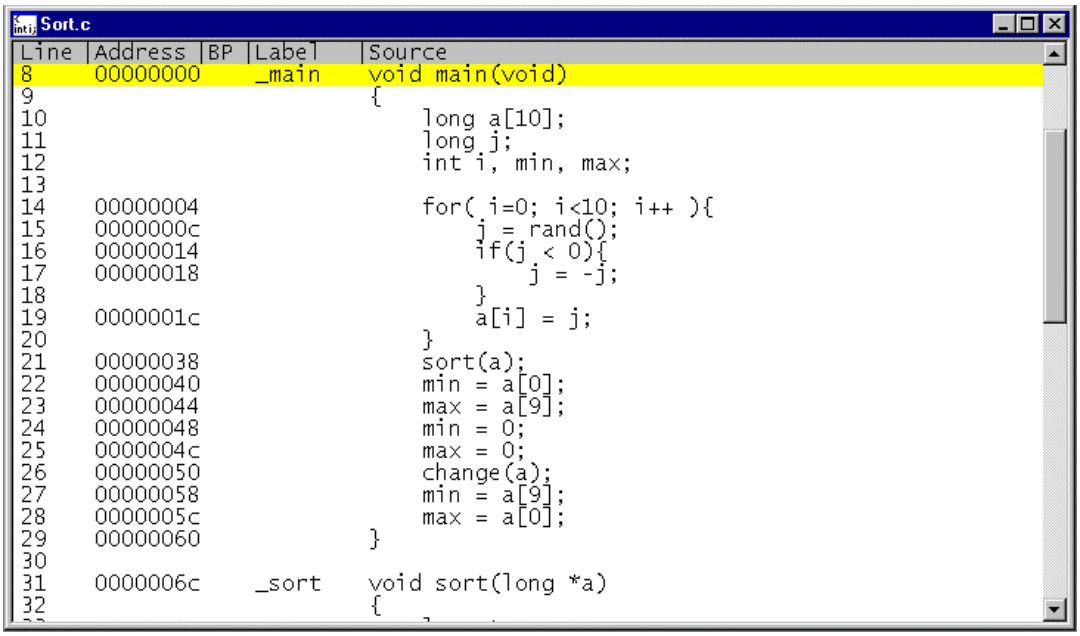

図 2.10 Source **ウィンドウ（ソースプログラムの表示）**

- 必要ならば、**[Setup]**メニューの **[Customize]**サブメニューから **[Font]**オプションを選択し、コンピュータに合ったフォントとサイズを選択してください。

Source ウィンドウは、最初はメインプログラムの先頭を表示しますが、スクロールバーを使ってプログラムをスクロールし、他の部分を見ることができます。

2.1.7 PC ブレークポイントの設定

Source ウィンドウによって、ブレークポイントを簡単に設定できます。たとえば、以下のようにして sort 関数のコール箇所にブレークポイントを設定します。

- sort 関数コールを含む行の **[BP]**コラムをダブルクリックしてください。

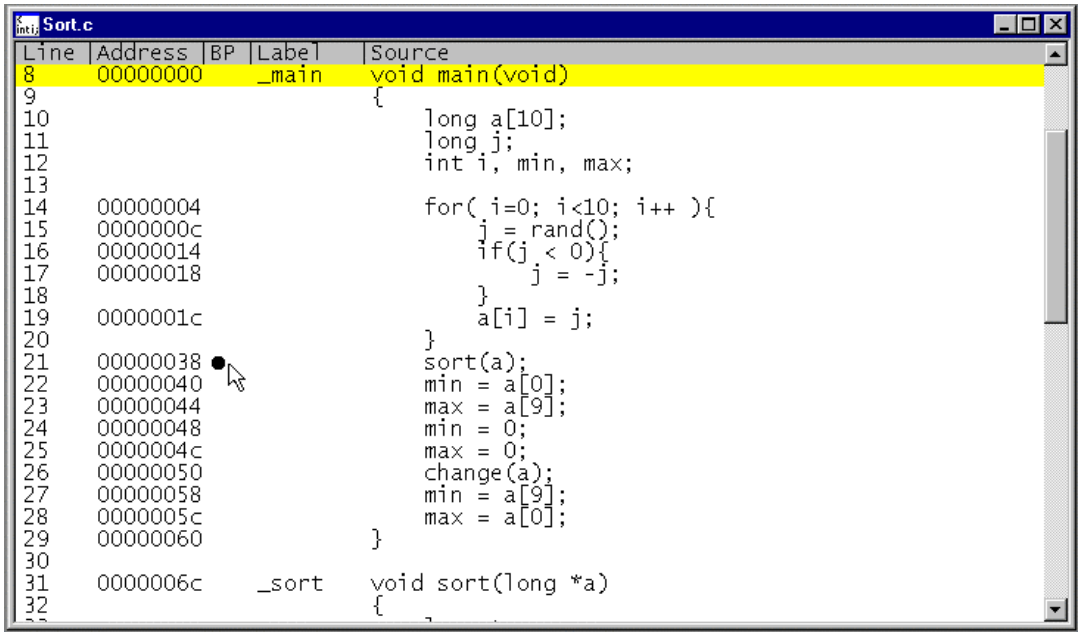

図 2.11 Source ウィンドウ（ブレークポイントの設定）

sort 関数コールを含む行に ●を表示し、そのアドレスに PC ブレークポイントを設定したことを示します。

2.1.8 トレース情報取得条件の設定

- **[View]**メニューから **[Trace]**を選択して、Trace ウィンドウを開いてください。さらに、Trace ウィンドウ上で右クリックしてポップアップメニューを表示し、**[Acquisition...]**を選択してください。

以下の Trace Acquisition ダイアログボックスを表示します。

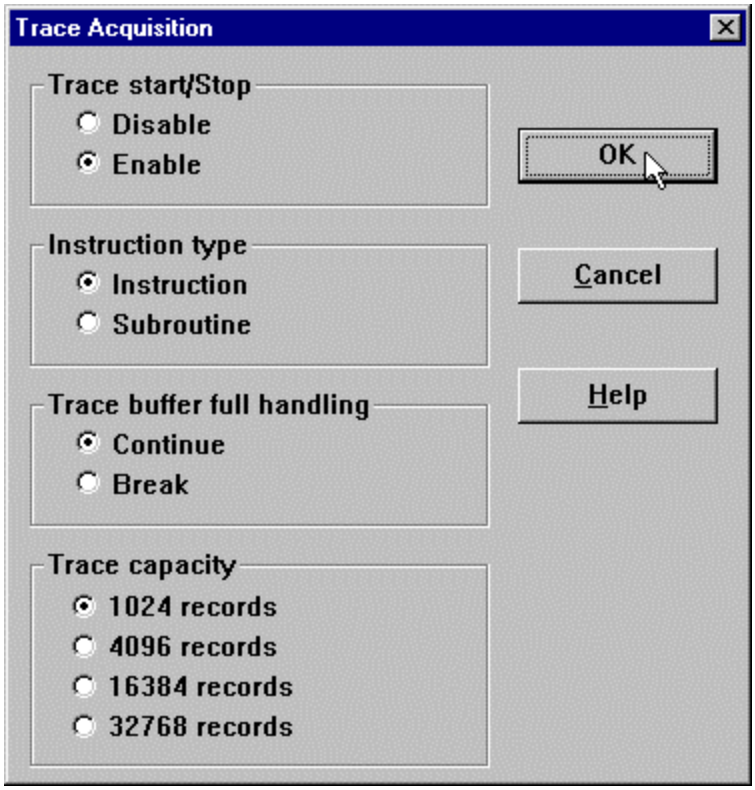

図 2.12　Trace Acquisition **ダイアログボックス**

- Trace Acquisition ダイアログボックスの **[Trace start/Stop]**を **[Enable]**に設定し、**[OK]**ボタンをクリックしてトレース情報取得を有効にしてください。

2.1.9　パフォーマンス・アナリシスの設定

- **[**View]メニューから [Performance Analysis]を選択して、Performance Analysis ウィンドウを開いてください。さらに、Performance Analysis ウィンドウ上で右クリックしてポップアップメニューを表示し、[Add Range...]を選択してください。

以下の Performance Option ダイアログボックスを表示します。

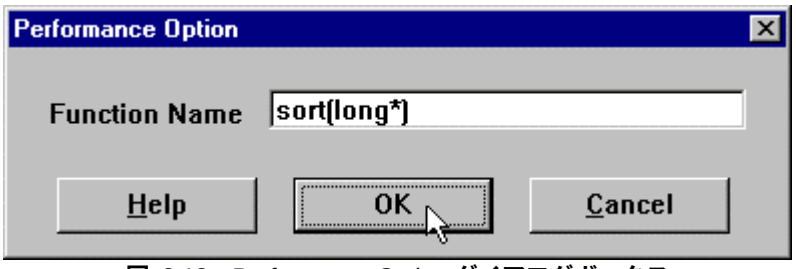

図 2.13　Performance Option **ダイアログボックス**

- Performance Option ダイアログボックスの **[Function Name]**に "sort(long*)"を設定し、**[OK]** ボタンをクリックしてください。

Performance Analysis ウィンドウの **[Function]**に " sort(long*)"を設定します。

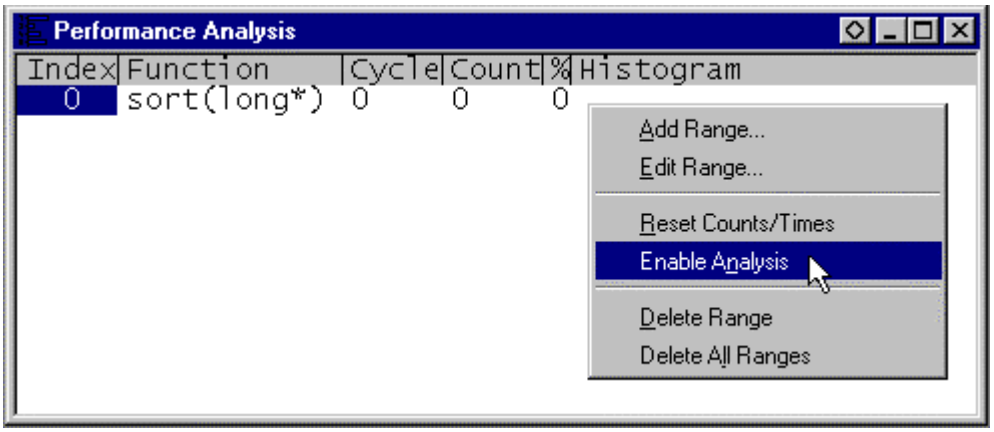

図 2.14 Performance Analysis **ウィンドウ（設定）**

- Performance Analysis ウィンドウ上で右クリックしてポップアップメニューを表示し、**[Analysis Enabled]**を選択して、パフォーマンス・アナリシス情報取得を有効にしてください。
- タイトルバーの**[×]**をクリックしてウィンドウを閉じてください。

2.1.10 スタックポインタの設定

プログラムを実行するために、スタックポインタを設定してください。

- **[View]**メニューから **[Registers]**を選択して、Registers ウィンドウを開いてください。
- スタックポインタ（R15）の値を変えるには、Registers ウィンドウで **[R15]**に対する**[Value]** コラムをダブルクリックしてください。

以下のダイアログボックスによって値を変更できます。

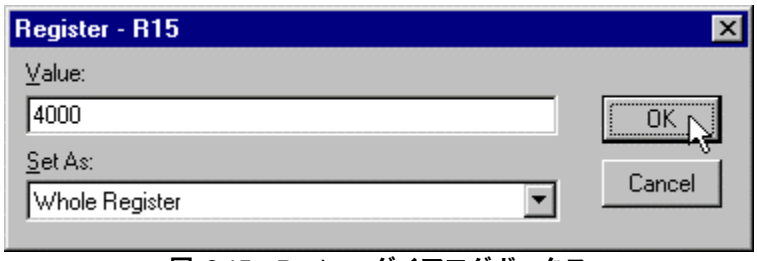

図 2.15 Register **ダイアログボックス**

- 本サンプルプログラムでは、H'4000 を設定し、**[OK]**ボタンをクリックしてください。

2.1.11 プログラムの実行

- プログラムを実行するには、**[Run]**メニューから **[Go]**を選択するか、またはツールバー上の **[Go]**ボタンをクリックしてください。

図 2.16 Go **ボタン**

プログラムはブレークポイントを設定したところまで実行します。プログラムが停止した位置を示すために Source ウィンドウ中でステートメントを強調表示します。また [Break=PC Breakpoint]メッセージをステータスバーに表示します。

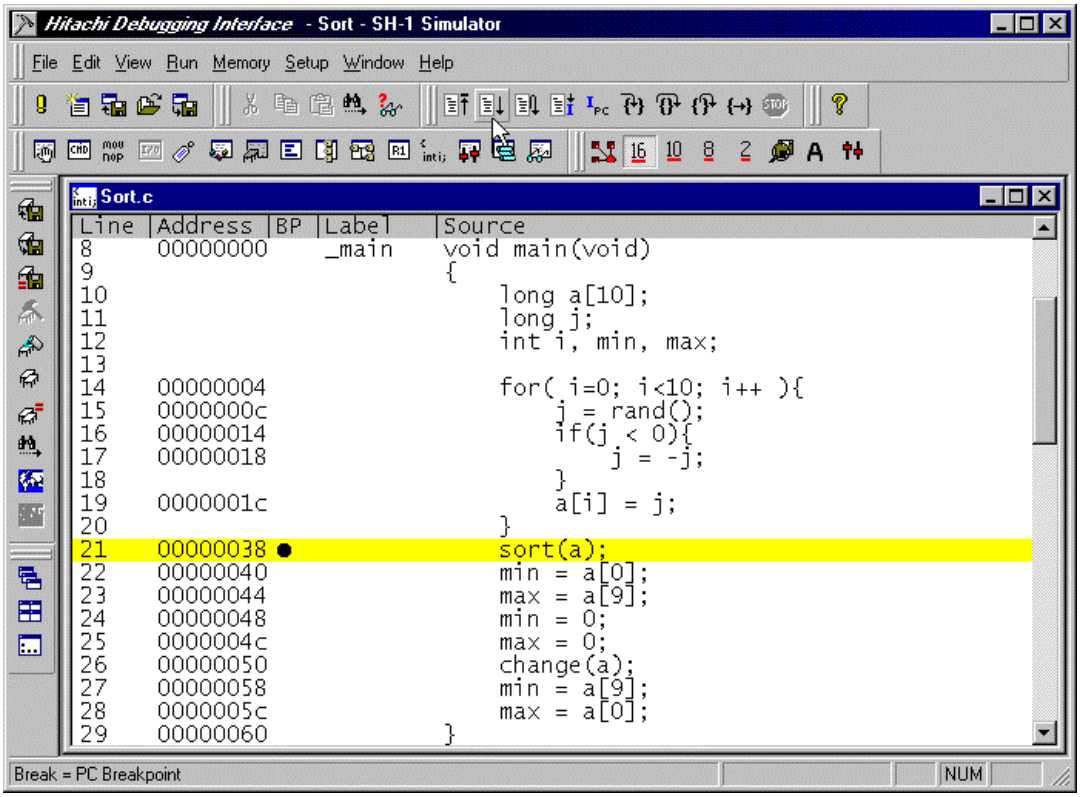

図 2.17 Source **ウィンドウ（ブレーク状態）**

HEW の Launch Debugger ボタン（**「**1.5 HDI **インタフェース」**を参照）から HDI を起動した場合、Source ウィンドウ中の **[Source]**コラムをダブルクリックすることにより、ソースウィンドウに表示しているソースファイルを HEW のテキストエディタで表示、編集することが可能です。

本チュートリアルのサンプルプログラムでは sort.c を HEW でコンパイル、リンクすることにより、本機能が使用できるようになります。

System Status ウィンドウで停止要因が確認できます。

- **[View]**メニューから **[Status]**を選択して、System Status ウィンドウを開いてください。さらに、System Status ウィンドウのうち **[Platform]**シートを表示してください。

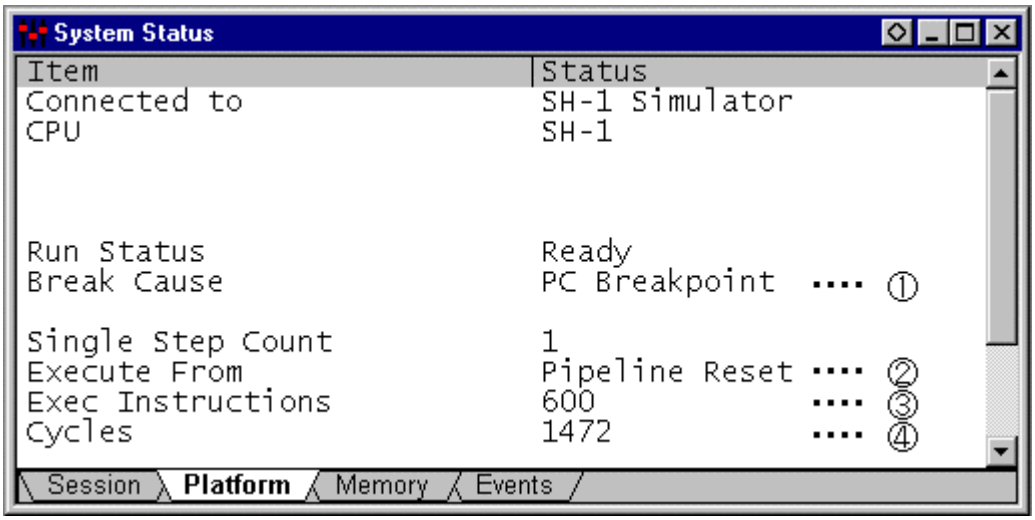

図 2.18 System Status **ウィンドウ**

これは、以下の実行内容を表示します。

① ブレークの原因は PCブレーク
② パイプラインリセットからの実行
③ GOコマンドでの実行命令数は600命令
④ パイプラインリセットからの実行サイクル数は1,472サイクル

Registers ウィンドウでレジスタの値が確認できます。

- **[View]**メニューから **[Registers]**を選択してください。

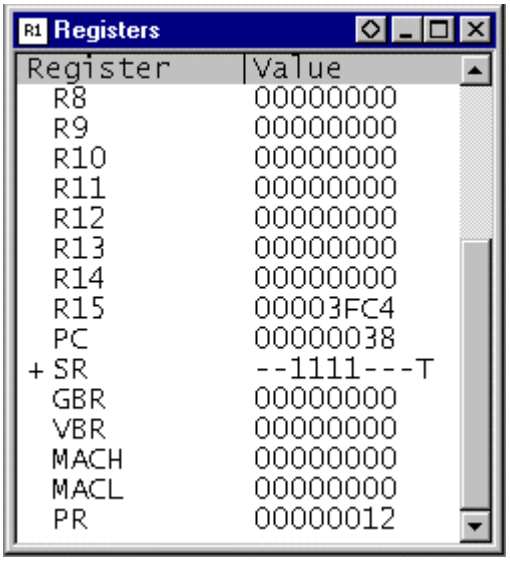

図 2.19 Registers **ウィンドウ**

プログラム停止時の各レジスタの値を確認することができます。

2.1.12 トレースバッファの使い方

トレースバッファを使って、命令実行の履歴を知ることができます。

- **[View]**メニューから **[Trace]**を選択して、Trace ウィンドウを開いてください。ウィンドウの最上部までスクロールアップしてください。
- 必要ならば、**[Setup]**メニューの **[Customize]**サブメニューから **[Font]**オプションを選択し、コンピュータに合ったフォントとサイズを選択してください。

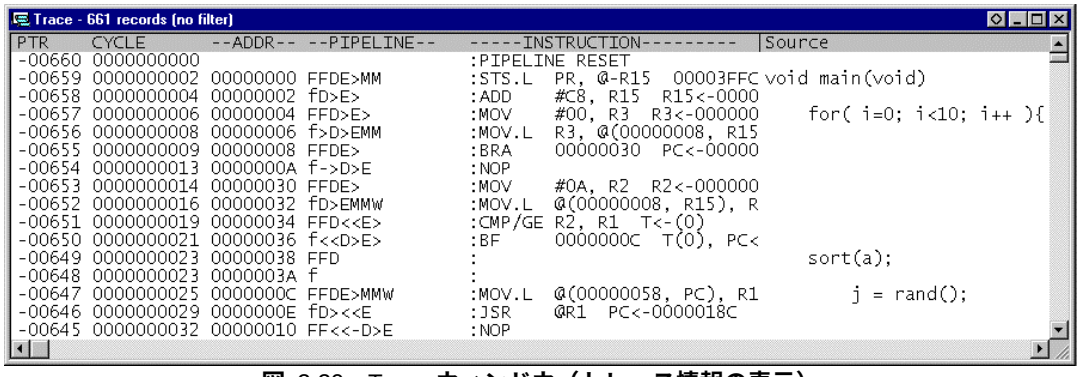

図 2.20 Trace **ウィンドウ（トレース情報の表示）**

トレースディスプレイの 0000000000 サイクル（PIPELINE RESET）から main 関数を実行したことがわかります。

2.1.13 トレースサーチの実行

最初に、Trace ウィンドウ上で右クリックしてポップアップメニューを表示し、**[Find...]**を選択して、Trace Search ダイアログボックスを開いてください。

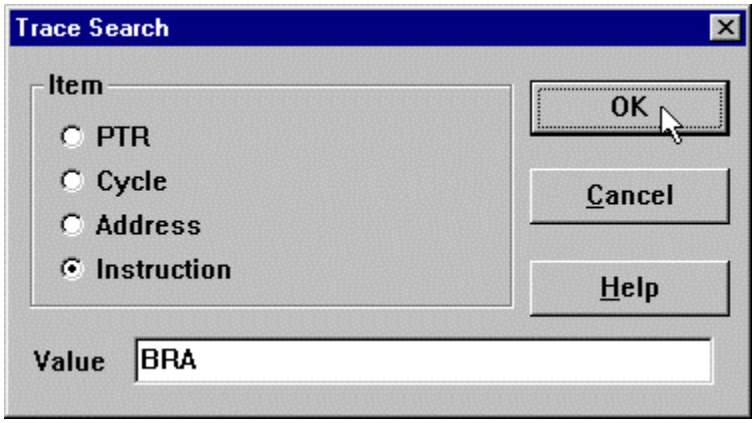

図 2.21 Trace Search **ダイアログボックス**

サーチ項目 **[Item]**とサーチ内容 **[Value]**を設定して **[OK]**ボタンをクリックすると、トレースサーチを実行します。該当するトレース情報があった場合、該当する最初の行を強調表示します。同じサーチ内容 **[Value]**でトレースサーチを続ける場合は、Trace ウィンドウ上で右クリックしてポップアップメニューを表示し、**[Find Next]**を選択してください。次に該当する行を強調表示します。

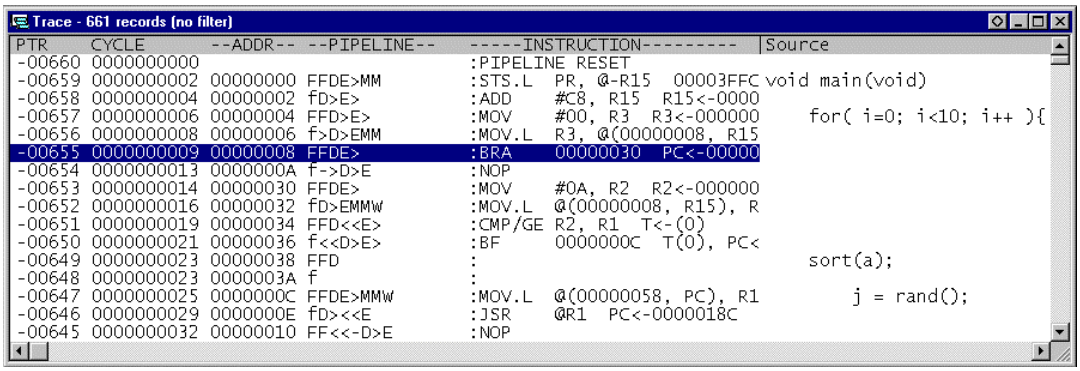

図 2.22 Trace **ウィンドウ（サーチ結果）**

2.1.14 ブレークポイントの確認

プログラムに設定した全てのブレークポイントのリストを Breakpoints ウィンドウで確認することができます。

- **[View]**メニューから **[Breakpoints]**を選択してください。

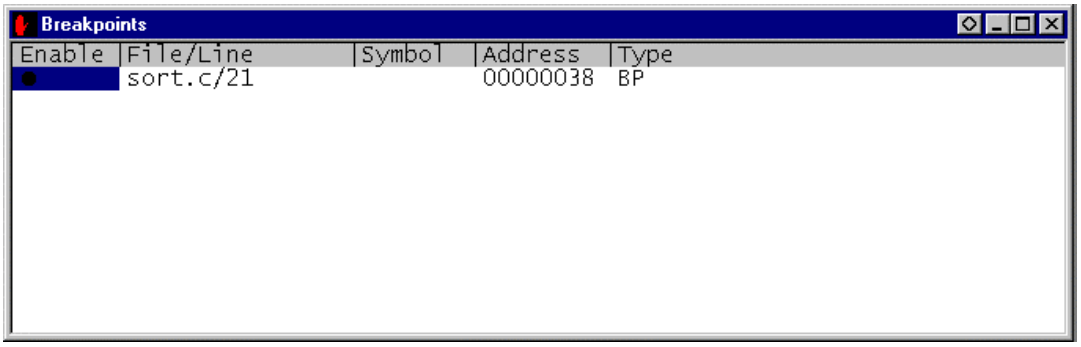

図 2.23 Breakpoints ウィンドウ

Breakpoints ウィンドウによって、ブレークポイントの設定、新しいブレークポイントの定義、およびブレークポイントの削除ができます。

- Breakpoints ウィンドウを閉じてください。

2.1.15 メモリ内容の確認

メモリブロックの内容を Memory ウィンドウで確認することができます。たとえば、ワードサイズで main 列に対応したメモリを確認する場合の手順を以下に示します。

- **[View]**メニューから **[Memory…]**を選択し、**[Address]**フィールドに "main"を入力し、**[Format]**を "Word"に設定してください。

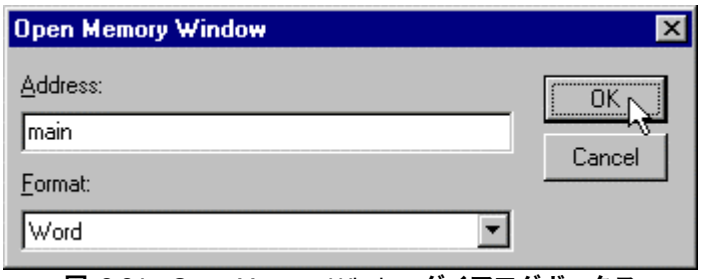

図 2.24 Open Memory Window ダイアログボックス

- **[OK]**ボタンをクリックして、指定したメモリ領域を示す Memory ウィンドウを開いてください。

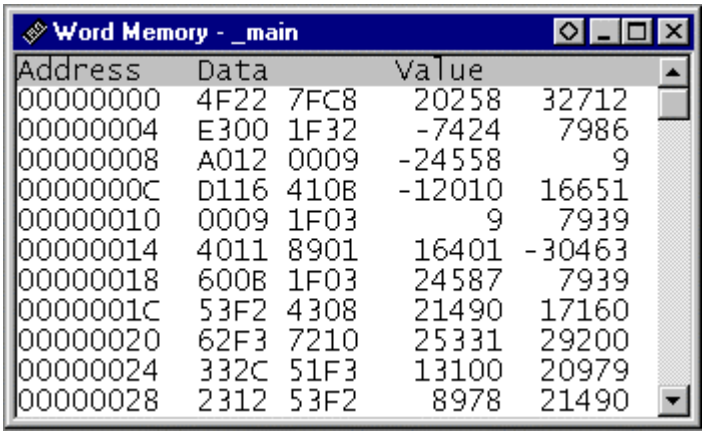

図 2.25 Word Memory **ウィンドウ**

2.1.16 変数の参照

変数の値を Watch ウィンドウで見ることができます。

プログラムの始めに宣言した long 型の配列 a を見る場合：

- Source ウィンドウの a の左にカーソルを置いてください。
- Source ウィンドウ上で右クリックしてポップアップメニューを表示し、 **[Instant Watch...]**を選択してください。

以下のダイアログボックスを表示します。

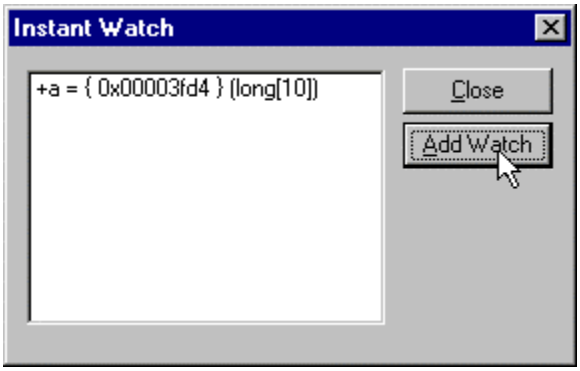

図 2.26 Instant Watch **ダイアログボックス**

- **[Add Watch]**をクリックし、Watch ウィンドウに変数を加えてください。

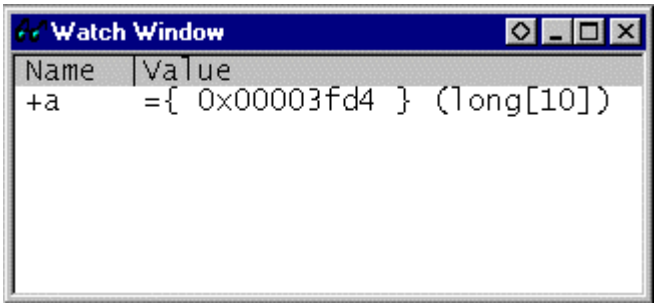

図 2.27 Watch **ウィンドウ（配列の表示）**

また、変数名を指定して、Watch ウィンドウに変数を加えることができます。

- Watch ウィンドウ上で右クリックしてポップアップメニューを表示し、**[Add Watch...]**を選択してください。

以下のダイアログボックスを表示します。

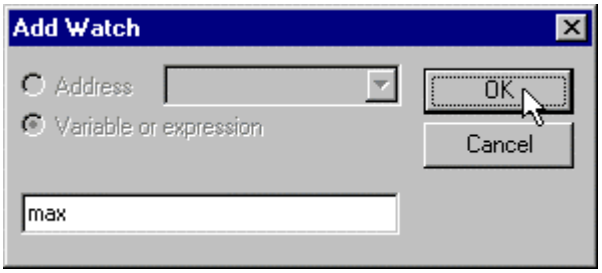

図 2.28 Add Watch **ダイアログボックス**

- 変数 "max" をタイプし、**[OK]**ボタンをクリックします。

Watch ウィンドウに、int 型の変数 max を表示します。

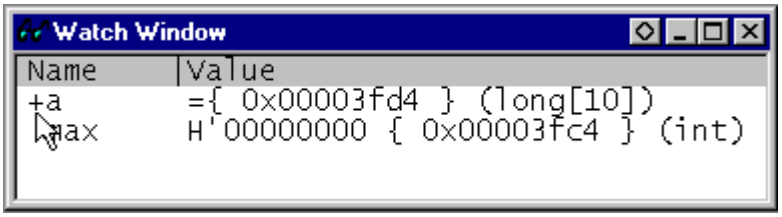

図 2.29 Watch **ウィンドウ（変数の表示）**

Watch ウィンドウの配列 a の前にあるシンボル＋をダブルクリックすると、配列を拡張して各要素を参照することができます。

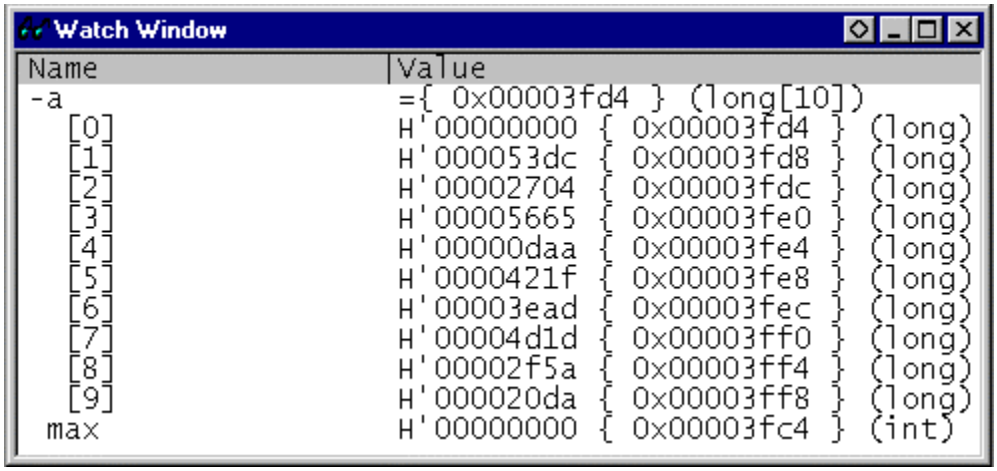

図 2.30 Watch **ウィンドウ（配列要素の表示）**

2.1.17 プログラムのステップ実行

シミュレータ・デバッガは、プログラムのデバッグに有効な各種のステップメニューを備えています。

メニュー	説明
Step In	各ステートメントを実行します（関数内のステートメントを含む）。
Step Over	関数コールを１ステップとして、ステップ実行します。
Step Out	関数を抜け出し、関数を呼び出したプログラムにおける次のステートメントで停止します。
Step…	指定した速度で指定回数分ステップ実行します。

プログラムのステップをデモンストレーションするために、アドレス H'00000038 の sort 関数ステートメントまで実行していることを確認してください。

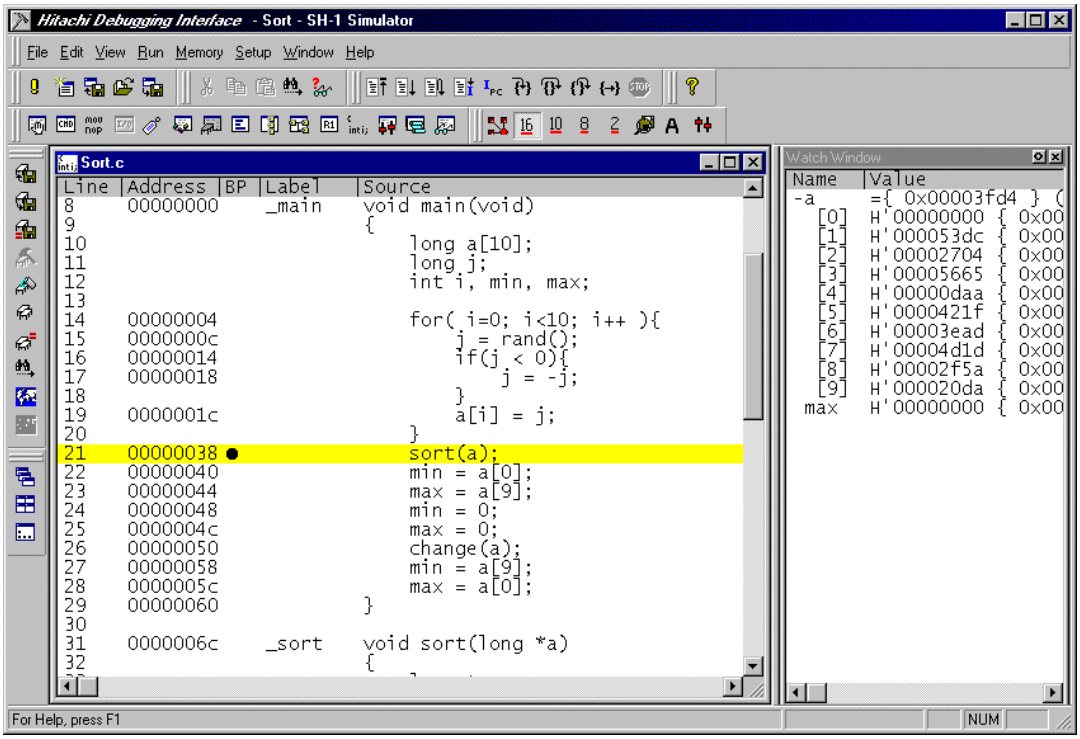

図 2.31　Source **ウィンドウ（ステップ実行）**

なお、Watch ウィンドウのタイトルバーにある **[◇]** ボタンをクリックすると、Watch ウィンドウは**図** 2.31のような位置へ自動的に配置します。

(1)　[Step In]の実行

[Step In]はコール関数の中に入り、コール関数の先頭のステートメントで停止します。

- sort 関数の中に入るために、**[Run]**メニューから **[Step In]**を選択するか、またはツールバーの **[Step In]**ボタンをクリックしてください。

図 2.32　Step In **ボタン**

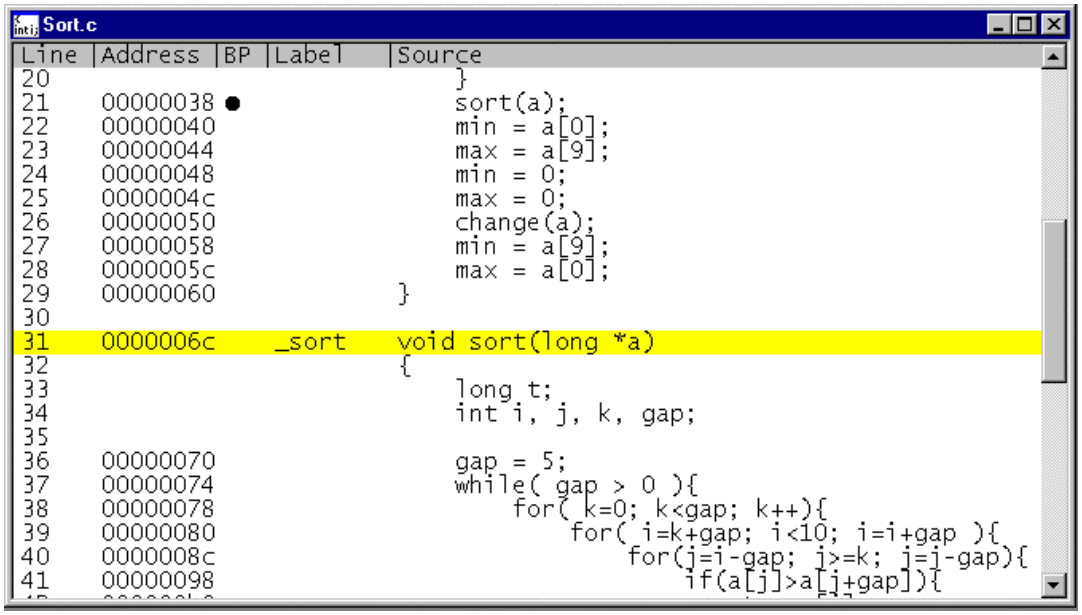

図 2.33 Source **ウィンドウ**（Step In）

- Source ウィンドウの強調表示が、sort 関数の先頭のステートメントに移動します。

(2) [Step Out]の実行

[Step Out]はコール関数の中から抜け出し、コール元プログラムの中の次のステートメントで停止します。

- sort 関数の中から抜け出すために、**[Run]**メニューから **[Step Out]**を選択するか、またはツールバーの **[Step Out]**ボタンをクリックしてください。

図 2.34 Step Out **ボタン**

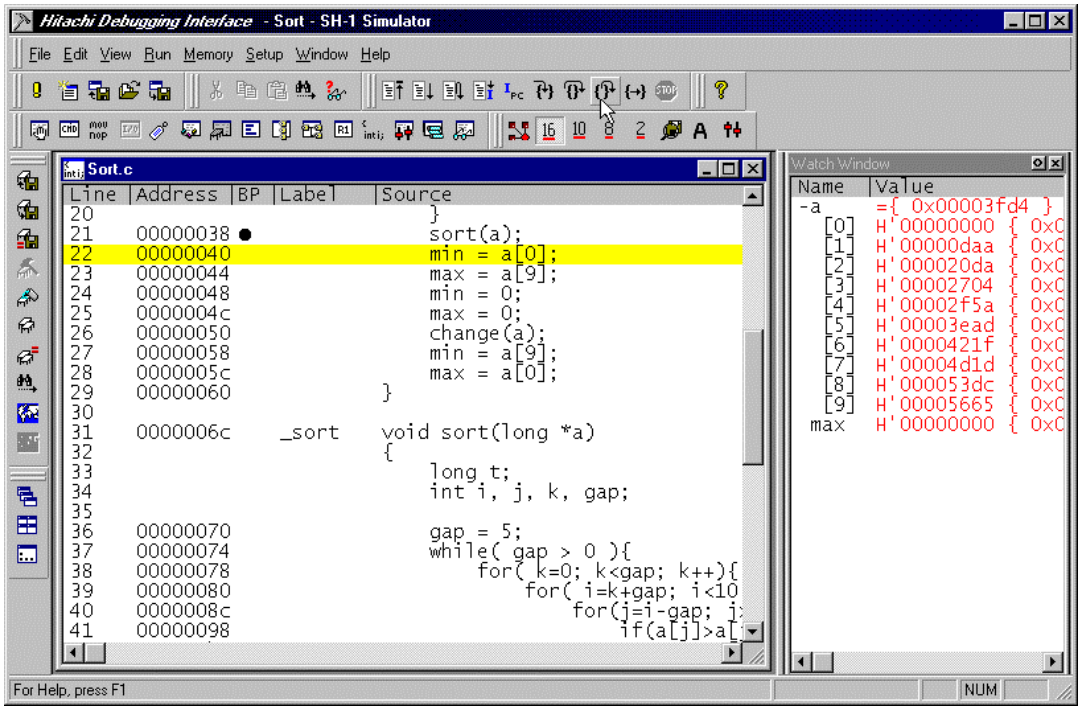

図 2.35 Source **ウィンドウ**（Step Out）

- Watch ウィンドウに表示した変数 a のデータを昇順にソートします。Watch ウィンドウでは、変更した変数の値を赤字で表示します。

- 次に**[Step In]**により、２ステップ実行してください。

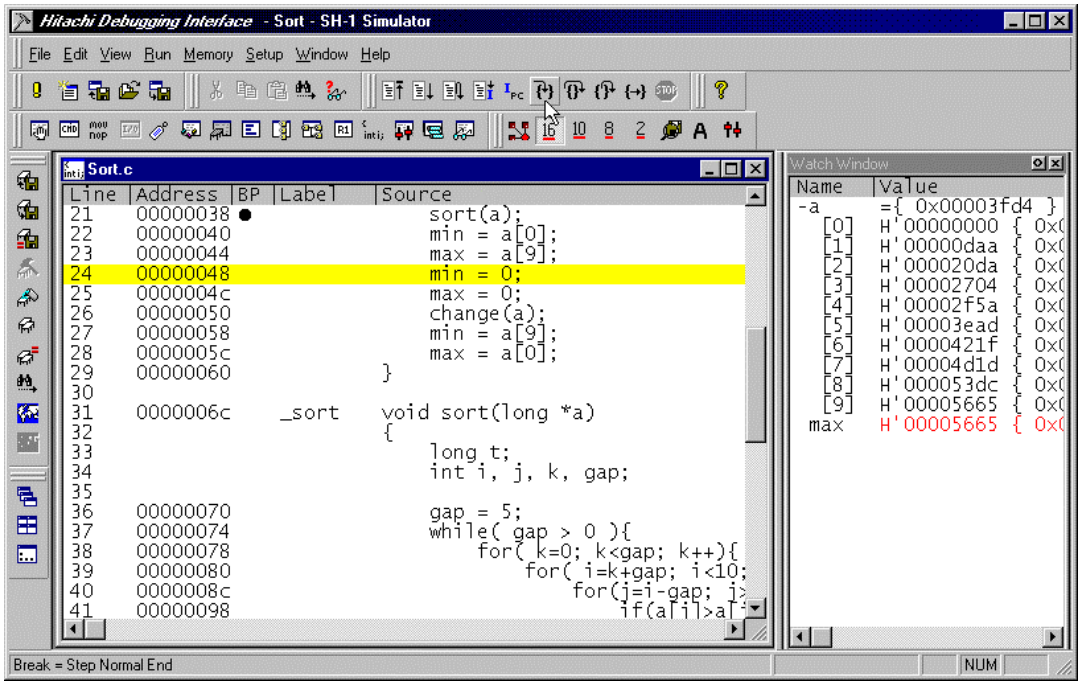

図 2.36 Source **ウィンドウ**（Step Out→Step In）

- Watch ウィンドウに表示した max を、データの最大値に変更します。

(3) [Step Over]の実行

[Step Over]は関数コールを１ステップとして実行して、メインプログラムの中の次のステートメントで停止します。

- **[Step Over]**をデモンストレーションするために、change 関数ステートメントまで２ステップ実行してください。

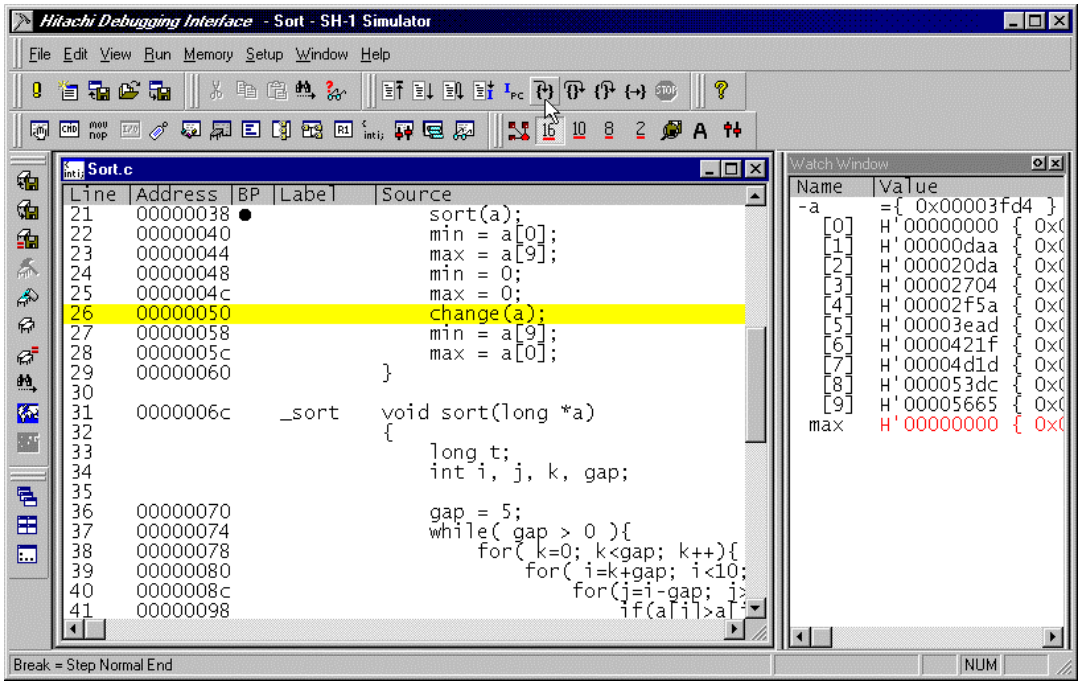

図 2.37　Source **ウィンドウ**（Step Over **実行前**）

change 関数中のステートメントを一度にステップ実行するために、**[Run]**メニューから**[Step Over]**を選択するか、またはツールバーの **[Step Over]**ボタンをクリックしてください。

図 2.38　Step Over **ボタン**

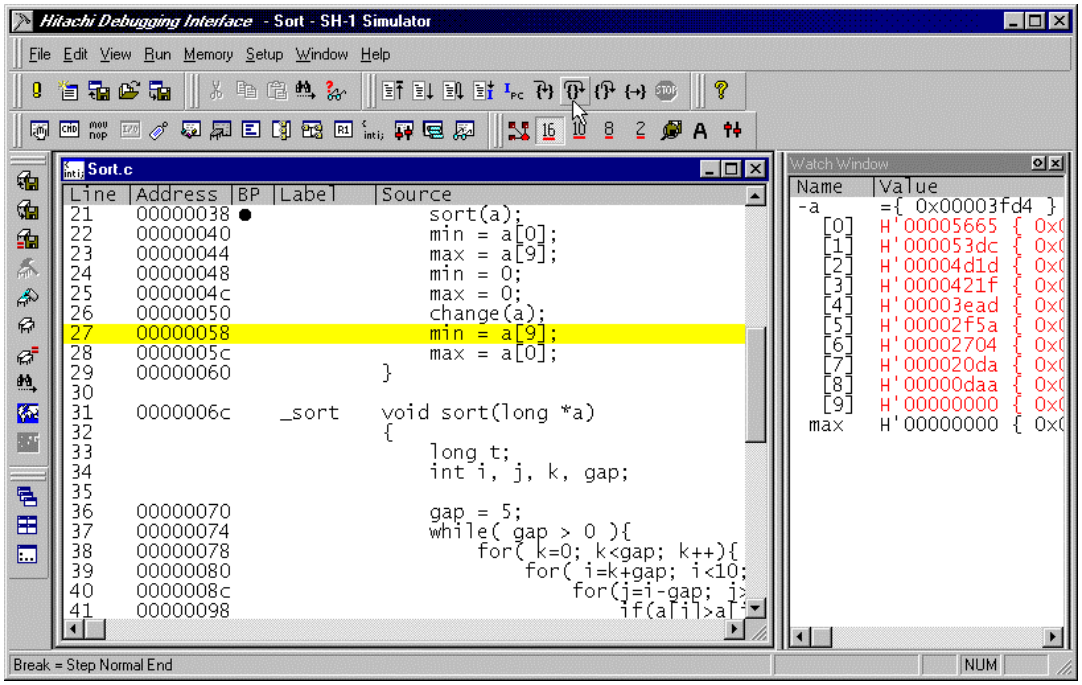

図 2.39 Source **ウィンドウ**（Step Over）

change 関数の最後のステートメントを実行すると、Watch ウィンドウに表示した変数 a のデータを降順に変更します。

2.1.18 ローカル変数の表示

Locals ウィンドウを使って関数内のローカル変数を表示させることができます。例として、main 関数のローカル変数を調べます。この関数は、５つのローカル変数 a、j、i、min、max を宣言しています。

- **[View]**メニューから **[Locals]**を選択することによって Locals ウィンドウを開いてください。
- ローカル変数が存在しない場合、Locals ウィンドウは何も表示しません。
- **[Run]**メニューから **[Step In]**を選択して、もう１ステップ実行してください。
- Locals ウィンドウは、ローカル変数とその値を表示します。

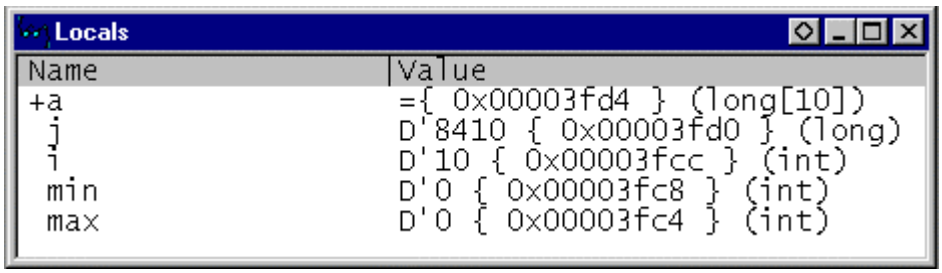

図 2.40 Locals **ウィンドウ**

- Locals ウィンドウの配列 a の前にあるシンボル＋をダブルクリックし、配列 a の構成要素を表示させてください。
- sort 関数実行前と実行後の配列 a の要素を参照し、ランダムデータを昇順、降順にソートしたことを確認してください。

2.1.19 パフォーマンス・アナリシスの確認

パフォーマンス・アナリシスの測定結果を Performance Analysis ウィンドウで確認することができます。

- **[View]**メニューから **[Performance Analysis]**を選択してください。

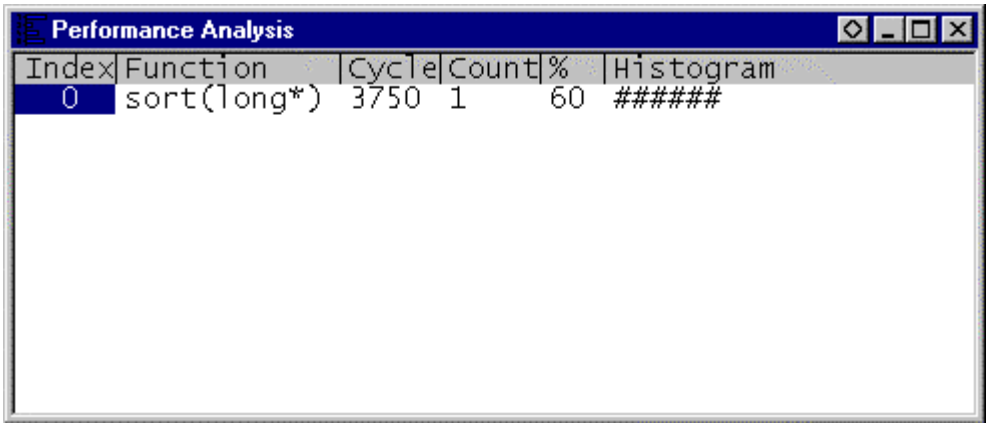

図 2.41 Performance Analysis **ウィンドウ（確認）**

Performance Analysis ウィンドウの **[Cycle]**には、sort 関数の累計実行サイクルを、**[%]**にはプログラム全体の実行サイクル数に占める sort 関数の実行サイクル数の割合を表示します。

Performance Analysis ウィンドウによって、パフォーマンス・アナリシス測定の許可または禁止、新たにパフォーマンス・アナリシスを測定する関数の設定、および削除ができます。

- Performance Analysis ウィンドウを閉じてください。

2.1.20 セッションの保存

終了する前に、今回のデバッグセッションを保存すれば、次回のデバッグセッションで同一の状態からデバッグを再開することができます。

- **[File]**メニューから **[Save Session]**を選択してください。
- **[File]**メニューから **[Exit]**を選択して、HDI を終了させてください。

SuperH™ RISC engine High-performance Embedded Workshop, Debugging Interface
チュートリアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

ADJ-702-306