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

RENESAS

HEWデバッガ
High-performance Embedded Workshop 2
ユーザーズマニュアル
(Windows®98/ME、Windows NT®4.0、および Windows®2000用)

HS6400EWIW2SJD

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00　2002.05

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。

このマニュアルについて

このマニュアルではHEWおよびデバッガ機能の使い方について述べています。ここでは、デバッグワークスペースを作成してダウンロードモジュールをロードし、プログラムを動作させるデバッグプロセスのあらゆる面について記述しています。HEWの使い方またはHEWの環境のカスタマイズに関する情報は、HEWのユーザーズマニュアルを参照してください。

注意　本マニュアルに記載されている HEW の画面は英語版 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されます。また、本マニュ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。

Microsoft, MS-DOS, Windows, Windows NTは米国Microsoft社の米国およびその他の国における登録商標です。Visual SourceSafe はMicrosoft社の米国およびその他の国における商標です。
IBMはInternational Business Machines Corporationの登録商標です。
その他、記載されている製品名は各社または各団体の商標または登録商標です。

このマニュアルの記号

このマニュアルで使用している記号の意味を説明します。

表 1: **記号一覧**

記号	**意味**
[Menu->Menu Option]	**太字と** '->' **はメニューオプションを示します (例** [File->Save As...])
FILENAME.C	**大文字の名前はファイル名を示します**
"**文字列の入力**"	**下線は入力する文字列を示します (**"" **を省く**)
Key + Key	**キー入力を示します。例えば、**CTRL+N **キーでは** CTRL **キーと** N **キーを同時に押します**
➲ (**「操作方法」マーク**)	**このマークが左端にあるとき、その右の文章は何かの操作方法を示します**

目次

1. 概要

この章ではHEWの基本概念を説明します。Windows®エクストラヘルプに慣れていないユーザのために、次章以降で必要となる情報を提供します。

1.1 ワークスペース、プロジェクト、ファイル

ワードプロセッサでドキュメントを作成、修正できるのと同じように、HEWではワークスペースを作成、修正できます。

ワークスペースはプロジェクトを入れる箱と考えることができます。同じように、プロジェクトはプロジェクトファイルを入れる箱と考えることができます。したがって各ワークスペースにはプロジェクトが一つ以上あり、各プロジェクトにはファイルが一つ以上あります。この構成を図 1.1に示します。

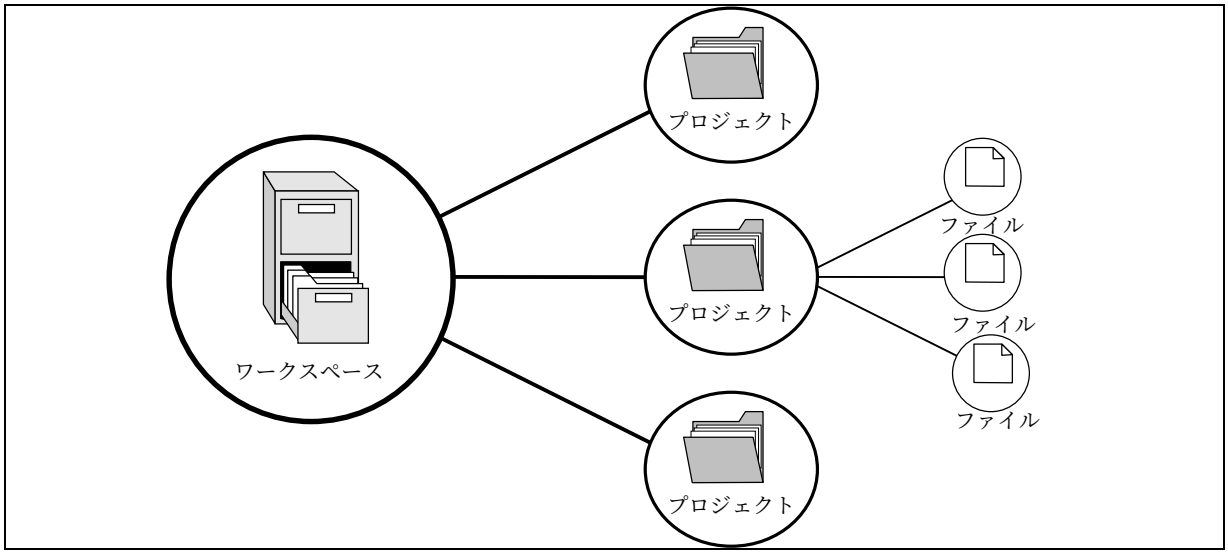


図 1.1: **ワークスペース、プロジェクト、ファイル**

ワークスペースでは関連したプロジェクトを一つにまとめることができます。例えば、異なるプロセッサに対して1つのアプリケーションを構築しなければならない場合、または、アプリケーションとライブラリを同時に開発している場合などに便利です。さらに、ワークスペース内でプロジェクトを階層的に関連づけることができます。つまり、1つのプロジェクトを構築すると、その子プロジェクトを最初に構築します。

ワークスペースを活用するには、ユーザは、まずワークスペースにプロジェクトを追加して、そのプロジェクトにファイルを追加しなければなりません。

1.2　HEW を起動する

HEWを起動するにはWindows®の“スタート”メニューを開き、“プログラム”を選択し、 “Hitachi Embedded Workshop2”を選択し、HEWのショートカットを選びます。デフォルトで図1.2に示すWelcome!ダイアログボックスが開きます。

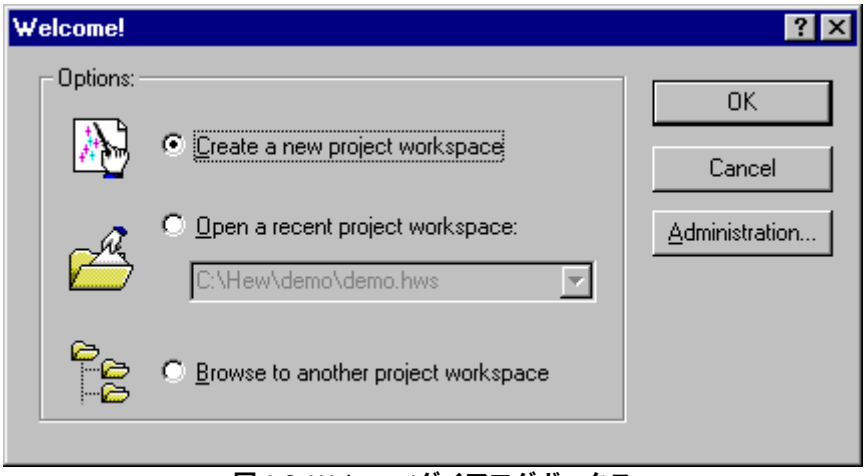


図 1.2: Welcome!**ダイアログボックス**

新規ワークスペースを作成するには“Create a new project workspace”ボタンを選択し、“OK”ボタンをクリックしてください。最近開いたワークスペースを開くには“Open a recent project workspace”ボタンを選択し、ドロップダウンリストから開きたいワークスペースを選択し、“OK”ボタンをクリックしてください。最近開いたワークスペースのリストには、最近使ったワークスペースファイルリストで見るものと同じ情報を表示します。このリストはファイルメニュー上に表示します。ワークスペースファイル（.HWSファイル）を指定してワークスペースを開くには“Browse to another project workspace”ボタンを選択し、“OK”ボタンをクリックしてください。HEWにツールを登録したり、HEWからツールの登録を外したりするには“Administration”ボタンをクリックしてください。ワークスペースを開かないでHEWを使うには“Cancel”ボタンをクリックしてください。

1.3　新規ワークスペースを作成する

➲新規にワークスペースを作成するには

1. Welcome!ダイアログボックス（図1.2）から“Create a new project workspace”オプションを選び、“OK”ボタンをクリックするか、[File->New Workspace...]を選んでください。New Project Workspace ダイアログボックスを表示します（図1.3）

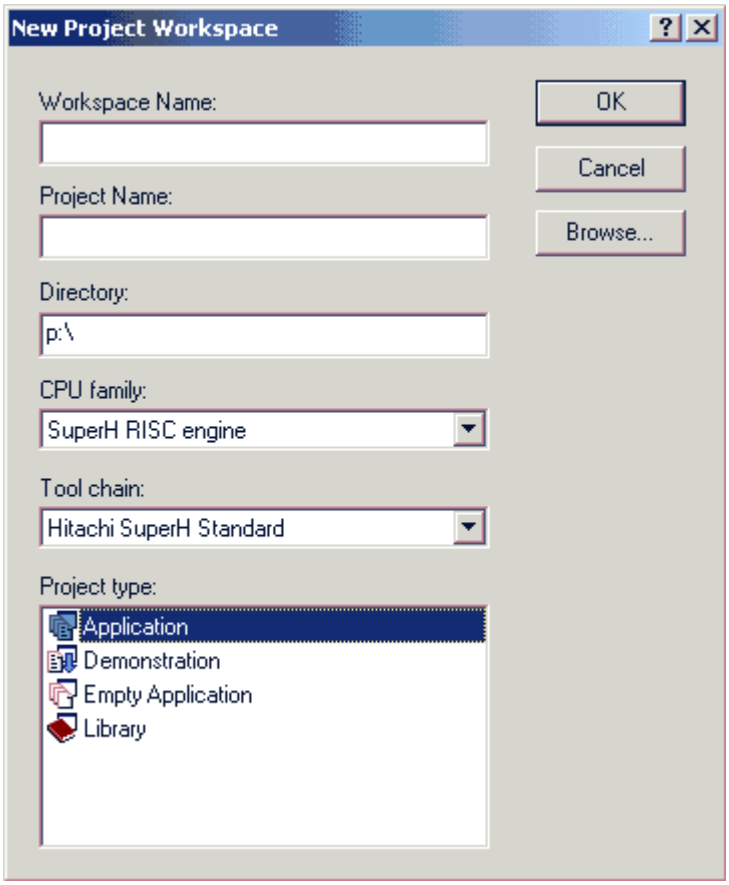


図 1.3: New Project Workspace **ダイアログボックス**

2. “Workspace Name”フィールドに新規ワークスペース名を入力してください。新規ワークスペース名は32文字以内で、半角英数字、半角下線を使用できます。ワークスペース名を入力すると、自動的にワークスペースのサブディレクトリおよびプロジェクト名をHEWに追加します。“Browse…”ボタンをクリックしてワークスペースを作成するディレクトリを選んだり、“Directory”フィールドに、ワークスペースを作成するディレクトリを手入力することができます。この場合ワークスペース名とプロジェクト名が異なります。
3. ワークスペースの基盤となるCPUファミリおよびツールチェインを選んでください。これらはワークスペースを作成した後で変更することができないので注意してください。
4. 新規ワークスペースを作成するとき、HEWは自動的にプロジェクト名フィールドで指定したプロジェクトを作成して、新規ワークスペースに追加します。“Project types”リストには、使用可能なプロジェクトの種類（アプリケーション、ライブラリなど）を表示します。作成するプロジェクトの種類をリストから選んでください。表示するプロジェクトの種類は、現在のCPUファミリおよびツールチェインの組み合わせに有効な全種類となります。ツールチェイン専用、デバッガ専用、またはHEWのデバッガおよびツールチェインの両方を構築するフルプロジェクトジェネレータがあります。
5. “OK”ボタンをクリックすると、新規のワークスペースとプロジェクトを作成します。

注意　同一ディレクトリにすでにワークスペースが存在する場合は、ワークスペースを作成できません。

1.4　ワークスペースを開く

➲ワークスペースを開くには

1. Welcome!ダイアログボックス（図1.2）から“Browse to another project workspace”オプションを選んで“OK”ボタンをクリックするか、 [File->Open Workspace…]を選んでください。Open Project Workspaceダイアログボックスを表示します。
2. 開きたいワークスペースファイルを選びます（.HWS ファイルのみ）。
3. “Open”ボタンをクリックしてワークスペースを開いてください。ワークスペースを開くときに情報を表示するように設定している場合、Workspace Propertiesダイアログボックスを表示します（図1.4）。設定していない場合、ワークスペースを開きます。
   Workspace Propertiesダイアログボックスを表示するかどうかはWorkspace Propertiesダイアログボックスの“Show workspace information on workspace open” チェックボックス、または、Tools Optionsダイアログボックス“Workspace”タブの “Display workspace information dialog on opening workspace” チェックボックスのチェックの有無によります。Workspace Propertiesダイアログボックスで“OK”ボタンをクリックするとワークスペースを開きます。“Cancel”ボタンをクリックするとワークスペースを開きません。

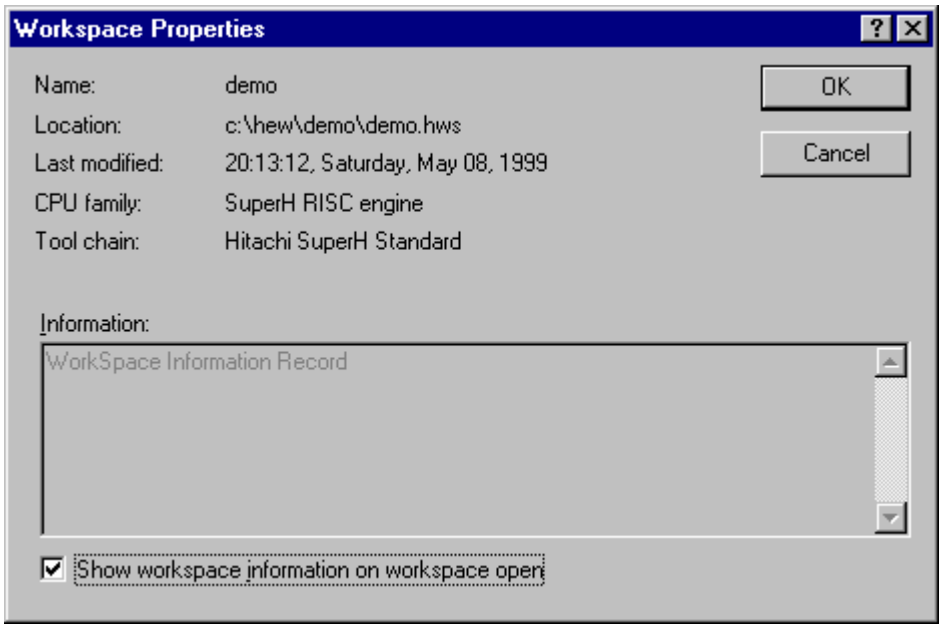


図 1.4　Workspace Properties **ダイアログボックス**

HEWでは、最近開いたファイル5つまでを“File”メニューの“Recent Workspaces”サブメニューに追加します。最近利用したファイルを再び開きたいときに便利です。

➲最近使ったワークスペースを開くには

1. Welcome!ダイアログボックスから“Open a recent project workspace”を選び、ドロップダウンリストからワークスペース名を選び、“OK”ボタンをクリックします。
   または、
2. [File->Recent Workspaces]を選び、そのサブメニューからワークスペース名を選びます。

注意　**HEW では、一度に 1 つのワークスペースしか開けません。したがって、すでに開いているワークスペースがあるときに別のワークスペースを開こうとすると、すでに開いているワークスペースを閉じてから新しいワークスペースを開きます。**

1.5　ワークスペースを保存する

[File->Save Workspace]を選ぶと、HEWのワークスペースが保存できます。

1.6　ワークスペースを閉じる

HEWのワークスペースを閉じるには、[File->Close Workspace]を選んでください。ワークスペースまたはそのプロジェクトに変更があった場合は、保存するかどうかを選んでください。

[File->Save Workspace]を選ぶと、HEWのワークスペースが保存できます。

1.7　古いワークスペースの使用

HEWでは、以前のバージョンのHEWで作成したワークスペースも開くことができます。これによって問題が発生することはなく、ワークスペースファイルの細部における違いもワークスペースを開いたときにアップグレードします。アップグレードしたファイルの現在のディレクトリに、初期のワークスペースまたはプロジェクトファイルのバックアップを保存しておく必要があります。日立デバッギングインタフェースで使用したセッションファイルをアップグレードすることはありません。

1.8　HEW を終了する

HEWを終了するには [File->Exit]を選ぶか、 Alt+F4アクセラレータを使用するか、システムメニューから“閉じる”オプションを選んでください。（システムメニューはHEWタイトルバーの最も左上側にあるアイコンをクリックすると開きます。）ワークスペースが開いているときは、前節で説明したワークスペースを閉じる操作を行います。

1.9　デバッガセッション

HEWは、ビルダオプションをコンフィグレーションへ保存することができます。同様に、HEWは、デバッガオプションをセッションに保存することもできます。セッションには、デバッギングプラットフォーム、ダウンロードするプログラム、各デバッギングプラットフォームのオプションを保存することができます。

セッションは、コンフィグレーションとは直接関連がありません。これは、複数のセッションが同じダウンロードモジュールを共有し、プログラムの不要なリビルドを避けられることを意味します。

各セッションのデータは、別々のファイルで HEW プロジェクトに保存します。詳細については、以下で説明します。

1.10　セッションを選択する

セッション選択するには、次の2通りの方法があります。

・ツールバーから選択する

1. ツールバーのドロップダウンリストボックス（図1.5）からセッションを選んでください。

図 1.5: **ツールバーの選択**

・ダイアログボックスから選択する

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。

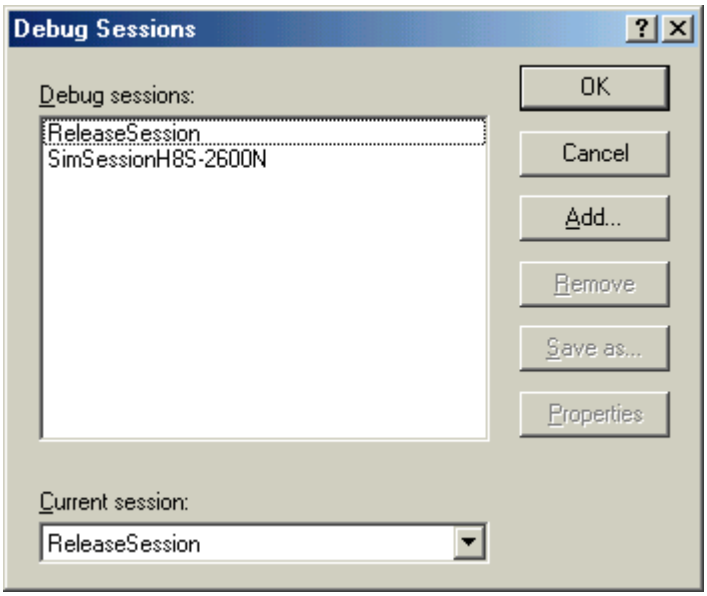


図 1.6: Debug Sessions **ダイアログボックス**

2. “Current session”ドロップダウンリストから使用したいセッションを選んでください。
3. “OK”をクリックして、セッションを設定してください。

1.10.1　セッションの追加と削除

別のセッションから設定をコピーしたり、セッションを削除したりして、新しいセッションを追加することができます。

・新しい空のセッションを追加する

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。
2. “Add...”ボタンをクリックしてください。Add new sessionダイアログボックスを表示します（図1.7）。
3. “Add new session”ラジオボタンをクリックしてください。
4. セッションの名前を入力してください。
5. “OK”をクリックし、Debug Sessionsダイアログボックスを閉じてください。
6. 入力したセッション名のファイルを新しく作成します。ファイルが既に存在する場合は、エラーを表示します。

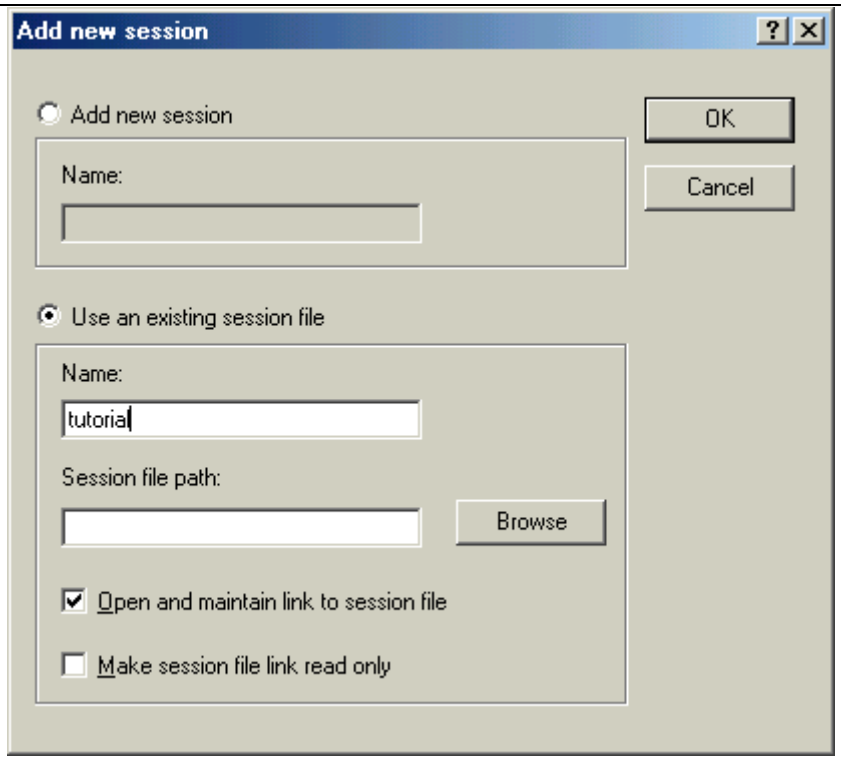


図 1.7: Add new sessions **ダイアログボックス**

・既存のセッションを新しいセッションファイルにインポートする

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。
2. “Add...”ボタンをクリックしてください。Add new sessionダイアログボックスを表示します（図1.7）。
3. “Use an existing session file”ラジオボタンをクリックしてください。
4. セッションの名前を入力してください。
5. 現在のプロジェクトにインポートしたい既存のセッションファイルをブラウズしてください。
   “Open and maintain link to session file”チェックボックスをクリックしない場合、プロジェクトディレクトリにインポートした新しいセッションファイルを生成します。
   “Open and maintain link to session file”チェックボックスをクリックした場合、プロジェクトディレクトリに新しいセッションファイルは生成せず、既存のセッションファイルにリンクします。
   “Make session file link read only”チェックボックスをクリックした場合、リンクしたセッションファイルをリードオンリーで使用します。
6. “OK”ボタンをクリックし、Debug Sessionsダイアログボックスを閉じてください。

・セッションを削除する

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。
2. 修正したいセッションを選んでください。
3. “Remove”ボタンをクリックしてください。
   現在のセッションを削除することはできません。
4. “OK”ボタンをクリックし、Debug Sessionsダイアログボックスを閉じてください。

・セッションのプロパティを見る

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。
2. 見たいプロパティのあるセッションを選んでください。
3. “Properties”ボタンをクリックしてください。Propertiesダイアログボックスを表示します（図1.8）。

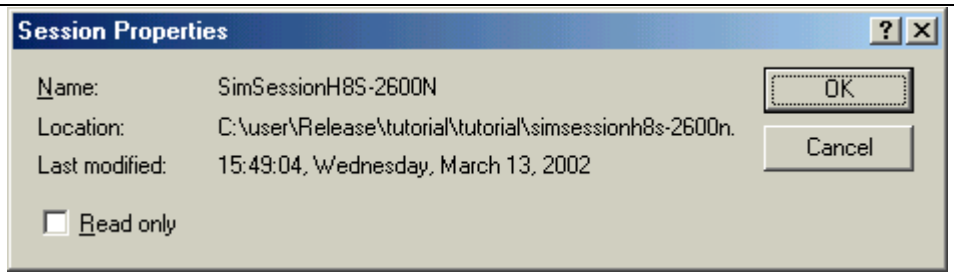


図 1.8: Session properties **ダイアログボックス**

・セッションをリードオンリーにする

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。
2. プロパティを見たいセッションを選んでください。
3. “Properties”ボタンをクリックしてください。Propertiesダイアログボックスを表示します（図1.8）。
4. “Read only”チェックボックスをクリックしてください。リンクをリードオンリーにします。これは、デバッガ設定ファイルを共有する場合、およびデータを間違って修正したくない場合に便利です。
5. “OK”ボタンをクリックしてください。

・セッションを別名で保存するには

1. [Options->Debug Sessions...]を選んでください。Debug Sessionsダイアログボックスを表示します（図1.6）。
2. 保存したいセッションを選んでください。
3. “Save as”ボタンをクリックしてください。Save Sessionダイアログボックスを表示します（図1.9）。
4. 新しいファイルの場所をブラウズしてください。
5. セッションファイルを別の場所へエクスポートしたい場合は、”Maintain link”チェックボックスをチェックしないでください。現在のセッションの場所の代わりに、この場所をHEWで使用したい場合は、”Maintain link”チェックボックスをチェックしてください。
6. “OK”ボタンをクリックしてください。

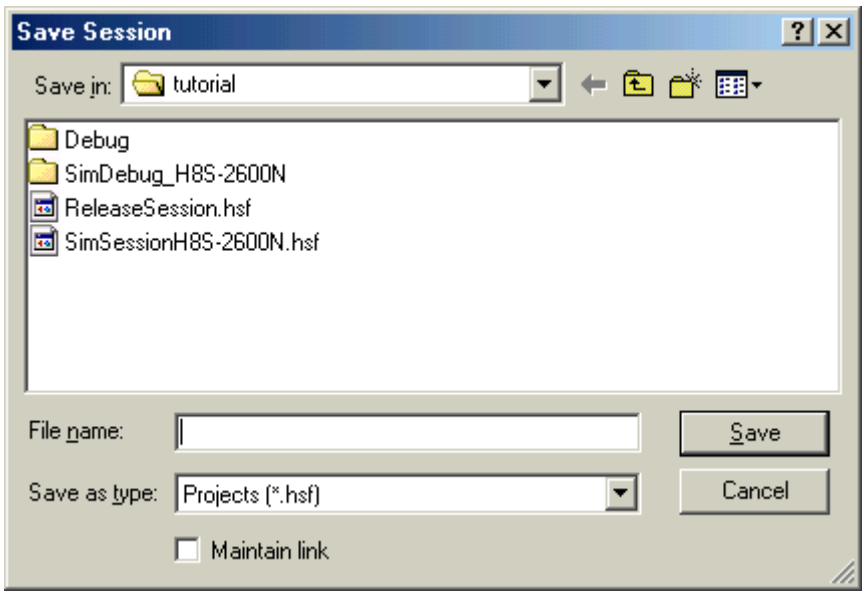


図 1.9: Session save as **ダイアログボックス**

1.10.2　セッション情報を保存する

➲セッションを保存するには

1. [File->Save Session]を選んでください。

2. デバッグの準備をする

この章では、デバッガが提供するすべての機能について説明し、プログラムのデバッグを開始するためのターゲットプラットフォームを設定します。ここでは、デバッグを行うためのデバッギングプラットフォームの選択および構築方法、またデバッグオブジェクトファイルのロードについて学びます。

2.1 デバッグの前にコンパイルを行う

プログラムをC/C++ソースレベルでデバッグするには、C/C++プログラムをコンパイルして、Debugオプションをイネーブルの状態でリンクする必要があります。このオプションがイネーブルの場合、コンパイラは、C/C++コードのデバッグに必要な情報をすべてアブソリュートファイルまたはマネージメントファイルに入れます。これらのファイルはその後「デバッグオブジェクトファイル」とよばれます。プロジェクトを作成すると、通常の初期セットアップでデバッグを設定します。

注意　デバッグ用のオブジェクトファイルを生成する際はコンパイラおよびリンカの Debug オプションを必ずイネーブルにしてください。
デバッグオブジェクトファイルにデバッグ情報(例えば S レコードフォーマットなど)がない場合でもデバッギングプラットフォームにロードできますが、アセンブリ言語レベルでのみデバッグすることができます。

2.2 デバッギングプラットフォームを選択する

デバッギングプラットフォームの選択はHEWのインストール方法に大きく依存します。HEWにツールチェインをインストールしている場合、アプリケーションプロジェクトジェネレータはツールチェインおよびデバッガのターゲットを同時にセットアップすることができます。これによってターゲットおよびツールチェインオプションを一致させ、矛盾が起こらないようにします。ツールチェインをインストールしていない場合には、デバッグ専用プロジェクトのみ選択することができます。HEWはデフォルトの設定で、New Workspaceダイアログボックスの中に、生成するそれぞれのCPUファミリのデバッグ専用プロジェクト生成タイプを表示します。このプロジェクトタイプは、前バージョンのHDI sessionダイアログボックスと同じ動作を提供します。

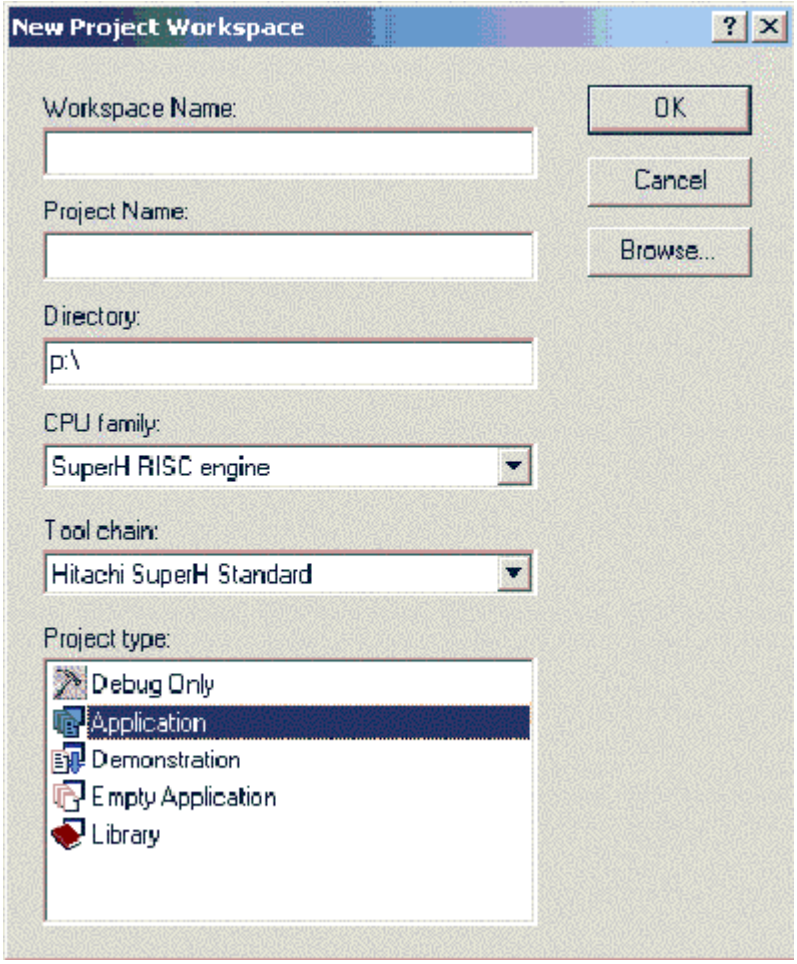


図 2.1: New Workspace **ダイアログボックス**

図2.1に示すダイアログボックスを使用して、ターゲットCPUに合ったプロジェクト生成タイプを選択することができます。

➲標準のデバッグプロジェクトタイプを使ってデバッグ専用のプロジェクトを生成するには

1. [File->New Workspace…]を選んでください。図2.1に示すダイアログボックスを表示します。
2. “Debug Only”プロジェクトタイプを選び、“OK”ボタンをクリックしてください。
3. “Debug Only”プロジェクトウィザードを表示します。図2.2に示すダイアログボックスを表示します。

注意　ツールチェインとシミュレータ・デバッガをインストールしている場合は、“ Empty Application”**がデバッグ専用プロジェクトになります。**

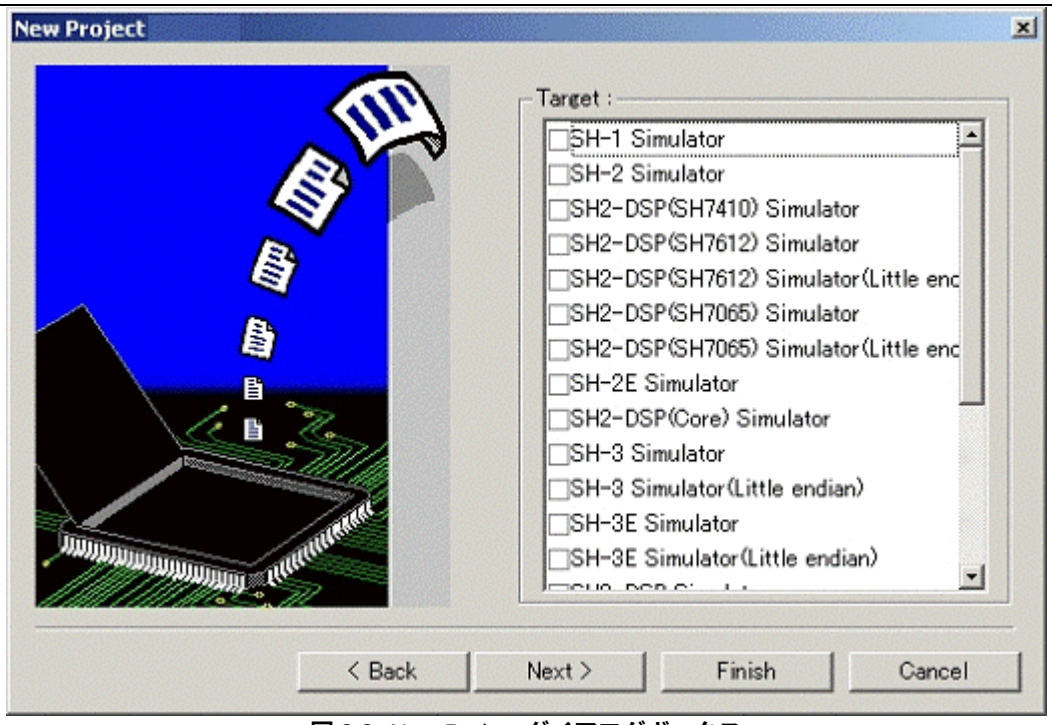


図 2.2: New Project **ダイアログボックス**

4. 現在インストールしている有効なデバッガターゲットをすべて表示します。ユーザは、この新しいデバッグプロジェクトでデバッグに使用するプラットフォームを選択することができます。ターゲットは複数選択することができ、各ターゲット用にデバッガセッションを生成します。
5. “Finish”ボタンをクリックして生成処理を完了します。Summaryダイアログボックスを表示します。生成処理が完了すると、デバッガプロジェクトジェネレータはプロジェクトを作成し、そのプロジェクトはそれぞれのデバッガセッションに対し、Debugオプションを正しく設定します。またジェネレータは、デフォルトのコンフィグレーションを1つ作成します。

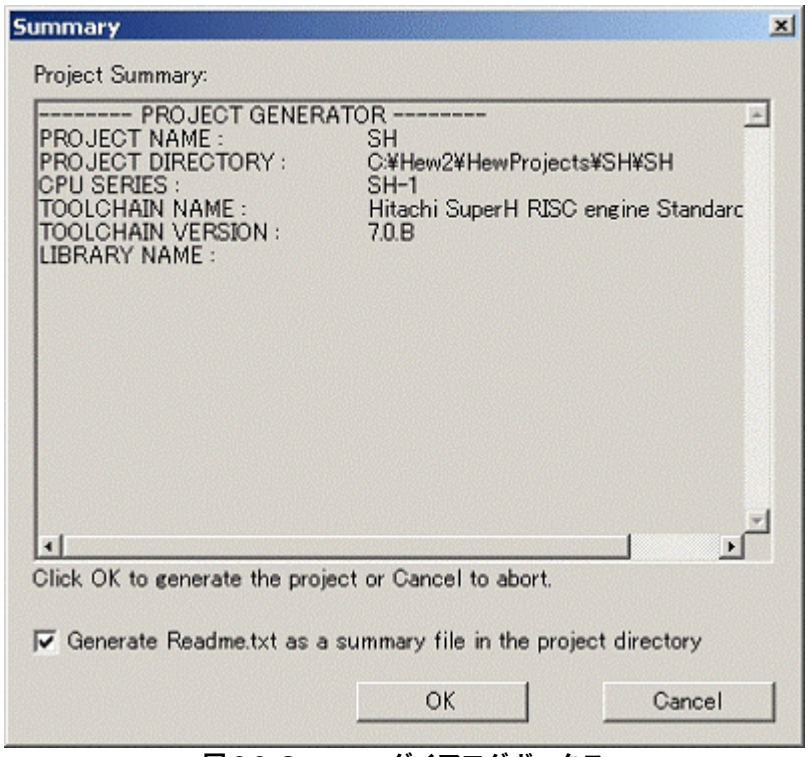


図 2.3: Summary **ダイアログボックス**

2.3　デバッギングプラットフォームを構築する

デバッギングプラットフォームにプログラムをロードする前に、アプリケーションシステムに合うようにプラットフォームを設定しなければなりません。必要な設定内容は、デバイスの種類、動作モード、クロックスピード、およびメモリマップなどです。その中でも特にメモリマップの設定が大切です。HEWでは、プロジェクトの生成処理でこの作業のほとんどが完了します。しかし、標準とは異なるボードのコンフィグレーションを使用する場合には、カスタマイズが必要になります。

2.3.1　メモリマップ

デバッギングプラットフォームは、デバイスのアドレス空間がROM、RAM、オンチップレジスタ、またはメモリのない領域なのかを知る必要があります。そのため、メモリマップを設定する必要があります。

プロジェクトジェネレータでデバイスの種類およびモードを選択すると、HEWは自動的にそのデバイスのマップおよびそのプロセッサが動作するモードを設定します。例えば、内部ROMおよびRAMを持つデバイスにおいて、これらのメモリがデバイスメモリマップ内で位置する領域をデフォルトで設定します。

また、外部メモリを使用する場合には、デバッギングプラットフォームに知らせる必要があります。

シミュレータを使用する場合、Simulator Systemダイアログボックスによりメモリマップの設定を変更することができます。詳細はデバッグプラットフォームのマニュアルを参照下さい。

また、シミュレータを使用する場合、Standard Toolchainダイアログボックスの”Simulator”タブで、設定内容を参照することができます。Standard Toolchainダイアログボックス（図2.4）の”View”ボタンをクリックすると、CPU hardware mapダイアログボックス（図2.5）を表示します。

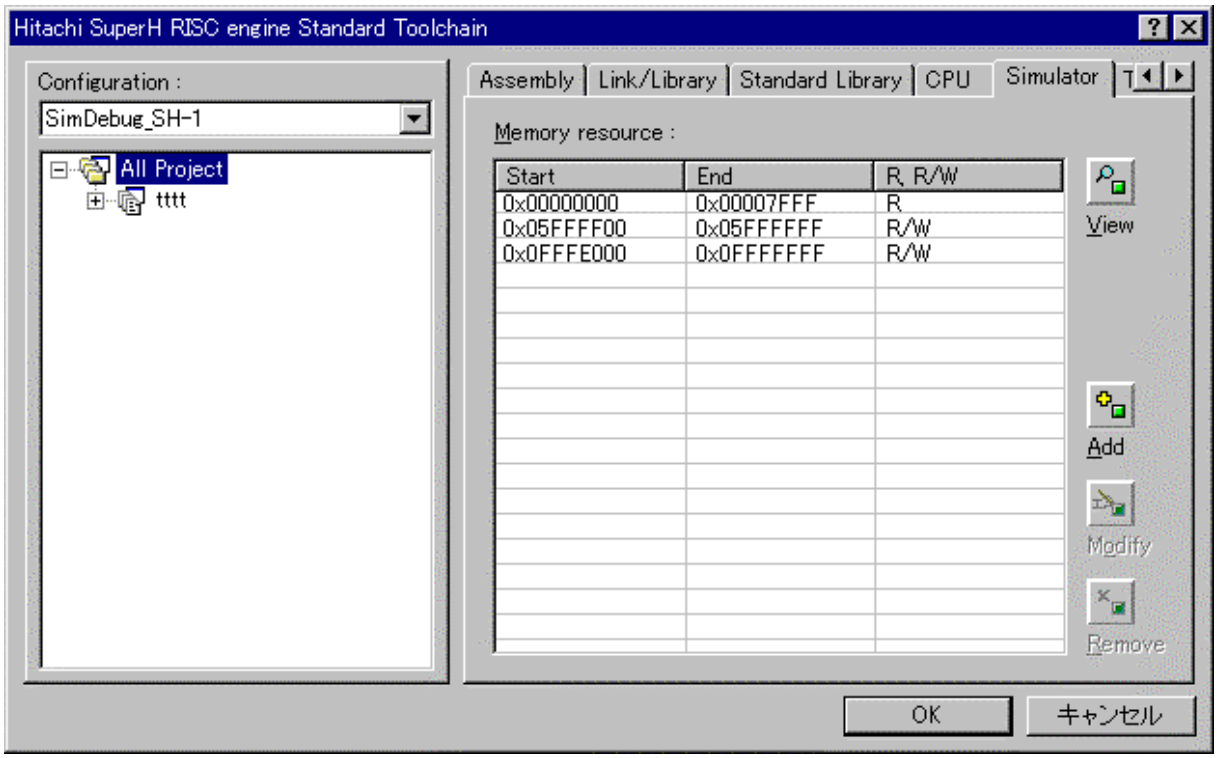


図 2.4: Standard Toolchain **ダイアログボックス**

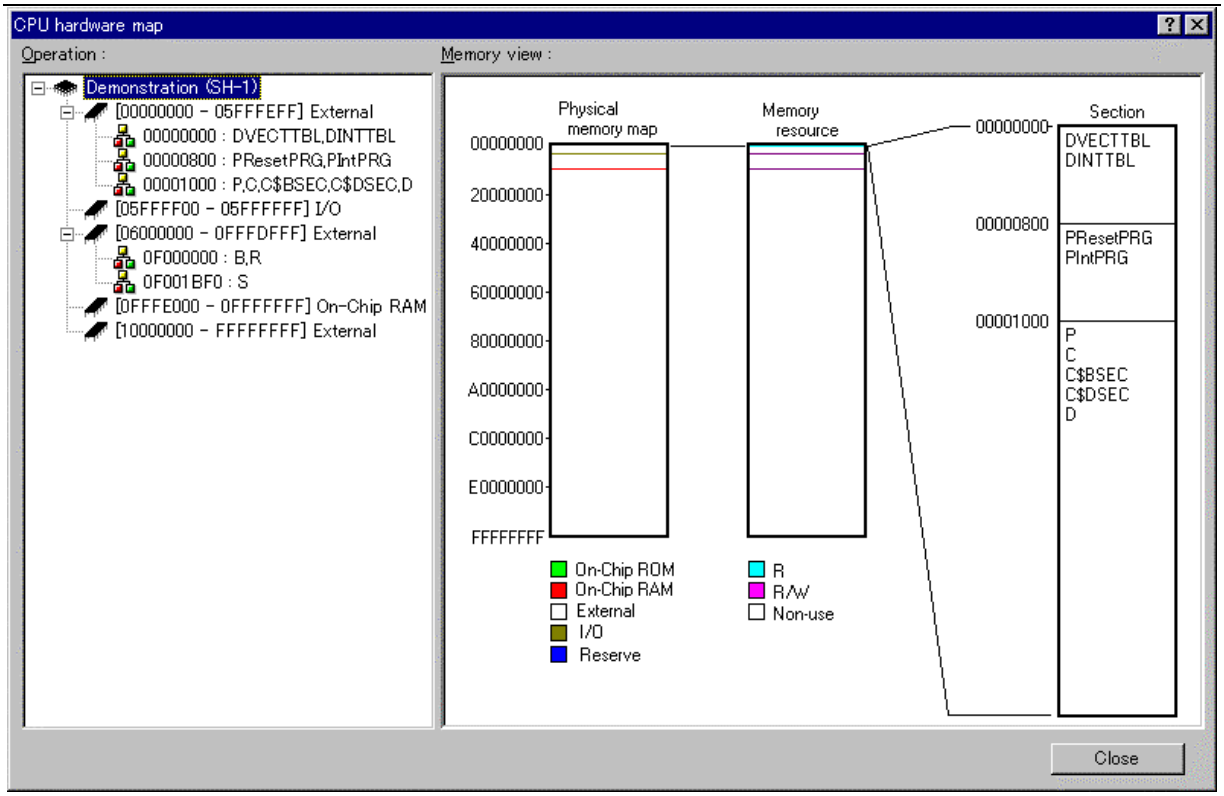


図 2.5: CPU hardware map **ダイアログボックス**

以下の情報を表示します。

[Operation]　　アドレス範囲を表示します。

[Memory view]　　メモリマップおよびメモリリソース情報を表示します。

2.3.2　メモリリソース

シミュレータを使用する場合、使用するアドレス範囲のメモリリソースを確保しなければ行けません。メモリリソースは、Simulator Memory Resourceダイアログボックスにより設定を変更することができます。詳細はデバッグプラットフォームのマニュアルを参照下さい。

また、シミュレータを使用する場合、図2.4のStandard Toolchainダイアログボックスの”Simulator”タブで、設定内容を参照することができます。”Simulator”タブには下記のボタンがあります。

[View]　　メモリマップ情報を表示します。

[Add]　　メモリリソースを追加します。

[Modify]　　選択したメモリリソースを変更します。

[Remove]　　選択したメモリリソースを削除します。

“Add”ボタンをクリックするとAdd memory resourceダイアログボックス（図2.6）を表示します。

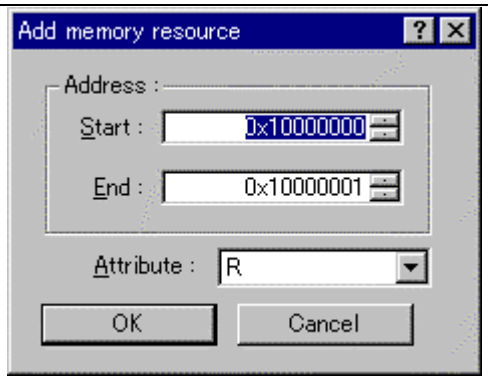


図 2.6: Add memory resource

Add memory resourceダイアログボックスには下記を設定してください。

[Start]　先頭アドレス

[End]　終了アドレス

[Attribute]　属性（R:リードのみ、R/W:リード／ライト可）

Simulator Memory Resourceダイアログボックス変更したメモリリソース情報は、Standard Toolchainダイアログボックスに反映され、Standard Toolchainダイアログボックスの変更もSimulator Memory Resourceダイアログボックスに反映されます。

2.3.3　プログラムをダウンロードする

プログラムをダウンロードするためのメモリがシステムに十分あるならば､デバッグするプログラムをダウンロードすることができます。デフォルト設定では、現在のプロジェクトのリンカが出力したプログラムを自動的にダウンロードします。

また、プロジェクトを作成したあとにダウンロードモジュールを手動で選択することができます。これは、図2.8に示すDebug Settingsダイアログボックスを使用して行うことができます。このダイアログボックスでは、ワークスペース全体に渡りデバッグの設定を制御することができます。ダイアログボックスの左にあるツリーは、現在のプロジェクトをすべて含みます。このツリーでプロジェクトを選ぶと、そのプロジェクトの設定の表示、およびセッションドロップダウンリストには、セッションの選択を表示します。このリストボックスでは、複数のセッションを選択することもできますし、すべてのセッションを選択することもできます。複数のセッションを選択した場合には、１つまたは複数のセッションの設定を同時に修正することができます。Debug Settingsダイアログボックスは、以下のデバッグオプションを表示します。

- 現在のプロジェクトのための現在のデバッグターゲットおよびセッションの選択
- 現在のプロジェクトのためのダウンロードモジュールおよびセッションの選択

ダウンロードモジュールリストは、ターゲットにダウンロードするファイルの順番を表示します。このリストの中でモジュールを追加、削除、修正、上へ移動、下へ移動することができます。

➲新規のダウンロードモジュールを追加するには

1. [Options->Debug Settings…]を選択してください。図2.8に示すダイアログボックスを表示します。
2. tree controlにおいて、ダウンロードモジュールを追加するプロジェクトおよびコンフィグレーションを選択します。
3. “Add”ボタンをクリックしてください。図2.7に示すダイアログボックスを表示します。
4. “Format”リストボックスは、サポートしているオブジェクトフォーマットリーダのリストを含みます。これによってプロジェクトに追加するダウンロードモジュールのリーダを正しく選択することができます。
5. “Filename”フィールドを使うと、使用中のディスクにあるダウンロードモジュールをブラウズすることができます。また、まだモジュールをビルドしていなければ、edit controlに入力することができます。
6. “Offset”フィールドにはオフセットアドレスを入力してください（一部のオブジェクトフォーマットに対してのみ入力できます）。
7. ユーザが“OK”ボタンをクリックすると、デバッグダウンロードモジュールをリストの最後に追加します。そのモジュールを他のモジュールと関連した別の位置に変えたいときには、モジュールを選択して“Up”および“Down”ボタンを使用してモジュールの位置を正しく設定してください。

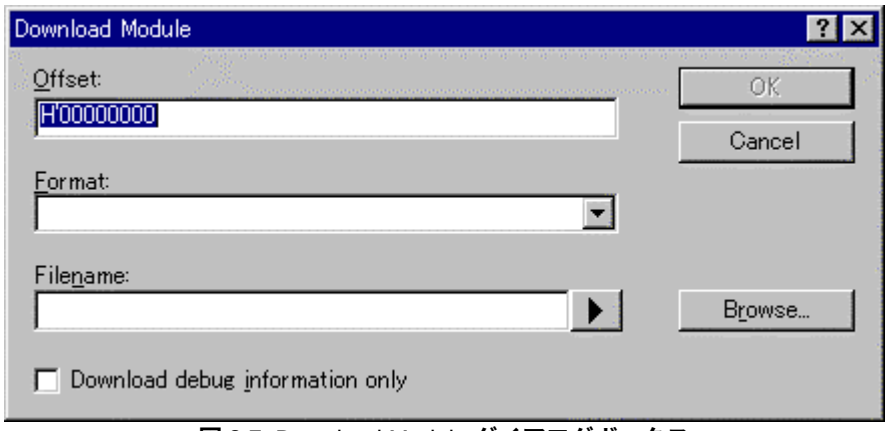


図 2.7: Download Module **ダイアログボックス**

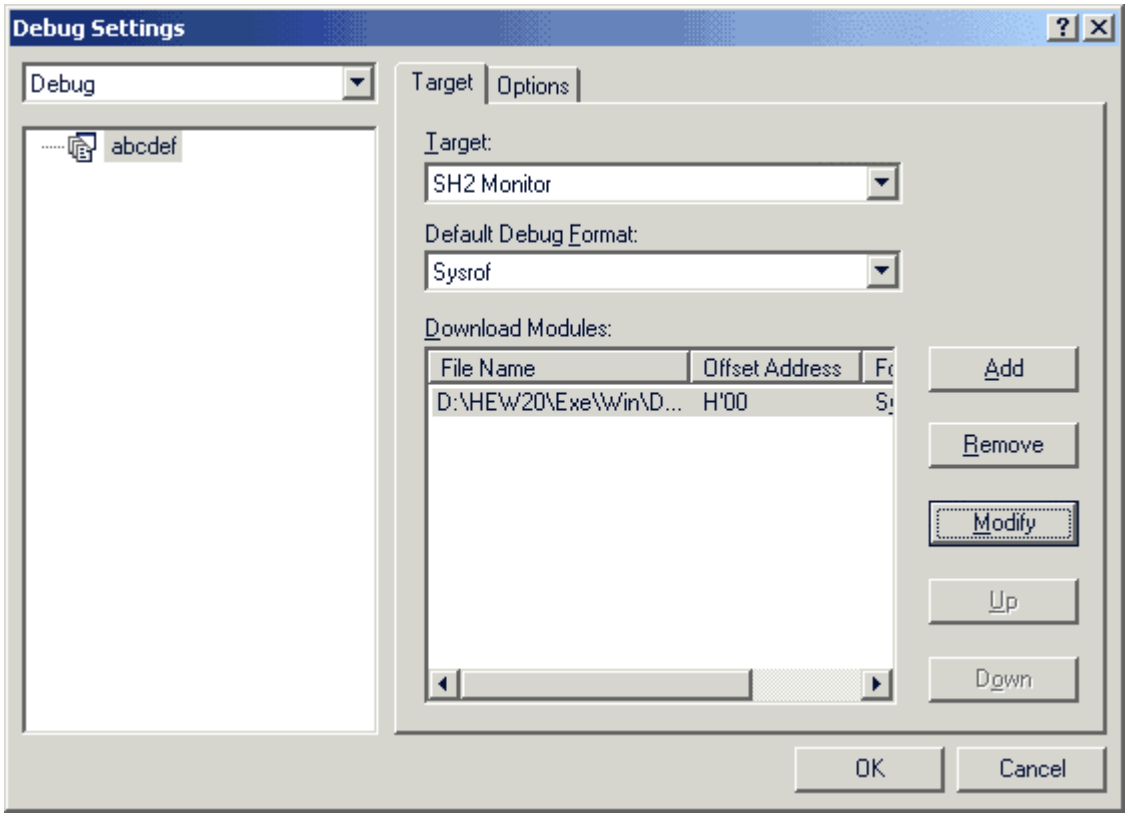


図 2.8: Debug Settings **ダイアログボックス**(Target **タブ**)

Debug Settingsダイアログボックスで行った変更はすべて、“OK”ボタンをクリックするまで更新しません。現在のターゲットに対し、ダウンロードモジュールに変更があった場合には、モジュールに印がつき、次回のデバッグの際、プログラムを実行するときにダウンロードすることができます。

デフォルトのデバッグフォーマットは、リスト中の最初のダウンロードモジュールに設定しています。1セッションにつき、デフォルトのデバッグオブジェクトフォーマットを1つだけ指定することができます。現在インストールしているデバッガフォーマットはすべてここに表示します。

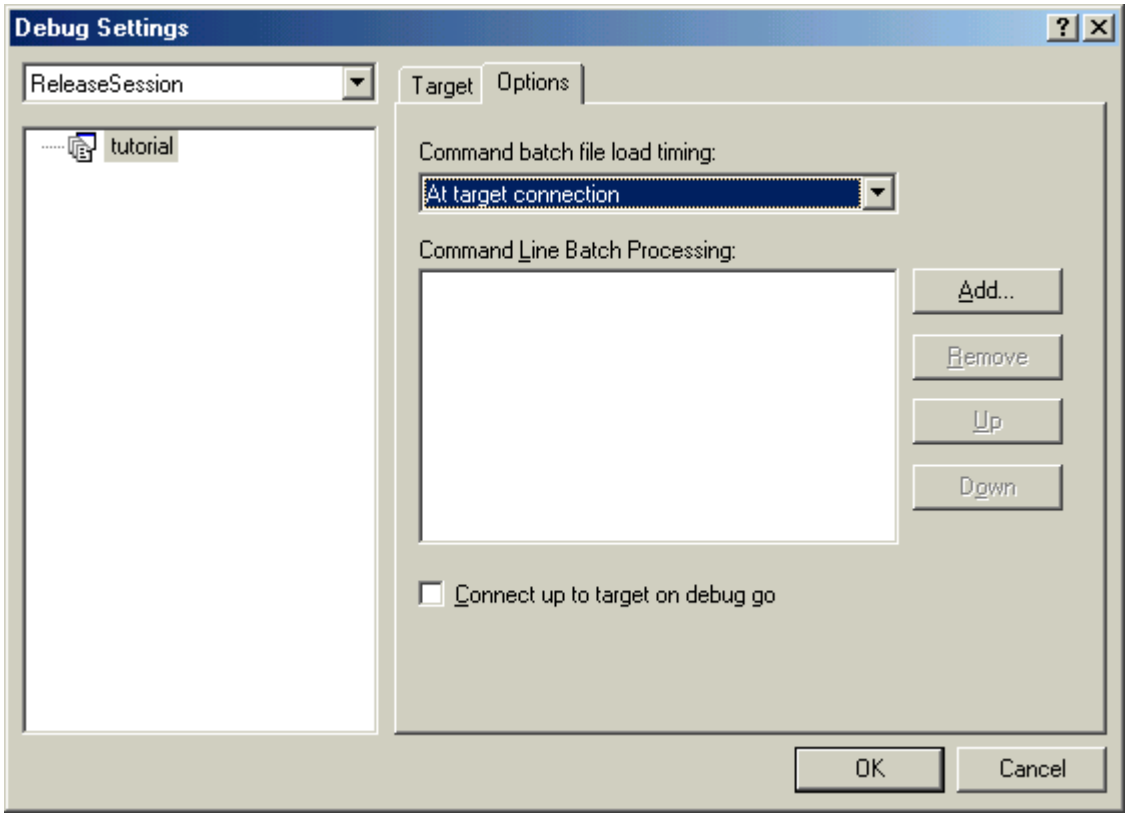


図 2.9: Debug Settings **ダイアログボックス**(Options **タブ**)

Debug Settingsダイアログボックスの“Options”タブ（図 2.9）では、ターゲットをいつ接続するのかを決定することができます。このオプションをチェックしていると、ユーザが[Build->Debug ->Go]を選択するまでターゲットを接続しません。チェックしていないと、コンフィグレーションを開く度にターゲットを接続します。新しいワークスペースやプロジェクトを開いたときやツールバーやBuild Configurationsダイアログボックスを使ってコンフィグレーションを切り替えたときにコンフィグレーションを開きます。

また、このタブの“Command Line Batch Processing”リストでは、コマンドラインバッチコマンドを指定できます。コマンドラインバッチコマンドは、コマンドラインを経由してデバッグを行います。このコマンドをいつ実行するか、”Command batch file load timing”ドロップダウンリストで選択することができます。”Command batch file load timing”ドロップダウンリストを選択してから、実行するバッチファイルを追加してください。リスト内の順番は、タイミングイベントが発生したときにコマンドラインバッチファイルを実行する順番です。

2.3.4　モジュールを手動でダウンロードする

いったんターゲットにダウンロードするモジュールを決定したら、接続したターゲット上のモジュールを手動で更新することができます。これは[Debug->Download Modules]で行います。ダウンロードするモジュールは、1つまたはすべてのモジュールを選択できます。また、Workspaceウィンドウにある”Download Module”フォルダの下のモジュールを右クリックしても更新できます。

2.3.5　モジュールを自動的にダウンロードする

[Debug->Run] を選択すると、HEWは前回のダウンロードからモジュールに影響を与えるファイルまたはデバッガの設定に変更があったかどうかをチェックします。HEWは変更を見つけると、それぞれのモジュールをダウンロードする必要があるかどうかをユーザに問い合わせます。ユーザが”Yes”を選択すると、モジュールをターゲットにダウンロードします。

複数のモジュールを同じ開始アドレスに構築した場合、そのアドレスの最初のモジュールのみをデフォルトでダウンロードします。よって、[Debug->Download Modules]および[Debug->Unload Modules]メニュー項目から、他のモジュールを手動でロードしたりアンロードしたりすることができます。

2.3.6　モジュールをアンロードする

[Debug->Unload Modules]メニュー項目から、ダウンロードしたモジュールを手動でアンロードすることができます。また、Workspaceウィンドウの”Download Module”ポップアップメニューからアンロードすることもできます。

モジュールをアンロードすると、そのモジュールのシンボルは、HEWデバッギングシステムからなくなりますが、ターゲットのメモリ内容は変更しません。一度アンロードしたモジュールは、再度ロードするまでデバッグすることはできません。

2.3.7　レジスタ内容を見る

アセンブリ言語レベルでデバッグを行なう場合に、CPUの汎用レジスタの内容を簡単に見ることができます。これは、Registersウィンドウを使用して行います。

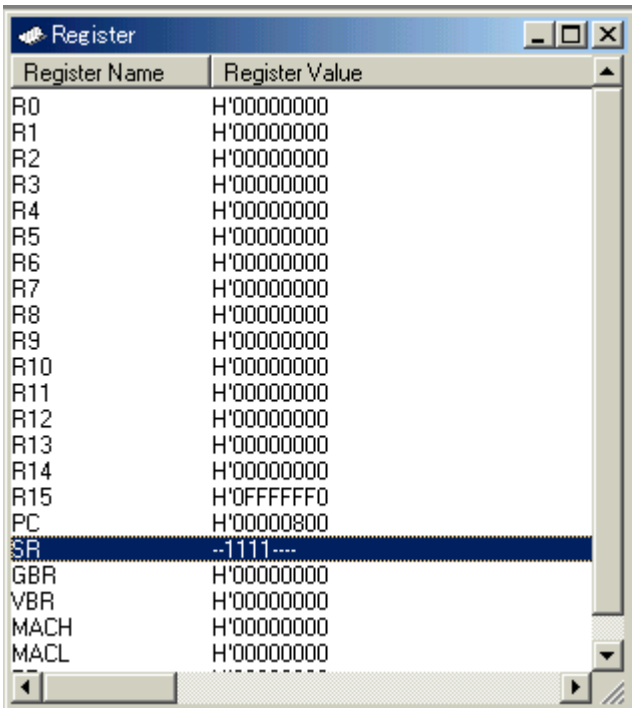


図 2.10: Registers **ウィンドウ**

Registersウィンドウを開くには、[View->Registers]を選択するか、Registersウィンドウツールバーボタン  をクリックします。Registersウィンドウが開きCPU汎用レジスタおよびその値(16進数)を表示します。

2.3.8　ビットレジスタを拡張する

ビットレベルのフラグで制御するレジスタの場合、数値ではなく記号でビットの状態を表示します。また、そのレジスタをダブルクリックするとEdit Registerダイアログを表示し、各ビット毎にオン／オフを設定できます。各ビットのチェックボックスをチェックしたときは１を、クリアしたときは０を設定します。

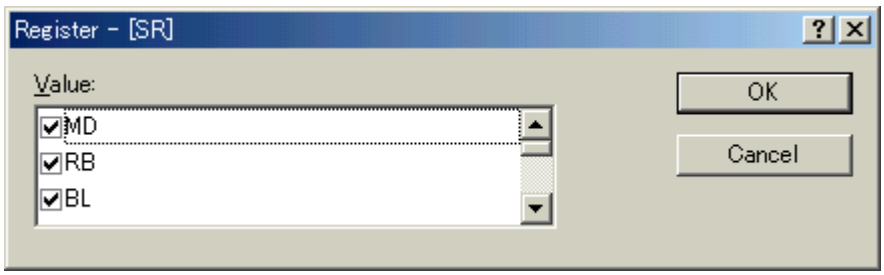


図 2.11: **フラグレジスタの拡張**

2.3.9　レジスタの内容を修正する

レジスタの内容を修正するには、以下のいずれかの方法でEdit Register ダイアログボックスを開きます。

(1)　修正したいレジスタをダブルクリックします。
(2)　修正したいレジスタを選択して、ポップアップメニューの[Edit…]を選択します。

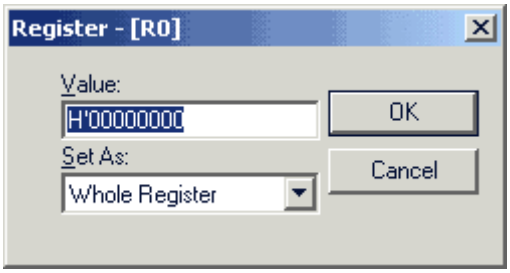


図 2.12: Edit Register **ダイアログボックス**

“Value”フィールドには数字またはC/C++の式を入力することができます。ドロップダウンリストからオプションを選択して、レジスタの内容のすべて、マスクした領域、フローティングビットまたはフラグビットを修正することができます(このリスト内容は、CPUのモデルおよび選択したレジスタによって異なります)。

新しい数字および式を入力したら“OK”ボタンをクリックするかENTERキーを押します。ダイアログボックスは閉じて、新しい値をレジスタに書き込みます。

2.3.10　レジスタの内容を使用する

DisassemblyまたはMemoryウィンドウのアドレス指定など、HEWの別のところで値を入力する場合、CPUレジスタの中にある値を使用するためには、#R1、#PC、#R6L、または#ER3などのように、レジスタ名の先頭に“＃”記号をつけてください。

3. プログラムを表示する

この章では、プログラムをソースコードおよびアセンブリ言語ニーモニックとして見る方法を説明します。HEWには、この章で説明しているコードまたはシンボル情報に対処するための機能があり、ユーザインタフェースでテキストファイルを見る方法を示します。

注意　ブレークが起こると、HEW はプログラムカウンタ(PC)の場所を表示します。多くの場合、例えば、SYSROF をベースにしたプロジェクトがもともとのパスから移動した場合、ソースファイルを自動的に見つけることができない場合があります。この場合、HEW はソースファイルブラウザダイアログボックスを開くので、ユーザは手動でファイルを探すことができます。このパスは、このデバッグプロジェクトのほかのソースファイルを更新するために使用します。

3.1 コードを表示する

ソースファイルを選択して”Open”ボタンをクリックすると、HEWは、統合化エディタのファイルを開きます。または、ワークスペースウィンドウのソースファイルをダブルクリックすることによって表示することができます。

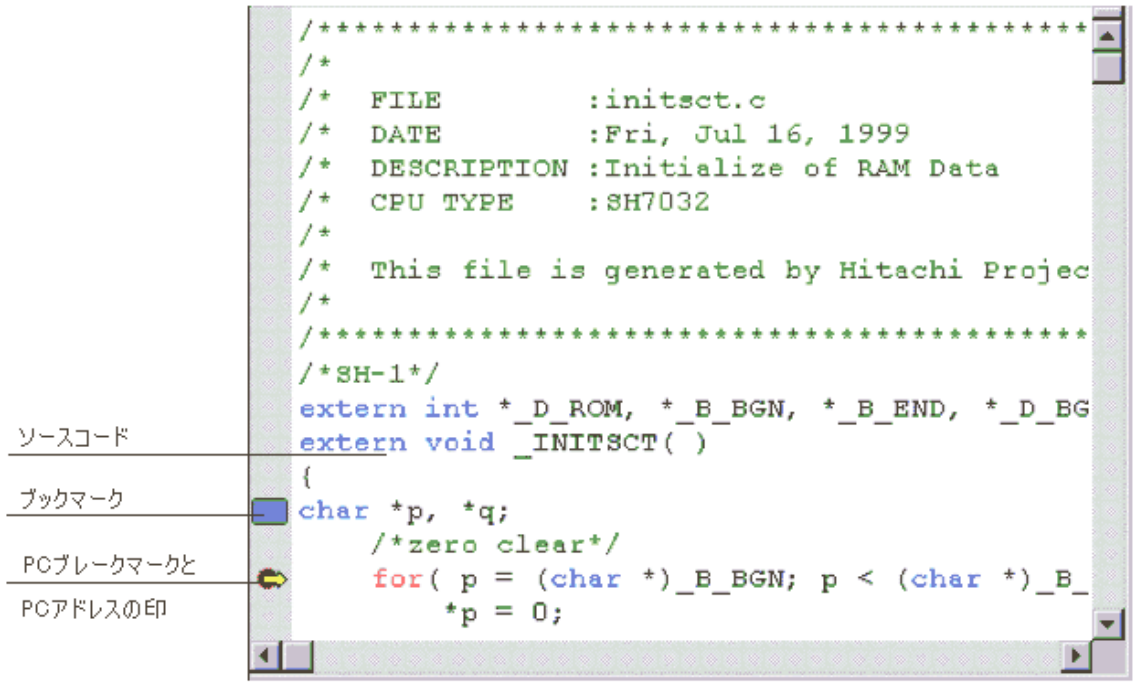


図 3.1: Source **ウィンドウ**

エディタを左余白領域（ブレークポイント、PCの場所などを表示します）およびテキスト領域（カラーシンタックスのハイライト表示したコードを含む）の２つの領域に分けます。これを図3.1（上記）に示します。

3.2　アセンブリ言語コードを表示する

ソースファイルが開いているときは、右ボタンをクリックしてポップアップメニューを開き、[View disassembly]を選択して現在のSourceウィンドウと同じアドレスにDisassemblyウィンドウを表示します。

ソースファイルが存在しなくてもアセンブリ言語レベルでコードを表示したい場合は、[View->Disassembly...]を選択するか、Ctrl+Dアクセラレータを使用するか、Disassemblyウィンドウツールバーボタン  をクリックします。

Disassemblyウィンドウは現在のPCの場所で開きます。また、ディスアセンブルニーモニック(可能なときはラベルも一緒に)を表示するAddress, Code (オプション)を表示します。

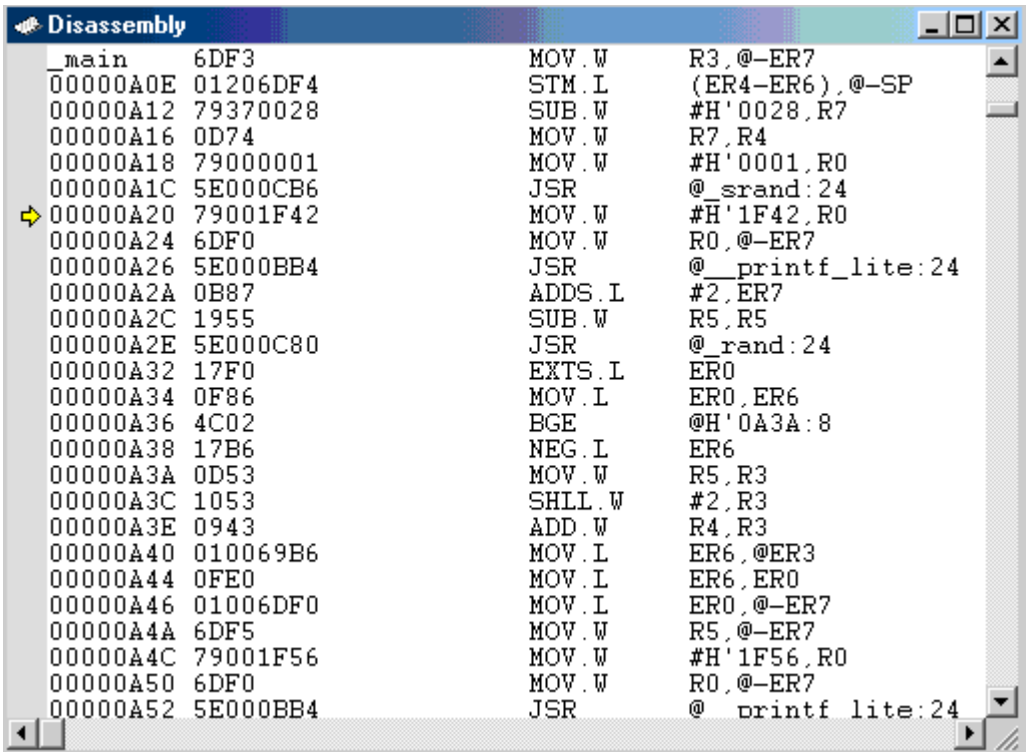


図 3.2: Disassembly **ウィンドウ**

3.3　アセンブリ言語コードを修正する

修正したい命令をダブルクリックすることによって、アセンブリ言語コードを修正することができます。Assemblerダイアログボックスが開きます。

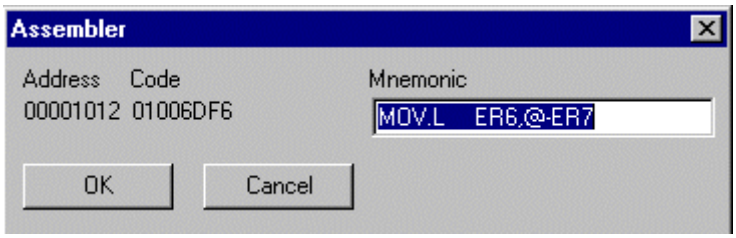


図 3.3: Assembler **ダイアログボックス**

アドレス、機械語コード、およびディスアセンブル命令を表示します。新しい命令を入力するか、”Mnemonics”フィールドの古い命令を編集します。ENTERキーを押すと、命令をメモリにアセンブルして、次の命令に移ります。”OK”ボタンをクリックすると、命令をメモリにアセンブルしてダイアログボックスを閉じます。”Cancel”ボタンをクリックするかESCキーを押すと、ダイアログボックスが閉じます。

注意　アセンブリ言語表示は、デバッギングプラットフォームのメモリの実際の機械語コードからディスアセンブルします。メモリの内容を修正すると、ダイアログボックス（および Disassembly **ウィンドウ）には新しいアセンブリ言語コードを表示します。しかし、**Source **ウィンドウは変更しません。これはソースファイルにアセンブラを含む場合も同じです。**

3.4　ラベルを見る

HEWがユーザプログラムソースコードをメモリ上の実際のコードにリンクする際に使用するデバッグ情報のほかに、デバッグオブジェクトファイルもシンボル情報を含みます。これはテキスト名の表で、プログラムのアドレスをあらわします。HEWではラベルと呼ばれています。Disassemblyウィンドウにおいて、対応するアドレスの代わりとして、また命令オペランドの一部として、ラベルの最初の8文字を表示します。

注意　オペランドとラベルの値が一致すれば、命令のオペランドはラベル名に置き換わります。同じ値を持つラベルが 2 **つ以上ある場合、アルファベット順で先に来るラベルを表示します。**

ヒント:edit control **にアドレスまたは値を入力できる場合には、代わりにラベルを使用することができます。**

3.5　ラベルを一覧にする

現在のデバッガセッションに定義したラベルのすべてを見るには、[View->Labels]を選択します。

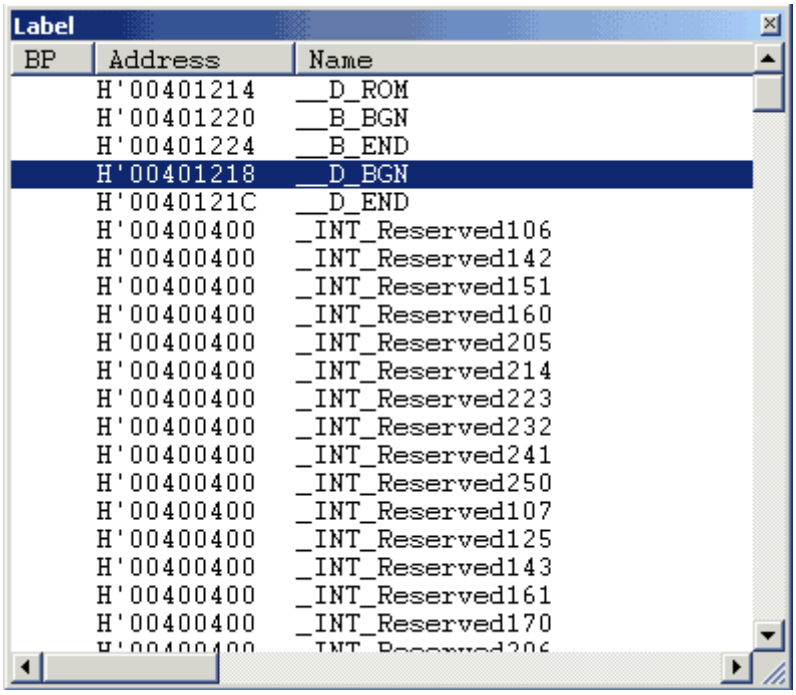


図 3.4: Labels **ウィンドウ**

それぞれのコラムのヘッダをクリックすることによりアルファベット順（ASCIIコードによって）またはアドレス値でソートしたシンボルを表示させることができます。

BPコラムをダブルクリックすることにより関数の入り口でソフトウェアブレークポイントをすばやく設定したり解除したりすることができます。または、右ボタンをクリックしてポップアップメニューを開いて[Break]を選択します。

3.6　特定のアドレスを見る

Disassemblyウィンドウを使って作成したプログラムを見ているとき、プログラム内のほかのところも見たいときがあります。そのような場合、プログラム内のコードをスクロールせずに特定のアドレスに直接行くことができます。ポップアップメニューから[Set Address]を選択します。図3.5に示すダイアログボックスを表示します。

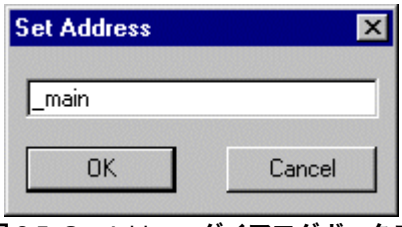


図 3.5: Set Address **ダイアログボックス**

エディットボックスにアドレスまたはラベル名を入力して、”OK”ボタンをクリックするかENTERキーを押します。Disassemblyウィンドウを更新して新しいアドレスコードを表示します。オーバーロード関数またはクラス名を入力した場合、Select Functionダイアログボックスを開くので、関数を選択してください。これについては、このマニュアルの「7章 Elf/Dwarf2のサポート」で詳細を説明します。

3.7　現在のプログラムカウンタアドレスを見る

HEWでアドレスまたは値を入力できるところでは、式も入力することができます。先頭にハッシュ文字をつけたレジスタ名を入力すると、そのレジスタ内容を式の値として使用します。従って、Set Addressダイアログボックスを開いて"#pc"という式を入力すると、SourceまたはDisassemblyウィンドウには、現在のPCアドレスを表示します。例えば、"#PC+0x100"といったPCレジスタおよびオフセットの式を入力することにより現在のPCのオフセットも表示することができます。

3.8　ソースアドレスカラム

プログラムをダウンロードすると、Sourceウィンドウにソースファイルに対するアドレスを表示することができます。（図 3.6）アドレスは、Sourceウィンドウの左側に表示します。この機能は、PCまたはハードウェアブレークポイントを設定するときに便利です。

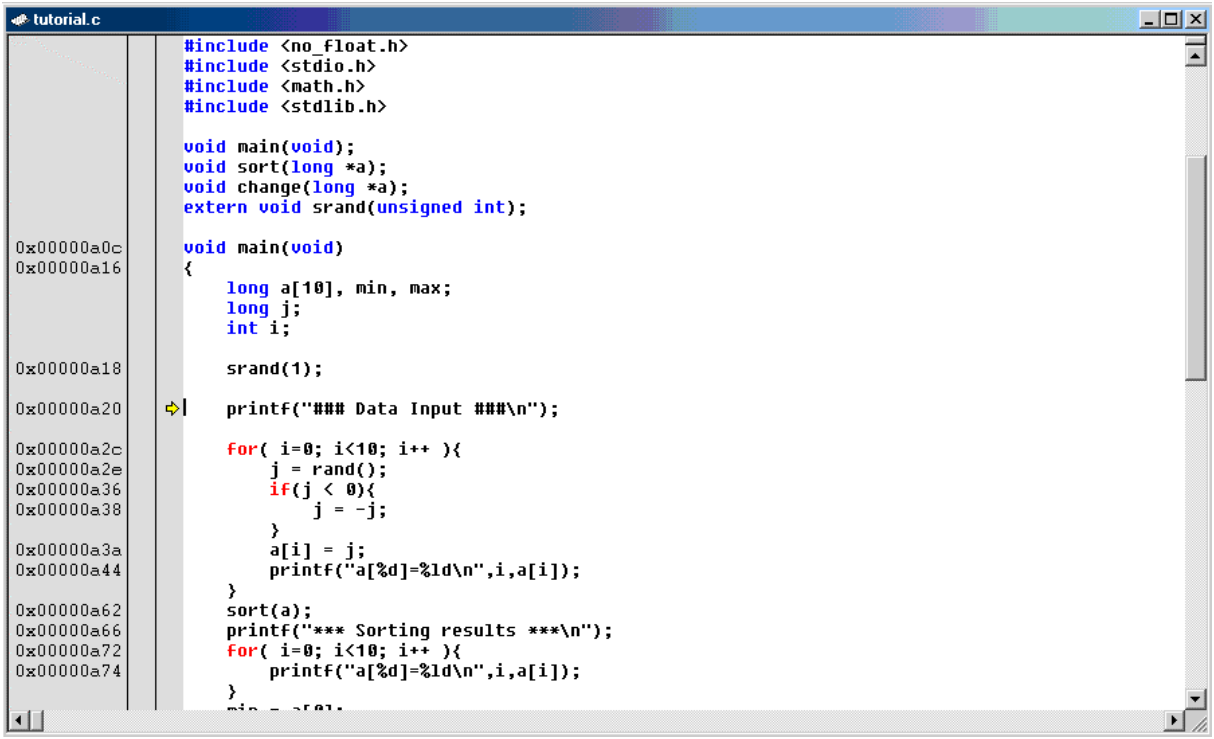


図 3.6: Source **ウィンドウとアドレスカラム**

➲すべてのソースファイルでカラムをオフにするには

1. Editorウィンドウを右クリックしてください。
2. “Define Column Format...”メニュー項目をクリックしてください。
3. Global Editor Column Statesダイアログボックスを表示します。
4. “Check status”チェックボックスは、そのカラムが有効か無効かを示します。チェックしている場合は有効です。チェックボックスがグレー表示の場合、一部のファイルではカラムが有効で、別のファイルでは無効であることを意味します。
5. “OK”ボタンをクリックして、新しいカラム設定を有効にしてください。

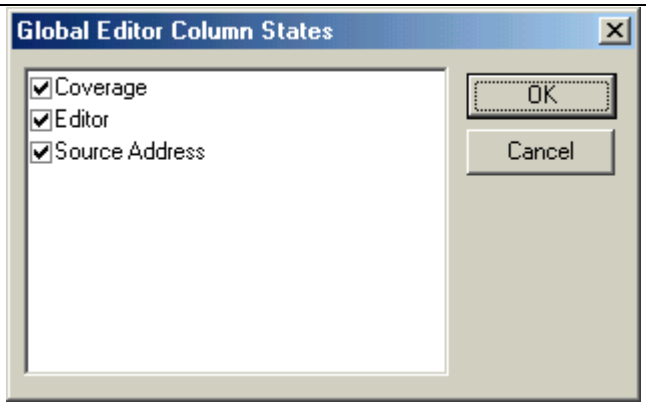


図 3.7: Global column state **ダイアログボックス**

➲1つのソースファイルでカラムをオフにするには

1. 削除したいカラムのあるEditorウィンドウを右クリックしてください。ポップアップメニューを表示します。
2. “Columns”メニュー項目をクリックしてください。カスケードしたメニュー項目が現れます。各カラムを、このポップアップメニューに表示します。カラムが有効である場合、名前の横にチェックマークがあります。エントリをクリックすると、カラムの表示、非表示を切り替えます。

3.9　デバッガカラム

デバッグプラットフォームによりSourceウィンドウに追加することができるカラムが変わります。カラムの例としては”Coverage”カラムがあり、デバッガの実行中に、プログラムカバレージをグラフィカルに表示します。別の例としては”Target”カラムがあり、ターゲットに設定したハードウェアブレークポイントを示します。

カラムを右クリックすると、そのカラムに対するポップアップメニューを表示します。また、カラム上でのダブルクリックは、カラムごとに異なった働きをします。例えば、”Editor”カラム、および”Target”カラムでは、ダブルクリックによりPCあるいはハードウェアブレークポイントを設定します。

4. メモリを操作する

この章では、CPUのアドレス空間におけるメモリ領域の見方を説明します。ここでは、１つのメモリ領域を異なるフォーマットで見る方法、メモリブロックをフィル、移動、およびテストする方法、ならびにディスクファイルを使用してメモリ領域をロードおよびベリファイする方法を説明します。

4.1 メモリ領域を見る

メモリ領域を見るには、Ctrl+Mアクセラレータを使用して[View->Memory…]を選択するか、Memoryウィンドウツールバーボタン  をクリックしてMemoryウィンドウを開きます。これにより、図4.1に示すSet Addressダイアログボックスが開きます。

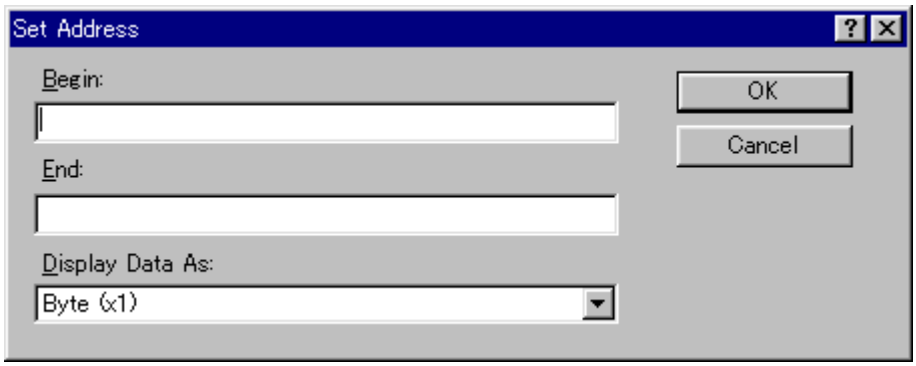


図 4.1: Set Address **ダイアログボックス**

“Address” フィールドに表示したいアドレスの開始アドレスまたは同等のシンボル、および終了アドレスを入力します。”Display Data As”ドロップダウンリストから表示するデータサイズを選択します。”OK”ボタンをクリックするか、ENTERキーを押すとダイアログボックスは閉じてMemoryウィンドウが開きます。入力した表示開始および終了アドレス内でスクロールすることができます。

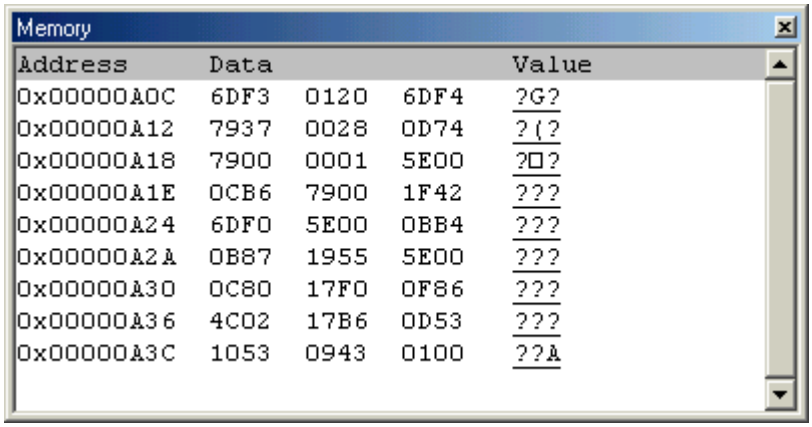


図 4.2: Memory **ウィンドウ**

コラムを3つ表示します。

1. Address　この行のDataコラムの最初のアイテムのアドレス
2. Data　デバッギングプラットフォーム物理メモリからアクセス幅でデータを読み出し、表示幅に変換します。
3. Value　他のフォーマットで表示するデータ

4.2　異なるフォーマットでデータを表示する

Memoryウィンドウの表示フォーマットを変更する場合は、ポップアップメニューから[Format]を選択します。このダイアログボックスを図4.3に示します。

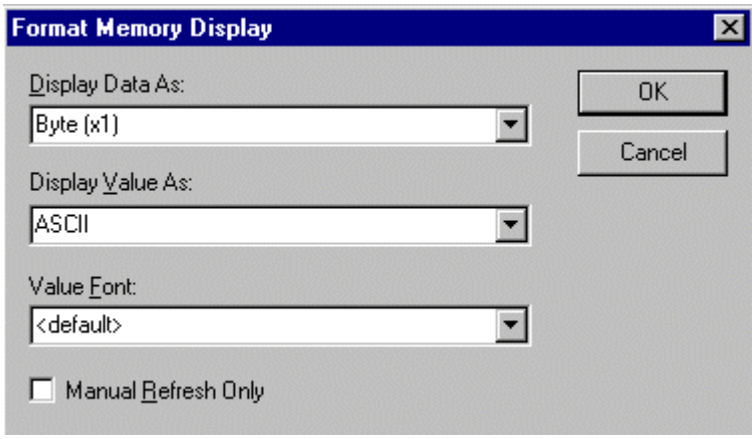


図 4.3: Format **メモリ表示**

メモリを異なる幅で表示して編集するには、”Display Data As”ドロップダウンリストを使用します。例えば、Byteオプションを選択すると、表示を更新して個々のバイトとしてメモリ領域を表示します。

データは、異なるフォーマットに変換することができます。これは3つ目の”Value”コラムに表示します。フォーマットのリストは、データ選択に依存します。

“Value”コラムのフォントを、データ表示に使用するフォントと異なるようにすることができます。これは、データをWordフォーマットで表示したときにdoubleバイト文字値を表示するのに便利です。”<default>”を選択すると、残りのウィンドウと同じように”Value”コラムと同じフォントを使用します。

4.3　異なるメモリ領域を見る

Memoryウィンドウの表示するメモリ領域を変更したいときは、スクロールバーを使用します。新しいアドレスをすぐに見たいときには、Set Addressダイアログボックスを使用します。これは、ポップアップメニューから[Start Address]を選択または”Address”コラムをダブルクリックすることによって開くことができます。

新しいアドレスを入力して”OK”ボタンをクリックするかENTERキーを押します。ダイアログボックスは閉じ、Memoryウィンドウの表示が新しいアドレスのデータに更新します。オーバーロード関数またはクラス名を入力すると、Select Functionダイアログボックスが開くので、関数を選択します。

4.4　メモリの内容を修正する

メモリ内容は、Editダイアログボックスで変更します。変更したいメモリユニット上にカーソルを移動します（Memoryウィンドウ表示選択にしたがって）。メモリユニットをダブルクリックするか、ENTERキーを押します。図4.4のダイアログボックスを表示します。

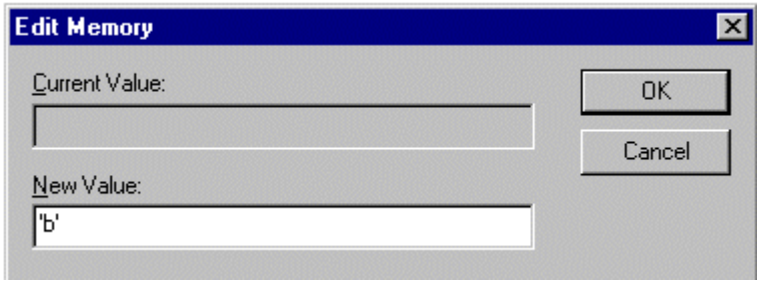


図 4.4: Edit Memory **ダイアログボックス**

また、メモリユニット上にマウスカーソルを移動し、キーボードより16進数を入力することにより、メモリの内容を変更することもできます。
“New Value”フィールドには数字またはC/C++の式を入力することができます。新しい数字または式を入力したら、”OK”ボタンをクリックまたはENTERキーを押すと、新しい値をメモリに書き込みます。

4.4.1　メモリ範囲を選択する

メモリアドレス範囲がMemoryウィンドウにある場合、最初のメモリユニット（Memoryウィンドウディスプレイでの選択に従い）をクリックして最後のユニットまでマウスをドラッグすることによって領域を選択することができます。選択した領域はハイライト表示します。

メモリアドレス範囲がMemoryウィンドウよりも大きい場合、またはMemoryウィンドウにはない場合、Memoryダイアログボックスのそれぞれのフィールドに開始アドレスおよびバイトカウントを入力することができます。

4.4.2　メモリ内の値を探す

メモリ内の値を探すには、Memoryウィンドウを開き、ポップアップメニューから[Search]を選択しなければなりません。あるいはMemoryウィンドウがすでに開いているときは、単にF3キーを押します。
図4.5に示すように、Search Memoryダイアログボックスを表示します。

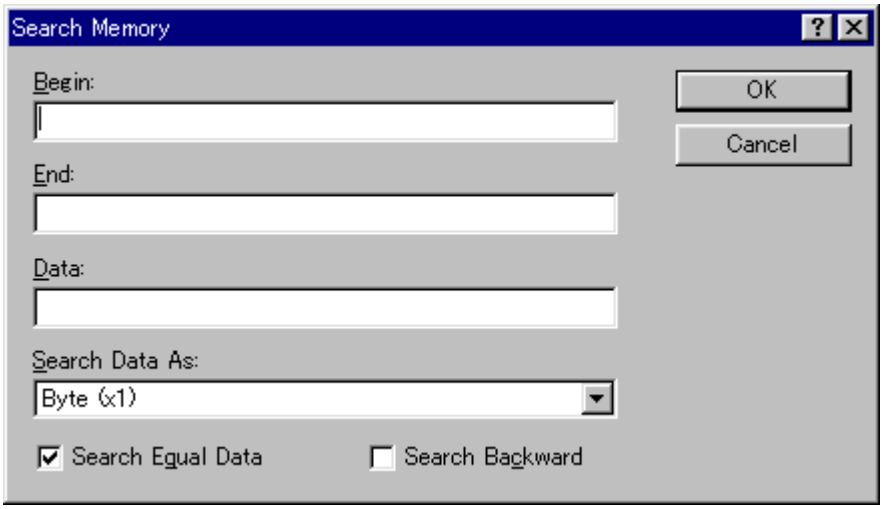


図 4.5: Search Memory **ダイアログボックス**

検索するアドレス範囲の開始および終了アドレス（Memoryウィンドウの中のメモリ領域を選択している場合には、開始および終了アドレス値を自動的にフィルします）、および検索するデータ値を入力します。また、検索条件として、一致／不一致、検索方向を指定できます。終了アドレスには先頭に'+'記号をつけることができ、この記号を入力すると、入力した値を範囲として使用します。（先頭アドレス＋入力した値が終了アドレスになります）

検索フォーマットを選択し、”OK”ボタンをクリックするかENTERキーを押します(デフォルトは、data displayフォーマットです)。ダイアログボックスは閉じてHEWは指定したデータの領域を検索します。データが見つかると、Memoryウィンドウにハイライト表示し、データが見つかったアドレスをStatusバーにメッセージとして表示します。

データを見つけることができなかった場合、Memoryウィンドウのカーソルの位置は以前と変わらず、データを見つけることができなかったことを知らせるメッセージをステータスバーに表示します。

4.5　メモリ範囲に値をフィルする

Memory fill機能を使って値をメモリアドレス範囲の内容に設定することができます。

4.5.1　範囲にフィルする

同じ値でメモリ範囲に入れるには、Memory ウィンドウのポップアップメニューの[Fill]を選択するか、Memory ドロップダウンメニューの[Fill]を選択します。Fill Memoryダイアログボックスを図4.6に示します。

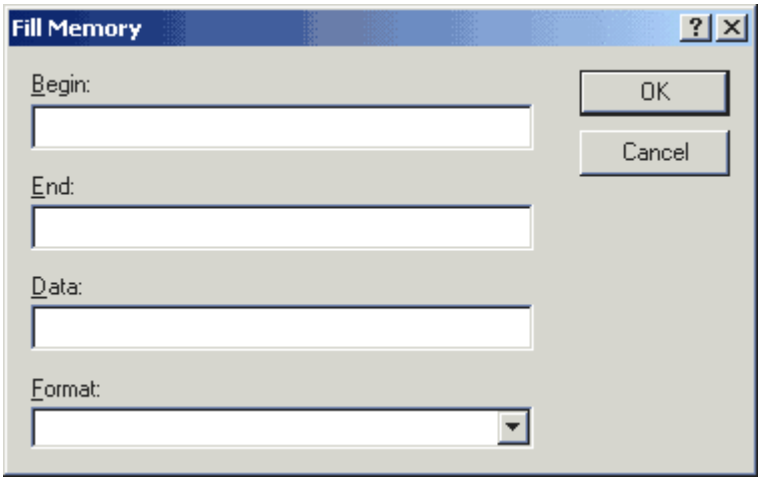


図 4.6: Fill Memory **ダイアログボックス**

Memoryウィンドウでアドレス範囲を選択した場合には、指定した開始および終了アドレスを表示します。”Format”ドロップダウンリストからフォーマットを選択して”Data”フィールドにデータ値を入力します。”OK”ボタンをクリックするかENTERキーを押すと、ダイアログボックスが閉じて新しい値をメモリ領域に書き込みます。

4.6　メモリ領域をコピーする

メモリコピー機能を使用してメモリ領域をコピーすることができます。メモリ領域を選択してポップアップメニューから[Copy…]を選択すると、Copy Memoryダイアログボックスを表示します。（図4.7）

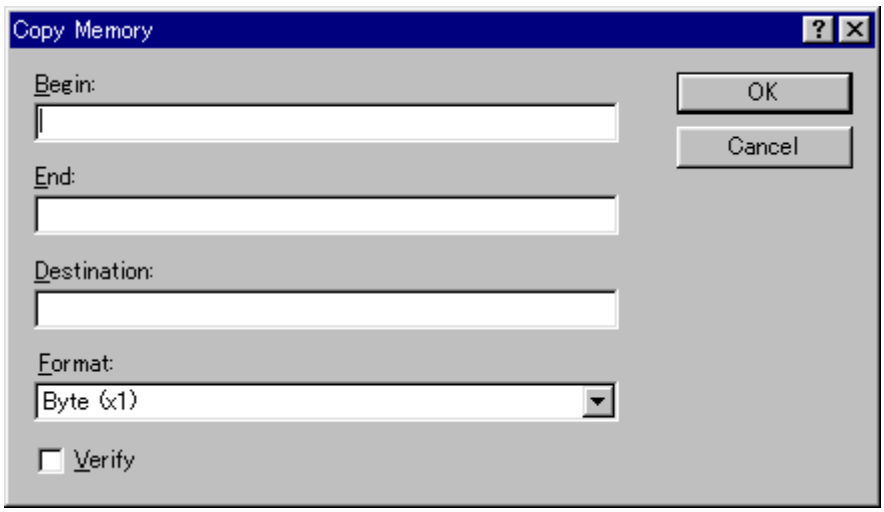


図 4.7: Copy Memory **ダイアログボックス**

Memoryウィンドウで選択したコピー元の開始アドレスおよび終了アドレスは、”Begin”および”End”フィールドに表示します。”Verify”チェックボックスをチェックすることによりコピー元とコピー先を比較しながらコピーすることもできます。”Format”リストボックスでコピー単位を選択することもできます。コピー先の開始アドレスを”Destination”フィールドに入力して”OK”ボタンをクリックするか、ENTERキーを押すと、ダイアログボックスを閉じてメモリブロックを新しいアドレスにコピーします。

4.7　メモリ領域をテストする

注意　メモリ領域のテストを行なう場合、メモリの現在の内容を上書きします。作成したプログラムまたはデータを消去しますのご注意ください。
なお、メモリテスト機能は、選択したデバッグプラットフォームによってはサポートしていない場合があります。

メモリテスト機能を使用してアドレス空間のメモリ領域をテストすることができます。メモリを選択してポップアップメニューから[Test]を選択します。Test Memoryダイアログボックスを表示します。（図4.8）

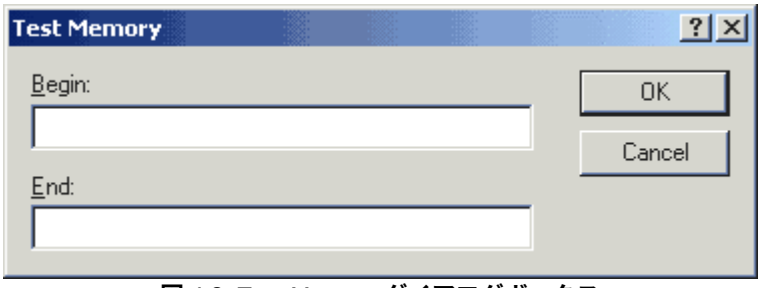


図 4.8: Test Memory **ダイアログボックス**

Memoryウィンドウに指定した開始および終了アドレスは、”Start”および”End”フィールドに表示します。”OK”ボタンをクリックするかENTERキーを押すと、ダイアログボックスを閉じて、HEWはメモリ領域をテストします。

4.8　メモリ領域を保存、検証する

メモリ保存機能を使用してアドレス空間のメモリ領域をディスクファイルに保存することができます。[File->Save Memory...]を選択してSave Memory Asダイアログボックスを開きます。

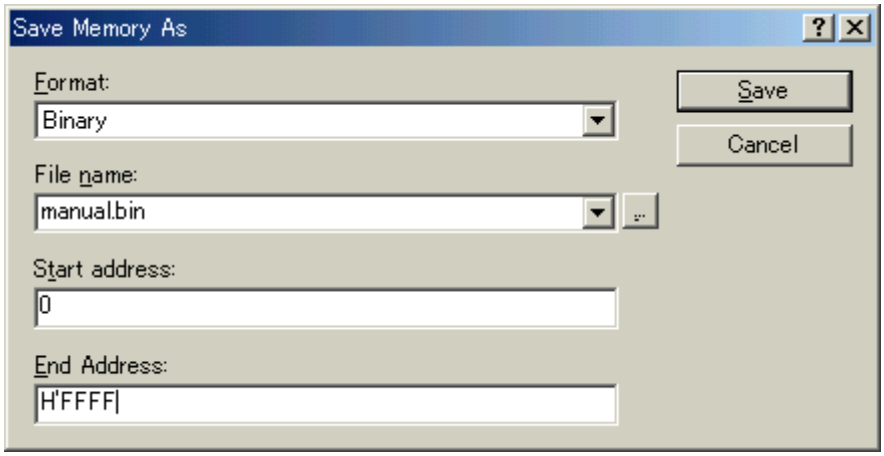


図 4.9: Save Memory As **ダイアログボックス**

保存するメモリブロックの開始および終了アドレス、ファイル名、ファイルフォーマットを入力します。“File name”ドロップダウンリストには、メモリを保存するために使用した過去4つのファイル名を表示します。

また“Browse…”ボタンをクリックすると、標準のFile Save Asダイアログボックスを開きます。“OK”ボタンをクリックするかENTERキーを押すと、ダイアログボックスを閉じて、メモリブロックをモトローラSレコードフォーマットファイルとしてディスクに保存します。ファイルの保存が完了すると、確認のメッセージボックスを表示することがあります。

メモリベリファイ機能を使用してアドレス空間のメモリ領域を検証することができます。[File->Verify Memory...]を選択してVerify Memoryダイアログボックスを開きます。

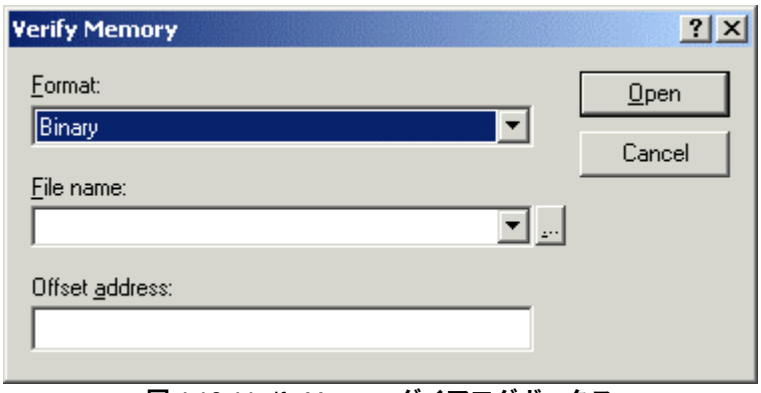


図 4.10: Verify Memory **ダイアログボックス**

4.9　I/O メモリを見る

CPUおよびROM/RAMと同様、マイクロコントローラには内蔵周辺モジュールがあります。デバイスによって周辺モジュールの数および型は異なりますが、代表的なモジュールとしては、DMAコントローラ、シリアルコミュニケーションインタフェース、A/Dコンバータ、インテグレーテッドタイマユニット、バスステートコントローラおよびウォッチドッグタイマなどがあります。マイクロコントローラのアドレス空間にマッピングしたアクセスレジスタは、内蔵周辺モジュールを制御します。

Memoryウィンドウは、メモリ内のデータをバイト、ワード、ロングワード、単精度浮動小数点、倍精度浮動小数点、またはASCII値として表示することができるので、HEWはIOウィンドウを提供しこれらのレジスタを簡単に確認したり設定したりすることができます。

4.9.1　IO ウィンドウを開く

IOウィンドウを開くには、[View->IO]を選択するか、IOウィンドウツールバーボタン  をクリックします。内蔵周辺と一致するモジュールがI/Oレジスタ情報を構成します。IOウィンドウを最初に開くと、モジュール名の一覧表のみを表示します。

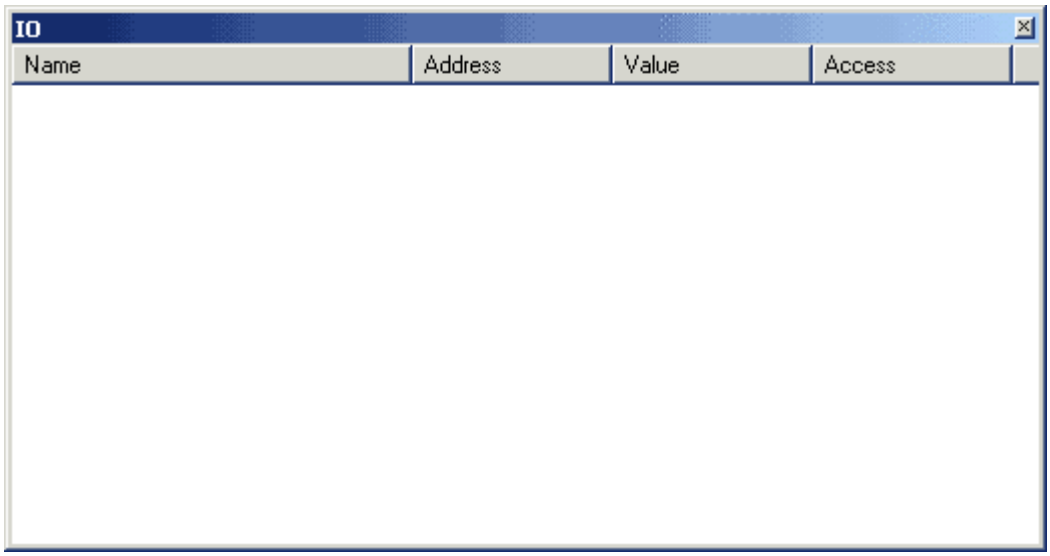


図 4.11: IO **ウィンドウ**

4.9.2　I/O Register 表示を拡張する

I/Oレジスタの名前、アドレス、および値を表示するには、モジュール名をダブルクリックするか、クリックまたはカーソルを使用することによってモジュール名を選択し、ENTERキーを押します。モジュールの表示が拡張し、その周辺モジュールのそれぞれのレジスタおよびその名前、アドレス、および値を表示します。モジュール名を再びダブルクリックする（またはENTERキーを押す）と、表示しているI/Oレジスタを閉じます。

ビットレベルで表示するには、Registersウィンドウと同様のやり方でI/Oレジスタを拡張します。

ビットのカラーコードは以下の通りです。

黒	通常の読み出し/書き込み
赤	値の変更
灰色	周辺モジュール使用不可(周辺コントロールレジスタによる)

4.9.3　I/O レジスタの内容を修正する

I/Oレジスタの値を編集するには、ウィンドウに対して16進数を直接入力します。さらに複雑な式を入力するには、レジスタをダブルクリックするかENTERキーを押してレジスタの内容を修正するためのダイアログボックスを開きます。新しい数字または式を入力したら、”OK”ボタンをクリックするかENTERキーを押します。ダイアログボックスは閉じて新しい値をレジスタに書き込みます。

5. プログラムを実行する

この章では、作成したプログラムコードの実行方法について説明します。ここで学ぶ実行方法は、プログラムを連続して実行させたり、シングルステップ実行を行ったり、同時に複数の命令を実行させたりすることです。

5.1 リセットから実行を開始する

ユーザシステムをリセットして*Reset Vector*アドレスから作成したプログラムを実行させるには、[Debug->Reset Go]を選択するか、Reset Goツールバーボタン  をクリックします。

プログラムは、ブレークポイントにヒットするまで、またはブレーク条件が成立するまで実行を続けます。プログラムの実行はいつでも手動で停止することができます。その方法としては、[Debug->Halt]を選択するかHaltツールバーボタン  をクリックします。

注意　プログラムはリセットベクタ位置に格納したアドレスから実行を開始します。したがって、この位置に自分のスタートアップコードのアドレスを含んでいることを確認することが重要です。

5.2 実行を継続する

作成したプログラムが停止し、デバッガが*Break mode*に入ると、HEWは、CPUの現在のプログラムカウンタ（PC）アドレス値に対応するエディタおよびDisassemblyウィンドウの行の左余白に黄色の矢印を表示します。ステップ実行を行った場合、または実行を続けた場合、この命令を次に実行します。

現在のPCアドレスから実行を継続するには、Continueツールバーボタン[]をクリックするか、[Debug->Go]を選択します。

5.3 カーソルまで実行する

アプリケーションを実行している途中で、シングルステップ実行を複数回行うだけの比較的小さいセクションコードのみを実行したいと考える場合があります。これは、Go To Cursor機能を使用して行うことができます。

➲ Go To Cursorを使用するには

1. SourceまたはDisassemblyウィンドウが開いていて、プログラムを停止するアドレスを表示していることを確認します。
2. アドレスフィールドをクリックするかカーソルキーを使用してプログラムを停止するアドレス上にテキストカーソルを置きます。
3. ポップアップメニューから[Go To Cursor]を選択します。

デバッギングプラットフォームは作成したコードを現在のPC値から実行し、カーソル位置が示すアドレスまで実行します。

注意

1. **作成したプログラムがこのアドレスのコードを決して実行しない場合、プログラムは停止しません。その場合、コードの実行を中止するには、Esc キーを押すか、[Debug ->Halt]を選択するか、Stop ツールバーボタン  をクリックします。**
2. **Go To Cursor 機能は、テンポラリーブレークポイントを必要とします。そのため、すでにその機能を使用している場合には、この機能は動作せず、メニューオプションを使用することができません。**

5.4　シングルステップ

作成したコードをデバッグするために、一度に一行だけまたは一つの命令だけステップ実行して、この命令がシステムにどのように影響するかを確認したい場合、Sourceウィンドウでは、ソースライン一行だけをステップ実行します。Disassemblyウィンドウにおいては、アセンブリ言語命令単位にステップ実行します。命令が他の関数またはサブルーチンをコールした場合、オプションでその関数にステップインまたはステップオーバーすることができます。その命令コールを行わない場合には、いずれのオプションでも、デバッガに命令を実行させ、次の命令で停止させることができます。

5.4.1　関数にステップイン実行する

関数にステップイン実行することを選択した場合にはデバッガは関数の行または命令でコールを実行します。関数にステップインするには、Step Inツールバーボタンをクリックするか、[Debug->Step In]を選択します。

5.4.2　関数コールをステップオーバー実行する

関数をステップオーバー実行することを選択した場合には、デバッガはコールおよび関数内のすべてのコード(および関数が行う可能性のある関数コールのすべて)を実行して、呼び出し元の関数の次の行または命令で停止します。関数をステップオーバーするには、Step Overツールバーボタンをクリックするか、[Debug ->Step Over]を選択します。

5.4.3　関数からステップアウト実行する

関数内の確認したい命令の実行が終了した場合や、誤って関数にステップインした場合に、Step Out機能を使用すると関数内の残りのコードをステップ実行せずに呼び出し元の関数に戻ることができます。

現在の関数からステップアウトするには、Step Outツールバーボタンをクリックするか、[Debug ->Step Out]を選択します。

5.5　複数のステップ

Step Programダイアログボックスを使用することにより、一度に複数のステップ実行ができます。このダイアログボックスでは、ステップ間の時間差を選択し、ステップ実行を自動的に行なうよう設定できます。このダイアログボックスは、[Debug -> Step...]を選択して開きます。

Step Programダイアログボックスを表示します。

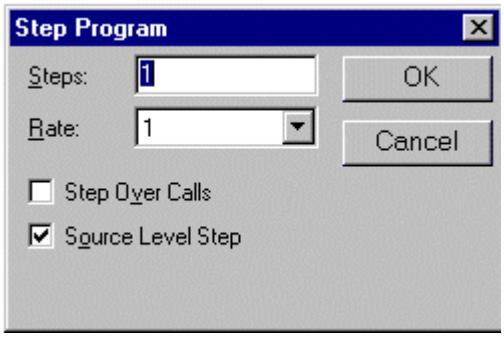


図 5.1: Step Program **ダイアログボックス**

“Steps”フィールドにステップ数を入力して、”Step Over Calls”チェックボックスを使用して関数コールをステップオーバーするかどうかを選択します。自動ステップ機能を使用している場合には、”Rate”フィールドのリストからステップレートを選択します。”OK”ボタンをクリックするか、ENTERキーを押してステップ実行を開始します。

6. プログラムを停止する

この章では、作成したプログラムの実行を停止する方法を説明します。停止手段として、Haltコマンドを使用して停止する方法、および作成したコードの特定の場所にブレークポイントを設定することによって停止する方法について説明します。

6.1 Halt による停止

作成したプログラムが実行中の場合、Haltツールバーボタン  (赤い停止の印)を使用することができます。しかし、プログラムが停止している場合は、使用できません  (STOPの印が灰色になります)。プログラムを停止させるには、Haltツールバーボタンをクリックするか、[Debug->Halt Program]を選択することによりプログラムが停止します。

Haltによりプログラムが停止したとき、Outputウィンドウの”Debug”タブに”Stop”というメッセージを表示します。

6.2 標準のブレークポイント(PC ブレークポイント)

作成したプログラムをデバッグする場合、PCブレークポイントにより指定した行または命令でプログラムの実行を停止させることができます。標準的なPCブレークポイントを設定したり解除したりする方法を以下に示します。より複雑な設定をする場合には、Breakpointsウィンドウを使用します。Breakpointsウィンドウは  ボタンをクリックするか、[View->Breakpoints]を選択することにより表示します。

➲PCブレークポイントを設定するには

1. PCブレークポイントを設定する位置のDisassembleまたはSourceウィンドウが開いていることを確認します。
2. プログラムを停止したい行でポップアップメニューの[Toggle Breakpoint] を選択するか、F9キーを押すかします。
3. 左余白に赤丸を表示します。これは、PC ブレークポイントを設定したことを示します。
4. [Edit->Source Breakpoints]を選択することによって表示するBreakpointsダイアログボックスにより、現在設定しているブレークポイントの有効／無効の切り替えおよび削除ができます。

作成したプログラムを実行して PCブレークポイントを設定したアドレスに達すると、Outputウィンドウの”Debug”タブに" PC Breakpoint"というメッセージを表示し、実行を停止し、SourceまたはDisassemblyウィンドウを更新し、停止位置を左余白に矢印で表示します。

注意　ブレーク発生時には、PC ブレークポイントを設定した行または命令を実行する直前で停止します。その PC ブレークポイントで停止した後に Go または Step を選択した場合、矢印で表示した行から実行します。

6.2.1　Breakpoint ダイアログボックスを使用する

Breakpointダイアログボックスを図6.1に示します。このダイアログボックスに現在設定しているブレークポイントを表示します。”Edit Code”ボタンによりブレークポイントが存在するソースを見ることができます。“Remove”、”Remove All”ボタンによりプレークポイントを１つまたはすべてを削除することができます。また、各ブレークポイントのチェックボックスにより有効／無効の切り替えができます。

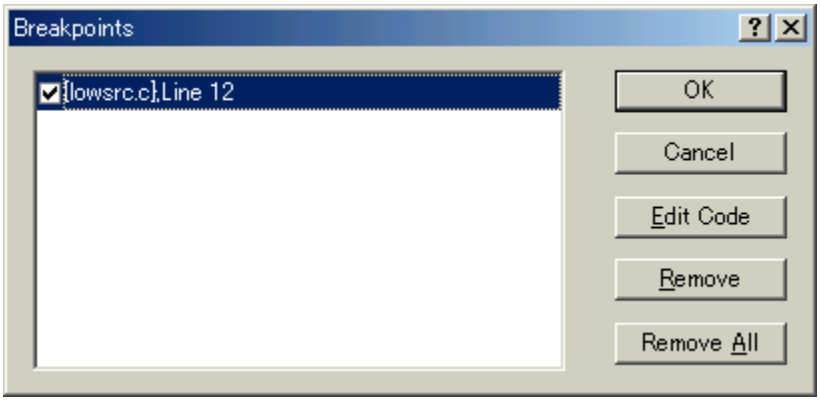


図 6.1: Breakpoints **ダイアログボックス**

6.2.2　PC ブレークポイントを切り替える

PCブレークポイントを設定している行のBPカラムをダブルクリックするか、その行にカーソルを置いてF9キーを使用すると、PC Breakpointsの設定を切り替えることができます。切り替わる設定内容はデバッグプラットフォームによって異なります。

7. Elf/Dwarf2 のサポート

HEWは、C/C++およびアセンブリ言語で書いたアプリケーションのデバッグのためにElf/Dwarf2オブジェクトファイルフォーマットをサポートします。Elf/Dwarf2オブジェクトファイルフォーマットを使用すると、実行中のアプリケーションに関するシンボルデバッグ情報を強力にアクセス、および修正できます。

主な特長

- ソースレベルデバッグ
- C/C++演算子
- C/C++式(キャスト、ポインタ、参照)
- あいまいな関数名
- オーバーレイメモリロード
- ウオッチ-ローカル、およびユーザ定義
- スタックトレース

7.1 C/C++演算子

以下のC/C++ 言語演算子を使用することができます。

```
+, -, *, /, &, |, ^, ~, !, >>, <<, %, (, ), <, >, <=, >=, ==, !=, &&, ||
Buffer_start + 0x1000
#R1 | B'10001101
((pointer + (2 * increment_size)) & H'FFFF0000) >> D'15
!(flag ^ #ER4)
```

7.2 C/C++の式

式の例

```
Object.value                        //メンバの直接参照を指定します(C/C++)
p_Object->value                     //メンバの間接参照を指定します(C/C++)
Class::value                        //クラスを持つメンバの参照を指定します(C++)
*value                              //ポインタを指定します(C/C++)
&value                              //参照を指定します(C/C++)
array[0]                            //アレイを指定します(C/C++)
Object.*value                       //ポインタを持つメンバの参照を指定します(C++)
::g_value                           //グローバル変数の参照を指定します(C/C++)
Class::function(short)              //メンバ関数を指定します(C++)
(struct STR) *value                 //キャスト動作を指定します(C/C++)
```

7.3　複数ラベルをサポートする

プログラム言語の中の、例えばC++オーバーロード関数などでは、1つのラベルが複数のアドレスを表す場合があります。各ダイアログボックスでこのようなラベル名を入力した場合、HEWはSelect Functionダイアログボックスを使用してオーバーロード関数およびメンバ関数を表示します。

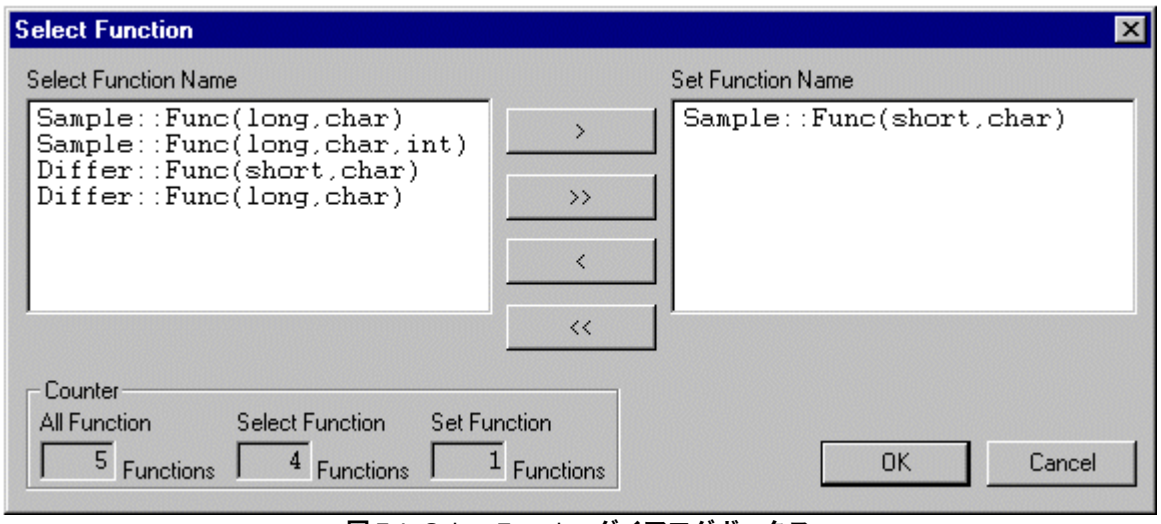


図 7.1: Select Function **ダイアログボックス**

Select Functionダイアログボックスでは、オーバーロード関数またはメンバ関数を選択します。通常、一度に一つの関数を選択します。ただし、ブレークポイントを設定する場合においてのみ、複数の関数を選択することができます。このダイアログボックスには3つの領域があります。

- “Select Function Name”リストボックス
  同じ名前をもつ関数またはメンバ関数、およびその詳細情報を表示します。
- “Set Function Name”リストボックス
  設定する関数およびそれらの詳細情報を表示します。
- “Counter group”エディットボックス

– All Function　同じ名前をもつ関数またはメンバ関数を表示します。
– Select Function　“Select Function Name” リストボックスに表示する関数の数を表示します。
– Set Function　“Set Function Name”リストボックスに表示する関数の数を表示します。

7.4　関数を選択する

“Select Function Name”リストボックスから選択したい関数をクリックして、**”>”**ボタンをクリックします。選択した関数を”Set Function Name”リストボックスに表示します。”Select Function Name”リストの関数すべてを選択するには、**”≫”**ボタンをクリックします。

7.5　関数の選択を解除する

“Set Function Name”リストボックスから選択を解除する関数をクリックして、**”<”**ボタンをクリックします。すべての関数の選択を解除するには、**”≪”**ボタンをクリックします。選択を解除した関数は、”Set Function Name”リストボックスから”Select Function Name”リストボックスへ戻します。

7.6　関数を設定する

“OK”ボタンをクリックして、”Set Function Name”リストボックスに表示した関数を設定します。関数を設定し、Select Functionダイアログボックスを閉じます。

“Cancel”ボタンをクリックすると、関数を設定せずにダイアログボックスを閉じます。

7.7　オーバーレイ関数

オーバーレイ関数を使用してプログラムをデバッグすることができます。この章では、オーバーレイ関数を使用するための設定について説明します。

7.8　セクショングループを表示する

オーバレイ関数（いくつかのセクショングループを同じアドレス範囲に割り当てる）を使用すると、アドレス範囲およびセクショングループをOverlayダイアログボックスに表示します。
[Memory->Configure Overlay]を選択してOverlayダイアログボックスを開きます。

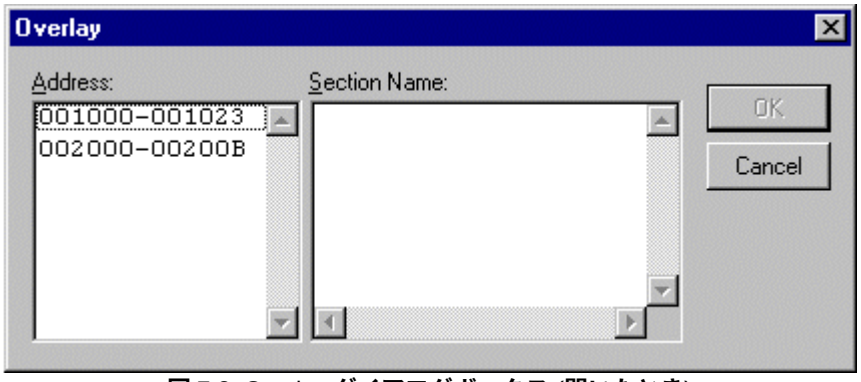


図 7.2: Overlay **ダイアログボックス (開いたとき)**

このダイアログボックスには2つの領域があります。”Address”リストボックスおよび”Section Name”リストボックスです。

“Address”リストボックスは、オーバーレイ関数が使用するアドレス範囲を表示します。アドレス範囲の１つをクリックして”Address”リストボックスのアドレス範囲を選択します。

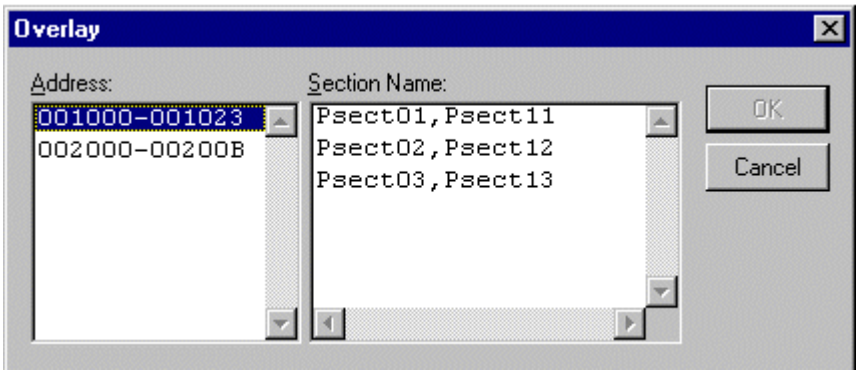


図 7.3: Overlay **ダイアログボックス (アドレス範囲を選択)**

“Section Name”リストボックスは、選択したアドレス範囲に割り当てたセクショングループを表示します。

➲セクショングループを設定するには
オーバーレイ関数を使用するときは、最も優先度の高いセクショングループをOverlayダイアログボックスで選択していなければ、HEWは正しく動作しません。
まず”Address”リストボックスに表示したアドレス範囲の一つをクリックします。選択したアドレス範囲に割り当てたセクショングループを”Section Name”リストボックスに表示します。
表示しているセクショングループの中から最も優先度の高いセクショングループをクリックします。

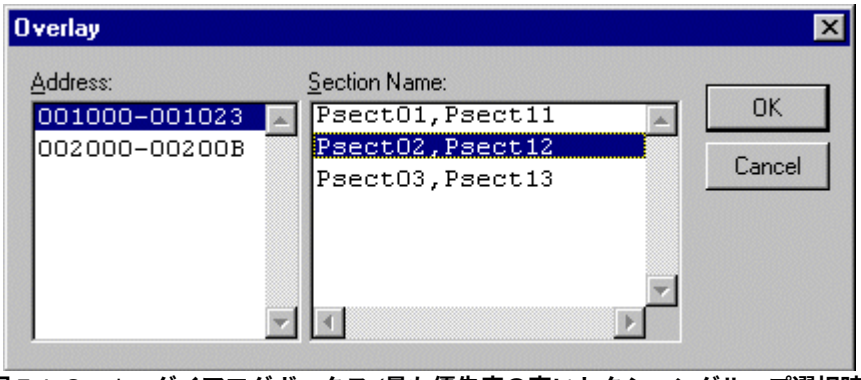


図 7.4: Overlay **ダイアログボックス (最も優先度の高いセクショングループ選択時)**

セクショングループを選択したら、”OK”ボタンをクリックして優先度の設定を保存して、ダイアログボックスを閉じます。”Cancel”ボタンをクリックすると、優先度の設定を保存せずにダイアログボックスを閉じます。

注意　オーバーレイ関数が使用するアドレス範囲内では Overlay **ダイアログボックスに指定したセクションのデバッグ情報を参照します。したがって、現在ロードしているプログラムと同じセクションを** Overlay **ダイアログボックスで選択しなければなりません。**

7.9　Tooltip Watch

作成したプログラムの変数を最もすばやく見るには、Tooltip Watch 機能を使用します。

➲Tooltip Watch を使用するには
確認したい変数を表示しているSourceウィンドウを開きます。
確認したい変数名の上にマウスのカーソルを静止させます。変数の近くにツールチップを表示し、その変数の基本的なWatch情報を表示します。

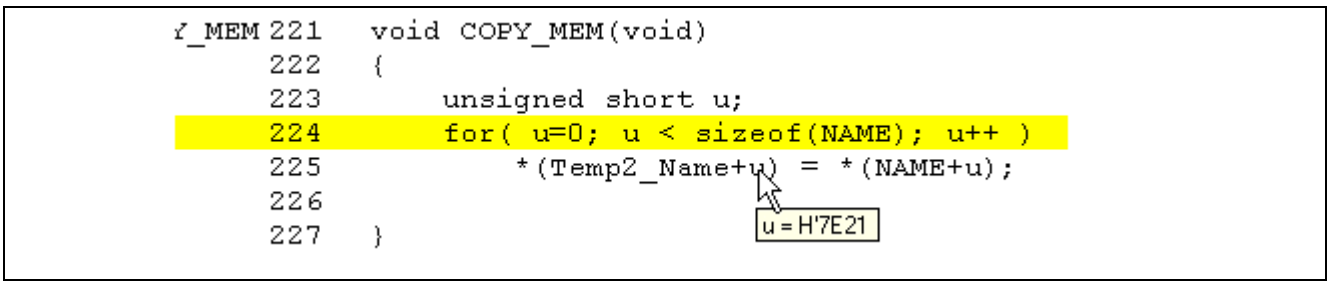


図 7.5: Tooltip Watch

7.10　ローカル変数

ローカル変数を見るには、[View->Locals]を選択してLocalsウィンドウを開きます。

Locals ウィンドウが開きます。

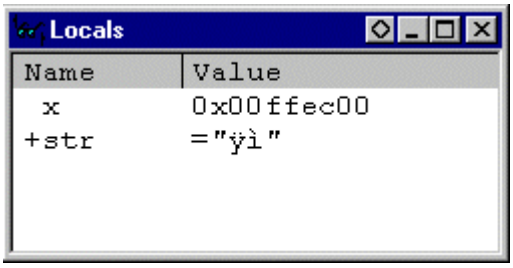


図 7.6: Locals ウィンドウ

プログラムをデバッグしていくに従い、Localsウィンドウを更新します。ローカル変数を定義した時点で初期化していないと、ローカル変数に値を代入するまでLocalsウィンドウの値を定義しません。

ローカル変数の値およびローカル変数の表示は、Watchウィンドウと同じ方法で修正することができます。

7.11　Watch アイテム

HEWでは、Watchウィンドウを開くことにより、変数の一覧とそれらの値を参照することができます。Watchウィンドウを開くには、[View->Watch]を選択するか、Watchウィンドウツールバーボタン が使用可能であれば、クリックします。Watchウィンドウが開きます。ウィンドウの中は最初空白です。

7.11.1　Watch を追加する

WatchアイテムをWatchウィンドウに追加するには、WatchウィンドウのAdd Watchダイアログボックスを使用します。

➲ WatchウィンドウからAdd Watch を使用するには
Watchウィンドウを開きます。
ポップアップメニューから[Add Watch]を選択します。
Add Watchダイアログボックスが開きます。

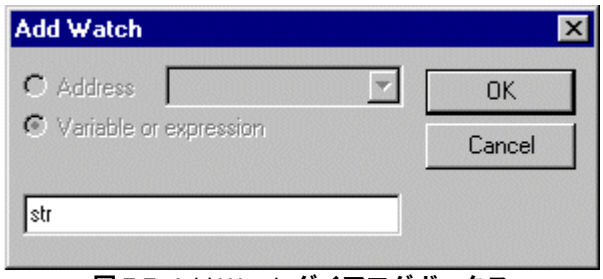


図 7.7: Add Watch ダイアログボックス

見たい変数名を入力して、”OK”ボタンをクリックします。その変数をWatchウィンドウに追加します。

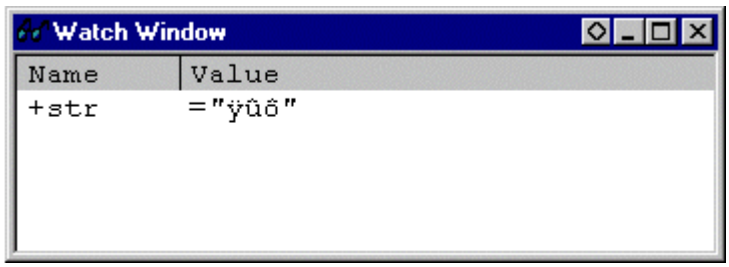


図 7.8: Watch ウィンドウ

注意　追加した変数がローカル変数で現在範囲外の場合には、HEW はその変数を Watch ウィンドウに追加しますが、その値は空白または、クエスチョンマーク'?'を表示します。

7.11.2　Watch を拡張する

Watchアイテムがポインタ、アレイ、または構造体のとき、その名前の左側にプラス記号(+)の拡張指示子を表示します。つまり、Watchアイテムを拡張できるという意味です。Watchアイテムを拡張するには、プラス記号(+)をダブルクリックします。１つのタブによってインデントをつけたアイテムは拡張し、その要素(構造体またはアレイの場合)またはデータ値(ポインタの場合)を表示し、プラス記号がマイナス記号に変わります。Watchアイテムが要素にポインタ、構造体、またはアレイを含む場合、その横に拡張指示子を表示します。

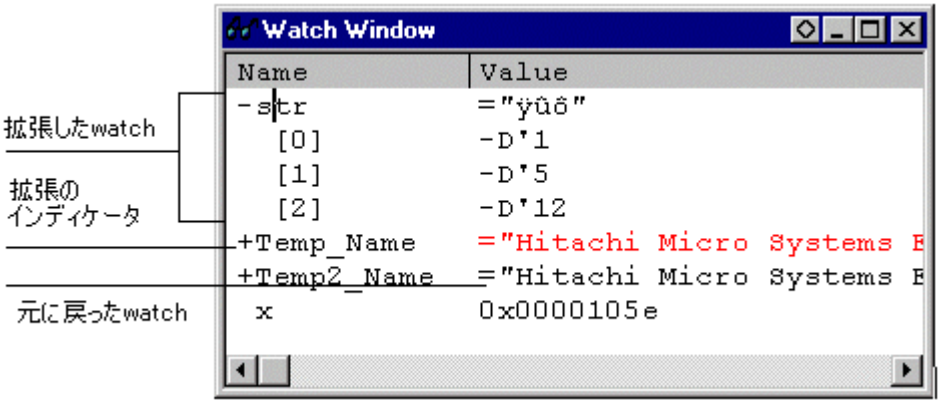


図 7.9: Watch を拡張する

拡張したWatchアイテムを元に戻すには、再びアイテムをダブルクリックします。アイテムの要素は、元の単一のアイテムに戻り、マイナス記号はプラス記号に戻ります。

7.11.3　Watchアイテムの値を編集する

テストのためや、プログラムにバグがあるために値が正しくないときには、Watch変数の値を変更することができます。Watchアイテムの値を変更するには、Edit Value機能を使用します。

➲ Watchアイテムの値を編集するには
編集するアイテムをクリックにより選択すると、アイテム上のカーソルが点滅します。
ポップアップメニューから[Edit Value]を選択します。
Edit Valueダイアログボックスオプションが開きます。

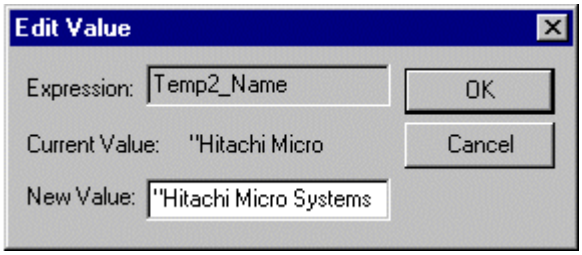


図 7.10: Edit Value **ダイアログボックス**

“New Value”フィールドに新しい値はまたは式を入力して”OK”ボタンをクリックします。Watchウィンドウを更新し、新しい値を表示します。

➲Watchを削除するには
Watchアイテムを削除するには、そのアイテムを選択してポップアップメニューから[Delete]を選択します。アイテムを削除し、Watchウィンドウを更新します。

8. プロファイル情報を見る

プロファイル機能は、アプリケーションプログラムの実行パフォーマンスを関数単位に測定します。アプリケーションプログラム中の性能劣化の原因となっている場所および要因を調査することができます。

HEWはプロファイルデータの参照方法、参照目的に応じて、３つのウィンドウでプロファイル測定結果を表示します。

8.1 スタック情報ファイル

プロファイル機能は、HITACHI最適化リンカ(Ver.7.0以降)が出力するスタック情報ファイル（拡張子".SNI"）を読み込むことができます。このファイルには、ソースファイル上の(静的な)関数呼び出し関係の情報が入っています。HEWがスタック情報ファイルを読み込むことで、ユーザアプリケーションが未実行（プロファイルデータの測定を行なう前）でも、関数の呼び出し関係を表示できるようになります。(但し、Profileウィンドウのポップアップメニューで[Setting->Only Executed Functions]をチェックしている場合を除きます。)

HEWがスタック情報ファイルを読み込まない場合、プロファイル機能で表示するデータは、プロファイルデータ測定中に実行した関数についてのみになります。

リンカでスタック情報ファイルを生成するには、Standard Toolchainダイアログボックスの"Link/Library"タブで"Category:"リストボックスを"Other"に指定し、"Stack information output"チェックボックスをチェックしてください。

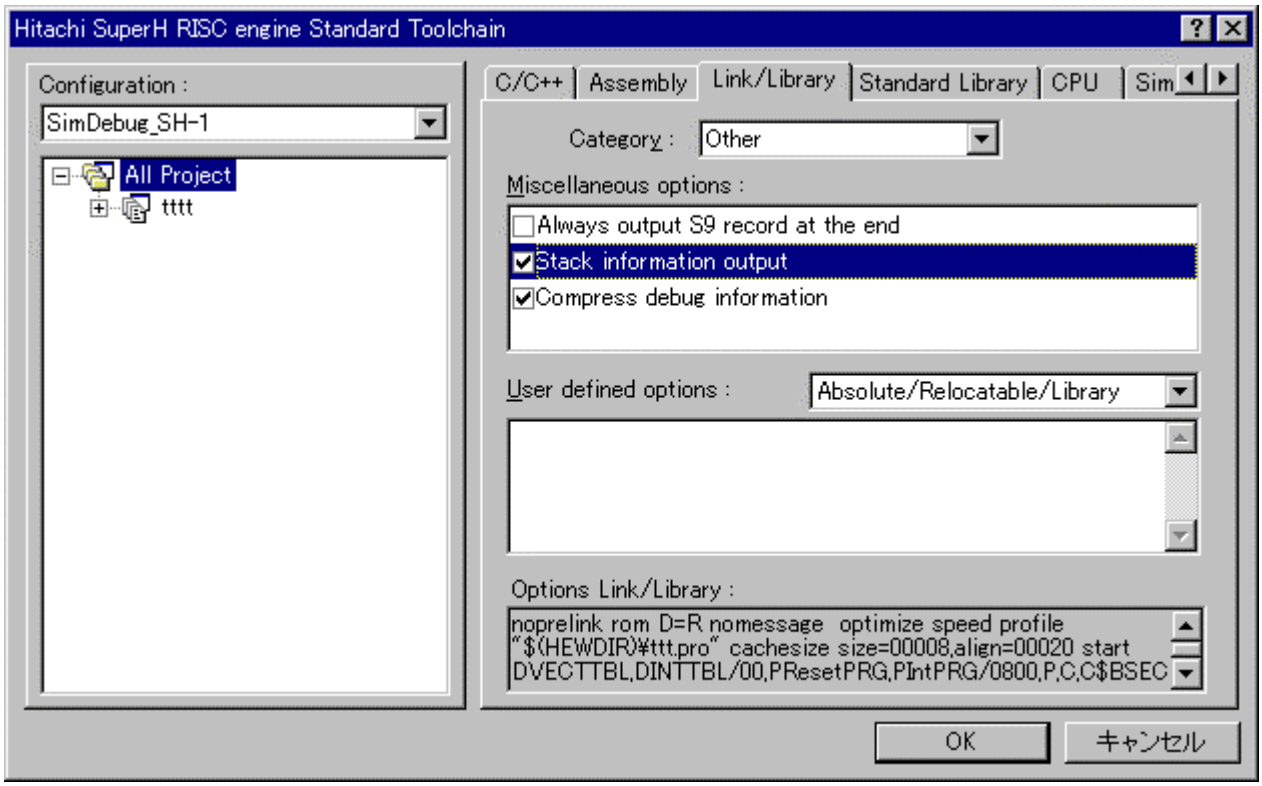


図 8.1: Standard Toolchain **ダイアログボックス**(1)

8.2　プロファイル情報ファイル

プロファイル情報ファイルを作成するためには、アプリケーションプログラムのプロファイルデータを測定後に、ProfileウィンドウのPop-upメニューで"Output Profile Information Files..."メニューオプションを選択し、ファイル名を指定します。

プロファイル情報ファイルには、関数の呼び出し回数とグローバル変数のアクセス回数の情報が入っています。HITACHI最適化リンカ(Ver.7.0以降)は、プロファイル情報ファイルを読み込み、関数および変数の配置を実際のプログラム動作状況に合わせた配置に最適化する機能を持っています。

プロファイル情報ファイルをリンカに入力するには、Standard Toolchainダイアログの"Link/Library"タブで"Category:"リストボックスを"Optimize"に指定し、"Include Profile:"チェックボックスをチェックして、プロファイル情報ファイル名を指定してください。

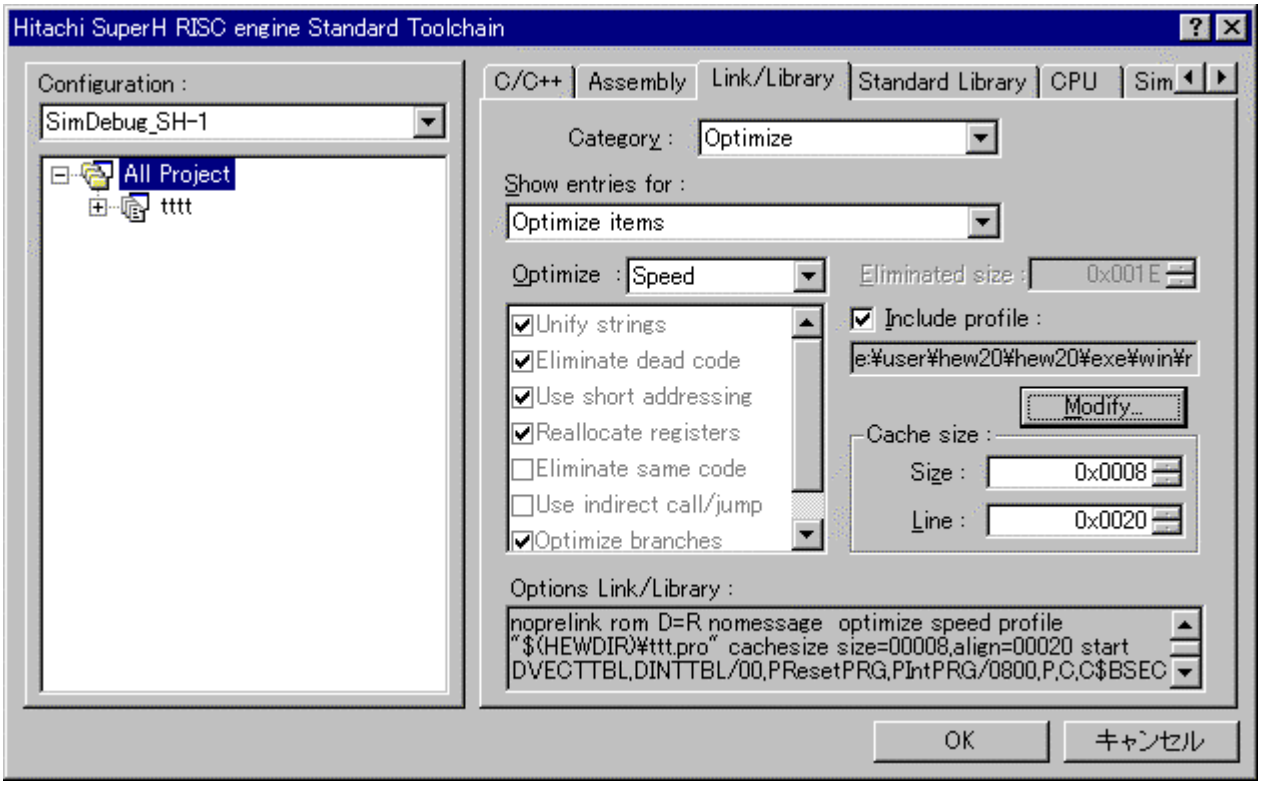


図 8.2: Standard Toolchain **ダイアログボックス**(2)

なお、"Include Profile:"チェックボックスを有効にするには、"Optimize"リストボックスを"None"以外に設定する必要があります。

8.3　スタック情報ファイルのロード

スタック情報ファイルを読み込むかどうかは、ロードモジュールロード時に表示する、確認のメッセージボックスで指定できます。メッセージボックスの“OK”ボタンをクリックするとスタック情報ファイルをロードします。

確認のメッセージボックスは、次の場合に表示します。

- スタック情報ファイルが存在する時
- Options ダイアログボックス(メインメニューの[Tools->Options…]を選択すると開きます)の”Confirmation”タブ（図 8.3）で”Load Stack Information Files (SNI files)”チェックボックスをチェックしている場合

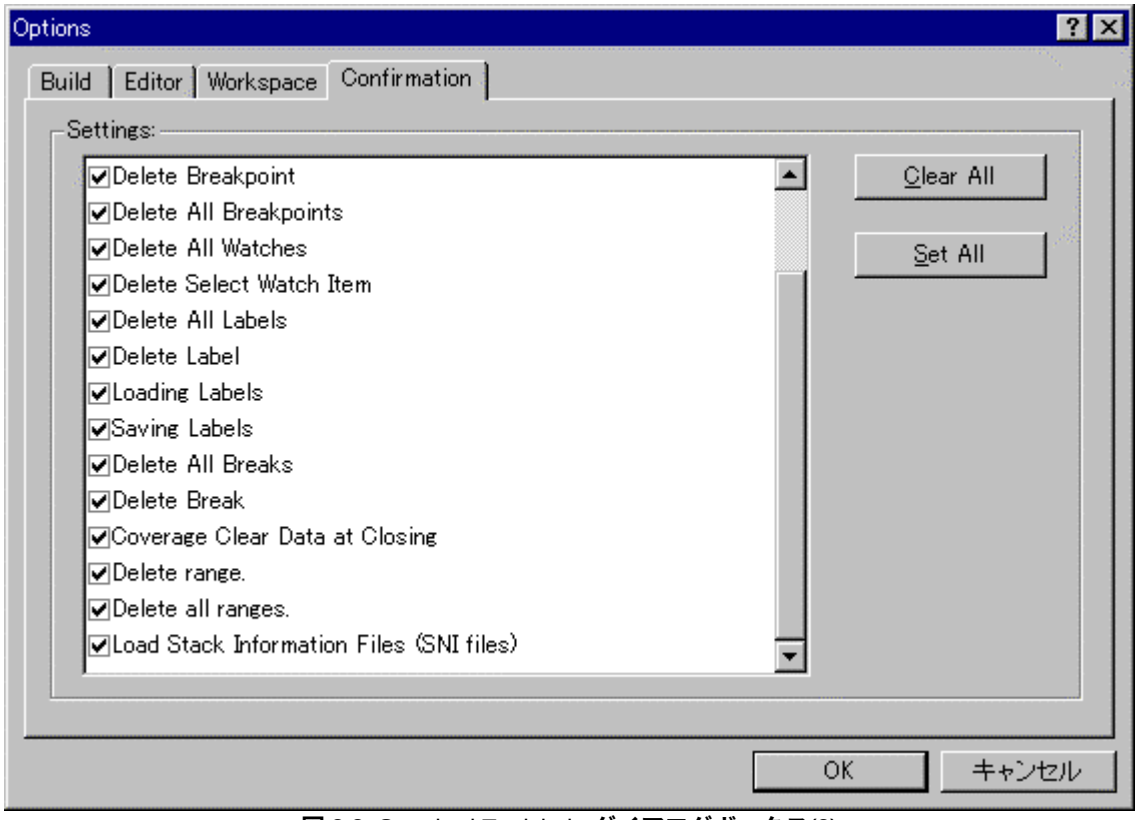


図 8.3: Standard Toolchain **ダイアログボックス**(2)

8.4　プロファイルを有効にする

[View->Profile]を選択し、Profileウィンドウをオープンします。

ProfileウィンドウのPop-upメニューで"Enable Profiler"メニューオプションを選択します。（メニューにチェックマークが付きます。）

8.5　測定方法を指定する

プロファイルデータの測定時に、関数呼び出しをトレースするかどうかを指定できます。関数呼び出しをトレースすると、ユーザプログラム実行時の関数呼び出し関係をツリー形式で表示できるようになります。関数呼び出しをトレースしないと、関数呼び出し関係を表示できませんが、Profileデータの測定時間を短縮することができます。

関数呼び出しをトレースしないようにするためには、ProfileウィンドウのPop-upメニュー”Disable Tree (Not traces function call)”を選択します。（メニューにチェックマークが付きます。）

また、OSによるタスクスイッチなど、通常の方法以外で関数を呼び出しているプログラムの場合、関数呼び出しを正しく表示できない場合がありますので、関数呼び出しをトレースせずにプロファイルデータを測定してください。

8.6　ユーザプログラムを実行し結果を確認する

ユーザプログラムを実行し、停止するとProfileウィンドウに測定結果を表示します。

Profileウィンドウには、”List”タブと”Tree”タブがあります。

8.6.1　List タブ

関数とグローバル変数をリスト表示し、各関数/変数のプロファイルデータを表示します。

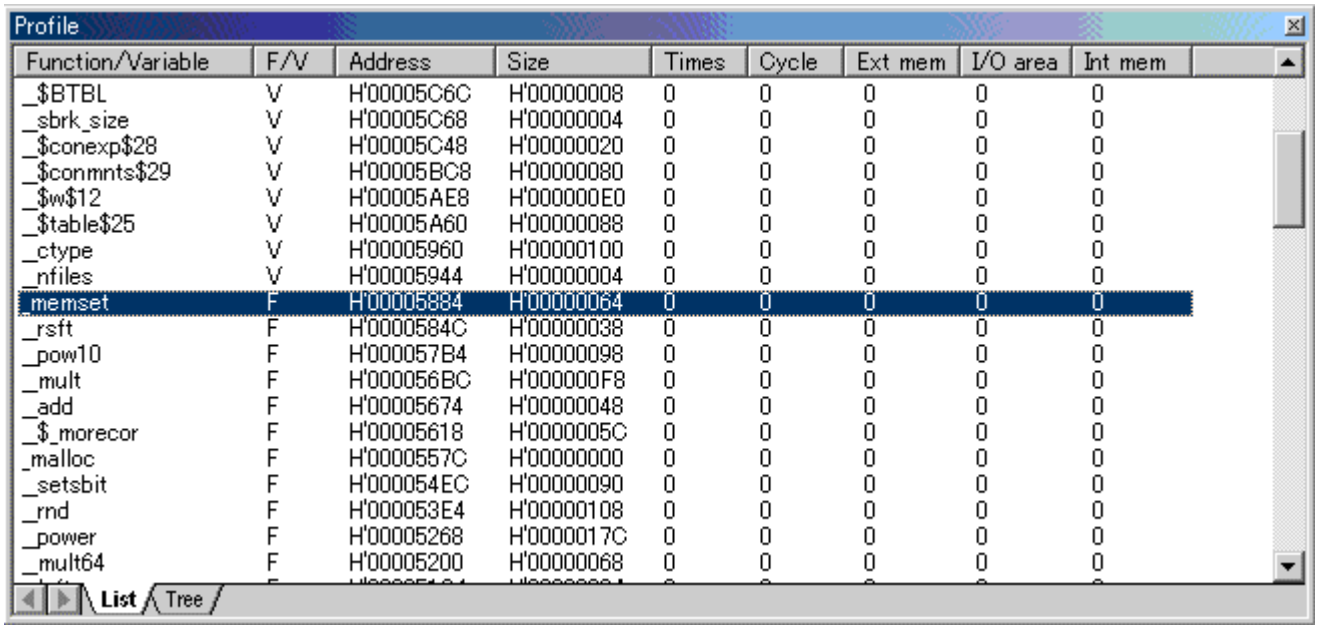


図 8.4: **List タブ**

8.6.2　Tree タブ

関数の呼び出し関係を表示し、各呼び出し位置におけるプロファイルデータを表示します。

”Tree”タブは、Profileウィンドウのポップアップメニュー”Disable Tree (Not traces function call)”をチェックしていない時のみ有効です。

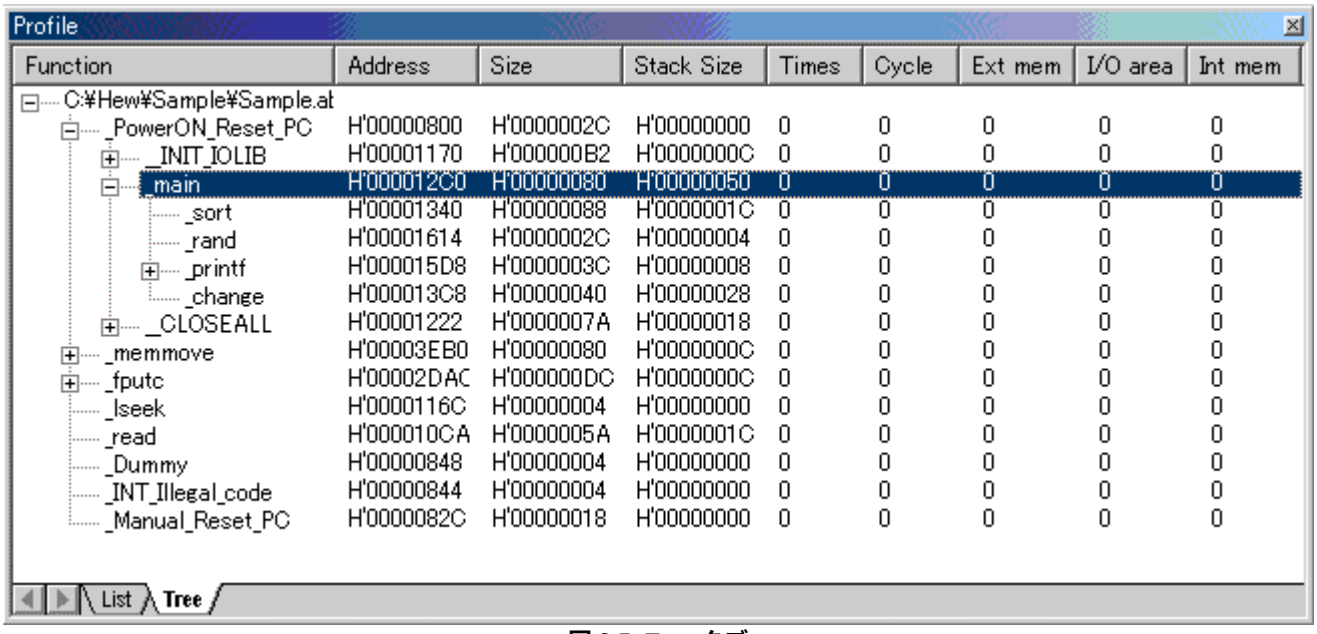


図 8.5: Tree タブ

8.6.3　Profile-Chart ウィンドウ

Profile-Chartウィンドウは、特定の関数に着目した関数の呼び出し関係を表示します。本ウィンドウは、着目する関数を中心に表示し、その左側には着目した関数を呼び出した関数、右側には、着目している関数が呼び出した関数を、それぞれ表示します。また、各呼び出しを行った回数も表示します。

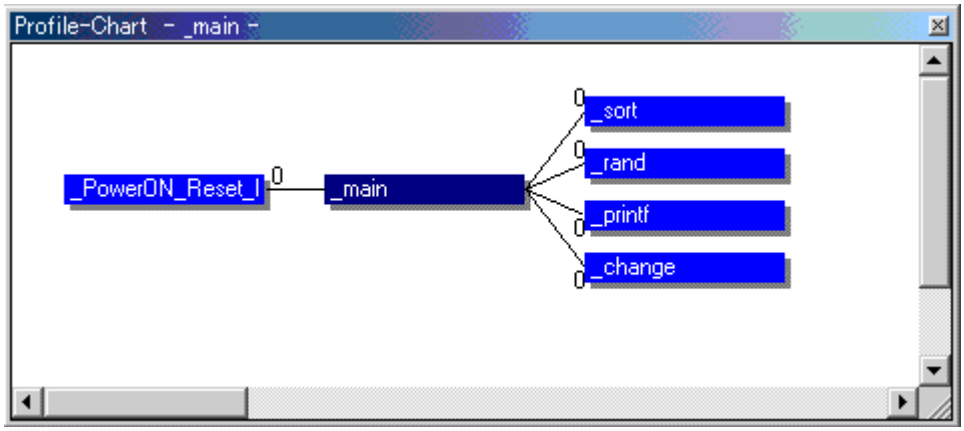


図 8.6: Profile-Chart ウィンドウ

8.7　表示データの種類および用途

プロファイル機能から下記情報を得ることができます。

・Address

関数を配置しているメモリ上の位置を知ることができます。アドレス順にソート表示することにより、メモリ上の配置イメージで関数とグローバル変数を並べることができます。

・Size

サイズ順にソート表示すれば、サイズが小さくて頻繁に呼び出している関数を見つけることができます。そのような関数があればinline関数にすることで、関数呼び出しのオーバヘッドを減らせる場合があります。

また、キャッシュ内蔵マイコンをご使用の場合、サイズの大きい関数を実行すると、更新するキャッシュのサイズが大きくなります。このような、キャッシュミスの原因となり得る関数を頻繁に呼び出していないかを容易に確認できます。

・Stack Size

関数呼び出しのネストが深い場合、関数呼び出し経路をたどり、その経路上の全関数のスタックサイズを合計することで、おおよそのスタック使用量を見積もれます。

・Times

呼び出し（アクセス）回数順にソート表示すれば、頻繁に呼び出している関数や頻繁にアクセスしている変数を容易に調べることができます。

・その他

ターゲットプラットフォームにより、さまざまなデータを測定できます。お使いのターゲット(シミュレータ/エミュレータ)マニュアルを参照してください。

8.8　プロファイル情報ファイルを作成する

プロファイル情報ファイルを作成する場合は、Pop-upメニューの"Output Profile Information Files..."メニューオプションを選択します。Save Profile Information Filesダイアログを表示します。ファイル名を選択して”Save”ボタンを押すと、選択したファイルにプロファイル情報を書きこみます。”Save All”ボタンを押すと、全てのファイルにプロファイル情報を書きこみます。

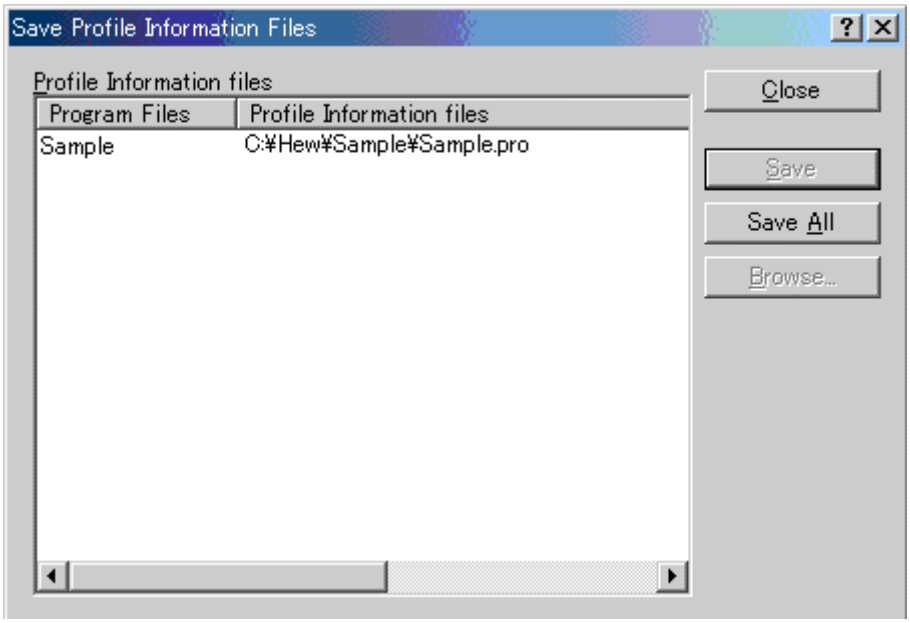


図 8.7: Save Profile Information Files **ダイアログボックス**

8.9　注意事項

（1） アプリケーションプログラムの実行サイクル数をプロファイル機能で測定した場合のデータには誤差があります。プロファイル機能では、アプリケーションプログラム全体の中で各関数が占める実行時間の比率を調べることができますが、より厳密に関数の実行サイクル数を測定したい場合には、Performance Analysis機能を使用してください。
（2） デバッグ情報が無いロードモジュールでプロファイル情報の測定を行った場合、関数名を表示しない場合があります。
（3） スタック情報ファイル(拡張子".SNI")はロードモジュールファイル(拡張子".ABS")と同一のディレクトリに置いてある必要があります。
（4） 測定結果の蓄積はできません。
（5） 測定結果の編集はできません。

付録 A　I/O ファイルフォーマット

HEWは、I/Oレジスタ定義ファイルで取得する情報に基づいて、IOウィンドウをフォーマットします。デバッギングプラットフォームを選択すると、HEWは、選択したデバイスに対応する“<device>.IO”ファイルを検索し、存在する場合にはこのファイルをロードします。これは、I/Oモジュール、およびそのレジスタのアドレスやサイズを記述するフォーマット済みテキストファイルです。ユーザはテキストエディタでこのファイルを編集し、ユーザアプリケーションに特有のメモリマップレジスタや周辺レジスタ（例えば、マイコンのアドレス空間にマップしたASICデバイスのレジスタ）のサポートを追加することができます。

A.1　ファイルフォーマット

各モジュール名を[Modules]定義セクションで定義し、モジュールの番号を、順番に付けていなければなりません。各モジュールはレジスタ定義セクションに対応しており、セクション内のエントリは、I/Oレジスタを定義します。

“BaseAddress”はデバイスのための定義であり、そのデバイスでは、CPUモードによってアドレス空間のI/Oレジスタの場所が移動します。この場合、”BaseAddress”値は、ある特有モードのI/Oレジスタのベースアドレスです。また、レジスタ定義で使用するアドレスは、同じモードにおけるレジスタのアドレス位置です。I/Oレジスタファイルを実際に使用する場合、定義したレジスタアドレスから”BaseAddress”値を引き、その結果のオフセットを選択したモードのベースアドレスに加算します。

各モジュールにはセクションがあり、オプションの依存性によって形成するレジスタを定義します。依存性は、モジュールがイネーブルかどうかを確認するためにチェックします。各レジスタ名をセクションで定義し、レジスタの番号を、順番に付けていなければなりません。依存性は、dep=<reg> <bit> <value>のようにセクションに入力します。

1. <reg>は依存性のレジスタIDです。
2. <bit>はレジスタのビット位置です。
3. <value>は値で、ビットは、イネーブルであるモジュールに使用しなければなりません。

[Register]定義エントリは、id=<name> <address> [<size>[<absolute>[<format>[<bitfields>]]]]のフォーマットで入力します。

1. <name>は表示するレジスタ名です。
2. <address>はレジスタのアドレスです。
3. <size>は、Bがバイトサイズ、Wがワードサイズ、Lがロングワードサイズを意味します（デフォルトはバイトです）。
4. <absolute>は、レジスタが絶対アドレスにある場合、Aと設定します。これは、異なるモードのCPUによってI/O空間アドレス範囲が移動する場合のみ関連します。レジスタが絶対アドレスにあると定義すると、ベースアドレスオフセットは計算せず、指定したアドレスを直接使用します。
5. <format>はレジスタを出力するためのフォーマットです。有効な値は、16進数の場合はH、10進数はD、2進数はBです。
6. <bitfields>セクションは、レジスタのビットを定義します。

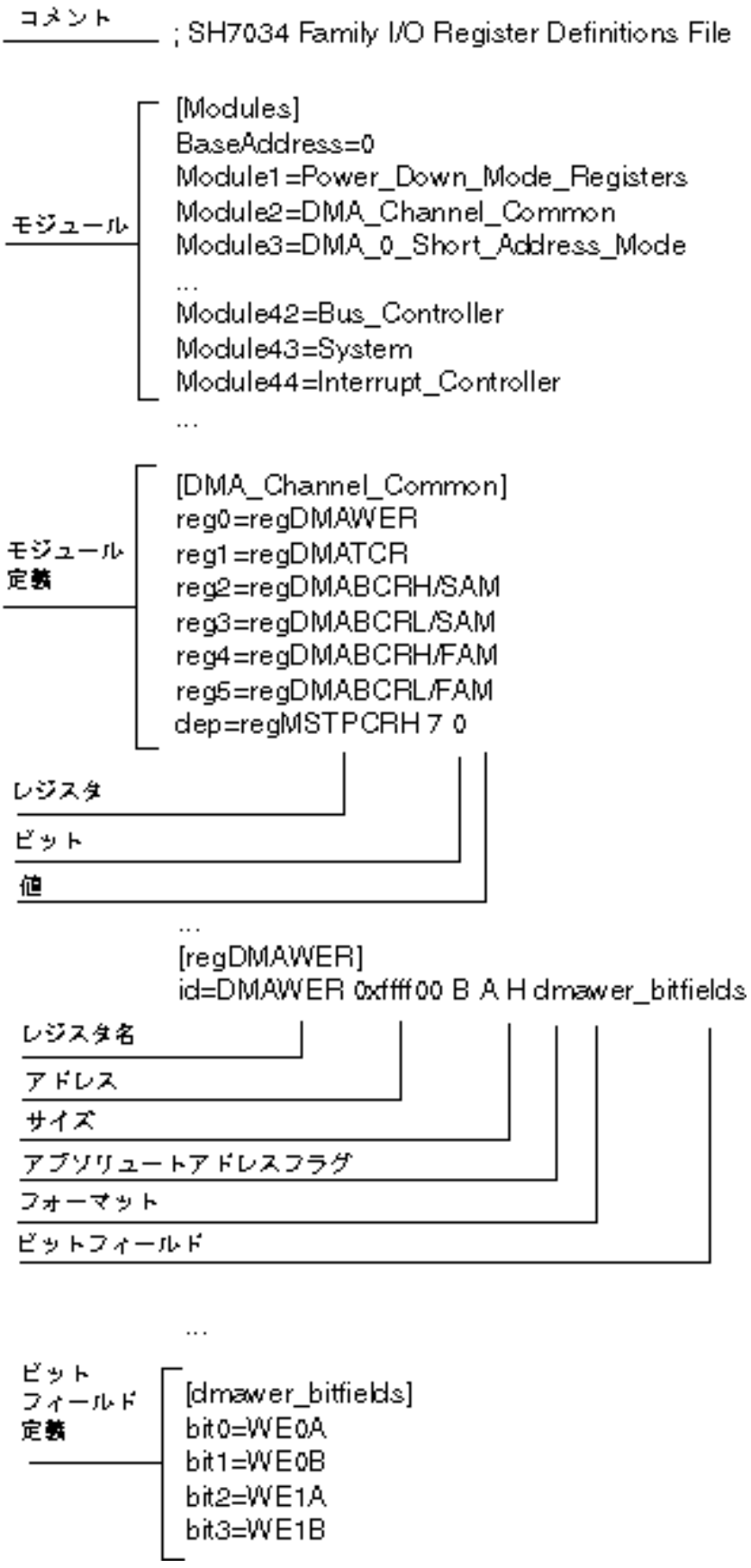
ビットフィールドセクションは、各エントリがbit<no>=<name>タイプのレジスタ内のビットを定義します。

1. <no>はビット番号です。
2. <name>はビットのシンボル名です。

コメント行を入れる場合、“;”で始めなければなりません。

次に例を示します。

例:


コメント
; SH7034 Family I/O Register Definitions File
モジュール
[Modules]
BaseAddress=0
Module1=Power_Down_Mode_Registers
Module2=DMA_Channel_Common
Module3=DMA_0_Short_Address_Mode
...
Module42=Bus_Controller
Module43=System
Module44=Interrupt_Controller
...
モジュール定義
[DMA_Channel_Common]
reg0=regDMAWER
reg1=regDMATCR
reg2=regDMABCRH/SAM
reg3=regDMABCRL/SAM
reg4=regDMABCRH/FAM
reg5=regDMABCRL/FAM
dep=regMSTPCRH 7 0
レジスタ
ビット
値
...
[regDMAWER]
id=DMAWER 0xffff00 B A H dmawer_bitfields
レジスタ名
アドレス
サイズ
アブソリュートアドレスフラグ
フォーマット
ビットフィールド
...
ビットフィールド定義
[dmawer_bitfields]
bit0=WE0A
bit1=WE0B
bit2=WE1A
bit3=WE1B

付録 B　シンボルファイルフォーマット

HEWを理解し、シンボルファイルを正確にデコードするためには、ファイルをPentica-Bファイルとしてフォーマットしなければなりません。

1. ファイルは、簡単なASCIIテキストファイルでなければなりません。
2. ファイルは、ワード“BEGIN”で始めなければなりません。
3. 各シンボルは、個々の行で、まず、“H”で終了する16進数の値から始まり、次にスペース、シンボルテキストの順でなければなりません。
4. ファイルは、ワード“END”で終了しなければなりません。

例：

```
BEGIN
11FAH Symbol_name_1
11FCH Symbol_name_2
11FEH Symbol_name_3
1200H Symbol_name_4
END
```

HEW デバッガ
High-performance Embedded Workshop 2
ユーザーズマニュアル

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-702-352A