

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ひ申し上げます。

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のですが、誤りが無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、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と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。

H8/300H Tiny コンパクトエミュレータデバッガ

ユーザーズマニュアル

ルネサスマイクロコンピュータ開発環境システム

Active X、Microsoft、MS-DOS、Visual Basic、Visual C++、WindowsおよびWindows NTは、米国Microsoft Corporationの米国およびその他の国における商標または登録商標です。

IBMおよびATは、米国International Business Machines Corporationの登録商標です。

Intel、Pentiumは、米国Intel Corporationの登録商標です。

AdobeおよびAcrobatは、Adobe Systems Incorporated（アドビシステムズ社）の登録商標です。

その他すべてのブランド名および製品名は個々の所有者の登録商標もしくは商標です。

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注意ください。

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株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズ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、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは責任を負いません。
- 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全ての情報は本資料発行時点のものであり、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、予告なしに、本資料に記載した製品又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、事前に株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又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。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、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズはその責任を負いません。
- 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。株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、適用可否に対する責任を負いません。
- 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、株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又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、文書による株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズの事前の承諾が必要です。
- 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株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店までご照会ください。

製品内容及び本書についてのお問い合わせ先

インストーラが生成する以下のテキストファイルに必要な事項を記入の上、ツール技術サポート窓口support_tool@renesas.comまで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス ソリューションズ

ツール技術サポート窓口	support_tool@renesas.com
ユーザ登録窓口	regist_tool@renesas.com
ホームページ	http://www.renesas.com/jp/tools

はじめに

High-performance Embedded Workshop は、ルネサスのマイクロコンピュータ用に、C/C++言語およびアセンブリ言語で書いたアプリケーションの開発およびデバッグを簡単に行うためのグラフィカルユーザインタフェースを提供します。アプリケーションを実行するエミュレータやシミュレータへのアクセス、計測、および変更に関して、高機能でしかも直観的な手段を提供することを目的としています。

本ヘルプでは、High-performance Embedded Workshop の主に「デバッガ」としての機能について説明しています。

対象システム

本デバッガは、コンパクトエミュレータシステム上で動作します。

対象 CPU

本ヘルプは、以下の CPU に対応したデバッグ機能を説明しています。

- H8/300H Tiny
(注)この CPU に依存する情報については、本ヘルプでは「H8/300H Tiny 用」と記載しています。

(このページは白紙です。)

1. 機能概要	3
1.1 リアルタイムRAMモニタ機能	3
1.2 ブレーク機能	3
1.2.1 ソフトウェアブレーク	3
1.2.2 ハードウェアブレーク	3
1.3 リアルタイムトレース機能	3
1.4 GUI入出力機能	4
2. コンパクトエミュレータについて	5
2.1 通信方式	5
2.2 機能表	5
3. デバッガを起動する前に	6
3.1 エミュレータとの通信方式	6
3.1.1 USB通信	6
3.2 ファームウェアのダウンロード	6
3.3 エミュレータ起動前の設定	7
3.3.1 USB通信	7
4. デバッグの準備	8
4.1 ワークスペース、プロジェクト、ファイルについて	8
4.2 High-performance Embedded Workshop の起動	9
4.2.1 新規にワークスペースを作成する（ツールチェーン使用）	10
4.2.2 新規にワークスペースを作成する場合（ツールチェーン未使用）	15
4.3 デバッグの起動	20
4.3.1 エミュレータの接続	20
4.3.2 エミュレータの終了	20
5. デバッガのセットアップ	21
5.1 Initダイアログ	21
5.1.1 MCUタブ	22
5.1.2 デバッグ情報 タブ	24
5.1.3 エミュレータ タブ	25
5.2 MCU Settingダイアログ(H8/300H Tiny用デバッガ)	26
5.2.1 MCUタブ	26

6. チュートリアル	31
6.1 はじめに	31
6.2 使用方法	32
6.2.1 Step1：デバッガの起動	32
6.2.2 Step2：RAMの動作チェック	33
6.2.3 Step3：チュートリアルプログラムのダウンロード	34
6.2.4 Step4：ブレークポイントの設定	35
6.2.5 Step5：プログラムの実行	36
6.2.6 Step6：ブレークポイントの確認	38
6.2.7 Step7：レジスタ内容の確認	39
6.2.8 Step8：メモリ内容の確認	40

6.2.9 Step9: 変数の参照	41
6.2.10 Step10: プログラムのステップ実行	43
6.2.11 Step11: プログラムの強制ブレーク	45
6.2.12 Step12: ローカル変数の表示	46
6.2.13 Step13: スタックトレース	47
6.2.14 さて次は?	48

リファレンス編

49

7. ウィンドウ一覧	51
7.1 RAMモニタウィンドウ	52
7.1.1 オプションメニュー	53
7.1.2 RAMモニタ領域を設定する	54
7.2 ASMウォッチウィンドウ	56
7.2.1 オプションメニュー	57
7.3 Cウォッチウィンドウ	58
7.3.1 オプションメニュー	59
7.4 S/Wブレークポイント設定ウィンドウ	60
7.4.1 コマンドボタン	61
7.4.2 エディタ(ソース)ウィンドウからブレークポイントを設定/解除する	62
7.5 H/Wブレークポイント設定ウィンドウ	63
7.5.1 ブレークイベント指定	64
7.5.2 組み合わせ条件指定	66
7.5.3 コマンドボタン	67
7.6 トレースポイント設定ウィンドウ	68
7.6.1 トレースイベント指定	69
7.6.2 組み合わせ条件指定	71
7.6.3 トレース範囲指定	72
7.6.4 トレース書き込み条件設定	73
7.6.5 コマンドボタン	73
7.7 トレースウィンドウ	74
7.7.1 バスモードの構成	74
7.7.2 逆アセンブルモードの構成	76
7.7.3 データアクセスモードの構成	77
7.7.4 ソースモードの構成	78
7.7.5 オプションメニュー	79
7.8 GUI入出力ウィンドウ	80
7.8.1 オプションメニュー	81
8. 機種依存コマンド一覧	82
9. 式の記述	83
9.1 式の記述	83
9.1.1 定数	83
9.1.2 シンボル、ラベル	83
9.1.3 レジスタ変数	84
9.1.4 文字定数	84
9.1.5 演算子	84
9.2 C/C++言語式の記述方法	85
9.3 C/C++言語式の表示形式	88
10. プログラム停止要因の表示	91
11. 注意事項	92
11.1 製品共通の注意事項	92

11.1.1 Windows上でのファイル操作.....	92
11.1.2 ソフトウェアブレークポイントの設定可能領域.....	92
11.1.3 C変数の参照・設定.....	92
11.1.4 C++での関数名.....	93
11.1.5 複数モジュールのデバッグ.....	93
11.1.6 同期デバッグ.....	93
11.1.7 コンパクトエミュレータのリセットスイッチ.....	93
11.2 H8/300H Tiny用デバッガの注意事項.....	94
11.2.1 コンパクトエミュレータが使用するスタック領域.....	94
11.2.2 H/Wブレーク指定.....	94
11.2.3 ハードウェアイベント.....	94

このページは白紙です。

デバッガの起動/セットアップ編

このページは白紙です。

1. 機能概要

本デバッガは、以下の機能を持っています。

1.1 リアルタイム RAM モニタ機能

ターゲットプログラム実行のリアルタイム性を損なわずにメモリ内容の変化を参照できる機能です。コンパクトエミュレータシステムは、1K バイトの RAM モニタ領域を備えています。この RAM モニタ領域は任意の連続アドレス、または、256 バイト単位で 4 ブロックの領域に分割して配置することができます。

1.2 ブレーク機能

以下のブレーク機能をサポートしています。

1.2.1 ソフトウェアブレーク

指定したアドレスの命令を実行する直前でターゲットプログラムを停止する機能です。設定可能なブレークポイント数は、64 点です。複数のソフトウェアブレークポイントを指定した場合、いずれかのブレークポイント到達でブレークします。

1.2.2 ハードウェアブレーク

メモリへのデータ書き込み/読み込み検出、命令実行検出、外部トレースケابلから入力された信号の立ち上がり/立ち下がりエッジ検出でターゲットプログラムを停止する機能です。設定可能なイベント内容は、ターゲット MCU によって異なります。指定したハードウェアブレークイベントは、以下のように組み合わせることができます。

- すべてのイベントが成立(And 条件)
- いずれかのイベントが同時に成立(And(same)条件)
- いずれかのイベントが成立(Or 条件)

1.3 リアルタイムトレース機能

ターゲットプログラムの実行履歴を記録する機能です。64K サイクルの実行履歴を記録することができます。サイクルごとのバス情報、実行した命令、ソースプログラムによる実行経路の参照が可能です。

1.4 GUI 入出力機能

ユーザーターゲットシステムのキー入力パネル(ボタン)や出力パネルをウィンドウ上で模擬する機能です。入力パネルにはボタン、出力パネルにはラベル(文字列)および LED が使用できます。

2. コンパクトエミュレータについて

コンパクトエミュレータは、手頃な価格と小さなボディでありながら、リアルタイムトレースやハードウェアブレイクなど本格的な開発に必要とされるデバッグ機能を備えた小型エミュレータです。

2.1 通信方式

エミュレータの種類によってサポートしている通信方式が異なります。

- R0E436640CPE00 は、通信インターフェースとして、USB をサポートしています。

2.2 機能表

サポートしている機能は、以下の通りです。

機能	コンパクトエミュレータ
S/W ブレーク	64 点
H/W ブレーク	2 点(組み合わせ可)
リアルタイムトレース	64K サイクル
RAM モニタ	1K バイト (256 バイト×4 ブロック) の領域
実行時間計測	Go→Stop

3. デバッグを起動する前に

デバッグを起動する前に以下の内容をご参照ください。

3.1 エミュレータとの通信方式

3.1.1 USB 通信

- 対応するホストマシンの OS は、Windows Me/98/2000/XP です。その他の OS 上では使用できません。
- USB 規格 1.1 に準拠しています。
- USB ハブ経由での接続はサポートしておりません。
- ホストマシンとエミュレータを USB ケーブルで接続することにより、対応するデバイスドライバをウィザード形式でインストールすることができます。
- 使用するケーブルは、エミュレータに付属しています。

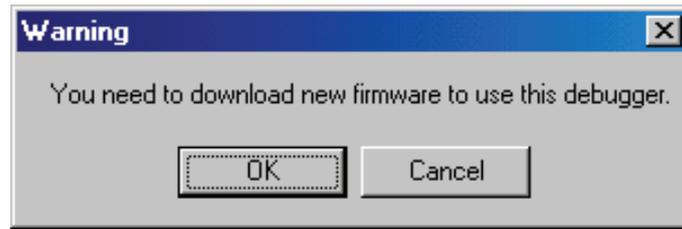
3.2 ファームウェアのダウンロード

コンパクトエミュレータに対応したファームウェアがダウンロードされている必要があります。以下のいずれかの条件に該当する場合は、エミュレータの電源投入後 2 秒以内に エミュレータのシステムリセットスイッチを押してください。エミュレータがファームウェアを強制的にダウンロードするモードとなります。

- エミュレータにダウンロードされているファームウェアが不明である。
- エミュレータデバッグを初めて使用する。
- エミュレータデバッグをバージョンアップした。

本製品は、デバッグ起動時にエミュレータにダウンロードされているファームウェアのバージョンを調べます。エミュレータにダウンロードされたファームウェアが古い場合もファームウェアをダウンロードするモードとなります。

エミュレータがファームウェアを強制的にダウンロードするモードになった状態で、デバッグを起動すると 起動時に以下のダイアログがオープンします。OK ボタンをクリックし、ファームウェアをダウンロードして下さい。



3.3 エミュレータ起動前の設定

3.3.1 USB 通信

3.3.1.1 USB デバイスドライバのインストール

Windows のプラグ&プレイ機能により USB デバイスが検出されます。USB デバイスを検出するとデバイスドライバをインストールするためのウィザードが 起動します。

以下の手順で USB デバイスドライバをインストールしてください。

1. ホストマシンとエミュレータを USB ケーブルで接続してください。
2. エミュレータの通信インタフェース設定スイッチを"USB"に設定し、電源を投入してください。
3. 以下のダイアログがオープンします。



そのままウィザードに従うとセットアップ情報ファイル(`inf` ファイル)を指定するためのダイアログがオープンします。本製品をインストールしたディレクトリ下の `musbdrv.inf` ファイルを指定してください。

注意事項

- USB デバイスドライバをインストールするには、あらかじめご使用になるエミュレータデバッガがインストールされている必要があります。先にエミュレータデバッガをインストールしてください。
- USB 通信は、Windows 98/Me/2000/XP 以外の OS では使用できません。
- Windows 2000/XP をご使用の場合、USB デバイスドライバのインストールは Administrator 権限を持つユーザが実施してください。
- インストール中にデバイスドライバ本体 `musbdrv.sys` が見つからないというメッセージが出る場合があります。 `musbdrv.sys` は、`musbdrv.inf` ファイルと同じディレクトリに格納されています。

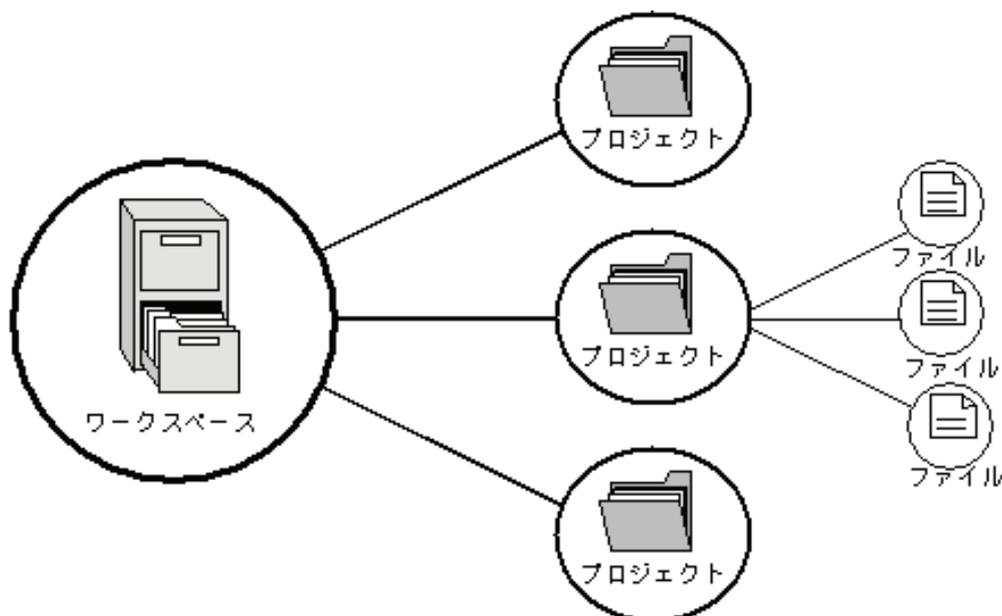
4. デバッグの準備

本製品を起動し、エミュレータに接続してデバッグを開始します。
なお、本製品でデバッグを行うためには、ワークスペースを作成する必要があります。

4.1 ワークスペース、プロジェクト、ファイルについて

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、本製品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1つ以上あり、各プロジェクトにはファイルが1つ以上あります。

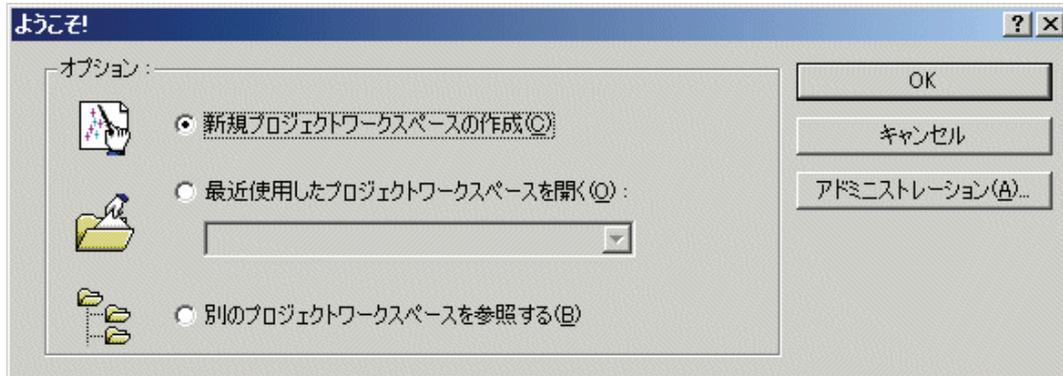


ワークスペースでは関連したプロジェクトを1つにまとめることができます。例えば、異なるプロセッサに対して1つのアプリケーションを構築しなければならない場合、または、アプリケーションとライブラリを同時に開発している場合などに便利です。さらに、ワークスペース内でプロジェクトを階層的に関連づけることができます。つまり、1つのプロジェクトを構築すると、その子プロジェクトを最初に構築します。

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。

4.2 High-performance Embedded Workshop の起動

[スタート]メニューの[プログラム]から High-performance Embedded Workshop を起動してください。
[ようこそ!]ダイアログボックスが表示されます。



このダイアログで、ワークスペースを作成/表示します。

- [新規プロジェクトワークスペースの作成]ラジオボタン
ワークスペースを新規作成する場合に選択します。
- [最近使用したプロジェクトワークスペースを開く]ラジオボタン
既存のワークスペースを使用する場合に選択します。
開いたワークスペースの履歴が表示されます。
- [別のプロジェクトワークスペースを参照する]ラジオボタン
既存のワークスペースを使用する場合に選択します。
開いた履歴が残っていない場合に使用します。

既存ワークスペースを指定する場合は、[最近使用したプロジェクトワークスペースを開く]または[別のプロジェクトワークスペースを参照する]ラジオボタンを選択し、ワークスペースファイル(拡張子.hws)を指定してください。

新規ワークスペースの作成方法については、以下を参照ください。

- 4.2.1 新規にワークスペースを作成する (ツールチェイン使用)
- 4.2.2 新規にワークスペースを作成する場合 (ツールチェイン未使用)
※既存のロードモジュールファイルを本製品でデバッグする場合などは、この方法でワークスペースを作成します。

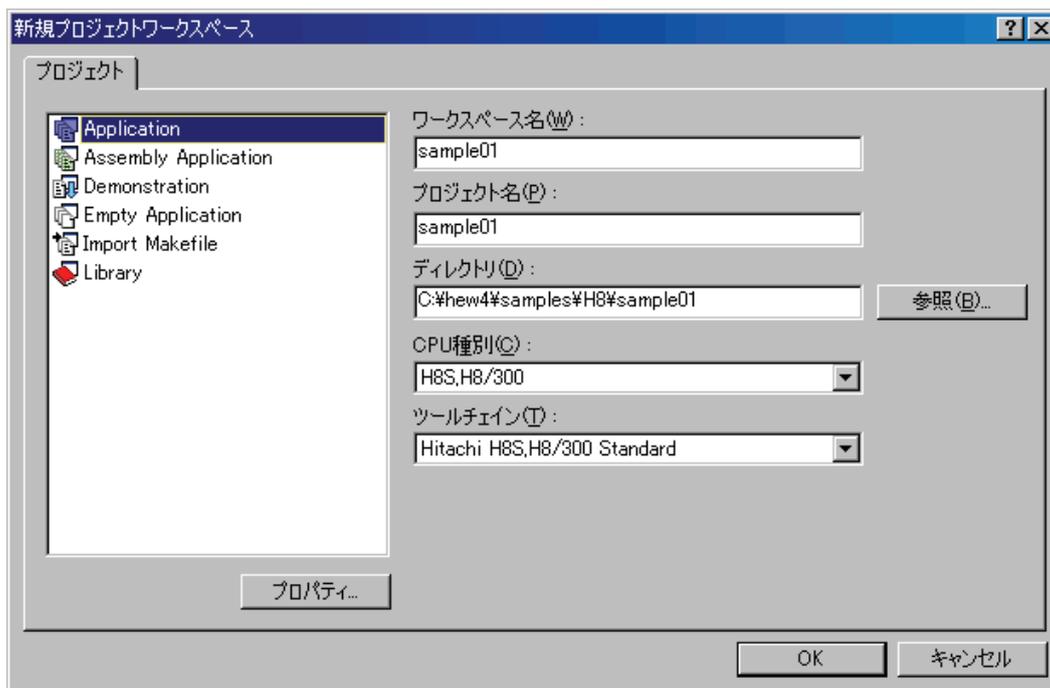
ツールチェインを使用する場合と使用しない場合では新規プロジェクトワークスペースの作成手順が異なります。本製品には、ツールチェインは含まれていません。ツールチェインはご使用のCPUに対応したC/C++コンパイラパッケージがインストールされている環境にて使用することができます。ツールチェインを使用した新規プロジェクトワークスペースの作成についての詳細は、C/C++コンパイラパッケージ付属のマニュアルを参照してください。

4.2.1 新規にワークスペースを作成する（ツールチェイン使用）

4.2.1.1 Step1：新規プロジェクトワークスペースの設定

High-performance Embedded Workshop 起動時に表示される、[ようこそ]ダイアログボックスで、[新規プロジェクトワークスペースの作成]ラジオボタンを選択し、[OK]ボタンをクリックしてください。新規プロジェクトワークスペースの作成を開始します。

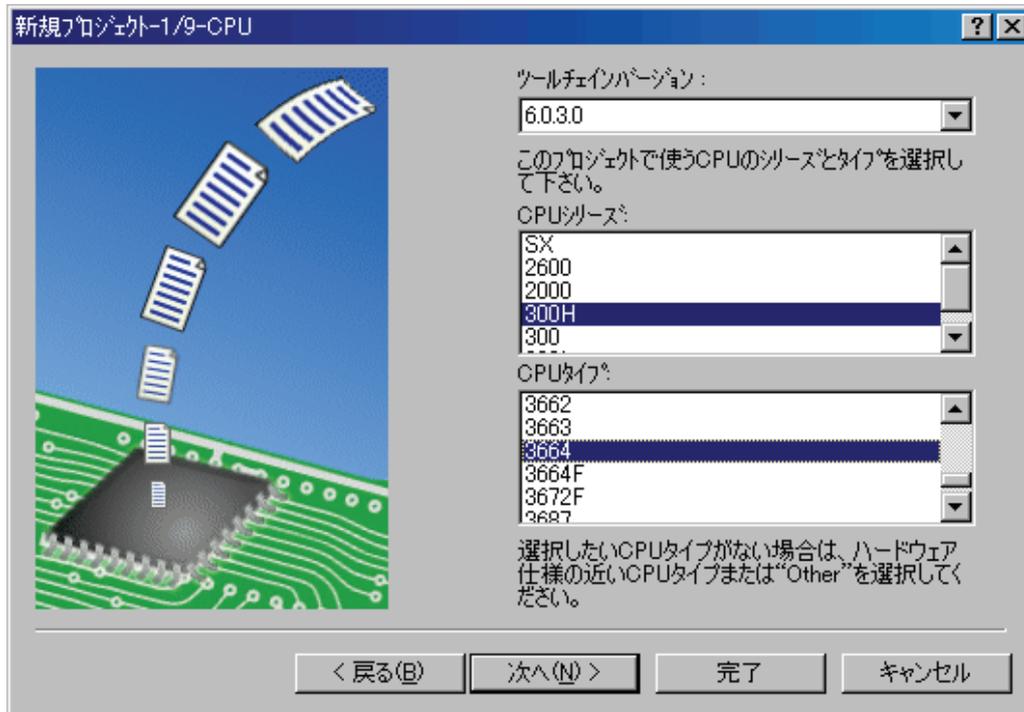
以下の画面が開きます。



1. CPU 種別を選択する
[CPU 種別]ドロップダウンリストボックスで、使用する CPU ファミリを選択してください。
2. ツールチェインを選択する
[ツールチェイン]ドロップダウンリストボックスで、該当するツールチェイン名を選択してください。
3. プロジェクトタイプを選択する
左の[プロジェクトタイプ]リストボックスで、使用したいプロジェクトタイプを選択します。
ここで、"Application" を選択してください。
(選択できるプロジェクトタイプの詳細については、C/C++コンパイラパッケージ付属のマニュアルを参照ください。)
4. ワークスペース名、プロジェクト名を指定する
 - [ワークスペース名]エディットボックスに、新規作成するワークスペース名を入力してください。
 - [プロジェクト名]エディットボックスに、プロジェクト名を入力してください。ワークスペース名と同じであれば、入力する必要はありません。
 - [ディレクトリ]エディットボックスに、ワークスペースを作成するディレクトリを入力してください。
[参照...]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択することもできます。
入力後、[OK]ボタンを押してください。

4.2.1.2 Step2：ツールチェインの設定

プロジェクト作成ウィザードが起動します。



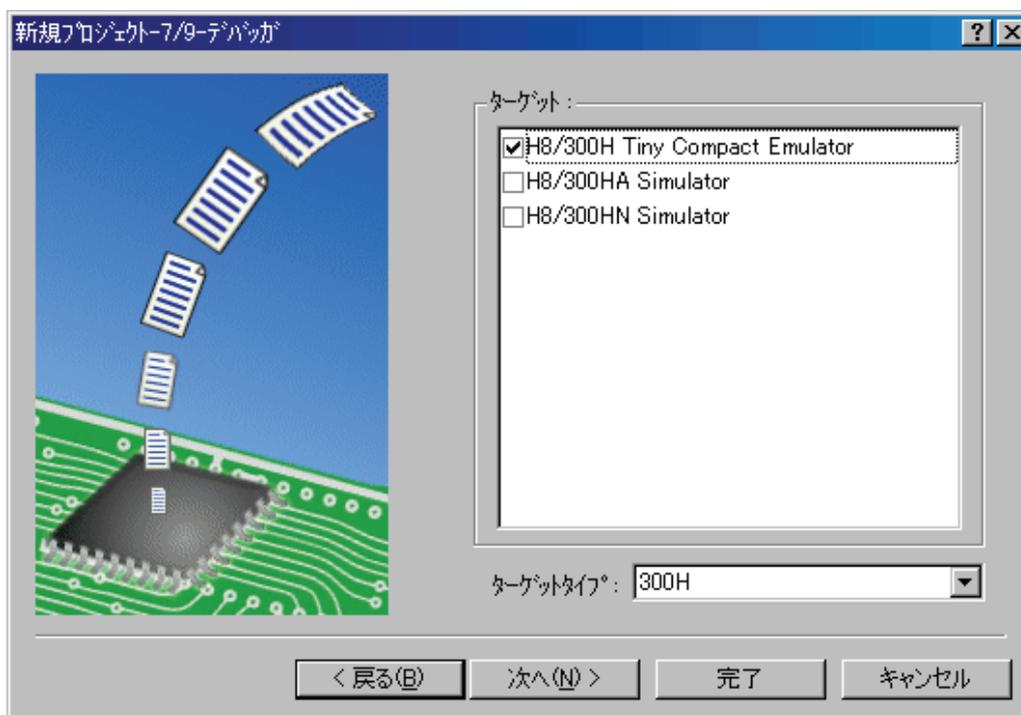
ウィザードの最初のほうでは以下の設定を行います。

- ツールチェインの設定
 - リアルタイム OS に関する設定（使用する場合）
 - 生成ファイル、ヒープ領域、スタック領域等の設定
- 必要な情報を入力し、[次へ]ボタンを押して行ってください。

設定内容はご使用の C/C++コンパイラパッケージにより異なります。設定内容の詳細については、C/C++コンパイラパッケージ付属のマニュアルを参照ください。

4.2.1.3 Step 3: ターゲットプラットフォームの選択

ウィザードの終盤で、使用するターゲット（エミュレータ、シミュレータ）の設定を行います。ツールチェーンの設定が終了したら、以下の画面が表示されます。



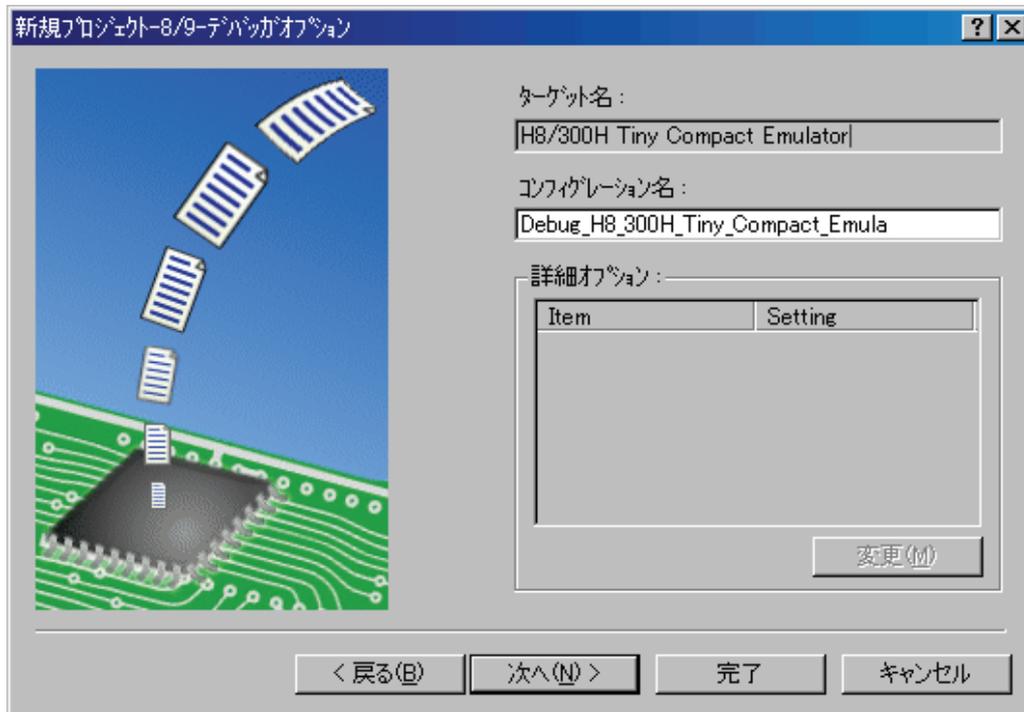
1. ターゲットタイプの選択
[Target type] ドロップダウンリストボックスで、使用するターゲットの CPU タイプを選択ください。
2. ターゲットプラットフォームの選択
[Targets] 領域に、使用可能なターゲットが表示されます。
使用するターゲットをチェックしてください（複数指定可能）。

入力後、[次へ]ボタンを押してください。

4.2.1.4 Step4：コンフィグレーションファイル名の設定

選択したターゲット毎にコンフィグレーションファイル名を設定します。

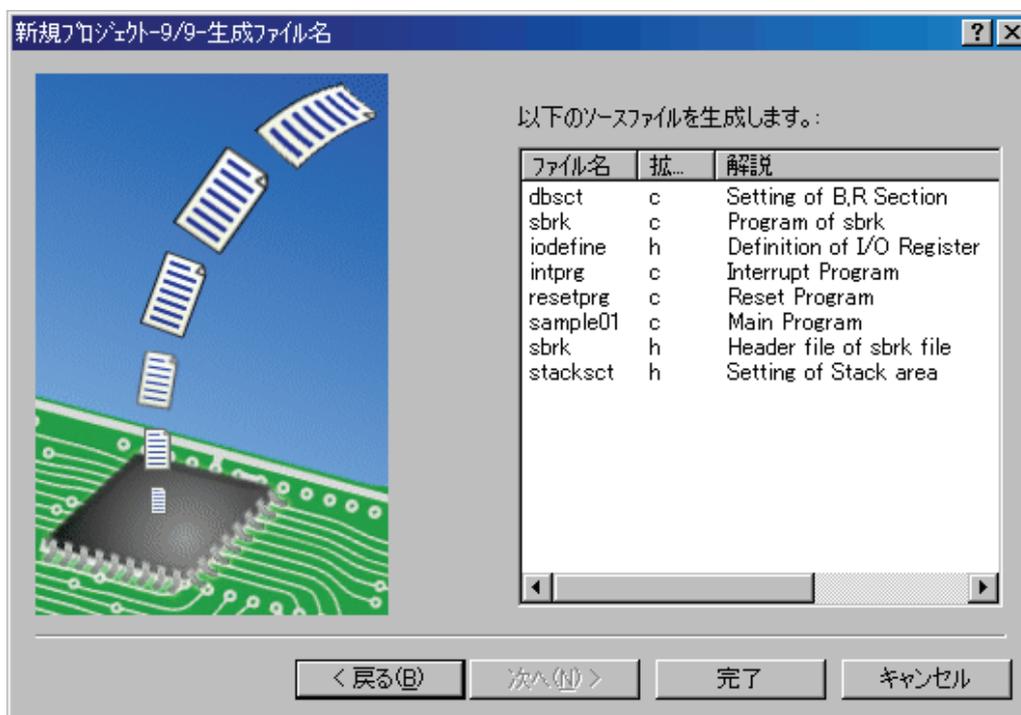
コンフィグレーションとは、ターゲット以外の High-performance Embedded Workshop の状態を保存するファイルです。



デフォルトの名前がすでに設定されていますので、変更する必要がなければそのまま[次へ]ボタンで進んでください。

4.2.1.5 Step5：生成ファイルの確認

これまでの設定により本製品が生成するファイルが表示されます。ファイル名を変更したい場合は、ファイル名を選択してクリック後、入力してください。



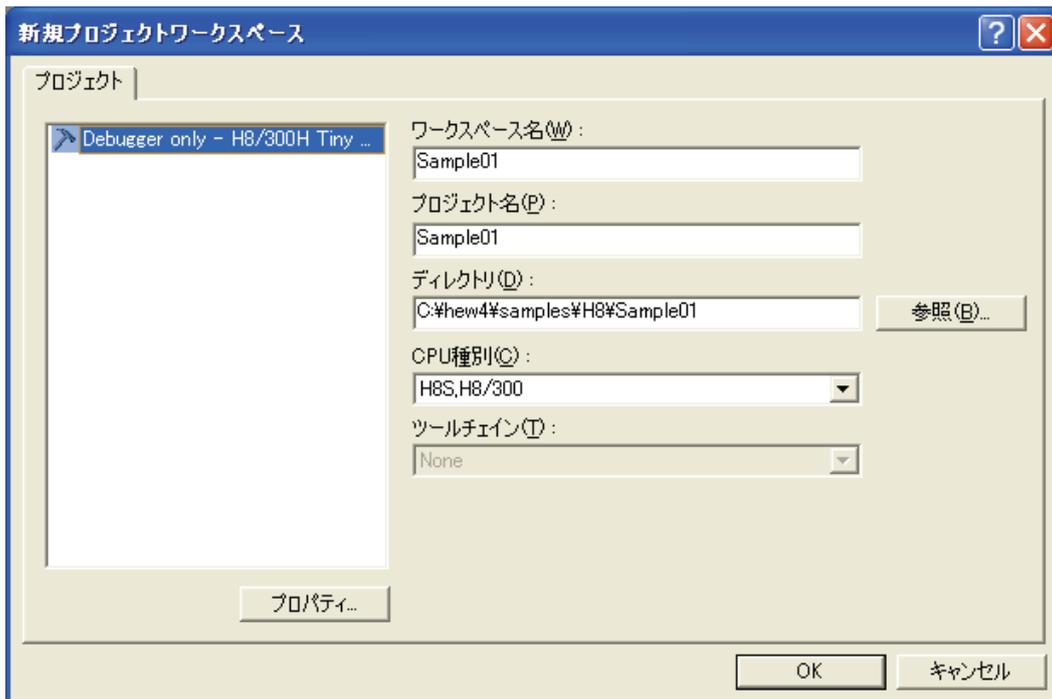
これで エミュレータ に関する設定は終了です。
画面の指示に従い、プロジェクト作成ウィザードを終了してください。

4.2.2 新規にワークスペースを作成する場合（ツールチェイン未使用）

既存のロードモジュールファイルを本製品でデバッグする場合などは、この方法でワークスペースを作成します。（ツールチェインがインストールされていない場合でも OK です。）

4.2.2.1 Step1：新規プロジェクトワークスペースの設定

High-performance Embedded Workshop 起動時に表示される、[ようこそ!]ダイアログボックスで、[新規プロジェクトワークスペースの作成]ラジオボタンを選択し、[OK]ボタンをクリックしてください。新規プロジェクトワークスペースの作成を開始します。以下の画面が開きます。

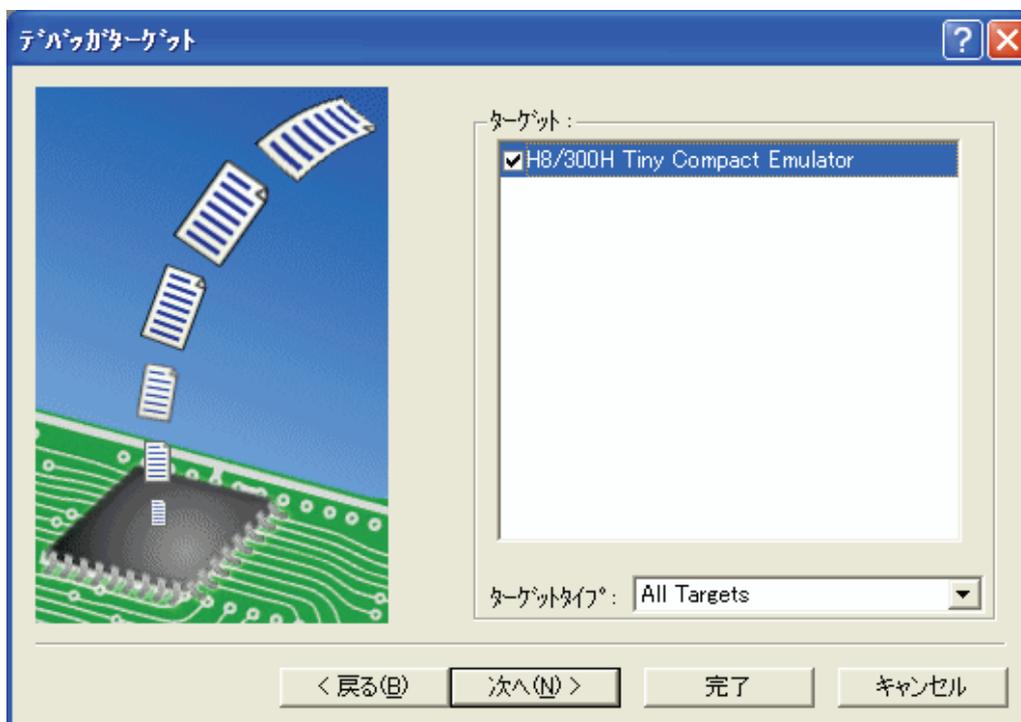


1. CPU 種別を選択する
[CPU 種別]ドロップダウンリストボックスで、使用する CPU ファミリを選択してください。
2. ツールチェインを選択する
ツールチェインは使用しませんので、[ツールチェイン]ドロップダウンリストボックスでは"None" を指定してください。
(ツールチェインが登録されていない場合は、このドロップダウンリストは選択できません（選択不要）。)
3. プロジェクトタイプを選択する
ツールチェインを使用しない場合、左の[プロジェクトタイプ]リストボックスには"Debugger only - ターゲット名" と表示されますので、それを選択ください（複数のターゲットが表示される場合は、使用するプロジェクトタイプを1つ選択してください）。
4. ワークスペース名、プロジェクト名を指定する
 - [ワークスペース名]エディットボックスに、新規作成するワークスペース名を入力してください。
 - [プロジェクト名]エディットボックスに、プロジェクト名を入力してください。ワークスペース名と同じであれば、入力する必要はありません。
 - [ディレクトリ]エディットボックスに、ワークスペースを作成するディレクトリを入力してください。[参照...]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択することもできます。

入力後、[OK]ボタンを押してください。

4.2.2.2 Step 2: ターゲットプラットフォームの選択

使用するターゲット（エミュレータ、シミュレータ）の設定を行います。
プロジェクト作成ウィザードが起動し、以下の画面を表示します。

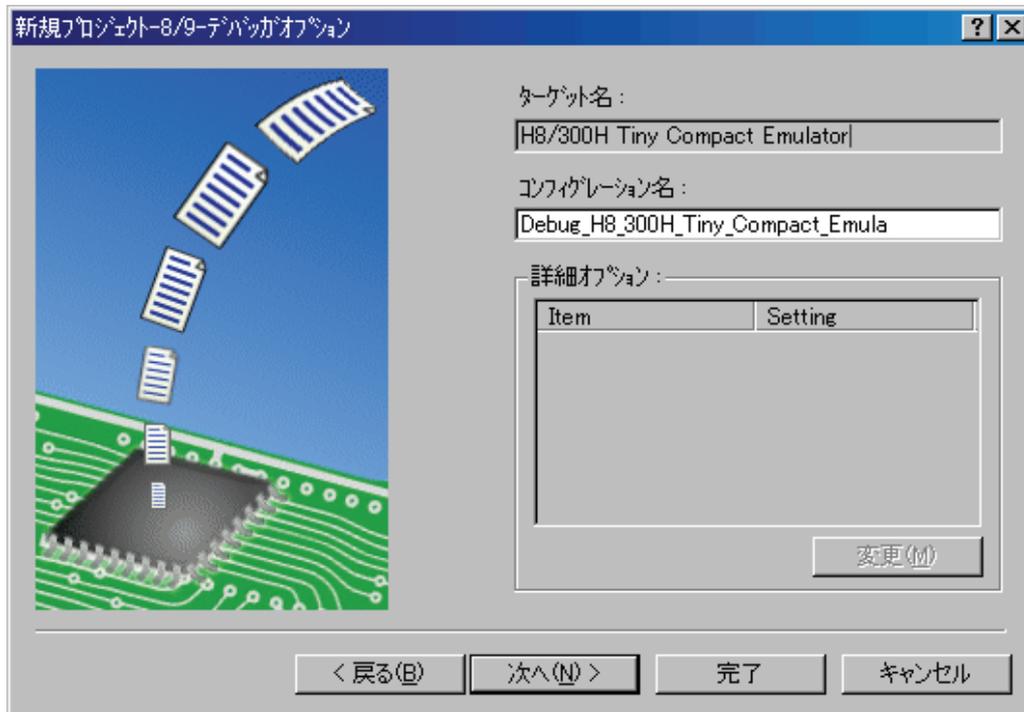


1. ターゲットタイプの選択
[Target type] ドロップダウンリストボックスで、使用するターゲットの CPU タイプを選択ください。
2. ターゲットプラットフォームの選択
[Targets] 領域に、使用可能なターゲットが表示されます。
使用するターゲットをチェックしてください（複数指定可能）。

入力後、[次へ]ボタンを押してください。

4.2.2.3 Step3 : コンフィグレーションファイル名の設定

選択したターゲット毎にコンフィグレーションファイル名を設定します。
コンフィグレーションとは、ターゲット以外の High-performance Embedded Workshop の状態を保存するファイルです。



デフォルトの名前がすでに設定されていますので、変更する必要がなければそのまま[次へ]ボタンで進んでください。

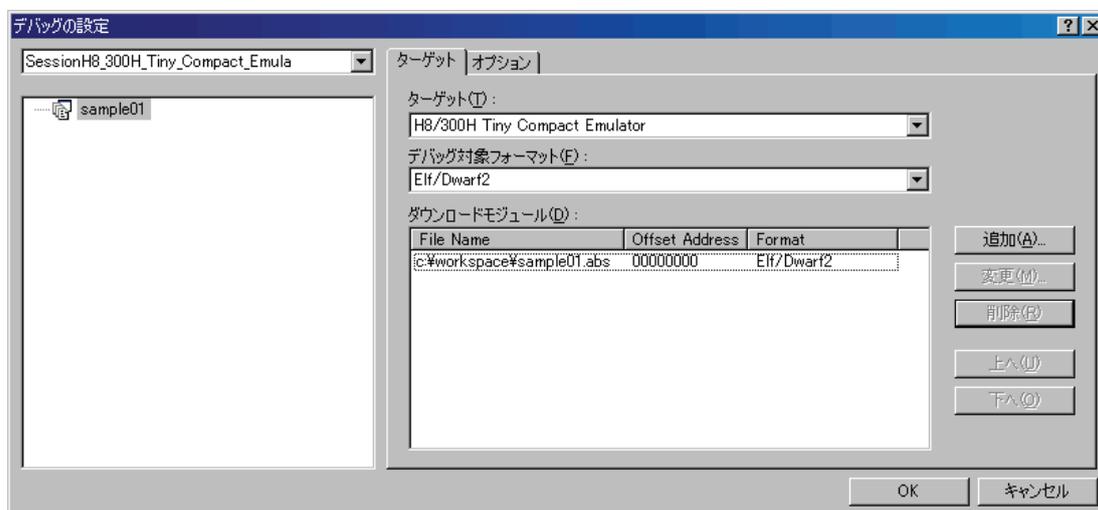
これで エミュレータ に関する設定は終了です。
画面の指示に従い、プロジェクト作成ウィザードを終了してください。

High-performance Embedded Workshop 起動と同時に、デバッガのセットアップを開始するダイアログが表示されます。エミュレータの準備が完了していれば、そのままセットアップを行い、エミュレータへ接続してください。

4.2.2.4 Step4：ダウンロードモジュールの登録

最後に、使用するロードモジュールファイルを登録します。

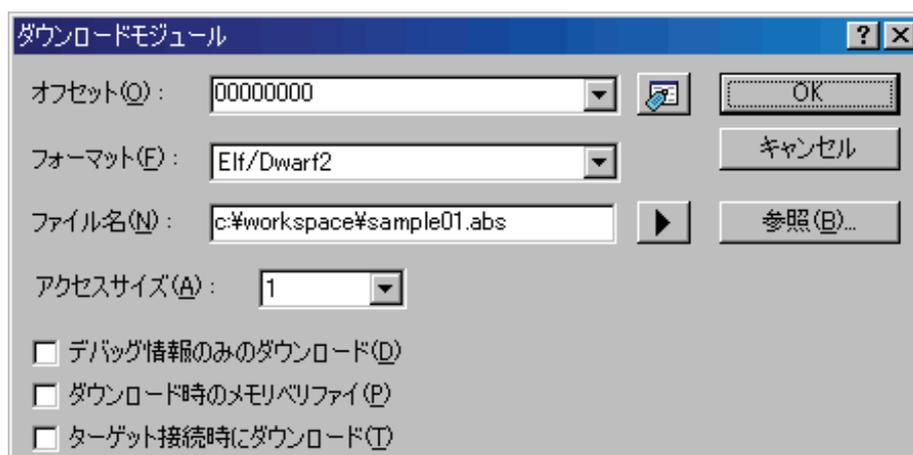
[デバッグ]メニューから[デバッグの設定...]を選択してください。開いたダイアログボックスで、以下の設定を行います。



1. [ターゲット]ドロップダウンリストボックスで接続したい製品名を選択してください。
2. [デバッグ対象フォーマット]ドロップダウンリストボックスで、ダウンロードするロードモジュールの形式を選択してください。

フォーマット名	種類
ELF/DWARF2	ELF/DWARF2 フォーマットファイル (弊社製クロスツール H8C で生成)

3. [ダウンロードモジュール]リストボックスに、ダウンロードモジュールを登録してください。ダウンロードモジュールは、[追加]ボタンで開く以下のダイアログで指定できます。



- [オフセット]エディットボックスに、ダウンロードモジュールをロードするオフセットを指定してください。
- [フォーマット]エディットボックスに、ダウンロードモジュールの形式を指定してください。形式名は、上記の一覧表を参照ください。
- [ファイル名]エディットボックスに、ダウンロードモジュールのフルパスとファイル名を入力してください。

- [アクセスサイズ]リストボックスに、ダウンロード時のメモリアクセスサイズを指定してください。
- 設定完了後、[OK]ボタンを押してください。

4.3 デバッガの起動

エミュレータ に接続することで、デバッグを開始できます。

4.3.1 エミュレータの接続

エミュレータ を使用する設定があらかじめ登録されているセッションファイルに切り替えることにより、エミュレータ を簡単に接続できます。

プロジェクト作成時にターゲットを選択すると、その選択したターゲットの個数分のセッションファイルがデフォルトで作成されています。

下記ツールバーのドロップダウンリストから、接続するターゲットに対応したセッションファイルを選択してください。



選択すると、デバッガのセットアップを行うためのダイアログが表示されます。
このセットアップが終了すると、接続は完了です。

4.3.2 エミュレータの終了

以下の方法があります。

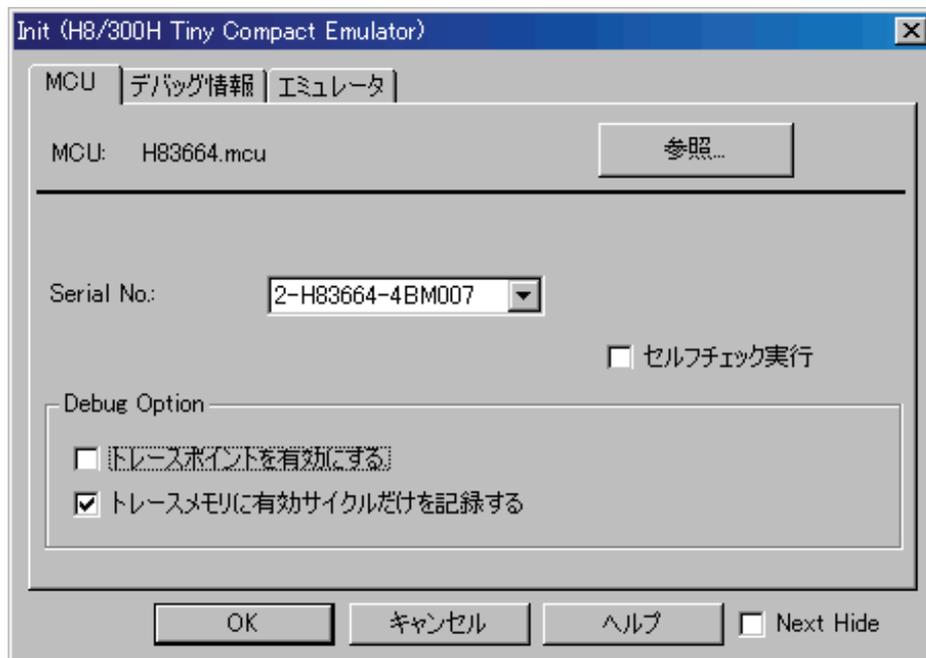
1. セッションを"DefaultSession" に切り替える
エミュレータ 接続時に使用したドロップダウンリストで,"DefaultSession" を選択してください。
2. High-performance Embedded Workshop 自体を終了する
[ファイル>アプリケーションの終了]を選択してください。High-performance Embedded Workshop は終了します。

セッション切り替え、および、High-performance Embedded Workshop 終了前には、セッション保存確認のメッセージボックスが表示されます。セッション保存が必要な場合は、[はい]ボタンをクリックしてください。不要なら、[いいえ]ボタンをクリックしてください。

5. デバッガのセットアップ

5.1 Init ダイアログ

Init ダイアログは、デバッガ起動時に設定が必要な項目を設定するためのダイアログです。このダイアログで設定した内容は、次回起動時にも有効となります。



各タブの詳細については、表のタブ名をクリックしてください。製品によってサポートしているタブが異なります。

タブ名	設定内容
MCU	MCU ファイル、通信インタフェースなどを指定します。
デバッグ情報	使用コンパイラ、デバッグ情報の格納方式を指定します。
エミュレータ	ターゲットクロックを指定します。

Init ダイアログ下部の Next Hide をチェックすると次回デバッガ起動時にこの Init ダイアログを オープンしないようにすることができます。

また、Init ダイアログは、以下のいずれかの方法で再表示できます。

- デバッガ起動後、メニュー[基本設定]→[エミュレータ]→[システム...]を選択する。
- Ctrl キーを押しながらデバッグセッションに切り替える。

5.1.1 MCU タブ

指定した内容は、次回起動時にも有効となります。



5.1.1.1 MCU ファイルの指定



[参照]ボタンをクリックして下さい。

ファイルセレクションダイアログがオープンしますので、該当する MCU ファイルを指定してください。

- MCU ファイルは、ターゲット MCU の固有情報を格納したファイルです。
- 指定した MCU ファイルは、MCU タブの MCU 領域に表示されます。

5.1.1.2 通信インタフェースの指定

使用するインタフェースを選択してください。

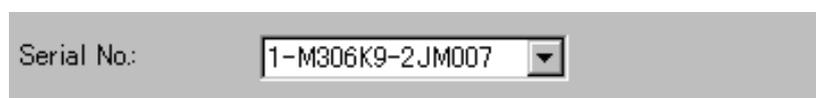
使用可能な通信インタフェースは、エミュレータによって異なります。

以下に通信インタフェースごとの設定を示します。

USB通信の設定

USB通信は、パーソナルコンピュータのUSBインタフェースを使用します。USB 1.1に準拠しています。

USB通信するには、あらかじめ専用のデバイスドライバがインストールされている必要があります。USBデバイスドライバのインストールについては、「3.3.1.1 USBデバイスドライバのインストール」を参照してください。



Serial No.領域には、現在 USB 接続されているエミュレータの一覧を表示します。
接続するエミュレータのシリアル No.を選択してください。

5.1.1.3 セルフチェックの実行

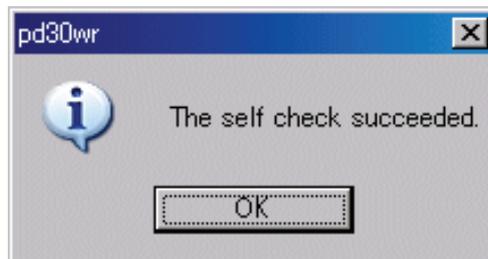
起動時にエミュレータの*セルフチェックを実行する場合に指定します。

セルフチェック実行

起動時にセルフチェックを行いたい場合のみ、上記チェックボックスをチェックしてください。次のような場合に指定してください。

- ファームウェアのダウンロードに失敗するとき
- ファームウェアのダウンロードは成功するが、デバッガの起動に失敗するとき
- MCU が暴走する、あるいは、トレース結果がおかしい場合などに、エミュレータが正常に動作しているか確認したいとき

チェックボックスをチェックして Init ダイアログを閉じると、エミュレータと接続しファームウェアを確認した直後にセルフチェックが始まります（セルフチェックの所要時間は、約 30 秒～1 分です）。セルフチェックでエラーが検出された場合は、エラー内容を表示しデバッガは終了します。セルフチェックが正常に終了した場合は、以下のダイアログが表示されます。OK ボタンを押すとそのままデバッガが起動します。



この指定は、デバッガ起動時のみ行えます。

* セルフチェックとは、エミュレータの内蔵基板のメモリ状態などを検査する機能です。セルフチェック機能に関する詳細は、ご使用のエミュレータのマニュアルを参照してください。

5.1.1.4 トレースポイント設定機能の使用/未使用

コンパクトエミュレータが持っている 2 点のイベントをトレースポイントとして使用するかどうかを指定します。

トレースポイントを有効にする。

トレースポイントとして使用する場合は、上記のチェックボックスをチェックしてください。
この指定は、デバッガ起動時のみ設定/変更が可能です。

補足事項

トレースポイント設定機能を有効にした場合、以下の機能は使用できません。

- ハードウェアブ레이크機能

5.1.1.5 トレースメモリに有効サイクルのみを記録する

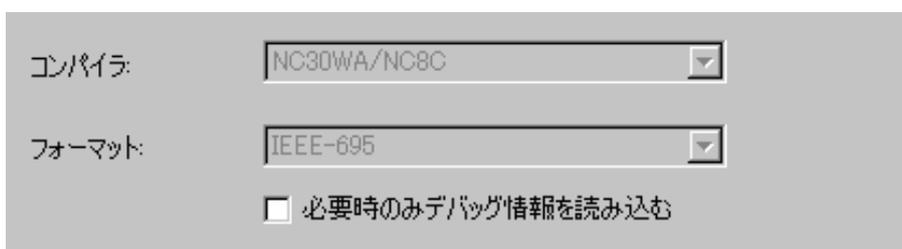
コンパクトエミュレータでは、トレースメモリに有効サイクルのみを記録するか、すべてのサイクルを記録するかを選択できます。

トレースデータとして有効サイクルのみ取得する

有効サイクルのみを記録する場合は、上記のチェックボックスをチェックしてください。

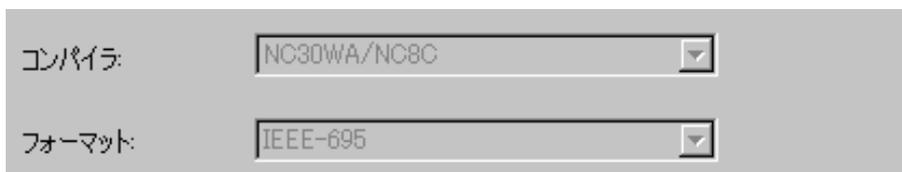
5.1.2 デバッグ情報 タブ

指定した内容は、次回ダウンロード時から有効です。



5.1.2.1 使用コンパイラ/オブジェクトフォーマットの参照

ご使用のコンパイラと、オブジェクトファイルのフォーマットを表示します。



本ダイアログで、現在の設定内容が確認できます。設定は、メニュー[デバッグ]→[デバッグの設定...]で開くダイアログで行ってください。

5.1.2.2 デバッグ情報の格納方式指定

デバッグ情報の格納方式には、オンメモリ方式とオンデマンド方式があります。デバッグ情報の格納方式を選択してください(デフォルトはオンメモリ方式です)。オンデマンド方式を選択する場合、[必要時のみデバッグ情報を読み込む]チェックボックスをチェックします。

- オンメモリ方式
デバッグ情報をパーソナルコンピュータのメモリ上に保持します。
ロードモジュール (ターゲットプログラム) の規模が小さい場合に適します。
- オンデマンド方式
デバッグ情報を再利用可能なテンポラリファイル上に保持します。
同一ロードモジュールに対する二度目以降のダウンロードでは、保持されたデバッグ情報を再利用するため、高速にダウンロード可能です。
ロードモジュール (ターゲットプログラム) の規模が大きい場合に適します。

注意事項

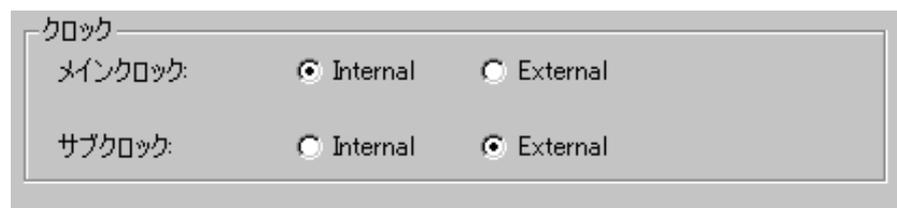
- ロードモジュールの規模が大きい場合、オンメモリ方式では、ダウンロード処理で非常に時間を要する場合があります。この場合は、オンデマンド方式を選択してください。
- オンデマンド方式では、ダウンロードしたロードモジュールが位置するフォルダに、再利用可能なテンポラリファイルを格納するフォルダを作成します。フォルダ名は、"~INDEX_" にロードモジュール名を付加した名称です。例えば、ロードモジュール名が "sample.abs" ならば、フォルダ名は "~INDEX_sample" です。このフォルダは、デバッガを終了しても削除されません。

5.1.3 エミュレータ タブ



5.1.3.1 ターゲットクロックの指定

MCU（メインクロック、サブクロック）への供給クロックを指定します。
ターゲットマイコンの使用クロックに合わせて設定を変更してください(デフォルトは Internal です)。



内部クロックに設定する場合は Internal、外部クロックに指定する場合は External を選択します。
指定した内容は、次回起動時にも有効となります。

5.2 MCU Setting ダイアログ(H8/300H Tiny 用デバッガ)

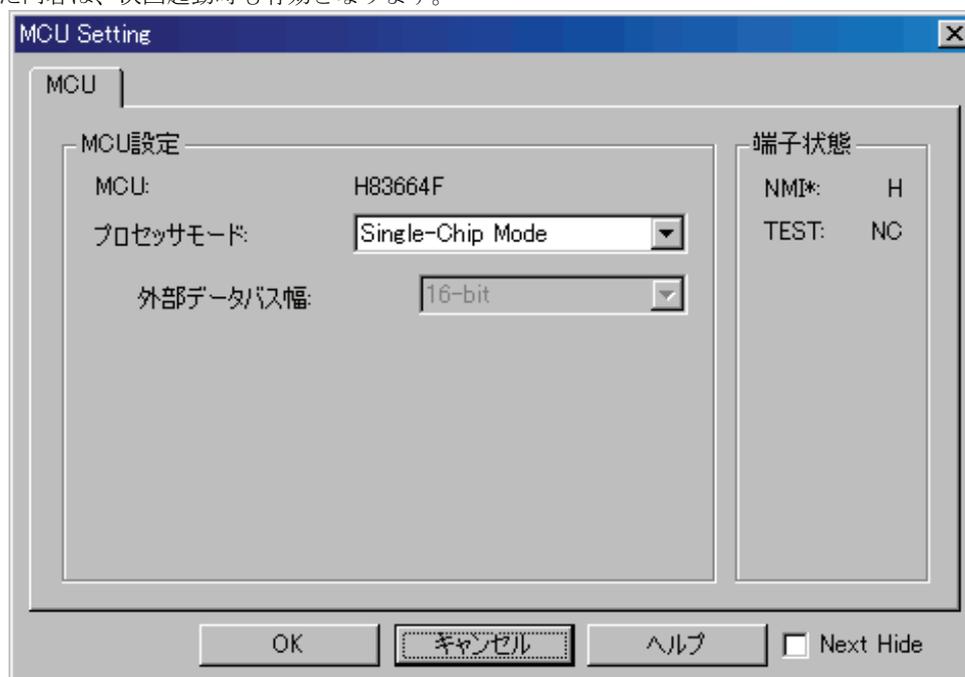
MCU Setting ダイアログは、ユーザターゲットの情報を設定するためのダイアログです。Init ダイアログをクローズした後にオープンします。

MCU Setting ダイアログ下部の Next Hide をチェックすると次回デバッガ起動時にこの MCU Setting ダイアログをオープンしないようにすることができます。また、MCU Setting ダイアログは、以下のいずれかの方法で再表示できます。

- デバッガ起動後、メニュー[基本設定]→[エミュレータ]→[ターゲット...]を選択する。

5.2.1 MCU タブ

指定した内容は、次回起動時にも有効となります。



5.2.1.1 プロセッサモードの指定

ターゲットシステムにあわせて、プロセッサモードを指定してください。



MCU設定

MCU: H83664F

プロセッサモード: Single-Chip Mode

外部データバス幅: 16-bit

以下のいずれかが指定できます。

- Single-chip Mode
シンプルチップモード

5.2.1.2 MCU Status の参照

MCU の各端子の状態を表示します。設定するプロセッサモードと一致しているかを確認できます。



端子状態

NMI*: H

TEST: NC

"NC"表示は、値が不定であることを表します。

[MEMO]

チュートリアル編

このページは白紙です。

6. チュートリアル

6.1 はじめに

本デバッガの主な機能を紹介するために、チュートリアルプログラムを提供しています。このプログラムを用いて各機能を説明します。

このチュートリアルプログラムは、C 言語で書かれており、10 個のランダムデータを昇順/降順にソートします。

チュートリアルプログラムでは、以下の処理を行います。

- `tutorial` 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 `tutorial` 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 `tutorial` 関数で生成した配列を入力し、降順にソートします。

注意事項

- 再コンパイルを行った場合、本章で説明しているアドレスと異なることがあります。

6.2 使用方法

以下のステップに沿ってお進みください。

6.2.1 Step1：デバッガの起動

6.2.1.1 デバッグの準備

High-performance Embedded Workshop を起動し、エミュレータに接続します。
詳細は「**4 デバッグの準備**」を参照ください。

6.2.1.2 デバッガのセットアップ

エミュレータに接続すると、デバッガをセットアップするためのダイアログが表示されます。このダイアログでデバッガの初期設定を行います。
詳細は「**5 デバッガのセットアップ**」を参照ください。

デバッガのセットアップが終了すると、デバッグできる状態になります。

6.2.2 Step2 : RAM の動作チェック

RAM が正常に動作することをチェックします。[メモリ]ウィンドウでメモリ内容を表示、編集し、メモリが正常に動作することを確認します。

注意事項

- マイコンによってはボード上にメモリをつけることができます。この場合、メモリ動作チェックは上記だけでは不完全な場合があります。メモリチェック用プログラムを作成し、チェックすることをお勧めします。

6.2.2.1 RAM の動作チェック

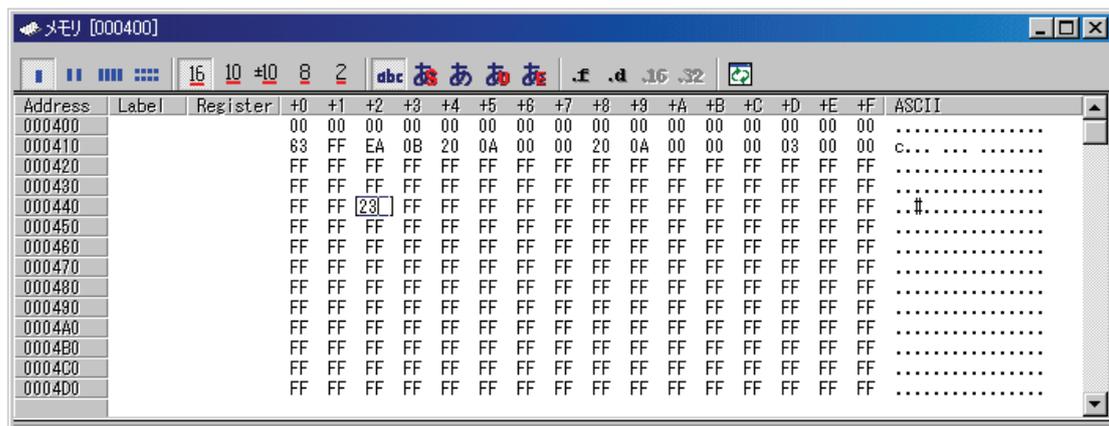
[表示]メニューの[CPU]サブメニューから[メモリ]を選択し、[表示開始アドレス]エディットボックスにRAM のアドレスを入力してください（ここでは"H'400"を入力しています）。[スクロール開始アドレス][スクロール終了アドレス]エディットボックスはデフォルトの設定のままにしておきます（デフォルトの場合、メモリ全空間がスクロール領域になります）。



注意事項

- 各製品ごとにRAM 領域の設定は異なります。各製品のハードウェアマニュアルを参照してください。

[OK]ボタンをクリックしてください。指定されたメモリ領域を示す[メモリ]ウィンドウが表示されます。



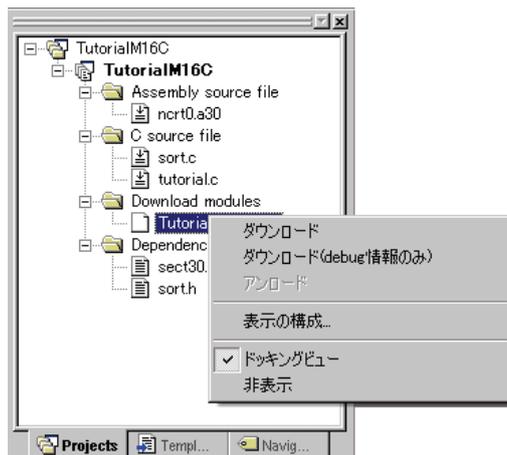
[メモリ]ウィンドウ上のデータ部分をダブルクリックすることにより、値が変更できます。

6.2.3 Step3: チュートリアルプログラムのダウンロード

6.2.3.1 チュートリアルプログラムをダウンロードする

デバッグしたいオブジェクトプログラムをダウンロードします。

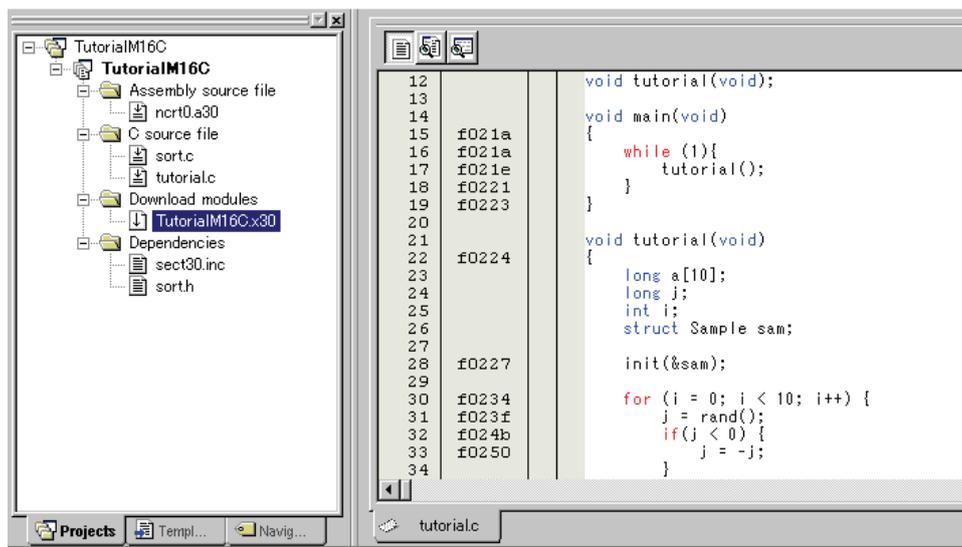
- **H8/300H Tiny 用デバッガの場合**
[Download modules]の[Tutorial.abs]から[ダウンロード]を選択します。



6.2.3.2 ソースプログラムを表示する

本デバッガでは、ソースレベルでプログラムをデバッグできます。

[C source file]の[Tutorial.c]をダブルクリックしてください。[エディタ(ソース)]ウィンドウが開き、"Tutorial.c"ファイルの内容を表示します。



必要であれば、[基本設定]メニューから[表示の形式]オプションを選択し、見やすいフォントとサイズを選択してください。

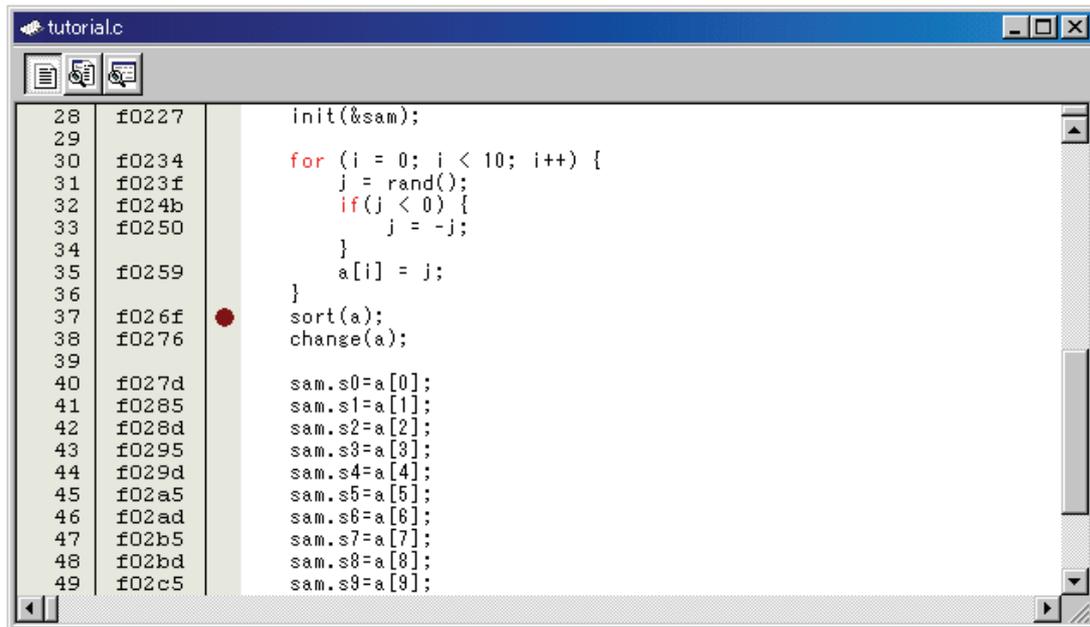
[エディタ(ソース)]ウィンドウは、最初はプログラムの先頭を示しますが、スクロールバーを使って他の部分を見ることができます。

6.2.4 Step4：ブレークポイントの設定

基本的なデバッグ機能の1つにソフトウェアブレークポイントがあります。
[エディタ(ソース)]ウィンドウにおいて、ソフトウェアブレークポイントを簡単に設定できます。

6.2.4.1 ソフトウェアブレークポイントを設定する

例えば、`sort` 関数のコール箇所にソフトウェア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[S/W ブレークポイント]カラムをダブルクリックしてください。



The screenshot shows a debugger window titled 'tutorial.c'. The left pane displays memory addresses and hex values, and the right pane shows the corresponding C code. A red dot is placed in the 'S/W Breakpoint' column next to the line containing the `sort(a);` function call.

```
28 f0227      init(&sam);
29
30 f0234      for (i = 0; i < 10; i++) {
31 f023f          j = rand();
32 f024b          if (j < 0) {
33 f0250              j = -j;
34
35 f0259              a[i] = j;
36          }
37 f026f          sort(a);
38 f0276          change(a);
39
40 f027d          sam.s0=a[0];
41 f0285          sam.s1=a[1];
42 f028d          sam.s2=a[2];
43 f0295          sam.s3=a[3];
44 f029d          sam.s4=a[4];
45 f02a5          sam.s5=a[5];
46 f02ad          sam.s6=a[6];
47 f02b5          sam.s7=a[7];
48 f02bd          sam.s8=a[8];
49 f02c5          sam.s9=a[9];
```

`sort` 関数を含む行に、赤色の印が表示されます。この表示によりソフトウェアブレークポイントが設定されたことを示しています。

6.2.5 Step5：プログラムの実行

プログラムの実行方法について説明します。

6.2.5.1 CPUのリセット

初期状態ではプログラムをダウンロード後、CPU はリセットされていません。

CPU をリセットする場合は、[デバッグ]メニューから[CPU のリセット]を選択するか、ツールバー上の

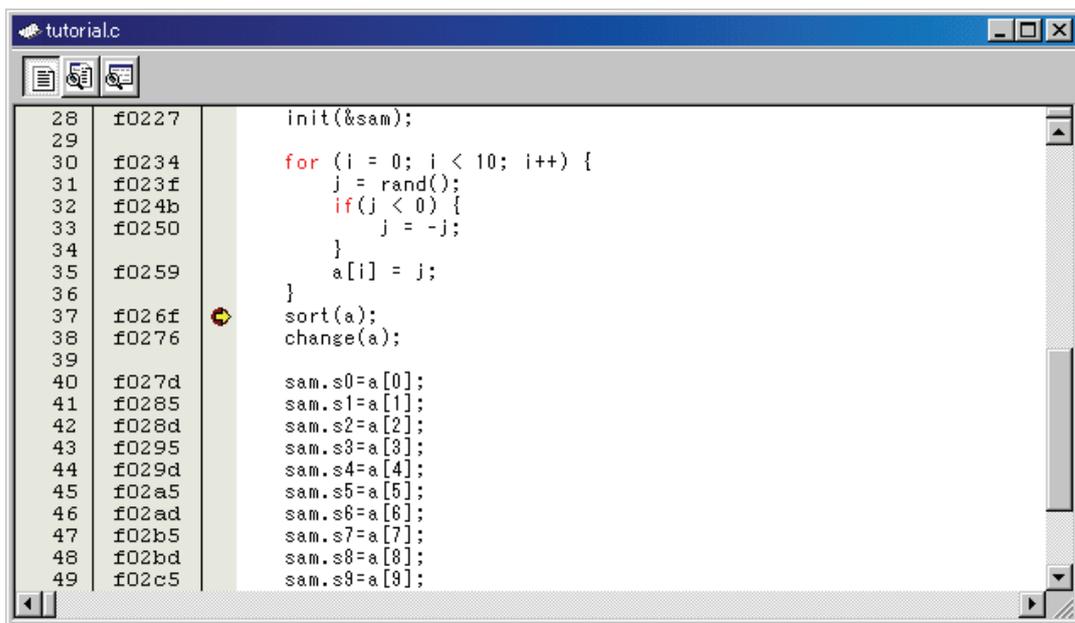
[CPU のリセット]ボタン  を選択してください。

6.2.5.2 プログラムを実行する

プログラムを実行する場合は、[デバッグ]メニューから[実行]を選択するか、ツールバー上の[実行]ボタン

 を選択してください。

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されます。プログラムが停止した位置を示すために[S/W ブレークポイント]カラム中に矢印が表示されます。



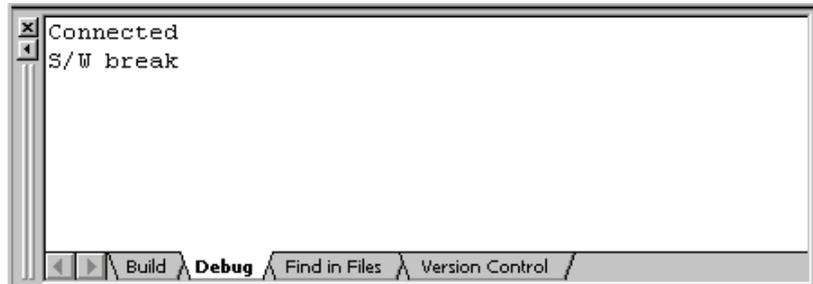
```
28 f0227      init(&sam);
29
30 f0234      for (i = 0; i < 10; i++) {
31 f023f          j = rand();
32 f024b          if(j < 0) {
33 f0250              j = -j;
34
35 f0259              a[i] = j;
36          }
37 f026f      sort(a);
38 f0276      change(a);
39
40 f027d      sam.s0=a[0];
41 f0285      sam.s1=a[1];
42 f028d      sam.s2=a[2];
43 f0295      sam.s3=a[3];
44 f029d      sam.s4=a[4];
45 f02a5      sam.s5=a[5];
46 f02ad      sam.s6=a[6];
47 f02b5      sam.s7=a[7];
48 f02bd      sam.s8=a[8];
49 f02c5      sam.s9=a[9];
```

注意事項

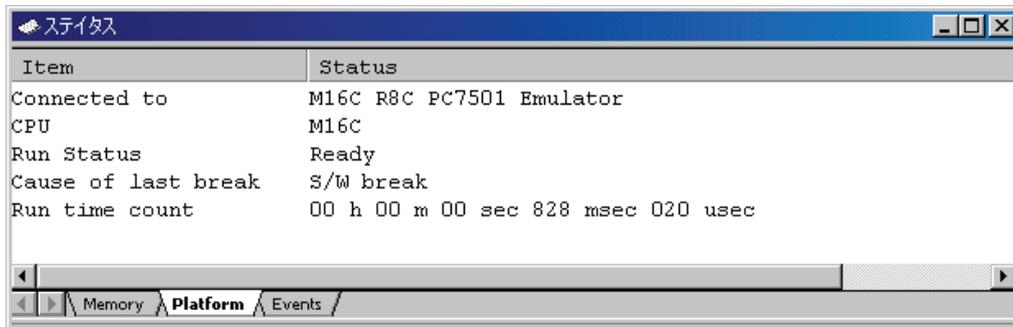
- ブレーク後にソースファイルを表示する際に、ソースファイルパスを問い合わせる場合があります。その場合は、ソースファイルの場所を指定してください。

6.2.5.3 ブレーク要因を確認する

[アウトプット]ウィンドウにブレーク要因を表示します。



また、[ステイタス]ウィンドウでも、最後に発生したブレークの要因を確認できます。
[表示]メニューの[CPU]サブメニューから[ステイタス]を選択してください。[ステイタス]ウィンドウが表示されますので、[Platform]シートを開いて Cause of last break の Status を確認してください。



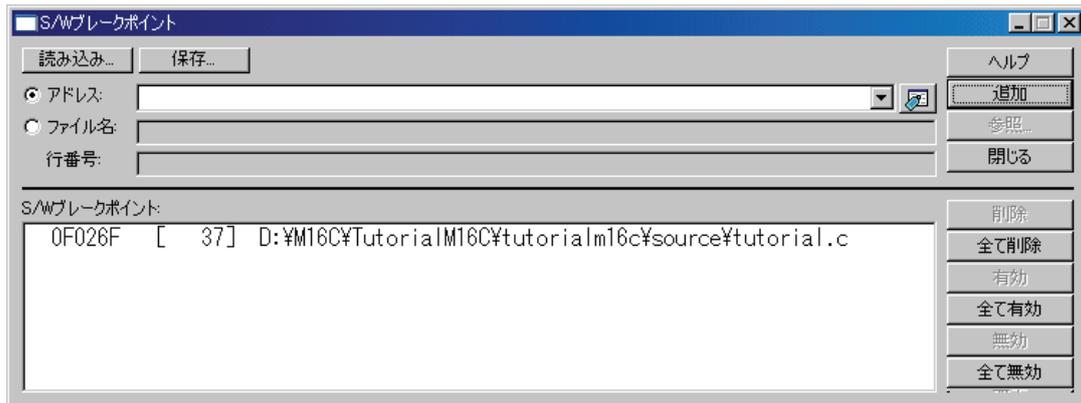
ブレーク要因の表記については、「10 プログラム停止要因の表示」を参照ください。

6.2.6 Step6：ブレイクポイントの確認

設定した全てのソフトウェアブレイクポイントは、[S/W ブレイクポイント]ウィンドウで確認することができます。

6.2.6.1 ブレイクポイントを確認する

[表示]メニューの[ブレイク]サブメニューから[S/W ブレイクポイント]を選択してください。[S/W ブレイクポイント]ウィンドウが表示されます。



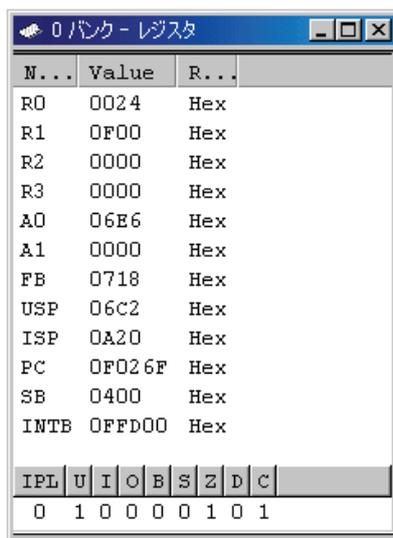
このウィンドウを使って、ブレイクポイントの設定/変更、新しいブレイクポイントの定義、およびブレイクポイントの削除、有効/無効の選択ができます。

6.2.7 Step7：レジスタ内容の確認

レジスタ内容は、[レジスタ]ウィンドウで確認することができます。

6.2.7.1 レジスタ内容を確認する

[表示]メニューの[CPU]サブメニューから[レジスタ]を選択してください。[レジスタ]ウィンドウが表示されます。



N...	Value	R...
R0	0024	Hex
R1	0F00	Hex
R2	0000	Hex
R3	0000	Hex
A0	06E6	Hex
A1	0000	Hex
FB	0718	Hex
USP	06C2	Hex
ISP	0A20	Hex
PC	0F026F	Hex
SB	0400	Hex
INTB	0FFD00	Hex

IPL	U	I	O	B	S	Z	D	C
0	1	0	0	0	0	1	0	1

6.2.7.2 レジスタ内容を変更する

任意のレジスタの内容を変更することができます。

変更するレジスタ行をダブルクリックして下さい。ダイアログが表示しますので、変更する値を入力ください。



PC - レジスタ値設定

値:

基数:

データ形式:

OK キャンセル

6.2.8 Step8：メモリ内容の確認

ラベル名を指定することによって、ラベルが登録されているメモリの内容を[ASM ウォッチ]ウィンドウで確認することができます。

6.2.8.1 メモリ内容を確認する

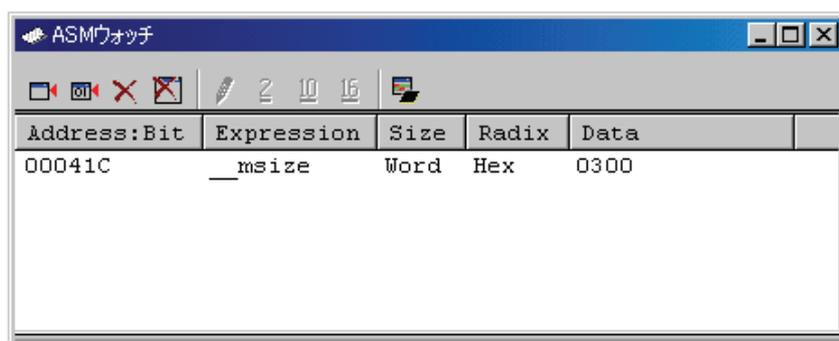
例えば、以下のように、ワードサイズで__msizeに対応するメモリ内容を確認します。

[表示]メニューの[シンボル]サブメニューから[ASM ウォッチ]を選択し、[ASM ウォッチ]ウィンドウを表示します。

[ASM ウォッチ]ウィンドウのポップアップメニュー[追加...]を選択し、[アドレス]エディットボックスに "__msize"を入力し、[サイズ]コンボボックスを"Word"に設定してください。



[OK]ボタンをクリックすると、[ASM ウォッチ]ウィンドウに指定されたメモリ領域が表示されます。



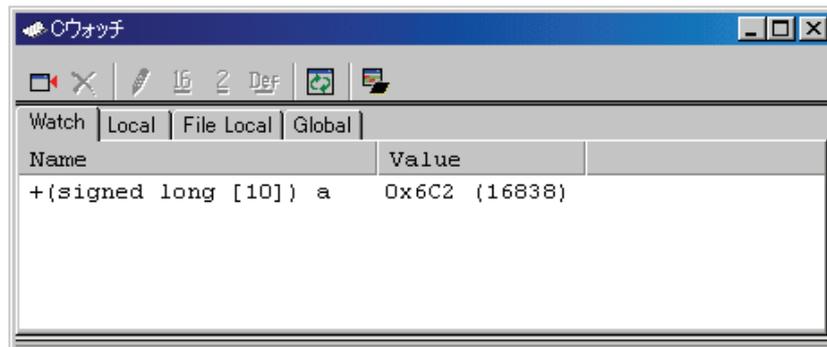
6.2.9 Step9：変数の参照

プログラムをステップ処理するとき、プログラムで使われる変数の値が変化を確認できます。

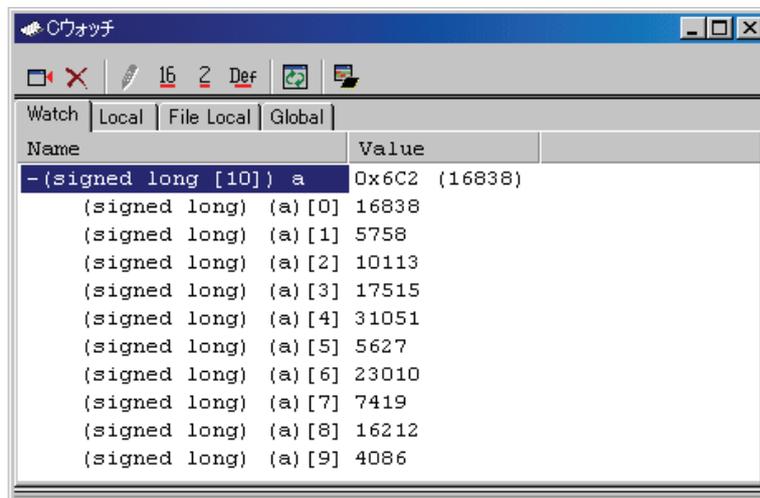
6.2.9.1 変数を参照する

例えば、以下の手順で、プログラムのはじめに宣言した long 型の配列 a を見ることができます。

[エディタ(ソース)]ウィンドウに表示されている配列 a を選択し、マウスの右ボタンで表示するポップアップメニューの[C ウォッチウィンドウに追加]を選択してください。[C ウォッチ]ウィンドウの[Watch]タブが開き、配列 a の内容を表示します。



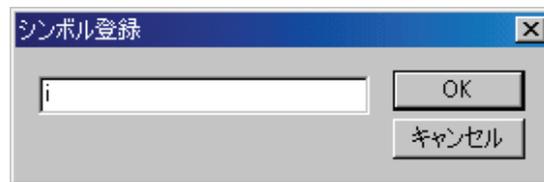
[C ウォッチ]ウィンドウの配列 a の左側にある"+"マークをクリックし、配列 a の各要素を参照することができます。



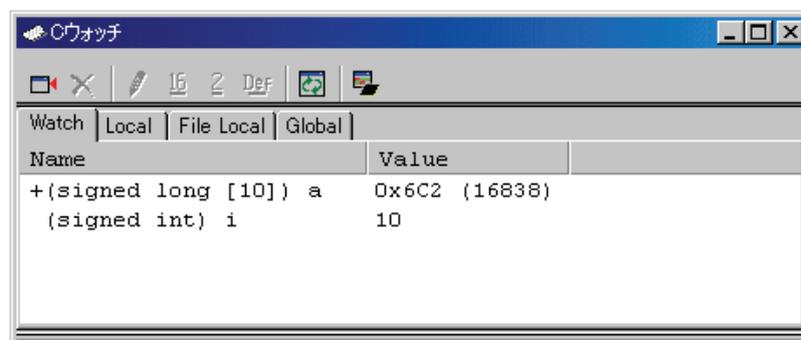
6.2.9.2 参照したい変数を登録する

また、変数名を指定して、[C ウォッチ]ウィンドウに変数を加えることもできます。

[C ウォッチ]ウィンドウのポップアップメニューから[シンボル登録...]を選択してください。以下のダイアログボックスが表示されますので、変数 `i` を入力してください。



[OK]ボタンをクリックすると、[C ウォッチ]ウィンドウに、`int` 型の変数 `i` が表示されます。



6.2.10 Step10：プログラムのステップ実行

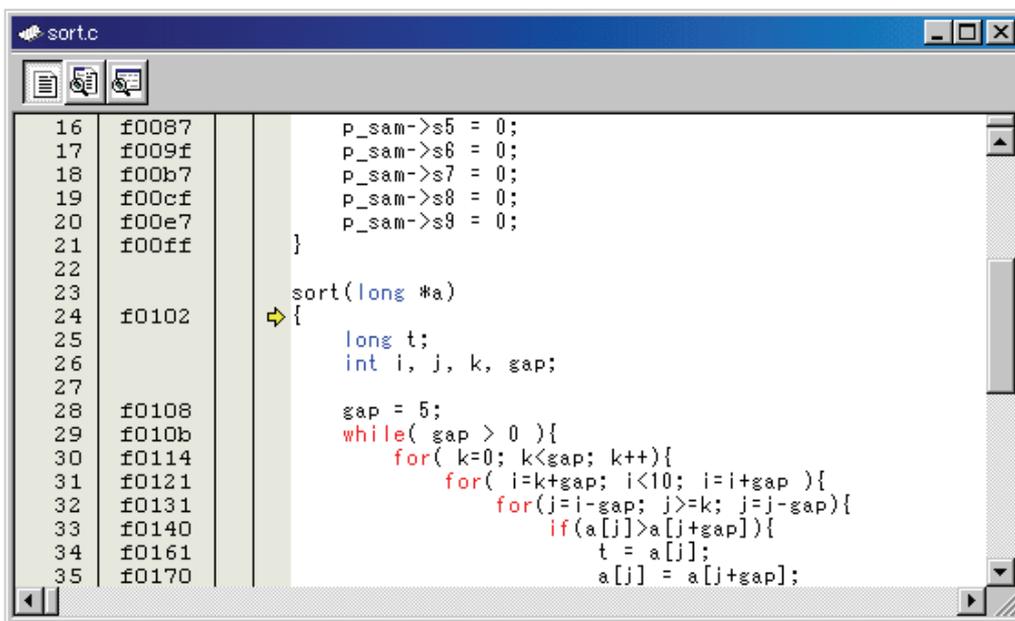
本デバッガは、プログラムのデバッグに有効な各種のステップコマンドを備えています。

1. **ステップイン**
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）。
2. **ステップアウト**
関数を抜け出し、関数を呼び出したプログラムの次のステートメントで停止します。
3. **ステップオーバ**
関数コールを1ステップとして、ステップ実行します。
4. **ステップ...**
指定した速度で指定回数分ステップ実行します。

6.2.10.1 ステップインの実行

ステップイン機能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、[デバッグ]メニューから[ステップイン]を選択するか、またはツールバーの[ステップイン]ボタン  をクリックしてください。
[エディット(ソース)]ウィンドウの PC 位置を示すカーソルが、sort 関数の先頭のステートメントに移動します。



```

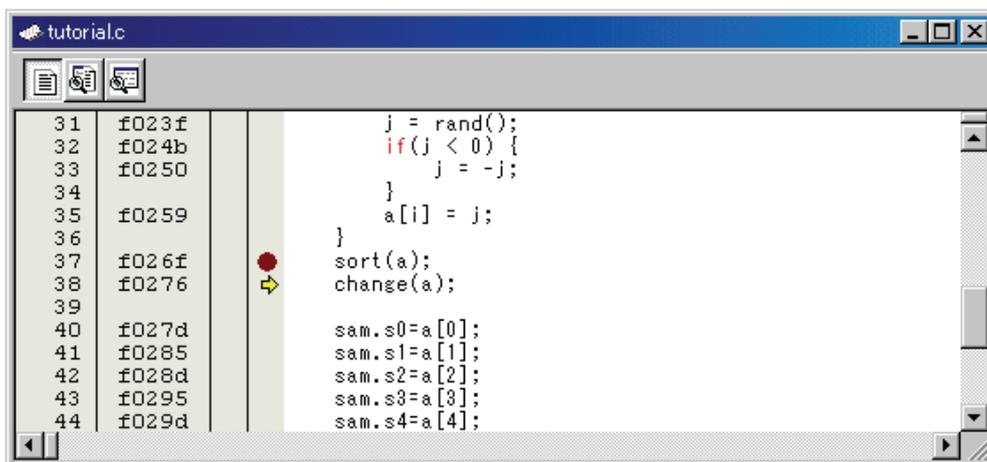
16 f0087 p_sam->s5 = 0;
17 f009f p_sam->s6 = 0;
18 f00b7 p_sam->s7 = 0;
19 f00cf p_sam->s8 = 0;
20 f00e7 p_sam->s9 = 0;
21 f00ff }
22
23 sort(long #a)
24 f0102 {
25     long t;
26     int i, j, k, gap;
27
28     gap = 5;
29     while( gap > 0 ){
30         for( k=0; k<gap; k++){
31             for( i=k+gap; i<10; i=i+gap ){
32                 for(j=i-gap; j>=k; j=j-gap){
33                     if(a[j]>a[j+gap]){
34                         t = a[j];
35                         a[j] = a[j+gap];

```

6.2.10.2 ステップアウトの実行

ステップアウト機能はコール関数の中から抜け出し、コール元プログラム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、[デバッグ]メニューから[ステップアウト]を選択するか、またはツールバーの[ステップアウト]ボタン  をクリックしてください。
[エディット(ソース)]ウィンドウの PC 位置を示すカーソルが、sort 関数を抜け出し、change 関数の手前に移動します。



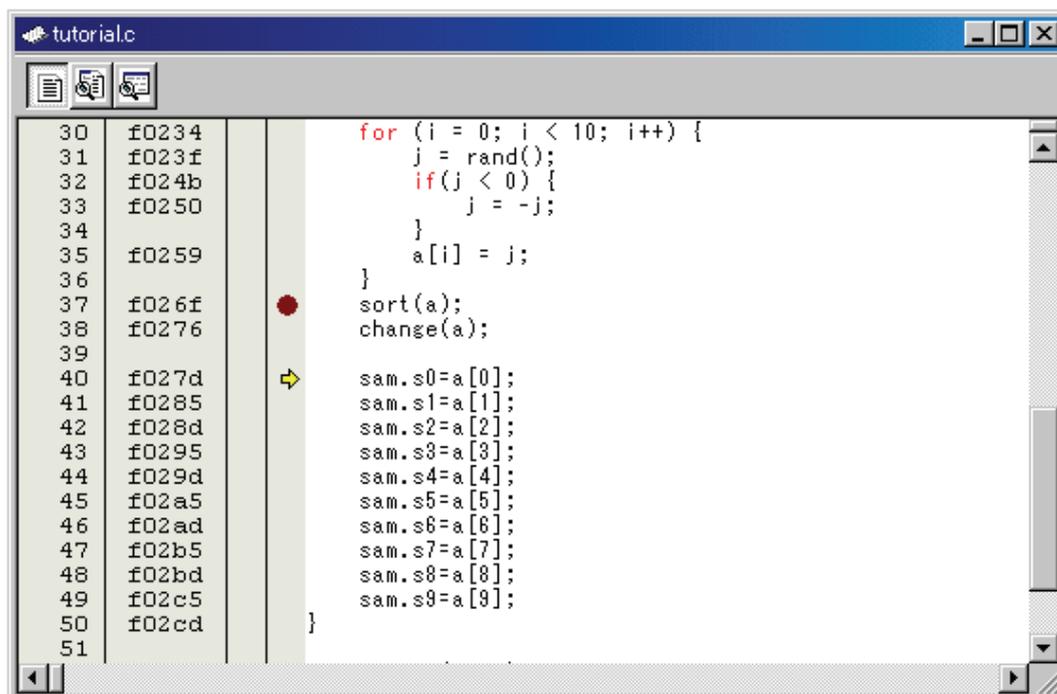
注意事項

- 本機能は処理時間がかかります。コール元が分かっている場合は、[カーソル位置まで実行]をご使用ください。

6.2.10.3 ステップオーバの実行

ステップオーバ機能は関数コールを 1 ステップとして実行して、メインプログラムの次のステートメントで停止します。

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、[デバッグ]メニューから[ステップオーバ]を選択するか、またはツールバーの[ステップオーバ]ボタン  をクリックしてください。[エディット(ソース)]ウィンドウの PC 位置を示すカーソルが、change 関数の次の位置に移動します。



6.2.11 Step11：プログラムの強制ブレーク

本デバッガは、プログラムを強制的にブレークすることができます。

6.2.11.1 プログラムを強制ブレークする

ブレークをすべて解除してください。

main 関数の残り部分を実行するために、[デバッグ]メニューから[実行]を選択するか、ツールバー上の[実行]ボタン  を選択してください。

プログラムは無限ループ処理を実行していますので、強制ブレークするために、[デバッグ]メニューから[プログラムの停止]を選択するか、ツールバー上の[停止]ボタン  を選択してください。

6.2.12 Step12： ローカル変数の表示

[C ウォッチ]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。

6.2.12.1 ローカル変数を表示する

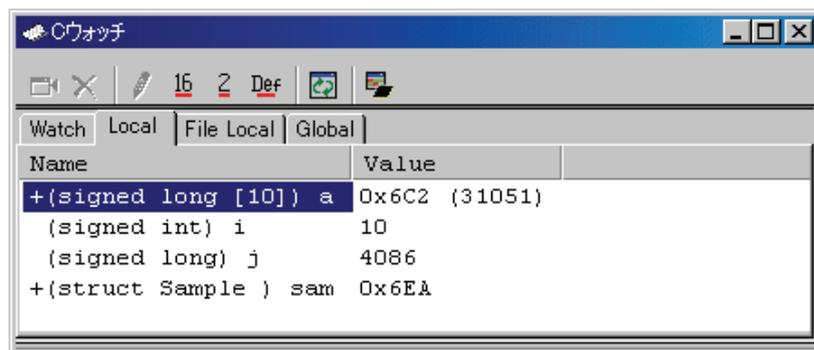
例として、tutorial 関数のローカル変数を調べます。

この関数は、4つのローカル変数 a, j, i, sam を宣言しています。

[表示]メニューの[シンボル]サブメニューから[C ウォッチ]を選択し、[C ウォッチ]ウィンドウを表示します。
[C ウォッチ]ウィンドウには、デフォルトで以下の4つのタブが存在します。

- **[Watch]タブ**
ユーザが登録した変数のみを表示します。
- **[Local]タブ**
現在 PC が存在しているブロックで参照可能なローカル変数がすべて表示されます。プログラム実行によりスコープが変更されると、[Local]タブの内容も切り替わります。
- **[File Local]タブ**
現在 PC が存在しているファイルのファイルローカル変数がすべて表示されます。プログラム実行によりスコープが変更されると、[File Local]タブの内容も切り替わります。
- **[Global]タブ**
ダウンロードしたプログラムで使用しているグローバル変数がすべて表示されます。

ローカル変数を表示する場合は、[Local]タブを選択してください。



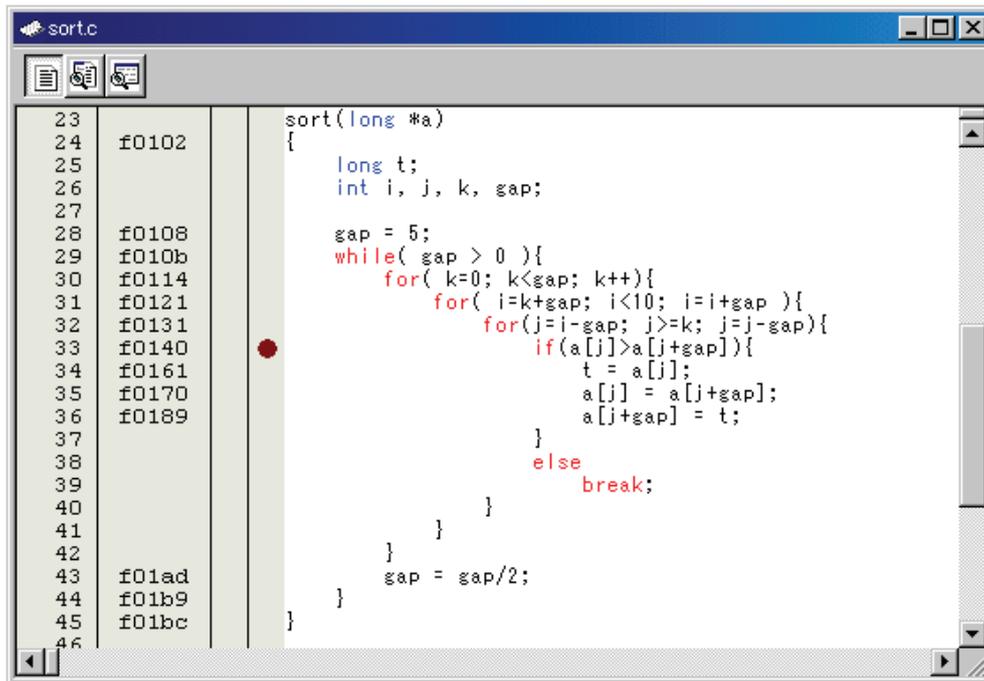
配列 a の左側にある"+"マークをクリックし、配列 a の構成要素を表示させてください。
sort 関数実行前と実行後の配列 a の要素を参照すると、ランダムデータが降順にソートされていることがわかります。

6.2.13 Step13：スタックトレース

本デバッガでは、スタック情報を用いて、現在の PC がある関数がどの関数からコールされているかを表示できます。

6.2.13.1 関数呼び出し状況を参照する

sort 関数内の行の[S/W ブレークポイント]カラムをダブルクリックして、ソフトウェアブレークポイントを設定してください。



プログラムを一旦リセットし、再実行します。[デバッグ]メニューから[リセット後実行]を選択するか、ツールバー上の[リセット後実行]ボタン  を選択してください。

プログラムブレーク後、[表示]メニューの[コード]サブメニューから[スタックトレース]を選択し[スタックトレース]ウィンドウを開いてください。

Kind	Name	Value
F	sort	0F0140
F	tutorial	0F0272
F	main	0F021E

現在 PC が sort()関数内にあり、sort()関数は tutorial()関数からコールされていることがわかります。

6.2.14 さて次は？

このチュートリアルでは、本デバッガの主な使い方を紹介しました。

ご使用のエミュレータで提供されるエミュレーション機能を使用することによって、さらに高度なデバッグを行うこともできます。それによって、ハードウェアとソフトウェアの問題が発生する条件を正確に分離し、識別すると、それらの問題点を効果的に調査することができます。

リファレンス編

このページは白紙です。

7. ウィンドウ一覧

本デバッガ用のウィンドウを以下に示します
ウィンドウ名をクリックするとそのリファレンスを表示します。

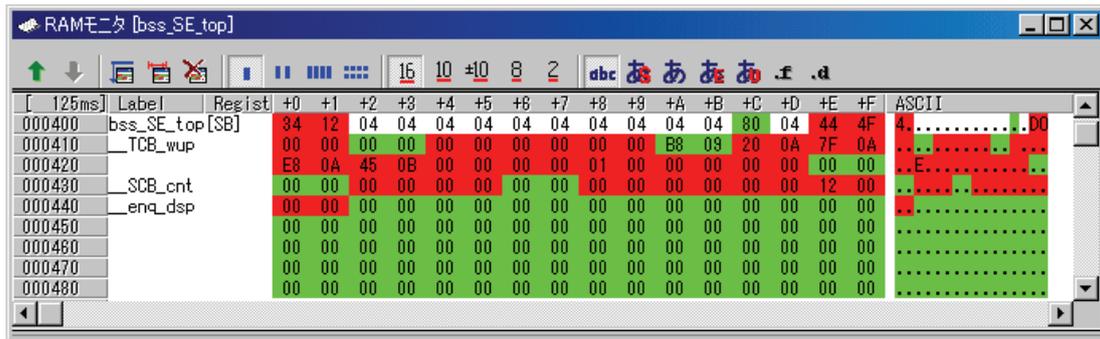
ウィンドウ名	表示用メニュー
RAM モニタウィンドウ	[表示]→[CPU]→[RAM モニタ]
ASM ウォッチウィンドウ	[表示]→[シンボル]→[ASM ウォッチ]
C ウォッチウィンドウ	[表示]→[シンボル]→[C ウォッチ]
H/W ブレークポイント設定ウィンドウ	[表示]→[ブレーク]→[H/W ブレークポイント]
トレースポイント設定ウィンドウ	[表示]→[トレース]→[トレースポイント]
トレースウィンドウ	[表示]→[トレース]→[トレース]
GUI 入出力ウィンドウ	[表示]→[グラフィック]→[GUI I/O]

なお、以下のウィンドウのリファレンスは High-performance Embedded Workshop 本体のヘルプに記載されていますので、そちらをご参照ください。

- 差分ウィンドウ
- マップウィンドウ
- コマンドラインウィンドウ
- ワークスペースウィンドウ
- アウトプットウィンドウ
- 逆アセンブリウィンドウ
- メモリウィンドウ
- IO ウィンドウ
- ステータスウィンドウ
- レジスタウィンドウ
- 画像ウィンドウ
- 波形ウィンドウ
- スタックトレースウィンドウ

7.1 RAM モニタウィンドウ

RAM モニタウィンドウは、ターゲットプログラム実行中のメモリの変化を表示するウィンドウです。リアルタイム RAM モニタ機能を使用し、RAM モニタ領域に該当するメモリ内容をダンプ形式で表示します。表示内容は、ターゲットプログラム実行中に一定間隔(デフォルトは 100msec)で更新されます。



- 1K バイトの RAM モニタ領域を備えています。この RAM モニタ領域は任意の連続アドレス、または、256 バイト単位で 4 ブロックの領域に分割して配置することができます。
- RAM モニタ領域は、任意のアドレス範囲に変更できます。
RAM モニタ領域の変更方法については、RAM モニタ領域を設定するを参照してください。
デフォルトの RAM モニタ領域は、内部 RAM 領域の先頭から 1K バイトの領域に割り当てられています。
- 表示内容の更新間隔はウィンドウごとに設定できます。
ターゲットプログラム実行中の実際の更新間隔は、Address 表示領域のタイトル部分に表示されます。
- データ表示領域及びビコード表示領域の背景色は、アクセス属性によって以下ようになります。

アクセス属性	背景色
Read アクセスされたアドレス	緑色
Write アクセスされたアドレス	赤色
アクセスされていないアドレス	白色

背景色は、変更可能です。

注意事項

- RAM モニタウィンドウには、バスアクセスのデータが表示されます。したがって、外部 I/O からメモリを直接書き換える等、ターゲットプログラムを介さないアクセスによる変化は、表示には反映されません。
- RAM モニタ領域の表示データ長が 1 バイト単位以外の場合、そのデータの 1 バイト単位でメモリに対するアクセス属性が異なる場合があります。このように 1 つのデータの中でアクセス属性が異なる場合は、そのデータが括弧に囲まれて表示されます。また、この時の背景色は、そのデータの 1 バイト目のアクセス属性を示します。

```

001B  00C8  00D2  0000  007C
0000  0000  0000  0000  0000
0000  (007C) FF8C  0000  0000
0000  0000  0000  0050  0000

```

- アクセス属性の表示は、ターゲットプログラムのダウンロードにより初期化されます。
- 表示の更新間隔は、動作状況(以下の要因)によって指定した更新間隔より長くなる場合があります。
 - ホストマシンの性能/負荷状況
 - 通信インタフェース
 - ウィンドウのサイズ(メモリ表示範囲)や表示枚数

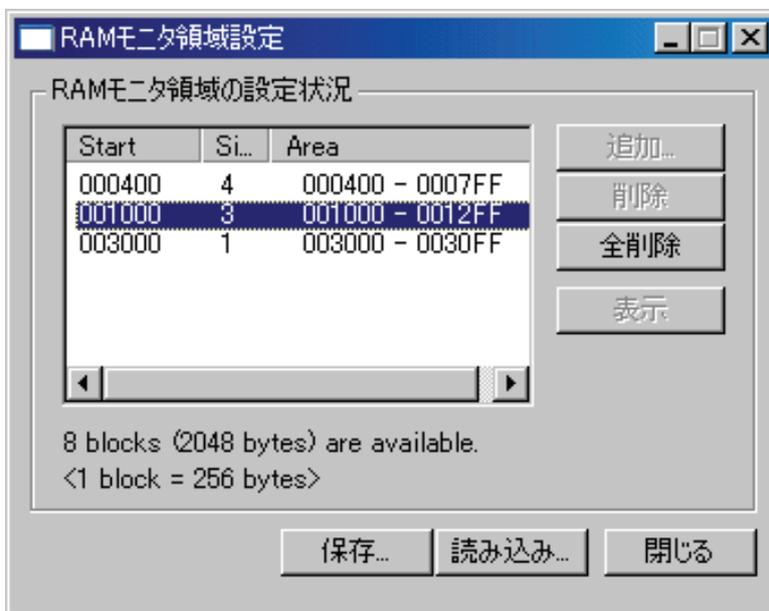
7.1.1 オプションメニュー

ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

メニュー名		機能
RAM モニタ領域設定...		RAM モニタ領域を設定します。
サンプリング周期...		サンプリング周期を設定します。
アクセス履歴の消去		アクセス履歴を消去します。
前方に移動		前方 (アドレスが小さい方) の RAM モニタ領域に表示位置を移動します。
後方に移動		後方 (アドレスが大きい方) の RAM モニタ領域に表示位置を移動します。
表示開始アドレス...		表示開始アドレスを変更します。
スクロール範囲...		スクロール範囲を設定します。
データ長	1byte	1Byte 単位で表示します。
	2byte	2Byte 単位で表示します。
	4byte	4Byte 単位で表示します。
	8byte	8Byte 単位で表示します。
基数	16 進数表示	16 進数で表示します。
	10 進数表示	10 進数で表示します。
	符号付 10 進数表示	符号付 10 進数で表示します。
	8 進数表示	8 進数で表示します。
	2 進数表示	2 進数で表示します。
表示コード	ASCII	ASCII コードで表示します。
	SJIS	SJIS コードで表示します。
	JIS	JIS コードで表示します。
	UNICODE	UNICODE コードで表示します。
	EUC	EUC コードで表示します。
	Float	Float 型で表示します。
	Double	Double 型で表示します。
レイアウト	ラベル	ラベル表示領域の表示/非表示を切り替えます。
	レジスタ	レジスタ表示領域の表示/非表示を切り替えます。
	コード	コード領域の表示/非表示を切り替えます。
カラム...		表示カラム数を変更します。
分割		ウィンドウを分割表示します。
ツールバー表示		ツールバーの表示/非表示を切り換えます。
ツールバーのカスタマイズ...		ツールバーをカスタマイズします。
ドッキングビュー		ウィンドウをドッキングします。
非表示		ウィンドウを非表示にします。

7.1.2 RAM モニタ領域を設定する

RAM モニタウィンドウのポップアップメニュー[RAM モニタ領域設定...]を選択してください。
RAM モニタ領域設定ウィンドウがオープンします。 現在設定されている RAM モニタ領域が、一覧に表示されています。



このウィンドウを使用して、RAM モニタ領域を追加、削除、変更します。

- RAM モニタ領域は、先頭アドレスとサイズ（ブロック数で指定）で指定します。
- 先頭アドレスは、0x100 バイト単位で指定できます。
端数のアドレス値を指定した場合は、0x100 バイト単位で丸め込まれた値が設定されます。
- サイズは、ブロック数で指定します。
コンパクトエミュレータの場合、1 ブロックのサイズは 256 バイトです。また、最大 4 ブロックまで指定できます。
- 使用ブロックの総数が 4 ブロックになるまで、RAM モニタ領域を追加できます。
(リストの下に、現時点で使用可能なブロック数（およびサイズ）が表示されます。)

7.1.2.1 RAM モニタ領域を変更する

RAM モニタ領域の先頭アドレスとサイズを変更できます。

- ダイアログで変更する
RAM モニタ領域の一覧から変更したい RAM モニタ領域を選択し、ダブルクリックしてください。以下のダイアログが表示されますので、[開始アドレス]領域に先頭アドレス、[サイズ]領域にサイズ（ブロック数で指定）を指定してください。



- ウィンドウ内で直接変更する
RAM モニタ領域の一覧から変更したい RAM モニタ領域を選択状態にして、その Start 表示欄または Size 表示欄を再度クリックしてください。エディットボックスが表示されますので、それぞれ変更内容を指定してください。ENTER キーで入力を確定、ESC キーで操作を取り消します。

Start	Size	Area
000400	4	000400 - (
001000	3	001000 - (

アドレスを変更

Start	Size	Area
000400	4	000400 - 0007FF
001000	3	001000 - 0012FF

サイズを変更

7.1.2.2 RAM モニタ領域を追加する

[追加...]ボタンをクリックしてください。

ダイアログが表示されますので、[開始アドレス]領域に先頭アドレス、[サイズ]領域にサイズ（ブロック数で指定）を指定してください。

7.1.2.3 RAM モニタ領域を削除する

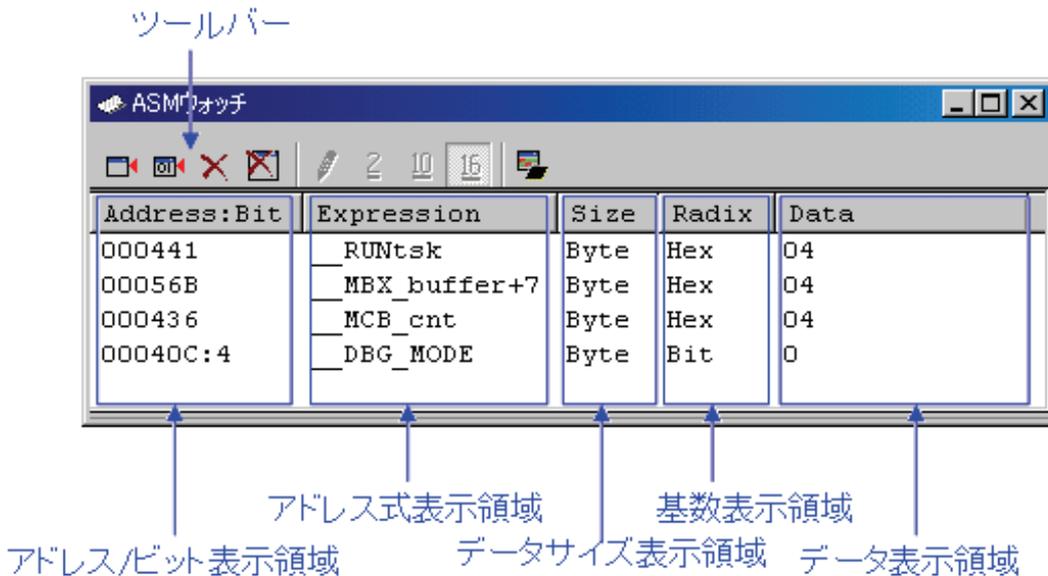
RAM モニタ領域の一覧から削除したい RAM モニタ領域を選択状態にして、[削除]ボタンをクリックしてください。

すべての RAM モニタ領域を削除したい場合は、[全削除]ボタンをクリックしてください。

7.2 ASM ウォッチウィンドウ

ASM ウォッチウィンドウは、ウォッチポイントとして特定のアドレスを登録し、メモリ内容を参照することができるウィンドウです。

登録したアドレスが RAM モニタ領域内であれば、ターゲットプログラム実行中に一定間隔(デフォルトは 100msec)でメモリ内容を更新します



- 登録するアドレスをウォッチポイントと呼びます。以下のいずれかを登録することができます。
 - アドレス(シンボルでの指定可)
 - アドレス+ビット番号
 - ビットシンボル
- 登録したウォッチポイントは、ASM ウォッチウィンドウクローズ時に保存され、再オープン時に自動登録されます。
- ウォッチポイントにシンボル/ビットシンボルを指定した場合、ウォッチポイントのアドレスはターゲットプログラムのダウンロード時に再計算されます。
- 無効なウォッチポイントは"--<not active>--"と表示します。
- (ドラッグ&ドロップ機能により)ウォッチポイントの並び順を変更することができます。
- ウォッチポイントのアドレス式、サイズ、基数、データはインプレイス編集により変更可能です。

注意事項

- RAM モニタは、バスアクセスのデータを取得します。ターゲットプログラムによるアクセス以外の変化は、反映されません。
- RAM モニタ領域の表示データ長が 1 バイト単位以外の場合、そのデータの 1 バイト単位でメモリに対するアクセス属性が異なる場合があります。このような 1 つのデータの中でアクセス属性が統一されていない場合は、そのデータの アクセス属性を正しく表示できません。この時の背景色は、そのデータの 1 バイト目のアクセス属性色となります。

7.2.1 オプションメニュー

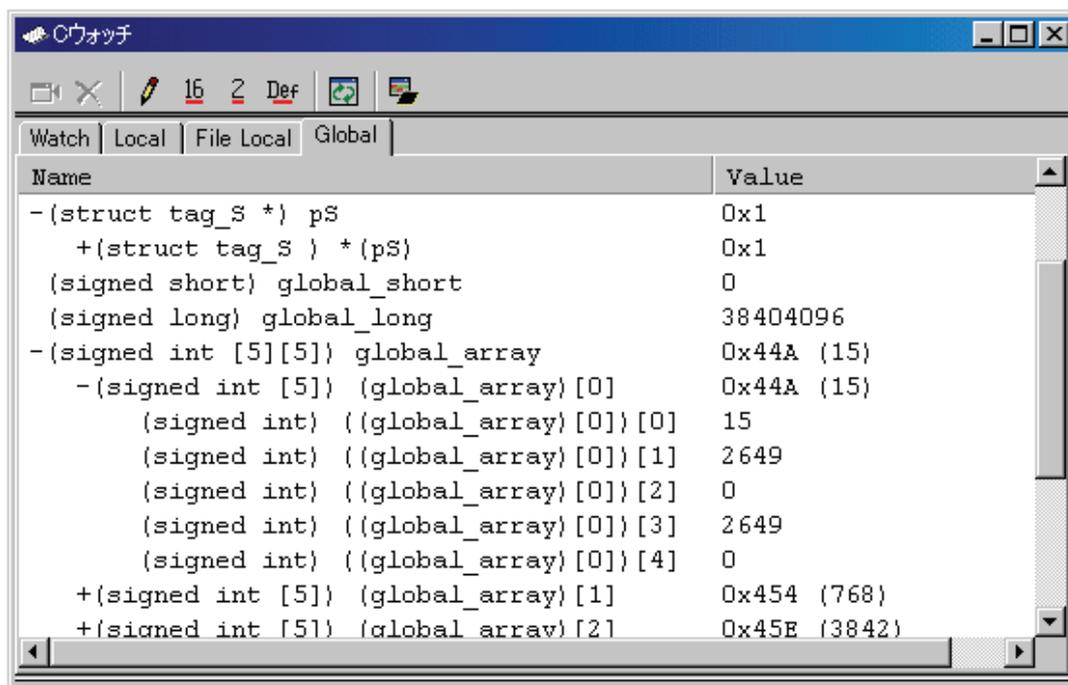
ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

メニュー名		機能
追加...		ウォッチポイントを追加します。
ビットの追加...		ビット形式のウォッチポイントを追加します。
削除		選択したウォッチポイントを削除します。
全て削除		全てのウォッチポイントを削除します。
値の編集...		選択したウォッチポイントの値を編集します。
基数	2進数表示	2進数で表示します。
	10進数表示	10進数で表示します。
	16進数表示	16進数で表示します。
最新の情報に更新		メモリをリフレッシュします。
レイアウト	アドレス	アドレスの表示/非表示を切り替えます。
	サイズ	サイズの表示/非表示を切り替えます。
RAM モニタ	RAM モニタ有効化	RAM モニタ機能を有効にします。
	サンプリング周期...	サンプリング周期を設定します。
ツールバー表示		ツールバーの表示/非表示を切り換えます。
ツールバーのカスタマイズ...		ツールバーをカスタマイズします。
ドッキングビュー		ウィンドウをドッキングします。
非表示		ウィンドウを非表示にします。

7.3 C ウォッチウィンドウ

C ウォッチウィンドウは、C 言語または C++ 言語で作成されたプログラムで使用されている変数を参照するウィンドウです。表示されている変数を C ウォッチポイントと呼びます。

登録したウォッチポイントが RAM モニタ領域内であれば、ターゲットプログラム実行中に一定間隔(デフォルトは 100msec)でメモリ内容を更新します。



- 変数をスコープ別（ローカル、ファイルローカル、グローバル）に参照することができます。
- PC 値の変化に応じて、表示が自動的に更新されます。
- 変数値を変更することができます。
- 変数ごとに表示基数を変更できます。
- 任意の変数を Watch タブに登録し、常時表示することができます。
 - 登録した内容は、プロジェクトごとに保存されます。
 - C ウォッチウィンドウを複数オープンした場合、Watch タブの登録内容は全ウィンドウで共有されます。
- Watch タブを追加し、C ウォッチポイントの登録先を分けることができます。
- ドラッグ&ドロップにより、他のウィンドウやエディタから変数を登録できます。
- 名前順、アドレス順にソートできます。
- RAM モニタ機能を使用し、プログラム実行中にリアルタイムに値を参照できます。

注意事項

- 以下に示す C ウォッチポイントは、値を変更できません。
 - ビットフィールド型変数
 - レジスタ変数
 - メモリの実体(アドレスとサイズ)を示さない C/C++ 言語式
- C/C++ 言語式が正しく計算できない場合(C シンボル未定義等)、無効な C ウォッチポイントとして登録されます。
- Local, File Local, Global タブの表示設定は保存されません。Watch タブ、および、新規に追加したタブの内容は保存されます。
- RAM モニタは、バスアクセスのデータを取得します。ターゲットプログラムによるアクセス以外の変化は、反映されません。
- リアルタイムに更新できるのは、グローバル変数、ファイルローカル変数のみです。
- RAM モニタ領域の表示データ長が 1 バイト単位以外の場合、そのデータの 1 バイト単位でメモリに対するアクセス属性が異なる場合があります。このように 1 つのデータの中でアクセス属性が異なる場合、そのデータの背景色は 1 バイト目のアクセス属性を示します。

7.3.1 オプションメニュー

ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

メニュー名		機能
シンボル登録...		シンボルを追加します。
シンボル削除		選択したシンボルを削除します。
初期化		選択したシンボルを再評価します。
値の編集...		選択したシンボルの値を編集します。
基数	16 進数表示	16 進数で表示します。
	2 進数表示	2 進数で表示します。
	デフォルト	デフォルト基数で表示します。
	トグル(全シンボル)	表示基数を変更します(トグル)。
最新の情報に更新		メモリをリフレッシュします。
型名の非表示		型名を非表示にします。
char* の文字列表示		char* の文字列を表示します。
ソート	名前順	シンボルを名前順に並び替えます。
	アドレス順	シンボルをアドレス順に並び替えます。
RAM モニタ	RAM モニタ有効化	RAM モニタ機能を有効にします。
	サンプリング周期...	サンプリング周期を設定します。
タブの追加...		タブを追加します。
タブの削除		タブを削除します。
ツールバー表示		ツールバーの表示/非表示を切り換えます。
ツールバーのカスタマイズ...		ツールバーをカスタマイズします。
ドッキングビュー		ウィンドウをドッキングします。
非表示		ウィンドウを非表示にします。

7.4 S/W ブレークポイント設定ウィンドウ

S/W ブレークポイント設定ウィンドウは、ソフトウェアブレークポイントを設定するためのウィンドウです。

ソフトウェアブレークは、指定アドレスの命令を実行する手前でブレークします。



- ブレークポイントは、"アドレス"または"ファイル名+行番号"で指定できます。
- ブレークポイントを複数設定した場合、いずれか1点のブレークポイントに到達するとターゲットプログラムを停止します(OR条件)。
- 各ブレークポイントに対して、削除、無効/有効を切り換えることができます。
- ブレークポイント情報は、ファイルに保存することができます。保存したブレークポイント情報を読み込むことも可能です。

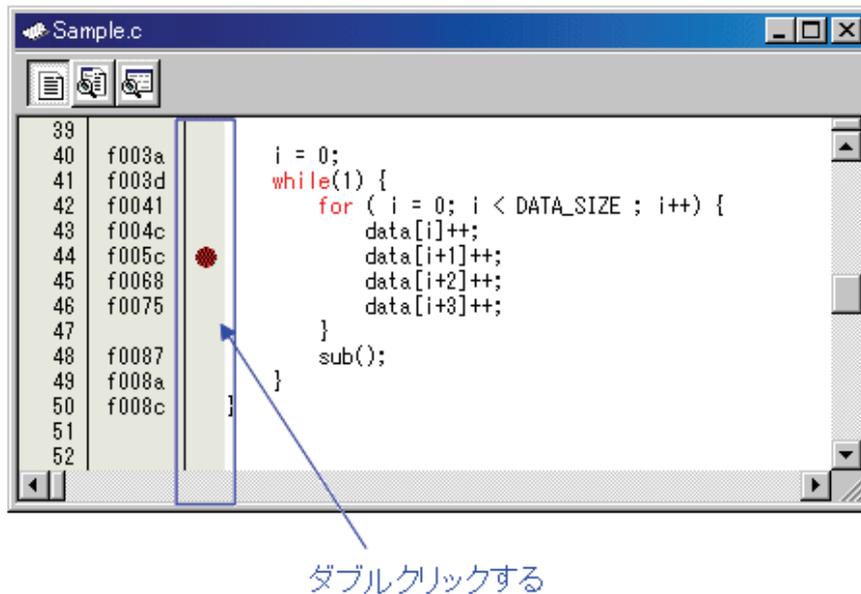
7.4.1 コマンドボタン

ウィンドウ上の各ボタンは、以下の意味を持っています。

ボタン名	機能
読み込み...	ファイルに保存した設定内容を読み込みます。
保存...	ウィンドウで設定した内容をファイルに保存します。
ヘルプ	ヘルプを表示します。
追加	ソフトウェアブレイクポイントを設定します。
参照...	ソースファイルを指定します。
閉じる	ウィンドウを閉じます。
削除	選択したソフトウェアブレイクポイントを解除します。
全て削除	全てのソフトウェアブレイクポイントを解除します。
有効	選択したソフトウェアブレイクポイントを有効にします。
全て有効	全てのソフトウェアブレイクポイントを有効にします。
無効	選択したソフトウェアブレイクポイントを無効にします。
全て無効	全てのソフトウェアブレイクポイントを無効にします。
表示	選択したソフトウェアブレイクポイントの位置をエディタ(ソース)ウィンドウに表示します。

7.4.2 エディタ(ソース)ウィンドウからブレークポイントを設定/解除する

エディタ(ソース)ウィンドウの S/W ブレークポイント設定用カラム上で、ブレークポイントを設定する行をダブルクリックして下さい (設定行に赤丸が表示されます)。



もう一度ダブルクリックするとブレークポイントの設定解除となります(赤丸の表示が消えます)。

エディタ(ソース)ウィンドウには、S/W ブレークポイント設定用カラムがデフォルトで表示されています。非表示にするには、メニュー[編集]→[表示カラムの設定...]で開くダイアログで、[S/W ブレークポイント]チェックボックスをオフにしてください。全てのエディタ(ソース)ウィンドウの、S/W ブレークポイント設定用のカラムが非表示になります。また、エディタ(ソース)ウィンドウのポップアップメニュー[カラム]→[S/W ブレークポイント]を選択することで、個々のエディタ(ソース)ウィンドウ毎にカラムを設定することもできます。

7.5 H/W ブレークポイント設定ウィンドウ

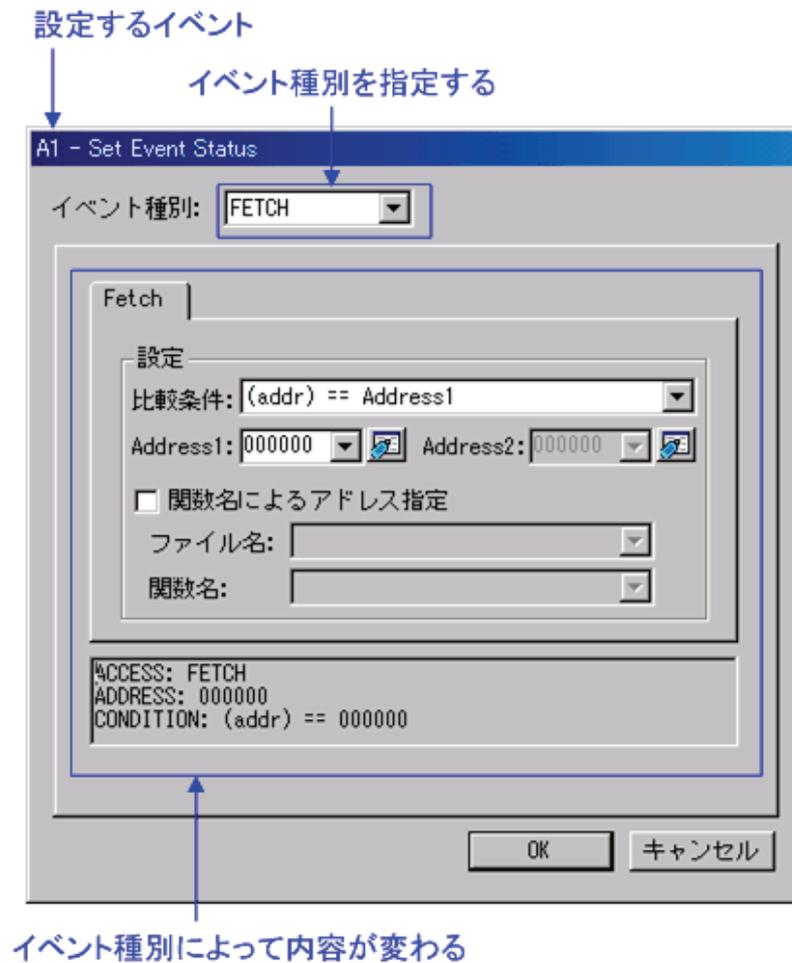
H/W ブレークポイント設定ウィンドウは、エミュレータのハードウェアブレークポイントを設定するウィンドウです。



- ブレークイベントとして、以下のイベントが指定できます。イベントの内容を変更するとタイトルバーに "*" を表示します。エミュレータへの設定後、 "*" は表示しません。
- 命令フェッチ、メモリアクセス、ビットアクセス
- 2点のイベントが使用できます。
ハードウェアブレーク機能とトレース機能で使用するイベントは、コンパクトエミュレータの同じ資源を使用します。 イベントをどちらの機能で使用するかの指定は、Init ダイアログの MCU タブで行います。このウィンドウを使用する場合は、このタブの[トレースポイントを有効にする]チェックボックスの チェックを外してください。
- 複数のイベントは、以下の方法で組み合わせで使用することができます。
 - 有効イベントのうち、すべてのイベントが成立した場合にブレーク(AND 条件)
 - 有効イベントのうち、すべてのイベントが同時に成立した場合にブレーク(同時 AND 条件)
 - 有効イベントのうち、いずれかのイベントが成立した場合にブレーク(OR 条件)
- デバッガ起動時、ハードウェアブレークは無効です。

7.5.1 ブレークイベント指定

イベントを設定するには、H/W ブレークポイント設定ウィンドウの[H/W ブレークを有効にする]チェックボックスをチェックし、 イベント指定領域から変更したいイベント行をダブルクリックします。 ダブルクリックすると以下のダイアログがオープンします。



[イベント種別]の指定により、以下のイベントが設定できます。

FETCHを選択した場合

命令フェッチでブレイクします。

Fetch

設定

比較条件: (addr) == Address1

Address1: _main Address2: 000000

関数名によるアドレス指定

ファイル名:

関数名:

ACCESS: FETCH
ADDRESS: _main
CONDITION: (addr) == 0F0042

DATA ACCESSを選択した場合

メモリアクセスでブレイクさせることができます。

Address Data

設定

比較条件: Data1 <= (data) <= Data2

Data1: 0000 Data2: 0000

アクセス条件: R/W マスク: FFFF

ACCESS: R/W
ADDRESS: _data
CONDITION: (addr) == 00042C, 0000 <= (data) <= 0000

BIT SYMBOL を選択した場合

ビットアクセスでブレイクさせることができます。

ビット

アドレス: 400 ビット: 2

シンボル:

条件

アクセス条件: WRITE

比較データ: 1

ACCESS: WRITE
ADDRESS: _global_float
CONDITION: (addr) == 000400, (data&0004) == 0004

7.5.2 組み合わせ条件指定

組み合わせ条件指定は、組み合わせ条件指定領域から指定します。

AND,OR を選択した場合

イベント指定領域で使用するイベントとそのパスカウント(通過回数)が指定できます。パスカウント(通過回数)を変更するには、変更するイベントを選択した状態でそのイベントのパスカウント値をクリックしてください。

H/Wブレイクを有効にする

PASS	E..	ADDRESS	AC...	
<input checked="" type="checkbox"/>	1	A1	000000	FETCH
<input type="checkbox"/>	1	A2	000000	FETCH
<input checked="" type="checkbox"/>	1	A3	000000	FETCH
<input type="checkbox"/>	1	A4	000000	FETCH
<input type="checkbox"/>	1	A5	000000	FETCH
<input type="checkbox"/>	1	A6	000000	FETCH

AND(Same Time)を選択した場合

イベント指定領域で使用するイベントが指定できます。パスカウント(通過回数)は指定できません。

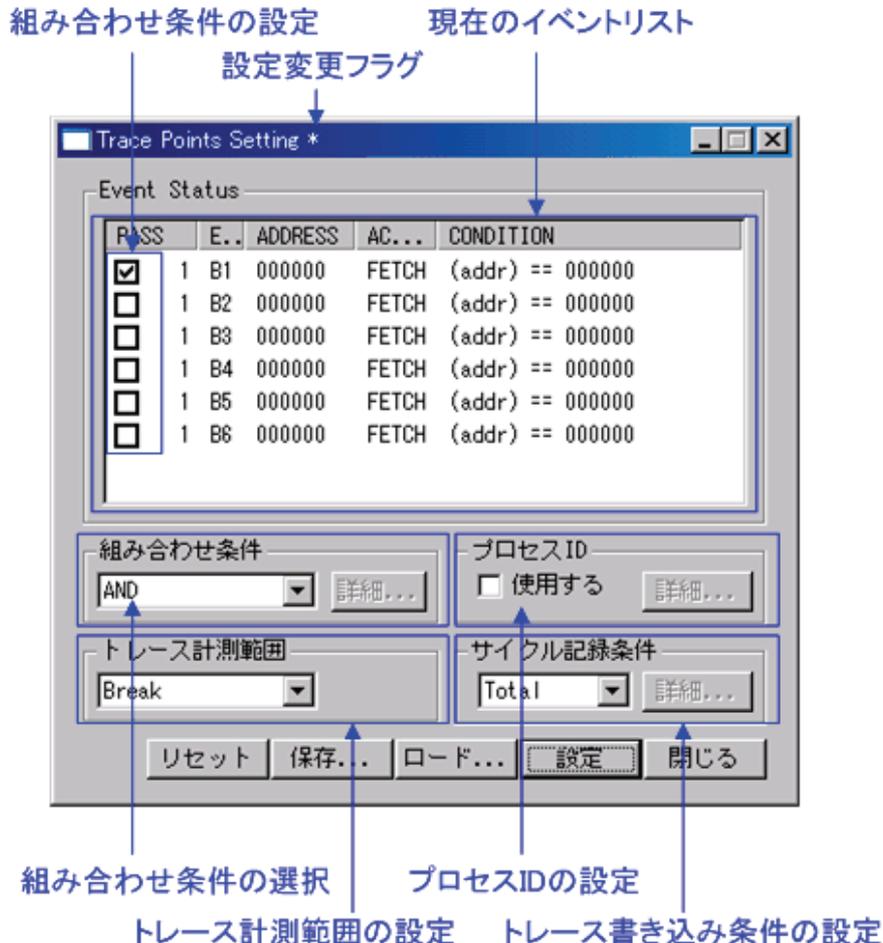
**7.5.3 コマンドボタン**

ウィンドウ上の各ボタンは、以下の意味を持っています。

ボタン名	機能
リセット	ウィンドウに表示中の内容を破棄し、エミュレータに設定されている内容をロードします。
保存...	ウィンドウで設定した内容をファイルに保存します。
読込...	ファイルに保存したイベント情報をロードします。
設定	ウィンドウで設定した内容をエミュレータに送信します。
閉じる	ウィンドウを閉じます。

7.6 トレースポイント設定ウィンドウ

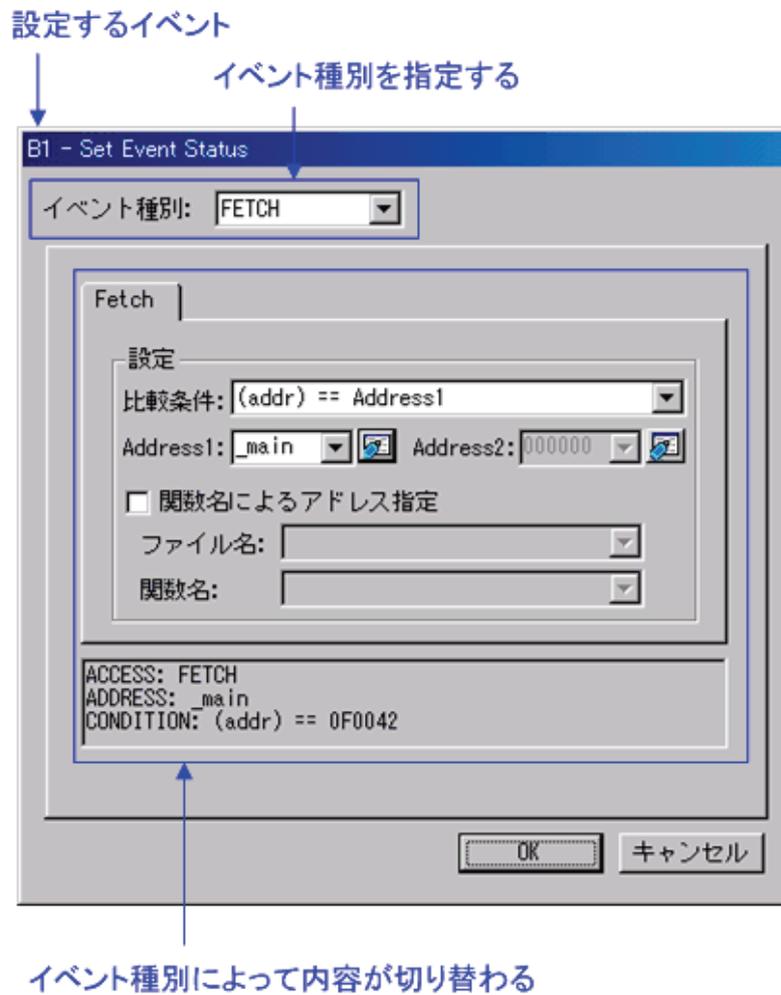
トレースポイント設定ウィンドウは、トレースポイントを設定するウィンドウです。



- トレースイベントとして、以下のイベントが指定できます。イベントの内容を変更するとタイトルバーに "*" を表示します。エミュレータへの設定後、 "*" は表示しません。
- 命令フェッチ、メモリアクセス、ビットアクセス
- 2点のイベントが使用できます。
ハードウェアブレイク機能とトレース機能で使用するイベントは、コンパクトエミュレータの同じ資源を使用します。 イベントをどちらの機能で使用するかの指定は、Init ダイアログの MCU タブで行います。このウィンドウを使用する場合は、このタブの[トレースポイントを有効にする]チェックボックスを チェックしてください。
- 複数のイベントは、以下の方法で組み合わせで使用することができます。
 - 有効イベントのうち、すべてのイベントが成立した場合にトレース(AND 条件)
 - 有効イベントのうち、すべてのイベントが同時に成立した場合にトレース(同時 AND 条件)
 - 有効イベントのうち、いずれかのイベントが成立した場合にトレース(OR 条件)

7.6.1 トレースイベント指定

イベントを設定するには、トレースポイント設定ウィンドウのイベント指定領域から変更したい イベント行をダブルクリックします。ダブルクリックすると以下のダイアログがオープンします。



[イベント種別]の指定により、以下のイベントが設定できます。

FETCH を選択した場合

命令フェッチでトレースします。

Fetch

設定

比較条件: (addr) == Address1

Address1: _main Address2: 000000

関数名によるアドレス指定

ファイル名:

関数名:

ACCESS: FETCH
ADDRESS: _main
CONDITION: (addr) == 0F0042

DATA ACCESS を選択した場合

メモリアクセスでトレースさせることができます。

Address Data

設定

比較条件: Data1 <= (data) <= Data2

Data1: 0000 Data2: 0000

アクセス条件: R/W マスク: FFFF

ACCESS: R/W
ADDRESS: _data
CONDITION: (addr) == 00042C, 0000 <= (data) <= 0000

BIT SYMBOL を選択した場合

ビットアクセスでトレースさせることができます。

ビット

アドレス: 400 シンボル:

ビット: 2

条件

アクセス条件: WRITE

比較データ: 1

```
ACCESS: WRITE
ADDRESS: _global_float
CONDITION: (addr) == 000400, (data&0004) == 0004
```

7.6.2 組み合わせ条件指定

組み合わせ条件指定は、組み合わせ条件指定領域から指定します。

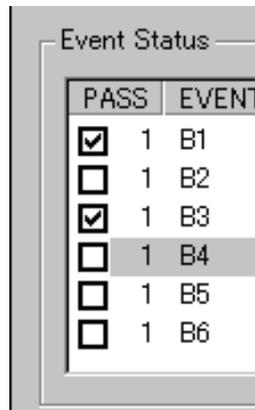
AND,OR を選択した場合

イベント指定領域で使用するイベントとそのパスカウントが指定できます。パスカウントを変更するには、変更するイベントを選択した状態でそのイベントのパスカウント値をクリックしてください。

Event Status		
PASS	EVENT	
<input checked="" type="checkbox"/>	1	B1
<input type="checkbox"/>	1	B2
<input checked="" type="checkbox"/>	1	B3
<input type="checkbox"/>	1	B4
<input type="checkbox"/>	1	B5
<input type="checkbox"/>	1	B6

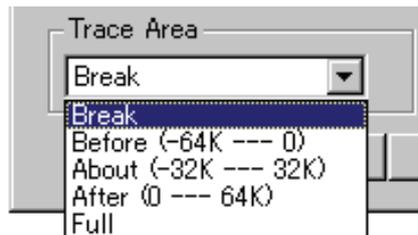
AND(Same Time)を選択した場合

イベント指定領域で使用するイベントが指定できます。パスカウントは指定できません。



7.6.3 トレース範囲指定

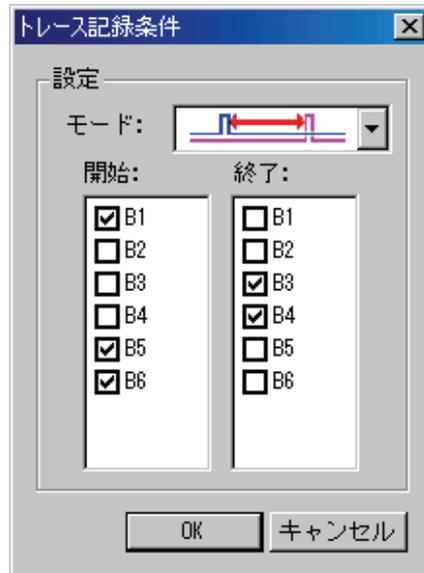
トレースイベントに対して、トレース範囲を指定することができます。
コンパクトエミュレータをご使用の場合、64K サイクル分を記録することができます。



Break	ターゲットプログラムが停止するまでの 64K サイクルを記録します。
Before	トレース条件成立まで 64K サイクルを記録します。
About	トレース条件成立の前後 32K サイクルを記録します。
After	トレース条件成立後の 64K サイクルを記録します。
Full	トレース開始からの 64K サイクルを記録します。

7.6.4 トレース書き込み条件設定

トレースメモリに書き込むサイクルの条件を指定することができます。



Total	全てのサイクルを書き込みます。
Pick up	指定した条件が成立したサイクルのみを書き込みます。
Exclude	指定した条件が非成立したサイクルのみを書き込みます。

また、書き込みモードとして、以下の3種類をサポートしています。

	指定イベント成立サイクルのみ
	指定イベント成立から指定イベント非成立までのサイクル
	開始イベント成立から終了イベント成立までのサイクル

7.6.5 コマンドボタン

ウィンドウ上の各ボタンは、以下の意味を持っています。

ボタン名	機能
リセット	ウィンドウに表示中の内容を破棄し、エミュレータに設定されている内容をロードします。
保存...	ウィンドウで設定した内容をファイルに保存します。
読込...	ファイルに保存したイベント情報を読み込みます。
設定	ウィンドウで設定した内容をエミュレータに送信します。
閉じる	ウィンドウを閉じます。

7.7 トレースウィンドウ

トレースウィンドウは、リアルタイムトレース計測結果を表示するウィンドウです。以下の表示形式で表示できます。

バスモード

サイクルごとのバス情報を参照できます。表示内容は、ご使用の MCU、エミュレータシステムに依存します。バス情報に加えて、逆アセンブル情報、ソース行情報、データアクセス情報を混合表示できます。

逆アセンブルモード

実行した命令を参照できます。逆アセンブル情報に加えて、ソース行情報、データアクセス情報を混合表示できます。

データアクセスモード

データの R/W サイクルを参照できます。データアクセス情報に加えて、ソース行情報を混合表示できます。

ソースモード

プログラムの実行経路をソースプログラム上で参照できます。

トレース計測が終了した時点で計測結果を表示します。トレース計測が再開されると、ウィンドウ表示はクリアされます。

トレース計測範囲は、トレースポイント設定ウィンドウで変更できます。トレースポイント設定ウィンドウの詳細については、「トレースポイント設定ウィンドウのリファレンス」を参照してください。初期状態では、プログラム停止直前の情報が記録されます。

7.7.1 バスモードの構成

バスモードが選択されている場合、バスモードの表示になります。バスモードは、以下の構成になっています。

バスモードの表示内容は、ご使用の MCU やエミュレータシステムにより異なります。

Cycle	Label	Address	Data	BUS	BIU	R/W	RWT	CPU	QN	B-T	Q-T	76543210	h' m' s: ms. us
-26525		0F011C	4A0C	16b	--	-	1	--	3	1	1	11111111	00'00'00:157.897
-26524		0F011E	C904	16b	IW	R	0	CW	3	1	1	11111111	00'00'00:157.897
-26523		0F011E	C904	16b	--	-	1	--	3	1	1	11111111	00'00'00:157.897
-26522		0F0120	FC1B	16b	IW	R	0	RW	3	1	1	11111111	00'00'00:157.898
-26521		0F0120	FC1B	16b	--	-	1	--	3	1	1	11111111	00'00'00:157.898
-26520	_global_arre	00044A	0000	16b	DW	W	0	CW	1	1	1	11111111	00'00'00:157.898
-26519		0007E2	0000	16b	DW	R	0	RB	0	1	1	11111111	00'00'00:157.898
-26518		0F0122	E4FE	16b	IW	R	0	--	2	1	1	11111111	00'00'00:157.898
-26517		0F0124	1BC9	16b	IW	R	0	--	4	1	1	11111111	00'00'00:157.898
-26516		0007E2	0001	16b	DW	W	0	CB	3	1	1	11111111	00'00'00:157.898
-26515		0007E2	0001	16b	--	-	1	RB	2	1	1	11111111	00'00'00:157.898
-26514		0F0107	D1FF	16b	IB	R	0	QC	1	1	1	11111111	00'00'00:157.899
-26513		0F0108	FC5B	16b	IW	R	0	--	3	1	1	11111111	00'00'00:157.899
-26512		0F0108	FC5B	16b	--	-	1	--	3	1	1	11111111	00'00'00:157.899
-26511		0F010A	CA7D	16b	IW	R	0	CW	3	1	1	11111111	00'00'00:157.899
-26510		0007E2	0001	16b	DW	R	0	RB	2	1	1	11111111	00'00'00:157.899
-26509		0F010C	7318	16b	IW	R	0	--	4	1	1	11111111	00'00'00:157.899
-26508		0F010E	FEB4	16b	IW	R	0	CW	4	1	1	11111111	00'00'00:157.899
-26507		0F010E	FEB4	16b	--	-	1	RB	3	1	1	11111111	00'00'00:157.899
-26506		0F0110	547D	16b	IW	R	0	CW	3	1	1	11111111	00'00'00:157.900

- (1) **サイクル表示領域：**
トレースサイクルを表示します。ダブルクリックすると、表示サイクルを変更するためのダイアログボックスが表示されます。
- (2) **ラベル表示領域：**
アドレスバス情報に対応するラベルを表示します。ダブルクリックすると、アドレスを検索するためのダイアログボックスが表示されます。
- (3) **バス情報表示領域：**
表示内容は、ご使用の MCU やエミュレータシステムにより異なります。
- (4) **時間情報表示領域：**
トレース計測結果の時間情報を表示します。以下の 3 通りの方法をメニューから選択できます。
 - **Absolute Time：**
プログラム実行開始時点からの経過時間を絶対時間で表示します（デフォルト）。
 - **Differences：**
直前のサイクルからの差分時間を表示します。
 - **Relative Time：**
選択したサイクルからの相対時間を表示します。なお、トレース計測結果が更新されると、絶対時間表示に変更されます。
- (5) **取得済みトレース計測結果の範囲：**
現在取得されているトレース計測結果の範囲を表示します。
- (6) **トレース計測範囲：**
現在設定されているトレース計測範囲を表示します。
- (7) **先頭行のサイクル：**
表示先頭行のサイクルを表示します。
- (8) **先頭行のアドレス：**
表示先頭行のアドレスを表示します。
- (9) **先頭行の時間：**
表示先頭行の時間を表示します。
- (10) **ウィンドウ分割ボックス：**
ダブルクリックするとウィンドウを分割表示します。

バス情報に加えて、逆アセンブル情報、ソース行情報、データアクセス情報を混合表示できます。次のような表示になります。

Cycle	Label	Address	Data	BUS	BIU	R/W	RWT	CPU	QN	B-T	Q-T	76543210	DataAccess
	exe.c, 38:												
		0F024C											
-00080		0007D9	0000	16b	DW	W	0	CW	2	1	1	11111111	{0007D9 00 W }
-00079		0007DA	0000	16b	DW	W	0	--	2	1	1	11111111	{0007DA 00 W }
-00078		0007DD	04FF	16b	DW	R	0	RB	1	1	1	11111111	{0007DD 04 R }
-00077		0007DE	FF00	16b	DW	R	0	--	1	1	1	11111111	{0007DE 00 R }
-00076		0007D9	00FF	16b	DW	R	0	RB	0	1	1	11111111	{0007D9 00 R }
-00075		0007DA	FF00	16b	DW	R	0	--	0	1	1	11111111	{0007DA 00 R }
-00074		0F0250	CA7D	16b	IW	R	0	--	2	1	1	11111111	

7.7.2 逆アセンブルモードの構成

バスモードが選択されておらず、逆アセンブルモードが選択されている場合、逆アセンブルモードの表示になります。逆アセンブルモードは、以下の構成になっています。

Cycle	Address	Obj-code	Label	Mnemonic	h" m' s: ms. us
-00080	0F024C	C1BBFEFA		CMP.W -2H[FB],-6H[FB]	00"00'00:161.203
-00072	0F0250	7DCA09		JGE F025BH	00"00'00:161.204
-00070	0F0253	C91BFC		ADD.W #1H,-4H[FB]	00"00'00:161.204
-00066	0F0256	C91BFA		ADD.W #1H,-6H[FB]	00"00'00:161.205
-00061	0F0259	FEF2		JMP.B F024CH	00"00'00:161.205
-00057	0F024C	C1BBFEFA		CMP.W -2H[FB],-6H[FB]	00"00'00:161.206
-00051	0F0250	7DCA09		JGE F025BH	00"00'00:161.206
-00049	0F0253	C91BFC		ADD.W #1H,-4H[FB]	00"00'00:161.207
-00045	0F0256	C91BFA		ADD.W #1H,-6H[FB]	00"00'00:161.207
-00040	0F0259	FEF2		JMP.B F024CH	00"00'00:161.208
-00036	0F024C	C1BBFEFA		CMP.W -2H[FB],-6H[FB]	00"00'00:161.208

(1) (2) (3) (4)

- (1) **アドレス表示領域：**
命令に対応するアドレスを表示します。ダブルクリックすると、アドレスを検索するためのダイアログボックスが表示されます。
- (2) **オブジェクトコード表示領域：**
命令のオブジェクトコードを表示します。
- (3) **ラベル表示領域：**
命令のアドレスに対応するラベルを表示します。ダブルクリックすると、アドレスを検索するためのダイアログボックスが表示されます。
- (4) **ニーモニック表示領域：**
命令のニーモニックを表示します。

その他の表示はバスモードと同様です。

逆アセンブル情報に加えて、ソース行情報、データアクセス情報を混合表示できます。次のような表示になります。

Cycle	Address	Obj-code	Label	Mnemonic	DataAccess	h" m' s: ms. us
-00080	exe.c, 38: 0F024C	C1BBFEFA		for(cnt=0; cnt<loopent; cnt++) CMP.W -2H[FB],-6H[FB]	(0007D9 00 W) (0007DA 00 W) (0007DD 04 R) (0007DE 00 R) (0007D9 00 R) (0007DA 00 R)	00"00'00:161.203
-00072	0F0250	7DCA09		JGE F025BH		00"00'00:161.204
-00070	exe.c, 39: 0F0253	C91BFC		{ ADD.W #1H,-4H[FB]	(0007DB 00 R)	00"00'00:161.204

7.7.3 データアクセスモードの構成

バスモードと逆アセンブルモードが選択されておらず、データアクセスモードが選択されている場合、データアクセスモードの表示になります。データアクセスモードは、以下の構成になっています。

Cycle	Label	DataAccess	h' m' s: ms. us
-00059		(0F023C 0517 R)	00"00'00:004.056
-00050	__RUNtsk	(000441 01 R)	00"00'00:004.057
-00041	__TCB_sp	(00041A 09B8 W)	00"00'00:004.058
-00032	__FCB_flg	(00042E 0000 R)	00"00'00:004.060
-00029	__FCB_flg	(00042E 0001 W)	00"00'00:004.060
-00024	__FCB_flgQ	(000541 03 R)	00"00'00:004.061
-00016	__FCB_nxt_tsk	(00055D 00 W)	00"00'00:004.062
-00015		(000546 03 R)	00"00'00:004.062
-00012	__FCB_flgQ	(000541 03 R)	00"00'00:004.062
-00003	__FCB_flg	(00042E 0001 R)	00"00'00:004.063
00000		(000555 03 R)	00"00'00:004.064

(1)

(1) データアクセス表示領域：

データアクセス情報を表示します。"(000400 1234 W)" と表示されている場合、000400H 番地にデータ 1234H が 2 バイト幅でライトされたことをあらわします。

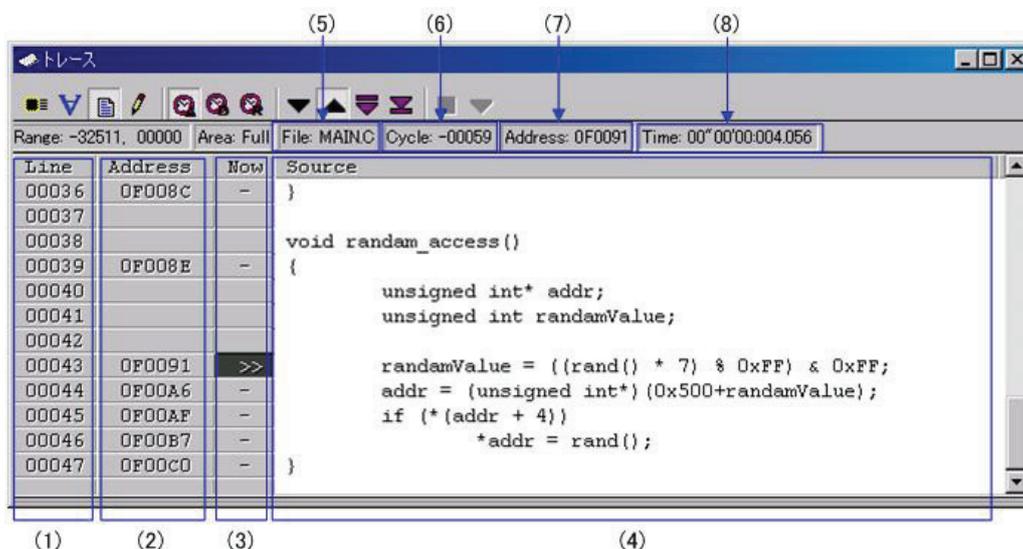
その他の表示はバスモードと同様です。

データアクセス情報に加えて、ソース行情報を混合表示できます。次のような表示になります。

Cycle	Label	DataAccess	h' m' s: ms. us
-00181	crtOmr.a30, 287:	nop	
	bss_SE_top	(000400 1234 W)	00"00'00:004.041
	crtOmr.a30, 288:	nop	
	crtOmr.a30, 290:	REIT	
-00178		(0009c8 0B60 R)	00"00'00:004.041
-00177		(0009CA 0FC0 R)	00"00'00:004.041
	test.c, 27:	ercd = set_flg(ID_flg1, FLGPTM);	
-00171		(0009CC FFC1 W)	00"00'00:004.042
-00153		(0009CA 0FC0 W)	00"00'00:004.044
-00151		(0009c8 0B6D W)	00"00'00:004.045
-00147		(0FFD82 0F R)	00"00'00:004.045
-00141		(0009c6 0014 W)	00"00'00:004.046

7.7.4 ソースモードの構成

ソースモードのみ選択されている場合、ソースモードの表示になります。ソースモードは、以下の構成になっています。



- (1) **行番号表示領域：**
表示されているファイルの行番号情報を表示します。ダブルクリックすると、表示ファイルを変更するためのダイアログボックスが表示されます。
- (2) **アドレス表示領域：**
ソース行に対応するアドレスを表示します。ダブルクリックすると、アドレスを検索するためのダイアログボックスが表示されます。
- (3) **参照サイクル表示領域：**
現在参照しているサイクルには">>"が表示されます。また、ソース行に対応するアドレスが存在する場合は"- "が表示されます。
- (4) **ソース表示領域：**
ソースファイルを表示します。
- (5) **ファイル名：**
現在表示中のソースファイル名を表示します。
- (6) **参照サイクル：**
現在参照中のサイクルを表示します。
- (7) **参照アドレス：**
現在参照中のサイクルに対応するアドレスを表示します。
- (8) **参照時間：**
現在参照中のサイクルに対応する時間を表示します。

その他の表示はバスモードと同様です。

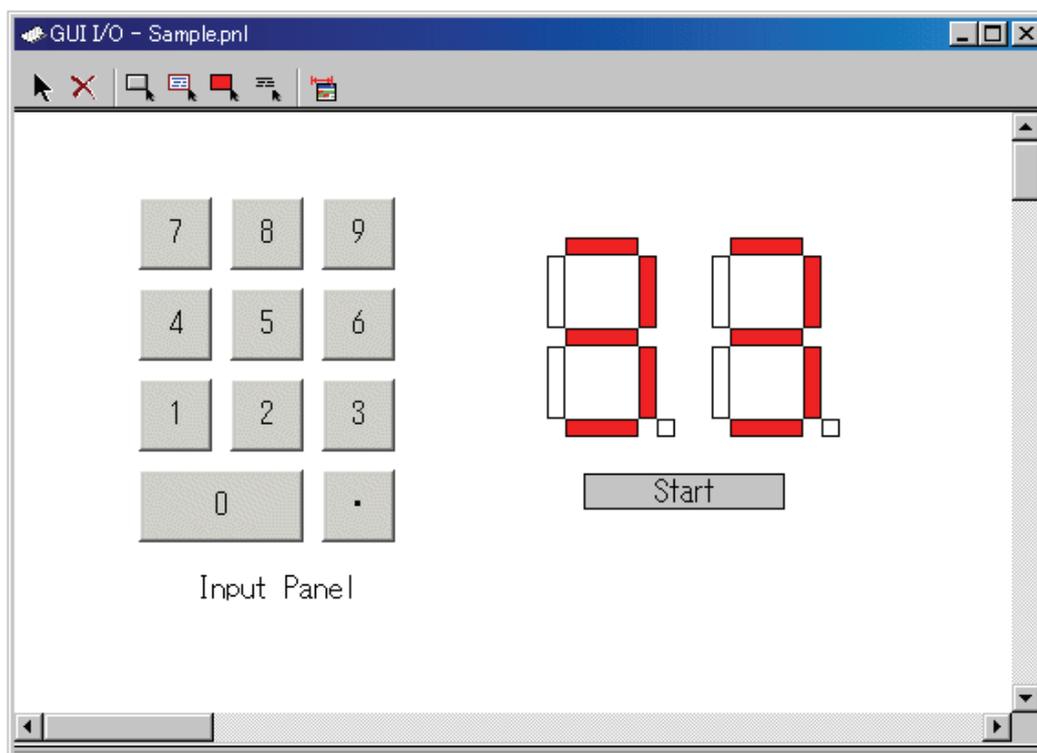
7.7.5 オプションメニュー

ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

メニュー名	機能	
BUS	バス (BUS) 情報を表示します。	
DIS	逆アセンブリ (DIS) 情報を表示します。	
SRC	ソース (SRC) 情報を表示します。	
DATA	データアクセス (DATA) 情報を表示します。	
表示	サイクル...	表示サイクルを指定します。
	アドレス...	アドレスを検索します。
	ソース...	表示するソースファイルを選択します。
時間表示	絶対時間	タイムスタンプを実行開始からの絶対時間で表示します。
	差分時間	タイムスタンプを前のサイクルとの差分時間で表示します。
	相対時間	タイムスタンプを指定したサイクルからの相対時間で表示します。
トレース操作	順方向	検索方向を順方向にします。
	逆方向	検索方向を逆方向にします。
	Step	指定方向にステップ実行します。
	Come	指定行の実行サイクルを検索します。
	計測中断	トレース計測を中断し結果を表示します。
	再計測	トレースデータを再計測します。
レイアウト...	表示カラムを選択します。	
コピー	選択されている行をクリップボードにコピーします。	
保存...	トレースデータをファイルに保存します。	
読み込み...	ファイルからトレースデータを読み込みます。	
ツールバー表示	ツールバーの表示/非表示を切り換えます。	
ツールバーのカスタマイズ...	ツールバーをカスタマイズします。	
ドッキングビュー	ウィンドウをドッキングします。	
非表示	ウィンドウを非表示にします。	

7.8 GUI 入出力ウィンドウ

GUI 入出力ウィンドウは、仮想的な入出力パネルを作成できるウィンドウです。ウィンドウ上に仮想のボタンを配置して入力したり、仮想 LED を配置してそこに出力したりできます。



- ウィンドウ上には、次のアイテムが配置できます。
 - ラベル
指定したアドレス(もしくはビット)に指定した値が書き込まれた際に、文字列を表示/消去します。
 - LED
指定したアドレス(もしくはビット)に指定した値が書き込まれた際に、指定した色で表示します(LED 点灯の代用)。
 - ボタン
押下することにより、仮想ポートへの入力が行えます。
 - テキスト
テキスト文字列を表示します。
- 作成した入出力パネルをファイル(入出力パネルファイル)に保存し、再読み込みすることもできます。
- 作成したアイテムに設定できるアドレスは、最大 200 点です。各アイテムに設定したアドレスがすべて異なる場合、配置できるアイテム数は 200 個になります。

7.8.1 オプションメニュー

ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

メニュー名	機能
アイテムの選択	クリックしたアイテムを選択状態にします。
削除	クリックしたアイテムを削除します。
コピー	クリックしたアイテムをコピーします。
貼り付け	コピーしたアイテムを貼り付けます。
ボタンの作成	新規にボタンを作成します。
ラベルの作成	新規にラベルを作成します。
LED の作成	新規に LED を作成します。
テキストの作成	新規にテキストを作成します。
グリッドの表示	グリッドを表示します。
保存...	入出力パネルファイルを保存します。
読み込み...	入出力パネルファイルを読み込みます。
サンプリング周期...	表示更新間隔を設定します。
ツールバー表示	ツールバーの表示/非表示を切り換えます。
ツールバーのカスタマイズ...	ツールバーをカスタマイズします。
ドッキングビュー	ウィンドウをドッキングします。
非表示	ウィンドウを非表示にします。

8. 機種依存コマンド一覧

下表に H8/300H Tiny コンパクトエミュレータで使用できるコマンドの一覧を示します。

- コマンドの引数に使用できる式の仕様については、「9.1 式の記述」を参照してください。
- 共通コマンドの仕様、およびコマンドラインの使用方法については、「High-performance Embedded Workshop ヘルプライブラリ」の「コマンドラインの使用」を参照してください。

項番	コマンド	短縮形	説明
1	ASSEMBLE	AS	指定したアドレスから 1 行単位でアセンブル
2	BREAKPOINT	BP	ソフトウェアブレークポイントの設定
3	BREAKPOINT_CLEAR	BC	ソフトウェアブレークポイントの削除
4	BREAKPOINT_DISPLAY	BI	ソフトウェアブレークポイントの表示
5	BREAKPOINT_ENABLE	BE	ソフトウェアブレークポイントの有効/無効化
6	CLOCK_SOURCE	CKS	MCU の供給クロック設定/参照
7	DISASSEMBLE	DA	指定した範囲の逆アセンブル結果を表示
8	EEPROM_COPY	EEPC	EEPROM メモリ領域のコピー
9	EEPROM_DISPLAY	EEPD	EEPROM メモリ内容の表示
10	EEPROM_EDIT	EEPE	EEPROM メモリ内容の変更
11	EEPROM_FILL	EEPF	EEPROM メモリ領域の内容の一括変更
12	EEPROM_SETMODE	EEPS	EEPROM メモリへの書込みモードの指定
13	EVALUATE	EV	指定したアセンブラ式の値を表示
14	HARDWAREBREAK_COMBINATION	HBC	ハードウェアブレークポイントの組合せ条件の指定
15	HARDWAREBREAK_DISPLAY	HBD	ハードウェアブレークポイントの表示
16	HARDWAREBREAK_ENABLE	HBE	ブレークモードの参照/設定
17	HARDWAREBREAK_SET	HBS	ハードウェアブレークポイントの指定
18	REGISTER_DISPLAY	RD	指定レジスタの値を参照
19	REGISTER_SET	RS	指定レジスタの値を参照
20	RESET	RE	ターゲットプログラムのリセット
21	TRACE_DISPLAY	TD	リアルタイムトレース結果のバス信号/逆アセンブル表示
22	TRACEPOINT_AREA	TPA	トレース範囲の指定
23	TRACEPOINT_COMBINATION	TPC	トレースポイントの組合せ条件の指定
24	TRACEPOINT_DISPLAY	TPD	トレースポイントの表示
25	TRACEPOINT_SET	TPS	トレースポイントの指定

9. 式の記述

9.1 式の記述

コマンドラインの引数として使用できる式の構成要素としては、以下のものが使用できます。

- 定数
- レジスタ変数
- シンボル、ラベル
- 文字定数
- 演算子

9.1.1 定数

2進数、8進数、10進数、16進数が入力可能です。数値の基数は、数値の先頭に基数を示す記号を付けて区別します。

	16進数	10進数	8進数	2進数
先頭	0x,0X,H',h'	D',d'	O',o'	B',b'
例	0xAB24 H'AB24 h'AB24	D'1234 d'1234	O'1234 o'1234	B'1001010010 b'1001010010

既定値と同じ基数で入力する場合は、基数を示す記号は省略可能です。
基数の既定値は、RADIX コマンドで設定します。

9.1.2 シンボル、ラベル

ターゲットプログラムで定義しているシンボル/ラベル、および Assemble コマンドで定義したシンボル/ラベルが使用できます。

シンボル/ラベル名には、英数字、アンダスコア(_)、ドルマーク('\$')が使用可能です。ただし、先頭文字に数字は使用できません。

大文字/小文字は区別します。

9.1.3 レジスタ変数

レジスタの値を式中で利用する場合に使用します。レジスタ変数は、レジスタ名の前に#を付加します。以下に使用できるレジスタ名を示します。

製品名	レジスタ名
H8/300H Tiny 用デバッグ	PC,CCR,SP,FP ER0,ER1,ER2,ER3,ER4,ER5,ER6,ER7 E0,E1,E2,E3,E4,E5,E6,E7 R0,R1,R2,R3,R4,R5,R6,R7 R0H,R1H,R2H,R3H,R4H,R5H,R6H,R7H R0L,R1L,R2L,R3L,R4L,R5L,R6L,R7L

レジスタ名の太文字／小文字は区別しません。どちらで指定しても結果は同じです。

9.1.4 文字定数

指定された文字または文字列を ASCII コードに変換し、定数として扱います。文字および文字列は、シングルクォーテーションで囲みます。文字列は 4 文字以内（32 ビット長）でなければなりません。4 文字を超えた場合エラーとなります。エスケープ文字には対応していません（例: > '¥n' と入力した場合、0x0A ではなく 0x5C6E となります）。

9.1.5 演算子

式に記述可能な演算子を以下に示します。演算子の優先度は、レベル 1 が最も高く、レベル 13 が最も低くなります。優先順位が同じ場合は、式の左から順番に計算します。

演算子	意味	優先度
()	括弧	レベル 1
+, -, ~, !	単項正、単項負、ビット毎の NOT、論理 NOT	レベル 2
*, /	二項乗算、二項除算	レベル 3
%	剰余	レベル 4
+, -	二項加算、二項減算	レベル 5
>>, <<	右シフト、左シフト	レベル 6
<, <=, >, >=	二項比較	レベル 7
==, !=	二項比較	レベル 8
&	ビット毎の AND	レベル 9
^	ビット毎の排他的 OR	レベル 10
	ビット単位の OR	レベル 11
&&	論理 AND	レベル 12
	論理 OR	レベル 13

9.2 C/C++言語式の記述方法

C ウォッチポイントの登録、及び C ウォッチポイントに代入する値の指定には、以下の字句(トークン)で構成された C/C++言語式が使用できます。

字句(トークン)	例
即値	10, 0x0a, 012, 1.12, 1.0E+3
スコープ解決	::name, classname::member
四則演算子	+, -, *, /
ポインタ	*, **, ...
参照	&
符号反転	-
"."演算子によるメンバ参照	Object.Member
"->"演算子によるメンバ参照	Pointer->Member, this->Member
メンバへのポインタ参照	Object.*var, Pointer->*var
括弧	(,)
配列	Array[2], DArray[2][3], ...
基本型へのキャスト	(int), (char*), (unsigned long *), ...
typedef された型へのキャスト	(DWORD), (ENUM), ...
変数名および関数名	var, i, j, func, ...
文字定数	'A', 'b', ...
文字列リテラル	"abcdef", "I am a boy.", ...

即値

即値としては、16進数、10進数、および8進数が使用できます。0x で始まれば16進数、0 で始まれば8進数として認識します。それ以外の数値は、10進数として認識します。また、変数に値を代入する場合、浮動小数点数値も使用できます。

注意

- 即値を C ウォッチポイントとして登録することはできません。
- 即値は、C ウォッチポイントを指定する C 言語式の中に用いる場合、および代入する値を指定する場合にのみ有効です。浮動小数点数値を使用する場合、1.0+2.0 等の演算はできません。

スコープ解決

スコープ解決演算子(::)が使用できます。以下に使用例を示します。

大域スコープ : ::変数名
::x, ::val

クラス指定 : クラス名::メンバ名、クラス名::クラス名::メンバ名 等
T::member, A::B::member

四則演算子

四則演算子は、加算(+), 減算(-), 乗算(*), 除算(/)が使用できます。以下に、計算の優先順位を示します。

(*), (/), (+), (-)

注意

- 浮動小数点に対する四則計算は、現在サポートしていません。

ポインタ

ポインタは、*で表され、ポインタのポインタ**、ポインタのポインタのポインタ ***、・・・が使用できます。

「*変数名」、「**変数名」、・・・という記述で使います。

注意

- 即値をポインタとして扱うことはできません。つまり、*0xE000などは、使用することができません。

参照

参照は、&で表され、「&変数名」のみが使用できます。「&&変数名」等は使用することができません。

符号反転

符号反転は、-で表され、「-即値」、「-変数名」のみが使用できます。-を2つ以上偶数個続けた場合には、符号反転は行われません。

注意

- 浮動小数点変数に対する符号反転は、現在サポートしておりません。

"."演算子によるメンバ参照

"."演算子によるクラス、構造体、共用体のメンバ参照は、「変数名.メンバ名」のみが使用できます。

(例)

```
class T {
public:
int member1;
char member2;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

この場合、t_cls.member1、(*pt_cls).member2は、正しくメンバを参照することができます。

メンバへのポインタ

"*"演算子や"->"演算子によるメンバへのポインタ参照は、「変数名.*メンバ名」、「変数名->*メンバ名」のみが使用できます。

(例)

```
class T {
public:
int member;
};
class T t_cls;
class T *pt_cls = &t_cls;

int T::*mp = &T::member;
```

この場合、t_cls.*mp、pt_cls->*mpは、正しくメンバを参照することができます。

注意

- print *mp という記述では、メンバへのポインタ変数を正しく参照できません。

括弧

式の途中で、計算の優先順位を指定する括弧として、'('と')'を使用することができます。

配列

配列の要素を指定する表現に '['と']'を使用することができます。配列は、「変数名[(要素番号または変数)]」、「変数名[(要素番号または変数)][(要素番号または変数)]」、・・・という記述で使います。

基本型へのキャスト

C の基本型のうち、char 型、short 型、int 型、long 型へのキャスト、およびこれらの基本型へのポインタ型へのキャスト演算が使用できます。ポインタ型へのキャストは、ポインタのポインタ、ポインタのポインタのポインタ、・・・なども使用できます。なお、signed、unsigned の指定がない場合のデフォルトは、以下のとおりです。

基本型	デフォルト
char	unsigned
short	signed
int	signed
long	signed

注意

- C++の基本型のうち、bool 型、wchar_t 型、浮動小数点型(float、double 型)へのキャストは使用できません。
- レジスタ変数に対するキャストは使用できません。

typedef された型へのキャスト

typedef された型(C/C++の基本型以外の型)、およびそれらへのポインタ型へのキャスト演算が使用できます。ポインタ型へのキャストは、ポインタのポインタ、ポインタのポインタのポインタ、・・・なども使用できます。

注意

- class 型、struct 型、union 型、およびそれらのポインタ型へのキャストは使用できません。

変数名

変数名は、C/C++の規約通りアルファベットで始まる文字列が使用できます。最大文字数は、255 文字です。また、this ポインタ変数を使用することができます。

関数名

関数名は、C の規約通りアルファベットで始まる文字列が使用できます。C++の場合、関数名は使用できません。

文字定数

文字定数として、シングルクォーテーション(')で囲まれた文字が使用できます。例えば、'A'、'b'等です。これらは、ASCII コードに変換され、1 バイトの即値として使用されます。

注意

- 文字定数を C ウォッチポイントとして登録することはできません。
- C ウォッチポイントを指定する C/C++言語式の中に用いる場合、および代入する値を指定する場合にのみ有効です（文字定数は即値と同じ扱いになります）。

文字列リテラル

文字列リテラルとして、ダブルクォーテーション(")で囲まれた文字列が使用できます。例えば、"abcde"、"I am a boy."等です。

注意

- 文字列リテラルは、右辺式(代入演算子の右辺)にのみ記述することができ、左辺式(代入演算子の左辺)が char 配列、または char ポインタ型の場合にのみ使用することができます。それ以外の場合には、文法エラーとなります。

9.3 C/C++言語式の表示形式

C ウォッチウィンドウのデータ表示領域における C/C++言語式の表示は、その型名、C/C++言語式(変数名)、計算結果(値)から構成されています。以下に、型別に表示形式を説明します。

列挙型の場合

- 計算結果の値が定義されているものであれば、その名前で表示します。
(DATE) date = Sunday (全Radix)
- 計算結果の値が定義されているものでなかった場合には、以下のように表示します。
(DATE) date = 16 (Radixが初期状態の場合)
(DATE) date = 0x10 (Radixが16進数の場合)
(DATE) date = 0000000000010000B (Radixが2進数の場合)

基本型の場合

- 計算結果が char 型および浮動小数点以外の基本型の場合には、以下のように表示します。
(unsigned int) i = 65280 (Radixが初期状態の場合)
(unsigned int) i = 0xFF00 (Radixが16進数の場合)
(unsigned int) i = 1111111100000000B (Radixが2進数の場合)
- 計算結果が char 型の場合には、以下のように表示します。
(unsigned char) c = 'J' (Radixが初期状態の場合)
(unsigned char) c = 0x4A (Radixが16進数の場合)
(unsigned char) c = 10100100B (Radixが2進数の場合)
- 計算結果が浮動小数点の場合には、以下のように表示します。
(double) d = 8.207880399131839E-304 (Radixが初期状態の場合)
(double) d = 0x10203045060708 (Radixが16進数の場合)
(double) d = 000000010.....1000B (Radixが2進数の場合)
(...は省略を表す)

ポインタ型の場合

- 計算結果が char*型以外のポインタ型の場合には、以下のように内容を 16 進数表示します。
(unsigned int *) p = 0x1234 (全Radix)
- 計算結果が char*型の場合には、C ウォッチウィンドウのメニュー [char*の文字列表示] で文字列/文字の表示が指定できます。表示例を以下に示します。

文字列表示の場合

(unsigned char *) str = 0x1234 "Japan" (全Radix)

文字表示の場合

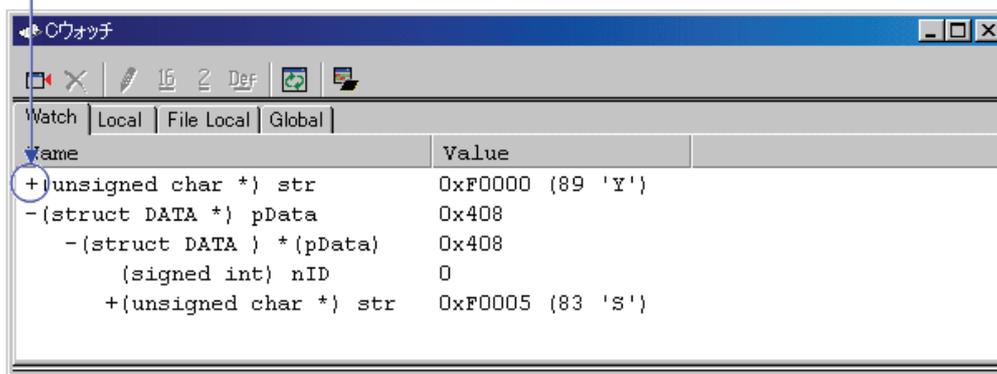
(unsigned char *) str = 0x1234 (74 'J') (全Radix)

文字列表示の場合、文字列の終わりを表すコード(0)までに、文字表示できないコードが格納されていた場合には、以下のように、閉じ(")を出力しません。

(unsigned char *) str = 0x1234 "Jap

また、文字列の長さが 80 文字を越えた場合も同様に、閉じ(")を出力しません。
なお、C/C++言語式がポインタ型の場合には、以下に示すように、型名の左側に '+' マークが現れます。

ポイント型を示す '+' マーク



この '+' マークが表示されている行をダブルクリックすると、そのポインタのオブジェクトが現れます。オブジェクトを表示すると、 '+' マークは '-' マークにかわります。なお、 '-' マークが表示されている行をダブルクリックすると、もとの状態に戻ります。このようにして、リスト構造やツリー構造等のデータも参照することができます。

配列型の場合

- 計算結果が char[] 型以外の配列型の場合には、以下のように先頭アドレスを 16 進数表示します。

```
(signed int [10]) z = 0x1234 (全Radix)
```
- 計算結果が char[] 型の場合には、以下のように表示します。

```
(unsigned char [10]) str = 0x1234 "Japan" (全Radix)
```

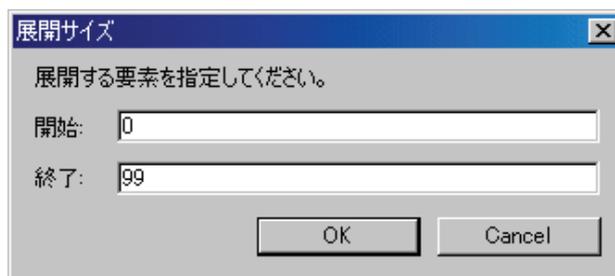
文字列の終わりを表すコード(0)までに、文字表示できないコードが格納されていた場合には、以下のように、閉じ(")を出力しません。

```
(unsigned char [10]) str = 0x1234 "Jap (全Radix)
```

また、文字列の長さが 80 文字を越えた場合も同様に、閉じ(")を出力しません。

なお、C/C++ 言語式が配列型の場合には、ポインタ型と同様、型名の左側に '+' マークが現れます。展開方法は、ポインタ型と同じです。詳細な説明については、「ポインタ型の場合」をご参照下さい。

配列のサイズが 100 以上の場合、下記ダイアログがオープンするので、展開する要素数を指定してください。



Start で指定した要素から End で指定した要素までを表示します。

配列の要素数の最大値を超える値を指定した場合は、配列の最大値を指定した事になります。

なお、Cancel ボタンを押下した場合、配列は展開しません。

関数型の場合

- 計算結果が関数型の場合には、以下のように関数の開始アドレスを 16 進数表示します。

```
(void()) main = 0xF000 (全Radix)
```

参照型の場合

- 計算結果が参照型の場合には、以下のように参照するアドレスを 16 進数表示します。
(signed int &) ref = 0xD038 (全Radix)

ビットフィールド型の場合

- 計算結果がビットフィールド型の場合には、以下のように表示します。
(unsigned int :13) s.f = 8191 (Radixが初期状態の場合)
(unsigned int :13) s.f = 0x1FFF (Radixが16進数の場合)
(unsigned int :13) s.f = 1111111111111B (Radixが2進数の場合)

C シンボルが見つからなかった場合

- 計算した式の中に発見できなかった C シンボルがあった場合には、以下のように表示します。
() x = <not active> (全Radix)

文法エラーの場合

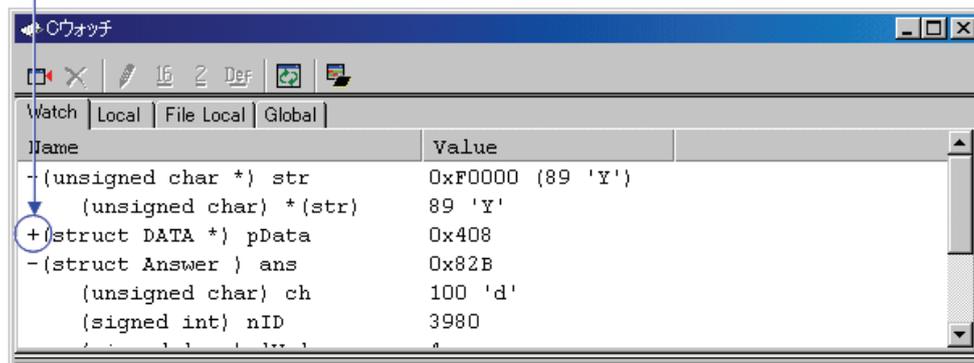
- 計算した式が文法的に間違っていた場合には、以下のように表示します。
() str*(p = <syntax error> (全Radix)
(str*(pは間違った記述)

構造体・共用体型の場合

- 計算結果が構造体・共用体型の場合には、以下のようにアドレスを 16 進数表示します。
(Data) v = 0x1234 (全Radix)

なお、C/C++言語式が構造体・共用体型のようにメンバを持つ場合は、以下に示すように、型名(タグ名)の左側に '+' マークが現れます。

構造体・共用対を示す '+' マーク



この '+' マークが表示されている行をダブルクリックすると、その構造体(または共用体)のメンバが現れます。メンバを表示すると、 '+' マークは '-' マークにかわります。なお、 '-' マークが表示されている行をダブルクリックすると、もとの状態に戻ります。このようにして、メンバを参照することができます。

注意

- typedef で宣言された型定義名と同一名の変数を宣言した場合、その変数を参照することはできません。

レジスタ変数の場合

- 計算結果がレジスタ変数の場合には、以下のように型名の先頭に "register" と表示します。
(register signed int) j = 100

10. プログラム停止要因の表示

デバッグ機能によりプログラムが停止した場合、その停止要因は アウトプットウィンドウ、および、ステータスウィンドウ ([Platform]シート) に表示されます。

停止要因の表示内容とその意味は、以下のとおりです。

表示	停止要因
Halt	[プログラムの停止]ボタン/メニューによる停止
S/W break	ソフトウェアブレイク
H/W event, Combination	ハードウェアブレイク、論理組合せ And 条件または同時 And 条件成立
H/W event, Combination, Ax	ハードウェアブレイク、論理組合せ Or 条件成立 (Ax : 成立したイベント番号)
H/W event, State transition, from xx	ハードウェアブレイク、状態遷移 State Transition 条件成立 (from xx : 直前の状態 (start, state1, state2))
H/W event, State transition, Timeout	ハードウェアブレイク、状態遷移 タイムアウト成立

注意事項

停止要因を表示可能かどうかは、接続しているターゲットに依存します。ターゲットによっては、常に "Halt" と表示されたり "---" と表示される場合があります。

11. 注意事項

11.1 製品共通の注意事項

11.1.1 Windows 上でのファイル操作

Windows 上でのファイル操作については、以下の点に注意してください。

ファイル名、及びディレクトリ名

- 空白文字を含むファイル名、ディレクトリ名は使用できません。
- 漢字のファイル名、ディレクトリ名は使用できません。
- .(ピリオド)が2つ以上ついたファイルは使用できません。

ファイル指定、及びディレクトリ指定

- "..."(2つ上のディレクトリ指定)は使用できません。
- ネットワークパス名は使用できません。 ネットワークパス名を使用する場合は、ドライブに割り当てて使用してください。

11.1.2 ソフトウェアブレイクポイントの設定可能領域

ソフトウェアブレイクポイントに設定可能な領域は、MCU によって異なります。詳細は、「ソフトウェアブレイクポイント設定可能領域」を参照して下さい。

11.1.3 C 変数の参照・設定

- typedef で宣言された型定義名と同一名の変数を宣言した場合、その変数を参照することはできません。
- レジスタ変数、ビットフィールド型変数への代入はできません。
- 64 ビット長の変数 (long long 型や double 型など) への代入はできません。
- メモリの実体 (アドレスとサイズ) を示さない変数への代入はできません。
- 複数のローカル変数が、コンパイラの最適化により同一領域に割り当てられている場合、その変数の値を正しく表示できない場合があります。
- リテラルな文字列を、char 配列あるいは char ポインタ型の変数以外に代入することはできません。
- 浮動小数点に対する四則演算はできません。
- 浮動小数点型変数に対する符号反転はできません。
- 浮動小数点型へのキャストはできません。
- レジスタ変数に対するキャストはできません。
- 構造体型、共用体型、及びそれらの型へのポインタ型へのキャストはできません。
- 文字定数およびリテラルな文字列には、エスケープシーケンスは記述できません。

11.1.4 C++での関数名

- ブレークポイント設定などで関数名を使用してアドレスを設定する場合、クラスのメンバ関数、operator 関数、および、オーバーロード（多重定義）関数を使用できません。
- C/C++言語式の記述に関数名を使用できません。
- アドレス値設定領域において、関数名を使用したアドレス指定はできません。

11.1.5 複数モジュールのデバッグ

一つのセッションに複数のアブソリュートモジュールファイルを登録し、同時にダウンロードすることはできません。ただし、一つのアブソリュートモジュールファイルと複数の機械語ファイルを同時にダウンロードすることは可能です。

11.1.6 同期デバッグ

同期デバッグには対応していません。

11.1.7 コンパクトエミュレータのリセットスイッチ

コンパクトエミュレータ本体のシステムリセットが正常に動作しない場合、デバッグを終了させた後コンパクトエミュレータの電源を再投入し、デバッグを再起動してください。その後、プログラムを再ダウンロードしてください。

11.2 H8/300H Tiny 用デバッガの注意事項

H8/300H Tiny 用デバッガに関する注意事項を以下に示します。

11.2.1 コンパクトエミュレータが使用するスタック領域

コンパクトエミュレータは、割り込みスタック領域をワーク領域として使用します(20 バイト)。
デバッグの際は、ユーザスタック領域+20 バイトの領域を確保して下さい。

11.2.2 H/W ブレーク指定

ターゲットプログラム実行中に以下の操作はできません。

- `HARDWAREBREAK_ENABLE` コマンドの実行
- H/W ブレークポイント設定ウィンドウのオープン

11.2.3 ハードウェアイベント

ハードウェアブレークおよびトレースポイントのデータアクセスにおいて、奇数アドレスからのワード長(2 バイト長)のデータをイベントに指定した場合、そのイベントは検出されません。

H8/300H Tiny コンパクトエミュレータデバッグ
ユーザーズマニュアル

発行年月日 2005年05月01日 Rev.1.00

発行 株式会社 ルネサス テクノロジ 営業企画統括部
〒100-0004 東京都千代田区大手町2-6-2

編集 株式会社 ルネサス ソリューションズ ツール開発部

© 2005. Renesas Technology Corp. and Renesas Solutions Corp., All rights reserved. Printed in Japan.

H8/300H Tiny コンパクトエミュレータデバッガ 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

RJJ10J0908-0100