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。

User's Manual

M32C シミュレータデバッガ V.1.04

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

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2009.05

本資料ご利用に際しての留意事項

1. 本資料は、お客様に用途に応じた適切な弊社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弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証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など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
3. 本資料に記載の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、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替および外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
4. 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などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊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、弊社ホームページ（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。
5. 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、弊社はその責任を負いません。
6. 本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独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。弊社は、適用可否に対する責任は負いません。
7. 本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会ください。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
8. 第7項にかかわらず、本資料に記載された製品は、下記の用途には使用しないでください。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
   1）生命維持装置。
   2）人体に埋め込み使用するもの。
   3）治療行為（患部切り出し、薬剤投与等）を行うもの。
   4）その他、直接人命に影響を与えるもの。
9. 本資料に記載された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、実装条件およ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ください。弊社保証値を越えて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、弊社はその責任を負いません。
10. 弊社は製品の品質および信頼性の向上に努めておりますが、特に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。弊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、誤動作防止設計などの安全設計（含むハードウェアおよびソフトウェア）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。
11. 本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
12. 本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断りいたします。
13. 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弊社営業窓口までご照会ください。

D039444

製品の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事項を記入の上、コンタクトセンタ csc@renesas.comまで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス テクノロジ

コンタクトセンタ　csc@renesas.com

ユーザ登録窓口　regist_tool@renesas.com

ホームページ　http://japan.renesas.com/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、シミュレータシステム上で動作します。

対応 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。

- R32C/100 シリーズ
  (注)この CPU に依存する情報については、本ヘルプでは「R32C 用」と記載しています。
- M32C/80, M16C/80 シリーズ
  (注)この CPU に依存する情報については、本ヘルプでは「M32C 用」と記載しています。
- M16C/60, M16C/50, M16C/30, M16C/Tiny, M16C/20, M16C/10 シリーズ, R8C ファミリ
  (注)この CPU に依存する情報については、本ヘルプでは「M16C/R8C 用」と記載しています。
- 740 ファミリ
  (注)この CPU に依存する情報については、本ヘルプでは「740 用」と記載しています。

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。

製品の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事項を記入の上、コンタクトセンタ csc@renesas.comまで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス テクノロジ
コンタクトセンタ　csc@renesas.com
ユーザ登録窓口　regist_tool@renesas.com
ホームページ　http://japan.renesas.com/tools

起動/セットアップ編 1

起動/セットアップ編

このページは白紙です

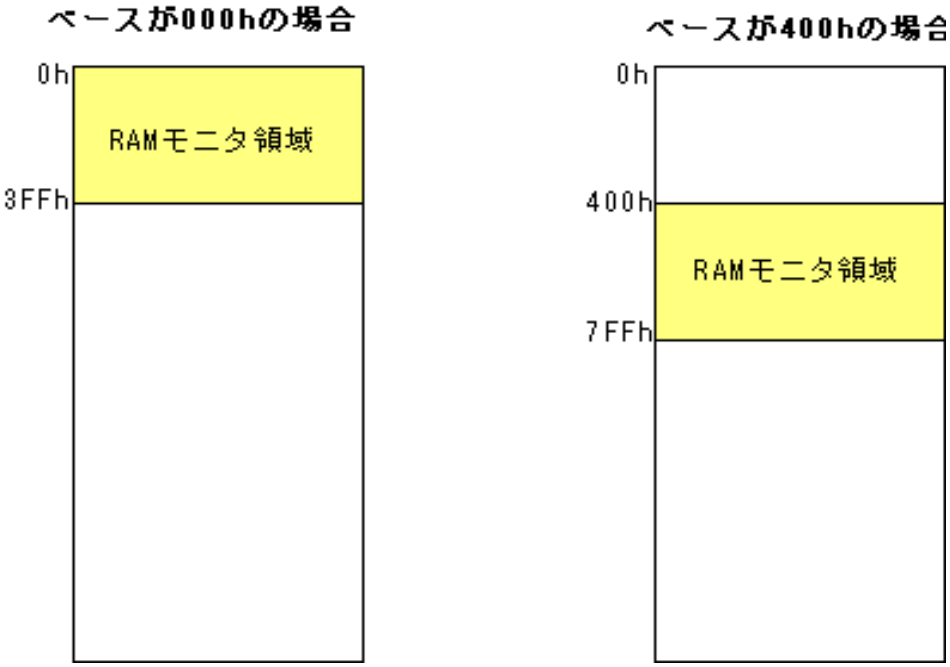
1. 機能概要

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。
シミュレータは、1K バイトの RAM モニタ領域を備えています（複数の領域に分割することはできません)。

1.1.1 RAM モニタ領域

本デバッガは、1K バイトの RAM モニタ領域を備えており、この RAM モニタ領域は、任意の連続アドレスに配置できます。



1.1.2 サンプリング周期

サンプリング周期とは、表示更新間隔を意味します。
RAM モニタ対応の各ウィンドウで指定できます(デフォルトは 100 ミリ秒)。動作状況によっては、指定したサンプリング周期より遅くなります(以下の状況に依存します)。

- 通信インタフェース
- RAM モニタウィンドウの表示ウィンドウの数
- RAM モニタウィンドウの表示ウィンドウのサイズ
- ASM ウォッチウィンドウの RAM モニタ領域内の ASM ウォッチポイント数
- C ウォッチウィンドウの RAM モニタ領域内の C ウォッチポイント数

1.1.3 関連ウィンドウ

リアルタイム RAM モニタの機能を使用できるウィンドウを以下に示します。

- RAM モニタウィンドウ
- ASM ウォッチウィンドウ
- C ウォッチウィンドウ

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機能

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ターゲットプログラムをブレークします。このブレークするポイントをソフトウェアブレークポイントと呼びます。 ソフトウェアブレークポイントは、エディタ(ソース)ウィンドウや S/W ブレークポイント設定ウィンドウで 設定/解除します。一時的に無効/有効にすることも可能です。
64 点のソフトウェアブレークポイントを指定することができます。 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。

1.2.1.1 ソフトウェアブレークポイントの設定/解除

ソフトウェアブレークポイントは、以下のウィンドウで設定/解除します。

- エディタ(ソース)ウィンドウ
- S/W ブレークポイント設定ウィンドウ

エディタ(ソース)ウィンドウでのソフトウェアブレークポイント設定/解除は、ダブルクリックで 操作できます。
S/W ブレークポイント設定ウィンドウでは、ソフトウェアブレークポイント設定/解除のほかに、 一時的無効/有効を切り換えることもできます。

1.2.1.2 ソフトウェアブレークポイントの設定可能領域

ソフトウェアブレークポイントに設定できる領域は、製品によって異なります。
設定できる領域については、「12.1.2 ソフトウェアブレークポイントが設定可能な領域」を参照ください。

1.2.2 ハードウェアブレーク機能

ハードウェアブレークは、メモリへのデータ書き込み/読み込み検出、命令実行検出でターゲットプログラムを停止する機能です。
このブレークするポイントをハードウェアブレークポイントと呼びます。 ハードウェアブレークポイントは、H/W ブレークポイント設定ダイアログで設定/解除します。

- 64 点のハードウェアブレークポイントが設定できます。
- パスカウントを指定することもできます。
- ハードウェアブレークポイントのアクセス条件には、命令フェッチ(Fetch)、メモリアクセス(Write,Read,R/W)が指定できます。
- ハードウェアブレークポイントに読み込み/書き込みされるデータが特定の値であればブレークするといった指定も可能です。
- 複数のハードウェアブレークポイントを設定した場合、いずれかのハードウェアブレークポイント到達でブレークします。

1.3 リアルタイムトレース機能

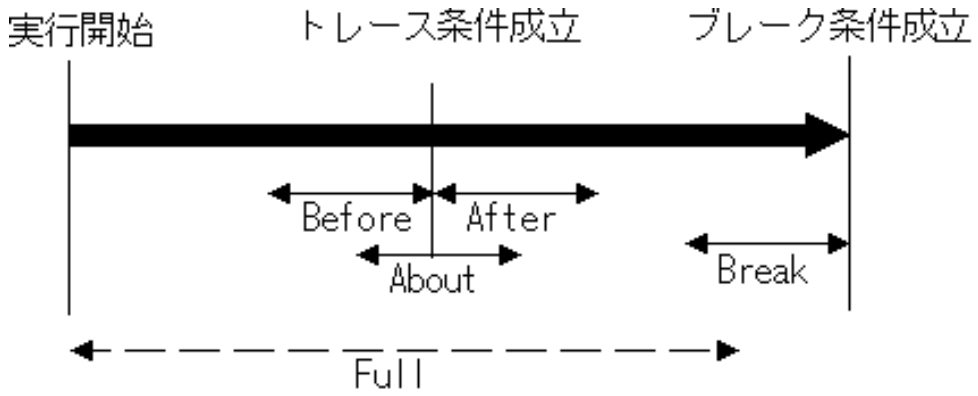
リアルタイムトレースは、ターゲットプログラムの実行履歴を参照する機能です。
740 用デバッガでは、リアルタイムトレース機能はサポートしていません。
実行履歴は、トレースウィンドウで参照します。
実行履歴の参照方法として、以下の表示モードをサポートしています。

- 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、シ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。

1.3.1 トレース範囲

シミュレータデバッガでは、初期設定で指定したサイクル分の実行履歴を参照できます。トレース範囲は、以下の 5 モードをサポートしています。

- Break
  ターゲットプログラム停止までの指定したサイクル
- Before
  トレースポイントを通過するまでの指定したサイクル
- About
  トレースポイントを通過した時点の前後指定したサイクル
- After
  トレースポイントを通過した後の指定したサイクル
- Full
  ターゲットプログラムを実行開始した後の指定したサイクル



デフォルトは、ターゲットプログラム停止までのサイクルを記録する"Break"です。
Break/Full 以外のトレース範囲を指定する場合やターゲットプログラムを継続実行したい場合は、トレースイベントを設定して下さい。

1.3.2 トレース条件設定

トレースイベントとして以下の指定ができます。

- アドレス指定
  - ・命令フェッチ
  - ・メモリアクセス
  - ・ビットアクセス
- 外部トリガ指定
- 割り込み

指定可能なイベント数は、6 点です。 各トレースイベントは、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)
- すべてのイベントが同時に成立(And(same time)条件)
- いずれかのイベントが成立(Or 条件)
- 状態遷移指定によるブレークステート突入(State Transition 条件)

H/W ブレークポイントと同様に、リアルタイム OS 対応として指定タスクのみ(または指定タスク以外)をトレース条件に指定することもできます。

1.3.3 トレースデータの書き込み条件

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。
このデバッガでは、以下の条件が指定できます。

- 書き込み条件の制限なし(デフォルト)
- 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
- 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以外
- 開始イベント成立のサイクルのみ
- 開始イベント成立のサイクル以外
- 開始イベント成立から開始イベント不成立となるまでのサイクル
- 開始イベント成立から開始イベント不成立となるまでのサイクル以外

1.4 カバレッジ計測機能

カバレッジ計測は、ターゲットプログラムが実行(アクセス)したアドレスを記録する機能です(C0 カバレッジ)。
ターゲットプログラムの実行停止後、未実行のアドレスを把握することができます。
このカバレッジ計測機能をテスト工程で用いることにより、テスト項目の抜けを把握することができます。

1.4.1 カバレッジ計測領域

シミュレータデバッガでは、全メモリ空間がカバレッジ計測領域になります。

1.4.2 関連ウィンドウ

カバレッジ計測結果は、以下のウィンドウで参照することができます。

- エディタ(ソース)ウィンドウ
- メモリウィンドウ
- カバレッジウィンドウ

1.5 リアルタイム OS デバッグ機能

リアルタイム OS を使用したターゲットプログラムのリアルタイム OS 依存部分をデバッグする機能です。
リアルタイム OS の状態表示やタスク実行履歴等を参照することができます。

740 用デバッガでは、リアルタイム OS デバッグ機能はサポートしていません。

1.6 GUI 入出力機能

ユーザ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。

1.7 I/O シミュレーション機能

- 仮想ポート入力

入力ポートのデータの変化を定義する機能です。入力ポートのデータの変化は、I/O タイミング設定ウィンドウで定義します。定義した内容を以下のタイミングでシミュレートすることができます。

・プログラムの実行が指定サイクルになった
・指定されたメモリをプログラムがリードアクセスした
・指定された仮想割り込みが発生した

この機能と I/O スクリプト機能と組み合わせることにより、命令フェッチ、メモリのライトアクセスなどのタイミングでシミュレートすることも可能です。

- 仮想ポート出力

出力ポートのデータの変化とその時のサイクルを記録する機能です。記録されたデータは、I/O タイミング設定ウィンドウでグラフや数値形式で確認できます。

本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。再実行した場合、前回のデータはクリアされます。

- 仮想割り込み

割り込み(タイマ割り込み,キー入力割り込みなど)を発生させる機能です。以下のタイミングで割り込みを発生させることができます。

・プログラムの実行が指定サイクルになった
・プログラムが指定アドレスを命令フェッチした
・指定時間間隔

仮想割り込みは、I/O タイミング設定ウィンドウで定義します。この機能と I/O スクリプト機能と組み合わせることにより、メモリのライトアクセス,リードアクセスなどのタイミングで割り込みを発生させることができます。

- 出力ポートシミュレート

出力ポートのデータの変化を記録する機能です。記録されたデータは、出力ポートウィンドウに表示させることができます(ファイルへの出力も可能です)。

本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。

- I/O スクリプト

仮想ポート入力や仮想割り込みの設定をスクリプトとして実行する機能です。 I/O タイミング設定ウィンドウで設定できる仮想ポート入力や仮想割り込みの定義よりもさらに柔軟な定義が可能です。
例) タイマレジスタに設定した分周比を読み込み、タイマ割り込みを周期的に発生する。

1.8 実行時間計測機能

実行したプログラムのおおよそのサイクル数と実行時間を計測する機能です。
実行時間は、実行した命令の累積サイクル数と、Init ダイアログの MCU タブで指定する MCU クロックから算出します。

1.9 スタック使用量計測機能

ターゲットプログラムが使用したスタックの最大アドレスと最小アドレスを検出する機能です。

2. シミュレーション仕様について

対応機種によって、シミュレーション仕様が異なります。

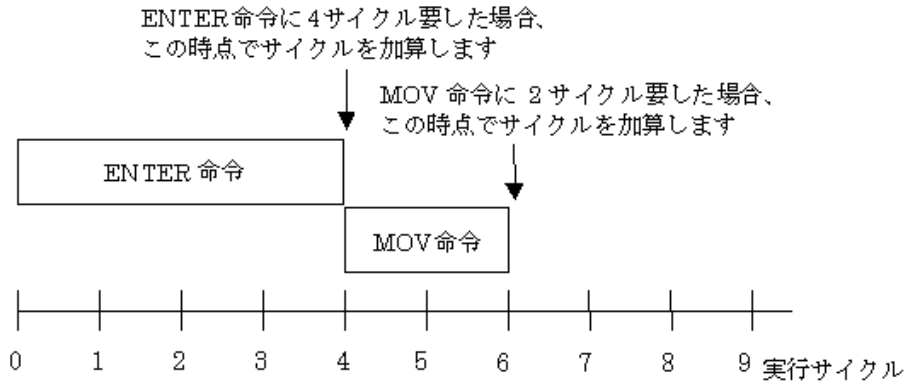
2.1 R32C 用シミュレーション仕様

2.1.1 命令動作

- 命令のサイクル数について
  時間管理は、サイクル単位で行われます。サイクル数は、マイクロコンピュータのソフトウエアマニュアルに記載されている値を使用しています。
  但し、実チップとは、次の点が異なります。
  - ・バス幅、キュー、ウェイト数を考慮したサイクル数の計測は行いません。
  - ・割り込みシーケンス実行サイクル数の計測は行いません（割り込みが発生した場合、割り込みシーケンス実行サイクル数は、0 サイクルになります）。
  - ・サイクル数の計測は、リセット直後から開始し（リセット直後のサイクル数は 0 になります）機械語 1 命令の実行に要したサイクル数を命令実行後に加算していきます（下図をご参照ください）。

注意事項

シミュレータで計測するサイクル数は、バス幅、キュー、ウェイト数等を考慮していないため、実チップのサイクル数と比べると誤差があります。



上記の例では、ENTER 命令や MOV 命令等が実行されている間は、サイクル数は加算されません。命令の実行後にサイクル数が加算されます。

- ターゲットプログラムの実行時間計測
  ターゲットプログラムの実行時間計測は、上記のサイクル数と、Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出します。

注意事項

シミュレータの実行時間計測は、上記サイクル数を使用して算出するため、実チップの実行時間と比べると誤差があります。

- STOP、WAIT、BRK2 命令
  NOP 命令として実行します。
  そのほかの命令は、実チップと同一の動作を行います。

- INT、INTO、UND、BRK 命令
  実チップと同様、割り込みが発生します（INTO 命令は O フラグが 1 のときのみ）。

2.1.2 リセット動作

- SFR 領域の初期化は行われません。
- 実行サイクル数は 0 になります。

上記以外の動作は、実チップと同一です。
なお、シミュレータ起動時にもリセットを行います。起動直後はリセットベクタ（FFFFFFFCh ～ FFFFFFFFh）に FFFF0000h という値が設定されているので、シミュレータ起動直後のプログラムカウンタ値は FFFF0000h となります。

2.1.3 メモリ

- メモリ空間
  マップとしてメモリ空間を設定していれば、0000000000h ～ FFFFFFFFh の 4GB のメモリ空間全てが RAM として読み書き可能です。
  ただし、00010000h ～ 0003FFFFh, 00050000h ～ FFFEFFFFh のメモリ空間は、初期状態ではマップが確保されていない状態になっているため、そのままでこの部分をアクセスしようとするとエラーとなります。これがプログラム実行中であれば、メモリの不正アクセスとして、プログラム実行が中断します。この部分にメモリを割り当てるためにはマップ機能（後述）を使用します。
- 起動直後のメモリ構成とその初期値
  起動直後は、次のようにメモリが設定されています。

00000000h ～ 000003FFh（SFR1 領域）	00h で埋められています。
00000400h ～ 0000FFFFh	FFh で埋められています。
00010000h ～ 0003FFFFh	起動直後はメモリが存在しません。
00040000h ～ 0004FFFFh（SFR2 領域）	00h で埋められています。
00050000h ～ FFFEFFFFh	起動直後はメモリが存在しません。
FFFF0000h ～ FFFFFFFBh	FFh で埋められています。
FFFFFFFCh ～ FFFFFFFFh（リセットベクタ）	FFFF0000h が設定されています。

- マップ機能･･･ MAP コマンド
  シミュレータでは、00000000h ～ FFFFFFFFh のメモリ空間を 65536 等分し、64KB ごとにマップとしてメモリの割り付けを行います。このうち、アドレスが 00000000h ～ 0000FFFFh、00040000h ～ 0004FFFFh、および FFFF0000h ～ FFFFFFFFh はシミュレータ起動時に既に割り付けられています。
  マップの割り付けは、MAP コマンドを使用してください。MAP コマンドで割り付けたメモリは、割り付け直後はすべて FFh で初期化されます。

なお、プログラムをダウンロードするとプログラム領域やデータ領域を自動的にマップ設定します。

注意事項

一度マップ設定して割り付けたメモリ空間は、削除することはできません。

- メモリのない領域にアクセスした場合
  00010000h ～ 0003FFFFh、00050000h～ FFFEFFFFh のメモリ空間は、メモリを確保しない限りメモリがありません。この領域をアクセスすると、メモリの不正アクセスとしてエラー中断します。これがプログラム実行中であれば、メモリの不正アクセスとして、プログラム実行が中断します。

2.1.4 I/O

- SFR
  実チップの CPU コア以外の、タイマ、DMAC、シリアル I/O などの周辺 I/O はサポートしていません。周辺 I/O が接続される SFR 領域も、シミュレータでは RAM として扱います。 ただし、SFR 等のメモリへのデータ入力や、タイマ割り込み等の割り込みを擬似的に実現する方法を用意しています。この方法に関しては、後述の「I/O スクリプト」、「割り込み」をご参照ください。

- I/O スクリプト
  ・仮想ポート入力機能
  仮想ポート入力機能とは、外部から指定アドレスのメモリに入力されるデータの変化を定義する機能です。この機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。 データをメモリに入力できるタイミングを以下に示します。
  1. プログラムの実行が指定サイクルになった時
  2. 指定されたメモリをプログラムがリードアクセスした時
  3. 指定された仮想割り込みが発生した時

  上記のタイミングにおける入力データの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、プログラムのフェッチや、プログラムがメモリをライトした時、命令を指定回数実行した時等、さらに詳細なデータの入力タイミングを指定することができます。

  ・仮想ポート出力機能
  仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。
  記録されたデータは、I/O タイミング設定ウィンドウでグラフや数値形式で確認できます。
  本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

  ・出力ポートシミュレート機能
  出力ポートのシミュレート機能を利用すると、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値を記録することができます。記録したデータは、ウィンドウに表示、およびファイルへ出力することができます。
  また、Printf 関数で UART へ出力されたデータを確認することができます。
  なお、本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

- 割り込み
  実チップでは、周辺 I/O（外部の割り込み信号も含む）が割り込みの発生要因となりますが、シミュレータでは周辺 I/O に該当するものがありません。 シミュレータではこれに代わるものとして、擬似的に割り込みをかける機能（仮想割り込み）を用意しています。仮想割り込みは、指定されたサイクルや実行アドレス等のタイミングで発生させることができます。

  ・仮想割り込み機能
  仮想割り込み機能とは、割り込みの発生を定義する機能です。この機能を利用すると、擬似的にタイマ割り込みやキー入力割り込み等を発生させることができます。
  仮想割り込みを発生することのできるタイミングを以下に示します。

  1. プログラムの実行が指定サイクルになった時
  2. プログラムが指定アドレスを実行した時
  3. 指定した時間間隔毎

  上記のタイミングにおける仮想割り込みの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、タイマの動作等を記述することができます。

  ・仮想割り込みと実チップの割り込みの相違点
  仮想割り込みは、実チップの割り込みと以下の点が異なります。
  1. ハードウェア割り込みの特殊割り込みは発生させることができません。
  2. 高速割り込みは発生させることができません。
  3. 同一優先順位の仮想割り込みが同時に発生した場合について
     実チップでは 2 つ以上の同じ優先順位の割り込みが同時に発生した場合、ハードウェアで設定されている優先度の高い割り込みが受け付けられますが、仮想割り込みでは、割り込み種別（周辺 I/O 割り込みの種別等）間の優先順位は全て同一として扱います。
     従って、同じ優先順位の仮想割り込みが同時に発生した場合、仮想割り込みが発生する順序は不定となります。

  なお、仮想割り込みを設定するには、以下の２つの方法があります。

  1. I/O タイミング設定ウィンドウを使用して設定する方法
  2. I/O スクリプト機能を使用して設定する方法

それぞれの方法で設定する仮想割り込みには、以下の制限があります。

1. I/O タイミング設定ウィンドウを使用して設定した仮想割り込み
   [仮想割り込みを発生させた場合の割り込み制御について]
   ・各割り込み制御レジスタの割り込み要求ビットは、１になりません。
   ・各割り込み制御レジスタの割り込み優先レベル選択ビットに指定された優先順位は、参照されません。
   仮想割り込みの優先順位は、I/O タイミング設定ウィンドウで仮想割り込みを設定する際に指定することができます。
   ・フラグレジスタ（FLG）の割り込み許可フラグ（I フラグ）とプロセッサ割り込み優先レベル（IPL）は、実チップと同様に参照されます。

2. I/O スクリプトを使用して設定した仮想割り込み
   [仮想割り込みを発生させた場合の割り込み制御について]
   ・割り込みが発生した時に、各割り込み制御レジスタの割り込み要求ビットを１にするような記述ができます。
   ・各割り込み制御レジスタの割り込み優先レベル選択ビットに指定された優先順位を、参照できます。 ただし、仮想割り込みが発生してシミュレータに登録された後に、ユーザプログラムで割り込み優先レベル選択ビットに指定された優先順位が変更されても、一旦登録された仮想割り込みの優先順位は変更されません。
   ・フラグレジスタ（FLG）の割り込み許可フラグ（I フラグ）とプロセッサ割り込み優先レベル（IPL）は、実チップと同様に参照されます。

   ・I/O スクリプト機能
   仮想ポート入力や仮想割り込みの設定をユーザがファイルにスクリプト形式で記述できます。この機能を利用すると、I/O タイミング設定ウィンドウで設定できる仮想ポート入力や仮想割り込みの定義よりもさらに柔軟な定義が可能です。具体的には、ユーザがタイマレジスタに設定した分周比を読み込み、タイマ割り込みを周期的に発生する等です。

注意事項

仮想ポート入出力機能や仮想割り込み機能は、命令実行後のタイミングで処理されます。

- ポート入出力

・GUI 入力機能

GUI 入力機能とは、ユーザターゲットシステムの簡単なキー入力パネルをウィンドウ上で模擬する機能です。キー入力パネルの作成は、GUI 入出力ウィンドウで行います。

入力パネルには、以下のパーツが配置できます。

[ボタン]

押下したタイミングで、仮想ポート入力や仮想割り込みを行うことができます。ボタンには以下のアクションが設定できます。

- 指定アドレスのメモリにデータを入力（仮想ポート入力）
- 指定仮想割り込みの発生
- 指定仮想割り込みと仮想ポート入力を同時に発生

[テキスト]

テキスト文字列を表示します。

・GUI 出力機能

GUI 出力機能とは、ユーザターゲットシステムの簡単な出力パネルをウィンドウ上で模擬する機能です。出力パネルの作成は、GUI 入出力ウィンドウで行います。

出力パネルには、以下のパーツが配置できます。

[文字列]

指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じてユーザが指定した文字列を表示／消去します。

[LED]

指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じて LED の点灯を行います。

[テキスト]

テキスト文字列を表示します。

2.1.5 サイクル数計測 ･･･ CYcle（CY）コマンド

CYcle コマンドを使用することで、実行したプログラムのおおよそのサイクル数と、実行時間を知ることができます。

サイクル数は、マイクロコンピュータのソフトウエアマニュアルに記載されている値を使用しています。

実行時間は､実行した CPU 命令の累積サイクル数と､Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出したターゲットプログラムの実行時間です。

2.1.6 スタック使用量計測 ･･･ StackMonitor（SM）コマンド

StackMonitor コマンドを使用することで、スタックの最大アドレスと最小アドレスが分かり、プログラムがスタック領域のどの部分をどれだけ使用していたかを判断できます。

スタック使用量計測は、Go あるいは GoFree コマンド発行から中断までに行われ、2 つのスタックポインタ（USP レジスタ、ISP レジスタ）のそれぞれで最大値・最小値が記録されます。

プログラム実行中、プログラムによってスタックポインタの値を変更すると、そのスタックポインタの使用量計測はそこで中断します。

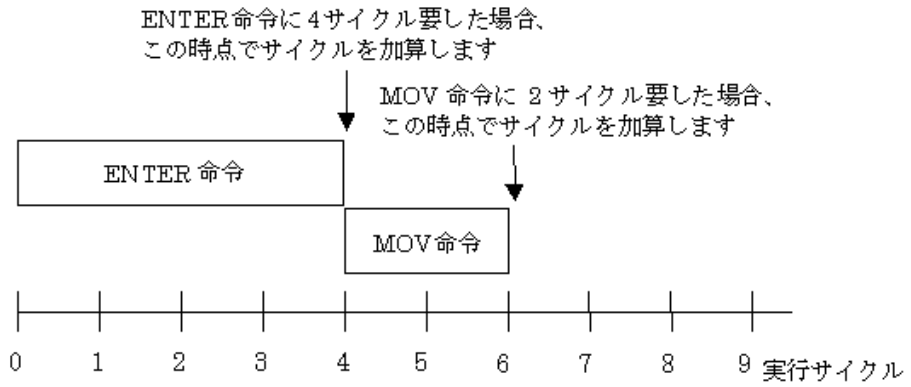
2.2 M32C 用シミュレーション仕様

2.2.1 命令動作

- 命令のサイクル数について
  時間管理は、サイクル単位で行われます。サイクル数は、マイクロコンピュータのソフトウエアマニュアルに記載されている値を使用しています。
  但し、実チップとは、次の点が異なります。
  - ・バス幅、キュー、ウェイト数を考慮したサイクル数の計測は行いません。
  - ・割り込みシーケンス実行サイクル数の計測は行いません（割り込みが発生した場合、割り込みシーケンス実行サイクル数は、0 サイクルになります）。
  - ・サイクル数の計測は、リセット直後から開始し（リセット直後のサイクル数は 0 になります）機械語 1 命令の実行に要したサイクル数を命令実行後に加算していきます（下図をご参照ください）。

注意事項

シミュレータで計測するサイクル数は、バス幅、キュー、ウェイト数等を考慮していないため、実チップのサイクル数と比べると誤差があります。



上記の例では、ENTER 命令や MOV 命令等が実行されている間は、サイクル数は加算されません。命令の実行後にサイクル数が加算されます。

- ターゲットプログラムの実行時間計測
  ターゲットプログラムの実行時間計測は、上記のサイクル数と、Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出します。

注意事項

シミュレータの実行時間計測は、上記サイクル数を使用して算出するため、実チップの実行時間と比べると誤差があります。

- WAIT、BRK2 命令
  NOP 命令として実行します。
  そのほかの命令は、実チップと同一の動作を行います。

- INT、INTO、UND、BRK 命令
  実チップと同様、割り込みが発生します（INTO 命令は O フラグが 1 のときのみ）。

2.2.2 リセット動作

- SFR 領域の初期化は行われません。
- 実行サイクル数は 0 になります。

上記以外の動作は、実チップと同一です。
なお、シミュレータ起動時にもリセットを行います。起動直後はリセットベクタ（FFFFFCh ～ FFFFFFh）に FF0000h という値が設定されているので、シミュレータ起動直後のプログラムカウンタ値は FF0000h となります。

2.2.3 メモリ

- メモリ空間
  マップとしてメモリ空間を設定していれば、000000h ～ FFFFFFh の 16MB のメモリ空間全てが RAM として読み書き可能です。
  ただし、020000h ～ FEFFFFh のメモリ空間は、初期状態ではマップが確保されていない状態になっているため、そのままでこの部分をアクセスしようとするとエラーとなります。これがプログラム実行中であれば、メモリの不正アクセスとして、プログラム実行が中断します。この部分にメモリを割り当てるためにはマップ機能（後述）を使用します。

- 起動直後のメモリ構成とその初期値
  起動直後は、次のようにメモリが設定されています。

000000h ～ 0003FFh（SFR 領域）	00h で埋められています。
000400h ～ 01FFFFh	FFh で埋められています。
020000h ～ FEFFFFh	起動直後はメモリが存在しません。
FF0000h ～ FFFFFBh	FFh で埋められています。
FFFFFCh ～ FFFFFFh（リセットベクタ）	0000FF00h が設定されています。

- マップ機能･･･ MAP コマンド
  シミュレータでは、000000h ～ FFFFFFh のメモリ空間を 256 等分し、64KB ごとにマップとしてメモリの割り付けを行います。このうち、アドレスが最下位のマップ（000000h～ 00FFFFh）とその次のマップ（010000h～ 01FFFFh）、アドレスが最上位のマップ（FF0000h ～ FFFFFFh）はシミュレータ起動時に既に割り付けられています。
  マップの割り付けは、MAP コマンドを使用してください。MAP コマンドで割り付けたメモリは、割り付け直後はすべて FFh で初期化されます。

なお、プログラムをダウンロードするとプログラム領域やデータ領域を自動的にマップ設定します。

注意事項

一度マップ設定して割り付けたメモリ空間は、削除することはできません。

- メモリのない領域にアクセスした場合
  020000h～ FEFFFFh 間の 253 個のメモリマップは、メモリを確保しない限りメモリがありません。この領域をアクセスすると、メモリの不正アクセスとしてエラー中断します。これがプログラム実行中であれば、メモリの不正アクセスとして、プログラム実行が中断します。

2.2.4 I/O

- SFR
  実チップの CPU コア以外の、タイマ、DMAC、シリアル I/O などの周辺 I/O はサポートしていません。周辺 I/O が接続される SFR 領域（000000h～ 0003FFh）も、シミュレータでは RAM として扱います。 ただし、SFR 等のメモリへのデータ入力や、タイマ割り込み等の割り込みを擬似的に実現する方法を用意しています。この方法に関しては、後述の「I/O スクリプト」、「割り込み」をご参照ください。

- I/O スクリプト

・仮想ポート入力機能
仮想ポート入力機能とは、外部から指定アドレスのメモリに入力されるデータの変化を定義する機能です。この機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。 データをメモリに入力できるタイミングを以下に示します。

1. プログラムの実行が指定サイクルになった時
2. 指定されたメモリをプログラムがリードアクセスした時
3. 指定された仮想割り込みが発生した時

上記のタイミングにおける入力データの定義は、I/O タイミング設定ウィンドウで行うことができます。なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、プログラムのフェッチや、プログラムがメモリをライトした時、命令を指定回数実行した時等、さらに詳細なデータの入力タイミングを指定することができます。

・仮想ポート出力機能
仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。
記録されたデータは、I/O タイミング設定ウィンドウでグラフや数値形式で確認できます。
本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

・出力ポートシミュレート機能
出力ポートのシミュレート機能を利用すると、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値を記録することができます。記録したデータは、ウィンドウに表示、およびファイルへ出力することができます。
また、Printf 関数で UART へ出力されたデータを確認することができます。
なお、本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

- 割り込み
  実チップでは、周辺 I/O（外部の割り込み信号も含む）が割り込みの発生要因となりますが、シミュレータでは周辺 I/O に該当するものがありません。 シミュレータではこれに代わるものとして、擬似的に割り込みをかける機能（仮想割り込み）を用意しています。仮想割り込みは、指定されたサイクルや実行アドレス等のタイミングで発生させることができます。

  ・仮想割り込み機能
  仮想割り込み機能とは、割り込みの発生を定義する機能です。この機能を利用すると、擬似的にタイマ割り込みやキー入力割り込み等を発生させることができます。
  仮想割り込みを発生することのできるタイミングを以下に示します。

  1. プログラムの実行が指定サイクルになった時
  2. プログラムが指定アドレスを実行した時
  3. 指定した時間間隔毎

  上記のタイミングにおける仮想割り込みの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、タイマの動作等を記述することができます。

  ・仮想割り込みと実チップの割り込みの相違点
  仮想割り込みは、実チップの割り込みと以下の点が異なります。
  1. ハードウェア割り込みの特殊割り込みは発生させることができません。
     リセット、NMI、DBC、監視タイマ、シングルステップ、アドレス一致割り込みは、発生させることができません。
  2. 高速割り込みは発生させることができません。
  3. 同一優先順位の仮想割り込みが同時に発生した場合について

  実チップでは 2 つ以上の同じ優先順位の割り込みが同時に発生した場合、ハードウェアで設定されている優先度の高い割り込みが受け付けられますが、仮想割り込みでは、割り込み種別（周辺 I/O 割り込みの種別等）間の優先順位は全て同一として扱います。
  従って、同じ優先順位の仮想割り込みが同時に発生した場合、仮想割り込みが発生する順序は不定となります。

  なお、仮想割り込みを設定するには、以下の２つの方法があります。

  1. I/O タイミング設定ウィンドウを使用して設定する方法
  2. I/O スクリプト機能を使用して設定する方法

それぞれの方法で設定する仮想割り込みには、以下の制限があります。

1. I/O タイミング設定ウィンドウを使用して設定した仮想割り込み
   [仮想割り込みを発生させた場合の割り込み制御について]
   ・各割り込み制御レジスタの割り込み要求ビットは、１になりません。
   ・各割り込み制御レジスタの割り込み優先レベル選択ビットに指定された優先順位は、参照されません。
   仮想割り込みの優先順位は、I/O タイミング設定ウィンドウで仮想割り込みを設定する際に指定することができます。
   ・フラグレジスタ（FLG）の割り込み許可フラグ（I フラグ）とプロセッサ割り込み優先レベル（IPL）は、実チップと同様に参照されます。

2. I/O スクリプトを使用して設定した仮想割り込み
   [仮想割り込みを発生させた場合の割り込み制御について]
   ・割り込みが発生した時に、各割り込み制御レジスタの割り込み要求ビットを１にするような記述ができます。
   ・各割り込み制御レジスタの割り込み優先レベル選択ビットに指定された優先順位を、参照できます。 ただし、仮想割り込みが発生してシミュレータに登録された後に、ユーザプログラムで割り込み優先レベル選択ビットに指定された優先順位が変更されても、一旦登録された仮想割り込みの優先順位は変更されません。
   ・フラグレジスタ（FLG）の割り込み許可フラグ（I フラグ）とプロセッサ割り込み優先レベル（IPL）は、実チップと同様に参照されます。

・I/O スクリプト機能
仮想ポート入力や仮想割り込みの設定をユーザがファイルにスクリプト形式で記述できます。この機能を利用すると、I/O タイミング設定ウィンドウで設定できる仮想ポート入力や仮想割り込みの定義よりもさらに柔軟な定義が可能です。具体的には、ユーザがタイマレジスタに設定した分周比を読み込み、タイマ割り込みを周期的に発生する等です。

注意事項

仮想ポート入出力機能や仮想割り込み機能は、命令実行後のタイミングで処理されます。

- ポート入出力

・GUI 入力機能

GUI 入力機能とは、ユーザターゲットシステムの簡単なキー入力パネルをウィンドウ上で模擬する機能です。キー入力パネルの作成は、GUI 入出力ウィンドウで行います。
入力パネルには、以下のパーツが配置できます。

[ボタン]
押下したタイミングで、仮想ポート入力や仮想割り込みを行うことができます。ボタンには以下のアクションが設定できます。
- 指定アドレスのメモリにデータを入力（仮想ポート入力）
- 指定仮想割り込みの発生
- 指定仮想割り込みと仮想ポート入力を同時に発生

[テキスト]
テキスト文字列を表示します。

・GUI 出力機能

GUI 出力機能とは、ユーザターゲットシステムの簡単な出力パネルをウィンドウ上で模擬する機能です。出力パネルの作成は、GUI 入出力ウィンドウで行います。
出力パネルには、以下のパーツが配置できます。

[文字列]
指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じてユーザが指定した文字列を表示／消去します。

[LED]
指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じて LED の点灯を行います。

[テキスト]
テキスト文字列を表示します。

2.2.5 サイクル数計測 ･･･ CYcle（CY）コマンド

CYcle コマンドを使用することで、実行したプログラムのおおよそのサイクル数と、実行時間を知ることができます。
サイクル数は、マイクロコンピュータのソフトウエアマニュアルに記載されている値を使用しています。
実行時間は､実行した CPU 命令の累積サイクル数と､Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出したターゲットプログラムの実行時間です。

2.2.6 スタック使用量計測 ･･･ StackMonitor（SM）コマンド

StackMonitor コマンドを使用することで、スタックの最大アドレスと最小アドレスが分かり、プログラムがスタック領域のどの部分をどれだけ使用していたかを判断できます。
スタック使用量計測は、Go あるいは GoFree コマンド発行から中断までに行われ、2 つのスタックポインタ（USP レジスタ、ISP レジスタ）のそれぞれで最大値・最小値が記録されます。
プログラム実行中、プログラムによってスタックポインタの値を変更すると、そのスタックポインタの使用量計測はそこで中断します。

2.3 M16C/R8C 用シミュレーション仕様

2.3.1 命令動作

- 命令のサイクル数について
  時間管理は、サイクル単位で行われます。

[シミュレーション速度モード]

サイクル数は、マイクロコンピュータのソフトウエアマニュアルに記載されている値を使用しています。

但し、実チップとは、次の点が異なります。

・バス幅、キュー、ウェイト数を考慮したサイクル数の計測は行いません。

・割り込みシーケンス実行サイクル数の計測は行いません（割り込みが発生した場合、割り込みシーケンス実行サイクル数は、0 サイクルになります）。

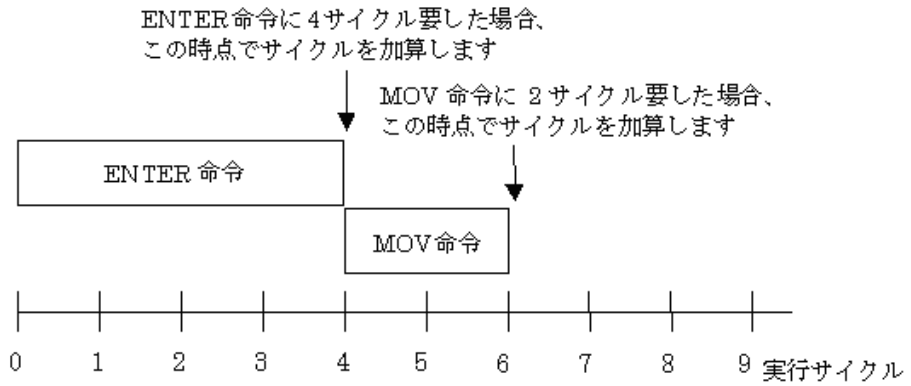
・サイクル数の計測は、リセット直後から開始し（リセット直後のサイクル数は 0 になります）機械語 1 命令の実行に要したサイクル数を命令実行後に加算していきます（下図をご参照ください）。

注意事項

サイクル数は、バス幅、キュー、ウェイト数等を考慮していないため、実チップのサイクル数と比べると誤差があります。

R8C ファミリの場合、命令キューバッファに命令コードが揃っており、メモリに対し 8 ビットのデータをソフトウエアウエイトなしに読み書きする場合のサイクル数です。

そのため、プログラムをエミュレータデバッガで実行したときの命令キューバッファ等の状態によっては、シミュレータデバッガでのサイクル数がエミュレータデバッガと比較して、1/2 程度になる場合もありますので、ご注意ください。



上記の例では、ENTER 命令や MOV 命令等が実行されている間は、サイクル数は加算されません。命令の実行後にサイクル数が加算されます。

[サイクル精度モード]

バス幅、命令キューバッファ、ウェイト数を考慮したサイクル数の計測を行います。ただし、実チップの結果と完全には一致しない場合がありますので、厳密な実行サイクル数の計測には、コンパクトエミュレータ、あるいは、フルスペックエミュレータをご使用ください。

- ターゲットプログラムの実行時間計測
  ターゲットプログラムの実行時間計測は、上記のサイクル数と、Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出します。

  注意事項

  シミュレータの実行時間計測は、上記サイクル数を使用して算出するため、実チップの実行時間と比べると誤差があります。

- WAIT 命令
  NOP 命令として実行します。
  そのほかの命令は、実チップと同一の動作を行います。

- INT、INTO、UND、BRK 命令
  実チップと同様、割り込みが発生します（INTO 命令は O フラグが 1 のときのみ）。

2.3.2 リセット動作

- SFR 領域の初期化は行われません。
- 実行サイクル数は 0 になります。

上記以外の動作は、実チップと同一です。 なお、シミュレータ起動時にもリセットを行います。起動直後はリセットベクタに 000F0000h という値が設定されているので、シミュレータ起動直後のプログラムカウンタ値は F000016 となります。

2.3.3 メモリ

- メモリ空間
  マップとしてメモリ空間を設定していれば、00000h ～ FFFFFh の 1MB のメモリ空間全てが RAM として読み書き可能です。
  ただし、10000h ～ EFFFFh のメモリ空間は、初期状態ではマップが確保されていない状態になっているため、そのままでこの部分をアクセスしようとするとエラーとなります。これがプログラム実行中であれば、メモリの不正アクセスとして、プログラム実行が中断します。この部分にメモリを割り当てるためにはマップ機能（後述）を使用します。

- 起動直後のメモリ構成とその初期値
  起動直後は、次のようにメモリが設定されています。

00000h ～ 003FFh	00h で埋められています。
00400h ～ 0FFFFh	FFh で埋められています。
10000h ～ EFFFFh	起動直後はメモリが存在しません。
F0000h ～ FFFFFh	FFh で埋められています。
リセットベクタ	000F0000h が設定されています。

- マップ機能･･･ MAP コマンド
  シミュレータでは、00000h ～ FFFFFh のメモリ空間を 256 バイトごとにマップとしてメモリの割り付けを行います。このうち、アドレスが最下位のマップ（00000h～ 0FFFFh）と、アドレスが最上位のマップ（F0000h ～ FFFFFh）はシミュレータ起動時に既に割り付けられています。
  マップの割り付けは、MAP コマンドを使用してください。MAP コマンドで割り付けたメモリは、割り付け直後はすべて FFh で初期化されます。

なお、プログラムをダウンロードするとプログラム領域やデータ領域を自動的にマップ設定します。

注意事項

- 一度マップ設定して割り付けたメモリ空間は、削除することはできません。
- Init ダイアログの MCU タブでシミュレーション速度モードを選択した時は、メモリの割り付けに MCU Setting ダイアログのメモリタブを使用できません。

- メモリのない領域にアクセスした場合
  10000h ～ EFFFFh 間のメモリマップは、メモリを確保しない限りメモリがありません。この領域をアクセスすると、メモリの不正アクセスとしてエラー中断します。これがプログラム実行中であれば、メモリの不正アクセスとして、プログラム実行が中断します。

2.3.4 I/O

- SFR
  実チップの CPU コア以外の、タイマ、DMAC、シリアル I/O などの周辺 I/O はサポートしていません。周辺 I/O が接続される SFR 領域も、シミュレータでは RAM として扱います。
  ただし、SFR 等のメモリへのデータ入力や、タイマ割り込み等の割り込みを擬似的に実現する方法を用意しています。この方法に関しては、後述の「I/O スクリプト」、「割り込み」をご参照ください。
- I/O スクリプト
  ・仮想ポート入力機能
  仮想ポート入力機能とは、外部から指定アドレスのメモリに入力されるデータの変化を定義する機能です。この機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。
  データをメモリに入力できるタイミングを以下に示します。
  1. プログラムの実行が指定サイクルになった時
  2. 指定されたメモリをプログラムがリードアクセスした時
  3. 指定された仮想割り込みが発生した時

  上記のタイミングにおける入力データの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、プログラムのフェッチや、プログラムがメモリをライトした時、命令を指定回数実行した時等、さらに詳細なデータの入力タイミングを指定することができます。

  ・仮想ポート出力機能
  仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。
  記録されたデータは、I/O タイミング設定ウィンドウでグラフや数値形式で確認できます。
  本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

  ・出力ポートシミュレート機能
  出力ポートのシミュレート機能を利用すると、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値を記録することができます。記録したデータは、ウィンドウに表示、およびファイルへ出力することができます。
  また、Printf 関数で UART へ出力されたデータを確認することができます。
  なお、本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

- 割り込み
  実チップでは、周辺 I/O（外部の割り込み信号も含む）が割り込みの発生要因となりますが、シミュレータでは周辺 I/O に該当するものがありません。 シミュレータではこれに代わるものとして、擬似的に割り込みをかける機能（仮想割り込み）を用意しています。仮想割り込みは、指定されたサイクルや実行アドレス等のタイミングで発生させることができます。

  ・仮想割り込み機能
  仮想割り込み機能とは、割り込みの発生を定義する機能です。この機能を利用すると、擬似的にタイマ割り込みやキー入力割り込み等を発生させることができます。
  仮想割り込みを発生することのできるタイミングを以下に示します。

  1. プログラムの実行が指定サイクルになった時
  2. プログラムが指定アドレスを実行した時
  3. 指定した時間間隔毎

  上記のタイミングにおける仮想割り込みの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、タイマの動作等を記述することができます。

  ・仮想割り込みと実チップの割り込みの相違点
  仮想割り込みは、実チップの割り込みと以下の点が異なります。
  1. ハードウェア割り込みの特殊割り込みは発生させることができません。
     リセット、NMI、DBC、監視タイマ、シングルステップ、アドレス一致割り込みは、発生させることができません。
  2. 同一優先順位の仮想割り込みが同時に発生した場合について
     実チップでは 2 つ以上の同じ優先順位の割り込みが同時に発生した場合、ハードウェアで設定されている優先度の高い割り込みが受け付けられますが、仮想割り込みでは、割り込み種別（周辺 I/O 割り込みの種別等）間の優先順位は全て同一として扱います。
     従って、同じ優先順位の仮想割り込みが同時に発生した場合、仮想割り込みが発生する順序は不定となります。

  なお、仮想割り込みを設定するには、以下の２つの方法があります。

  1. I/O タイミング設定ウィンドウを使用して設定する方法
  2. I/O スクリプト機能を使用して設定する方法

それぞれの方法で設定する仮想割り込みには、以下の制限があります。

1. I/O タイミング設定ウィンドウを使用して設定した仮想割り込み
   [仮想割り込みを発生させた場合の割り込み制御について]
   - 各割り込み制御レジスタの割り込み要求ビットは、1 になりません。
   - 各割り込み制御レジスタの割り込み優先レベル選択ビットに指定された優先順位は、参照されません。
     仮想割り込みの優先順位は、I/O タイミング設定ウィンドウで仮想割り込みを設定する際に指定することができます。
   - フラグレジスタ（FLG）の割り込み許可フラグ（I フラグ）とプロセッサ割り込み優先レベル（IPL）は、実チップと同様に参照されます。
2. I/O スクリプトを使用して設定した仮想割り込み
   [仮想割り込みを発生させた場合の割り込み制御について]
   - 割り込みが発生した時に、各割り込み制御レジスタの割り込み要求ビットを 1 にするような記述ができます。
   - 各割り込み制御レジスタの割り込み優先レベル選択ビットに指定された優先順位を、参照できます。 ただし、仮想割り込みが発生してシミュレータに登録された後に、ユーザプログラムで割り込み優先レベル選択ビットに指定された優先順位が変更されても、一旦登録された仮想割り込みの優先順位は変更されません。
   - フラグレジスタ（FLG）の割り込み許可フラグ（I フラグ）とプロセッサ割り込み優先レベル（IPL）は、実チップと同様に参照されます。

・I/O スクリプト機能
仮想ポート入力や仮想割り込みの設定をユーザがファイルにスクリプト形式で記述できます。この機能を利用すると、I/O タイミング設定ウィンドウで設定できる仮想ポート入力や仮想割り込みの定義よりもさらに柔軟な定義が可能です。具体的には、ユーザがタイマレジスタに設定した分周比を読み込み、タイマ割り込みを周期的に発生する等です。

注意事項

仮想ポート入出力機能や仮想割り込み機能は、命令実行後のタイミングで処理されます。

- ポート入出力
  ・GUI 入力機能
  GUI 入力機能とは、ユーザターゲットシステムの簡単なキー入力パネルをウィンドウ上で模擬する機能です。キー入力パネルの作成は、GUI 入出力ウィンドウで行います。
  入力パネルには、以下のパーツが配置できます。

  [ボタン]
  押下したタイミングで、仮想ポート入力や仮想割り込みを行うことができます。ボタンには以下のアクションが設定できます。
  ・ 指定アドレスのメモリにデータを入力（仮想ポート入力）
  ・ 指定仮想割り込みの発生
  ・ 指定仮想割り込みと仮想ポート入力を同時に発生

  [テキスト]
  テキスト文字列を表示します。

  ・GUI 出力機能
  GUI 出力機能とは、ユーザターゲットシステムの簡単な出力パネルをウィンドウ上で模擬する機能です。出力パネルの作成は、GUI 入出力ウィンドウで行います。
  出力パネルには、以下のパーツが配置できます。
  [文字列]
  定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じてユーザが指定した文字列を表示／消去します。

  [LED]
  指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じて LED の点灯を行います。

  [テキスト]
  テキスト文字列を表示します。

2.3.5 サイクル数計測 ･･･ CYcle（CY）コマンド

CYcle コマンドを使用することで、実行したプログラムのおおよそのサイクル数と、実行時間を知ることができます。
実行時間は、実行した CPU 命令の累積サイクル数と、Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出したターゲットプログラムの実行時間です。

2.3.6 スタック使用量計測 ･･･ StackMonitor（SM）コマンド

StackMonitor コマンドを使用することで、スタックの最大アドレスと最小アドレスが分かり、プログラムがスタック領域のどの部分をどれだけ使用していたかを判断できます。
スタック使用量計測は、Go あるいは GoFree コマンド発行から中断までに行われ、2 つのスタックポインタ（USP レジスタ、ISP レジスタ）のそれぞれで最大値・最小値が記録されます。
プログラム実行中、プログラムによってスタックポインタの値を変更すると、そのスタックポインタの使用量計測はそこで中断します。

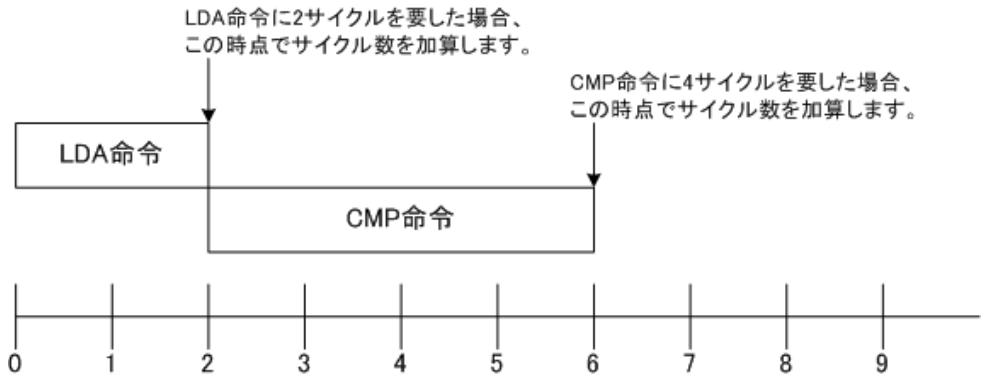
2.4 740 用シミュレーション仕様

2.4.1 命令動作

- 命令のサイクル数について
  時間管理は、サイクル単位で行われます。サイクル数は、「740 ファミリ ソフトウェアマニュアル」に記載されている値を使用しています。
  但し、実チップとは、次の点が異なります。
  ・サイクル数の計測は、リセット直後から開始し（リセット直後のサイクル数は 0 になります）機械語 1 命令の実行に要したサイクル数を命令実行後に加算していきます（下図をご参照ください）。

注意事項

シミュレータで計測するサイクル数は、バス幅、キュー、ウェイト数等を考慮していないため、実チップのサイクル数と比べると誤差があります。



上記の例では、LDA 命令や CMP 命令等が実行されている間は、サイクル数は加算されません。命令の実行後にサイクル数が加算されます。

- ターゲットプログラムの実行時間計測
  ターゲットプログラムの実行時間計測は、上記のサイクル数と、Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出します。

注意事項

シミュレータの実行時間計測は、上記サイクル数を使用して算出するため、実チップの実行時間と比べると誤差があります。

- WIT,STP 命令
  NOP 命令として実行します。
  そのほかの命令は、実チップと同一の動作を行います。

2.4.2 リセット動作

- S レジスタは 0xFF16、PS レジスタは I フラグのみ"1"となり、プログラムカウンタはリセットベクタの値になります。それ以外のレジスタは 0 で初期化されます。
- SFR 領域の初期化は行われません。
- 実行サイクル数は 0 になります。

上記以外の動作は、実チップと同一です。 なお、シミュレータ起動時にもリセットを行います。起動直後はリセットベクタに 0000h という値が設定されているので、シミュレータ起動直後のプログラムカウンタ値は 0000h となります。

2.4.3 メモリ

- メモリ空間
  メモリ空間(0000h～ FFFFh)は、全てが RAM として読み書き可能です。

- 起動直後のメモリ構成とその初期値
  起動直後は、次のようにメモリが設定されています。

0000h～ FFFFh	00h で埋められています。

2.4.4 I/O

- SFR
  実チップの CPU コア以外の、タイマ、シリアル I/O などの周辺 I/O はサポートしていません。周辺 I/O が接続される SFR 領域も、シミュレータでは RAM として扱います。
  ただし、CPU モードレジスタのスタックページ選択ビットおよび割り込み制御レジスタは、それぞれ SFR として扱います(各レジスタの配置は、チップのユーザズマニュアルを参照ください。)
  スタックページ選択ビットを 1 にした時、1 ページ内の RAM をスタック領域として使用することができます。 割り込み制御レジスタのビットを 1 にした時、そのビットに対応する割り込み発生が許可されます。 また、タイマ割り込みなどの割り込みや、SFR 等のメモリへのデータ入力を擬似的に実現する方法を用意しています。 この方法に関しては、後述の仮想ポート入出力機能、仮想割り込み機能を参照下さい。
- I/O スクリプト
  ・仮想ポート入力機能
  仮想ポート入力機能とは、外部から指定アドレスのメモリに入力されるデータの変化を定義する機能です。この機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。
  データをメモリに入力できるタイミングを以下に示します。

  1. プログラムの実行が指定サイクルになった時
  2. 指定されたメモリをプログラムがリードアクセスした時
  3. 指定された仮想割り込みが発生した時

  上記のタイミングにおける入力データの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、プログラムのフェッチや、プログラムがメモリをライトした時、命令を指定回数実行した時等、さらに詳細なデータの入力タイミングを指定することができます。

  ・仮想ポート出力機能
  仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。
  記録されたデータは、I/O タイミング設定ウィンドウでグラフや数値形式で確認できます。
  本機能で記録できるデータの個数は、プログラム実行開始時から Init ダイアログの I/O Script タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。

- 割り込み
  実チップでは、周辺 I/O（外部の割り込み信号も含む）が割り込みの発生要因となりますが、シミュレータでは周辺 I/O に該当するものがありません。 シミュレータではこれに代わるものとして、擬似的に割り込みをかける機能（仮想割り込み）を用意しています。仮想割り込みは、指定されたサイクルや実行アドレス等のタイミングで発生させることができます。
  ・仮想割り込み機能
  仮想割り込み機能とは、割り込みの発生を定義する機能です。この機能を利用すると、擬似的にタイマ割り込みや A-D 変換割り込み等を発生させることができます。
  仮想割り込みを発生することのできるタイミングを以下に示します。

  1. プログラムの実行が指定サイクルになった時
  2. プログラムが指定アドレスを実行した時
  3. 指定した時間間隔毎

  上記のタイミングにおける仮想割り込みの定義は、I/O タイミング設定ウィンドウで行うことができます。
  なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能）を利用すると、タイマの動作等を記述することができます。

  ・仮想割り込みと実チップの割り込みの相違点
  仮想割り込みは、実チップの割り込みと以下の点が異なります。

  割り込み制御レジスタ/割り込み要求レジスタについて
  仮想割り込みを発生させた場合、割り込み制御レジスタの割り込み制御ビットを参照し、仮想割り 込み発生をシミュレートしています。 割り込み発生不許可時に仮想割り込みが発生した場合、割り込み要求がシミュレータ内部に保存さ れ、割り込み発生許可後に仮想割り込みが発生します。 ただし、割り込み要求ビットはシミュレートされないため、割り込み要求保存時でも、割り込み 要求ビットはセットされません。 また、割り込み要求ビットをクリアした場合にも、保存されている仮想割り込みは削除できません。（リセット時にシミュレータ内部に保存されている仮想割り込みは削除されます。）なお、I/O スクリプト機能を利用すると、割り込みが発生した時に割り込み要求ビットをセットす るような記述を行うことが可能です。
  リセット割り込みは、発生させることができません。

  なお、仮想割り込みを設定するには、以下の２つの方法があります。

  1. I/O タイミング設定ウィンドウを使用して設定する方法
  2. I/O スクリプト機能を使用して設定する方法

  ・I/O スクリプト機能
  仮想ポート入力や仮想割り込みの設定をユーザがファイルにスクリプト形式で記述できます。この機能を利用すると、I/O タイミング設定ウィンドウで設定できる仮想ポート入力や仮想割り込みの定義よりもさらに柔軟な定義が可能です。具体的には、ユーザがタイマレジスタに設定した分周比を読み込み、タイマ割り込みを周期的に発生する等です。

注意事項

仮想ポート入出力機能や仮想割り込み機能は、命令実行後のタイミングで処理されます。

- ポート入出力
  ・GUI 入力機能
  GUI 入力機能とは、ユーザターゲットシステムの簡単なキー入力パネルをウィンドウ上で模擬する機能です。キー入力パネルの作成は、GUI 入出力ウィンドウで行います。
  入力パネルには、以下のパーツが配置できます。

  [ボタン]
  押下したタイミングで、仮想ポート入力や仮想割り込みを行うことができます。ボタンには以下のアクションが設定できます。
  ・ 指定アドレスのメモリにデータを入力（仮想ポート入力）
  ・ 指定仮想割り込みの発生
  ・ 指定仮想割り込みと仮想ポート入力を同時に発生

  [テキスト]
  テキスト文字列を表示します。

  ・GUI 出力機能
  GUI 出力機能とは、ユーザターゲットシステムの簡単な出力パネルをウィンドウ上で模擬する機能です。出力パネルの作成は、GUI 入出力ウィンドウで行います。
  出力パネルには、以下のパーツが配置できます。

  [文字列]
  指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じてユーザが指定した文字列を表示／消去します。

  [LED]
  指定アドレスのメモリにある値が書き込まれた（ライトされた）時、あるいは、ビットの 1/0 に応じて LED の点灯を行います。

  [テキスト]
  テキスト文字列を表示します。

2.4.5 サイクル数計測 ･･･ CYcle（CY）コマンド

CYcle コマンドを使用することで、実行したプログラムのおおよそのサイクル数と、実行時間を知ることができます。
サイクル数は、「740 ファミリ ソフトウエアマニュアル」に記載されている値を使用しています。
実行時間は､実行した CPU 命令の累積サイクル数と､Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出したターゲットプログラムの実行時間です。

2.4.6 スタック使用量計測 ･･･ StackMonitor（SM）コマンド

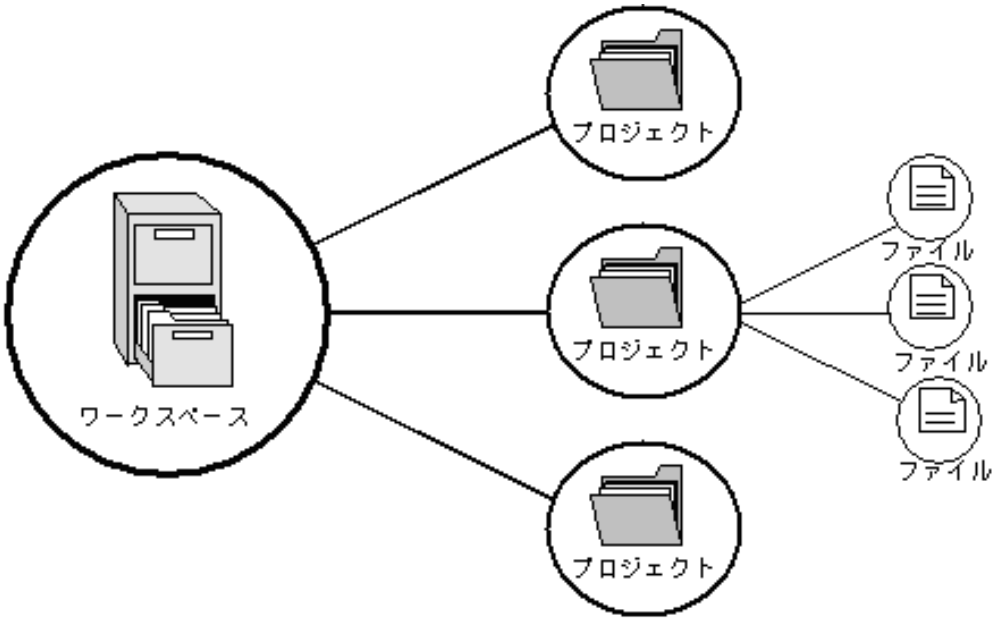
StackMonitor コマンドを使用することで、スタックの最大アドレスと最小アドレスが分かり、プログラムがスタック領域のどの部分をどれだけ使用していたかを判断できます。
スタック使用量計測は、Go あるいは GoFree コマンド発行から中断までに行われ、スタックポインタ（S レジスタ）の最大値・最小値が記録されます。
プログラム実行中、プログラムによってスタックポインタの値を変更すると、そのスタックポインタの使用量計測はそこで中断します。

3. デバッグの準備

本製品を起動し、 シミュレータ 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。

3.1 ワークスペース，プロジェクト，ファイルについて

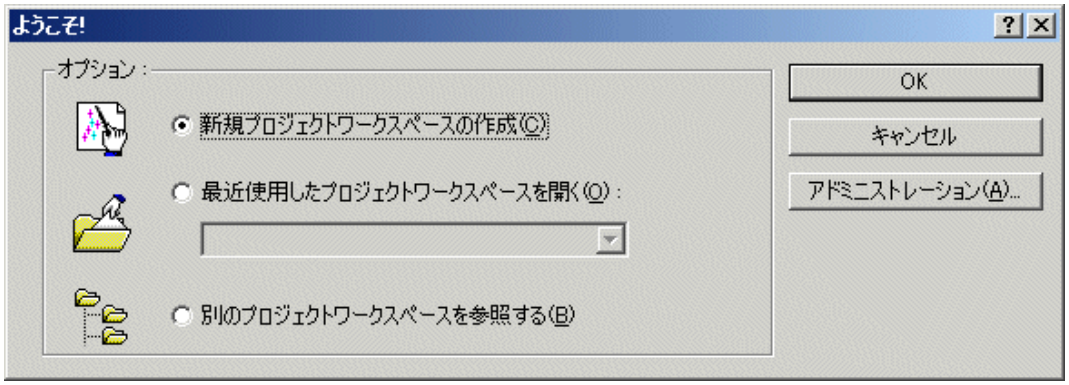
ワードプロセッサでドキュメントを作成、修正できるのと同じように、本製品ではワークスペースを作成、修正できます。
ワークスペースはプロジェクトを入れる箱と考えることができます。 同じように、プロジェクトはプロジェクトファイルを入れる箱と考えることができます。 したがって各ワークスペースにはプロジェクトが 1 つ以上あり、各プロジェクトにはファイルが 1 つ以上あります。



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。

3.2 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。

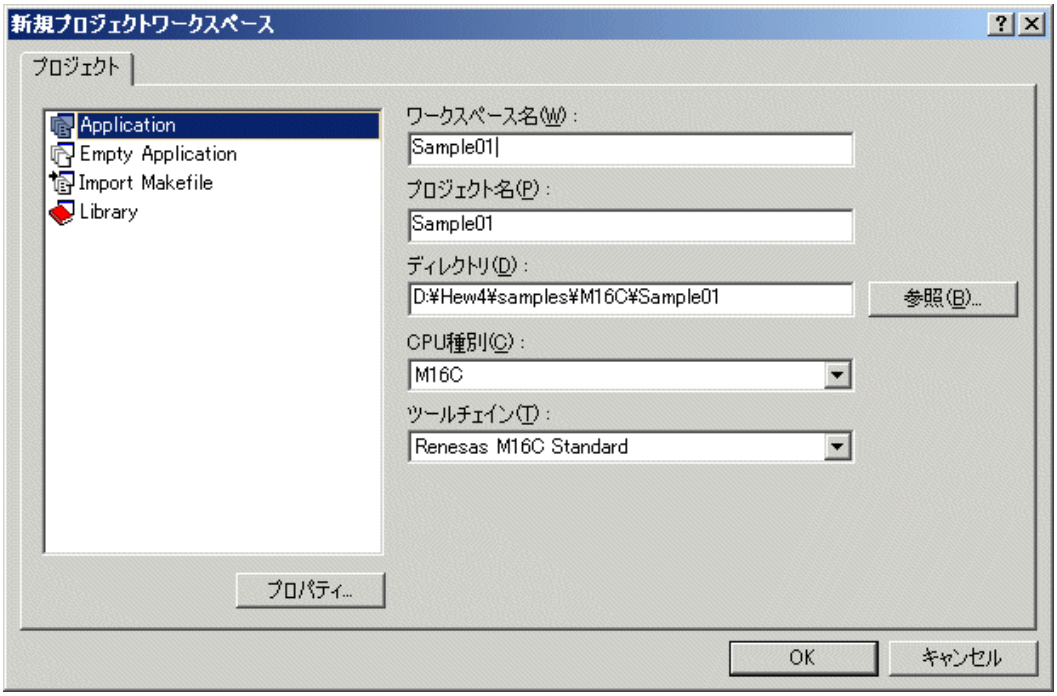
新規ワークスペースの作成方法については、以下を参照ください。
「3.2.1 新規にワークスペースを作成する（ツールチェイン使用）」を参照ください。
「3.2.2 新規にワークスペースを作成する場合（ツールチェイン未使用）」を参照ください。
※既存のロードモジュールファイルを本製品でデバッグする場合などは、この方法でワークスペースを作成します。

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。

3.2.1 新規にワークスペースを作成する（ツールチェイン使用）

3.2.1.1 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。

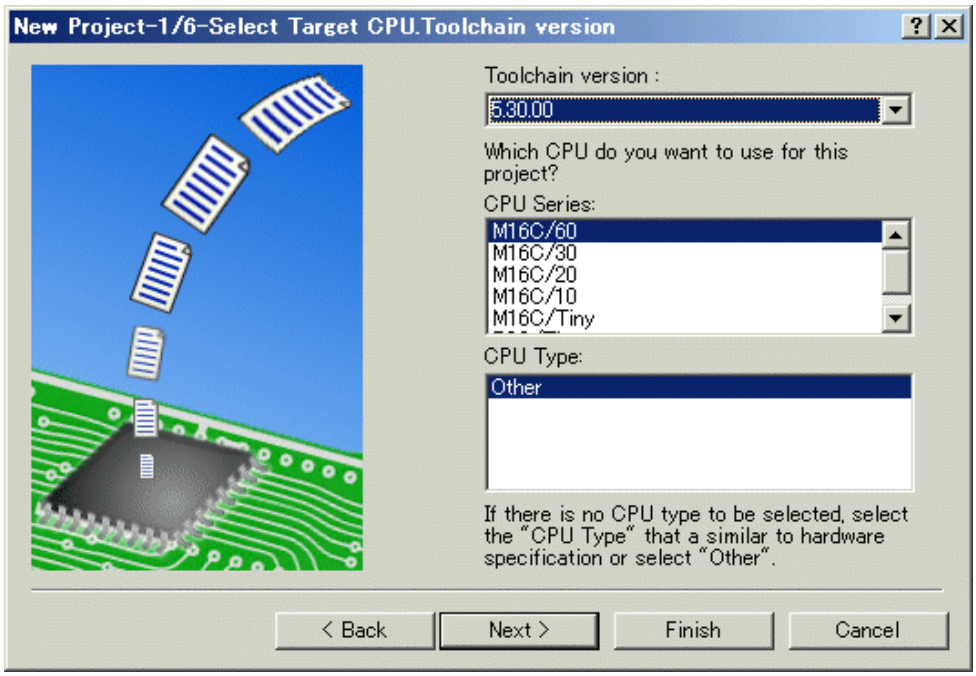


1. CPU 種別を選択する
   [CPU 種別]ドロップダウンリストボックスで、使用する CPU ファミリを選択してください。
2. ツールチェインを選択する
   [ツールチェイン]ドロップダウンリストボックスで、該当するツールチェイン名を選択してください。
3. プロジェクトタイプを選択する
   左の[プロジェクトタイプ]リストボックスで、使用したいプロジェクトタイプを選択します。
   ここで、"Application" を選択してください。
   （選択できるプロジェクトタイプの詳細については、C/C++コンパイラパッケージ付属のマニュアルを参照ください。）
4. ワークスペース名、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よろしければ、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。

3.2.1.2 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。

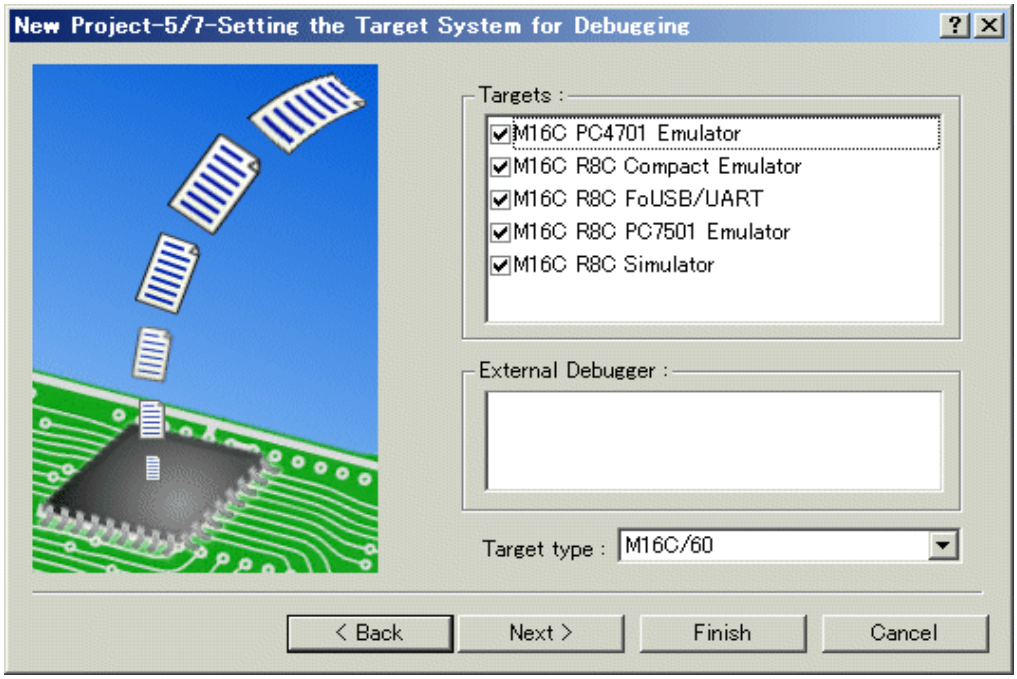
- ツールチェインの設定
- リアルタイム OS に関する設定（使用する場合）
- 生成ファイル、ヒープ領域、スタック領域等の設定

必要な情報を入力し、[次へ]ボタンを押していってください。

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。

3.2.1.3 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。
ツールチェインの設定が終了したら、以下の画面が表示されます。

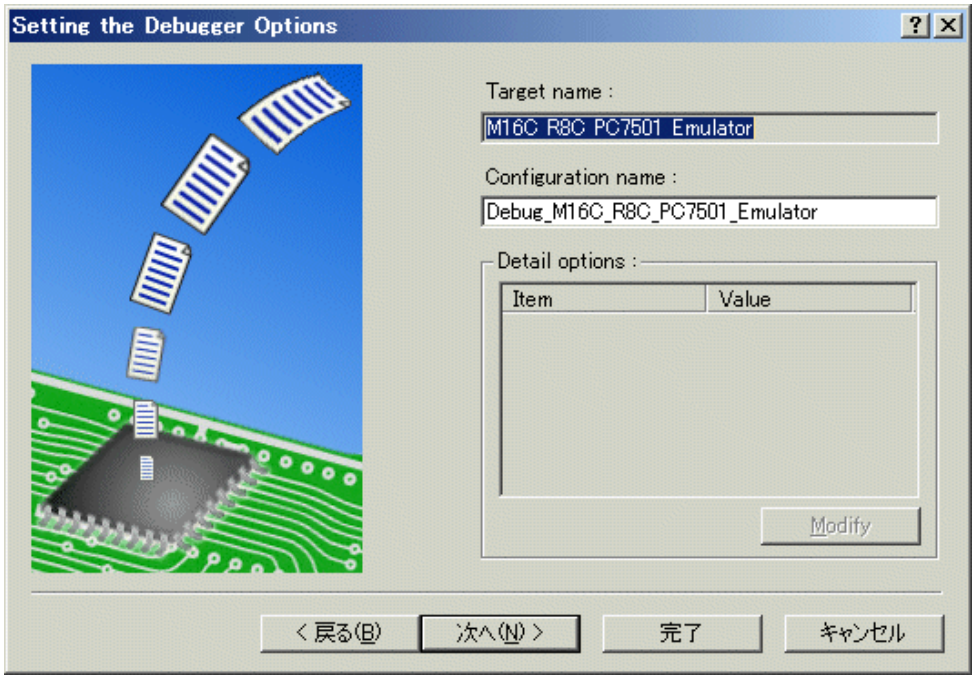


1. ターゲットタイプの選択
   [Target type]ドロップダウンリストボックスで、使用するターゲットの CPU タイプを選択ください。
2. ターゲットプラットフォームの選択 [Targets]領域に、使用可能なターゲットが表示されます。使用するターゲットをチェックしてください（複数指定可能）。

入力後、[次へ]ボタンを押してください。

3.2.1.4 Step4： コンフィグレーションファイル名の設定

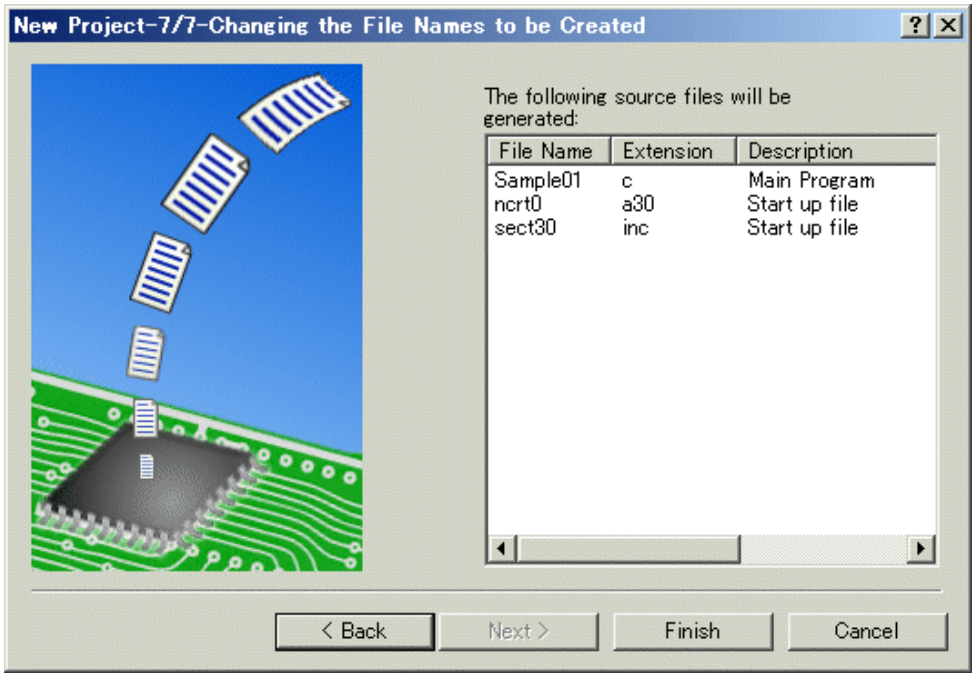
選択したターゲット毎にコンフィグレーションファイル名を設定します。
コンフィグレーションとは、ターゲット以外の High-performance Embedded Workshop の状態を保存するファイルです。



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。

3.2.1.5 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。



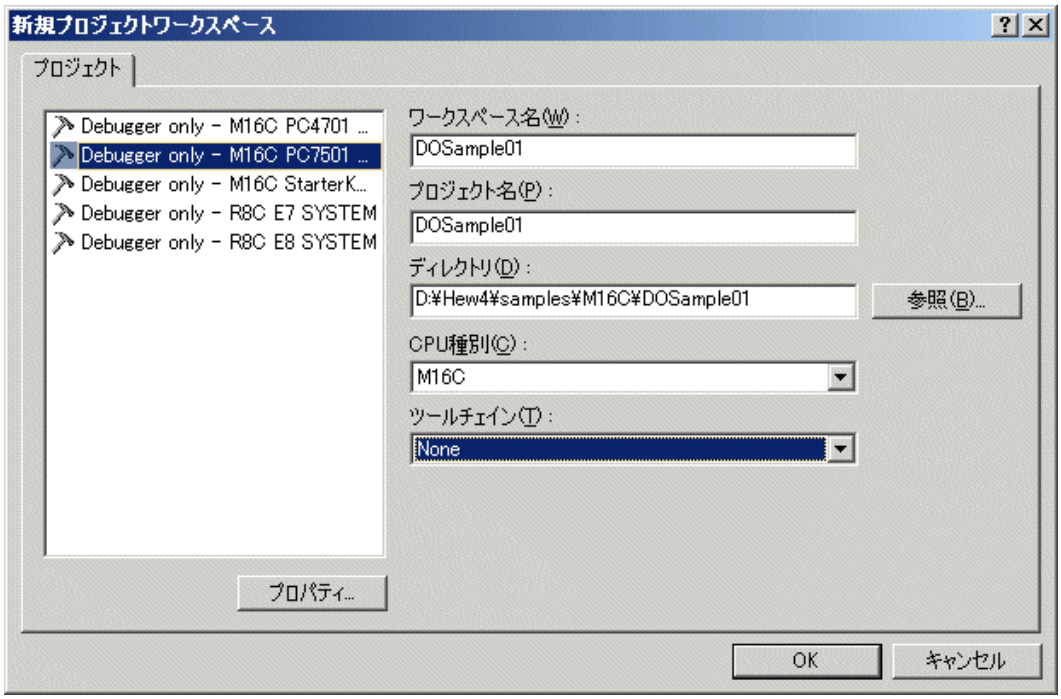
これで シミュレータ に関する設定は終了です。
画面の指示に従い、プロジェクト作成ウィザードを終了してください。

3.2.2 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くても OK です。）

3.2.2.1 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。

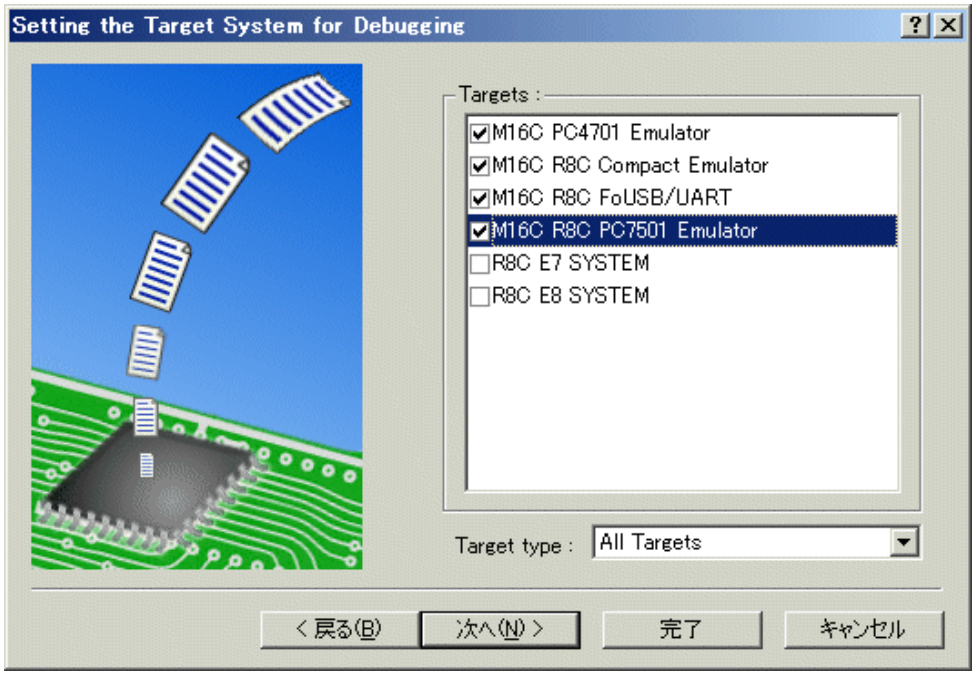


1. CPU 種別を選択する
   [CPU 種別]ドロップダウンリストボックスで、使用する CPU ファミリを選択してください。
2. ツールチェインを選択する
   ツールチェインは使用しませんので、[ツールチェイン]ドロップダウンリストボックスでは"None" を指定してください。
   （ツールチェインが登録されていない場合は、このドロップダウンリストは選択できません（選択不要）。）
3. プロジェクトタイプを選択する
   ツールチェインを使用しない場合、左の[プロジェクトタイプ]リストボックスには"Debugger only - ターゲット名" と表示されますので、 それを選択ください（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よろしければ、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。

3.2.2.2 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。

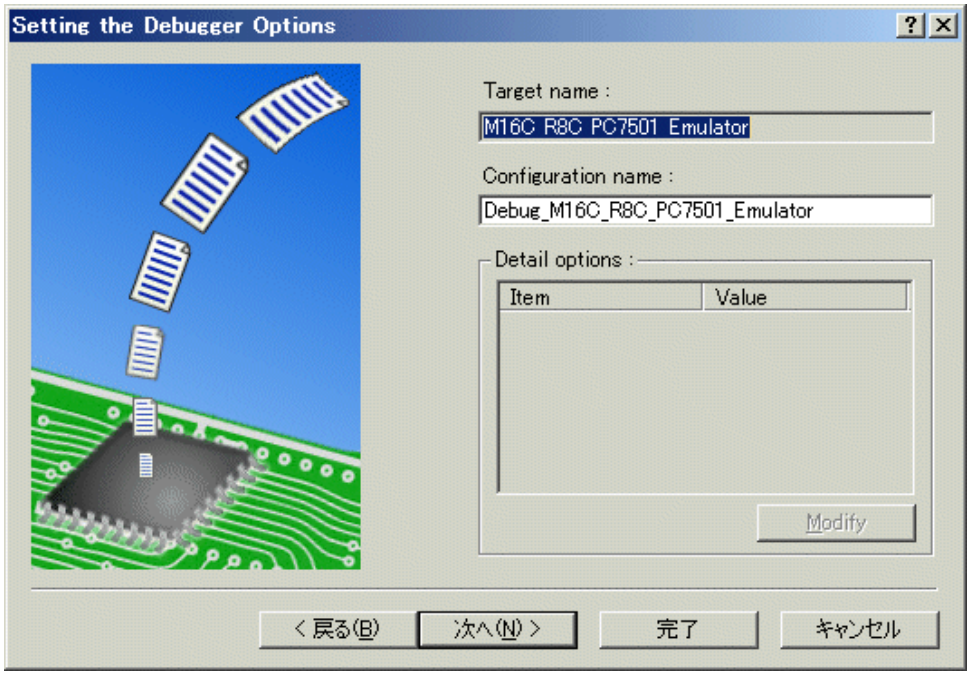


1. ターゲットタイプの選択
   [Target type]ドロップダウンリストボックスで、使用するターゲットの CPU タイプを選択ください。
2. ターゲットプラットフォームの選択
   [Targets]領域に、使用可能なターゲットが表示されます。
   使用するターゲットをチェックしてください（複数指定可能）。

入力後、[次へ]ボタンを押してください。

3.2.2.3 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。

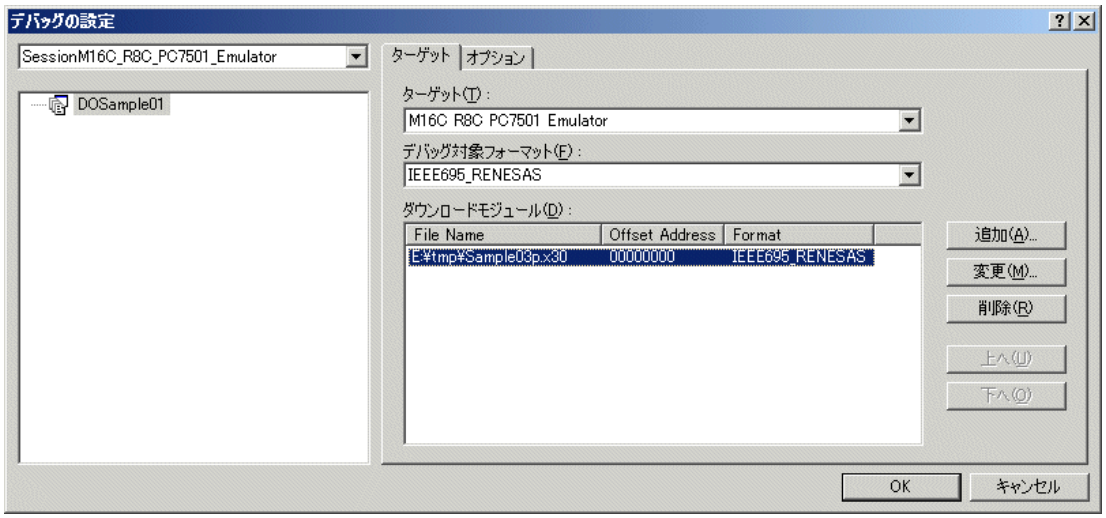


デフォルトの名前がすでに設定されていますので、変更する必要がなければそのまま[次へ]ボタンで進んでください。
これで シミュレータ に関する設定は終了です。
画面の指示に従い、プロジェクト作成ウィザードを終了してください。
High-performance Embedded Workshop 起動と同時に、 デバッガのセットアップを開始するダイアログが表示されます。 シミュレータ の準備が完了していれば、そのままセットアップを行い、 シミュレータへ接続してください。

3.2.2.4 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。

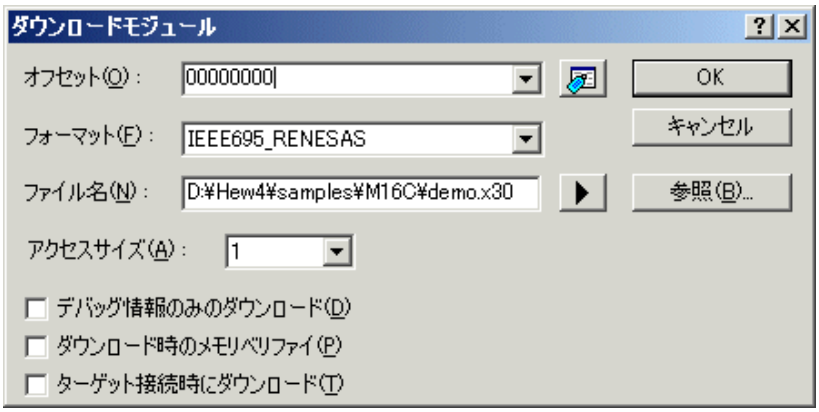


1. [ターゲット]ドロップダウンリストボックスで接続したい製品名を選択してください。
2. [デバッグ対象フォーマット] ドロップダウンリストボックスで、ダウンロードするロードモジュールの形式を選択してください。

フォーマット名	種類
IEEE695_RENESAS	IEEE-695 フォーマットファイル（弊社製ツールチェインで生成）
IEEE695_IAR	IEEE-695 フォーマットファイル（IAR 社製ツールチェインで生成）
IEEE695_TASKING	IEEE-695 フォーマットファイル（TASKING 社製ツールチェインで生成）
ELF/DWARF2	ELF/DWARF2 フォーマットファイル（弊社製ツールチェインで生成）
ELF/DWARF2_IAR	ELF/DWARF2 フォーマットファイル（IAR 社製ツールチェインで生成）
ELF/DWARF2_TASKING	ELF/DWARF2 フォーマットファイル（TASKING 社製ツールチェインで生成）
ELF/DWARF2_KPIT	ELF/DWARF2 フォーマットファイル（KPIT 社製ツールチェインで生成）
Intel-Hex+Sym	IntelHex フォーマットファイルとシンボルファイル（弊社製ツールチェイン SRA74 で生成）
IEEE695_ICC740	IEEE-695 フォーマットファイル（弊社製ツールチェイン ICC740 で生成）

なお、ドロップダウンリストに表示されないオブジェクトフォーマットには対応していません。

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。



- [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。
- ターゲットに接続したときすぐにプログラムをダウンロードする場合、[ターゲット接続時にダウンロード]をチェックしてください。

設定完了後、[OK]ボタンを押してください。

注意事項

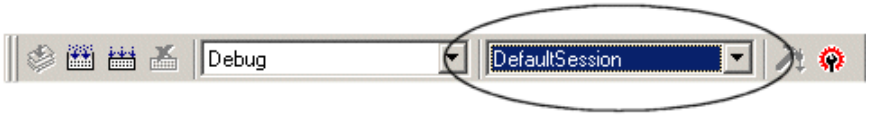
[オフセット]、[アクセスサイズ] および [ダウンロード時のメモリベリファイ] はサポートしていません。
常にオフセット 0、アクセスサイズは 1、メモリベリファイなしとなります。

3.3 デバッガの起動

シミュレータ に接続することで、デバッグを開始できます。

3.3.1 シミュレータの接続

シミュレータ 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、シミュレータ 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。
表示されない場合は、[デバッグ->接続]を選択します。

このセットアップが終了すると、接続は完了です。

3.3.2 シミュレータの終了

以下の方法があります。

1. “接続解除” を選択する。
   [デバッグ->接続解除] を選択してください。

2. セッションを"DefaultSession" に切り替える
   シミュレータ 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。

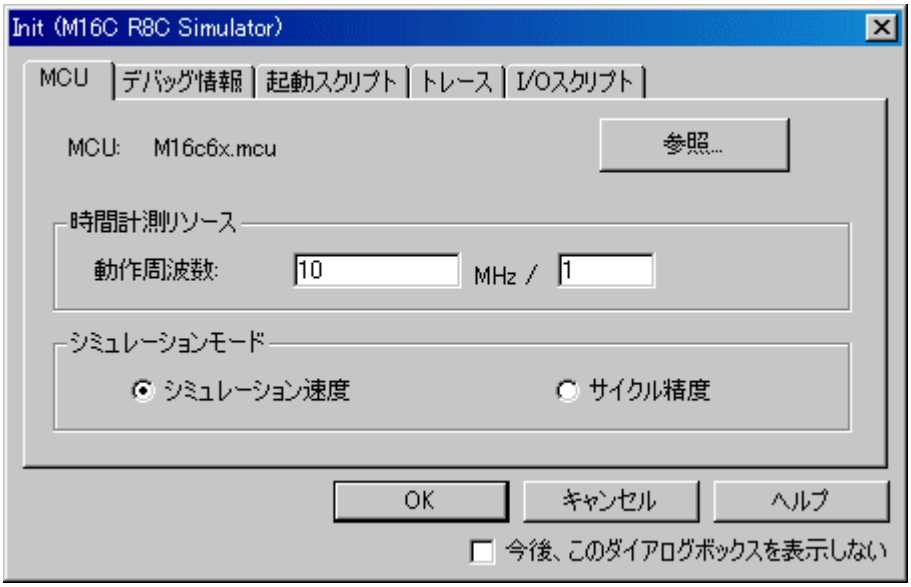
3. High-performance Embedded Workshop 自体を終了する
   [ファイル->アプリケーションの終了]を選択してください。High-performance Embedded Workshop は終了します。

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。

4. デバッガのセットアップ

4.1 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、次回起動時も有効となります。



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	製品名			
	R32C 用 デバッガ	M32C 用 デバッガ	M16C/R8C 用 デバッガ	740 用デバッガ
MCU	○	○	○	○
デバッグ情報	○	○	○	○
起動スクリプト	○	○	○	○
トレース	○	○	○	×
I/O スクリプト	○	○	○	×

なお、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]→[シミュレータ]→[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。

4.1.1 MCU タブ

指定した内容は、次回起動時も有効となります。

4.1.1.1 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。

対応する MCU ファイルがデバッガに含まれていない場合､MCU ファイルを新規に作成していただく必要があります。
詳細は「4.4 MCUファイルの作成」を参照ください。

4.1.1.2 時間計測リソースの指定

ターゲット MCU の動作クロックを指定します。

MCU クロックとクロック分周比を、それぞれ指定してください。
MCU を 10MHz・4 分周で使用する場合、左側に"10"、右側に"4"を指定します。
クロック分周比を指定する領域に値が設定されなかった場合は、 分周なし("1"を指定した場合と同じ)として動作します。

この設定は、デバッガが次回接続される時に有効になります。

4.1.1.3 シミュレーションモードの選択（M16C/R8C 用）

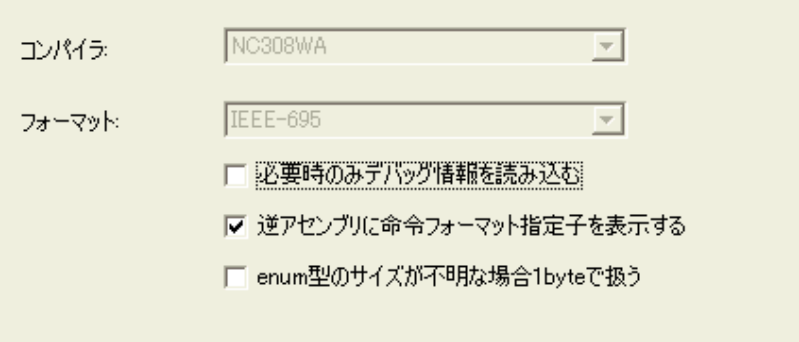
シミュレーションモードを選択します。

- シミュレーション速度モード
  シミュレーション速度モードで計測するサイクル数は、「バス幅、命令キューバッファ、メモリウェイト数」を考慮したサイクル数の計測は行いません。サイクル数は、マイクロコンピュータのソフトウェアウェアマニュアルに記載されている値を使用しています。
- サイクル精度モード
  サイクル精度モードで計測するサイクル数は、「バス幅、命令キューバッファ、メモリウェイト数」を考慮したサイクル数を計測します。そのため、実チップに近いサイクル数を計測できます。
  ただし、サイクル精度モードは、シミュレーション速度モードに比べ、シミュレーション速度は遅くなります。

この指定は、デバッガ起動時のみ設定が可能です。

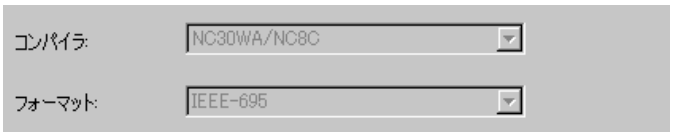
4.1.2 デバッグ情報 タブ

指定した内容は、次回起動時も有効となります。



4.1.2.1 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。

4.1.2.2 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。
指定した内容は、次回ダウンロード時から有効です。

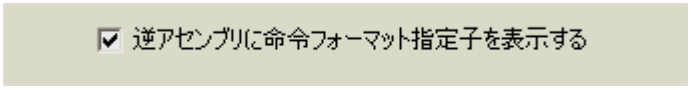
- オンメモリ方式
  デバッグ情報をパーソナルコンピュータのメモリ上に保持します。
  通常はこちらを選択します。
- オンデマンド方式
  デバッグ情報を再利用可能なテンポラリファイル上に保持します。
  同一ロードモジュールに対する二度目以降のダウンロードでは、 保持されたデバッグ情報を再利用するため、高速にダウンロード可能です。
  PC に搭載されているメモリ量がロードモジュールの規模に対して少ないなどの理由でデバッグ情報のダウンロードに時間がかかる場合に適します。

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。

4.1.2.3 命令フォーマット指定子の表示/非表示

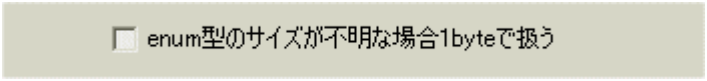
逆アセンブル表示で、命令フォーマット指定子を表示するかどうかを指定します。
740 用デバッガでは、サポートしていません。



命令フォーマット指定子を表示する場合は、上記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。

4.1.2.4 不明サイズの列挙型のサイズ指定

デバッグ情報中にサイズ情報を持たない列挙型の情報があった場合、その列挙型のサイズを常に 1byte で扱うかを指定できます。 NC30、NC308、および NC100 には、列挙型を標準の int と同サイズではなく、1byte にしてメモリ効率を上げるオプションがあります。また、NC30、NC308、および NC100 では、デバッグ情報に列挙型のサイズ情報が出力されないため、デバッガは常に int と同じサイズで扱います。このため、列挙型を 1byte にするオプションを指定したターゲットプログラムでは、列挙型変数の値を正しく表示することが出来ない場合がありました。このオプションはその問題を解決します。
列挙型のサイズを 1byte にするオプションの詳細については、それぞれのコンパイラ製品のマニュアルをご参照ください。
740 用デバッガでは、サポートしていません。



サイズ情報のない列挙型のサイズを常に 1byte で扱いたい場合は、上記チェックボックスをチェックしてください。
この設定を反映するには、ターゲットプログラムの再ダウンロードが必要です。

4.1.3 起動スクリプト タブ

指定した内容は、起動時のみ反映されます。起動後に Init ダイアログで再設定した内容は、有効になりません。

4.1.3.1 スクリプトコマンドの自動実行

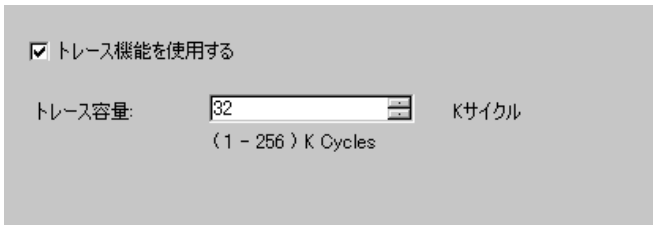
デバッガ起動時にスクリプトコマンドを自動実行するには、"参照..."ボタンをクリックし、 実行するスクリプトファイルを指定してください。

"参照..."ボタンをクリックすることにより、ファイルセレクションダイアログがオープンします。
指定されたスクリプトファイルは、ファイル名:領域に表示されます。
スクリプトコマンドを自動実行しないようにするには、ファイル名:領域に表示された文字列を消去してください。

4.1.4 トレース タブ

トレース計測の有効／無効、および有効時のトレースバッファのサイズを指定します。
このタブは、740 用デバッガではありません。

指定した内容は、次回起動時も有効となります。

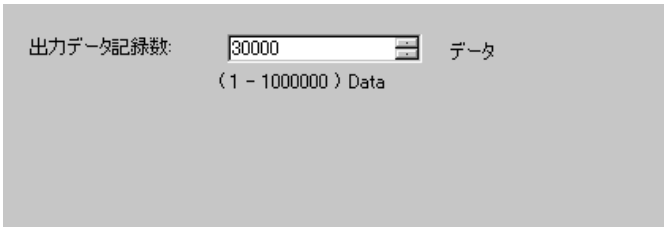


トレース計測を有効にするには、[トレース機能を使用する]チェックボックスをチェックしてください。
トレース計測を無効にした場合、トレースウィンドウ、トレースポイント設定ウィンドウはオープンできません。
[トレース容量]領域でトレースデータを格納するバッファのサイズを指定してください(K サイクル単位)。

4.1.5 I/O スクリプト タブ

I/O ウィンドウ、出力ポートウィンドウのポート出力機能で記録する、データの個数を指定します。
このタブは、740 用デバッガではありません。740 用デバッガの場合、データ個数は 30000 個に固定されています。

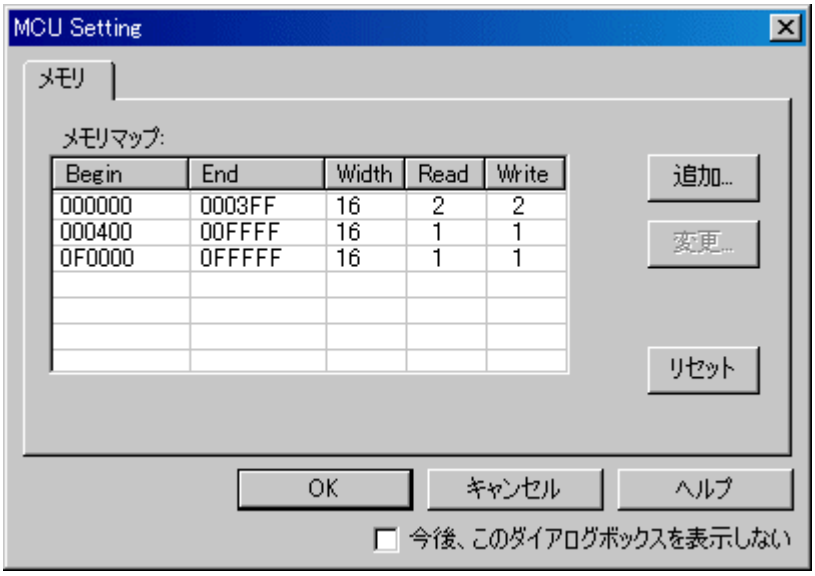
指定した内容は、次回起動時も有効となります。



[出力データ記録数]領域に、出力データを記録する個数を指定してください。

4.2 MCU Setting ダイアログ(M16C/R8C 用)

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 ダイアログは、以下の方法で再表示できます。

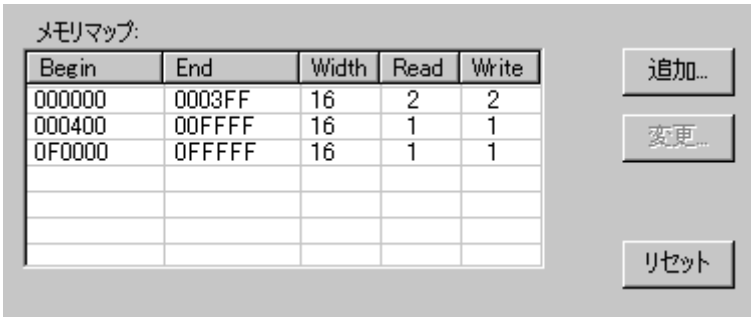
- デバッガ起動後、メニュー[基本設定]→[シミュレータ]→[ターゲット...]を選択する。

注意事項

MCU Setting ダイアログは、Init ダイアログの MCU タブでサイクル精度モードを選択した時のみオープンします。

4.2.1 メモリタブ

指定した内容は、次回起動時も有効となります。



4.2.1.1 メモリマップの設定

マップを設定するメモリ領域を 256 バイト単位で設定します。シミュレータデバッガは、これらの値をサイクル数の算出に使用します。
なお、アドレスが最下位のマップ（00000h～ 0FFFFh）と、アドレスが最上位のマップ（F0000h ～ FFFFFh）はシミュレータデバッガ起動時に既に割り付けられています。

注意事項

一度マップ設定して割り付けたメモリ空間は、削除することはできません。

メモリマップは、追加・変更ができます。
追加ボタンをクリックすると、メモリマップ設定ダイアログが開き、追加することができます。
変更したい項目をリストボックス上で選択後、変更ボタンをクリックすると、メモリマップ設定ダイアログが開き、変更することができます。
なお、リセットボタンによりデフォルト値にリセットすることができます。リセットボタンは、デバッガ起動後、メニュー[基本設定]→[シミュレータ]→[ターゲット...]を選択して MCU Setting ダイアログを開いた場合は、クリックできません。

変更内容は、OK ボタンをクリックすることにより設定します。
キャンセルボタンをクリックすると、設定しないでダイアログを閉じます。

4.2.1.1.1. メモリマップ設定ダイアログ

メモリマップ設定ダイアログでは、メモリマップを設定します。
シミュレータデバッガは、これらの値をサイクル数の算出に使用します。

本ダイアログでは以下の項目を設定します。

項目	設定内容
開始アドレス	メモリの開始アドレス
終了アドレス	メモリの終了アドレス
バス幅	メモリのバス幅（8 / 16 ビット）
リードサイクル数	メモリのリードサイクル数（1 ～ 255）
ライトサイクル数	メモリのライトサイクル数（1 ～ 255）

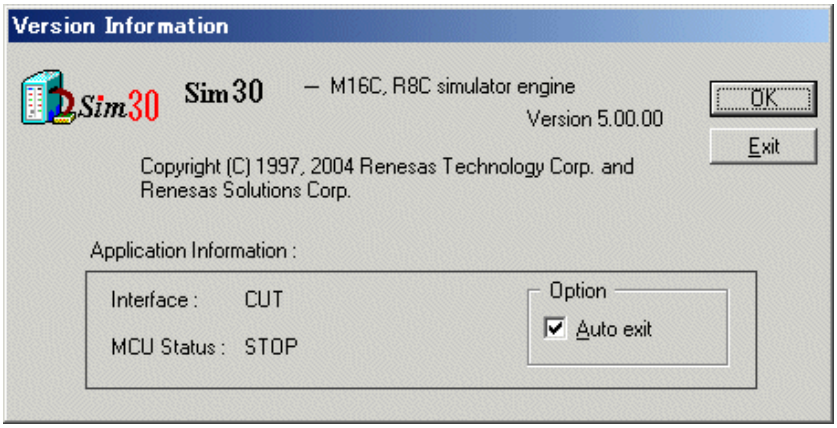
変更内容は、OK ボタンをクリックすることにより設定します。
キャンセルボタンをクリックすると、設定しないでダイアログを閉じます。

4.3 シミュレータエンジンのセットアップ

シミュレータエンジン simxx は、シミュレータデバッガを起動すると、システムトレイに登録されます。

起動中の simxx のアイコンを右クリックしてメニュー[Version...]を選択すると、Version Information ダイアログがオープンします。 simxx のセットアップは、この Version Information ダイアログで設定します。

※以下の図は、M16C/R8C 用シミュレータデバッガ対応シミュレータエンジンのものです。



- 自動終了スイッチの設定
  Auto exit チェックボックスをチェックすることにより、シミュレータデバッガ（フロントエンド）の終了と同時に simxx を終了させることができます。
- 通信接続状況
  シミュレータデバッガとの接続時には CONNECT を、切断時には CUT を表示します。
- シミュレータ MCU 実行状況
  シミュレータ MCU の実行時には RUN を、停止時には STOP を表示します。
- OK ボタン
  Version Information ダイアログをクローズします。
- Exit ボタン
  simxx を終了します。

4.4 MCU ファイルの作成

4.4.1 MCU ファイルの作成(R32C 用デバッガ)

MCU ファイルには、以下の内容を順番に記述します。
ファイル名は、MCU 名を指定してください。また、拡張子は、"mcu"と指定してください。
1. UART0 送受信制御レジスタ 1 アドレス
2. UART1 送受信制御レジスタ 1 アドレス
3. UART0 送信バッファレジスタアドレス
4. UART1 送信バッファレジスタアドレス

各アドレスは 16 進数で記述してください。また、基数を示すプレフィックスは付けないでください。

4.4.1.1 記述例

以下に例を示します。

```
365
36D
362
36A
```

4.4.2 MCU ファイルの作成(M32C 用デバッガ)

MCU ファイルには、以下の内容を順番に記述します。
ファイル名は、MCU 名を指定してください。また、拡張子は、"mcu"と指定してください。

1. MCU タイプ（”0” or “1”）
2. UART0 送受信制御レジスタ 1 アドレス
3. UART1 送受信制御レジスタ 1 アドレス
4. UART0 送信バッファレジスタアドレス
5. UART1 送信バッファレジスタアドレス

MCU タイプは、M32C 用デバッガのときのみ指定する必要があります。

“0” … M16C/8x を選択
“1” … M32C/8x を選択

各アドレスは 16 進数で記述してください。また、基数を示すプレフィックスは付けないでください。

4.4.2.1 記述例

以下に例を示します。

```
0
3A5
3AD
3A2
3AA
```

4.4.3 MCU ファイルの作成(M16C/R8C 用デバッガ)

MCU ファイルには、以下の内容を順番に記述します。
ファイル名は、MCU 名を指定してください。また、拡張子は、"mcu"と指定してください。

1. MCU タイプ
2. UART0 送受信制御レジスタ 1 アドレス
3. UART1 送受信制御レジスタ 1 アドレス
4. UART0 送信バッファレジスタアドレス
5. UART1 送信バッファレジスタアドレス
6. リセットベクタアドレス
7. 未定義命令割り込みベクタアドレス
8. オーバフロー割り込みベクタアドレス
9. BRK 命令割り込みベクタアドレス

MCU タイプの先頭には、必ず';' (セミコロン)を付けて下さい。

- ";8" … R8C ファミリ
- ";16" … M16C シリーズ

各アドレスは 16 進数で記述してください。また、基数を示すプレフィックスは付けないでください。

4.4.3.1 記述例

以下に例を示します。

```
;16
3A5
3AD
3A2
3AA
FFFFC
FFFDC
FFFE0
FFFE4
```

4.4.4 MCU ファイルの作成(740 用デバッガ)

MCU ファイルには、以下の内容を順番に記述します。 ファイル名は、MCU 名を指定してください("m3xxxx.mcu")。拡張子は、"mcu"と指定してください。 各アドレスは 16 進数で記述してください。また、基数を示すプレフィックスは付けないでください。
3～6 の情報は、ご使用になられる MCU のデータブック等をご参照の上、記述してください。

1. MCU 名
2. 予約番号
3. CPU モードレジスタアドレスとスタックページ選択ビット番号
4. リセットベクタアドレス
5. BRK ベクタアドレス
6. 割り込みベクタ情報

- MCU 名、予約番号
  先頭には、必ず';' (セミコロン)を付けて下さい。

- CPU モードレジスタのアドレスとスタックページ選択ビット番号
  CPU モードレジスタのアドレスとスタックページ選択ビット番号は':' (コロン)で区切って下さい。

- リセットベクタアドレス
  アドレスの後ろに":RST"を付けて下さい。

- BRK ベクタアドレス
  アドレスの後ろに":BRK"を付けて下さい。

- 割り込みベクタ情報
  割り込みベクタアドレス、対応する割り込み制御レジスタアドレス、割り込み制御ビット番号の情報を記述します。 各情報は、':' (コロン)で区切って下さい。割り込みベクタ情報は最大 32 点まで記述可能です。

4.4.4.1 記述例

以下に例(m38000.mcu)を示します。

```
;M38000
;1
3B:2
FFFC:RST
FFDC:BRK
FFFA:3E:0
FFF8:3E:1
FFF6:3E:2
FFF4:3E:3
FFF2:3E:4
FFF0:3E:5
FFEE:3E:6
FFEC:3E:7
FFEA:3F:0
FFE8:3F:1
FFE6:3F:2
FFE4:3F:3
FFE2:3F:4
FFE0:3F:5
```

チュートリアル編

このページは白紙です

5. チュートリアル

5.1 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。チュートリアルプログラムは High-performance Embedded Workshop をインストールしたドライブの¥WorkSpace¥Tutorial にターゲットごと、MCU ごとに格納されています。ご使用のシステムに対応したチュートリアルのワークスペースファイル（*.hws）を［ワークスペースを開く］から開いてください。
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。

740 用デバッガで 740 ファミリ用アセンブラパッケージを使用する場合

740 ファミリ用アセンブラパッケージを使用したアセンブリ言語用の チュートリアルプログラムを用意しています。
740 ファミリ用アセンブラパッケージをご使用の場合は、こちらのチュートリアルプログラムをご使用ください。

- チュートリアル中の関数名の記述(例:"sort 関数")は、サブルーチン名に適宜置き換えてください(例:"サブルーチン sort")。
- チュートリアル中のソースファイル名の記述(例:"Tutorial.c")は、適宜置き換えてください。
- チュートリアル中の図は、C 言語プログラム用です。アセンブリ言語用のチュートリアルプログラムでは、 表示が異なる場合があります。
- 「Step9:変数の参照」と「Step12: ローカル変数の参照」は、C 言語プログラム専用の機能です。

5.2 使用方法

以下のステップに沿ってお進みください。

5.2.1 Step1： デバッガの起動

5.2.1.1 デバッグの準備

High-performance Embedded Workshopを起動し、シミュレータ に接続します。
詳細は「3 デバッグの準備」を参照ください。

5.2.1.2 デバッガのセットアップ

シミュレータ 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「4 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。

5.2.2 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

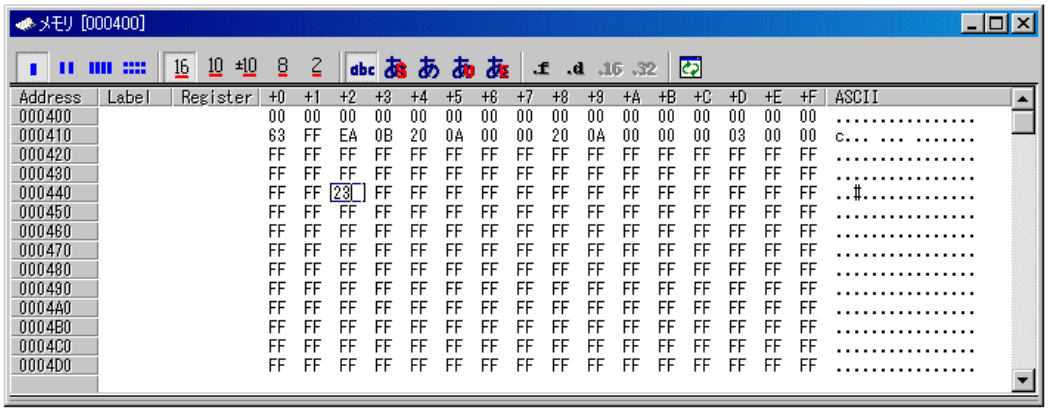
マイコンによってはボード上にメモリをつけることができます。 この場合、メモリ動作チェックは上記だけでは不完全な場合があります。 メモリチェック用プログラムを作成し、チェックすることをお勧めします。

5.2.2.1 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"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。



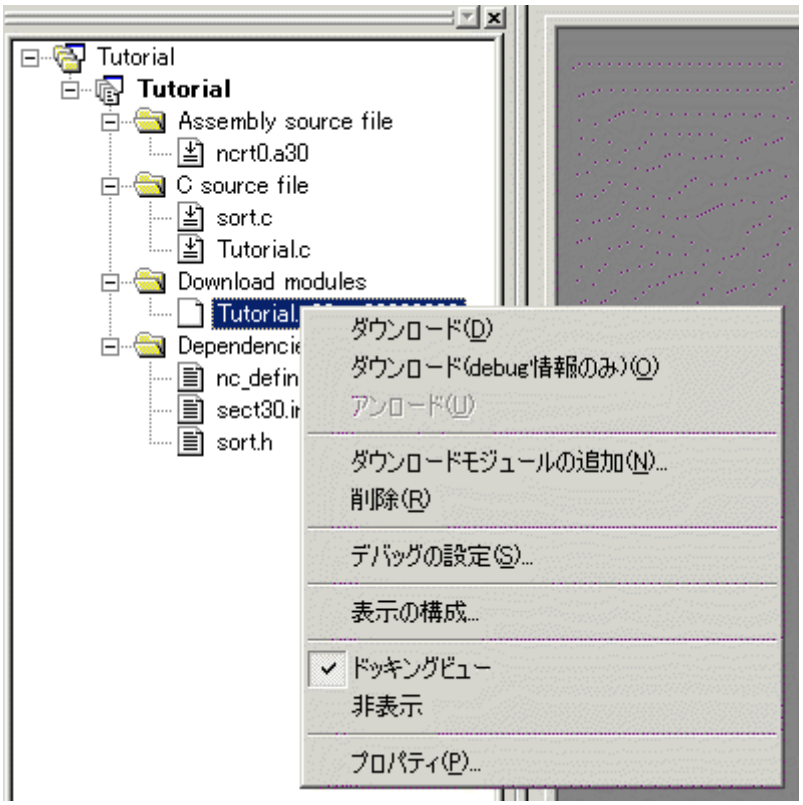
[メモリ]ウィンドウ上のデータ部分をダブルクリックすることにより、値が変更できます。

5.2.3 Step3： チュートリアルプログラムのダウンロード

5.2.3.1 チュートリアルプログラムをダウンロードする

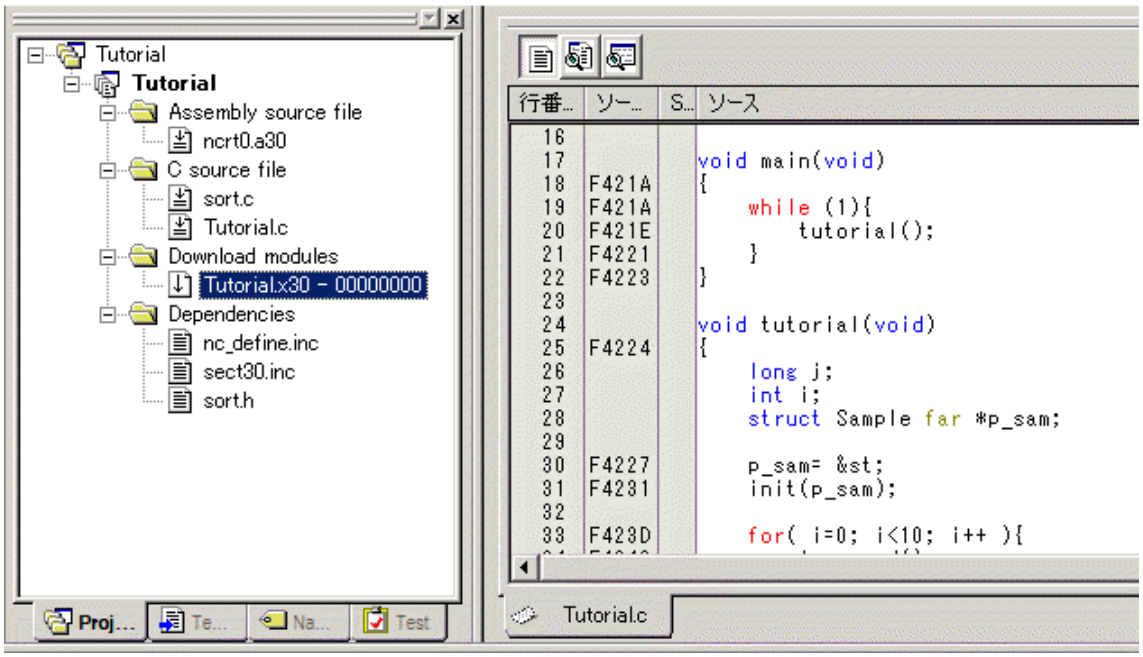
デバッグしたいオブジェクトプログラムをダウンロードします。なお、ダウンロードするプログラムやダウンロード先のアドレスは使用しているマイコンによって異なります。画面の表示などはご使用のマイコンに合わせて適宜読み替えてください。

- M16C/R8C、M32C、または、R32C 用デバッガの場合
  [Download modules] の [Tutorial.x30] から [ダウンロード] を選択します。
- 740 用デバッガの場合
  740 ファミリ用 C コンパイラパッケージをご使用の場合は [Download modules] の [Tutorial.695] から [ダウンロード] を選択します。
  740 ファミリ用アセンブラパッケージをご使用の場合は [Download modules] の [Tutorial.hex] から [ダウンロード] を選択します。



5.2.3.2 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。



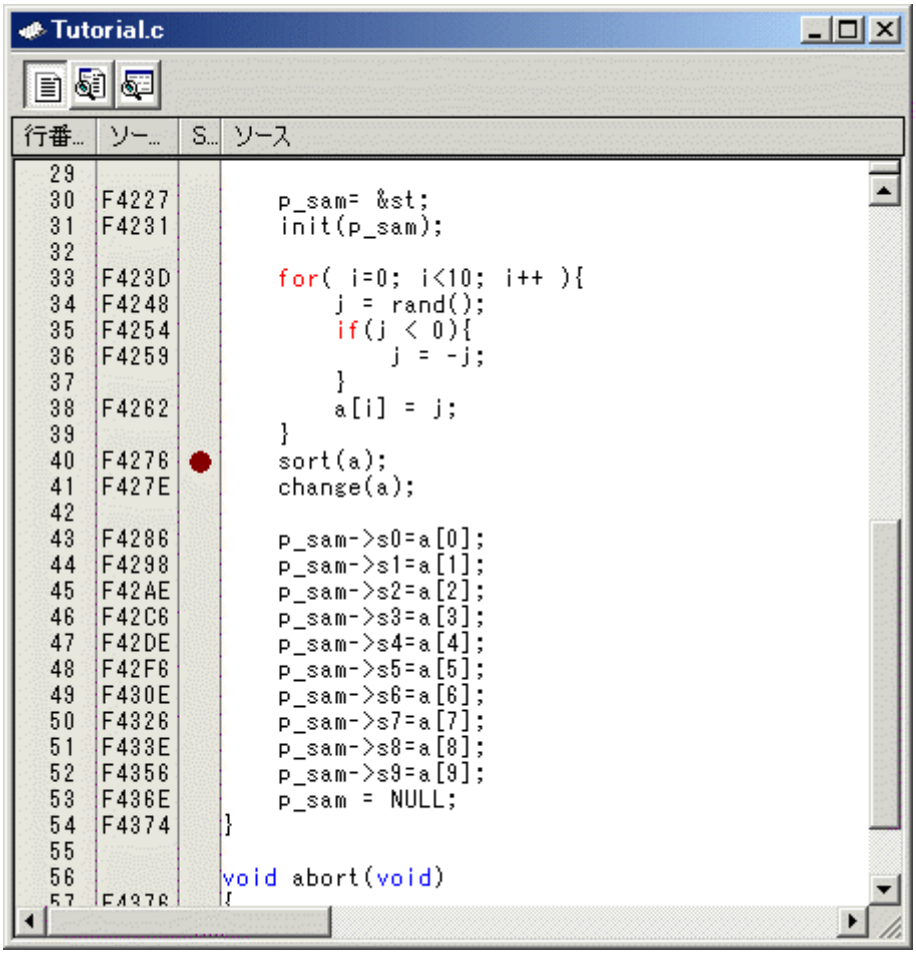
必要であれば、[基本設定]メニューから[表示の形式]オプションを選択し、見やすいフォントとサイズを選択してください。
[エディタ(ソース)]ウィンドウは、最初はプログラムの先頭を示しますが、スクロールバーを使って他の部分を見ることができます。

5.2.4 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。

5.2.4.1 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。



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。

5.2.5 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。

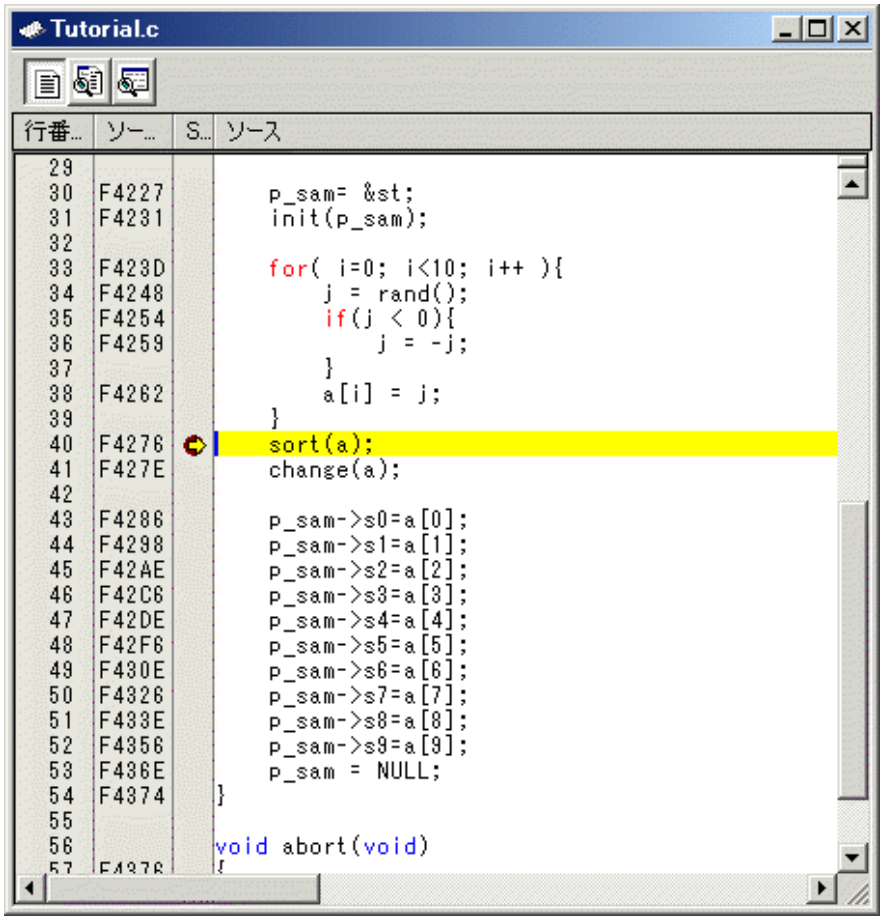
5.2.5.1 CPU のリセット

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。

5.2.5.2 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。

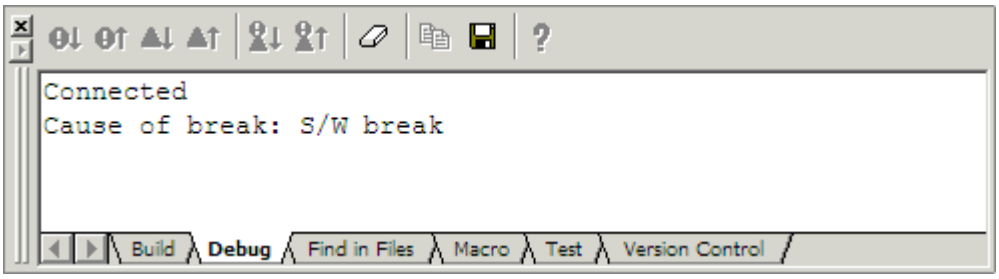


注意事項

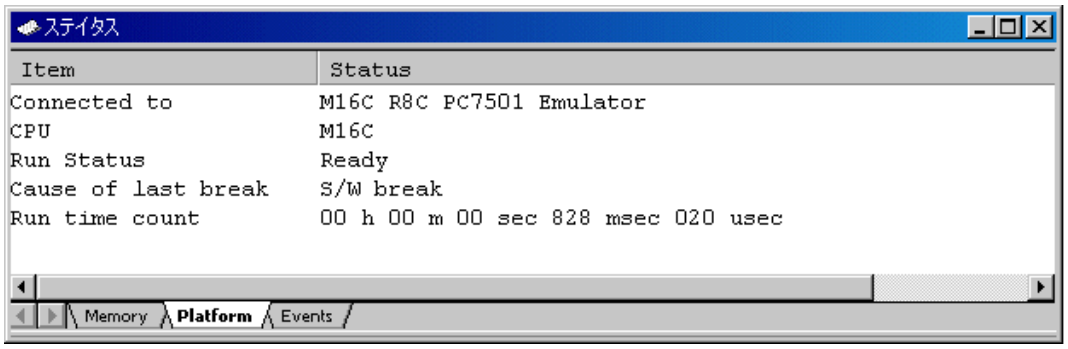
ブレーク後にソースファイルを表示する際に、ソースファイルパスを問い合わせる場合があります。 その場合は、ソースファイルの場所を指定してください。

5.2.5.3 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。
740 用デバッガでは、ブレーク要因を表示する機能はサポートしていません。



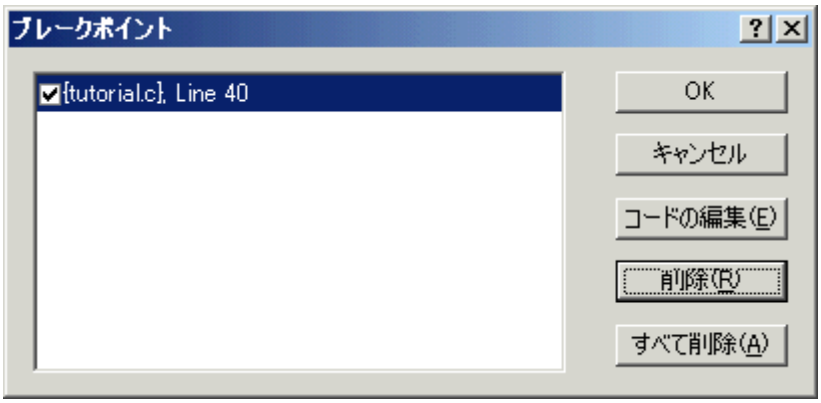
ブレーク要因の表記については、「11 プログラム停止要因の表示」を参照ください。

5.2.6 Step6： ブレークポイントの確認

設定した全てのソフトウェアブレークポイントは、[ブレークポイント]ダイアログボックスで確認できます。

5.2.6.1 ブレークポイントを確認する

キーボードで Ctrl+B を押してください。 [ブレークポイント]ダイアログボックスが表示されます。



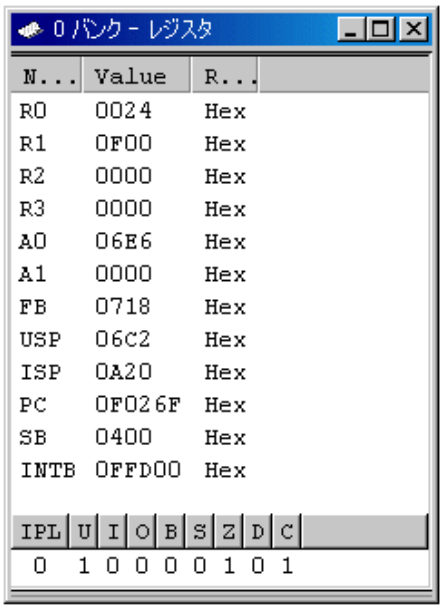
このダイアログボックスを使って、ブレークポイントの削除や有効/無効の選択ができます。

5.2.7 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。

5.2.7.1 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。
以下の図は、M16C/R8C 用デバッガの表示です。



5.2.7.2 レジスタ内容を変更する

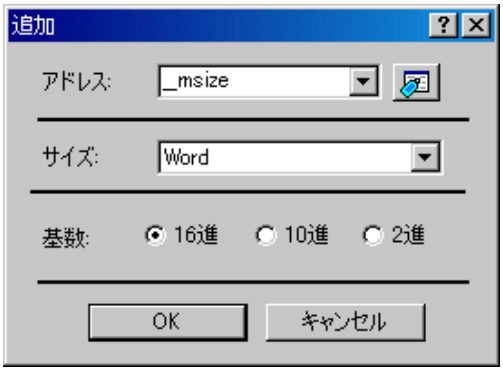
任意のレジスタの内容を変更することができます。
変更するレジスタ行をダブルクリックして下さい。ダイアログが表示しますので、変更する値を入力ください。

5.2.8 Step8： メモリ内容の確認

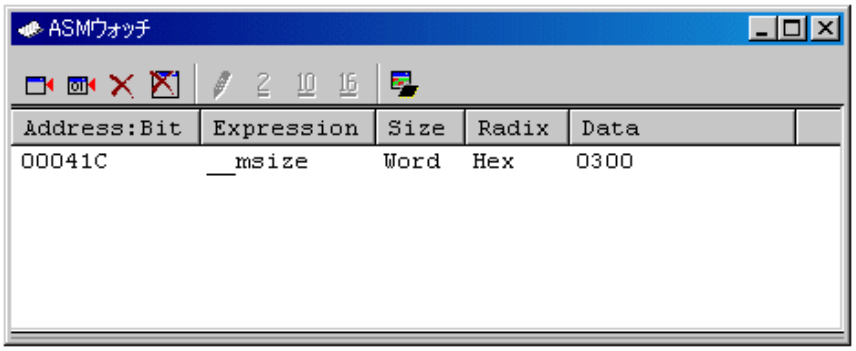
ラベル名を指定することによって、ラベルが登録されているメモリの内容を[ASM ウォッチ]ウィンドウで確認することができます。

5.2.8.1 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。

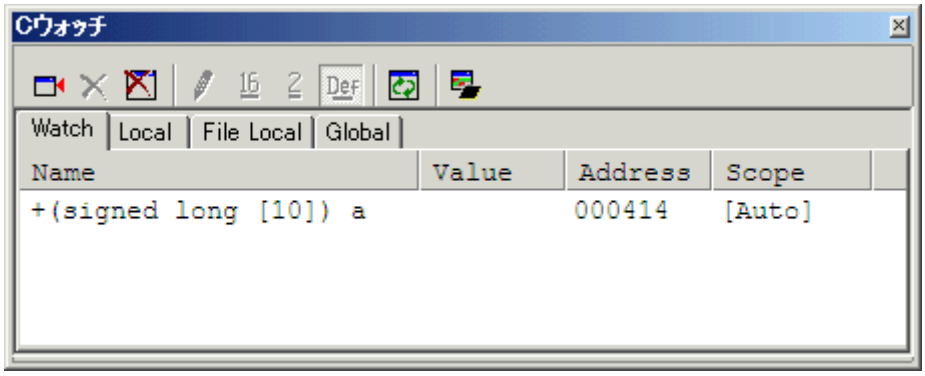


5.2.9 Step9： 変数の参照

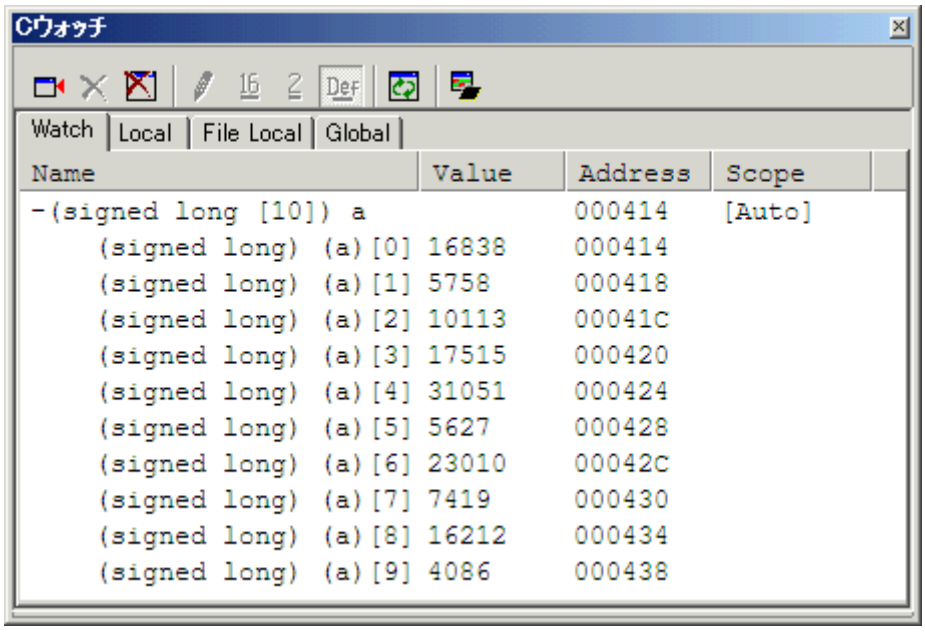
プログラムをステップ処理するとき、プログラムで使われる変数の値が変化することを確認できます。
740用デバッガで740ファミリ用アセンブラパッケージ用のチュートリアルプログラムを ダウンロードされた場合は、変数を参照することができません。

5.2.9.1 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。



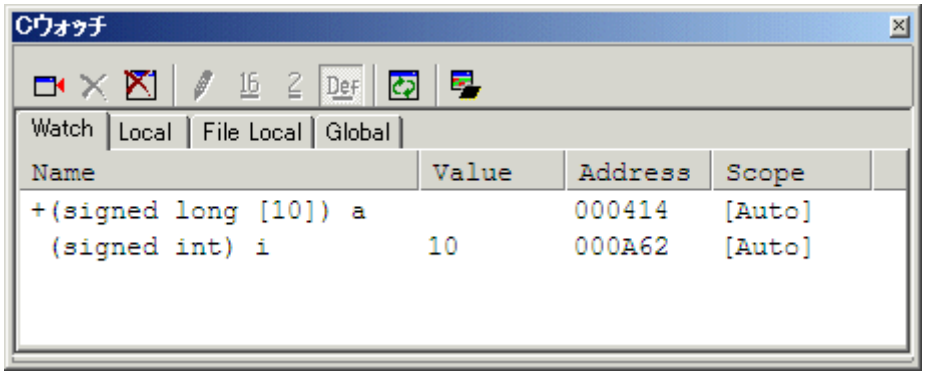
[C ウォッチ]ウィンドウの配列 a の左側にある"+"マークをクリックし、配列 a の各要素を参照することができます。



5.2.9.2 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。



5.2.10 Step10： プログラムのステップ実行

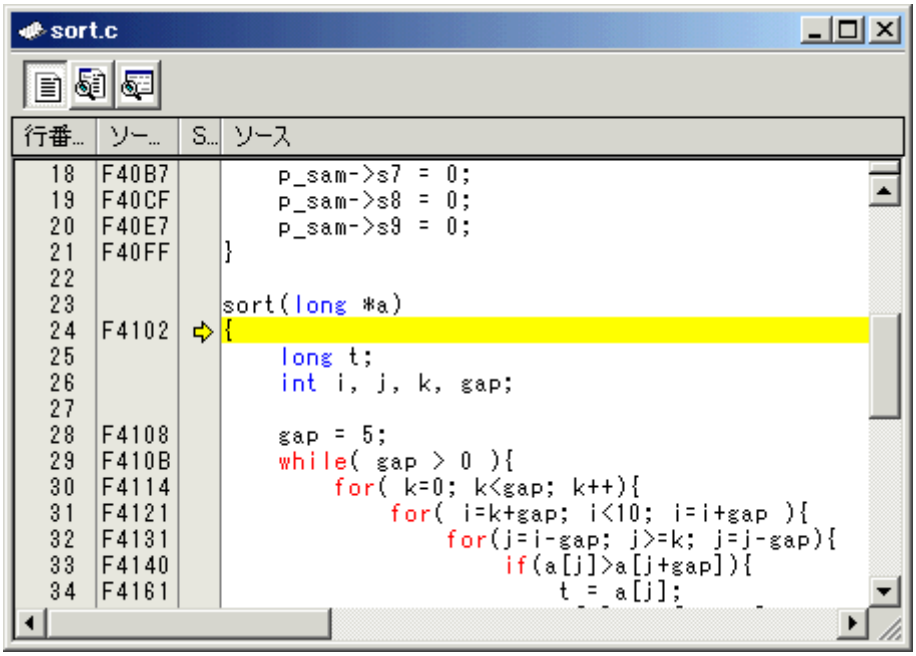
本デバッガは、プログラムのデバッグに有効な各種のステップコマンドを備えています。

1. ステップイン
   各ステートメントを実行します（関数(サブルーチン)内のステートメントを含む）。
2. ステップアウト
   関数(サブルーチン)を抜け出し、関数(サブルーチン)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(サブルーチン)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。

5.2.10.1 ステップインの実行

ステップイン機能はコール関数(サブルーチン)の中に入り、コール関数(サブルーチン)の先頭のステートメントで停止します

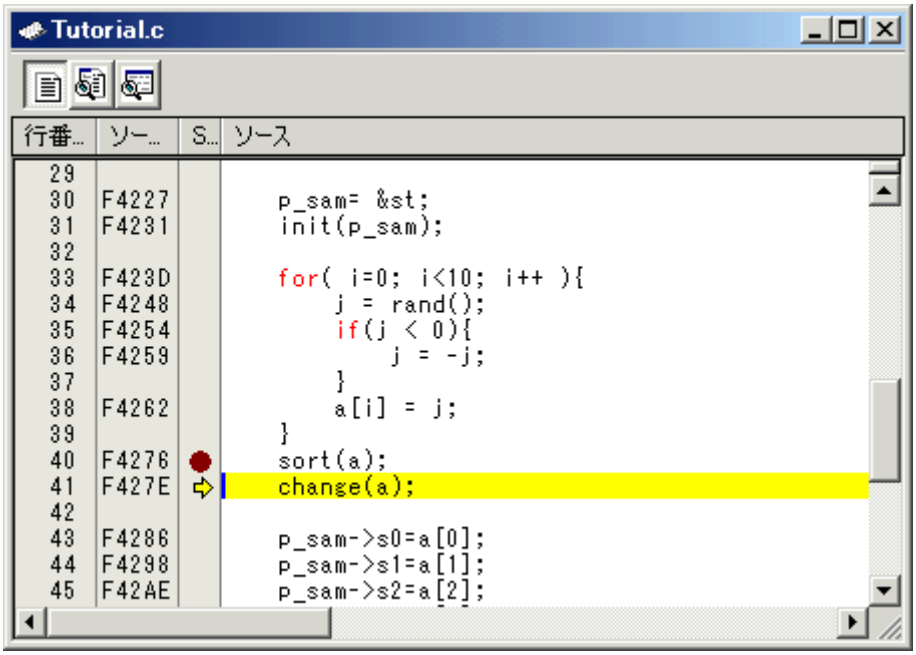
sort 関数の中に入るために、[デバッグ]メニューから[ステップイン]を選択するか、またはツールバーの[ステップイン]ボタン  をクリックしてください。
[エディット(ソース)]ウィンドウの PC 位置を示すカーソルが、sort 関数の先頭のステートメントに移動します。



5.2.10.2 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(サブルーチン)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。

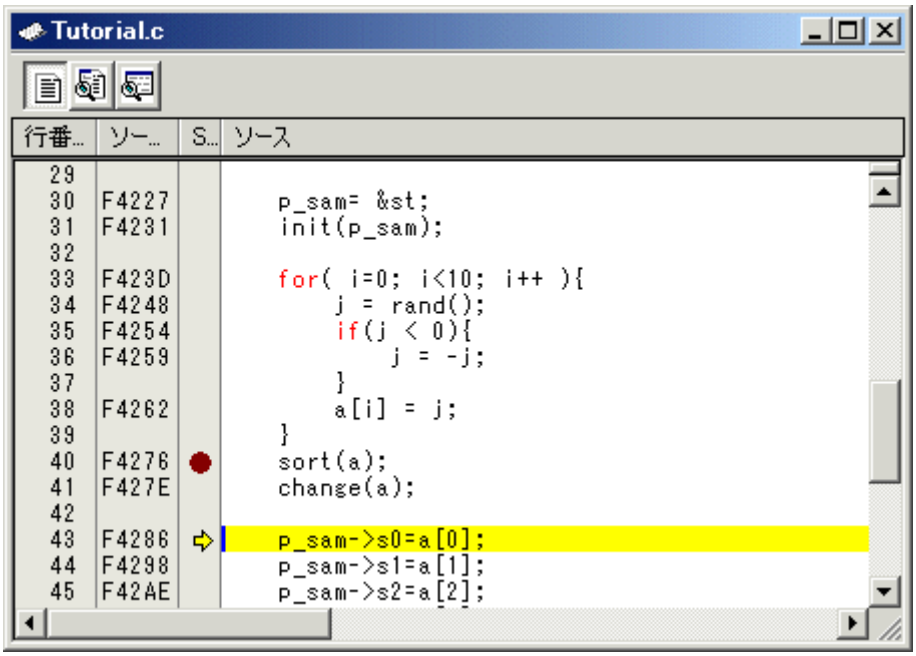


注意事項

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。

5.2.10.3 ステップオーバの実行

ステップオーバ機能は関数(サブルーチン)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。



5.2.11 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。

5.2.11.1 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。

5.2.12 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。
740 用デバッガで 740 ファミリ用アセンブラパッケージ用のチュートリアルプログラムを ダウンロードされた場合は、変数を参照することができません。

5.2.12.1 ローカル変数を表示する

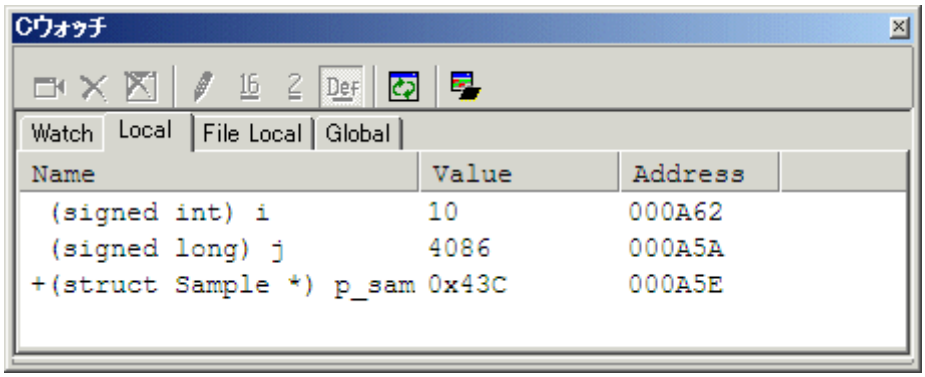
例として、tutorial 関数のローカル変数を調べます。

この関数は、3 つのローカル変数 i, j, p_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。



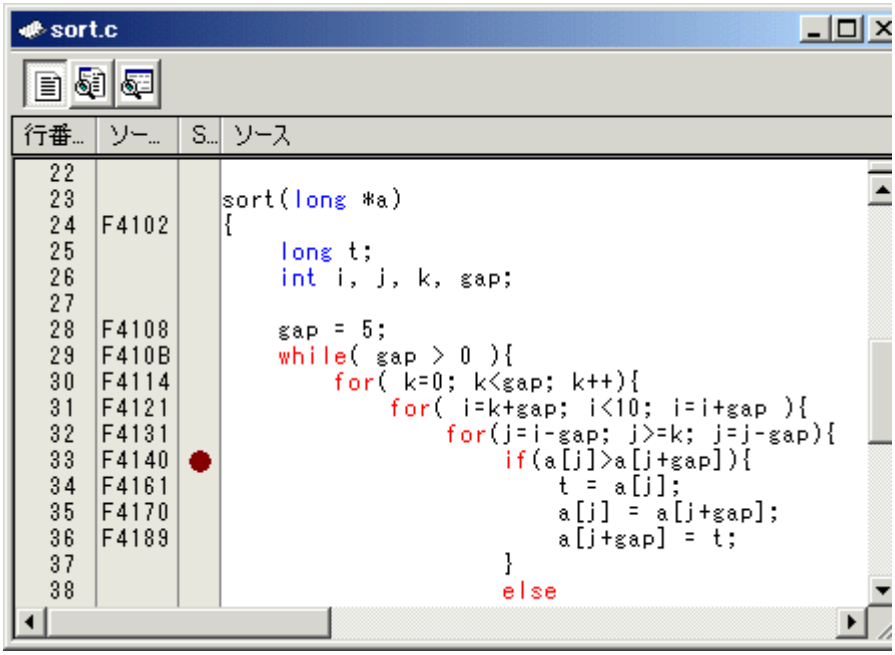
ポインタ変数 p_sam の左側にある"+"マークをダブルクリックし、構造体 *(p_sam) を表示させてください。
Tutorial 関数の最後で*(p_sam) の構造体メンバを参照すると、ランダムデータが降順にソートされていることがわかります。

5.2.13 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。
740 用デバッガでは、スタックトレースはサポートしていません。

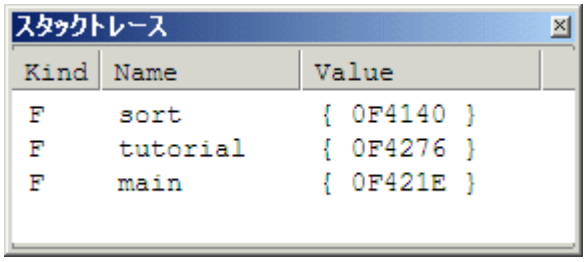
5.2.13.1 関数呼び出し状況を参照する

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。



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。

5.2.14 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。
ご使用のシミュレータで提供されるシ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。

6. 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]
I/O タイミング設定ウィンドウ	[表示]→[CPU]→[I/O タイミング設定]
出力ポートウィンドウ *	[表示]→[CPU]→[出力ポート]
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]
カバレッジウィンドウ	[表示]→[コード]→[カバレッジ]
スクリプトウィンドウ	[表示]→[スクリプト]
S/W ブレークポイント設定ウィンドウ	[表示]→[ブレーク]→[S/W ブレークポイント]
H/W ブレークポイント設定ダイアログ	[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]
データトレースウィンドウ *	[表示]→[トレース]→[データトレース]
GUI 入出力ウィンドウ	[表示]→[グラフィック]→[GUI I/O]
MR ウィンドウ *	[表示]→[RTOS]→[MR]
MRトレースウィンドウ #*	[表示]→[RTOS]→[MR トレース]
MRアナライズウィンドウ #*	[表示]→[RTOS]→[MR アナライズ]
タスクトレースウィンドウ #*	[表示]→[RTOS]→[タスクトレース]
タスクアナライズウィンドウ #*	[表示]→[RTOS]→[タスクアナライズ]

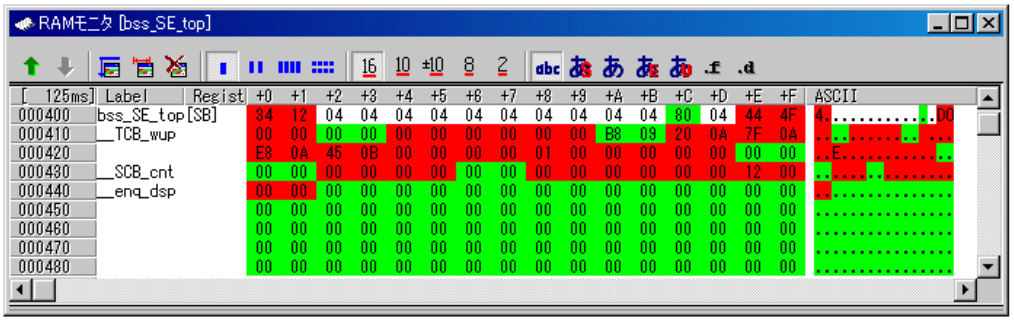
#: R32C用デバッガではサポートしていません。
*: 740 用デバッガではサポートしていません。

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
- ステイタスウィンドウ
- 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

6.1 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。



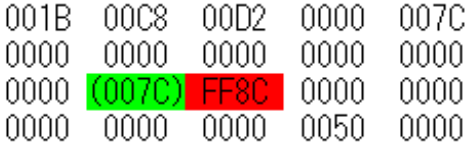
- 1K バイトの RAM モニタ領域を備えています。この RAM モニタ領域は、任意の連続アドレスに配置できます。
- RAMモニタ領域は、任意のアドレス範囲に変更できます。
  RAMモニタ領域の変更方法については、「6.1.2 RAMモニタ領域を設定する」を参照してください。
  デフォルトのRAMモニタ領域は､内部RAM領域の先頭から 1Kバイトの領域に割り当てられています。
- 表示内容の更新間隔はウィンドウごとに設定できます。
  ターゲットプログラム実行中の実際の更新間隔は、Address 表示領域のタイトル部分に表示されます。
- データ表示領域及びコード表示領域の背景色は、アクセス属性によって以下のようになります。

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。



- 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

6.1.1 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。

6.1.2 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。

以下のダイアログがオープンします。
[開始アドレス]領域には現在設定されている RAM モニタ領域の先頭アドレス、[RAM モニタ領域]領域には RAM モニタ領域の範囲が表示されています（[サイズ]領域は入力不可）。

このダイアログを使用して、RAM モニタ領域の位置を変更します。

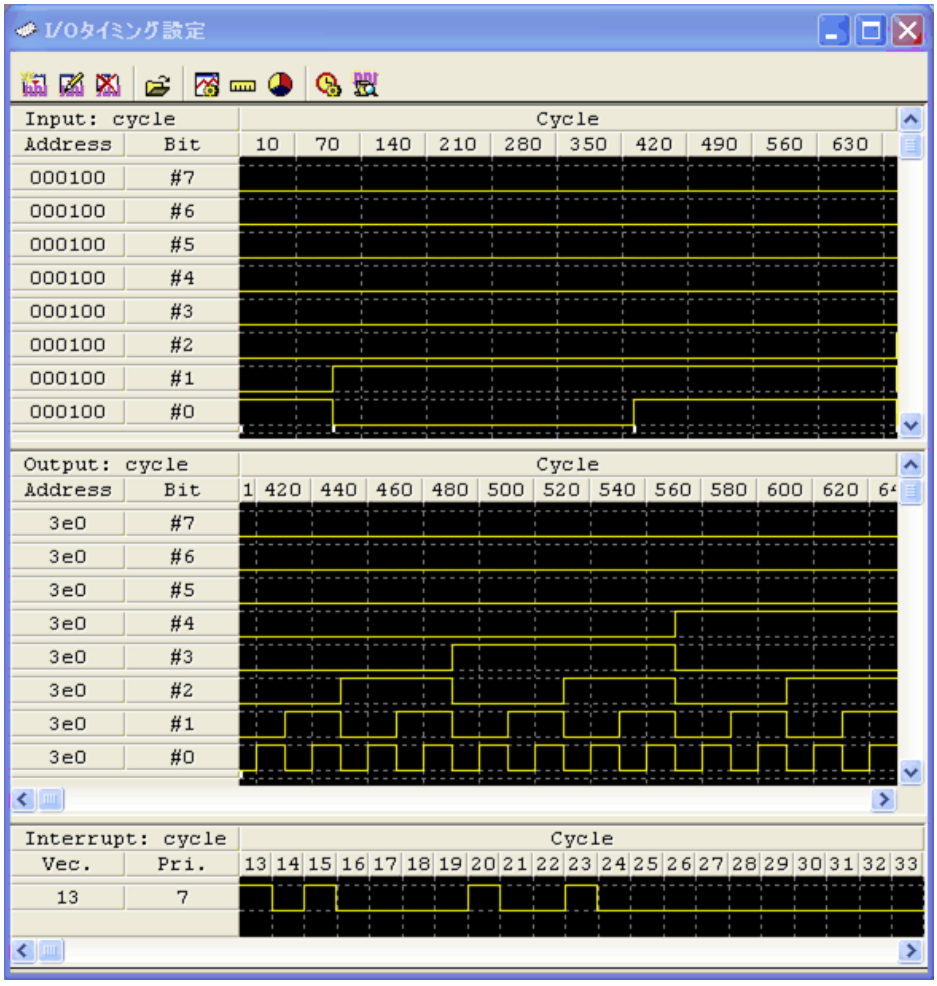
- RAM モニタ領域は、先頭アドレスで指定します。サイズは変更できません（1K バイト固定）。
- 先頭アドレスは、0x10 バイト単位で指定できます。
  端数のアドレス値を指定した場合は、0x10 バイト単位で丸め込まれた値が設定されます。

6.1.2.1 RAM モニタ領域を変更する

RAM モニタ領域の先頭アドレスを変更できます。
表示したダイアログの[開始アドレス]領域に、先頭アドレスを指定してください（[サイズ]領域は入力不可）。

6.2 I/O タイミング設定ウィンドウ

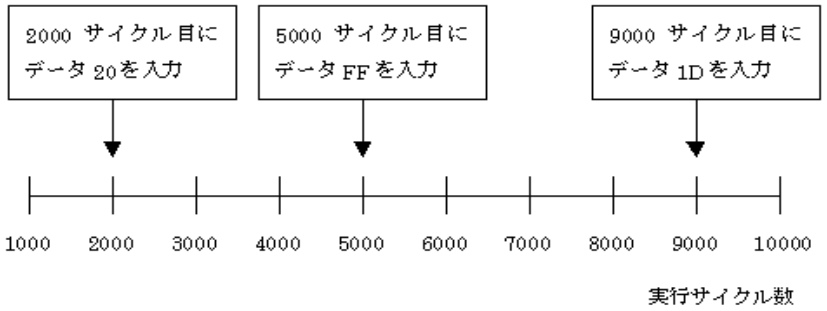
I/O タイミング設定ウィンドウは、仮想ポート入力や仮想ポート出力、仮想割り込みの設定、表示を行うウィンドウです。
仮想ポート入力、仮想割り込みの設定及び、仮想ポート出力の結果をチャート、数値、グラフ形式の表示で参照できます。
I/O タイミング設定ウィンドウは、仮想ポート入力の設定内容、仮想ポート出力の出力結果、及び仮想割り込みの設定内容をそれぞれ個別の画面で構成されており、 下図に示す 3 画面で構成されています。



6.2.1 仮想ポート入力

仮想ポート入力とは、外部から指定アドレスのメモリに入力されるデータの変化を定義する機能です。この機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。 定義した入力データは、チャート、16 進数値、グラフ形式で参照できます。 仮想ポート入力には、以下の 3 種類の入力があります。

1. サイクル同期入力
   プログラムの実行が指定サイクルになった時に、入力データをメモリに書き込むことができます。入力するデータのサイズは 1 バイトです。 サイクルに同期した仮想ポート入力の例を以下に示します。



上記のように、ユーザが指定した任意のサイクルにおいて 3E0 番地のメモリにデータを入力するように設定することができます。

2. リードアクセス同期入力
   指定されたメモリをプログラムがリードアクセスした時にデータを入力することができます。入力するデータのサイズは 1 バイトです。
   メモリのリードアクセスに同期した仮想ポート入力の例を以下に示します。
   下記のプログラム例は、ポート 0（3E0 番地）の読み出しを行う関数の例です。

```
#pragma  ADDRESS  port0 = 3e0H
char port0;

read_port()
{
        char key;

        key = port0;  /* ポート 0 からの入力 */

          :
          :
}
```

この関数では、変数 key にポート 0 の値を代入しようとしています。このような場合、ポート 0（3E0 番地）をプログラムがリードアクセスした時に、ポート 0 に対して値を入力すれば変数 key に値を代入することができます。
このような関数の処理を支援する機能として、指定したアドレスのメモリをリードした回数に応じて入力するデータを定義する機能（メモリのリードアクセスに同期した仮想ポート入力）を用意しています。この機能を利用すると、次のように 3E0 番地を 1 回目にリードした時に 0x10、2 回目にリードした時に 0x20 を 3E0 番地のメモリに入力する処理が行えます。

3E0 番地をリードした回数	3E0 番地に入力するデータ
1 回目	0x10
2 回目	0x20
3 回目 : :	0x30 : :

3. 割り込み同期入力
   仮想割り込みが発生した時に、指定されたメモリにデータを入力することができます。入力するデータのサイズは 1 バイトです。
   仮想割り込みに同期した仮想ポート入力の例を以下に示します。
   下記のプログラムの例は、ポート 1（3E1 番地）の読み出しを割り込みハンドラ（この場合は、タイマ割り込みの割り込みハンドラの例です。）ルーチンでおこなっている場合の例です。

```
#pragma  ADDRESS  port1 = 3e1H
char port1;

#pragma INTERRUPT        read_port

/* ポート 1 のポーリング用の割り込みハンドラ */
read_port()
{
          char key;

          key = port1;  /* ポート 1 からの入力 */

             :
             :
}
```

この割り込みハンドラルーチンでは、仮想割り込みが発生した時に変数 key にポート 1 の値を代入しようとしています。このような場合、仮想割り込みが発生（この場合は、タイマ割り込み）した時に、ポート 1 に対して値を入力すれば変数 key に値を代入することができます。
なお、タイマ割り込みは、別途仮想割り込みの機能を利用して発生させているものとします（後述の仮想割り込みの項を参照下さい）。
このような割り込みハンドラの処理を支援する機能として、仮想割り込みが発生した回数に応じて入力するデータを定義する機能（仮想割り込みに同期した仮想ポート入力）を用意しています。この機能を利用すると、次のように仮想割り込みが 1 回目に発生した時に 0xFF、2 回目に発生した時に 0xFE を 3E1 番地のメモリに入力する処理が行えます。

仮想割り込みが発生した回数	3E1 番地に入力するデータ
1 回目	0xFF
2 回目	0xFE
3 回目 : :	0xFD : :

6.2.2 仮想ポート出力

仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。 記録されたデータは、チャート形式、数値形式やグラフ形式で確認できます。 なお、記録されるデータの個数は、Init ダイアログの I/O スクリプトタブで指定できます。プログラム実行開始から、指定した個数までデータを記録できます。

例えば、下記のようなプログラムを実行してポート 0（3E0 番地）にデータを書き込んだ場合、

```
#pragma  ADDRESS  port0  = 3e0H
char port0;

out_port(char  data)
{
          port0 = data;  /* ポート 0 にデータを出力 */
            :
            :
}
```

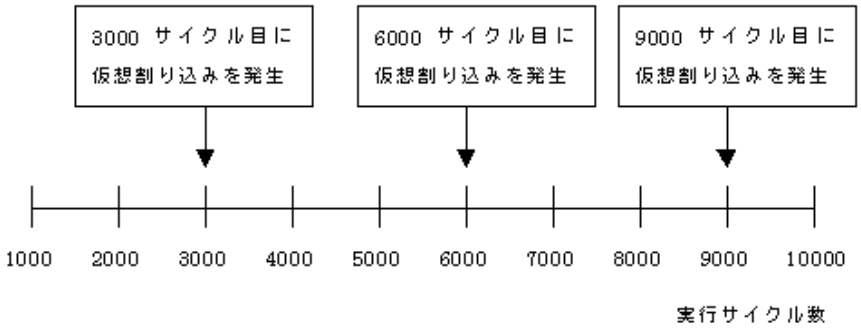
3E0 番地に書き込まれたデータとその時のサイクル数を記録します。

6.2.3 仮想割り込み

仮想割り込み機能とは、割り込みの発生を定義する機能です。この機能を利用すると、擬似的にタイマ割り込みやキー入力割り込み等を発生させることができます。
仮想割り込みには、以下の 3 種類があります。

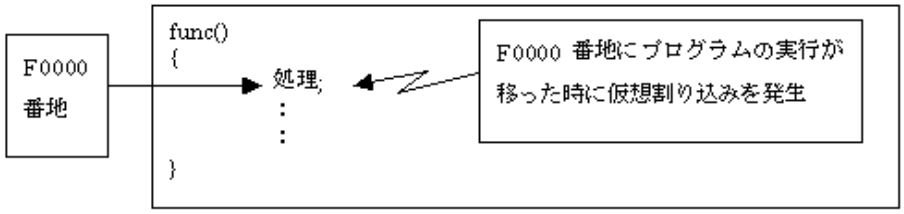
1. サイクル同期割り込み
   プログラムの実行が指定サイクルになった時に、指定した仮想割り込みを発生することができます。
   サイクルに同期した仮想割り込みの例を以下に示します。

「ソフトウェア割り込み番号 21 番（タイマ A0）の仮想割り込みの定義例」



上記のように、仮想割り込み（この場合タイマ A0 の割り込み）を任意のサイクルで発生させることができます。

2. 実行アドレス同期割り込み
プログラムが指定アドレスを実行した時に、仮想割り込みを発生することができます。
アドレスに同期した仮想割り込みの例を以下に示します。

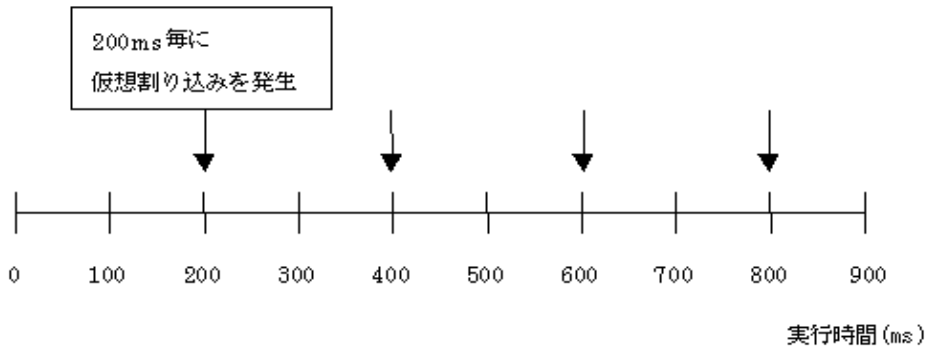


上記のように、プログラムの実行が F0000 番地に移った時に指定した仮想割り込みを発生するように定義することができます。
この機能を利用すると､次のようにプログラムが F0000 番地を 1 回目に実行した時に仮想割り込みを発生、2 回目に実行した時は割り込みを発生させないというような指定が行えます。

F0000 番地を実行した回数	仮想割り込み発生の有無
1 回目	仮想割り込みの発生
2 回目	仮想割り込みを発生させない
3 回目 : :	仮想割り込みを発生 : :

3. 時間間隔同期割り込み
指定した時間間隔で仮想割り込みを発生することができます。
指定した時間間隔に同期した仮想割り込みの例を以下に示します。

「ソフトウェア割り込み番号 21 番(タイマ A0)の仮想割り込みの定義例」



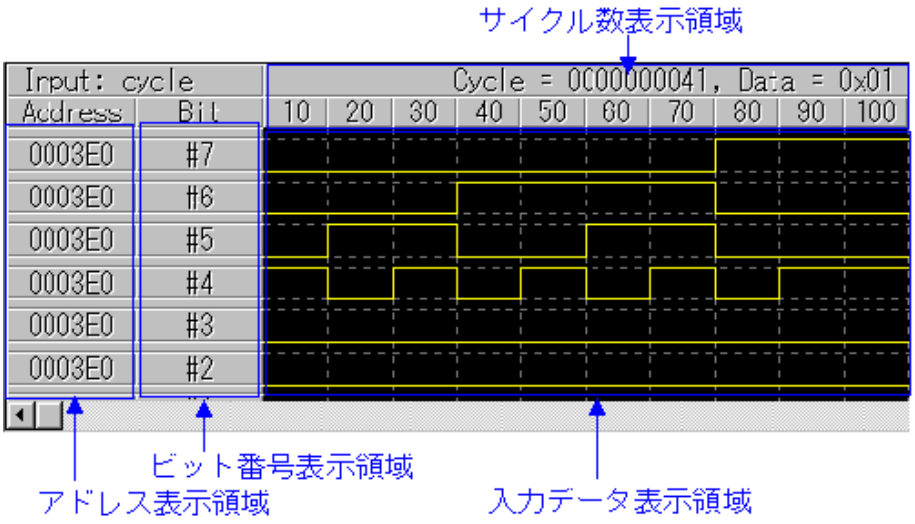
上記のように、仮想割り込み（この場合タイマ A0 の割り込み）を指定した時間間隔で発生させることができます。

6.2.4 仮想ポート入力画面の構成

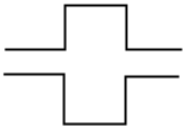
6.2.4.1 サイクル同期入力の画面構成

サイクルに同期した仮想ポート入力を設定した場合、以下の３つの形式で表示することができます。表示形式の変更は[表示形式...]メニューで行います。

1. チャート形式（ビット単位）
   設定された仮想ポート入力をビット単位のチャート形式で表示します。



- アドレス表示領域は仮想ポート入力を行うメモリのアドレスを表示します。
- ビット番号表示領域は仮想ポート入力を行うメモリのビット番号を表示します。
- 入力データ表示領域は設定された仮想ポート入力のデータをビットごとにチャート形式で表示します。

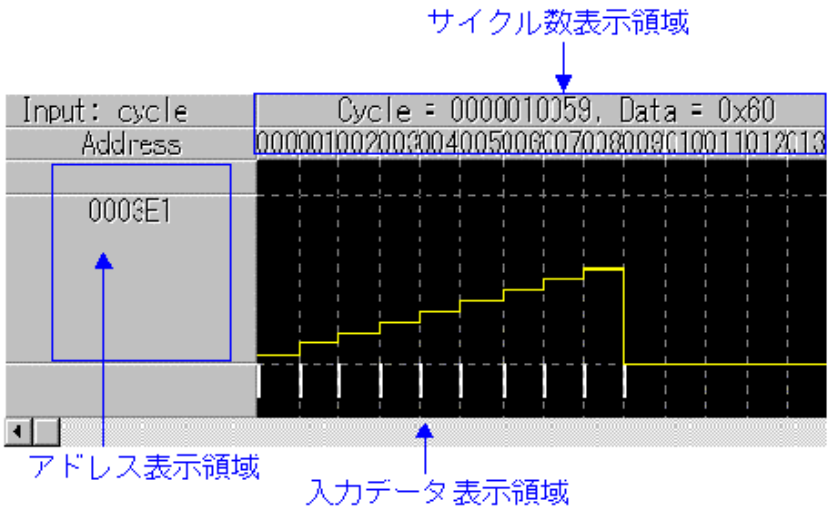

は、メモリのビットが１の状態です。

は、メモリのビットが０の状態です。

入力データ表示領域の一番下に表示されている短い白線は、データが入力されるポイントを示しています。データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

- サイクル数表示領域はサイクル数を表示します。

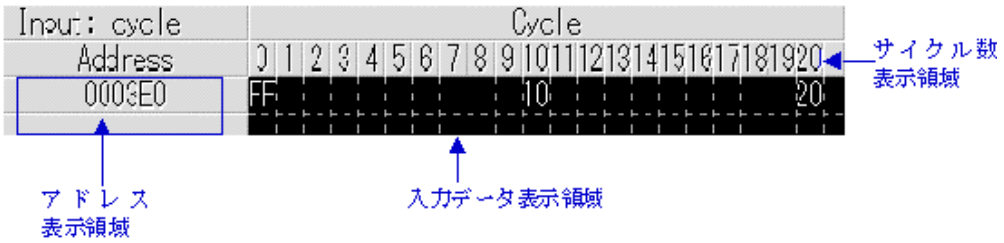
2. グラフ形式（バイト単位）
設定された仮想ポート入力をバイト単位のグラフ形式で表示します。



- アドレス表示領域は、仮想ポート入力を行うメモリのアドレスを表示します。
- 入力データ表示領域は、設定された仮想ポート入力のデータをグラフ形式で表示します。
表示されるグラフの凸は、データを表示している領域の高さを 255（1 バイトデータの最大値）等分してデータ値を表示しています。
入力データ表示領域の一番下に表示されている短い白線は、データが入力されるポイントを示しています。
データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

- サイクル数表示領域は、サイクル数を表示します。

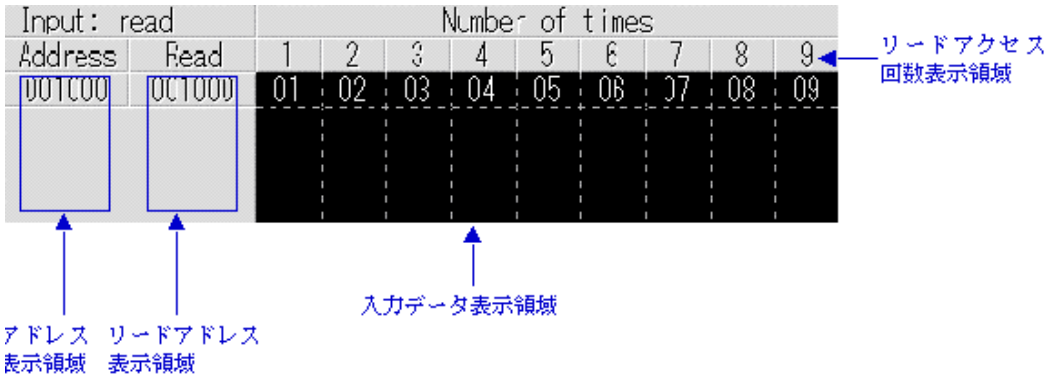
3. 16 進数値形式
設定された仮想ポート入力を 16 進数値形式で表示します。



- アドレス表示領域は、仮想ポート入力を行うメモリのアドレスを表示します。
- 入力データ表示領域は、設定された仮想ポート入力のデータを 16 進数値で表示します。
データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。
- サイクル数表示領域は、サイクル数を表示します。

6.2.4.2 リードアクセス同期入力の画面構成

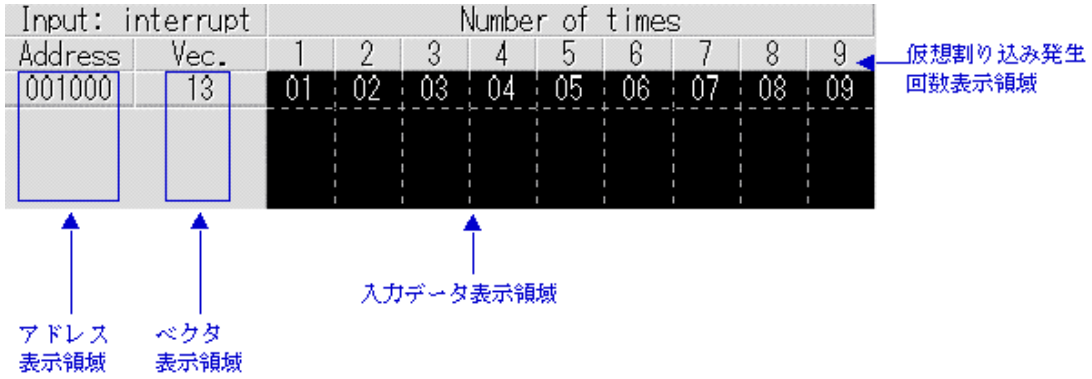
メモリのリードアクセスに同期した仮想ポート入力を設定した場合、次のような表示画面構成となります。



- アドレス表示領域は、仮想ポート入力を行うメモリのアドレスを表示します。
- リードアドレス表示領域は、リードアクセスの監視を行うアドレスを表示します。
- 入力データ表示領域は、設定された仮想ポート入力のデータを 16 進数値で表示します。
  データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。
- リードアクセス回数表示領域は、リードアクセス回数を表示します。

6.2.4.3 割り込み同期入力の画面構成

仮想割り込みに同期した仮想ポート入力を設定した場合、次のような表示画面構成となります。

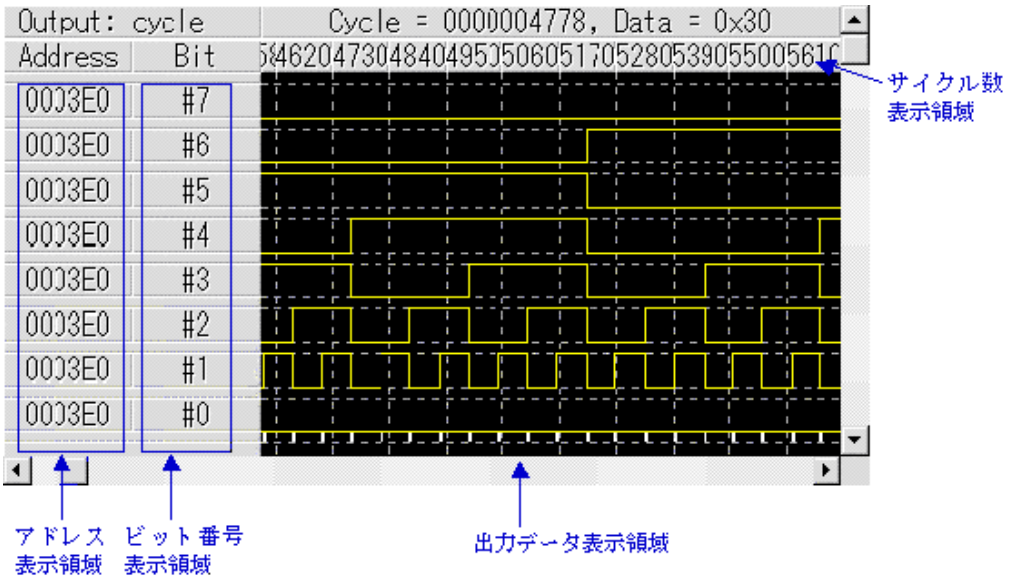


- アドレス表示領域は、仮想ポート入力を行うメモリのアドレスを表示します。
- ベクタ番号表示領域は、監視する仮想割り込みのベクタ番号を表示します。
- 入力データ表示領域は、設定された仮想ポート入力のデータを 16 進数値で表示します。
  データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。
- 仮想割り込み発生回数表示領域は、仮想割り込み発生回数を表示します。

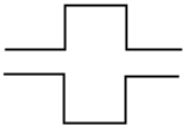
6.2.5 仮想ポート出力画面の構成

仮想ポート出力の結果は、以下の３つの形式で表示することができます。表示形式の変更は[表示形式...]メニューで行います。

1. チャート形式（ビット単位）
   仮想ポート出力の結果をビット単位のチャート形式で表示します。



- アドレス表示領域は仮想ポート出力を行うメモリのアドレスを表示します。
- ビット番号表示領域は仮想ポート出力を行うメモリのビット番号を表示します。
- 出力データ表示領域は設定された仮想ポート出力のデータをビットごとにチャート形式で表示します。

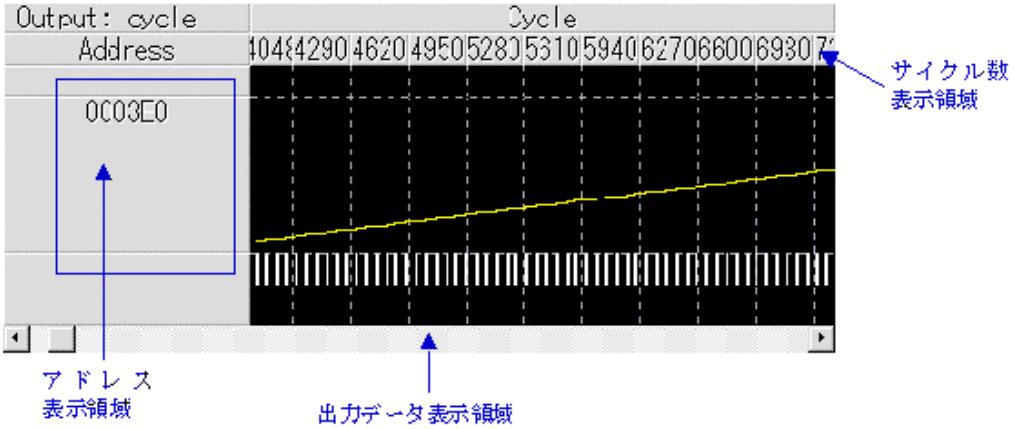

は、メモリのビットが１の状態です。

は、メモリのビットが０の状態です。

出力データ表示領域の一番下に表示されている短い白線は、データが出力されるポイントを示しています。データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

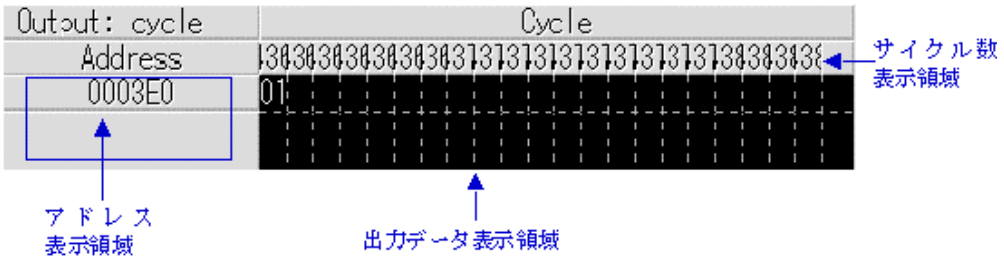
- サイクル数表示領域はサイクル数を表示します。

2. グラフ形式（バイト単位）
設定された仮想ポート入力をバイト単位のグラフ形式で表示します。



- アドレス表示領域は、仮想ポート出力を行うメモリのアドレスを表示します。
- 出力データ表示領域は、設定された仮想ポート出力のデータをグラフ形式で表示します。
  表示されるグラフの凸は、データを表示している領域の高さを 255（1 バイトデータの最大値）等分してデータ値を表示しています。
  出力データ表示領域の一番下に表示されている短い白線は、データが入力されるポイントを示しています。
  データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。
- サイクル数表示領域は、サイクル数を表示します。

3. 16 進数値形式
設定された仮想ポート入力を 16 進数値形式で表示します。

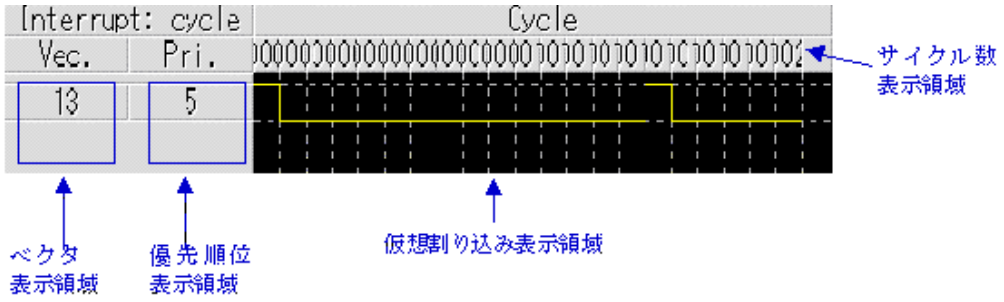


- アドレス表示領域は、仮想ポート出力を行うメモリのアドレスを表示します。
- 出力データ表示領域は、設定された仮想ポート出力のデータを 16 進数値で表示します。
  データ値を参照するには、この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。
- サイクル数表示領域は、サイクル数を表示します。

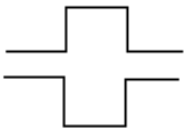
6.2.6 仮想割り込み画面の構成

6.2.6.1 サイクル同期割り込みの画面構成

サイクルに同期した仮想割り込みを設定した場合、次のような表示画面構成となります。



- ベクタ番号表示領域は仮想割り込みのベクタ番号を表示します。
- 優先順位表示領域は、仮想割り込みの優先順位を表示します。
- 仮想割り込み表示領域は、設定された仮想割り込みの発生タイミングをグラフ形式で表示します。



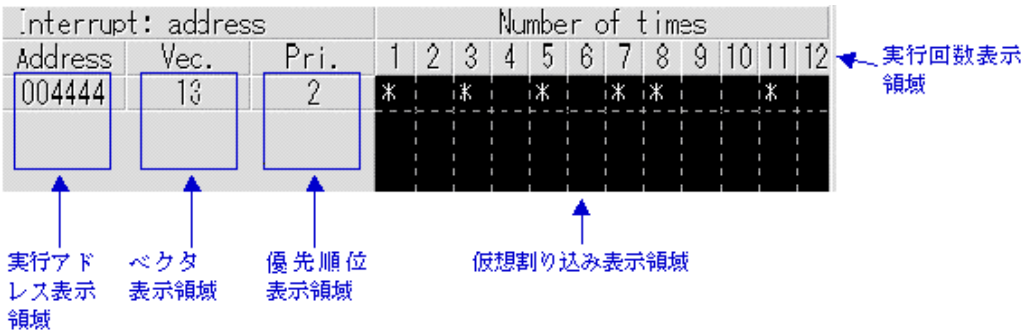
は、仮想割り込みを発生することを意味します。

は、仮想割り込みを発生させないことを意味します。

- サイクル数表示領域はサイクル数を表示します。

6.2.6.2 実行アドレス同期割り込みの画面構成

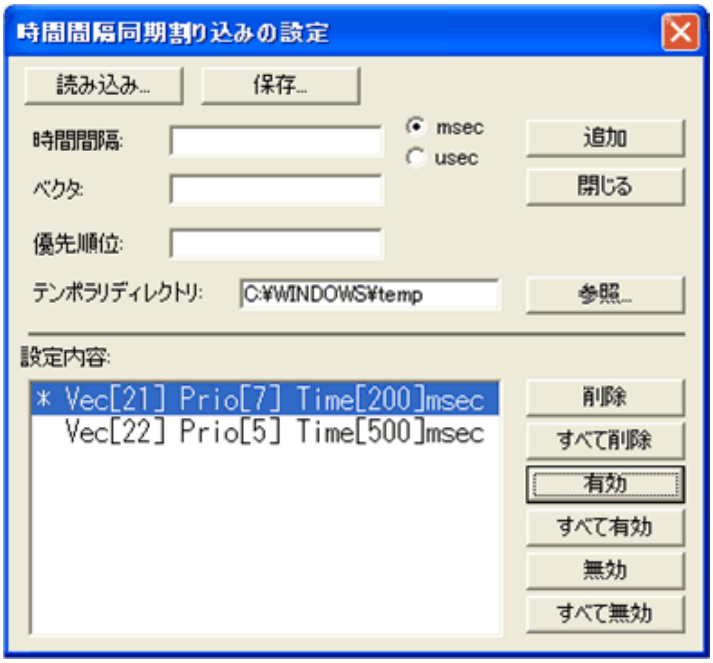
実行アドレスに同期した仮想割り込みを設定した場合、次のような表示画面構成となります。



- 実行アドレス表示領域は、仮想割り込みの発生タイミングとなるフェッチアドレス（プログラムが実行されるアドレス）を表示します。
- ベクタ番号表示領域は、仮想割り込みのベクタ番号を表示します。
- 優先順位表示領域は、仮想割り込みの優先順位を表示します。
- 仮想割り込み表示領域は、設定された仮想割り込みの発生タイミングをアスタリスク（＊）で表示します。
  アスタリスク（＊）が表示されていると、仮想割り込みを発生することを意味します。表示されていない時は、仮想割り込みが発生しないことを意味します。
- 実行回数表示領域は、実行回数（プログラムが指定アドレスを実行した回数）を表示します。

6.2.6.3 時間間隔同期割り込みの画面構成

時間間隔に同期した仮想割り込みを設定する場合は、[時間間隔同期割り込み]ボタンをクリックしてオープンする時間間隔同期割り込みの設定ダイアログを使用します。時間間隔同期割り込みの設定ダイアログは、次のような表示画面構成になります。



- 仮想割り込み登録領域は、仮想割り込みと発生タイミングとなる時間間隔を指定します。
- 時間間隔は、実行サイクル数と、Init ダイアログの MCU タブで指定する MCU クロックと分周比から算出します。
- 仮想割り込み表示領域は、登録されている仮想割り込みと発生タイミングとなる時間間隔を表示します。
- 仮想割り込みに対する操作ボタンは、各仮想割り込みに対して、削除、無効／有効を切り換えることができます。
- 仮想割り込みの保存／読み込み操作ボタンは、仮想割り込み情報をファイルに保存することができます。保存した仮想割り込み情報を読み込むことも可能です。

6.2.7 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
設定...	データ（仮想ポート入力、仮想ポート出力、仮想割り込みデータ）をセットアップします。
変更...	セットアップデータを変更します。
削除...	セットアップデータを削除します。
読み込み...	保存した I/O スクリプトファイルを読み込みます。
表示形式...	表示形式を変更します。
スケール...	スケールを変更します。
表示色の設定...	表示色を変更します。
時間間隔同期割り込み...	時間間隔同期割り込みを設定します。
登録 I/O スクリプトファイル一覧...	登録されている I/O スクリプトファイルを表示します。
仮想ポート入力データの前方検索	仮想ポート入力データの前方検索します。
仮想ポート入力データの後方検索	仮想ポート入力データの後方検索します。
仮想ポート出力データの前方検索	仮想ポート出力データの前方検索します。
仮想ポート出力データの後方検索	仮想ポート出力データの後方検索します。
仮想割り込みデータの前方検索	仮想割り込みデータの前方検索します。
仮想割り込みデータの後方検索	仮想割り込みデータの後方検索します。
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。

6.2.8 仮想ポート入力の設定

仮想ポート入力機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。 データをメモリに入力できるタイミングとして以下のものがあります。

1. あるメモリに対して時間の変化とともにデータを入力したい場合 プログラムの実行が指定サイクルになった時に、データを入力することができます。 この場合、サイクル同期入力の設定を行って下さい。
2. あるメモリがリードされたタイミングで、データを入力したい場合 指定されたメモリをプログラムがリードアクセスした時にデータを入力することができます。例えば、ある変数（グローバル変数等のアドレスが固定番地に割り付けられている変数）をリードしたタイミングでその変数に値を代入したい場合などに利用します。 この場合、リードアクセス同期入力の設定を行って下さい。

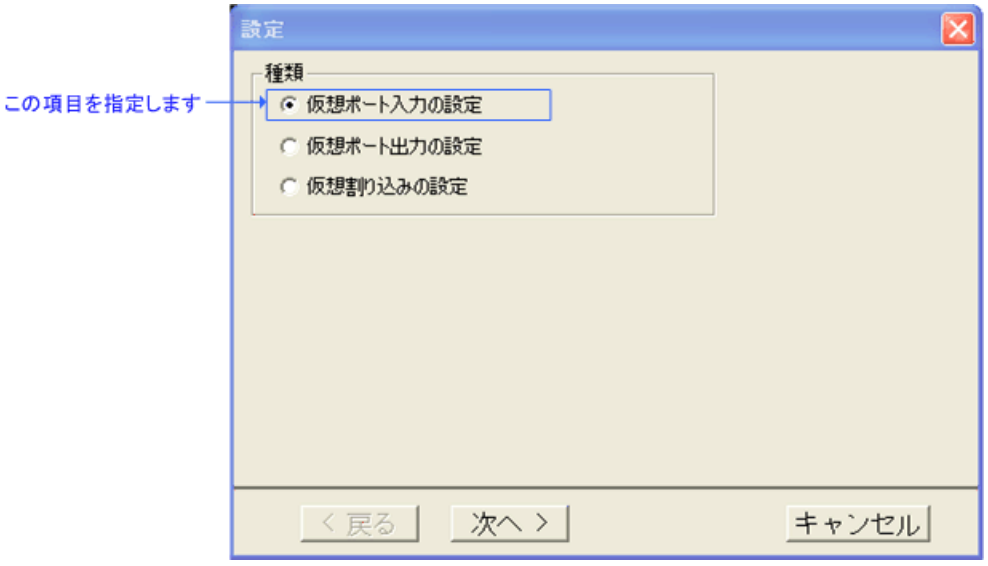
3. ある仮想割り込みが発生したタイミングで、データを入力したい場合 指定された仮想割り込みが発生した時に、データを入力することができます。 例えば、割り込みハンドラの中で SFR のメモリを参照している場合などに利用します。 この場合、割り込み同期入力の設定を行って下さい。

注意事項

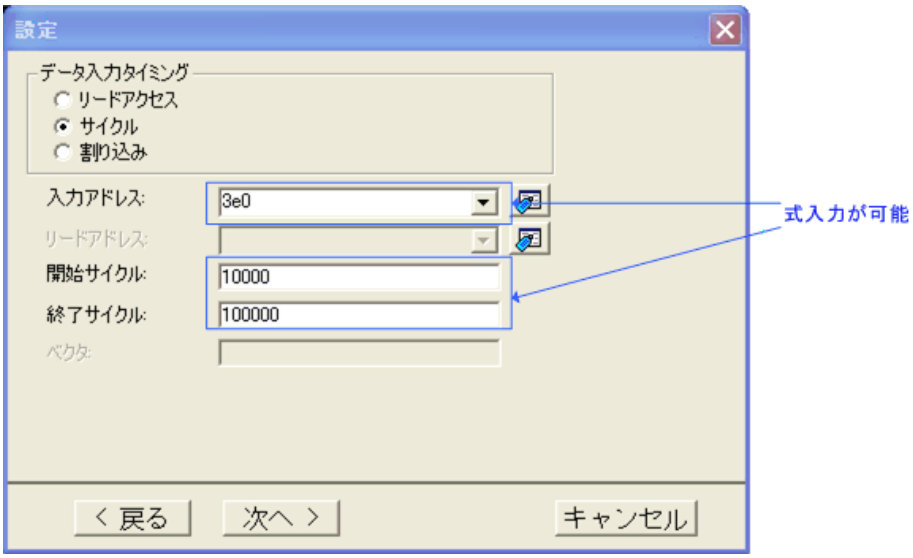
仮想ポート入力、仮想割り込み、I/O スクリプトのプロシジャー数は、合わせて 50 個まで設定できます。ただし､出力ポートウィンドウで Printf 関数の出力機能を使用している場合､設定できる仮想ポート入力、仮想割り込み、I/O スクリプトのプロシジャー数は、合わせて 48 個までになります 。

6.2.8.1 サイクル同期入力の設定

サイクルに同期した仮想ポート入力を設定するには、I/O タイミング設定ウィンドウの[設定...]メニューを選択して下さい。
選択すると次のダイアログがオープンします。



ここで、仮想ポート入力の設定の欄を選択し[次へ]ボタンを押して下さい（キャンセルボタンを押すと設定を取りやめてダイアログをクローズします）。
仮想ポート入力のタイミングを設定するダイアログがオープンします。

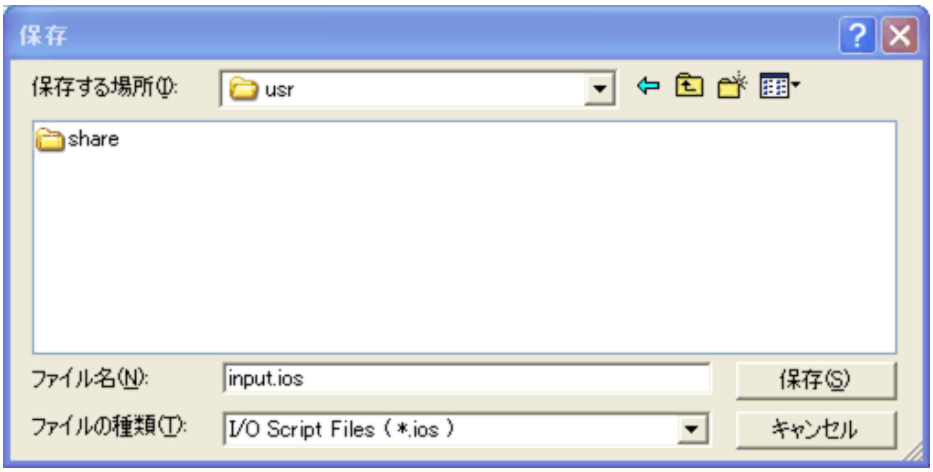


まず、データ入力タイミングの欄でサイクルを指定します。次に、入力アドレス欄に仮想ポート入力を行いたいアドレス（データを入力するアドレス）を 16 進数値で指定します。そして、仮想ポート入力を開始するサイクルと終了サイクルを 10 進数値で、開始サイクル,終了サイクルにそれぞれ指定します。その後、[次へ]ボタンを押して下さい（ここで[戻る]ボタンを押すと前の設定ダイアログに戻ることができます）。仮想ポート入力のデータを設定するマトリクスダイアログがオープンします。

このダイアログでは、実際にメモリに入力するデータを設定します。
データの設定手順は以下のように行って下さい。

1. データを設定したいサイクルの箇所（エレメントといいます）までマウスを移動し（画面をスクロールすることもできます）左ボタンをダブルクリックします。
2. 選択した箇所でデータを 16 進数値で入力して下さい。
   入力できるデータのサイズは 1 バイトです（0x0～0xFF まで）。
3. 入力データ数分 1、2 を繰り返して下さい。

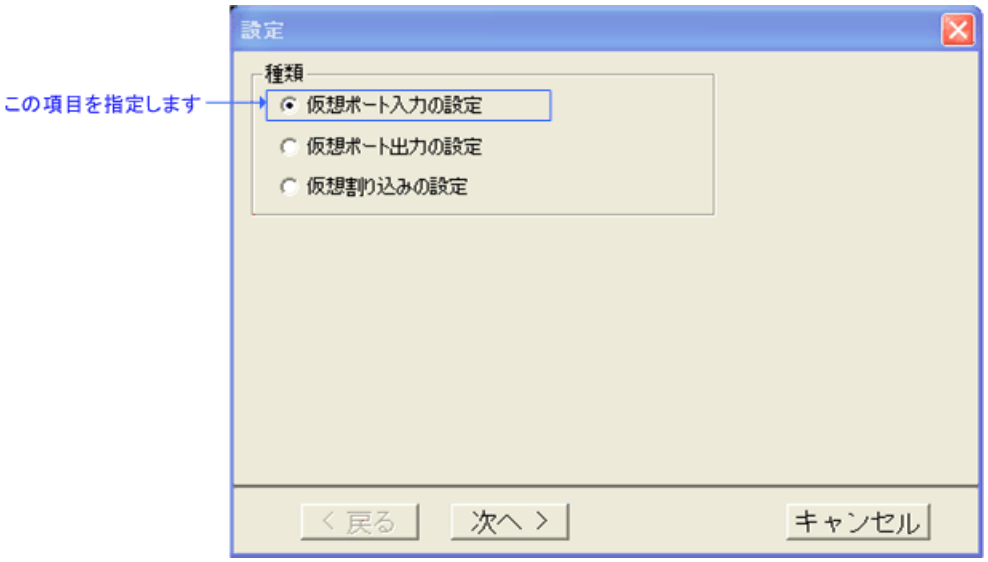
すべてのデータの入力後、[次へ]ボタンを押してください。
設定した仮想ポート入力のデータをファイル（仮想ポート入力ファイル）に保存するためのダイアログがオープンします。



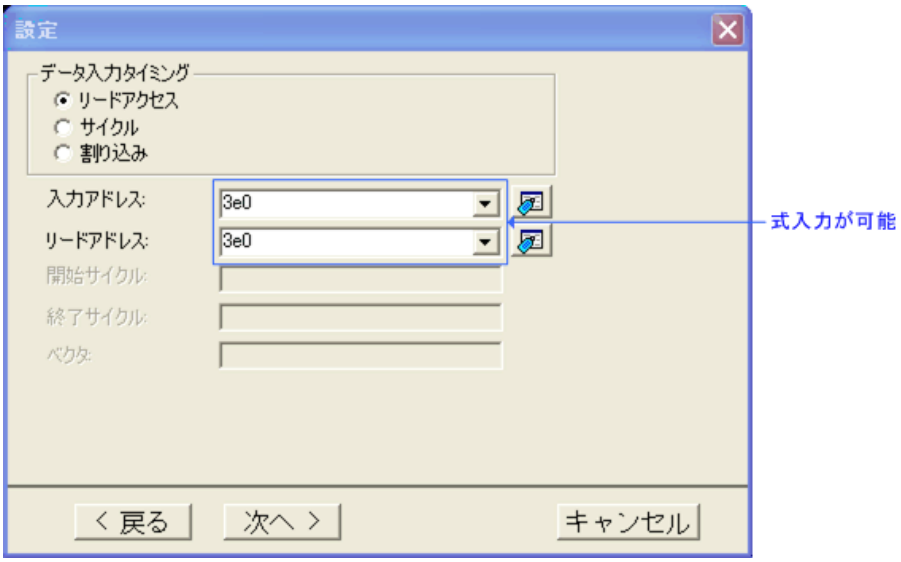
ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。 保存したファイルは、I/O タイミング設定ウィンドウの[読み込み...]メニューで再度読み込むことができます。ファイル名を入力したら、保存ボタンを押して下さい。 これで、サイクルに同期した仮想ポート入力の設定が完了します。

6.2.8.2 リードアクセス同期入力の設定

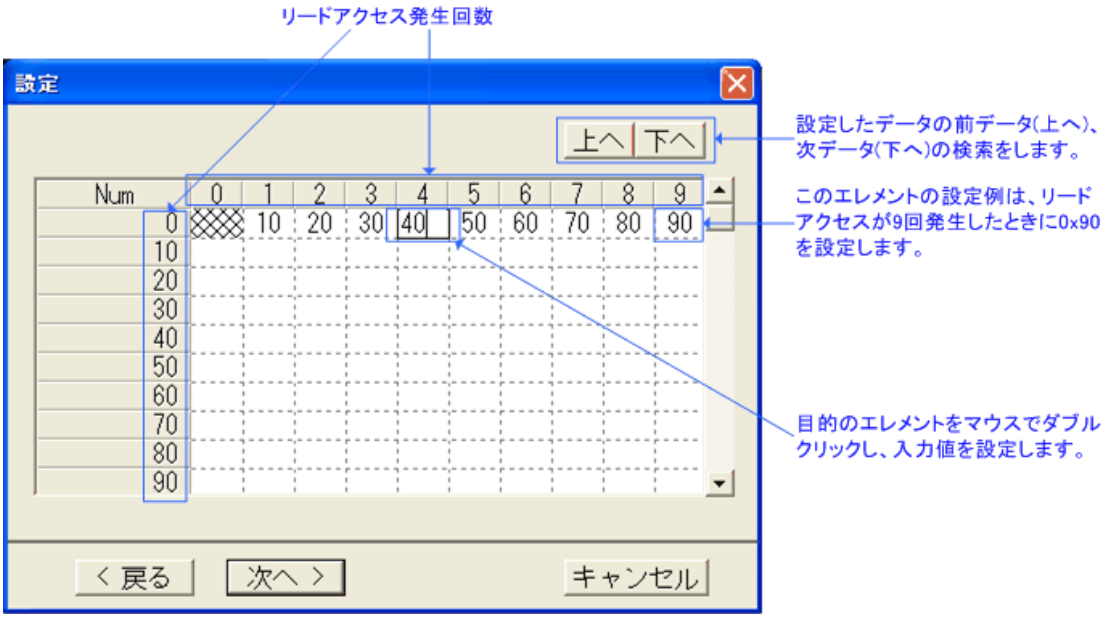
リードアクセスに同期した仮想ポート入力を設定するには、I/O タイミング設定ウィンドウの[設定...]メニュー を選択して下さい。選択すると次のダイアログがオープンします。



ここで、仮想ポート入力の設定の欄を選択し[次へ]ボタンを押して下さい（キャンセルボタンを押すと設定を取りやめてダイアログをクローズします）。仮想ポート入力のタイミングを設定するダイアログがオープンします。



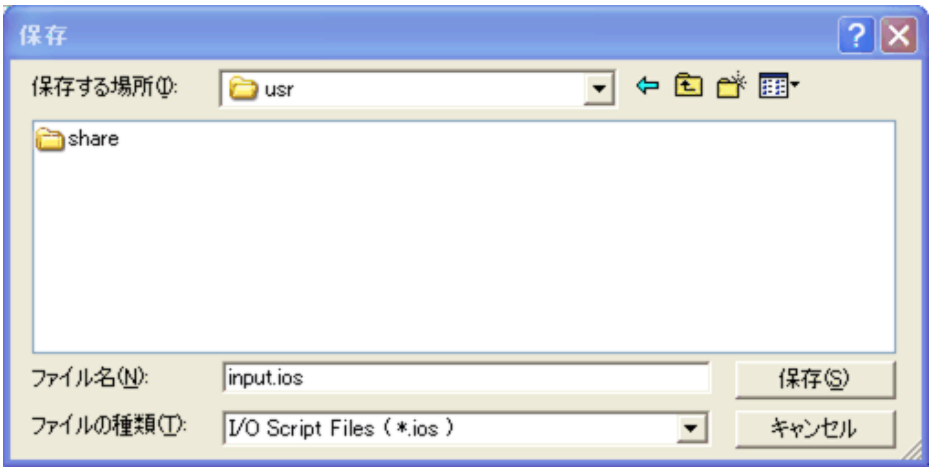
まず、データ入力タイミングの欄でリードアクセスを指定します。次に、入力アドレス欄に仮想ポート入力を行いたいアドレス（データを入力するアドレス）を 16 進数値で指定します。そして、リードアドレス欄にリードアクセス(メモリの読み込み)が行われるアドレスを入力して下さい（ここで指定したメモリにリードアクセスが発生した時に仮想ポート入力を行います）。その後、[次へ]ボタンを押して下さい（ここで[戻る]ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想ポート入力のデータを設定するマトリクスダイアログがオープンします。



このダイアログでは、実際にメモリに入力するデータを設定します。 データの設定手順は以下のように行って下さい。

1. データを設定したいリードアクセス発生回数の箇所（エレメントといいます）までマウスを移動し（画面をスクロールすることもできます）左ボタンをダブルクリックします。
2. 選択した箇所でデータを 16 進数値で入力して下さい。
   入力できるデータのサイズは 1 バイトです（0x0～0xFF まで）。
3. 入力データ数分 1、2 を繰り返して下さい。

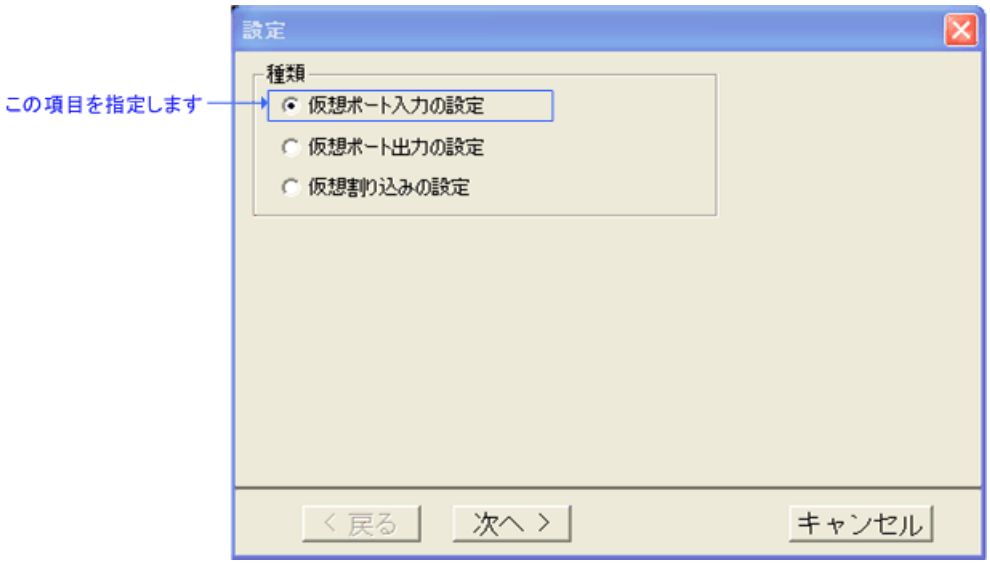
すべてのデータの入力後、[次へ]ボタンを押してください。
設定した仮想ポート入力のデータをファイル（仮想ポート入力ファイル）に保存するためのダイアログがオープンします。



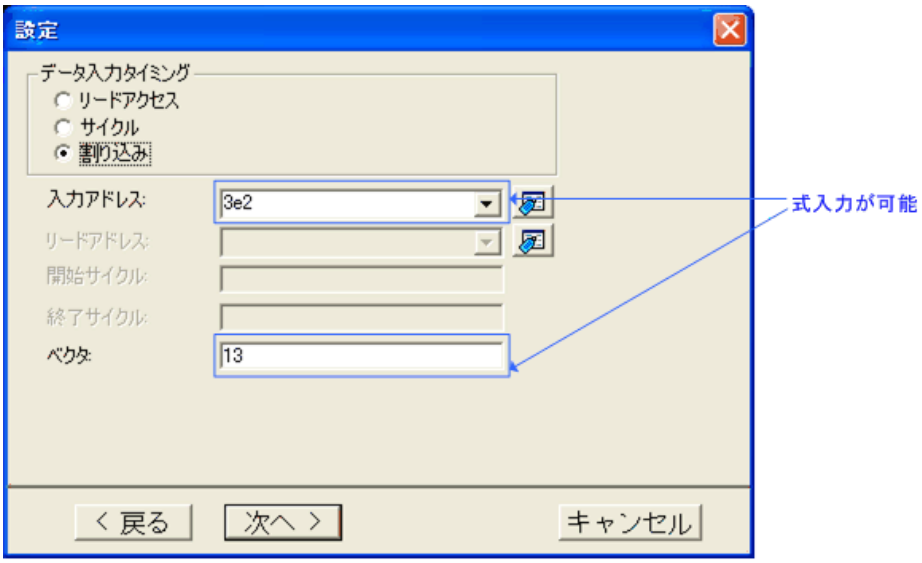
ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。 保存したファイルは、I/O タイミング設定ウィンドウの[読み込み...]メニュー で再度読み込むことができます。ファイル名を入力したら、保存ボタンを押して下さい。 これで、リードアクセスに同期した仮想ポート入力の設定が完了します。

6.2.8.3 割り込み同期入力の設定

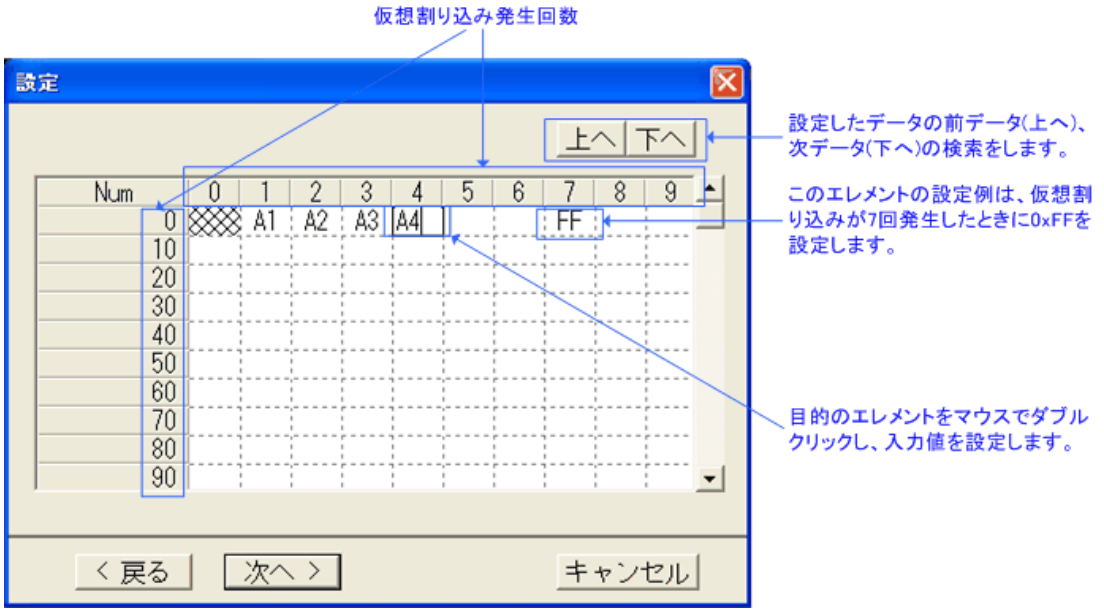
仮想割り込みに同期した仮想ポート入力を設定するには、I/O タイミング設定ウィンドウの[設定...]メニューを選択して下さい。選択すると次のダイアログがオープンします。



ここで、仮想ポート入力の設定の欄を選択し[次へ]ボタンを押して下さい（キャンセルボタンを押すと設定を取りやめてダイアログをクローズします）。仮想ポート入力のタイミングを設定するダイアログがオープンします。



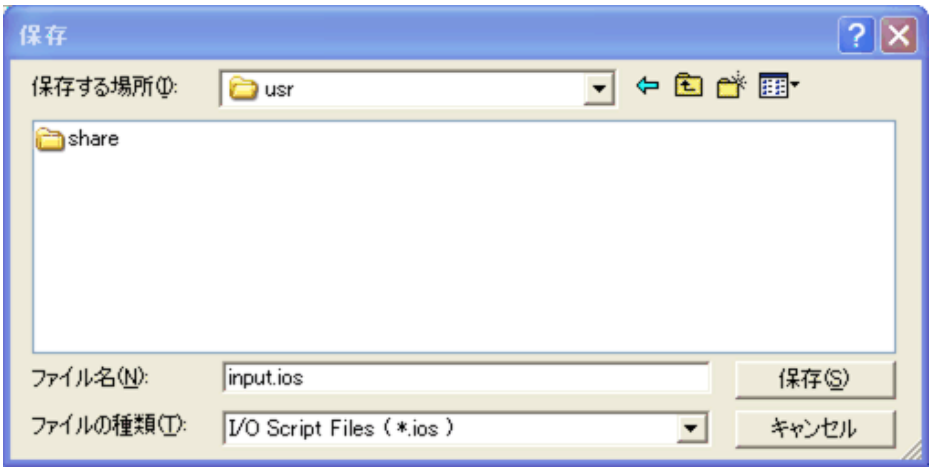
まず、データ入力タイミングの欄で[割り込み]を指定します。次に、入力アドレス欄に仮想ポート入力を行いたいアドレス（データを入力するアドレス）を 16 進数値で指定します。そして、ベクタ欄に仮想ポート入力のタイミングとなる仮想割り込みのベクタ番号を入力して下さい（M32R 用シミュレータの場合、ベクタアドレスは固定です）。その後、[次へ]ボタンを押して下さい（ここで[戻る]ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想ポート入力のデータを設定するマトリクスダイアログがオープンします。



このダイアログでは、実際にメモリに入力するデータを設定します。
データの設定手順は以下のように行って下さい。

1. データを設定したい割り込み発生回数の箇所（エレメントといいます）までマウスを移動し（画面をスクロールすることもできます）左ボタンをダブルクリックします。
2. 選択した箇所でデータを 16 進数値で入力して下さい。
   入力できるデータのサイズは 1 バイトです（0x0～0xFF まで）。
3. 入力データ数分 1、2 を繰り返して下さい。

すべてのデータの入力後、[次へ]ボタンを押してください。
設定した仮想ポート入力のデータをファイル（仮想ポート入力ファイル）に保存するためのダイアログがオープンします。



ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O タイミング設定ウィンドウの[読み込み...]メニューで再度読み込むことができます。ファイル名を入力したら、保存ボタンを押して下さい。
これで、仮想割り込みに同期した仮想ポート入力の設定が完了します。

6.2.9 仮想ポート出力の設定

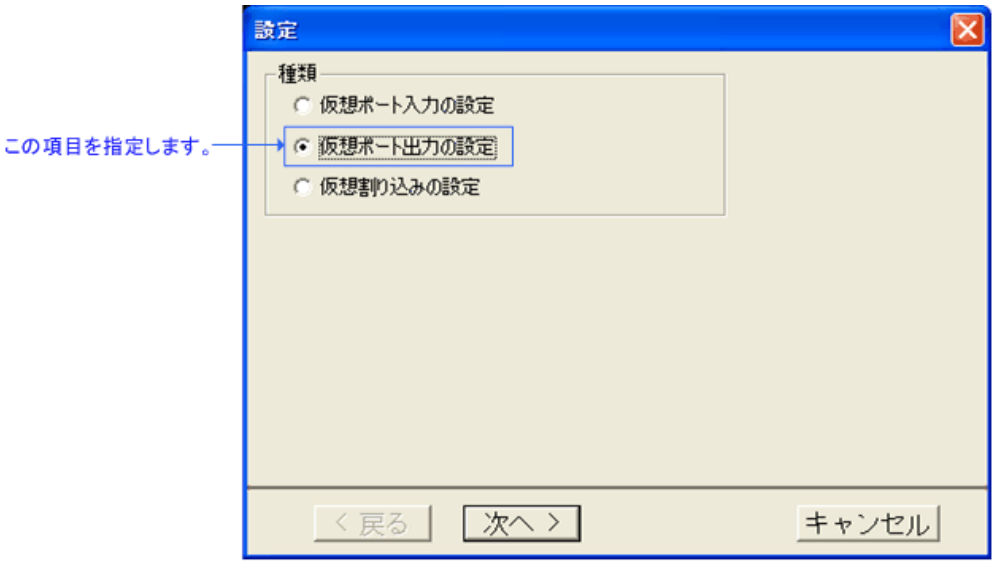
仮想ポート出力機能を利用すると、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録することができます。
記録されたデータは、グラフや数値形式で確認できます。

注意事項

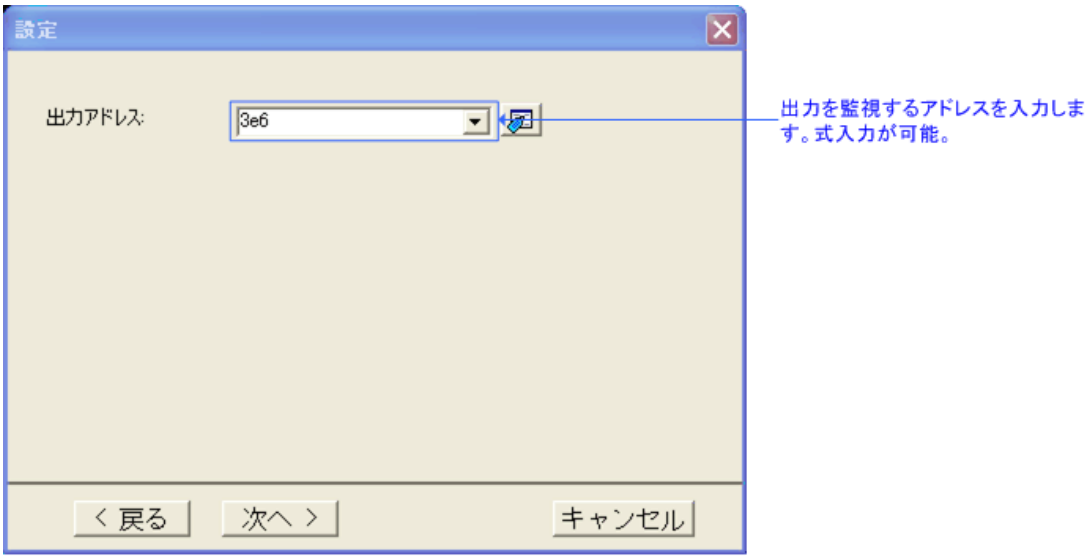
- 記録するデータの個数は、プログラム実行開始時から Init ダイアログの[I/O スクリプト]タブで指定した個数分です。 再実行した場合、前回のデータはクリアされます。
- 仮想ポート出力は 200 個まで設定できます。ただし、出力ポートウィンドウを使用している場合、設定できる仮想ポート出力は、199 個までになります。

6.2.9.1 仮想ポート出力の設定

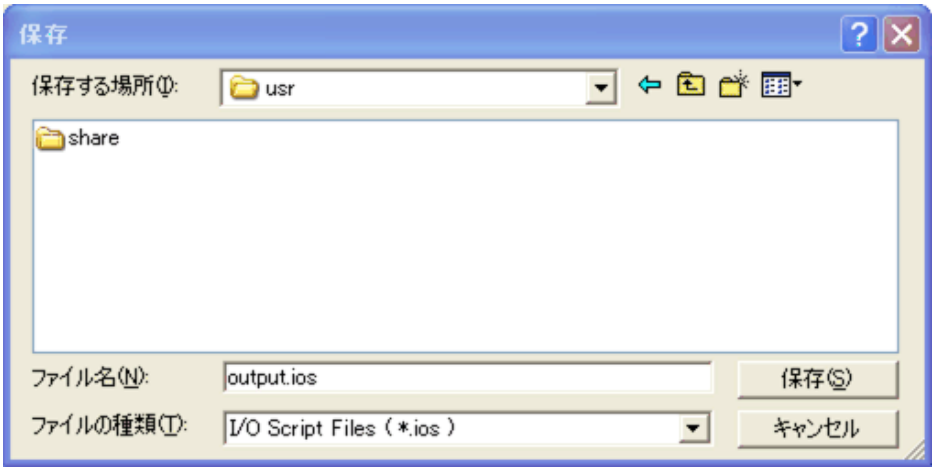
仮想ポート出力を設定するには、I/O タイミング設定ウィンドウの[設定...]メニューを選択して下さい。選択すると次のダイアログがオープンします。



ここで、仮想ポート出力の設定の欄を選択し[次へ]ボタンを押して下さい（キャンセルボタンを押すと設定を取りやめてダイアログをクローズします）。仮想ポート出力の監視を行うアドレスを設定するダイアログがオープンします。



仮想ポート出力の監視を行うアドレスを出力アドレス欄に入力して下さい。
入力後、[次へ]ボタンを押してください。
仮想ポート出力の結果を保存(記録)するファイル（仮想ポート出力ファイル）を指定するためのダイアログがオープンします。（デバッガは、プログラム実行中に発生した仮想ポート出力をファイルに保存し、プログラム停止時にそのファイルを参照します）。



ここで、仮想ポート出力ファイルを保存するディレクトリとファイル名を入力して下さい。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、仮想ポート出力の設定が完了します。

6.2.10 仮想割り込みの設定

仮想割り込み機能を利用すると、割り込みを擬似的に発生させることができます。この機能を利用すると、擬似的にタイマ割り込みやキー入力割り込み等の設定が行えます。
仮想割り込みを発生することのできるタイミングを以下に示します。

1. 時間の変化とともに、仮想割り込みを発生させる場合
   プログラムの実行が指定サイクルになった時に、仮想割り込みを発生させることができます。
   この場合、サイクル同期割り込みの設定を行って下さい。

2. プログラムが指定アドレスを実行した時に、仮想割り込みを発生させる場合
   ある関数を実行した時に仮想割り込みを発生させたい場合等に使用します。
   この場合、実行アドレス同期割り込みの設定を行って下さい。

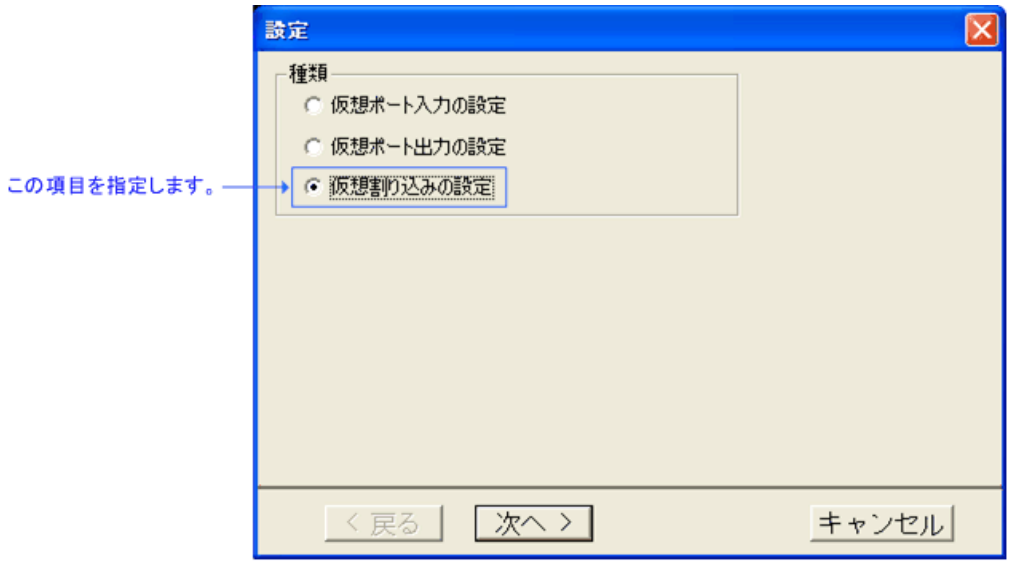
3. 一定時間間隔で仮想割り込みを発生させる場合
   一定時間間隔で仮想割り込みを発生させたい場合に使用します。
   この場合、時間間隔同期割り込みの設定を行って下さい。

注意事項

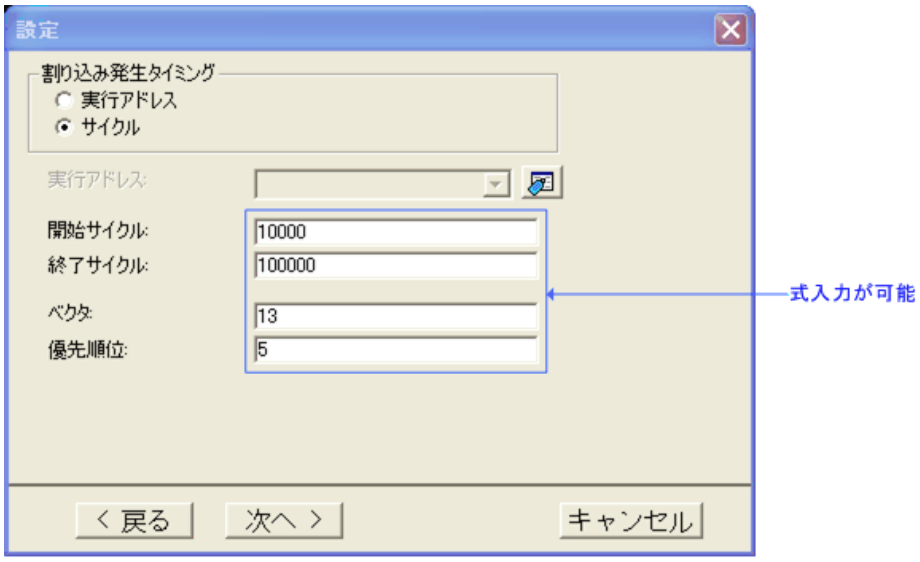
仮想ポート入力、仮想割り込み、I/O スクリプトのプロシジャー数は、合わせて 50 個まで 設定できます。ただし、出力ポートウィンドウで printf 関数の出力機能を使用している場合、設定できる仮想ポート入力、仮想割り込み、I/O スクリプトのプロシジャー数は、 合わせて 48 個までになります。

6.2.10.1 サイクル同期割り込みの設定

サイクルに同期した仮想割り込みを設定するには、I/O タイミング設定ウィンドウの[設定...]メニューを選択して下さい。選択すると次のダイアログがオープンします。



ここで、仮想割り込みの設定の欄を選択し[次へ]ボタンを押して下さい（キャンセルボタンを押すと設定を取りやめてダイアログをクローズします）。仮想割り込みのタイミングを設定するダイアログがオープンします。



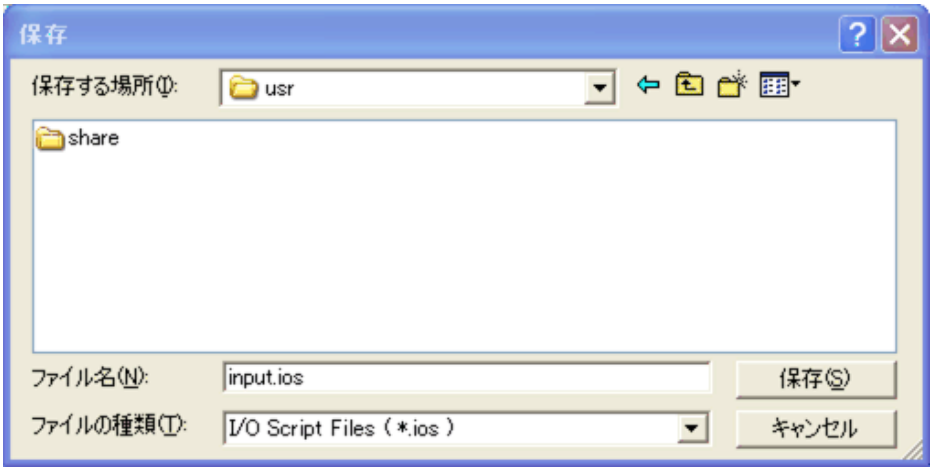
まず、割り込み発生タイミングの欄で[サイクル]を指定します。 次に、仮想割り込みを開始するサイクルと終了サイクルを 10 進数値で、開始サイクル,終了サイクルにそれぞれ指定します。 そして、発生させる仮想割り込みのベクタ番号と優先順位を 10 進数値で、ベクタ, 優先順位にそれぞれ指定します（M32R 用シミュレータの場合、ベクタアドレスは固定で優先順位はありません）。 その後、[次へ]ボタンを押して下さい（ここで[戻る]ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想割り込みを設定するマトリクスダイアログがオープンします。

このダイアログでは、実際に仮想割り込みの発生の有無を設定します。 仮想割り込みの設定手順は以下のように行って下さい。

1. 仮想割り込みを設定したいサイクルの箇所（エレメントといいます）までマウスを移動し（画面をスクロールすることもできます）左ボタンをクリックします。
2. クリックすると＊マークが付きます。再度、同じ箇所をクリックすると仮想割り込みの設定を解除できます。その時は、＊マークが消えます。
3. 仮想割り込みの発生数分 1、2 を繰り返して下さい。

すべての仮想割り込みの設定後、[次へ]ボタンを押してください。

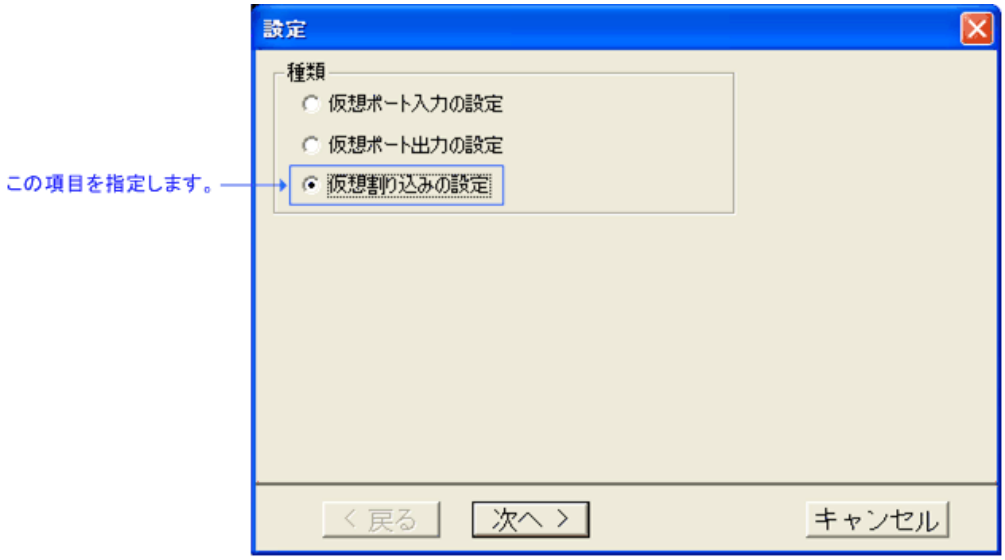
設定した仮想割り込みのデータをファイル（仮想割り込みファイル）に保存するためのダイアログがオープンします。



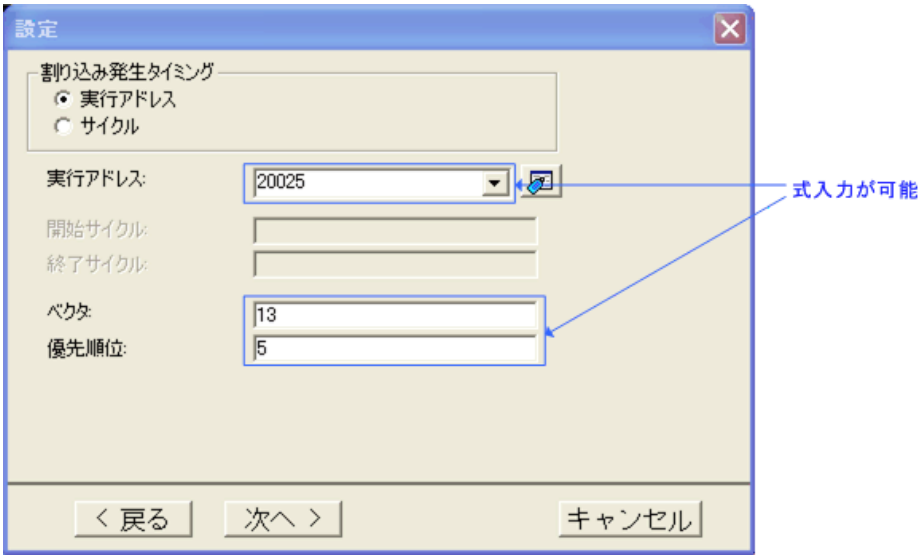
ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O タイミング設定ウィンドウの[読み込み...]メニューで再度読み込むことができます。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、サイクルに同期した仮想割り込みの設定が完了します。

6.2.10.2 実行アドレス同期割り込みの設定

アドレスに同期した仮想割り込みを設定するには、I/O タイミング設定ウィンドウの[設定...]メニューを選択して下さい。選択すると次のダイアログがオープンします。



ここで、仮想割り込みの設定の欄を選択し[次へ]ボタンを押して下さい（キャンセルボタンを押すと設定を取りやめてダイアログをクローズします）。仮想割り込みのタイミングを設定するダイアログがオープンします。



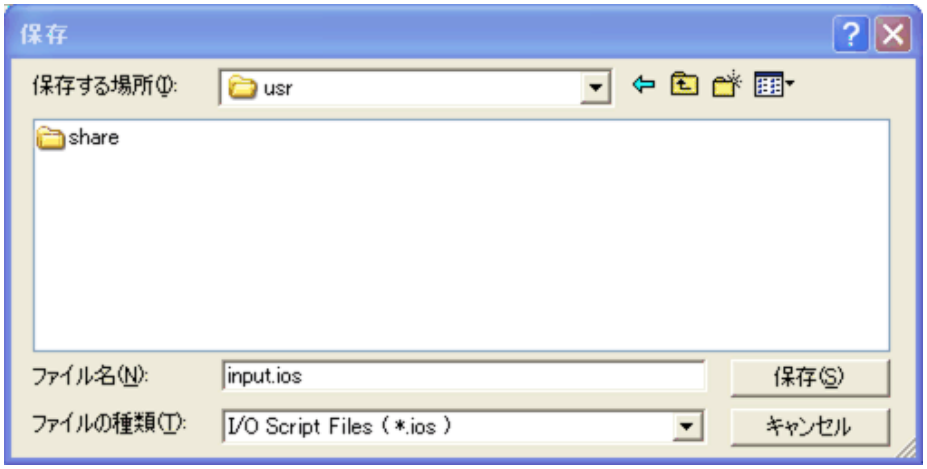
まず、割り込み発生タイミングの欄で実行アドレスを指定します。 次に、仮想割り込みを行うタイミングとなる実行アドレス（ここで指定したアドレスを実行した時に仮想割り込みを行います）を[実行アドレス]に指定します。 そして、発生させる仮想割り込みのベクタ番号と優先順位を 10 進数値で、ベクタ ,優先順位にそれぞれ指定します（M32R 用シミュレータの場合、ベクタアドレスは固定で優先順位はありません）。 その後、[次へ]ボタンを押して下さい（ここで[戻る]ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想割り込みを設定するマトリクスダイアログがオープンします。

このダイアログでは、実際に仮想割り込みの発生の有無を設定します。 仮想割り込みの設定手順は以下のように行って下さい。

1. 仮想割り込みを設定したいサイクルの箇所（エレメントといいます）までマウスを移動し（画面をスクロールすることもできます）左ボタンをクリックします。
2. クリックすると＊マークが付きます。再度、同じ箇所をクリックすると仮想割り込みの設定を解除できます。その時は、＊マークが消えます。
3. 仮想割り込みの発生数分 1、2 を繰り返して下さい。

すべての仮想割り込みの設定後、[次へ]ボタンを押してください。

設定した仮想割り込みのデータをファイル（仮想割り込みファイル）に保存するためのダイアログがオープンします。



ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O タイミング設定ウィンドウの[読み込み...]メニューで再度読み込むことができます。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、アドレスに同期した仮想割り込みの設定が完了します。

6.2.10.3 時間間隔同期割り込みの設定

一定時間間隔に同期した仮想割り込みを設定するには、I/O タイミング設定ウィンドウの[時間間隔同期割り込み...]メニューを選択して下さい。選択すると一定時間間隔に同期した仮想割り込みを設定するためのダイアログがオープンします。
ダイアログでは以下の項目を行うことができます。

- 仮想割り込みを設定する
- 仮想割り込みを削除する
- 仮想割り込みを一時的に無効にする
- 無効にした仮想割り込みを有効にする
- 仮想割り込みを保存する
- 仮想割り込みを読み込む

各項目の指定方法は以下の通りです。

6.2.10.3.1. 仮想割り込みを設定する

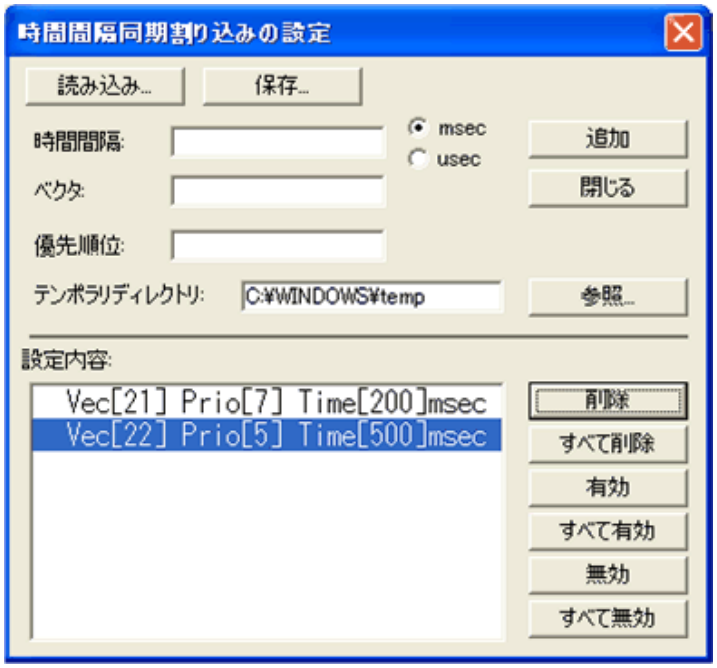
M16C/6X 用シミュレータデバッガでの指定例を示します。
例）ベクタ番号 21 の割り込みを優先順位(IPL)7 で、200ms 毎に発生させる

以下のように設定します。なお、テンポラリディレクトリ欄は、仮想割り込み設定のためにデバッガ内部で使用するテンポラリの領域ですので、書き込み可能なディレクトリを設定してください。

[追加]ボタンをクリックすると、ダイアログ下部の仮想割り込み一覧に、設定した仮想割り込みが追加されます。
仮想割り込みの設定完了後は、[閉じる]ボタンをクリックして下さい。

6.2.10.3.2. 仮想割り込みを削除する

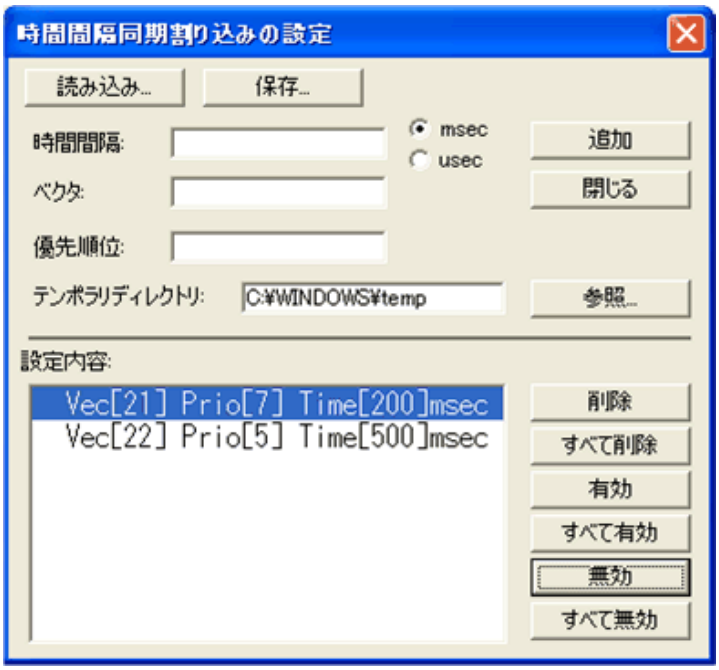
ダイアログ下部の仮想割り込み一覧から削除する仮想割り込みをクリックし、 [削除]ボタンをクリックして下さい。



すべての仮想割り込みを削除する場合は、[すべて削除]ボタンをクリックして下さい。
仮想割り込みの設定完了後は、[閉じる]ボタンをクリックして下さい。

6.2.10.3.3. 仮想割り込みを一時的に無効にする

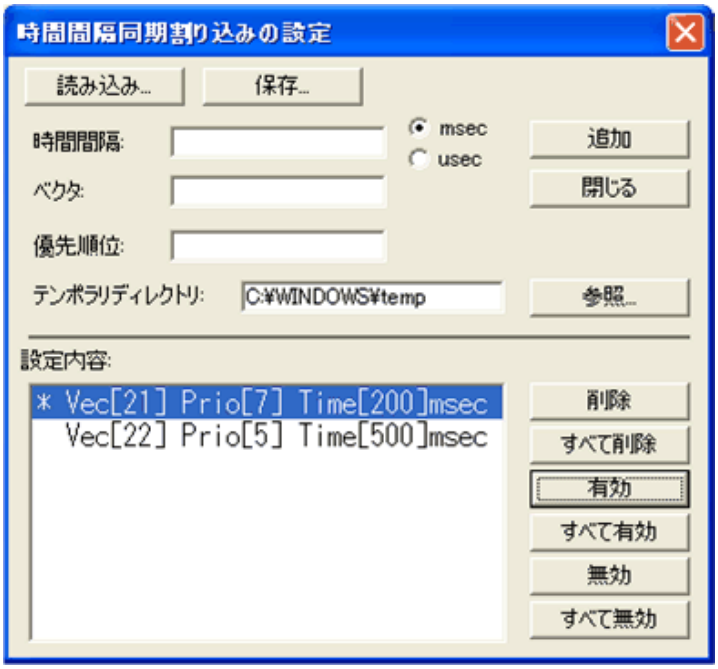
ダイアログの仮想割り込み一覧から一時的に無効にする仮想割り込みをクリックし、 [無効]ボタンをクリックして下さい。
仮想割り込みをダブルクリックすることにより、一時的に無効にすることもできます。



一時的に無効にした仮想割り込みは、仮想割り込み一覧の左側に"*"が表示されます。
すべての仮想割り込みを一時的に無効にする場合は、[すべて無効]ボタンをクリックして下さい。
仮想割り込みの設定完了後は、[閉じる]ボタンをクリックして下さい。

6.2.10.3.4. 無効にした仮想割り込みを有効にする

ダイアログの仮想割り込み一覧から有効にする仮想割り込みをクリックし、 [有効]ボタンをクリックして下さい。 仮想割り込みをダブルクリックすることにより、有効にすることもできます。



有効にした仮想割り込みは、仮想割り込み一覧の左側の"*"が消えます。 すべての仮想割り込みを有効にする場合は、[すべて有効]ボタンをクリックして下さい。 仮想割り込みの設定完了後は、[閉じる]ボタンをクリックして下さい。

6.2.10.3.5. 仮想割り込みを保存する

ダイアログの[保存...]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。仮想割り込みファイル名を指定して下さい。拡張子を省略した場合は、拡張子.stm が付加されます。

6.2.10.3.6. 仮想割り込みを読み込む

ダイアログの[開く...]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。読み込む仮想割り込みファイルを指定して下さい。現在設定されている仮想割り込みにファイルから読み込んだ仮想割り込みを追加します。

6.2.11 仮想ポート入力、仮想割り込み、I/O スクリプトファイルの評価タイミングについて

設定した仮想ポート入力、仮想割り込み、及び I/O スクリプトファイルの評価は以下のタイミングで行います。

6.2.11.1 評価タイミング

1. プログラム実行時、カム実行時
2. ステップ実行時
3. オーバステップ実行時
4. リターン実行時

6.2.11.2 プログラムがリセットされた場合の処理

設定した仮想ポート入力、仮想割り込み、及び I/O スクリプトファイルは再評価されます。
つまり、プログラムがリセットされると設定されている仮想ポート入力、仮想割り込み、 I/O スクリプトファイルが再設定されます。

6.2.11.3 I/O ウィンドウをクローズした場合の処理

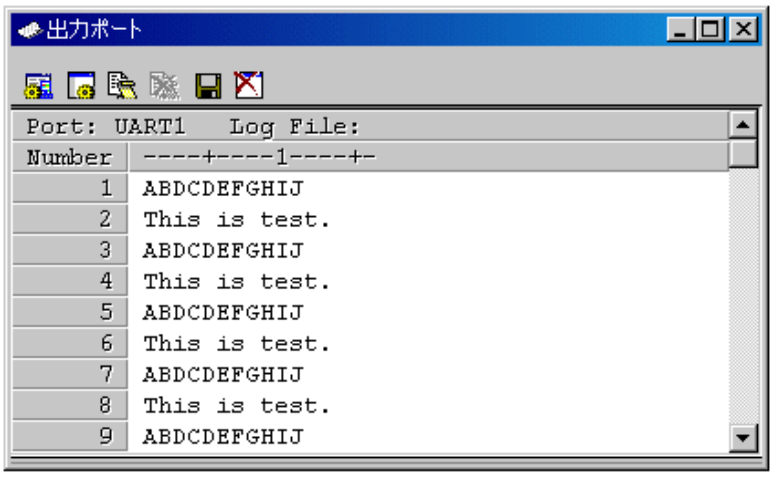
I/O ウィンドウをクローズした場合、設定した仮想ポート入力、仮想割り込み、及び I/O スクリプトファイルは評価されません。設定が削除された状態と同じです。

6.3 出力ポートウィンドウ

出力ポートウィンドウは、ポートに出力されるデータをウィンドウへ表示、およびファイルへ出力するウィンドウです。また、Printf 関数で UART へ出力されたデータを確認することができます。
740 用デバッガでは、サポートしていません。

出力データのウィンドウへの表示、およびファイルへの出力には、以下のフォーマットが指定できます。

出力先	フォーマット
ウィンドウ	ASCII 表示
	16 進数表示
ファイル	ASCII 出力
	16 進数出力
	バイナリ出力（ただし、Printf 関数データの出力時は除く）



- 出力ポートには、任意のポート、および Printf 関数の出力先である UART0 または UART1 を選択できます。 なお、Printf 関数の出力先は、お使いのルネサス製 C コンパイラ NCxx のユーザーズマニュアルをご参照ください。
- ポートへの出力データは、あらかじめ指定したファイル(ログファイル)に保存することができます。
- 出力ポートウィンドウは、最新 10000 バイト分の実行結果を保存するバッファを持っています。ログファイルの指定を忘れた場合でもポートへの出力データを、ファイル(ビューファイル)に保存することができます。
- 出力データは、ターゲットプログラムを停止したタイミングで、ウィンドウへ表示、およびファイルへ出力します。

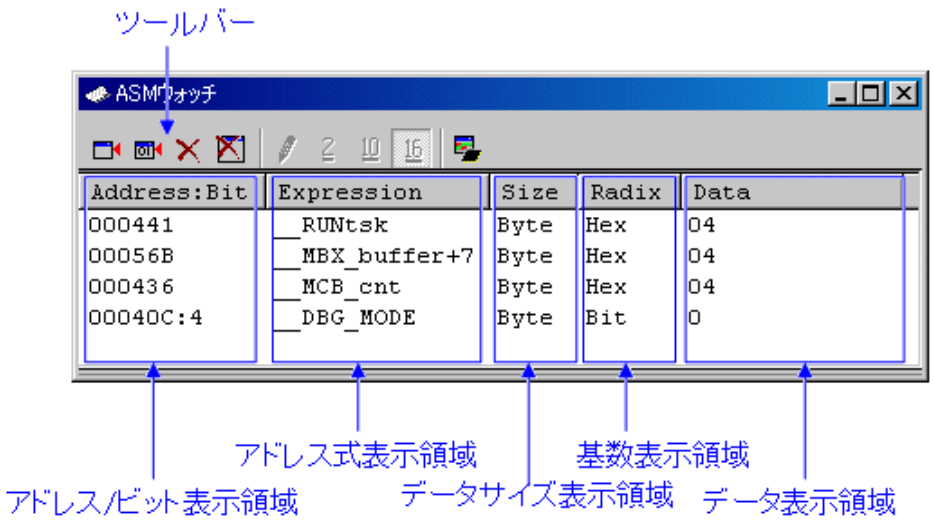
6.3.1 オプションメニュー

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
ポート設定...		出力ポートを設定します。
カラム設定...		カラムを設定します。
ログ	開始...	ログファイルをオープンします（出力開始）。
	終了	ログファイルをクローズします（出力終了）。
表示	保存...	表示内容を保存します。
	消去	表示内容を消去します。
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。

6.4 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。

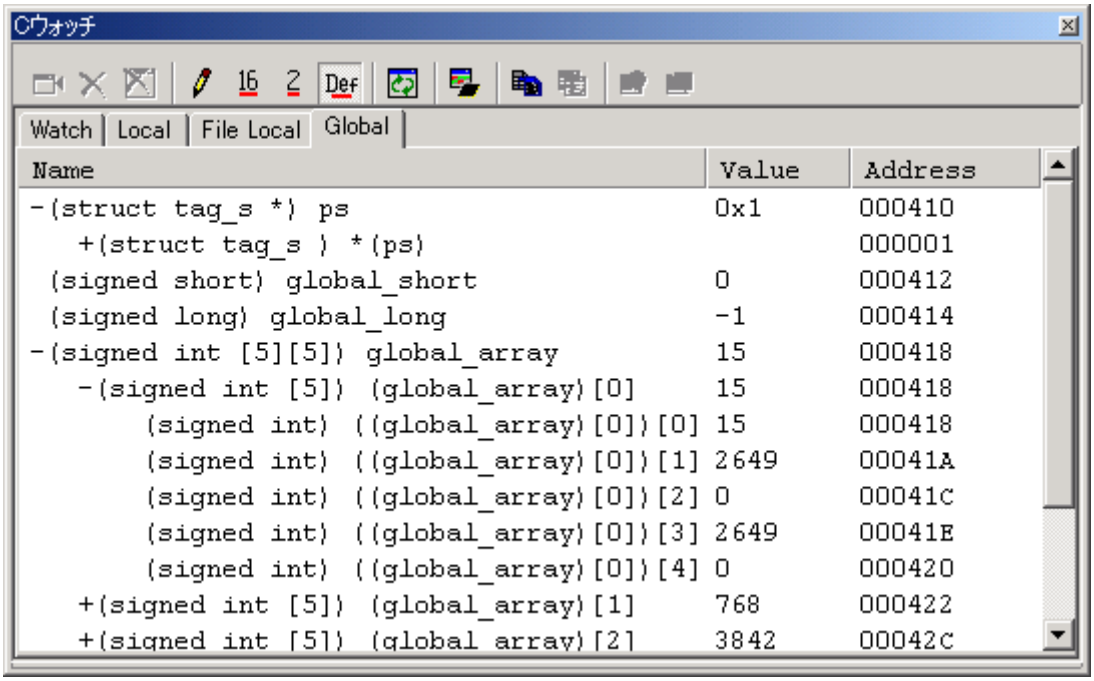
6.4.1 オプションメニュー

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。

6.5 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。
  - ・デフォルトの表示基数を変更できます。
  - ・16 進数で表示する場合、上位桁の 0 の表示/非表示を選択できます。
- 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。
  - ・参照先を停止時点のスコープ([Auto])、グローバル([Global])、各ファイル内のスタティックから選択することができます。
- 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。
- 指定した変数のアドレスに、RAM モニタを配置することができます。

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。
  - ・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。

その他、C変数の扱いについては、「12.1.3 C変数の参照・設定」を参照してください。

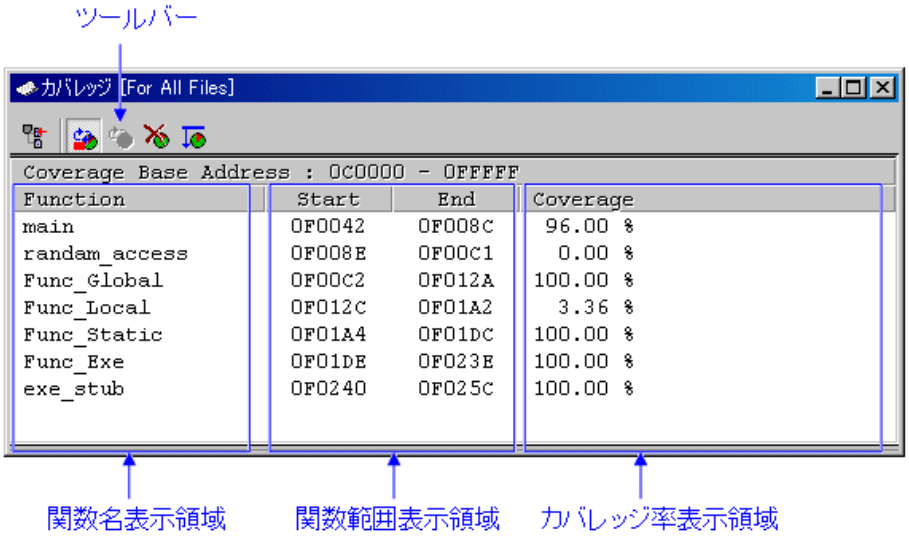
6.5.1 オプションメニュー

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。
全て削除		全ての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)。
	初期表示の設定...	初期表示基数を設定します。
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。
16 進表示のゼロサプレス		16 進表示時にゼロサプレスします。
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。
	RAM モニタ領域をこの変数に設定	RAM モニタ領域をこの変数に設定します。
	記録開始...	値更新の記録を開始します。
	記録終了	値更新の記録を終了します。
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。
コピー		選択されたアイテムをクリップボードにコピーします。
全てコピー		シート内の全アイテムをクリップボードにコピーします。
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。

6.6 カバレッジウィンドウ

カバレッジウィンドウは、各関数の開始アドレス/終了アドレスとカバレッジ計測結果を参照するためのウィンドウです。 計測可能なカバレッジは、C0 カバレッジです。 ソース行単位の実行/未実行を確認するには、エディタウィンドウを使用します。



- 全メモリ空間がカバレッジ計測領域になります。
- 関数の任意の行をダブルクリックすることにより、該当する関数をエディタ(ソース)ウィンドウに表示します。
- カバレッジ計測中は、カバレッジ表示領域が"-%"と表示されます。
- 関数名表示領域/関数範囲表示領域間は、表示割合をマウスで変更することができます。

6.6.1 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
ソース選択...		カバレッジ計測結果を表示するソースファイルを選択します。
最新の情報に自動更新		カバレッジ計測結果の表示をターゲット停止時に自動更新します。
最新の情報に更新		カバレッジ計測結果の表示を更新します。
初期化		カバレッジ計測結果を初期化します。
ベース...*		カバレッジ計測領域を変更します。
ファイル	保存...	カバレッジ計測結果をファイルに保存します。
	読み込み...	カバレッジ計測結果をファイルから読込みます。
レイアウト	アドレス	アドレス表示領域の表示/非表示を切り換えます。
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。

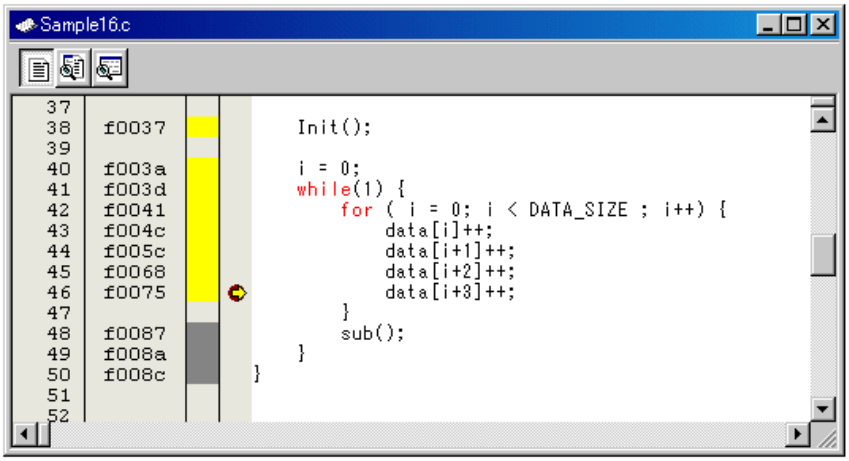
*:シミュレータデバッガでは、全メモリ空間がカバレッジ計測領域ですので選択できません。

6.6.2 実行したソース行/アドレスを参照する

カバレッジ計測結果は、エディタ(ソース)ウィンドウとメモリウィンドウで参照できます。

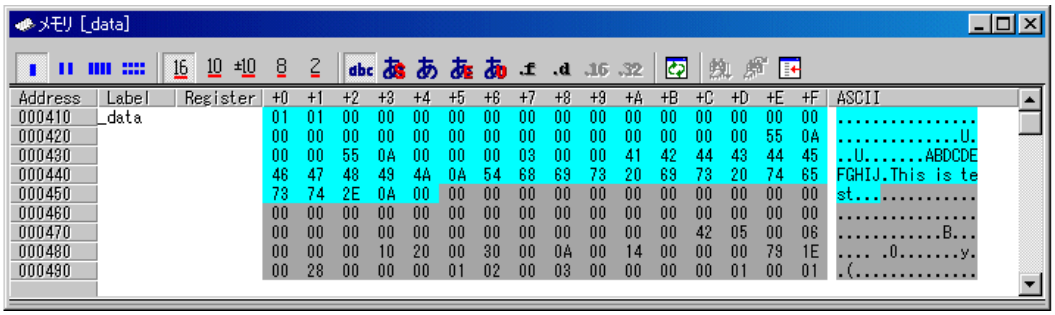
6.6.2.1 エディタ(ソース)ウィンドウで参照する

エディタ(ソース)ウィンドウは、カバレッジ計測表示がデフォルトで無効になっています。
有効にするには、メニュー[編集]→[表示カラムの設定...]で開くダイアログで、 [カバレッジ]チェックボックスをオンにしてください。 全てのエディタ(ソース)ウィンドウに、カバレッジ計測表示用のカラムが表示されます。
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]→[カバレッジ]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。



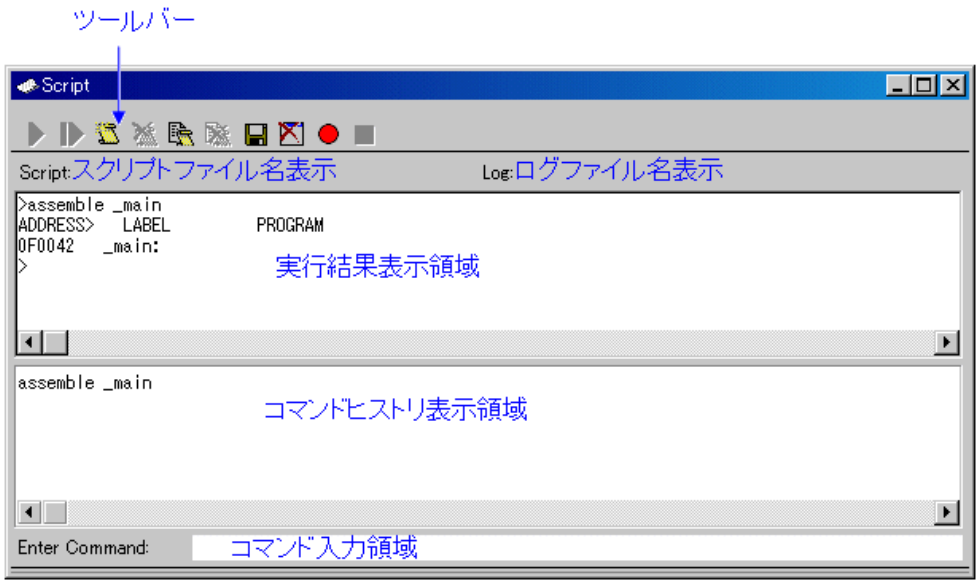
6.6.2.2 メモリウィンドウで参照する

メモリウィンドウは、カバレッジ計測表示がデフォルトで無効になっています。
有効にするには、メモリウィンドウのポップアップメニュー[カバレッジ]→[有効]を選択してください。



6.7 スクリプトウィンドウ

スクリプトウィンドウは、スクリプトコマンドを実行するためのウィンドウです。
スクリプトコマンドは、ウィンドウ下部のコマンド入力領域から入力します。 コマンドの実行結果は、実行結果表示領域に表示します。
主要な操作は、ツールバーのボタンに割り付けています。



- 実行するスクリプトコマンドをあらかじめファイル(スクリプトファイル)に記述することにより、一括実行することができます。
- スクリプトコマンドの実行結果は、あらかじめ指定したファイル(ログファイル)に保存することができます。
- スクリプトウィンドウは、最新 1000 行分の実行結果を保存したバッファ(ビューバッファ)を持っています。 ログファイルの指定を忘れた場合でもスクリプトコマンドの実行結果をファイル(ビューファイル)に保存することができます。
- 実行するコマンドは、あらかじめ指定したファイルに保存することができます(スクリプトファイルとして再使用できます)。

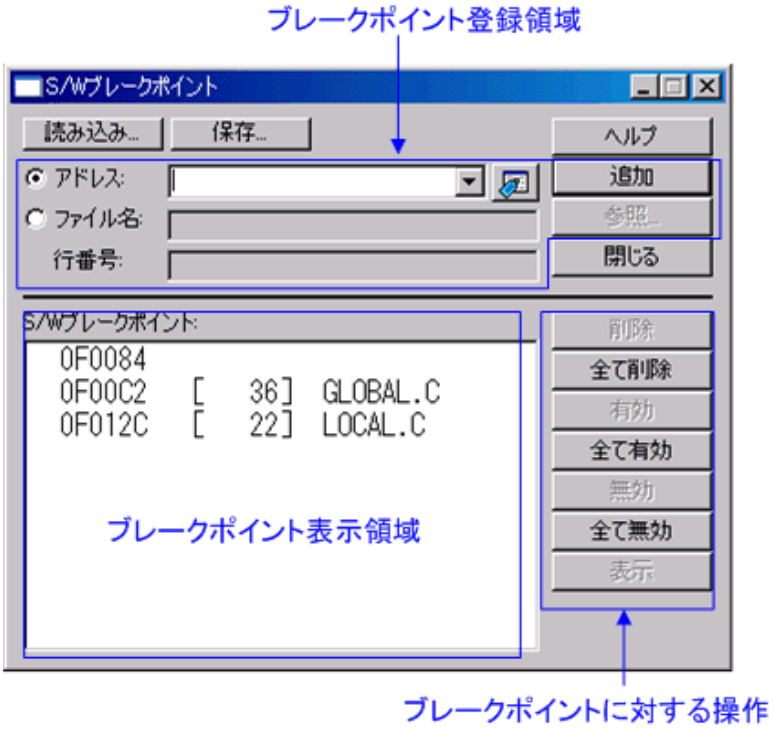
6.7.1 オプションメニュー

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
スクリプト	開く...	スクリプトファイルを開きます。
	実行	スクリプトファイルを実行します。
	ステップ実行	スクリプトファイルをステップ実行します。
	閉じる	スクリプトファイルを閉じます。
表示	保存...	実行結果表示をファイルに保存します。
	消去	実行結果表示を消去します。
ログ	開始...	ログファイルを開き出力を開始します。
	停止	出力を終了しログファイルを閉じます。
コマンドの記録	開始...	コマンドのファイルへの記録を開始します。
	停止	コマンドのファイルへの記録を停止します。
コピー		選択範囲をコピーしてクリップボードに保存します。
貼り付け		クリップボードの内容を貼り付けます。
切り取り		選択範囲を切り取ってクリップボードに保存します。
削除		選択範囲を消去します。
元に戻す		直前に行った操作を元に戻します。
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。

6.8 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。

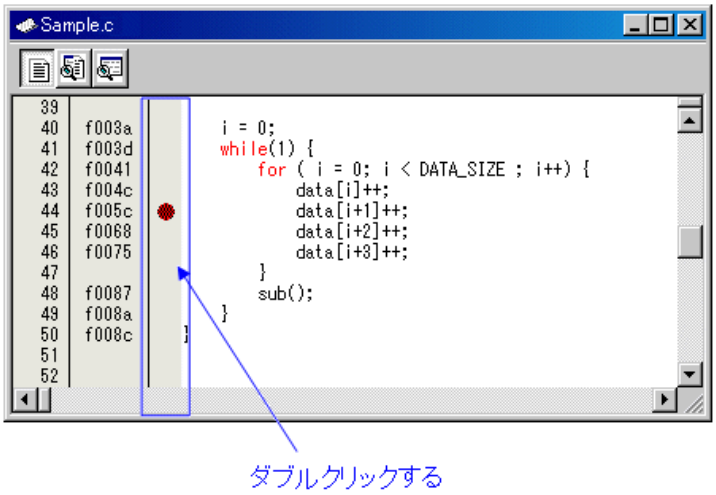
6.8.1 コマンドボタン

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	ソフトウェアブレー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ー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ー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ー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ー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ー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ー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ー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。

6.8.2 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

製品によっては、ソフトウェアブレークポイントに設定できる領域が異なります。
詳細は、「12.1.2 ソフトウェアブレークポイントが設定可能な領域」を参照してください。
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)。

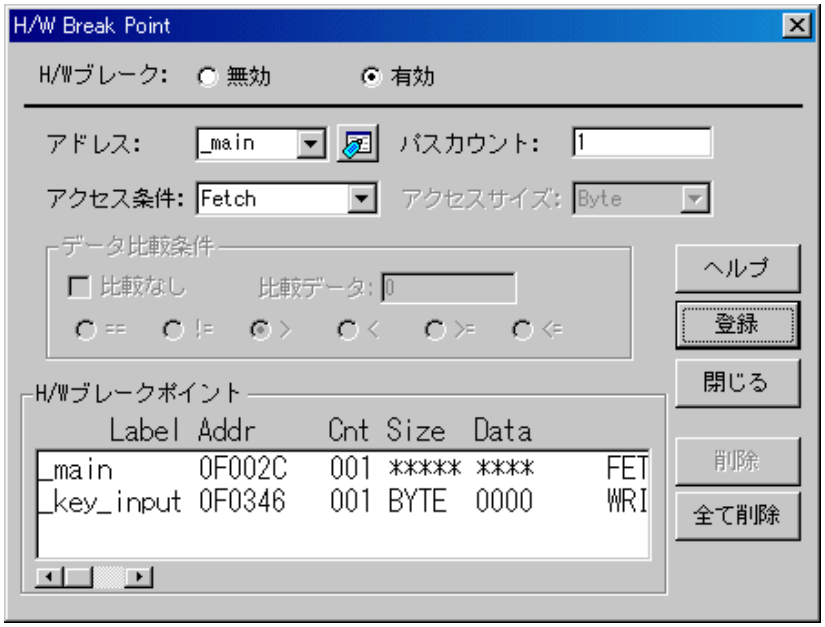


もう一度ダブルクリックするとブレークポイントの設定解除となります(赤丸の表示が消えます)。

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。

6.9 H/W ブレークポイント設定ダイアログ

H/W ブレークポイント設定ダイアログは、ハードウェアブレークポイントを設定するためのダイアログです。

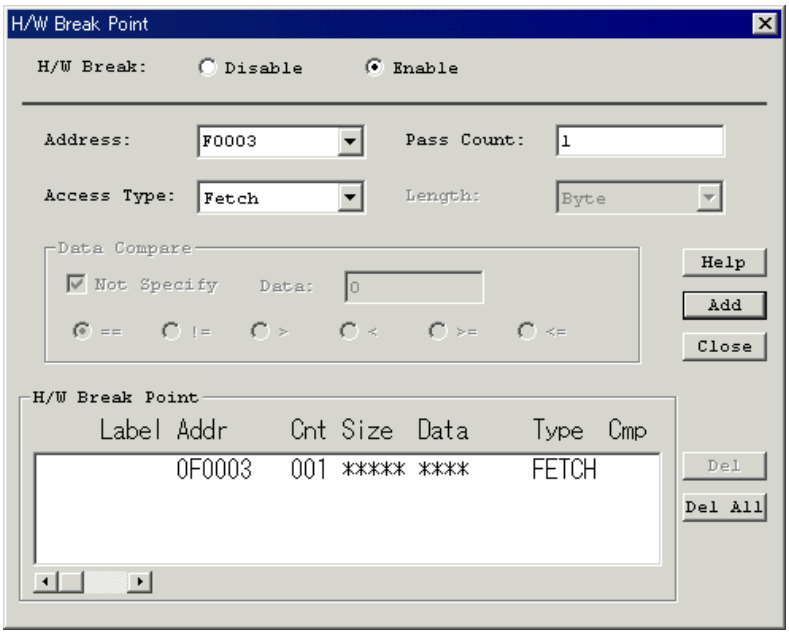


- 64 点のハードウェアブレークポイントが設定できます。パスカウントを指定することもできます。
- ハードウェアブレークポイントのアクセス条件には、命令フェッチ(Fetch)、メモリアクセス(Write,Read,R/W)が指定できます。
- ハードウェアブレークポイントに読み込み/書き込みされるデータが特定の値であればブレークするといった指定も可能です。
- ブレークポイントを複数設定した場合、いずれか 1 点の ブレークポイントに到達するとターゲットプログラムを停止します(OR 条件)。

6.9.1 イベントを設定する

6.9.1.1 命令フェッチ

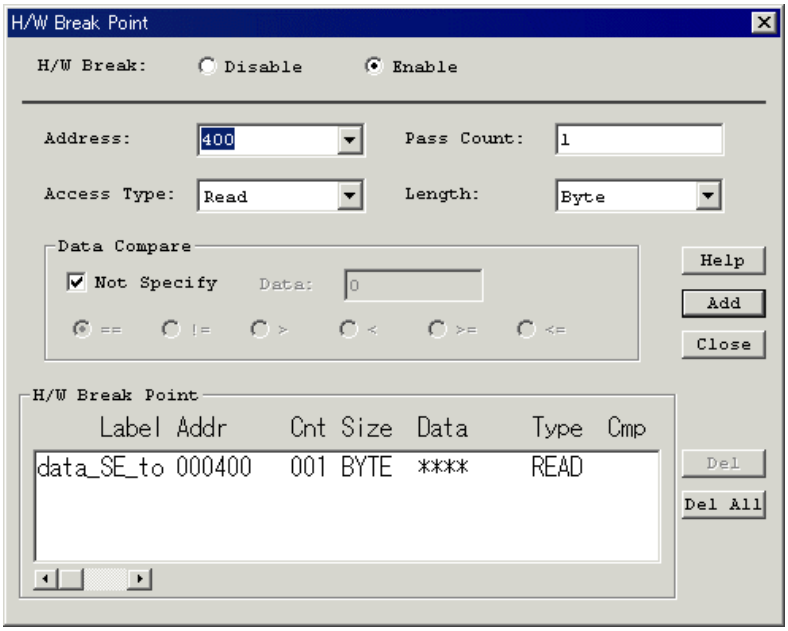
以下のように設定します。
例)アドレス F0003h の命令実行



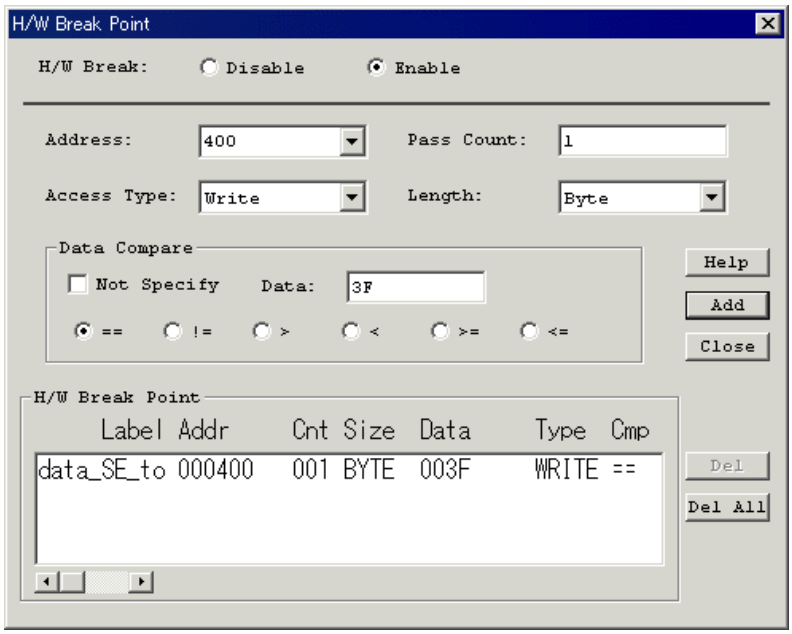
Add ボタンをクリックすると、ダイアログ下部のブレークポイント一覧に設定したブレークポイントが追加されます。
ハードウェアブレークポイントの設定完了後は、Close ボタンをクリックして下さい。

6.9.1.2 指定アドレスへアクセス

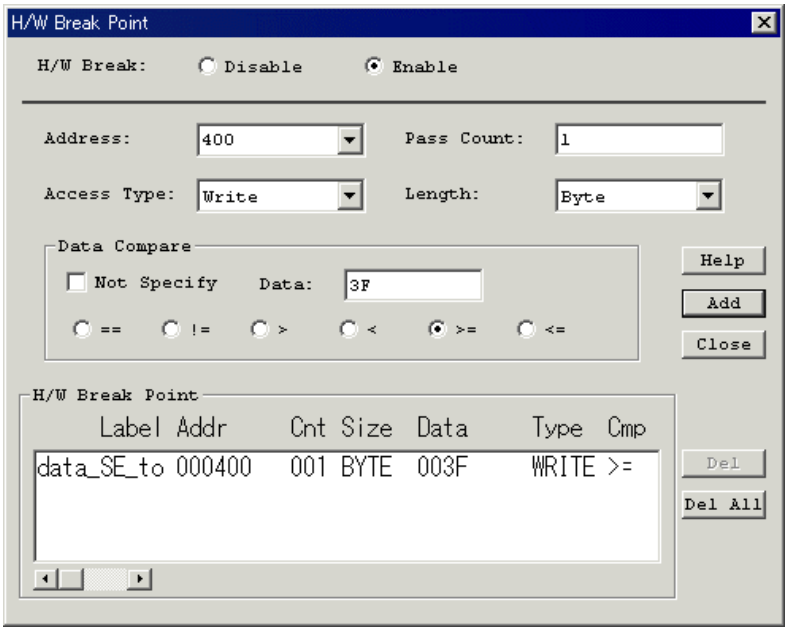
以下のように設定します。
例)アドレス 400h の読み込み



例)アドレス 400h へデータ 3Fh の書き込み

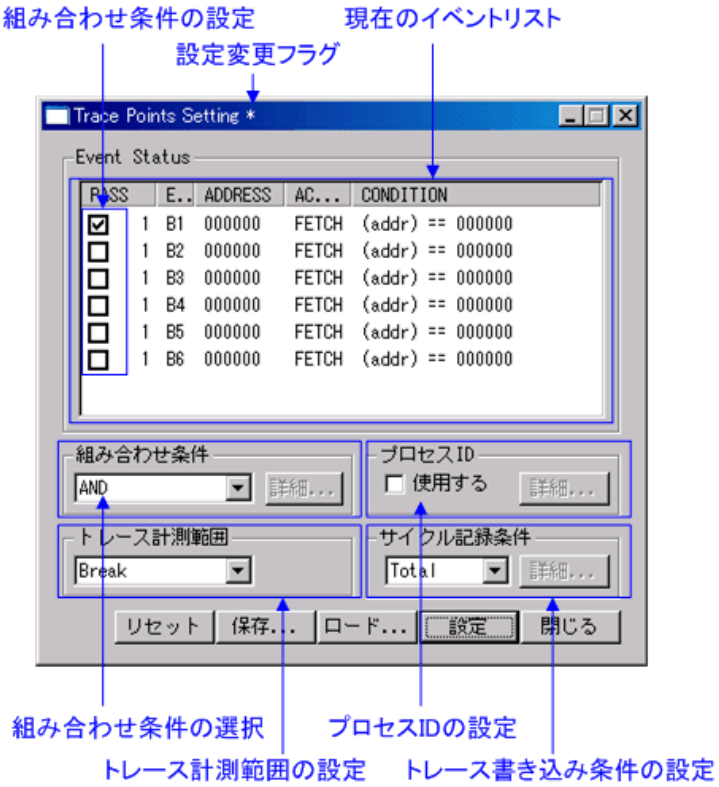


例)アドレス 400h へデータ 3Fh 以上の書き込み



6.10 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。
740 用デバッガでは、サポートしていません。



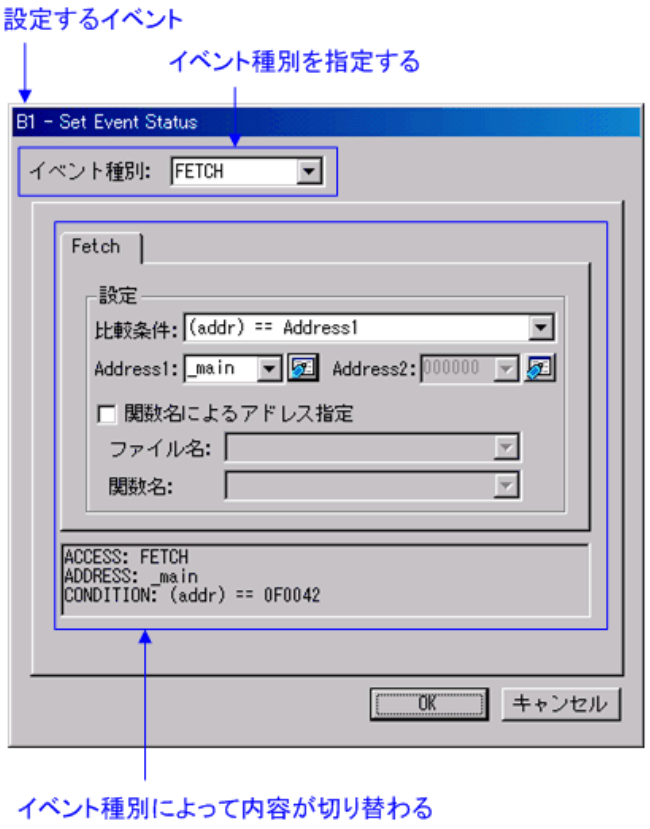
- トレースイベントとして、以下のイベントが指定できます。イベントの内容を変更するとタイトルバーに "*"を表示します。シミュレータへの設定後、"*"は表示しません。

  命令フェッチ、メモリアクセス、ビットアクセス、割り込み

- 6 点のイベントが使用できます。
- 複数のイベントは、以下の方法で組み合わせて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)
  - ・状態遷移でトレースステートに遷移した場合にトレース(State Transition 条件)

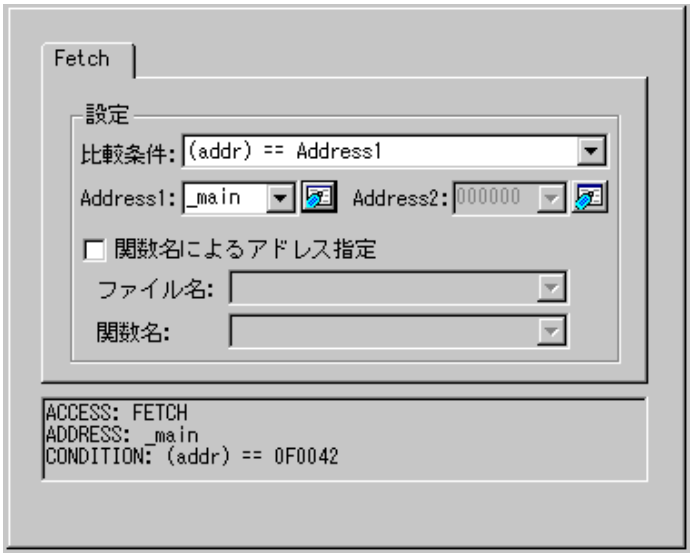
6.10.1 トレースイベント指定

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。



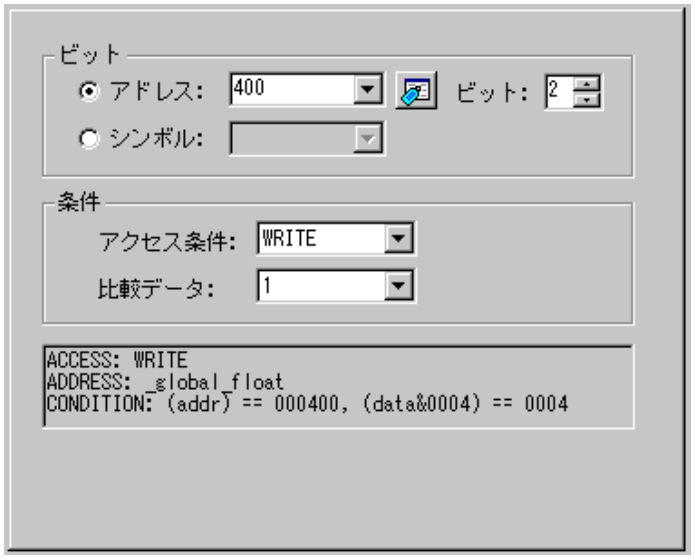
[イベント種別]の指定により、以下のイベントが設定できます。

- FETCH を選択した場合
  命令フェッチでトレースします。

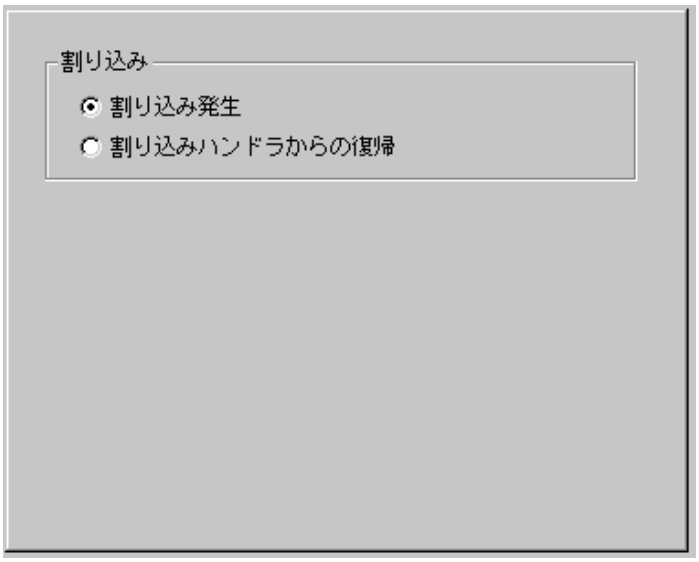


- DATA ACCESS を選択した場合
  メモリアクセスでトレースさせることができます。

- 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。



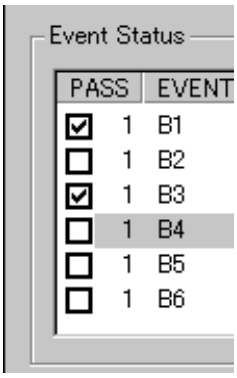
- INTERRUPT を選択した場合
  割り込み発生/割り込み終了時にトレースさせることができます。



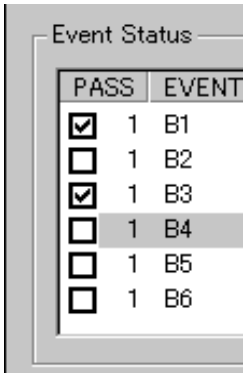
6.10.2 組み合わせ条件指定

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。

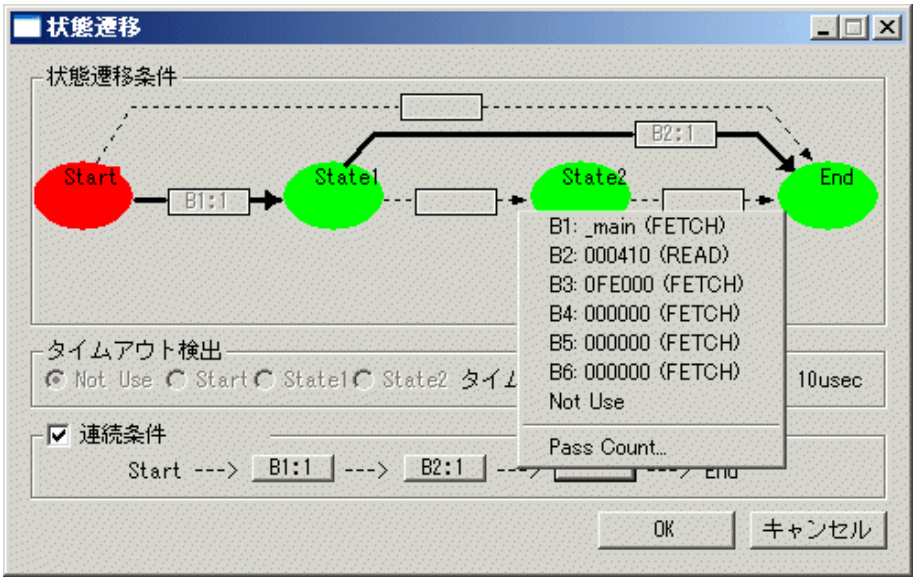
- AND,OR を選択した場合
  イベント指定領域で使用するイベントとそのパスカウントが指定できます。 パスカウントを変更するには、変更するイベントを選択した状態でそのイベントのパスカウント値をクリックしてください。



- 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。



- State Transition を選択した場合
  [詳細...]ボタンをクリックすると以下のウィンドウがオープンします。 シーケンシャルイベントによる指定ができます。 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。シミュレータへの設定後、"*"は表示しません。



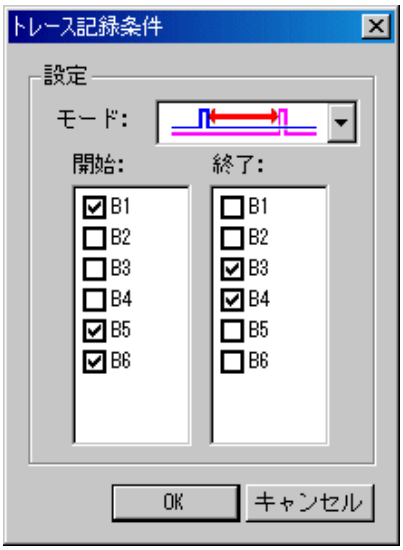
6.10.3 トレース範囲指定

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。
シミュレータをご使用の場合、Init ダイアログの[トレース]タブで指定したサイクル分を記録することができます（以下の記述は、32K サイクルを指定した場合)。

Break	ターゲットプログラムが停止するまでの 32K サイクルを記録します。
Before	トレース条件成立までの 32K サイクルを記録します。
About	トレース条件成立の前後 16K サイクルを記録します。
After	トレース条件成立後の 32K サイクルを記録します。
Full	トレース開始からの 32K サイクルを記録します。

6.10.4 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

6.10.5 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。

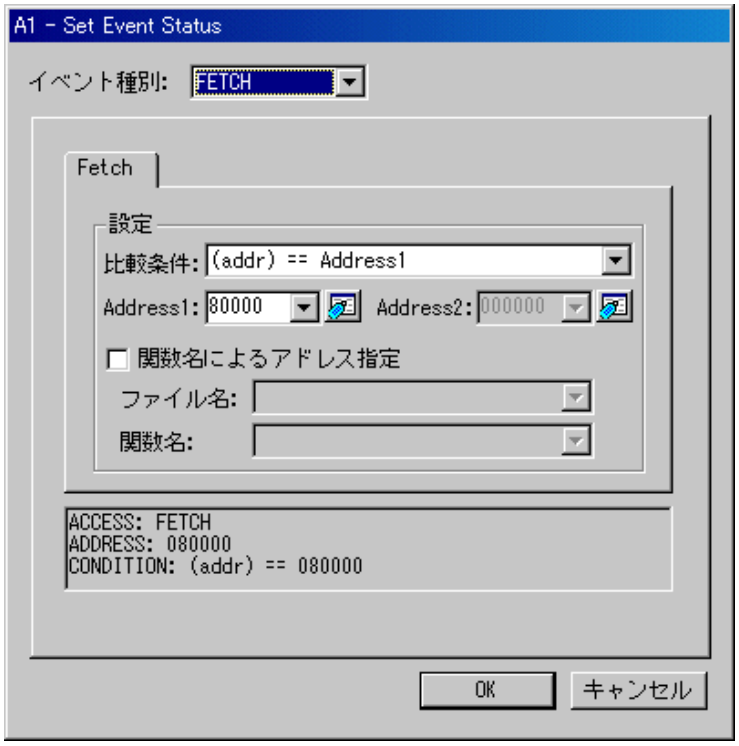
ボタン名	機能
リセット	ウィンドウに表示中の内容を破棄し、シミュレータに設定されている内容をロードします。
保存...	ウィンドウで設定した内容をファイルに保存します。
読込...	ファイルに保存したイベント情報を読み込みます。
設定	ウィンドウで設定した内容をシミュレータに送信します。
閉じる	ウィンドウを閉じます。

6.10.6 イベントを設定する（命令フェッチ）

命令フェッチのイベントを指定する場合は、イベント選択ダイアログの[イベント種別]を"FETCH"に 変更してください。 指定アドレス、または指定アドレス範囲の任意のアドレスがフェッチされた場合にイベントが成立します。

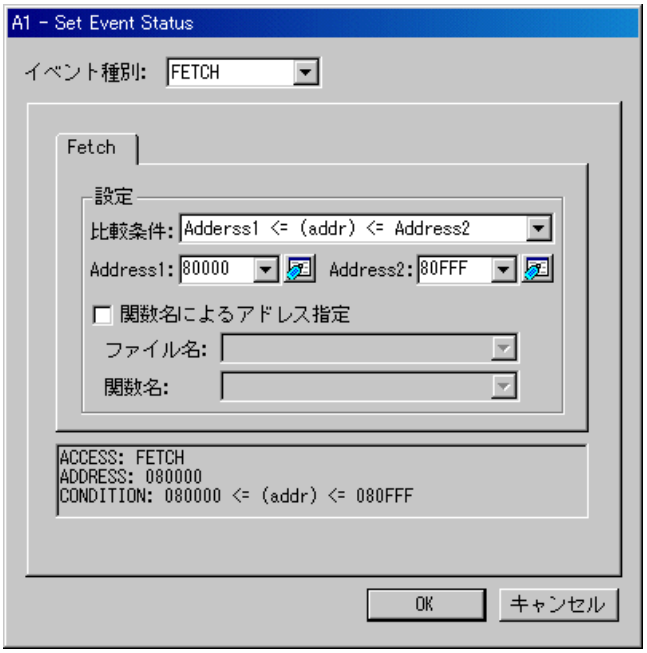
6.10.6.1 指定アドレスの命令フェッチ

以下のように設定します。
例)アドレス 80000h の命令実行



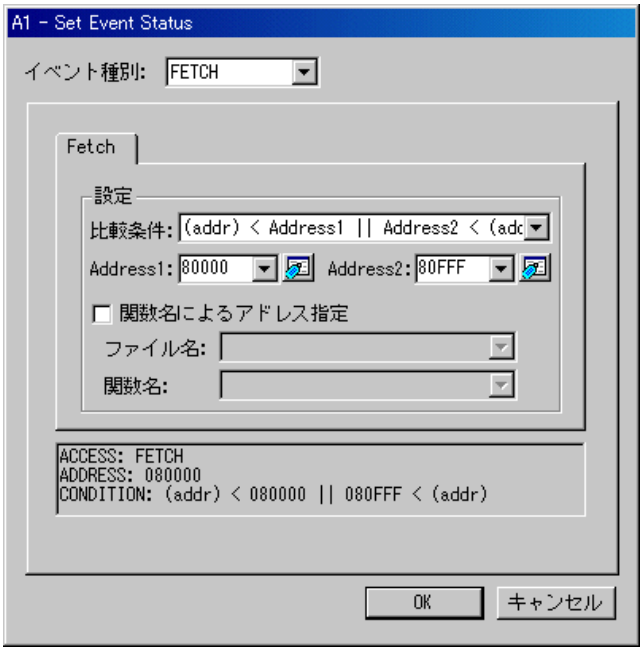
6.10.6.2 指定アドレス範囲内の命令フェッチ

以下のように設定します。
例)アドレス 80000h～80FFFh の命令実行



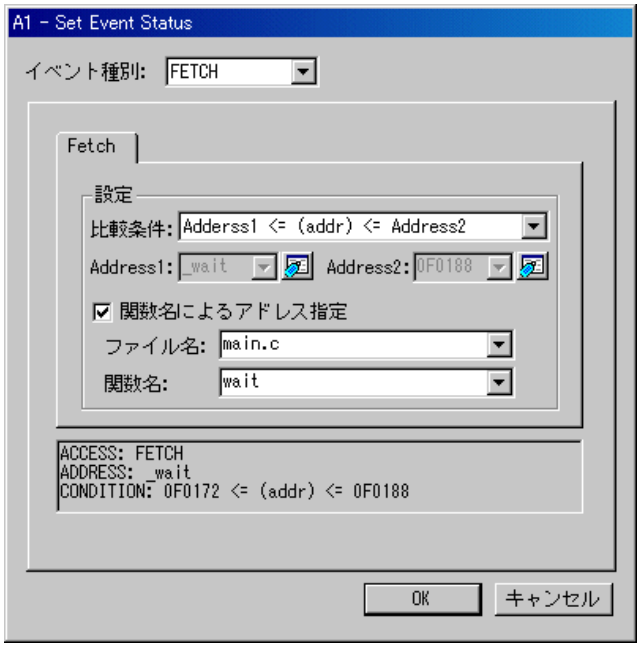
6.10.6.3 指定アドレス範囲外の命令フェッチ

以下のように設定します。
例)アドレス 80000h～80FFFh 以外の命令実行

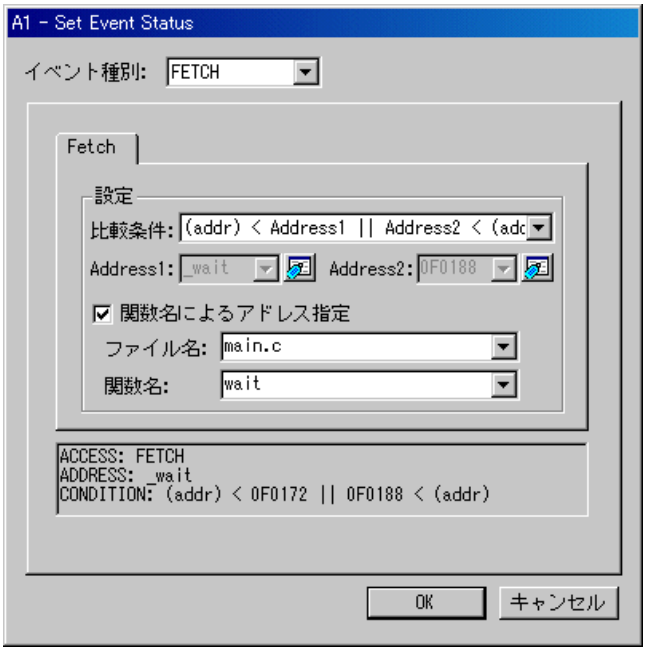


6.10.6.4 指定関数への突入/脱出

以下のように設定します。
例)関数名 wait への突入



例)関数名 wait からの脱出



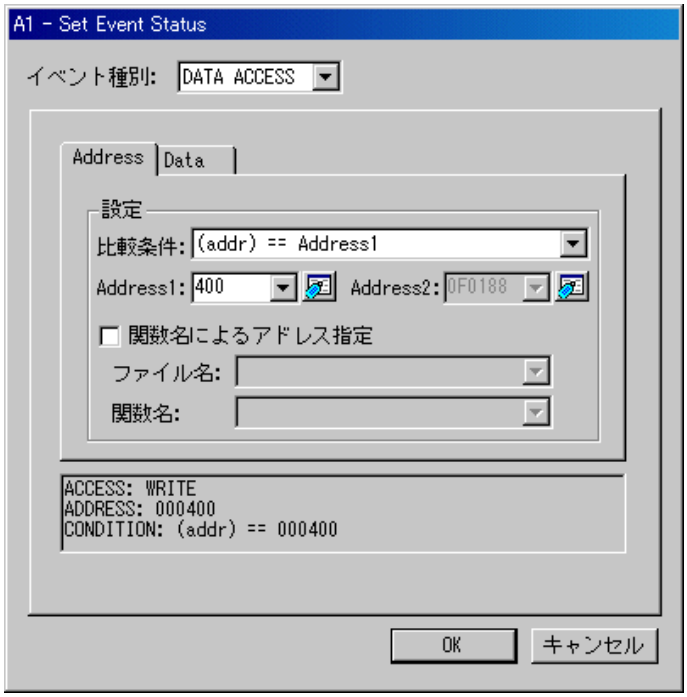
6.10.7 イベントを設定する（メモリアクセス）

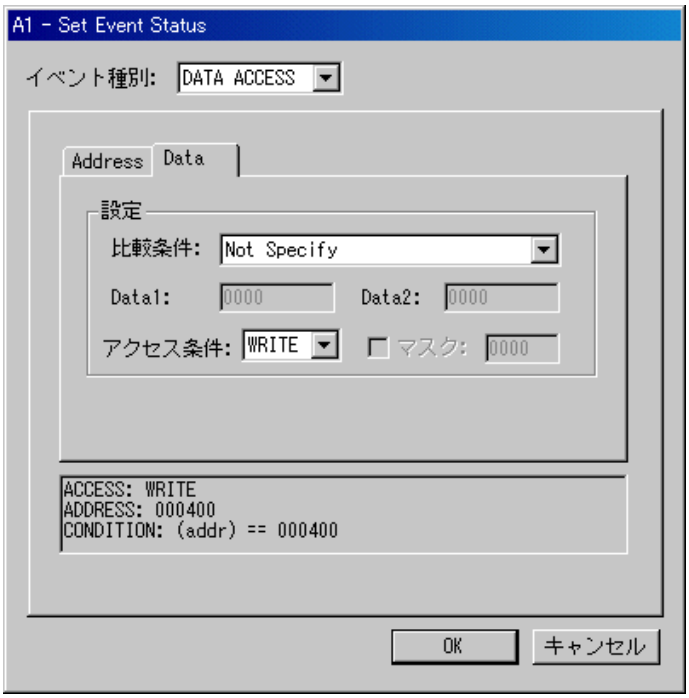
メモリアクセスのイベントを指定する場合は、イベント選択ダイアログの[イベント種別]を"DATA ACCESS"に 変更してください。 指定アドレス、または指定アドレス範囲に設定した条件でアクセスした場合にイベントが成立します。

6.10.7.1 メモリアクセス(R32C 用デバッガ)

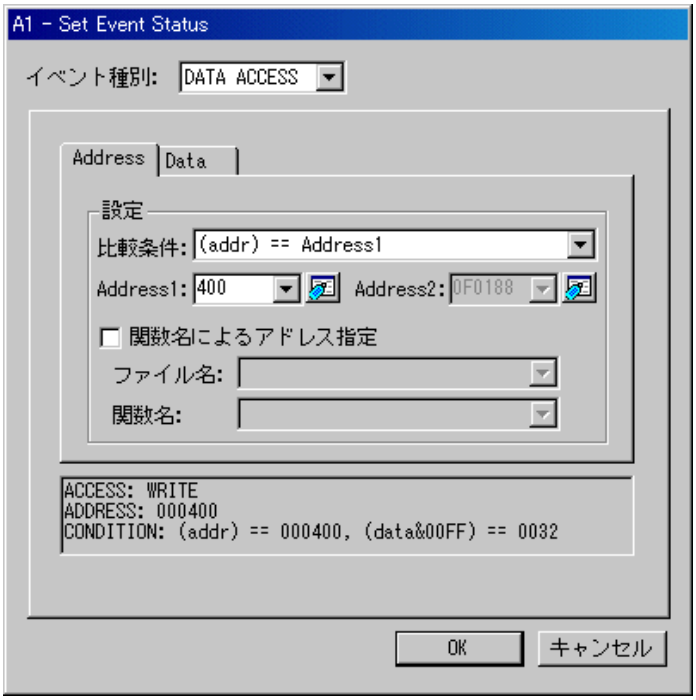
6.10.7.1.1. 指定アドレスへのデータ書き込み/読み込み

以下のように設定します。
例)偶数アドレス 400h へのデータ書き込み

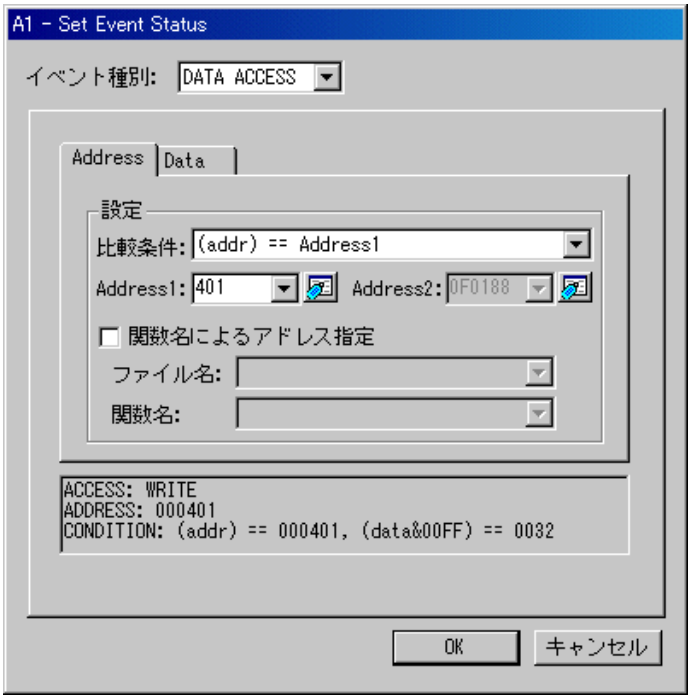




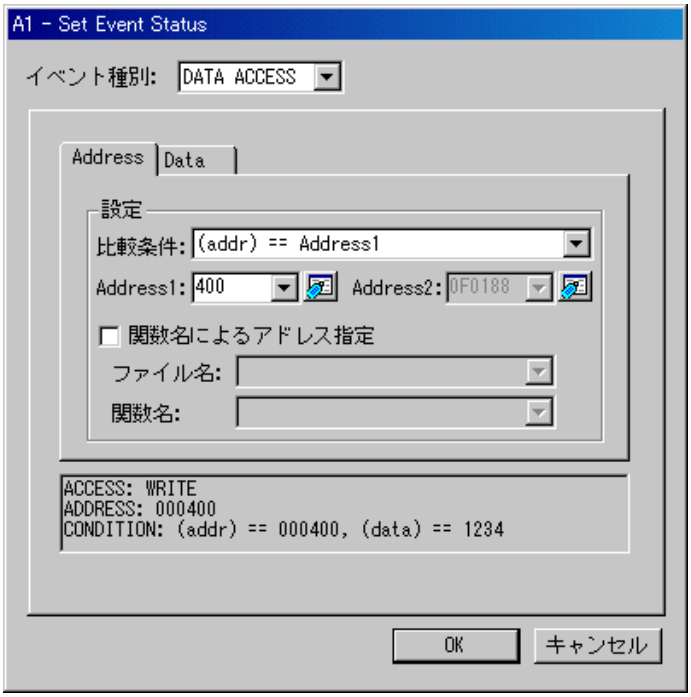
例)偶数アドレス 400h へのバイト長データ 32h の書き込み

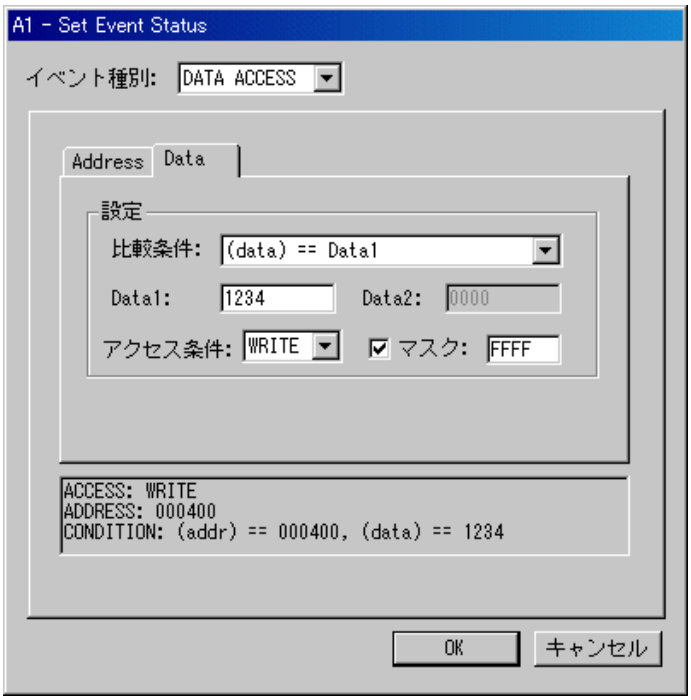


例)奇数アドレス 401h へのバイト長データ 32h の書き込み

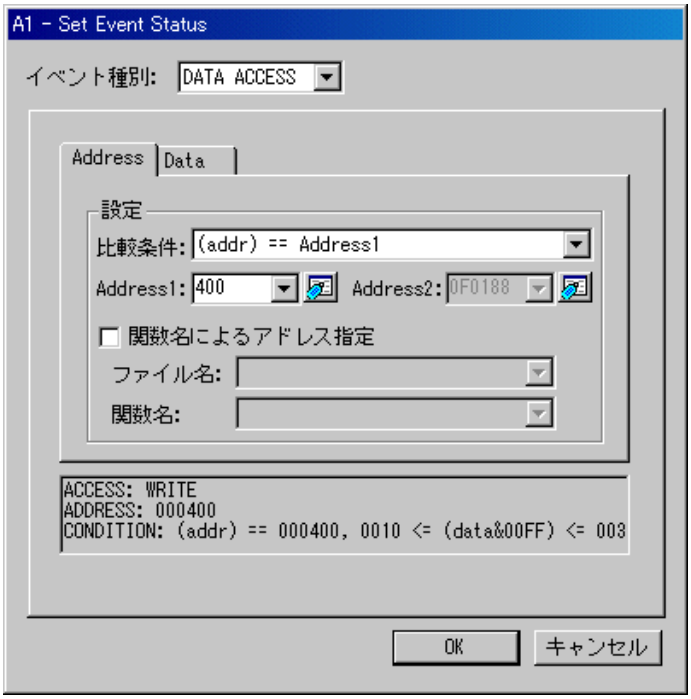


例)偶数アドレス 400h へのワード長データ 1234h の書き込み





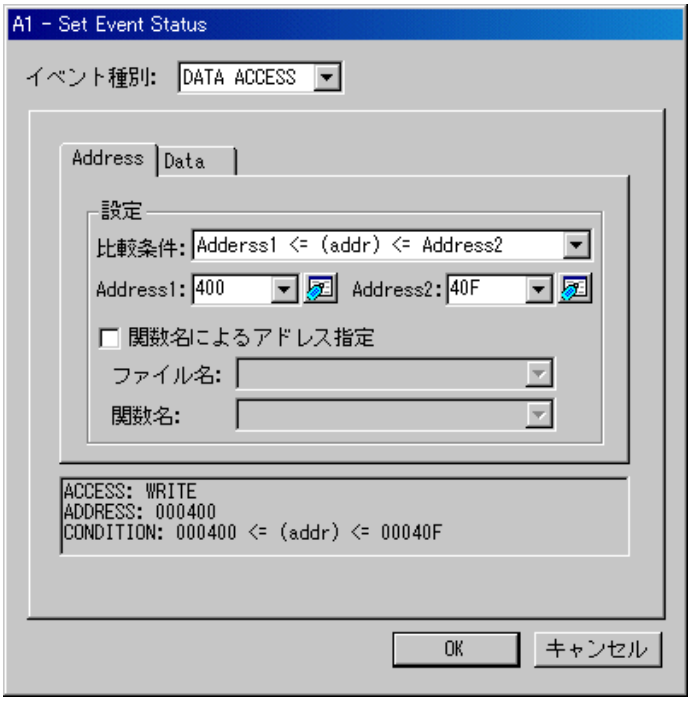
例)偶数アドレス 400h へのバイトデータ 10h～3Fh の書き込み

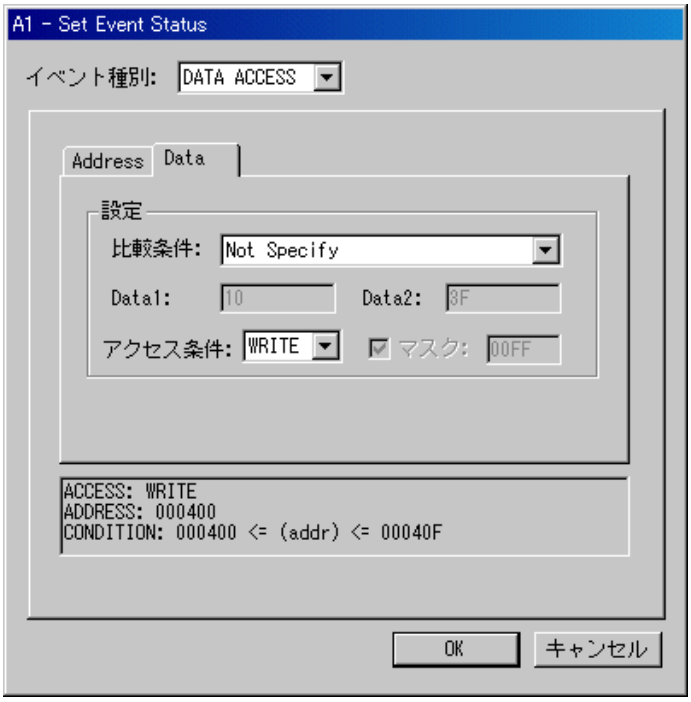


6.10.7.1.2. 指定アドレス範囲内へのデータ書き込み/読み込み

以下のように設定します。

例)アドレス 400h～40Fh へのデータ書き込み

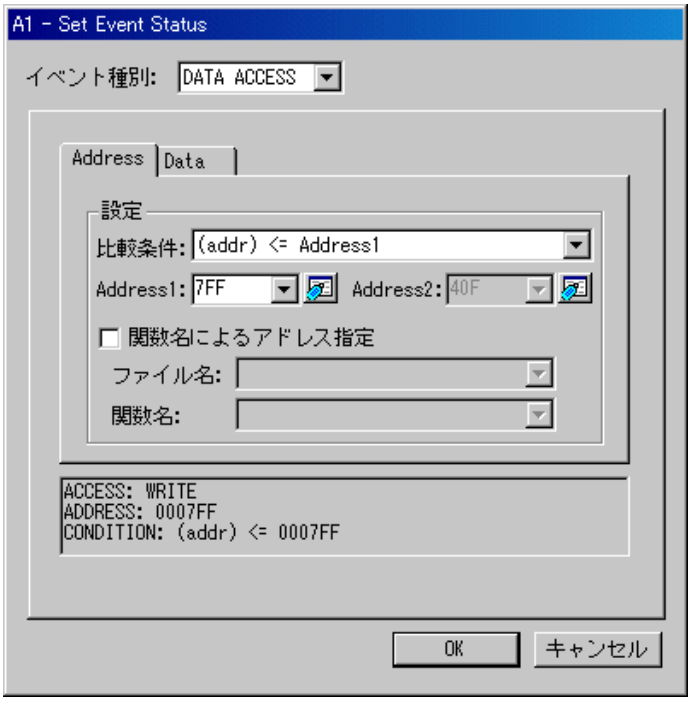




6.10.7.1.3. 指定アドレス範囲外へのデータ書き込み/読み込み

以下のように設定します。

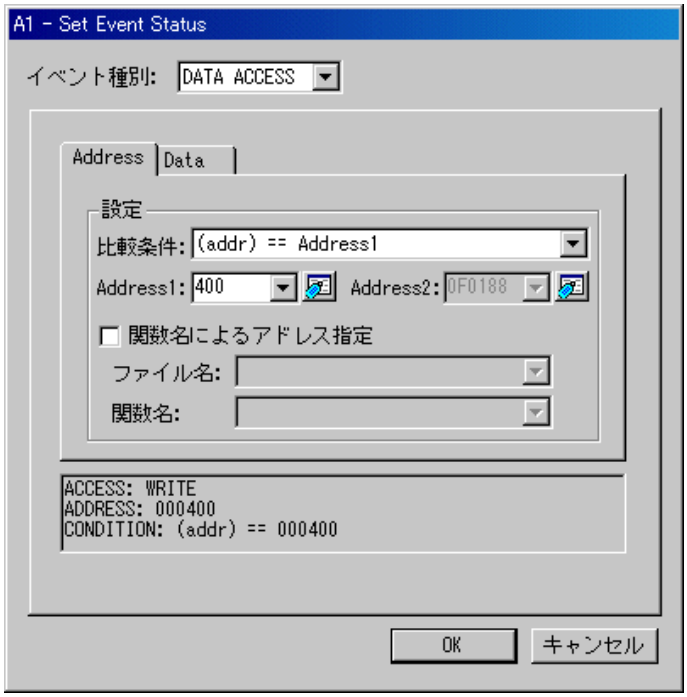
例)アドレス 7FFh 以下へのデータ書き込み

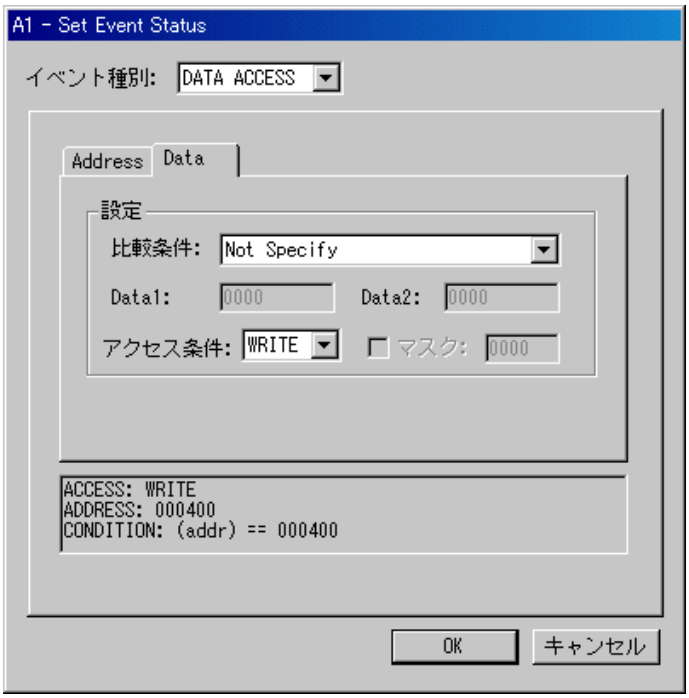


6.10.7.2 メモリアクセス(M32C 用デバッガ)

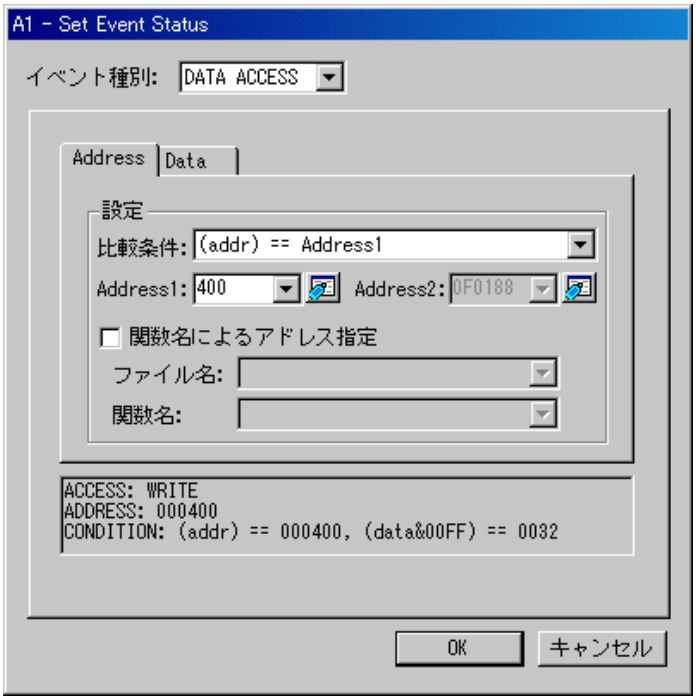
6.10.7.2.1. 指定アドレスへのデータ書き込み/読み込み

以下のように設定します。
例)偶数アドレス 400h へのデータ書き込み

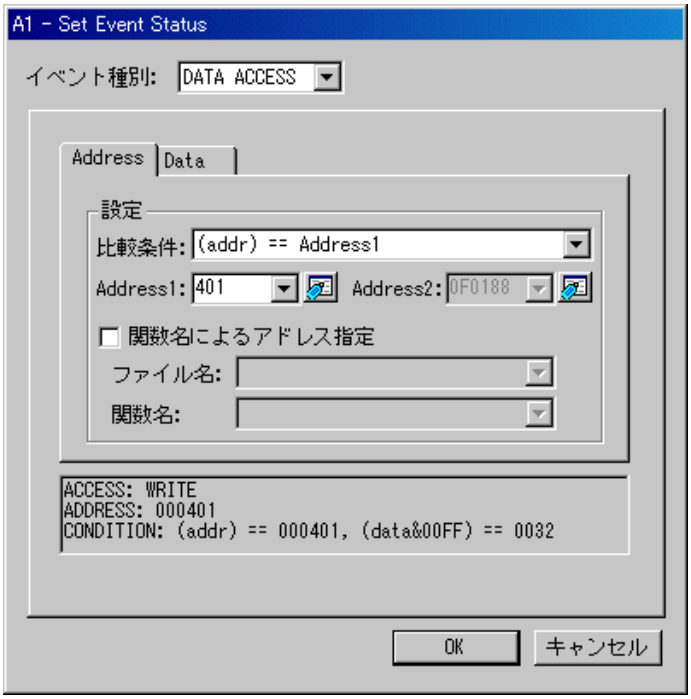




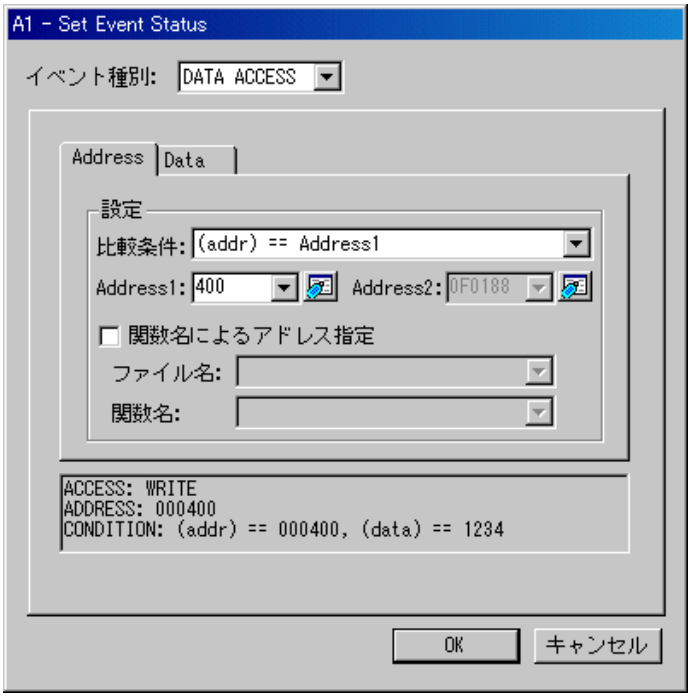
例)偶数アドレス 400h へのバイト長データ 32h の書き込み

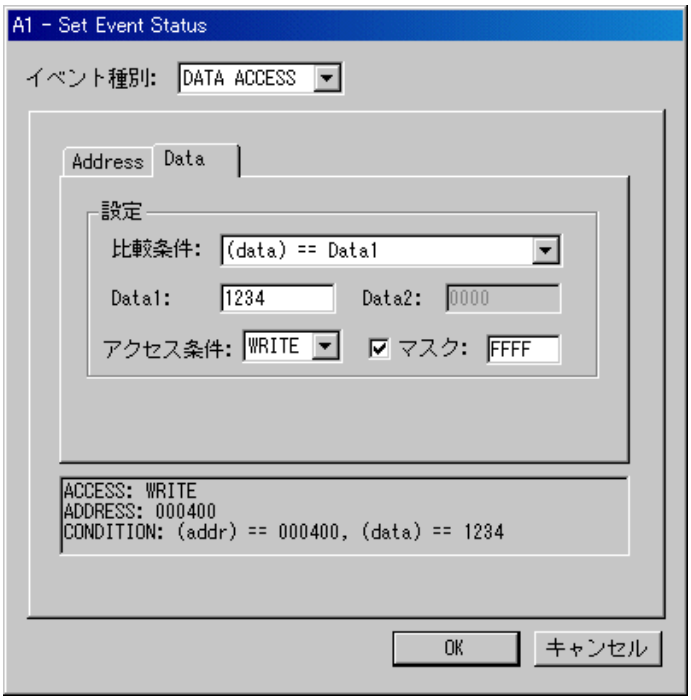


例)奇数アドレス 401h へのバイト長データ 32h の書き込み

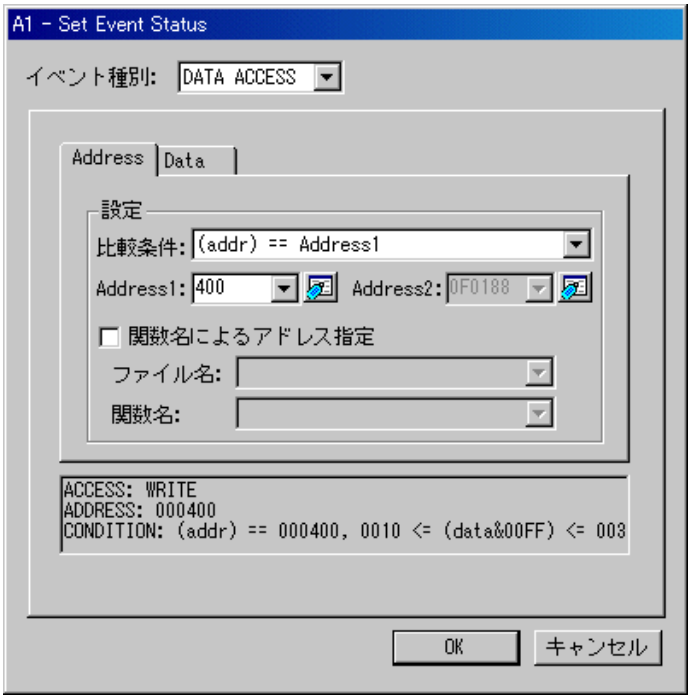


例)偶数アドレス 400h へのワード長データ 1234h の書き込み





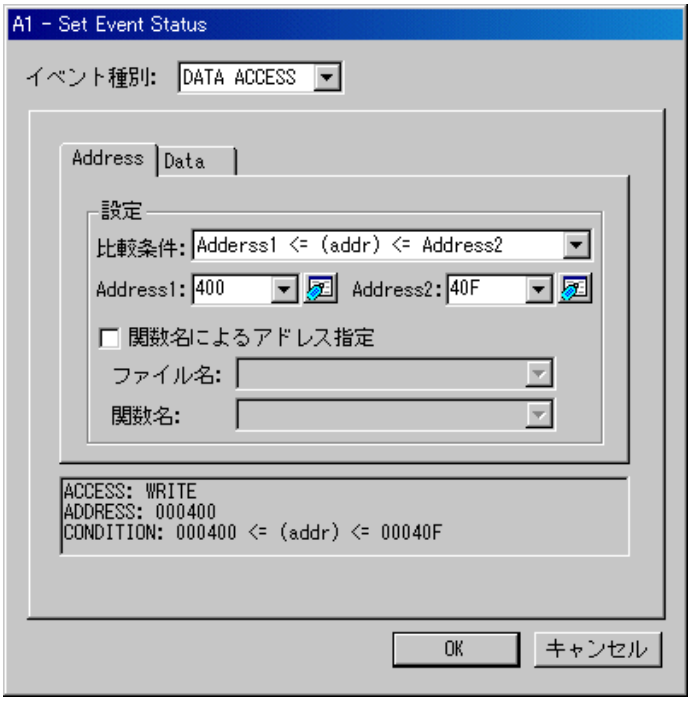
例)偶数アドレス 400h へのバイトデータ 10h～3Fh の書き込み

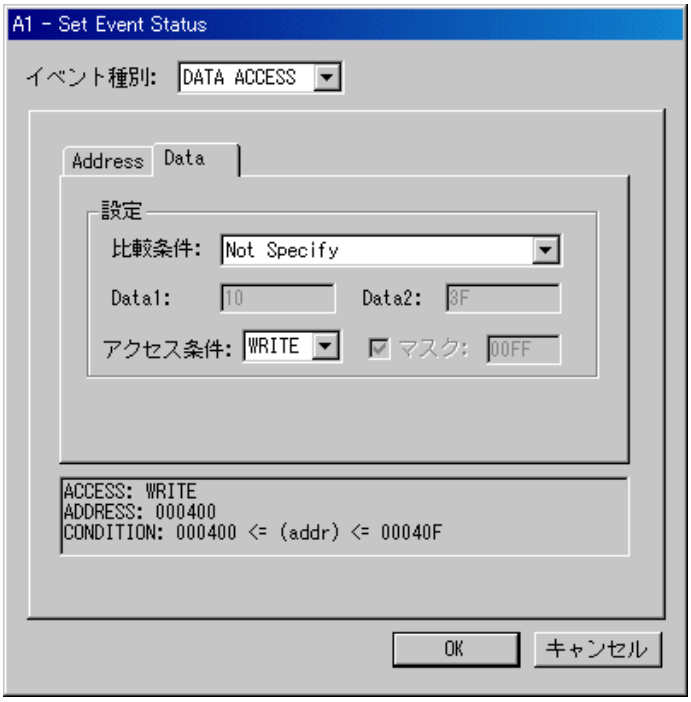


6.10.7.2.2. 指定アドレス範囲内へのデータ書き込み/読み込み

以下のように設定します。

例)アドレス 400h～40Fh へのデータ書き込み

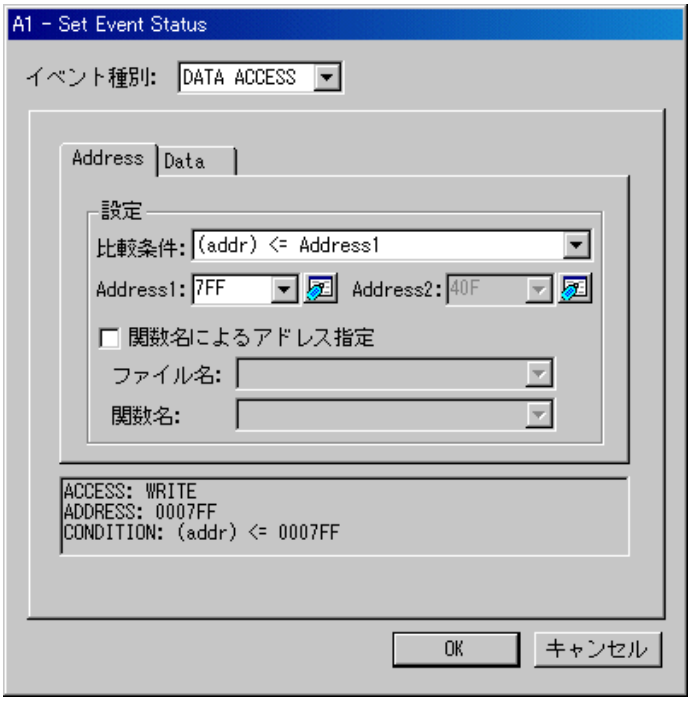




6.10.7.2.3. 指定アドレス範囲外へのデータ書き込み/読み込み

以下のように設定します。

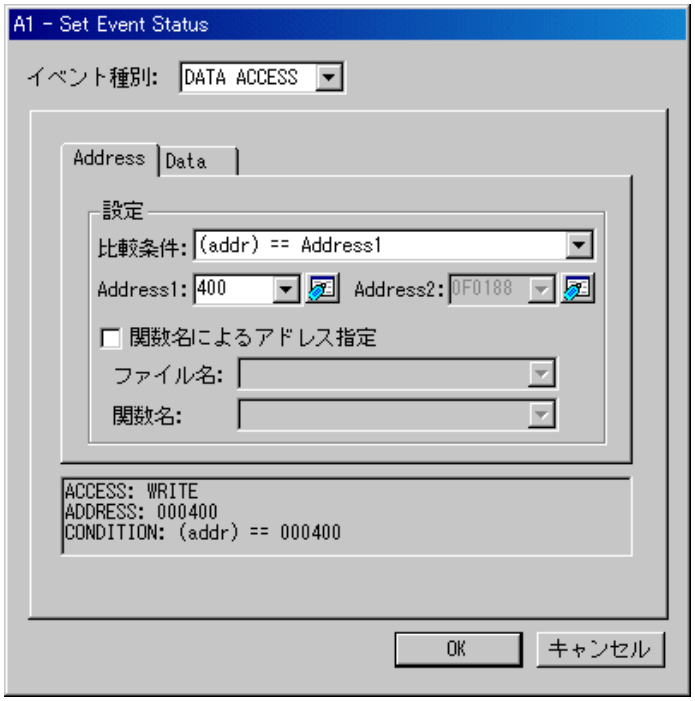
例)アドレス 7FFh 以下へのデータ書き込み



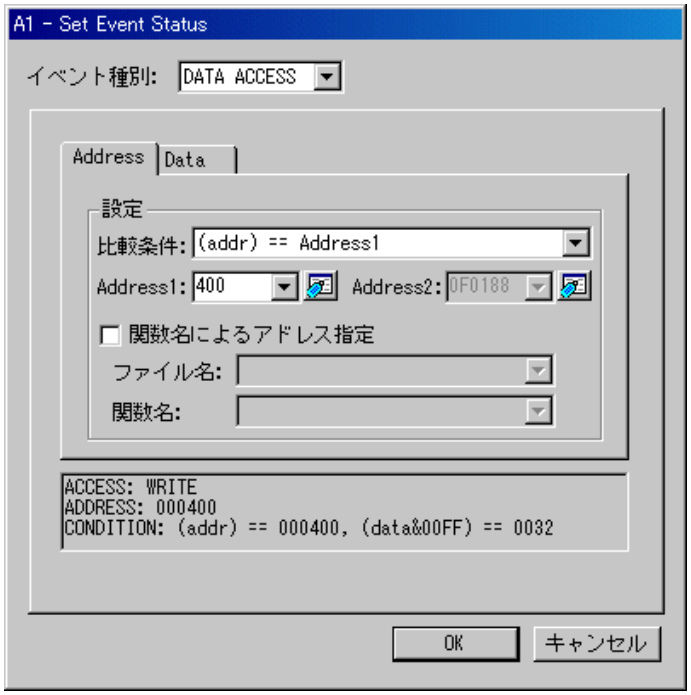
6.10.7.3 メモリアクセス(M16C/R8C 用デバッガ)

6.10.7.3.1. 指定アドレスへのデータ書き込み/読み込み

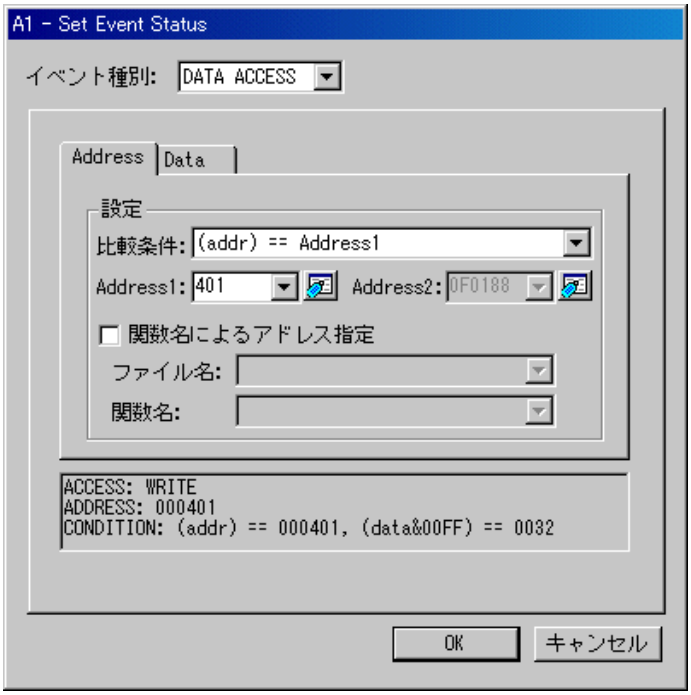
以下のように設定します。
例)偶数アドレス 400h へのデータ書き込み



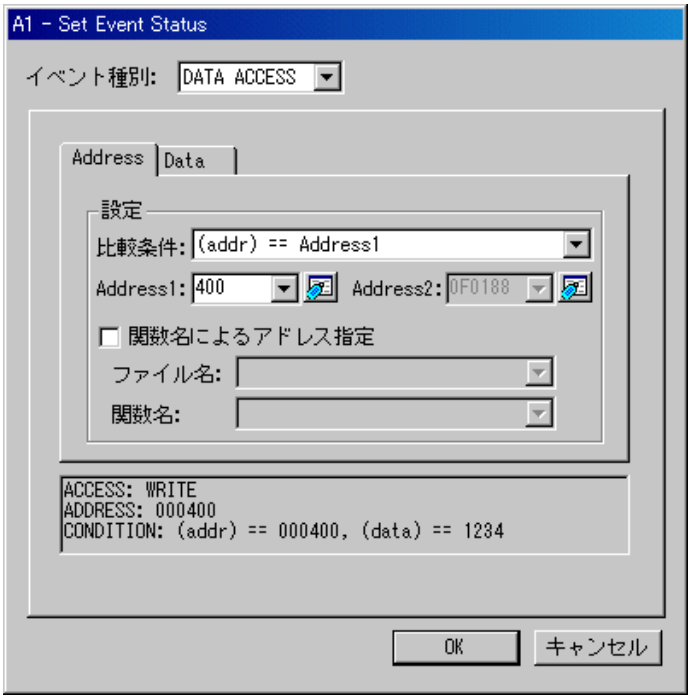
例)偶数アドレス 400h へのバイト長データ 32h の書き込み

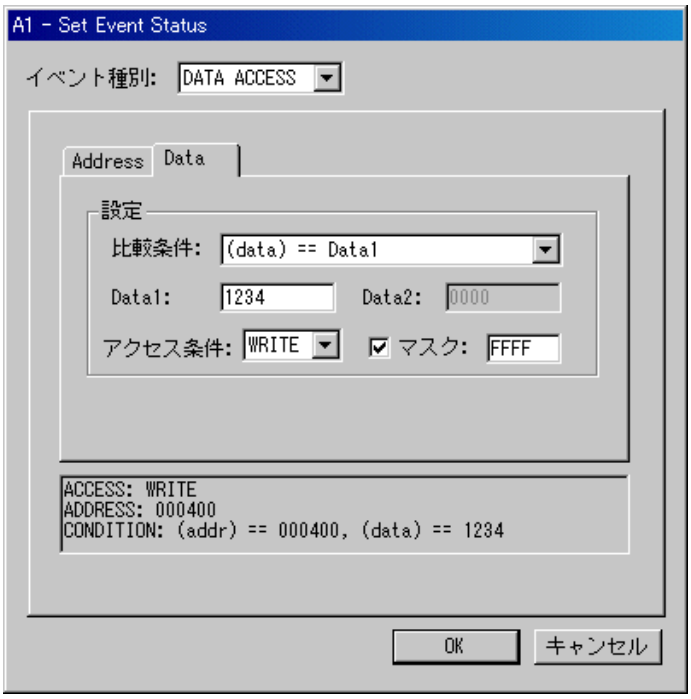


例)奇数アドレス 401h へのバイト長データ 32h の書き込み

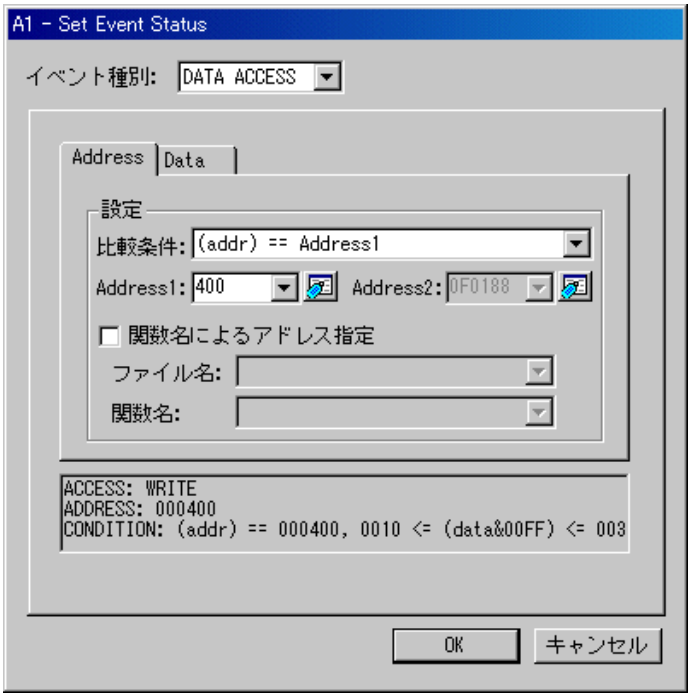


例)偶数アドレス 400h へのワード長データ 1234h の書き込み





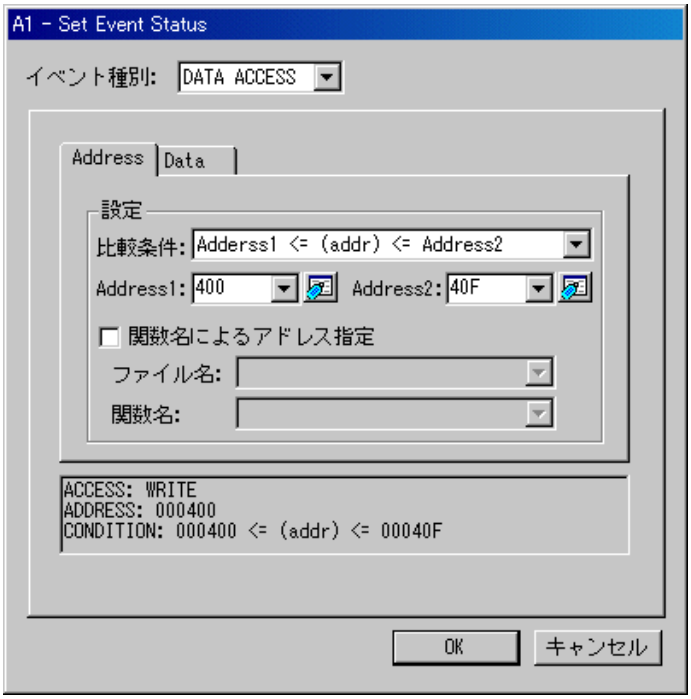
例)偶数アドレス 400h へのバイトデータ 10h～3Fh の書き込み

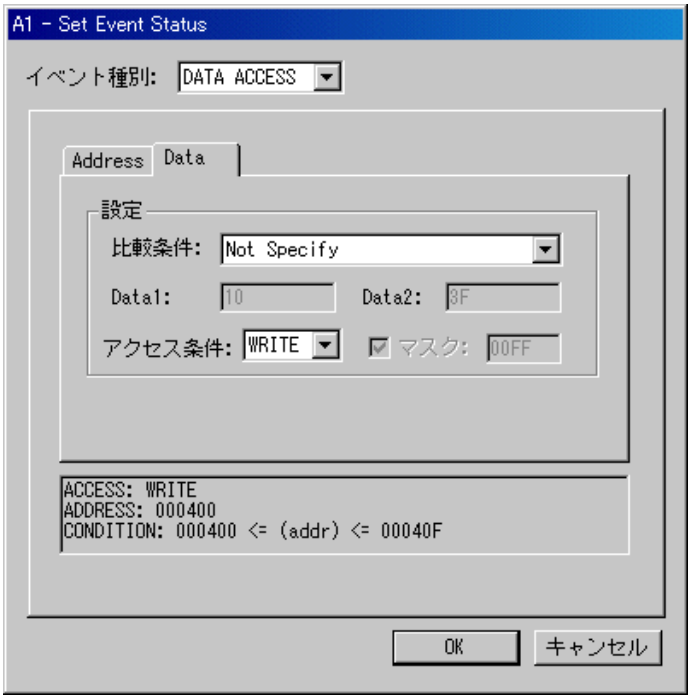


6.10.7.3.2. 指定アドレス範囲内へのデータ書き込み/読み込み

以下のように設定します。

例)アドレス 400h～40Fh へのデータ書き込み

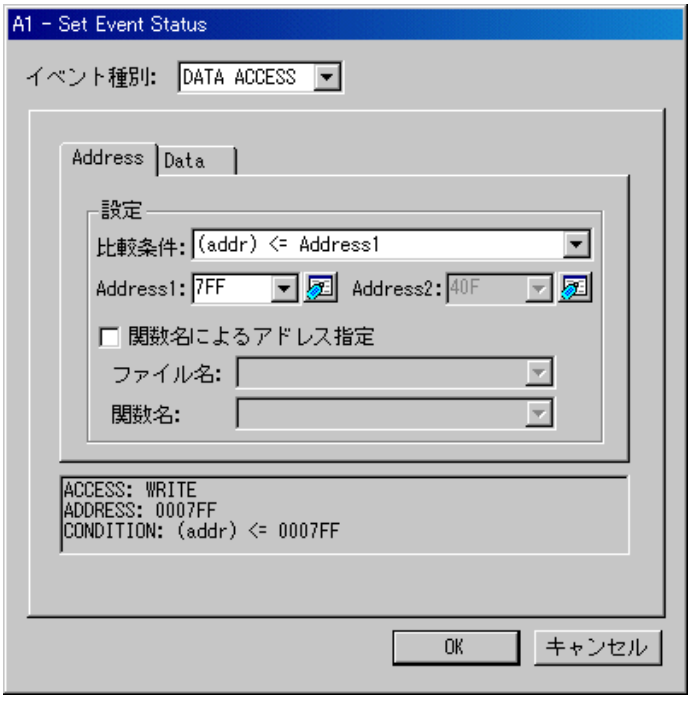


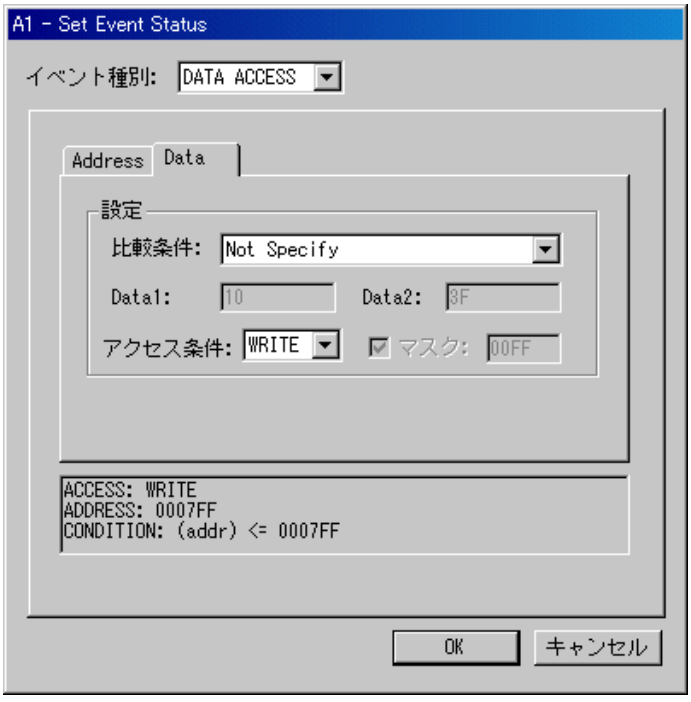


6.10.7.3.3. 指定アドレス範囲外へのデータ書き込み/読み込み

以下のように設定します。

例)アドレス 7FFh 以下へのデータ書き込み



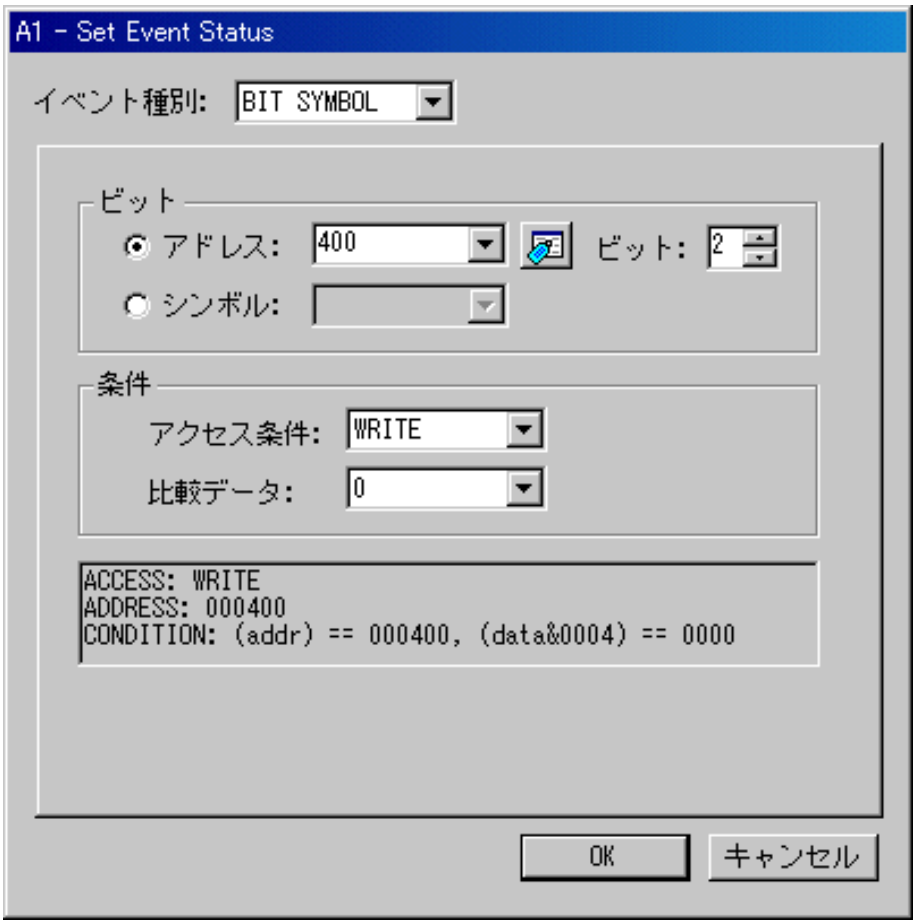


6.10.8 イベントを設定する（ビットアクセス）

ビットアクセスのイベントを指定する場合は、イベント選択ダイアログの[イベント種別]を"BIT SYMBOL"に 変更してください。 指定アドレスの指定ビットまたはビットシンボルに指定した条件でアクセスした場合にイベントが成立します。

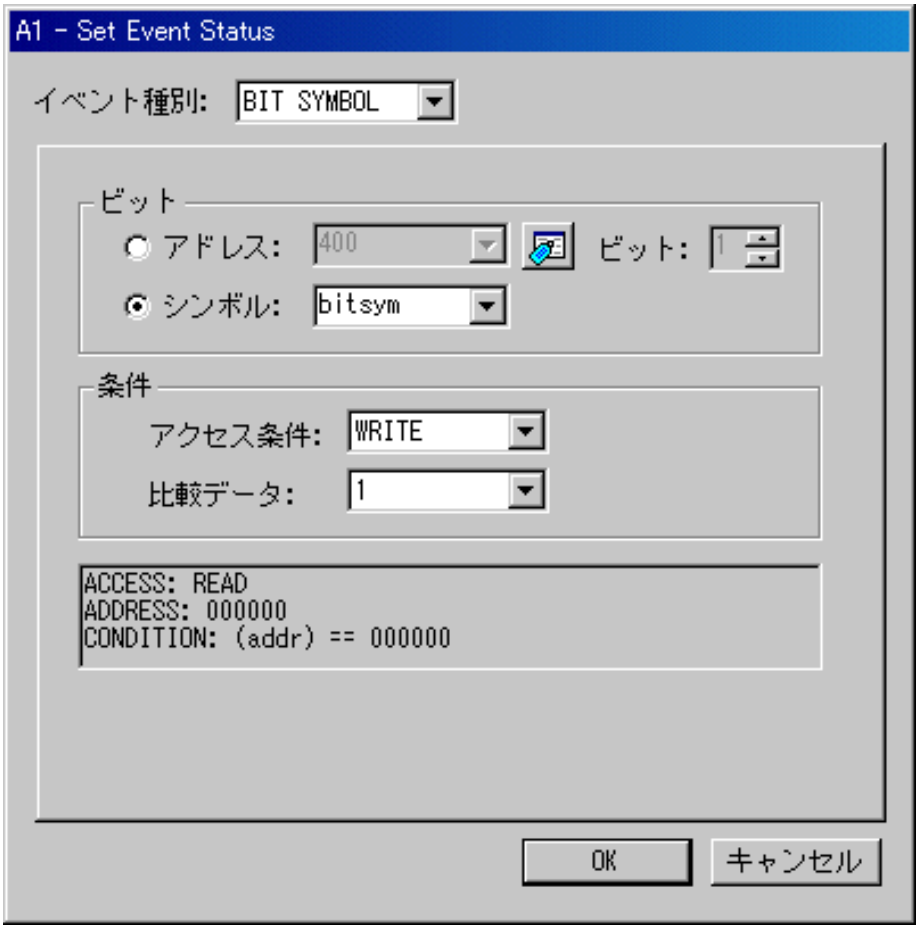
6.10.8.1 指定ビットへの書き込み/読み込み

以下のように設定します。
例)アドレス 400h のビット 2 へ"0"を書き込み



6.10.8.2 指定ビットシンボルへの書き込み/読み込み

以下のように設定します。
例)ビットシンボル bitsym へ"1"を書き込み

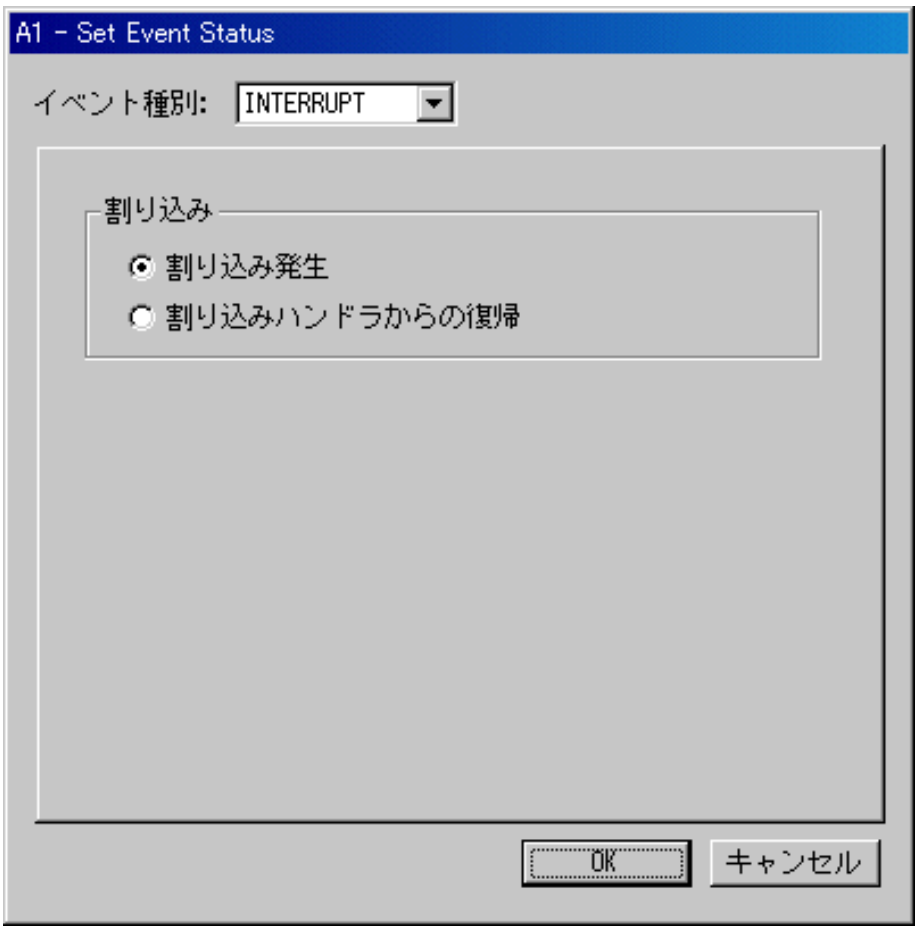


6.10.9 イベントを設定する（割り込み）

割り込みのイベントを指定する場合は、イベント選択ダイアログの[イベント種別]を "INTERRUPT"に 変更してください。 割り込み発生または割り込み終了した場合にイベントが成立します。

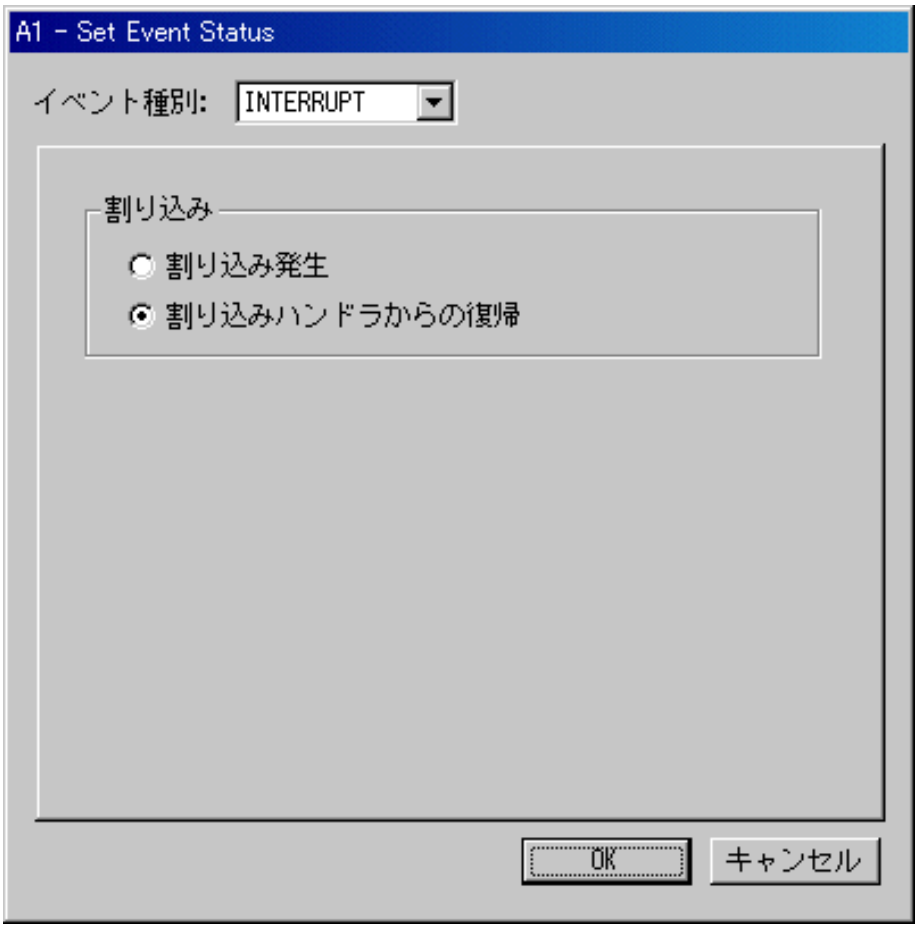
6.10.9.1 割り込み発生

以下のように設定します。



6.10.9.2 割り込み終了

以下のように設定します。



6.10.10 イベントの組み合わせ条件を設定する

イベント設定ウィンドウの[組み合わせ条件]グループで指定します。

複数のイベントを組み合わせることができます。(トレースイベントの場合は、B1～B6)　組み合わせ条件は、以下のいずれかを選択できます。

AND	指定イベントがすべて成立
AND(Same Time)	指定イベントが同時に成立
OR	指定イベントのいずれかが成立
STATE TRANSITION	状態遷移図におけるブレークステート突入で成立

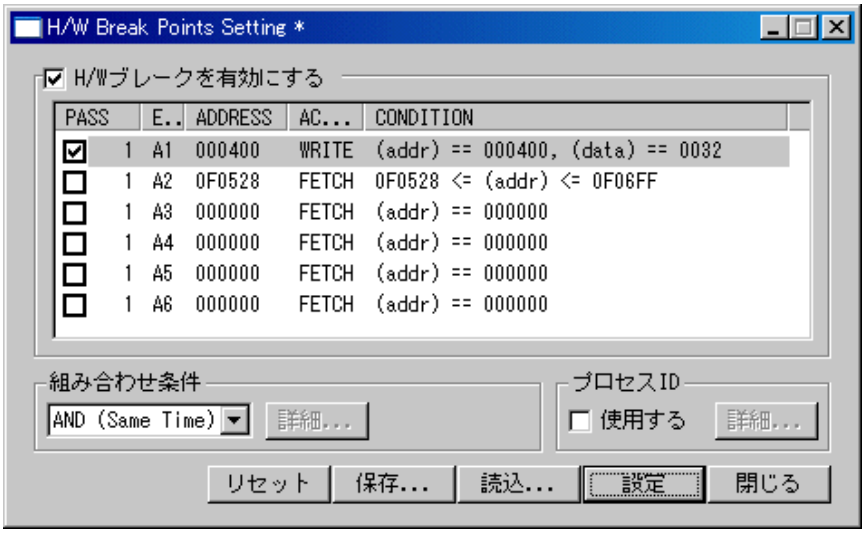
それぞれのイベントには、パスカウント(通過回数)の指定ができます(1～255)。　組み合わせ条件にAnd(same time)を指定した場合は、パスカウント(通過回数)は指定できません(1 固定です)。

6.10.10.1 AND,OR 指定

組み合わせ条件を AND とする場合は[組み合わせ条件]グループを"AND"、 OR とする場合は"OR"に変更してください。 次にイベント指定領域で使用するイベントをチェックし、そのイベントのパスカウント(通過回数) を指定してください。 パスカウント(通過回数)を変更する場合は、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。

6.10.10.2 AND(Same Time)指定

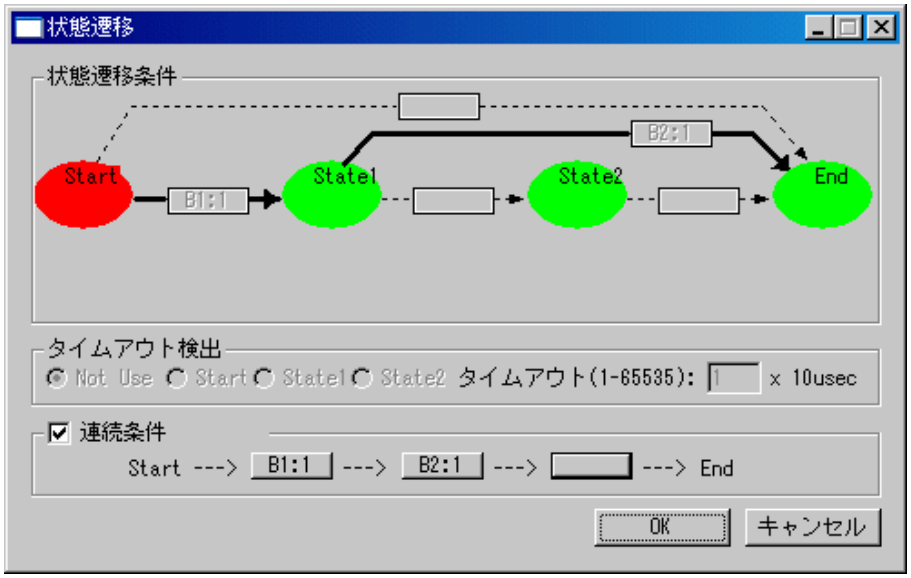
[組み合わせ条件]グループを"AND(Same Time)"に変更してください。 次にイベント指定領域で使用するイベントをチェックしてください。 パスカウント(通過回数)は指定できません(1 固定です)。



6.10.10.3 State Transition 指定

[組み合わせ条件]グループを"State Transition"に変更してください。[組み合わせ条件]グループの[詳細...]ボタンが有効になりますので、そのボタンをクリックしてください。 状態遷移ウィンドウがオープンします。 状態遷移ウィンドウでは、シーケンシャル指定ができます。
[連続条件]グループのボタンを使用します。 パスカウント(通過回数)は、イベント選択時のポップアップメニューから指定することができます。 設定した内容は、状態遷移図に反映されます。

例)B1→B2 と連続して発生するイベントが成立



6.10.11 書き込み条件を設定する

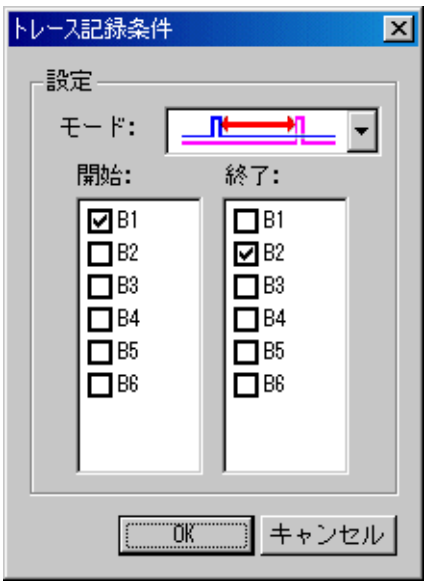
トレースデータの書き込み条件を指定することができます。
以下の書き込み条件を指定することができます。

1. 書き込み条件の制限なし(デフォルト)
2. 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
3. 開始イベント成立のサイクルのみ
4. 開始イベント成立から開始イベント不成立となるまでのサイクル
5. 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以外
6. 開始イベント成立のサイクル以外
7. 開始イベント成立から開始イベント不成立となるまでのサイクル以外

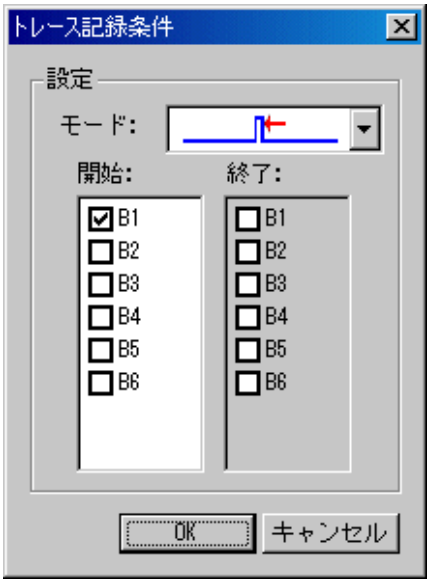
1 を指定するには、ウィンドウ内の［サイクル記録条件］の項目で、リストボックスより"Total"を選択します。

2 ～ 4 を指定するには同様に、"Pick Up"を選択後、ボタン[詳細...]"をクリックして、トレース記録条件ダイアログをオープンします。

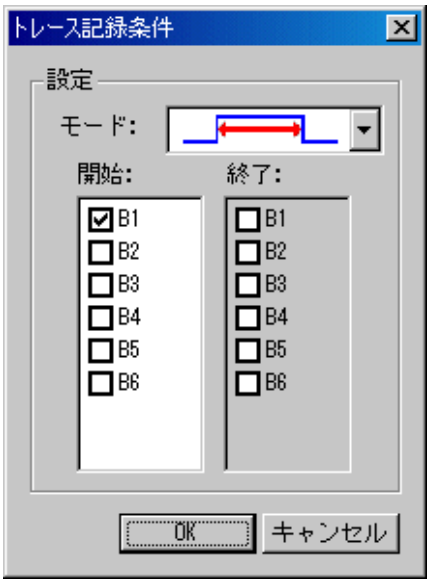
- 2 の場合、以下のモードを選択して、開始と終了のイベントをそれぞれ設定します。



- 3 の場合、以下のモードを選択して、開始イベントをそれぞれ設定します。

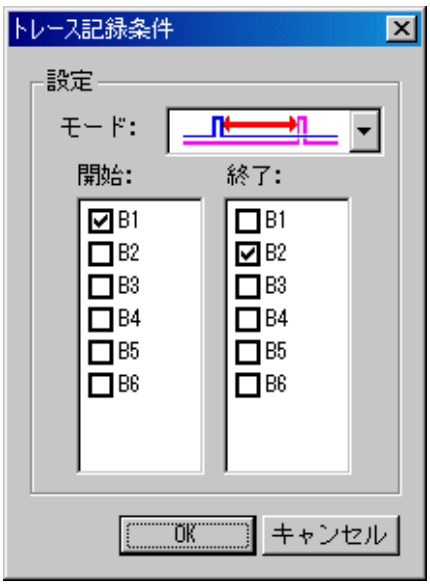


- 4 の場合、以下のモードを選択して、開始イベントをそれぞれ設定します。

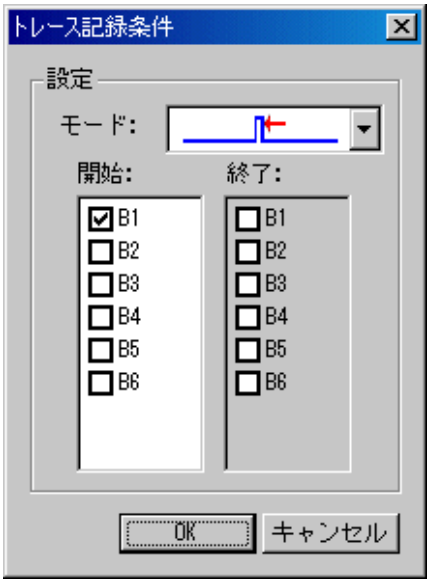


5 ～ 7 を指定するにも同様に、"Exclude"を選択後、ボタン [詳細...]をクリックして、トレース記録条件ダイアログをオープンします。

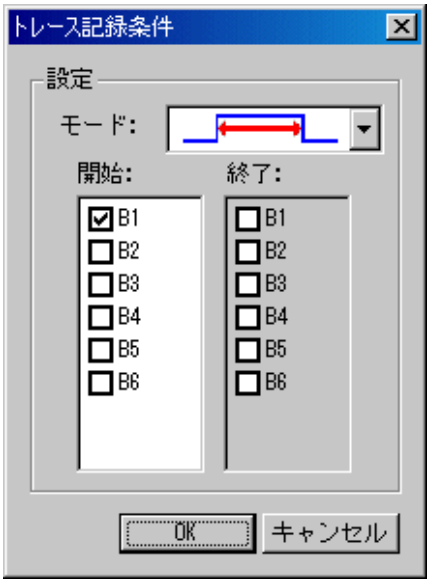
- 5 の場合、以下のモードを選択して、開始と終了のイベントをそれぞれ設定します。



- 6 の場合、以下のモードを選択して、開始イベントをそれぞれ設定します。



- 7 の場合、以下のモードを選択して、開始イベントをそれぞれ設定します。



6.11 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。
740 用デバッガでは、サポートしていません。
以下の表示形式で表示できます。

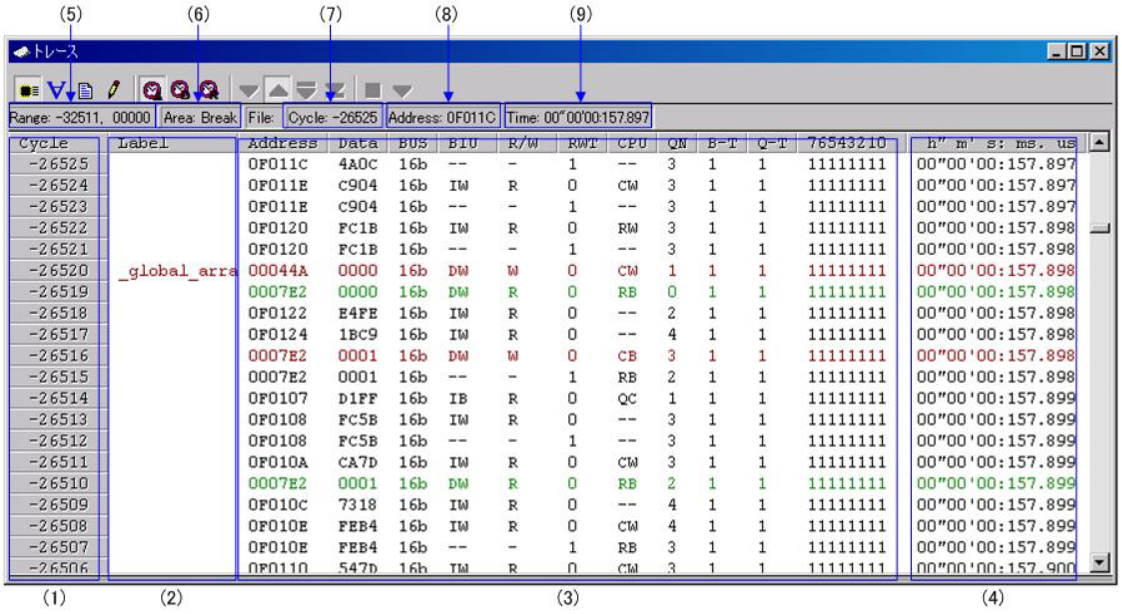
- バスモード
  サイクルごとのバス情報を参照できます。表示内容は、ご使用の MCU、シミュレータシステムに依存します。バス情報に加えて、逆アセンブル情報、ソース行情報、データアクセス情報を混合表示できます。
- 逆アセンブルモード
  実行した命令を参照できます。逆アセンブル情報に加えて、ソース行情報、データアクセス情報を混合表示できます。
- データアクセスモード
  データの R/W サイクルを参照できます。データアクセス情報に加えて、ソース行情報を混合表示できます。
- ソースモード
  プログラムの実行経路をソースプログラム上で参照できます。

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、「6.10 トレースポイント設定ウィンドウ」 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。

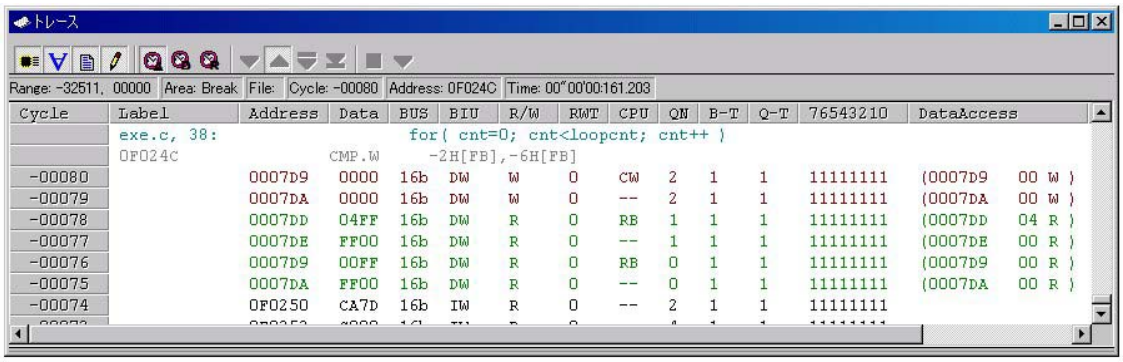
6.11.1 バスモードの構成

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 やシミュレータシステムにより異なります。



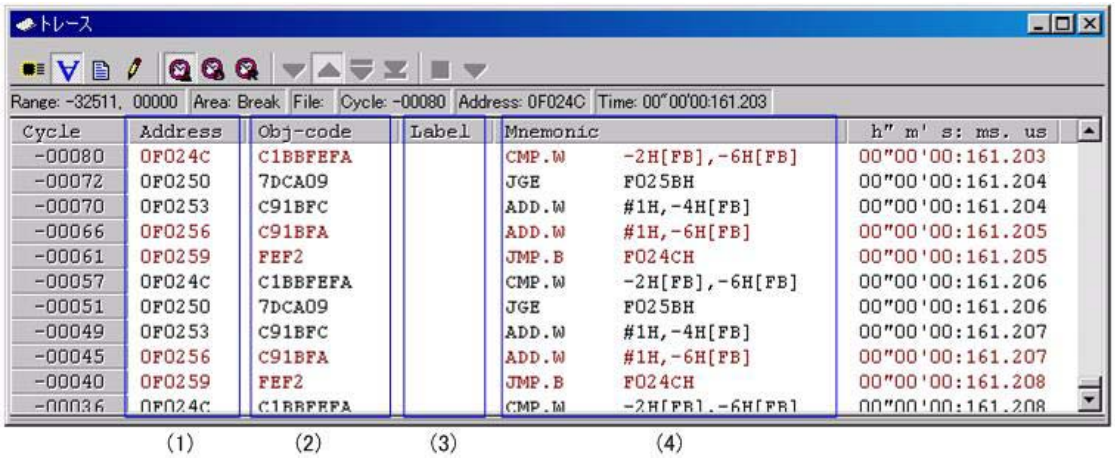
1. サイクル表示領域：
   トレースサイクルを表示します。ダブルクリックすると、表示サイクルを変更するためのダイアログボックスが表示されます。
2. ラベル表示領域：
   アドレスバス情報に対応するラベルを表示します。ダブルクリックすると、アドレスを検索するためのダイアログボックスが表示されます。
3. バス情報表示領域：
   表示内容は、ご使用の MCU やシミュレータシステムにより異なります。
   - 「6.11.6 シミュレータデバッガでのバス情報表示」を参照ください。
4. 時間情報表示領域：
   トレース計測結果の時間情報を表示します。以下の 3 通りの方法をメニューから選択できます。
   - 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。



6.11.2 逆アセンブルモードの構成

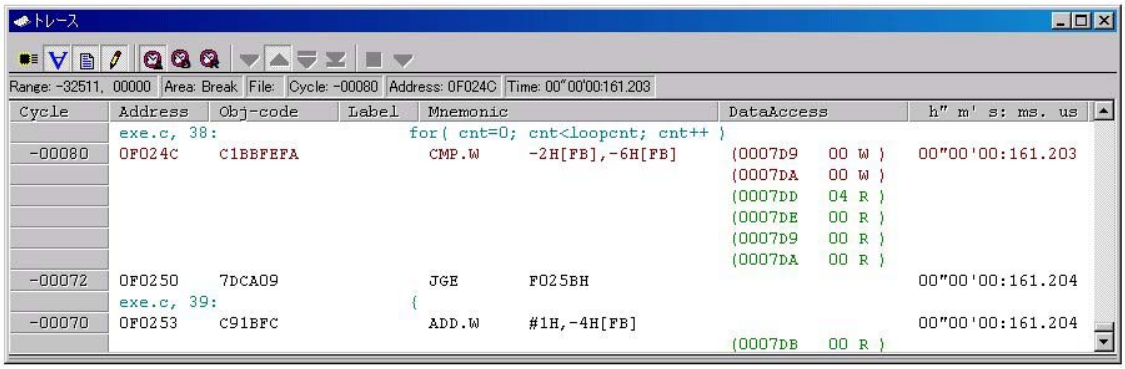
バスモードが選択されておらず、逆アセンブルモードが選択されている場合、逆アセンブルモードの表示になります。逆アセンブルモードは、以下の構成になっています。



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。

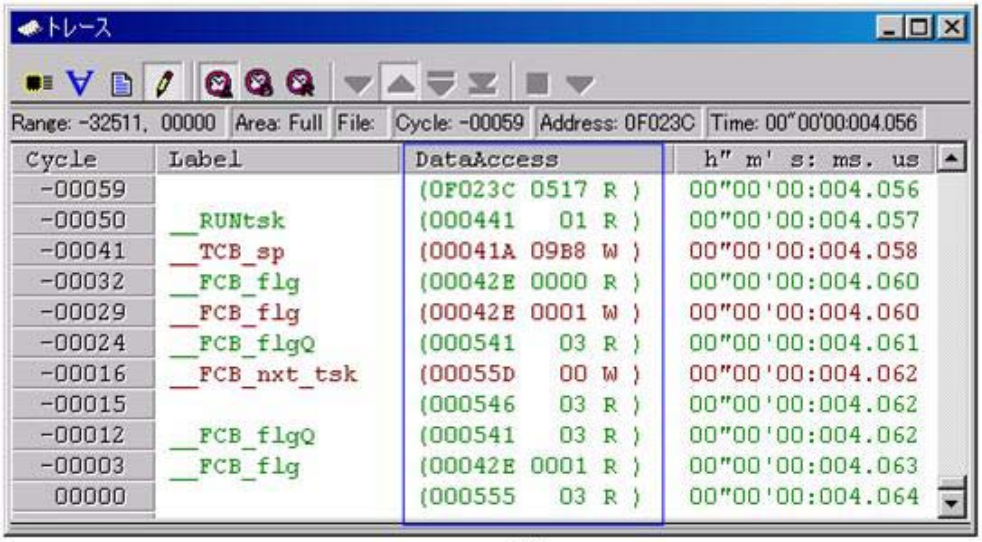
その他の表示はバスモードと同様です。

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。



6.11.3 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。

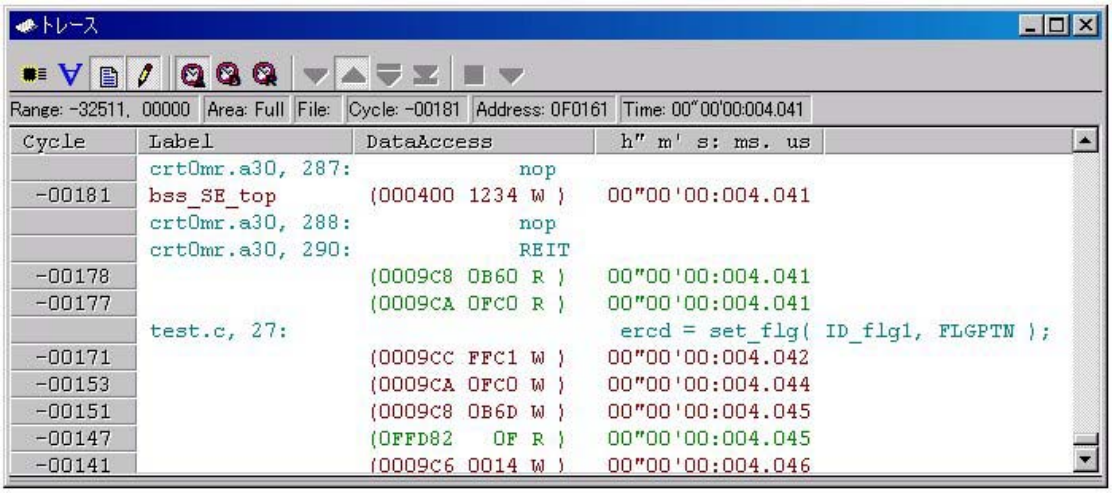


(1)

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。

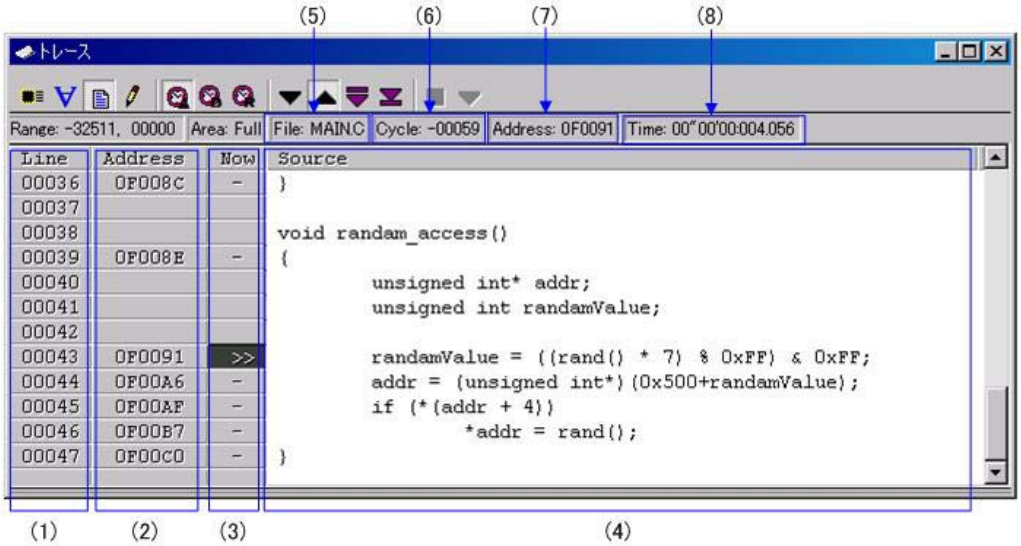
その他の表示はバスモードと同様です。

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。



6.11.4 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。

6.11.5 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。

6.11.6 シミュレータデバッガでのバス情報表示

左端より以下の内容を意味します。

- Address
  アドレスバスの状態を示します。
- Data
  データバスの状態を示します。
- Size
  データアクセスサイズを示します。

製品	表示形式	サイズ
R32C 用デバッガ	DB DW DL	8 ビット 16 ビット 32 ビット
M32C 用デバッガ M16C/R8C 用デバッガ	DB DW	8 ビット 16 ビット

- Type
  データアクセスの種類を示します。

表示形式	ステータス
Code *1	ニーモニックがフェッチされた状態を示します。
Data	データがアクセスされたことを示します。

*1 R32C 用デバッガの Code データ表示は 32 ビット固定であり、それ以降の表示は省略されます。
M32C 用デバッガ, M16C/R8C 用デバッガの Code データ表示は 16 ビット固定であり、それ以降の表示は省略されます。

- R/W
  データのアクセス状態を示します。

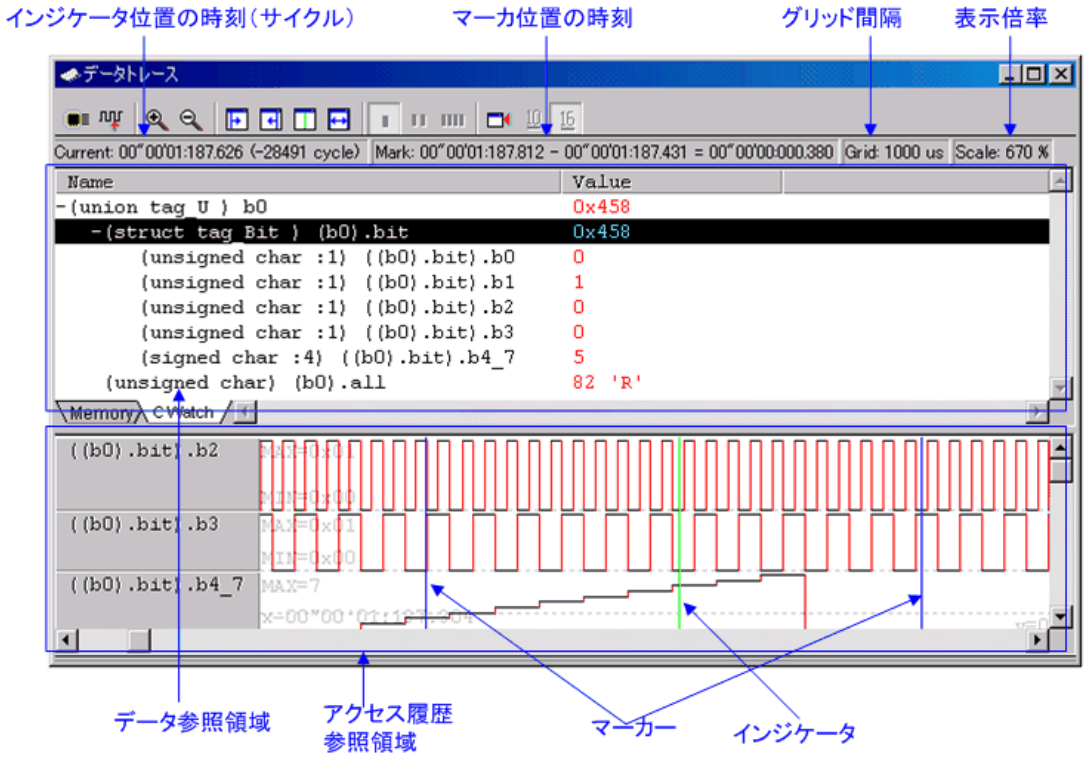
表示形式	ステータス
R	データリード
W	データライト

Type が Code の場合は常に R (コードリード) になります。

- h" m' s: ms.us
  ターゲットプログラム開始からの経過時間を示します。
  また、後に続く "( )" で括られた値はプログラム実行開始時を 0 とする命令実行サイクルの合計値を示します。

6.12 データトレースウィンドウ

データ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解析し、データアクセス情報をグラフィカルに表示するウィンドウです。
トレースウィンドウと連携して動作します。
740 用デバッガでは、サポートしていません。



- データ参照領域では、現在注目しているサイクル時点でのメモリの値、または、登録した C 変数の値を参照できます。
- アクセス履歴参照領域では、登録したアドレスへのアクセス履歴ををチャート形式で参照できます。
- トレースウィンドウと連携し、トレースウィンドウで注目しているサイクル時点でのメモリの値を参照できます。逆に、データトレースウィンドウで注目しているサイクルをトレースウィンドウで表示できます。

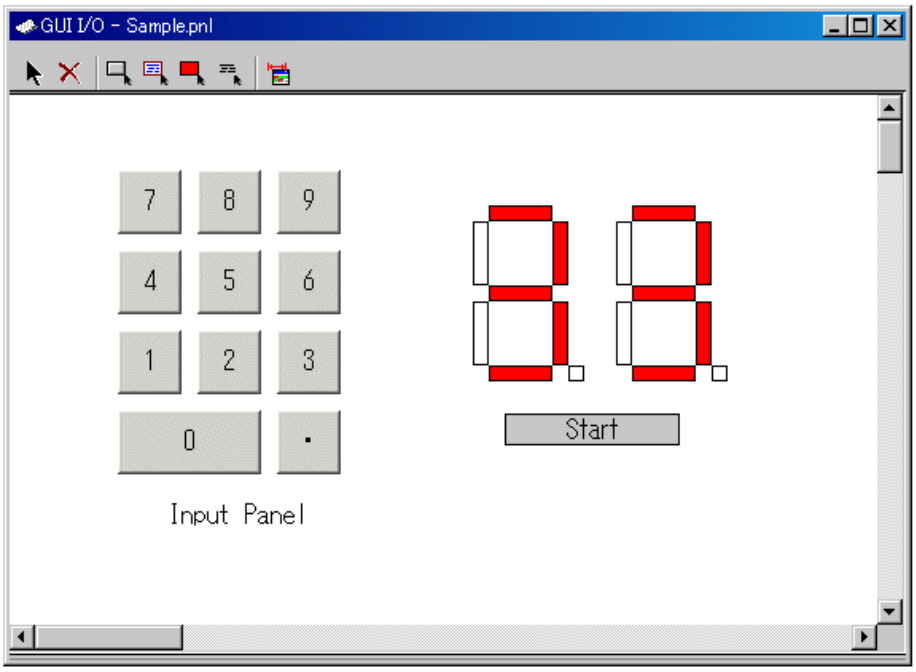
6.12.1 オプションメニュー

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
トレースデータの解析		トレースデータを解析します。
サイクル指定...		表示サイクルを指定します。
トレースウィンドウと連動		トレースウィンドウと連携して動作します。
データ長	1byte	1 バイト幅で表示します。
	2byte	2 バイト幅で表示します。
	4byte	4 バイト幅で表示します。
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。
アドレス...		表示アドレスを指定します。
シンボル登録		C ウォッチタブに C 変数を追加します。
シンボル削除		選択されている変数を削除します。
型名の非表示		型名を非表示にします。
リストに項目を追加...		アクセス履歴表示領域に項目を追加します。
リストから項目を削除		アクセス履歴表示領域から項目を削除します。
ズーム	拡大	表示を拡大します。
	縮小	表示を縮小します。
	倍率指定...	表示倍率を指定します。
マーカー	始点マーカー	始点マーカーを表示領域に移動します。
	終点マーカー	終点マーカーを表示領域に移動します。
	現在位置マーカー	現在位置マーカーを表示領域に移動します。
	表示倍率の調整	マーカー間をウィンドウいっぱいに表示します。
グリッド間隔の変更...		グリッド間隔を変更します。
項目の設定...		選択されている項目の表示設定を変更します。
表示色の設定...		表示色を変更します。
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。

6.13 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、仮想ポートに入力したり、仮想割り込みを発生させたりできます。 (仮想割り込みはシミュレータデバッガのみ)。
  - ・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。

6.13.1 オプションメニュー

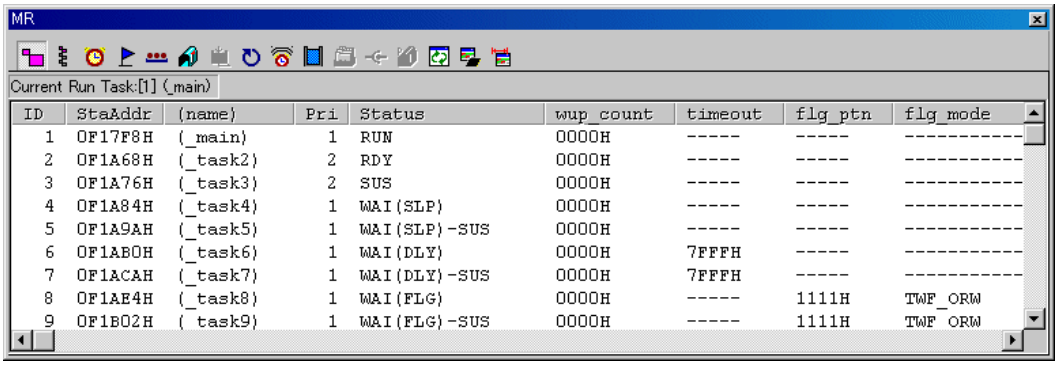
ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。 これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

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。

6.14 MR ウィンドウ

MR ウィンドウは、リアルタイム OS の状態を表示するウィンドウです。
740 用デバッガでは、サポートしていません。

リアルタイム OS を使用したプログラムをダウンロードした場合にのみ使用することができます。
ダウンロードしたプログラムが MR を使用していなかった場合、MR ウィンドウをオープンしても MR ウィンドウには何も表示されません。



- MR ウィンドウは、表示モードの種類数分までオープンすることができます。
- 各ボタンをクリックすることにより、MR ウィンドウの表示モードが切り換わり、表示内容も切り換わります。
- 各タスクの行をダブルクリックすることにより、そのタスクのコンテキスト内容を表示させることができます。
- 各モードの各表示領域は、ドラッグ操作により、表示幅を変更することができます。
- ダウンロードしたプログラムが MR を使用していなかった場合、表示モードを選択するメニューはすべて選択できなくなります。
- 選択可能な表示モードは、ご使用の MR によって異なります。

注意事項

ターゲットプログラム作成の際、ご使用の MRxx のバージョンに対応したスタートアップファイル(crt0mr.xxx)をご使用下さい。対応していない場合、リアルタイム OS に依存する部分のデバッグができなくなる場合があります。

6.14.1 オプションメニュー

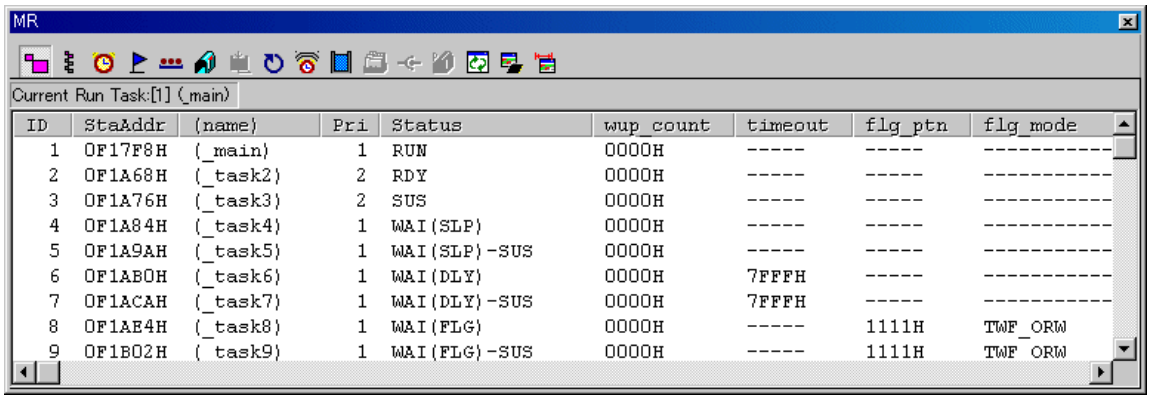
ウィンドウ内でマウスの右ボタンをクリックすると以下のポップアップメニューを表示します。 これらのメニューの主要な機能は、ツールバーのボタンにも割り付けられています。

メニュー名		機能
表示モード	タスク	タスクの状態を表示します。
	レディキュー	レディキューの状態を表示します。
	タイムアウトキュー	タイムアウトキューの状態を表示します。
	イベントフラグ	イベントフラグの状態を表示します。
	セマフォ	セマフォの状態を表示します。
	メールボックス	メールボックスの状態を表示します。
	データキュー	データキューの状態を表示します。
	周期起動ハンドラ	周期起動ハンドラの状態を表示します。
	アラームハンドラ	アラームハンドラの状態を表示します。
	メモリプール	メモリプールの状態を表示します。
	メッセージバッファ*	メッセージバッファの状態を表示します。
	ポート	ポートの状態を表示します。
	メールボックス(優先度付き)	メールボックス(優先度付き)の状態を表示します。
コンテキスト表示...		タスクのコンテキストを表示します。
レイアウト	ステータスバーの表示	ステータスバーの表示/非表示を切り替えます。
最新の情報に更新		メモリをリフレッシュします。
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。

*: R32C 用デバッガではサポートしていません。

6.14.2 タスクの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[タスク]を選択してください。



任意行をダブルクリックすることにより、Context ダイアログにそのタスクのコンテキスト情報を表示します。
Contextダイアログの詳細については、「6.14.12 タスクのコンテキストを参照/設定する」を参照してください。
ステータスバーには、現在実行中のタスク ID とタスク名を表示します。

6.14.2.1 タスクの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

コンフィグレーションで定義されたすべてのタスクを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
ID	タスク ID の番号を表示します。
StaAddr	タスクの開始アドレスを表示します。
(name)	タスク名を表示します。
Pri	優先度を表示します。
Status*1	タスクの状態を表示します。
wup_count	ウェイクアップカウント値を表示します。
timeout	タスクが時間待ち状態の場合、そのタイムアウト値を表示します。
flg_ptn	タスクがイベントフラグ待ち状態の場合、その待ちビットパターンを表示します。
flg_mode*2	タスクがイベントフラグ待ち状態の場合、その待ち解除条件を表示します。

- *1 タスクの状態表示

表示	状態
RUN	実行状態
RDY	実行可能状態
SUS	強制待ち状態
DMT	休止状態
WAI(SLP)	起床待ち状態
WAI(SLP)-SUS	起床待ち状態(二重待ち)
WAI(SLP-TMO)	タイムアウト付起床待ち状態
WAI(SLP-TMO)-SUS	タイムアウト付起床待ち状態(二重待ち)
WAI(DLY)	dly_tsk による時間経過待ち状態
WAI(DLY)-SUS	dly_tsk による時間経過待ち状態(二重待ち)
WAI(FLG)	イベントフラグ待ち状態
WAI(FLG)-SUS	イベントフラグ待ち状態(二重待ち)
WAI(FLG-TMO)	タイムアウト付イベントフラグ待ち状態
WAI(FLG-TMO)-SUS	タイムアウト付イベントフラグ待ち状態(二重待ち)
WAI(SEM)	セマフォ資源の獲得待ち状態
WAI(SEM)-SUS	セマフォ資源の獲得待ち状態(二重待ち)
WAI(SEM-TMO)	タイムアウト付セマフォ資源の獲得待ち状態
WAI(SEM-TMO)-SUS	タイムアウト付セマフォ資源の獲得待ち状態(二重待ち)
WAI(MBX)	メールボックスからの受信待ち状態
WAI(MBX)-SUS	メールボックスからの受信待ち状態(二重待ち)
WAI(MBX-TMO)	タイムアウト付メールボックスからの受信待ち状態
WAI(MBX-TMO)-SUS	タイムアウト付メールボックスからの受信待ち状態(二重待ち)

- *2 イベントフラグの待ち解除条件表示

flg_mode	状態
TWF_ANDW	待ちビットパターンで設定されているビットのすべてが、イベントフラグにセットされるのを待ちます(AND 待ち)。
TWF_ANDW+TWF_CLR	AND 待ちが発生し、タスクが待ち解除になった場合に、イベントフラグの値を 0 クリアします。
TWF_ORW	待ちビットパターンで設定されているビットのいずれかがイベントフラグにセットされるのを待ちます(OR 待ち)。
TWF_ORW+TWF_CLR	OR 待ちが発生し、タスクが待ち解除になった場合に、イベントフラグの値を 0 クリアします。

6.14.2.2 タスクの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

コンフィグレーションで定義されたすべてのタスクを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

項目	内容
ID	タスク ID の番号を表示します。
Name	タスク名を表示します。
Pri	優先度を表示します。
Status*1	タスクの状態を表示します。
Wupcnt	ウェイクアップカウント値を表示します。
Actcnt	起動要求キューイング数を表示します。
Tmout	タスクが時間待ち状態の場合、そのタイムアウト値(ms 単位)を表示します。
Flgptn	タスクがイベントフラグ待ち状態の場合、その待ちビットパターンを表示します。
Wfmode*2	タスクがイベントフラグ待ち状態の場合、その待ち解除条件を表示します。

- *1 タスクの状態表示

Status	状態
RUN	実行状態
RDY	実行可能状態
SUS	強制待ち状態
DMT	休止状態
WAI(SLP)	起床待ち状態
WAI(SLP)-SUS	起床待ち状態(二重待ち)
WAI(SLP-TMO)	タイムアウト付起床待ち状態
WAI(SLP-TMO)-SUS	タイムアウト付起床待ち状態(二重待ち)
WAI(DLY)	dly_tsk による時間経過待ち状態
WAI(DLY)-SUS	dly_tsk による時間経過待ち状態(二重待ち)
WAI(FLG)	イベントフラグ待ち状態
WAI(FLG)-SUS	イベントフラグ待ち状態(二重待ち)
WAI(FLG-TMO)	タイムアウト付イベントフラグ待ち状態
WAI(FLG-TMO)-SUS	タイムアウト付イベントフラグ待ち状態(二重待ち)
WAI(SEM)	セマフォ資源の獲得待ち状態
WAI(SEM)-SUS	セマフォ資源の獲得待ち状態(二重待ち)
WAI(SEM-TMO)	タイムアウト付セマフォ資源の獲得待ち状態
WAI(SEM-TMO)-SUS	タイムアウト付セマフォ資源の獲得待ち状態(二重待ち)
WAI(MBX)	メールボックスからの受信待ち状態
WAI(MBX)-SUS	メールボックスからの受信待ち状態(二重待ち)
WAI(MBX-TMO)	タイムアウト付メールボックスからの受信待ち状態
WAI(MBX-TMO)-SUS	タイムアウト付メールボックスからの受信待ち状態(二重待ち)
WAI(SDTQ)	データキューへの送信待ち状態
WAI(SDTQ)-SUS	データキューへの送信待ち状態(二重待ち)
WAI(SDTQ-TMO)	タイムアウト付データキューへの送信待ち状態
WAI(SDTQ-TMO)-SUS	タイムアウト付データキューへの送信待ち状態(二重待ち)
WAI(RDTQ)	データキューからの受信待ち状態
WAI(RDTQ)-SUS	データキューからの受信待ち状態(二重待ち)
WAI(RDTQ-TMO)	タイムアウト付データキューからの受信待ち状態
WAI(RDTQ-TMO)-SUS	タイムアウト付データキューからの受信待ち状態(二重待ち)
WAI(VSDTQ)	拡張データキューへの送信待ち状態
WAI(VSDTQ)-SUS	拡張データキューへの送信待ち状態(二重待ち)
WAI(VSDTQ-TMO)	タイムアウト付拡張データキューへの送信待ち状態
WAI(VSDTQ-TMO)-SUS	タイムアウト付拡張データキューへの送信待ち状態(二重待ち)
WAI(VRDTQ)	拡張データキューからの受信待ち状態
WAI(VRDTQ)-SUS	拡張データキューからの受信待ち状態(二重待ち)
WAI(VRDTQ-TMO)	タイムアウト付拡張データキューからの受信待ち状態
WAI(VRDTQ-TMO)-SUS	タイムアウト付拡拡張データキューからの受信待ち状態(二重待ち)
WAI(MPF)	固定長メモリブロックの獲得待ち状態
WAI(MPF)-SUS	固定長メモリブロックの獲得待ち状態
WAI(MPF-TMO)	タイムアウト付固定長メモリブロックの獲得待ち状態
WAI(MPF-TMO)-SUS	タイムアウト付固定長メモリブロックの獲得待ち状態(二重待ち)
WAI(SMBF) *	メッセージバッファへの送信待ち状態
WAI(SMBF)-SUS *	メッセージバッファへの送信待ち状態(二重待ち)
WAI(SMBF-TMO) *	タイムアウト付メッセージバッファへの送信待ち状態
WAI(SMBF-TMO)-SUS *	タイムアウト付メッセージバッファへの送信待ち状態(二重待ち)
WAI(RMBF) *	メッセージバッファからの受信待ち状態
WAI(RMBF)-SUS *	メッセージバッファからの受信待ち状態(二重待ち)
WAI(RMBF-TMO) *	タイムアウト付メッセージバッファからの受信待ち状態
WAI(RMBF-TMO)-SUS *	タイムアウト付メッセージバッファからの受信待ち状態(二重待ち)
WAI(MTX) *	ミューテックスのロック待ち状態

WAI(MTX)-SUS *	ミューテックスのロック待ち状態(二重待ち)
WAI(MTX-TMO) *	タイムアウト付ミューテックスのロック待ち状態
WAI(MTX-TMO)-SUS *	タイムアウト付ミューテックスのロック待ち状態(二重待ち)

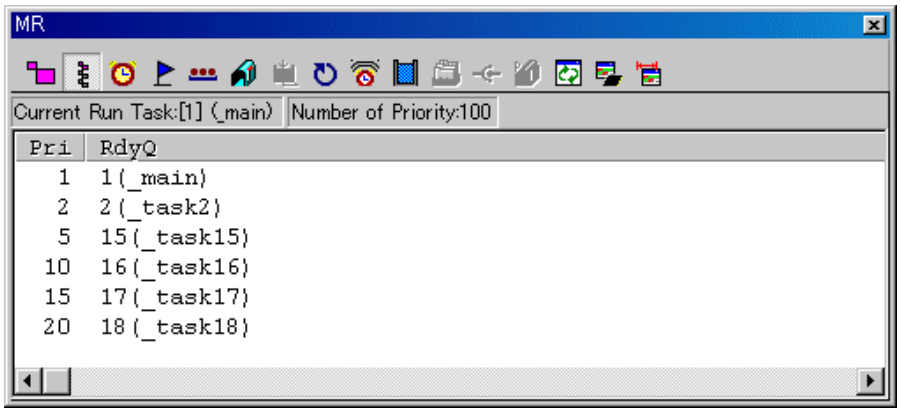
*: M32C 用デバッガではサポートしていません。
*: M16C/R8C 用デバッガではサポートしていません。

- *2 イベントフラグの待ち解除条件表示

Wfmode	状態
TWF_ANDW	待ちビットパターンで設定されているビットのすべてが、イベントフラグにセットされるのを待ちます(AND 待ち)。
TWF_ORW	待ちビットパターンで設定されているビットのいずれかがイベントフラグにセットされるのを待ちます(OR 待ち)。

6.14.3 レディキューの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[レディキュー]を選択してください。



ステータスバーには、現在実行中のタスク ID とタスク名、最大優先度数を表示します。

6.14.3.1 レディキューの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

レディキューにつながっているタスクを優先度の高い順に表示します。各項目の内容は、以下の通りです。
(μITRON3 準拠の MRxx をご使用の場合)

項目	内容
Pri	優先度を表示します。
RdyQ	レディキューに並んでいるタスクの ID 番号を表示します。

- RdyQ 領域に表示されるタスク名の表示文字数は、最大 8 文字までです。 タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.3.2 レディキューの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

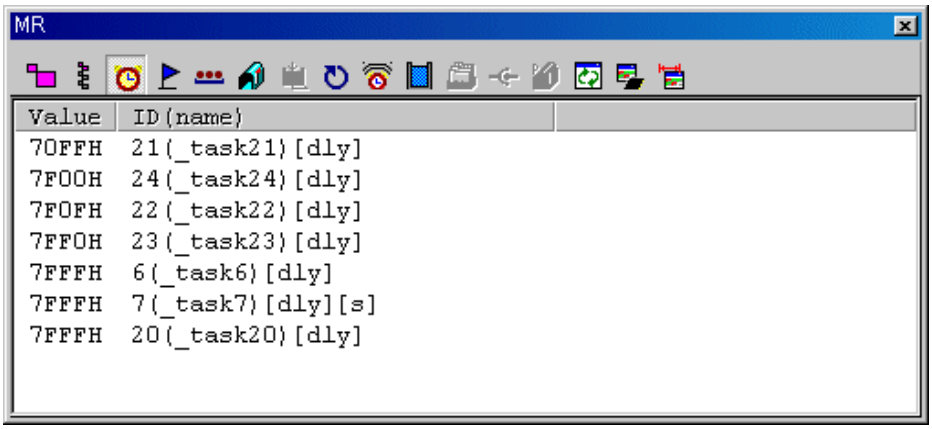
レディキューにつながっているタスクを優先度の高い順に表示します。各項目の内容は、以下の通りです。
(μITRON4 準拠の MRxx をご使用の場合)

項目	内容
Pri	優先度を表示します。
Ready Queue	レディキューに並んでいるタスクの ID 番号を表示します。

- RdyQ 領域に表示されるタスク名の表示文字数は、最大 8 文字までです。 タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.4 タイムアウトキューの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[タイムアウトキュー]を選択してください。



6.14.4.1 タイムアウトキューの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

現時点で時間待ち状態になっているタスクをタイムアウト値の小さい順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
Value	各タスクの現時点からのタイムアウト値を表示します。
ID(name)	タイムアウトキューに並んでいるタスク ID 番号とタスク名、および待ち状態の種類を表示します。

- 待ち状態の種類を示す文字列には、以下の種類があります。

文字列	待ち状態
[slp]	tslp_tsk による待ち
[dly]	dly_tsk による待ち
[flg]	twai_flg による待ち
[sem]	twai_sem による待ち
[mbx]	trcv_msg による待ち

- タイムアウトキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、 ID(name)領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

普通の場合の表示	26(_task26)
二重待ち状態の場合の表示	26(_task26)[s]

6.14.4.2 タイムアウトキューの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

現時点で時間待ち状態になっているタスクをタイムアウト値の小さい順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

項目	内容
Tmout	各タスクの現時点からのタイムアウト値を ms 単位で表示します。
ID(Name)	タイムアウトキューに並んでいるタスク ID 番号とタスク名、および待ち状態の種類を表示します。

- 待ち状態の種類を示す文字列には、以下の種類があります。

文字列	待ち状態
[slp]	tslp_tsk による待ち
[dly]	dly_tsk による待ち
[flg]	twai_flg による待ち
[sem]	twai_sem による待ち
[mbx]	trcv_mbx による待ち
[mpf]	tget_mpf による待ち
[sdtq]	tsnd_dtq による待ち
[rdtq]	trcv_dtq による待ち
[vsdtq]	vtsnd_dtq による待ち
[vrdtq]	vtrcv_dtq による待ち
[smbf] *	tsnd_mbf による待ち
[rmbf] *	trcv_mbf による待ち
[mtx] *	tloc_mtx による待ち

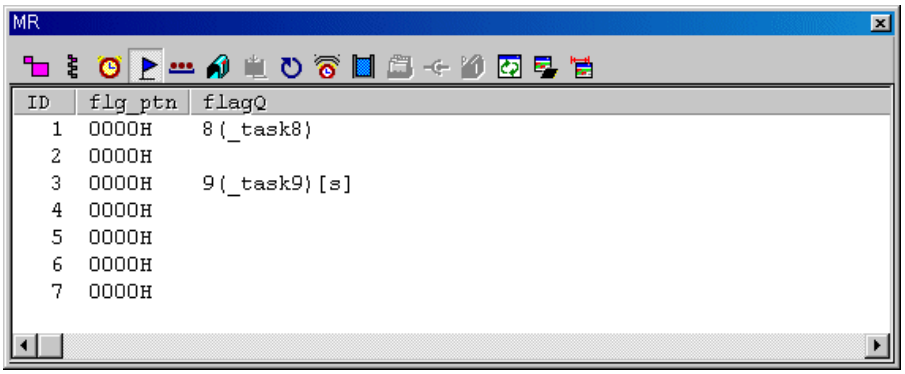
*: M32C 用デバッガではサポートしていません。
*: M16C/R8C 用デバッガではサポートしていません。

- タイムアウトキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、 ID(Name) 領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

普通の場合の表示	26(_task26)
二重待ち状態の場合の表示	26(_task26)[s]

6.14.5 イベントフラグの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[イベントフラグ]を選択してください。



6.14.5.1 イベントフラグの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

すべてのイベントフラグを ID 番号順に表示します。 各項目の内容は、以下の通りです。
(μITRON3 準拠の MRxx をご使用の場合)

項目	内容
ID	イベントフラグの ID 番号を表示します。
flg_ptn	イベントフラグのビットパターンを表示します。
flagQ	ベントフラグキューに並んでいるタスクの ID 番号を表示します。

- イベントフラグキューにつながったタスクがさらにタイムアウト有りの待ち状態(twai_flg による 待ち状態)の場合は、flagQ 領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字 列"[tmo]"が付加されます。
  また、イベントフラグキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、flagQ 領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- flagQ 領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.5.2 イベントフラグの状態を表示する（μITRON4 拠の MRxx をご使用の場合）

すべてのイベントフラグを ID 番号順に表示します。 各項目の内容は、以下の通りです。
(μITRON4 準拠の MRxx をご使用の場合)

項目	内容
ID	イベントフラグの ID 番号を表示します。
Flgatr	イベントフラグの属性を表示します。
Flgptn	イベントフラグのビットパターンを表示します。
Flag Queue	ベントフラグキューに並んでいるタスクの ID 番号とタスク名を表示します。

- Flgatr 領域の表示内容は、以下の種類があります。

TA_TFIFO	待ちタスクのキューイングは FIFO
TA_TPRI	待ちタスクのキューイングは優先度順
TA_WSGL	複数タスクの待ちを禁止
TA_WMUL	複数タスクの待ちを許可
TA_CLR	クリア指定

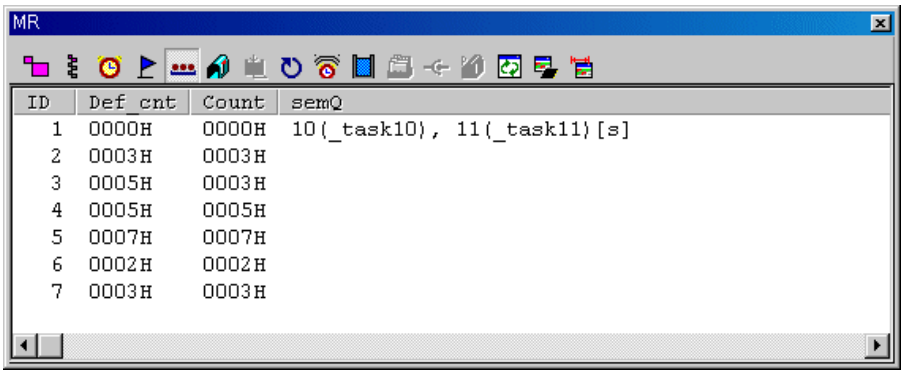
- イベントフラグキューにつながったタスクがさらにタイムアウト有りの待ち状態(twai_flg による 待ち状態)の場合は、Flag Queue 領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字 列"[tmo]"が付加されます。
  また、イベントフラグキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、Flag Queue 領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- Flag Queue 領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.6 セマフォの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[セマフォ]を選択してください。



6.14.6.1 セマフォの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

すべてのセマフォを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
ID	セマフォの ID 番号を表示します。
Def_cnt	セマフォカウンタの初期値を表示します。
Count	現時点のセマフォカウンタを表示します。
semQ	セマフォキューに並んでいるタスク ID 番号とタスク名を表示します。

- セマフォキューにつながったタスクがさらにタイムアウト有りの待ち状態(twai_sem による 待ち状態)の場合は、semQ 領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字 列"[tmo]"が付加されます。
  また、セマフォキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、semQ 領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- semQ 領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.6.2 セマフォの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

すべてのセマフォを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

項目	内容
ID	セマフォの ID 番号を表示します。
Sematr	セマフォの属性を表示します。
Semcnt	現時点のセマフォカウンタを表示します。
Semaphore Queue	セマフォキューに並んでいるタスク ID 番号とタスク名を表示します。

- Sematr 領域の表示内容は、以下の種類があります。

TA_TFIFO	待ちタスクのキューイングは FIFO
TA_TPRI	待ちタスクのキューイングは優先度順

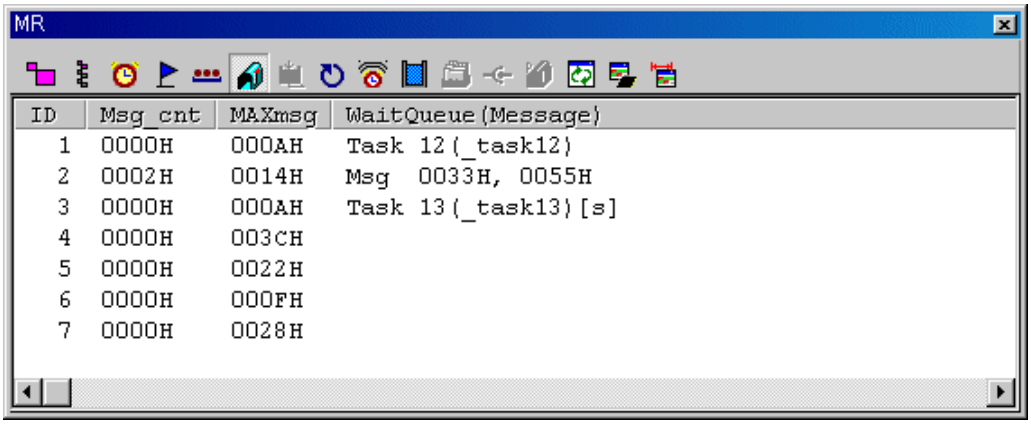
- セマフォキューにつながったタスクがさらにタイムアウト有りの待ち状態(twai_sem による 待ち状態)の場合は、Semaphore Queue 領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字 列"[tmo]"が付加されます。
  また、セマフォキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、Semaphore Queue 領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- Semaphore Queue 領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.7 メールボックスの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[メールボックス]を選択してください。



6.14.7.1 メールボックスの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

すべてのメールボックスを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
ID	メールボックスの ID 番号を表示します。
Msg_cnt	メールボックスに格納されているメッセージ数を表示します。
MAXmsg	メールボックスに格納可能なメッセージ数を表示します。
Wait Queue(Message)	メールボックスに格納されているメッセージ、またはメッセージ待ちの タスク ID 番号とタスク名を表示します。

- WaitQueue (Message)領域の表示内容は、メッセージが格納されている場合（上記の Msg_cnt が 0 以外の場合)には、文字列"Msg"を表示し、続いて格納されている メッセージを表示します。
  メッセージが格納されていない場合(上記の Msg_cnt が 0 の場合)で、メッセージ待ちのタ スクが存在している場合には、文字列"Task"を表示し、続いてメッセージ待ちのタスク ID 番号とタスク名を表示します。

- メールボックスキューにつながったタスクがさらにタイムアウト有りの待ち状態(trcv_msg による 待ち状態)の場合は、WaitQueue(Message)領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字 列"[tmo]"が付加されます。

また、メールボックスキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、WaitQueue(Message)領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- WaitQueue(Message)領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.7.2 メールボックスの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

すべてのメールボックスを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

項目	内容
ID	メールボックスの ID 番号を表示します。
Mbxatr	メールボックスの属性を表示します。
Mailbox Queue (Wait)	メッセージ待ちのタスク ID 番号とタスク名を表示します。
Mailbox Queue (Message)	メールボックスに格納されているメッセージを表示します。

- Mbxatr 領域の表示内容は、以下の種類があります。

TA_TFIFO	待ちタスクのキューイングは FIFO
TA_TPRI	待ちタスクのキューイングは優先度順
TA_MFIFO	メッセージのキューイングは FIFO
TA_MPRI	メッセージのキューイングは優先度順

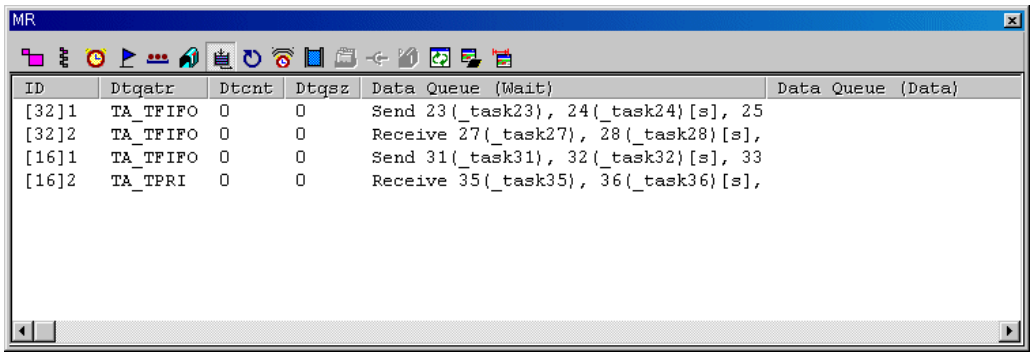
- メールボックスキューにつながったタスクがさらにタイムアウト有りの待ち状態(trcv_mbx による待ち状態)の場合は、Mailbox Queue (Wait)領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字 列"[tmo]"が付加されます。
  また、メールボックスキューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、Mailbox Queue (Wait)領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- Mailbox Queue (Wait)領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.8 データキューの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[データキュー]を選択してください



6.14.8.1 データキューの状態を表示する（(μITRON4 準拠の MRxx をご使用の場合)

すべてのデータキューを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合)

項目	内容
ID	データキューの ID 番号を表示します。
Dtqatr	データキューの属性を表示します。
Dtcnt	データキューに格納されているデータ数を表示します。
Dtqsz	データキューに格納可能なデータ数を表示します。
Data Queue (Wait)	データ送信待ち、または受信待ちのタスク ID 番号とタスク名を表示します。
Data Queue (Data)	データキューに格納されているデータを表示します。

- ID 領域は、標準データ, 拡張データの違いにより、以下のように表示内容が変わります。

  MR100/4 の場合
  ・標準データ(32bit)の場合、文字列"[32]"とデータキューの ID 番号を表示します。
  ・拡張データ(16bit)の場合、文字列"[16]"とデータキューの ID 番号を表示します。
  MR308/4 の場合
  ・標準データ(32bit)の場合、文字列"[32]"とデータキューの ID 番号を表示します。
  ・拡張データ(16bit)の場合、文字列"[16]"とデータキューの ID 番号を表示します。
  MR30/4 の場合
  ・標準データ(16bit)の場合、文字列"[16]"とデータキューの ID 番号を表示します。
  ・拡張データ(32bit)の場合、文字列"[32]"とデータキューの ID 番号を表示します。

- Dtqatr 領域の表示内容は、以下の種類があります。

TA_TFIFO	待ちタスクのキューイングは FIFO
TA_TPRI	待ちタスクのキューイングは優先度順

- Data Queue (Wait)領域の表示内容は、 送信待ちのタスクの場合には、文字列"Send"を表示し、続いてタスク ID とタスク名を表示します。 受信待ちのタスクの場合には、文字列"Receive"を表示し、続いてタスク ID とタスク名を表示します。

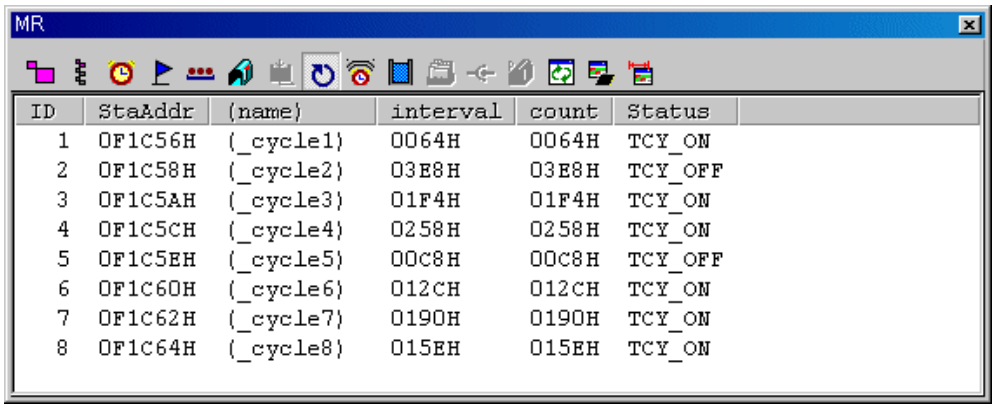
- キューにつながったタスクがさらにタイムアウト有りの待ち状態の場合は、 Data Queue (Wait)領域に表示される文字列の後ろにタイムアウト有りの待ち状態を示す文字列"[tmo]"が付加されます。
  また、キューにつながったタスクがさらに強制待ち状態(二重待ち状態)の場合は、 Data Queue (Wait)領域に表示される文字列の後ろに二重待ち状態を示す文字列"[s]"が付加されます。

通常	26(_task26)
二重待ち状態	26(_task26)[s]
タイムアウト有りの待ち状態+二重待ち状態	26(_task26)[tmo][s]

- Data Queue (Wait)領域に表示されるタスク名の表示文字数は、最大 8 文字までです。
  タスク名が 8 文字を超える場合、それ以降は省略されます。

6.14.9 周期起動ハンドラの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[周期起動ハンドラ]を選択してください。



6.14.9.1 周期ハンドルの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

すべての周期起動ハンドラを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
ID	周期起動ハンドラの ID 番号を表示します。
StaAddr	周期起動ハンドラの開始アドレスを表示します
(name)	周期起動ハンドラ名を表示します
interval	周期起動ハンドラの周期起動間隔を表示します。
count	周期起動ハンドラが次に起動するまでの割り込み回数(残数)を表示します。
Status	周期起動ハンドラの活性状態を表示します。

- Status 領域の表示内容は、以下の種類があります。

TCY_ON	周期起動ハンドラが有効です。
TCY_OFF	周期起動ハンドラが無効です。

6.14.9.2 周期ハンドルの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

すべての周期起動ハンドラを ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

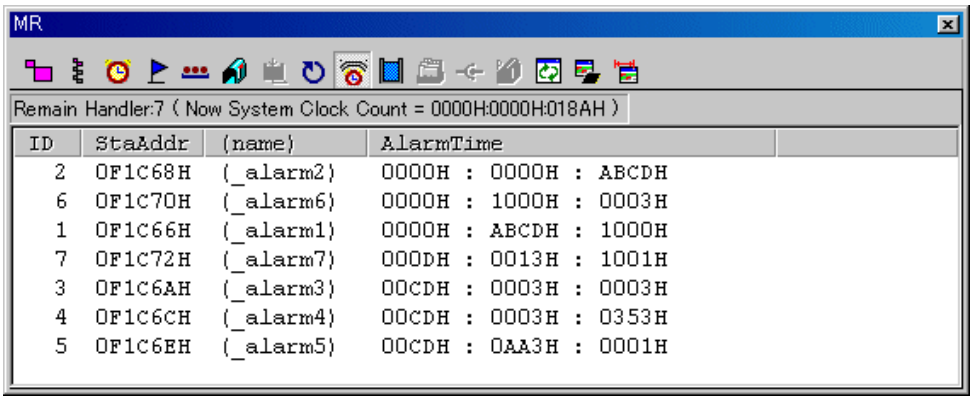
項目	内容
ID	周期起動ハンドラの ID 番号を表示します。
Name	周期起動ハンドラ名を表示します
Cycphs	起動位相を ms 単位で表示します。
Cyctim	周期起動間隔を ms 単位で表示します。
Tmout	次に起動するまでの時間を ms 単位で表示します。
Status	周期起動ハンドラの活性状態を表示します。

- Status 領域の表示内容は、以下の種類があります。

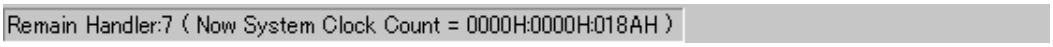
TCYC_STA	活性中
TCYC_STP	停止中

6.14.10 アラームハンドラの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[アラームハンドラ]を選択してください。



ステータスバーには、起動待ちのアラームハンドラ数、現在のシステムクロックカウントを表示します(μITRON3 準拠の MRxx をご使用の場合のみ)。



6.14.10.1 アラームハンドラの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

現時点で起動していないアラームハンドラのみを起動時刻の早い順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
ID	アラームハンドラの ID 番号を表示します。
StaAddr	アラームハンドラの開始アドレスを表示します。
(name)	アラームハンドラ名を表示します。
AlarmTime	アラームハンドラの起動時刻を表示します。

6.14.10.2 アラームハンドラの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

現時点で起動していないアラームハンドラのみを起動時刻の早い順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

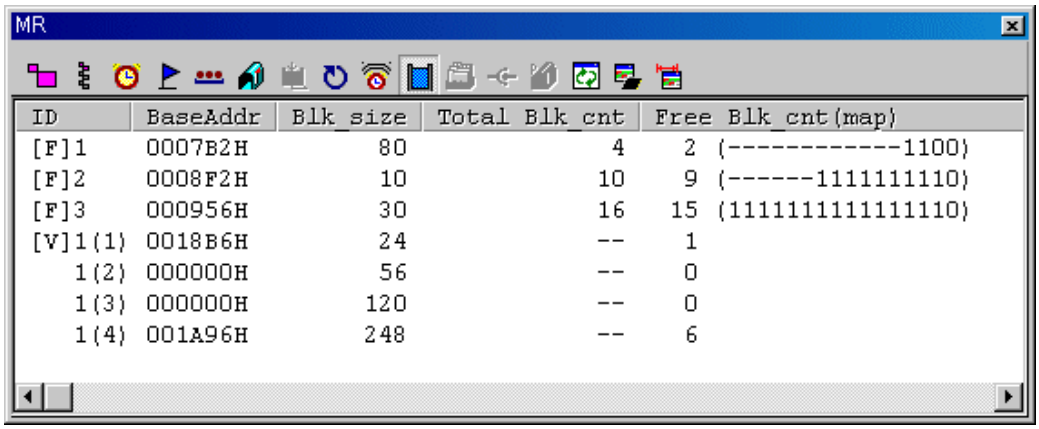
項目	内容
ID	アラームハンドラの ID 番号を表示します。
Name	アラームハンドラ名を表示します。
Almtim	アラームハンドラが次に起動するまでの時間を ms 単位で表示します。
Status	アラームハンドラの起動状態を表示します。

- Status 領域の表示内容は、以下の種類があります。

TALM_STA	動作中
TALM_STP	停止中

6.14.11 メモリプールの状態を表示する

MR ウィンドウのポップアップメニュー[表示モード]→[メモリプール]を選択してください。



6.14.11.1 メモリプールの状態を表示する（μITRON3 準拠の MRxx をご使用の場合）

メモリプールを(固定長・任意長の順で)ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON3 準拠の MRxx をご使用の場合）

項目	内容
ID	メモリプールの ID 番号を表示します。
BaseAddr	メモリプールのベースアドレスを表示します。
Blk_size	メモリプールのブロックサイズを表示します。
Total Blk_cnt	メモリプールの全ブロック数を表示します。
Free Blk_cnt(map)	未使用のブロック数、およびメモリブロック情報(ビット情報)を表示します。

- ID 領域は、固定長・任意長の違いにより、以下のように表示内容が変わります。
  - ・固定長の場合、文字列"[F]"とメモリプールの ID 番号を表示します。
  - ・任意長の場合、1 行目には文字列"[V]"、メモリプール ID 番号、ブロック ID 番号を表示します。 2～4 行目にはメモリプール ID 番号、ブロック ID 番号を表示します。 ブロック ID 番号はカッコで囲んで表示します。

- 任意長メモリプールの場合、Total Blk_cnt 領域には"--"を表示します。
  また、Free Blk_cnt(map)領域のビット情報は表示されません。

- 固定長メモリプールの場合、Free Blk_cnt(map)領域のメモリブロック情報の 各ビットの表示形式は次のようになります。

表示	内容
'0'	メモリブロック使用不可(使用中)
'1'	メモリブロック使用可能(未使用)
'-'	もともとメモリブロックが存在しない

6.14.11.2 メモリプールの状態を表示する（μITRON4 準拠の MRxx をご使用の場合）

メモリプールを(固定長・任意長の順で)ID 番号順に表示します。 各項目の内容は、以下の通りです。（μITRON4 準拠の MRxx をご使用の場合）

項目	内容
ID	メモリプールの ID 番号を表示します。
Mplatr	メモリプールの属性を表示します。
Mpladr	メモリプール領域の先頭番地を表示します。
Mplsz	メモリプール領域のサイズを表示します。
Blkcnt	固定長メモリプールの全ブロック数を表示します。
Fblkcnt	未使用のブロック数を表示します。
Memory Pool Queue	メモリプール待ちのタスク ID 番号とタスク名を表示します。

- Mplatr 領域の表示内容は、以下の種類があります。

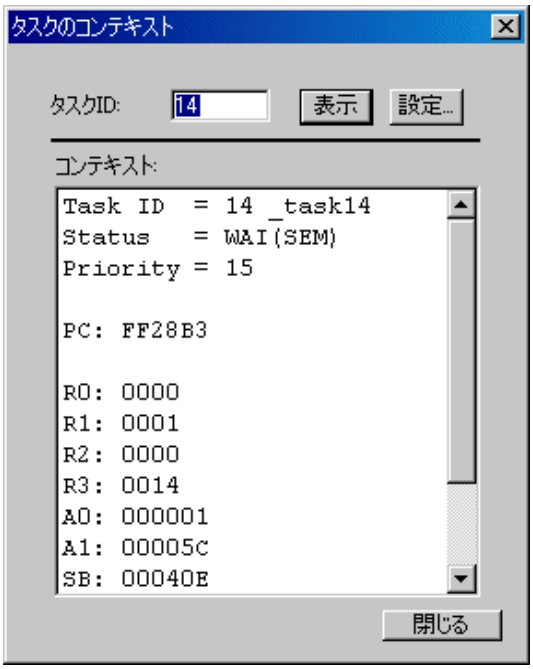
TA_TFIFO	待ちタスクのキューイングは FIFO
TA_TPRI	待ちタスクのキューイングは優先度順

- ID 領域は、固定長・任意長の違いにより、以下のように表示内容が変わります。
  - ・固定長の場合、文字列"[F]"とメモリプールの ID 番号を表示します。
  - ・任意長の場合、1 行目には文字列"[V]"、メモリプール ID 番号、ブロック ID 番号を表示します。 2～4 行目にはメモリプール ID 番号、ブロック ID 番号を表示します。 ブロック ID 番号はカッコで囲んで表示します。

6.14.12 タスクのコンテキストを参照/設定する

6.14.12.1 タスクのコンテキストを参照する

MR ウィンドウでポップアップメニュー[コンテキスト表示...]を選択してください。
以下のダイアログがオープンします。この[タスクのコンテキスト]ダイアログは、指定タスクの コンテキスト情報を参照/設定するためのダイアログです。
このダイアログは、タスク状態表示モードでデータ表示部分を ダブルクリックすることによりオープンすることもできます。

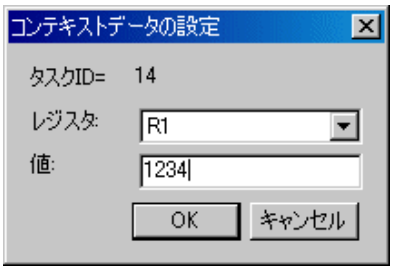


[タスク ID]領域にタスク ID 番号を入力し、[表示]ボタンをクリック(または Enter キー入力) してください。
[コンテキスト]領域に指定タスクのコンテキストが表示されます。

- [表示]ボタンクリック時、[タスク ID]領域に入力したタスクが"RUN"または"DMT"状態の 場合は、コンテキストは表示されません ([コンテキスト]領域には、タスク ID とタスクの状態のみが表示されます)。
- [表示]ボタンクリック時、[タスク ID]領域に存在しないタスク ID 番号を入力した場合は、 エラーとなります。

6.14.12.2 タスクのコンテキストを変更する

[タスクのコンテキスト]ダイアログの[タスク ID]領域にタスク ID 番号を入力し、[設定]ボタンをクリックしてください。
以下のダイアログがオープンします。 この[コンテキストデータの設定]ダイアログは、指定タスクの指定コンテキストレジスタ値を設定するためのダイアログです。



[レジスタ]領域のリストボックスで変更するレジスタを指定し、[値]領域に設定する値を 入力してください。
[値]領域に設定した式の記述に誤りがあった場合、指定レジスタに設定できる値の範 囲を超えた場合などには、エラーとなります。

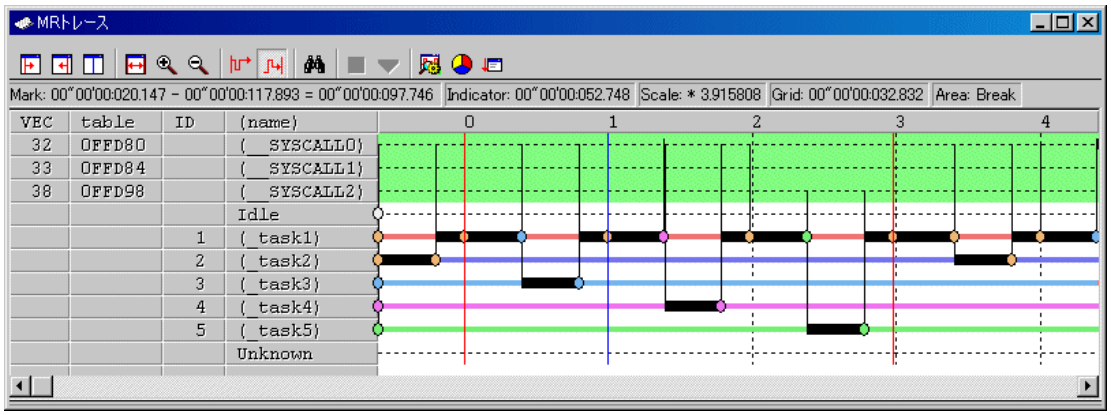
6.15 MR トレースウィンドウ

MR トレースウィンドウは、リアルタイム OS を使用したプログラムのタスク実行履歴等を計測しグラフィカルに表示するウィンドウです。
R32C 用デバッガでは、サポートしていません。
740 用デバッガでは、サポートしていません。

タスク実行履歴の他に、割り込み処理・タスク状態遷移・システムコール発行の各履歴も同時に計測し表示します。
弊社リアルタイム OS(MRxx)を使用したターゲットプログラムをダウンロードした場合のみ使用できます。

MR30 の場合
- MR30 V.2.00 以上を対象とします。MR30 V.1.00 で作成されたターゲットプログ ラムをダウンロードした場合は、MR トレースウィンドウは機能せず何も表示しません。

MR308 の場合
- 高速割り込み処理の履歴は計測、表示できません。



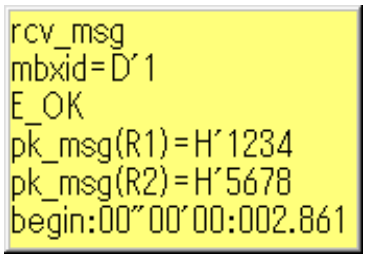
各項目の内容は、以下の通りです。

項目	内容
VEC*1	ソフトウェア割り込み番号を表示します。
table	割り込みベクタテーブル番地を表示します。
ID	タスクの ID 番号を表示します。
(name)	割り込みルーチン名、タスク名、アイドル処理("idle"と表示)、不明("unknown"と表示)を表示します。

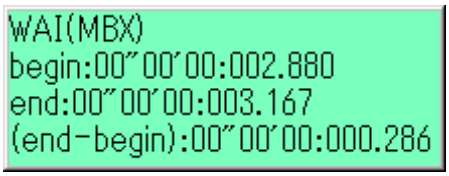
ウィンドウに表示された各情報にマウスを移動することにより、以下のようなポップアップウィンドウをオープン し詳細な情報を表示します。

割り込み処理・タスク実行履歴の詳細表示情報

システムコール発行履歴の詳細表示情報



タスク状態遷移履歴の詳細表示情報



ステータスバーには、以下の情報を表示します。

- 始点マーカー位置の時刻値
- 終点マーカー位置の時刻値
- 始点マーカー、終点マーカー間の時間幅
- インジケーター位置の時刻値
- 表示倍率
- グリッド線間隔時間幅
- 計測(トレース)範囲

グリッド線は、始点マーカーを基点として表示しています。目盛りは始点マーカーが位置する時刻を 0 として、左側(時間的に前方)を負、右側(時間的に後方)を正にして表 示しています。
グリッド線により、割り込み発生周期や処理時間等をおおまかに把握することができます。
表示しているグリッド線の間隔時間幅は、ステータスバーの"Grid"領域に示します。

MR トレースウィンドウでの時刻値は、すべてプログラム実行開始時点を 0 とする実行経過 時間を意味します。
これに対し、MR トレースウィンドウのグリッド線(目盛り)上部の数字は、始点マーカ ーを 0 とする相対値(グリッド間隔は、設定ダイアログで指定)であり、時刻値とは関 係ありません(ウィンドウを見易くするためのものです)。

補足事項

VEC 列*1 のソフトウェア割り込み番号は、製品によって異なります。
どのシステムコールがどの割り込み番号に割り当てられているかは、 MRxx のリファレンスマニュアルを参照ください。

6.15.1 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
始点マーカー		始点マーカーを表示画面内へ移動します。
終点マーカー		終点マーカーを表示画面内へ移動します。
現在位置マーカー		現在位置マーカー(インジケーター)の表示画面内への移動します。
表示倍率の調整		始点/終点マーカーの範囲を横幅一杯に表示します。
表示倍率の拡大		表示倍率を拡大します。
表示倍率の縮小		表示倍率を縮小します。
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。
検索...		システムコール発行履歴を検索します。
計測範囲条件	After	計測範囲条件を After に設定します。
	Break	計測範囲条件を Break に設定します。
設定...		グリッド間隔、表示倍率を設定します。
表示色の設定...		各種表示色を設定します。
表示順序の初期化...		表示順序を初期化します。
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。

6.15.2 タスクの実行履歴を参照する(MRxx ウィンドウ)

タスクの実行履歴は、MR トレースウィンドウで参照します。また、実行履歴の統計処理結果は、MR アナライズウィンドウで参照します。
これらのウィンドウは、弊社リアルタイム OS(MRxx)を使用したターゲットプログラムの場合に使用できます。

6.15.2.1 トレース範囲を選択し、プログラムを実行する

リアルタイム OS に依存する部分をデバッグするための事前準備が完了していれば、タスクの実行履歴を参照することができます。 MR トレースウィンドウでトレース範囲を選択してください。

MR トレースウィンドウの After ボタン(メニューでは、[計測範囲条件]→[After])または、Break ボタン(メニューでは、[計測範囲条件]→[Break])をクリックしてください。

After	リアルタイムトレースのトレースイベント成立からのタスク実行履歴を記録
Break	ターゲットプログラム停止以前のタスク実行履歴を記録

ターゲットプログラムを実行し、タスクの実行履歴を記録してください。

注意事項

トレースポイント設定ウィンドウで設定したトレースポイントは無効になります。

6.15.2.2 タスクの実行履歴計測を中断する

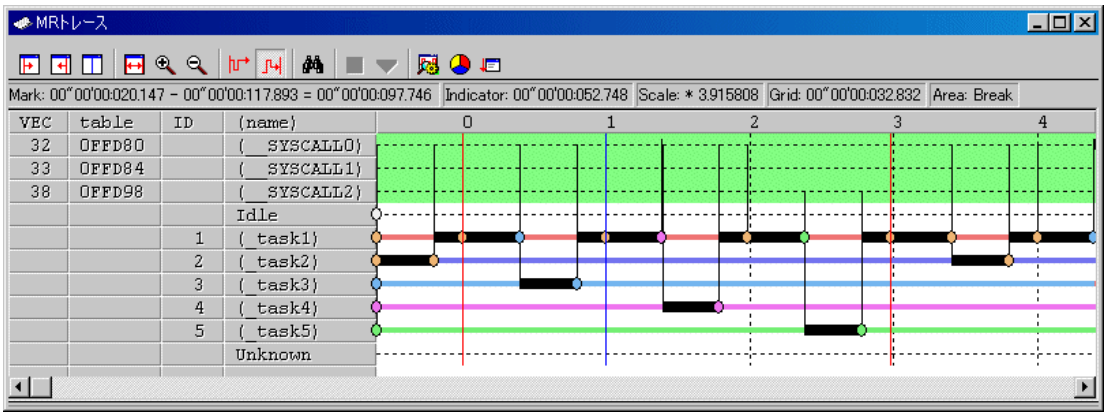
MR トレースウィンドウのツールバーから"計測中断"ボタンをクリックして下さい（メニューの場合、[計測中断])。それまでの計測結果を MR トレースウィンドウに表示します。

6.15.2.3 タスクの実行履歴計測を再開する

MR トレースウィンドウのツールバーから"再計測"ボタンをクリックして下さい（メニューの場合、[再計測])。それまでの計測結果はすべて削除されます。

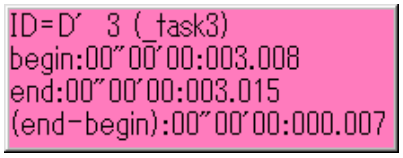
6.15.2.4 タスクの実行遷移を参照する

タスクの実行遷移は、MR トレースウィンドウで参照します。

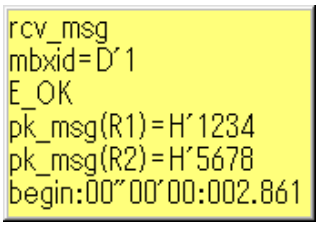


ウィンドウに表示された各情報にマウスを移動することで、以下の例のようなウィンドウがオープンし、詳細な情報を表示します。

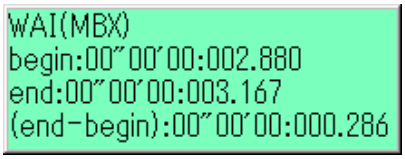
割り込み処理・タスク実行履歴の詳細情報表示



システムコール発行履歴の詳細情報表示

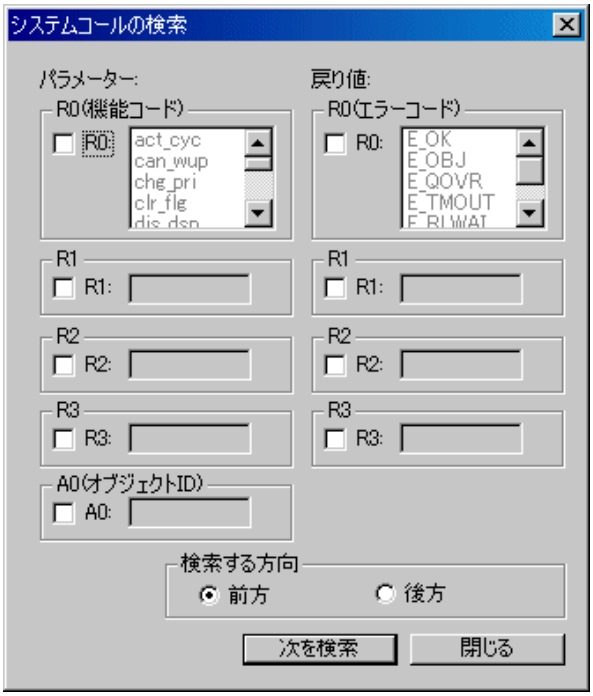


タスク状態遷移履歴の詳細情報表示



6.15.2.4.1. システムコールの発行履歴を検索する

ツールバーの"検索"ボタンをクリックしてください。システムコールの検索ダイアログがオープンします(メニューでは、[検索...])。

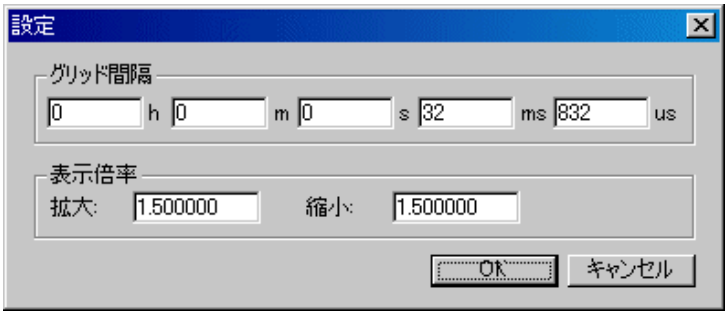


検索条件を指定してください。
機能コード、エラーコードでは、複数の値を指定することができます(OR 条件)。
それ以外の検索項目は、AND 条件で検索します。
次に検索方向を指定してください。 インジケータが指し示す位置を基点として、ダイアログで指定した方向に検索します。
すべての検索項目がチェックされなかった場合は、検索方向にある次のシステムコール発行履歴が検索結果となります。
"次を検索"ボタンをクリックしてください。指定した条件に該当するシステムコールの発行履歴を検索します。 指定した各項目は、AND 条件で検索します。
検索条件に一致した場合、その位置にインジケータを移動します。

6.15.2.4.2. 表示倍率を変更する

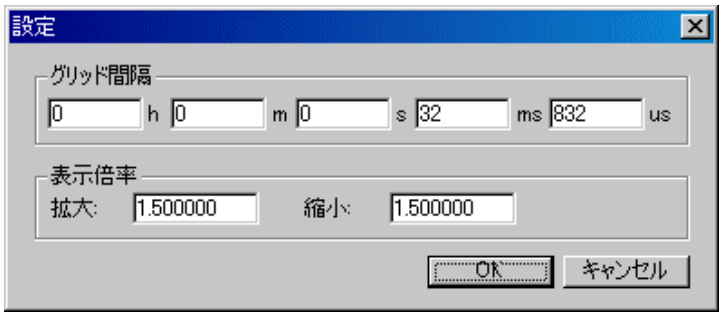
ツールバーの"表示倍率の拡大"ボタンもしくは"表示倍率の縮小"ボタンをクリックしてください（メニューでは、それぞれ[表示倍率の拡大]、[表示倍率の縮小]）。
グラフ表示領域の左端を基点として表示を拡大/縮小します。 デフォルトでは 1.5 倍ずつ拡大/縮小して表示します。
表示倍率は、ステータスバーの"Scale:*"領域に示します。

拡大/縮小率のデフォルトは、1.5 倍です。拡大/縮小率を変更するには、 メニュー[設定...]を選択してください。
設定ダイアログがオープンします。表示拡大率/縮小率を指定してください。



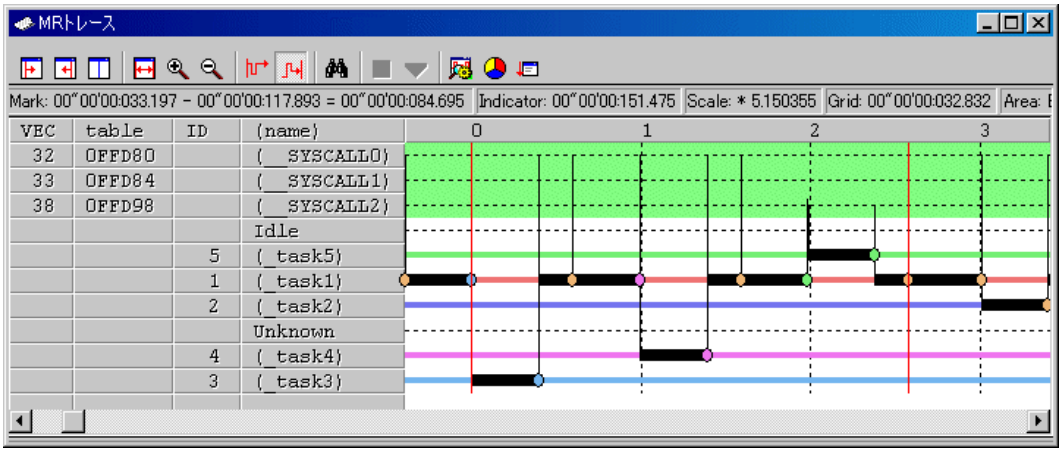
6.15.2.4.3. グリッド線の表示間隔を変更する

メニュー[設定...]を選択してください。設定ダイアログがオープンします。
グリッド間隔の時間幅を指定してください。



6.15.2.4.4. タスクの表示順序を変更する

移動するタスク・割り込みルーチン(グラフ表示の左側部分)を移動先までドラッグしてください。



表示順序を初期化するには、メニュー[表示順序の初期化]を選択してください。

6.15.2.4.5. 特定タスクのみを表示する

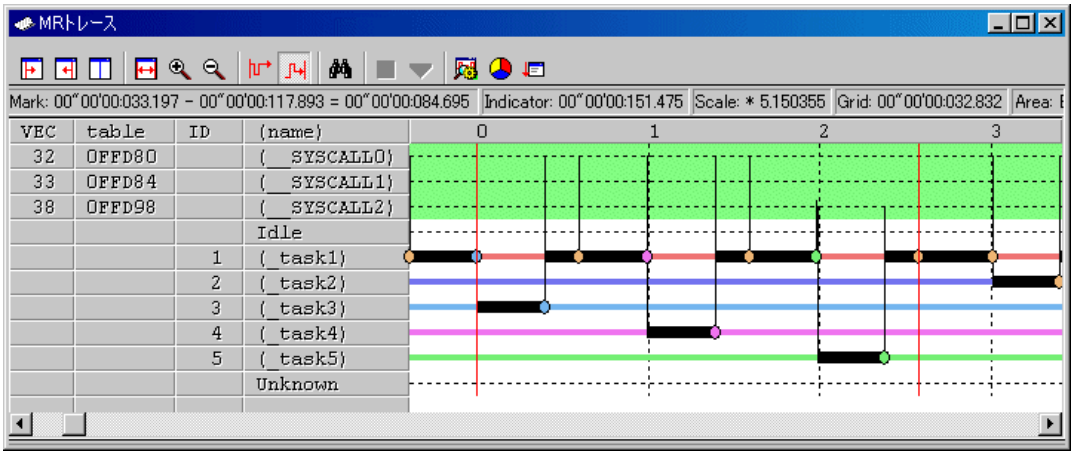
表示しないタスク・割り込みルーチン(グラフ表示の左側部分)をクリックしてください。
クリックするごとに表示/非表示が切り換わります。

6.15.2.4.6. 表示カラーを変更する

メニュー[表示色の設定...]を選択してください。表示色の設定ダイアログがオープンします。
各項目に対応したボタンをクリックしてください。色の設定ダイアログがオープンしますので 表示色を変更してください。

6.15.2.5 タスクの実行時間を計測する

MR トレースウィンドウの始点マーカー、終点マーカー位置を変更することにより、 マーカー間の実行時間を計測することができます。



始点マーカー位置、及び終点マーカー位置をドラッグしてください。
ステータスバーにマーカー間の時間幅を表示します。

補足事項

[MR トレースウィンドウの時刻値の定義]
MR トレースウィンドウでの時刻値は、すべてプログラム実行開始時点を 0 とする実行経 過時間を意味します。
これに対し、MR トレースウィンドウのグリッド線(目盛り)上部の数字は、始点マー カーを 0 とする相対値(グリッド間隔は、設定ダイアログで指定)であり、 時刻値とは関係ありません(ウィンドウを見やすくするためのものです)。

6.15.2.5.1. マーカーを移動する

各マーカーは、ドラッグにより移動ができます。 マーカー上にマウスを移動するとカーソルが変化しますので、その状態でドラッグしてください。

始点マーカーは、ツールバーの"始点マーカー"ボタンをクリックすることにより、ウィンドウ内(左部)へ 移動します(メニューでは、[始点マーカー])。
終点マーカーは、"終点マーカー"ボタンをクリックすることにより、ウィンドウ内(右部)へ移動します(メニューでは、[終点マーカー])。
インジケータは、"現在位置マーカー"ボタンをクリックすることにより、 ウィンドウ内(中央部)へ移動します(メニューでは、[現在位置マーカー])。
ただし、各マーカーは、以下の場所にのみ、移動することができます。

- 割り込み処理・タスク実行が遷移した位置
- タスク状態が遷移した位置
- システムコール発行履歴表示位置

6.16 MR アナライズウィンドウ

MR アナライズウィンドウは、MR トレースウィンドウの始点マーカーと終点マーカーで指定された範 囲の計測データを統計処理した結果を表示するウィンドウです。
R32C 用デバッガでは、サポートしていません。
740 用デバッガでは、サポートしていません。

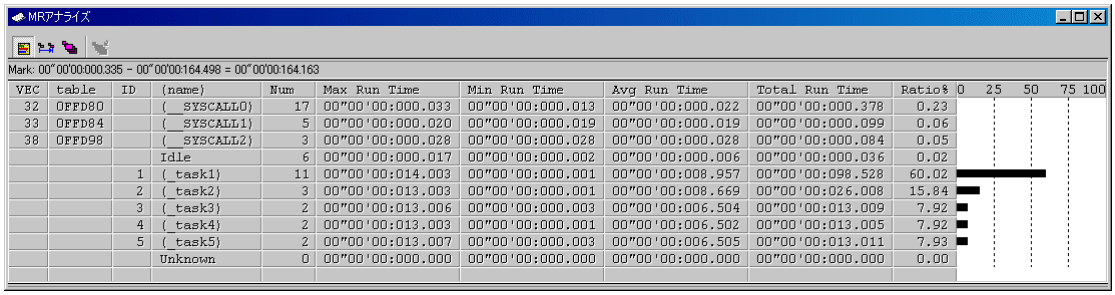
MR アナライズウィンドウでは、以下の 3 つの表示モードをサポートしています。

- 割り込み処理ごと・タスクごとの CPU 占有状況
- タスクごとのレディ状態時間
- システムコール発行履歴の一覧表示(特定条件指定による抽出表示可能)

MR アナライズウィンドウは、MR トレースウィンドウと共に機能します。
弊社リアルタイム OS(MRxx)を使用したターゲットプログラムをダウンロードした場合のみ使用できます。

6.16.1 CPU 占有状況表示モードの構成

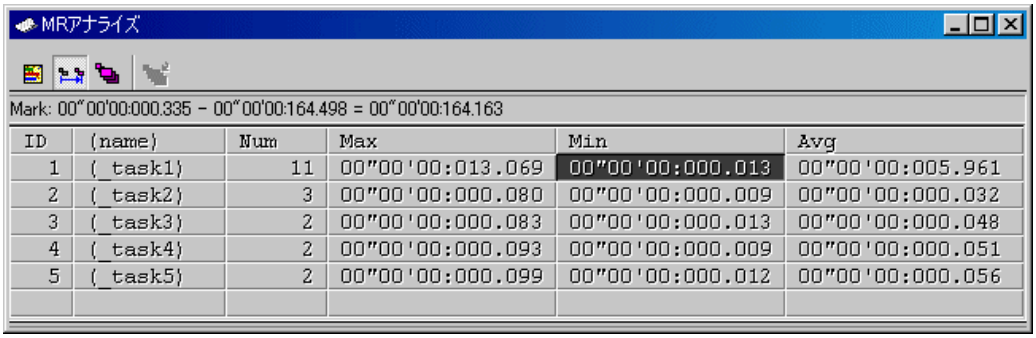
CPU 占有状況表示モードは、割り込み処理ごと・タスクごとの CPU 占有時間と比率を表示するためのモードです。
MR トレースウィンドウで始点マーカーと終点マーカーで指定した範囲内での統計結果を表示します。



各行の最大実行時間・最小実行時間表示領域をクリックすることで、クリックした行に対応する割り 込み処理もしくはタスクの最大実行時間・最小実行時間の処理履歴を検索することが可能です。
検索結果は、MR トレースウィンドウのインジケータが対象位置に移動して指示します。

6.16.2 タスクごとのレディ状態時間表示モードの構成

タスクごとのレディ状態時間表示モードは、タスクごとの実行可能状態から実行状態に遷移するまでの時間 を統計処理した結果を表示するためのモードです。
MR トレースウィンドウで始点マーカーと終点マーカーで指定した範囲内での統計結果を表示します。

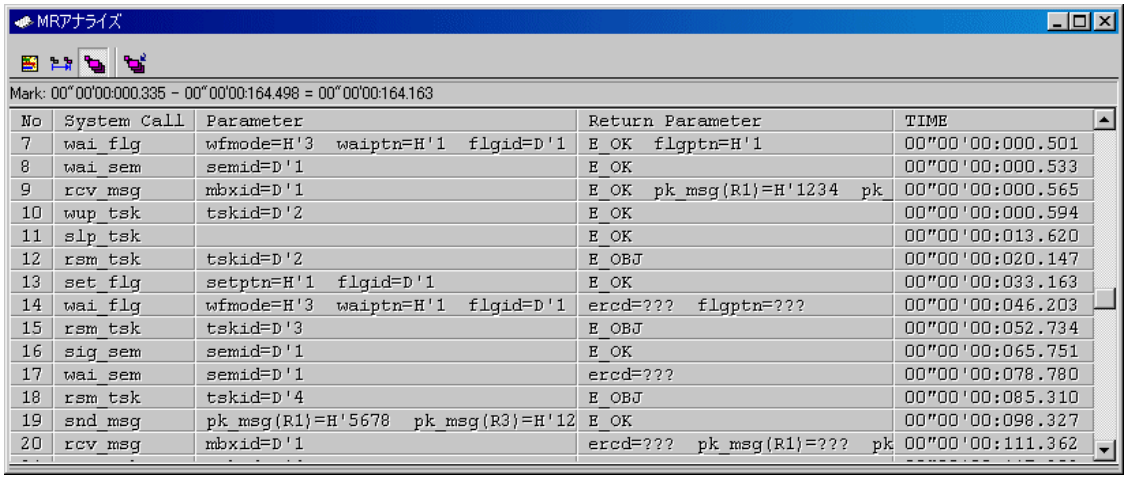


ID	(name)	Num	Max	Min	Avg
1	(_task1)	11	00"00'00:013.069	00"00'00:000.013	00"00'00:005.961
2	(_task2)	3	00"00'00:000.080	00"00'00:000.009	00"00'00:000.032
3	(_task3)	2	00"00'00:000.083	00"00'00:000.013	00"00'00:000.048
4	(_task4)	2	00"00'00:000.093	00"00'00:000.009	00"00'00:000.051
5	(_task5)	2	00"00'00:000.099	00"00'00:000.012	00"00'00:000.056

各行の最大レディ時間・最小レディ時間表示領域をクリックすることで、クリックした行に対応する タスクの最大レディ時間・最小レディ時間の処理履歴を検索することが可能です。
検索結果は、MR トレースウィンドウのインジケータが対象位置に移動して指示します。

6.16.3 システムコール発行履歴の一覧表示モードの構成

システムコール発行履歴の一覧表示モードは、発行されたシステムコールのリストを表示するためのモ ードです。
MR トレースウィンドウで始点マーカーと終点マーカーで指定した範囲内でのシステムコール 発行履歴の一覧をリスト形式で表示します。
ただし、番号は計測できた範囲内で先頭のシステムコールから数えた数値を示します。



No	System Call	Parameter	Return Parameter	TIME
7	wai_flg	wfmode=H'3 waiptn=H'1 flgid=D'1	E_OK flgptn=H'1	00"00'00:000.501
8	wai_sem	semid=D'1	E_OK	00"00'00:000.533
9	rcv_msg	mbxid=D'1	E_OK pk_msg(R1)=H'1234 pk_	00"00'00:000.565
10	wup_tsk	tskid=D'2	E_OK	00"00'00:000.594
11	slp_tsk		E_OK	00"00'00:013.620
12	rsm_tsk	tskid=D'2	E_OBJ	00"00'00:020.147
13	set_flg	setptn=H'1 flgid=D'1	E_OK	00"00'00:033.163
14	wai_flg	wfmode=H'3 waiptn=H'1 flgid=D'1	ercd=??? flgptn=???	00"00'00:046.203
15	rsm_tsk	tskid=D'3	E_OBJ	00"00'00:052.734
16	sig_sem	semid=D'1	E_OK	00"00'00:065.751
17	wai_sem	semid=D'1	ercd=???	00"00'00:078.780
18	rsm_tsk	tskid=D'4	E_OBJ	00"00'00:085.310
19	snd_msg	pk_msg(R1)=H'5678 pk_msg(R3)=H'12	E_OK	00"00'00:098.327
20	rcv_msg	mbxid=D'1	ercd=??? pk_msg(R1)=??? pk	00"00'00:111.362

各行をクリックすることで、クリックした行に対応するシステムコール発行履歴を検索することが可 能です。
検索結果は、MR トレースウィンドウのインジケータが対象位置に移動して指示します。

6.16.4 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
CPU 占有状況	CPU 占有状況を表示します。
レディー状態時間	タスクごとのレディ状態時間を表示します。
システムコール発行履歴の一覧表示	システムコール発行履歴の一覧を表示します。
システムコール発行履歴の抽出...	システムコール発行履歴の一覧を、特定条件指定により抽出します。
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。

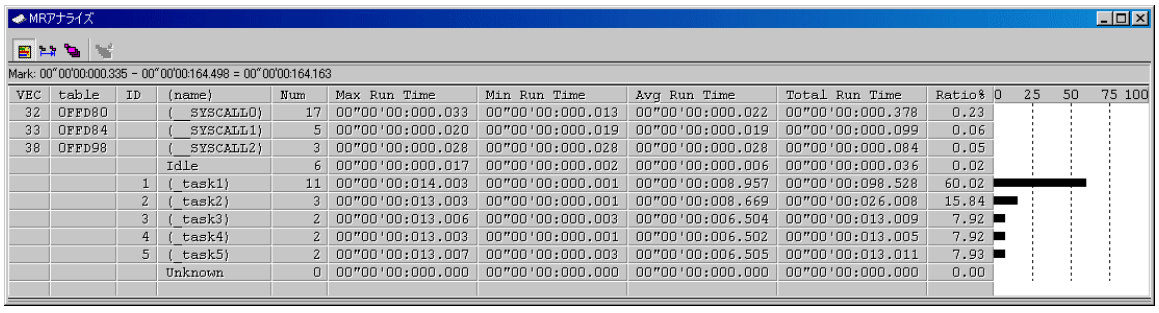
6.16.5 タスクの実行履歴を統計処理する

実行履歴の統計処理は、MR アナライズウィンドウで参照します。
MR アナライズウィンドウは、MR トレースウィンドウと共に機能します。MR トレースウィンドウがオープンしていない場合、 及び MR トレースウィンドウ上に何も表示していない場合は、機能しません。

実行履歴の統計処理では、以下の内容を参照することができます。

6.16.5.1 CPU 占有状況を参照する

ツールバーの"CPU 占有状況"ボタンをクリックしてください(メニューでは、[CPU 占有状況])。 MR アナライズウィンドウが CPU 占有状況表示モードに変わります。

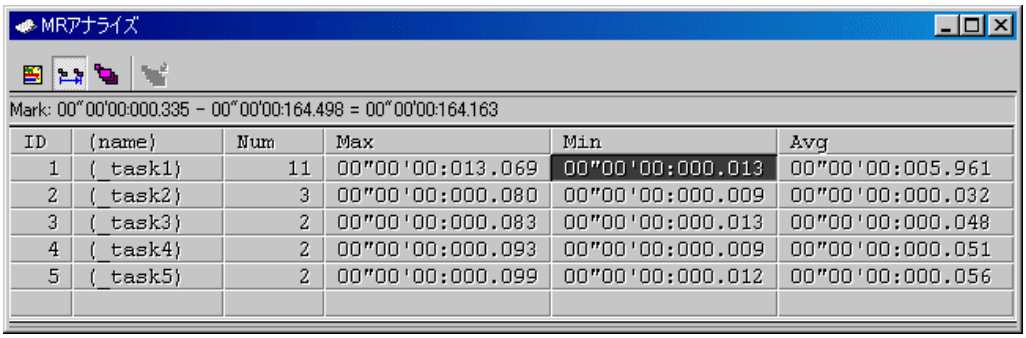


割り込み処理ごと・タスクごとの CPU 占有時間と比率を表示します。
表示内容は、MR トレースウィンドウの始点マーカー、終点マーカーで指定した範囲の統計結果です。

各行の最大実行時間・最小実行時間表示領域をクリックすることにより、 クリックした行に対応するタスクの最大実行時間・最小実行時間の処理履歴を 検索することが可能です。
検索結果は、MR トレースウィンドウのインジケータが対象位置に移動して指示します。

6.16.5.2 レディ状態時間を参照する

ツールバーの"レディ状態時間"ボタンをクリックしてください(メニューでは、[レディ状態時間])。



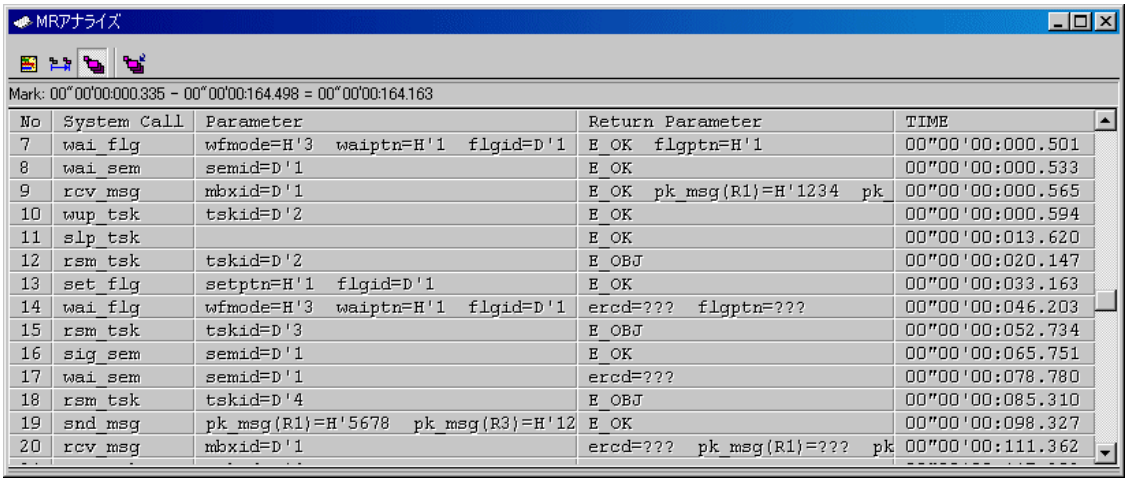
ID	(name)	Num	Max	Min	Avg
1	(_task1)	11	00"00'00:013.069	00"00'00:000.013	00"00'00:005.961
2	(_task2)	3	00"00'00:000.080	00"00'00:000.009	00"00'00:000.032
3	(_task3)	2	00"00'00:000.083	00"00'00:000.013	00"00'00:000.048
4	(_task4)	2	00"00'00:000.093	00"00'00:000.009	00"00'00:000.051
5	(_task5)	2	00"00'00:000.099	00"00'00:000.012	00"00'00:000.056

タスクごとの実行可能状態から実行状態に遷移するまでの時間を統計処理し、表示します。
表示内容は、MR トレースウィンドウの始点マーカー、終点マーカーで指定した範囲の統計結果です。

各行の最大レディ時間・最小レディ時間表示領域をクリックすることにより、 クリックした行に対応するタスクの最大レディ時間・最小レディ時間の処理履歴を 検索することが可能です。
検索結果は、MR トレースウィンドウのインジケータが対象位置に移動して指示します。

6.16.5.3 システムコール発行履歴を参照する

ツールバーの"システムコール発行履歴の一覧表示"ボタンをクリックしてください(メニューでは、[システムコール発行履歴の一覧表示])。



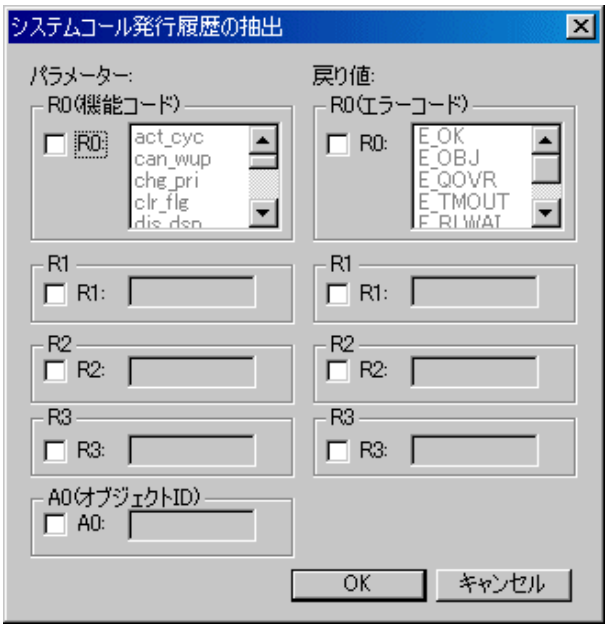
No	System Call	Parameter	Return Parameter	TIME
7	wai_flg	wfmode=H'3 waiptn=H'1 flgid=D'1	E_OK flgptn=H'1	00"00'00:000.501
8	wai_sem	semid=D'1	E_OK	00"00'00:000.533
9	rcv_msg	mbxid=D'1	E_OK pk_msg(R1)=H'1234 pk_	00"00'00:000.565
10	wup_tsk	tskid=D'2	E_OK	00"00'00:000.594
11	slp_tsk		E_OK	00"00'00:013.620
12	rsm_tsk	tskid=D'2	E_OBJ	00"00'00:020.147
13	set_flg	setptn=H'1 flgid=D'1	E_OK	00"00'00:033.163
14	wai_flg	wfmode=H'3 waiptn=H'1 flgid=D'1	ercd=??? flgptn=???	00"00'00:046.203
15	rsm_tsk	tskid=D'3	E_OBJ	00"00'00:052.734
16	sig_sem	semid=D'1	E_OK	00"00'00:065.751
17	wai_sem	semid=D'1	ercd=???	00"00'00:078.780
18	rsm_tsk	tskid=D'4	E_OBJ	00"00'00:085.310
19	snd_msg	pk_msg(R1)=H'5678 pk_msg(R3)=H'12	E_OK	00"00'00:098.327
20	rcv_msg	mbxid=D'1	ercd=??? pk_msg(R1)=??? pk	00"00'00:111.362

発行されたシステムコールのリストをシステムコール順に表示します。
表示内容は、MR トレースウィンドウの始点マーカー、終点マーカーで指定した範囲の統計結果です。

各行をクリックすることにより、クリックした行に対応するシステムコール発行履歴を検 索することが可能です。検索結果は、MR トレースウィンドウのインジケータが対象 位置に移動して指示します。

6.16.5.3.1. 発行履歴を抽出する

ツールバーから"システムコール発行履歴の抽出"ボタンをクリックしてください(メニューでは[システムコール発行履歴の抽出])。 以下のダイアログがオープンします。抽出して表示するシステムコールの検索条件を指定してください。

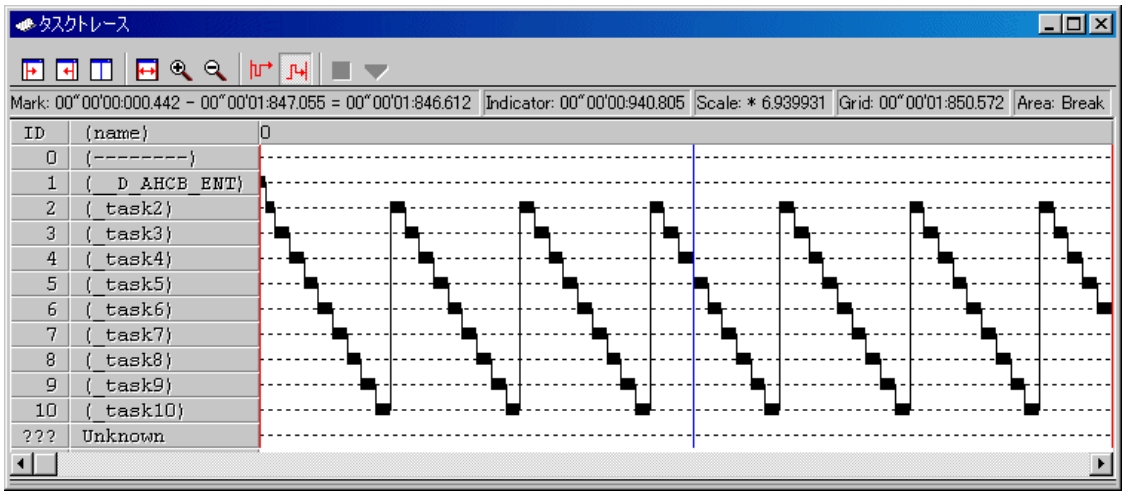


指定した条件に該当するシステムコールの発行履歴を抽出し、表示します。

6.17 タスクトレースウィンドウ

タスクトレースウィンドウは、リアルタイム OS を使用したプログラムのタスク実行履歴を計測しグラフィカルに表示するウィンドウです。
弊社リアルタイム OS(MRxx)以外の OS を使用したターゲットプログラムをダウンロードした場合でも使用できます。

R32C 用デバッガでは、サポートしていません。
740 用デバッガでは、サポートしていません。



各項目の内容は、以下の通りです。

項目	内容
ID	タスクの ID 番号を表示します。
(name)	割り込みルーチン名、タスク名、アイドル処理("idle"と表示)、不明("unknown"と表示)を表示します。

ウィンドウに表示された各情報にマウスを移動することにより、以下のようなポップアップウィンドウをオープンし詳細な情報を表示します。

タスク実行履歴詳細情報

```
ID=D'   7 (_task7)
begin:00"00'00:722.055
end:00"00'00:753.305
(end-begin):00"00'00:031.250
```

ステータスバーには、以下の情報を表示します。

- 始点マーカー位置の時刻値
- 終点マーカー位置の時刻値
- 始点マーカー、終点マーカー間の時間幅
- 現在位置マーカー位置の時刻値
- 表示倍率
- グリッド線間隔時間幅
- 計測(トレース)範囲

グリッド線は、始点マーカーを基点として表示しています。
目盛りは始点マーカーが位置する時刻を 0 として、左側(時間的に前方)を負、右側(時間的に後方)を正にして表示しています。
グリッド線により、割り込み発生周期や処理時間等をおおまかに把握することができます。
表示しているグリッド線の間隔時間幅は、ステータスバーの"Grid"領域に示します。

タスクトレースウィンドウでの時刻値は、すべてプログラム実行開始時点を 0 とする実行経過 時間を意味します。
これに対し、タスクトレースウィンドウのグリッド線(目盛り)上部の数字は、開始マーカ ーを 0 とする相対値(グリッド間隔は、Value ダイアログで指定)であり、時刻値とは関 係ありません(ウィンドウを見易くするためのものです)。

6.17.1 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
始点マーカー		始点マーカーを表示領域に移動します
終点マーカー		終点マーカーを表示領域に移動します
現在位置マーカー		現在位置マーカーを表示領域に移動します
表示倍率の調整		始点/終点マーカーの範囲を横幅一杯に表示します
表示倍率の拡大		表示倍率を拡大します
表示倍率の縮小		表示倍率を縮小します
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
計測範囲条件	After	トレース計測範囲条件を After に設定します
	Break	トレース計測範囲条件を Break に設定します
設定		グリッド間隔、表意倍率を設定します
表示色の設定		各種表示色を設定します。
リアルタイム OS 情報の設定		ご使用のリアルタイム OS に関する情報を設定します
ツールバー表示		ツールバーの表示/非表示を切り替えます
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

6.17.2 タスクの実行履歴を参照する(Taskxx ウィンドウ)

タスクの実行履歴は、タスクトレースウィンドウで参照します。
また、実行履歴の統計処理結果は、タスクアナライズウィンドウで参照します。
これらのウィンドウは、弊社リアルタイム OS(MRxx)以外の OS を使用したターゲットプログラムの場合にも使用できます。

6.17.2.1 タスクの実行履歴の計測を準備する

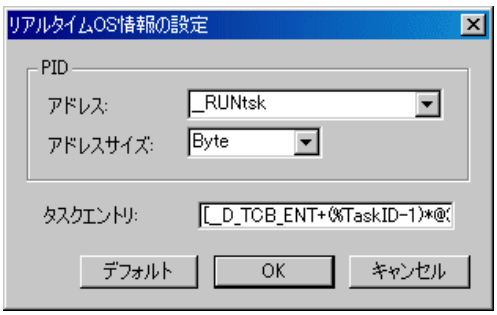
リアルタイム OS を使用したプログラムのタスク実行履歴等を計測するには、 タスクトレースウィンドウにおいてトレース範囲を選択した後、ターゲットプログラムを実行する必要があります。

6.17.2.1.1. 対象リアルタイム OS の情報を設定する

タスクトレースウィンドウを使用するためには、ダウンロードしたプログラムが使用しているリアルタ イム OS(対象リアルタイム OS)に関する以下の情報を設定する必要があります。

- 実行タスク ID 格納領域のラベル名(アドレス値)とそのサイズ
- タスクの開始アドレス計算式

タスクトレースウィンドウをオープンします。 メニュー[表示]→[RTOS]→[タスクトレース]を選択してください。
このメニューを PDxx 起動後、はじめて選択した場合、タスクトレースウィンドウがオープンする前に リアルタイム OS 情報の設定ダイアログがオープンします。



- 弊社リアルタイム OS(MRxx)を使用している場合

1. デフォルトボタンをクリックしてください。MRxx 用の情報が設定されます。
2. OK ボタンをクリックしてください。タスクトレースウィンドウがオープンします。

- MRxx 以外のリアルタイム OS を使用している場合

1. PID のアドレス領域で実行タスク ID 格納領域のラベル(アドレス指定も可能)、PID のアクセスサイズリストボックスで 実行タスク ID 格納領域のサイズを指定してください。
   この情報が正しく設定されていない場合は、タスクトレースウィンドウが使用できません。
2. タスクエントリ領域にタスクの開始アドレス計算式を指定します。
   式は、「式の記述方法」に従った書式で記述してください。
   またタスク ID 番号を代入するべき位置には、マクロ変数「%TaskID」を使用します。 この情報が正しく設定されていない場合、タスクトレースウィンドウでのタスク名が表示できません。
3. OK ボタンをクリックしてください。タスクトレースウィンドウがオープンします。

740 用 デバッガ では、デフォルトボタンをクリックすると OSEK OS 用の情報が設定されます。
このダイアログで 1 度リアルタイム OS 情報を設定すると、次回からはその情報が有効になります。
設定内容を変更したい場合は、ポップアップメニュー→[RTOS...]を選択してください。 リアルタイム OS 情報の設定ダイアログが再オープンします。

注意事項

リアルタイム OS 情報の設定ダイアログの PID 設定で、アクセスサイズに Word を指定する場合は、以下の制限があります（これらの条件を満たさない場合、正常に動作しません)。

- PID 情報格納領域が偶数アドレスに割り付けられている。
- PID 情報格納領域が 16 ビットバス幅でアクセスされる領域に割り付けられている。

6.17.2.1.2. トレース範囲を選択する

タスクの実行履歴計測には、リアルタイムトレース機能を使用しています。
タスクトレースウィンドウの After ボタン(ポップアップメニュー→[After])または、Break ボタン(ポップアップメニュー→[Break])をクリックしてください。

After	トレースメモリが記録データで満たされるまでのタスク実行履歴を記録
Break	ターゲットプログラム停止以前のタスク実行履歴(トレースメモリ分)を記録

トレースメモリには、タスクの実行履歴を知るために必要な特定のサイクルのみが記録されます。

注意事項

トレースポイント設定ウィンドウで設定したトレースポイントは、無効になります。

6.17.2.1.3. ターゲットプログラムを実行する

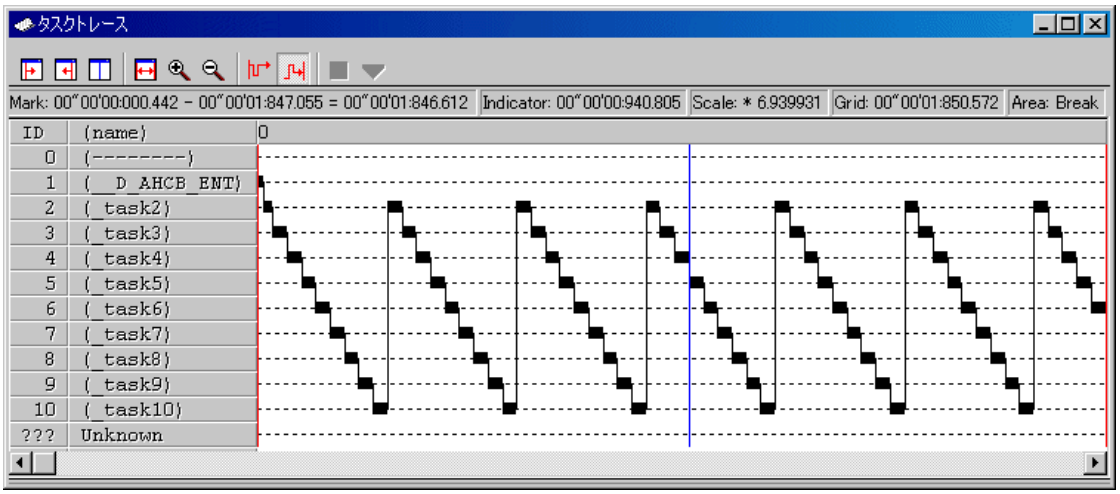
ターゲットプログラムを実行してください。タスクの実行履歴を知るために必要な情報をトレースメモリに記録します。
トレース範囲に After を選択した場合は、トレースメモリが満たされた直後、または、ターゲットプログラムが停止した直後 にタスクトレースウィンドウへ表示します。
トレース範囲に Break を選択した場合は、ターゲットプログラムが停止した直後にタスクトレースウィンドウへ表示します。

タスクの実行履歴計測は、中断させることができます。
中断させるには、タスクトレースウィンドウの Stop ボタンをクリックしてください(もしくはポップアップメニュー→[計測中断])。

タスクの実行履歴計測を再開するには、タスクトレースウィンドウの再計測ボタンをクリックしてください(もしくは、ポップアップメニュー→[再計測])。

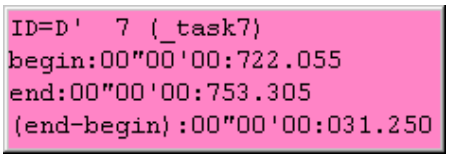
6.17.2.2 タスクの実行遷移を参照する

タスクの実行遷移は、タスクトレースウィンドウで参照します。



ウィンドウに表示された各情報にマウスを移動することで、以下の例のようなウィンドウがオープンし、詳細な情報を表示します。

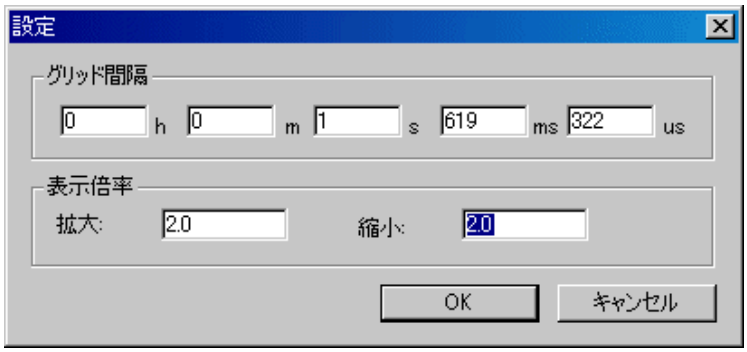
タスク実行履歴詳細情報表示



6.17.2.2.1. 表示倍率を変更する

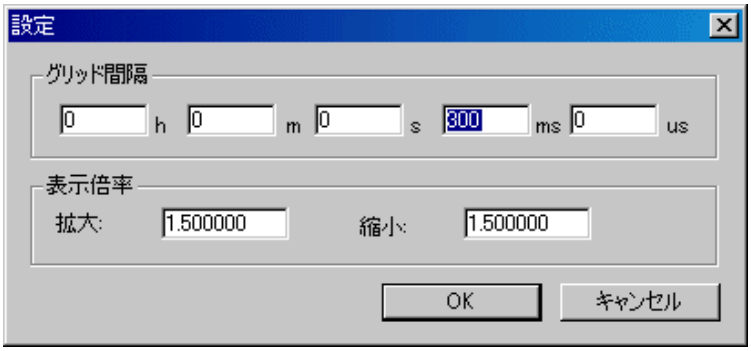
ツールバーの表示倍率の拡大ボタンもしくは表示倍率の縮小ボタンをクリックしてください (もしくは、ポップアップメニュー→[表示倍率の拡大]、→[表示倍率の縮小])。
グラフ表示領域の左端を基点として表示を拡大/縮小します。 デフォルトでは 1.5 倍ずつ拡大/縮小して表示します。
表示倍率は、ステータスバーの"Scale:*"領域に示します。

拡大/縮小率のデフォルトは、1.5 倍です。拡大/縮小率を変更するには、 ポップアップメニュー→[設定...]を選択してください。
設定ダイアログがオープンします。表示拡大率/縮小率を指定してください。



6.17.2.2.2. グリッド線の表示間隔を変更する

ポップアップメニュー→[設定...]を選択してください。設定ダイアログがオープンします。
表示間隔の時間幅を指定してください。

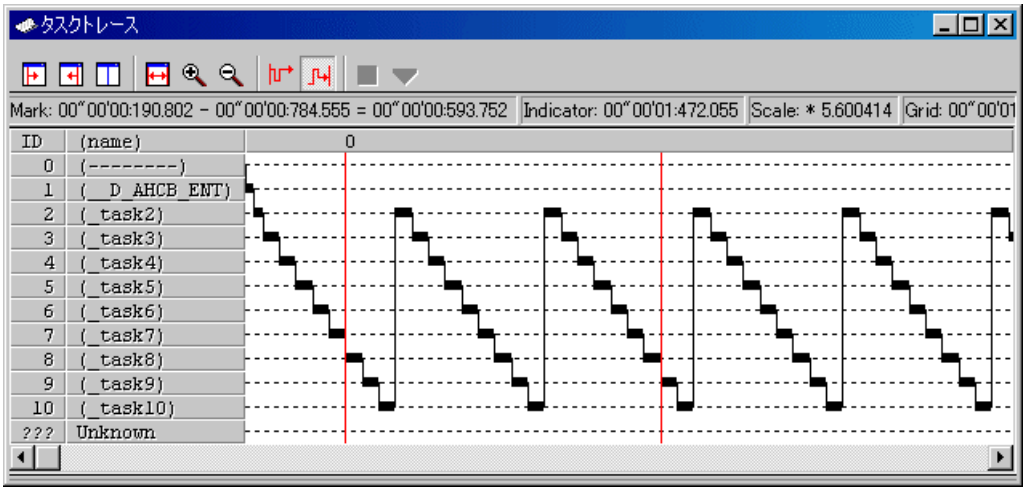


6.17.2.2.3. 表示カラーを変更する

ポップアップメニュー→[表示色の設定...]を選択してください。表示色の設定ダイアログがオープンします。
各項目に対応したボタンをクリックしてください。カラー設定ダイアログがオープンしますので 表示色を変更してください。

6.17.2.3 タスクの実行時間を計測する

タスクトレースウィンドウの始点マーカー、終点マーカー位置を変更することにより、 マーカー間の実行時間を計測することができます。



始点マーカー位置、及び終点マーカー位置をドラッグしてください。
ステータスバーにマーカー間の時間幅を表示します。

補足事項

- タスクトレースウィンドウの時刻値の定義
  タスクトレースウィンドウでの時刻値は、すべてプログラム実行開始時点を 0 とする実行経 過時間を意味します。
  これに対し、タスクトレースウィンドウのグリッド線(目盛り)上部の数字は、開始マー カーを 0 とする相対値(グリッド間隔は、設定ダイアログで指定)であり、 時刻値とは関係ありません(ウィンドウを見易くするためのものです)。

6.17.2.3.1. マーカーを移動する

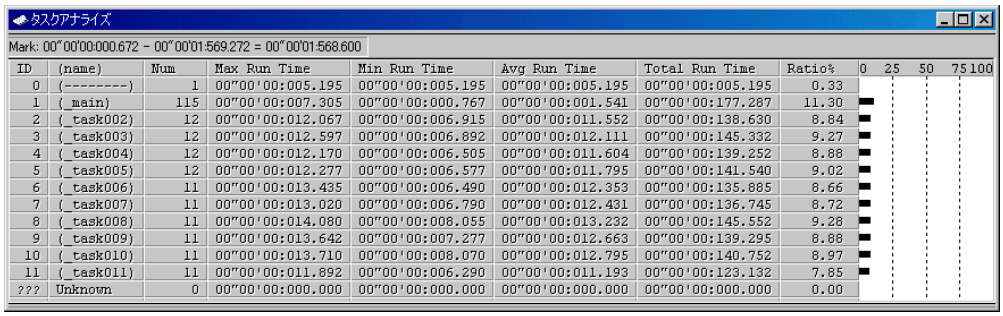
各マーカーは、ドラッグで移動できます。 マーカー上にマウスを移動するとカーソルが変化しますので、その状態でドラッグしてください。

始点マーカーは、ツールバーの始点マーカーボタンをクリックすることにより、ウィンドウ内(左部)へ 移動します(もしくは、ポップアップメニュー→[始点マーカー])。
終点マーカーは、終点マーカーボタンをクリックすることにより、ウィンドウ内(右部)へ移動します(もしくは、ポップアップメニュー→[終点マーカー])。
現在位置マーカーは、現在位置マーカーボタンをクリックすることにより、 ウィンドウ内(中央部)へ移動します(もしくは、ポップアップメニュー→[現在位置マーカー])。
ただし、各マーカーは、各イベントの成立点にのみ、移動することができます。

6.18 タスクアナライズウィンドウ

タスクアナライズウィンドウは、タスクトレースウィンドウの始点マーカーと終点マーカーで指定された範囲の 計測データを統計処理した結果を表示するウィンドウです。
タスクアナライズウィンドウでは、CPU 占有状況を表示します。
R32C 用デバッガでは、サポートしていません。
740 用デバッガでは、サポートしていません。

タスクアナライズウィンドウは、タスクトレースウィンドウと共に機能します。
弊社リアルタイム OS(MRxx)以外の OS を使用したターゲットプログラムをダウンロードしたでも使用できます。
CPU 占有状況表示モードは、タスクごとの CPU 占有時間と比率を表示するためのモードです。
タスクトレースウィンドウで始点マーカーと終点マーカーで指定した範囲内での統計結果を表示します。



各行の最大実行時間・最小実行時間表示領域をクリックすることで、クリックした行に対応する タスクの最大実行時間・最小実行時間の処理履歴を検索することが可能です。
検索結果は、タスクトレースウィンドウの現在位置マーカーが対象位置に移動して指示します。

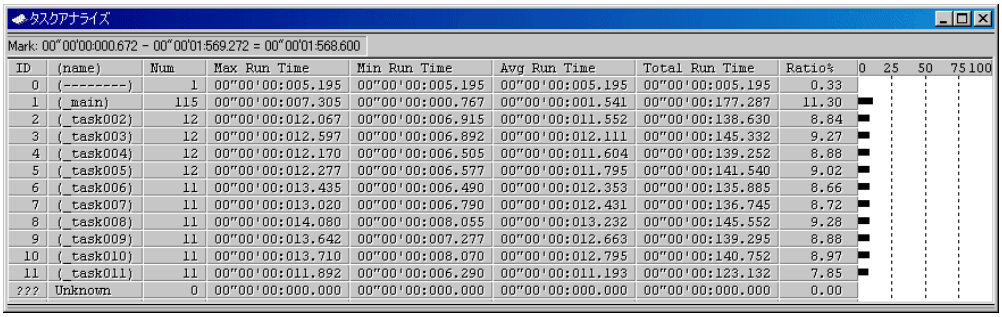
6.18.1 オプションメニュー

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
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。

6.18.2 タスクの実行履歴を統計処理する

実行履歴の統計処理は、タスクアナライズウィンドウで参照します。 このウィンドウは、タスクごとの CPU 占有時間と比率を表示します。
タスクアナライズウィンドウは、タスクトレースウィンドウと共に機能します。 タスクトレースウィンドウがオープンしていない場合、及びタスクトレースウィンドウ上に何も 表示していない場合は、機能しません。



ID	(name)	Num	Max Run Time	Min Run Time	Avg Run Time	Total Run Time	Ratio%	0 25 50 75 100
0	(--------)	1	00"00'00:005.195	00"00'00:005.195	00"00'00:005.195	00"00'00:005.195	0.33	
1	(_main)	115	00"00'00:007.305	00"00'00:000.767	00"00'00:001.541	00"00'00:177.287	11.30	
2	(_task002)	12	00"00'00:012.067	00"00'00:006.915	00"00'00:011.552	00"00'00:138.630	8.84	
3	(_task003)	12	00"00'00:012.597	00"00'00:006.892	00"00'00:012.111	00"00'00:145.332	9.27	
4	(_task004)	12	00"00'00:012.170	00"00'00:006.505	00"00'00:011.604	00"00'00:139.252	8.88	
5	(_task005)	12	00"00'00:012.277	00"00'00:006.577	00"00'00:011.795	00"00'00:141.540	9.02	
6	(_task006)	11	00"00'00:013.435	00"00'00:006.490	00"00'00:012.353	00"00'00:135.885	8.66	
7	(_task007)	11	00"00'00:013.020	00"00'00:006.790	00"00'00:012.431	00"00'00:136.745	8.72	
8	(_task008)	11	00"00'00:014.080	00"00'00:008.055	00"00'00:013.232	00"00'00:145.552	9.28	
9	(_task009)	11	00"00'00:013.642	00"00'00:007.277	00"00'00:012.663	00"00'00:139.295	8.88	
10	(_task010)	11	00"00'00:013.710	00"00'00:008.070	00"00'00:012.795	00"00'00:140.752	8.97	
11	(_task011)	11	00"00'00:011.892	00"00'00:006.290	00"00'00:011.193	00"00'00:123.132	7.85	
???	Unknown	0	00"00'00:000.000	00"00'00:000.000	00"00'00:000.000	00"00'00:000.000	0.00	

表示内容は、タスクトレースウィンドウの始点マーカー、終点マーカーで指定した範囲の統計結果です。

各行の最大実行時間・最小実行時間表示領域をクリックすることにより、 クリックした行に対応するタスクの最大実行時間・最小実行時間の処理履歴を検索することが可能です。
検索結果は、タスクトレースウィンドウの現在位置マーカーが対象位置に移動して指示します。

7. スクリプトコマンド一覧

本デバッガは、以下のスクリプトコマンドが使用できます。
網掛け表示しているスクリプトコマンドは、ランタイム実行可能です。
後ろに*の付いたコマンドは、製品によってはサポートしていません。

7.1 スクリプトコマンド一覧(機能順)

7.1.1 実行関連

コマンド名	短縮名	内容
Go	G	ターゲットプログラムの実行
GoFree	GF	ターゲットプログラムのフリーラン実行
GoProgramBreak*	GPB	ターゲットプログラムのブレーク付き実行(アドレス指定)
GoBreakAt*	GBA	ターゲットプログラムのブレーク付き実行(行番号指定)
Stop	-	ターゲットプログラムの停止
Status	-	ターゲットプログラムの実行状態表示
Step	S	ソース行単位のステップ実行
StepInstruction	SI	機械語単位のステップ実行
OverStep	O	ソース行単位のオーバーステップ実行
OverStepInstruaction	OI	機械語単位のオーバーステップ実行
Return	RET	ソース行単位のリターン実行
ReturnInstruction	RETI	機械語単位のリターン実行
Reset	-	ターゲットプログラムのリセット

7.1.2 ダウンロード関連

コマンド名	短縮名	内容
Load	L	ターゲットプログラムの一括ダウンロード
LoadHex	LH	機械語情報(インテル HEX フォーマットファイル)のダウンロード
LoadMot*	LM	機械語情報(モトローラ S フォーマットファイル)のダウンロード
LoadSymbol	LS	ソース行/アセンブラシンボル情報のダウンロード
LoadIeee*	LI	C 言語変数/関数情報のダウンロード
Reload	-	ターゲットプログラムの再ダウンロード
UploadHex	UH	機械語情報のインテル HEX フォーマットファイルへのアップロード
UploadMot*	UM	機械語情報のモトローラ S フォーマットファイルへのアップロード

7.1.3 レジスタ操作関連

コマンド名	短縮名	内容
Register	R	指定レジスタの値を参照

7.1.4 メモリ操作関連

コマンド名	短縮名	内容
DumpByte	DB	メモリ内容の１バイト単位表示
DumpWord*	DW	メモリ内容の２バイト単位表示
DumpLword*	DL	メモリ内容の４バイト単位表示
SetMemoryByte	MB	メモリ内容の１バイト単位変更
SetMemoryWord*	MW	メモリ内容の２バイト単位変更
SetMemoryLword*	ML	メモリ内容の４バイト単位変更
FillByte	FB	メモリ内容の１バイト単位充填
FillWord*	FW	メモリ内容の２バイト単位充填
FillLword*	FL	メモリ内容の４バイト単位充填
Move	-	メモリ内容の１バイト単位転送
MoveWord*	MOVEW	メモリ内容の２バイト単位転送
MoveLword*	MOVEL	メモリ内容の４バイト単位転送

7.1.5 アセンブル/逆アセンブル関連

コマンド名	短縮名	内容
Assemble	A	指定したアドレスから１行単位でアセンブル
DisAssemble	DA	指定した範囲の逆アセンブル結果を表示
Module	MOD	全モジュール(オブジェクト名)を表示
Scope	-	現在のスコープ表示/スコープの変更
Section	SEC	セクション情報を表示
Bit*	-	ビットシンボルの参照/設定
Symbol	SYM	シンボルの表示
Label	-	ラベルの表示
Express	EXP	指定したアセンブラ式の値を表示

7.1.6 ソフトウェアブレーク設定関連

コマンド名	短縮名	内容
SoftwareBreak	SB	ソフトウェアブレークポイントの表示/設定
SoftwareBreakClear	SBC	ソフトウェアブレークポイントの削除
SoftwareBreakClearAll	SBCA	全ソフトウェアブレークポイントの削除
SoftwareBreakDisable	SBD	ソフトウェアブレークポイントの無効化
SoftwareBreakDisableAll	SBDA	全ソフトウェアブレークポイントの無効化
SoftwareBreakEnable	SBE	ソフトウェアブレークポイントの有効化
SoftwareBreakEnableAll	SBEA	全ソフトウェアブレークポイントの有効化
BreakAt	-	行番号でのソフトウェアブレークポイント指定
BreakIn	-	関数の先頭にソフトウェアブレークポイントを指定

7.1.7 ハードウェアブレーク設定関連

コマンド名	短縮名	内容
HardwareBreak	HB	ハードウェアブレークポイントの指定
HardwareBreakClear	HBC	ハードウェアブレークポイントの削除
HardwareBreakClearAll	HBCA	全ハードウェアブレークポイントの削除
BreakMode	BM	ブレークモードの参照/設定

7.1.8 リアルタイムトレース関連

コマンド名	短縮名	内容
TracePoint*	TP	トレースポイントの指定
TraceData*	TD	リアルタイムトレース結果のバス信号表示
TraceList*	TL	リアルタイムトレース結果の逆アセンブル表示

7.1.9 カバレッジ計測関連

コマンド名	短縮名	内容
Coverage	CV	カバレッジ計測結果の表示

7.1.10 スタック使用量計測関連

コマンド名	短縮名	内容
StackMonitor	SM	スタック使用量計測モードの設定/参照

7.1.11 サイクル計測関連

コマンド名	短縮名	内容
Cycle	CY	実行サイクル数と実行時間の計測方式の参照/設定

7.1.12 スクリプト/ログファイル関連

コマンド名	短縮名	内容
Script	-	スクリプトファイルのオープン
Exit	-	スクリプトファイルのクローズ
Wait	-	コマンド入力待機
Pause	-	指定メッセージを表示し、ボタン入力待ち
Sleep	-	指定秒数のコマンド入力待機
Logon	-	ログファイルのオープン
Logoff	-	ログファイルのクローズ
Exec	-	外部アプリケーションの起動

7.1.13 プログラム表示関連

コマンド名	短縮名	内容
Func	-	関数名の参照/関数内容の表示
Up*	-	呼び出し元関数の表示
Down*	-	呼び出し先関数の表示
Where*	-	関数の呼び出し状況の表示
Path	-	ソースファイルのパス指定
AddPath	-	ソースファイルのパス指定の追加
File	-	指定ソースファイルの表示

7.1.14 マップ関連

コマンド名	短縮名	内容
Map*	-	マップの参照/設定

7.1.15 C 言語関連

コマンド名	短縮名	内容
Print	-	C 言語変数式の参照
Set	-	C 言語変数式へのデータ指定

7.1.16 リアルタイム OS 関連

コマンド名	短縮名	内容
MR*	-	リアルタイム OS(MRxx)の状態表示

7.1.17 ユーティリティ関連

コマンド名	短縮名	内容
Radix	-	定数の既定値設定/参照
Alias	-	コマンドの別名定義/定義状況の参照
UnAlias	-	コマンドの別名定義削除
UnAliasAll	-	全コマンドの別名定義削除
Help	H	スクリプトコマンドのヘルプ表示
Version	VER	デバッガのバージョン表示
Date	-	現在の日時表示
Echo	-	メッセージの表示
CD	-	カレントディレクトリの設定/参照

7.2 スクリプトコマンド一覧(アルファベット順)

コマンド名	短縮名	内容
AddPath	-	ソースファイルのパス指定の追加
Alias	-	コマンドの別名定義/定義状況の参照
Assemble	A	指定したアドレスから 1 行単位でアセンブル
Bit*	-	ビットシンボルの参照/設定
BreakAt	-	行番号でのソフトウェアブレークポイント指定
BreakIn	-	関数の先頭にソフトウェアブレークポイントを指定
BreakMode	BM	ブレークモードの参照/設定
CD	-	カレントディレクトリの設定/参照
Coverage	CV	カバレッジ計測結果の表示
Cycle	CY	実行サイクル数と実行時間の計測方式の参照/設定
Date	-	現在の日時表示
DisAssemble	DA	指定した範囲の逆アセンブル結果を表示
Down*	-	呼び出し先関数の表示
DumpByte	DB	メモリ内容の 1 バイト単位表示
DumpLword*	DL	メモリ内容の 4 バイト単位表示
DumpWord*	DW	メモリ内容の 2 バイト単位表示
Echo	-	メッセージの表示
Exec	-	外部アプリケーションの起動
Exit	-	スクリプトファイルのクローズ
Express	EXP	指定したアセンブラ式の値を表示
File	-	指定ソースファイルの表示
FillByte	FB	メモリ内容の 1 バイト単位充填
FillLword*	FL	メモリ内容の 4 バイト単位充填
FillWord*	FW	メモリ内容の 2 バイト単位充填
Func	-	関数名の参照/関数内容の表示
Go	G	ターゲットプログラムの実行
GoBreakAt*	GBA	ターゲットプログラムのブレーク付き実行(行番号指定)
GoFree	GF	ターゲットプログラムのフリーラン実行
GoProgramBreak*	GPB	ターゲットプログラムのブレーク付き実行(アドレス指定)
HardwareBreak	HB	ハードウェアブレークポイントの指定
HardwareBreakClear	HBC	ハードウェアブレークポイントの削除
HardwareBreakClearAll	HBCA	全ハードウェアブレークポイントの削除
Help	H	スクリプトコマンドのヘルプ表示
Label	-	ラベルの表示
Load	L	ターゲットプログラムの一括ダウンロード
LoadHex	LH	機械語情報(インテル HEX フォーマットファイル)のダウンロード
LoadIeee*	LI	C 言語変数/関数情報のダウンロード
LoadMot*	LM	機械語情報(モトローラ S フォーマットファイル)のダウンロード
LoadSymbol	LS	ソース行/アセンブラシンボル情報のダウンロード
Logoff	-	ログファイルのクローズ
Logon	-	ログファイルのオープン
Map*	-	マップの参照/設定
Module	MOD	全モジュール(オブジェクト名)を表示
Move	-	メモリ内容の 1 バイト単位転送
MoveLword*	MOVEL	メモリ内容の 4 バイト単位転送
MoveWord*	MOVEW	メモリ内容の 2 バイト単位転送
MR*	-	リアルタイム OS 状態表示

OverStep	O	ソース行単位のオーバーステップ実行
OverStepInstruaction	OI	機械語単位のオーバーステップ実行
Path	-	ソースファイルのパス指定
Pause	-	指定メッセージを表示し、ボタン入力待ち
Print	-	C 言語変数式の参照
Radix	-	定数の既定値設定/参照
Register	R	指定レジスタの値を参照
Reload	-	ターゲットプログラムの再ダウンロード
Reset	-	ターゲットプログラムのリセット
Return	RET	ソース行単位のリターン実行
ReturnInstruction	RETI	機械語単位のリターン実行
Scope	-	現在のスコープ表示/スコープの変更
Script	-	スクリプトファイルのオープン
Section	SEC	セクション情報を表示
Set	-	C 言語変数式へのデータ指定
SetMemoryByte	MB	メモリ内容の 1 バイト単位変更
SetMemoryLword*	ML	メモリ内容の 4 バイト単位変更
SetMemoryWord*	MW	メモリ内容の 2 バイト単位変更
Sleep	-	指定秒数のコマンド入力待機
SoftwareBreak	SB	ソフトウェアブレークポイントの表示/設定
SoftwareBreakClear	SBC	ソフトウェアブレークポイントの削除
SoftwareBreakClearAll	SBCA	全ソフトウェアブレークポイントの削除
SoftwareBreakDisable	SBD	ソフトウェアブレークポイントの無効化
SoftwareBreakDisableAll	SBDA	全ソフトウェアブレークポイントの無効化
SoftwareBreakEnable	SBE	ソフトウェアブレークポイントの有効化
SoftwareBreakEnableAll	SBEA	全ソフトウェアブレークポイントの有効化
StackMonitor	SM	スタック使用量計測モードの設定/参照
Status	-	ターゲットプログラムの実行状態表示
Step	S	ソース行単位のステップ実行
StepInstruction	SI	機械語単位のステップ実行
Stop	-	ターゲットプログラムの停止
Symbol	SYM	シンボルの表示
TraceData*	TD	リアルタイムトレース結果のバス信号表示
TraceList*	TL	リアルタイムトレース結果の逆アセンブル表示
TracePoint*	TP	トレースポイントの指定
UnAlias	-	コマンドの別名定義削除
UnAliasAll	-	全コマンドの別名定義削除
Up*	-	呼び出し元関数の表示
UploadHex	UH	機械語情報のインテル HEX フォーマットファイルへのアップロード
UploadMot*	UM	機械語情報のモトローラ S フォーマットファイルへのアップロード
Version	VER	デバッガのバージョン表示
Wait	-	コマンド入力待機
Where*	-	関数の呼び出し状況の表示

8. スクリプトファイルの記述

スクリプトファイルは、スクリプトコマンドを自動実行するために、その制御文などを記述したファイルです。
スクリプトファイルは、スクリプトウィンドウで実行します。

8.1 スクリプトファイルの構成要素

スクリプトファイルには、以下の文が記述できます。
- スクリプトコマンド
- 代入文
- 判断文(if,else,endi)
  式の結果を判断して、実行する文を分岐します。
- 繰り返し文(while,endw)
  式の結果を判断して、文を繰り返し実行します。
- break 文
  最も内側の繰り返し実行から抜けます。
- コメント文
  スクリプトファイルにコメント(注釈)を記述できます。スクリプトコマンド実行の際、コメント文は無視されます。

スクリプトファイルには、一行につき 1 つの文を記述してください。一行に複数の文を記述したり、1 つの文を複数行にまたがって記述することはできません。

注意事項
- スクリプトコマンドのコメントとして同一行に記述することはできません。
- スクリプトファイルのネストは 10 段までです。
- if 文と while 文のネストはそれぞれ 32 段までです。
- 一つのスクリプトファイルで if と endi 文、while と endw が対になっていなければいけません。
- スクリプトファイルに記述する式は、unsigned 型で計算します。したがって、if 文、while 文の式で負の値を比較した場合の動作は不定になります。
- 1 行に記述できる文字数は、4096 文字までです。これを越える行を実行した場合、エラーになります。
- 不適当な記述のあるスクリプトファイルを自動実行した場合、スクリプト行自身が読み込めない場合を除いて、エラー検出後もスクリプトファイルの終わりまで実行処理は続けられます。 ただしこの場合、エラー検出後の動作は不定であり、したがってエラー検出後の実行結果は信頼性がありません。

8.1.1 スクリプトコマンド

スクリプトウィンドウで入力するコマンドを、そのまま記述することができます。
またスクリプトファイルからスクリプトファイルを呼び出すこともできます(ネストは 10 段まで)。

8.1.2 代入文

代入文は、マクロ変数の定義や初期化、および代入を行います。以下に記述書式を示します。

```
%マクロ変数名 = 式
```

- マクロ変数名には、英数字と'_'が使用できます。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- マクロ変数に代入する式が扱える値の範囲は、0h から FFFFFFFFh までの整数です。
  負の数を指定した場合は 2 の補数として扱います。
- マクロ変数は、式の中で使用することができます。
- マクロ変数は、先頭に'%'を付加して使用します。

8.1.3 判断文

判断文は、式の結果を判断し、実行する文を分岐します。以下に記述書式を示します。

```
if ( 式 )
    文1
 else
    文2
 endi
```

- 式が真（0 以外）のとき文 1 を実行します。式が偽（0）のとき文 2 を実行します。
- else 文は省略することができます。else 文を省略時に式が偽の場合、endi 文の次の行から実行します。
- if 文は、32 段までネストすることができます。

8.1.4 繰り返し文(while,endw)と break 文

繰り返し文は、式の結果を判断し、文を繰り返し実行します。以下に記述書式を示します。

```
while ( 式 )
   文
 endw
```

- 式が真の場合、文を繰り返し実行します。式が偽の場合、ループから抜けます（endw の次の文から実行します）。
- while 文は、32 段までネストすることができます。
- while 文を強制的の抜ける場合は、break 文を使用します。while 文がネストしている場合は、最も内側のループから抜けます。

8.1.5 コメント文

コメント文は、スクリプトファイルにコメント（注釈）を記述する場合に使用します。以下に記述書式を示します。

```
; 文字列
```

- セミコロン（';'）から文を記述します。セミコロンの前には、空白文字とタブのみ記述可能です。
- コメント文の行は、スクリプトファイル実行時に無視されます。

8.2 式の記述

アドレス、データ、通過回数などの指定に式を記述することができます。
以下に式を使用したコマンド例を示します。

```
>DumpByte TABLE1
>DumpByte TABLE1+20
```

式の構成要素としては、以下のものが使用できます。

- 定数
- シンボル、ラベル
- マクロ変数
- レジスタ変数
- メモリ変数
- 行番号
- 文字定数
- 演算子

8.2.1 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、数値の先頭または、末尾に基数を示す記号を付けて区別します。
M32C 用デバッガ、M16C/R8C 用デバッガ、740 用デバッガの場合

	16 進数	10 進数	8 進数	2 進数 *2
先頭	0x,0X	@	なし	%
末尾	h,H	なし	o,O	b,B
例	0xAB24 AB24h	@1234	1234o	%10010 10010b

*2　基数の既定値が 16 進数のときは、'%'のみ指定可能

- 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（2 進数は除く)。
- 基数の既定値は、RADIX コマンドで設定します。ただし、以下のデータに関する入力を行う場合は、RADIX コマンドの設定に関係なく、基数は固定です。

種別	基数
アドレス	16 進
行番号 実行回数 通過回数	10 進

8.2.2 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('_')、ピリオド('.')、クエスチョンマーク('?')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。
- シンボル／ラベル名は、255 文字まで記述できます。
- 大文字／小文字は区別します。

製品名	注意事項
M32R 用デバッガ, R32C 用デバッガ, M32C 用デバッガ, M16C/R8C 用デバッガ	• アセンブラの構造化命令、擬似命令、マクロ命令、オペコード、予約語は使用できません。 （.SECTION, .BYTE, switch, if など） • ".."で始まる文字列は、シンボル／ラベル名には使用できません。
740 用デバッガ	• レジスタ名(A,X,Y,S,PC,PS,P)は使用できません。 • アセンブラの構造化命令、擬似命令、マクロ命令、オペコード、予約語は使用できません。 （.SECTION, .BYTE, switch, if など） • 以下に示す文字列は、シンボル／ラベル名には使用できません。 .D0～.D65535、.F0～.F65535、.I0～.I56635、.S0～.S65535、..0～..65535、??0～??65535

8.2.2.1 ローカルラベルシンボルとスコープ

プログラムの全領域から参照可能なグローバルラベルシンボルと、宣言したファイル内でのみ 参照可能なローカルラベルシンボルの 2 種類をサポートしています。
ローカルラベルシンボルの有効範囲をスコープといいます。 スコープの単位は、オブジェクト(リロケータブル)ファイルです。
下記の場合に応じて、スコープを切り替えます。

- コマンド入力時
  プログラムカウンタが示すアドレスを含むオブジェクトファイルが、現在のスコープとなります。また SCOPE コマンドでスコープを設定した場合、設定したスコープが有効になります。
- コマンド実行中
  コマンドが扱うプログラムアドレスによって現在のスコープを自動的に切り替えます。

8.2.2.2 ラベル/シンボルの優先順位

値からラベル／シンボルへの変換、ラベル／シンボルから値への変換は、下記の優先順位で行います。

- アドレス値を変換する場合

1. ローカルラベル
2. グローバルラベル
3. ローカルシンボル
4. グローバルシンボル
5. スコープ範囲外のローカルラベル
6. スコープ範囲外のローカルシンボル

- データ値を変換する場合

1. ローカルシンボル
2. グローバルシンボル
3. ローカルラベル
4. グローバルラベル
5. スコープ範囲外のローカルシンボル
6. スコープ範囲外のローカルラベル

- ビット値を変換する場合

1. ローカルビットシンボル
2. グローバルビットシンボル
3. スコープ範囲外のローカルビットシンボル

8.2.3 マクロ変数

マクロ変数は、スクリプトファイル中の代入文で定義します。マクロ変数は、変数名の先頭に' % 'を付加して使用します。
詳細については、「8.1.2 代入文」を参照してください。

- パーセント文字（' % '）の後の変数名には、英数字と'_'が使用可能です。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- 変数名には、レジスタ名は使用できません。
- 変数名の大文字／小文字を区別します。
- マクロ変数は、255 個まで定義できます。一度定義したマクロ変数は、デバッガを終了するまで有効です。

マクロ変数は、while 文の繰り返し回数を指定する際に利用すると便利です。

8.2.4 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(740 用デバッガの場合、'_')。 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
R32C 用デバッガ	PC, USP, ISP, INTB, FLB, SVF, SVP, VCT, DMD0, DMD1, DMD2, DMD3, DCT0, DCT1, DCT2, DCT3, DCR0, DCR1, DCR2, DCR3, DSA0, DSA1, DSA2, DSA3, DSR0,DSR1, DSR2, DSR3, DDA0, DDA1, DDA2, DDA3, DDR0,DDR1, DDR2, DDR3, 0R0, 0R1, 0R2, 0R3, 0R4, 0R5, 0R6, 0R7, 0A0, 0A1, 0A2, 0A3, 0FB, 0SB←レジスタバンク 0 1R0, 1R1, 1R2, 1R3, 1R4, 1R5, 1R6, 1R7, 1A0, 1A1, 1A2, 1A3, 1FB, 1SB←レジスタバンク 1
M32C 用デバッガ	PC, USP, ISP, INTB, FLB, SVF, SVP, VCT, DMD0,DMD1, DCT0, DCT1, DRC0, DRC1, DMA0,DMA1, DCA0, DCA1, DRA0, DRA1, 0R0, 0R1, 0R2, 0R3, 0A0, 0A1, 0FB, 0SB←レジスタバンク 0 1R0, 1R1, 1R2, 1R3, 1A0, 1A1, 1FB, 1SB←レジスタバンク 1
M16C/R8C 用デバッガ	PC, USP, ISP, SB, INTB, FLG 0R0, 0R1, 0R2, 0R3, 0A0, 0A1, 0FB←レジスタバンク 0 1R0, 1R1, 1R2, 1R3, 1A0, 1A1, 1FB←レジスタバンク 1
740 用デバッガ	PC, A, X, Y, S, PS

レジスタ名の大文字／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。

8.2.5 メモリ変数

メモリの値を式中で利用する際に使用します。メモリ変数の書式を以下に示します。
[アドレス] .データサイズ

- アドレスには、式が記述できます（メモリ変数も指定可能）。
- データサイズは､以下のように指定します（740用デバッガでは4バイト長はサポートしていません）。

データ長	対応デバッガ	指定
1 バイト	すべて	B または b
2 バイト	M32R 用デバッガ	H または h
	その他	W または w
4 バイト	M32R 用デバッガ	W または w
	R32C 用デバッガ、M32C 用デバッガ、M16C/R8C 用デバッガ	L または l

例：8000h 番地のメモリ内容を 2 バイト長で参照する場合

```
[0x8000].w
```

- データサイズの指定を省略した場合、ワード長を指定したことになります。

8.2.6 行番号

ソースファイルの行番号です。行番号の書式を以下に示します。

#行番号
#行番号."ソースファイル名"

- 行番号は、10 進数で指定します。
- 行番号に指定できるのは、ソフトウェアブレークが設定できる行だけです。 コメント行や空白行などのアセンブラの命令が生成されない行を指定することはできません。
- ソースファイル名を省略した場合､現在フォーカスがあるエディタ(ソース)ウィンドウに表示しているソースファイルの行番号になります。
- ソースファイル名は、ファイル属性も指定してください。
- 行番号とソースファイル名の間に空白文字を挿入することはできません。

8.2.7 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。

- 文字は、シングルクォーテーションで囲みます。
- 文字列は、ダブルクォーテーションで囲みます。
- 文字列は 2 文字以内（16 ビット長）でなければなりません。 2 文字を越えた場合も、記述した文字列の最後の 2 文字が処理の対象となります。例えば、"ABCD" と記入した場合、 文字列の最後の 2 文字 "CD" が処理対象となり、値は 4344h となります。

8.2.8 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 8 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、単項論理否定	レベル 2
*, /	二項乗算、二項除算	レベル 3
+, -	二項加算、二項減算	レベル 4
>>,	右シフト、左シフト	レベル 5
&	二項論理積	レベル 6
|, ^	二項論理和、二項排他的論理和	レベル 7
<, <=, >, >=, ==, !=	二項比較	レベル 8

9. I/O スクリプト

仮想ポート入力や仮想割り込みの設定を、ファイルにスクリプト形式で記述できます。このスクリプトのことを I/O スクリプトといいます。また、I/O スクリプトを記述したファイルのことを I/O スクリプトファイルといいます。 I/O スクリプトを利用することにより、I/O タイミング設定ウィンドウでの仮想ポート入力や仮想割り込みの設定よりさらに柔軟な設定を行うことができます。例えば、I/O タイミング設定ウィンドウでは設定できない次のような設定が行えます。

- タイマ割り込みのように周期的な仮想割り込みを発生させる場合、仮想割り込みの発生の繰り返しを while 文で指定できます。
- 仮想割り込みを発生させる際の割り込み優先順位を、割り込み制御レジスタの割り込み優先レベル選択ビットに設定された優先順位を参照するように指定できます。
- 仮想ポート入力や仮想割り込みの入力/発生条件として、プログラムのフェッチ、メモリへのリード/ライトアクセス、メモリの比較等を組み合わせて利用できます。

その他にも様々な I/O の設定が可能です。

9.1 I/O スクリプトの記述方法

I/O スクリプトに記述する仮想ポート入力や仮想割り込み等の I/O の定義方法について定義例を用いて説明します。 I/O スクリプトを定義するには、プロシジャーを記述します。プロシジャーは{ }で囲んで記述します。プロシジャーは、1 つのファイルに複数記述できます。 各プロシジャーには、仮想ポート入力や仮想割り込みの設定、タイミング等を記述します。 定義された複数の各プロシジャーは、プログラム実行中に並列に処理されます。但し、各プロシジャーが評価される順序は不定です。 I/O スクリプトファイルは、I/O タイミング設定ウィンドウのポップアップメニュー [Load] でデバッガに登録します。 I/O スクリプトファイルは複数登録できます。
また、 登録できるプロシジャーの数は、仮想ポート入力、仮想割り込みと合わせて 50 個までです。ただし、出力ポートウィンドウで Printf 関数の出力機能を使用している場合、設定できる仮想ポート入力、仮想割り込み、I/O スクリプトのプロシジャー数は、合わせて 48 個までになります。

- 下記例のプロシジャー1 は、タイマ A0 のタイマモードの定義例です。
  タイマ A0 に指定された分周比（サイクル数）毎に、タイマ A0 割り込みを発生させる例です。タイマ割り込みの優先順位は、割り込み制御レジスタに指定された値を参照します。
- 下記例のプロシジャー2 は、サイクルに同期した仮想ポート入力の定義例です。
  プログラムが 10000 サイクル実行された時に仮想ポート入力を行う例です。I/O タイミング設定ウィンドウでは、バイト単位の仮想ポート入力しかできませんが、I/O スクリプトではワード、ロングワード単位での仮想ポート入力が可能です。

```
; I/O スクリプトファイルの定義例                → コメント文
;プロシジャー1 の定義（仮想割り込みの例）
{                                              → プロシジャー1 の開始
    while(1){                                  → while 文
        if( ([0x380].b & 0x01) == 0x01){       → タイマ A0 のカウント開始フラグのチェック
            waitc [0x386].w + 1                → タイマ A0 に設定された分周比のサイクル数分
                                                  I/O スクリプトの実行をウェイトする
            int 21, [0x55] & 0x7               → タイマ A0 の仮想割り込みを発生
                                                  （優先順位は、割り込み制御レジスタを参照）
        }else{
            waiti 100                          → 100 命令分 I/O スクリプトの実行をウェイト
        }
    }
}                                              → プロシジャー1 の終了
;プロシジャー2 の定義（仮想ポート入力の例）
{                                              → プロシジャー2 の開始
    waitc 10000                                → 10000 サイクル分 I/O スクリプトの実行をウェイト
    set [0x3e0] = 0x20                         → 0x3e0 番地に 0x20 を入力
    waitc 10000
    set [0x3e0].w = 0x4143                     → 0x3e0 番地から 2 バイトデータ 0x4143 を入力
}                                              → プロシジャー2 の終了
```

9.2 I/O スクリプトの構成要素

I/O スクリプトは以下の文が記述できます。

- プロシジャー
- I/O スクリプト文
- 判断文（if,else）
  式の結果を判断して、実行する文を分岐します。
- 繰り返し文（while)と Break 文
  式の結果を判断して、文を繰り返し実行します。
- コメント文
  I/O スクリプトにコメント（注釈）を記述できます。I/O スクリプト実行の際、コメント文は無視されます。
  I/O スクリプトには、一行につき 1 文を記述してください。一行に複数の文を記述したり、1 つの文を複数行にまたがって記述することはできません。

9.2.1 プロシジャー

プロシジャーは、I/O スクリプトの定義ブロックを指定します。プロシジャーは 1 つのファイルに複数指定できます。 なお、プロシジャーの定義は、仮想ポート入力、仮想割り込みと合わせて 50 個までです。 ただし、出力ポートウィンドウで Printf 関数の出力機能を使用している場合、プロシジャーの定義は、仮想ポート入力、仮想割り込みと合わせて 48 個までになります。 以下に記述書式を示します。

```
{
  文
}
{
  文
}

以下、複数定義可能
 :
```

9.2.2 I/O スクリプト文

I/O スクリプト文には、以下の５つの文が指定できます。

9.2.2.1 waiti 文

「書式」waiti 機械語命令数
「機能」
指定された機械語命令数分、次の文の実行を抑止します。
機械語命令数には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。
例えば、次のような記述の場合、

```
waiti 100
set [0x800] = 0x10
```

set 文の実行は、機械語命令が 100 命令実行された後に行われます。

9.2.2.2 waitc 文

「書式」 waitc サイクル数
「機能」
指定されたサイクル数分、次の文の実行を抑止します。
サイクル数には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。
例えば、次のような記述の場合、

```
waitc 10000
set [0x800] = 0x10
```

set 文の実行は、プログラムの実行が 10000 サイクル実行された後に行われます。

9.2.2.3 int 文

「書式」 int ベクタ番号, 優先順位
「機能」
指定されたベクタ番号の仮想割り込みを、指定された優先順位で発生させます。
ベクタ番号、優先順位には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。
例えば、次のような記述の場合、

```
int 21 , 5
```

タイマ A0（ベクタ番号 21）割り込みを優先順位 5 で発生させます。

9.2.2.4 set 文

set 文には、3 通りの書式があります。
「書式１」 set メモリアドレス = 入力値
「機能」
指定したメモリアドレスに入力値を入力します（仮想ポート入力を行います）。
メモリアドレスには左辺式が、入力値には右辺式が指定できます（左辺式、右辺式の仕様は、後述を参照下さい）。
例えば、次のような記述の場合、

```
set [0x3e0] = 0x1d
```

0x3e0 番地のメモリに 0x1d を入力します。

「書式２」 set 条件式 , メモリアドレス = 入力値１, 入力値２, ...
「機能」
条件式が成立するごとに、指定したメモリアドレスに入力値１、入力値２を順次入力します。
メモリアドレスには左辺式が、条件式, 入力値には右辺式が指定できます（左辺式、右辺式の仕様は、後述を参照下さい）。
例えば、次のような記述の場合、

```
set #isfetch:0xf0000 , [0x3e1] = 0x10 , 0x20
```

;#isfetch はプログラムが指定番地を実行した時に真（成立）になります。

プログラムが 0xf0000 番地を実行するごとに、0x3e1 番地のメモリにデータ 0x10,0x20 を順次入力します。
つまり、0xf0000 を 1 回目に実行した時には、0x3e1 番地のメモリにデータ 0x10 が、2 回目に実行した時は 0x20 が入力されます。

「書式３」 set %マクロ変数 = 右辺式
「機能」
指定したマクロ変数に右辺式を代入します（マクロ変数の仕様は、後述を参照下さい）。
例えば、次のようなマクロ変数の記述が行えます。

```
set %val = 10 ; マクロ変数valを10に初期化します。
set %val = %val + 1 ; マクロ変数の値に１を足します。
```

9.2.2.5 pass 文

「書式」 pass 条件式 , パス回数
「機能」
パス回数で指定された回数分、条件式が成立するまで、次の文の実行を抑止します。
条件式、パス回数には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。
例えば、次のような記述の場合、

```
pass #isint:21 , 3
```

;#isint は指定された仮想割り込みが発生すれば真（成立）になります。

```
set [0x800] = 0x10
```

set 文の実行は、タイマ A0 割り込み（ベクタ番号 21）が 3 回発生した後に行われます。

9.2.3 判断文（if, else）

判断文は、式の結果を判断し、実行する文を分岐します。 以下に記述書式を示します。

```
if (条件式 ) {
   文１
 } else if (条件式) {
   文２
 } else {
   文3
 }
```

- if(条件式)が真（０以外）のとき文１を実行します。条件式が偽（０）のとき else if(条件式)を評価し、条件式が真であれば文２を実行します。そうでなければ else 文の文 3 を実行します。
- else if, else 文は省略することができます。
- if 文は、32 段までネストすることができます。
- 条件式には右辺式が使用できます。
- I/O スクリプトに記述する条件式は、unsigned 型で計算します。したがって、if 文、の式で負の値を比較した場合の動作は不定になります。

9.2.4 繰り返し文（while)と break 文

繰り返し文は、式の結果を判断し、文を繰り返し実行します。以下に記述書式を示します。

```
while ( 条件式 ) {
   文 または break文
 }
```

- 条件式が真の場合、文を繰り返し実行します。条件式が偽の場合、ループから抜けます。
- while 文は、32 段までネストすることができます。
- while 文を強制的の抜ける場合は、break 文を使用します。while 文がネストしている場合は、最も内側のループから抜けます。
- 条件式には右辺式が使用できます。
- I/O スクリプトに記述する条件式は、unsigned 型で計算します。したがって、while 文の式で負の値を比較した場合の動作は不定になります。

9.2.5 コメント文

コメント文は、I/O スクリプトにコメント（注釈）を記述する場合に使用します。以下に記述書式を示します。

; 文字列

- セミコロン（' ; '）から文を記述します。
- ;以降、行の最後までをコメントとして扱います。
- コメント文の行は、I/O スクリプト実行時に無視されます。

9.3 右辺式の記述方法

I/O スクリプト文の機械語命令数、サイクル数、ベクタ番号、優先順位、入力値、条件式、パス回数や if,while 文の式の指定に右辺式を記述することができます。以下に右辺式を使用した I/O スクリプト文の例を示します。

```
waitc LABEL
waiti [0x800] + 20
if( [0x1ff] == 0x30 )
while( #isfetch:0xf0000 )
```

右辺式の構成要素として、以下のものが使用できます。

- 定数
- シンボル、ラベル
- マクロ変数
- メモリ変数
- 文字定数
- 演算子
- #isfetch, #isint, #isread, #iswrite

以下に、各構成要素について説明します。

9.3.1 定数

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

	16 進数	10 進数	2 進数※
先頭	0x,0X	なし	%
例	0xAB24	1234	%10010

9.3.2 シンボル、ラベル

ターゲットプログラムで定義しているグローバルシンボル／グローバ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('_')、ピリオド('.')、クエスチョンマーク('?')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。
- シンボル／ラベル名は、255 文字まで記述できます。
- 大文字／小文字は区別します。
- アセンブラ asxx の構造化命令、擬似命令、マクロ命令、オペコード予約語は使用できません(.SECTION, .BYTE, switch, if など)。
- ".."で始まる文字列は、シンボル／ラベル名には使用できません。

9.3.3 マクロ変数

マクロ変数は、変数名の先頭に'%'を付加して使用します。

- パーセント文字（'%'）の後の変数名には、英数字と'_'が使用可能です。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- 変数名には、レジスタ名は使用できません。
- 変数名の大文字／小文字を区別します。
- マクロ変数は、32 個まで定義できます。一度定義したマクロ変数は、デバッガが終了するまで有効です。

9.3.4 メモリ変数

メモリの値を式中で利用する際に使用します。メモリ変数の書式を以下に示します。

[アドレス] .データサイズ

- アドレスには、式が記述できます（メモリ変数も指定可能）。
- データサイズは、以下のように指定します。

バイト長の場合	Bまたはb
ワード（2バイト）長の場合	またはw
ロングワード（4バイト）長の場合	Lまたはl

- データサイズの指定を省略した場合、バイト長を指定したことになります。

例１：800016番地のメモリ内容をバイトで参照する場合

[0x8000].B または [0x8000]

例２：800016番地のメモリ内容をワードで参照する場合

[0x8000].w

例３：800016番地のメモリ内容をロングワードで参照する場合

[0x8000].L

9.3.5 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。

- 文字は、シングルクォーテーションで囲みます。
- 文字列は、ダブルクォーテーションで囲みます。
- 文字列は2文字以内（16ビット長）でなければなりません。

2文字を越えた場合も、記述した文字列の最後の2文字が処理の対象となります。例えば、"ABCD" と記入した場合、文字列の最後の2文字 "CD" が処理対象となり、値は 434416 となります。

9.3.6 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１２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、単項ビット論理	レベル２
*, /	二項乗算、二項除算	レベル３
+, -	二項加算、二項減算	レベル４
>>,	右シフト、左シフト	レベル５
<, <=, >, >=	二項比較	レベル６
==, !=	二項比較	レベル７
&	二項論理積	レベル８
^	二項排他的論理和	レベル９
|	二項論理和	レベル１０
&&	積結合	レベル１１
||	和結合	レベル１２

9.3.7 #isfetch, #isint, #isread, #iswrite

#isfetch,#isint,#isread,#iswrite 式は、 I/O スクリプト文や if 文、while 文の条件式に使用します。

9.3.7.1 #isfetch 式

「書式」 #isfetch: アドレス
「機能」
指定されたアドレスにプログラムの PC 値が移った時に、式の値が真(1)になります。
それ以外は偽（0）となります。
例えば、次の if 文は、

```
if ( #isfetch:0xfc000)
```

プログラムのアドレス（PC 値）が 0xfc000 になった時に真(1)になります。

9.3.7.2 #isint 式

「書式」 #isint: ベクタ番号
「機能」
指定されたベクタ番号の仮想割り込みが発生した直後に式の値が真(1)になります。
それ以外は偽(0)となります。
例えば、次の if 文は、

```
if ( #isint:13)
```

この if 文が評価された時（評価される直前）に、ベクタ番号 13 の仮想割り込みが発生していたら真(1)になります。

9.3.7.3 #isread 式

「書式」 #isread: アドレス
「機能」
指定されたアドレスのメモリがリードアクセス（メモリの読み込み）された直後に式の値が真(1)となります。それ以外は偽(0)となります。
例えば、次の if 文は、

```
if ( #isread:0x800)
```

この if 文が評価された時（評価される直前）に、0x800 番地のメモリにリードアクセスがあった場合、真(1)となります。

9.3.7.4 #iswrite 式

「書式」 #iswrite: アドレス
「機能」
指定されたアドレスのメモリがライトアクセス（メモリへの書き込み）された直後に式の値が真(1)となります。それ以外は偽(0)となります。
例えば、次の if 文は、

```
if ( #iswirte:0x800)
```

この if 文が評価された時（評価される直前）に、0x800 番地のメモリにライトアクセスがあった場合、真(1)となります。

9.4 左辺式の記述方法

左辺式は、I/O スクリプト文の set 文のメモリアドレス、マクロ変数に記述することができます。以下に左辺式を使用した I/O スクリプト文の例を示します。

```
set [0x3e0] = 0x1a
set %val = 10
```

左辺式の構成要素として、以下のものが使用できます。

- マクロ変数
- メモリ変数

以下に、各構成要素について説明します。

9.4.1 マクロ変数

マクロ変数は、変数名の先頭に' % 'を付加して使用します。

- マクロ変数名には、英数字と'_'が使用できます。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- マクロ変数に代入する式が扱える値の範囲は、016 から FFFFFFFF16 までの整数です。負の数を指定した場合は 2 の補数として扱います。

while 文の繰り返し回数を指定する際に、マクロ変数を利用すると便利です。

set %val = 0	;マクロ変数%val に 0 を代入します。
while(%val	;%val が 10 になるまで while 文を繰り返します。
waitc 10000	
int 13,5	
set %val = %val + 1	;%val の値に 1 を足します。
}	

9.4.2 メモリ変数

メモリに値を書き込む際に使用します。メモリ変数の書式を以下に示します。

[アドレス］.データサイズ

- アドレスには、式が記述できます（メモリ変数は記述できません）。
- データサイズは、以下のように指定します。

バイト長の場合	B または b
ワード（2 バイト）長の場合	W または w
ロングワード（4 バイト）長の場合	L または l

- データサイズの指定を省略した場合、バイト長を指定したことになります。

例１：800016 番地のメモリにバイトで書き込む場合

```
set [0x8000].B = 0x10 または set [0x8000] = 0x10
```

例２：800016 番地のメモリにワードで書き込む場合

```
set [0x8000].w = 0x1234
```

例３：800016 番地のメモリにロングワードで書き込む場合

```
set [0x8000].L = 0x12345678
```

10. C/C++言語式の記述

10.1 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.", ...

10.1.1 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。

10.1.2 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`

10.1.3 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、現在サポートしておりません。

10.1.4 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」、・・・という記述で使用します。**

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。

10.1.5 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。

10.1.6 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、符号反転は行なわれません。

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。

10.1.7 "."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。

10.1.8 "->"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。また、メンバ関数内では this->member1 等の this ポインタを使用した変数参照ができます。

10.1.9 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。

10.1.10 括弧

式の途中に、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。

10.1.11 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)]」、・・・という記述で使用します。

10.1.12 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。

10.1.13 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。

10.1.14 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。

10.1.15 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。

注意

- C++の場合、関数名は使用できません。

10.1.16 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）。

10.1.17 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。

10.2 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。

10.2.1 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)

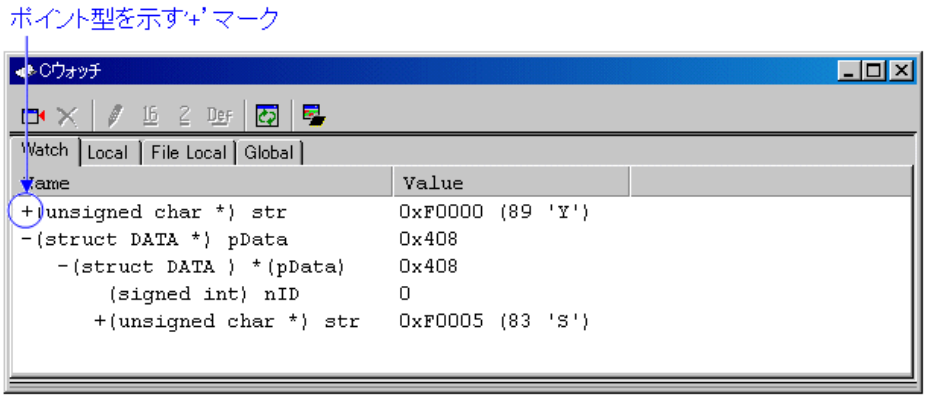
10.2.2 基本型の場合

- 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 = 0000000010.....1000B(Radixが2進数の場合)
  (...は省略を表す)

10.2.3 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(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++言語式がポインタ型の場合は、以下に示すように、型名の左側に'+'マークが現れます。

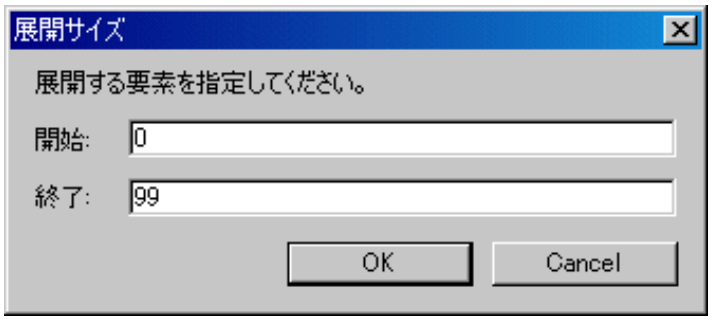


この'+'マークが表示されている行をダブルクリックすると、そのポインタのオブジェクトが現れます。 オブジェクトを表示すると、'+'マークは'-'マークにかわります。なお、'-'マークが表示されている行をダブルクリックすると、 もとの状態に戻ります。このようにして、リスト構造やツリー構造等のデータも参照することができます。

10.2.4 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10.2.3 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。

10.2.5 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)

10.2.6 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)

10.2.7 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)

10.2.8 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)

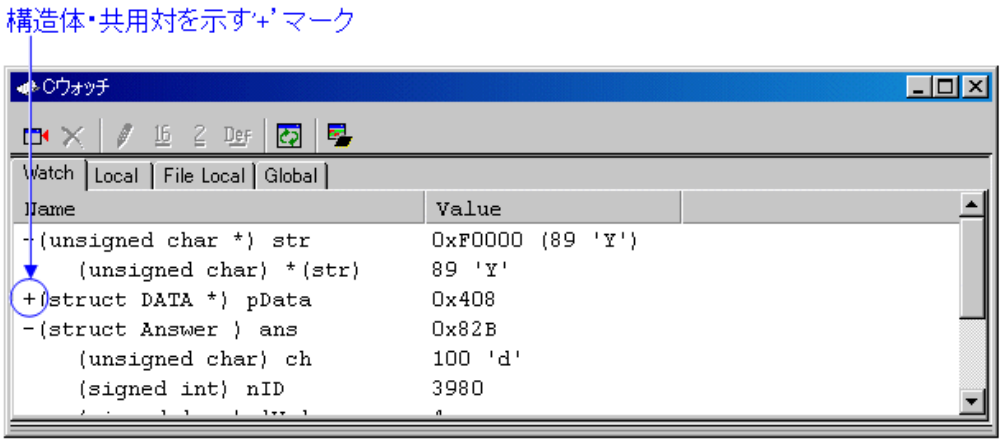
10.2.9 文法エラーの場合

- 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)

10.2.10 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。



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

11. 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、ステイタスウィンドウ（[Platform]シート）に表示されます。
740 用デバッガでは、サポートしていません。

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	ソフトウェアブレーク
H/W break	ハードウェアブレーク
Memory access error	メモリアクセスエラー
Undefined instruction	未定義命令

12. 注意事項

12.1 製品共通の注意事項

12.1.1 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。

1. ファイル名、及びディレクトリ名
   - ・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。
2. 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。

12.1.2 ソフトウェアブレークポイントが設定可能な領域

全領域にソフトウェアブレークポイントが設定可能です。

12.1.3 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。
- ビットフィールドメンバへは、以下の値を代入できます。
  - ・整数定数、文字定数、列挙子
  - ・bool 型変数、文字型変数、整数型変数、列挙型変数
  - ・ビットフィールドメンバ

  上記の代入する値が、ビットフィールドメンバのビットサイズを越える値の場合、超えた値(上位ビット)は切り捨てて代入します。
- メモリを読み出すと値が変更される SFR 領域に割り当てられたビットフィールドメンバは、値が正しく変更されません。
- プログラム実行中は、ローカル変数、および、ビットフィールドメンバの値を変更できません。

12.1.4 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。
- 引数に関数名を指定するスクリプトコマンド（breakin, func 等）は使用できません。
- 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。
- メンバ関数へのポインタは、C ウォッチウィンドウでは正しく参照できません。

12.1.5 ターゲットプログラムダウンロードの設定

ダウンロードモジュールを登録する際に設定するオプションのうち、以下については対応していません。

- オフセット：常に 0 として処理されます。
- アクセスサイズ：常に 1 として処理されます。
- ダウンロード時のメモリベリファイ：対応していません。

12.1.6 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。

12.1.7 同期デバッグ

同期デバッグには対応していません。

12.1.8 ポート出力機能

I/O タイミング設定ウィンドウの仮想ポート出力や、出力ポートウィンドウで取得できるデータの最大数を Init ダイアログで指定できます。指定するデータ数は、なるべく 50 万以下の値を設定してください。740 用デバッガの場合、データ個数は 30000 個に固定されています。
大きな値にすると、シミュレータデバッガや Windows のパフォーマンスが低下する可能性があります。

12.2 R32C 用デバッガの注意事項

12.2.1 コンパイラ/アセンブラ/リンカのオプション

デバッグするには、コンパイル・リンク時のオプション設定を考慮する必要があります。
設定内容については、以下を参照して下さい。
「12.6 コンパイラ/アセンブラ/リンカのオプション」を参照してください

R32C 用デバッガで使用可能のコンパイラ：
- 弊社製 C コンパイラ NCxx
- IAR 社製 EC++コンパイラ
- IAR 社製 C コンパイラ

12.3 M32C 用デバッガの注意事項

12.3.1 コンパイラ/アセンブラ/リンカのオプション

デバッグするには、コンパイル・リンク時のオプション設定を考慮する必要があります。
設定内容については、以下を参照して下さい。
「12.6 コンパイラ/アセンブラ/リンカのオプション」を参照してください

M32C 用デバッガで使用可能のコンパイラ：
- 弊社製 C コンパイラ NCxx
- IAR 社製 EC++コンパイラ
- IAR 社製 C コンパイラ

12.4 M16C/R8C 用デバッガの注意事項

12.4.1 コンパイラ/アセンブラ/リンカのオプション

デバッグするには、コンパイル・リンク時のオプション設定を考慮する必要があります。
設定内容については、以下を参照して下さい。
「12.6 コンパイラ/アセンブラ/リンカのオプション」を参照してください。

M16C/R8C 用デバッガで使用可能のコンパイラ：
- 弊社製 C コンパイラ NCxx
- IAR 社製 EC++コンパイラ
- IAR 社製 C コンパイラ
- TASKING 社製 C コンパイラ

12.4.2 TASKING 社製 C コンパイラ ビットフィールドメンバの参照

TASKING 社製 C コンパイラ CCM16 をご使用の場合、ビットフィールドのメンバは常に unsigned short int 型で表示されます。これは、CCM16 が出力するデバッグ情報によるものです。

12.4.3 M16C/62 グループを使用する際の注意事項について

M16C/62 グループのメモリ空間拡張モードには対応しておりません(ノーマルモードのみ対応しております)。

12.5 740 用デバッガの注意事項

12.5.1 コンパイラ/アセンブラ/リンカのオプション

デバッグするには、コンパイル・リンク時のオプション設定を考慮する必要があります。
設定内容については、「12.6 コンパイラ/アセンブラ/リンカのオプション」を参照してください。

740 用デバッガで使用可能のコンパイラ：
- 弊社製 740 ファミリ用アセンブラパッケージ SRA74
- IAR 社製 C コンパイラ

12.5.2 使用できない機能

- 740 用デバッガでは、リアルタイムトレース機能はサポートしていません。 そのため、以下のウィンドウおよびスクリプトコマンドは使用できません。
  - ・出力ポートウィンドウ
  - ・トレースポイント設定ウィンドウ
  - ・トレースウィンドウ
  - ・データトレースウィンドウ
  - ・TracePoint コマンド
  - ・TraceData コマンド
  - ・TraceList コマンド

- 740 用デバッガでは、メモリ全空間がデバッグ対象領域になります。

  - ・メモリマップを確保する Map コマンドはありません。
  - ・カバレッジ対象の領域を指定することはありません。

- 740 用デバッガのシミュレータエンジンは、プログラム停止要因を検出することができません。 そのため、アウトプットウィンドウおよびステイタスウィンドウ([Platform]シート)にはプログラム停止要因を 表示しません。

- 740 用デバッガでは、スタックトレースはサポートしていません。 そのため、以下のウィンドウおよびスクリプトコマンドは使用できません。
  - ・スタックトレースウィンドウ
  - ・Up コマンド
  - ・Down コマンド
  - ・Where コマンド

12.6 コンパイラ/アセンブラ/リンカのオプション

デバッグするには、コンパイル・リンク時のオプション設定を考慮する必要があります。
設定内容については、以下を参照して下さい

これ以外の設定では動作チェックを行っておりません。 これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.1 弊社 C コンパイラ NCxx をご使用の場合

コンパイル時に-O, -OR, -OS オプションを指定した場合、最適化のためにソース行情報が 正しく生成されず、ステップ実行等が正しく行われない場合があります。
この問題を回避するには、-O,-OR, -OS オプションと同時に-ONBSD(もしくは-Ono_Break_source_debug)オプションも指定してください。

12.6.2 740 ファミリ用アセンブラパッケージ SRA74 をご使用の場合

以下のオプションを指定してください。

アセンブル時

- "-c"オプション
  ソースラインデバッグ情報がリロケータブルファイルに出力されます。

（注意）

ソースファイル中に擬似命令.FUNC で関数名を指定した場合は、"-c"オプションを付けると 指定した関数名が無効になります。.FUNC で指定した関数名を有効にする場合は、"-c"オプションを 付けないでください。

- "-s"オプション
  ローカルなラベル、シンボル、ビットシンボル情報がリロケータブルファイルに出力されます。

リンク時

- "-s"オプション
  シンボルファイルが生成されます。

これ以外の設定では動作チェックを行っておりません。 これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.2.1 コマンド入力例

以下にコマンド入力例を示します。

- 740 用デバッガの場合

```
>sra74 -c -s main.a74<Enter>
>sra74 -c -s sub.a74<Enter>
>link74 main sub ,,,-s<Enter>
```

12.6.3 IAR 社製 EC++コンパイラをワークベンチ(EW)でご使用の場合

以下の手順でプロジェクトを設定してください。

1. IAR Embedded Workbench でのプロジェクト設定
   メニュー[Project]→[Options...]を選択すると Options For Target"xxx"ダイアログが開きます。
   このダイアログの Category で XLINK を選択し、以下のように設定してください。
   - ・Output タブ
     Format 領域で Other をチェックし、Output format に elf/dwarf を選びます。
   - ・Include タブ
     XCL file name 領域で、ご使用の XCL ファイル(例：lnkm32cf.xcl)を指定してください。

2. XCL ファイルの編集
   ご使用の XCL ファイルに -y オプションを追記してください。"-y"オプションの指定は、製品によって異なります。

製品名	-y オプション
R32C 用デバッガ	-yspc
M32C 用デバッガ	-yspc
M16C/R8C 用デバッガ	-yspc

3. プログラムのビルド
   上記設定後、ターゲットプログラムをビルドしてください。

これ以外の設定では動作チェックを行っておりません。 これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.4 IAR 社製 C コンパイラをワークベンチ(EW)でご使用の場合

以下の手順でプロジェクトを設定してください。

1. IAR Embedded Workbench でのプロジェクト設定
   メニュー[Project]→[Options...]を選択すると Options For Target"xxx"ダイアログが開きます。
   このダイアログの Category で XLINK を選択し、以下のように設定してください。
   - ・Output タブ
     Format 領域で Other をチェックし、Output format に ieee-695 を選びます。
   - ・Include タブ
     XCL file name 領域で、ご使用の XCL ファイル(例：lnkm16c.xcl)を指定してください。
2. XCL ファイルの編集
   ご使用の XCL ファイルに -y オプションを追記してください。"-y"オプションの指定は、製品によって異なります。

製品名	-y オプション
R32C 用デバッガ	-ylmba
M32C 用デバッガ	-ylmb
M16C/R8C 用デバッガ	-ylmb
740 用デバッガ	-ylmba

3. プログラムのビルド
   上記設定後、ターゲットプログラムをビルドしてください。

これ以外の設定では動作チェックを行っておりません。 これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.5 IAR 社製 C コンパイラをコマンドラインでご使用の場合

12.6.5.1 オプション指定

以下の手順でコンパイル・リンクしてください。

- コンパイル時
  "-r"オプションを指定して下さい。
- リンク前
  リンク時に読み込むリンカのオプション定義ファイル(拡張子.xcl)をオープンし、 "-FIEEE695"及び"-y"オプションを追加してください。
  "-y"オプションの指定は、製品によって異なります。

製品名	-y オプション
R32C 用デバッガ	-ylmba
M32C 用デバッガ	-ylmb
M16C/R8C 用デバッガ	-ylmb
740 用デバッガ	-ylmba

- リンク時
  "-f"オプションでリンカのオプション定義ファイル名を指定して下さい。

これ以外の設定では動作チェックを行っておりません。 これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.5.2 コマンド入力例

以下にコマンド入力例を示します。

- R32C 用デバッガの場合

```
>ICCR32C -r file1.c<Enter>
>ICCR32C -r file2.c<Enter>
>XLINK -o filename.695 -f lnkr32c.xcl file1 file2<Enter>
```

- M32C 用デバッガの場合

```
>ICCMC80 -r file1.c<Enter>
>ICCMC80 -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm80.xcl file1 file2<Enter>
```

- M16C/R8C 用デバッガの場合

```
>ICCM16C -r file1.c<Enter>
>ICCM16C -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm16c.xcl file1 file2<Enter>
```

- 740 用デバッガの場合

```
>ICC740 -r file1.c<Enter>
>ICC740 -r file2.c<Enter>
>XLINK -o filename.695 -f lnk7400t.xcl file1 file2<Enter>
```

XCL ファイル名は、製品やメモリモデルによって異なります。詳細は、ICCxxxx のマニュアルを参照して下さい。

12.6.6 TASKING 社製 C コンパイラをワークベンチ(EDE)でご使用の場合

以下の手順でプロジェクトを設定してください。

1. メニュー[EDE]→[C Compiler Option]→[Project Options...]を選択して下さい。 "M16C C Compiler Options [プロジェクト名]"ダイアログが開きます。
   このダイアログで以下のように設定してください。
   - ・Optimeze タブ
     Optimization level に"No optimization"を指定して下さい。
   - ・Debug タブ
     "Enable generation of any debug information(including type checkeing)"と "Genarate symbolic debug information"のみをチェックして下さい。

2. メニュー[EDE]→[Linker/Locator Options...]を選択して下さい。 "M16C Linker/Locator Options [プロジェクト名]"ダイアログが開きます。
   このダイアログで以下のように設定してください。
   - ・Format タブ
     Output Format に"IEEE 695 for debuggers(abs)"を指定して下さい。

3. 上記設定後、ターゲットプログラムをビルドしてください。

これ以外の設定では動作チェックを行っておりません。 これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.7 TASKING 社製 C コンパイラをコマンドラインでご使用の場合

12.6.7.1 オプション指定

コンパイル時に"-g"、"-O0"オプションを指定して下さい。

これ以外の設定では動作チェックを行っておりません。
これ以外の設定は、推奨いたしかねますのでご了承ください。

12.6.7.2 コマンド入力例

以下にコマンド入力例を示します。

```
>CM16 -g -O0 file1.c<Enter>
```

[MEMO]

M32C シミュレータデバッガ V.1.04
ユーザーズマニュアル

発行年月日　2009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

© 2009. Renesas Technology Corp. and Renesas Solutions Corp., All rights reserved. Printed in Japan.

M32C シミュレータデバッガ V.1.04
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

RJJ10J2466-0100