お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

お客様各位

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

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

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

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

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

User's Manual

SuperH RISC engine シミュレータ・デバッガ

ユーザーズマニュアル

ルネサスマイクロコンピュータ開発環境システム

HS0700SDCS4SJ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00　2000.09

ご注意

1 本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合，または国外に持ち出す場合は日本国政府の許可が必要です。
2 本書に記載された情報の使用に際して，弊社もしくは第三者の特許権，著作権，商標権，その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合，弊社はその責を負いませんので予めご了承ください。
3 製品及び製品仕様は予告無く変更する場合がありますので，最終的な設計，ご購入，ご使用に際しましては，事前に最新の製品規格または仕様書をお求めになりご確認ください。
4 弊社は品質・信頼性の向上に努めておりますが，宇宙，航空，原子力，燃焼制御，運輸，交通，各種安全装置， ライフサポート関連の医療機器等のように，特別な品質・信頼性が要求され，その故障や誤動作が直接人命を脅かしたり，人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は，事前に弊社営業担当迄ご相談をお願い致します。
5 設計に際しては，特に最大定格，動作電源電圧範囲，放熱特性，実装条件及びその他諸条件につきましては，弊社保証範囲内でご使用いただきますようお願い致します。
保証値を越えてご使用された場合の故障及び事故につきましては，弊社はその責を負いません。
また保証値内のご使用であっても半導体製品について通常予測される故障発生率，故障モードをご考慮の上，弊社製品の動作が原因でご使用機器が人身事故，火災事故，その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。
6 本製品は耐放射線設計をしておりません。
7 本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。
8 本書をはじめ弊社半導体についてのお問い合わせ，ご相談は弊社営業担当迄お願い致します。

はじめに

SuperH™ RISC engine シミュレータ・デバッガ（以下、シミュレータ・デバッガと略します）はホストコンピュータ上で、SuperH™ RISC engine マイコンのシミュレーションを行うことにより、ソフトウェア開発をサポートするソフトウェアです。

本マニュアルは、シミュレータ・デバッガの概要および取り扱い方法について記載してあります。シミュレータ・デバッガをご使用になる前に本マニュアルをよくお読み下さい。また、本シミュレータ・デバッガと関連する C/C++コンパイラ、アセンブラ、モジュール間最適化ツール、ライブラリアンについては、下記のマニュアルをお読み下さい。

- SuperH™ RISC engine　C コンパイラ ユーザーズマニュアル
- SuperH™ RISC engine　C++コンパイラ ユーザーズマニュアル
- SuperH™ RISC engine　クロスアセンブラ ユーザーズマニュアル
- H シリーズ　モジュール間最適化ツール ライブラリアン オブジェクトコンバータ ユーザーズマニュアル

SuperH™ RISC engine マイコンの詳細については、該当品種のプログラミングマニュアルおよびハードウェアマニュアルをお読み下さい。

本マニュアルでは、概要、使用前の準備および操作方法を説明します。詳細な操作については、シミュレータ・デバッガを起動し、オンラインマニュアルをお読みください。

また、ワークステーションからのコマンド入力は、C シェルが起動されている状態を仮定して説明します。

他のシェルを使用している場合は、ホストシステムマニュアル等を参照してください。

本マニュアルの説明の中で用いられる記号は、次の意味を示しています。

＜　＞	：	この記号で囲まれた内容を指定することを示します。
［　］	：	省略してもよい項目を示します。
{A｜B}	：	AまたはBを選択することを示します。
△	：	一個以上の空白を示します。
（ＳＰ）	：	スペースキーを示します。
（ＲＥＴ）	：	リターンキーを示します。
<u>　　</u>	：	入力を意味します。（アンダーライン）
%	：	C シェルのプロンプトを意味します。

目次

表目次

図目次

1. 概要

シミュレータ・デバッガは、SuperH™ RISC engine マイコンの CPU シミュレーション機能およびデバッグ機能を持っており、SH-1、SH-2、SH-3、SH-4、SH-2E、SH-3E、SH-3DSP、SH-DSP シリーズのシミュレーションをサポートします。シミュレータ・デバッガ Ver.4 では、C 言語やアセンブリ言語で作成されたプログラムに加え、C++言語で作成されたプログラムを効率よくデバッグすることができます。

本シミュレータ・デバッガは、ワークステーションで動作するインタフェースソフトとともに動作します。

なお､SH-DSP シリーズにはキャッシュをサポートしていない SH-DSP および SH-2DSP とサポートしている SH-DSPwithCache があります。本マニュアルでは、SH-DSP シリーズと表記した場合は、SH-DSP、SH-2DSP および SH-DSPwithCache を表わします。

1.1 動作環境

ホストシステムとして以下のマシン環境をサポートしています。

(1) SPARC※1 搭載機（以降、SPARC と示します。）

OS	：Solaris※2 2.4（OSF/Motif※3 環境）
ウィンドウシステム	：OSF/Motif
メモリ容量	：32M バイト以上（システム動作状態によって異なります。）
ディスク容量	：30M バイト以上（動作するための空容量を含みます。）

(2) HP9000 シリーズ 700※4（以降、HP9000 と示します。）

OS	：HP-UX10.2※5
ウィンドウシステム	：OSF/Motif
メモリ容量	：32M バイト以上（システム動作状態によって異なります。）
ディスク容量	：30M バイト以上（動作するための空容量を含みます。）

(3) ソフトウェア構成

シミュレータ・デバッガの構成を示します。

インストーラ	：cas_install
インタフェースソフト	：csdsh,dbgif
シミュレータ･デバッガ	：sdsh12,sdshdsp,sdsh2e,sdsh3e,sdsh4,sdsh3dsp,sdshdspc,sdsh2dsp
CPU 情報ファイル作成プログラム	：ciash,ciashdsp,ciash4
協調検証用ライブラリ	：libeaglei_xxx.yy、libseamless_xxx.yy、libexttool_xxx.yy、

（xxx：CPU 名－sh4 または sh2dsp、yy：拡張子－so（SPARC)または sl（HP9000））

※1 SPARC は、米国 SPARC International 社の商標です。
※2 Solaris は、米国 Sun Microsystems 社の商標です。
※3 OSF/Motif は、米国 Open Software Foundation 社の商標です。
※4 HP9000 シリーズ 700 は、米国 Hewlett-Packard 社の商品名称です。
※5 HP-UX は、米国 Hewlett-Packard 社の商品名称です。

1.2 特長

本シミュレータ・デバッガには次のような特長があります。

（1） ホストシステム上で動作するので、実機がなくてもプログラムのデバッグを開始することができ、システム全体の開発期間を短縮できます。

（2） パイプラインシミュレーションを行い、プログラムの命令実行サイクル数を計算します。これにより実機がなくても性能評価がおこなえます。

（3） 下記のような機能を持ち、プログラムのテスト、およびデバッグを効率よく進めることができます。

- SuperH™ RISC engine の各 CPU に対応
- 全命令またはサブルーチン命令のみのトレース機能
- デバッグ対象プログラムの実行中に異常が発生した場合、異常を無視して続行するか、または停止するかを制御する機能
- 関数単位のパフォーマンス測定
- 豊富なブレーク機能
- メモリマップの設定・編集
- 関数呼び出し履歴の表示

（4） 協調検証ツールとの接続をサポートしています。

1.3 シミュレーション範囲

（1） シミュレータ・デバッガは、SuperH™ RISC engineマイコンの下記機能をサポートしています。

- 全実行命令（パイプラインシミュレーション）
- 例外処理
- レジスタ
- 全アドレス空間
- MMU（SH-3/SH-3E/SH-3DSP/SH-4 シリーズのみ）
- キャッシュ（SH-3/SH-3E/SH-3DSP/SH-4 シリーズ および SH-DSP with Cache のみ）
- DMAC（SH-4 シリーズのみ）
- BSC（SH-4 シリーズのみ）
- FPU（SH-2E/SH-3E/SH-4 シリーズのみ）

（2） シミュレータ・デバッガは SuperH™ RISC engineマイコンの下記機能をサポートしていません。下記機能を使用したプログラムは、SuperH™ RISC engine用エミュレータを使用してデバッグしてください。

- 16 ビットフリーランニングタイマ（FRT）
- シリアルコミニュケーションインタフェース（SCI）
- I/O ポート
- 割り込みコントローラ（INTC）

1.4 留意事項

（1） 動作マシンの負荷が高い（動作中のプロセスが多い等）場合、「１．１　動作環境」に示したメモリ容量があっても起動できないことがあります。

（2） X Window System※6端末を使用する場合は、ホストシステムのウィンドウシステムが動作できるものをご用意ください。

（3） リソースで指定されているフォントセットによって、日本語の表示ができないことがあります、この場合は、ホームディレクトリ下の".Xdefaults"ファイルに次の設定を行い、ウィンド

ウシステムを再起動してください。ただし、ウィンドウ内の入力フィールドにおいても、日本語表示を行う場合は、設定例の先頭行だけを指定してください。また、リソースで指定するフォントのサイズが大きいとき、入力フィールドが重なることがあります。この場合は、小さいサイズのフォントを指定してください。

＜設定例＞

```
csdsh*fontList:     -*-fixed-medium-r-normal-14-*-*:
csdsh*filld*fontList:     -*-fixed-medium-r-normal-14-*-*
csdsh*frame*text*fontList:     -*-fixed-medium-r-normal-14-*-*
```

（4） サブウィンドウ表示（Dumpウィンドウ等）の中断はCloseまたはCancelボタンだけではできません。ベースウィンドウのSTOPボタンを押してからCloseまたはCancelボタンを押して、サブウィンドウを閉じてください。

（5） デバッガコマンド等の処理中はシミュレータ・デバッガを終了できません。ベースウィンドウのSTOPボタンを押し、処理を中断してからシミュレータ・デバッガを終了してください。

（6） シミュレータ・デバッガはユーザインタフェースcsdshおよびdbgifの２つのプロセスで動作しています。ウィンドウメニューの"終了"はcsdshプロセスの強制終了となるため、dbgifプロセスが残ることがあります。dbgifプロセスが残っている場合、シミュレータ・デバッガが再起動できないことがありますので、次のようにして、dbgifプロセスを強制終了してください。

例） シミュレータ・デバッガが動作していないときに、"dbgif"プロセスが存在しているか確認します。

```
%ps -e |grep dbgif(RET)
  689 pts/4    1:32 dbgif
```

存在していた場合は kill コマンドを使用して dbgif プロセスを強制終了させてください。

```
%kill -9 <PID>(RET)
```

<PID>は前記 ps コマンドで左端に表示されるプロセス番号（例では 689）です。

（7） SPARCでシミュレータ・デバッガ起動時、下記エラーメッセージが出力される場合は、ダイナミックリンクライブラリ（libXt等）が格納されているディレクトリ名を環境変数LD_LIBRARY_PATHに設定してください。

＜エラーメッセージ＞

```
ld.so.x:csdsh:fatal:libXt.so.x:can't open file:error=2
```

- LD_LIBRARY_PATH が環境変数に設定されていない場合。
  %setenv LD_LIBRARY_PATH ライブラリ格納ディレクトリ名（RET）
- LD_LIBRARY_PATH が既に、環境変数に設定されている場合。
  %setenv LD_LIBRARY_PATH
  登録ディレクトリ名：ライブラリ格納ディレクトリ名（RET）

※６ X Window System **は、米国マサチューセッツ工科大学の製品です。**

2. シミュレータ・デバッガの機能

本章では、SuperH™ RISC engine シミュレータ・デバッガの機能について説明します。なお、エンディアンの指定、キャッシュ、制御レジスタは、SH-3/SH-3E,SH-DSPwithCache,SH-3DSP,SH-4 シリーズのみ、MMU は SH-3/SH-3E,SH-3DSP,SH-4 シリーズのみ、BSC、DMAC は SH-4 シリーズのみサポートしています。

なお、SH-4 にはシミュレーション機能を一部制限することでシミュレーション速度を向上させた版と高機能版があり、それぞれ SH-4 および SH-4BSC と記述し区別します。SH-4 シリーズと表記した場合は、SH-4 および SH-4BSC を表わします。

表2.1　**シミュレータ・デバッガの機能と** CPU **の対応**

	エンディアン指定	MMU	**キャッシュ**	**制御レジスタ**	BSC	DMAC
SH-1	―	―	―	―	―	―
SH-2／SH-2E	―	―	―	―	―	―
SH-3／SH-3E	○	○	○	○	―	―
SH-3DSP	○	○	○	○	―	―
SH-4	○	○	○	○	△	―
SH-4BSC	○	○	○	○	○	○
SH-DSP	―	―	―	―	―	―
SH-DSP with Cache	○	―	○	○	―	―
SH-2DSP	―	―	―	―	―	―

2.1 シミュレータ・デバッガのメモリ管理

(1) メモリマップ

メモリマップの設定は、シミュレーション時のメモリアクセスサイクル数の計算に使用します。シミュレータ・デバッガでは、表 2.2に示すメモリ種別をサポートしています。

表 2.2　**メモリ種別**

メモリ種別	**デバッグ対象プログラムの実行**
内蔵 ROM **空間**（X-ROM,Y-ROM）	**可能**
内蔵 RAM **空間**（X-RAM ,Y-RAM）	**可能**
外部バス空間	**可能**
内蔵 I/O **空間**	**不可能**

(2) メモリマップの設定

メモリマップは、CPU 情報ファイルで指定します。CPU 情報ファイルの作成方法は、**「**7. CPU **情報ファイルの作成」**を参照ください。

(3) メモリリソースの確保

デバッグ対象プログラムをロードすると自動的にメモリリソースが確保されます。ただし、デバッ

グ対象プログラムに登録されていない領域は、確保されませんので、MAP_SET コマンドでメモリリソースの確保を行ってください。

(4) SH-4 のメモリマップ設定

エリア 0 のバス幅と MPX メモリ／NORMAL メモリの区別は CPU 情報ファイルで指定してください。その他の設定は BSC レジスタで設定してください。

2.2 エンディアン

SH-3/SH-3E,SH-DSPwithCache,SH-3DSP,SH-4 シリーズでは、メモリ上のデータ格納形式としてビッグエンディアンの他にリトルエンディアンのバイト順をサポートしています。これにより、リトルエンディアンで作成されたデバッグ対象プログラムのシミュレーション、デバッグが可能になります。エンディアンは、シミュレータを起動する時のオプションで指定してください。オプション指定方法の詳細は**「3.2 起動」**を参照してください。

2.3 パイプラインリセット処理

シミュレータ・デバッガでは、パイプラインのシミュレーションを行っていますが、以下の場合にパイプラインをリセットします。

① 命令シミュレーション停止後再実行までにPCが変更された
② 実行開始アドレスを指定したGoコマンドが実行された
③ イニシャライズまたはプログラムのロードが行われた
④ 現在フェッチおよびデコードされているメモリの内容が書き換えられた

パイプラインがリセットされると、すでにフェッチおよびデコードされている内容を捨てて、現在の PC からフェッチおよびデコードをやり直します。また、命令実行数および命令実行サイクル数をゼロクリアします。

2.4 MMU（メモリマネージメントユニット）

SH-3/SH-3E,SH-3DSP,SH-4 シリーズでは、TLB、アドレス変換機構、MMU 関連例外（TLB ミス、TLB 保護例外、TLB 無効例外、初期ページ書き込み）等 MMU の動作をシミュレーションします。これにより、MMU でのアドレス変換機構を使用しているデバッグ対象プログラムのシミュレーション、デバッグが可能となります。また本シミュレータ・デバッガは、MMU 関連例外のハンドラルーチン作成時にも利用することができます。

MMU のアドレス変換機構はシミュレーション時だけでなくウィンドウ上のアドレスにも働きます。ユーザはウィンドウ上でのメモリ操作をデバッグ対象と同じ論理アドレスで行うことができます。MMU は CPU によって異なります。

2.5 キャッシュ

SH-3/SH-3E,SH-DSPwithCache,SH-3DSP,SH-4 シリーズでは、キャッシュの動作をシミュレーションします。これにより、デバッグ対象プログラム実行時のキャッシュの動作を確認することができます。

SH-3/3E シリーズではキャッシュ容量を起動時のオプションで指定できます。（**「3.2 起動」**参照）

また、シミュレータ・デバッガでは STATUS コマンドでキャッシュヒット率を表示します。

(1) キャッシュヒット率の取得、表示

キャッシュへのアクセス回数に対するヒット回数の割合をキャッシュヒット率として取得します。キャッシュヒット率は Status コマンドで 100 分率で表示します。キャッシュヒット率は、キャッシュヒット回数とキャッシュミス回数の和を分母とし、キャッシュヒット回数を分子として算出しています。

(2) キャッシュヒット率の初期化

キャッシュヒット率は、起動時、パイプラインリセット時、CCR 制御レジスタ値変更時に初期化します。

【注】 キャッシュアドレスアレイに格納するアドレスタグの上位 3bit は本シミュレータ・デバッガでは 0 にしません。
キャッシュがマッピングされている空間へ[File Load]メニューを選択することによってメモリをロードする場合は、MMUCR の AT bit をオフにし、MMU ディスエーブルにしてください。

2.6 BSC（バスステートコントローラ）

SH-4 シリーズでは、BSC に対応したメモリマップの設定、変更ができます。これにより、BSC を利用したプログラムのデバッグが可能になります。SH-4 シリーズで設定できるメモリ種別を**表** 2.3に示します。

表2.3 SH-4 **シリーズで設定できるメモリ種別**

アドレス	設定できるメモリ種別
H'00000000〜H'03FFFFFF（エリア０）	**通常メモリ、バースト** ROM、MPX
H'04000000〜H'07FFFFFF（エリア１）	**通常メモリ、バイト制御** SRAM、MPX
H'08000000〜H'0BFFFFFF（エリア２）	**通常メモリ、**DRAM、SDRAM、MPX
H'0C000000〜H'0FFFFFFF（エリア３）	**通常メモリ、**DRAM、SDRAM、MPX
H'10000000〜H'13FFFFFF（エリア４）	**通常メモリ、バイト制御** SRAM、MPX
H 14000000〜H'17FFFFFF（エリア５）	**通常メモリ、バースト** ROM、MPX
H'18000000〜H'1BFFFFFF（エリア６）	**通常メモリ、バースト** ROM、MPX
H'1C000000〜H'1FFFFFFF（エリア７）	**設定不可**
H'7C000000〜H'7C001FFF	**内蔵** RAM（変更不可）
H'E0000000〜H'FFFFFFFF	I/O（変更不可）

なお、**表** 2.3では、エリア０〜７に対するアドレスは、上位３ビットを無視して表示しています。つまり、H '00000000 と H '20000000 はともにエリア０に入ります。

シミュレータ・デバッガでは、PCMCIA はサポートしていません。

2.7 DMAC（ダイレクトメモリアクセスコントローラ）

SH-4 シリーズでは、４チャネルの DMAC の動作をシミュレーションします。これにより、DMAC を利用したプログラムのデバッグが可能になります。

2.8 例外処理

シミュレータ・デバッガでは、TRAPA 命令、一般不当命令、スロット不当命令、アドレスエラー例外の発生を検出します。さらに、SH-3/SH-3E/SH-3DSP/SH-4 シリーズでは MMU 関連の例外（TLB ミス例外、TLB 保護例外、TLB 無効例外、初期ページ書き込み例外）処理を、SH-2E/SH-3E/SH-4 シリーズでは FPU 例外処理をシミュレーションします。これにより、例外発生時のシミュレーションも行うことができます。

例外処理のシミュレーションは、EXEC_MODE における実行モードの選択に従って、以下の手順で行います。

(1) SH-1/SH-2/SH-2E/SH-DSP シリーズ

① [Continue]を選択した場合（続行モード）
- (a) 命令の実行中に例外の発生を検出します。
- (b) PCとSRを退避します。
- (c) ベクタ番号に対応するベクタアドレスから、スタートアドレスを読み出します。
- (d) スタートアドレスから命令実行を行います。スタートアドレスが０の場合は、例外処理を中止し、例外処理エラーが発生したことを表示した後、シミュレータ・デバッガのコマンド待ち状態に戻ります。

② [Stop]を選択した場合（停止モード）

前項の（a）（b）（c）を行い停止します。

(2) SH-3/SH-3E/SH-3DSP シリーズ

① [Continue]を選択した場合（続行モード）
- (a) 命令の実行中に例外の発生を検出します。
- (b) SPCとSSRにそれぞれPCとSRを退避します。
- (c) SRのBLビット、RBビット、MDビットを１にセットします。
- (d) 制御レジスタEXPEVTに例外コードを設定します。なお、必要に応じて他の制御レジスタにも適切な値を設定します。
- (e) 例外の要因に応じたベクタアドレスをPCに設定します。
  （SRのBLビットが１のときに例外が検出されると例外の要因にかかわらずリセットのベクタアドレスH'A0000000が設定されます。）
- (f) PCに設定されたアドレスから命令実行を行います。

② [Stop]を選択した場合（停止モード）

前項の（a）（b）（c）（d）（e）を行い停止します。

(3) SH-4 シリーズ

① [Continue]を選択した場合（続行モード）
- (a) 命令に実行中に例外の発生を検出します。
- (b) SPCとSSRにそれぞれPCとSRを退避します。
- (c) SRのBLビット、RBビット、MDビットを１にセットします。
- (d) リセット時SRのFD（FPUディスエーブル）ビットを０にセットします。
- (e) 制御レジスタEXPEVTに例外コードを設定します。なお、必要に応じて他の制御レジスタにも適切な値を設定します。
- (f) 例外の要因に応じたベクタアドレスをPCに設定します。
  （SRのBLビットが１のときに例外が検出されると例外の要因にかかわらずリセットのベ

クタアドレスH'A0000000が設定されます。）

(g) PCに設定されたアドレスから命令実行を行います。

② [Stop]を選択した場合（停止モード）

前項の（a）（b）（c）（d）（e）（f）を行い停止します。

2.9 制御レジスタ

SH-3/SH-3E/SH-3DSP/SH-4 シリーズでは、例外処理、MMU、キャッシュの制御で使用するメモリにマッピングされた制御レジスタを、さらに SH-4 シリーズでは、BSC、DMAC の制御で使用する制御レジスタをサポートしています。また、SH-DSP with Cache ではキャッシュの制御で使用する CCR レジスタのみをサポートしています。これにより、例外処理、MMU 制御、キャッシュ制御、BSC 制御 および DMAC 制御を行っているデバッグ対象プログラムのシミュレーション、デバッグを行うことができます。

本シミュレータ・デバッガでサポートしている制御レジスタを以下の通りです。

・MMU	PTEH	ページテーブルエントリ上位レジスタ
	PTEL	ページテーブルエントリ下位レジスタ
	TTB	変換テーブルベースレジスタ
	TEA	TLB 例外アドレスレジスタ
	MMUCR	MMU 制御レジスタ
・例外処理	TRA	TRAPA 例外レジスタ
	EXPEVT	例外事象レジスタ
	INTEVT	割り込み事象レジスタ
・キャッシュ	CCR	キャッシュ制御レジスタ
	CCR2*[1]	キャッシュ制御レジスタ２
	QACR0,1*[2]	キューアドレス制御レジスタ０、１
・BSC	BCR1,2*[2]	バスコントロールレジスタ１、２
	WCR1~3*[2]	ウェイトステートコントロールレジスタ１～３
	MCR*[2]	個別メモリコントロールレジスタ
	RTCSR*[2]	リフレッシュタイマコントロール/ステータスレジスタ
	RTCNT*[2]	リフレッシュタイマカウンタ
	RTCOR*[2]	リフレッシュタイムコンスタントレジスタ
	RFCR*[2]	リフレッシュカウントレジスタ
・DMAC	SAR0～3*[2]	DMA ソースアドレスレジスタ０～３
	DAR0～3*[2]	DMA デスティネーションアドレスレジスタ０～３
	DMATCR0～3*[2]	DMA トランスファカウントレジスタ０～３
	CHCR0～3*[2]	DMA チャンネルコントロールレジスタ０～３
	DMAOR*[2]	DMA オペレーションレジスタ

【注】 *[1] **は、SH-3DSP シリーズのみサポートしています。**

*[2] **は、SH-4 シリーズのみサポートしています。**

SH-DSP with Cache **では** CCR **レジスタのみサポートしています。**

シミュレータ・デバッガでは、PCMCIA インタフェース、シンクロナス DRAM モードレジスタはサポートしていません。

2.10 トレース

シミュレータ・デバッガは、実行結果をトレースバッファに書き込みます。トレースバッファの大きさは TRACE_CONDITION コマンドで 1024,4096,16384,32768 命令から選択できます（「4.60 TRACE_CONDITION」参照）。トレース情報の取得条件は、TRACE_CONDITION コマンドで指定します。取得したトレース情報は、Trace ウィンドウに表示します。表示する内容は、以下のとおりです。

（1） SH-1/SH-2/SH-2E/SH-DSPシリーズ
- 累計命令実行サイクル数
- 命令アドレス
- パイプライン実行状況
- 命令ニーモニック
- データアクセス情報（転送先および転送データ）
- C/C++またはアセンブラソース

（2） SH-3/SH-3Eシリーズ
- 累計命令実行サイクル数
- アドレスバス上のデータ
- データバス上のデータ
- 命令コード
- １命令ごとにつけられる番号
- 命令ニーモニック
- フェッチした命令の番号（メモリアクセスを行わないとき [ ]で囲んで表示）
- デコードした命令の番号
- 実行した命令の番号
- メモリアクセスした命令の番号
- ライトバックした命令の番号
- データアクセス情報（転送先および転送データ）
- C/C++またはアセンブラソース

（3） SH-3DSPシリーズ
- 累計命令実行サイクル数
- プログラムカウンタ値
- 命令コード
- フェッチした命令の番号（メモリアクセスを行わないとき [ ]で囲んで表示）
- デコードした命令の番号
- 実行した命令の番号
- メモリアクセスした命令の番号
- ライトバックした命令の番号
- １命令ごとにつけられる番号
- 命令ニーモニック
- データアクセス情報（転送先および転送データ）
- C/C++またはアセンブラソース

（4） SH-4シリーズ
- 累計命令実行サイクル数（CPU 内部クロック）
- プログラムカウンタ値

- フェッチした命令コード
- EXパイプラインで実行、メモリアクセス、ライトバックした命令の番号
- LSパイプラインで実行、メモリアクセス、ライトバックした命令の番号
- BRパイプラインで実行、メモリアクセス、ライトバックした命令の番号
- FPパイプラインで実行、メモリアクセス、ライトバックした命令の番号
- 実行する命令に割り振られた命令番号
- 実行する命令のメモリ上のアドレス、命令コード、ニーモニック
- データアクセス情報（転送先および転送データ）
- C/C++またはアセンブラソース

2.11 標準入出力およびファイル入出力処理

シミュレータ・デバッガでは、デバッグ対象プログラムからの標準入出力（コンソールおよびキーボード)およびファイル入出力の処理をサポートしています。サポートしている入出力処理を以下に示します。

表2.4 **入出力機能一覧**

番号	機能コード	機能名	内　容
1	H'21	GETC	**標準入力からの１バイト入力**
2	H'22	PUTC	**標準出力への１バイト出力**
3	H'23	GETS	**標準入力からの１行入力**
4	H'24	PUTS	**標準出力への１行出力**
5	H'25	FOPEN	**ファイルのオープン**
6	H'06	FCLOSE	**ファイルのクローズ**
7	H'27	FGETC	**ファイルからの１バイト入力**
8	H'28	FPUTC	**ファイルへの１バイト出力**
9	H'29	FGETS	**ファイルからの１行入力**
10	H'2A	FPUTS	**ファイルへの１行出力**
11	H'0B	FEOF	**エンドオブファイルのチェック**
12	H'0C	FSEEK	**ファイルポインタの移動**
13	H'0D	FTELL	**ファイルポインタの現在位置を得る**

この機能を実現するために、TRAP_ADDRESS コマンドを使用します。ユーザは、デバッグ対象プログラム内に入出力用の特定の位置へのサブルーチン分岐命令（BSR、JSR、BSRF）を記述します。シミュレータ・デバッガ起動後 TRAP_ADDRESS コマンドで、その特定の位置を指定して、プログラムを実行します。シミュレータ・デバッガでは、デバッグ対象プログラムの命令を実行中に、指定された位置へのサブルーチンコール命令（BSR、JSR、BSRF）を検出すると、R0、R1 の内容をパラメータとして、入出力処理を実行します。入出力処理が終了すると、サブルーチンコール命令の次の命令からシミュレーションを再開します。

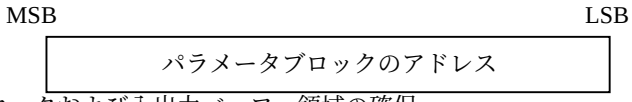
詳細は**「**4.62　**ＴＲＡＰ＿＿ＡＤＤＲＥＳＳ」**を参照ください。

従って、システムコールを行う前にデバッグ対象プログラムの中で次の設定をしておきます。

・R0 レジスタ：**表** 2.4に示す機能コード

MSB　１バイト	１バイト		LSB
H'01	機能コード	- - -	- - -

・R1 レジスタ：パラメータブロックのアドレス
（パラメータブロックの内容は各機能の説明を参照してください。）



・パラメータブロックおよび入出力バッファ領域の確保

なお、パラメータブロックの各パラメータにアクセスする場合は、該当するパラメータのサイズでアクセスしてください。

入出力処理が終了すると、システムコール命令の次の命令からシミュレーションを再開します。

【注】 JSR、BSR、BSRF **命令をシステムコール命令として実行すると、**JSR、BSR、BSRF **の次命令はスロット命令ではなく、通常の命令として実行されます。このため、スロット命令と通常命令で実行結果に違いが発生する命令をシステムコール用** JSR、BSR、BSRF **命令の次命令には、記述しないでください。**

各入出力機能を**図** 2.1の形式で説明します。

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

【パラメータブロック】 (5)
【パラメータ】 (6)

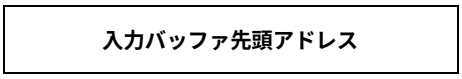
図2.1　**入出力機能の説明形式**

各項目の内容は、以下の通りです。

（1） **表**2.4に対応する番号
（2） 機能名
（3） 機能コード
（4） 入出力の機能
（5） 入出力のパラメータブロック
（6） 入出力のパラメータ

1	GETC H'21	標準入力からの１バイト入力

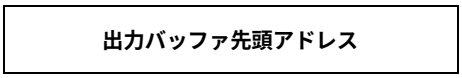
【パラメータブロック】



【パラメータ】 **・入力バッファ先頭アドレス（入力）**
入力データを書き込むバッファの先頭アドレス

2	PUTC H'22	標準出力への１バイト出力

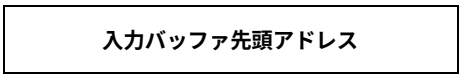
【パラメータブロック】



【パラメータ】 **・出力バッファ先頭アドレス（入力）**
出力データを格納しているバッファの先頭アドレス

3	GETS H'23	標準入力からの１行入力

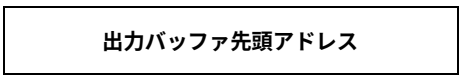
【パラメータブロック】



【パラメータ】 **・入力バッファ先頭アドレス（入力）**
入力データを書き込むバッファの先頭アドレス

4	PUTS H'24	標準出力への１行出力

【パラメータブロック】



【パラメータ】 **・出力バッファ先頭アドレス（入力）**
出力データを格納しているバッファの先頭アドレス

5	FOPEN H'25	ファイルのオープン

FOPEN によってファイルをオープンすると、ファイル番号が返されます。以後のファイル入出力、ファイルクローズ等では、このファイル番号を用います。同時にオープンできる最大ファイル数は 256 です。

【パラメータブロック】

	１バイト	１バイト
+0	実行結果	ファイル番号
+2	オープンモード	未使用
+4 +6	ファイル名先頭アドレス	

【パラメータ】

・実行結果（出力）

0 **正常終了**

–1 **エラー**

・ファイル番号（出力）

オープン処理以降のファイルアクセスで使用する番号

・オープンモード（入力）

H'00	"r"
H'01	"w"
H'02	"a"
H'03	"r+"
H'04	"w+"
H'05	"a+"
H'10	"rb"
H'11	"wb"
H'12	"ab"
H'13	"r+b"
H'14	"w+b"
H'15	"a+b"

各モードの内容は以下の通りです。

"r"	**読み出し用にオープンする。**
"w"	**空ファイルを書き込み用にオープンする。**
"a"	**ファイルの最後から書き込み用にオープンする。**
"r+"	**読み出し，書き込み用にオープンする。**
"w+"	**空ファイルを読み出し，書き込み用にオープンする。**
"a+"	**読み出し追加用にオープンする。**
"b"	**バイナリモードでオープンする。**

・ファイル名先頭アドレス（入力）

ファイル名を格納している領域の先頭アドレス

6	FCLOSE H'06	ファイルのクローズ

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号

【パラメータ】

・実行結果（出力）

0　正常終了

–1　エラー

・ファイル番号（入力）

ファイルオープン時に返される番号

7	FGETC H'27	ファイルから1バイトのデータ読み出し

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号
+2	未使用	
+4 +6	入力バッファ先頭アドレス	

【パラメータ】

・実行結果（出力）

0　正常終了

–1　EOF 検出

・ファイル番号（入力）

ファイルオープン時に返される番号

・入力バッファ先頭アドレス（入力）

入力データを書き込むバッファの先頭アドレス

8	FPUTC H'28	ファイルへ1バイトのデータ書き込み

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号
+2	未使用	
+4 +6	出力バッファ先頭アドレス	

【パラメータ】

・実行結果（出力）

0　正常終了

–1　エラー

・ファイル番号（入力）

ファイルオープン時に返される番号

・出力バッファ先頭アドレス（入力）

出力データを格納しているバッファの先頭アドレス

9	FGETS H'29	ファイルから文字列データの読み出し

改行コードまたは NULL コードを検出するまで，またはバッファサイズに達するまでファイルから文字列データを読み出します。

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号
+2	バッファサイズ	
+4 +6	入力バッファ先頭アドレス	

【パラメータ】

- **実行結果（出力）**
  - **0　正常終了**
  - **–1**　EOF **検出**
- **ファイル番号（入力）**
  - **ファイルオープン時に返される番号**
- **バッファサイズ（入力）**
  - **データを格納する領域のサイズ**
  - **（バイト単位で最大** 256 **バイトまで）**
- **入力バッファ先頭アドレス（入力）**
  - **入力データを書き込むバッファの先頭アドレス**

10	FPUTS H'2A	ファイルへ文字列データ書き込み

ファイルへ文字列データ書き込みます。文字列終端記号の NULL コードはファイルには書き込まれません。

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号
+2	未使用	
+4 +6	出力バッファ先頭アドレス	

【パラメータ】

- **実行結果（出力）**
  - 0　**正常終了**
  - –1　**エラー**
- **ファイル番号（入力）**
  - **ファイルオープン時に返される番号**
- **出力バッファ先頭アドレス（入力）**
  - **出力データを格納しているバッファの先頭アドレス**

11	FEOF H'0B	エンドオブファイルのチェックを行う

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号

【パラメータ】

・実行結果（出力）

0　EOF でない

–1　EOF 検出

・ファイル番号（入力）

ファイルオープン時に返される番号

12	FSEEK H'0C	指定位置にファイルポインタを移動

【パラメータブロック】

	1バイト	1バイト
+0	実行結果	ファイル番号
+2	ディレクション	未使用
+4 +6	オフセット	

【パラメータ】

・実行結果（出力）

0　正常終了

–1　エラー

・ファイル番号（入力）

ファイルオープン時に返される番号

・ディレクション（入力）

0　オフセットはファイルの先頭からのバイト数

1　オフセットは現在のファイルポインタからのバイト数

2　オフセットはファイルの最後尾からのバイト数

・オフセット（入力）

ディレクションで指定した位置からのバイト数

13	FTELL H'0D	ファイルポインタの現在位置を調査

【パラメータブロック】

	１バイト	１バイト
+0	**実行結果**	**ファイル番号**
+2	**未使用**	
+4 +6	**オフセット**	

【パラメータ】

・実行結果（出力）

0　**正常終了**

–1　**エラー**

・ファイル番号（入力）

ファイルオープン時に返される番号

・オフセット（出力）

現在のファイルポインタの位置
（ファイル先頭からのバイト数）

以下に標準入力（キーボード）から１文字入力する例を示します。

```
            MOV.L     PAR_ADR、R1
            MOV.L     REQ_COD、R0
            MOV.L     CALL_ADR、R3
            JSR       @R3
            NOP
STOP        NOP
SYS_CALL    NOP
            .ALIGN    4
CALL_ADR    .DATA.L   SYS_CALL
REQ_COD     .DATA.L   H'01210000
PAR_ADR     .DATA.L   PARM
PARM        .DATA.L   INBUF
INBUF       .RES.B    2
            .END
```

2.12 ブレーク条件

デバッグ対象プログラムのシミュレーションを中断する条件として以下のものがあります。

- ブレーク系コマンドの条件成立によるブレーク
- デバッグ対象プログラムの実行時エラー検出によるブレーク
- トレースバッファ満杯によるブレーク
- SLEEP 命令実行によるブレーク
- [STOP]ボタンによるブレーク

(1) ブレーク系コマンドの条件成立によるブレーク

ブレーク条件を設定するコマンドには次の 5 種類があります。

- BREAKPOINT ：命令実行位置によるブレーク
- BREAKACCESS ：メモリ範囲のアクセスによるブレーク
- BREAKDATA ：メモリ書き込みデータ値によるブレーク
- BREAKREGISTER ：レジスタ書き込みデータ値によるブレーク
- BREAKSEQUENCE ：実行順序を指定したブレーク

デバッグ対象プログラム実行中にブレーク条件が成立した場合、ブレークポイントの命令を実行しないで停止するか、実行してから停止するかを**表** 2.5に示します。

表2.5 **ブレーク条件成立時の処理**

コマンド名	ブレーク条件成立命令	
	実行する	実行しない
BREAKPOINT		○
BREAK_ACCESS	○	
BREAK_DATA	○	
BREAK_REGISTER	○	
BREAK_SEQUENCE		○

BREAKPOINT、BREAKSEQUENCE の場合、実行命令の先頭位置以外にブレークポイントを設定するとブレークを検出できません。

デバッグ対象プログラム実行中にブレーク条件が成立すると、ブレーク条件成立のメッセージをステータスバーに表示して、命令実行を中断します。

(2) デバッグ対象プログラムの実行時エラー検出によるブレーク

シミュレータ・デバッガでは、CPU の例外発生機能では検出できないプログラムの誤りを検出するためにシミュレーションエラーを設けています。これらのエラーが発生した場合に、シミュレーションを停止するか、続行するかを Exec_Mode コマンドまたは Exec_Mode ウィンドウにより選択できます。エラーの種類、エラーメッセージ、エラー発生要因、および続行時のシミュレータ・デバッガの動作を**表** 2.6に示します。

表2.6 シミュレーションエラー一覧

エラーの種類/メッセージ	エラー発生要因	続行モード時処理
メモリアクセスエラー/ Memory Access Error	• 確保されていないメモリ領域をアクセスしようとした • 書き込み不可属性を持つメモリへ書き込みを行おうとした • 読み出し不可属性を持つメモリから読み出しを行おうとした • メモリが存在しない領域をアクセスしようとした	メモリへの書き込み時、何も書き込まない。メモリ読み出し時、全ビット"1"を読み出す
命令実行不正/ Illegal Operation	• DIV1 命令でゼロ除算を行おうとした • SETRC 命令でゼロを書き込もうとした	デバイスと同一動作をする
DSP 命令実行不正/ Illegal DSP Operation	• PSHA 命令で 32 ビットを越えるシフトを行おうとした • PSHL 命令で 16 ビットを越えるシフトを行おうとした	デバイスと同一動作をする
DSP 命令コード不正/ Invalid DSP Instruction Code	• DSP 命令の命令コードが不正である	常に停止する
TLB マルチヒット/ TLB Multiple Hit	• MMU によるアドレス変換で複数の TLB エントリにヒットした • （SH-3/SH-3E/SH-3DSP シリーズのみ）	動作不定

停止モードの場合、シミュレーションエラーが発生するとシミュレータ・デバッガは、命令実行を中止してエラーメッセージを表示後、コマンド待ち状態に戻ります。シミュレーションエラー停止後の PC の状態を表 2.7に示します。なお、シミュレーションエラー停止後 SR の内容は変化しません。

表2.7 シミュレーションエラー停止時のレジスタ

エラーの種類	PC の内容
メモリアクセスエラー	命令読込時： ・SH-DSP/SH-3DSP シリーズ エラーが発生した命令の３命令前のアドレス ・SH-1/SH-2/SH-2E/SH-3/SH-3E/SH-4 シリーズ エラーが発生した命令の前命令のアドレス ただし、分岐先を読み込んだ時にエラーが発生した時は、スロット位置のアドレス 命令実行時： エラーが発生した命令の次命令のアドレス
命令実行不正	エラーが発生した命令の次命令のアドレス
DSP 命令実行不正	エラーが発生した命令の２命令先のアドレス
DSP 命令コード不正	エラーが発生した命令の２命令先のアドレス
TLB マルチヒット	エラーが発生した命令のアドレス

シミュレーションエラーが発生する命令を組み込んだプログラムのデバッグは、次の手順で行ってください。

(a) 最初は停止モードで実行させて、意図している箇所以外にエラーがないかどうかを確認してください。

(b) 確認が完了したら、続行モードで実行してください。

【注】 停止モードでエラーが発生して停止した状態から、モードを続行モードに変更してシミュレーションを再開すると、正しくシミュレーションされない場合があります。シミュレーションを再開する場合は、レジスタ内容（汎用レジスタ、コントロールレジスタ、システムレジスタ）、メモリの内容をエラー発生前の状態に戻してから再実行するようにしてください。

(3) トレースバッファ満杯によるブレーク

TRACE_CONDITION コマンドで Break モードを指定し、命令実行中にトレースバッファが満杯になると、シミュレータ・デバッガは、実行を中断します。中断時には以下のメッセージを表示します。

+++ 5006 : Trace buffer full

(4) SLEEP 命令実行によるブレーク

命令実行時に、SLEEP 命令を実行すると、シミュレータ・デバッガは実行を中断します。中断時には、以下のメッセージをステータスバーに表示します。

+++ 5007 :Sleep

【注】 実行を再開する場合は、PC の値を再開位置の命令アドレスに変更してください。

(5) [STOP]ボタンおよび Ctrl+C キーによるブレーク

命令実行中にユーザにより強制的に実行を中断することができます。中断時には以下のメッセージをステータスバーに表示します。

*** 503 : User break

Go、Step コマンドにより実行を再開できます。

2.13 浮動小数点データ

実数データとして浮動小数点数を指定することができます。これにより、データ値等で浮動小数点を扱う場合の操作が容易になります。浮動小数点を指定できる項目は次の通りです。

・[Break Data]や[Break Register]メニューを開いた時のデータ指定
・Memory Display ウィンドウにおけるデータ
・Registers ウィンドウ（SH4 以外）または Floating Point Registers ウィンドウにおけるレジスタ値表示および入力値

浮動小数点データフォーマットは、ANSI C の浮動小数点フォーマットに準拠しています。

シミュレータ・デバッガでは、ROUND_MODE コマンドにより、浮動小数点数の 10 進→2 進変換で発生する丸めのモードを選択することができます。次の 2 通りより選択します。

- RN（Round to Nearest）
- RZ（Round to Zero）

なお、10 進→2 進変換および 2 進→10 進変換で非正規化数が指定された場合、RZ モードでは 0 に変換し、RN モードでは非正規化数のまま処理します。また、10 進→2 進変換時にオーバーフローが発生した場合、RZ モードでは浮動小数点数の最大値を、RN モードでは無限大を設定します。

2.14 SH-4 のサポート機能

2.14.1 BSC

SH-4 では、BSC によるバス制御機能を削除して、SRAM、バス幅、およびステート数指定のみをサポートしています。

SH-4 シミュレータ・デバッガで設定できるメモリ種別を下表に示します。

表2.8 SH-4 **シミュレータ・デバッガで設定できるメモリ種別**

アドレス	設定できるメモリ種別
H'00000000 ～ H'03FFFFFF (**エリア** 0)	SRAM
H'04000000 ～ H'07FFFFFF (**エリア** 1)	
H'08000000 ～ H'0BFFFFFF (**エリア** 2)	
H'0C000000 ～ H'0FFFFFFF (**エリア** 3)	
H'10000000 ～ H'13FFFFFF (**エリア** 4)	
H'14000000 ～ H'17FFFFFF (**エリア** 5)	
H'18000000 ～ H'1BFFFFFF (**エリア** 6)	
H'1C000000 ～ H'1FFFFFFF (**エリア** 7)	**設定不可**
H'7C000000 ～ H'7C001FFF	**内蔵** RAM (**変更不可**)
H'E0000000 ～ H'FFFFFFFF	I/O (**変更不可**)

2.14.2 DMA

DMA 機能は削除しました。

2.14.3 外部/内部クロック比

外部/内部クロック比を 3:1 から 1:1 に変更しました。

2.14.4 制御レジスタ

SH-4 シミュレータ・デバッガの制御レジスタサポート状況を下表に示します。

表2.9 SH-4 **シミュレータ・デバッガの制御レジスタサポート状況(その** 1)

レジスタ名	サポート状況
PTEH	○
PTEL	○
TTB	○
TEA	○
MMUCR	○
EXPEVT	○
INTEVT	○
TRA	○
CCR	○
QACR0,1	○
SAR0-3	×
DAR0-3	×
DMATCR0-3	×
CHCR0-3	×
DMAOR	×

MCR	×
BCR1,2	△
WCR1,2	△
RTCSR	×
RTCNT	×
RFCR	×

【注】 ○はサポート。
×は未サポート
△は部分サポート
未サポート部分については、制御レジスタダイアログボックス等を使用して値の変更、および参照は行なえますが、シミュレータ・デバッガの実行には影響を与えません。

次に部分サポートの各制御レジスタについてフィールドごとのサポート状況を示します。

表2.10　SH-4 **シミュレータ・デバッガの制御レジスタサポート状況(その** 2))

レジスタ名	フィールド名	サポート状況
BCR1	ENDIAN	○
	MASTER	×
	A0MPX	×
	A0BST	×
	A5BST	×
	A6BST	×
	DRAMTP	×
	IPUP	×
	OPUP	×
	A1MBC	×
	A4MBC	×
	BREQEN	×
	PSHR	×
	MEMMPX	×
	HIZMEM	×
	HIZCNT	×
	A56PCM	×
BCR2	A6SZ-A0SZ	○
	PORTEN	×
WCR1	DMAW	×
	A6IW-A0IW	○
WCR2	A6W-A0W	○
	A6B	×
	A5B	×
	A0B	×

【注】 ○はサポート。
×は未サポート
未サポート部分については、制御レジスタダイアログボックス等を使用して値の参照および変更は行なえますが、シミュレータ・デバッガの実行には影響を与えません。

3. 操作方法

本章では、インタフェースソフトの起動方法、およびシミュレータ・デバッガの動作の確認方法を説明します。

本章では、ホストシステム上でのウィンドウシステムの操作知識があることを前提として説明します。

3.1 パスおよび環境変数の設定

インストーラを用いてパスおよび環境変数をシェルスクリプトに追加しない場合は、手動でパスおよび環境変数を設定してください。

(1) パスの設定

パスに、インタフェースソフト（cas）をインストールしたディレクトリを追加します。

% set△path=($path△<インタフェースソフトディレクトリパス>)(RET)

(2) ライブラリパスの設定

ライブラリパスに、協調検証用ライブラリをインストールしたディレクトリを追加します。

- SPARC の場合

  % setenv△LD_LIBRARY_PATH= ($LD_LIBRARY_PATH△
  <協調検証用ライブラリディレクトリパス>)(RET)

- HP9000 の場合

  % setenv△SHLIB_PATH= ($SHLIB_PATH△
  <協調検証用ライブラリディレクトリパス>)(RET)

(3) 環境変数の設定

インタフェースソフトが使用する環境変数を次に示します。

- HS_CA_HOM

  インタフェースソフト定義ファイルをインストールしたディレクトリを指定します。

  % setenv△HS_CA_HOM△<定義ファイルディレクトリへのパス>(RET)

- SIMCPU

起動するシミュレータデバッガの CPU 名を環境変数 SIMCPU に設定してください。環境変数 SIMCPU を省略した場合、SH1 が選択されます。

% setenv△SIMCPU△<CPU 名>(RET)

<CPU 名> ： SH1 （SH-1）
SH2 （SH-2）
SH3 （SH-3）
SH4 （SH-4）

SH4BSC（SH-4withBSC）
SH2E　（SH-2E）
SH3E　（SH-3E）
SHDSP　（SH-DSP）
SHDSPC（SH-DSPwithCache）
SH3DSP（SH-3DSP）
SH2DSP（SH-2DSP）

- HS_CA_SIM
  インタフェースソフト定義ファイルの環境ファイル名を指定します。

  % setenv△HS_CA_SIM△<環境ファイル名>(RET)

- EXTTOOLATTRIBUTE
  使用する外部ツールの種別を環境変数に EXTTOOLATTRIBUTE 設定してください。

  % setenv EXTTOOLATTRIBUTE "Co-Verification Tool" (RET)　　：協調検証ツールの場合

- EXTTOOLNAME
  使用する外部ツール名を環境変数に EXTTOOLNAME 設定してください。

  % setenv EXTTOOLNAME Seamless (RET)　　：Seamless の場合
  % setenv EXTTOOLNAME Eaglei (RET)　　：Eaglei の場合

3.2　起動

(1)　インタフェースソフトの起動

インタフェースソフトのコマンド形式を示します。

% csdsh [△<セットアップファイル名>](RET)

起動後、次のメッセージを表示します。

SH SERIES CYCLE-ACCURATE SIMULATOR/DEBUGGER Vn.m
Copyright (C) Hitachi,Ltd.1998
Copyright (C) Hitachi ULSI System Co.,Ltd.1998
Licensed Material of Hitachi,Ltd.

(2)　セットアップファイル

セットアップファイルで、使用するシミュレータ、バックアップファイルおよびリプレイファイルを指定します。セットアップファイルはエディタで作成してください。なお、起動時にセットアップファイルを指定しない場合は、**図** 3.1 で示すセットアップウィンドウを表示しますので、下記の各項目を入力してください。

セットアップファイルまたはセットアップウィンドウでの各項目の指定方法を以下に示します。

(a) シミュレータの指定

シミュレータの指定形式を示します
起動時パラメータについては下記参照してください。

<シミュレータ・デバッガ名>[△<起動時パラメータ>]

SH-1/SH-2 シリーズの場合：sdsh12 [△-cpu=<CPU 情報ファイル名>] (RET)
（１） （２）

SH-2E シリーズの場合：sdsh2e [△-cpu=<CPU 情報ファイル名>] (RET)
（１） （２）

SH-DSP の場合：sdshdsp [△-cpu=<CPU 情報ファイル名>] (RET)
（１） （２）

SH-DSPwithCache の場合：sdshdspc [△-cpu=<CPU 情報ファイル名>] [△-endian={big | little}] (RET)
（１） （２） （３）

SH-2DSP の場合：sdsh2dsp [△-cpu=<CPU 情報ファイル名>] (RET)
（１） （２）

SH-3DSP シリーズの場合：sdsh3dsp [△-cpu=<CPU 情報ファイル名>] [△-endian={big | little}] (RET)
（１） （２） （３）

SH-3/SH-3E シリーズの場合：sdsh3e [△-cpu=<CPU 情報ファイル名>] [△-endian={big | little}]
（１） （２） （３）
[△-cache=<キャッシュサイズ>] (RET)
（４）

SH-4 の場合：sdsh4 [△-cpu=<CPU 情報ファイル名>] [△-endian={big | little}] (RET)
（１） （２） （３）

SH-4withBSC の場合：sdsh4bsc [△-cpu=<CPU 情報ファイル名>] [△-endian={big | little}] (RET)
（１） （２） （３）

（説明）
（１）ホスト計算機に登録されたシミュレータ・デバッガプログラムのコマンド名です。
（２）本オプションを指定するとシミュレータ・デバッガは指定されたファイルからＣＰＵ情報を読み込んでメモリマップとします。ファイル形式を省略すると".cpu"を仮定します。
（３）エンディアンを指定します。（SH-3/SH-3E,SH-DSPwithCache,SH-3DSP,SH-4 シリーズのみ）
big ：ビッグエンディアン（デフォルト）
little：リトルエンディアン
（４）キャッシュ容量を指定します。（SH-3/SH-3E シリーズのみ）
2： 2KB
4： 4KB
8： 8KB（デフォルト）

(b) バックアップファイルの指定（省略可）

バックアップファイルは、終了時のウィンドウ位置とサイズおよびウィンドウ上の設定内容を保存するファイルで、終了時に作成することができます。起動時にバックアップファイル指定すると終了時の設定内容等を回復することができます。バックアップファイルの指定形式を以下に示します。

BAK△<バックアップファイル名>

(c) リプレイファイル

リプレイファイルは、シミュレータ・デバッガ上の操作（シミュレータ・デバッガコマンドの入力およびボタンのクリック）を保存しているファイルで、シミュレータ・デバッガのレコーディング機能で作成できます。起動時にリプレイファイルを指定するとデバッグに必要な初期設定等を自動的に実行することができます。リプレイファイルの指定形式を以下に示します。

REP△<リプレイファイル名>

(d) セットアップファイル例

SH-3 シミュレータを使用し、バックアップファイルおよびリプレイファイルを指定した例を次に示します。

```
#  Set  up  file
sdsh3e
BAK  backup
REP  recover
```

(3) セットアップウィンドウ

セットアップウィンドウは、セットアップファイルに対応した項目を表示します。

セット アップファイルがコマンドラインで指定された場合は、セットアップファイルで指定された内容で起動し、セットアップウィンドウは表示されません。

セットアップファイルが指定されなかった場合、環境変数 SIM_CPU で指定した CPU に対応したセットアップサンプルファイルの内容で、セットアップウィンドウを表示します。

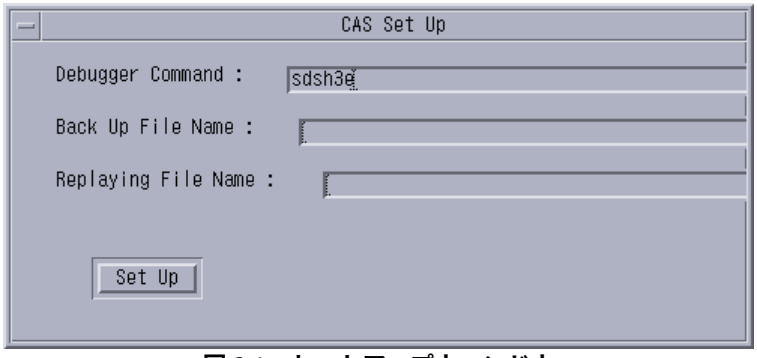


図 3.1 **セットアップウィンドウ**

3.3 ウィンドウ

(1) ベースウィンドウ

セットアップ後、インタフェースソフトはベースウィンドウを表示します。ベースウィンドウ上で、ソースファイルの表示、プログラムの実行、ブレークポイントの設定・解除・シンボル内容表示等のソースレベルデバッグ機能のほかにサブウィンドウの選択、シミュレータ・デバッガのコマンド入力を行うことができます。

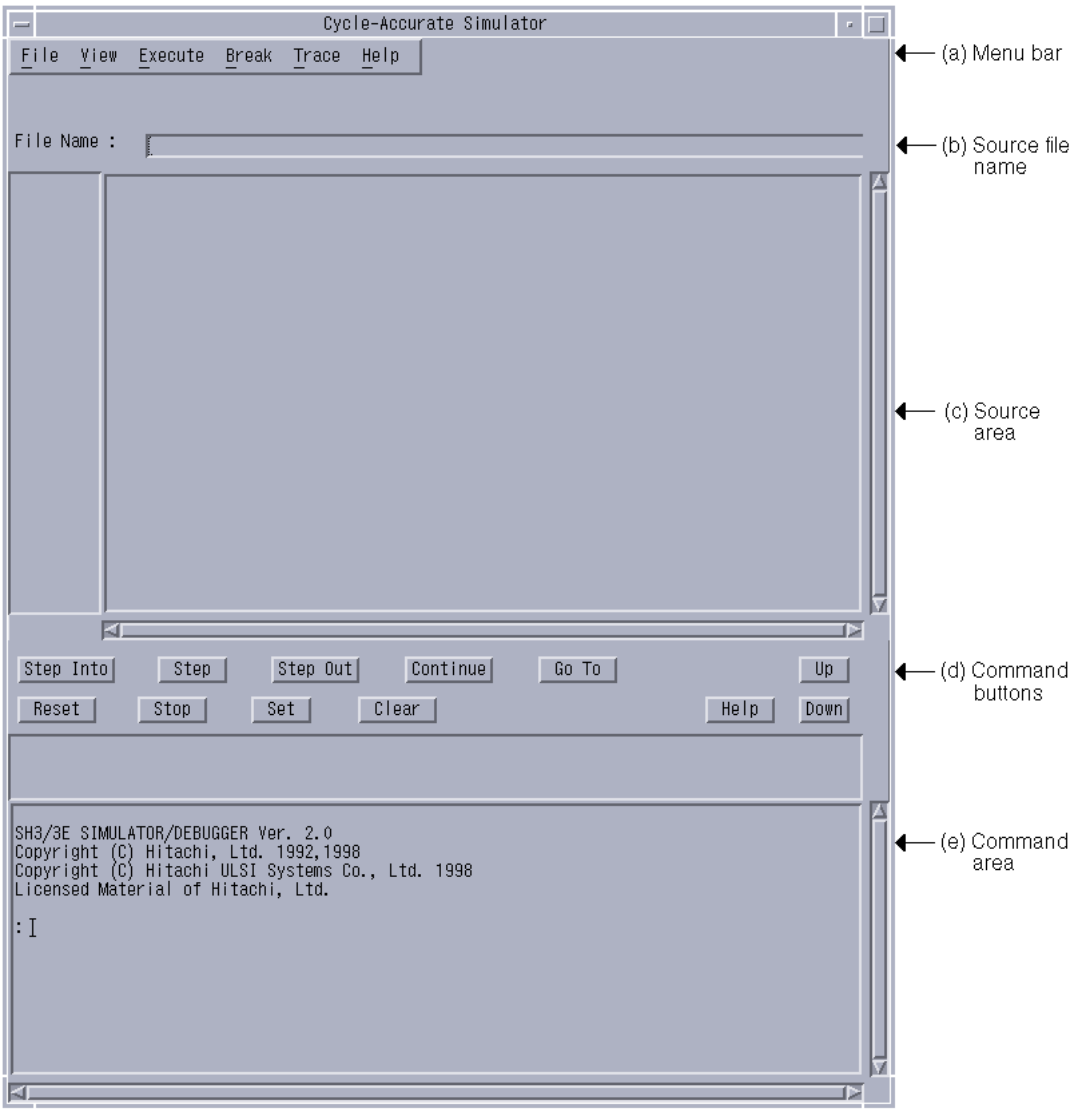


図 3.2 ベースウィンドウ

(a) **メニューバー**

メニューバーの各項目を以下に示します。各項目の詳細は**「6　ウィンドウ」**を参照ください。

- File

  ファイル関連操作（ロード、セーブ等）を行うサブウィンドウを選択することができます。
- View

  レジスタ、逆アセンブルなどの表示関連のサブウィンドウを選択することができます。
- Execute

  実行関連の操作を行うサブウィンドウを選択することができます。
- Break

  ブレーク関連の操作を行うサブウィンドウを選択することができます。
- Trace

  トレース関連の操作を行うサブウィンドウを選択することができます。
- Help

  ヘルプ機能をもつサブウィンドウを選択することができます。

(b) **ソースファイル名**

ソースエリアに表示しているソースファイル名を絶対パス名で表示します。

(c) **ソースエリア**

デバッグ対象ロードモジュールのソースファイルの内容を表示します。
"PC->"マークは、実行が停止した行を示します。（プログラムカウンタの値）
また、ソースファイルの内容は、実行停止時に更新して表示します。
"BP->"マークは、ブレークポイントが設定されている行を示します。

(d) **コマンドボタン**

コマンドボタンの各項目を以下に示します。

- Step Into

  プログラムカウンタが示す行を実行します。プログラムカウンタが示すソース行内に関数呼び出しがある場合、その関数に移った行まで実行します。
- Step

  プログラムカウンタが示す行を実行します。ソース行内に関数の呼び出しがあってもその関数内では停止しません。
- Step Out

  ある関数内にプログラムカウンタがあると、きその関数から呼び出し元に戻った行まで実行します。
- Continue

  プログラムカウンタが示すアドレスから実行を開始します。
- Go To

  ソースエリアで選択した行（クリックした行）まで実行します。
- Reset

  デバッガの Reset コマンドを実行します。

- Stop

  実行中のデバッガコマンドを強制終了させます。

- Set

  ソースエリアで選択した行（クリックした行）にブレークポイントを設定します。
  ("BP->"マークを表示します）
  なお、選択した行にブレークポイントを設定できない（対応するアドレスがない）場合、
  行番号が小さくなる方向に検索し、ブレークポイントを設定します。

- Clear

  ソースエリアで選択した行（クリックした行）のブレークポイントを解除します。
  ("BP->"マークを消去します）

- Help

  本説明をヘルプウィンドウに出力します。

- Up

- Down

  ソースエリアとコマンドエリアの表示比率（縦方向)を変更することができます。
  Up　：ソースエリアが縮小され、コマンドエリアが拡大します。
  Down：ソースエリアが拡大され、コマンドエリアが縮小します。

【注】　最適化コンパイルしたロードモジュールでは Step、Step Into、Step Out、Continue、Go To **ボタンによるプログラム実行によってソースエリアの**"PC->"**マークがソースファイルの記述通りに移動しないことがあります。**

(e)　コマンドエリア

シミュレータ・デバッガのコマンドを直接入力することができます。また、コマンドの実行結果も表示します。

実行できるコマンドについては「4　シミュレータ・デバッガのコマンド」を参照ください。

(2) サブウィンドウとヘルプウィンドウ

ベースウィンドウのメニューボタンをプルダウンし、メニュー項目（”メニュー項目...”）を選択するとサブウィンドウを開きます。各サブウィンドウにはベースウィンドウと同様に Help ボタンがあり、そのサブウィンドウの説明を表示するヘルプウィンドウを開くことができます。

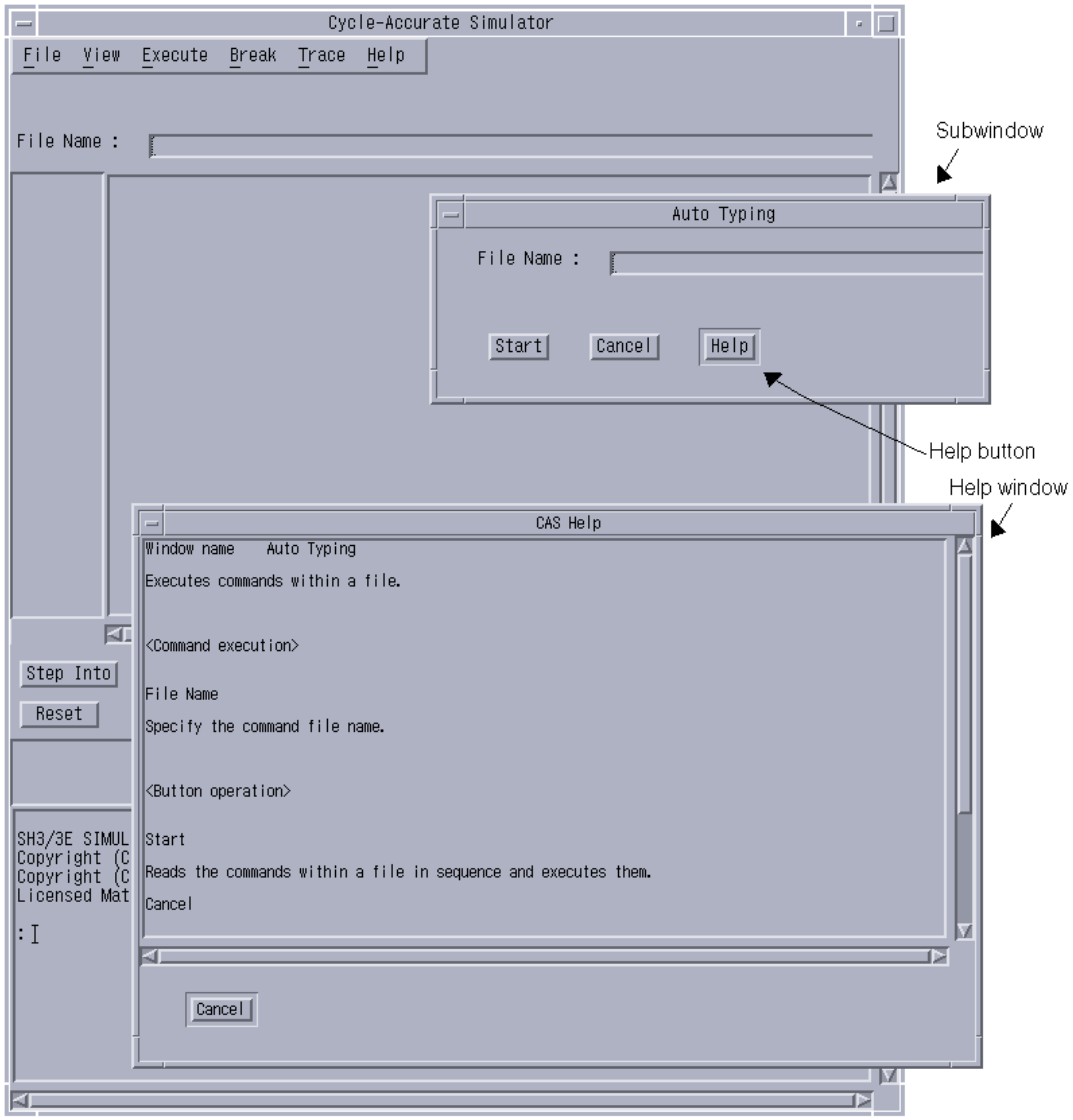


図 3.3 サブウィンドウとヘルプウィンドウ

(3) エラーウィンドウとマニュアルウィンドウ

エラーが発生した場合、エラーウィンドウを開いてエラーメッセージを表示します。エラーウィンドウ上の Manual ボタンによって、エラーメッセージについての説明を表示するマニュアルウィンドウを開くことができます。エラーメッセージが複数ある場合は、エラーウィンドウで説明するエラーメッセージを選択してから、Manual ボタンを押してください。

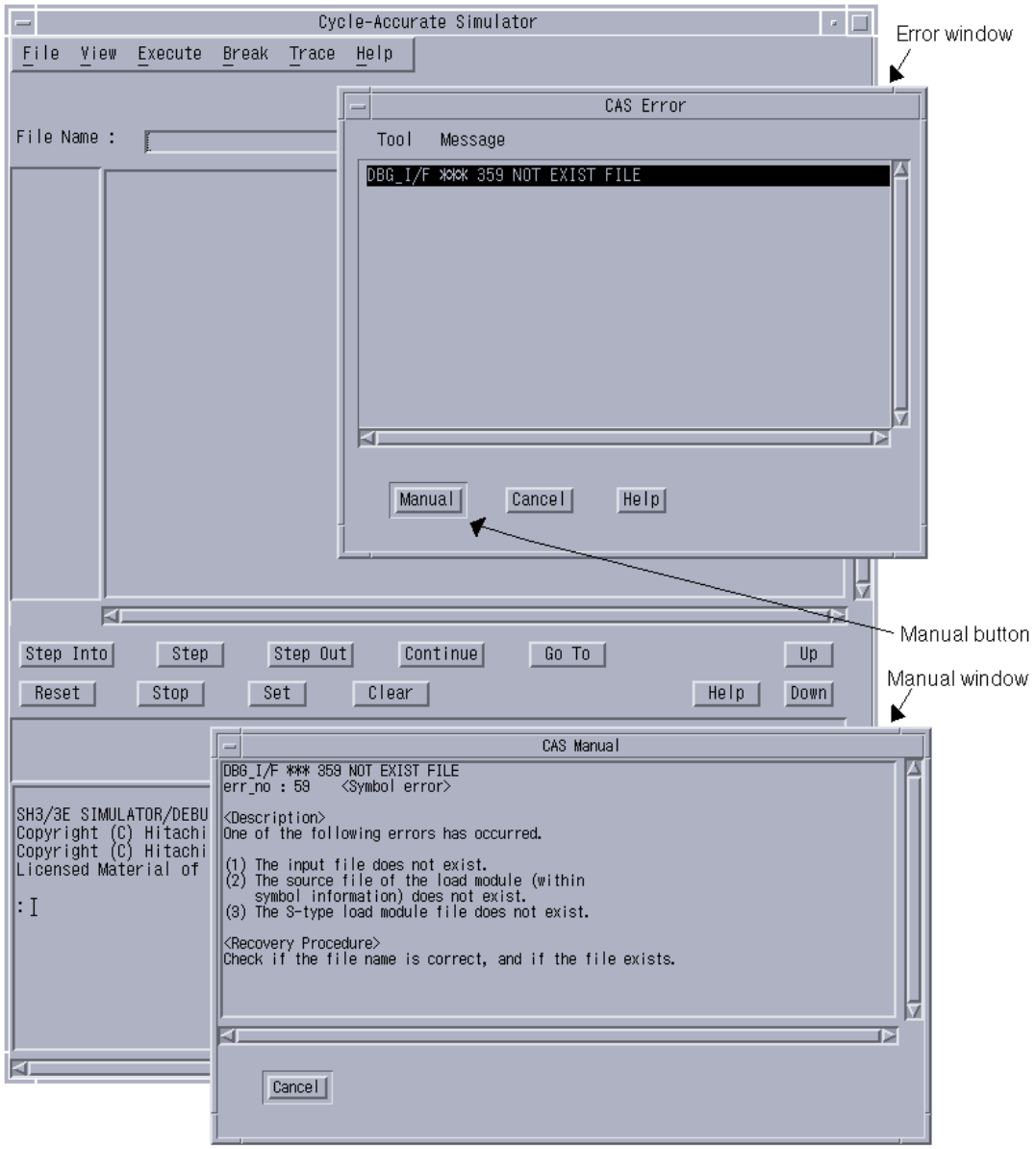


図 3.4 エラーウィンドウとマニュアルウィンドウ

3.4 ロードモジュールのロード

ロードウィンドウを用いて、ロードモジュールをロードします。ロードはモジュール間最適化ツールで出力したファイル名を入力して、CA&DEBUGGER ボタンを押してください。シミュレータ・デバッガは自動的にロードモジュールのメモリの確保を行います。

ロードモジュールをモジュール間最適化ツールで出力した時と違うディレクトリにソースファイルがある場合、ロードウィンドウの Old Path Name にモジュール間最適化ツールで出力したときのソースファイルのパスを、New Path Name に現在のソースファイルのパスを入力してください。

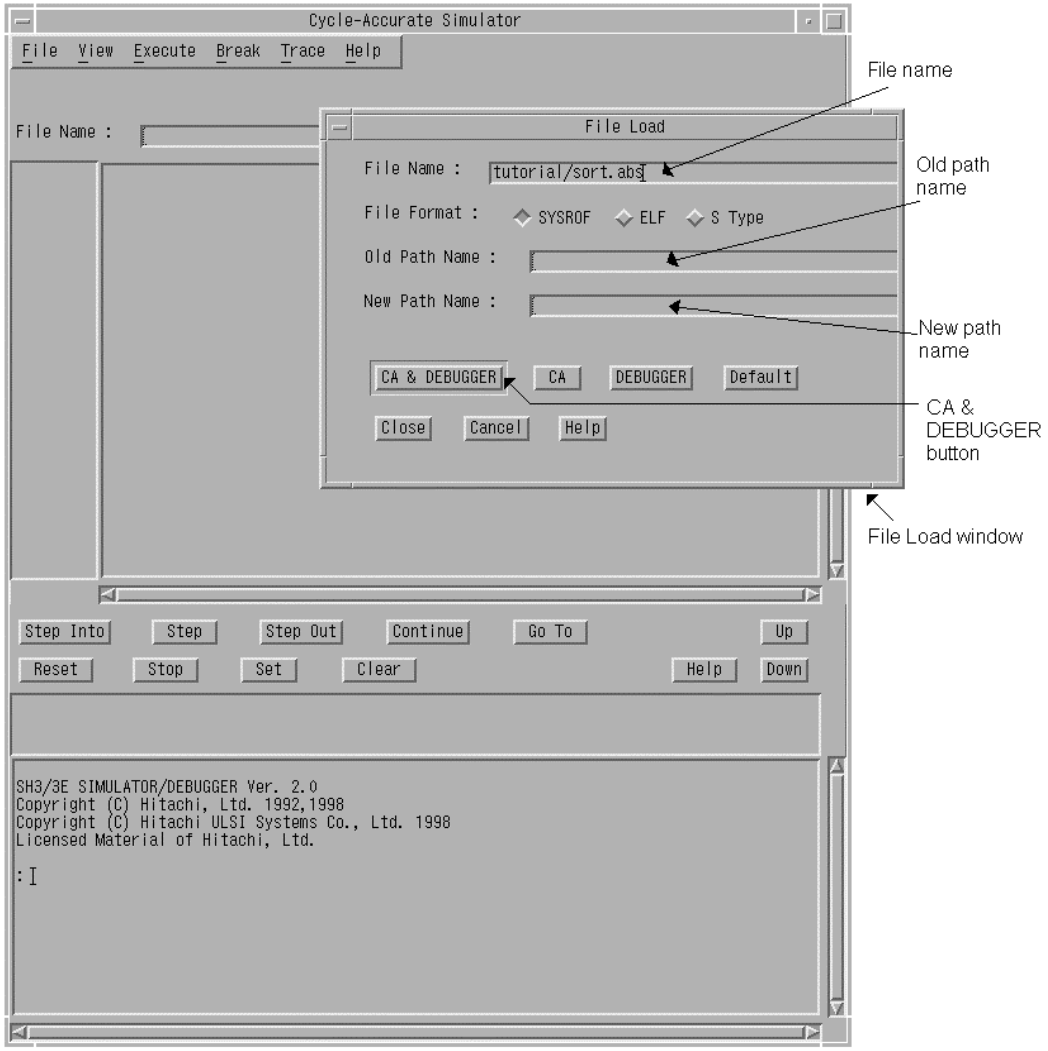


図 3.5 ロードの入力例

3.5 ソースファイルの表示

ソースファイルは、次の手順で表示することができます。また、表示するソースファイルを変更す

ることができます。

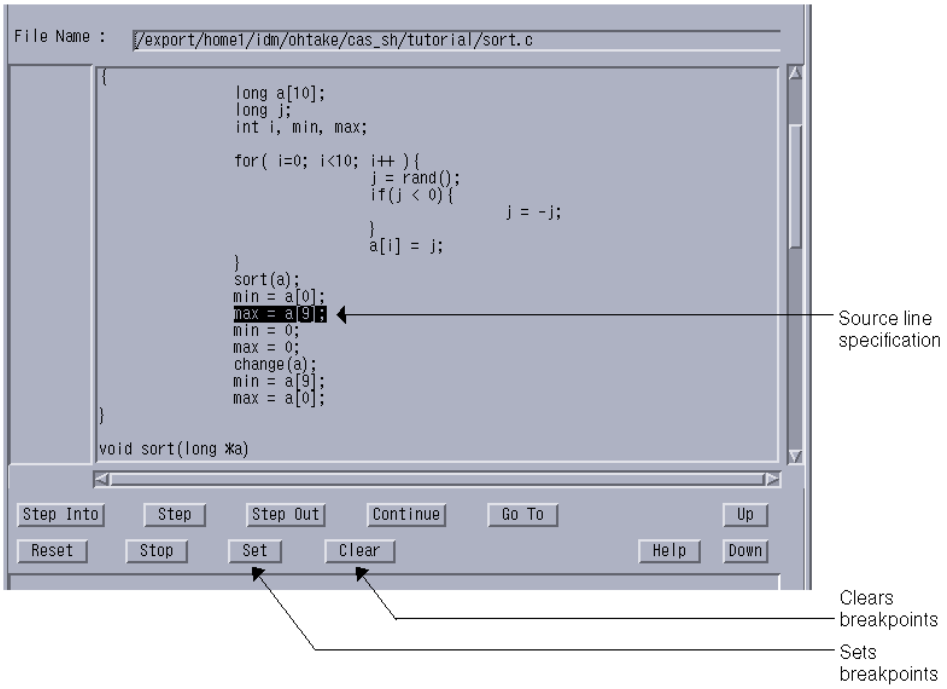
ロード終了後、ソースファイル名ウィンドウまたは関数名ウィンドウを開いてください。
表示したいソースファイル名または関数名を選択し、Base Window Display ボタンを押してください。

図 3.6 ソースファイルの選択例

3.6 ブレークポイントの設定

ブレークポイントはソースエリアでソース行を指定し、Set ボタンを押すと設定できます。ブレー

クポイントの解除は、同様にソース行を指定して Clear ボタンを押します。



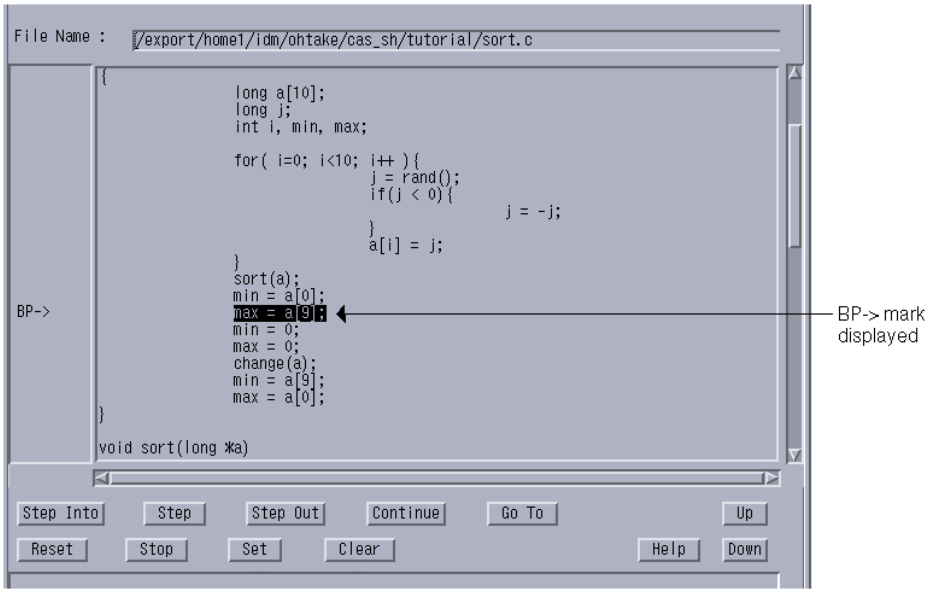


図 3.7　ブレークポイントの設定例

3.7 アドレスのシンボリックデバッグ指定

アドレスは、数値の代わりに行番号、関数名または変数名を用いてシンボリックに入力することができます。（**図** 3.8 参照）

シンボルの指定形式を以下に示します。＜ユニット名＞は、通常、コンパイルまたはアセンブルで出力されるオブジェクトモジュールのファイル名から、”．＜サフィックス＞”を除いた文字列です。＜ユニット名＞および＜ファイル名＞に特殊文字（英数字、'_'および'$'以外の文字）が含まれていない場合は、”（ダブルクォーテション）を省略することができます。

(1) 関数名

＜関数名＞

(2) 変数名

変数名としては、C 言語の変数名および定数名、アセンブリ言語のラベル名および EQU 名が指定できます。EQU 名では、値をアドレスとします。C 言語の構造体／共用体メンバで指定する場合は、＜構造体名．メンバ名＞の形式で入力してください。

！＜変数名＞

3.8 プログラムの実行

Register ウィンドウまたはコマンドエリアで、プログラムカウンタ(PC)およびスタックポインタ(SP)を設定し、コマンドボタン(Step Into、Step、Step Out、Continue、Go To)を使用することでプログラムの実行を行うことができます。

また、Go ウィンドウで実行開始アドレスを指定し、実行を開始することもできます。

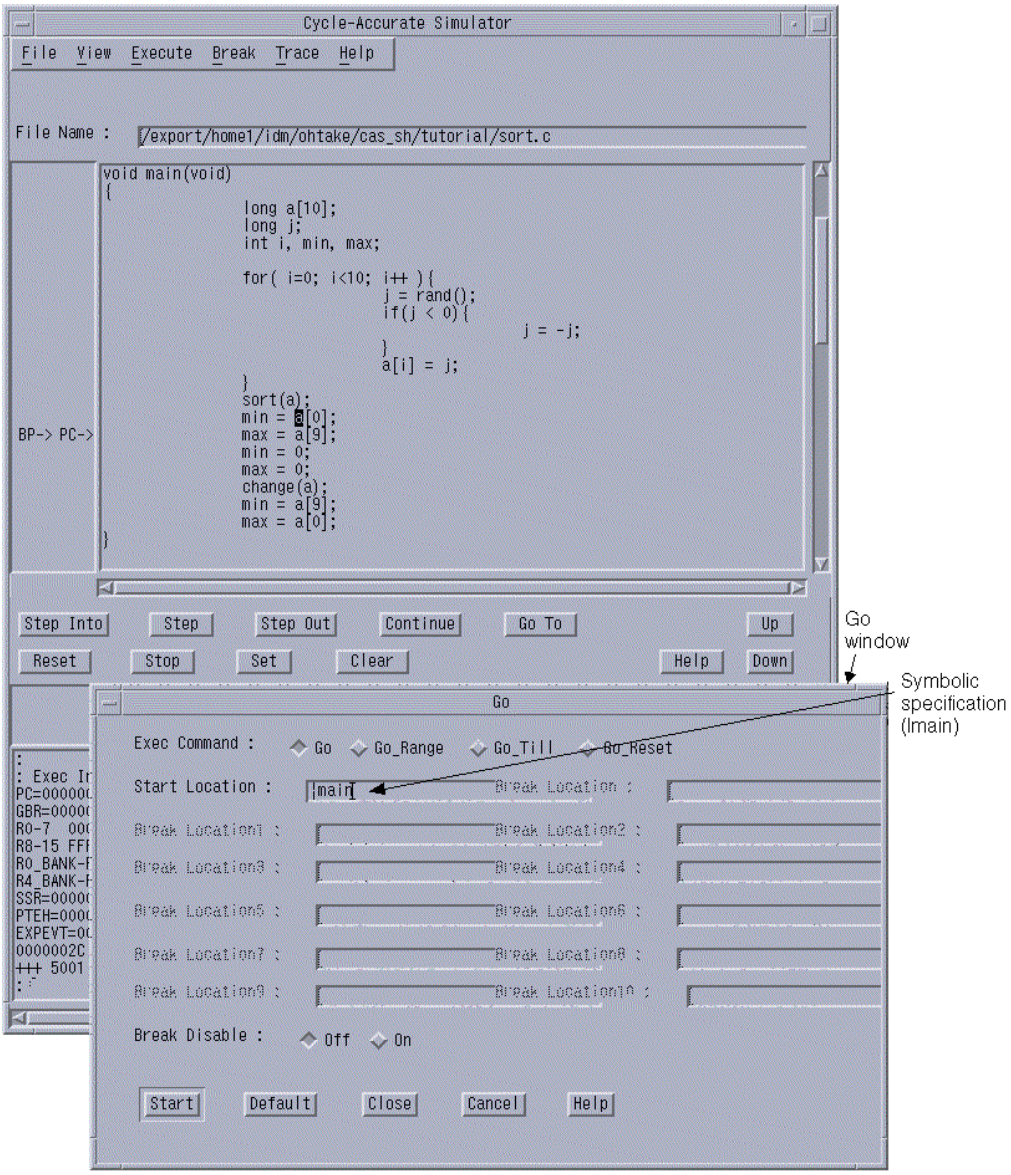


図 3.8 実行ウィンドウの入力例

3.9 変数内容の表示

変数内容は、次の手順で表示することができます。

- **ソースエリアで変数名を選択します。（図** 3.9 **では変数** a **を選択）**
- View **メニューで** Symbol Value **ウィンドウを開きます。**

  （変数名が Symbol Name **に取得されます）**
- Symbol Value **ウィンドウの** Set **ボタンを押します。**

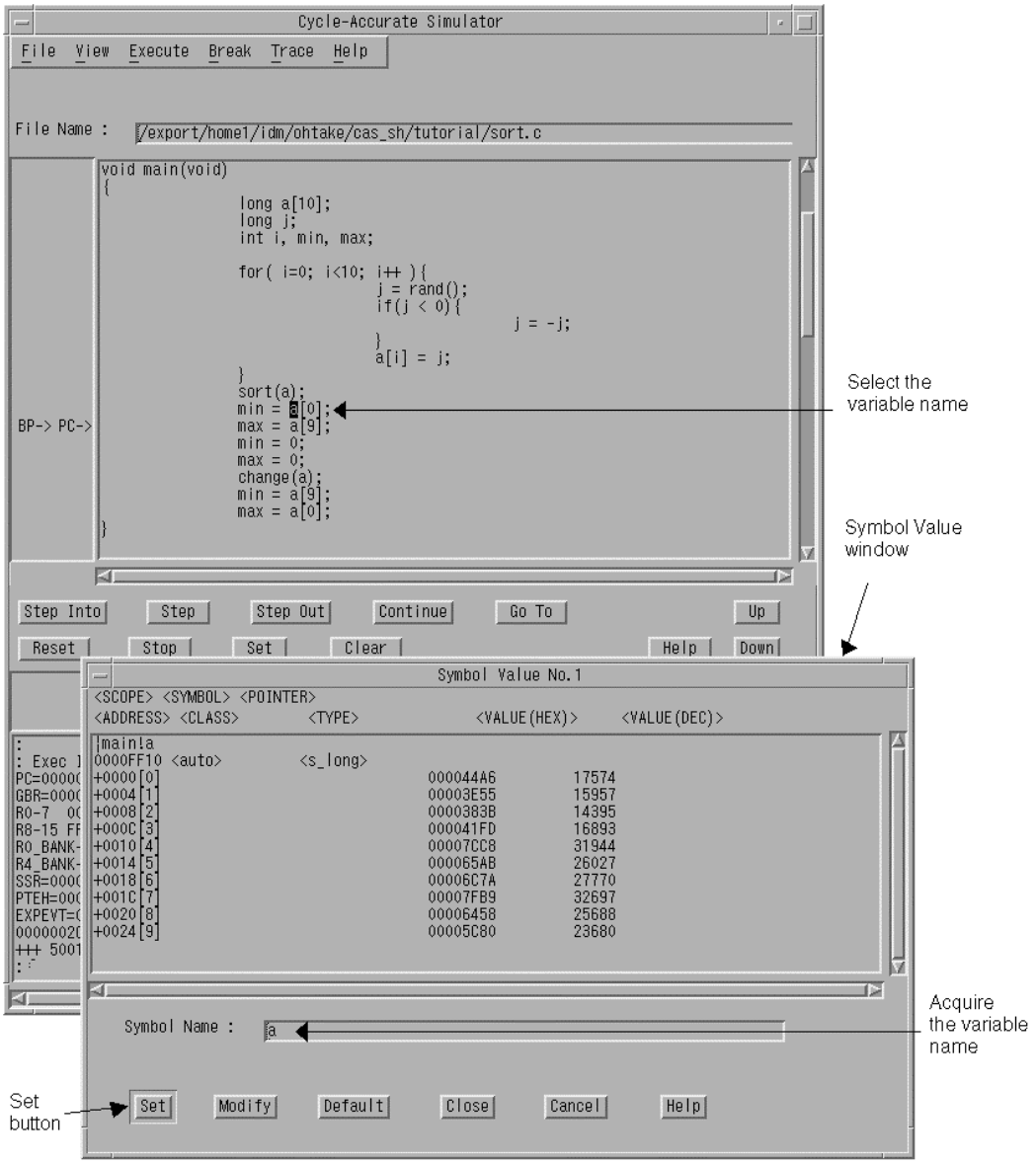


図 3.9 **変数内容の表示例**

3.10 実行パフォーマンスの測定

次の手順で実行パフォーマンスを測定することができます。

- View メニューで Performance Analysis ウィンドウを開きます。
- Function Name or Delete Index に測定する関数名を入力し Add ボタンを押します。
- Start ボタンを押すと測定が開始されます。
- Display ボタンで測定結果を値で表示します。
- Graph ボタンで測定結果をグラフ表示します。

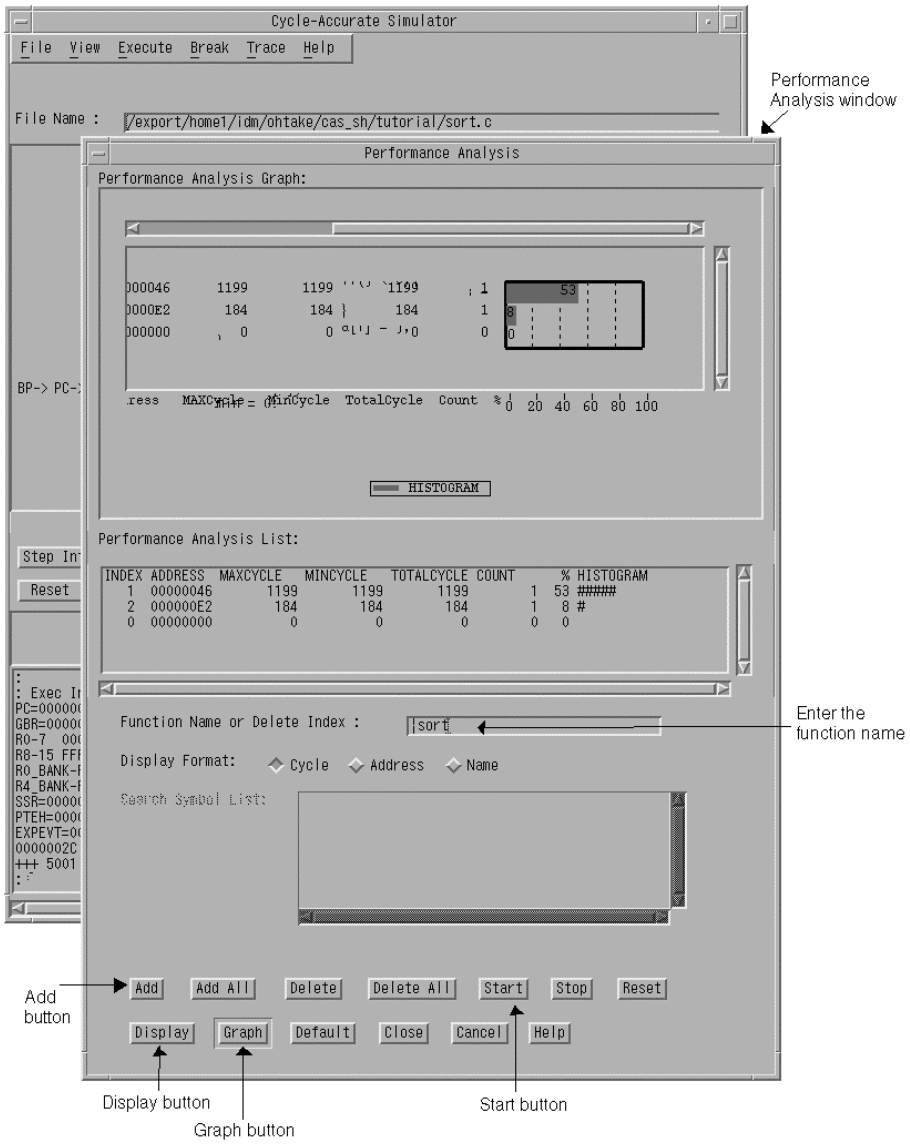


図 3.10 パフォーマンス表示例

3.11 スタック使用状況の解析

次の手順でスタック使用状況の解析をすることができます。

- View メニューで Stack Analysis ウィンドウを開きます。
- Start ボタンを押すとスタック使用状況の解析が開始されます。
- Display ボタンで解析結果を値で表示します。
- Graph ボタンで解析結果をグラフ表示します。

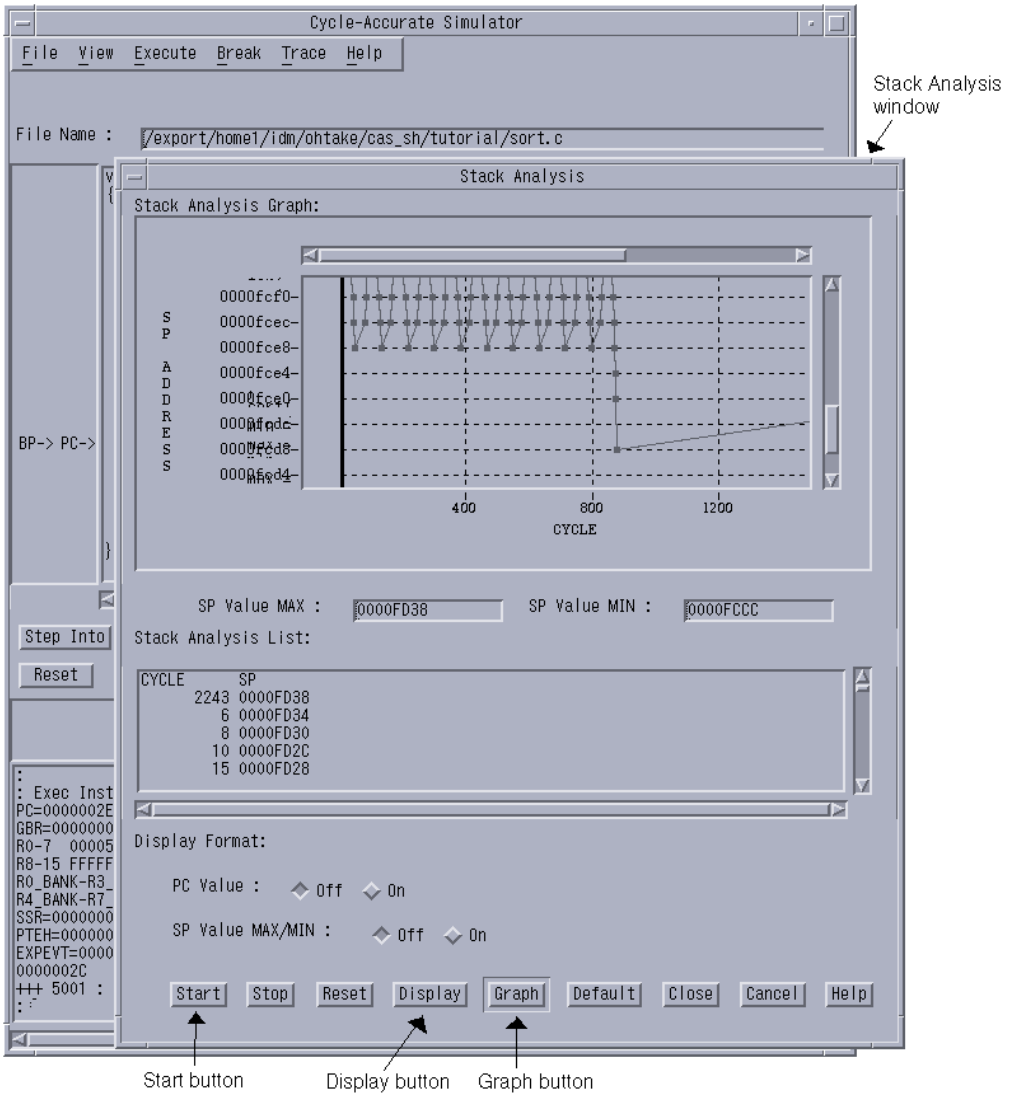


図 3.11 スタックトレース表示例

3.12 終了

File メニューの Quit を選択すると、Quit ウィンドウを表示します。Quit ウィンドウの Quit ボタンを押すと終了します。

バックアップ選択で"Window"を選択すると、インタフェースソフトのウィンドウ位置とサイズおよびウィンドウ上の設定情報を保存することができます。

設定情報を保存するファイル名のデフォルトは"HS_CA.BAK"です。

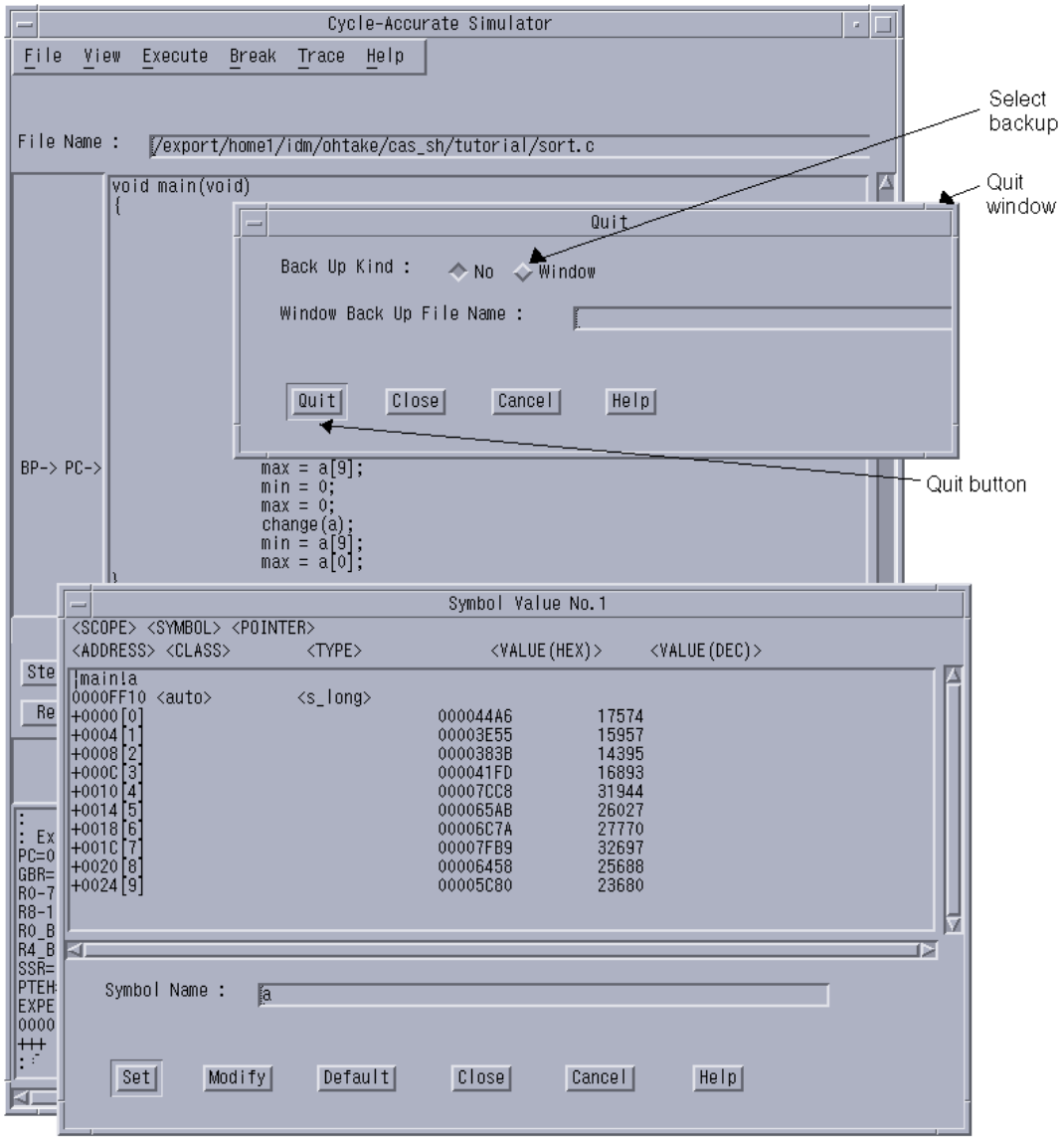


図 3.12 終了ウィンドウの入力例

4. シミュレータ・デバッガのコマンド

コマンドラインで使用できるシミュレータ・デバッガのコマンド一覧を**表** 4.1 に示します。

表 4.1　**コマンド一覧**

項番	コマンド名	短縮形	機 能
1	ASSEMBLE	AS	**ラインアセンブル**
2	BREAK_CLEAR	BC	**ブレークポイントの解除**
3	BREAK_ENABLE	BE	**ブレークポイントの有効／無効**
4	BREAKACCESS	BA	**メモリ範囲のアクセスによるブレーク条件の設定**
5	BREAKACCESS_DISPLAY	BAD	**メモリ範囲のアクセスによるブレーク条件の表示**
6	BREAKDATA	BD	**メモリのデータ値によるブレーク条件の設定**
7	BREAKDATA_DISPLAY	BDD	**メモリのデータ値によるブレーク条件の表示**
8	BREAKPOINT	BP	**命令実行位置によるブレークポイントの設定**
9	BREAKPOINT_DISPLAY	BPD	**命令実行位置によるブレークポイントの表示**
10	BREAKREGISTER	BR	**レジスタのデータ値によるブレーク条件の設定**
11	BREAKREGISTER_DISPLAY	BRD	**レジスタのデータ値によるブレーク条件の表示**
12	BREAKSEQUENCE	BS	**実行順序を指定したブレークポイントの設定**
13	BREAKSEQUENCE_DISPLAY	BSD	**実行順序を指定したブレークポイントの表示**
14	COMPARE	CMP	**メモリ内容の比較**
15	DATA_SEARCH	DS	**データ検索**
16	DISASSEMBLE	DA	**メモリ内容の逆アセンブル表示**
17	DISPLAY_CHARACTERS	DCH	**文字列の表示**
18	EXEC_MODE	EM	**実行モードの切り替え**
19	EXTTOOL	EX	**外部ツールの操作**
20	FILE_LOAD	FL	**ファイルのロード**
21	FILE_SAVE	FS	**ファイルへの保存**
22	GO	G	**命令の連続実行**
23	GO_RANGE	GR	**命令の連続実行（範囲指定）**
24	GO_RESET	GS	**ベクタアドレスからの実行**
25	GO_TILL	GT	**命令の連続実行（停止位置指定）**
26	HELP	HE	**コマンド名、コマンド入力形式の表示**
27	LOAD_STATUS	LS	**シミュレータ・デバッガのメモリとレジスタ内容の状態回復**
28	LOG	LO	**実行履歴ファイルの作成開始**
29	LOG_ENABLE	LE	**実行履歴ファイルの作成の有効／無効**
30	LOG_STOP	LT	**実行履歴ファイルの作成終了**

項番	コマンド名	短縮形	機　能
31	MAP_CLEAR	MC	メモリ領域の解除
32	MAP_DISPLAY	MI	メモリ領域の表示
33	MAP_SET	MS	メモリ領域の設定
34	MEMORY_DISPLAY	MD	メモリ内容の表示
35	MEMORY_EDIT	ME	メモリ内容の変更
36	MEMORY_FILL	MF	メモリ領域の初期設定
37	MEMORY_MOVE	MV	メモリブロックの転送
38	PERFORMANCE_ANALYSIS	PA	実行効率測定のための設定
39	PERFORMANCE_ANALYSIS_CLEAR	PC	実行効率測定の解除
40	PERFORMANCE_ANALYSIS_DISPLAY	PD	実行効率測定結果表示
41	PERFORMANCE_ANALYSIS_ENABLE	PE	実行効率測定の有効／無効
42	QUIT	Q	シミュレータ・デバッガの終了
43	RADIX	RX	基数の設定
44	REGISTER	R	レジスタ内容の表示
45	RESET	RS	シミュレータ・デバッガのリセット
46	ROUND_MODE	RM	浮動小数点丸めモードの設定、表示
47	SAVE_STATUS	SS	現在のシミュレータ・デバッガの状態保存
48	STACK_ANALYSIS	SA	スタック使用量測定の有効／無効、初期化
49	STACK_ANALYSIS_DISPLAY	SD	スタック使用量測定結果の表示
50	STATUS	ST	シミュレータ・デバッガの状態表示
51	STEP	S	サブルーチンを１ステップとしてステップ実行
52	STEP_G	SG	範囲指定ステップ実行
53	STEP_INTO	SI	ステップ実行
54	STEP_INTO_G	SIG	範囲指定ステップ実行
55	TLB	TLB	ＴＬＢ内容の変更
56	TLB_DUMP	TLBD	ＴＬＢ内容の表示
57	TLB_FLUSH	TLBF	ＴＬＢのフラッシュ
58	TLB_SEARCH	TLBS	ＴＬＢ内容の検索
59	TRACE	T	トレースバッファの表示
60	TRACE_CONDITION	TC	トレース条件の設定、トレースの開始、終了
61	TRACE_CLEAR	TL	トレース条件の初期化
62	TRAP_ADDRESS	TA	システムコール開始位置の設定
63	TRAP_ADDRESS_DISPLAY	TD	システムコール開始位置の表示
64	TRAP_ADDRESS_ENABLE	TE	システムコール開始位置の有効／無効
65	.<register>	.	レジスタ内容の変更

各コマンド名、省略形および機能は、**図** 4.1の形式で説明します。

(1) (2)	(3)

【形式】 (4)
【パラメータ】 (5)
【機能】 (6)
【説明】 (7)
【留意事項】 (8)
【使用例】 (9)

図 4.1 **コマンドの説明形式**

各項目の内容は、以下の通りです。

(1) コマンド名
(2) コマンドの省略形
(3) コマンドの機能
(4) コマンドの入力形式
なお、△は 1 つ以上の空白（スペースまたはタブ）、[A]は A が省略可能であることを示します。
(5) コマンドのパラメータ
(6) コマンドの機能の説明
(7) コマンドの使用方法
(8) コマンドを使用する上での留意事項
(9) コマンドの使用例

4.1 ASSEMBLE

ASSEMBLE AS	ラインアセンブル

【形式】 ASSEMBLE△<開始位置>(RET)

【パラメータ】 ・<開始位置> アセンブルを行いたい位置を指定します。

【機能】 対話形式で入力されたアセンブリ言語記述を行単位で機械語に変換し、指定された位置から格納します。
アセンブルモードでは ．は終了、^ は 1 バイト戻り、Enter キーを押すと 1 バイト進みます。
アセンブリ言語の記述については「SuperH™ RISC engine クロスアセンブラユーザズマニュアル」を参照してください。

【使用例】 入力したアセンブリ言語記述を機械語に変換し、指定された位置 H'400 番地より格納します。

```
: ASSEMBLE 400(RET)
00000400 : (RET)
00000401 : (RET)
00000402 : MOV #H'02E,R1(RET)
00000402 MOV #H'02E,R1
00000404 : ADD R1,R2(RET)
00000404 ADD R1,R2
00000406 : ^(RET)
00000405 : .(RET)
:
```

4.2 BREAK_CLEAR

BREAK_CLEAR BC	ブレークポイントの解除

【形式】 BREAK_CLEAR[△<インデックス>](RET)

【パラメータ】 ・<インデックス> ブレーク No.を指定します。（ブレーク No.はブレークポイント表示で確認します。）
省略すると全てのブレークポイントが対象となります。

【機能】 指定されたブレーク No.のブレークポイントを解除します。
以下のコマンドのブレークポイントを解除できます。
BREAKACCESS、BREAKDATA、BREAKPOINT、BREAKREGISTER、BREAKSEQUENCE

【使用例】 １番目のブレークポイントを解除します。

```
: BREAK_CLEAR 0(RET)
:
```

4.3 BREAK_ENABLE

BREAK_ENABLE BE	ブレークポイントの有効／無効

【形式】 BREAK_ENABLE△{E｜D}[△<インデックス>](RET)

【パラメータ】
- ・有効／無効 {E｜D}
  - E(Enable) ：設定したブレーク条件を有効にします。
  - D(Disable)：設定したブレーク条件を無効にします。
- ・<インデックス> ブレーク No.を指定します。（ブレーク No.はブレークポイント表示で確認します。）
  - 省略すると全てのブレークポイントが対象となります。

【機能】 指定されたブレーク No.のブレークポイントを有効／無効にします。
以下のコマンドのブレークポイントを有効／無効にできます。
BREAKACCESS、BREAKDATA、BREAKPOINT、BREAKREGISTER、BREAKSEQUENCE

【使用例】 (1) 1 番目のブレークポイントを無効にします。

```
: BREAK_ENABLE D 0(RET)
:
```

(2)全てのブレークポイントを有効にします。

```
: BREAK_ENABLE E(RET)
:
```

4.4 BREAKACCESS

BREAKACCESS BA	メモリ範囲のアクセスによるブレーク条件の設定

【形式】 BREAKACCESS△<開始位置>[△<終了位置>][△{R｜W｜RW}](RET)

【パラメータ】
- ・<開始位置>[△<終了位置>] アクセスすると実行を停止する位置、または位置の範囲を指定します。
  - 終了位置を省略すると、開始位置のみが位置の範囲となります。
- ・アクセス種別 {R｜W｜RW}
  - R ：(Read)指定位置をリードしたときにブレークします。
  - W ：(write)指定位置をライトしたときにブレークします。
  - RW：(Read/write)指定位置をリードまたはライトしたときにブレークします。（デフォルト）

【機能】 メモリアクセスによるブレークの設定を行います。
指定したメモリの範囲をアクセスしたときに実行を停止するようメモリの範囲を設定します。
メモリの範囲は、最大 2 個まで設定できます。
また、設定を行うとブレークポイントは自動的に有効となります。

【使用例】　H'1000 番地から H'1100 番地までの範囲をリードまたはライトしたとき、
実行を停止するようブレークポイントを設定します。

```
: BREAKACCESS 1000 1100 RW(RET)
:
```

4.5 BREAKACCESS_DISPLAY

BREAKACCESS_DISPLAY	メモリ範囲のアクセスによるブレーク条件の表示
BAD	

【形式】　BREAKACCESS_DISPLAY(RET)

【機能】　以下のフォーマットでメモリアクセスによるブレークの設定状況を表示します。

<INDEX>　：ブレーク No.
<E/D>　：有効／無効
<START>　：開始位置
<END>　：終了位置
<ATTR>　：アクセス種別

【使用例】　現在の設定状況を表示します。（アドレスは 16 進表示）

```
: BREAKACCESS_DISPLAY(RET)
  <INDEX> <E/D> <START>   <END>       <ATTR>
    001       E    00001000    00001100        RW
:
```

4.6 BREAKDATA

BREAKDATA	メモリのデータ値によるブレーク条件の設定
BD	

【形式】　BREAKDATA△<ブレーク位置>△<データ>[;<サイズ>][△<オプション>](RET)

【パラメータ】
・<ブレーク位置>　ブレーク判定を行うメモリの位置を指定します。
・<データ>　ブレーク条件となるデータ値を指定します。
・<サイズ>　データのサイズ　{B｜W｜L｜D｜S}。
B(Byte)　：バイトデータ
W(Word)　：ワードデータ
L(Long)　：ロングワードデータ（デフォルト）
D(Double Float)　：倍精度浮動小数点データ
S(Single Float)　：単精度浮動小数点データ
・<オプション>　データの一致／不一致　{EQ｜NE}
EQ(Equal)：データが一致したときにブレークします。（デフォルト）
NE(Not Equal)：データが不一致となったときにブレークします。

【機能】　メモリへの書き込みによるブレークの設定を行います。
ユーザプログラム実行中にブレーク条件が成立すると命令実行を停止します。
メモリのデータ値によるブレーク条件は最大 8 個まで設定できます。
また、設定を行うとブレークポイントは自動的に有効となります。

【使用例】　(1)H'2000 番地のメモリにワードサイズで 10 を書き込んだときに実行を停止するようにブレークポイントを設定します。

```
: BREAKDATA 2000 10;W (RET)
:
```

(2)H'AF00 番地のメモリがバイトサイズの 20 以外の値に書き換えられたときに実行を停止するようにブレークポイントを設定します。

```
: BREAKDATA 0AF00 20;B NE (RET)
:
```

4.7 BREAKDATA_DISPLAY

BREAKDATA_DISPLAY BDD	メモリのデータ値によるブレーク条件の表示

【形式】　BREAKDATA_DISPLAY(RET)

【機能】　以下のフォーマットでメモリのデータ値によるブレークの設定状況を表示します。

<INDEX>　：ブレーク No.
<E/D>　：有効／無効
<ADDRESS>　：ブレーク位置
<DATA>　：書き込みデータおよびデータのサイズ
<EQ/NE>　：データの一致／不一致

【使用例】　現在の設定されているブレークポイントを表示します。（位置、データは 16 進表示）

```
: BREAKDATA_DISPLAY (RET)
<INDEX> <E/D> <ADDRESS>          <DATA>              <EQ/NE>
   006      D    0000FF00                  0010:W        EQ
   005      E    0000AF00                    20:B        NE
   004      E    00000100               00000100:L       EQ
   003      E    00000020             1.234568e-12:S     EQ
   002      E    00000010  1.234567890123457e-123:D      EQ

:
```

4.8 BREAKPOINT

BREAKPOINT BP	命令実行位置によるブレークポイントの設定

【形式】 BREAKPOINT△<命令位置>［△<回数>］(RET)

【パラメータ】 ・<命令位置> ブレークポイントの位置を指定します。
・＜回数＞ 指定位置の命令をフェッチする回数を指定します。（H'1～H'3FFF）
省略すると 1 になります。

【機能】 命令実行位置によるブレークポイントの設定を行います。
ユーザプログラム実行中にブレーク条件が成立すると命令実行を停止します。
ブレーク位置の命令は、実行しないで停止します。
ブレークポイントは、最大 255 個まで設定できます。
また、設定を行うとブレークポイントは自動的に有効となります。

【留意事項】 (1)命令の先頭位置以外にブレークポイントを設定するとブレーク成立を検出できません。
(2)命令実行が中断した時点で通過回数はリセットされます。

【使用例】 H'2000 番地の命令を 8 回目に実行しようとしたとき、実行を停止するようブレークポイントを設定します。

```
: BREAKPOINT 2000 8(RET)
:
```

4.9 BREAKPOINT_DISPLAY

BREAKPOINT_DISPLAY BPD	命令実行位置によるブレークポイントの表示

【形式】 BREAKPOINT_DISPLAY(RET)

【機能】 以下のフォーマットで命令実行位置によるブレークの設定状況を表示します。
<INDEX> ：ブレーク No.
<E/D> ：有効／無効
<ADDRESS> ：ブレークポイント位置
<COUNT> ：指定位置の命令をフェッチする回数

【使用例】 現在の設定状況を表示します。（アドレス、回数は 16 進表示）

```
: BREAKPOINT_DISPLAY(RET)
<INDEX> <E/D> <ADDRESS> <COUNT>
  000     E   00002000       8
:
```

4.10 BREAKREGISTER

BREAKREGISTER BR	レジスタのデータ値によるブレーク条件の設定

【形式】 BREAKREGISTER△<レジスタ名>[△<データ>[;<サイズ>][△<オプション>]] (RET)

【パラメータ】
- ・<レジスタ名> ブレーク条件を設定するレジスタ名を指定します。（制御レジスタ名は指定できません）
- ・<データ> ブレーク条件のデータ値を指定します。
  省略した場合は指定レジスタに書き込みが発生したときにブレークします。
- ・<サイズ> データのサイズ {B｜W｜L｜S｜D}
  省略した場合は指定レジスタのサイズとします。
  ただし、単精度浮動小数点データ指定時は省略不可となります。
  B(Byte) ： バイトデータ
  W(Word) ： ワードデータ
  L(Long) ： ロングワードデータ
  S(Single Float)： 単精度浮動小数点データ
  D(Double Float)： 倍精度浮動小数点データ（SH-4 のみ）
- ・<オプション> データの一致／不一致 {EQ｜NE}
  EQ(Equal)：データが一致したときにブレークします。（デフォルト）
  NE(Not Equal)：データが不一致となったときにブレークします。

【機能】 レジスタへのデータ書き込みによるブレークの設定を行います。
R15 の変わりに SP と指定することができます。
指定レジスタをアクセスしたときに実行を停止するようブレーク条件を設定します。
また、設定を行うと、ブレークポイントは自動的に有効になります。
ブレークレジスタは、最大 8 個まで設定できます。

【使用例】 (1)レジスタ R0 に書き込み時、実行を停止するようブレークポイントを設定します。

```
: BREAKREGISTER R0(RET)
:
```

(2) レジスタ R1 が FF になった時、実行を停止するようブレークポイントを設定します。

```
: BREAKREGISTER R1 FF;B(RET)
:
```

(3) レジスタ R2 が FF 以外の値に書き換えられた時、実行を停止するようブレークポイントを設定します。

```
: BREAKREGISTER R2 FF;B NE(RET)
:
```

(4)レジスタ FR1 が 1.0E-5 になった時、実行を停止するようブレークポイントを設定します。

```
: BREAKREGISTER FR1 1.0E-5;S(RET)
:
```

4.11 BREAKREGISTER_DISPLAY

BREAKREGISTER_DISPLAY	レジスタのデータ値によるブレーク条件の表示
BRD	

【形式】 BREAKREGISTER_DISPLAY(RET)

【機能】 以下のフォーマットでレジスタのデータ値によるブレークの設定状況を表示します。

<INDEX> ：ブレーク No.
<E/D> ：有効／無効
<REGISTER> ：レジスタ名
<DATA> ：書き込みデータおよびデータのサイズ
<EQ/NE> ：データの一致／不一致

【使用例】 現在の設定されているブレークポイントを表示します。
データのサイズが S の時は浮動小数点形式、その他のデータは 16 進数で表示します。

```
: BREAKREGISTER_DISPLAY(RET)
<INDEX> <E/D> <REGISTER>          <DATA>        <EQ/NE>
   003     E       FR1          1.000000e-05      EQ
   002     E       R2                     FF      NE
   001     E       R1                     FF      EQ
   000     E       R0                --------     EQ
```

4.12 BREAKSEQUENCE

BREAKSEQUENCE	実行順序を指定したブレークポイント設定
BS	

【形式】 BREAKSEQUENCE△<命令位置１>［△<命令位置２>…△<命令位置８>］(RET)

【パラメータ】 ・<命令位置> シーケンシャルブレークポイントとなる位置を指定します。

【機能】 実行順序指定によるブレークの設定を行います。
指定した位置順に命令を実行し、指定した順序の最後の位置を実行するとき停止するようブレーク条件を設定します。
このコマンドでは位置を最大 8 個まで設定できます。

【留意事項】 (1)命令の先頭位置以外にブレークポイントを設定するとブレーク成立を検出できません。
(2)命令実行が中断した時点で通過情報はリセットされます。

【使用例】 H'2000、 H'2100、 H'3000 番地にシーケンシャルブレークポイントを設定します。

```
: BREAKSEQUENCE 2000 2100 3000(RET)
:
```

4.13 BREAKSEQUENCE_DISPLAY

BREAKSEQUENCE_DISPLAY BSD	実行順序を指定したブレークポイント表示

【形式】 BREAKSEQUENCE_DISPLAY(RET)

【機能】 以下のフォーマットで実行順序指定によるブレークの設定状況を表示します。

<INDEX> ：ブレーク No.
<E/D> ：有効／無効
1ST BREAK POINT = xxxxxxxx ：命令位置１
2ND BREAK POINT = xxxxxxxx ：命令位置２
: :

【使用例】 現在設定しているシーケンシャルブレークポイントを表示します。

```
: BREAKSEQUENCE_DISPLAY(RET)
<INDEX> <E/D>
  008      E    1ST BREAK POINT = 00002000
                2ND BREAK POINT = 00002100
                3RD BREAK POINT = 00002200
:
```

4.14 COMPARE

COMPARE CMP	メモリ内容の比較

【形式】 COMPARE△<開始位置>△<終了位置>△<比較先位置>(RET)

【パラメータ】
・<開始位置> 比較元の開始位置を指定します。
・<終了位置> 比較元の終了位置を指定します。
・<比較先位置> 比較先の先頭位置を指定します。

【機能】 指定範囲内のメモリ内容と比較先のメモリ内容を 1 バイト単位で比較します。
内容の一致しない部分が見つかると、そのアドレスと不一致データを表示します。

【使用例】 H'1000 番地から H'500 バイト分の内容と H'2000 番地から H'500 バイト分の内容を比較します。
データの内容が異なる場合、SOURCE DATA は比較元のアドレスと内容を、COMPARED DATA は比較先のアドレスと内容を表示します。

```
: COMPARE 1000 14FF 2000(RET)
SOURCE DATA       COMPARED DATA
00001005 3F         00002005 42
     :                   :
000014FE 00         000024FE 80
:
```

4.15 DATA_SEARCH

DATA_SEARCH	データ検索
DS	

【形式】 DATA_SEARCH△<開始位置>△<終了位置>△<データ>[;<サイズ>] (RET)

【パラメータ】
- <開始位置> 検索の開始位置を指定します。
- <終了位置> 検索の終了位置を指定します。
- <データ> 検索するデータを指定します。
- <サイズ> データのサイズ {B｜W｜L｜D｜S}
  - B(Byte) ： バイトデータ（デフォルト）
  - W(Word) ： ワードデータ
  - L(Long) ： ロングワードデータ
  - D(Double Float)： 倍精度浮動小数点データ
  - S(Single Float)： 単精度浮動小数点データ

【機能】 指定位置の範囲に指定データが存在するか検索し、存在したアドレスを表示します。

【留意事項】 ワードサイズのデータを検索する場合、開始位置がワード境界（2 の倍数）になるようにしてください。
単精度、倍精度浮動小数点またはロングワードサイズのデータを検索する場合、開始位置がロングワード境界（4 の倍数）になるようにしてください。

【使用例】 H'1000 番地から H'14FF 番地までの間に H'005E があるか検索します。

```
: DATA_SEARCH 1000 14FF 005E;W(RET)
ADDRESS
00001004
00001100
000011A8
:
```

4.16 DISASSEMBLE

DISASSEMBLE	メモリ内容の逆アセンブル
DA	

【形式】 DISASSEMBLE△<開始位置>[△<命令数>] (RET)

【パラメータ】
- <開始位置> 逆アセンブルの開始位置を指定します。
- <命令数> 逆アセンブルする命令数を指定します。（デフォルト=16、最大＝65535）

【機能】 開始位置から命令数で指定された範囲を逆アセンブル表示します。
命令先頭アドレス、命令ニモニック、オペランドを表示します。
不当命令は、命令コードを 16 進数形式で表示します。

【使用例】　H'400 番地から 4 命令分逆アセンブル表示します。

```
: DISASSEMBLE 400 4 (RET)
00000400   STS.L      PR,@-R15
00000402   ADD        #H'C8,R15
00000404   MOV        #H'00,R3
00000406   MOV.L      R3,@(H'08:4,R15)
:
```

4.17 DISPLAY_CHARACTERS

DISPLAY_CHARACTERS DCH	文字列の表示

【形式】　DISPLAY_CHARACTERS△<文字列>(RET)

【パラメータ】　・<文字列>　　任意の文字列を指定します。

【機能】　コマンド名の後ろのスペース１文字以降の文字を表示します。

【使用例】　コンソールに SIMULATOR を表示します。

```
: DISPLAY_CHARACTERS SIMULATOR (RET)
SIMULATOR
:
```

4.18 EXEC_MODE

EXEC_MODE EM	実行モードの切り替え

【形式】　設定　：EXEC_MODE△{S｜C}(RET)
表示　：EXEC_MODE(RET)

【パラメータ】　・実行ﾓｰﾄﾞ指定　{S｜C}

【機能】　デバッグ対象プログラムの命令実行中に、異常を検出した時、実行を停止するか続行するかを選択します。
実行モードを省略すると、現在設定中の実行モードを表示します。
デバッグ対象プログラムの実行中に起きる異常については、**「**2.12(2)　**デバッグ対象プログラム実行時エラー検出によるブレーク」**を参照してください。

【説明】　S(Stop)　：シミュレータ・デバッガでデバッグ対象プログラムの異常（シミュレーションエラー）を検出したとき、実行を停止するモードです。（デフォルト）

C(Continue)：シミュレータ・デバッガでデバッグ対象プログラムの異常を検出したとき、異常を無視して続行するモードです。

設定　：テストの初期段階では停止モードを設定し、それ以降では続行モードを設定してテストを行うと有効です。

表示　：停止モードの時は、Stop を表示します。続行モードの時は、Continue を表示します。

【使用例】　(1)実行モードを続行モードに変更します。

```
: EXEC_MODE C (RET)
:
```

(2)現在の実行モードを表示します。

```
: EXEC_MODE (RET)
Continue mode
:
```

4.19　EXTTOOL

EXTTOOL	外部ツールの操作
EX	

【形式】　EXTTOOL△<操作名>△<操作パラメータ> (RET)

【パラメータ】　・<操作名>　接続／切断／操作を指定します。

CONNECT ： 外部ツールと接続します。
TERMINATE ： 外部ツールとの接続を切断します。
STATE　：外部ツールとの接続状態を表示します。

外部ツールとして Eeaglei を指定した場合

COUPLE　： Eaglei をカップルモードに指定します。
UNCOUPLE： Eaglei をアンカップルモードに指定します。
UPDATEMAP△<マップファイル名>：Eaglei のマップを更新します。

外部ツールとして Seamless を指定した場合

CONNECT、TERMINATE 以外を指定した場合、
<操作パラメータ>を Seamless へコマンドとして送信します。

【機能】 ・接続
外部ツールへの接続します。外部ツールからシミュレータ・デバッガを起動した場合のみ接続が可能です。

・切断
外部ツールとの接続を切断します。一度切断して再接続したい場合、シミュレータ・デバッガを再起動してください。

・Eaglei の操作
Eaglei に対し、カップリングモードの指定、マップの更新操作を行なうことができます。

・Seamless の操作
Seamless に対してコマンドを送信することができます。

【使用例】 (1)Seamless に接続します。
:exttool connect (RET)
:
(2)Seamless との接続を切断します。
:exttool terminate (RET)
:
(3)Seamless に break コマンドを送信します。
:exttool break（RET）
:

4.20 FILE_LOAD

FILE_LOAD FL	ファイルのロード

【形式】 FILE_LOAD△<ファイル名>[△{ ELF｜STY}](RET)

【パラメータ】 ・<ファイル名> ロードするファイル名を指定します。
ファイル拡張子を省略した場合、ファイル形式の指定があれば下記のように拡張子を付加します。
ファイル形式の指定がなければ".abs"をファイル拡張子として付加します。
ELF ： .abs
STY ： .mot

・ファイル形式の指定 { ELF｜STY}
ELF(ELF)：ELF ファイルをロードします。
STY(STYPE)：STYPE ファイル（モトローラ S レコードタイプのみ）をロードします。

【機能】 デバッグ対象プログラムをロードします。
ELF の場合は、ロードに必要なメモリは FILE_LOAD コマンドが確保しますが、STYPE の場合は確保しません。

【説明】　デバッグ対象プログラムをロードする前にシミュレータ・デバッガのリセットを行います。（ELF のみ）
デバッグ対象プログラムをロードした後の初期設定を次に示します。
メモリ領域‥‥デバッグ対象プログラム領域を確保します。（ELF のみ）
PC‥‥‥‥‥‥‥‥‥‥STYPE の場合
デバッグ対象プログラム内にエントリアドレスを設定している場合、
そのアドレスを設定します。エントリ指定がないときはロードモジュールの
先頭アドレスを設定します。
ELF の場合
最初に現れたセクションの先頭アドレスを設定します。
SP‥‥‥‥‥‥‥‥‥‥‥‥内蔵 RAM 空間の最終アドレス+1 を設定します。
内蔵 RAM 空間がなかった場合は、０を設定します。
上記以外のレジスタ、フラグは、設定しません。

【使用例】　(1)ELF タイプのデバッグ対象プログラム test1.abs をロードします。

```
: FILE_LOAD test1.abs(RET)
:
```

(2)STYPE ファイル test2.mot をロードします。

```
: FILE_LOAD test2.mot STY(RET)
:
```

4.21　FILE_SAVE

FILE_SAVE FS	ファイルへの保存

【形式】　FILE_SAVE△<ファイル名>△<開始位置>△<終了位置>(RET)

【パラメータ】
・<ファイル名>　保存するファイル名を指定します。
・<開始位置>　保存するメモリの開始位置を指定します。
・<終了位置>　保存するメモリの終了位置を指定します。

【機能】　メモリエリアをファイルへ保存します。 データはモトローラ S レコードフォーマットで保存されます。
すでに存在するファイル名を指定した場合は上書きします。
ファイル名に拡張子が指定されていない場合は .mot を付加します。

【使用例】　H'2000 番地から H'2FFF 番地までのメモリデータを sample.mot というファイルに保存します。

```
: FILE_SAVE sample.mot 2000 2FFF(RET)
:
```

4.22 GO

GO G	命令の連続実行

【形式】 GO△[開始位置>][;D](RET)

【パラメータ】
- ・<開始位置> プログラムの実行を開始する位置を指定します。
  省略時は現在のプログラムカウンタが指しているアドレスから実行を開始します。
- ・ブレーク無効 ;D (Disable)
  ブレーク系コマンドで設定されているブレークポイントを一時的に無効にします。

【機能】 開始位置からプログラムを連続実行します。
ブレーク無効の指定は一時的なものであり、実行停止と同時に解除されます。
実行中断時に、命令実行数（１０進数）、実行中断時のレジスタ、最後に実行した命令の逆アセンブル表示、および実行中断インフォメーションメッセージを表示します。
開始位置を指定すると、実行開始時にパイプラインがリセットされます。

【使用例】 H'1000 番地から H'101E 番地までデバッグ対象プログラムを連続実行します。（SH-1 の場合）

```
: GO 1000(RET)
Exec Instructions = 30
PC=00001020 SR=00000000:-MRB----------IIII---- SP=05000000
GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
R0-7    00000000 0000FFFF 00000000 00000000 01000000 00000000 00000000 00000000
R8-15 00000000 00000000 00000010 00000000 0000FFFF 00000000 00000000 05000000
0000101E    MOV         #H'00,R3
+++5001 : PC breakpoint
:
```

4.23 GO_RANGE

GO_RANGE GR	命令の連続実行（範囲指定）

【形式】 GO_RANGE△<開始位置>△<ブレーク位置>[;D](RET)

【パラメータ】
- ・<開始位置> プログラムの実行を開始する位置を指定します。
- ・<ブレーク位置> プログラムの実行を停止する位置を指定します。
- ・ブレーク無効 ;D (Disable)
  ブレーク系コマンドで設定されているブレークポイントを一時的に無効にします。

【機能】　開始位置からブレーク位置までプログラムを連続実行します。
ブレーク位置の命令は実行しないで停止します。
ブレーク無効の指定は一時的なものであり、実行停止と同時に解除されます。
実行中断時に、命令実行数（１０進数）、実行中断時のレジスタ、最後に実行した命令の逆アセンブル表示、および実行中断インフォメーションメッセージを表示します。
開始位置を指定すると、実行開始時にパイプラインがリセットされます。

【留意事項】　命令の先頭位置以外にブレーク位置を設定するとブレーク成立を検出できません。

【使用例】　H'1000 番地から H'1020 番地までデバッグ対象プログラムを連続実行します。（SH-3 の場合）
（H'1020 番地の命令は実行しないで停止します）

```
: GO_RANGE 1000 1020(RET)
Exec Instructions = 30
PC=00001020 SR=700000F0:-MRB----------1111---- SP=00001FEC
GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
R0-7    00000000 00000000 00000003 00000002 000001E4 00000000 00000000 00000000
R8-15 00000000 00000000 00000000 00000000 00000000 00000001 000001EC 00001FEC
R0_BANK-R3_BANK 00000000 00000000 00000000 00000000
R4_BANK-R7_BANK 00000000 00000000 00000000 00000000
SSR=00000000 SPC=00000000
PTEH=00000000 PTEL=00000000 TTB=00000000 TEA=00000000 MMUCR=00000000
EXPEVT=00000000 INTEVT=00000000 TRA=00000000 CCR=00000000
0000101E    MOV          #H'00,R3
+++5001 : PC breakpoint
:
```

4.24　GO_RESET

GO_RESET GS	ベクタアドレスからの実行

【形式】　GO_RESET(RET)

【機能】　リセットベクタで指定されているアドレスから始まるデバッグプラットフォームプログラムを実行します。
実行中断時に、命令実行数（10 進数）、実行中断時のレジスタ、最後に実行した命令の逆アセンブル表示、および実行中断インフォメーションメッセージを表示します。

【説明】　SH-1/SH-2/SH-2E/SH-DSP シリーズの場合：
実行前には、あらかじめリセット例外処理ベクタテーブルがメモリ上に設定されている必要があります。
リセット例外処理のテーブルには、PC と SP の初期値を格納しておきます。

ベクタテーブル

格納する項目レジスタ	ベクタ番号	格納するベクタテーブルアドレス
PC	0	H'00000000～H'00000003
SP	1	H'00000004～H'00000007

SH-3/SH-3E/SH-3DSP/SH-4 シリーズの場合：
リセットベクタアドレスは H'A0000000 に固定されています。

【使用例】　ユーザプログラムをリセットベクタから実行を開始します。（SH-3 の場合）

```
: GO_RESET
Exec Instructions = 12
PC=A0000018 SR=700000F0:-MRB----------1111---- SP=7F001000
GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
R0-7    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
R8-15 00000000 00000000 00000000 00000000 00000000 00000000 00000000 7F001000
R0_BANK-R3_BANK 00000000 00000000 00000000 00000000
R4_BANK-R7_BANK 00000000 00000000 00000000 00000000
SSR=700000F0 SPC=00000000
PTEH=00000000 PTEL=00000000 TTB=00000000 TEA=00000000 MMUCR=00000000
EXPEVT=00000000 INTEVT=00000000 TRA=00000000 CCR=00000000
A0000016        NOP
+++ 5001 : PC breakpoint
:
```

4.25　GO_TILL

GO_TILL GT	命令の連続実行（停止位置指定）

【形式】　GO_TILL△<ブレーク位置 1>[△<ブレーク位置 2>△<ブレーク位置 3>･･･△<ブレーク位置 10>][;D](RET)

【パラメータ】
・<ブレーク位置 n>　プログラムの実行を停止する位置を指定します。（最大 10 カ所）
・ブレーク無効　;D (Disable)
ブレーク系コマンドで設定されているブレークポイントを一時的に無効にします。

【機能】　現在のプログラムカウンタが指しているアドレスからブレーク位置までを連続実行します。
ブレーク位置の命令は実行しないで停止します。
ブレーク位置は 10 個所まで設定できます。
ブレーク無効の指定は一時的なものであり、実行停止と同時に解除されます。
実行中断時に、命令実行数（10 進数）、実行中断時のレジスタ、最後に実行した命令の
逆アセンブル表示、および実行中断インフォメーションメッセージを表示します。

【留意事項】　命令の先頭位置以外にブレーク位置を設定するとブレーク成立を検出できません。

【使用例】　現在のプログラムカウンタが指しているアドレスから、H'1000、H'1010 または H'1020 番地までデバッグ対対象プログラムを連続実行します。（SH-1 の場合）

```
: GO_TILL 1000 1010 1020(RET)
Exec Instructions = 30
PC=00001020 SR=00000000:-MRB----------IIII---- SP=05000000
GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
R0-7    00000000 0000FFFF 00000000 00000000 01000000 00000000 00000000 00000000
R8-15 00000000 00000000 00000010 00000000 0000FFFF 00000000 00000000 05000000
0000101E    MOV          #H'00,R3
  +++5001 : PC breakpoint
:
```

4.26　HELP

HELP HE	コマンド名、コマンド入力形式の表示

【形式】　HELP[△<コマンド名>](RET)

【パラメータ】　・<コマンド名>　　ヘルプメッセージを表示させたいコマンド名を指定します。

【機能】　指定されたコマンドのヘルプメッセージを表示します。
コマンド名を省略した場合は、コマンド一覧を表示します。

【使用例】 (1)コマンドの一覧を表示します。

```
: HELP(RET)
 .<register>                    ;
 ASsemble                        Break_Clear
 Break_Enable                   BreakAccess
 BreakAccess_Display            BreakData
 BreakData_Display              BreakPoint
 BreakPoint_Display             BreakRegister
        :                              :
 Trace                           Trace_cLear
 Trace_Condition                 Trap_Address
 Trap_address_Display            Trap_address_Enable
 :
```

(2)HELP コマンドのコマンド入力形式を表示します。

```
:HELP HELP(RET)
 HE|HELP[ <command name> ]
:
```

4.27 LOAD_STATUS

LOAD_STATUS	シミュレータ・デバッガの状態回復
LS	

【形式】 LOAD_STATUS[△<ファイル名>](RET)

【パラメータ】 ・<ファイル名> シミュレータ･デバッガの動作状態を保存してあるファイルのファイル名を指定します。
ファイル名を省略した場合は、"sdsh.sav"をファイル名として使用します。
ファイル拡張子を省略した場合は、".sav"をファイル形式として付加します。

【機能】 レジスタ内容等の状態を SAVE_STATUS コマンド実行時の状態に戻します。
SAVE_STATUS コマンド実行時にロードしていたロードモジュールをロードし直します。

【使用例】 ファイル test1.sav に保存された状態をロードします。

```
: LOAD_STATUS test1.sav(RET)
:
```

4.28 LOG

LOG LO	実行履歴ファイルの作成開始

【形式】　LOG△<ファイル名>[△A](RET)

【パラメータ】　・<ファイル名>　実行履歴を出力するファイル名を指定します。
・アペンドモードの指定　A(Append) ：指定したファイルに実行履歴を追加します。
本オプションを省略した場合は、指定した
ファイルの先頭から実行履歴を格納します。

【機能】　実行履歴ファイルへの出力を開始します。
もし、指定したファイルが存在する場合は、そのファイルを削除し、改めてファイルを作成します。
開始後停止せずに再度開始が指定された場合は、出力中のファイルを閉じてから、あらためて指定されたファイルへ出力を開始します。

【留意事項】　システムコールの入出力処理でエラーが発生した場合、入出力データは出力ファイルに書き込みません。

【使用例】　(1)コマンド入力および表示データを書き込む sample.log ファイルを指定し、ファイルへの書き込みを開始します。

```
: LOG sample.log(RET)
:
```

(2)sample.log に実行履歴を追加します。

```
: LOG sample.log A(RET)
:
```

4.29 LOG_ENABLE

LOG_ENABLE LE	実行履歴ファイルの作成の有効／無効

【形式】　LOG_ENABLE△{E｜D}(RET)

【パラメータ】　・ファイル出力の一時停止と再開　{E｜D}
E(Enable) ：ファイルへの出力を再開します。
D(Disable)：ファイルへの出力を一時停止します。

【機能】　Ｄオプション(Disable)を指定するとファイルの出力は一時停止され、Ｅオプション(Enable)を指定すると、ファイルへの出力を再開します。

【使用例】　(1)ファイルへの書き込みを一時停止します。
: LOG_ENABLE D(RET)
:

(2)ファイルへの書き込みを再開します。
: LOG_ENABLE E(RET)
:

4.30 LOG_STOP

LOG_STOP	実行履歴ファイルの作成終了
LT	

【形式】　LOG_STOP(RET)

【機能】　実行履歴ファイルへの出力を終了します。

【留意事項】　システムコールの入出力処理でエラーが発生した場合、入出力データは出力ファイルに書き込みません。

【使用例】　ファイルへの書き込みを終了します。

: LOG_STOP(RET)
:

4.31 MAP_CLEAR

MAP_CLEAR	メモリ領域の解除
MC	

【形式】　MAP_CLEAR△<開始位置>△<終了位置>(RET)

【パラメータ】　・<開始位置>　メモリ領域の先頭位置を指定します。
・<終了位置>　メモリ領域の終了位置を指定します。

【機能】　MAP_SET コマンドで確保したメモリ領域を解除します。

【使用例】　既に設定済みの H'3000 番地から H'301F 番地を解除します。

: MAP_CLEAR 3000 301F(RET)
:

4.32 MAP_DISPLAY

MAP_DISPLAY MI	メモリ領域の表示

【形式】　MAP_DISPLAY[△M](RET)

【パラメータ】　・メモリマップ情報表示　M(Map) ： CPU 情報ファイル内のメモリマップ情報の表示を指定します。

【機能】　メモリマップの設定情報を以下のフォーマットで表示します。

(1)メモリマップ情報表示

<START>　： 開始位置
<END>　： 終了位置
<ATTR>　：アクセス種別(R-Read、W-Write、RW-Read/Write)
<SECT_NAME>：セクション名

(2)CPU 情報ファイル内のメモリマップ情報表示

<KIND> ：メモリ種別（I/O - 内蔵 I/O、RAM または INTRAM - 内蔵 RAM、ROM または INTROM - 内蔵 ROM、XRAM/XROM/YRAM/YROM – XY メモリ、EXT - 外部バス空間）
<START>：開始位置
<END> ：終了位置
<STATE>：メモリのアクセスステート数（SH-4 では- - -を表示します）
<BUS> ：メモリのデータバス幅

【使用例】　(1)現在のメモリ領域の状態を表示します

```
: MAP_DISPLAY (RET)
<START>   <END>       <ATTR>   <SECT_NAME>
00000000-000003FF       W
00002000-000020EF      RW      SECT1
00003000-0000301F       W
:
```

(2)CPU 情報のメモリマップを表示します。

```
: MAP_DISPLAY M (RET)
<KIND> <START>   <END>          <STATE>          <BUS>
 EXT     00000000-7EFFFFFF      00000001         00000032
 RAM     7F000000-7F000FFF      00000001         00000032
 EXT     7F001000-DFFFFFFF      00000001         00000032
 I/O     E0000000-FFFFFFFF      00000001         00000032
:
```

4.33 MAP_SET

MAP_SET MS	メモリ領域の設定

【形式】　MAP_SET△<開始位置>[△<終了位置>][△{R｜W｜RW}](RET)

【パラメータ】
- ・<開始位置>　メモリ領域の先頭位置を指定します。
- ・<終了位置>　メモリ領域の終了位置を指定します。
  省略すると開始位置と同じになります。
- ・アクセス種別　{R｜W｜RW}
  - R(Read)　：読み出しのみ可能に設定します。
  - W(write)　：書き込みのみ可能に設定します。
  - RW(Read/write)：読み書きを可能に設定します。（デフォルト）

【機能】　デバッグ対象プログラムで使用するメモリ領域の設定を行います。

【留意事項】
(1)MAP コマンドで設定しようとした範囲が既に確保されていた場合でも、新たに指定した内容で設定し直します。
(2)複数の領域にまたがって再設定および解除が可能です。

【使用例】
(1)H'3000 番地から H'301F 番地を読み出しのみ可能なメモリ領域として確保します。

```
: MAP_SET 3000 301F R(RET)
:
```

(2)H'0 番地から H'03FF 番地に確保されているメモリ領域のアクセス種別を書き込みのみ可能に変更します。

```
: MAP_SET 0 3FF W(RET)
:
```

4.34 MEMORY_DISPLAY

MEMORY_DISPLAY MD	メモリ内容の表示

【形式】　MEMORY_DISPLAY△<開始位置> [△<長さ>] [;<サイズ>](RET)

【パラメータ】
- ・<開始位置>　メモリ内容表示の開始位置を指定します。
- ・<長さ>　表示するデータの長さ（バイト数）を指定します。（デフォルト=H'100、最大=H'4000）
- ・<サイズ>　データのサイズ　{B｜W｜L｜D｜S｜A}
  - B(Byte)　：バイトデータ（デフォルト）
  - W(Word)　：ワードデータ
  - L(Long)　：ロングワードデータ
  - D(Double Float)：倍精度浮動小数点データ
  - S(Single Float)：単精度浮動小数点データ
  - A(ASCII)　：ASCII データ

【機能】　メモリ内容を表示します。

【留意事項】　ワードサイズのデータを表示する場合、開始位置がワード境界（2 の倍数）になるようにしてください。
単精度、倍精度浮動小数点またはロングワードサイズのデータを表示する場合、開始位置がロングワード境界（４の倍数）になるようにしてください。

【使用例】　(1)H'1000 番地からバイト単位で表示します。

```
: MEMORY_DISPLAY 1000;B(RET)
address   +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
00001000 4F 22 7F C8 E3 00 1F 32 A0 12 00 09 D1 1E 41 0B
00001010 00 09 1F 03 40 11 89 01 60 0B 1F 03 53 F2 43 08
      :                                   :
000010E0 A0 08 00 09 63 F2 52 F2 32 38 1F 22 53 F1 51 F2
000010F0 31 33 89 D1 63 F2 52 F3 32 3C 1F 23 E3 0A 51 F3
:
```

(2)H'1000 番地から 16 バイト分をワード単位で表示します。

```
: MEMORY_DISPLAY 1000 10;W(RET)
address   +0    +2    +4    +6    +8    +A    +C    +E
00001000 4F22 7FC8 E300 1F32 A012 0009 D11E 410B
:
```

(3)H'2000 番地から 8 バイト分の倍精度浮動小数点データを表示します。

```
: MEMORY_DISPLAY 2000 8;D(RET)
address   +0 +1 +2 +3 +4 +5 +6 +7
00002000 32 1F 00 E3 C8 7F 22 4F 2.87495706857453e-67
:
```

(4)H'3000 番地から 22 バイトの ASCII データを表示します。

```
: MEMORY_DISPLAY 3000 16;A(RET)
address           ASCII
00003000 O".....2......A.
00003010 ....@.
:
```

4.35　MEMORY_EDIT

MEMORY_EDIT	メモリ内容の変更
ME	

【形式】　変更　：　MEMORY_EDIT△<開始位置>△<データ>[;<サイズ>](RET)
対話形式：　MEMORY_EDIT△<開始位置> [;<サイズ>](RET)

【パラメータ】
・<開始位置>　メモリ内容変更の開始位置を指定します。
・<データ>　変更内容を指定します。

・<サイズ>　　データのサイズ　{B｜W｜L｜D｜S｜A}
B(Byte)　　：バイト単位で変更します。（デフォルト）
W(Word)　　：ワード単位で変更します。
L(Long)　　：ロングワード単位で変更します。
D(Double Float)：倍精度浮動小数点単位で変更します。
S(Single Float)：単精度浮動小数点単位で変更します。
A(ASCII)：ASCII 文字列単位で変更します。

【機能】　メモリ内容を任意の値に変更します。

【説明】　対話形式のコマンドが入力されると、指定されたアドレスの内容を表示した後、対話形式となります。

MEMORY_EDIT△<開始位置>(RET)
address data : [{<データ>｜^}](RET)
address data : [{<データ>｜^}](RET)
・
・
address data : .(RET)

address data　　：変更前のデータを示します。
<データ>　　：変更データを指定します。
^　　：直前のアドレスの内容を表示します。
(RET)のみ入力　　：次のアドレスの内容を表示します。
.(ピリオド)　　：MEMORY_EDIT コマンドを終了します。

【留意事項】　ワードサイズのデータを変更する場合、開始位置がワード境界（2 の倍数）になるようにしてください。
単精度、倍精度浮動小数点またはロングワードサイズのデータを変更する場合、開始位置がロングワード境界（4 の倍数）になるようにしてください。

【使用例】　(1)H'1000 番地の 1 バイトのメモリ内容を H'3E に変更します。

```
: MEMORY_EDIT 1000 3E;B(RET)
:
```

(2)H'1000 番地から対話形式で、1 バイトごとにメモリ内容を変更します。

```
: MEMORY_EDIT 1000;B(RET)
00001000 3E : 5F(RET)
00001001 FF : (RET)
00001002 55 : 25(RET)
    :    :
00001005 CC : .(RET)
:
```

(3)H'2000 番地から対話形式で、単精度浮動小数点単位毎にメモリ内容を変更します。

```
: MEMORY_EDIT 2000;S(RET)
00002000 1.413991E-3 : F'-3.1415922E+1(RET)
00002004 1.234567E+5 : .(RET)
:
```

4.36 MEMORY_FILL

MEMORY_FILL MF	メモリ領域の初期設定

【形式】　MEMORY_FILL△<開始位置>△<終了位置>△<データ値>[;<サイズ>] [△<ベリファイフラグ>](RET)

【パラメータ】
- ・<開始位置>　初期設定するメモリの開始位置を指定します。
- ・<終了位置>　初期設定するメモリの終了位置を指定します。
- ・<データ値>　設定するデータを指定します。
- ・<サイズ>　データのサイズ　{B｜W｜L｜D｜S}
  - B(Byte)　：バイトデータ（デフォルト）
  - W(Word)　：ワードデータ
  - L(Long)　：ロングワードデータ
  - D(Double Float)：倍精度浮動小数点データ
  - S(Single Float)：単精度浮動小数点データ
- ・<ベリファイフラグ>　データ設定後のベリファイの有無　{V｜N}
  - V　：ベリファイ有り（デフォルト）
  - N　：ベリファイ無し

【機能】　指定した位置の範囲に初期化データを設定します。

【留意事項】　ワードサイズのデータを設定する場合、開始位置がワード境界（2 の倍数）になるようにしてください。
単精度、倍精度浮動小数点またはロングワードサイズのデータを設定する場合、開始位置がロングワード境界（４の倍数）になるようにしてください。

【使用例】　(1)H'1000 番地から H'1FFF 番地まで 0 クリア後、ベリファイを行います。

```
: MEMORY_FILL 1000 1FFF 0(RET)
:
```

(2)H'2000 番地から H'2FFF 番地までワードで H'FF00 を設定します。ベリファイは行いません。

```
: MEMORY_FILL 2000 2FFF FF00;W N(RET)
:
```

4.37　MEMORY_MOVE

MEMORY_MOVE	メモリブロックの転送
MV	

【形式】　MEMORY_MOVE△<開始位置>△<終了位置>△<転送先位置>(RET)

【パラメータ】
- <開始位置>　転送元の開始位置を指定します。
- <終了位置>　転送元の終了位置を指定します。
- <転送先位置>　転送先の開始位置を指定します。

【機能】　指定範囲のメモリデータを指定された転送先にコピーします。
転送先の領域はあらがじめ MAP_SET コマンドで確保しておいてください。

【使用例】　H'1000 番地から H'14FF 番地までの内容を H'2000 番地以降に順次コピーします。

```
: MEMORY_MOVE 1000 14FF 2000 (RET)
:
```

4.38　PERFORMANCE_ANALYSIS

PERFORMANCE_ANALYSIS	実行効率測定の設定
PA	

【形式】　PERFORMANCE_ANALYSIS[△<開始位置>](RET)

【パラメータ】
- <開始位置>　実行効率測定を行う関数の開始アドレスを指定します。
開始位置を省略すると全関数（実際に実行した関数のみ）の実行効率測定を行います。

【機能】　指定した関数の最大実行サイクル数、最小実行サイクル数、合計実行サイクル数、コール回数を測定します。

【留意事項】　測定関数の途中で.<register>コマンド等で PC 値が変更(パイプラインリセット)された場合、測定値は不正となる場合があります。

【使用例】　全関数の実行効率測定を指定します。

```
: PERFORMANCE_ANALYSIS(RET)
:
```

4.39 PERFORMANCE_ANALYSIS_CLEAR

PERFORMANCE_ANALYSIS_CLEAR	実行効率測定の解除
PC	

【形式】　PERFORMANCE_ANALYSIS_CLEAR[△<インデックス>](RET)

【パラメータ】　・<インデックス>　解除する関数の番号を指定します。（インデックスは実行効率の表示で確認します。）
インデックスを省略すると全ての測定結果を削除します。

【機能】　実行効率測定の解除を行います。

【使用例】　インデックス No.1 の測定結果を削除します。

```
:PERFORMANCE_ANALYSIS_CLEAR 1(RET)
:
```

4.40 PERFORMANCE_ANALYSIS_DISPLAY

PERFORMANCE_ANALYSIS_DISPLAY	実行効率測定の表示
PD	

【形式】　PERFORMANCE_ANALYSIS_DISPLAY[△{A｜C}](RET)

【パラメータ】　・表示種別　{A｜C}
A(Address)：アドレス順（昇順）に表示します。（デフォルト）
C(Cycle)：サイクル数順（降順）で表示します。

【機能】　実行効率測定結果を以下のフォーマットで表示します。

INDEX	： 関数の登録番号
ADDRESS	： 関数の開始アドレス
MAXCYCLE	： 関数の最大実行サイクル数
MINCYCLE	： 関数の最小実行サイクル数
TOTALCYCLE	： 関数の合計実行サイクル数
COUNT	： 関数のコール回数
%	： 関数の合計実行サイクル数が、デバッグ対象プログラム全体の実行サイクル数に占める割合
HISTOGRAM	： 上記割合を棒グラフで表示

【説明】　(1)本機能は、指定関数ごとの実行サイクル数を表示します。
(2) 実行サイクル数は、指定関数コール命令実行時の累計実行サイクル数と指定関数からのリターン命令実行時の累計実行サイクル数の差から求めています。
(3)関数は9999個まで設定可能です。

【使用例】　(1)アドレス順に測定結果を表示します。

```
: PERFORMANCE_ANALYSIS_DISPLAY(RET)
INDEX ADDRESS    MAXCYCLE    MINCYCLE    TOTALCYCLE COUNT %
HISTOGRAM
    0   00001234        20000        10000          50000        3     45 ####
    1   00005678        15000         5000         400000        9     55 #####
:
```

(2)サイクル数順に測定結果を表示します。

```
: PERFORMANCE_ANALYSIS_DISPLAY C(RET)
INDEX ADDRESS    MAXCYCLE    MINCYCLE    TOTALCYCLE COUNT %
HISTOGRAM
    1   00005678        15000         5000         400000        9     55 #####
    0   00001234        20000        10000          50000        3     45 ####
:
```

4.41 PERFORMANCE_ANALYSIS_ENABLE

PERFORMANCE_ANALYSIS_ENABLE PE	実行効率測定の有効／無効、初期化

【形式】　PERFORMANCE_ANALYSIS_ENABLE△{E｜D｜R} (RET)

【パラメータ】　・指定区分　{E｜D｜R}
E(Enable)： 測定を有効にします。
D(Disable)： 測定を無効にします。
R(Reset) ： 測定結果をリセットします。

【機能】　実行効率測定の有効／無効の設定または測定結果のリセットを行います。

【説明】　(1)シミュレータ起動時は、実行効率測定は無効になっています。
(2)初期化では測定結果のみをリセットします。
　実行効率測定の有効／無効、登録済みの開始位置（全関数を含む）は変更しません。

【使用例】　(1)実行効率測定を無効にします。

```
: PERFORMANCE_ANALYSIS_ENABLE D(RET)
:
```

(2)実行効率測定をリセットします。

```
: PERFORMANCE_ANALYSIS_ENABLE R(RET)
:
```

4.42 QUIT

QUIT Q	シミュレータ・デバッガの終了

【形式】　QUIT(RET)

【機能】　シミュレータ・デバッガを終了して OS に戻ります。
実行履歴ファイルおよびコマンドファイルがオープンされている状態の時はクローズします。

【使用例】　シミュレータ・デバッガの処理を終了します。

```
: QUIT (RET)
```

（csdsh およびシミュレータ・デバッガが終了し、OS のプロンプトが表示されます）

4.43 RADIX

RADIX RX	基数の設定

【形式】　設定：　RADIX△{B｜O｜D｜H}(RET)
　　表示：　RADIX(RET)

【パラメータ】　・基数　　{B｜O｜D｜H}
B　：　基数を２進数にします。
O　：　基数を８進数にします。
D　：　基数を１０進数にします。
H　：　基数を１６進数にします。
起動時は H になります。

【機能】　デフォルトの基数を設定、または表示します。パラメータを指定しないと基数を表示します。
数値データの前に B'/H'/D'/O' を入力した場合は、その指定がデフォルトの基数より優先されます。
表示内容
B　：　Binary
O　：　Octal
D　：　Decimal
H　：　Hexadecimal

【使用例】　(1)現在の基数を表示します。
: RADIX(RET)
Hexadecimal
:

(2)基数を 10 進数に変更します。
: RADIX D(RET)
: RADIX(RET)
Decimal
:

4.44　REGISTER

REGISTER	レジスタ内容の表示
R	

【形式】　REGISTER [△{C｜F｜A}](RET)

【パラメータ】　・表示レジスタの指定　　{C｜F｜A}
C(Cpu)：　CPU レジスタ(DSP レジスタも表示)の内容を表示します。（デフォルト）
F(Fpu)：　FPU レジスタの内容を表示します。
（SH-2E、SH-3E、SH-4 シリーズでのみ有効）
A(All)：　CPU、FPU、制御レジスタの内容を表示します。
（制御レジスタは SH-3、SH-3E、SH-DSPwithCache、SH-3DSP、SH-4 シリーズのみ有効）

【機能】　レジスタの内容を表示します。表示するレジスタは以下のとおりです。

SH-1/SH-2 シリーズの場合：

CPU レジスタ	
汎用レジスタ	R0-R15
コントロールレジスタ	SR GBR VBR
システムレジスタ	MACH MACL PR PC

SH-DSP、SH-2DSP の場合：

CPU レジスタ	
汎用レジスタ	R0-R15
コントロールレジスタ	SR RS RE GBR VBR MOD
システムレジスタ	MACH MACL PR PC
DSP レジスタ	
データレジスタ	A0 A0G A1 A1G M0 M1 X0 X1 Y0 Y1
コントロールレジスタ	DSR

SH-DSPwithCache の場合：

CPU レジスタ	
汎用レジスタ	R0-R15
コントロールレジスタ	SR RS RE GBR VBR MOD
システムレジスタ	MACH MACL PR PC
制御レジスタ	CCR
DSP レジスタ	
データレジスタ	A0 A0G A1 A1G M0 M1 X0 X1 Y0 Y1
コントロールレジスタ	DSR

SH-3DSP シリーズの場合：

CPU レジスタ	
汎用レジスタ	R0-R15 R0_BANK-R7_BANK
コントロールレジスタ	SR GBR VBR SSR SPC
システムレジスタ	MACH MACL PR PC
制御レジスタ	PTEH PTEL TTB TEA MMUCR EXPEVT INTEVT TRA CCR CCR2
DSP レジスタ	
データレジスタ	A0 A0G A1 A1G M0 M1 X0 X1 Y0 Y1
コントロールレジスタ	DSR

SH-3 シリーズの場合：

CPU レジスタ	
汎用レジスタ	R0-R15 R0_BANK-R7_BANK
コントロールレジスタ	SR GBR VBR SSR SPC
システムレジスタ	MACH MACL PR PC
制御レジスタ	PTEH PTEL TTB TEA MMUCR EXPEVT INTEVT TRA CCR

【機能】

SH-2E シリーズの場合：

CPU レジスタ	
汎用レジスタ	R0-R15
コントロールレジスタ	SR GBR VBR
システムレジスタ	MACH MACL PR PC
FPU レジスタ	
浮動小数点レジスタ	FR0-FR15
コントロールレジスタ	FPSCR
システムレジスタ	FPUL

SH-3E シリーズの場合：

CPU レジスタ	
汎用レジスタ	R0-R15 R0_BANK-R7_BANK
コントロールレジスタ	SR GBR VBR SSR SPC
システムレジスタ	MACH MACL PR PC
制御レジスタ	PTEH PTEL TTB TEA MMUCR EXPEVT INTEVT TRA CCR
FPU レジスタ	
浮動小数点レジスタ	FR0-FR15
コントロールレジスタ	FPSCR
システムレジスタ	FPUL

SH-4 シリーズの場合：

CPU レジスタ	
汎用レジスタ	R0-R15 R0_BANK-R7_BANK
コントロールレジスタ	SR GBR VBR SSR SPC SGR DBR
システムレジスタ	MACH MACL PR PC
制御レジスタ	PTEH PTEL TTB TEA MMUCR EXPEVT INTEVT TRA CCR QACR0 QACR1
FPU レジスタ	
浮動小数点レジスタ	FR0-FR15 XF0-XF15 DR0 DR2 DR4 DR6 DR8 DR10 DR12 DR14 XD0 XD2 XD4 XD6 XD8 XD10 XD12 XD14
コントロールレジスタ	FPSCR
システムレジスタ	FPUL

【使用例】　(1)全レジスタの内容を表示します。（SH-DSP の場合）

```
: REGISTER A(RET)
PC=00000000 SR=000000F0:----000000000000--------111100-- SP=00000000
GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
R0-7    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
R8-15 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
RS=00000000 RE=00000000 MOD=00000000
DSR=00000000:----------------------------COB-
A0G=00 A0=00000000 M0=00000000 X0=00000000 Y0=00000000
A1G=00 A1=00000000 M1=00000000 X1=00000000 Y1=00000000
:
```

(2)CPU レジスタの内容を表示します。（SH-3 シリーズの場合）

```
: REGISTER C(RET)
PC=00000000 SR=700000F0:-MRB--------------------1111---- SP=00000000
GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
R0-7    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
R8-15 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
R0_BANK-R3_BANK 00000000 00000000 00000000 00000000
R4_BANK-R7_BANK 00000000 00000000 00000000 00000000
SSR=00000000 SPC=00000000
:
```

(3)FPU レジスタの内容を表示します。（SH-2E シリーズの場合）

```
: REGISTER F(RET)
PC=00000000 SR=000000F0:-------------------------1111---- SP=00000000
FPUL=00000000 FPSCR=00040001:-------------D-----------------RZ
FR0-7    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
FR8-15 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
FR0- 3     0.000000e+00     0.000000e+00     0.000000e+00     0.000000e+00
FR4- 7     0.000000e+00     0.000000e+00     0.000000e+00     0.000000e+00
FR8-11     0.000000e+00     0.000000e+00     0.000000e+00     0.000000e+00
FR12-15    0.000000e+00     0.000000e+00     0.000000e+00     0.000000e+00
:
```

4.45　RESET

RESET	シミュレータ・デバッガのリセット
RS	

【形式】　RESET(RET)

【機能】　シミュレータ･デバッガをリセットします。
本コマンドを実行するとシミュレータ･デバッガの設定は以下のように設定されます。
パイプライン　：リセットします。
レジスタ　：以下のように初期化されます
・SH-1/SH-2/SH-DSP シリーズの場合
SR　：　H'F0
上記以外：　H'0
・SH-3,SH-3DSP の場合
SR　：　H'700000F0
上記以外：　H'0
・SH-2E の場合
SR　：　H'F0
FPSCR　：　H'40001
上記以外　：　H'0
・SH-3E/SH-4 シリーズの場合
SR　：　H'700000F0
FPSCR　：　H'40001
上記以外　：　H'0
メモリ　：全てのメモリ設定を解除します。
デバッグ対象プログラム：デバッグ対象プログラムに関する情報を全て削除して、
プログラム未ロードの状態に設定します。
コマンド　：RADIX コマンドを除く全てのコマンドの設定を解除して、
初期状態に戻します。

【使用例】　シミュレータ・デバッガをリセットします。

```
: RESET (RET)
:
```

4.46　ROUND_MODE

ROUND_MODE RM	浮動小数点の丸めモード設定、表示

【形式】　設定　：　ROUND__MODE△｛Z｜N｝(RET)
表示　：　ROUND__MODE(RET)

【パラメータ】　・丸めモード種別　｛Z｜N｝
Z（Round to zero）　：ゼロ方向への丸め（デフォルト）
N（Round to nearest）　：近傍への丸め

【機能】　浮動小数点の丸めモードを設定します。

【使用例】　(1)現在の丸めモードを表示します。
: ROUND_MODE(RET)
ROUND TO ZERO
:

(2)丸めモードを近傍への丸めに変更します。
: ROUND_MODE N (RET)
:

4.47 SAVE_STATUS

SAVE_STATUS	現在のシミュレータ・デバッガの状態保存
SS	

【形式】　SAVE_STATUS[△<ファイル名>](RET)

【パラメータ】　・<ファイル名>　シミュレーション状態を保存するファイル名を指定します。
ファイル名を省略した場合は、"sdsh.sav"をファイル名として使用します。
ファイル形式を省略した場合は、".sav"をファイル形式として付加します。

【機能】　現在のシミュレーション状態を保存します。以下の項目が保存されます。
・最後にロードされたプログラムファイルのファイル名
・ＣＰＵ情報
・デフォルトの基数
・メモリマップ設定情報
・トレース条件
・ブレーク情報
・レジスタ情報

【使用例】　シミュレーション状態を test1.sav というファイルに保存します。

: SAVE_STATUS test1.sav(RET)
:

4.48 STACK_ANALYSIS

STACK_ANALYSIS	スタック使用量測定の有効／無効、初期化
SA	

【形式】　STACK_ANALYSIS△{E｜D｜R}(RET)

【パラメータ】　・有効／無効／初期化　{E｜D｜R}
E(Enable)：スタック使用量測定を有効にします。
D(Disable)：スタック使用量測定を無効にします。
R(Reset)：スタック使用量測定結果を初期化します。

【機能】　スタック使用量測定の有効／無効の設定または初期化を行います。

【使用例】　(1)スタック使用量の測定を有効にします。

```
: STACK_ANALYSIS E(RET)
:
```

(2)スタック使用量の測定結果を初期化します。

```
: STACK_ANALYSIS R(RET)
:
```

(3) スタック使用量の測定を無効にします。

```
: STACK_ANALYSIS D(RET)
:
```

4.49 STACK_ANALYSIS_DISPLAY

STACK_ANALYSIS_DISPLAY SD	スタック使用量測定の表示

【形式】　STACK_ANALYSIS_DISPLAY[△PC][△M](RET)

【パラメータ】
- ・プログラムカウンタ出力　PC
  プログラムカウンタを表示します。
  （省略時は、表示しません。）
- ・スタックポインタ最大／最小値出力　M
  スタックポインタの最大値、最小値を表示します。
  （省略時は、表示しません。）

【機能】　スタック使用量測定結果としてスタックが変化した時のサイクル数とスタックの値を表示します。
また、PC を指定した時は、スタックが変わった時のプログラムカウンタの値を表示します。
M が指定された時は、スタックの最大値と最小値を表示します。

【説明】 (1)本機能は、スタック使用量の測定を行います。
(2)プログラム実行中の SP 値の変化を保存します。
バッファはリング状になっており、9999 データ格納できます。
10000 データ以上格納した場合は、バッファの先頭からオーバライトします。
(3)表示する項目は以下の通りです。

```
CYCLE          SP           PC
XXXXXXXXXX XXXXXXXX XXXXXXXX
     :          :          :
XXXXXXXXXX XXXXXXXX XXXXXXXX
    CYCLE          SP           PC
Max XXXXXXXXXX XXXXXXXX XXXXXXXX
Min XXXXXXXXXX XXXXXXXX XXXXXXXX
```

[CYCLE]サイクル数
[SP]スタックポインタ値
[PC]プログラムカウンタ値
[Max]スタックポインタが最大の時の値を表示します。
[Min]スタックポインタが最小の時の値を表示します。

(4)シミュレータ起動時は、スタック使用量測定は無効になっています。
(5)初期化では測定結果のみをリセットします。スタック使用量測定の有効／無効は変更しません。

【使用例】 (1)スタック使用量測定結果を表示します。

```
: STACK_ANALYSIS_DISPLAY(RET)
CYCLE          SP
          333   00000FF0
        10000   00000900
       999998   00000FFC
:
```

(2)スタック使用量測定結果をプログラムカウンタ、スタックポインタの最大値、最小値を含めて表示します。

```
: STACK_ANALYSIS_DISPLAY PC M(RET)
CYCLE          SP           PC
          333   00000FF0   000000F0
        10000   00000090   00000F00
9999999999   00000FFC   00000FF0
    CYCLE          SP           PC
Max 9999999999   00000FFC   00000FF0
Min         10000   00000090   00000F00
:
```

4.50 STATUS

STATUS ST	シミュレータ・デバッガの状態表示

【形式】 STATUS(RET)

【機能】 シミュレータ・デバッガ起動時の CPU 種別、メモリのエンディアン、および実行停止時の実行サイクル数、キャッシュヒット率を表示します。

・ SH-1,SH-2,SH-2E,SH-DSP,SH-2DSP の場合

```
CPU=xxx      ENDIAN=xxxx     CYCLE=xxxxx
       (1)             (2)             (3)
TRACE BUFFER SIZE=xxxxxrecords
                        (4)
```

・ SH-3/SH-3E,SH-DSPwithCache,SH-3DSP の場合

```
CPU=xxx      ENDIAN=xxxx     CYCLE=xxxxx
       (1)             (2)             (3)
CACHE HIT=xx%   (Hit Counts = xxxxxxxxxx,Miss Counts = xxxxxxxxxx)
             (6)                (7)                (8)
TRACE BUFFER SIZE=xxxxxrecords     CACHE SIZE=xKbyte
                        (4)                (5)
```

・ SH-4 シリーズの場合

```
CPU=xxx      ENDIAN=xxxx     CYCLE=xxxxx
       (1)             (2)             (3)
INSTRUCTION CACHE HIT=xx%   (Hit Counts = xxxxxxxxxx,Miss Counts = xxxxxxxxxx)
                        (9)             (7)             (8)
OPERAND CACHE HIT=xx%       (Hit Counts = xxxxxxxxxx,Miss Counts = xxxxxxxxxx)
                        (10)            (7)             (8)
TRACE BUFFER SIZE=xxxxxrecords
                        (4)
```

(1) CPU 種別
(2) エンディアン種別
(3) 実行サイクル数（10 進 10 桁）
(4) トレース容量（10 進５桁）
(5) キャッシュサイズ（10 進１桁：SH-3/SH-3E の時のみ表示）
(6) キャッシュヒット率（パーセント）
(7) キャッシュヒット回数（10 進 10 桁）
(8) キャッシュミスヒット回数（10 進 10 桁）
(9) 命令キャッシュヒット率（パーセント）
(10)オペランドキャッシュヒット率（パーセント）

【使用例】 (1) CPU 種別 SH-1、BIG エンディアンで起動した場合のシミュレータ・デバッガの状態を表示します。

```
: STATUS (RET)
CPU=SH1          ENDIAN=BIG          CYCLES=128
TRACE BUFFER SIZE=1024records
:
```

(2) CPU 種別 SH-3E、BIG エンディアンで起動した場合のシミュレータ・デバッガの状態を表示します。

```
: STATUS (RET)
CPU=SH3E         ENDIAN=BIG          CYCLES=24186
CACHE HIT= 89% (Hit Counts =          7854,Miss Counts =          970)
TRACE BUFFER SIZE=4096records     CACHE SIZE=8Kbyte
:
```

(3) CPU 種別 SH-4、LITTLE エンディアンで起動した場合のシミュレータ・デバッガの状態を表示します。

```
: STATUS (RET)
CPU=SH4          ENDIAN=LITTLE     CYCLES=157353
INSTRUCTION CACHE HIT= 89% (Hit Counts =          47105,Miss Counts =          52926)
OPERAND CACHE HIT= 34%         (Hit Counts =          2289,Miss Counts =          6732)
TRACE BUFFER SIZE=32768records
:
```

4.51 STEP

STEP S	サブルーチンを１ステップとしてステップ実行

【形式】 STEP[△<ステップ数>][△R](RET)

【パラメータ】 ・<ステップ数> 命令実行ステップ数を指定します（H'1～H'FFFF）
省略時は、１ステップです。
・レジスタ内容の表示 R
R(Register) ：命令実行後のレジスタ内容を表示します。

【機能】 現在のプログラムカウンタから 1 命令ごとに指定ステップ数分実行します。

【説明】 (1) 1 命令実行するたびに、実行したニモニックを表示します。
オプション R を指定した場合は命令実行後のレジスタ内容も表示します。
(2)本コマンドでは、BSR、JSR、BSRF 命令で分岐したサブルーチンは、サブルーチンの開始から RTS 命令の次の命令（RTS は遅延分岐命令のため）までを 1 ステップとして実行します。
(3)ブレーク系コマンドで設定したブレーク条件を満たしたときやシミュレータ・デバッガがエラーを検出したときは、実行を停止します。
実行停止時は、停止要因を表示します。

【使用例】 5 命令実行します。サブルーチン内部は、 1 ステップとして実行します。

```
: STEP 5 (RET)
00000000      MOV.L       R3,@R14
00000002      MOV.L       @(H'0084:8,PC),R1
00000004      JSR         @R1
00000006      NOP
00000008      LDS.L       @R15+,PR
+++ 5000 : Step normal end
:
```

4.52 STEP_G

STEP_G SG	サブルーチンを 1 ステップとしてステップ実行のアドレス範囲指定

【形式】 STEP_G△<開始 PC 位置>△<終了 PC 位置>[△R](RET)

【パラメータ】
- <開始 PC 位置> ダミーの値（0～H'FFFFFFFF）を指定してください。
  本シミュレータ・デバッガでは、ステップ実行アドレス範囲の開始 PC 位置は常に現在 PC 位置になります。
- <終了 PC 位置> ステップ実行範囲終了 PC 位置を指定します。
- レジスタ内容の表示 R
  R(Register) ：命令実行後のレジスタ内容を表示します。

【機能】 現在 PC 位置から終了 PC 位置までを 1 命令ごとに実行します。

【説明】 (1)最後に実行したニモニックを表示します。
オプション R を指定した場合は命令実行後のレジスタ内容も表示します。
(2)本コマンドでは、BSR、JSR、BSRF 命令で分岐したサブルーチンは、サブルーチンの開始から RTS 命令の次の命令（RTS は遅延分岐命令のため）までを１ステップとして実行します。
(3)ブレーク系コマンドで設定したブレーク条件を満たしたときやシミュレータ・デバッガがエラーを検出したときは、実行を停止します。
実行停止時は、停止要因を表示します。
(4)開始 PC 位置、終了 PC 位置の関係を次に説明します。
・終了 PC 位置が奇数の場合、偶数にするために、最終ビットを０にする。
（0xfffffffe との＆を取り、偶数にする）
EX:1→0、101→100、ffff→fffe
・現在 PC 位置＝終了 PC 位置の時、現在 PC 位置を１ステップ実行する。
・現在 PC 位置＞終了 PC 位置の時、現在 PC 位置を１ステップ実行する。
・現在 PC 位置＜終了 PC 位置の時、現在 PC 位置から終了 PC 位置までステップを実行する。
PC 値が現在 PC 位置と終了 PC 位置の範囲からはずれた場合には、ステップを停止します。
（サブルーチン実行中は除きます）

【使用例】 現在 PC 位置から H'8 番地まで実行します。

```
: STEP_G 0 8 (RET)
00000008      LDS.L       @R15+,PR
+++ 5000 : Step normal end
:
```

4.53 STEP_INTO

STEP_INTO SI	ステップ実行

【形式】 STEP_INTO[△<ステップ数>][△R](RET)

【パラメータ】
・<ステップ数> 命令実行ステップ数を指定します。（H'1～H'FFFF）
省略時は、１ステップです。
・レジスタ内容の表示 R
R(Register) ：命令実行後のレジスタ内容を表示します。

【機能】 PC 位置から指定ステップ数分命令を実行します。
プログラムで関数コールを行った場合、コール先の関数内も１ステップずつ実行します。

【説明】 (1)１命令実行するたびに、実行したニモニックを表示します。
オプション R を指定した場合は命令実行後のレジスタ内容も表示します。
(2)ブレーク系コマンドで設定したブレーク条件を満たしたときやシミュレータ・デバッガがエラーを検出したときは、実行を停止します。
実行停止時は、停止要因を表示します。

【使用例】 (1)1 命令実行後、実行した命令のニモニックと命令実行後のレジスタ内容を表示します。
(SH-3 の場合)

```
: STEP_INTO R(RET)
  PC=00001002 SR=700000F0:-MRB----------IIII---- SP=00000000
  GBR=00000000 VBR=00000000 MACH=00000000 MACL=00000000 PR=00000000
  R0-7    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  R8-15 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  R0_BANK-R3_BANK 00000000 00000000 00000000 00000000
  R4_BANK-R7_BANK 00000000 00000000 00000000 00000000
  SSR=00000000 SPC=00000000
  PTEH=00000000 PTEL=00000000 TTB=00000000 TEA=00000000 MMUCR=00000000
  EXPEVT=00000000 INTEVT=00000000 TRA=00000000 CCR=00000000
  00001000    MOV          #H'00,R3
  +++ 5000 : Step normal end
:
```

(2)3 命令実行します。

```
: STEP_INTO 3 (RET)
00000404     MOV.L        #0000002E,R4
00000406     MOV.L        #FFFFFFFF,R3
00000408     ADD.L        R1,R2
  +++ 5000 : Step normal end
:
```

4.54 STEP_INTO_G

STEP_INTO_G SIG	ステップ実行の範囲指定

【形式】 STEP_INTO_G△<開始 PC 位置>△<終了 PC 位置>[△R](RET)

【パラメータ】
- <開始 PC 位置> ダミーの値（0～H'FFFFFFFF）を指定してください。
  本シミュレータ・デバッガでは、ステップ実行アドレス範囲の開始 PC 位置は常に現在 PC 位置になります。
- <終了 PC 位置> ステップ実行範囲終了 PC 位置を指定します。
- レジスタ内容の表示 R
  R(Register)：命令実行後のレジスタ内容を表示します。

【機能】 現在 PC 位置から終了 PC 位置までを 1 命令ごとに実行します。
プログラムで関数コールを行った場合、コール先の関数内も 1 ステップずつ実行します。

【説明】 (1)最後に実行したニモニックを表示します。
オプション R を指定した場合は命令実行後のレジスタ内容も表示します。
(2)プログラムで関数コールを行った場合、コール先の関数内も１ステップずつ実行します。
(3)ブレーク系コマンドで設定したブレーク条件を満たしたときやシミュレータ・デバッガがエラーを検出したときは、実行を停止します。
実行停止時は、停止要因を表示します。
(4)開始 PC 位置、終了 PC 位置の関係を次に説明します。
・終了 PC 位置が奇数の場合、偶数にするために、最終ビットを０にする。
（0xfffffffe との＆を取り、偶数にする）
EX:1→0、101→100、ffff→fffe
・現在 PC 位置＝終了 PC 位置の時、現在 PC 位置を１ステップ実行する。
・現在 PC 位置＞終了 PC 位置の時、現在 PC 位置を１ステップ実行する。
・現在 PC 位置＜終了 PC 位置の時、現在 PC 位置から終了 PC 位置までステップを実行する。
PC 値が現在 PC 位置と終了 PC 位置の範囲からはずれた場合には、ステップを停止します。（サブルーチン実行中を含む。）

【使用例】 現在 PC 位置から H'6 番地まで実行します。

```
: STEP_INTO_G 0 6(RET)
 00000006     ADD.L          R1,R2
 +++5000 : Step normal end
:
```

4.55 TLB (SH-3/ SH-3E,SH-3DSP,SH-4 シリーズのみ)

TLB	TLB の内容の変更
TLB	

【形式】 ・SH-3,SH-3E,SH-3DSP の場合
変更： TLB△<インデックス>△<ウェイ>[△<AA データ>][△:△<DA データ>](RET)
対話形式： TLB△<インデックス>△<ウェイ>(RET)
・SH-4 の場合
変更： TLB[△ {I｜U}]△<エントリ>[△<AA データ>][△:△<DA データ>](RET)
対話形式： TLB[△ {I｜U}]△<エントリ>(RET)

【パラメータ】
・<インデックス> 変更する TLB のインデックスを指定します。（H'00～H'1F）
・<ウェイ> 変更する TLB のウェイを指定します。（H'0～H'3）
・<AA データ> アドレスアレイに書き込むデータを指定します。
・<DA データ> データアレイに書き込むデータを指定します。
・変更する TLB の種別{I｜U} {I｜U}
I： 命令 TLB（ITLB）を指定します。（デフォルト）
U： 共有 TLB（UTLB）を指定します。
・<エントリ> 変更する TLB のエントリを指定します。

【機能】 TLB アドレスアレイ、データアレイの内容の変更を行います。

【説明】 (1)変更（直接）

指定されたデータで TLB の内容を変更します。

(2)変更（対話形式）

データ（<AA データ>、<DA データ>）を省略すると対話形式で TLB の内容を変更します。
この場合、現在のデータを表示し、変更データを要求します。
対話形式の入力フォーマットを以下に示します。

・SH-3/SH-3E,SH-3DSP の場合

: TLB△<インデックス>△<ウェイ>(RET)
ii w aaaaaaaa/dddddddd : [△<AA データ>][△:△<DA データ>](RET)
ii w aaaaaaaa/dddddddd :
(1)(2) (3) (4)

(1)インデックス（16 進数２桁）
(2)ウェイ（16 進数１桁）
(3)現在のアドレスアレイの値（16 進数８桁）
(4)現在のデータアレイの値（16 進数８桁）

・SH-4 の場合

: TLB△<エントリ>(RET)
ee aaaaaaaa/dddddddd : [△<AA データ>][△:△<DA データ>](RET)
ee aaaaaaaa/dddddddd :
(1) (2) (3)

(1)エントリ（16 進数２桁）
(2)現在のアドレスアレイの値（16 進数８桁）
(3)現在のデータアレイの値（16 進数８桁）

変更データの入力ではデータ以外に次のものを入力することができます。
.(ピリオド)：コマンドを終了します。
^：前のエントリに戻ります。
(RET)のみ入力：次のエントリに進みます。

【使用例】　(1)SH-3 でインデックス 0、ウェイ 0 のエントリを変更します。

```
: TLB 0 0 00000000 : 00000000(RET)
:
```

(2)SH-3 でインデックス 0、ウェイ 0 のエントリから順に TLB の内容を変更します。

```
: TLB 0 0(RET)
00 0 00000000/00000000 : 00000101 : 00000500 (RET)
00 1 00000000/00000000 : 00000101 : 00000900 (RET)
00 2 00000000/00000000 : (RET)
00 3 00000000/00000000 : ^(RET)
00 2 00000000/00000000 : 00000101 : 00001100 (RET)
00 3 00000000/00000000 : 00000101 : 00001100 (RET)
01 0 00000000/00000000 : .(RET)
:
```

4.56　TLB_DUMP (SH-3/ SH-3E,SH-3DSP,SH-4 シリーズのみ)

TLB_DUMP	TLB の内容の表示
TLBD	

【形式】　・SH-3/SH-3E,SH-3DSP の場合
TLB_DUMP (RET)
・SH-4 の場合
TLB_DUMP[△｛I｜U｝] (RET)

【パラメータ】　・表示する TLB の種別　｛I｜U｝
I： 命令 TLB（ITLB）を指定します。（デフォルト）
U： 共有 TLB（UTLB）を指定します。

【機能】　TLB のアドレスアレイおよびデータアレイの内容を表示します。

【説明】　・SH-3/SH-3E,SH-3DSP の場合
TLB を以下のフォーマットで表示します。

```
<NO>      <WAY0>              <WAY1>              <WAY2>              <WAY3>
 ii   aaaaaaaa/dddddddd aaaaaaaa/dddddddd aaaaaaaa/dddddddd aaaaaaaa/dddddddd
(1)     (2)     (3)     (2)     (3)     (2)     (3)     (2)     (3)
```

(1)インデックス（１６進数２桁）
(2)各ウェイのアドレスアレイの値（16 進数８桁）
(備考) bit16-bit12 の 5 ビットは常に 0 となります。
(3)各ウェイのデータアレイの値（16 進数８桁）

・SH-4 の場合
TLB を以下のフォーマットで表示します。

```
<NO> <ADDR ARAY>   <DATA ARRAY>
 ee      aaaaaaaa/dddddddd
(1)        (2)        (3)
```

(1)エントリ（16 進数２桁）
(2)アドレスアレイの値（16 進数８桁）
(3)データアレイの値（16 進数８桁）

【使用例】　SH-3 で全インデックスの内容を表示します。

```
: TLB_DUMP(RET)
<NO>      <WAY0>              <WAY1>              <WAY2>              <WAY3>
00   00000000/00000000 00000000/00000000 00000000/00000000 00000000/00000000
01   00000000/00000000 00000000/00000000 00000000/00000000 00000000/00000000
 :          :                   :                   :                   :
1F   00000000/00000000 00000000/00000000 00000000/00000000 00000000/00000000
:
```

4.57　TLB_FLUSH (SH-3/ SH-3E,SH-3DSP,SH-4 シリーズのみ)

TLB_FLUSH TLBF	TLB のフラッシュ

【形式】　・SH-3/SH-3E,SH-3DSP の場合
TLB_FLUSH(RET)
・SH-4 の場合
TLB_FLUSH[△｛I｜U｝] (RET)

【パラメータ】　・フラッシュする TLB の種別　｛I｜U｝
I： 命令 TLB（ITLB）を指定します。（デフォルト）
U： 共有 TLB（UTLB）を指定します。

【機能】　TLB をフラッシュします。

【使用例】　SH3 で TLB をフラッシュします。

```
: TLB_FLUSH(RET)
:
```

4.58 TLB_SEARCH (SH-3/3E,SH-3DSP,SH-4 シリーズのみ)

TLB_SEARCH	TLB の内容の検索
TLBS	

【形式】
・SH-3/SH-3E,SH-3DSP の場合
TLB_SEARCH△<アドレス>[△<アドレス種別>] (RET)
・SH-4 の場合
TLB_SEARCH[△ {I｜U}]△<アドレス>[△<アドレス種別>] (RET)

【パラメータ】
・<アドレス>　検索するアドレスを指定します。
・<アドレス種別>　検索するアドレスの種別を指定します。
P：物理アドレス
V：論理アドレス（デフォルト）

・検索する TLB の種別　{I｜U}
I： 命令 TLB（ITLB）を指定します。（デフォルト）
U： 共有 TLB（UTLB）を指定します。

【機能】　指定した論理アドレスが登録されている TLB のアドレスアレイを検索します。
また、指定された物理アドレスが登録されている TLB のデータアレイを検索します。
該当するエントリのインデックス、WAY 番号、アドレスアレイ値、データアレイ値を表示します。

【使用例】　(1)SH-3 で論理アドレス H'00000000 で TLB の内容を検索します。

```
: TLB_SEARCH 0 (RET)
  00 0 00000101/20000158
  00 2 00000103/20100158
:
```

(2)SH-3 で物理アドレス H'20000000 で TLB の内容を検索します。

```
: TLB_SEARCH 20000000 P (RET)
  00 0 00000101/20000158
:
```

4.59 TRACE

TRACE T	トレースバッファの表示

【形式】　TRACE[△<オフセット>[△<カウント>]](RET)

【パラメータ】
・<オフセット>　表示する最初のサイクルを指定します。（0～トレースバッファ容量－１）
トレースバッファの先頭から何サイクル目から表示するかを意味します。
省略した場合は、デフォルトをバッファの最後から 9 サイクルの位置とします。
・<カウント>　表示するカウントを指定します。（1～32768）
省略した場合は、10 サイクル分表示します。
カウントを指定する場合は、必ずオフセットの指定も必要です。

【機能】　トレースバッファの内容を表示します。
バッファでの最後の（最も最近実行された）サイクルはサイクル 0 で、それ以前のサイクルは負の値を持っています。

【説明】　(1)表示する内容は、以下のとおりです。
なお、パイプライン動作の詳細については、各デバイスのプログラミングマニュアルを参照してください。
SH-1/SH-2、SH-2E、SH-DSP シリーズの場合：

```
PTR      CYCLE          ADDR-BUS PIPELINE          INSTRUCTION
XXXXXX XXXXXXXXXX XXXXXXXX XXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

[PTR]:トレースバッファ内ポインタ（最後に実行した命令が 0 となります）
[CYCLE]:累計命令実行サイクル数（パイプラインリセットによりクリアされます）
[ADDR-BUS]:命令アドレス
[PIPELINE]:パイプラインの実行状況各記号の意味は以下の通りです。

- F: 命令フェッチ（メモリアクセスあり）
- f: 命令フェッチ（メモリアクセスなし）
- D: 命令デコード
- E: 命令実行
- M: メモリアクセス
- W: ライトバック
- P: DSP(SH-DSP シリーズのみ)
- m: 乗算器実行
- -: 命令固有のストール
- >: スプリット
- <: 競合によるストール

[INSTRUCTION]:命令ニモニックとデータアクセス（転送先←転送データの形式で表示）

【説明】　SH-3/SH-3E シリーズの場合：

```
PTR      CYCLE          ADDR-BUS DATA-BUS    CODE     NO    INSTRUCTION
IF DE EX MA SW
ACCESS DATA
XXXXXX XXXXXXXXXX    XXXXXXXX XXXXXXXX    XXXX     XX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX    XX XX XX XX XX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
```

[PTR]：トレースバッファ内ポインタ（最後に実行した命令が 0 となります）
[CYCLE]：累計命令実行サイクル数（パイプラインリセットによりクリアされます）
[ADDR-BUS]：アドレスバス上のデータ
[DATA-BUS]：データバス上のデータ
[CODE]：命令コード
[No]：命令実行番号（各ステージの実行番号と対応しています）
[INSTRUCTION]：命令ニモニック
[IF]：フェッチした命令の実行番号
[DE]：デコードした命令の実行番号
[EX]：実行した命令の実行番号
[MA]：メモリアクセスした命令の実行番号
[SW]：ライトバックした命令の実行番号
[ACCESS DATA]：データアクセスの内容（転送先←転送データの形式で表示）

SH-3DSP シリーズの場合：

```
PTR      CYCLE          ADDRESS    CODE          IF DE EX MA SW    NO
INSTRUCTION
XXXXXX XXXXXXXXXX    XXXXXXXX XXXX          XX XX XX XX XX    XX
XXXXXXXXXXXX
```

[PTR]：トレースバッファ内ポインタ（最後に実行した命令が 0 となります）
[CYCLE]：累計命令実行サイクル数（パイプラインリセットによりクリアされます）
[ADDRESS]：プログラムカウンタ
[CODE]：命令コード
[IF]：フェッチした命令の実行番号
[DE]：デコードした命令の実行番号
[EX]：実行した命令の実行番号
[MA]：メモリアクセスした命令の実行番号
[SW]：ライトバックした命令の実行番号
[No]：命令実行番号（各ステージの実行番号と対応しています）
[INSTRUCTION]：命令ニモニックとデータアクセスの内容
（転送先←転送データの形式で表示）

【説明】　SH-4 シリーズの場合：

```
PTR      CYCLE          ADDRESS     code1 code2 EX-EAS LS-EAS BR-EAS FP-EXASD
INSTRUCTION    ACCESS DATA
XXXXXX XXXXXXXXXX XXXXXXXX   XXXX   XXXX   X X X   X X X   X X X   X X
X X X XXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

[PTR]　：トレースバッファ内ポインタ（最後に実行した命令が 0 となります）
[CYCLE]：累計命令実行サイクル数（パイプラインリセットによりクリアされます）
[ADDRESS]　：プログラムカウンタ値
[code1]　：フェッチした命令のコード 1
[code2]　：フェッチした命令のコード 2
[EX-EAS]：EX パイプラインで実行(E)、メモリアクセス(A)、ライトバック(S)した命令の番号
[LS-EAS]：LS パイプラインで実行、メモリアクセス、ライトバックした命令の番号
[BR-EAS]：BR パイプラインで実行、メモリアクセス、ライトバックした命令の番号
[FP-EXASD]　：FP パイプラインで実行、メモリアクセス、ライトバックした命令の番号
(X ステージは、FSCA,FSRRA,FIPR,FTRV 命令、D ステージは、FDIV,FSQRT 命令のみ使用します。)
[INSTRUCTION]：実行する命令に割り振られた命令番号、実行する命令のメモリ上のアドレス、命令コード、ニーモニック
[ACCESS DATA]：データアクセスの内容（転送先←転送データの形式で表示）

【留意事項】　トレース情報は（0～）トレースバッファ容量まで表示します。トレースバッファ容量は TRACE_CONDITION コマンドで設定できます。

【使用例】　(1)トレースバッファの先頭から 5 サイクル分表示します。（SH-1 の場合）

```
: TRACE 0 5(RET)
PTR      CYCLE    ADDR-BUS PIPELINE          INSTRUCTION
-01023 0000010193 000001B0 FFDE>MM      :MOV.L    R1, @R6    000001EC<-00000001
-01022 0000010195 000001B2 fD>E>        :TST      R5, R5    T<-(0)
-01021 0000010197 000001B4 FFD>E>       :BT       000001BA    T(0)
-01020 0000010199 000001B6 f>D>E        :MOV      #FF, R2    R2<-FFFFFFFF
-01019 0000010200 000001B8 FFDE>MM      :MOV.L    R2, @R6    000001EC<-FFFFFFFF
:
```

(2)トレースバッファの先頭から 5 サイクル分表示します。（SH-4 の場合）

```
: t 0 5(RET)
PTR     CYCLE    ADDRESS    code1 code2 EX-EAS LS-EAS BR-EAS FP-EXASD
INSTRUCTION    ACCESS DATA
-01023 0000004426 00000060   xxxx   xxxx   x x x    x x x    x x x    x x x x x
-01022 0000004427 00000060   xxxx   xxxx   x x x    x x x    x x x    x x x x x
-01021 0000004428 00000064   xxxx *E30A   x x x    x x x    x x x    x x x x x
[5 (0000005E): E30A    MOV     #0A, R3]
-01020 0000004429 00000064   xxxx   xxxx   5 x x    x x x    x x x    x x x x x
-01019 0000004430 00000064   xxxx   xxxx   x 5 x    x x x    x x x    x x x x x
:
```

4.60 TRACE_CONDITION

TRACE_CONDITION TC	トレース条件の設定、トレースの開始、終了

【形式】 開始： TRACE_CONDITION[[△{I｜S}][△E][△{C｜B}] [△BUF={1｜4｜16｜32}]](RET)
終了： TRACE_CONDITION△D(RET)

【パラメータ】

・命令の種類 {I｜S}
I(Instruction)：すべての命令をトレースバッファに格納します。（デフォルト）
S(Subroutine)：サブルーチンコール命令（BSR、JSR、BSRF）のみトレースバッファに格納します。

・ﾄﾚｰｽの開始、終了 {E｜D}
E(Enable)：トレースバッファへの格納を開始します。（デフォルト）
D(Disable)：トレースバッファへの格納を終了します。

・ﾄﾚｰｽﾊﾞｯﾌｧ満杯時の処理 {C｜B}
C(Continue)：トレースバッファ満杯時にオーバーライトします。（デフォルト）
B(Break)：トレースバッファ満杯時に実行を中断します。

・ﾄﾚｰｽﾊﾞｯﾌｧ容量 {1｜4｜16｜32}
1：1024 サイクル（デフォルト）
4：4096 サイクル
16：16384 サイクル
32 ：32768 サイクル

【機能】 トレース条件の設定を行います。

【説明】 シミュレータ起動時は、トレースバッファへの格納は終了になっています。

【使用例】 (1)本コマンド以降の命令実行において、すべての命令をトレースバッファに格納します。

```
: TRACE_CONDITION I(RET)
:
```

(2)サブルーチンコール命令のみトレースバッファに格納します。また、トレースバッファのサイズを 16384 サイクルにします。

```
: TRACE_CONDITION S E BUF=16(RET)
:
```

(3)トレースバッファへの格納を終了します。

```
: TRACE_CONDITION D(RET)
:
```

4.61 TRACE_CLEAR

TRACE_CLEAR	トレース条件の初期化
TL	

【形式】 TRACE_CLEAR(RET)

【機能】 トレースバッファの内容を初期化します。

【説明】 初期化ではトレースバッファのみをクリアします。
命令の種類、トレースの開始／終了、トレースバッファ満杯時の処理は変更しません。

【使用例】 トレースバッファの内容を初期化します。

```
: TRACE_CLEAR(RET)
:
```

4.62 TRAP_ADDRESS

TRAP_ADDRESS	システムコール開始位置の設定
TA	

【形式】 RAP_ADDRESS△<命令位置>(RET)

【パラメータ】 ・<命令位置> システムコール開始位置を指定します。
設定を行うとシステムコールが有効となります。

【機能】 デバッグ対象プログラムから標準入出力への文字入出力を行うシステムコールの処理開始位置の設定を行います。設定できる位置は、１個です。
JSR、BSR 命令を実行する際に、分岐アドレスが、このコマンドで指定された位置ならば、通常のシミュレーションは行わないで、機能コードで示すシステムコールを実行します。
デバッグ対象プログラムの中でパラメータブロックおよび入出力バッファの領域を確保してください。
JSR、BSR 命令を実行するときに、R0、R1、パラメータブロック、および入出力バッファの設定を行っておいてください。
システムコール処理が終了すると、JSR、BSR 命令の次の命令から、シミュレーションを再開します。
システムコール使用方法の詳細は、**「2.11 標準入出力およびファイル入出力処理」**を参照ください。

【使用例】 システムコール開始位置を H'10 に設定します。

```
:TRAP_ADDRESS 10(RET)
:
```

4.63 TRAP_ADDRESS_DISPLAY

TRAP_ADDRESS_DISPLAY	システムコール開始位置の表示
TD	

【形式】 TRAP_ADDRESS_DISPLAY(RET)

【機能】 システムコール開始位置および有効／無効を表示します。

【説明】 システムコール開始位置および有効／無効を以下のフォーマットで表示します。

```
:TD(RET)
 aaaaaaaa b
:
```

aaaaaaaa：システムコール開始位置
b：E(Enable)システムコールが有効です。
D(Disable)システムコールが無効です。

【使用例】 システムコール開始位置および有効／無効を表示します。

```
:TRAP_ADDRESS_DISPLAY(RET)
00000010 E
:
```

4.64 TRAP_ADDRESS_ENABLE

TRAP_ADDRESS_ENABLE	システムコール開始位置の有効／無効
TE	

【形式】 TRAP_ADDRESS_ENABLE△{E｜D}(RET)

【パラメータ】
・<命令位置> システムコール開始位置を指定します。
設定を行うとシステムコールが有効となります。
・有効／無効 {E｜D} {E｜D}
E(Enable)：設定したシステムコール開始位置を有効にします。
D(Disable)：設定したシステムコール開始位置を無効にします。

【機能】 デバッグ対象プログラムから標準入出力への文字入出力を行うシステムコールの有効／無効の設定を行います。

【使用例】 システムコール開始位置を無効にします。

```
:TRAP_ADDRESS_ENABLE D(RET)
:
```

4.65 .<register>

.<register>	レジスタ内容の変更

【形式】 コマンド一般形 ： .<レジスタ>△{<データ>｜<実数>}(RET)
対話形式 ： .<レジスタ>(RET)

【パラメータ】 ・<レジスタ> 汎用レジスタ名、コントロールレジスタ名、システムレジスタ名、制御レジスタ名、DSP レジスタ名または FPU レジスタ名を指定します。
・<データ> 変更値を指定します。
・<実数> 変更値を単精度浮動小数点数（数値の前に F' を付けます）で指定します。

【機能】 レジスタ内容を指定された値に変更します。

【説明】 (1)汎用レジスタ名には R0 から R15 が指定できます。R15 のかわりに SP と指定することができます。
SH-3/SH-3E、SH-3DSP、SH-4 シリーズでは上記に加えて R0_BANK から R7_BANK が指定できます。
なお、R0-R7、R0_BANK-R7_BANK は、SR の MD ビット、RB ビットの内容にしたがって相互に変更の結果が反映されます。
(2)コントロールレジスタ名には、SR、GBR、VBR が指定できます。
SH-3DSP、SH-DSP シリーズでは上記に加えて RS、RE、MOD が指定できます。
SH-3/SH-3E、SH-3DSP、SH-4 シリーズでは上記に加えて SSR、SPC が指定できます。
SH-4 シリーズでは上記に加えて SGR、DBR が指定できます。
(3)システムレジスタ名には、MACH、MACL、PR、PC が指定できます。
(4)FPU レジスタ名には、FPUL、FPSCR、FR0-FR15 が指定できます。（SH-2E、SH-3E、SH-4 シリーズ）
SH-4 シリーズでは上記に加えて XF0-XF15、DR0-DR14、XD0-XD14 の指定ができます。
(5) 制御レジスタ名には下記のレジスタが指定できます。
SH-DSPwithCache：CCR
SH-3/SH-3E：CCR、PTEH、PTEL、TTB、TEA、MMUCR、EXPEVT、INTEVT、TRA
SH-4 シリーズ： CCR 、PTEH、PTEL、TTB、TEA、MMUCR、EXPEVT、INTEVT、TRA QACR0、QACR1
SH-3DSP シリーズ： CCR 、CCR2、PTEH、PTEL、TTB、TEA、MMUCR、EXPEVT、INTEVT、TRA
(6)DSP レジスタ名には、A0、A0G、A1、A1G、M0、M1、X0、X1、Y0、Y1、DSR が指定できます。
(SH-3DSP、SH-DSP シリーズのみ)
A0G、A1G レジスタは、8 ビットの範囲内で値を設定できます。
8 ビットを越えた値を指定した場合は、下位 8 ビット分が有効となります。

【説明】 (7)対話形式のコマンドが入力されると、指定されたレジスタの内容を表示した後、対話形式となります。

.<register>(RET)
register data : [{<データ>｜<実数>｜^}](RET)
register data : [{<データ>｜<実数>｜^}](RET)
・
・
・
register data : .(RET)

register data：変更前のデータを示します。
<データ>：変更データを指定します。
<実数>：変更値を単精度浮動小数点数で指定します。
^：直前のレジスタ内容を表示します。
(RET)のみ入力：次のレジスタ内容を表示します。
.(ピリオド)：.<register>コマンドを終了します。

【使用例】 (1)レジスタ R0 の内容を H'1000 に変更します。

: .R0 1000(RET)
:

(2)レジスタ R1 の内容を H'9999 に変更します。

: .R1 9999(RET)
:

(3)レジスタ R1 から対話形式で順次レジスタ内容の変更および直前直後のレジスタの内容表示を行います。

: .R1(RET)
R1 00009999 : 5678(RET)
R2 00000000 : ^(RET)
R1 00005678 : (RET)
R2 00000000 : 345(RET)
R3 00000000 : ^(RET)
R2 00000345 : .(RET)
:

(4)システムレジスタ PC から対話形式でレジスタの内容の表示および変更を行います。

: .PC(RET)
PC 00000100 : 400(RET)
SR 00000000 : .(RET)
:

(5)FPU レジスタ FR0 の内容を F'9876543e+21 に変更します。

: .FR0 F'9876543e+21(RET)
:

4.66 制限事項

SuperH™ RISC engine シミュレータ・デバッガを使用する際の制限事項を下記に示します。

(1) コマンド入力時の演算子

コマンド入力時に入力した文字列内に演算子が含まれていた場合は、演算子の前後の数字を計算し使用します。

但し、演算子の前後にスペースが入っていた場合は計算の対象外となりますので御注意願います。

また、浮動小数点データの場合も計算の対象外となります。

例：

```
: me 0 10+1 ;b
: md 0 10
address    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
00000000 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
:
```

(2) 実行ニモニック表示

Go、Step 系のコマンドで、最後に実行したニモニックコードを表示する時に、表示されたニモニックコードが最後に実行したコードではない場合があります。これは 1step の単位を E ステージから E ステージとしているために、1step の間に 2 回デコードが入る場合があり、その時に最初に実行したコードが表示されます。

(3) メモリマップ

メモリ領域の下端に遅延分岐命令があると、その命令を実行したときに「Memory Access Error」が発生します。

遅延分岐命令では、次命令をフェッチまで行います。

従ってこの読み捨てになる命令のプリフェッチの動作で、メモリがないためにエラーとなっています。

(4) Memory_Display コマンド

Memory_Display コマンドのサイズに W、L、D、S 指定時は、それぞれのサイズの境界を開始位置に指定してください。

境界以外のアドレスを指定した場合は、オール F で表示されます。

例：

```
: md 0 10 ;w
 address   +0     +2     +4     +6     +8     +A     +C     +E
 00000000 0001 0002 0003 0004 0005 0006 0007 0008
 : md 1 10 ;w
 address   +0     +2     +4     +6     +8     +A     +C     +E
 00000001 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 :
```

(5) 浮動小数点表現

浮動小数点の特殊な値の表現について示します。

表現方法は、使用しているマシンに依存します。

表4.2　単精度における特殊な値の表現

値	SPARC	HP9000
0x00000000	0.000000e+00	0.000000e+00
0x80000000	-0.000000e+00	-0.000000e+00
0x7F800000	Inf	+.+00000e+01
0xFF800000	-Inf	--.-00000e+01
0x7F800001	NaN	?.000000e+00

表4.3　倍精度における特殊な値の表現

値	SPARC	HP9000
0x0000000000000000	0.000000000000000e+00	0.000000000000000e+00
0x8000000000000000	-0.000000000000000e+00	-0.000000000000000e+00
0x7FF0000000000000	Infinity	+.+00000000000000e+01
0xFFF0000000000000	-Infinity	--.-00000000000000e+01
0x7FF8000000000001	NaN	?.000000000000000e+00

(6)　浮動小数点演算

HP9000 版シミュレータ・デバッガでは浮動小数点の演算に下記の制限があります。

- **クアイアット非数**

演算結果がクアイアット非数になるところがシグナリング非数になる場合があります。

- FPSCR **レジスタの無効演算フラグ**

演算結果が無効演算であった場合でも無効演算フラグがたたない場合があります。

(7)　コマンド入力の Tab コード

コマンド入力時に Tab コードは区切り文字とはなりませんので、Tab コードを入力しないでください。

5. メッセージ一覧

5.1 インフォメーションメッセージ

シミュレータ・デバッガは実行経過をユーザに知らせるため、インフォメーションメッセージを出力します。シミュレータ・デバッガの出力するインフォメーションメッセージを**表** 5.1に示します。

表 5.1 **インフォメーションメッセージ一覧**

番号	メッセージ	内容
5000	Step normal end	**ステップ実行が正常に終了しました。**
5001	PC breakpoint	**ブレークポイント条件が成立して実行を中断しました。**
5002	Break sequence	**ブレークシーケンス条件が成立して実行を中断しました。**
5003	Break data	**ブレークデータ条件が成立して実行を中断しました。**
5004	Break register	**ブレークレジスタ条件が成立して実行を中断しました。**
5005	Break access	**ブレークアクセス条件が成立して実行を中断しました。**
5006	Trace buffer full	TRACE_CONDITION **コマンドで** B（Break）**モードが選択され、かつトレースバッファが満杯となったので実行を中断しました。**
5007	Sleep	SLEEP **命令により実行を中断しました。**

5.2 エラーメッセージ

シミュレータ・デバッガはデバッグ対象プログラムや操作の誤りをユーザに知らせるため、エラーメッセージを出力します。シミュレータ・デバッガの出力するエラーメッセージを**表** 5.2に示します。

表 5.2 **エラーメッセージ一覧**

番号	メッセージ	内容･対策
0001	Program error	**内部エラーが発生しました。** **最寄りの当社営業所へ連絡してください。**
0007	Not enough memory	**シミュレータ・デバッガで使うメモリを確保できません。** **メモリの拡張、またはデバッグ対象プログラムの変更を行ってください。**
0009	User aborted	**ブレークキー入力により処理を中断しました。**
0010	File not found	**指定されたファイルが見つかりません。**
0011	Invalid CPU kind	**指定されたデバッグ情報ファイルと** CPU **の設定が一致しません。**
0501	Invalid parameter	**パラメータ値が不正です。**
0502	Invalid address	**アドレス値として不正です。** **正しく指定してください。**
0503	User break	**ブレークキー入力により処理を中断しました。**

番号	メッセージ	内容･対策
0505	Not found	見つかりませんでした。
0507	Cannot load file	ロードできません。
0508	Cannot save file	セーブできません。
0509	Divide by zero	整数式において除数が 0 です。 除数は 0 以外の数値に変更してください。
0510	Number out of range	範囲外のデータを指定しました。
0511	Invalid command	不正なコマンドを実行しました。
0512	Invalid operator	演算子が不正です。
0513	Mismatched parentheses	括弧()の対応が不正です。
0514	Invalid character constant	文字定数が不正です。
0515	Invalid register name	レジスタ名が不正です。
0516	Invalid function name	関数名が不正です。
0517	File write error	ファイルの書き込みで不正が発生しました。
0519	Out of analysis space	解析範囲を超えています
0520	Analysis ranges overlap	解析の範囲が重なっています。
0521	Not an analysis range	解析範囲ではありません。
0522	No trace data available	有効なトレースデータがありません。
0525	File verify error	ファイルのベリファイで差違がありました。
0526	File format error	ファイルのフォーマットが不正です。
0532	Cannot load as program. No Source level debugging available	デバッグ対象プログラムがロードされていません。 デバッグ対象プログラム作成時にデバッグオプションが指定されていないと考えられます。 ソースレベルデバッグができません。
0533	File does not exist	指定したファイルは存在しませんでした。
0534	Not enough memory	シミュレータ・デバッガで使うメモリを確保できません。 メモリの拡張、またはデバッグ対象プログラムの変更を行ってください。
0536	No function selected	関数が選択されていません。
0537	File read error	ファイルリードでエラーが発生しました。
0600	Not logging	LOG コマンドによるロギングがされていません。
0601	No log file set	LOG ファイルが指定されていません。
0602	Program did not start.	デバッグ対象プログラムは実行されていません。
0603	Program has stopped.	デバッグ対象プログラムの実行は中断されています。

番号	メッセージ	内容･対策
0604	Must specify go till address.	ブレーク位置を指定してください。
0606	Memory map not available	メモリマップが設定されていません。
0607	Trace record out of range	トレース取得した範囲から外れています。
0609	Trace not available	トレースが有効ではありません。
0905	Invalid expression	不正な表現を行いました。
0906	String too long	文字列の文字数が長すぎます。
1002	Register not found	レジスタが見つかりません。
1003	Invalid register index	レジスタのインデックスが不正です。
1006	Invalid memory map mode specified	メモリマップの設定が不正です。
1007	Invalid endian type	エンディアンの設定が不正です。
1500	Invalid mnemonic	ニモニックが不正です。
1501	Invalid operand	オペランドが不正です。
1502	Syntax error	コマンドのパラメータが不正です。
1503	Too many operands	オペランドの指定が多すぎます。
1504	Operand out of range	オペランドで指定したアドレスが範囲を超えています。
1505	Bad address operand alignment	アドレスのアライメントが不正です。
1507	Invalid numeric constant	入力された数値が不正です。
1508	Divide by zero	整数式において除数が 0 です。 除数は 0 以外の数値に変更してください。
1509	Invalid mnemonic specifier	不正なニモニックの記述をしました。
2001	Not currently available	本コマンドはサポートしていないか、現在実行できない状態です。 また、リセットベクタが読み出し可能なメモリ領域に設定されていないことが考えられます。 読み出し可能または読み書き可能なメモリ領域に設定してください。
2010	Invalid address value	アドレス値が不正です。
2011	Invalid length value	長さの指定が不正です。
2012	Invalid index value	インデックスの指定が不正です。
2013	Invalid memory space value	不正なメモリマップを設定しようとしています。
2019	Invalid parameter	パラメータの指定が不正です。
2020	Compare failed	比較で不一致が生じました。

番号	メッセージ	内容･対策
2021	Find failed	一致する項目が見つかりませんでした。
2022	Verify failed	ベリファイで不一致が生じました。
2023	Table full	テーブルがいっぱいです。
2030	Not an instruction	ブレークポイントに指定したアドレスは命令ではありません。
2100	Invalid address	アドレスの指定が不正です。
2101	Invalid count	回数の指定が不正です。
2102	Too many parameter	パラメータの指定が多すぎます。
2103	No breakpoint set	ブレークポイントは設定されていません。
2104	Invalid data	データの指定が不正です。
2105	Invalid size	サイズの指定が不正です。
2106	Invalid option	オプションの指定が不正です。
2107	No breakdata set	ブレークが設定されていません。
2108	Invalid start address	開始アドレスの指定が不正です。
2109	Invalid end address	終了アドレスの指定が不正です。
2110	Invalid access type	アクセス種別の指定が不正です。
2111	No breakaccess set	ブレークが設定されていません。
2112	Invalid register name	レジスタ名の指定が不正です。
2113	No set parameter	パラメータが指定されていません。
2114	No breaksequence set	ブレークが設定されていません。
2115	Invalid index value	インデックスの指定が不正です。
2116	Get break failure	ブレークの設定を失敗しました。
2117	No resource	プログラムがロードされていません。
2118	No set performance range	時間測定機能が設定されていません。
2119	No breakregister set	ブレークが設定されていません。
2120	Address re-use	アドレスが再利用されました。
2121	Can't read	リードできませんでした。
2122	Exception handling error	例外のハンドリンクで不正処理が発生しました。
2123	No get memory area	メモリマップを設定できませんでした。
2124	Invalid address	アドレス指定が不正です。
2125	Address already use	指定されたアドレスは既に使われています。

番号	メッセージ	内容･対策
2126	Exception error	例外処理でエラーが発生しました。 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2127	Memory access error	以下のいずれかの状態になりました。 (1)確保されていないメモリ領域をアクセスしようとした (2)書き込み不可属性を持つメモリへの書き込みを行おうとした (3)読み出し不可属性を持つメモリからの読み出しを行おうとした (4)メモリが存在しない領域をアクセスしようとした メモリの確保､属性変更を行うか､当該メモリアクセスが発生しないようにデバッグ対象プログラムを修正してください。
2128	Address error	以下のいずれかの状態になりました。 (1)PC 値が奇数である (2)内蔵 I/O 空間から命令読み出しを行おうとした (3)ワードデータを(2n)番地以外からアクセスしようとした (4)ロングワードデータを(4n)番地以外からアクセスしようとした (5)VBR、SP が 4 の倍数以外である (6)アドレスエラーの例外処理でエラーが発生した エラーが発生しないようにデバッグ対象プログラムを修正してください。
2129	Memory already set	メモリマップが既に設定されています。
2130	Memory area not exist	指定されたアドレスはメモリマップ設定されていません。
2131	Invalid value	不正な値を指定しました。
2132	General invalid instruction	以下のいずれかの状態になりました。 (1)命令ではないコードを実行しようとした (2)一般不当命令の例外処理でエラーが発生した エラーが発生しないようにデバッグ対象プログラムを修正してください。
2133	Illegal operation	以下の状態になりました。 (1)DIV1 命令でゼロ除算が発生した エラーが発生しないようにデバッグ対象プログラムを修正してください。
2134	Invalid slot instruction	以下のいずれかの状態になりました。 (1)遅延分岐命令直後の PC を変える命令(分岐命令)を実行した (2)スロット不当命令の例外処理でエラーが発生した エラーが発生しないようにデバッグ対象プログラムを修正してください。
2135	Illegal DSP operation	以下のいずれかの状態になりました。 (1)PSHA 命令で 32 ビットを越えるシフトを行おうとした (2)PSHL 命令で 16 ビットを越えるシフトを行おうとした エラーが発生しないようにデバッグ対象プログラムを修正してください。

番号	メッセージ	内容･対策
2136	Invalid DSP instruction code	DSP パラレル命令で不正な命令コードを検出しました。 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2137	TLB miss	シミュレーション実行中またはコマンド実行中に TLB ミスが発生しました。 TLB の内容を更新する等必要な処置をとってください。
2138	TLB invalid	シミュレーション実行中またはコマンド実行中に TLB 無効例外が発生しました。 TLB の内容を更新する等必要な処置をとってください。
2139	TLB protection violation	シミュレーション実行中に TLB 保護違反が発生しました。
2140	Initial page write	シミュレーション実行中に初期ページ書き込みが発生しました。
2141	TLB multiple hit	シミュレーション実行中またはコマンド実行中にアクセスした論理アドレスが TLB の複数エントリにヒットしました。 TLB が適正に設定されていません。 TLB の内容を修正するとともにプログラム(ハンドラルーチン)を修正してください。
2142	Multiple exception	多重例外が発生しました。 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2143	Illegal LRU set	キャッシュの LRU の値が不正です。 設定を確認してください。
2144	Simulated I/O	システムコールエラーが発生しました。 レジスタ R0、R1 およびパラメータブロックの内容の誤りを修正してください。
2145	System call error	システムコールでエラーが発生しました。
2146	FPU error	浮動小数演算で､以下のいずれかの状態になりました。 (1)FPU エラー発生 (2)無効演算発生 (3)ゼロによる除算発生 (4)オーバフロー発生 (5)アンダーフロー発生 (6)不正確発生 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2147	No get stack data	スタック情報が取得されていません。
2148	Unified TLB miss	メモリアクセスで Unified TLB ミスが発生しました。 共用 TLB 内容を更新する等必要な処置をとってください。
2150	Unified TLB protection violation	メモリアクセスで Unified TLB 保護例外が発生しました。 共用 TLB 内容を更新する等必要な処置をとってください。
2151	Initial Page Write	シミュレーション実行中に初期ページ書き込みが発生しました。 TLB の内容を更新する等必要な処置をとってください。

番号	メッセージ	内容･対策
2152	Unified TLB multiple hit	シミュレーション実行中またはコマンド実行中にアクセスした論理アドレスが Unified TLB の複数エントリにヒットしました。 共用 TLB が適正に設定されていません。 共用 TLB の内容を修正するとともにプログラム(ハンドラルーチン)を修正してください。
2153	Instruction TLB miss	メモリアクセスで Instruction TLB ミスが発生しました。 TLB 内容を更新する等必要な処置をとってください。
2155	Instruction TLB protection violation	メモリアクセスで Instruction TLB 保護例外が発生しました。 命令 TLB 内容を更新する等必要な処置をとってください。
2156	Instruction TLB multiple hit	シミュレーション実行中またはコマンド実行中にアクセスした論理アドレスが Instruction TLB の複数エントリにヒットしました。 命令 TLB が適正に設定されていません。 命令 TLB の内容を修正するとともにプログラム(ハンドラルーチン)を修正してください。
2157	FPU disable	FPU が使用できない状態(SR.FD=1)で FPU 命令を実行しようとしました。エラーが発生しないようにプログラムを修正してください。
2158	Slot FPU disable	FPU が使用できない状態(SR.FD=1)で遅延スロットにある FPU 命令を実行しようとしました。 エラーが発生しないようにプログラムを修正してください。
2159	Instruction TLB Illegal LRU	命令 TLB の LRU の値が不正です。 設定を確認してください。
2160	Illegal PR bit	FPSCR の PR ビットが不正な状態で FPU 命令を実行しようとしました。 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2161	Illegal combination BSC register	BSC レジスタの設定が不正なエリアへアクセスしようとしました。 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2162	Illegal CCR2 set	CCR2 レジスタに不正な値を代入しようとしました。 エラーが発生しないようにデバッグ対象プログラムを修正してください。
2163	Interrupt exception	割込み例外が発生しました。
2164	Power on reset exception	パワーオンリセット例外が発生しました。
2165	Manual reset exception	マニュアルリセット例外が発生しました。
2997	(Seamless からのメッセージ)	Seamless からのウォーニングエラーメッセージです。
2998	(Seamless からのメッセージ)	Seamless からのエラーメッセージです。
2999	(Seamless からのメッセージ)	Seamless からのフェイタルエラーメッセージです。
3002	Incorrect object module format	ファイルのフォーマットが不正です。

番号	メッセージ	内容･対策
3003	Object module allocation	シミュレータ・デバッガで使うメモリを確保できません。 メモリの拡張、またはデバッグ対象プログラムの変更を行ってください。
3004	Object module not absolute format	ファイルのフォーマットが不正です。
3005	Incorrect object module cpu	他の CPU のデバッグ対象プログラムをロードしようとしました。
3030	Can't convert	変換できません。
3041	Not enough memory	メモリが足りません。
5995	MANUAL RESET	HW シミュレータからマニュアルリセット例外発生が通知されました。
5996	POWER-ON RESET	HW シミュレータからパワーオンリセット例外発生が通知されました。
5997	HW SIMULATOR WAS RESET	HW シミュレータはリセットされました。
5998	HW SIMULATOR WAS RESTARTED	HW シミュレータはリスタートされました。
5999	HW SIMULATOR IS EXITING	HW シミュレータは終了します。

6. ウィンドウ

シミュレータデバッガのメニューバーとそれに対応するプルダウンメニューの一覧を以下に示します。

表 6.1 **インフォメーションメッセージ一覧**

メニューバー	プルダウンメニュー	サブメニュー	機　　能
File	Auto Typing		ファイルからのデバッガコマンド入力
	Logging		デバッガコマンド結果のファイル出力
	Recording		シミュレータ・デバッガ操作手順の保存
	Replaying		保存されている操作手順の再生
	File Selection		ファイル名の選択
	File Load		ロード
	Load Status		デバッガ状態の回復
	File Save		セーブ
	Save Status		デバッガ状態の保存
	Quit		シミュレータ・デバッガの終了
View	Register		レジスタ表示および変更 （SH4 は浮動小数点レジスタ。制御レジスタ以外のレジスタ）
	Floating Point Register		SH4 の浮動小数点レジスタの表示および変更
	Control Register		SH4 の制御レジスタの表示および変更
	Disassemble		逆アセンブル表示
	Show Calls		関数呼び出し表示
	Source Files		ソースファイル名の表示およびソースファイルの表示
	Function List		関数名の表示およびソースファイルの表示
	Expression Value		式の値表示
	Localized Dump		メモリ内容の日本語対応
	Stack		スタックの内容表示
	Symbol List	Symbol	PC 位置での内容表示可能な変数名表示
		Object	現在スコープのオブジェクト名表示
		Class	現在スコープのクラス名表示
	Symbol Value	Symbol Value No.x	変数内容の表示および変更　x：１～４
	Memory Display	Memory Display No.x	メモリ内容の表示　x：１～４
	Analysis	Performance	実行パフォーマンス測定の設定、表示
		Stack	スタックトレースの設定、表示

メニューバー	プルダウンメニュー	サブメニュー	機　　能
Execute	Go		実行開始
	Exec Mode		実行モードの設定および表示
	Exttool		外部ツールの操作
Break	Break Point		ブレークポイントの設定、表示、解除
	Break Access		アクセスによるブレーク条件の設定、表示、解除
	Break Data		メモリ値によるブレーク条件の設定、表示、解除
	Break Register		レジスタ値によるブレーク条件の設定、表示、解除
	Break Sequence		実行順序によるブレーク条件の設定、表示、解除
Trace	Trace		トレース情報の表示
	Trace Condition		トレース条件の設定、トレースの開始、終了
Help	General Operation		シミュレータ・デバッガの操作説明
	Symbolic Input		アドレスのシンボリック指定の説明
	Command Name		デバッガコマンドの説明

7. CPU 情報ファイルの作成

7.1 CPU 情報ファイル作成プログラム CIA の機能

シミュレータ・デバッガは、マイコンごとにメモリマップに沿ってセクションをロードしたり、異なったメモリ種別をまたがってロードしていないかをチェックするため CPU 情報ファイルを使います。CPU 情報ファイルは CIA（CPU Information Analyzer）を用いて作成します。

CIA は次の３つの機能を持ちます。

(1) **CPU 情報ファイルの作成**

使用するマイコンの CPU のメモリマップ情報をファイルに作成します。

(2) **CPU 情報ファイルの内容表示**

作成済みの CPU 情報ファイルの内容を確認することができます。

(3) **CPU 情報の編集（削除／追加）**

作成済みの CPU 情報ファイルの内容を削除・追加機能を使って変更できます。

7.2 CIA の実行

CIA を起動するためのコマンド形式は次の通りです。

- SH-1、SH-2、SH-2E、SH-3、SH-3E の場合
  % ciash△<CPU情報ファイル名>(RET)
- SH-3DSP、SH-DSP シリーズの場合
  % ciashdsp△<CPU情報ファイル名>(RET)
- SH-4 シリーズの場合
  % ciash4△<CPU情報ファイル名>(RET)

＜CPU 情報ファイル名＞として新規または既存の CPU ファイル名を指定します。既存の CPU 情報ファイルを指定した場合は、出力用の CPU 情報ファイルを指定する様に要求します。＜CPU 情報ファイル名＞のファイル形式を省略した場合は、".cpu"を仮定します。

7.3 CIA の使用手順と選択メニュー

CIA の使用手順を以下に示します。

CIA 起動 → (1)モードの選択 → (2)コメント入力 → (3)メモリマップ設定処理 ’．’ → (4)編集処理 ’．’（終了） → CIA 終了

(1) モード選択

- ciash の場合
  SH-1、SH-2、SH-2E、SH-3、SH-3EからCPUを選択します。
- ciashdsp の場合
  SH-DSP、SH-DSPwithCache（SHDSPC）、SH-3DSP、SH-2DSPからCPUを選択します。
  SH-DSPの時は例外ベクタ内蔵メモリモードか例外ベクタ外部メモリモードかを選択します。
  SH-2DSPの時は内蔵ROM有効モードか無効モードかを選択します。
- ciash4 の場合
  モード選択はありません。

(2) コメント入力

コメントは CPU 情報の識別用として、文字列を指定することができます。コメントは、127 文字まで設定できます。

コメントは、新規ファイルの場合のみです。既存ファイルの場合は(4)の編集処理から始まります。

(3) メモリマップ設定

CPU 情報の入力メニューとして次の選択肢があり、’. ’（終了）が指定されるまでメモリマップの設定処理を繰り返します。

- ciash の場合
  0:ROM 1:EXTERNAL 2:RAM 3:I/O .:END
- ciashdsp の場合（SH-DSP または SH-3DSP 選択時）
  0:X-ROM 1:X-RAM 2:Y-ROM 3:X-RAM 4:I/O 5:EXTERNAL .:END
- ciashdsp の場合（SH-DSPC 選択時）
  0:X-RAM 1:Y-RAM 2:I/O 3:EXTERNAL 4:INTRAM .:END
- ciashdsp の場合（SH-2DSP 選択時）
  0:X-ROM 1:X-RAM 2:Y-ROM 3:X-RAM 4:I/O 5:EXTERNAL 6:INTROM
  7:INTRAM .:END
- ciash4 の場合
  0:NORMAL 1:MPX

ROM : ciash の場合の内蔵 ROM
RAM : ciash の場合の内蔵 RAM
INTROM : ciashdsp の場合の内蔵 ROM
INTRAM : ciashdsp の場合の内蔵 RAM
I/O : 内蔵 I/O
X-ROM : 内蔵 X-ROM
X-RAM : 内蔵 X-RAM
Y-ROM : 内蔵 Y-ROM
Y-RAM : 内蔵 Y-RAM
EXTERNAL : 外部メモリ
NORMAL : エリア 0 をノーマルメモリで使用する
MPX : エリア 0 をマルチプレクスメモリで使用する

(4) 編集処理

CPU 情報の編集用メニューとして、次の選択肢があります。

- ciash、ciashdsp の場合
  1:ADD 2:DELETE 3:COMMENT 4:CIA ABORT .:CIA END

  '1'(ADD)を選択した場合は、(3)のメモリマップ設定処理を行います。
  '2'(DELETE)を選択した場合は、消去したいアドレスの範囲を番号で入力します。
  '3'(COMMENT)を選択した場合は、新規のコメントを入力します。
  '4'(CIA ABORT)を選択した場合は、CPU情報ファイルに出力せずに、CIAの処理を終了します。
  '.'(CIA END)を選択した場合は、メモリマップ情報をCPU情報ファイルに出力し、正常にCIAの処理を終了します。

- ciash4 の場合
  1:MODIFY 2:MODIFY 3:COMMENT 4:CIA ABORT .:CIA END

  '1'(MODIFY)を選択した場合は、(3)のメモリマップ設定処理を行います。
  '2'(MODIFY)を選択した場合は、'1'と同様に(3)のメモリマップ設定処理を行います。
  '3'(COMMENT)を選択した場合は、新規のコメントを入力します。
  '4'(CIA ABORT)を選択した場合は、CPU情報ファイルに出力せずに、CIAの処理を終了します。
  '.'(CIA END)を選択した場合は、メモリマップ情報をCPU情報ファイルに出力し、正常にCIAの処理を終了します。

7.4 CIA の使用例

SH-3 を使用する場合の CIA の使用例を以下に示します。アンダーラインはユーザの入力部分です。

(1) CPU 情報ファイルの新規作成例

```
% ciash sh3

SH SERIES CIA Ver. 3.1(HS0700CICS3SM)
Copyright (C) Hitachi, Ltd. 1992,1998
Copyright (C) Hitachi ULSI Systems Co., Ltd. 1993,1998
Licensed Material of Hitachi, Ltd.

*** NEW FILE ***

   *** CPU MENU ***
   1:SH1   2:SH2   3:SH3   4:SH3E   5:SH2E
   ? 3
  COMMENT?     :SH3  CPU INFORMATION
```

```
            *** MAP MENU ***
            0:ROM   1:EXTERNAL   2:RAM   3:I/O   .:END
            ? 1
                * EXTERNAL START ADDRESS? 0
                        END  ADDRESS? fffff
                  STATE COUNT         ? 3
                  DATA BUS SIZE       ? 32
                * EXTERNAL START ADDRESS? .

            *** MAP MENU ***
            0:ROM   1:EXTERNAL   2:RAM   3:I/O   .:END
            ? .


                  ***** CPU INFORMATION *****
                  CPU : SH3
                  SH3  CPU INFORMATION
                  BIT SIZE : 32
                     No   Device    Start     End     State   Bus
                      1 : EXTERNAL : 00000000 - 000FFFFF  3      32

      ** EDIT MENU **
      1:ADD    2:DELETE   3:COMMENT  4:CIA ABORT  .:CIA END
      ? .

*** CIA COMPLETED ***
%
```

（説明）

(a) CIA 起動時に新規作成ファイルを指定します。

(b) CPU の種類を指定します。

(c) コメントを入力します。省略した場合は空白を表示します。128 文字以上入力した場合は、ウォーニングメッセージを出力し、128 文字目以降を無視します。

(d) 入力メニューに対し、メモリ種別を番号で入力します。

(e) 当該メモリ領域の先頭アドレスを 16 進数で入力します。

(f) 当該メモリ領域の終了アドレスを 16 進数で入力します。

(g) 当該メモリ領域のステート数を 10 進数で入力します。

(h) 当該メモリ領域のデータバス幅を 10 進数で入力します。

(i) 当該メモリ種別入力の入力を終了したい場合は'.'を指定します。

(j) 入力メニューを終了すると編集結果を表示します。

- CPU の種類
- コメント
- ビットサイズ
- マップ番号
- メモリ種別
- 先頭アドレス
- 終了アドレス
- ステート数
- データバス幅

(k) CIA の処理を正常に終了します。CIA 起動時に指定したファイル（sh3.cpu）にメモリマップ情報を出力します。

(2) CPU 情報ファイルの編集例

ここでは、外部メモリのアドレス範囲、データバス幅を変更します。

```
% ciash sh1.cpu

SH SERIES CIA Ver. 3.1(HS0700CICS3SM)
Copyright (C) Hitachi, Ltd. 1992,1998
Copyright (C) Hitachi ULSI Systems Co., Ltd. 1993,1998
Licensed Material of Hitachi, Ltd.
```

```
*** OLD FILE ***
  NEW CPU FILE NAME? sh1.cpu

                    ***** CPU INFORMATION *****
                    CPU : SH1
                    SH1 CPU INFORMATION
                    BIT SIZE : 32
                       No   Device    Start     End     State   Bus
                        1 : ROM AREA : 00000000 - 0000FFFF  1      32
                        2 : EXTERNAL : 01000000 - 010FFFFF  4      16
                        3 : RAM AREA : 0F000000 - 0F000FFF  1      32

        ** EDIT MENU **
        1:ADD    2:DELETE   3:COMMENT  4:CIA ABORT  .:CIA END
        ? 2
        DELETE MAP NUMBER? 2

                    ***** CPU INFORMATION *****
                    CPU : SH1
                    SH1 CPU INFORMATION
                    BIT SIZE : 32
                       No   Device    Start     End     State   Bus
                        1 : ROM AREA : 00000000 - 0000FFFF  1      32
                        2 : RAM AREA : 0F000000 - 0F000FFF  1      32

        ** EDIT MENU **
        1:ADD    2:DELETE   3:COMMENT  4:CIA ABORT  .:CIA END
        ? 1

              *** MAP MENU ***
              0:ROM   1:EXTERNAL   2:RAM   3:I/O   .:END
              ? 1
                  *  EXTERNAL START ADDRESS?  9000000
                         END   ADDRESS?  90fffff
                     STATE COUNT         ?  4
                     DATA BUS SIZE       ?  8
```

```
            *  EXTERNAL START ADDRESS?  .

        *** MAP MENU ***
        0:ROM    1:EXTERNAL    2:RAM    3:I/O    .:END
        ? .

              ***** CPU INFORMATION *****
              CPU : SH1
              SH1 CPU INFORMATION
              BIT SIZE : 32
                 No   Device     Start      End      State    Bus
                  1 : ROM AREA : 00000000 - 0000FFFF   1      32
                  2 : EXTERNAL : 09000000 - 090FFFFF   4      8
                  3 : RAM AREA : 0F000000 - 0F000FFF   1      32

    ** EDIT MENU **
    1:ADD    2:DELETE    3:COMMENT   4:CIA ABORT   .:CIA END
    ? .

*** CIA COMPLETED ***
%
```

（説明）

(a) CIA **起動時に編集対象のファイル名を入力します。ファイル形式を省略した場合は、**".cpu"**を仮定します。**

(b) **編集後の新規ファイル名を指定します。**(RET)**のみ入力した場合は、**(1)**と同じファイル名で出力します。ファイル形式のみ省略した場合は、**".cpu"**を仮定します。マップ情報を表示します。**

(c) **編集メニューで変更したい情報の削除をするため、**"DELETE"**を指定します。**

(d) **削除したい情報をマップ番号で指定します。削除した結果のマップ情報が表示されます。**

(e) **変更後の情報を入力するため、**"ADD"**を指定します。**

(f) **入力メニューが表示され、新規作成時と同様にメモリ種別を入力します。**

(g) "END"**を指定すると追加した結果のマップ情報が表示されます。**

7.5 CIA の制限事項一覧

表 7.1 に CIA の制限事項一覧を示します。CIA ではこれらの制限値を超える処理はできません。

表 7.1 シミュレータ・デバッガの機能と CIA の対応

項番	項目	制限値
1	入力ファイル形式	・CIA 出力の CPU 情報ファイル
2	ビットサイズ	・32 ビット固定
3	アドレス指定	・16 進数により指定のみ ・指定範囲はビットサイズによる
4	メモリ容量	・SH-DSP 選択時の X/Y-ROM/RAM のみ指定可 ・10 進数による指定のみ ・指定範囲は 1〜64
5	ステート数	・10 進数による指定のみ ・指定範囲は 1〜65535
6	データバス幅	・8,16,32,64 のいずれかのみ （64 は ciash4 のみ可）
7	コメントの長さ	・127 文字まで
8	MAP 情報数	・最大 65535 個

【注】 本シミュレータに付属する CIA で作成した CPU 情報ファイルは、モジュール間最適化ツールでは読み込むことができません。
モジュール間最適化ツールには、モジュール間最適化ツールに付属の CIA で作成した CPU 情報ファイルを使用してください。

8. 協調検証対応機能

本章では、Mentor Graphics 社の Seamless を用いた協調検証方法について説明します。なお、Seamless およびハードウェアシミュレータに関しては、該当製品のマニュアルをお読みください。

8.1 特長

シミュレータ・デバッガは、SuperH™ RISC engine マイコンの CPU シミュレーション機能およびデバッグ機能を持っており、SH-4、SH-2DSP では協調検証をサポートします。協調検証を行なうことによって、ハードウェアの完成していない初期の段階で C/C++言語やアセンブリ言語で作成されたプログラムを用いてシステムを評価することができます。

協調検証では下記のような機能を持ち、プログラムのテスト、およびデバッグを効率よく進めることができます。

- NMI、IRQ、IRL、タイマーによる割り込みのサポート
- バス幅、WAIT、IDLE のサポート
- タイマーのサポート

8.2 シミュレータ・デバッガの機能

協調検証でサポートするシミュレータ・デバッガの機能について説明します。

8.2.1 シミュレータ・デバッガのメモリ管理

(1) メモリリソースの確保

ハードウェアシミュレータで動作するメモリモデルを使用する場合でも、シミュレータ・デバッガにメモリリソースを設定する必要があります。メモリリソースは、MAP_SET コマンドで設定できます。

8.2.2 エンディアン

SH-4 のエンディアンは MD ピンの設定で決まります。ハードウェアシミュレータのデザイン上で MD ピンの High／Low を設定してください。

8.2.3 BSC（バスステートコントローラ）

BSC でバス幅、プログラマブルウェイト、アイドルサイクルを設定できます。
また SDRAM などのメモリ種別も設定できます。

8.2.4 INTC（割込みコントローラ）

協調検証では NMI、IRQ、IRL、タイマーによる割り込みをサポートします。

8.3 操作方法

協調検証を行なうためのシミュレータ・デバッガの操作方法について説明します。

8.3.1 はじめに

シミュレータ・デバッガを起動する前に Seamless およびハードウェアシミュレータを動作できる状態にしてください。Seamless およびハードウェアシミュレータの設定に関しては各製品のマニュアルを参照ください。

8.3.2 Seamless の設定、シミュレータ・デバッガの実行

はじめに Seamless を起動してください。
Seamless にはハードウェアシミュレータのための設定がしてあるものとします。
CVE ウィンドウで setup メニューの Software Simulator を選択し Setup Software Simulator ウィンドウの Invocation に sdsh2dsp または sdsh4 を指定してください。

8.3.3 Seamless に接続する

EXTTOOL コマンドで Seamless に接続します。

(例) Seamless に接続します。
:EXTTOOL CONNECT （RET）

【注】 Seamless との接続を終了した場合、再接続はできません。

8.3.4 プログラムのダウンロード

メモリにプログラムをダウンロードする方法には 3 種類あります。

(1) **シミュレータ・デバッガよりダウンロード**

FILE_LOAD コマンドでプログラムをロードしてください。

(例) プログラム test.abs をロードします。
:FILE_LOAD test.abs （RET）

(2) **ハードウェアシミュレータデバッガ経由でダウンロード**

ハードウェアシミュレータのデザイン上で、メモリのモデルに初期値を設定してください。

8.3.5 メモリリソースの確保

メモリモデルをアクセスするためにメモリリソースの確保を行ないます。
MAP_SET コマンドでメモリマップを設定してください。

(例) 0～FFFF 番地のﾒﾓﾘリソースを Read/Write 属性で確保します。
:MAP_SET 0 FFFF RW（RET）

【注】 メモリモデルをアクセスするためには必ずマップの設定をしてください。

8.3.6 ハードウェアシミュレータの停止

波形を表示する等、ハードウェアシミュレータの操作をする場合、EXTTOOL コマンドでハードウェアシミュレータを停止してください。

(例) ハードウェアシミュレータを停止します。
:EXTTOOL BREAK（RET）

8.4 注意事項

協調検証を行なうためのシミュレータ・デバッガの注意事項について説明します。

8.4.1 内蔵メモリ、X／Yメモリ

内蔵メモリ、X／Yメモリのサイズおよびアクセスサイクル数は CPU 情報ファイルで設定してください。また、例外ベクタのモード（内蔵メモリモードまたは外部メモリモード）を CPU 情報ファイルで設定してください。

CPU 情報ファイルは CPU 情報ファイル作成プログラム（ＣＩＡ）を使用して作成してください。CPU 情報ファイル作成プログラムについては**「７章　CPU 情報ファイルの作成」**を参照してください。

作成した CPU 情報ファイルはシミュレータ・デバッガ起動時の cpu オプションで指定してください。

（例）CPU 情報ファイル（sdsh2dsp.cpu）を指定して起動します。
%sdsh2dsp –cpu=sdsh2dsp.cpu

SuperH RISC engine シミュレータ・デバッガ
ユーザーズマニュアル

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

ADJ-702-277A