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

- 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、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。

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

 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

- 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。
- 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。
- 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。
- 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。



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



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
- SuperHTM 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

第1章	章 HEW チュートリアル	
1.1	HEW の起動	1
1.2	プロジェクトの作成	
	1.2.1 CPUの選択	
	1.2.2 CPU 固有のオプションの設定	5
	1.2.3 生成ファイルの設定	7
	1.2.4 標準ライブラリの設定	9
	1.2.5 スタック領域の設定	
	1.2.6 ベクタの設定	
	1.2.7 生成ファイル名の変更	
1.2	1.2.8 設定唯認	
1.5		
	1.3.2 ノロンエクトへのファイル追加と削除 1.2.2 プロジェクトで使うスファイルの種類	
	1.3.5 フロシェットで使えるファイルの種類 1.3.4 ロークスペースウィンドウのカスタマイブ	
14	1.5.4 ノーノスマースティントラのカスノマース	
1.1	1/1 ビルドの機能	
	1.4.1 C/VTの機能	
	1.4.3 コンフィグレーションのカスタマイズ	
	1.4.4 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	
1.5	HDI インタフェース	
	1.5.1 HDI との連動	
	1.5.2 Demonstration プロジェクト	
1.6	HEW の終了	
第2章	章 HDI チュートリアル	
2.1	HDI を使ったデバッグ	
	2.1.1 サンプルプログラム	
	2.1.2 HDI の実行	
	2.1.3 ターゲットの 選択	
	2.1.4 メモリリソースの確保	
	2.1.5 チュートリアルブログラムのダウンロード	
	2.1.6 ソー人ノロクフムの表示	
	2.1./ rC ノレーク小1 ノトの設定 2.1.8 トレーフ結却取得冬姓の設定	
	2.1.0 I レーヘ用報報時ボロの政定 210 パフォーマンス・アナリシスの設定	
	2.1.10 スタックポインタの設定	

2.1.11	プログラムの実行	47
2.1.12	トレースバッファの使い方	49
2.1.13	トレースサーチの実行	
2.1.14	ブレークポイントの確認	
2.1.15	メモリ内容の確認	51
2.1.16	変数の参照	
2.1.17	プログラムのステップ実行	54
2.1.18	ローカル変数の表示	60
2.1.19	パフォーマンス・アナリシスの確認	61
2.1.20	セッションの保存	61

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。

VICTOR NATIONAL STREET FOR STREET STR	 Accessories Hitachi Call Walker Hitachi Call Walker Hitachi Debugging Interface [SuperH 3 Hitachi Debugging Interface Help [Sup Hitachi Debugging Interface Help [Sup Hitachi Debugging Interface Help [Sup Hitachi Embedded Workshop Hitachi Embedded Workshop Read N Hitachi Embedded Workshop Adobe Acrobat 4.0 Hitachi Embedded Workshop Startup Startup 	Series] berH Series] 4e lish(00 09)
💈 🜒 Sh <u>u</u> t Down		
Start		

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。

Welcome!			? ×
Options:-			
<u>~</u>	© Create a new project workspace		Cancel
	O Open a recent project workspace:	7	Administration
200	O <u>B</u> rowse to another project workspace		

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。

New Project Workspace	? ×
Name:	ОК
tutorial	
-	Cancel
Directory:	
c:\tutorial	<u>B</u> rowse
CDLI familur	
SuperH RISC engine	
Tool chain:	
Hitachi SuperH Standard 🔹 💌	
Project type:	
Application	
Experimentation	

図 1.3 New Project Workspace ダイアログボックス

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。)

表 1.1 プロジェクトタイプ

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。

New Project -Step 2	×
(III)	Specify global options.
	Endian: Big
	FPU; Mix
	Round:
	Denormalized number allowed as a result
A AND	Position independent code (PIC)
. Manual and a statement	Treat double as float
	Use try, throw and catch of C++
	Enable/disable runtime type information
< Back	Next > Finish Cancel

図 1.5 プロジェクトジェネレータの CPU 固有オブション設定画面(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。

New Project -Step 3	×
A CONTRACTOR OF	What kind of initialization routine would you like to create? Use I/O Library Number of I/O Streams: 20 * 20 *
< Back	Next > Finish Cancel

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。

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	tutorial.c
	C++ source file	tutorial.cpp
	None	-
		(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	hwsetup.src
	C/C++ source file	hwsetup.c (または、cpp)
		(main 関数ファイルと同じ拡張子)

表 1.2 生成ファイル画面の設定内容と生成ファイル

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。

New Project -Step 4		×
A A A A A A A A A A A A A A A A A A A	Library : Variable forms in the second string string (EC++) : Performs string	
< Back	Next > Finish Cancel	

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。

New Project -Step 5	×
ET.	What are the stack settings?
	Stack Pointer Address: (power-on reset) FYFFFFFF Stack Size: H'100
< Back	Next > Finish Cancel

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。

New Project -Step 6	×
A CONTRACTOR OF	What supporting files would you like to create? Vector Definition Files Vector Handlers: Handler Vector PowerON_Reset_PC 0 Power On F Manual_Reset_PC 2 Manual Re
< Back	Next > Finish Cancel

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 プ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 extern void INT_Illegal_code(void);	> >	#pragma interrupt UserIntProg4 に変更。 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(アセンプリモジュール名)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 TL .global	B miss/invalid (load) _INT_TLBMiss_Load	>	.global	_UserIntProg0	40 に変更。
intprg.src					
;H'040 TLB mi _INT_TLBMis:	ss/invalid (load) s_Load	> 봄	削除し、_U をプロジェ	JserIntProg040 を言 クトに追加。	記述したファイル
vecttbl.src ;H'040 TL .data.1 _1	B miss/invalid (load) 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。

New Project -Step 7					
A HA	The follow generated	ing sou	urce files will be		
	initsct dbsct sbrk iodefine intprg vecttbl vect resetprg tutorial sbrk stacksct	C src c h c c h c c h src	Initialize of RAM Dat Setting of B,R Section Program of sbrk Definition of I/O Reg Interrupt Program Initialize of Vector T Definition of Vector Reset Program Main Program Header file of sbrk f Setting of Stack are		
	•		<u> </u>		
< Back	Next >		Finish	Cancel	

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。

Project PROJECT GENERATOR PROJECT NAME : PROJECT DIRECTORY : CPU SERIES : CPU TYPE : TOOLCHAIN NAME : TOOLCHAIN VERSION : GENERATION FILES : c:\tutorial\tutorial\initsct.c Initialize of RAM Data c:\tutorial\tutorial\bsrk.c Setting of B,R Section c:\tutorial\tutorial\sbrk.c Program of sbrk c:\tutorial\tutorial\iodefine.h Definition of I/O Register c:\tutorial\tutorial\intprg.c Interrupt Program	tutorial c:\tutorial SH-1 SH7020 Hitachi SuperH RISC engine Standar 6.0.0
•	
Click OK to generate the project	or Cancel to abort.
🔽 Generate Readme.txt as a si	ummary file in the project directory
	OK Cancel

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。

🛞 tutorial - Hitachi Embedded Workshop					_ 🗆 ×
<u>F</u> ile <u>E</u> dit <u>Project Options</u> <u>Build</u> <u>T</u> ools <u>W</u> indow <u>H</u> elp	p				
	* 🕮 🖽 🗡	Debug			
		ລ			
Interial Image: Second secon					
Projects Navigat					
C:\tutorial\tutorial\stacksct.src Phase SH Assembler finished					
Phase OptLinker starting Phase OptLinker finished					
Build Finished O Errors, O Warnings					_
Build Find in Files Version Control					<u> </u>
For Help, press F1			_	INS	

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)

ł

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。

Add File(s)		? ×
Look <u>i</u> n:	🔄 tutorial 💽 🖻	8-6- 8-6- 8-6-
Debug Release dbsct.src initsct.c intprg.c iodefine.h newprog.c resetprg.src	❷ sbrk.c 폐 sbrk.h 폐 stacksct.src ❷ tutorial.c 폐 vect.h ❷ vectbl.c	
File <u>n</u> ame:	newprog.c	Add
Files of type:	Project Files	Cancel
	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。

🛞 tutorial - Hitachi Embedded Workshop - [newprog	c] _ 🗆 🗶
Eile Edit Project Options Build Tools Window He	alp _ 문 ×
D & ■ Ø Ø X B & M 🙀 {} T 🖬 {	🖻 🛗 👗 Debug 💽 💽 🗐 🗐
💽 🕹 🕯 関 🕞 🕒	÷ ÷ .
Image: Second	am (void)
Projects Navigat	rog.c
Z:\tutorial\tutorial\stacksct.src Phase SH Assembler finished	×
Phase OptLinker starting Phase OptLinker finished Build Finished O Errors, O Warnings	×
Build (Find in Files Version Control /	
For Help, press F1	Read-write 1 3/3 INS //

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	プロジェクトフラ	▽イルの除外 」	を参照して	ください。
--------	----------	----------	-------	-------

Remove Project Files		? ×
<u>P</u> roject files:		ОК
dbsct.src initsct.c initsrc.c	[C:\tutorial\tutorial] [C:\tutorial\tutorial] [C:\tutorial\tutorial]	Cancel
newprog.c resetprg.src	[C:\tutorial\tutorial] [C:\tutorial\tutorial] [C:\tutorial\tutorial]	<u>R</u> emove
sbrk.c stacksct.src tutorial.c vecttbl.c	[C:\tutorial/tutorial] [C:\tutorial\tutorial] [C:\tutorial\tutorial] [C:\tutorial\tutorial]	Remove <u>A</u> ll
4		Þ

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。

File Extensions			? ×
Extension	Group	•	ок
]*.abs	Absolute file		
∎*.inc	Assembly include file		Cancel
≣*.lis	Assembly list file		
🖹 *.src	Assembly source file		Add
🗋 *.bin	Binary file		<u>A</u> uu
≣ *.h	C header file		
≣*.lst	C list file		<u>R</u> emove
≣*.c	C source file		
🖹 *.hpp	C++ header file		<u>O</u> pen with
≣i*.lpp	C++ list file		
≣*.cpp	C++ source file		
🗋 *.cpu	CPU information file		
I≣*.exp	Expanded assembly source file		
≣*.hex	Hex file	_	
]*.lib	Librarv file	▼	
	• _		

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)。

Define File Extension	? ×
File extension:	OK Cancel
File group:	
Assembly source file	
O <u>N</u> ew group:	
☑ _ext based file	

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

File Extensions		? ×
File Extensions Extension *.abs *.inc *.lis *.src *.bin *.lst *.c *.lpp *.lpp *.cpp *.cpu *.cpu *.cpu	Group Absolute file Assembly include file Assembly list file Assembly source file Binary file C header file C list file C++ header file C++ header file C++ list file C++ source file	OK Cancel Add Remove Open with
☐*.cpu ■*.exp ■*.hex	CPU information file Expanded assembly source file Hex file	

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。

Configure View	? ×
Show <u>d</u> ependencies under each file	ОК
Show <u>s</u> tandard library includes	Cancel
Show file paths	
Show file groups in separate folders	

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。

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。

SH C/C++ Compiler Options(Debug)		? ×
C source file C source file sbrk.c tutorial.c vectbl.c intpr.c newprog.c Default Options C++ source file Default Options	Source Object List Optimize Other CPU Show entries for : Include file directories Add Insert Remove Move up Move gown	
	OK Cance	el

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。

Build Configurations	? ×
Build configurations:	OK
Debug Release	Cancel
	<u>A</u> dd
	<u>R</u> emove
Current configuration:	
Debug	

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。

🛞 tutorial - Hitachi Embedded Workshop - [tuto	orial.c]
Eile Edit Project Options Build Tools Winds	ow <u>H</u> elp
ÍDCI 🛛 🖉 🎒 👗 🖻 🖻 🙌 🐂 {} T 🕻	🚨 🕼 👑 🚣 🛆 Debug 💽 💽 🖳 🗐
Image: start start start #ifdef Image: start s	<pre>cplusplus</pre>
Projects Navigat	×
C:\tutorial\tutorial\stacksct.src	A
Phase SH Assembler finished Phase OptLinker starting Phase OptLinker finished Build Finished O Errors, O Warnings Build /Find in Files / Version Control /	
For Help, press F1	Read-write 27 28/35 INS ///

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

For Help, press F1

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。 🔞 tutorial - Hitachi Embedded Workshop - [newprog.c] _ 🗆 × SEILE Edit Project Options Build Tools Window Help _ 8 × 🗅 😅 🖬 🍠 🎒 👗 🖻 🛍 🙌 🙀 🦂 {} T 🛄 🛛 🕸 🛗 🚈 Debug 🔻 🕄 🔉 💷 🗐 T & A 🐂 🛛 🗗 🗗 🗳 🛓 🍭 - × void NewProgram(void) 🖻 🗟 tutorial -🗄 🚯 tutorial 🗄 🔄 Assembly source error= dbsct.src resetpra.src - 🗐 stacksct.src 🚊 🔄 C source file initsct.c intprq.c newprog.c 🖹 sbrk.c 🔳 tutorial.c 🖹 vecttbl.c 🗄 🔄 Dependencies 🖹 sbrk.h 🖹 vecth Ŧ • Þ 🔄 Projects 🛛 🔍 Navigat. newprog.c Phase SH C/C++ Compiler starting ٠ C:\tutorial\tutorial\newprog.c C:\tutorial\tutorial\newprog.c 3: C2225 (E) Undeclared name "error" C:\tutorial\tutorial\newprog.c 4: C2500 (E) Illegal token "}" C:\tutorial\tutorial\newprog.c 4: C2500 (E) Illegal token "}" Phase SH C/C++ Compiler finished Phase SH Assembler starting Nothing to do - skipping Build (Find in Files) Version Control /

図 1.27 ビルド時のエラー修正

Read-write 1

3/4

INS

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

-	
=	
	100

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。

Tools Customize	? ×
Toolbars Commands Menu Debugger Log Help	
	1
HDI location (V4.0 or greater):	
C:\Hew\HDI5\SIM\SH\Hdi.exe	Browse
Session file:	
\$(CONFIGDIR)\\$(PROJECTNAME).hds	Bro <u>w</u> se
Download module:	
\$(CONFIGDIR)\\$(PROJECTNAME).abs	B <u>r</u> owse

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。

$\gg I$	litachi .	Debugg	ging Interface-1	utorial - SH	-1 Simulator	_ 🗆 ×
Eil	e <u>E</u> dit	⊻iew <u>P</u>	<u>}</u> un <u>M</u> emory <u>S</u> e	tup <u>W</u> indow	Help	
<u> </u>	1 🖬	🗳 🖬	X 🖻 🖻 💾	• 🇞 📗 🗄 🛛	1 ፤1 ፤፤ ·. የ የ የ (+ 🐵 📗 የ	
0	CHD MOU DOP	170 Ø	Ş 🗐 🗉 🖪 🖗	🖁 🖪 🕻 inti; 🐺	🗟 🖄 🔢 10 8 2 🔊 A 🕂	
0 11 10 11 10 10 10 10 10 10 10 10 10 10		Line 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 •	ial.c Address BP 00001094 00001098 0000109a	Jelint; #	Source //#include <ios> //int ios_base::Init::init_cnt; #endif #ifdefcplusplus extern "C" { #endif void abort(void); #ifdefcplusplus } #endif void main(void) { /* User Program */ } void abort(void) { /* User Program */ }</ios>	
		51				
Fort	Help, pre	ess F1				

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。

System Configuration	×
CPU SH-1	•
<u>B</u> it size 32	S <u>v</u> stem Call Address ⊠ Enable H'00000000
<u>E</u> ndian Big Endian 🔽	Execution Mode
Memory Map	● <u>S</u> top ● Con <u>t</u> inue
00000000 FFFFFFF EXT 16 1	Round Mode © Round to <u>n</u> earest
	⊙ Round to <u>z</u> ero
Add Modify Delete	Help OK <u>C</u> ancel

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。



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。



HDI for SH series simulator 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。

Select Session		×
• Create a <u>n</u> ew session on:		OK
SH-1 Simulator		Exit
© Previous session file:		
	T	BIowse

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。

Memory Map		×
System Configuration	Memory map	
CPU:SH-1 Bit Size:32 Exec Mode:STOP	00000000 FFFFFFF EXT 16 1	
System memory resource		
Add Modify	Delete Reset Help Clos	e
図 2.4	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。

System Memory Re	source Modify	×
Start address	H'0000000	ОК
End address	H'00003FFF	<u>C</u> ancel
C Read	Write © Read/Write	<u>H</u> elp
·		

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'0000000、[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。

Load Program		×
Offset: H'0 File name:	Verify	<u>O</u> pen Cancel
C:\hew\HDI5_SH\Tutorial\Sort.abs	•	B <u>r</u> owse
Load only debugging information Load stack information file(SNI file)		

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。

Open					? ×
Look in: 🔁	Tutorial	-	<u></u>	<u>e</u>	
Sort.abs					
File <u>n</u> ame:	Sort.abs				pen
Files of tune:	ELE/DW/ABE Files (* abs)		Ţ	C	ancel
, not of grps.					/

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。



• [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。

Open					? ×
Look jn: 🔁	Tutorial	-	<u></u>		
Sort.c					
					- 1
					- 1
I					
File <u>n</u> ame:	Sort.c			<u>О</u> ре	en 🚬
Files of <u>type</u> :	C Files (*.c;*.inl)		•	Can	cel
					///

図 2.9 Open ダイアログボックス ([Source...])

inti; Sort	.C			
Line	Address	BP Label	Source	
8	00000000	_main	void main(void)	
9			{]]]]]]]]]]]]]]]]]]]	
11			long i:	
12			int i. min. max:	
13			ing if milling makely	
14	00000004		for(_i=0; i<10; i++){	
15	0000000€		j_=_rand();	
16	00000014		1†(] < 0){	
18	00000018		ן = −ן; נ	
19	0000001c		a[i] = i	
20	00000010		α[·] =], }	
21	00000038		<pre>sort(a);</pre>	
22	00000040		min = a[0];	
23	00000044		max = a[9];	
24	00000048		m = 0;	
26	00000040		change(a):	
27	00000058		min = a[9];	
28	0000005c		$\max = a[0];$	
29	00000060		}	
10	0000006-		void cont(long *=)	
1 32	00000066	_Sort	vola soriciong "a) S	_
L34				Ľ

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。

inti; Sort.	c		_ 🗆 ×
Line 8	Address BP Label	Source void main(void)	_
9 10 11 12 13		{ long a[10]; long j; int i, min, max;	
14 15 16 17	00000004 0000000⊂ 00000014 00000018	for(i=0; i<10; i++){ j = rand(); if(j < 0){ j = -j;	
18 19 20	0000001c	} a[i] = j;	_
21 22 23 24 25 26 27 28 29 30	00000038 00000040 00000044 00000042 0000004 00000050 00000050 00000058 00000055 00000055 00000055	<pre>sort(a); min = a[0]; max = a[9]; min = 0; max = 0; change(a); min = a[9]; max = a[0]; }</pre>	
31 32	0000006c _sort	void sort(long *a) {	•

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。

Trace Acquisition	×
Trace start/Stop O Disable © Enable	ОК
Instruction type Instruction Subroutine	<u>C</u> ancel
Trace buffer full handling Continue Break	<u>H</u> elp
Trace capacity 1024 records 4096 records 16384 records 32768 records 	

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。

Performance Option		×
Function Name	sort(long*)	
<u>H</u> elp	ОК	<u>C</u> ancel

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。

Performance Analysis	◇ _ □ ×
Index Function Cycle Count %	Histogram
sort(long*) U U U	<u>A</u> dd Range <u>E</u> dit Range
	<u>R</u> eset Counts/Times
	Enable Analysis
	Delete Range
	Delete All Ranges

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。

Value: 4000 Set As: Whole Register	Register - R15	×
4000 Set As: Whole Register Cancel	<u>V</u> alue:	
Set As: Whole Register	4000	
Whole Register	<u>S</u> et As:	
	Whole Register	

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。



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。

>> H	itachi Di	ebugging Interface	- Sort - SH-1 Simulator	
<u>F</u> ile	<u>E</u> dit ⊻i	ew <u>R</u> un <u>M</u> emory <u>S</u> e	tup <u>W</u> indow <u>H</u> elp	
0	12 🖬	🗳 🖬 🛛 X 🖻	🖀 🙏 📗 티 티 티 타 다. 근) ① 단 (-) 💷 🛛 💡	
	CHD MOU			
	nop			
-	_{inti;} Sor	t.c		
1	8	00000000	_main void main(void)	
1	9		{	
*	$10 \\ 11$		long a[10]; long i:	
Â	12		int [°] i, min, max;	
ø	14	00000004	for(i=0: i<10: i++){	
æ	15	0000000	j = rand(0);	
<u>#1</u> ,	17	00000014	i = -i:	
6 52	18		}	
	19 20	0000001c	a[1] =]; }	
	21	00000038 •	sort(a);	
	22	00000040	min = a[O]; max = a[9]·	
	24	00000048	min = 0;	
	25	0000004c	max = 0; change(a):	
	27	00000058	min = a[9];	
	28	0000005c 00000060	ma× = a[0];	-
Break	= PC Brea	akpoint		

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。

👬 System Status		◇ _ 🗆 ×
Item Connected to CPU	Status SH-1 Simulator SH-1	<u>•</u>
Run Status Break Cause	Ready PC Breakpoint	0
Single Step Count Execute From Exec Instructions Cycles	1 Pipeline Reset 600 1472	0 9 9
Session) Platform / Memory / Ever	its_/	

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。 ブレークの原因は PCブレーク パイプラインリセットからの実行 GOコマンドでの実行命令数は600命令 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。

R1 Registers	0 <u> </u>
Register	Value 🔺
R8	00000000 🦳
R9	00000000
R10	0000000
R11	0000000
R12	0000000
R13	00000000 🔲
R14	00000000
R15	00003FC4
PC .	00000038
+ SR	1111T
GBR	00000000
VBR	00000000
MACH	00000000
MACL	00000000
PR	00000012 🔽

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。

🔄 Trace -	661 records (no f	ilter)					0 _ 🗆 ×
PTR	CYCLE	ADDR	PIPELINE	IN	STRUCTION	Source	
-00660	0000000000			:PIPELI	NE RESET		1000
-00659	0000000002	00000000	FFDE>MM	:STS.L	PR, @-R15 00003FFC	:void main(void)	
-00658	0000000004	00000002	fD>E>	: ADD	#C8, R15 R15<-0000)	
-00657	000000006	00000004	FFD>E>	:MOV	#00, R3 R3<-000000) for(i=0; i<10;	i++){
-00656	8000000000	00000006	f>D>EMM	:MOV.L	R3, @(00000008, R15		
-00655	0000000009	80000008	FFDE>	:BRA	00000030 PC<-00000)	
-00654	000000013	A0000000	f->D>E	:NOP			
-00653	000000014	00000030	FFDE>	:MOV	#0A, R2 R2<-000000)	
-00652	0000000016	00000032	fD>EMMW	:MOV.L	@(00000008, R15), R		
-00651	. 0000000019	00000034	FFD< <e></e>	:CMP/GE	R2, R1 T<-(0)		
-00650	0000000021	00000036	f< <d>E></d>	:BF	0000000C T(0), PC<		
-00649	000000023	00000038	FFD	:		sort(a);	
-00648	000000023	AE000000	f	:			
-00647	0000000025	0000000C	FFDE>MMW	:MOV.L	@(00000058, PC), R1	. j = rand();	
-00646	0000000029	0000000E	fD>< <e< td=""><td>:JSR</td><td>@R1 PC<-0000018C</td><td></td><td></td></e<>	:JSR	@R1 PC<-0000018C		
-00645	000000032	00000010	FF<<-D>E	:NOP			•
•							• //

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。

Trace Search	×
Item	
O PTR	
O Cycle	Cancel
C Address	
Instruction	<u>H</u> elp
Value BRA	

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。

l	🚍 Trace -	661 records (no fi	ilter)					0 _ 🗆 ×
Γ	PTR	CYCLE	ADDR	PIPELINE	INS	STRUCTION	Source	
I	-00660	0000000000			:PIPELI	NE RESET		
	-00659	0000000002	00000000	FFDE>MM	:STS.L	PR. @-R15 00003FFC	:void main(void)	
	-00658	0000000004	00000002	fD>E>	:ADD	#C8, R15 R15<-0000)	
	-00657	0000000006	00000004	FFD>E>	:MOV	#00, R3 R3<-000000) for(i=0; i<10;	i++){
	-00656	8000000008	00000006	f>D>EMM	:MOV.L	R3. @(00000008, R15	. , ,	, .
	-00655	000000009	80000008	FFDE>	:BRA	00000030 PC<-00000	0	
E	-00654	0000000013	A0000000	f->D>E	:NOP			
	-00653	0000000014	00000030	FFDE>	:MOV	#0A, R2 R2<-000000)	
	-00652	0000000016	00000032	fD>EMMW	:MOV.L	@(00000008, R15), R		
	-00651	0000000019	00000034	FFD< <e></e>	:CMP/GE	R2, R1 T<-(0)		
	-00650	0000000021	00000036	f< <d>E></d>	:BF	0000000C T(0), PC<		
	-00649	0000000023	00000038	FFD	:		sort(a);	
	-00648	0000000023	0000003A	f	:			
	-00647	0000000025	0000000C	FFDE>MMW	:MOV.L	@(00000058. PC). R1	i = rand():	
	-00646	0000000029	0000000E	fD>< <e< td=""><td>:JSR</td><td>@R1 PC<-0000018C</td><td>, , , , , , , , , , , , , , , , , , ,</td><td></td></e<>	:JSR	@R1 PC<-0000018C	, , , , , , , , , , , , , , , , , , ,	
	-00645	000000032	00000010	FF<<-D>E	:NOP			-
L	•							► //

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。

🕑 Breakpoints			0 _ 🗆 ×
Enable File/Line	Symbol Address	; Type	
sort.c/21	0000003	38 Rh	

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。

Open Memory Window	×
<u>A</u> ddress:	
main	
Eormat:	Lancei
Word	

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。

🛷 Word Mem	orymai	n		◇ _ □	×
Address	Data		Value		
00000000	4F22	7FC8	20258	32712	
00000004	E300	1F32	-7424	7986	
80000008	A012	0009	-24558	9	
0000000C	D116	410B	-12010	16651	
00000010	0009	1F03	9	7939	
00000014	4011	8901	16401	-30463	
00000018	600B	1F03	24587	7939	
0000001C	53F2	4308	21490	17160	
00000020	62F3	7210	25331	29200	
00000024	332C	51F3	13100	20979	
00000028	2312	53F2	8978	21490	-

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。

😽 Watch	Window	○ _ □ ×	
Name	Value		
+a	={ 0x00003fd4	}	(long[10])

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。

Add Watch	×
C Address Solution	Cancel
max	

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。

& Watch Wir	ndow 🦷) _ 🗆 ×
Name	Value	
+a Laax	={ 0x00003fd4 } (long[10] H'00000000 { 0x00003fc4 }) (int)

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。

& Watch Window	
Name	Value
-a [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] ma×	={ 0x00003fd4 } (long[10]) H'00000000 { 0x00003fd4 } (long) H'000053dc { 0x00003fd8 } (long) H'00005665 { 0x00003fdc } (long) H'00000daa { 0x00003fe0 } (long) H'0000421f { 0x00003fe4 } (long) H'00003ead { 0x00003fe8 } (long) H'00004d1d { 0x00003ff0 } (long) H'00002f5a { 0x00003ff4 } (long) H'000020da { 0x00003ff8 } (long) H'000020da { 0x00003ff8 } (long) H'0000000 { 0x00003fc4 } (int)

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。

<i>]</i> ≥ h	litachi De	ebugging Interface	- Sort - SH-1	Simulator				_ 🗆 ×
<u> </u>	<u>E</u> dit ⊻ie	ew <u>R</u> un <u>M</u> emory <u>S</u> e	etup <u>W</u> indow	Help				
0	🎦 🏭	🗳 🖬 🛛 X 🖻	Ca 🛤 🐕	ET EL EL EL I+c 7 7 7 (P (→	8			
10	CHD MOU NOP	🖂 🧳 😺 💭 🖻	👔 🔁 RI	📊 🐺 🖼 🔝 🔢 🌆 🛛 🖉 A	† ‡			
	inti, Sort	.c		1.05	- 🗆 ×	Watch Wind	low	<u>ox</u>
s = 0 + > = = = = = =	Line 8 9 10 11 12 13 14 15 16 17 18	Address BP 00000000 00000004 0000000c 00000014 00000018	Label _main	<pre>Source void main(void) { long a[10]; long j; int i, min, max; for(i=0; i<10; i++){ j = rand(); if(j < 0){</pre>	-	ra [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]	={ 0x0003f H 0000000 H 000053dc H 00002704 H 00002704 H 00004aa H 0000421f H 0000421f H 0000421f H 0000421f H 00002f5a H 00002f5a	d4 } ({ 0×00 { 0×00
	19 20	0000001c		ā[i] = j; }		ma×	H.00000000	{ 0×00
	21 22 23 24 25 26 27 28 29 30 31	0000038 ● 0000044 0000044 0000042 0000050 0000055 00000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 000055 00	sort	<pre>sort(a); min = a[0]; max = a[9]; min = 0; max = 0; change(a); min = a[9]; max = a[0]; } yoid sort(long *a)</pre>				
	32 •		_3010		•	•		F
For H	elp, press F	7					NUM	1

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

inti; Sort	.c			_ 🗆 ×
Line 20 21 22 23 24	Address BP 00000038 ● 00000040 00000044 00000048	Labe]	<pre>Source</pre>	<u> </u>
25 26 27 28 29 30	0000004 00000050 00000058 0000005 0000005 00000060		<pre>max = 0; change(a); min = a[9]; max = a[0]; }</pre>	
31 32 33 34 35	0000060	_sort	<pre>void sort(long *a) { long t; int i, j, k, gap;</pre>	
36 37 38 39 40 41	00000070 00000074 00000078 00000080 0000008c 0000008c		gap = 5; while(gap > 0){ for(k=0; k <gap; k++){<br="">for(i=k+gap; i<10; i=i+gap) for(j=i-gap; j>=k; j=j-ga if(a[j]>a[j+gap]){</gap;>){ ap){ ▼

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。



>> H	litachi De	bugging Interface	e - Sort - SH-1 Simula	tor				_ 🗆 ×
Eile	<u>E</u> dit <u>V</u> ie	ew <u>R</u> un <u>M</u> emory .	<u>S</u> etup <u>W</u> indow <u>H</u> elp					
<u>]</u> 0	12 🖬	😂 🖬 🗍 X 🖻	d (2 🤐 🐎 🗍 Et	⊒↓ ⊒↓ ⊒ <mark>† ↓</mark> ₀ ? } ()	Ŷ ((→ ⊕) ¶ Ŷ			
	CHD MOU NOP	🖂 🧳 🖉 🖾 🛛	E 👌 😭 🖻 inti; 🖡) 🗟 🖉 🗍 🔀 🔟 1	0 8 2 🔊 A 👫			
4	inti; Sort	te				Watch Win	idow	<u>ox</u>
	Line	Address B	P Label Sou	unce	•	Name -a	Value ={ 0x00003f	d4 }
	20	00000038	•	sort(a);		[0] [1]	н [°] 00000000 Н'00000daa	{ 0x0 { 0x0
1 2 2 2 2 2 2 2 3 1 1 1 1 1 1 1 1 1 1 1	22 23 24 25 26 27 28 29 30 31 32 33 33 34	0000044 00000048 00000042 00000050 00000050 00000058 0000005c 0000006c	} _sort vo {	<pre>min = a[0]; max = a[9]; min = 0; max = 0; change(a); min = a[9]; max = a[0]; id sort(long *a long t; int i, j, k,</pre>	a)	[2] [2] [4] [5] [6] [7] [8] [9] max	H'000020da H'00002704 H'00002f5a H'00003ead H'0000421f H'000041d H'000053dc H'00005665 H'00000000	2 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
	35 36 37 38 39 40 41 •	00000070 00000074 00000078 0000080 000008c 00000098		gap = 5; while(gap > for(k=0; for(0){ ik <gap; k++){<br="">i=k+gap; i<10 for(j=i-gap; j: if(a[j]>a[</gap;>		(NDM	×
For He	eip, press H						INUM	11.

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。

>> H	litachi Di	ebugging Interface	e - Sort - SH-1	Simulator					_ 🗆 ×
<u>File</u>	<u>E</u> dit ⊻i	ew <u>R</u> un <u>M</u> emory <u>S</u>	<u>S</u> etup <u>W</u> indow	<u>H</u> elp					
0	12 🖬	🗠 🖬 🛛 X 🖻	a (2 触 🐍		Pc 🔁 🔂 🖓 🖓 💮	8			
	CHD MOU		ଅଧି କରି ଲୋ			+			
	- nop			inti; 💵 🗠 👦) (state) (inse	tow	
1	inti; Sor	t.c				. 🗆 ×	Name	Jue	
4	Line	Address B	P Label	Source			-a	={ 0x00003	fd4 }
VIII O	141	00000038	,	sort(a,	;		[TO]	н'ооооооо	{ 0x0
	23	00000040		max = a	[0], [0]:		[1]	H'00000daa	{ 0x(
*	24	00000048		max = 0 min = 0	·,		[2]	H'000020da	{ 0x(
æ	25	0000004c		$ma \times = 0$;			H'00002704	{ 0x0
a	26	00000050		cḥange (<u>a);</u>			H 0000215a	
	27	00000058		min = a	[9];		l tét	H'0000421f	{ Őxd
4	20	00000050		nax = a	.[0];		[7]	H'00004d1d	{ 0x(
# <u>4</u>	30	00000000		J			[8]	H'000053dc	{ 0×0
6 52	31	000006c	_sort	void sort(1	ong *a)		[9]	H'00005665	{ 0×0
	32			{ long t			max	11 00003003	1 0.00
	34			int i.	i. k. gap:				
	35			··· · ,	5,, 5-4,				
	36	00000070		gap_=_5	;				
	3/	00000074		while(gap > 0){ / k_0, k/app, k//	10			
<u></u>	30	00000078		101	for(i=k+dap; k++	10.			
	4õ	0000008c			for(i=i-gap; i<	i>			
	41	00000098			if(alil>	at			
						• //	4		F
Break	= Step No	ormal End						NUM	

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。



Hitachi Debugging Interface - Sort - SH-1 Simulator	
Eile Edit View Bun Memory Setup Window Help	
• 1 = 5 • 6 • 4 • 1 = 1 = 1 = 1 • • • • • • • • • • • •	
R Sort.c	Watch Window OX
Line Address BP Label Source 21 00000038 ● sort(a); 22 00000040 min = a[0]; 23 00000044 max = a[9]; 24 00000048 min = 0; 25 00000044 max = 0; 26 00000050 change(a); 27 0000005c max = a[0]; 29 00000060 } 30 31 0000006c 31 0000006c _sort(long *a) 32 { 33 long t; 34 int i, j, k, gap; 35 gap = 5; 37 0000070 gap = 5; 39 00000074 while(gap > 0){ 40 00000080 for(i=k+gap; i<10; 40 00000080 for(i=k-gap; i) ; 41, 0000098 if(ali)alix	-a ={ 0×00003fd4 } [0] H'00005665 { 0×([1] H'00005dc { 0×([2] H'00004d1d { 0×([3] H'00004d1d { 0×([4] H'00002f5a { 0×([5] H'00002704 { 0×([6] H'000020da { 0×([7] H'0000000a a { 0×([8] H'00000000 { 0×(ma× H'00000000 { 0×(
Break = Step Normal End	

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。

Sery Locals	
Name	Value
+a j i min max	={ 0x00003fd4 } (long[10]) D'8410 { 0x00003fd0 } (long) D'10 { 0x00003fcc } (int) D'0 { 0x00003fc8 } (int) D'0 { 0x00003fc4 } (int)

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。

E Performance Analysis	0 _ 🗆 ×
Index Function Cycle Count % Histogram O sort(long*) 3750 1 60 ######	
J	

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

