

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

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社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、ディスクリット半導体、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

## 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注意ください。

### 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すとは、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。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を負いません。
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、その他お気づ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。

# H8S,H8/300シリーズ 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

HSS008SDIW3SJ

## ご注意

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

## 商標

Microsoft®および Windows®、 は、米国マイクロソフトコーポレーションの米国およびその他の国における登録商標です。

IBM PC は、米国 IBM 社により管理されている計算機の名称です。

ELF/DWARF は、the Tool Interface Standards Committee で開発された、オブジェクトフォーマットの名称です。

本マニュアルで使用されているすべての製品名またはブランド名は、それぞれの会社の商標または登録商標です。

---

## まえがき

---

日立デバッグインタフェースおよびシミュレータ・デバッガの使用前に本マニュアルを必ずお読みください。

ユーザズマニュアルは、なくさないよう保管してください。

本システムの使用方法を十分理解してから、使用してください。

## 本書について

本書では、日立のマイクロコンピュータの開発ツールに対応した Hitachi Debugging Interface (HDI) およびシミュレータ・デバッガの使用法について説明します。次章、「はじめに」では、デバッグインタフェースおよびシミュレータ・デバッガについて簡単に紹介し、おもな特長を示します。

次の各章、「システムの概要」、「シミュレータ・デバッガの機能」、「メニュー」、「ウィンドウおよびダイアログボックス」、「コマンドライン」および「メッセージ一覧」は、それぞれの操作方法や利便性についてのリファレンスです。

「プログラムの表示」、「メモリの操作」、「プログラムの実行」、「プログラムの停止」、「変数の表示」、「オーバーレイ機能」、「関数の設定」および「ユーザインタフェースの構成」は、HDIを使用したデバッグに関する“how to”ガイドです。

本マニュアルは、動作環境を IBM PC 上の Microsoft® Windows®95 operating system 英語版として記述しています。



## 前提事項

本書では、読者の方々に、C/C++プログラミング言語およびデバッグ対象プロセッサ用のアセンブラモニターについての十分な知識があること、および Microsoft® Windows®アプリケーションの使用経験があることを前提としています。

## マニュアルの規約

本書には、以下の表記上の規約があります。

表 1 表記上の規約

表 記	意 味
[Menu->Menu Option]	'->'の付いた太字の表記は、メニューオプションを指定する場合に使用します (例： [File->Save As...])。
FILENAME.C	大文字表記の名前は、ファイル名を指定する場合に使用します。
"enter this string"	入力必須のテキストを指定する場合に使用します(引用符" "は除く)。
キー + キー	必要なキーを押すよう指示する場合に使用します。たとえば、CTRL+N は、CTRL キーを押したまま N キーを押すことを意味します。
☞ ("手順"記号)	この記号は、必ず左端に記載されます。この記号が使われる場合は、すぐ右に実行手順が記載されていることを意味します。

---

# 目次

---

第 1 章	はじめに	
1.1	おもな特長	1
1.2	特長	1
1.3	デバッグ対象プログラム	2
1.4	シミュレーション範囲	4
第 2 章	システムの概要	
2.1	ユーザインタフェース	5
2.2	データ入力	5
2.2.1	演算子	5
2.2.2	データ形式	5
2.2.3	精度	6
2.2.4	式の例	6
2.2.5	シンボル形式	6
2.2.6	シンボル指定の例	6
2.3	ヘルプ	6
2.3.1	コンテキスト依存型ヘルプ	7
第 3 章	シミュレータ・デバッガの機能	
3.1	シミュレータ・デバッガと CPU の対応	9
3.2	シミュレータ・デバッガのメモリ管理	9
3.3	命令実行リセット処理	10
3.4	例外処理	10
3.5	H8S/2600CPU 特殊機能	11
3.6	制御レジスタ	11
3.7	トレース	12
3.8	標準入出力およびファイル入出力処理	13
3.9	命令実行サイクル数の計算	22
3.10	ブレイク条件	23
3.11	浮動小数点データ	25
3.12	関数呼び出し履歴の表示	26
第 4 章	メニュー	
4.1	File	27
4.1.1	New Session	27
4.1.2	Load Session	27
4.1.3	Save Session	27

4.1.4	Save Session As.....	28
4.1.5	Load Program.....	28
4.1.6	Initialize.....	28
4.1.7	Exit .....	28
4.2	Edit .....	28
4.2.1	Cut .....	28
4.2.2	Copy .....	28
4.2.3	Paste .....	28
4.2.4	Find... ..	29
4.2.5	Evaluate.....	29
4.3	View .....	29
4.3.1	Breakpoints.....	29
4.3.2	Command Line.....	29
4.3.3	Disassembly.....	29
4.3.4	Labels .....	29
4.3.5	Locals .....	29
4.3.6	Memory.....	29
4.3.7	Performance Analysis.....	29
4.3.8	Profile-List .....	30
4.3.9	Profile-Tree .....	30
4.3.10	Registers.....	30
4.3.11	Source.....	30
4.3.12	Status .....	30
4.3.13	Trace.....	30
4.3.14	Watch .....	30
4.3.15	Localized Dump... ..	30
4.3.16	Simulated I/O .....	30
4.3.17	Stack Trace.....	30
4.3.18	External Tool.....	31
4.4	Run .....	31
4.4.1	Reset CPU .....	31
4.4.2	Go .....	31
4.4.3	Reset Go .....	31
4.4.4	Go To Cursor.....	31
4.4.5	Set PC To Cursor.....	31
4.4.6	Run... ..	31
4.4.7	Step In .....	31
4.4.8	Step Over.....	32
4.4.9	Step Out.....	32
4.4.10	Step.....	32
4.4.11	Halt.....	32
4.5	Memory .....	32
4.5.1	Refresh .....	32
4.5.2	Load.....	32
4.5.3	Save.....	32
4.5.4	Verify... ..	32

4.5.5	Test.....	33
4.5.6	Fill.....	33
4.5.7	Copy.....	33
4.5.8	Compare.....	33
4.5.9	Configure Map.....	33
4.5.10	Configure Overlay.....	33
4.6	Setup .....	33
4.6.1	Status Bar .....	33
4.6.2	Options.....	34
4.6.3	Radix .....	34
4.6.4	Customize.....	34
4.6.5	Configure Platform.....	34
4.7	Window .....	34
4.7.1	Cascade .....	34
4.7.2	Tile .....	34
4.7.3	Arrange Icons .....	35
4.7.4	Close All.....	35
4.8	Help .....	35
4.8.1	Index.....	35
4.8.2	Using Help.....	35
4.8.3	Search for Help on.....	35
4.8.4	About HDI.....	35

## 第 5 章 ウィンドウおよびダイアログボックス

5.1	Breakpoints.....	37
5.1.1	Add.....	37
5.1.2	Edit.....	38
5.1.3	Delete .....	38
5.1.4	Delete All .....	38
5.1.5	Disable/Enable.....	38
5.1.6	Go to Source.....	38
5.2	Set Break ダイアログボックス.....	39
5.3	Break Sequence ダイアログボックス.....	40
5.4	Command Line .....	41
5.4.1	Set Batch File.....	41
5.4.2	Play.....	41
5.4.3	Set Log File.....	41
5.4.4	Logging .....	42
5.4.5	Select All.....	42
5.4.6	Copy .....	42
5.5	Disassembly.....	42
5.5.1	Copy .....	43
5.5.2	Set Address.....	43
5.5.3	Go To Cursor.....	43
5.5.4	Set PC Here .....	44
5.5.5	Instant Watch.....	44

5.5.6	Add Watch.....	44
5.5.7	Go to Source.....	44
5.6	Labels .....	44
5.6.1	Add.....	45
5.6.2	Edit.....	45
5.6.3	Find.....	46
5.6.4	Find Next.....	46
5.6.5	View Source .....	46
5.6.6	Copy .....	46
5.6.7	Delete .....	47
5.6.8	Delete All .....	47
5.6.9	Load.....	48
5.6.10	Save .....	48
5.6.11	Save As.....	48
5.7	Locals .....	49
5.7.1	Copy .....	49
5.7.2	Edit Value.....	49
5.7.3	Radix .....	49
5.8	Memory .....	50
5.8.1	Refresh .....	50
5.8.2	Load.....	50
5.8.3	Save.....	50
5.8.4	Test.....	50
5.8.5	Fill.....	51
5.8.6	Copy.....	51
5.8.7	Compare.....	51
5.8.8	Search.....	51
5.8.9	Set Address.....	51
5.8.10	ASCII/Byte/Word/Long/Single Float/Double Float .....	51
5.9	Performance Analysis.....	52
5.9.1	Add Range.....	52
5.9.2	Edit Range .....	52
5.9.3	Delete Range .....	52
5.9.4	Reset Counts/Times.....	53
5.9.5	Delete All Ranges.....	53
5.9.6	Enable Analysis.....	53
5.10	Performance Option ダイアログボックス .....	53
5.11	Registers .....	54
5.11.1	Copy .....	54
5.11.2	Edit.....	54
5.11.3	Toggle Bit.....	54
5.12	Source .....	55
5.12.1	Copy .....	56
5.12.2	Find.....	56
5.12.3	Set Address.....	56

5.12.4	Set Line.....	56
5.12.5	Go To Cursor.....	56
5.12.6	Set PC Here.....	56
5.12.7	Instant Watch.....	56
5.12.8	Add Watch.....	56
5.12.9	Go to Disassembly.....	57
5.13	System Status.....	57
5.13.1	Update.....	58
5.13.2	Copy.....	58
5.14	Trace.....	58
5.14.1	Find.....	59
5.14.2	Find Next.....	59
5.14.3	Filter.....	59
5.14.4	Acquisition.....	59
5.14.5	Halt.....	59
5.14.6	Restart.....	59
5.14.7	Snapshot.....	59
5.14.8	Clear.....	59
5.14.9	Save.....	59
5.14.10	View Source.....	59
5.14.11	Trim Source.....	60
5.15	Trace Acquisition ダイアログボックス.....	60
5.16	Trace Search ダイアログボックス.....	61
5.17	Watch.....	61
5.17.1	Copy.....	62
5.17.2	Delete.....	62
5.17.3	Delete All.....	62
5.17.4	Add Watch.....	62
5.17.5	Edit Value.....	62
5.17.6	Radix.....	62
5.18	System Configuration ダイアログボックス.....	63
5.19	Memory Map Modify ダイアログボックス.....	64
5.20	Memory Map ダイアログボックス.....	65
5.21	System Memory Resource Modify ダイアログボックス.....	66
5.22	Control Registers.....	66
5.23	SYSCR ダイアログボックス.....	67
5.24	Simulated I / O.....	67
5.25	Stack Trace.....	68
5.25.1	Copy.....	68
5.25.2	Go to Source.....	68
5.25.3	View Setting.....	69
5.26	Profile-List.....	70
5.26.1	View Source.....	70
5.26.2	View Profile-Tree.....	70

5.26.3	View Profile-Chart .....	70
5.26.4	Enable Profiler.....	70
5.26.5	Find... ..	71
5.26.6	Clear Data.....	71
5.26.7	Output Profile Information File.....	71
5.26.8	Output Text File... ..	71
5.26.9	Select Data.....	71
5.26.10	Setting... ..	71
5.27	Profile-Tree.....	73
5.27.1	View Source .....	73
5.27.2	View Profile-List.....	73
5.27.3	View Profile-Chart .....	73
5.27.4	Enable Profiler.....	74
5.27.5	Find... ..	74
5.27.6	Find Data... ..	74
5.27.7	Clear Data.....	74
5.27.8	Output Profile Information File... ..	75
5.27.9	Output Text File... ..	75
5.27.10	Select Data.....	75
5.27.11	Setting... ..	75
5.28	Profile-Chart .....	76
5.28.1	Expands Size .....	76
5.28.2	Reduces Size .....	76
5.28.3	View Source .....	76
5.28.4	View Profile-List.....	76
5.28.5	View Profile-Tree.....	76
5.28.6	View Profile-Chart .....	77
5.28.7	Enable Profiler.....	77
5.28.8	Clear Data.....	77
5.28.9	Multiple View.....	77
5.28.10	Output Profile Information File.....	78
第 6 章	コマンドライン.....	79
第 7 章	メッセージ一覧	
7.1	インフォメーションメッセージ.....	105
7.2	エラーメッセージ .....	105
第 8 章	プログラムの表示	
8.1	デバッグを行うためのコンパイル.....	107
8.2	コードの表示 .....	107
8.2.1	ソースコードの表示 .....	107
8.2.2	アセンブラコードの表示 .....	108
8.2.3	アセンブラコードの変更 .....	109
8.3	ラベルの表示 .....	110
8.3.1	ラベルのリスト表示 .....	110
8.3.2	Source ウィンドウまたは Disassembly ウィンドウでのラベルの追加.....	111

8.4	特定アドレスのプログラム表示.....	111
8.4.1	現在の PC のプログラム表示 .....	112
8.5	テキストの検索 .....	112
<b>第 9 章 メモリの操作</b>		
9.1	メモリ領域の表示 .....	113
9.1.1	ASCII でのメモリ表示.....	114
9.1.2	バイト単位でのメモリ表示 .....	114
9.1.3	ワード単位でのメモリ表示 .....	114
9.1.4	ロングワード単位でのメモリ表示 .....	114
9.1.5	単精度浮動小数点数形式でのメモリ表示.....	114
9.1.6	倍精度浮動小数点数形式でのメモリ表示.....	114
9.1.7	別領域のメモリ表示 .....	115
9.2	メモリ内容の変更 .....	115
9.2.1	簡易変更方式 .....	115
9.2.2	詳細変更方式 .....	115
9.2.3	メモリ範囲の選択 .....	116
9.3	メモリ内の値の検索 .....	116
9.4	メモリ領域に値を埋める .....	117
9.4.1	範囲を埋める .....	117
9.5	メモリ領域の転送 .....	118
9.6	メモリ領域の保存 .....	118
9.7	メモリ領域のロード .....	119
9.8	メモリ領域のペリファイ .....	120
<b>第 10 章 プログラムの実行</b>		
10.1	リセットからの実行 .....	121
10.2	連続実行.....	121
10.3	カーソル位置まで実行 .....	122
10.4	複数ポイントまで実行 .....	122
10.5	シングルステップ .....	122
10.5.1	関数内の各命令をステップ実行 .....	123
10.5.2	関数全体を 1 ステップで実行 .....	123
10.6	関数の実行停止 .....	123
10.7	複数のステップ .....	123
<b>第 11 章 プログラムの停止</b>		
11.1	実行の停止 .....	125
11.2	PC ブレークポイント .....	125
11.3	Breakpoints ウィンドウ .....	126
11.3.1	ブレークポイントの追加 .....	127
11.3.2	ブレークポイントの変更 .....	127
11.3.3	ブレークポイントの削除 .....	127
11.3.4	全ブレークポイントの削除 .....	127



11.4	ブレークポイントを無効にする.....	127
11.4.1	ブレークポイントを無効にする.....	127
11.4.2	ブレークポイントを有効にする.....	128
11.5	テンポラリブレークポイント.....	128
<b>第 12 章 変数の表示</b>		
12.1	ツールチップウォッチ.....	131
12.2	インスタントウォッチ.....	131
12.3	ウォッチ項目の使用.....	132
12.3.1	ウォッチ項目の追加.....	132
12.3.2	ウォッチ項目の拡張.....	133
12.3.3	ウォッチ項目の表示基数の変更.....	134
12.3.4	ウォッチ項目の値の変更.....	134
12.3.5	ウォッチ項目の削除.....	135
12.4	ローカル変数の表示.....	135
12.5	レジスタの表示.....	136
12.5.1	ビットレジスタの拡張.....	136
12.5.2	レジスタ内容の変更.....	137
<b>第 13 章 オーバーレイ機能</b>		
13.1	セクショングループの表示.....	139
13.2	セクショングループの設定.....	140
<b>第 14 章 関数の設定</b>		
14.1	関数の表示.....	141
14.2	関数の設定.....	142
14.2.1	関数の選択.....	142
14.2.2	関数の削除.....	142
14.2.3	関数の設定.....	142
<b>第 15 章 ユーザインタフェースの構成</b>		
15.1	ウィンドウの配置.....	143
15.1.1	ウィンドウの最小化.....	143
15.1.2	アイコンの整列.....	144
15.1.3	ウィンドウのタイル表示.....	145
15.1.4	ウィンドウのカスケード表示.....	145
15.2	現在開いているウィンドウの検索.....	145
15.2.1	次のウィンドウの検索.....	145
15.2.2	特定のウィンドウの検索.....	145
15.3	ステータスバーの表示 / 非表示.....	146
15.4	ツールバーのカスタマイズ.....	146
15.4.1	全体概要.....	147
15.4.2	ツールバーのカスタマイズ.....	148
15.4.3	ボタンカテゴリ.....	148
15.4.4	ツールバーへのボタンの追加.....	148

15.4.5	ツールバーのボタンの位置変更 .....	149	
15.4.6	ツールバーからのボタンの削除 .....	149	
15.5	フォントのカスタマイズ .....	149	
15.6	ファイルフィルターのカスタマイズ .....	150	
15.7	セッションの保存 .....	151	
15.8	セッションのロード .....	151	
15.9	HDI オプションの設定 .....	152	
15.10	デフォルト基数の設定 .....	154	
付録 A. システムモジュール			
A.1	グラフィカルユーザインタフェース .....	155	
A.2	オブジェクト DLL .....	156	
A.3	CPU DLL .....	156	
A.4	ターゲット DLL .....	156	
付録 B. GUI コマンド一覧 .....			157
付録 C. シンボルファイルのフォーマット .....			161

---

## 図表目次

---

### 図目次

図 1-1	デバッグ対象プログラム作成方法	3
図 3-1	入出力機能の説明形式	14
図 5-1	Breakpoints ウィンドウ	37
図 5-2	Set Break ダイアログボックス	39
図 5-3	Break Sequence ダイアログボックス	40
図 5-4	Command Line ウィンドウ	41
図 5-5	Disassembly ウィンドウ	42
図 5-6	Labels ウィンドウ	44
図 5-7	Add Label ダイアログボックス	45
図 5-8	Edit Label ダイアログボックス	45
図 5-9	Find Label Containing ダイアログボックス	46
図 5-10	ラベル削除確認メッセージボックス	47
図 5-11	Confirming All Label Deletion メッセージボックス	47
図 5-12	Load Symbols ダイアログボックス	48
図 5-13	Locals ウィンドウ	49
図 5-14	Memory ウィンドウ	50
図 5-15	Performance Analysis ウィンドウ	52
図 5-16	Performance Option ダイアログボックス	53
図 5-17	Registers ウィンドウ	54
図 5-18	Source ウィンドウ	55
図 5-19	System Status ウィンドウ	57
図 5-20	Trace ウィンドウ	58
図 5-21	Trace Acquisition ダイアログボックス	60
図 5-22	Trace Search ダイアログボックス	61
図 5-23	Watch ウィンドウ	61
図 5-24	System Configuration ダイアログボックス	63
図 5-25	Memory Map Modify ダイアログボックス	64
図 5-26	Memory Map ダイアログボックス	65
図 5-27	System Memory Resource Modify ダイアログボックス	66
図 5-28	Control Registers ウィンドウ	66
図 5-29	SYSCR ダイアログボックス	67

☒ 5-30	Simulated I/O ウィンドウ.....	67
☒ 5-31	Stack Trace ウィンドウ.....	68
☒ 5-32	Stack Trace Setting ダイアログボックス.....	69
☒ 5-33	Profile-List ウィンドウ.....	70
☒ 5-34	Profiler および Analysis の同時設定不可警告メッセージボックス.....	71
☒ 5-35	Setting Profile-List ダイアログボックス.....	72
☒ 5-36	Profile-Tree ウィンドウ.....	73
☒ 5-37	Profiler および Analysis の同時設定不可警告メッセージボックス.....	74
☒ 5-38	Find Data ダイアログボックス.....	74
☒ 5-39	Setting Profile-Tree ダイアログボックス.....	75
☒ 5-40	Profile-Chart ウィンドウ.....	76
☒ 5-41	Profiler および Analysis の同時設定不可警告メッセージボックス.....	77
☒ 8-1	Source ウィンドウ.....	108
☒ 8-2	Disassembly ウィンドウ.....	109
☒ 8-3	Assembler ダイアログボックス.....	109
☒ 8-4	Labels ウィンドウ.....	110
☒ 8-5	Label ダイアログボックス.....	111
☒ 8-6	Set Address ダイアログボックス.....	111
☒ 8-7	Find ダイアログボックス.....	112
☒ 9-1	Open Memory Window ダイアログボックス.....	113
☒ 9-2	Memory ウィンドウ(バイト単位).....	113
☒ 9-3	Set Address ダイアログボックス.....	115
☒ 9-4	Edit ダイアログボックス.....	116
☒ 9-5	Search Memory ダイアログボックス.....	116
☒ 9-6	Fill Memory ダイアログボックス.....	117
☒ 9-7	Copy Memory ダイアログボックス.....	118
☒ 9-8	Save Memory As ダイアログボックス.....	118
☒ 9-9	Load Memory ダイアログボックス.....	119
☒ 9-10	Verify S-Record File with Memory ダイアログボックス.....	120
☒ 10-1	PC が示すアドレス値に対応する行の強調表示.....	121
☒ 10-2	Step Program ダイアログボックス.....	123
☒ 11-1	プログラムブレークポイントの設定.....	126
☒ 11-2	Breakpoints ウィンドウ.....	126
☒ 11-3	Run Program ダイアログボックス.....	128
☒ 12-1	Tooltip Watch.....	131
☒ 12-2	Instant Watch ダイアログボックス.....	132
☒ 12-3	Add Watch ダイアログボックス.....	133
☒ 12-4	Watch ウィンドウ.....	133

図 12-5	ウォッチ項目の拡張	134
図 12-6	Edit Value ダイアログボックス	134
図 12-7	Locals ウィンドウ	135
図 12-8	Registers ウィンドウ	136
図 12-9	ビットレジスタの拡張	137
図 12-10	Register ダイアログボックス	138
図 13-1	Overlay ダイアログボックス (表示時)	139
図 13-2	Overlay ダイアログボックス (アドレス範囲選択時)	139
図 13-3	Overlay ダイアログボックス (優先セクショングループ選択時)	140
図 14-1	Select Function ダイアログボックス	141
図 15-1	ウィンドウの最小化	143
図 15-2	Disassembly ウィンドウのアイコン	143
図 15-3	整列前	144
図 15-4	整列後	144
図 15-5	ウィンドウの選択	146
図 15-6	Customize ダイアログボックス (Toolbars シート)	147
図 15-7	Customize ダイアログボックス (Commands シート)	148
図 15-8	Font ダイアログボックス	149
図 15-9	Customize File Filter ダイアログボックス	150
図 15-10	セッション名の表示	151
図 15-11	HDI Options (Session) ダイアログボックス	152
図 15-12	HDI オプション (Confirmation) ダイアログボックス	153
図 15-13	HDI オプション (Viewing) ダイアログボックス	153
図 15-14	基数の設定	154
図 A-1	HDI システムモジュール	155

## 表目次

表 3-1	プラットフォームと CPU の対応	9
表 3-2	メモリ種別	9
表 3-3	入出力機能一覧	13
表 3-4	ブレーク条件成立時の処理	23
表 3-5	シミュレーションエラー一覧	24
表 3-6	シミュレーションエラー停止時のレジスタ	24
表 6-1	コマンド一覧	79
表 7-1	インフォメーションメッセージ一覧	105
表 7-2	エラーメッセージ一覧	105

---

# 1. はじめに

---

Hitachi Debugging Interface (HDI)は、日立のマイクロコンピュータ用に、C/C++言語およびアセンブリ言語で書かれたアプリケーションの開発およびデバッグを簡単に行うためのグラフィカルユーザインタフェースです。アプリケーションを実行するデバッグプラットフォームのアクセス、計測、および変更に関して、HDIは高機能でしかも直観的な手段を提供することを目的としています。

## 1.1 おもな特長

- Windows®グラフィカルユーザインタフェースを用いたデバッグ
- 直観的なユーザインタフェース
- オンラインヘルプ
- 共通した表示と操作性

【注】 HDIはWindows®3.1では動作しません。

シミュレータ・デバッガは、H8S、H8/300シリーズマイコンの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、H8/300、H8/300L、H8/300H、H8S/2600、およびH8S/2000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・デバッガを利用することによって、C/C++言語やアセンブリ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。

また、当社製の下記ソフトウェアと共に使用することにより、プログラムの開発工数を削減することができます。

- HEW (Hitachi Embedded Workshop)
- H8S、H8/300シリーズC/C++コンパイラ
- H8S、H8/300シリーズクロスアセンブラ
- 最適化リンケージエディタ

## 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) ホスト計算機上で動作するので、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) シミュレーション時に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。
  - H8S、H8/300シリーズの各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
  - プロファイルデータ取得、および関数単位のパフォーマンス測定

## 1. はじめに

---

- 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) Windows®上で動作し、ブレークポイント・メモリマップ・パフォーマンス・トレースをダイアログボックス上で設定することができます。H8S、H8/300マイコンの各々のメモリマップに対応した環境設定もダイアログボックス上で行うことができます。

### 1.3 デバッグ対象プログラム

シミュレータ・デバッガでは、ELF/DWARF フォーマット、および S-type フォーマットのロードモジュールがデバッグ可能です。これらのロードモジュールをデバッグ対象プログラムと呼びます。デバッグ対象プログラム作成方法を図 1-1 に示します。

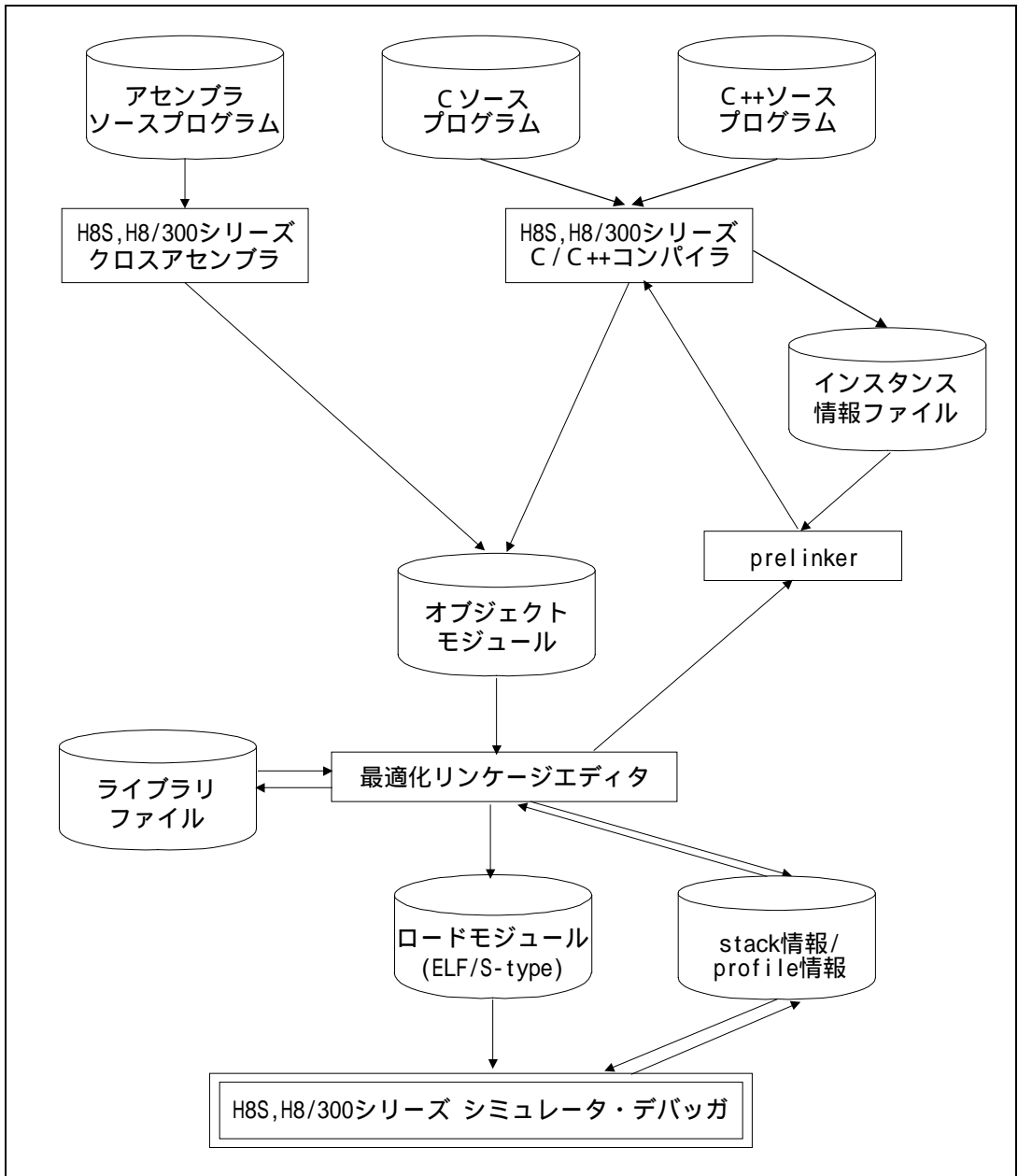


図1-1 デバッグ対象プログラム作成方法



## 1.4 シミュレーション範囲

- (1) シミュレータ・デバッガは、H8S、H8/300シリーズマイコンの下記機能をサポートしていません。
- 全実行命令
  - 例外処理
  - レジスタ
  - 全アドレス空間
- (2) シミュレータ・デバッガは H8S、H8/300シリーズ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、H8S、H8/300シリーズ用エミュレータを使用してデバッグしてください。
- デュアルポート RAM
  - タイマ
  - パルス幅変換器 (PWM)
  - シリアルコミュニケーションインタフェース (SCI)
  - A/D 変換器
  - I/O ポート
  - 割り込みコントローラ

---

## 2. システムの概要

---

HDI はモジュール方式のソフトウェアシステムで、それぞれに独立したモジュールを使用します。これらのモジュールは汎用グラフィカルユーザインタフェースへリンクされており、これにより、システムを構成するモジュールに関係なく共通した操作性が得られます。

### 2.1 ユーザインタフェース

HDI グラフィカルユーザインタフェースは、ユーザにデバッグプラットフォームを提供し、システムの設定および変更を可能にする Windows®対応のアプリケーションです。Windows®アプリケーションの詳しい操作方法は、Windows®標準のユーザズマニュアルを参照してください。

### 2.2 データ入力

ダイアログボックスまたはフィールドに数値を入力する場合には、単純な数値だけでなく式も入力できます。この式には、符号を含めたり、C/C++言語の演算子を使用することができます。C/C++言語のデバッグをサポートする ELF/DWARF フォーマットのオブジェクト DLL が使われている場合に、配列や構造体などの C/C++言語機能を使用できます。

一部のダイアログボックスでは、終了アドレスを入力する際に値の前に+符号を付けることによりアドレス範囲を入力することが可能です。

この場合、+符号を付けて入力された値と先頭アドレスの和が、終了アドレスになります。

#### 2.2.1 演算子

以下の C/C++言語演算子を使用できます。

`+, -, *, /, &, |, ^, ~, !, >>, <<, %, (, ), <, >, <=, >=, ==, !=, &&, ||`

#### 2.2.2 データ形式

接頭コードのないデータ値は、[Setup->Radix]メニューオプションで設定されたデフォルトの基数を使用していると見なします。ただし、回数を入力するところは例外で 10 進数の値をデフォルトにとります。

名前にはシンボルを使用できます。また、一重引用符で囲めば、ASCII 文字列を入力できます (例: `demo`)。

次の接頭コードを使用して、基数を指定できます。

O` 8 進  
B` 2 進  
D` 10 進  
H` 16 進  
0x` 16 進

## 2. システムの概要

---

先頭コードに#文字を使用してレジスタ名を指定すると、そのレジスタの内容を使用できます。次に例を示します。

```
#R1, #ER3, #R4L
```

### 2.2.3 精度

式の評価では、すべての数値演算は 32 ビット(符号付き)を使用して行われます。32 ビットを超える値はすべて切り捨てられます。

### 2.2.4 式の例

```
Buffer_start + 0x1000
#R1 | B'10001101
((pointer + (2 * increment_size)) & H'FFFF0000) >> D'15
!(flag ^ #ER4)
```

### 2.2.5 シンボル形式

シンボルには、C/C++ベースの指定ができます。これにより、C/C++言語ソースと同様な記述でシンボルの参照ができます。また、シンボルにはキャスト演算子をつけることができます。これにより、型変換後のデータを参照することができます。ただし、以下のような制限事項があります。

- ポインタ指定は 4 レベルまで可能です
- 配列指定は 3 次元まで可能です
- typedef 名は使用できません

### 2.2.6 シンボル指定の例

Object.value	: メンバの直接参照指定(C/C++)
p_Object->value	: メンバの間接参照指定(C/C++)
Class::value	: クラス指定付きメンバ参照指定(C++)
*value	: ポインタ指定(C/C++)
array[0]	: 配列指定(C/C++)
Object.*value	: メンバへのポインタ参照指定(C++)
::g_value	: グローバル変数参照指定(C/C++)
Class::function(short)	: メンバ関数指定(C++)
(struct STR)*value	: キャスト指定(C/C++)

## 2.3 ヘルプ

HDI には、Windows<sup>®</sup>標準のコンテキスト依存型ヘルプシステムが備わっています。ヘルプを使用すると、デバッグシステムの使用法に関するオンライン情報を表示します。

F1 キーを押すか、Help メニューを介して、ヘルプを呼び出すことができます。

さらに、一部のウィンドウおよびダイアログボックスには、専用のヘルプボタンがあります。

### 2.3.1 コンテキスト依存型ヘルプ

HDI で、特定の項目に関するヘルプを得るには、ヘルプカーソルを使用します。ヘルプカーソルを有効状態にするには、SHIFT+F1 キーを押すか、または、ツールバー上のボタンをクリックします。

SHIFT+F1 キーを押すと、カーソルが変化して、“?”マークが現れます。次に、ヘルプを必要とする項目上でクリックすると、適切な内容のヘルプが開きます。

## 2. システムの概要

---

---

## 3. シミュレータ・デバッガの機能

---

本章では、H8S、H8/300 シリーズ シミュレータ・デバッガの機能について説明します。

### 3.1 シミュレータ・デバッガと CPU の対応

シミュレータ・デバッガのデバッグプラットフォームと CPU の対応を表 3-1 に示します。

ご使用になる CPU に合わせてプラットフォームを選択してください。プラットフォームの選択は、Select Session ダイアログボックスで行います。

表3-1 プラットフォームと CPU の対応

デバッグ プラットフォーム名	対応 CPU
H8/300 Simulator	H8/300
H8/300L Simulator	H8/300L
H8/300HA Simulator	H8/300Hアドバンスモード
H8/300HN Simulator	H8/300Hノーマルモード
H8S/2600A Simulator	H8S/2600アドバンスモード
H8S/2600N Simulator	H8S/2600ノーマルモード
H8S/2000A Simulator	H8S/2000アドバンスモード
H8S/2000N Simulator	H8S/2000ノーマルモード

### 3.2 シミュレータ・デバッガのメモリ管理

#### (1) メモリマップの設定

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計算に使用します。シミュレータ・デバッガでは、表 3-2 に示すメモリ種別をサポートしています。

表3-2 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	可能
内蔵 RAM	可能
外部メモリ	可能
内蔵 I/O	不可能
EEPROM	可能

### 3. シミュレータ・デバッガの機能

---

メモリマップは、System Configuration ダイアログボックス上で設定することができ、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。設定できる項目は次の通りです。

- メモリ種別
- メモリ領域の先頭位置、終了位置
- メモリアクセスのサイクル数
- メモリのデータバス幅

設定できるメモリ種別は、CPU によって異なります。詳細は、「5.18 System Configuration ダイアログボックス」を参照してください。なお、デバッグ対象プログラムは、内蔵 I/O 空間を除くすべてのメモリで実行可能です。

#### (2) 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して実行させるため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、System Memory Resource Modify ダイアログボックスで設定できます。設定できる項目は以下の通りです。

- 開始アドレス
- 終了アドレス
- アクセス種別

アクセス種別は、読み書き可能、読み出しのみ可能、書き込みのみ可能があります。

デバッグ対象プログラムで、読み出しのみ可能メモリへ書き込みを行う等の不正なアクセスを行ったときはエラーとなるので、誤ったメモリアクセスを検出することができます。

EEPROM については、他のメモリと異なりアクセス種別が読み出しのみ可能な場合でも、EEPROM 命令により書き込みできます。反対に書き込み可能でも、EEPROM 命令以外では書き込みできません。

## 3.3 命令実行リセット処理

シミュレータ・デバッガでは、以下の場合に命令実行数および命令実行サイクル数をリセットします。

- (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Run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

## 3.4 例外処理

シミュレータ・デバッガでは、TRAPA 命令 (H8/300H、H8S シリーズのみ)、トレース例外 (H8S シリーズのみ) の発生を検出し、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、次の手順で行います。

- (a) 命令の実行中に例外の発生を検出します。
- (b) スタック領域に PC と CCR を退避します。EXR の有効ビットが ON のときには EXR も退避します。退避処理でエラーが発生した場合は、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後、シミュレータ・デバッガのコマンド待ちに戻ります。
- (c) CCR の I ビットを 1 にセットします。
- (d) 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。読み出し処理でエラーが発生した場合は、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後、シミュレータ・デバッガのコマンド待ちに戻ります。
- (e) 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。

### 3.5 H8S/2600CPU 特殊機能

#### (1) MAC 命令

H8S/2600CPU では、積和演算 (MAC 命令) が行えます。この命令では飽和演算、非飽和演算が選択できます。本シミュレータ・デバッガでは内蔵 I/O の SYSCR レジスタのビット 7 (以下 MACS ビットとする) の値により判定します。

MACS ビット 0 : 非飽和演算  
1 : 飽和演算

#### (2) EXR レジスタ

H8S/2600CPU では EXR レジスタを使用できます。また、このレジスタの有効/無効を設定することもできます。本シミュレータ・デバッガでは内蔵 I/O の SYSCR レジスタのビット 5 (以下 EXR ビットとする) の値により判定します。

EXR ビット 0 : EXR 無効  
1 : EXR 有効

SYSCR アドレスは、System Configuration ダイアログボックスの [SYSCR Address] で設定します。

【注】 SYSCR アドレスは内蔵 I/O に設定してください。内蔵 I/O 以外に SYSCR アドレスが設定されている場合は MACS ビットを 0 (非飽和)、EXR ビットを 0 (EXR 無効) と判断しますのでご注意ください。

詳しくは、「5.18 System Configuration ダイアログボックス」、「5.22 Control Registers」、および「5.23 SYSCR ダイアログボックス」を参照してください。

### 3.6 制御レジスタ

H8S/2600 シリーズでは、メモリにマッピングされた制御レジスタとして、SYSCR (システムコントロールレジスタ) をサポートしています。これにより、積和演算、EXR アクセス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。

SYSCR アドレスは、System Configuration ダイアログボックスの [SYSCR Address] で設定します。制御レジスタの変更や表示は Control Registers ウィンドウと SYSCR ダイアログボックスをご利用ください。

詳しくは、「5.18 System Configuration ダイアログボックス」、「5.22 Control Registers」お



よび「5.23 SYSCR ダイアログボックス」を参照してください。

## 3.7 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1024行までです。トレース情報の取得条件は、Trace Acquisition ダイアログボックスで指定します。Trace Acquisition ダイアログボックスは、Trace ウィンドウ上で右クリックしてポップアップメニューを表示し、[Acquisition]を選択することによって表示できます。取得したトレース情報は、Trace ウィンドウに表示します。Trace ウィンドウに表示する内容は、以下の通りです

- 累計命令実行サイクル数
- 命令アドレス
- CCR
- 乗算器内部フラグ ( H8S/2600 シリーズでのみ有効 )
- 命令モニター
- 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

トレース情報はサーチすることができます。サーチ条件は、Trace Search ダイアログボックスで設定します。Trace Search ダイアログボックスは、Trace ウィンドウ上で右クリックしてポップアップメニューを表示し、[Find]を選択することによって表示できます。

詳しくは、「5.14 Trace」から「5.16 Trace Search ダイアログボックス」を参照してください。

### 3.8 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標準入出力およびファイル入出力を行うことができます。入出力機能を利用する場合は、必ず Simulated I/O ウィンドウをオープンしておいてください。

サポートしている入出力処理は以下の通りです。機能コードには、16 ビットアドレス版、24 ビットアドレス版、32 ビットアドレス版があります。使用する CPU に合わせて選択してください。

表3-3 入出力機能一覧

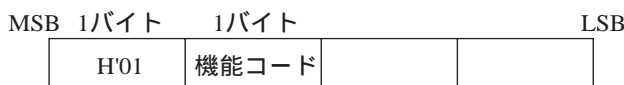
番号	機能コード	機能名	内容
1	H'01 (16 ビットアドレス) H'11 (24 ビットアドレス) H'21 (32 ビットアドレス)	GETC	標準入力からの 1 バイト入力
2	H'02 (16 ビットアドレス) H'12 (24 ビットアドレス) H'22 (32 ビットアドレス)	PUTC	標準出力への 1 バイト出力
3	H'03 (16 ビットアドレス) H'13 (24 ビットアドレス) H'23 (32 ビットアドレス)	GETS	標準入力からの 1 行入力
4	H'04 (16 ビットアドレス) H'14 (24 ビットアドレス) H'24 (32 ビットアドレス)	PUTS	標準出力への 1 行出力
5	H'05 (16 ビットアドレス) H'15 (24 ビットアドレス) H'25 (32 ビットアドレス)	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'017 (16 ビットアドレス) H'17 (24 ビットアドレス) H'27 (32 ビットアドレス)	FGETC	ファイルからの 1 バイト入力
8	H'08 (16 ビットアドレス) H'18 (24 ビットアドレス) H'28 (32 ビットアドレス)	FPUTC	ファイルへの 1 バイト出力
9	H'09 (16 ビットアドレス) H'19 (24 ビットアドレス) H'29 (32 ビットアドレス)	FGETS	ファイルからの 1 行入力
10	H'0A (16 ビットアドレス) H'1A (24 ビットアドレス) H'2A (32 ビットアドレス)	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

### 3. シミュレータ・デバッガの機能

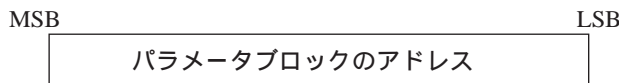
この機能を実現するためには、まず入出力用の特定の位置を System Configuration ダイアログボックスの [System Call Address] で指定し、[Enable] をチェック後、デバッグ対象プログラムを実行します。シミュレータ・デバッガでは、デバッグ対象プログラムの命令を実行中に、指定された位置へのサブルーチン分岐命令（BSR、JSR）すなわちシステムコール命令を検出すると、R0、R1（H8/300、H8/300L シリーズ）または ER1（H8/300H、H8S シリーズ）の内容をパラメータとして入出力処理を行います。

したがって、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 3-3 に示す機能コード



- ・ 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。

各入出力機能を図3-1の形式で説明します。

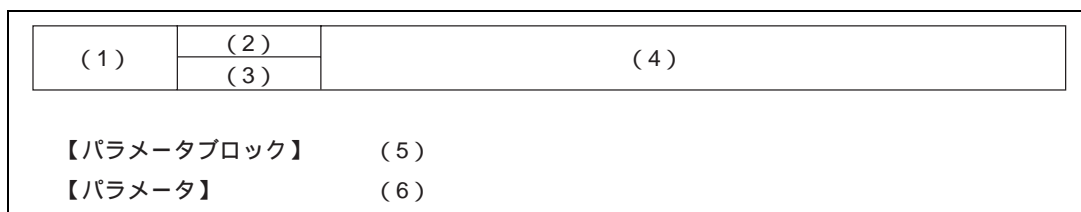


図3-1 入出力機能の説明形式

各項目の内容は、以下の通りです。

- (1) 表3-3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	標準入力からの1バイト入力
	H'01、H'11、H'21	

【パラメータブロック】

・機能コード：H'01 (16 ビットアドレス)  
1バイト                      1バイト

+0

・機能コード：H'11 (24 ビットアドレス)、H'21 (32 ビットアドレス)  
1バイト                      1バイト

+0   
+2

【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	標準出力への1バイト出力
	H'02、H'12、H'22	

【パラメータブロック】

・機能コード：H'02 (16 ビットアドレス)  
1バイト                      1バイト

+0

・機能コード：H'12 (24 ビットアドレス)、H'22 (32 ビットアドレス)  
1バイト                      1バイト

+0   
+2

【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	標準入力からの1行入力
	H'03、H'13、H'23	

【パラメータブロック】

・機能コード：H'03 (16 ビットアドレス)  
1バイト                      1バイト

+0

・機能コード：H'13 (24 ビットアドレス)、H'23 (32 ビットアドレス)  
1バイト                      1バイト

+0   
+2

【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

### 3. シミュレータ・デバッガの機能

---

4	PUTS	標準出力への1行出力
	H'04、H'14、H'24	

【パラメータブロック】

・機能コード：H'04 (16 ビットアドレス)  
                   1 バイト                  1 バイト

+0

・機能コード：H'14 (24 ビットアドレス)、H'24 (32 ビットアドレス)  
                   1 バイト                  1 バイト

+0   
 +2

【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	ファイルのオープン
	H'05、H'15、H'25	

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】

・機能コード：H'05 (16 ビットアドレス)

	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	ファイル名先頭アドレス	

・機能コード：H'15 (24 ビットアドレス)、H'25 (32 ビットアドレス)

	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	ファイル名先頭アドレス	
+6		

【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。

### 3. シミュレータ・デバッガの機能

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	ファイルのクローズ
	H'06	

【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	ファイルから 1 バイトのデータ読み出し
	H'07、H'17、H'27	

【パラメータブロック】

・機能コード：H'07 (16 ビットアドレス)

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

・機能コード：H'17 (24 ビットアドレス)、H'27 (32 ビットアドレス)

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

【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	ファイルへ 1 バイトのデータ書き込み
	H'08、H'18、H'28	

【パラメータブロック】

・機能コード：H'08 (16 ビットアドレス)  
1 バイト                      1 バイト

+0	実行結果	ファイル番号
+2	出力バッファ先頭アドレス	

・機能コード：H'18 (24 ビットアドレス)、H'28 (32 ビットアドレス)  
1 バイト                      1 バイト

+0	実行結果	ファイル番号
+2	出力バッファ先頭アドレス	
+4		

【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	ファイルから文字列データの読み出し
	H'09、H'19、H'29	

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】

・機能コード：H'09 (16 ビットアドレス)  
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	入力バッファ先頭アドレス	

・機能コード：H'19 (24 ビットアドレス)、H'29 (32 ビットアドレス)  
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	入力バッファ先頭アドレス	
+6		

【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）



### 3. シミュレータ・デバッガの機能

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	ファイルへ文字列データ書き込み
	H'0A、H'1A、H'2A	

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】

・機能コード : H'0A (16 ビットアドレス)  
 1 バイト                      1 バイト

+0	実行結果	ファイル番号
+2	出力バッファ先頭アドレス	

・機能コード : H'1A (24 ビットアドレス)、H'2A (32 ビットアドレス)  
 1 バイト                      1 バイト

+0	実行結果	ファイル番号
+2	出力バッファ先頭アドレス	
+4		

#### 【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	エンドオブファイルのチェック
	H'0B	

#### 【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	指定位置にファイルポインタを移動
	H'0C	

## 【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	オフセット上位ワード	
+6	オフセット下位ワード	

## 【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	ファイルポインタの現在位置を調査
	H'0D	

## 【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	オフセット上位ワード	
+4	オフセット下位ワード	

## 【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）

### 3. シミュレータ・デバッガの機能

---

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。システムコールアドレスとしてラベル SYS\_CALL を指定します。

```
                MOV.W    #H'0101, R0
                MOV.W    #PARM, R1
                JSR      @SYS_CALL
STOP           NOP
SYS_CALL      NOP
PARM          DATA.W   .INBUF
INBUF         .RES.B    2
                .END
```

## 3.9 命令実行サイクル数の計算

シミュレータ・デバッガでの命令実行サイクル数計算は、H8S、H8/300 シリーズ プログラミング マニュアルに記載されている計算式と System Configuration ダイアログボックスで設定したメモリのデータバス幅およびアクセスサイクル数を使用して行います。ただし、以下の理由によりシミュレータ・デバッガで計算した命令実行サイクル数と、実機でプログラムを実行した場合の命令実行サイクル数が異なることがあります。

#### (1) MOVFPE、MOVTPE 命令

E クロック同期命令のデータ転送サイクル数は、9～16 と幅を持った値となっています。シミュレータ・デバッガでは「11+オペランドアクセスサイクル数」で計算します。オペランドアクセスサイクル数は、メモリのデータバス幅およびアクセスサイクル数より求めます。

#### (2) EEPMOV 命令

EEPROM 書き込み専用命令のサイクル数は、命令読み出しに要するサイクル数と、データを転送するのに要するサイクル数の合計となります。

#### (3) SLEEP 命令

シミュレータ・デバッガでは、SLEEP 命令がプログラム停止用命令として使用される場合を考慮して、サイクル数を加算しません。

#### (4) 標準入出力およびファイル入出力処理

標準入出力およびファイル入出力処理は、シミュレータ・デバッガ固有の機能であるため、サイクル数に加算しません。なお、標準入出力およびファイル入出力処理とは、BSR、JSR 命令でシステムコールアドレスで指定された位置への分岐が完了してから入出力処理を行いコール元に戻るまでです。

### 3.10 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
- BREAK\_ACCESS : メモリ範囲のアクセスによるブレーク
- BREAK\_DATA : メモリ書き込みデータ値によるブレーク
- BREAK\_REGISTER : レジスタ書き込みデータ値によるブレーク
- BREAK\_SEQUENCE : 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 3-4 に示します。

表3-4 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、BREAK\_SEQUENCE 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 System Configuration ダイアログボックス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 3-5 に示します。

### 3. シミュレータ・デバッガの機能

表3-5 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
アドレスエラー/Address Error	<ul style="list-style-type: none"> <li>PC 値が奇数</li> <li>内蔵 I/O 空間からの命令フェッチ</li> <li>奇数アドレスからのワードアクセス</li> <li>奇数アドレスからのロングワードアクセス</li> </ul>	デバイスと同一動作をする
メモリアクセスエラー/ Memory Access Error	<ul style="list-style-type: none"> <li>確保されていないメモリ領域をアクセスしようとした</li> <li>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</li> <li>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</li> <li>メモリが存在しない領域をアクセスしようとした</li> <li>EEPROM 命令以外の命令での EEPROM 書き込み</li> </ul>	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
不当命令/ Illegal Instruction	<ul style="list-style-type: none"> <li>命令ではないコードの実行</li> <li>MOV.B Rn,@-SP または MOV.B @SP+,Rn の実行</li> </ul>	常に停止する そのまま実行するが結果は保証しない
命令実行不正/ Illegal Operation	<ul style="list-style-type: none"> <li>DAA 命令、DAS 命令で CCR の C フラグ、H フラグと補正前の値の関係不正</li> <li>DIVXU 命令、DIVXS 命令のゼロ除算またはオーバーフロー</li> </ul>	そのまま実行するが結果は保証しない

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 3-6 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3-6 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
アドレスエラー、 メモリアクセスエラー	命令読込時： エラーが発生した命令の先頭アドレス 命令実行時： エラーが発生した命令の次命令のアドレス
不当命令	エラーが発生した命令の先頭アドレス
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

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、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 Acquisition ダイアログボックスの [Trace buffer full handling] 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ステータスバーに表示します。

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。

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。

Stop

Go、Step コマンドにより実行を再開できます。

## 3.11 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。

- Set Break ダイアログボックスにおいて、ブレーク種別を [Break Data] や [Break Register] と指定したときのデータ
- Memory ウィンドウにおけるデータ
- Fill Memory ダイアログボックスにおけるデータ
- Search Memory ダイアログボックスにおけるデータ

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、System Configuration ダイアログボックスにより、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。

## 3.12 関数呼び出し履歴の表示

シミュレーションの中断時に、関数の呼び出し履歴を Stack Trace ウィンドウに表示します。これにより、プログラムの動作の流れを確認することができます。また、Stack Trace ウィンドウ上で関数名を選択することにより、該当するソースプログラムを Source ウィンドウ上に表示します。これにより、中断している関数の他に、その関数を呼び出した元の関数をチェックすることができます。関数呼び出し履歴を更新するのは、以下のような場合です。

- 「3.10 ブレーク条件」に示す条件によりシミュレーションが中断した時
- 上記中断した状態で、レジスタの値を変更した時
- シミュレーションをステップ実行している時

詳しくは、「5.25 Stack Trace」を参照してください。

---

## 4. メニュー

---

このマニュアルでは、標準的な Microsoft®メニュー命名規約を使用しています。

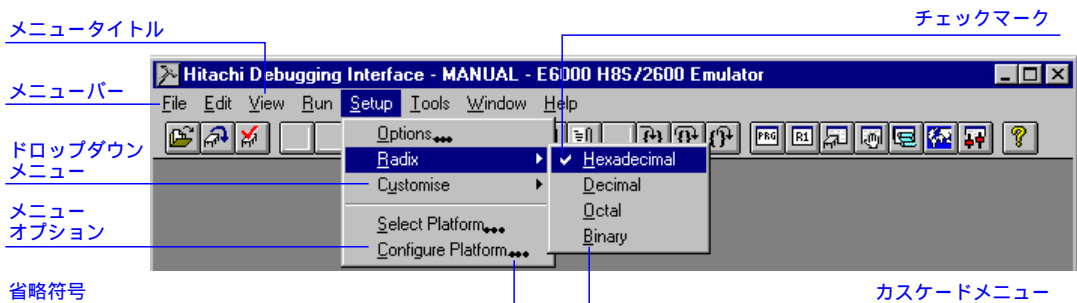


図4-1 メニュー

チェックマークは、そのメニューオプションにより提供する機能が選択されていることを示します。


省略符号は、そのメニューオプションを選択すると、追加情報の入力が必要なダイアログボックスを表示することを示します。

Windows®メニューシステムの使用法については、Windows®ユーザーズマニュアルを参照してください。


### 4.1 File

File メニューは、プログラムファイルにアクセスする場合に使用します。


#### 4.1.1 New Session...

 新しいデバッグプラットフォームを選択ができるように Select Session ダイアログボックスを表示します。

#### 4.1.2 Load Session...

 セッションファイル(拡張子が hds)を指定してロードできるように Select Session ダイアログボックスを表示します。セッションファイルには、デバッグプラットフォームの設定、シンボル、ブレークポイント、レジスタ値などのウィンドウ情報、位置情報が含まれています。

#### 4.1.3 Save Session

 現在のセッションファイルを更新します。現在のセッションファイルが定義されていない場合は、[Save Session As...]メニューオプションと同様の動作をします。




## 4. メニュー

---


### 4.1.4 Save Session As...

現在のセッションを新しいファイル名でセーブするために Save As ダイアログボックスを表示します。セッションファイルには、デバッグプラットフォームの設定、シンボル、ブレークポイント、レジスタ値などウィンドウの情報、位置が含まれています。

### 4.1.5 Load Program...

 Load Program ダイアログボックスを表示します。S-Record フォーマット(拡張子は\*.mot、\*.s20、\*.obj)または ELF/DWARF フォーマット(拡張子は\*.abs)のオブジェクトファイルを選択してデバッグプラットフォームのメモリにダウンロードします。選択したファイルでシンボルが使用できる場合は、シンボルもロードします。

### 4.1.6 Initialize

 デバッグシステムを再初期化します。現在オープンしているウィンドウをクローズし、デバッグプラットフォームとの接続を解除します。そして、デバッグプラットフォームの再接続をします。ステータスバーの最左端に'Link up'と表示されれば、初期化を完了したことを意味します。「4.4.1 Reset CPU」も参照してください。


### 4.1.7 Exit

HDI を終了します。HDI Option ダイアログボックスの'On Exit'セクションに指定された動作をします。「4.6.2 Options...」も参照してください。

## 4.2 Edit

Edit メニューは、プログラムのうち、各ウィンドウのデータを変更する場合や、デバッグプラットフォームのメモリデータに対して、それらのデータを変更する場合に使用します。


### 4.2.1 Cut

 内容を変更できるウィンドウでブロックが反転表示されている場合にのみ使用できます。反転表示されているブロックの内容をウィンドウから削除し、Windows®標準のクリップボードに格納します。


### 4.2.2 Copy

 内容が変更できるウィンドウでブロックが反転表示されている場合にのみ使用できます。反転表示されているブロックの内容を Windows®標準のクリップボードにコピーします。

### 4.2.3 Paste


 ウィンドウの内容が変更可能な場合のみ使用できます。Windows®標準のクリップボードの内容をウィンドウの現在のカーソル位置にコピーします。

#### 4.2.4 Find...

 ウィンドウにテキストが含まれている場合のみ使用できます。

Find ダイアログボックスを表示し、ユーザが単語を入力してテキスト中の出現箇所を検索できるようにします。一致する単語を見つけると、カーソルがその単語の先頭に移動します。


#### 4.2.5 Evaluate...

 Evaluate ダイアログボックスを表示します。たとえば"#pc + 205)\*2"のような数式を入力して、その結果を現在サポートしているすべての基数で表示します。


### 4.3 View

View メニューは、新規に各ウィンドウを開くために使用します。メニューオプションがグレー表示されていれば、そのウィンドウによって提供する機能は現在のデバッグプラットフォームで使用できません。


#### 4.3.1 Breakpoints

 Breakpoints ウィンドウを開きます。現在設定されているブレークポイントを表示して変更できるようにします。


#### 4.3.2 Command Line

 Command Line ウィンドウを開きます。テキストベースのコマンドを使用して、デバッグプラットフォームを制御できます。これらのコマンドは、バッチファイルから読み込み、結果をログファイルに書き出すことができます。これにより、自動テストが実行できます。


#### 4.3.3 Disassembly...

 表示を開始するアドレスを入力できるように Set Address ダイアログボックスを表示します。


#### 4.3.4 Labels

 プログラム中のシンボル・ラベルを操作できるよう Labels ウィンドウを表示します。


#### 4.3.5 Locals

 Locals ウィンドウを開きます。現在の関数において定義されている変数の値を表示し、変更できるようにします。PC が C/C++ ソースレベル関数の中になれば、ウィンドウは空白となります。


#### 4.3.6 Memory...

 Open Memory Window ダイアログボックスを表示します。Memory ウィンドウ内に表示するメモリブロック位置と表示フォーマットを指定できるようにします。


#### 4.3.7 Performance Analysis

 Performance Analysis ウィンドウを開きます。ユーザプログラムの特定のセクションが呼び出される回数を計測・表示できるようにします。


### 4.3.8 Profile-List

 Profile-List ウィンドウを開きます。関数とグローバル変数のアドレス、サイズ、関数呼出し回数、およびプロファイルデータを表示します。


### 4.3.9 Profile-Tree

 Profile-Tree ウィンドウを開きます。関数の呼出し関係をつリー構造で表示します。また、各関数のアドレス、サイズ、スタックサイズ、関数呼出し回数、およびプロファイルデータを表示します。スタックサイズ、関数呼出し回数、およびプロファイルデータは、実際の関数が呼出された経路における値を表示します。


### 4.3.10 Registers

 Registers ウィンドウを開きます。現在の CPU のすべてのレジスタとその内容を見ることができます。


### 4.3.11 Source...

 Open ダイアログボックスを表示します。表示したいソースファイル (C/C++ 言語フォーマットまたはアセンブラ言語フォーマット) のファイル名を入力します。ソースファイルが現在のプログラムに含まれていない場合、あるいはアブソリュートファイル (\*.abs) 内にそのファイルのデバッグ情報がない場合は、"Cannot load program. No Source level debugging available" というメッセージを表示します。


### 4.3.12 Status

 System Status ウィンドウを開きます。デバッグプラットフォームの現在の状態、セッション、プログラム名を見られるようにします。

### 4.3.13 Trace

 Trace ウィンドウを開きます。現在のトレース情報を見られるようにします。

### 4.3.14 Watch

 Watch ウィンドウを開きます。C/C++ ソースレベルの変数を入力し、その内容を表示・変更できるようにします。

### 4.3.15 Localized Dump...

Open Localized Dump ダイアログボックスを開きます。Localized Dump ウィンドウ内に表示するメモリ内容の開始アドレス、表示バイト数、および DBSC の種類を入力できるようにします。

### 4.3.16 Simulated I/O

Simulated I/O ウィンドウを開きます。標準入出力、ファイル I/O を使用できるようにします。

### 4.3.17 Stack Trace

Stack Trace ウィンドウを開きます。現在のスタックトレース情報を表示できるようにします。


### 4.3.18 External Tool

External Tools ウィンドウを開きます。協調検証ツールを使用できるようにします。

## 4.4 Run

Run メニューは、デバッグプラットフォームにおけるユーザプログラムの実行を制御するために使用します。


### 4.4.1 Reset CPU

 ターゲットハードウェアをリセットし、PC をリセットベクタアドレスに設定します(デバッグプラットフォームのリセットについては「4.1.6 Initialize」を参照)。


### 4.4.2 Go

 現在の PC からユーザプログラムを実行します。


### 4.4.3 Reset Go

 リセットベクタアドレスからユーザプログラムを実行します。

### 4.4.4 Go To Cursor

 現在の PC からユーザプログラムの実行を開始し、PC が現在のテキストカーソル(マウスカーソルではありません)の位置によって示されたアドレスに到達するまで続きます。


### 4.4.5 Set PC To Cursor

 PC を現在のテキストカーソル(マウスカーソルではありません)の位置によって示されるアドレスに設定します。アドレスが有効でなければ、無効です。


### 4.4.6 Run...

Run Program ダイアログボックスを表示します。ユーザプログラムの実行開始前にブレークポイントを入力できます。


### 4.4.7 Step In

 ユーザプログラムの 1 ブロックを実行して停止します。このブロックのサイズは、通常は単一の命令ですが、ユーザが複数の命令または C/C++ ソース行に設定することも可能です(「4.4.10 Step...」参照)。サブルーチンを呼び出した場合は、そのサブルーチンに入って実行を停止し、サブルーチンのコードを表示します。


### 4.4.8 Step Over

 ユーザプログラムの 1 ブロックを実行して停止します。このブロックのサイズは、通常は単一の命令ですが、ユーザが複数の命令または C/C++ ソース行に設定することも可能です(「12.4.10 Step...」参照)。サブルーチン呼び出しの場合は、そのサブルーチンには入らず、現在の PC 位置が現在の表示の次行に移動するまでユーザプログラムを実行します。


### 4.4.9 Step Out

 現在の関数の終わりに到達するまでユーザプログラムを実行し、呼び出す関数の次の行に PC を設定して停止します。

### 4.4.10 Step...

 Step Program ダイアログボックスを表示します。ステップ動作の設定を変更できるようにします。

### 4.4.11 Halt

 ユーザプログラムの実行を停止します。


## 4.5 Memory

Memory メニューは、ユーザプログラムがアクセスするメモリの設定に使われます。


### 4.5.1 Refresh

すべてのオープンしている Memory ウィンドウの内容を強制的にアップデートします。


### 4.5.2 Load...

 メモリ領域のアドレスのオフセット、S-record フォーマットのファイルを選択できるように、Load Memory ダイアログボックスを表示します。


### 4.5.3 Save...

 メモリ領域の開始、終了アドレスを指定し、S-record フォーマットでファイルを保存するための Save Memory As ダイアログボックスを表示します。ダイアログボックスを表示したときに自動的に設定する開始、終了アドレスは、Memory ウィンドウ中の反転表示されたメモリブロックです。


### 4.5.4 Verify...

 ディスク上の S-record ファイルとベリファイするメモリ領域の開始、終了アドレスを設定するための Verify S-Record File with Memory ダイアログボックスを表示します。


### 4.5.5 Test...

 Test Memory ダイアログボックスを表示します。メモリブロックを指定して、デバッグプラットフォームのメモリブロックに対して、読み取り/書き込み動作が正しく行われているかをテストします。このテストは、ターゲットに依存します。しかし、すべてのケースで現在のメモリ内容は上書きされ、プログラムとデータは削除されます。本シミュレータ・デバッガではサポートしていません。


### 4.5.6 Fill...

 Fill Memory ダイアログボックスを表示します。デバッグプラットフォームのメモリブロックに値を書き込みます。開始、終了フィールドは、Save オプション(「4.5.3 Save...」も参照)と同様に設定します。


### 4.5.7 Copy...

 Copy Memory ダイアログボックスを表示します。デバッグプラットフォームの同一メモリスペース内で、メモリブロックを別の位置にコピーします。これらのブロックは重なってもかまいません。開始、終了フィールドは、Save オプション(「4.5.3 Save...」も参照)と同様に設定します。


### 4.5.8 Compare...

 メモリ領域の開始アドレスと終了アドレスを指定し、別のメモリ領域と比較するための Compare Memory ダイアログボックスを表示します。開始、終了フィールドは、Save オプション(「4.5.3 Save...」も参照)と同様に設定します。

### 4.5.9 Configure Map...

 Memory Mapping ウィンドウを開きます。デバッグプラットフォームの現在のメモリマップを表示し、変更できるようにします。デバッグプラットフォームにより、Memory Map ダイアログボックスが開きます。

### 4.5.10 Configure Overlay...

 Overlay...ダイアログボックスを表示します。オーバーレイ機能を利用した場合、優先するセクショングループを設定することができます。


## 4.6 Setup

Setup メニューは、HDI ユーザインタフェースの設定変更と、デバッグプラットフォームの設定に使用します。


### 4.6.1 Status Bar

ステータスバーの表示/非表示を切り換えます。ステータスバーが表示になっている場合は、メニューテキストの左にチェックマークを表示します。

### 4.6.2 Options...

 HDI Options ダイアログボックスを表示します。HDI 固有の設定(デバッグプラットフォーム依存の設定ではありません)を変更できるようにします。

### 4.6.3 Radix

 数値を表示したり入力する場合の、(基数の接頭部を入力しなかった場合の)基数のデフォルトの設定をカスケードメニューとして表示します。現在選択されている基数は左にチェックマークが付いていて、ツールバーの対応するボタンが押し下げられています。

たとえば現在の基数が Decimal ならば、10 進法の 10 は"10"と表示され、"10"、"H'A"、"0x0a"などと入力できます。現在の基数が Hexadecimal ならば、10 進法の 10 は"0A"と表示され、"A"、"D'10"などと入力できます。

### 4.6.4 Customize


 ユーザがカスタマイズできるオプションのリストをカスケードメニューとして表示します。

Toolbar : このカスケードメニューオプションを選択すると Customize ダイアログボックスを表示します。

Font : このカスケードメニューオプションを選択すると Font ダイアログボックスを表示します。固定長ピッチのフォントを選択することができます。

File Filter : このカスケードメニューオプションを選択すると Customize File Filter ダイアログボックスを表示します。オブジェクト、ソースファイル、メモリファイルのファイルフィルタを変更できます。


### 4.6.5 Configure Platform...

 Set-up ダイアログボックスを表示します。デバッグプラットフォームの設定を変更できるようにします。詳しくは「5.18 System Configuration ダイアログボックス」を参照してください。


## 4.7 Window

Window メニューは、現在開いている各ウィンドウの表示変更に使用します。以下のメニューオプションは常に表示しています。また、現在開いているウィンドウの番号付きリストも一緒に表示します。一番手前に表示しているウィンドウにはチェックマークがついています。


### 4.7.1 Cascade

 ウィンドウを標準的なカスケード方式で、すなわち各ウィンドウのタイトルバーが見えるように左上から配置します。

### 4.7.2 Tile

 ウィンドウを標準的なタイル方式で表示、すなわちすべてのウィンドウが重ならずに表示されるよう各ウィンドウのサイズを変更します。

### 4.7.3 Arrange Icons

アイコン化されたウィンドウを、親フレームの下辺に沿って標準的な方式できれいに整列させます。


### 4.7.4 Close All

すべてのウィンドウをクローズします。

## 4.8 Help

Help メニューは、HDI が提供する機能の使用方法に関する追加情報へのアクセスに使用します。

### 4.8.1 Index

メインヘルプファイルのインデックスを開きます。

### 4.8.2 Using Help

Windows®ハイパーテキストヘルプシステムの使用方法のヘルプファイルを開きます。

### 4.8.3 Search for Help on

メインヘルプファイルを開き、Search ダイアログボックスを表示します。このダイアログボックスで、ファイルのキーワードを入力・閲覧することができます。

### 4.8.4 About HDI

About HDI ダイアログボックスを表示します。HDI と現在ロードされている DLL のバージョンを確認することができます。





---

## 5. ウィンドウおよびダイアログボックス

---

本章では、各ウィンドウおよびダイアログボックスの種類と、それぞれがサポートしている機能、および関連ポップアップメニューにより使用できるオプションについて説明します。

### 5.1 Breakpoints

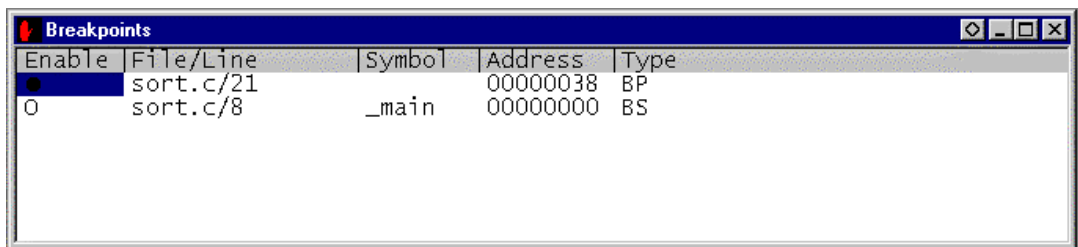


図5-1 Breakpoints ウィンドウ

本ウィンドウは、設定されている全ブレークポイントをリスト表示します。表示する項目は以下の通りです。

- [Enable] 該当ブレークポイントの有効/無効を示します。 またはOが表示されているブレークポイントは有効であることを示します。
- [File/Line] ブレークポイントの存在するファイル名および行番号を表示します。
- [Symbol] ブレークポイントの設定されている位置に対応するシンボルを表示します。対応するシンボルがない場合は何も表示しません。
- [Address] ブレークポイントの設定されている位置を示します。
- [Type] ブレーク種別を表示します。
  - BP : PC ブレーク
  - BA : ブレークアクセス
  - BD : ブレークデータ
  - BR : ブレークレジスタ ( 括弧内にレジスタ名を表示します。 )
  - BS : ブレークシーケンス

また、本ウィンドウでブレークポイントをダブルクリックすると、Set Break ダイアログボックスが開き、ブレーク条件を変更することができます。ただし、ブレークシーケンスをダブルクリックすると、Break Sequence ダイアログボックスが開きます。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

#### 5.1.1 Add...

ブレークポイントを設定します。クリックすると、Set Break ダイアログボックスが開き、ブレーク

## 5. ウィンドウ

---

ク条件を設定することができます。

### 5.1.2 Edit...

ブレークポイントが選択されている場合のみ有効です。変更したいブレークポイントを選択後クリックすると、Set Break ダイアログボックスが開き、ブレーク条件を変更することができます。ただし、ブレークシーケンスが選択された場合は、Break Sequence ダイアログボックスが開きます。

### 5.1.3 Delete

ブレークポイントが選択されている場合のみ有効です。選択されているブレークポイントを削除します。ブレークポイントを削除しないで、詳細情報は保持したまま、条件が成立した場合に実行を停止させないようにする場合は、Disable オプションを使用します(「5.1.5 Disable/Enable」参照)。

### 5.1.4 Delete All

すべてのブレークポイントをリストから削除します。

### 5.1.5 Disable/Enable

ブレークポイントが選択されている場合のみ有効です。選択されているブレークポイントの有効(Enable)/無効(Disable)を切り替えます(無効にした場合は、ブレークポイントはリストには残りますが、指定した条件が成立しても実行を停止しません)。ブレークポイントが有効となっていれば、メニューテキストの左にチェックマークが付きます(かつ、そのブレークポイントの Enable 列に丸を表示します)。

### 5.1.6 Go to Source

ブレークポイントのある Source または Disassembly ウィンドウをオープンします。

## 5.2 Set Break ダイアログボックス

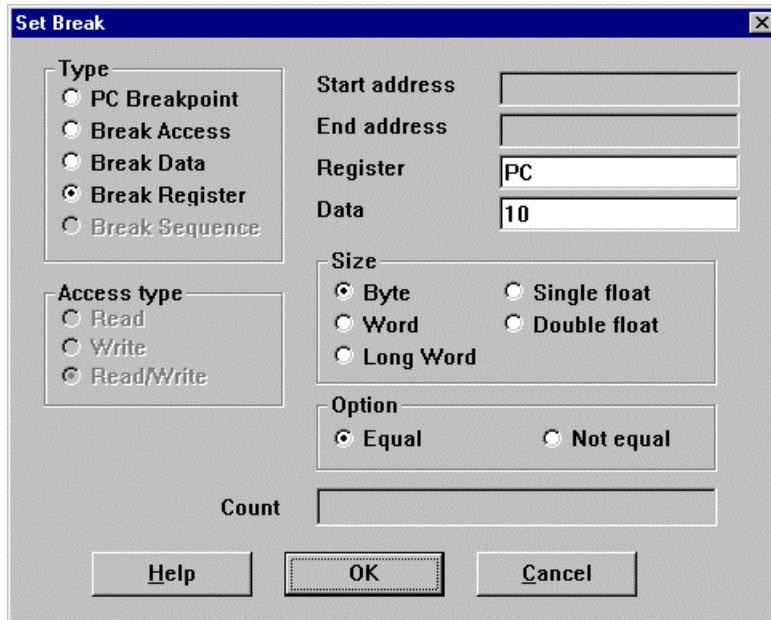


図5-2 Set Break ダイアログボックス

本ダイアログボックスでは、ブレーク条件を設定します。

まず、設定するブレーク種別を [Type]ラジオボタンで指定します。各種別において設定可能な項目を以下に示します。

[PC Breakpoint]	[Start address]	ブレークする命令の位置
	[Count]	指定位置の命令をフェッチする回数 (省略すると1となります)
[Break Access]	[Start address]	アクセスするとブレークするメモリの開始位置
	[End address]	アクセスするとブレークするメモリの終了位置 (省略すると開始位置のみが範囲となります)
	[Access type]	アクセス種別
[Break Data]	[Start address]	ブレーク判定を行うメモリの位置
	[Data]	ブレーク条件となるデータ値
	[Size]	データのサイズ
	[Option]	データの一致/不一致
[Break Register]	[Register]	ブレーク条件を設定するレジスタ名
	[Data]	ブレーク条件となるデータ値 (省略するとレジスタへ書き込む度にブレークします)
	[Size]	データのサイズ
	[Option]	データの一致/不一致
[Break Sequence]	Break Sequence	ダイアログボックスが開くので、そこで設定します。

[PC Breakpoint]の設定時に、[Start Address]に多重定義関数あるいはメンバ関数を含むクラス名を入力した場合、Select Function ダイアログボックスが開くので設定する関数を選択します。詳細は、「14 関数の設定」を参照してください。

指定したブレーク条件は、[OK]ボタンをクリックすることにより設定します。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

### 5.3 Break Sequence ダイアログボックス

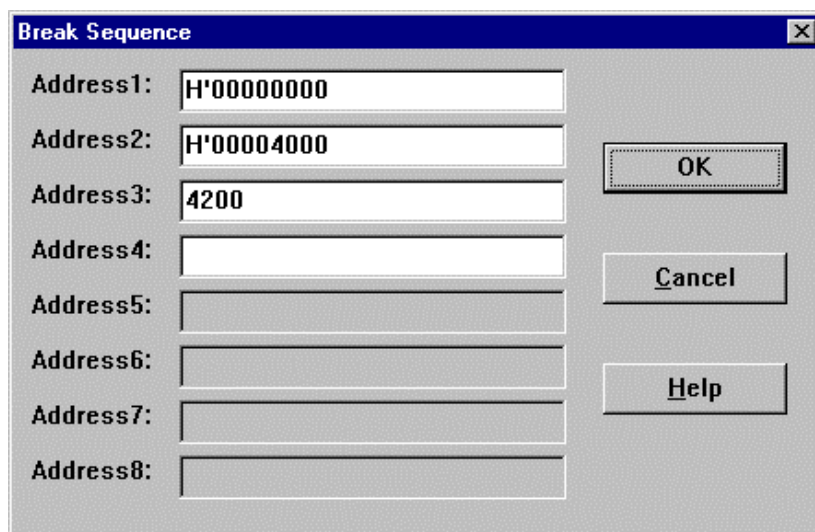


図5-3 Break Sequence ダイアログボックス

本ダイアログボックスでは、ブレークの発生条件となる通過アドレスを設定します。

[Address1]~[Address8]にアドレスを指定します。なお、8ポイントすべてを設定する必要はありません。アドレスとして多重定義関数あるいはメンバ関数を含むクラス名を入力した場合、Select Function ダイアログボックスが開くので設定する関数を選択します。詳細は、「14 関数の設定」を参照してください。

通過アドレスは、[OK]ボタンをクリックすることにより設定することができます。[Cancel]ボタンをクリックすると、新たに通過アドレスを設定しないでダイアログボックスを閉じます。

## 5.4 Command Line

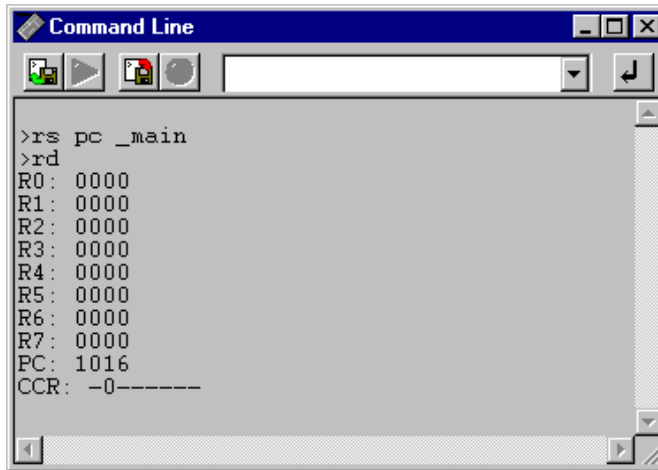



図5-4 Command Line ウィンドウ

ウィンドウメニューやウィンドウコマンドを使用しないで、テキストベースのコマンドを入力してデバッグプラットフォームを制御できるウィンドウです。あらかじめ定義した一連のコマンドをバッチファイルから呼び出してデバッグプラットフォームに送り、出力結果をログファイルに記録する必要がある場合に便利です。コマンドの実行は、テキストボックス入力後に Enter キーを押すか、テキストボックス右にあるボタンをクリックします。使用できるコマンドについては、オンラインヘルプを参照してください。


ウィンドウタイトルとしてバッチファイル名とログファイル名をコロンで区切って表示します。

ツールバーボタンの機能は、以下に示すポップアップメニューオプションと同じです。


### 5.4.1 Set Batch File...

 Set Batch File ダイアログボックスを表示します。HDI コマンドファイル名(\*.hdc)を入力できません。コマンドファイルは、自動的に実行します。ファイル名をウィンドウのタイトルバーに表示します。

### 5.4.2 Play

 最後に実行したコマンドファイルを実行します。このボタンは、バッチファイルの実行中はグレー表示となり、コマンドファイルの実行が停止してユーザに制御が戻ったときに有効表示となります。

### 5.4.3 Set Log File...

 Open Log File ダイアログボックスを表示します。出力結果を記録する HDI ログファイル(\*.log)の名前を入力します。ロギングオプションは自動的に設定され、ログファイル名をウィンドウのタイトルバーに表示します。

既に存在しているログファイル名を指定すると、ログを追加するか、以前のログを消去して、新しいログを上書きするかを尋ねられます。

## 5. ウィンドウ

### 5.4.4 Logging

ファイルへのロギング処理を実行するか、停止するかを切り替えます。ロギングが実行状態になっていれば、ボタンが有効状態になります。ログファイルの内容は、ロギングが終了するか、チェックボックスをクリアしてロギングを一時的に停止しなければ表示できないことにご注意ください。ロギングを再び開始すると、ログファイルに追加します。

### 5.4.5 Select All

Command Line ウィンドウ出力をすべて選択します。

### 5.4.6 Copy

テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストを Windows® クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

## 5.5 Disassembly

本ウィンドウは、アセンブラレベルでプログラムを表示します。

本ウィンドウは、ソースウィンドウとは異なったレイアウトです。列にシンボル名を表示することがあります。アセンブラ情報はメモリ内容を逆アセンブルします。オブジェクトファイルからデバッグ情報を読まずにメモリ内容を直接編集、表示します。

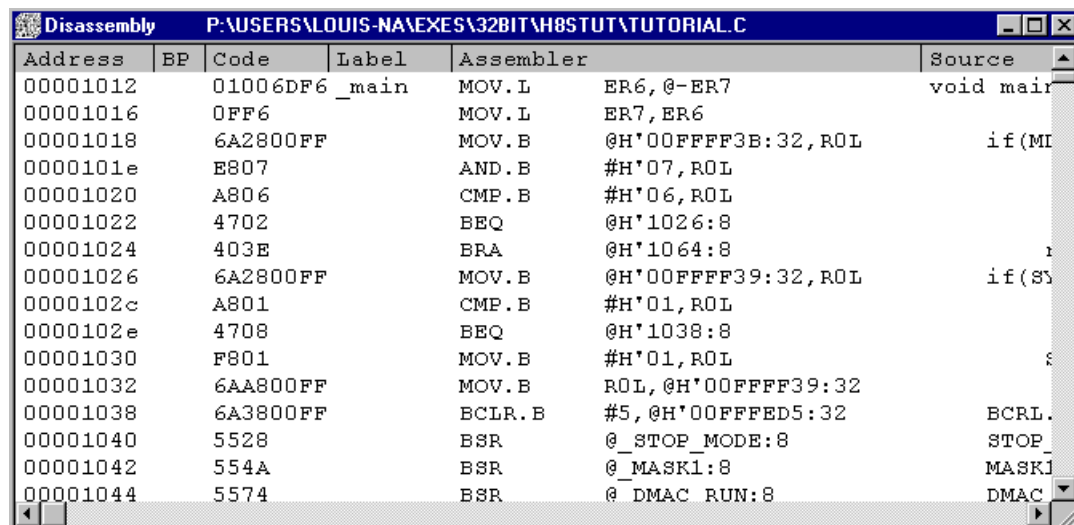


図5-5 Disassembly ウィンドウ


各列に対してダブルクリック動作をサポートしています。

- BP  
そのアドレスに対し、標準ブレークポイントを設定または解除します。
- Address  
Set Addressダイアログボックスを表示します。新しいアドレスを入力できます。そのアドレスが、ソースファイル中であれば、新しいウィンドウをオープンし、カーソルが該当個所をさします。もしくは、ソースプログラムの該当個所を表示します。アドレスに対応するソースファイルがないときには、本ウィンドウがそのアドレスまでスクロールします。アドレスとして多重定義関数名やクラス名を入力するとSelect Functionダイアログボックスが開くので、関数を選択してください。
- Code and Assembler  
Assemblerダイアログボックスを表示します。任意のアドレスの命令を変更できます。機械語を変更しても、ソースファイルには反映されません。また、そのセッション終了時に変更した情報が失われるので注意してください。
- Label  
Labelダイアログボックスを表示します。新しいラベルの入力やラベルの削除、編集ができます。
- Source  
スタートメニューのHDIショートカットで設定されているエディタが該当個所を表示します。

BP 列で右クリックすると現在サポートしている標準ブレークポイントタイプを表示します。現在、選択されているブレークポイントタイプがメニューの左側にチェックされています。

ウィンドウ内の BP 列以外のところで右クリックすると使用可能なオプションをポップアップメニューで表示します。


### 5.5.1 Copy

 テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストを Windows® クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

### 5.5.2 Set Address...

Set Address ダイアログボックスを表示します。新しい開始アドレスを入力すると、ウィンドウが更新され、ユーザが入力したアドレスが左上端のアドレスとなります。アドレスとして多重定義関数あるいはメンバ関数を含むクラス名を入力した場合、Select Function ダイアログボックスが開くので、設定する関数を選択します。

### 5.5.3 Go To Cursor

 現在の PC アドレスからプログラムを実行します。プログラムは、PC がテキストカーソル(マウスカーソルではありません)の位置によって指定されたアドレスに達するか、別のブレーク条件が成立するまで実行します。



## 5. ウィンドウ

### 5.5.4 Set PC Here

PC の値をテキストカーソル(マウスカーソルではありません)の位置によって指定されたアドレスに変更します。

### 5.5.5 Instant Watch...

変数名を現在のテキストカーソル(マウスカーソルではありません)の位置から抜き出して名前とともに Instant Watch ダイアログボックスを表示します。選択しているソース行が有効なときのみ機能します。

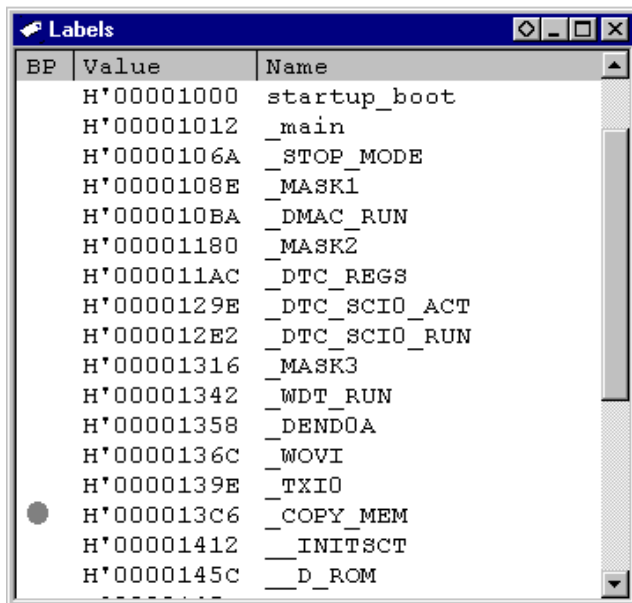
### 5.5.6 Add Watch

テキストカーソル (マウスカーソルではありません)の位置の表示から抽出した名前を、ウォッチ変数のリストに追加します。Watch ウィンドウが開いていない場合はこれを開いて、他のウィンドウの一番上に表示します。選択しているソース行が有効なときのみ機能します。

### 5.5.7 Go to Source

テキストカーソル (マウスカーソルではありません)の位置に対応する Source ウィンドウをオープンします。ソース行が有効なときのみ機能します。

## 5.6 Labels



BP	Value	Name
	H*00001000	startup_boot
	H*00001012	_main
	H*0000106A	_STOP_MODE
	H*0000108E	_MASK1
	H*000010BA	_DMAC_RUN
	H*00001180	_MASK2
	H*000011AC	_DTC_REGS
	H*0000129E	_DTC_SCI0_ACT
	H*000012E2	_DTC_SCI0_RUN
	H*00001316	_MASK3
	H*00001342	_WDT_RUN
	H*00001358	_DEND0A
	H*0000136C	_WOVI
	H*0000139E	_TXIO
●	H*000013C6	_COPY_MEM
	H*00001412	_INITSCT
	H*0000145C	_D_ROM

図5-6 Labels ウィンドウ

カラムヘッダをクリックすることで、シンボルをアルファベット順またはアドレス順にソートして表示します。

各列に対してダブルクリック動作をサポートしています。

- BP  
そのアドレスに対し、標準ブレークポイントを設定または解除します。
- Address  
関数の先頭アドレスでソースファイルをオープンします。
- Name  
Edit Labelダイアログボックスをオープンします。

右クリックすると BP 列に現在サポートしている標準ブレークポイントタイプを表示します。現在、選択されているブレークポイントタイプがメニューの左側にチェックされています。

ウィンドウ内の BP 列以外のところで右クリックすると使用可能なオプションをポップアップメニューで表示します。

### 5.6.1 Add...

Add Label ダイアログボックスを表示します。

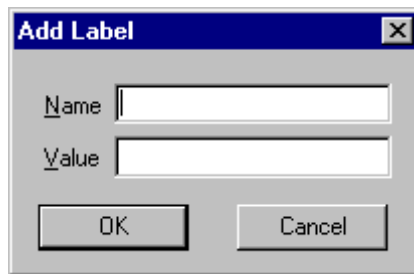


図5-7 Add Label ダイアログボックス

新しいラベル名を Name フィールドに入力し、対応する値を Value フィールドに入力して[OK]を押します。Add Label ダイアログボックスがクローズし、ラベルリストに新しいラベルが追加され更新されます。多重定義関数やクラス名を入力したときは、Select Function ダイアログボックスが開くので、関数を選択して Value フィールドを設定します。詳細は「14 関数の設定」を参照してください。

### 5.6.2 Edit...

Edit Label ダイアログボックスを表示します。

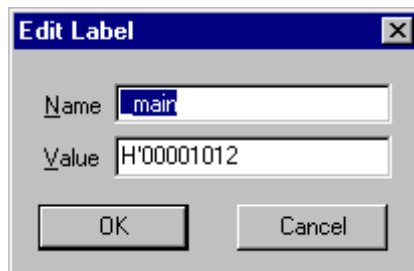


図5-8 Edit Label ダイアログボックス

## 5. ウィンドウ

---

ラベル名と対応する値を編集して、[OK]を押すとラベルリストに編集が反映され保存されます。多重定義関数やクラス名を入力したときは、Select Function ダイアログボックスが開くので、関数を選択して Value フィールドを設定します。詳細は「14 関数の設定」を参照してください。

### 5.6.3 Find...

Find Label Containing ダイアログボックスを表示します。

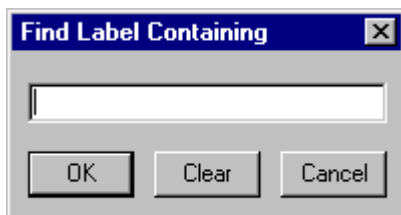


図5-9 Find Label Containing ダイアログボックス

検索したいラベル名の一部、全部をエディットボックスに入力し、[OK]ボタンまたは、ENTER キーを押すと、ダイアログボックスはクローズし、指定された文字列を含んでいるテキストファイルを検索します。

【注】 ラベルは、はじめの 1024 文字分しか情報を保持していません。したがって、ラベル名のはじめの 1024 文字分は重複しないようにしてください。ラベルは、大文字、小文字を区別します。


### 5.6.4 Find Next

ラベルが検索できた後、検索条件に一致する次のラベルを検索します。

### 5.6.5 View Source

ラベルのアドレスに該当する Source または、Disassembly ウィンドウをオープンします。

### 5.6.6 Copy

 テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストを Windows® クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

### 5.6.7 Delete

シンボルリストから選択されたラベルを削除します。Delete キーでも同様の動作です。確認メッセージを表示します。



図5-10 ラベル削除確認メッセージボックス

Yes ボタンをクリックするとラベルリストから削除されウィンドウが更新されます。メッセージボックスの表示不要のときは、HDI Options ダイアログボックスの Confirmation シートの Delete Label オプションを選択しないでください。

### 5.6.8 Delete All

リストからすべてのラベルを削除します。確認メッセージボックスを表示します。

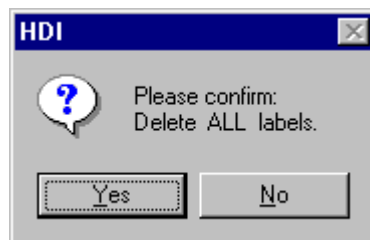


図5-11 Confirming All Label Deletion メッセージボックス

Yes ボタンをクリックすると、すべてのラベルが HDI のシンボルテーブルから削除され、リスト表示もクリアされます。メッセージボックスの表示が不要のときは、HDI Options ダイアログボックスの Confirmation シートの Delete All Labels オプションを選択しないでください。

### 5.6.9 Load...

現在の HDI のシンボルテーブルに結合します。Load Symbols ダイアログボックスをオープンします。

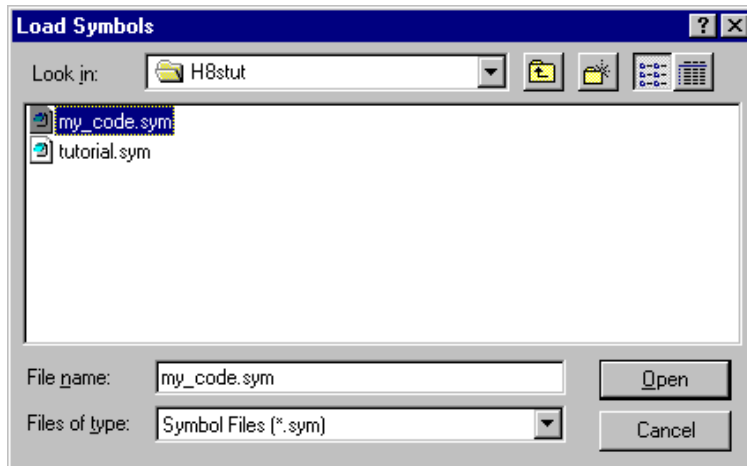


図5-12 Load Symbols ダイアログボックス

ダイアログボックスは、Windows®標準の open file ダイアログボックスと同様です。ファイルを選択し、Open をクリックするとロードを開始します。シンボルファイルの標準拡張子は".sym"です。シンボルのロードが完了するとロードしたシンボル数を表示したメッセージボックスを表示します。このメッセージボックスは、HDI Options ダイアログボックスの Confirmation シートで非表示にもできます。

### 5.6.10 Save

現在のシンボルテーブルをシンボルファイルに保存します。

### 5.6.11 Save As...

Save Symbols ダイアログボックスは Windows®標準の Save File As ダイアログボックスと同様に操作できます。File name フィールドにファイル名を入力し Open をクリックするとシンボルファイルにラベルリストを保存します。標準ファイル拡張子は".sym"です。

シンボルファイルフォーマットが「付録 C」にあるので参照してください。

## 5.7 Locals

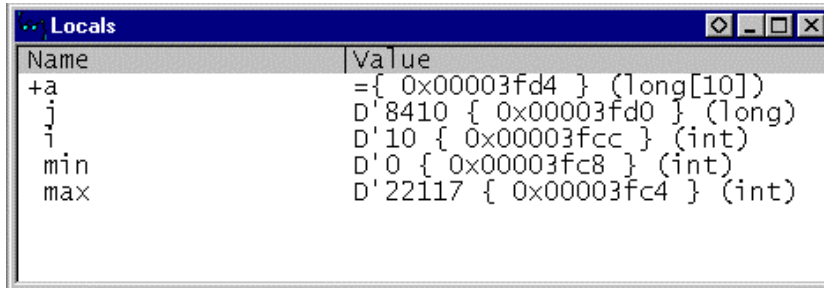


図5-13 Locals ウィンドウ

すべてのローカル変数を表示・変更できるウィンドウです。このウィンドウは、アブソリュートファイル (\*.abs)に含まれるデバッグ情報によって、現在の PC 位置から関数内にあるローカル変数に関連づけることができない場合は空白となります。


表示する項目は以下の通りです。

- [Name]           変数名を表示します
- [Value]          変数の値、割付け位置、および型を表示します  
割付け位置は{}で囲んで表示し、型は()で囲んで表示します

変数はリスト表示され、'+ '記号は変数名をダブルクリックすれば情報を拡張表示できることを、'- '記号は情報を収縮表示できることを示します。また、'+ ' / '- 'キーでも拡張 / 収縮表示することができます。情報の表示については、「12.3.2 ウォッチ項目の拡張」を参照してください。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには次のオプションが含まれています。

### 5.7.1 Copy

 テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストを Windows® クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

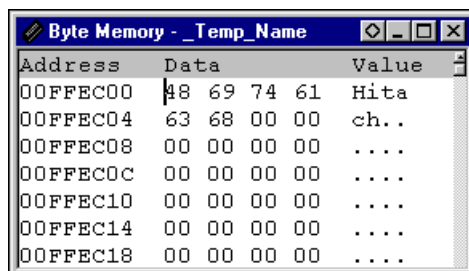
### 5.7.2 Edit Value...

選択しているローカル変数の値を変更するダイアログボックスを表示して、変数の値を変更できます。

### 5.7.3 Radix

選択しているローカル変数の表示基数を変更します。

## 5.8 Memory



Address	Data	Value
00FFEC00	48 69 74 61	Hita
00FFEC04	63 68 00 00	ch..
00FFEC08	00 00 00 00	....
00FFEC0C	00 00 00 00	....
00FFEC10	00 00 00 00	....
00FFEC14	00 00 00 00	....
00FFEC18	00 00 00 00	....

図5-14 Memory ウィンドウ

デバッグプラットフォームのメモリ内容を表示・変更できるウィンドウです。メモリは ASCII、バイト、ワード、ロングワード、単精度浮動小数点、倍精度浮動小数点の各フォーマットで表示できます。タイトルバーは現在の表示スタイルと、直前のラベル(シンボル)からのオフセットで示したアドレスを示します。

メモリ内容は、現在のカーソル位置で入力するか、データ項目をダブルクリックして変更することができます。データ項目をダブルクリックすると Edit ダイアログボックスを表示するので、複雑な式を使用して新しい値を入力できます。そのアドレスのデータが変更できない(すなわち ROM または未使用領域上にある)場合は、"Invalid address value"というメッセージを表示します。

Address 列でダブルクリックすると Set Address ダイアログボックスが表示されて、新しい開始アドレスを入力できるようになります。[OK]をクリックするとウィンドウが更新されて、入力したアドレスが、1 行目になるように表示します。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

### 5.8.1 Refresh

Memory ウィンドウの内容を強制的にアップデートします。

### 5.8.2 Load...

Load Memory ダイアログボックスを表示します。現在のデバッグ情報を削除せずにデバッグプラットフォームのメモリに S-record ファイル(\*.mot)をロードします。offset フィールドは、アドレス値を変更するときに指定します。オプションナリファイフラグはダウンロードが正常に行われたかどうかをチェックします。

### 5.8.3 Save...

Save Memory As ダイアログボックスを表示します。デバッグプラットフォームのブロックを S-record ファイル(\*.mot)でセーブします。開始フィールドと終了フィールドは、Search オプション(「5.8.8 Search...」も参照)と同様に設定します。

### 5.8.4 Test...

Test Memory ダイアログボックスを表示します。デバッグプラットフォーム内のメモリブロックを検査します。テストの詳細は、デバッグプラットフォームに依存します。開始フィールドと終了

フィールドは、Search オプション(「5.8.8 Search...」も参照)と同様に設定します。本シミュレータ・デバッガではサポートしていません。

### 5.8.5 Fill...

Fill Memory ダイアログボックスを表示します。デバッグプラットフォームのメモリブロックに指定した値を書き込みます。開始フィールドと終了フィールドは、Search オプション(「5.8.8 Search...」も参照)と同様に設定します。

### 5.8.6 Copy...

Copy Memory ダイアログボックスを表示します。デバッグプラットフォームのメモリブロックを同一メモリ空間内の別の場所にコピーします。ブロックはオーバーラップしても構いません。開始フィールドと終了フィールドは、Search オプション(「5.8.8 Search...」も参照)と同様に設定します。

### 5.8.7 Compare...

Compare Memory ダイアログボックスを表示します。メモリ領域の開始アドレスと終了アドレスを選択して別のメモリ領域と比較します。Memory ウィンドウでメモリブロックが反転表示されていれば、ダイアログボックスを表示したときに開始アドレスと終了アドレスを自動的に設定します。

メモリベリファイに似ていますが、本機能は、2つのメモリブロックを比較します。

### 5.8.8 Search...

Search Memory ダイアログボックスを表示します。デバッグプラットフォームのメモリブロックの中で指定したデータ値を検索します。メモリブロックが反転表示されている場合は、ダイアログボックスの開始フィールドと終了フィールドに、反転表示されているブロックに対応する開始アドレスと終了アドレスを自動的に設定します。

### 5.8.9 Set Address...

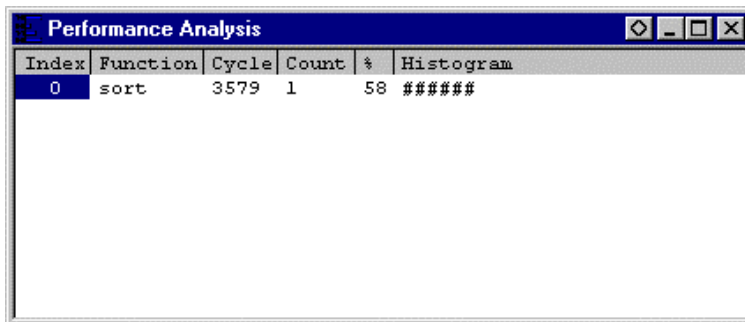
Set Address ダイアログボックスを表示します。新しい開始アドレスを入力すると、ウィンドウが更新され、ユーザが入力したアドレスが左上端のアドレスとなります。アドレスとして多重定義関数あるいはメンバ関数を含むクラス名を入力した場合、Select Function ダイアログボックスが開くので、設定する関数を選択します。

### 5.8.10 ASCII/Byte/Word/Long/Single Float/Double Float

これらの6つの項目の左に付いているチェックマークは、現在の表示フォーマットを示しています。異なる項目を選択して、そのフォーマットに変更することができます。



## 5.9 Performance Analysis



Index	Function	Cycle	Count	%	Histogram
0	sort	3579	1	58	#####

図5-15 Performance Analysis ウィンドウ

本ウィンドウは、指定関数ごとの実行サイクル数を表示します。  
 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。

表示する項目は以下の通りです。

[Index]	設定条件のインデックス番号
[Function]	測定対象の関数名（または関数の開始アドレス）
[Cycle]	当該関数の累計実行サイクル数
[Count]	当該関数の累計呼び出し回数
[%]	プログラム全体の実行サイクル数に占める当該関数の実行サイクル数の割合
[Histogram]	上記割合のヒストグラム表示

また、評価関数をダブルクリックすると、Performance Option ダイアログボックスが開き、関数を変更することができます。なお、関数は 255 個まで設定可能です。

表示領域内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

### 5.9.1 Add Range...

評価関数を追加します。クリックすると、Performance Option ダイアログボックスが開き、関数を追加することができます。

### 5.9.2 Edit Range

反転表示カーソルバーがユーザ定義範囲にある場合にのみ有効です。Performance Option ダイアログボックスを表示して、範囲の設定を変更します。

### 5.9.3 Delete Range

反転表示カーソルバーがユーザ定義範囲にある場合にのみ有効です。範囲設定を削除し、他の範囲のデータを再計算します。

#### 5.9.4 Reset Counts/Times

現在のプログラムの実行効率データをクリアします。

#### 5.9.5 Delete All Ranges

現在のユーザ定義範囲をすべて削除し、実行効率測定データをクリアします。

#### 5.9.6 Enable Analysis

実行効率データ収集のオン・オフを切り替えます。実行効率測定が現在アクティブであれば、テキストの左にチェックマークを表示します。

### 5.10 Performance Option ダイアログボックス



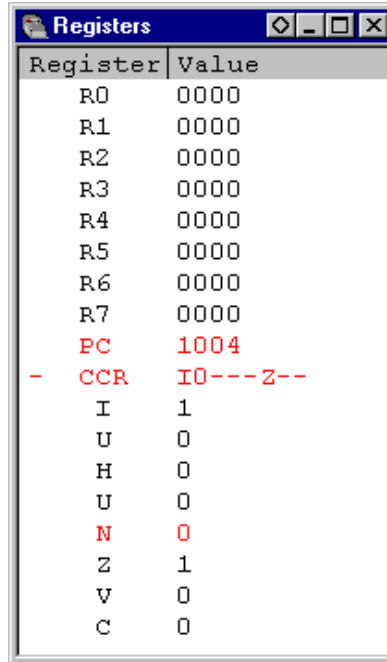
図5-16 Performance Option ダイアログボックス

本ダイアログボックスでは、性能評価する関数(ラベルも可)を設定します。評価結果は Performance Analysis ウィンドウで表示します。

多重定義関数あるいはメンバ関数を含むクラス名を入力した場合、Select Function ダイアログボックスが開くので設定する関数を選択します。詳細は、「14 関数の設定」を参照してください。

[OK]ボタンをクリックすることにより、評価関数を設定します。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

## 5.11 Registers




Register	Value
R0	0000
R1	0000
R2	0000
R3	0000
R4	0000
R5	0000
R6	0000
R7	0000
PC	1004
- CCR	I0---Z--
I	1
U	0
H	0
U	0
N	0
Z	1
V	0
C	0

図5-17 Registers ウィンドウ

現在のレジスタ値を表示・変更できるウィンドウです。ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

### 5.11.1 Copy

 テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストを Windows® クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

### 5.11.2 Edit...

Register ダイアログボックスを表示します。テキストカーソル (マウスカーソルではありません) の位置によって示されたレジスタの値を設定します。

### 5.11.3 Toggle Bit

テキストカーソルが、たとえばステータスレジスタ内のフラグなどのビットフィールド上にある場合にのみ使用できます。ビットの現在のステータスを、もう一方のステータスに変更します。たとえば、設定したオーバーフローフラグをクリアすることができます。

## 5.12 Source

Source ウィンドウは、C/C++、アセンブラプログラムで、デバッグオプションが指定されていれば表示できます。

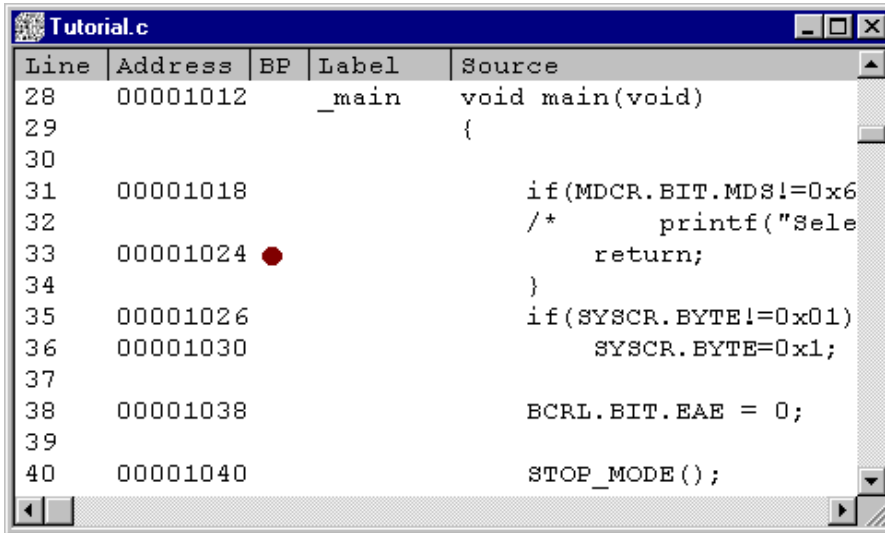


図5-18 Source ウィンドウ

各列に対してダブルクリック動作をサポートしています。

- BP  
クリックしたアドレス(PC)のブレークポイントの設定/解除
- Address  
Set Address ダイアログボックスを表示します。アドレスの入力ができます。本ファイルの範囲内のアドレスのときに、カーソルは指定された位置までスクロールします。入力されたアドレスが別ファイルのときには、ウィンドウを新規にオープンしカーソルは、該当アドレスを表示します。入力されたアドレスに対応するファイルがないときには、新規に Disassembly ウィンドウをオープンします。多重定義関数名やクラス名が入力されたとき、Select Function ダイアログボックスが開くので、関数を選択します。
- Label  
Label ダイアログボックスを表示します。新しいラベルを入力するかラベル名の編集ができます。
- Line  
Set Line ダイアログボックスを表示します。ソースファイルの指定した行を直接表示することができます。
- Source  
スタートメニューのHDIショートカットで設定されているエディタが該当個所を表示します。


## 5. ウィンドウ

---


右クリックすると BP 列に現在サポートしている標準ブレークポイントタイプを表示します。現在、選択されているブレークポイントタイプがメニューの左側にチェックされています。

ウィンドウ内の BP 列以外のところで右クリックすると使用可能なオプションがポップアップメニューを表示します。

### 5.12.1 Copy

 テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストを Windows® クリップボードにコピーし、他のアプリケーションに貼り付けられます。

### 5.12.2 Find...

 Find ダイアログボックスを表示して、ソースファイル内で文字列を検索します。


### 5.12.3 Set Address...

Set Address ダイアログボックスを表示します。新しい開始アドレスを入力すると、ウィンドウが更新され、ユーザが入力したアドレスが左上端のアドレスとなります。アドレスとして多重定義関数あるいはメンバ関数を含むクラス名を入力した場合、Select Function ダイアログボックスが開くので、設定する関数を選択します。

### 5.12.4 Set Line...

Set Line ダイアログボックスを表示します。テキストカーソル (マウスカーソルではありません) を指定された行に移動します。

### 5.12.5 Go To Cursor

 現在の PC アドレスからユーザプログラムの実行を開始します。プログラムは、PC がテキストカーソル(マウスカーソルではありません)の位置によって指定されたアドレスに達するか、別のブレーク条件が成立するまで実行します。デバッグプラットフォームによってサポートされていない場合、このメニューオプションはグレー表示となります。

### 5.12.6 Set PC Here

PC の値をテキストカーソル(マウスカーソルではありません)の位置によって指定されたアドレスに変更します。

### 5.12.7 Instant Watch...

変数名を現在のテキストカーソル(マウスカーソルではありません)の位置から抜き出して名前とともに Instant Watch ダイアログボックスを表示します。ソース行が有効なときのみ機能します。

### 5.12.8 Add Watch

テキストカーソル (マウスカーソルではありません)の位置の表示から抽出した名前を、ウォッチ変数のリストに追加します。Watch ウィンドウが開いていない場合はこれを開いて、他のウィンドウの一番上に表示します。ソース行が有効なときのみ機能します。

### 5.12.9 Go to Disassembly

現在のソース行に対応したアドレスの Disassembly 表示を行います。

## 5.13 System Status

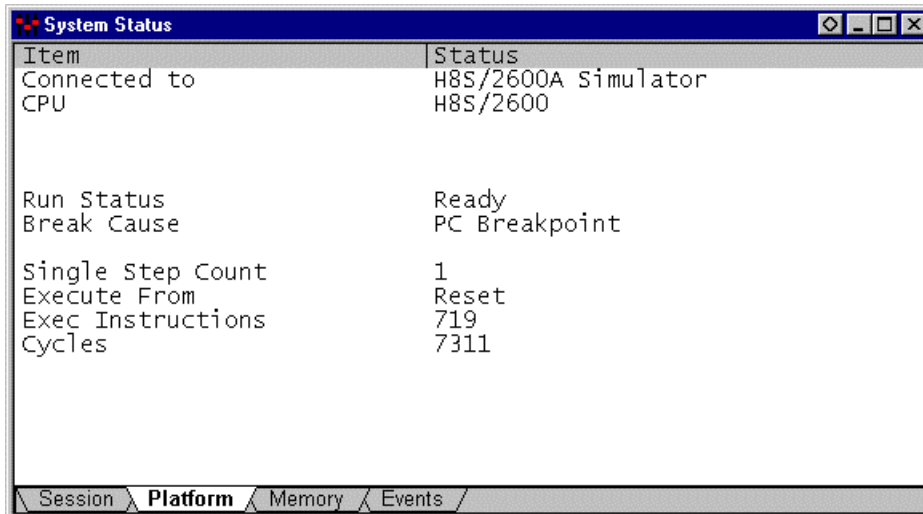


図5-19 System Status ウィンドウ

デバッグプラットフォームの現在のステータスを表示できるウィンドウです。

System Status ウィンドウは、4枚のシートに分割されています。

- Session シート  
接続しているデバッグプラットフォームおよびロードしたファイルの名前など、現在のセッションに関する情報を含んでいます。
- Platform シート  
CPU種別および動作モードなど、デバッグプラットフォームのステータス情報、実行状態および実行統計情報を含んでいます。
- Memory シート  
メモリマッピングおよび現在ロードしたオブジェクト・ファイルによって使用されるメモリアリアなど、現在のメモリステータスに関する情報を含んでいます。
- Events シート  
リソース情報およびブレークポイント等のイベント情報に関する情報を含んでいます。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。



### 5.14.1 Find...

Trace Search ダイアログボックスを表示します。現在のトレースバッファ内の特定のトレースレコードを検索できます。

### 5.14.2 Find Next

トレースレコードが検索できた後、次の同様のトレースレコードをさらに検索できます。

### 5.14.3 Filter...

Filter Trace ダイアログボックスを表示します。必要でないトレースエントリを、マスクできます。本シミュレータ・デバッガではサポートしていません。

### 5.14.4 Acquisition...

Trace Acquisition ダイアログボックスを表示します。トレースするユーザプログラム領域を指定します。これは問題のある領域にトレースを集中させるために使用することができます。

### 5.14.5 Halt

ユーザプログラムの実行を中止せずにトレースデータの収集を中止し、トレース情報を更新します。本シミュレータ・デバッガではサポートしていません。

### 5.14.6 Restart

トレースデータの収集を開始します。本シミュレータ・デバッガではサポートしていません。

### 5.14.7 Snapshot

ユーザプログラムの実行を中止せずに、トレース情報を更新してデバッグプラットフォームの現在のステータスを表示します。本シミュレータ・デバッガではサポートしていません。

### 5.14.8 Clear

トレースバッファを空にします。複数の Trace ウィンドウが開いているときは、それらは同じバッファをアクセスしているため、すべてのトレースバッファをクリアします。

### 5.14.9 Save...

Save As ファイルダイアログボックスを表示します。トレースバッファの内容をテキストファイルとして保存します。Cycle の範囲によって指定するか、すべてのバッファを指定することができます(すべてのバッファをセーブするには、数分かかることがあります)。このファイルはトレースバッファに再ロードできないことに注意してください。

### 5.14.10 View Source

該当アドレスに対応したソースプログラムや逆アセンブルを表示します。



### 5.14.11 Trim Source

ソースプログラムの左側の空白を取り除きます。

## 5.15 Trace Acquisition ダイアログボックス

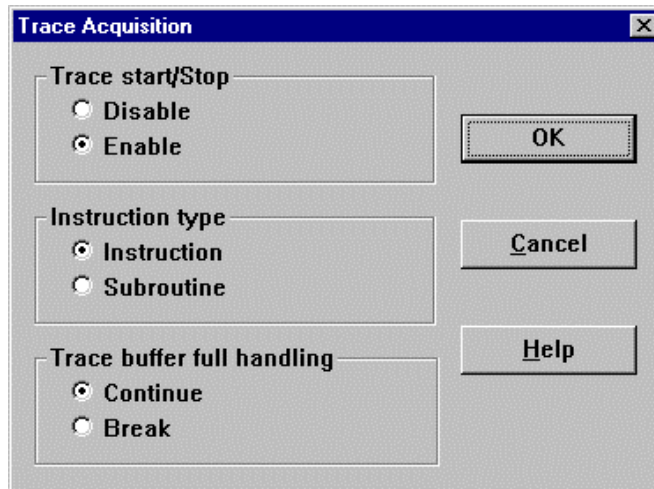


図5-21 Trace Acquisition ダイアログボックス

本ダイアログボックスでは、トレース情報の取得条件を設定します。

[Trace start/Stop]

- [Disable]      トレース情報の取得停止
- [Enable]      トレース情報の取得開始

[Instruction type]

- [Instruction]    全命令のトレース情報を取得
- [Subroutine]    サブルーチン命令のみトレース情報を取得

[Trace buffer full handling]

- [Continue]      トレース情報取得バッファが満杯になっても取得を続行
- [Break]        トレース情報取得バッファが満杯になった場合、実行停止

指定した内容は、[OK]ボタンをクリックすることにより設定します。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

## 5.16 Trace Search ダイアログボックス

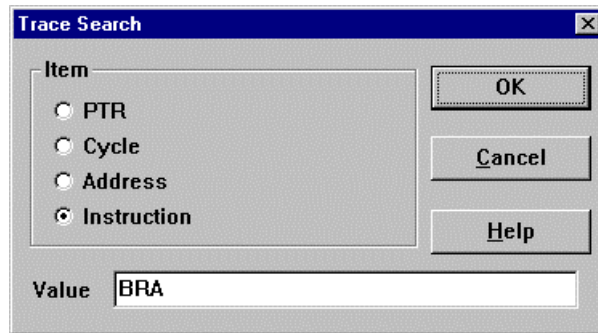


図5-22 Trace Search ダイアログボックス

本ダイアログボックスは、トレース情報のサーチ条件を設定します。[Item]でサーチ対象項目を指定し、[Value]で指定された内容をサーチします。

- [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。  
                  -*nnn* の形式で指定してください。
- [Cycle]        累計命令実行サイクル数
- [Address]      命令アドレス
- [Instruction]  命令ニーモニック

[OK]ボタンをクリックすることにより、サーチ条件を設定します。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

## 5.17 Watch

Name	Value
-a	= { 0x00003fd4 } (long[10])
[0]	H' 00000000 { 0x00003fd4 } (long)
[1]	H' 00000daa { 0x00003fd8 } (long)
[2]	H' 000020da { 0x00003fdc } (long)
[3]	H' 00002704 { 0x00003fe0 } (long)
[4]	H' 00002f5a { 0x00003fe4 } (long)
[5]	H' 00003ead { 0x00003fe8 } (long)
[6]	H' 0000421f { 0x00003fec } (long)
[7]	H' 00004d1d { 0x00003ff0 } (long)
[8]	H' 000053dc { 0x00003ff4 } (long)
[9]	H' 00005665 { 0x00003ff8 } (long)
max	H' 00005665 { 0x00003fc4 } (int)

図5-23 Watch ウィンドウ

## 5. ウィンドウ

---

C/C++ソースレベルの変数を表示・変更することができるウィンドウです。本ウィンドウの内容は、アブソリュートファイル(\*.abs)内のデバッグ情報から、C/C++ソースファイルとのリンクが取れば表示します。リンクが取れない場合には、表示しません。

表示する項目は以下の通りです。

[Name] 変数名を表示します


[Value] 変数の値、割付け位置、および型を表示します

割付け位置は{}で囲んで表示し、型は()で囲んで表示します

変数はリスト表示され、'+'記号は変数名をダブルクリックすれば情報を拡張表示できることを、'-'記号は情報を収縮表示できることを示します。また、'+'/ '-'キーでも拡張/収縮表示することができます。

ウィンドウ内でマウスの右ボタンをクリックすると、ポップアップメニューを表示します。このメニューには以下のオプションが含まれます。

### 5.17.1 Copy

テキストブロックが反転表示されている場合にのみ使用できます。反転表示されたテキストをWindows®クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

### 5.17.2 Delete

テキストカーソル (マウスカーソルではありません)の位置によって示された変数を、Watch ウィンドウから削除します。

### 5.17.3 Delete All

すべての変数を、Watch ウィンドウから削除します。

### 5.17.4 Add Watch...

Add Watch ダイアログボックスを表示します。監視する変数または式を入力します。

### 5.17.5 Edit Value...

Edit Value ダイアログボックスを表示して、変数の値を変更します。ポインタの値を変更する場合は、そのポインタが有効なデータを指し示さなくなる可能性があるため、特に注意が必要です。

### 5.17.6 Radix

選択しているウォッチ項目の表示基数を変更します。

## 5.18 System Configuration ダイアログボックス

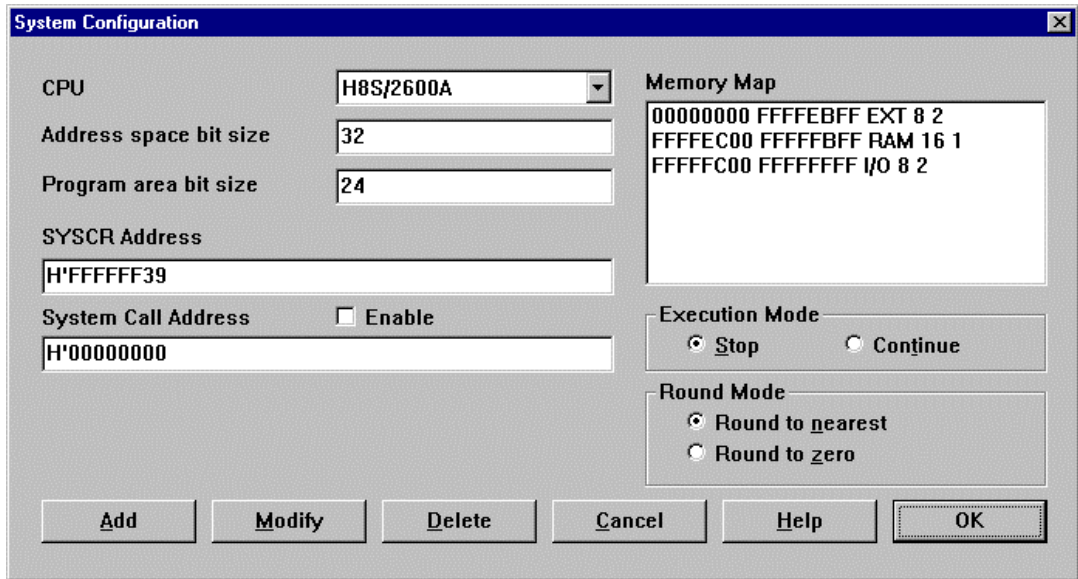


図5-24 System Configuration ダイアログボックス

本ダイアログボックスでは、アドレス空間ビット数、プログラム領域ビット数、SYSCR アドレス、システムコールの開始位置、実行モード、浮動小数点の丸めモード、およびメモリマップの設定を行います。

- |                          |  |
|--------------------------|--|
| [CPU]                    | 現在設定されている CPU。<br>( CPU は、Select Session ダイアログボックスにより設定します。 )  |
| [Address Space Bit Size] | アドレス空間のビット数を指定します。各 CPU で指定できる値は以下の通りです。<br>H8/300、H8/300L、H8/300HN、H8S/2600N、H8S/2000N : 16<br>H8/300HA : 17 ~ 24<br>H8S/2600A、H8S/2000A : 17 ~ 32            |
| [Program Area Bit Size]  | プログラム領域のビット数を指定します。各 CPU で指定できる値は以下の通りです。<br>H8/300、H8/300L、H8/300HN、H8S/2600N、H8S/2000N : 16<br>H8/300HA : アドレス空間ビット数と同<br>—<br>H8S/2600A、H8S/2000A : 17 ~ 24 |
| [System Call Address]    | デバッグ対象プログラムから標準入出力またはファイル入出力を行うためのシステムコールの開始位置を指定します。<br>[Enable] チェックするとシステムコールが有効となります。  |
| [Execution Mode]         | シミュレーションエラーが発生した場合のシミュレータ・デバッガ動作を規定します。  |

## 5. ウィンドウ

	[Stop]	シミュレーションを停止します。
	[Continue]	シミュレーションを続行します。
[Round Mode]	浮動小数点数 10 進 2 進変換で発生する丸めのモードを指定します。	
	[Round to nearest]	最近値丸め
	[Round to zero]	ゼロ方向丸め

[Memory Map]には、メモリ情報として、先頭アドレス・終了アドレス、メモリ種別、データバス幅、アクセスサイクル数の順に表示し、[Memory Map]は、以下の各ボタンにより設定・変更・削除ができます。

- [Add] [Memory Map]の項目を設定します。クリックすると、Memory Map Modify ダイアログボックスが開き、設定することができます。
- [Modify] [Memory Map]の項目を変更します。変更したい項目をリストボックス上で選択後、ボタンをクリックします。クリックすると、Memory Map Modify ダイアログボックスが開き、変更することができます。
- [Delete] [Memory Map]の項目を削除します。削除したい項目をリストボックス上で選択後、ボタンをクリックします。

変更内容は、[OK]ボタンをクリックすることにより設定します。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

### 5.19 Memory Map Modify ダイアログボックス

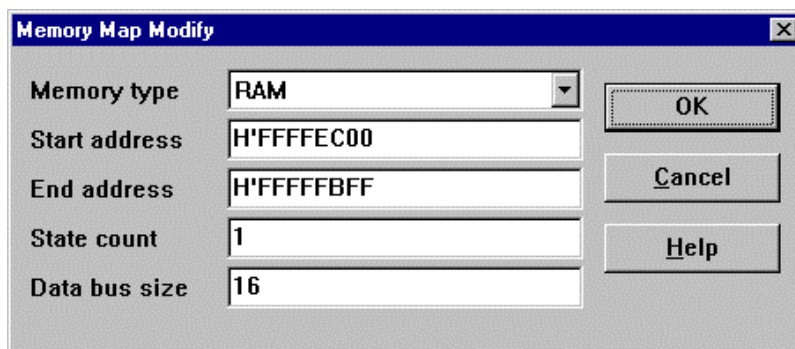


図5-25 Memory Map Modify ダイアログボックス

本ダイアログボックスでは、シミュレータ・デバッガの対象 CPU のメモリマップを設定します。各項目に表示する内容は、対象 CPU によって異なります。これらの値は、メモリアクセスサイクル数の算出に使用します。

[Memory type]	メモリ種別
[Start address]	メモリ種別に対応するメモリの先頭アドレス
[End address]	メモリ種別に対応するメモリの終了アドレス
[State count]	メモリアクセスサイクル数
[Data bus size]	メモリのデータバス幅

変更内容は、[OK]ボタンをクリックすることにより設定されます。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

【注】 メモリリソースを確保している領域のメモリマップは、削除・変更することができません。あらかじめ、Memory Map ダイアログボックスによりメモリリソースを削除してから、メモリマップを削除・変更してください。

## 5.20 Memory Map ダイアログボックス

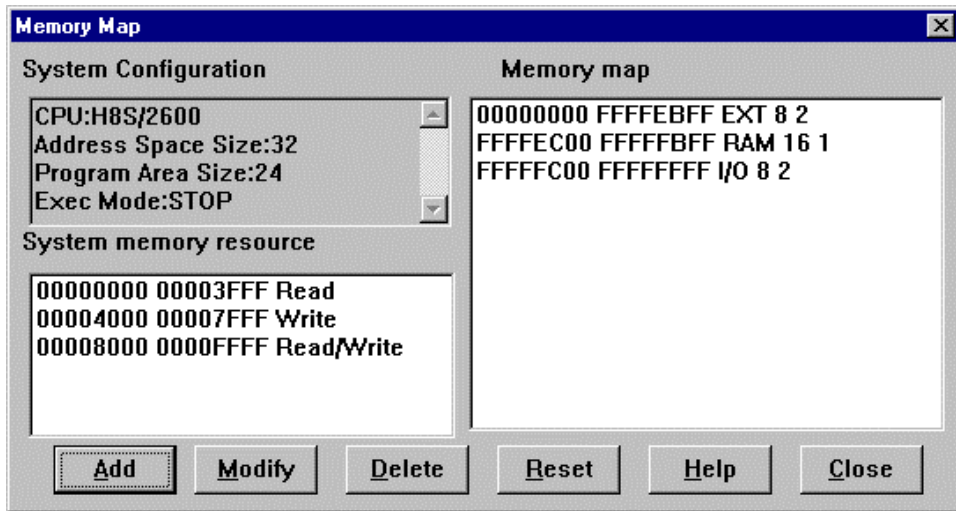


図5-26 Memory Map ダイアログボックス

本ダイアログボックスは、対象 CPU に関する情報とメモリマップを表示します。

表示する項目は以下の通りです。

- |                          |  |
|--------------------------|--|
| [System Configuration]   | シミュレータ・デバッガの対象 CPU、アドレスバス幅、実行モードを表示します。                |
| [System memory resource] | 現在設定されているメモリリソースのアクセス種別とその先頭アドレス・終了アドレスを表示します。         |
| [Memory map]             | メモリ情報として、メモリ種別とその先頭アドレス・終了アドレス、データバス幅、アクセスサイクル数を表示します。 |

[System memory resource]は次の各ボタンにより設定・変更・削除することができます。

- |          |  |
|----------|--|
| [Add]    | [System memory resource]の項目を設定します。クリックすることにより、System Memory Resource Modify ダイアログボックスが開き、設定することができます。                                  |
| [Modify] | [System memory resource]の項目を変更します。変更したい項目をリストボックス上で選択後、ボタンをクリックします。クリックすることにより、System Memory Resource Modify ダイアログボックスが開き、変更することができます。 |
| [Delete] | [System memory resource]の項目を削除します。削除したい項目をリストボックス上で選択後、ボタンをクリックします。  |

なお、[Memory map]・[System memory resource]は、[Reset]ボタンによりデフォルト値にリセット

することができます。また、本ダイアログボックスは、[Close]ボタンをクリックすることにより閉じます。

### 5.21 System Memory Resource Modify ダイアログボックス

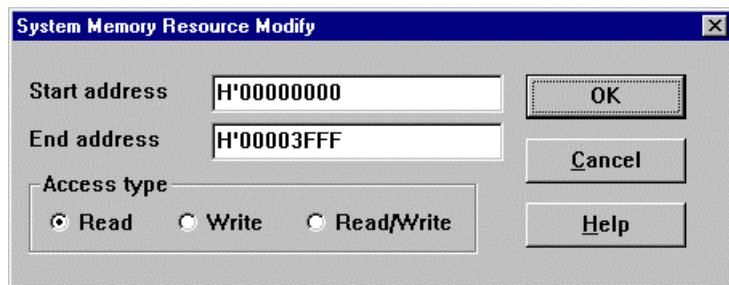


図5-27 System Memory Resource Modify ダイアログボックス

本ダイアログボックスでは、メモリリソースの設定・変更を行います。設定する項目は以下の通りです。

[Start address]	確保するメモリ領域の先頭アドレス
[End address]	確保するメモリ領域の終了アドレス
[Access type]	アクセス種別
Read	読み出しのみ可能
Write	書き込みのみ可能
Read/Write	読み書き可能

各項目を指定後、[OK]ボタンをクリックすることによりメモリリソースの設定・変更を行います。[Cancel]ボタンをクリックすると、設定しないでダイアログボックスを閉じます。

### 5.22 Control Registers

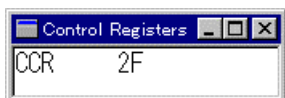


図5-28 Control Registers ウィンドウ

本ウィンドウは、制御レジスタの値を表示します。なお、本ウィンドウは H8S/2600 シリーズのみサポートします。

表示する制御レジスタは以下の通りです。

[SYSCR]	システムコントロールレジスタ
---------	----------------

制御レジスタ値は、ウィンドウ上で直接変更できます。また、レジスタをダブルクリックすると、レジスタ値をビット単位で設定できる SYSCR ダイアログボックスが開きます。

## 5.23 SYSCR ダイアログボックス

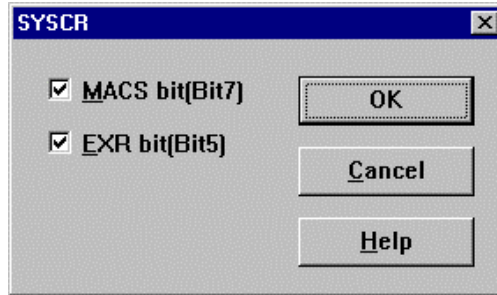


図5-29 SYSCR ダイアログボックス

本ダイアログボックスでは、SYSCR レジスタ（システムコントロールレジスタ）の値を設定します。なお、本ダイアログボックスは H8S/2600 シリーズのみサポートします。

設定する項目は以下の通りです。

- [MACS bit]      積和演算（MAC 命令）の飽和演算/非飽和演算を設定します。チェックすると飽和演算になります。
- [EXR bit]        EXR レジスタの有効/無効を設定します。チェックすると有効になります。

[OK]ボタンをクリックすることにより、変更した値をメモリ上に設定します。[Cancel]ボタンをクリックすると、変更内容をメモリ上に設定しないでダイアログボックスを閉じます。

## 5.24 Simulated I/O

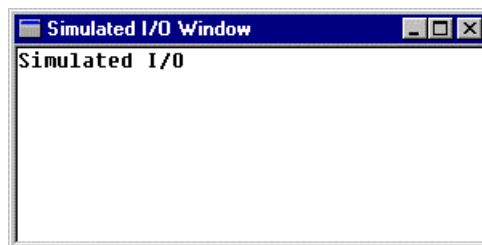


図5-30 Simulated I/O ウィンドウ

本ウィンドウは、デバッグ対象プログラムから標準入出力およびファイル入出力のシステムコールを行うためのウィンドウです。



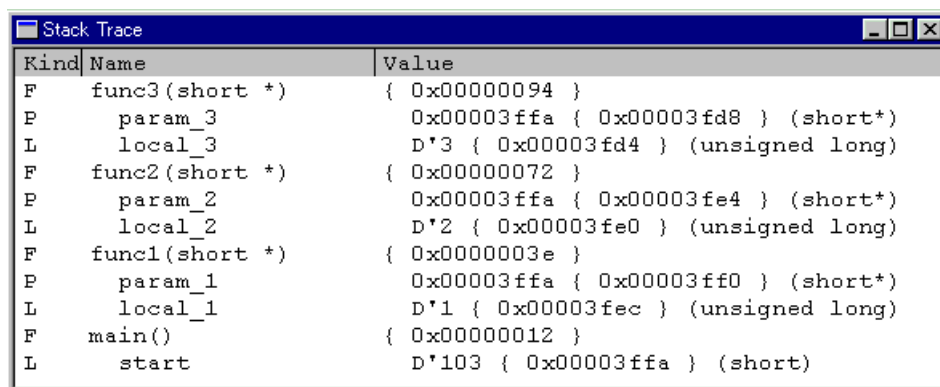
## 5. ウィンドウ

本ウィンドウ上で右クリックすると、以下のポップアップメニューを表示します。

- [Copy] 反転表示されたテキストを Windows®クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。
- [Paste] Windows®クリップボードの内容を貼り付けます。
- [Clear window] 本ウィンドウの内容をクリアします。

入出力を行うための手順は、「3.12 標準入出力およびファイル入出力処理」を参照してください。

### 5.25 Stack Trace



Kind	Name	Value
F	func3(short *)	{ 0x00000094 }
P	param_3	0x00003ffa { 0x00003fd8 } (short*)
L	local_3	D*3 { 0x00003fd4 } (unsigned long)
F	func2(short *)	{ 0x00000072 }
P	param_2	0x00003ffa { 0x00003fe4 } (short*)
L	local_2	D*2 { 0x00003fe0 } (unsigned long)
F	func1(short *)	{ 0x0000003e }
P	param_1	0x00003ffa { 0x00003ff0 } (short*)
L	local_1	D*1 { 0x00003fec } (unsigned long)
F	main()	{ 0x00000012 }
L	start	D*103 { 0x00003ffa } (short)

図5-31 Stack Trace ウィンドウ

本ウィンドウは、関数呼び出し履歴を表示します。  
表示する項目は以下の通りです。

- [Kind] 該当シンボルのシンボル種別を示します。
  - F: 関数
  - P: 関数パラメータ
  - L: ローカル変数
- [Name] シンボル名を示します。
- [Value] シンボルの値、アドレス、型を示します。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

#### 5.25.1 Copy

反転表示されたテキストを Windows®クリップボードにコピーし、他のアプリケーションに貼り付けられるようにします。

#### 5.25.2 Go to Source

選択した関数に該当するソースプログラムを Source ウィンドウ上に表示します。

### 5.25.3 View Setting...

Stack Trace Setting...ダイアログボックスが開き、Stack Trace ウィンドウの表示形式を設定します。

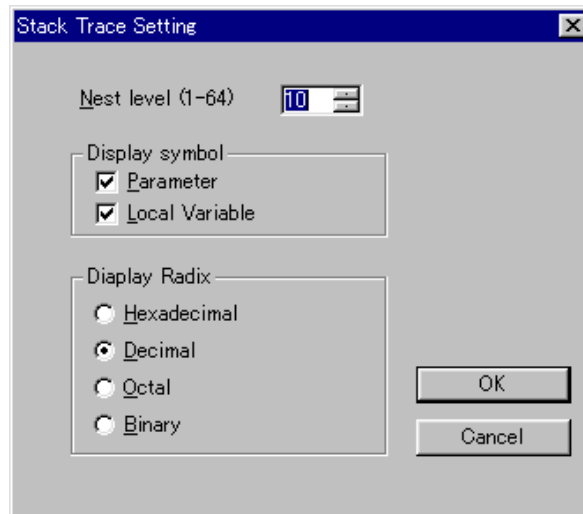
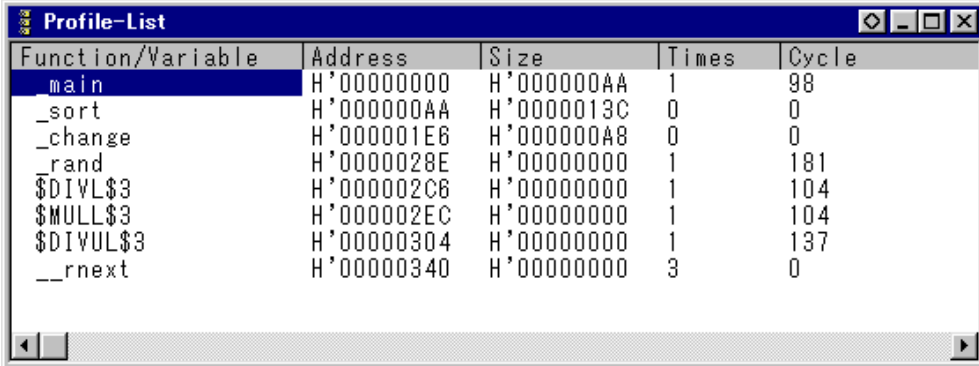


図5-32 Stack Trace Setting ダイアログボックス

Nest level は、Stack Trace ウィンドウに表示する関数コールネスト数を指定します。  
Display symbol グループのチェックボックスは、関数以外に表示するシンボルを指定します。  
Display Radix グループのラジオボタンは、Stack Trace ウィンドウの表示基数を指定します。

## 5.26 Profile-List



Function/Variable	Address	Size	Times	Cycle
main	H'00000000	H'000000AA	1	98
_sort	H'000000AA	H'0000013C	0	0
_change	H'000001E6	H'000000A8	0	0
_rand	H'0000028E	H'00000000	1	181
\$DIVL\$3	H'000002C6	H'00000000	1	104
\$MULL\$3	H'000002EC	H'00000000	1	104
\$DIVUL\$3	H'00000304	H'00000000	1	137
__rnext	H'00000340	H'00000000	3	0

図5-33 Profile-List ウィンドウ

本ウィンドウは、関数とグローバル変数のアドレス、サイズ、関数呼出し回数、およびプロファイルデータを表示します。表示するプロファイルデータは以下の通りです。

- Called (グローバル変数アクセス回数)、Cycle (実行サイクル数)

実行サイクル数は、当該関数コール命令実行時の累計実行サイクル数と当該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。

カラムヘッダをクリックすると、アルファベット順または昇降順にソートして表示します。

Function/Variable 列または Address 列をダブルクリックすると、該当するアドレスに対応したソースプログラムまたは逆アセンブルを表示します。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

### 5.26.1 View Source

選択している行の該当アドレスに対応したソースプログラムまたは逆アセンブルを表示します。選択している行がグローバル変数の場合、このメニューオプションはグレー表示となります。

### 5.26.2 View Profile-Tree

Profile-Tree ウィンドウを表示します。

### 5.26.3 View Profile-Chart

選択している行の関数に着目した Profile-Chart ウィンドウを表示します。

### 5.26.4 Enable Profiler

プロファイルデータ収集のオン・オフを切り替えます。プロファイルデータ測定がアクティブであれば、メニューテキストの左にチェックマークを表示します。なお、プロファイルデータと実行効率データは同時に計測することはできません。プロファイルデータ収集をオンにするときに、実

行効率データ収集がオンされている場合（Performance Analysis ウィンドウの Enable Analysis にチェックされている場合）、警告メッセージボックスを表示します。

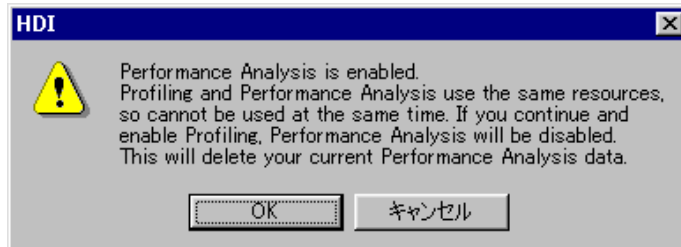


図5-34 Profiler および Analysis の同時設定不可警告メッセージボックス

[OK]ボタンを選択すると、実行効率データ収集がオフに、プロファイルデータ収集がオンに切り替わります。

### 5.26.5 Find...

Function/Variable 列の文字列を検索する Find Text ダイアログボックスを表示します。検索したい文字列をエディットボックスに入力し、[Find Next]ボタンまたは、ENTER キーを入力すると、検索を開始します。

### 5.26.6 Clear Data

関数呼び出し回数のカウントおよびプロファイルデータをクリアします。Profile-Tree ウィンドウおよび Profile-Chart ウィンドウのデータもクリアします。

### 5.26.7 Output Profile Information File...

Save Profile Information File ダイアログボックスを表示します。プロファイル結果をプロファイル情報ファイル（拡張子は".pro"）に保存します。最適化リンケージエディタは、プロファイル情報を元に、ユーザプログラムの最適化を行うことができます。プロファイル情報を使用した最適化についての詳細は、最適化リンケージエディタのマニュアルを参照してください。

### 5.26.8 Output Text File...

Save Text of Profile Data ダイアログボックスを表示します。表示している状態をテキストファイルに保存します。

### 5.26.9 Select Data...

プロファイルデータの種類を選択することができます。プロファイルデータの種類は、デバッグプラットフォームにより異なります。デバッグプラットフォームがサポートしていない場合、このメニューオプションはグレー表示となります。

### 5.26.10 Setting...

表示状態の設定を行う Setting Profile-List ダイアログボックスを表示します。

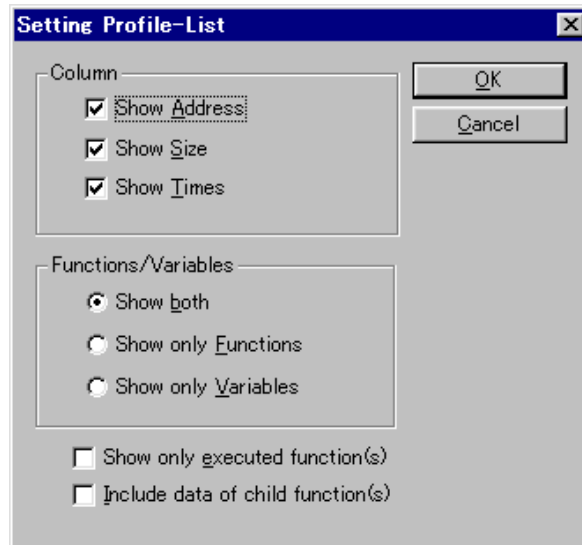


図5-35 Setting Profile-List ダイアログボックス

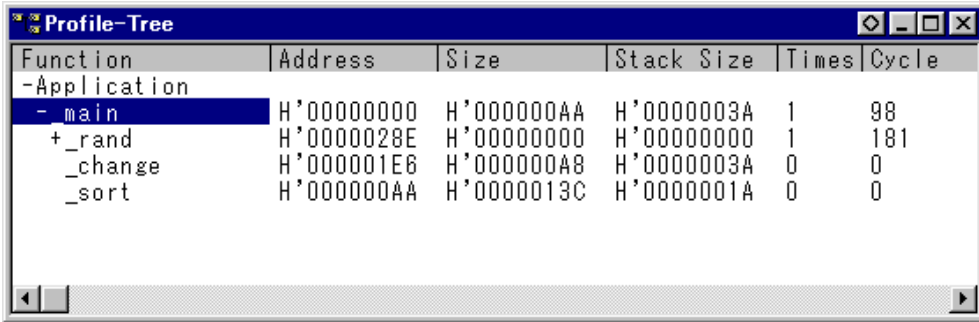
Column グループのチェックボックスは、カラムの表示 / 非表示を設定します。

Functions/Variables グループのラジオボタンは、Function/Variable 列で表示している関数およびグローバル変数を両方表示するかどちらか一方のみ表示するかを設定します。

Show Only Executed Function(s)チェックボックスは、未実行関数の表示を抑制することができます。最適化リンケージエディタが出力するスタック使用量情報ファイル（拡張子は".sni"）が無い場合、このチェックボックスの設定に関わらず、未実行関数は表示しません。

Include Data of Child Function(s)チェックボックスは、表示するプロファイルデータに、関数内で呼び出されている子関数のプロファイルデータを含めるかどうかを設定します。

## 5.27 Profile-Tree



Function	Address	Size	Stack Size	Times	Cycle
-Application					
- main	H'00000000	H'000000AA	H'0000003A	1	98
+ _rand	H'0000028E	H'00000000	H'00000000	1	181
_change	H'000001E6	H'000000A8	H'0000003A	0	0
_sort	H'000000AA	H'0000013C	H'0000001A	0	0

図5-36 Profile-Tree ウィンドウ

本ウィンドウは、関数の呼出し関係をつリー構造で表示します。また、各関数のアドレス、サイズ、スタックサイズ、関数呼出し回数、およびプロファイルデータを表示します。スタックサイズ、関数呼出し回数、およびプロファイルデータは、実際の関数が呼出された経路における値を表示します。

表示するプロファイルデータは以下の通りです。

- Cycle (実行サイクル数)

実行サイクル数は、当該関数コール命令実行時の累計実行サイクル数と当該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。

【注】 スタックサイズは実際の値とは異なります。関数の呼び出し経路における目安としてください。また、最適化リンケージエディタが出力するスタック使用量情報ファイル（拡張子は".sni"）が無い場合には、スタックサイズを表示しません。スタック使用量情報ファイルについては、最適化リンケージエディタのマニュアルを参照してください。

Function 列の関数をダブルクリックすると、ツリー構造を拡張または収縮表示します。また、'+'/-キーでも拡張/収縮表示することができます。Address 列をダブルクリックすると、該当するアドレスに対応したソースプログラムまたは逆アセンブルを表示します。

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには以下のオプションが含まれています。

### 5.27.1 View Source

選択している行の該当アドレスに対応したソースプログラムまたは逆アセンブルを表示します。

### 5.27.2 View Profile-List

Profile-List ウィンドウを表示します。

### 5.27.3 View Profile-Chart

選択している行の関数に着目した Profile-Chart ウィンドウを表示します。

### 5.27.4 Enable Profiler

プロファイルデータ収集のオン・オフを切り替えます。プロファイルデータ測定がアクティブであれば、メニューテキストの左にチェックマークを表示します。なお、プロファイルデータと実行効率データは同時に計測することはできません。プロファイルデータ収集をオンにするときに、実行効率データ収集がオンされている場合（Performance Analysis ウィンドウの Enable Analysis にチェックされている場合）、警告メッセージボックスを表示します。

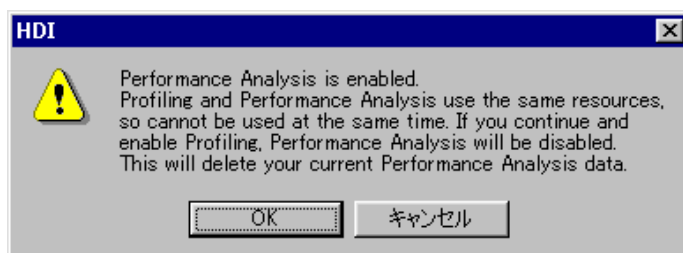


図5-37 Profiler および Analysis の同時設定不可警告メッセージボックス

[OK]ボタンを選択すると、実行効率データ収集がオフに、プロファイルデータ収集がオンに切り替わります。

### 5.27.5 Find...

Function 列の文字列を検索する Find Text ダイアログボックスを表示します。検索したい文字列をエディットボックスに入力し、[Find Next]ボタンまたは、ENTER キーを入力すると、検索を開始します。

### 5.27.6 Find Data...

Find Data ダイアログボックスを表示します。なお、カーソルが Function 列にある場合、このメニューオプションはグレー表示となります。

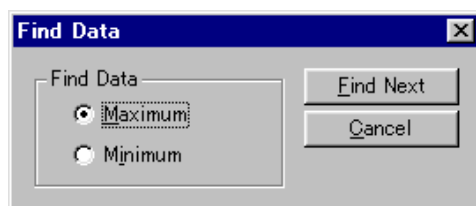


図5-38 Find Data ダイアログボックス

Find Data グループより検索方向を設定し、[Find Next]ボタンまたは、ENTER キーを入力すると、検索を開始します。また、連続して[Find Next]ボタンまたは ENTER キーを入力すると、次に大きいデータ（Minimum の場合は小さいデータ）を検索します。

### 5.27.7 Clear Data

関数呼び出し回数のカウントおよびプロファイルデータをクリアします。Profile-List ウィンドウ

および Profile-Chart ウィンドウのデータもクリアします。

### 5.27.8 Output Profile Information File...

Save Profile Information File ダイアログボックスを表示します。プロファイル結果をプロファイル情報ファイル（拡張子は".pro"）に保存します。最適化リンケージエディタは、プロファイル情報を元に、ユーザプログラムの最適化を行うことが出来ます。プロファイル情報を使用した最適化についての詳細は、最適化リンケージエディタのマニュアルを参照してください。

### 5.27.9 Output Text File...

Save Text of Profile Data ダイアログボックスを表示します。表示している状態をテキストファイルに保存します。

### 5.27.10 Select Data...

プロファイルデータの種類を選択することができます。プロファイルデータの種類は、デバッグプラットフォームにより異なります。デバッグプラットフォームがサポートしていない場合、このメニューオプションはグレー表示となります。

### 5.27.11 Setting...

表示状態の設定を行う Setting Profile-Tree ダイアログボックスを表示します。

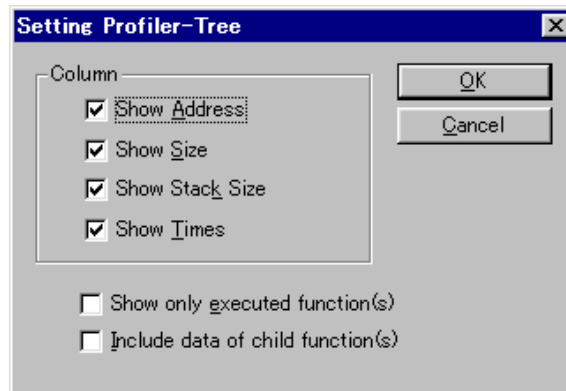


図5-39 Setting Profile-Tree ダイアログボックス

Column グループのチェックボックスは、カラムの表示 / 非表示を設定します。

Show Only Executed Function(s) チェックボックスは、未実行関数の表示を抑制することができます。最適化リンケージエディタが出力するスタック使用量情報ファイル（拡張子は".sni"）が無い場合、このチェックボックスの設定に関わらず、未実行関数は表示しません。

Include Data of Child Function(s) チェックボックスは、表示するプロファイルデータに、関数内で呼び出されている子関数のプロファイルデータを含めるかどうかを設定します。



## 5.28 Profile-Chart

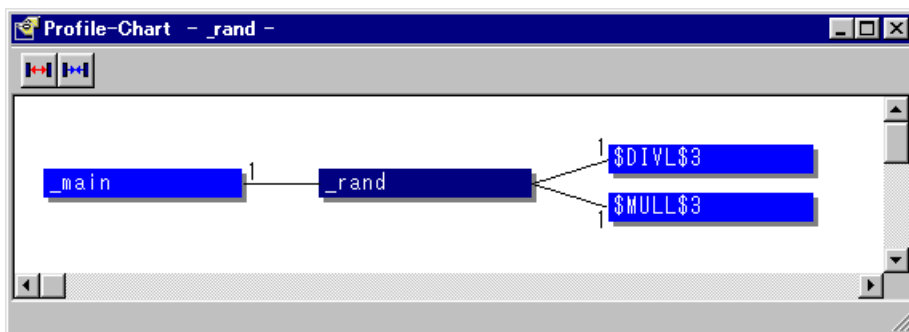


図5-40 Profile-Chart ウィンドウ


ある関数に着目した呼び出し関係を表示します。Profile-Chart ウィンドウは、Profile-List ウィンドウまたは Profile-Tree ウィンドウから開きます。Profile-List ウィンドウまたは Profile-Tree ウィンドウで選択した関数に着目した呼び出し関係を表示します。着目した関数を中心に、左側に呼び出し元関数、右側に呼び出し先関数を表示します。呼び出し元関数および呼び出し先関数横の数値は、呼び出し回数を示します。

Profile-Chart ウィンドウには以下のツールバーボタンが含まれています。

- Expands Size
- Reduces Size

ウィンドウ内でマウスの右ボタンをクリックするとポップアップメニューを表示します。このメニューには「5.28.3 View Source」以降のオプションが含まれています。

### 5.28.1 Expands Size

 各関数の間隔を広げて表示します。また、`+`キーでも広げて表示することができます。

### 5.28.2 Reduces Size

 各関数の間隔を縮めて表示します。また、`-`キーでも縮めて表示することができます。

### 5.28.3 View Source

マウスの右ボタンをクリックしたときの位置にある関数の該当アドレスに対応したソースプログラムまたは逆アセンブルを表示します。マウスの右ボタンをクリックしたときの位置が関数ではない場合、このメニューオプションはグレー表示となります。

### 5.28.4 View Profile-List

Profile-List ウィンドウを表示します。

### 5.28.5 View Profile-Tree

Profile-Tree ウィンドウを表示します。

### 5.28.6 View Profile-Chart

マウスの右ボタンをクリックしたときの位置にある関数に着目した Profile-Chart ウィンドウを表示します。マウスの右ボタンをクリックしたときの位置が関数ではない場合、このメニューオプションはグレー表示となります。

### 5.28.7 Enable Profiler

プロファイルデータ収集のオン・オフを切り替えます。プロファイルデータ測定がアクティブであれば、メニューテキストの左にチェックマークを表示します。なお、プロファイルデータと実行効率データは同時に計測することはできません。プロファイルデータ収集をオンにするときに、実行効率データ収集がオンされている場合（Performance Analysis ウィンドウの Enable Analysis にチェックされている場合）、警告メッセージボックスを表示します。

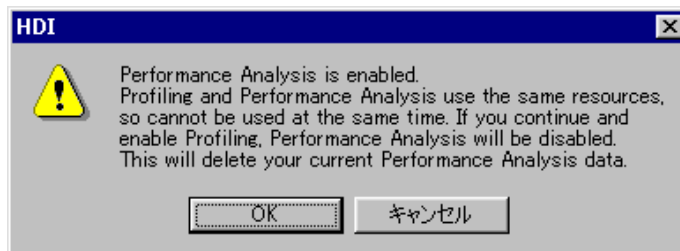


図5-41 Profiler および Analysis の同時設定不可警告メッセージボックス

[OK]ボタンを選択すると、実行効率データ収集がオフに、プロファイルデータ収集がオンに切り替わります。

### 5.28.8 Clear Data

関数呼び出し回数のカウントおよびプロファイルデータをクリアします。Profile-List ウィンドウおよび Profile-Tree ウィンドウのデータもクリアします。

### 5.28.9 Multiple View

Profile-Chart ウィンドウを表示する際、既に Profile-Chart ウィンドウが開いているとき、別のウィンドウを開くか同ウィンドウに表示するかを設定します。メニューテキストの左にチェックマークが表示されていれば、別のウィンドウを開きます。

### 5.28.10 Output Profile Information File...

Save Profile Information File ダイアログボックスを表示します。プロファイル結果をプロファイル情報ファイル（拡張子は".pro"）に保存します。最適化リンケージエディタは、プロファイル情報を元に、ユーザプログラムの最適化を行うことが出来ます。プロファイル情報を使用した最適化についての詳細は、「最適化リンケージエディタのマニュアル」を参照してください。

---

## 6. コマンドライン

---

コマンド一覧を表 6-1 に示します。

表 6-1 コマンド一覧

コマンド名	短縮形	説明
!	-	コメント
ANALYSIS	AN	性能分析機能の有効化 / 無効化
ANALYSIS_RANGE	AR	性能評価関数の設定、表示
ANALYSIS_RANGE_DELETE	AD	性能分析範囲の削除
ASSEMBLE	AS	アセンブルの実行
ASSERT	-	コンディションのチェック
BREAKPOINT	BP	実行命令位置によるブレークポイントの設定
BREAK_ACCESS	BA	メモリ範囲のアクセスによるブレーク条件の設定
BREAK_CLEAR	BC	ブレークポイントの削除
BREAK_DATA	BD	メモリのデータ値によるブレーク条件の設定
BREAK_DISPLAY	BI	ブレークポイント一覧の表示
BREAK_ENABLE	BE	ブレークポイントの有効/無効の切り換え
BREAK_REGISTER	BR	レジスタのデータ値によるブレーク条件の設定
BREAK_SEQUENCE	BS	実行順序を指定したブレークポイントの設定
DISASSEMBLE	DA	逆アセンブル表示
ERASE	ER	Command Line ウィンドウの内容のクリア
EVALUATE	EV	式の計算
FILE_LOAD	FL	オブジェクト(プログラム)ファイルのロード
FILE_SAVE	FS	メモリ内容のファイルセーブ
FILE_VERIFY	FV	ファイル内容とメモリ内容の比較
GO	GO	ユーザプログラムの実行
GO_RESET	GR	リセットベクタからのユーザプログラムの実行
GO_TILL	GT	テンポラリブレークポイントまでのユーザプログラムの実行
HALT	HA	ユーザプログラムの停止
HELP	HE	コマンドラインまたはコマンドに対するヘルプ表示
INITIALISE	IN	HDI の初期化
LOG	LO	ロギングファイルの操作
MAP_DISPLAY	MA	メモリマッピング情報の表示
MAP_SET	MS	メモリリソースの設定
MEMORY_DISPLAY	MD	メモリ内容の表示
MEMORY_EDIT	ME	メモリ内容の変更
MEMORY_FILL	MF	指定データによるメモリ内容の一括変更
MEMORY_MOVE	MV	メモリブロックの移動
MEMORY_TEST	MT	メモリブロックのテスト
QUIT	QU	HDI の終了

## 6. コマンドライン

コマンド名	短縮形	説明
RADIX	RA	入力ラディックス(基数)の設定
REGISTER_DISPLAY	RD	CPU レジスタ値の表示
REGISTER_SET	RS	CPU レジスタ値の設定
RESET	RE	CPU のリセット
SLEEP	-	コマンド実行の遅延
STEP	ST	ステップ実行(命令単位またはソース行単位)
STEP_OUT	SP	PC 位置の関数を終了するまでのステップ実行
STEP_OVER	SO	ステップオーバー実行
STEP_RATE	SR	ステップ実行速度の設定、表示
SUBMIT	SU	コマンドファイルの実行
SYMBOL_ADD	SA	シンボルの設定
SYMBOL_CLEAR	SC	シンボルの削除
SYMBOL_LOAD	SL	シンボル情報ファイルのロード
SYMBOL_SAVE	SS	シンボル情報のファイルセーブ
SYMBOL_VIEW	SV	シンボルの表示
TRACE	TR	トレース情報の表示
TRACE_ACQUISITION	TA	トレース情報取得の有効/無効の切換え

以下に各コマンドのシンタックスを示します。

!(コメント)

省略形: なし

説明:

ログファイル等への記録に便利な、コメントを出力することができます。

シンタックス

! <text>

パラメータ	型	説明
<text>	テキスト	出力するテキスト

例:

! Start of test routine

Command Line ウィンドウ (ログが有効の場合はログファイル) にコメント "Start of test routine" を出力します

ANALYSIS

省略形: AN

説明:

性能分析を有効、または無効にします。分析数は、実行前に自動的にリセットしません。

## シンタックス

an [ &lt;state&gt; ]

パラメータ	型	説明
なし		分析状況を表示します
<state>	キーワード	分析を設定します
	enable	分析を可能にします
	disable	分析を無効にします
	reset	分析数をリセットします

例:

ANALYSIS            分析状況を表示します  
 AN enable            分析を可能にします  
 AN disable           分析を無効にします  
 AN reset             分析数をリセットします

## ANALYSIS\_RANGE

省略形: AR

説明:

性能評価を行う関数を設定するか、またはパラメータなしで性能評価を行う関数を表示します。

## シンタックス

ar [ &lt;関数名 t&gt; ]

パラメータ	型	説明
なし		すべての性能評価を行う関数を表示します
<関数名>	文字列	性能評価を行う関数名

例:

ANALYSIS\_RANGE sort    関数 sort の性能評価を行います  
 AR                        性能評価を行う関数を表示します

## ANALYSIS\_RANGE\_DELETE

省略形: AD

説明:

指定された項目番号の関数、またはパラメータが何も指定されていなかったらすべての関数を削除します(確認を求められません)。

## シンタックス

ad [ &lt;index&gt; ]

パラメータ	型	説明
なし		すべての関数を削除します
<index>	数値	削除する関数の番号

## 6. コマンドライン

---

例:

ANALYSIS\_RANGE\_DELETE 6 項目番号 6 の関数を削除します  
AD 全ての関数を削除します

### ASSEMBLE

省略形: AS

説明:

アセンブルし、メモリに書き込みます。  
アセンブルモードでは "." は終了、"^" は1バイト戻り、ENTERキーを押すと先に進みます。  
シンタックス

as <address>

パラメータ	型	説明
<address>	数値	アセンブルを開始するアドレス

例:

AS H'1000 H'1000 からアセンブルします

### ASSERT

省略形: なし

説明:

式が真であることを調べます。式が偽のときはバッチファイルを終了するため、バッチファイルで使えます。式が偽のときエラーを返します。このコマンドは、サブルーチンのテストハーネスを記述するために使うことができます。

シンタックス

assert <expression>

パラメータ	型	説明
<expression>	式	判定する式

例:

ASSERT #R0 == 0x100 R0 が 0x100 を含んでいないときエラーを返します

### BREAKPOINT

省略形: BP

説明:

実行命令位置によるブレークポイントを設定します。

## シンタックス

bp &lt;address&gt; [&lt;count&gt;]

パラメータ	型	説明
<address>	数値	ブレークポイントのアドレス
<count>	数値	指定アドレスの命令をフェッチする回数（任意、デフォルト=1）

例:

BREAKPOINT 0 2      0 番地の命令を 2 回目に実行しようとしたとき、ブレークするように設定します。

BP C0                C0 番地の命令を実行しようとしたとき、ブレークするように設定します。

## BREAK\_ACCESS

省略形: BA

説明:

メモリ範囲のアクセスによるブレーク条件の設定します。

## シンタックス

ba &lt;start address&gt; [&lt;end address&gt;] [&lt;mode&gt;]

パラメータ	型	説明
<start address>	数値	ブレークポイントの開始アドレス
<end address>	数値	ブレークポイントの終了アドレス（任意、デフォルト=開始アドレス）
<mode>	キーワード	アクセス種別（任意、デフォルト=RW）
	R	リードした場合にブレーク
	W	ライトした場合にブレーク
	RW	リードまたはライトした場合にブレーク

例:

BREAK\_ACCESS 0 1000 W      0 番地から 1000 番地の範囲でライトアクセスが発生したときブレークするように設定します。

BA FFFF                FFFF 番地でリードまたはライトアクセスが発生したときブレークするように設定します。

## BREAK\_CLEAR

省略形: BC

説明:

ブレークポイントを削除します。

## シンタックス

bc &lt;index&gt;

パラメータ	型	説明
<index>	数値	削除するブレークポイントのインデックス（省略した場合は全てのブレークポイントを削除します。）





例:  
 BREAK\_DISPLAY      ブレークポイントの一覧を表示します。  
 BI                    ブレークポイントの一覧を表示します。

## BREAK\_ENABLE

省略形: BE

説明:  
 ブレークポイントの有効/無効を切換えます。

## シンタックス

be <flag> [<index>]

パラメータ	型	説明
<flag>	キーワード	有効/無効
	E	有効
	D	無効
<index>	数値	有効/無効を切りかえるブレークポイントのインデックス(省略した場合は全てのブレークポイントを対象とします。)

例:  
 BREAK\_ENABLE D 0      1番目のブレークポイントを無効にします。  
 BE E                    全てのブレークポイントを有効にします。

## BREAK\_REGISTER

省略形: BR

説明:  
 レジスタのデータ値によるブレーク条件を設定します。

## シンタックス

br <register> [<data> <size>] [<option>]

パラメータ	型	説明
<register>	文字列	レジスタ名
<data>	数値	アクセスデータ
<size>	キーワード	アクセスサイズ(任意、省略した場合は指定レジスタのサイズとします。ただし、データ値設定時は省略不可となります。)
	B	バイトデータ
	W	ワードデータ
	L	ロングワードデータ
	S	単精度浮動小数点データ
	D	倍精度浮動小数点データ
<option>	キーワード	データ的一致/不一致(任意、デフォルト=EQ)
	EQ	データが一致したときブレーク
	NE	データが不一致となったときブレーク

## 6. コマンドライン

---

例:

BREAK\_REGISTER R0 FFFF W EQ R0 レジスタの下位 2 バイトが FFFF になったときブレークします。

BR R10 R10 レジスタにライトアクセスが発生したときブレークします。

### BREAK\_SEQUENCE

省略形: BS

説明:

実行順序を指定したブレークポイントを設定します。

シンタックス

bs <address1> [<address2> [<address3> [...]]]

パラメータ	型	説明
<address1> ~ <address8>	数値	シーケンシャルブレークポイントとなるアドレス (最大 8 個まで指定できます。)

例:

BREAK\_SEQUENCE 1000 2000 通過位置が 1000 番地、2000 番地するときブレークします。

BS 1000 1000 番地を実行するときブレークするように設定します。

### DISASSEMBLE

省略形: DA

説明:

メモリ内容を逆アセンブル表示します。逆アセンブル表示は完全なシンボリックです。

シンタックス

da <address> [<length>]

パラメータ	型	説明
<address>	数値	開始アドレス
<length>	数値	命令数 (任意、デフォルト=16)

例:

DISASSEMBLE H'100 5 アドレス H'100 からコード 5 行を逆アセンブル表示します

DA H'3E00 20 アドレス H'3E00 からコード 20 行を逆アセンブル表示します

### ERASE

省略形: ER

説明:

Command Lineウィンドウをクリアします。



## 6. コマンドライン

---

### FILE\_LOAD

省略形: FL

#### 説明:

指定されたオフセットで、メモリに対して指定されたファイルをロードします。現在のシンボルはクリアしますが、新しいシンボルが同一名で現在のシンボルを重複定義します。オフセットが指定されるとシンボルに加えられます。ファイルの拡張子はデフォルトとして ".MOT" を設定します。

#### シンタックス

fl <filename> [ <offset> ] [ <state> ]

パラメータ	型	説明
<filename>	文字列	ファイル名
<offset>	数値	ロードアドレスに加えるオフセットを設定します (任意、デフォルト=0)
<state>	キーワード	ベリファイフラグ (任意、デフォルト=V)
	V	ベリファイあり
	N	ベリファイなし

#### 例:

FILE\_LOAD A:¥¥BINARY¥¥TESTFILE.A22 S レコードファイル "testfile.a22" をロードします  
FL ANOTHER.MOT H'200 モトローラ S レコードファイル "another.mot" をオフセット H'200 バイトからロードします

## FILE\_SAVE

省略形: FS

## 説明:

メモリ内容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します。すでに存在するファイル名を指定した場合は、上書きするかどうかユーザに確認します。ファイルの拡張子のデフォルトは、".MOT"です。シンボルは自動的にセーブしません。

## シンタックス

fs &lt;filename&gt; &lt;start&gt; &lt;end&gt;

パラメータ	型	説明
<filename>	文字列	ファイル名
<start>	数値	開始アドレス
<end>	数値	終了アドレス

## 例:

FILE\_SAVE TESTFILE H'0 H'2013

アドレス範囲 H'0 - H'2013 をモトローラ S レコードファイル "TESTFILE.MOT" として保存します

FS D:¥¥USER¥¥ANOTHER.A22 H'4000 H'4FFF

アドレス範囲 H'4000 - H'4FFF を S レコードフォーマットファイル "ANOTHER.A22" として保存します

## FILE\_VERIFY

省略形: FV

## 説明:

メモリとファイル内容をベリファイします。ファイルデータはモトローラ S レコードフォーマットです。ファイルの拡張子はデフォルトとして".MOT"を設定します。

## シンタックス

fv &lt;filename&gt; [ &lt;offset&gt; ]

パラメータ	型	説明
<filename>	文字列	ファイル名
<offset>	数値	ファイルアドレスへ加えるオフセットを設定します (任意、デフォルト=0)

## 例:

FILE\_VERIFY A:¥¥BINARY¥¥TEST.A22

メモリに対して S レコードファイル "TEST.A22" をベリファイします

FV ANOTHER 200

メモリに対して S レコードファイル "ANOTHER.MOT" にオフセット H'200 バイトを加えたアドレスからベリファイします



例:  
GR                   リセットベクタで指定されているアドレスから始まる、ユーザプログラムを実行します (コマンド処理を続けることができません)

## GO\_TILL

省略形: GT

## 説明:

テンポラリブレークポイントを設定し、現在の PC からプログラムを実行します。 テンポラリブレークポイントとして、複数のアドレスを指定することができます (テンポラリブレークポイントは、コマンド実行中のみ有効です)。

## シンタックス

gt [ <state> ] <address>...

パラメータ	型	説明
<state>	キーワード	プログラム実行中でのコマンド処理の有効 / 無効
	wait	コマンド処理を続けることができません
	continue	コマンド処理を続けることができます
<address>...	数値	テンポラリブレークポイントのアドレス (複数指定可)

wait はデフォルトで、プログラムが実行を停止するまでコマンド処理ができません。

continue は、コマンド処理を続けることができます (デバッグプラットフォームの機能により動作しないものもあります)。

## 例:

GO\_TILL H'1000 PC がアドレス H'1000 に到達するまでユーザプログラムを実行します

## HALT

省略形: HA

## 説明:

ユーザプログラムを停止します ("go continue"コマンドの後で、使うことができます)。

## シンタックス

ha

パラメータ	型	説明
なし		ユーザプログラムを停止します

## 例:

HA                   ユーザプログラムを停止します



## 6. コマンドライン

---

### HELP

省略形: HE

説明:

ヘルプファイルを開きます。

コンテキスト詳細ヘルプは、F1 キーで表示できます。HELP または HE に続いてコマンド名を入力することにより検索できます。

シンタックス

he [ <command> ]

パラメータ	型	説明
なし		ヘルプの目次を表示します
<command>	文字列	コマンドを指定します

例:

HE                   ヘルプの目次を表示します  
HE GO               GO コマンドのヘルプを表示します

### INITIALISE

省略形: IN

説明:

HDI ( デバッグプラットフォーム、ターゲットを含む ) を初期化 ( ターゲットライブラリを再選択する ) します。すべてのブレイクポイント、メモリマッピングなどもリセットします。

シンタックス

in

パラメータ	型	説明
なし		HDI を初期化します

例:

IN                   HDI を初期化します

## LOG

省略形: LO

## 説明:

ファイルへのコマンド出力のログを制御します。パラメータなしでは、現在のログ状況を表示します。存在するファイルを指定すると、追加するかをユーザに確認します。No と答えるとデータはファイルに上書きされ、Yes と答えるとファイルに追加されます。ロギングはコマンドラインインタフェースでのみサポートします。

## シンタックス

lo [ &lt;state&gt; | &lt;filename&gt; ]

パラメータ	型	説明
なし		ロギング状態を表示します
<state>	キーワード	ロギングを再開 / 停止します
	+	ロギングを再開します
	-	ロギングを停止します
<filename>	文字列	ロギングを出力するファイル名を指定します

## 例:

LOG TEST      ファイル TEST への出力をロギングします  
 LO -            ロギングを停止します  
 LOG +          ロギングを再開します  
 LOG            現在のロギング状態を表示します

## 6. コマンドライン

---

### MAP\_DISPLAY

省略形: MA

説明:

現在のメモリマップ構成を表示します。

シンタックス

ma

パラメータ	型	説明
なし		現在のメモリマップ構成を表示します

例:

MA 現在のメモリマップ構成を表示します

### MAP\_SET

省略形: MS

説明:

メモリリソースを設定します。

シンタックス

ms <start address> [ <end address> ] [ <mode> ]

パラメータ	型	説明
<start address>	数値	開始アドレス
<end address>	数値	終了アドレス
<mode>	キーワード	アクセス種別 (任意、デフォルト=RW)
	R	リードのみ可
	W	ライトのみ可
	RW	リード/ライト可

例:

MAP\_SET 0000 3FFF RW 0000 番地から 3FFF 番地をリード/ライト可能な領域として確保します。

MS 5000 5000 番地をリード/ライト可能な領域として確保します。

### MEMORY\_DISPLAY

省略形: MD

説明:

メモリ内容を表示します。

## シンタックス

md &lt;address&gt; [ &lt;length&gt; ] [ &lt;mode&gt; ]

パラメータ	型	説明
<address>	数値	開始アドレス
<length>	数値	長さ(任意、デフォルト=H'100 バイト)
<mode>	キーワード	フォーマット(任意、デフォルト=byte)
	byte	バイトとして表示します
	word	ワードとして表示します(2 バイト)
	long	ロングワードとして表示します(4 バイト)
	ascii	ASCII として表示します
	single	単精度浮動小数点として表示します
	double	倍精度浮動小数点として表示します

例:

MEMORY\_DISPLAY H'C000 H'100 WORD      H'C000 から始まるメモリを H'100 バイト分ワード  
 フォーマットで表示します

MEMORY\_DISPLAY H'1000 H'FF              H'1000 から始まるメモリを H'FF バイト分バイトフ  
 ォーマットで表示します

## MEMORY\_EDIT

省略形: ME

説明:

メモリ内容を変更します。メモリを変更するとき、現在位置は ASSEMBLE コマンドで記述したときと同じような方法で変更することができます。"." を入力すると変更モードを終了し、"^" は1ユニット戻り、空白は変更せずに進みます。

## シンタックス

me &lt;address&gt; [ &lt;mode&gt; ] [ &lt;state&gt; ]

パラメータ	型	説明
<address>	数値	変更するアドレス
<mode>	キーワード	フォーマット(任意、デフォルト=byte)
	byte	バイトとして変更します
	word	ワードとして変更します
	long	ロングワードとして変更します
	ascii	ASCII として変更します
	single	単精度浮動小数点として表示します
	double	倍精度浮動小数点として表示します
<state>	キーワード	ベリファイフラグ(任意、デフォルト=V)
	V	ベリファイあり
	N	ベリファイなし

例:

ME H'1000 WORD      H'1000 から word フォーマットでメモリ内容を変更します(ベリファイあり)



MV H'FB80 H'FF7F H'3000

領域 H'FB80 - H'FF7F を H'3000 へ移動します

## MEMORY\_TEST

省略形: MT

### 説明:

指定されたアドレス範囲で、リード・ライト・ベリファイのテストを行います。このときのデータは破壊されます。マップ設定に従ってメモリテストを行います。本シミュレータ・デバッガではサポートしていません。

### シンタックス

mt <start> <end>

パラメータ	型	説明
<start>	数値	開始アドレス
<end>	数値	終了アドレス (この値を含む)

### 例:

MEMORY\_TEST H'8000 H'BFFF

H'8000 から H'BFFF までテストします

MT H'4000 H'5000

H'4000 から H'5000 までテストします

## 6. コマンドライン

---

### QUIT

省略形: QU

説明:

HDI を終了します。 ログファイルがオープンしていると閉じられます。

シンタックス

qu

パラメータ	型	説明
なし		HDI を終了します

例:

QU HDI を終了します

### RADIX

省略形: RA

説明:

デフォルトの基数を設定、または表示します。 パラメータなしで基数を表示します。 基数は数値データの前の B/H/D/O を使って変更できます。

シンタックス

ra [ <mode> ]

パラメータ	型	説明
なし		現在の基数を表示します
<mode>	キーワード	基数指定子
	H	16 進数
	D	10 進数
	O	8 進数
	B	2 進数

例:

RADIX 現在の基数を表示します

RA H 基数を 16 進数にします

### REGISTER\_DISPLAY

省略形: RD

説明:

CPUレジスタ値を表示します。

シンタックス

rd

パラメータ	型	説明
なし		全レジスタの内容を表示します

例:  
RD           全レジスタの内容を表示します

## REGISTER\_SET

省略形: RS

説明:  
CPUレジスタ値を変更します。

## シンタックス

rs <register> <value> <mode>

パラメータ	型	説明
<register>	キーワード	レジスタ名
<value>	数値	レジスタ値
<mode>	キーワード	データサイズ (任意、デフォルト=レジスタサイズ)
	byte	バイト
	word	ワード
	long	ロングワード
	single	単精度浮動小数点
	double	倍精度浮動小数点

例:  
RS PC \_SstartUp           プログラムカウンタをシンボル\_StartUp に設定します  
RS R0 H'1234 WORD       R0 にワードデータ H'1234 に設定します

## RESET

省略形: RE

説明:  
プロセッサをリセットします。すべてのレジスタ値はデバイスの初期化状態となります。  
メモリマッピングとブレークポイントには影響しません。

## シンタックス

re

パラメータ	型	説明
なし		プロセッサをリセットします

例:  
RE           プロセッサをリセットします



## 6. コマンドライン

---

### SLEEP

省略形: なし

説明:

指定されたミリ秒の間コマンド実行を遅延します。

シンタックス

sleep <milliseconds>

パラメータ	型	説明
<milliseconds>	数値	遅延時間 (ミリ秒)

基数 D' は、明示して使わないとデフォルトの基数 (必ずしも 10 進ではありません) が使われま  
す。

例:

SLEEP D'9000      9 秒間遅延します

### STEP

省略形: ST

説明:

シングルステップ (ソース行、または命令) を行ないます。現在の PC から指定された命  
令数分ステップします。ソースデバッグが有効な時、デフォルトのステップはソース行と  
なります。ステップ数のデフォルトは1です。

シンタックス

st [ <mode> ] [ <count> ]

パラメータ	型	説明
<mode>	キーワード	ステップの種類 (任意)
	instruction	アセンブラの 1 命令を基準にステップします
	line	ソースコードの 1 行を基準にステップします
<count>	数値	ステップ数 (任意、デフォルト=1)

例:

STEP 9              9 ステップコードをステップします

### STEP\_OUT

省略形: SP

説明:

現在の関数の外へプログラムをステップします (即ち、ステップアップ)。アセンブラと  
ソースレベルデバッグの両方に有効です。

## シンタックス

sp

パラメータ	型	説明
なし		ステップアップします

例:

SP 現在の関数の外へプログラムをステップします

## STEP\_OVER

省略形: SO

説明:

現在の PC から指定された命令数分ステップします。  
このコマンドはサブルーチン、または割り込みルーチンの中でステップしないという点で STEPとは異なります。フルスピードで実行します。

## シンタックス

so [&lt;mode&gt;] [&lt;count&gt;]

パラメータ	型	説明
<mode>	キーワード	ステップの種類 (任意)
	instruction	アセンブラの 1 命令を基準にステップします
	line	ソースコードの 1 行を基準にステップします
<count>	数値	ステップ数 (任意、デフォルト=1)

例:

SO 1 ステップコードをステップオーバーします

## STEP\_RATE

省略形: SR

説明:

STEPとSTEP\_OVERコマンドでステップの速度をコントロールします。6レートは最大限に速くステップします。値が1の場合は最も遅いステップです。

## シンタックス

sr [&lt;rate&gt;]

パラメータ	型	説明
なし		ステップレートを表示します
<rate>	数値	ステップレート 1 から 6 で 6 が最も速くなります

例:

SR 現在設定されているステップレートを表示します

SR 6 ステップレートを最速にします

## 6. コマンドライン

---

### SUBMIT

省略形: SU

説明:

コマンドファイル进行处理します。 処理するファイル中でもこのコマンドを使用できます。 エラーが発生するとファイルの処理を中止します。 [stop]ボタンでプロセスを中止することができます。

シンタックス

su <filename>

パラメータ	型	説明
<filename>	文字列	ファイル名

例:

SUBMIT COMMAND.HDC           COMMAND.HDC ファイル进行处理します  
SU A:SETUP.TXT                 ドライブ A: の SETUP.TXT ファイル进行处理します

### SYMBOL\_ADD

省略形: SA

説明:

新しいシンボルを追加するか、または存在しているシンボルを変更します。

シンタックス

sa <symbol> <value>

パラメータ	型	説明
<symbol>	文字列	シンボル名
<value>	数値	値

例:

SYMBOL\_ADD start H'1000       H'1000 に start を定義します  
SA END\_OF\_TABLE 1000         現在のデフォルト基数を使い、1000 に END\_OF\_TABLE を定義します

### SYMBOL\_CLEAR

省略形: SC

説明:

シンボルを削除します。 パラメータを指定しないとすべてのシンボルを削除します。

シンタックス

sc [<symbol>]

パラメータ	型	説明
なし		すべてのシンボルを削除します
<symbol>	文字列	シンボル名

例:

SYMBOL\_CLEAR                    すべてのシンボルを削除します  
SC start                         シンボル start を削除します

## SYMBOL\_LOAD

省略形: SL

説明:

ファイルからシンボルをロードします。ファイルは XLINK Pentica-b フォーマット（即ち "XXXXH name"）である必要があります。シンボルは存在するシンボルテーブルに加えられます。

シンタックス

sl <filename>

パラメータ	型	説明
<filename>	文字列	ファイル名

例:

SYMBOL\_LOAD TEST.SYM         ファイル TEST.SYM をロードします  
SL MY\_CODE.SYM                ファイル MY\_CODE.SYM をロードします

## SYMBOL\_SAVE

省略形: SS

説明:

すべてのシンボルをファイルへ保存します。ファイルは XLINK Pentica-b フォーマットで、シンボルファイル拡張子は".SYM" をデフォルトとします。ファイル名は、既に存在するときファイルを上書きするプロンプトを表示します。

シンタックス

ss <filename>

パラメータ	型	説明
<filename>	文字列	ファイル名

例:

SYMBOL\_SAVE TEST                TEST.SYM にシンボルテーブルを保存します  
SS MY\_CODE.SYM                MY\_CODE.SYM にシンボルテーブルを保存します

## SYMBOL\_VIEW

省略形: SV

説明:

定義されたすべてのシンボル、または指定した文字列（大文字 / 小文字は区別する）を含んでいるシンボルを表示します。

## 6. コマンドライン

---

### シンタックス

sv [<pattern>]

パラメータ	型	説明
なし		すべてのシンボルを表示します
<pattern>	文字列	文字列を含んでいるシンボル

例:

SYMBOL\_VIEW BUFFER      BUFFER を含んでいるすべてのシンボルを表示します  
SV                              すべてのシンボルを表示します

### TRACE

省略形: TR

説明:

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。

### シンタックス

tr [<start rec> [<count>]]

パラメータ	型	説明
<start rec>	数値	オフセット（任意、デフォルト=最も新しいサイクル-9）
<count>	数値	カウント（任意、デフォルト=10）

例:

TR 0 5      バッファ先頭から 5 行トレースバッファの内容を表示します

### TRACE\_ACQUISITION

省略形: TA

説明:

トレース情報取得の有効/無効を切替えます。

### シンタックス

ta <mode>

パラメータ	型	説明
<mode>	キーワード	トレース情報取得の有効/無効
	E	有効
	D	無効

例:

TRACE\_ACQUISITION E      トレース情報の取得を有効にします  
TA D                              トレース情報の取得を無効にします

---

## 7. メッセージ一覧

---

### 7.1 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 7-1 に示します。

表 7-1 インフォメーションメッセージ一覧

メッセージ	内容
Break Access	ブレイクアクセス条件が成立して実行を中断しました。
Break Data	ブレイクデータ条件が成立して実行を中断しました。
Break Register	ブレイクレジスタ条件が成立して実行を中断しました。
Break Sequence	ブレイクシーケンス条件が成立して実行を中断しました。
PC Breakpoint	ブレイクポイント条件が成立して実行を中断しました。
Sleep	SLEEP 命令により実行を中断しました。
Step Normal End	ステップ実行が正常に終了しました。
Stop	[STOP]ボタンにより実行を中断しました。
Trace Buffer Full	Trace Acquisition ダイアログボックスの Trace buffer full handling で 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。

### 7.2 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 7-2 に示します。

表 7-2 エラーメッセージ一覧

メッセージ	内容・対策
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 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。
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。

## 7. メッセージ一覧

メッセージ	内容・対策
Illegal 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) MOV.B Rn,@-SP または、MOV.B @SP+,Rn を実行しようとした 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。
Illegal Operation	以下のいずれかの状態になりました。 (1) DAA 命令、DAS 命令で CCR の C フラグ、H フラグと補正前の値の関係が不正である (2) DIVXU 命令、DIVXS 命令でゼロ除算または、オーバフローが発生した 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。
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 (5) EEPMOV 命令以外で EEPROM へ書き込みを行おうとした メモリの確保、属性変更を行うか、当該メモリアccess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。
System Call Error	システムコールエラーが発生しました。 レジスタ R0,R1 または ER1 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。

---

## 8. プログラムの表示

---

本章では、プログラムリストをソースコードやアセンブラニーモニックで表示する方法について説明します。また、HDI を使用してコード、ラベル、テキストファイルを表示する方法について述べます。

### 8.1 デバッグを行うためのコンパイル


C/C++言語ソースレベルでプログラムをデバッグするためには、プログラムをデバッグオプションを指定してコンパイル、リンクする必要があります。

【注】 デバッグ用のデバッグオブジェクトファイルを作成する際には、コンパイラおよびリンカージェネレータの両方で、デバッグオプションを指定していることを確認してください。

デバッグオブジェクトファイルにデバッグ情報が入っていない場合でも、そのファイルを HDI へロードできますが、C/C++言語ソースレベルでプログラムをデバッグすることができず、アセンブラレベルのみとなります。

### 8.2 コードの表示

#### 8.2.1 ソースコードの表示

プログラムのソースコードを見るには、[View->Source...]メニューオプションを選択するか、キーボードで Ctrl+K を入力します。あるいは、ツールバー上に Source ボタン  が表示されていれば、このボタンをクリックします。

ソースファイルを選択して[Open]をクリックすると、Source ウィンドウが開きます。



## 8. プログラムの表示

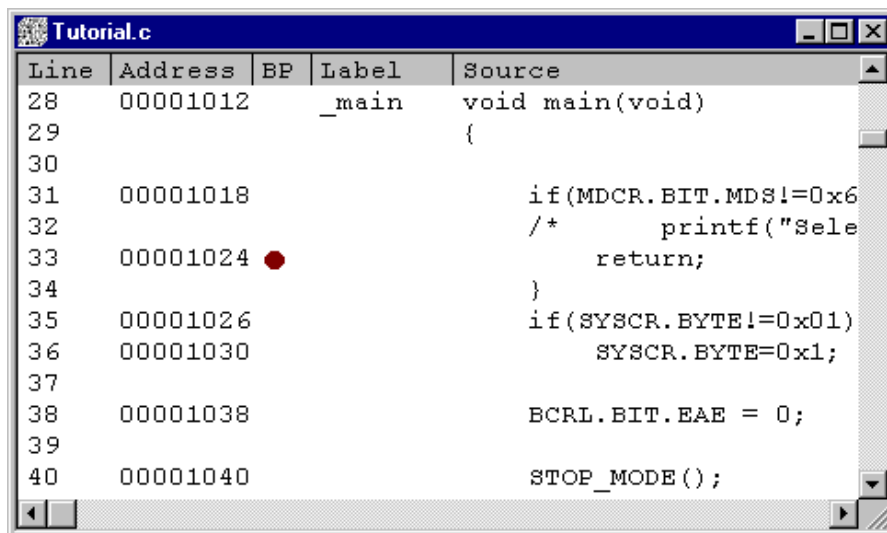



図8-1 Source ウィンドウ

Source ウィンドウは、ヘッダー領域とメインウィンドウ領域の、2つの領域に分割されています。また、縦に、Line、Address、BP (Break Point)、Label および Source の5つの列に分かれています。各列の幅を調整するには、ヘッダーに表示された各列のタイトル間の境界線をドラッグします。カーソルが+⇄に変わり、ドラッグした位置にあわせて、メインウィンドウ領域に縦線が表示されます。変更したい列幅に合わせてマウスボタンを離すと、新しい列幅で表示が更新されます。

### 8.2.2 アセンブラコードの表示

Source ウィンドウがオープンされている場合は、ポップアップメニューの[Go to Disassembly]メニューオプションを選択すると、Disassembly ウィンドウをオープンします。Disassembly ウィンドウには、Source ウィンドウに現在表示されているアドレスと同一のアドレスを表示します。

ソースファイルがなく、アセンブラレベルでコードを表示したいという場合は、[View->Disassembly...]メニューオプションを選択するか、キーボードでCtrl+Dを入力します。あるいは、ツールバー上に Disassembly ボタンが表示されていれば、このボタンをクリックします。Set Address ダイアログボックスには、逆アセンブルの開始アドレスを指定してください。

Disassembly ウィンドウにはアドレス、ブレークポイント、マシンコード値、ラベル、逆アセンブルされたニーモニック(使用可能であればラベル付きで)を表示します。さらに、対応するソースファイルを表示し、混在モード表示を提供します。

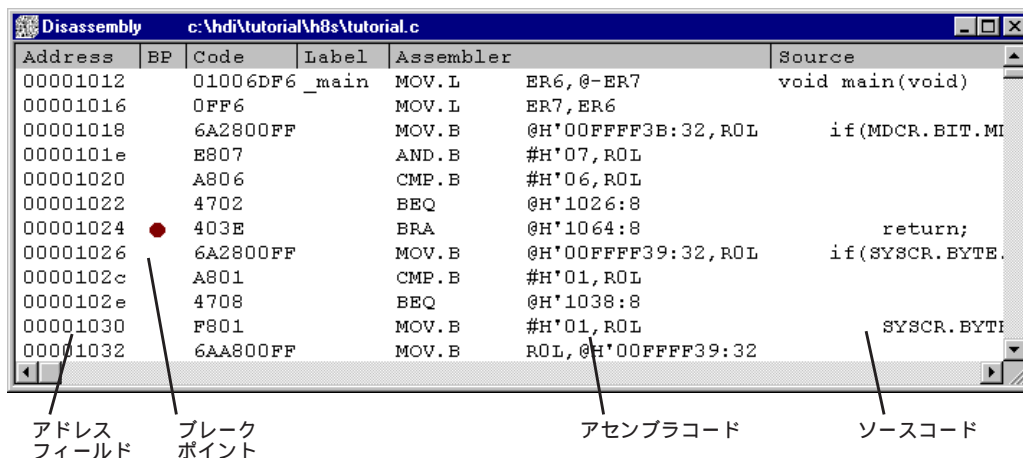


図8-2 Disassembly ウィンドウ

### 8.2.3 アセンブラコードの変更

アセンブラコードを変更するには、変更したい命令をダブルクリックし、Assembler ダイアログボックスを開きます。

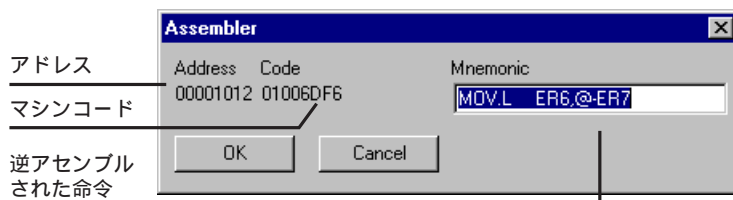


図8-3 Assembler ダイアログボックス

アドレス、マシンコード、および逆アセンブルされた命令を表示します。Mnemonic フィールドに新しい命令を入力するか、表示されている命令を変更してください。ENTER キーを押すと、その命令がアセンブルされてメモリに格納され、次の命令に移ります。[OK]をクリックすると、その命令がアセンブルされてメモリに格納され、このダイアログボックスが閉じます。[Cancel]をクリックするか、ESC キーを押すと、このダイアログボックスが閉じます。

【注】 アセンブラ表示は、デバッグプラットフォームのメモリに格納された実際のマシンコードから逆アセンブルします。メモリの内容が変更されると、それに対応する新しいアセンブラコードを示します。ただし、この内容は、ソース表示とは一致しません。

### 8.3 ラベルの表示

デバッグオブジェクトファイルに含まれるシンボル情報には、プログラム中でアドレスを表すラベルが含まれています。ラベルは対応するアドレスの Label フィールドに表示します。また、Assembler フィールドにオペランドの一部として表示します。

- 【注】
1. 命令のオペランドとラベルが一致する場合は、そのオペランドをラベルに置き換えます。2つ以上のラベルが同じ値をもつ場合は、アルファベット順で最初にくるラベルを表示します。
  2. HDI では、アドレスまたは値を入力できる場所では、どこでもラベルを代わりに使用できます。

#### 8.3.1 ラベルのリスト表示

HDI で定義した全ラベルのリストを見るには、[View->Labels]メニューオプションを選択して、Labels ウィンドウを開きます。

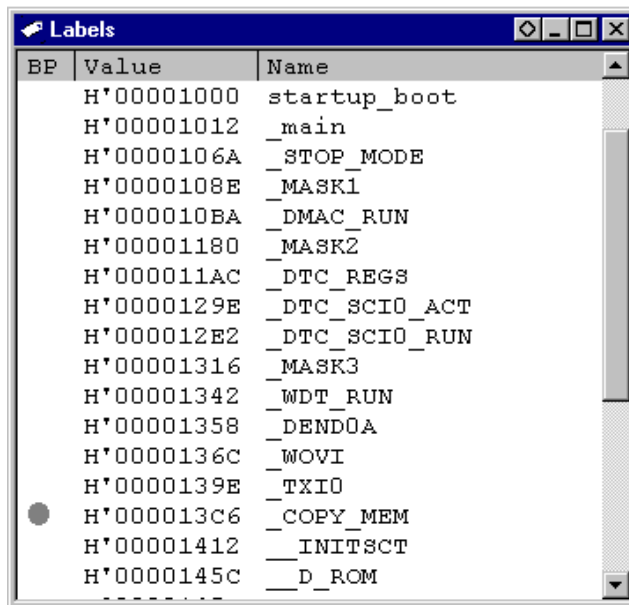


図8-4 Labels ウィンドウ

それぞれのカラムヘッダをクリックすることによって、アルファベット順(ASCII コード順)またはアドレスの昇順にソートします。

BP 列をダブルクリックする(または、ポップアップメニューの [Break]メニューオプションを選択する)により、対応するアドレスにソフトウェアブレークを迅速に設定することができます。

### 8.3.2 Source ウィンドウまたは Disassembly ウィンドウでのラベルの追加

Source ウィンドウまたは Disassembly ウィンドウで、ラベルを割り当てたいアドレスのラベル列をダブルクリックすることで、ラベルを迅速に追加することができます。Label ダイアログボックスで、ラベル名を入力してください。

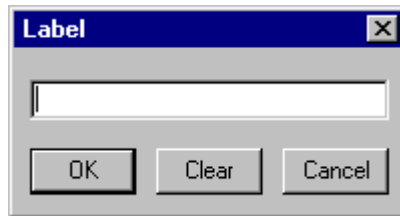


図8-5 Label ダイアログボックス

ラベル名を入力し[OK]をクリックすると、その行のアドレス列に表示されているアドレス値がラベルに設定され、ラベルリストに追加されます。Source ウィンドウの表示内容が更新され、追加したラベルが表示されます。[Clear]ボタンはラベルを削除するために使用します。

また、この方法を使用して、既存のラベルのテキストを簡単に変更することもできます。Label 列で変更するラベルをダブルクリックすると、Label ダイアログボックスのエディットボックスにそのラベルのテキストがコピーされます。次に、そのテキストを変更すると、変更されたテキストがラベルリストに保存されます。Source ウィンドウの表示内容が更新され、新しいラベルが表示されます。

【注】 追加・変更したラベルを再度使用する場合は、ラベル情報をファイルに保存してください。詳細は、「5.6.11 Save As ...」を参照してください。

## 8.4 特定アドレスのプログラム表示

Source ウィンドウでプログラムを参照時に、別の領域のプログラムを参照したいことがあります。その場合には、メインウィンドウ領域をスクロールするのではなく、特定のアドレスへ直接移動することができます。Address 列でダブルクリックすると、Set Address ダイアログボックスが開きます。

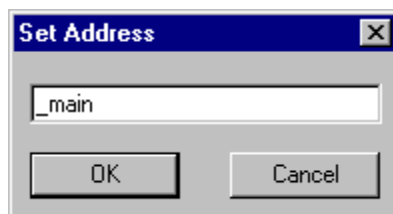


図8-6 Set Address ダイアログボックス

該当するアドレスまたはシンボル名をエディットボックスに入力して、[OK]をクリックするか、ENTER キーを押します。そのアドレスのコードが同じソースファイル内にある場合は、Source ウィンドウが新しいアドレスのコードに更新されます。ただし、多重定義関数あるいはクラス名を入力した場合、Select Function ダイアログボックスが開くので、設定する関数を選択してください。詳細

## 8. プログラムの表示

---

については、「14 関数の設定」を参照してください。

新しいアドレスが別のソースファイル内にある場合は、新しい Source ウィンドウが開き、そのアドレスのコードが表示されます。ソースファイルが参照可能であれば、デフォルトで新規ウィンドウにソースが表示されます。新規アドレスに対して参照可能なソースファイルがない場合は、Source ウィンドウはアセンブラコードで表示されます。

新しいアドレスが、すでに開いている Source ウィンドウのソースファイル内にある場合は、そのウィンドウが前面に呼び出されて、新しいアドレスのコードが表示されます。

### 8.4.1 現在の PC のプログラム表示

HDI にアドレスまたは値を入力できる場所では、式を入力することも可能です(「2.2 データ入力」を参照)。先頭に“#”文字を付けてレジスタ名を入力すると、そのレジスタの内容が値として式に使用できます。つまり、Set Address ダイアログボックスを呼び出して、“#PC”という式を入力すると、Source ウィンドウまたは Disassembly ウィンドウには現在の PC のアドレスが表示されます。また、PC+オフセット (例：“#PC+0x100”)の形で式を入力して、現在の PC アドレスからオフセット値分はなれた箇所を表示することも可能です。

## 8.5 テキストの検索

検索オプションを使用して、Source ウィンドウから特定の文字列を検索できます。これを実行するには、ポップアップメニューの[Find...]メニューオプションを選択するか、キーボードで F3 を押します。

Find ダイアログボックスを表示します。

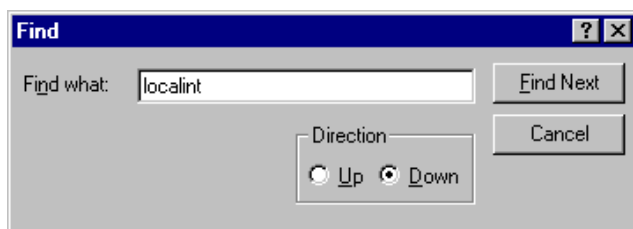


図8-7 Find ダイアログボックス

検索したいテキストを入力して、[Find Next]をクリックするか、ENTER キーを押します。テキストが検出されると、Source ウィンドウにはそのテキストが反転表示されます。そのテキストが次に現れる箇所を検索するには、再度、[Find Next]をクリックするか、ENTER キーを押します。Find ダイアログボックスを閉じるには、[Cancel]をクリックするか、ESC キーを押します。


---

## 9. メモリの操作

---

本章では、CPU のアドレス空間で、メモリの各領域を見る方法について説明します。ここでは、様々なフォーマットでメモリ領域を表示する方法、メモリブロックにデータを埋める方法、メモリブロックの移動およびテスト方法、ディスクファイルを使用したメモリ領域の保存、ロード、およびペリファイ方法について述べます。

### 9.1 メモリ領域の表示

メモリ領域を見るには、[View->Memory...]メニューオプションを選択するか Ctrl+M キーを押して、Memory ウィンドウを開きます。あるいは、ツールバーに Memory ボタンが表示されていれば、このボタンをクリックして Memory ウィンドウを表示することもできます。これらの操作により Open Memory Window ダイアログボックスが開きます。

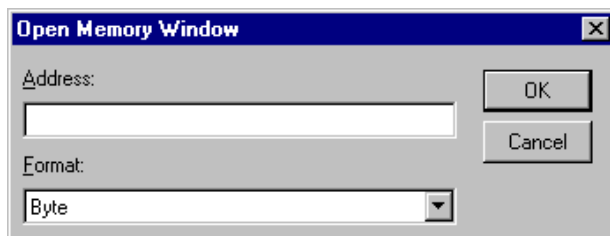
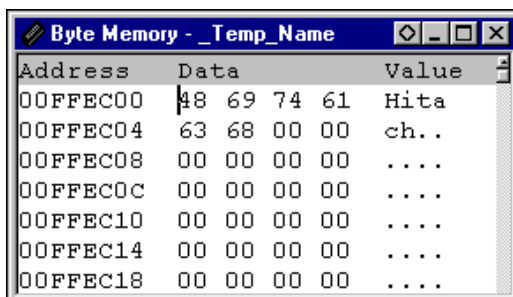


図9-1 Open Memory Window ダイアログボックス

Address フィールドに、ウィンドウ表示の開始アドレスまたはそれに相当するシンボルを入力して、Format リストから必要な表示フォーマットを選択します。[OK]をクリックするか、ENTER キーを押すと、このダイアログボックスが閉じ、Memory ウィンドウが開きます。

A screenshot of the 'Byte Memory - \_Temp\_Name' window. The window title is 'Byte Memory - \_Temp\_Name'. It displays a table with three columns: 'Address', 'Data', and 'Value'. The data is as follows:

Address	Data	Value
00FFEC00	48 69 74 61	Hita
00FFEC04	63 68 00 00	ch..
00FFEC08	00 00 00 00	....
00FFEC0C	00 00 00 00	....
00FFEC10	00 00 00 00	....
00FFEC14	00 00 00 00	....
00FFEC18	00 00 00 00	....

図9-2 Memory ウィンドウ(バイト単位)

Memory ウィンドウは、アドレス列を除いて2つの表示列を持っています。

- Data  
表示するデータはデバッグプラットフォームから読み込みます。データは表示サイズ単位に物理メモリから読み込みます。データの変更ができます。
- Value  
データは選択されたフォーマットで表示します。値の変更はできません。

表示フォーマットを、ウィンドウのオープン時に選択したもから変更したい場合は、ポップアップメニューから変更できます。

### 9.1.1 ASCII でのメモリ表示

メモリを ASCII 文字として表示・変更するにはポップアップメニューの[ASCII]メニューオプションを選択します。[ASCII]メニューオプションを選択すると、表示内容が更新され、メモリ内容が ASCII 文字で表示されます。

### 9.1.2 バイト単位でのメモリ表示

メモリをバイト単位で表示・変更するにはポップアップメニューの[Byte]メニューオプションを選択します。[Byte]メニューオプションを選択すると、表示内容が更新され、メモリ内容がバイト単位で表示されます。(図 9-2)

### 9.1.3 ワード単位でのメモリ表示

メモリをワード単位で表示・変更するにはポップアップメニューの[Word]メニューオプションを選択します。[Word]メニューオプションを選択すると、表示内容が更新され、メモリ内容が 16 ビットのワード単位で表示されます。

### 9.1.4 ロングワード単位でのメモリ表示

メモリをロングワード単位で表示するにはポップアップメニューの [Long]メニューオプションを選択します。[Long]メニューオプションを選択すると、表示内容が更新され、メモリ内容が 32 ビットのロングワード単位で表示されます。

### 9.1.5 単精度浮動小数点数形式でのメモリ表示

メモリを単精度浮動小数点数形式で表示するにはポップアップメニューの [Single float]メニューオプションを選択します。[Single float]メニューオプションを選択すると、表示内容が更新され、メモリ内容が単精度浮動小数点数形式で表示されます。

### 9.1.6 倍精度浮動小数点数形式でのメモリ表示

メモリを倍精度浮動小数点数形式で表示するにはポップアップメニューの[Double float]メニューオプションを選択します。[Double float]メニューオプションを選択すると、表示内容が更新され、メモリ内容が倍精度浮動小数点数形式で表示されます。

### 9.1.7 別領域のメモリ表示

Memory ウィンドウに表示されたメモリ領域を変更するには、スクロールバーを使用できます。表示する領域を変更するには、Set Address ダイアログボックスを使用します。ポップアップメニューの[Set Address...]メニューオプションを選択するか、またはアドレス列をダブルクリックすることで、このダイアログボックスを呼び出すことができます。

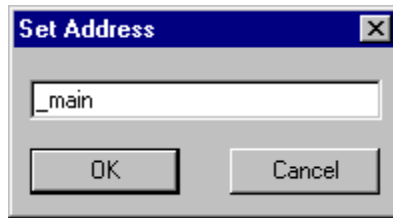


図9-3 Set Address ダイアログボックス

Set Address ダイアログボックスが開いたら、新しいアドレス値を入力します。[OK]をクリックするか、ENTER キーを押すと、このダイアログボックスが閉じ、Memory ウィンドウの表示内容が更新され、新しいアドレスのデータが表示されます。ただし、アドレス値として多重定義関数あるいはクラス名を入力した場合、Select Function ダイアログボックスが開くので、設定する関数を選択してください。詳細については、「14 関数の設定」を参照してください。

## 9.2 メモリ内容の変更

アドレスのメモリ内容を変更する方法は2つあります。一つは簡易変更方式です。本方式は、ウィンドウに値を直接入力できますが、ASCII 表示時は ASCII 入力のみ、それ以外での表示時は 16 進数値の入力のみで限定されます。もう一つは、詳細変更方式です。本方式は、ダイアログボックスで値を入力します。浮動小数点数や式を値として入力できます。

### 9.2.1 簡易変更方式

該当する桁をクリックするか、または、クリック・ドラッグにより変更したい桁を選択し、反転表示させます。その桁に新しい値を入力してください。ASCII 表示時以外のとき、値は、0-9 および a-f の範囲内であればなりません。ASCII 表示時のときには、ASCII 入力であればなりません。変更値がその桁に書き込まれ、カーソルがメモリの次の桁へ移動します。

### 9.2.2 詳細変更方式

Edit ダイアログボックスを使用します。変更するメモリ単位 (Memory ウィンドウの表示フォーマットに依存します) 上にカーソルを移動して ENTER キーを押す、またはダブルクリックすると、Edit ダイアログボックスが開きます。



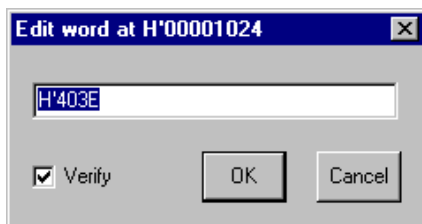


図9-4 Edit ダイアログボックス

データ入力フィールドには、フォーマットに従った数値またはC/C++言語の式を入力できます(「2.2 データ入力」を参照)。新しい数値または式を入力したら、[OK]ボタンをクリックするか、ENTER キーを押します。ダイアログボックスが閉じ、新しい値がメモリに書き込まれます。

### 9.2.3 メモリ範囲の選択

選択する範囲が Memory ウィンドウに表示されている範囲内の場合、範囲の先頭位置でマウスの左ボタンを押して、そのまま範囲の終了位置までドラッグします。選択された範囲は強調表示されます。

## 9.3 メモリ内の値の検索

Memory ウィンドウの検索機能を使用して、メモリ内の値を検索できます。値を検索するには、ポップアップメニューの[Search]メニューオプションをクリックするか、または Memory ウィンドウが入力フォーカスを持つときに F3 を押します。これらの操作により Search Memory ダイアログボックスを開きます。

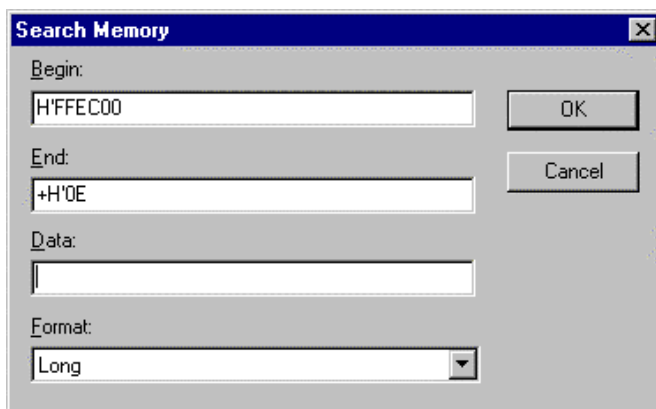


図9-5 Search Memory ダイアログボックス

検索したい範囲の開始アドレスと終了アドレス、および検索するデータ値を入力します。開始アドレスと終了アドレスは Memory ウィンドウで範囲が選択されている場合は、自動的に設定されます。また、終了アドレスには、符号 + を値の前に付けて、サイズとして指定することができます。

検索フォーマットを選択して、[OK]をクリックするか、ENTER キーを押します。ダイアログボックスが閉じ、HDI は該当する範囲から指定したデータを検索します。データが検出されると、その

データが Memory ウィンドウに強調表示されます。データが検出されなかった場合は、Memory ウィンドウの表示は変更されず、データが検出されなかったことを示すメッセージがメッセージボックスに表示されます。

## 9.4 メモリ領域に値を埋める

メモリ埋め込み機能を使用して、指定した範囲のメモリアドレスに値を設定できます。

### 9.4.1 範囲を埋める

同じ値で指定した範囲のメモリを埋めるためには、[Memory->Fill...]メニューオプションまたは Memory ウィンドウのポップアップメニューの[Fill...]メニューオプションを選択し、Fill Memory ダイアログボックスを開きます。

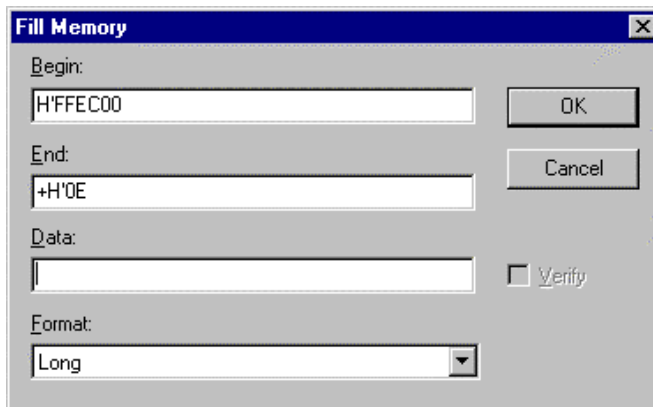


図9-6 Fill Memory ダイアログボックス

Memory ウィンドウでアドレス範囲を選択してある場合、先頭アドレスおよび終了アドレスは、自動的に設定されます。検索フォーマットを選択して、[OK]をクリックするか、ENTER キーを押します。ダイアログボックスが閉じ、新規の値がメモリ範囲に書き込まれます。

## 9.5 メモリ領域の転送

メモリ転送機能を使用して、メモリ領域を転送できます。メモリ範囲を選択して(「9.2.3 メモリ範囲の選択」を参照)、ポップアップメニューの[Copy]メニューオプションを選択し、Copy Memory ダイアログボックスを開きます。

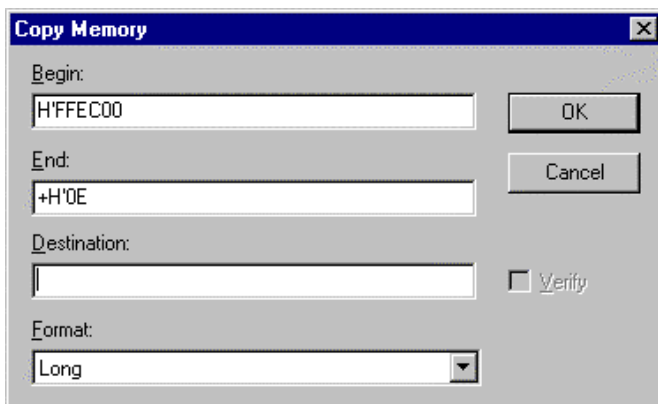


図9-7 Copy Memory ダイアログボックス

Begin および End フィールドには、Memory ウィンドウで選択した転送元のそれぞれのアドレスが設定されています。Destination フィールドに転送先の開始アドレスを入力して、[OK]ボタンをクリックするか、ENTER キーを押すと、このダイアログボックスが閉じ、指定されたメモリブロックが新しいアドレスへコピーされます。

## 9.6 メモリ領域の保存

メモリ保存機能を使用して、アドレス空間内のメモリ領域をディスクファイルに保存できます。[Memory->Save...]メニューオプションを選択して、Save Memory As ダイアログボックスを開きます。

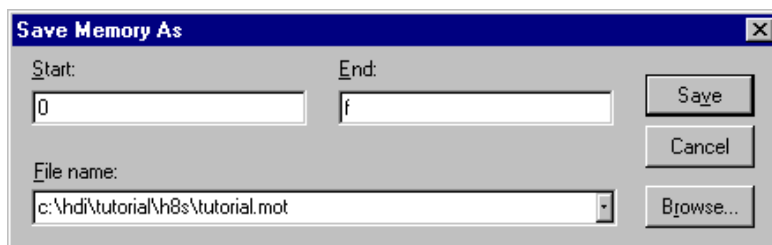


図9-8 Save Memory As ダイアログボックス

保存したいメモリブロックの開始アドレスと終了アドレス、およびファイル名を入力します。ファイル名のドロップダウンリストは、以前にセーブしたファイル名を4つ持っています。また、[Browse...]ボタンを押すと、標準の Save As ダイアログボックスが開きます。[Save]ボタンをクリックするか、ENTER キーを押すと、このダイアログボックスが閉じ、指定したメモリブロックが S-Record フォーマットファイルとしてディスクに保存されます。ファイルの保存が完了すると、確認のメッセージボックスが開きます。HDI Options ダイアログボックスの Confirmation タグで確認メッセージの出力抑止できます。

## 9.7 メモリ領域のロード

メモリ領域のロード機能を使用して、現在のデバッグ情報を削除せずにメモリ領域へ S-Record ファイルをロードすることができます。[Memory->Load...]メニューオプションを選択すると Load Memory ダイアログボックスが開きます。

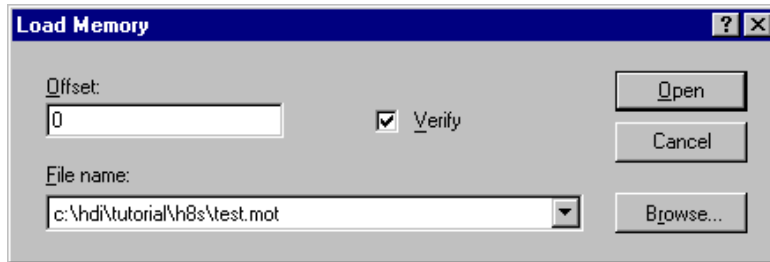


図9-9 Load Memory ダイアログボックス

Offset フィールドに値(正または負)を入力して、S-Record で指定されたアドレスから指定した値の分はなれた位置をロードアドレスとすることができます。[Open]ボタンをクリックするか、ENTER キーを押すと、このダイアログボックスが閉じ、データがメモリにロードされます。データのロードが完了すると、確認のメッセージボックスが開きます。HDI Options ダイアログボックスの Confirmation タグで確認メッセージの出力抑止できます。

## 9.8 メモリ領域のベリファイ

メモリ比較機能を使用して、現在のメモリブロックと以前に保存したメモリブロックを比較することができます。[Memory->Verify...]メニューオプションを選択すると Verify S-Record File with Memory ダイアログボックスが開きます。

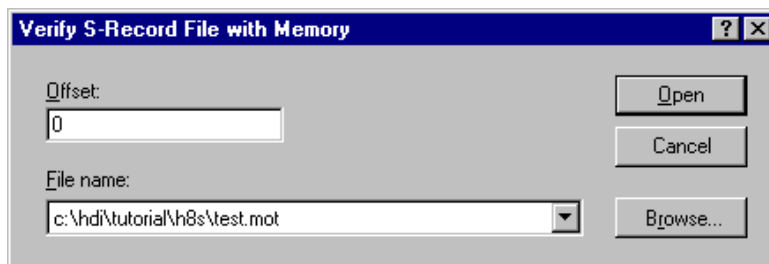



図9-10 Verify S-Record File with Memory ダイアログボックス


Offset フィールドに値(正または負)を入力して、S-Record で指定されたアドレスから指定した値の分はなれた位置からベリファイを開始することができます。[Open]ボタンをクリックするか、ENTER キーを押すと、このダイアログボックスが閉じ、ファイルとメモリの内容がベリファイされます。ベリファイが完了すると、確認のメッセージボックスが開きます。HDI Options ダイアログボックスの Confirmation タグで確認メッセージの出力抑止できます。

## 10. プログラムの実行

本章では、ユーザプログラムの実行方法について説明します。ここでは、命令を連続実行する方法と、単一命令または複数命令をステップ実行する方法について述べます。

### 10.1 リセットからの実行

ユーザシステムをリセットして、リセットベクタアドレスからアプリケーションを実行するには、ツールバーの Reset Go ボタンをクリックするか、[Run->Reset Go]メニューオプションを選択します。

プログラムは、ブレークポイントに到達するかブレーク条件が成立するまで実行されます。ツールバーの Halt ボタンをクリックするか、[Run->Halt]メニューオプションを選択して、プログラムを任意のタイミングで停止できます。

【注】 プログラムは、リセットベクタ位置に格納されたアドレスから実行を開始します。したがって、この位置に開始コードのアドレスが格納されているかどうかを確認してください。

### 10.2 連続実行

プログラムが停止し、デバッガがブレークモードになると、HDI は、Source ウィンドウまたは Disassembly ウィンドウ内で、CPU のプログラムカウンタ(PC)が示す現在のアドレス値に対応する行を強調表示します。これは、ステップ実行または連続実行の場合に、次に実行される命令となります。

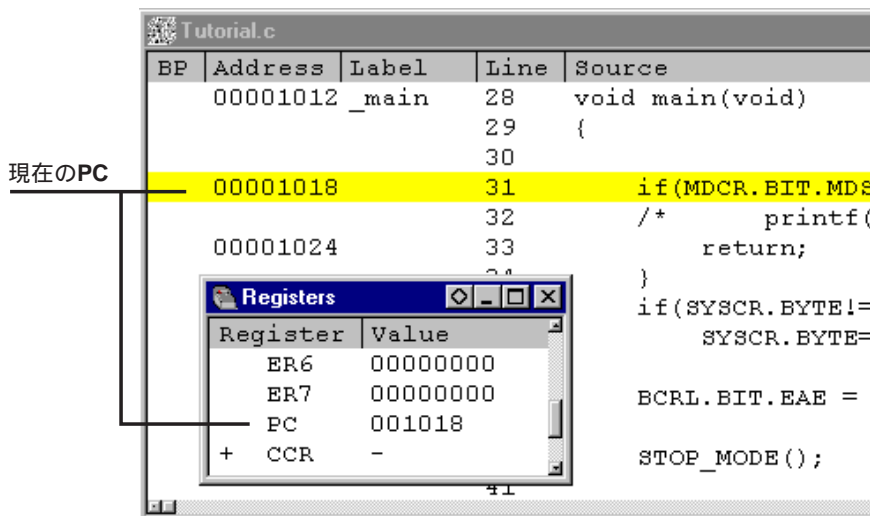



図10-1 PC が示すアドレス値に対応する行の強調表示

## 10. プログラムの実行

---

現在の PC のアドレスから連続実行するには、Go ボタンをクリックするか、[Run->Go]メニューオプションを選択します。


### 10.3 カーソル位置まで実行

プログラム的一部分のみを実行する機能として、特定アドレスまで実行する Go To Cursor 機能があります。

#### ●Go To Cursor の使用手順

1. プログラムを停止させたいアドレスが、SourceウィンドウまたはDisassemblyウィンドウに表示されていることを確認してください。
2. Address列でクリックするか、カーソルキーを使用して、停止させたいアドレス上にテキストカーソルを合わせます。
3. ポップアップメニューの [Go To Cursor]メニューオプションを選択します。

これにより、現在の PC 値から、カーソル位置で指定したアドレスに到達するまでプログラムが実行されます。

- 【注】 1. カーソル位置のアドレスのプログラムを実行しない場合は、プログラムは停止しません。このようなことが起きた場合、ツールバーの Halt ボタンをクリックするか、ESC キーを押す、または、[Run->Halt]メニューオプションを選択して、プログラムを手操作で停止できます。
2. Go To Cursor 機能は、テンポラリブレークポイントを使用します。ブレークポイント設定数が上限値の場合には、この機能は使用できません。メニューオプションが無効状態になります。


### 10.4 複数ポイントまで実行

Go To Cursor のような処理を実行したいが、停止アドレスが Source ウィンドウの外部にある、または複数のアドレスで停止させたいという場合もあります。このような場合には、HDI のテンポラリブレークポイント機能を使用します(「11.5 テンポラリブレークポイント」を参照)。


### 10.5 シングルステップ

プログラムをデバッグ時に、1 行単位または 1 命令単位に実行して、システムに対するその命令の効果を検証できる機能です。Source ウィンドウでは、ステップ処理はソースの 1 行をステップ実行します。Disassembly ウィンドウでは、ステップ処理はアセンブラの 1 命令をステップ実行します。命令が別の関数またはサブルーチンを呼び出す場合は、その関数内の各命令をステップ実行するか、またはその関数全体を 1 ステップで実行するかをオプションで指定します。命令が呼び出しを実行しない場合は、いずれのオプションを指定していても、デバッガはその命令を実行して、次の命令で停止します。

### 10.5.1 関数内の各命令をステップ実行


関数内の各命令のステップ実行を選択すると、関数の呼び出しを実行し、その関数の最初の行または命令で停止します。関数内の各命令をステップ実行するには、Step In ボタンをクリックするか、[Run->Step In]メニューオプションを選択します。

### 10.5.2 関数全体を1ステップで実行

関数全体の1ステップ実行を選択すると、デバッガは呼び出しとその関数（および、その関数が実行するすべての関数呼び出し）を実行し、呼び出し元関数の次の行または命令で停止します。関数全体を1ステップで実行するには、Step Over ボタンをクリックするか、[Run->Step Over]メニューオプションを選択します。

## 10.6 関数の実行停止

デバッグ時に、関数の実行を開始し、検証したい各命令の実行を完了した後、その関数の残りの全コードを実行して呼び出し元関数に戻りたい場合があります。あるいは、関数全体の1ステップ実行をするつもりが、誤って関数内の各命令のステップ実行したために、現在の関数の全コードを実行して呼び出し元関数に戻りたい場合があります(この場合の方に、より有用です)。このような場合には、Step Out 機能を使用します。

現在の関数の残コードを実行して呼び出し元関数に戻るには、Step Out ボタンをクリックするか、[Run->Step Out]メニューオプションを選択します。

## 10.7 複数のステップ

1度に複数の命令を実行してから停止させるには、Step Program ダイアログボックスを使用します。また、このダイアログボックスには、ステップ実行の間隔を選択できる自動ステップ機能も備わっています。このダイアログボックスを呼び出すには、[Run-> Step...]メニューオプションを選択します。

Step Program ダイアログボックスを表示します。



図10-2 Step Program ダイアログボックス



## 10. プログラムの実行

---

Steps フィールドにはステップ数を入力します。Step Over Calls チェックボックスにより、関数呼び出し全体を 1 ステップで実行するかどうかを選択します。Source Level Step チェックボックスにより、ソースプログラムの 1 行と 1 ステップを対応させるかどうかを選択します。自動ステップ機能を使用する場合は、Rate フィールドのリストからステップの速度を選択します。ステップ実行を開始するには、[OK]をクリックするか、ENTER キーを押します。



---

## 11. プログラムの停止

---

本章では、プログラムの実行を停止する方法を説明します。停止コマンドを使用して直接停止させる方法と、コード中の特定の位置にブレークポイントを設定し、停止させる方法について述べます。

### 11.1 実行の停止

ツールバーの Halt ボタンは、プログラムの実行中は有効状態  (赤色の STOP マーク)、プログラム停止時は無効状態  (グレー表示の STOP マーク) となります。実行中に、Halt ボタンをクリックするか、ESC キーを押すか、メニューで [Run->Halt] を選択することにより実行が停止します。

プログラムの実行が停止され、メッセージ “ Break = Stop ” がステータスバーに表示されます、また、開いているすべてのウィンドウが HDI によって更新されます。

最後にブレークしたときの要因は、System Status ウィンドウの Platform シートに表示されます。

### 11.2 PC ブレークポイント

プログラムのデバッグで、PC が 1 個所または複数の特定個所に達した時点で実行を停止させたい場合には、その行または命令にプログラムブレークポイントを設定してプログラムを停止させることができます。以下に、プログラムブレークポイントを簡単に設定・削除する方法を示します。Breakpoints ウィンドウで、より複雑なブレークポイント操作を行うこともできます。これについては後で説明します。

#### 🔍 プログラムブレークポイントの設定手順

1. プログラムブレークポイントを設定したい位置の Source ウィンドウが開いていることを確認します。
2. プログラムを停止したいアドレスが表示されている行の BP 列をダブルクリックするか F9 キーを押します。
3. プログラムブレークポイントが設定されたことを示す “Break” という語と丸が、その列に表示されます。

## 11. プログラムの停止

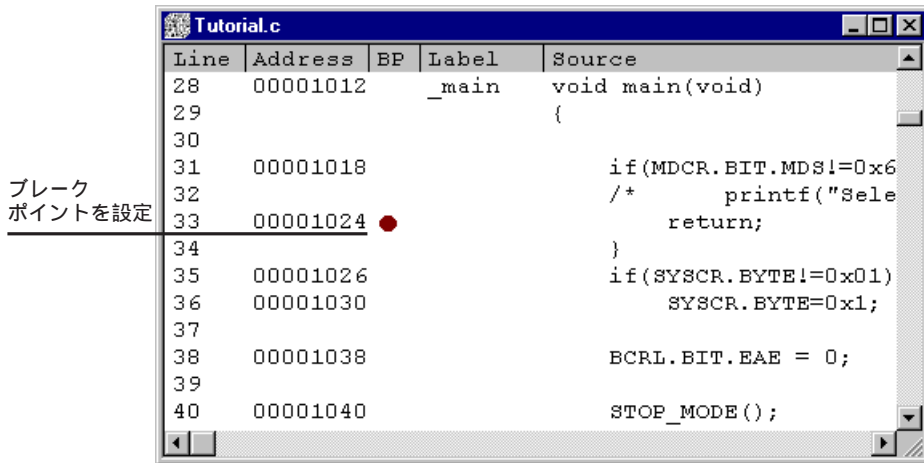



図11-1 プログラムブレークポイントの設定

これにより、プログラムがプログラムブレークポイントを設定したアドレスに到達すると、実行が停止して、メッセージ“ Break = PC Breakpoint ” がステータスバーに表示されます。また、 Source ウィンドウの表示が更新されてプログラムブレークポイント行が強調表示されます。

【注】 プログラムがプログラムブレークポイントを設定したアドレスで停止した場合、プログラムブレークポイントを設定した行または命令は、実行されません。プログラムはその行または命令の実行が開始される直前に停止します。プログラムブレークポイントで停止した後、Go または Step を選択すると、次に実行される命令は強調表示された行となります。

### 11.3 Breakpoints ウィンドウ

Breakpoints ウィンドウを使用すると、複雑なブレークポイントを設定することができ(デバッグプラットフォームが各ブレークポイントをサポートしている場合)、ブレークポイントの設定/削除および有効/無効に関して、さらに詳細に制御できるようになります。Breakpoints ウィンドウを開くには、[View->Breakpoints]メニューオプションを選択します。あるいは、ツールバーに Breakpoints ボタンが表示されていれば、このボタンをクリックして Breakpoints ウィンドウを開くこともできます。

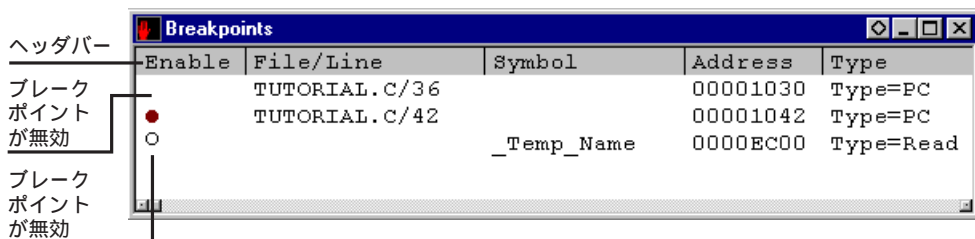


図11-2 Breakpoints ウィンドウ

このウィンドウは、システムに設定されたブレークポイントのリストを表示します。ブレークポイントリストは、横方向に Enable、File/Line、Symbol、Address、および Type の 5 列に分かれています。各列の幅を調整するには、ヘッダバーに表示された各列のタイトル間の境界線をドラッグしま

す。カーソルが $\leftarrow$ に変わり、ドラッグした位置にあわせて、リスト領域に縦線が表示されます。マウスボタンを離れた位置の列幅で、表示が新しい列幅に更新されます。

### 11.3.1 ブレークポイントの追加

Breakpoints ウィンドウに新しいブレークポイントを追加するには、ポップアップメニューの[Add...]メニューオプションを選択してください。

Set Break ダイアログボックスが開き、ここで設定したいブレークポイントの種類とパラメータを選択できます。

### 11.3.2 ブレークポイントの変更

Breakpoints ウィンドウで既存のブレークポイントを変更するには、リスト上の該当するブレークポイントに対応する行をダブルクリックしてください。または、クリックしてそのブレークポイントを選択後、ポップアップメニューの[Edit...]メニューオプションを選択してください。

Set Break ダイアログボックスが開き、設定を変更したいブレークポイントの種類とパラメータを選択できます。ただし、ブレークシーケンスを選択した場合は、Break Sequence ダイアログボックスが開きます。

### 11.3.3 ブレークポイントの削除

Breakpoints ウィンドウから既存のブレークポイントを削除するには、リスト上の該当するブレークポイントに対応する行をクリックしてブレークポイントを選択します。次に、ポップアップメニューの[Delete]メニューオプションを選択してください。

選択したブレークポイントが削除され、ウィンドウが更新されます。

### 11.3.4 全ブレークポイントの削除

Breakpoints ウィンドウにリストされたブレークポイントをすべて削除するには、ポップアップメニューの[Delete All]メニューオプションを選択してください。

すべてのブレークポイントが削除され、ウィンドウが更新されます。

## 11.4 ブレークポイントを無効にする

ある領域のプログラムをデバッグ後、別の領域のプログラムをデバッグして、また前の領域に戻りたいという場合に、ある領域にブレークポイントを設定してデバッグし、別の領域をデバッグするために設定したすべてのブレークポイントを削除し、別の領域のデバッグ後、再び削除したブレークポイントを設定しなければならぬのでは、効率が悪くなります。HDI では、ブレークポイントリストにブレークポイントを残したまま、それらを無効にすることが可能です。

### 11.4.1 ブレークポイントを無効にする

各ブレークポイントを無効にするには、リスト上の該当するブレークポイントに対応する行をクリックしてそのブレークポイントを選択します。次に、ポップアップメニューの[Disable]メニューオプションを選択してください。

## 11. プログラムの停止

Enable 列をダブルクリックしても、ブレークポイントが無効になります。

メニューが閉じ、ブレークポイントリストが更新され、Enable 列のチェックマークが消去され、ブレークポイントが無効となります。

### 11.4.2 ブレークポイントを有効にする

Breakpoints ウィンドウでブレークポイントを再度有効にするには、上述のように、リスト上の該当するブレークポイントに対応する行をクリックしてそのブレークポイントを選択します。次に、ポップアップメニューの[Enable]メニューオプションを選択してください。

Enable 列をダブルクリックしても、ブレークポイントが有効になります。

メニューが閉じ、ブレークポイントリストが更新され、Enable 列にチェックマークが付き、選択したブレークポイントが有効となります。

## 11.5 テンポラリブレークポイント

プログラムの実行を開始して、1つまたは複数のアドレスに達した場合にプログラムを停止したいが、それらのアドレスに恒久的なブレークポイントを設定したくないという場合があります。たとえば、Go To Cursor と同じような処理を実行したいが、停止アドレスが Source ウィンドウの外部にある、または複数のアドレスで停止させたいという場合などが当てはまります。このような場合には、HDI のテンポラリブレークポイント機能を使用して、最大 10 箇所のテンポラリブレークポイントを設定して実行します。それらのブレークポイントは、プログラムが停止すると削除されます。テンポラリブレークポイントは、Run Program ダイアログボックスで設定します。このダイアログボックスを開くには、[Run-> Run...]メニューオプションを選択します。

Run Program ダイアログボックスが開きます。

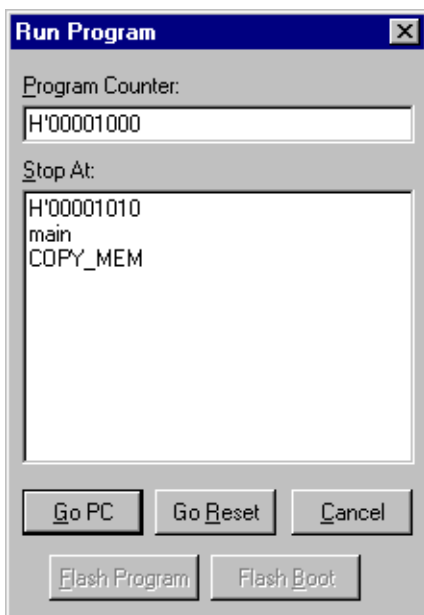


図11-3 Run Program ダイアログボックス

Stop At フィールドに、プログラムを停止する各ポイント(最大 10 まで)を表すシンボルまたはアドレスを入力します。ただし、多重定義関数あるいはクラス名を入力した場合、Select Function ダイアログボックスが開くので、設定する関数を選択してください。詳細については、「14 関数の設定」を参照してください。

Program Counter フィールドに表示されている、現在の PC アドレスから実行を開始するには、[Go PC]ボタンをクリックします。CPU をリセットして、リセットベクタアドレスから実行を開始するには、[Go Rset]ボタンをクリックします。

テンポラリブレークポイントでプログラムが停止すると、テンポラリブレークポイントがブレークポイントリストから削除されます。Run Program ダイアログボックスを再度選択すると、Stop At フィールドにテンポラリブレークポイントがリストされ、[Go PC]ボタンまたは[Go Rset]ボタンをクリックすると、再度設定されます。



---

## 12. 変数の表示

---

本章では、プログラムが使用する変数およびデータオブジェクトを見る方法について説明します。ここでは、変数の表示方法、ウォッチ項目の設定方法、および CPU の汎用レジスタ、FPU レジスタ、DSP レジスタおよび内蔵周辺レジスタを見る方法について述べます。

### 12.1 ツールチップウォッチ

Tooltip Watch 機能を使用してプログラム内の変数を迅速に見ることができます。

#### ☛ Tooltip Watch の使用手順

1. Source ウィンドウを開き、ウォッチしたい変数が表示された状態にします。
2. ウォッチしたい変数の変数名の上にマウスカーソルを移動して静止します。

```
✓_MEM 221 void COPY_MEM(void)
      222 {
      223     unsigned short u;
      224     for( u=0; u < sizeof(NAME); u++ )
      225         *(Temp2_Name+u) = *(NAME+u);
      226
      227 }
```

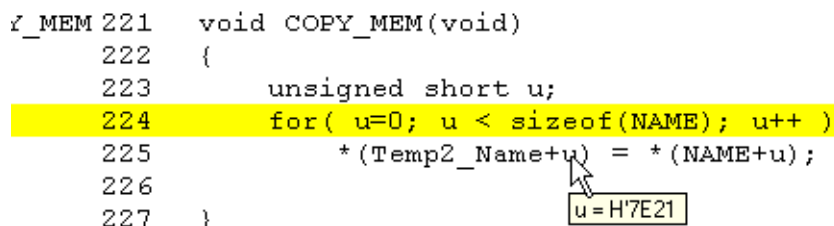


図12-1 Tooltip Watch

### 12.2 インスタントウォッチ

Instant Watch 機能を使用して変数の詳細を見ることができます。

#### ☛ Instant Watch の使用手順

1. Source ウィンドウを開き、ウォッチしたい変数が表示された状態にします。
2. 該当する変数をクリックして、変数上にカーソルを移動します。
3. ポップアップメニューの[Instant Watch]メニューオプションを選択します。

Instant Watch ダイアログボックスが開きます。



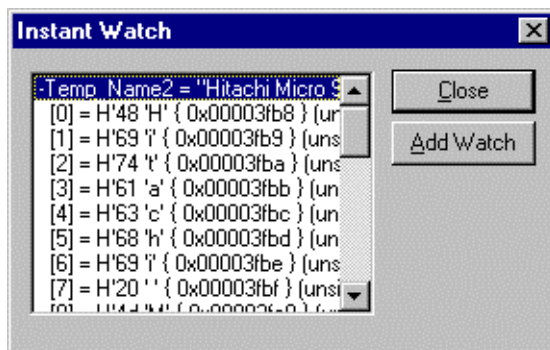



図12-2 Instant Watch ダイアログボックス

この変数を Watch ウィンドウのウォッチ項目リストに追加するには、[Add Watch]ボタンをクリックします。

## 12.3 ウォッチ項目の使用

プログラムを実行しながら、変数の値の変化を確認するには、Watch ウィンドウを使用します。Watch ウィンドウを開くには、[View->Watch]メニューオプションを選択するか、ツールバーの Watch ボタンをクリックします。Watch ウィンドウは、最初は空白です。

### 12.3.1 ウォッチ項目の追加

Watch ウィンドウにウォッチ項目を追加する方法は2つあります。一方は、Source ウィンドウからアクセスする簡易方式で、もう一方は、Watch ウィンドウの Add Watch ダイアログボックスを使用する詳細方式です。

#### 簡易方式

変数を Watch ウィンドウに簡単に追加するには、Add Watch 機能を使用します。

#### Source ウィンドウでの Add Watch の使用手順

1. Source ウィンドウを開き、ウォッチしたい変数が表示された状態にします。
2. 該当する変数をクリックして、変数上にカーソルを移動します。
3. ポップアップメニューの[Add Watch...]メニューオプションを選択します。

指定した変数がウォッチ項目として追加され、Watch ウィンドウが更新されます。

### 詳細方式

詳細方式では、配列、構造体やポインタなどの複雑な表現を使用することができます。

### Watch ウィンドウでの Add Watch の使用手順

1. Watchウィンドウを開きます。
2. ポップアップメニューの[Add Watch...]メニューオプションを選択します。

Add Watch ダイアログボックスが開きます。

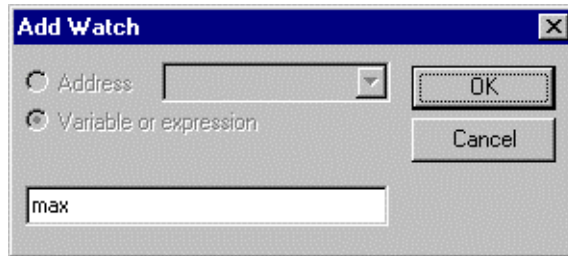


図12-3 Add Watch ダイアログボックス

ウォッチしたい変数名を入力して、[OK]をクリックします。指定した変数が Watch ウィンドウに追加されます。

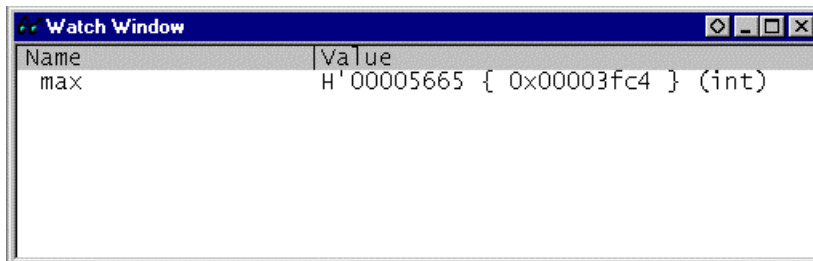


図12-4 Watch ウィンドウ

### 12.3.2 ウォッチ項目の拡張

ウォッチ項目がポインタ、配列、または構造体の場合は、その項目名の左側にプラス記号(+)の拡張インジケータが表示されます。これは、そのウォッチ項目を拡張できることを表しています。ウォッチ項目を拡張するには、その項目をダブルクリックします。その項目が拡張され、各構成要素(構造体または配列の場合)またはデータ値(ポインタの場合)が、タブ1つ分インデントされて表示されます。プラス記号はマイナス記号(-)に変わります。ウォッチ項目の構成要素にもポインタ、構造体、または配列が含まれている場合は、それらの構成要素の横にもまた、拡張インジケータが表示されます。

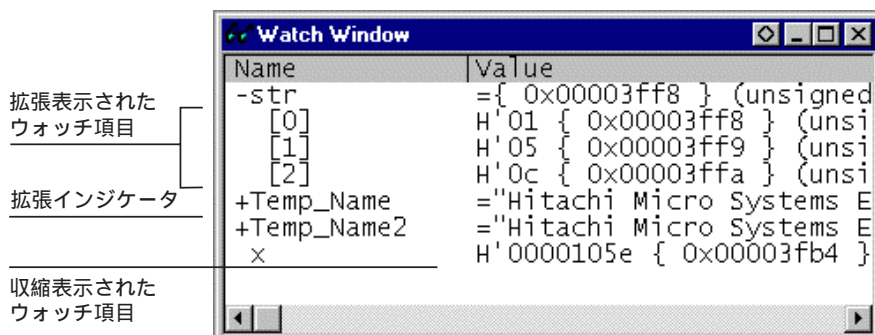


図12-5 ウォッチ項目の拡張

拡張されたウォッチ項目を圧縮するには、その項目を再度ダブルクリックします。項目の構成要素が圧縮され、1つの項目に戻ります。マイナス記号は再度プラス記号に変わります。

### 12.3.3 ウォッチ項目の表示基数の変更

ウォッチ項目の表示基数を変更するには、該当する項目をクリックして、変更する項目を選択します。ポップアップメニューの[Radix]メニューオプションを選択するとサブメニューに基数のリストが表示されます。表示したい基数を選択しクリックすると、選択した項目の表示基数が変更されます。

### 12.3.4 ウォッチ項目の値の変更

ウォッチ対象の変数の値を変更したい場合があります。たとえば、テストを実行したい場合や、プログラムにバグがあるため値が正しくない場合などです。ウォッチ項目の値を変更するには、Edit Value 機能を使用します。

#### ①ウォッチ項目の値の変更手順

1. 該当する項目をクリックします。クリックした項目上でカーソルが点滅します。
2. ポップアップメニューの[Edit Value]メニューオプションを選択します。

Edit Value ダイアログボックスが開きます。

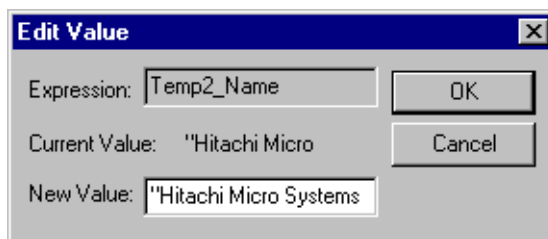


図12-6 Edit Value ダイアログボックス

New Value フィールドに新しい値または式を入力して、[OK]をクリックします。Watch ウィンドウが更新され、新しい値が表示されます。

### 12.3.5 ウォッチ項目の削除

ウォッチ項目を削除するには、該当する項目をクリックして、削除する項目を選択し、ポップアップメニューの[Delete]メニューオプションを選択します。選択した項目が削除され、Watch ウィンドウが更新されます。

【注】 Watch ウィンドウで設定したウォッチ項目をセッションファイルに保存できます。詳細については、「15 ユーザインタフェースの構成」を参照してください。

## 12.4 ローカル変数の表示

ローカル変数を見るには、[View->Locals]メニューオプションを選択して、Locals ウィンドウを開きます。

Locals ウィンドウが開きます。

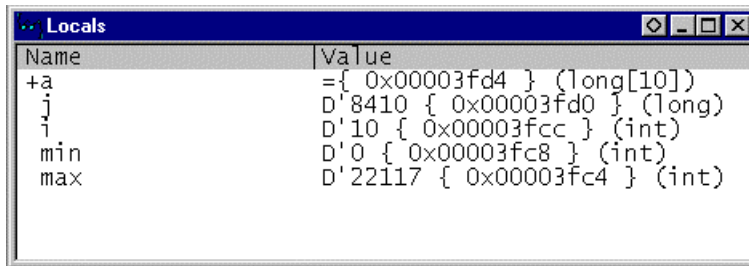


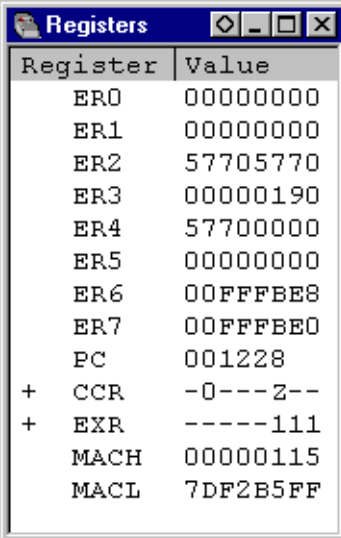
図12-7 Locals ウィンドウ

プログラムをデバッグする際には、実行開始後のステップまたはブレークに従って、Locals ウィンドウが更新され、現在のローカル変数およびそれらの値が表示されます。定義時に初期設定されていないローカル変数の場合は、値が代入されるまで、Locals ウィンドウ内の値は不定値を表示します。

ローカル変数の表示基数および値の変更は、Watch ウィンドウと同様に行えます。


## 12.5 レジスタの表示

アセンブラ表示または混合表示の Source ウィンドウを使用して、アセンブラレベルでデバッグする場合は、汎用、FPU、DSP レジスタの内容を確認できると便利です。これを行うには、Registers ウィンドウを使用します。



Register	Value
ER0	00000000
ER1	00000000
ER2	57705770
ER3	00000190
ER4	57700000
ER5	00000000
ER6	00FFFBE8
ER7	00FFFBE0
PC	001228
+ CCR	-0---2--
+ EXR	-----111
MACH	00000115
MACL	7DF2B5FF

図12-8 Registers ウィンドウ

Registers ウィンドウを開くには、[View->Registers]メニューオプションを選択するか、ツールバーの CPU Registers ボタンをクリックします。Registers ウィンドウが開き、汎用、FPU、DSP レジスタの値(16進数表記)がすべて表示されます。

### 12.5.1 ビットレジスタの拡張

コントロールレジスタやステータスレジスタのようにビット単位で使用されるレジスタの場合、レジスタ名の左側に拡張インジケータ(+)が付いています。これは拡張表示できることを意味します。(+)記号をダブルクリックするとそのレジスタが拡張表示され、拡張インジケータが(+)から(-)に変わります。

拡張表示されたレジスタがレジスタマスクのように、さらにサブグループを持つ場合は、それらの左側にも拡張インジケータ(+)が付きます。

Register	Value
R0	0000
R1	0000
R2	0000
R3	0000
R4	0000
R5	0000
R6	0000
R7	0000
PC	1004
- CCR	I0---Z--
I	1
U	0
H	0
U	0
N	0
Z	1
V	0
C	0

図12-9 ビットレジスタの拡張

拡張表示を解除するには拡張インジケータ(-)記号をダブルクリックします。拡張表示を解除すると、拡張インジケータは(-)から(+)に戻ります。

## 12.5.2 レジスタ内容の変更

レジスタの内容を変更する方法は2つあります。一方は簡易変更方式で、ウィンドウに直接入力することで値を入力できます。ただし、この方式は16進数値のみに限定されます。もう一方は詳細変更方式で、ダイアログボックスを使用して値を入力する必要がありますが、この方式を使用すると、基数の指定ができ、また複雑な式も入力できます。

### 簡易変更方式

レジスタの内容を簡単に変更するには、該当する桁をクリックするか、または、クリック・ドラッグにより変更したい桁を選択し、反転表示させます。その桁に新しい値を入力してください。値は、0-9およびa-fの範囲内であればなりません。新しい値がその桁に書き込まれ、カーソルがレジスタ内の次の桁へ移動します。レジスタの最下位桁に値を入力すると、カーソルは、次のレジスタの最上位桁へ移動します。表示されているレジスタの桁が、たとえば、CPUの条件コードレジスタ(CCR)などのビットを示している場合は、SPACEキーを押して、そのビットの値を切り替えることができます。

### 詳細変更方式

レジスタの内容を詳細変更方式で変更する場合は、Register ダイアログボックスを使用します。以下の操作のいずれかを行なって、Register ダイアログボックスを開いてください。

- 変更したいレジスタをダブルクリックする。
- 変更したいレジスタを選択し、ENTER キーを押す。
- 変更したいレジスタを選択し、ポップアップメニューの[E]dit...]メニューオプションを選択する。

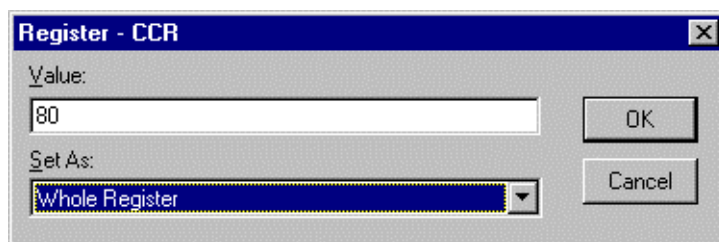


図12-10 Register ダイアログボックス

HDI の他のデータ入力フィールドと同様に、フォーマットに従った数値または C/C++ 言語の式を入力できます(「2.2 データ入力」を参照)。

ドロップダウンリスト(このリストの内容は CPU モデルおよび選択されたレジスタに依存します)からオプションを選択することにより、レジスタのどの部分の値を変更するのか (High Word、Low Word など) や、値の指定方法 (マスク形式、浮動小数点数など)、あるいはどのフラグビットを修正するのかを指定することができます。

新しい数値または式を入力し、[OK] ボタンをクリックするか、ENTER キーを押します。ダイアログボックスが閉じ、新しい値がレジスタに書き込まれます。

### レジスタ内容の使用

値を入力する際に、CPU レジスタに設定された値を使用できると便利な場合があります。たとえば、Source ウィンドウまたは Memory ウィンドウで特定アドレスを表示する場合などです。これを行うには、先頭に“#”文字を付けて、レジスタ名を指定します(例：#R1、#PC、#R6L、#ER3 など)。

---

## 13. オーバーレイ機能

---

本章では、オーバーレイを実現するための設定方法について説明します。

### 13.1 セクショングループの表示

オーバーレイ機能を利用した場合、つまり同一アドレスに複数のセクショングループを割り当てた場合、Overlay ダイアログボックスにそのアドレス範囲とセクショングループを表示します。

Overlay ダイアログボックスを開くには、[Setup->Overlay]メニューオプションを選択します。

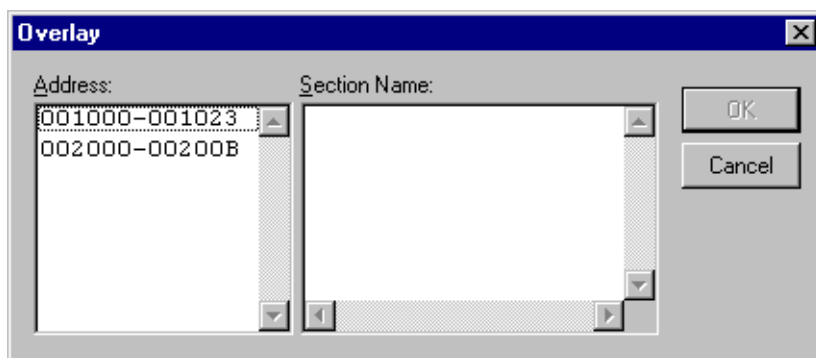


図13-1 Overlay ダイアログボックス (表示時)

このダイアログボックスには、Address リストボックスと Section Name リストボックスがあります。Address リストボックスには、オーバーレイ指定されているアドレス範囲を表示します。

Address リストボックスの中から、アドレス範囲を選択しクリックします。

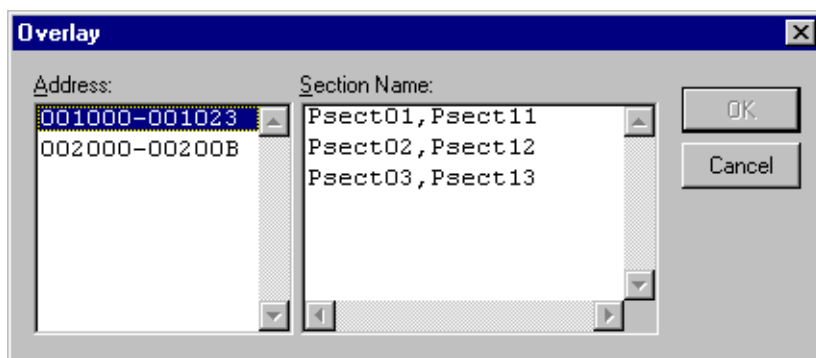


図13-2 Overlay ダイアログボックス (アドレス範囲選択時)

Section Name リストボックスに、選択したアドレス範囲に割り付けられた複数のセクショングループを表示します。



## 13.2 セクショングループの設定

オーバーレイの指定をした場合、Overlay ダイアログボックスにより、優先するセクショングループを設定する必要があります。設定しないで実行すると不正な動作をします。

まず、Address リストボックスに表示されたアドレスをクリックします。すると、そのアドレスに割り付けられた複数のセクショングループが Section Name リストボックスに表示されます。

表示された複数のセクショングループの中から、優先するセクショングループを選択しクリックします。

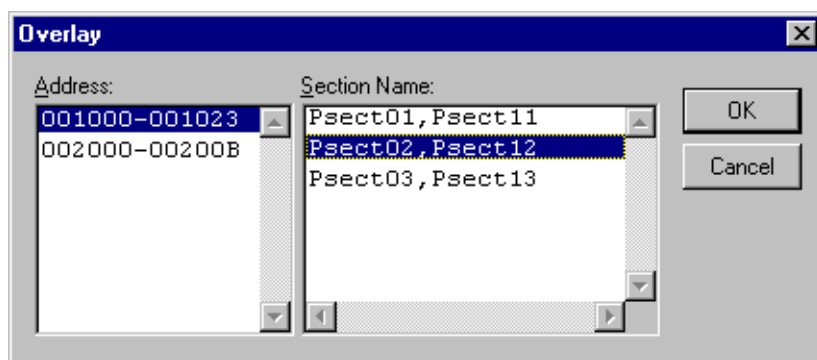


図13-3 Overlay ダイアログボックス（優先セクショングループ選択時）

セクショングループを選択後、[OK]ボタンをクリックすることにより、優先するセクショングループを設定しダイアログボックスを閉じます。

[Cancel]ボタンをクリックすると、セクションを設定しないでダイアログボックスを閉じます。

**【注】** オーバーレイ指定のアドレス範囲では、Overlay ダイアログボックスで指定したセクションのデバッグ情報を使用します。そのため、現在ロードしているプログラムのセクションと同一のセクションを Overlay ダイアログボックスで設定してください。

---

## 14. 関数の設定

---

本章では、C++プログラムの多重定義関数およびメンバ関数の設定方法について説明します。

### 14.1 関数の表示

多重定義関数およびメンバ関数は、Select Function ダイアログボックスで表示します。

次のような時、関数名による設定が可能です。

- ブレークポイントの設定
- Run Program ダイアログボックスでの関数設定
- Source ウィンドウ表示時に開く Set Address ダイアログボックスによる設定
- Memory ウィンドウ表示時に開く Set Address ダイアログボックスによる設定
- シンボルの追加および変更
- パフォーマンス・アナリシスの関数設定

上記項目に設定した関数に多重定義関数が存在する場合、あるいはメンバ関数を含むクラス名を設定した場合、Select Function ダイアログボックスが開きます。

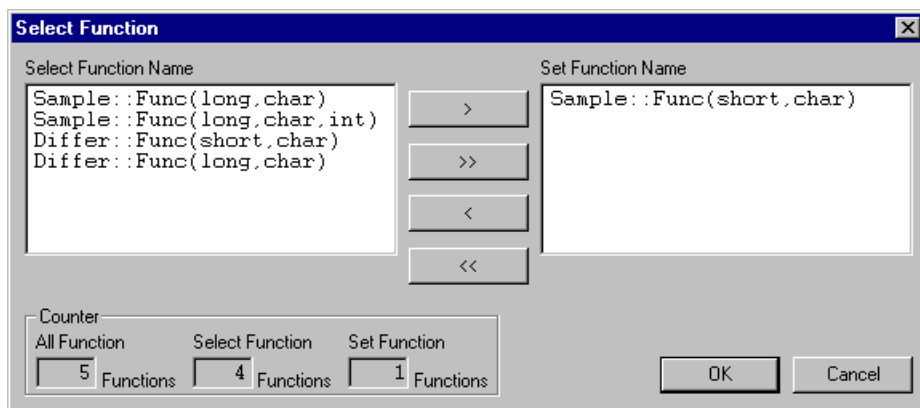


図14-1 Select Function ダイアログボックス

このダイアログボックスは、3つの領域に分割されています。

- Select Function Name リストボックス  
多重定義関数あるいはメンバ関数を詳細情報付きで表示します。
- Set Function Name リストボックス  
設定する関数を詳細情報付きで表示します。
- Counter グループエディットボックス
  - All Function :すべての同一名関数あるいはメンバ関数の個数を表示します。
  - Select Function :Select Function Name リストボックスに表示している関数の個数を表示します。
  - Set Function :Set Function Name リストボックスに表示している関数の個数を表示します。

### 14.2 関数の設定

多重定義関数およびメンバ関数は、Select Function ダイアログボックス上で選択し設定します。選択できる関数は通常1つですが、ブレークポイントを設定する場合、Run Program ダイアログボックスでの関数設定、パフォーマンス・アナリシスの関数設定では、複数選択できます。

#### 14.2.1 関数の選択

関数を選択するには、Select Function Name リストボックス上で関数を選択後 [ > ] ボタンをクリックします。クリックすることによって、選択した関数が Set Function Name リストボックス上に表示されます。また、[ ] ボタンをクリックすることによって Select Function Name リストボックス上に表示されているすべての関数を選択することができます。

#### 14.2.2 関数の削除

Set Function Name リストボックス上に表示されている関数を削除する場合は、Set Function Name リストボックス上で関数を選択後 [ < ] ボタンをクリックします。また、[ ] ボタンをクリックすることによって Set Function Name リストボックス上に表示されているすべての関数を削除することができます。

#### 14.2.3 関数の設定

Set Function Name リストボックス上に表示されている関数を設定するには、[ OK ] ボタンをクリックします。クリックすることにより、関数を設定し、ダイアログボックスを閉じます。

[ Cancel ] ボタンをクリックすると、関数を設定しないでダイアログボックスを閉じます。

## 15. ユーザインタフェースの構成

HDI ユーザインタフェースは、頻繁に行う操作にすばやくアクセスできるように、関連のある操作を論理的な順序でグループ分けをしています。しかし、デバッグ中には、ユーザインタフェース項目の配置をユーザの使いやすいように変更したり、ユーザの好みに応じて配置できるように、ユーザインタフェースをカスタマイズできるようになっています。本章では、ユーザインタフェースウィンドウの配置を変更、表示形式のカスタマイズ、設定を保存する方法について説明します。

### 15.1 ウィンドウの配置

#### 15.1.1 ウィンドウの最小化

開いたウィンドウを一時的に終了して、現在の状態で再度表示する場合は、そのウィンドウをアイコン化することができます。つまり、ウィンドウの最小化ができます。ウィンドウを最小化するには、ウィンドウの最小化ボタンをクリックするか、ウィンドウメニューで[最小化]メニューオプションを選択します。

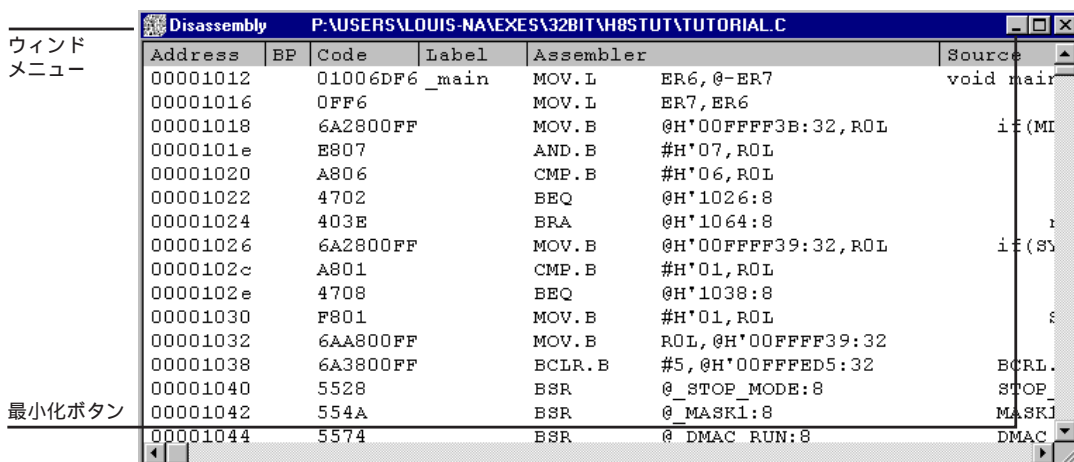


図15-1 ウィンドウの最小化

ウィンドウが最小化され、HDI アプリケーションウィンドウ左下隅にアイコンとして表示されます。上の Disassembly ウィンドウの場合、アイコンは次のようになります。



図15-2 Disassembly ウィンドウのアイコン

【注】 画面の下部に開いているウィンドウがあると、このアイコンが見えない場合があります。

アイコンをウィンドウに復元するには、アイコンをダブルクリックするか、ウィンドウメニュー

## 15. ユーザインタフェースの構成

の[Restore]を選択します。

### 15.1.2 アイコンの整列

アイコンは、デフォルトでHDIアプリケーションウィンドウの左下隅に置かれます。アイコンは、クリックして新しい位置にドラッグすればアプリケーションウィンドウ内の任意の位置に移動させることができます。アイコンをウィンドウに復元すると、最小化される前と同じ位置にウィンドウが表示されます。同様に、再び最小化すると、アイコンは最後に移動した位置に表示されます。

最小化してアイコンとなったウィンドウがいくつもあると、見づらくなります。アイコンを整理するには、[Window->Arrange Icons]メニューオプションを選択します。

アイコンがアプリケーションウィンドウの左下隅から整列します。

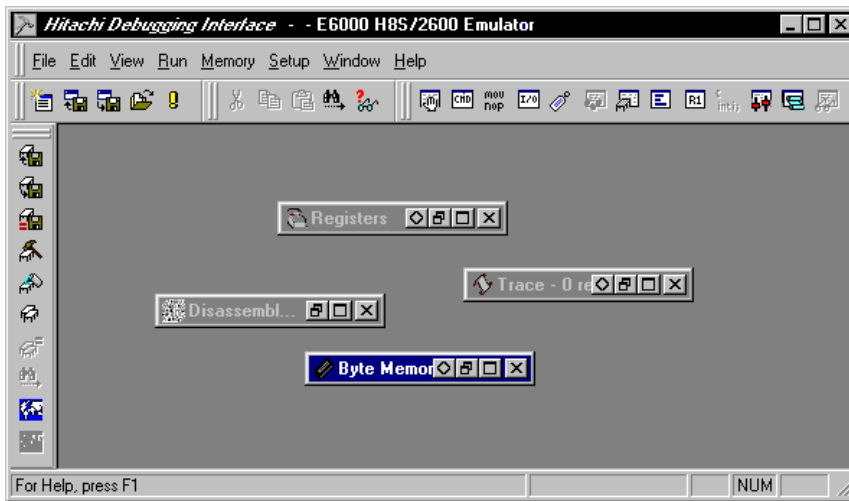


図15-3 整列前

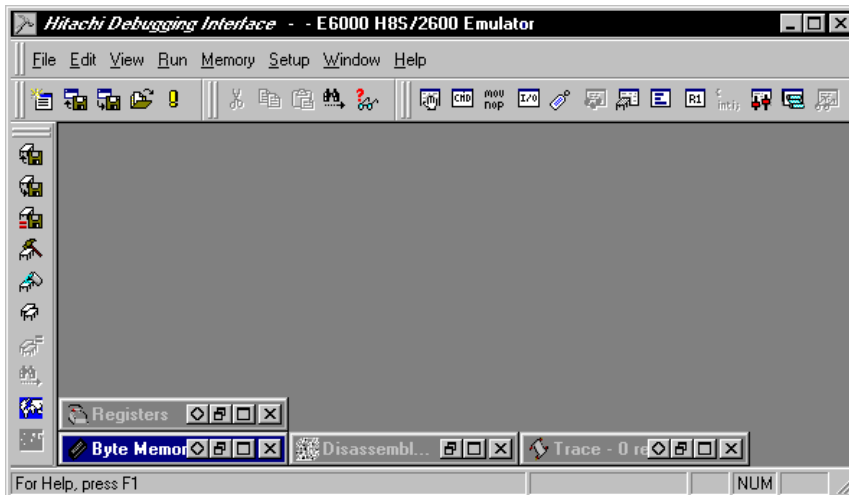


図15-4 整列後

### 15.1.3 ウィンドウのタイル表示

デバッグ後には、画面上に多くのウィンドウが開いている場合があります。Tile 機能を使用すれば、どのウィンドウも他のウィンドウと重ならないタイルフォーマットで、すべてのウィンドウを配置することができます。これを行うには、[Window->Tile]メニューオプションを選択します。

現在開いているすべてのウィンドウが、タイルフォーマットで配置されます。最小化されアイコンとなっているウィンドウは影響を受けません。

### 15.1.4 ウィンドウのカスケード表示

ウィンドウを、手前のウィンドウの後ろにウィンドウの左と上の端だけが表示されるカスケードフォーマットで配置することができます。これを行うには、[Window->Cascade]メニューオプションを選択します。現在開いているすべてのウィンドウが、カスケードフォーマットで配置されます。最小化されアイコンとなっているウィンドウは影響を受けません。

## 15.2 現在開いているウィンドウの検索

HDI アプリケーションの中に多くのウィンドウが開いていると、他のウィンドウの後ろに隠れたウィンドウを見失ってしまうことがあります。見失ったウィンドウを見つけるには、2つの方法があります。

### 15.2.1 次のウィンドウの検索

ウィンドウリスト中の次のウィンドウを手前に表示するには、ウィンドウメニューを呼び出し [Next]を選択するか、CTRL+F6 を押します。この操作を繰り返すと、すべてのウィンドウ(開いているものと最小化されているもの)を順に選択できます。

### 15.2.2 特定のウィンドウの検索

特定のウィンドウを選択するには、[Window]メニューの一番下にあるウィンドウリスト(開いているものと最小化されているもの)の中で、選択したいウィンドウをクリックします。ウィンドウリストでは、現在選択されているウィンドウの横にチェックマークが付いています。

次の例では Disassembly ウィンドウが、現在選択されているウィンドウです。

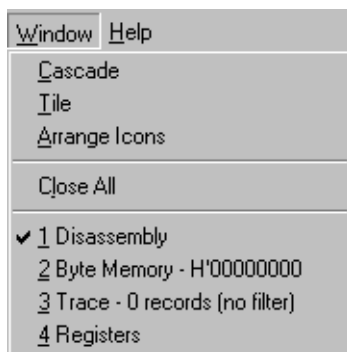


図15-5 ウィンドウの選択

選択したウィンドウが手前に表示されます。そのウィンドウが最小化されている場合は、アイコンがウィンドウに復元されます。

### 15.3 ステータスバーの表示 / 非表示

HDI アプリケーションウィンドウの下部にステータスバーを表示するかどうかを選択できます。デフォルトでは表示します。ステータスバーを非表示にするには、[Setup->Status Bar]メニューオプションを選択します。

ステータスバーが、HDI アプリケーションウィンドウの表示から削除されます。ステータスバーを再表示するには、もう一度[Setup->Status Bar]メニューオプションを選択します。ステータスバーが、HDI アプリケーションウィンドウの表示に追加されます。

### 15.4 ツールバーのカスタマイズ

ツールバーに表示されるボタンの種類と配列をカスタマイズすることができます。表示を変更するには、[Setup->Customize->Toolbar]メニューオプションを選択します。

Customize ダイアログボックスをオープンすると2枚のシートがあります。1枚目のシートは、ツールバーの表示を設定します。2枚目のシートは、ツールバーの個々のボタンを設定します。

### 15.4.1 全体概要

Toolbars シートで、表示するツールバーを選択します。

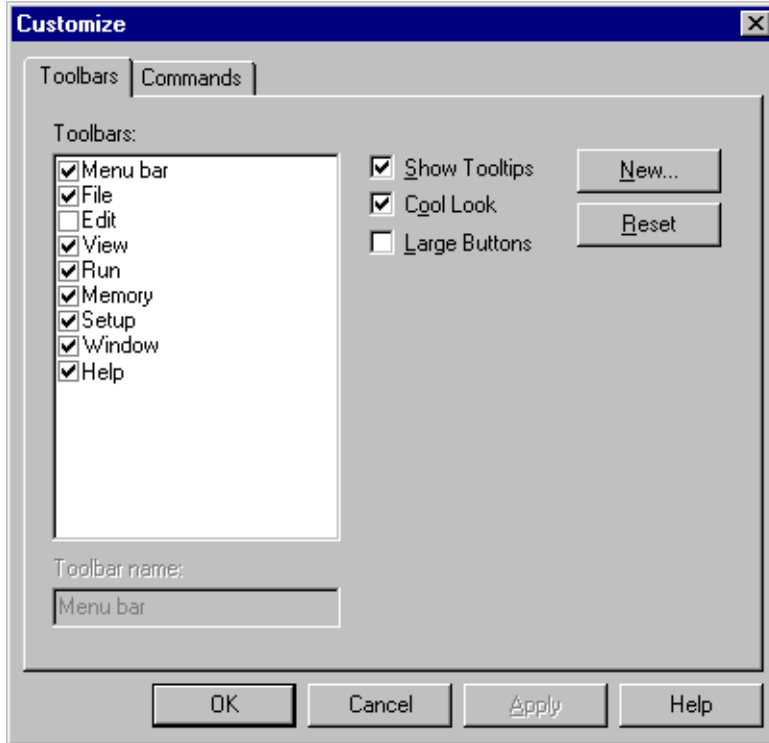


図15-6 Customize ダイアログボックス(Toolbars シート)

ツールバーは、複数選択可能なリストボックスに表示されています。個々のツールバーを非表示にする場合は、ツールバー名(ツールバーがメインフレームウィンドウに固定されていない場合に表示されるタイトルバーの名前)の隣のチェックをクリアします。

【注】 メニューバーは、チェックをクリアすることはできません。

簡素化されたデスクトップエリアで HDI を使用する場合には 'Cool Look' のチェックをはずすと Windows® 3.1 スタイルのメニュー、ツールバーになります。

ユーザ定義のツールバーを追加することが可能です。[New...] ボタンをクリックし、定義するツールバー名を入力してください。Toolbar Name エディットボックスで編集することができます。新しいツールバー「My Toolbar」として説明します。「My Toolbar」は、メインフレームの左上に現れます。ボタンがないのでボタンを追加するには、ツールバーをカスタマイズしなければなりません。



## 15.4.2 ツールバーのカスタマイズ

ユーザ定義のツールバーをカスタマイズするには、マウスまたは、他のポインティングデバイスが必要になります。キーボードしかない場合は、カスタマイズできません。ツールバーは、マウスのみでしか操作できないため、マウスがない場合にはカスタマイズする必要がないためです。

Customize ダイアログボックスの Commands シートでそれぞれのツールバーのボタンを設定できます。

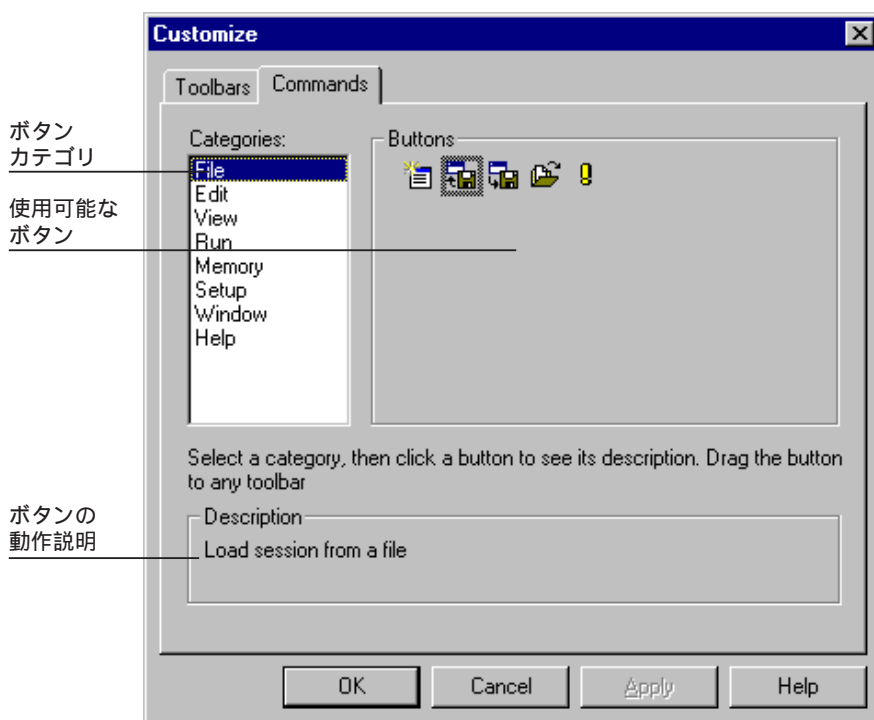


図15-7 Customize ダイアログボックス(Commands シート)

## 15.4.3 ボタンカテゴリ

ダイアログボックスの左上部は、ボタンカテゴリのリストです。それぞれのカテゴリに対応するボタンは右側に表示されます。リスト中のボタンをクリックすると、Description にボタンの動作についての説明が表示されます。

## 15.4.4 ツールバーへのボタンの追加

● ツールバーにボタンを追加する方法を以下に示します。

1. ボタンカテゴリのリストから該当するボタンカテゴリを選択します。
2. 動作リストからボタン項目を選択します。
3. ボタン項目をダイアログからドラッグするとツールバーにボタンが追加されます。

### 15.4.5 ツールバーのボタンの位置変更

☛ ツールバーのボタンの位置を変更するには

1. ツールバー中の移動するボタンを選択します。
2. そのボタンをツールバー内の移動先の位置でドロップします。

【注】 Ctrl キーを押しながらドロップするとボタンをコピーできます。

### 15.4.6 ツールバーからのボタンの削除

☛ ツールバーのボタンを削除するには

1. ツールバーから削除するボタンを選択します。
2. 選択したボタンをメインフレーム内のツールバーの外に移動します。

## 15.5 フォントのカスタマイズ

テキスト形式のウィンドウのフォントをカスタマイズすることができます (Source ウィンドウや Memory ウィンドウ)。また、新しいウィンドウをオープンしたときに使われるデフォルトのフォントを設定できます。

フォントを変更するには、[Setup->Customize->Font]メニューオプションを選択します。Font ダイアログボックスがオープンします。

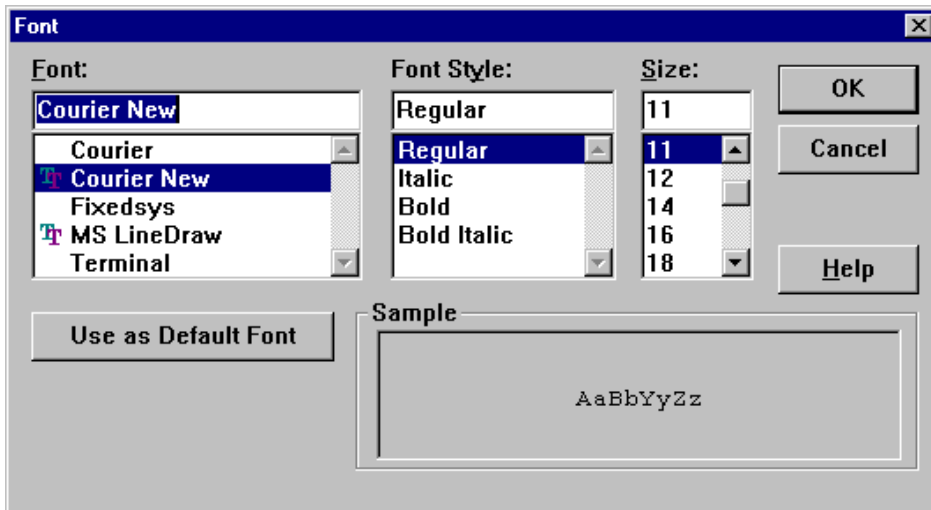


図15-8 Font ダイアログボックス

このダイアログボックスは、標準的な Windows®のフォントダイアログボックスと同じ操作ができますが、Font リストボックスには、固定長幅のフォントのみ表示されます。また、[Use as Default Font] ボタンを押すと、新しいウィンドウをオープンしたときに使われるフォントを設定することができます。

## 15.6 ファイルフィルターのカスタマイズ

Open ダイアログのファイルフィルタをカスタマイズできます。

フィルタを変更するには、[Setup->Customize->File Filter]メニューオプションを選択します。Customize File Filter ダイアログボックスがオープンします。

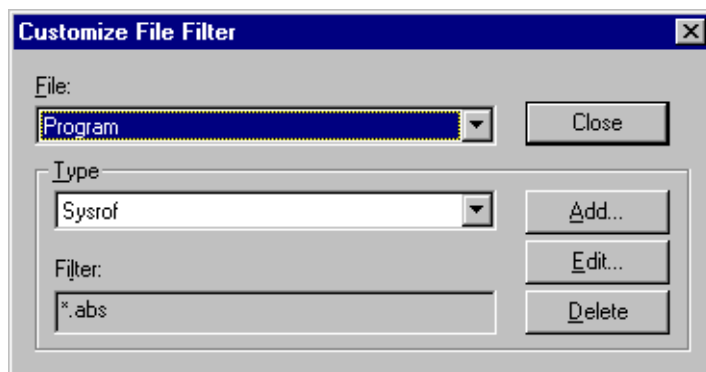


図15-9 Customize File Filter ダイアログボックス

【注】 このダイアログボックスでの変更は直ちに反映されます。変更を取り消す事はできません。

フィルターを編集するには

1. File一覧からファイルグループを選択します。
2. Type一覧から対応するタイプ名を選択します。
3. [Edit...]ボタンをクリックするとEdit Filterダイアログボックスが開きます。ダイアログボックスのタイトルには、選択されているファイルグループが表示されます。エディットボックスにはフィルタタイプまたは拡張子として指定できる文字以外は入力できません。
4. ファイル拡張子またはフィルタタイプを編集します。同時に2つ以上の拡張子を指定する場合は、各拡張子をセミコロンで区切ってください。以下に例を示します。

例

\*.mot; \*.a20; \*.a37

新しいフィルターを入力するには

1. File一覧からファイルグループを選択します。
2. [Add...]ボタンをクリックするとAdd Filterダイアログボックスが開きます。ダイアログボックスのタイトルには、選択されているファイルグループが表示されます。エディットボックスにはフィルタとして指定できる文字以外は入力できません。
3. 追加するフィルタタイプと、フィルタに使用したい拡張子を登録します。

【注】 指定されたフィルタタイプと同じタイプのフィルタが既に存在する場合は、新しく入力されたフィルタが有効となります。

フィルタを削除するには

1. File一覧からファイルグループを選択します。
2. Type一覧から対応するタイプ名を選択します。

3. [Delete]ボタンをクリックすれば、タイプ名を削除します。

## 15.7 セッションの保存

ユーザプログラムがデバッグプラットフォームにダウンロードされ、対応するソースファイルが表示されていて、かつ多くのウィンドウが開いている場合は、次回このプログラムをロードする時にこうした情報のセットアップに時間がかかる場合があります。HDI では、セットアップ時間短縮のために現在の設定をファイルに保存することができます。

すでに命名されているセッションや、現在のオブジェクトファイルと同名のセッションを新規に生成する場合は、[File->Save Session]メニューオプションを選択してセッションを更新することができます。

現在の設定を新しい名前では保存するには、[File->Save Session As...]メニューオプションを選択します。これにより、ファイル名を要求する標準的なファイルダイアログボックスを表示します。HDI セッションファイル(\*.hds)、ターゲットセッションファイル(\*.hdt)とウォッチセッションファイル(\*.hdw)の3つのファイルが保存されます。HDI セッションファイルには、すべての開いているウィンドウとその位置などの HDI インタフェース設定が含まれます。ターゲットセッションファイルには、デバッグプラットフォームの名前と構成など、デバッグプラットフォーム/ターゲットシステムに固有の設定が含まれます。ウォッチセッションファイルには、現在の Watch ウィンドウの変数の情報等を保存します。

これらのファイルが保存されると、HDI タイトルバーの第2 エントリとしてセッション名が表示されます。

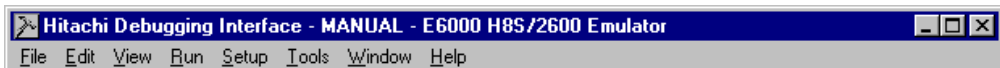


図15-10 セッション名の表示

- 【注】 セッションファイルにはシンボルやメモリ情報を保存しないため、変更した情報を再度使用したい場合は別途それぞれのファイルに保存してください。詳細は、「9.6 メモリ領域の保存」、「5.6.11 Save As ...」を参照してください。

## 15.8 セッションのロード

保存したセッションを再ロードするには、[File->Load Session...]メニューオプションを選択します。これにより、HDI セッションファイル名(\*.hds)を要求する標準的な Windows® ファイルダイアログボックスを表示します。

現在開いているウィンドウがあればクローズされ、デバッグプラットフォームへの接続が初期化されます。ユーザプログラムがターゲットにダウンロードされている場合は、ステータスバーが進行状況を表示します。ダウンロードが完了すると、ウィンドウが開かれ、更新されて、ターゲットからの最新情報が表示されます。

## 15.9 HDI オプションの設定

HDI インタフェースを使用するとき、役立つ設定があります。[Setup->Options...]メニューオプションを選択すると、HDI Options ダイアログボックスを表示します。

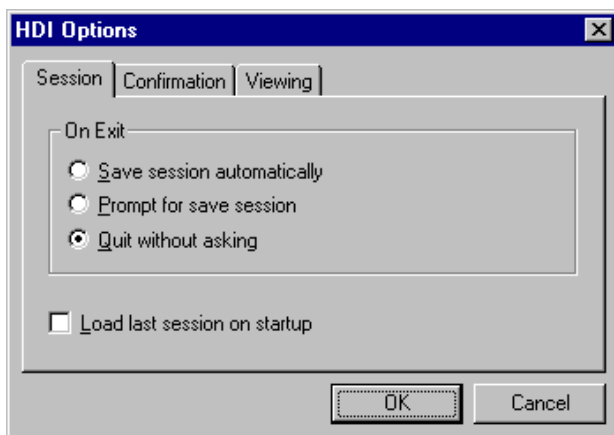


図15-11 HDI Options ( Session ) ダイアログボックス

On Exit のラジオボタングループは、プログラム終了時のカレントセッションの自動保存に使用できます。

- Save session automatically - カレントセッションファイルのセッション情報を保存します。カレントセッションファイルがない場合は、HDI セッションファイル名を入力するよう求められます。
- Prompt for save session - プログラム終了時に、カレントセッションを保存したいかどうかを毎回尋ねてきます。Yes を選択すると、カレントセッションファイルにセッション情報が保存されます。カレントセッションファイルがない場合は、HDI セッションファイル名を入力するよう求められます。
- Quit without asking - カレントセッション情報を保存するかどうかを尋ねず、保存も行わないでプログラムを終了します。

次にプログラムを起動する際、最後に保存したセッションを自動ロードしたい場合は、Load last session on startup チェックボックスをチェックします。

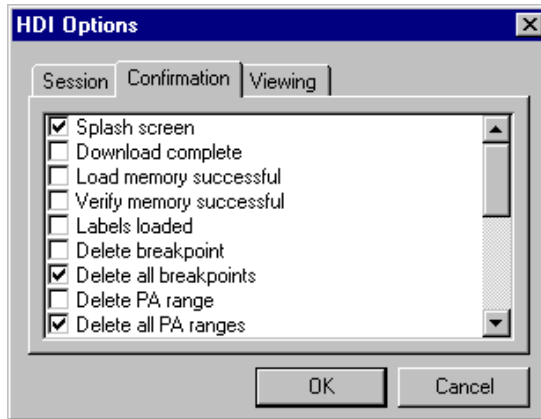


図15-12 HDI オプション(Confirmation)ダイアログボックス

Confirmation シートで、確認メッセージボックスの表示 / 非表示を切り替えられます。

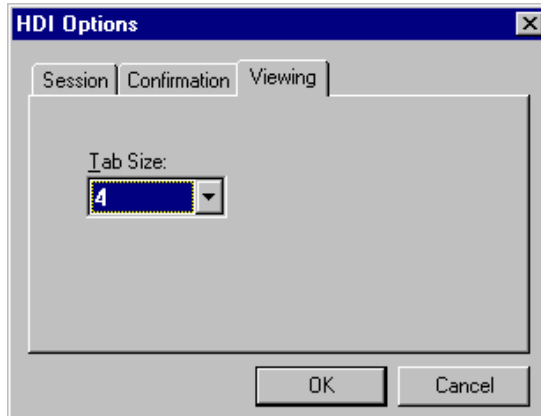


図15-13 HDI オプション(Viewing)ダイアログボックス

Tab Size リストボックスは、タブを何文字の空白にするかを設定します。指定可能な値は 2 から 8 までです。通常使用しているエディタの空白数に合わせることを推奨します。

## 15.10 デフォルト基数の設定

HDI では、いくつかの基数で数値を表示できます。デフォルトは、16進数です。ただし、Count フィールドは、常に10進数です。「2.2.2 データ形式」で説明した接頭コードのいずれかを使用することができます。使用入力を簡単にするために、これらのフォーマットのいずれかをデフォルトとして選択することができます。すなわち、その基数を使用する際に対応する接頭コードを入力する必要がありません。

デフォルトの基数を変更するには、[Setup->Radix]メニューオプションを選択します。これにより、使用可能な数値表示システムのリストを表示します。現在選択されている基数の左にチェックマークが付いています。

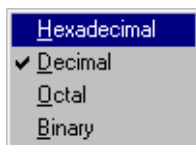


図15-14 基数の設定

## 付録A. システムモジュール

本章では、HDI デバッグシステムのアーキテクチャを説明します。

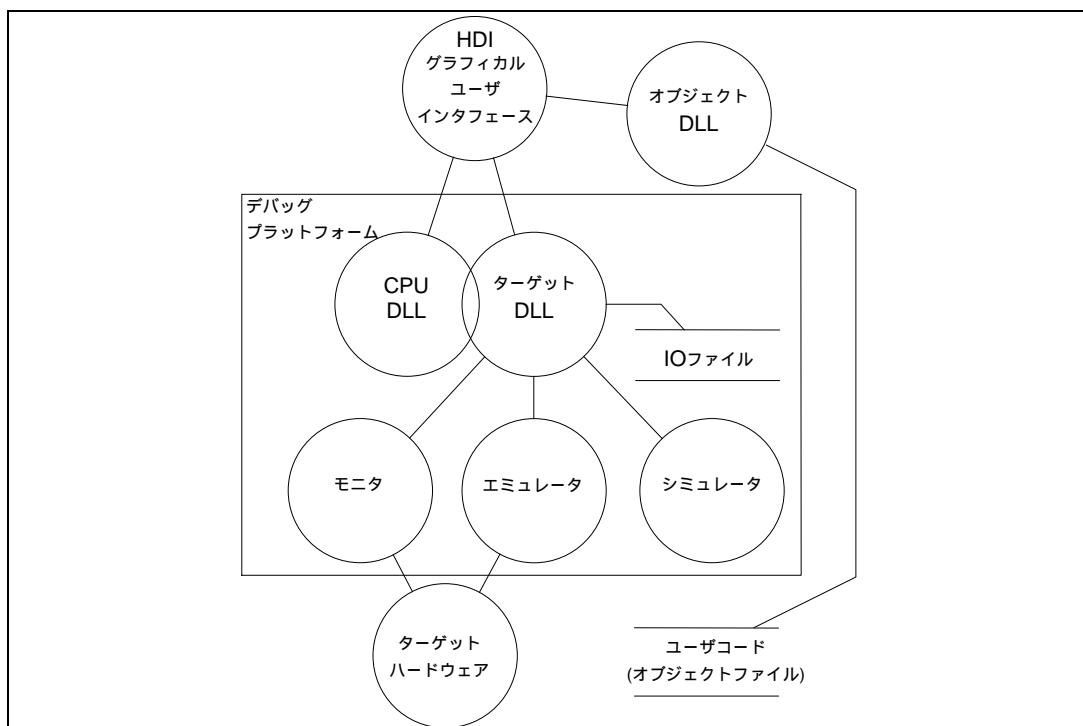


図 A-1 HDI システムモジュール

通常の動作では、ユーザプログラムをターゲットハードウェアに直接格納します(たとえば EPROM として)。HDI はこの情報を使用して、Windows®ベースのデバッグシステムを提供します。

異なるデバッグプラットフォームやターゲットハードウェアに交換したとき、デバッグシステムの使用法を習熟する時間を短縮するため、HDI は、統一したインタフェース(GUI)とターゲット固有のモジュールファミリーを提供しています。通常は、標準 GUI は共通です。ユーザが適切なターゲットモジュールを選択すれば、システムの残りの部分が自動的に適切なモジュールをロードして自己設定します。

### A.1 グラフィカルユーザインタフェース

これは、Windows®上で動くメインの HDI.EXE プログラムです。ユーザになじみのある Windows®の操作を採用しており、メニューとウィンドウは、デバッグシステムの内容をユーザフレンドリーな方法で表示します。GUI は、ユーザがシステムの残りの部分が接する唯一の接点であり、コマン



ドを処理して、ユーザプログラムに関する必要な情報を提供します。また、モジュール DLL とホストファイルシステム、すなわち PC との間のインタフェースも提供します。

## A.2 オブジェクト DLL

コンパイラツールはユーザプログラムを作成する際、アブソリュートファイルを作成します。このファイルには、ターゲットアプリケーションを構成する機能を実行するため、マイクロコンピュータが処理する実際のマシンコードおよびデータが含まれます。オリジナルソースコードとしてユーザプログラムをデバッグするためには、コンパイラツールはデバッガにさらに情報を提供しなければなりません。このため、コンパイラツールは、ソースコードのデバッグに必要なすべての情報をアブソリュートファイルに格納するデバッグオプションを備えています。このようなアブソリュートファイルは通常、デバッグオブジェクトファイルと呼ばれます。

オブジェクト DLL は、オブジェクトファイルからこの情報を抽出し、ユーザに対して表示します。データのフォーマットはコンパイラツールに依存するため、HDI ディレクトリには複数のオブジェクト DLL を入れておくことができます。HDI はオブジェクトファイルのフォーマットが理解できるオブジェクト DLL をみつけるまで、1つ1つを試します。
























## A.3 CPU DLL
























CPU DLL モジュールには、ターゲットのマイクロコンピュータに関する情報が含まれています。たとえば、マイクロコンピュータが使用できる、レジスタの数と種類が含まれており、また、ターゲット中のマシンコードと Source ウィンドウに表示するアセンブラニーモニックとの間で、翻訳を行います。













## A.4 ターゲット DLL

ターゲット DLL は、デバッグプラットフォームの機能を HDI に通知し、正しい CPU DLL を選択します。ターゲットの幾つかの機能は、一般的になり得ません(たとえばターゲット構成)。ターゲット DLL には、これらの機能へのアクセスをユーザに提供するため、標準 GUI への拡張機能も含まれています。

## 付録B. GUI コマンド一覧

メニュー	メニューオプション	ショートカットキー	ツールバーボタン
File	New Session...	Ctrl+N	    
	Load Session...	Ctrl+O	
	Save Session	Ctrl+S	
	Save Session As...		
	Load Program...		
	Initialize		
Exit	Alt+F4		
Edit	Cut	Ctrl+X	    
	Copy	Ctrl+C	
	Paste	Ctrl+V	
	Find	F3	
	Evaluate		
View	Breakpoints	Ctrl+B	            
	Command Line	Ctrl+L	
	Disassembly...	Ctrl+D	
	I/O Area	Ctrl+I	
	Labels	Ctrl+A	
	Locals	Ctrl+Shift+W	
	Memory...	Ctrl+M	
	Performance Analysis	Ctrl+P	
	Profile-List	Ctrl+F	
	Profile-Tree	Ctrl+Shift+F	
	Registers	Ctrl+R	
	Source...	Ctrl+K	
	Status	Ctrl+U	

メニュー	メニューオプション	ショートカットキー	ツールバーボタン
View	Trace	Ctrl+T	
	Watch	Ctrl+W	
Run	Reset CPU		
	Go	F5	
	Reset Go	Shift+F5	
	Go To Cursor		
	Set PC To Cursor		
	Run...		
	Step In	F8	
	Step Over	F7	
	Step Out		
	Step...		
Halt	Esc		
Memory	Refresh	F12	
	Load...		
	Save...		
	Verify...		
	Test...		
	Fill...		
	Copy...		
	Compare...		
	Search...		
	Configure Map...		
	Configure Overlay...		

メニュー	メニューオプション	ショートカットキー	ツールバーボタン
Setup	Options... Radix (Input) > Hexadecimal Decimal Octal Binary Customize > Toolbar... Font... File Filter...		       
	Configure Platform...		
Window	Cascade Tile Arrange Icons		  
	Close All		
Help	Index Using Help Search for Help on	F1	
	About HDI		



---

## 付録C. シンボルファイルのフォーマット

---

HDI がシンボルファイルを正しく理解しデコードするためには、決められた方法でファイルをフォーマットする必要があります。

1. ファイルは、ASCIIテキストファイルでなければなりません。
2. ファイルの先頭には、“BEGIN”という語を指定する必要があります。
3. 各シンボルは、別個の行にあり、最初に“H”で終わる16進数の値、続いてスペース、次にシンボルテキストが指定されていなければなりません。
4. ファイルの終わりには、“END”という語を指定する必要があります。

例:

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



---

## 索引

---

.pro .....	71, 75, 78
.sni .....	73, 75
About HDI.....	35
Acquisition.....	59
Add.....	37, 45
Add Range .....	52
Add Watch.....	44, 56, 62, 132, 133
Address .....	43, 45, 55
Arrange Icons.....	35, 144
ASCII.....	5, 161
Assembler .....	109
BP .....	45, 55
Break	
Break Access .....	39
Break Data.....	25, 39
Break Register .....	25, 39
Break Sequence .....	39
– Break Sequence ダイアログボックス .....	39, 40
Breakpoints	
– Breakpoints ウィンドウ.....	37
PC Breakpoint .....	39
BREAK	
BREAK_ACCESS ( BA ) .....	23
BREAK_DATA ( BD ) .....	23
BREAK_REGISTER ( BR ) .....	23
BREAK_SEQUENCE ( BS ) .....	23
BREAKPOINT ( BP ) .....	23
Breakpoints .....	29, 37, 125, 126, 127, 128
Called.....	70
Cascade .....	34, 145
Clear.....	59
Clear Data.....	71, 74, 77
Close All.....	35



Code and Assembler.....	43
Command Line.....	29, 41, 80, 86
Compare .....	33, 51
Configure Map.....	33
Configure Overlay .....	33
Configure Platform .....	34
Continue.....	64
Control Registers .....	66
Copy.....	28, 33, 42, 43, 46, 49, 51, 54, 58, 62, 68
Copy Memory.....	33, 51, 118
Customize .....	34, 149, 150
Cut .....	28
Cycle .....	70, 73
Delete.....	38, 47, 62, 135
Delete All.....	38, 47, 62
Delete All Ranges .....	53
Delete Range .....	52
Description .....	148
Disable/Enable.....	38
Disassembly.....	29, 42, 108, 109, 121, 122, 143, 146
Disassembly ウィンドウ.....	58
Edit.....	28, 38, 45, 54, 115, 127, 138
Edit Range .....	52
Edit Value.....	49, 62, 134
EEPMOV 命令.....	10, 22
EEPROM.....	10
ELF/DWARF フォーマット.....	2
Enable Analysis.....	53
Enable Profiler.....	70, 74, 77
Evaluate .....	29
Execution Mode .....	63
Exit.....	28, 152
Expands Size .....	76
EXR.....	11
External Tool.....	31
File .....	126, 151
File Filter .....	150
Fill .....	33, 51

---

Fill Memory ダイアログボックス .....	25
Fill Memory .....	33, 51
Filter .....	59
Find .....	29, 46, 56, 59, 71, 74
Find Data .....	74
Find Next .....	46, 59
Font .....	149
Go .....	31, 122, 126, 129
Go Reset .....	129
Go To Cursor .....	31, 43, 56, 122, 128
Go to Disassembly .....	57
Go to Source .....	38, 44, 68
Halt .....	32, 59, 121, 125
HDI .....	1
HDI Options .....	34, 152
HDI セッションファイル .....	151
Help .....	6, 35
Index .....	35
Initialize .....	28
Instant Watch .....	44, 56, 131, 132
Label .....	43, 55
Labels .....	29, 44, 110
Line .....	55
Load .....	32, 48, 50
Load Memory .....	32, 119
Load Program .....	28
Load Session .....	27, 151
Localized Dump .....	30
Locals .....	29, 49, 135
Logging .....	42
MACS ビット .....	11
MAC 命令 .....	11
Memory .....	29, 32, 50, 113, 115, 117, 118, 141, 149
Memory ウィンドウ .....	25
Memory Map .....	64, 65
Memory Map Modify ダイアログボックス .....	64
Memory Map ダイアログボックス .....	65
Memory Mapping .....	33

Minimize .....	143
MOVFP 命令 .....	22
MOVTPE 命令 .....	22
Multiple View .....	77
Name.....	45
New.....	147
New Session .....	27
Next.....	145
Open .....	30
Open Localized Dump.....	30
Open Memory Window.....	29, 113
Options.....	34, 152
Output Profile Information File.....	71, 75, 78
Output Text File.....	71, 75
Overlay.....	33, 139, 140
Paste.....	28
PC .....	10, 24, 29, 31, 43, 44, 56, 80, 91, 100, 101, 112, 121, 129, 138
PC Breakpoint .....	39
Performance	
Performance Analysis	
– Performance Analysis ウィンドウ.....	52, 53
Performance Option ダイアログボックス .....	53
Performance Analysis .....	29, 52, 90
Play .....	41
Profile-Chart.....	76
Profile-List .....	30, 70
Profile-Tree .....	30, 73
Radix.....	5, 34, 49, 62, 134, 154
Reduces Size .....	76
Refresh.....	32, 50
Register.....	138
Registers.....	30, 54, 136
Reset Counts/Times.....	53
Reset CPU .....	31
Reset Go .....	31, 121
Restart.....	59
Restore.....	144
RN ( Round to Nearest ) .....	25, 64

Round Mode.....	64
Run .....	31, 121, 122, 123, 125
Run Program.....	31, 128, 129, 141
RZ ( Round to Zero ) .....	25, 64
Save.....	32, 48, 50, 59, 118
Save As.....	48
Save Memory As.....	32, 118
Save Profile Information File .....	71, 75, 78
Save Session .....	27, 151
Save Session As .....	28, 151
Save Text of Profile Data .....	71, 75
Search.....	51, 116
Search Memory ダイアログボックス.....	25
Search for Help on .....	35
Select	
Select Function ダイアログボックス.....	40, 53
Select Session ダイアログボックス.....	63
Select All .....	42
Select Data.....	71, 75
Select Function .....	43, 51, 56, 111, 115, 129, 141, 142
Set	
Set Break ダイアログボックス.....	25, 39
Set Address.....	43, 51, 56, 111, 112, 115, 141
Set Batch File.....	41
Set Break .....	127
Set Line .....	56
Set Log File.....	41
Set PC Here.....	44, 56
Set PC To Cursor.....	31
Setting.....	71, 75
Setting Profile-List .....	71
Setting Profile-Tree .....	75
Setup.....	5, 33, 139, 146, 149, 150, 152, 154
Simulated I/O .....	30
Simulated I/O ウィンドウ.....	13, 67
SLEEP.....	23, 25
SLEEP 命令 .....	22
Snapshot.....	59

Source.....	30, 43, 55, 107, 108, 111, 112, 121, 122, 126, 128, 131, 132, 138, 141, 149
Source ウィンドウ .....	26, 58
SR .....	24
Stack Trace .....	30
Stack Trace ウィンドウ .....	26, 68
Stack Trace Setting .....	69
Status .....	30
Status Bar .....	33, 146
Step.....	31, 32
Step In.....	31, 123
Step Out.....	32, 123
Step Over .....	32, 123, 124
Step Program .....	32, 123
Stop.....	64
Stop ボタン.....	23, 25
S-type フォーマット .....	2
Symbol .....	37
SYSCR.....	66, 67
SYSCR.....	11
SYSCR アドレス.....	11
SYSCR ダイアログボックス.....	11
System Call Address.....	14, 63
System Configuration .....	65
System Configuration ダイアログボックス.....	10, 14, 23, 25
System Memory Resource .....	65, 66
System Memory Resource Modify ダイアログボックス.....	10, 65, 66
System Status .....	30, 57
Test .....	33, 50
Test Memory .....	33, 50
Tile .....	34, 145
Toggle Bit.....	54
Toolbar.....	34, 146
Tooltip Watch .....	131
Trace .....	30, 58
Trace Acquisition ダイアログボックス.....	25, 60
Trace Search ダイアログボックス.....	61
Trace Acquisition.....	59
Trace Search.....	59

Trim Source.....	60
Update.....	58
Using Help.....	35
Verify.....	32
Verify S-Record File with Memory.....	32
View.....	29, 107, 108, 110, 113, 126, 132, 135, 136
View Profile-Chart.....	70, 73, 77
View Profile-List.....	73, 76
View Profile-Tree.....	70, 76
View Setting.....	69
View Source.....	46, 59, 70, 73, 76
Watch.....	30, 44, 56, 61, 62, 131, 132, 133, 134, 135
Window.....	34
アクセス	
アクセスサイクル数.....	10, 64, 65
アクセス種別.....	10, 39, 65, 66
メモリアクセスエラー.....	24
インフォメーションメッセージ.....	105
エラー	
アドレスエラー.....	24
エラーメッセージ.....	23, 105
シミュレーションエラー.....	23, 63
メモリアクセスエラー.....	24
関数呼び出し回数.....	30
関数呼び出し履歴.....	26, 68
クラス名.....	40, 53
コマンド.....	79
サイクル数	
アクセスサイクル数.....	10, 64, 65
実行サイクル数.....	1, 12, 22, 52, 58, 61
システムコール.....	63, 67
システムコントロールレジスタ.....	66
実行	
実行モード.....	65
実数データ.....	25
シミュレーションエラー.....	23, 63
乗算器内部フラグ.....	12
多重定義関数.....	40, 53

デバッグ対象プログラム.....	2
デバッグプラットフォーム名.....	9
トレース.....	60, 61
トレース情報取得条件の設定.....	60
トレース情報のサーチ.....	61
トレース情報の表示.....	58
トレースバッファ.....	23, 25, 58, 61
内蔵 I/O.....	10
ニーモニック.....	12, 58, 61
パイプライン	
パイプラインリセット.....	58
バス幅	
アドレスバス幅.....	65
データバス幅.....	10, 64, 65
パフォーマンス・アナリシス.....	52
パフォーマンス・アナリシスの設定.....	53
パラメータブロック.....	14
非正規化数.....	25
標準入出力.....	13, 22, 63, 67
ファイル入出力.....	13, 22, 63, 67
浮動小数点.....	25
不当命令.....	24
ブレーク	
通過アドレスの設定.....	40
ブレーク種別.....	25, 37, 39
ブレーク条件.....	23
- ブレーク条件の設定.....	39
ブレークポイント.....	37
- ブレークポイントの表示.....	37
プロファイルデータ.....	70, 73
丸めモード.....	25, 64
命令実行不正.....	24
命令実行リセット.....	10
メモリ	
メモリアクセスエラー.....	24
メモリ種別.....	10, 64, 65
メモリマップ.....	65
- メモリマップの設定.....	9, 64

---

- メモリマップの表示.....	65
メモリリソース.....	65
メモリリソースの確保.....	10
メモリリソースの設定.....	66
メンバ関数.....	40, 53
モード	
続行モード.....	24
停止モード.....	24, 64
例外	
例外処理.....	4, 10





# H8S, H8/300 シリーズ 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-702-163F