

お客様各位

---

## カタログ等資料中の旧社名の扱いについて

---

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

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

2010年4月1日

ルネサスエレクトロニクス株式会社

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

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

## ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りが無いことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。  
標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット  
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）  
特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

お客様各位

---

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

---

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

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

2003年4月1日  
株式会社ルネサス テクノロジ  
カスタマサポート部

## ご注意

### 安全設計に関するお願い

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

### 本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (<http://www.renesas.com>)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものです。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、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

# ルネサスデバッグインタフェース

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

本マニュアルを使用する際には、以下の点に留意してください。

1. 本マニュアルは、全体または一部が予告なしに変更される場合があります。
2. 本マニュアルの一部または全部を（株）日立製作所の許可を得ることなく、複製または複写することはできません。
3. 日立は、本マニュアルに従って操作を行っている間の事故または他の原因によるユーザの装置への損害については、責任を負いません。

## まえがき

日立デバッグインタフェースの使用前に本マニュアルを必ずお読みください。

ユーザーズマニュアルは、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) 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について簡単に紹介し、このインタフェースのおもな特長を示します。

次の各章、「デバッグの準備」、「プログラムの表示」、「プログラムの実行」、「プログラムの停止」、「変数の表示」、「メモリの操作」、「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。

別冊のデバッグプラットフォームのマニュアルには、以下の内容が記載されています。

「セットアップ」には、コンピュータにシステムハードウェアおよびソフトウェアをインストールする手順、およびすべてのシステム要素が正しくインストールされたかどうかを検証する手順が記載されています。

「チュートリアル」には、デバッグシステムの簡単な開始・実行方法、および一般的なセッションでの各デバッグ機能の使用例が記載されています。

「xxxxxxの機能」(xxxxxxにはデバッグプラットフォームの名称が入ります)には、ブレークポイントの変更やトレース設定情報などのデバッグプラットフォーム特有なユーザインタフェースについて記載されています。

### 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ニーモニック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。

## 商標

Microsoft® およびWindows®、Windows®95、Windows®98、WindowsNT®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。



# 目次

1	はじめに	1
2	システムの概要	3
2.1	ユーザインタフェース	3
2.2	データ入力	3
2.2.1	演算子	3
2.2.2	データ形式	3
2.2.3	精度	4
2.2.4	式の例	4
2.2.5	シンボル形式	4
2.2.6	シンボル指定の例	4
2.3	ヘルプ	4
2.3.1	コンテキスト依存型ヘルプ	5
3	デバッグの準備	7
3.1	デバッグを行うためのコンパイル	7
3.2	デバッグプラットフォームの選択	7
3.3	デバッグプラットフォームの構成	8
3.3.1	設定	8
3.3.2	マッピング	8
3.3.3	ステータス	10
3.4	プログラムのダウンロード	11
4	プログラムの表示	15
4.1	コードの表示	15
4.1.1	ソースコードの表示	15
4.1.2	アセンブラコードの表示	16
4.1.3	アセンブラコードの変更	16
4.2	ラベルの表示	17
4.2.1	ラベルのリスト表示	17
4.2.2	Source ウィンドウまたは Disassembly ウィンドウでのラベルの追加	18
4.3	特定アドレスのプログラム表示	19
4.3.1	現在の PC のプログラム表示	19
4.4	テキストの検索	20
5	メモリの操作	21
5.1	メモリ領域の表示	21
5.1.1	ASCII でのメモリ表示	22
5.1.2	バイト単位でのメモリ表示	22
5.1.3	ワード単位でのメモリ表示	22
5.1.4	ロングワード単位でのメモリ表示	22
5.1.5	単精度浮動小数点形式でのメモリ表示	22
5.1.6	倍精度浮動小数点形式でのメモリ表示	22
5.1.7	別領域のメモリ表示	23
5.2	メモリ内容の変更	23
5.2.1	簡易変更方式	23
5.2.2	詳細変更方式	23
5.2.3	メモリ範囲の選択	24
5.3	メモリ内の値の検索	24
5.4	メモリ領域に値を埋める	25

5.4.1	範囲を埋める	25
5.5	メモリ領域の転送	26
5.6	メモリ領域のテスト	26
5.7	メモリ領域の保存	27
5.8	メモリ領域のロード	28
5.9	メモリ領域のベリファイ	28
6	プログラムの実行	29
6.1	リセットからの実行	29
6.2	連続実行	29
6.3	カーソル位置まで実行	30
6.4	複数ポイントまで実行	30
6.5	シングルステップ	30
6.5.1	関数内の各命令をステップ実行	31
6.5.2	関数全体を1ステップで実行	31
6.6	関数の実行停止	31
6.7	複数のステップ	31
7	プログラムの停止	33
7.1	実行の停止	33
7.2	標準ブレークポイント(PCブレークポイント)	33
7.2.1	標準ブレークポイントのタイプ切り替え	34
7.2.2	標準ブレークポイントの削除	34
7.3	Breakpoints ウィンドウ	34
7.3.1	ブレークポイントの追加	35
7.3.2	ブレークポイントの変更	35
7.3.3	ブレークポイントの削除	36
7.3.4	全ブレークポイントの削除	36
7.4	ブレークポイントを無効にする	36
7.4.1	ブレークポイントを無効にする	36
7.4.2	ブレークポイントを有効にする	36
7.5	テンポラリブレークポイント	37
7.6	ハードウェアブレークポイント(イベント)	38
8	変数の表示	39
8.1	ツールチップウォッチ	39
8.2	インスタントウォッチ	39
8.3	ウォッチ項目の使用	40
8.3.1	ウォッチ項目の追加	40
8.3.2	ウォッチ項目の拡張	41
8.3.3	ウォッチ項目の表示基数の変更	42
8.3.4	ウォッチ項目の値の変更	42
8.3.5	ウォッチ項目の削除	43
8.4	ローカル変数の表示	43
8.5	レジスタの表示	44
8.5.1	ビットレジスタの拡張	44
8.5.2	レジスタ内容の変更	45
8.5.3	レジスタ内容の使用	46
8.6	I/Oレジスタの表示	46
8.6.1	I/O Registers ウィンドウのオープン	47
8.6.2	I/Oレジスタ表示の拡張	47
8.6.3	I/Oレジスタの変更	48
9	オーバーレイ機能	49

9.1	セクショングループの表示	49
9.2	セクショングループの設定	50
10	関数の設定	51
10.1	関数の表示	51
10.2	関数の設定	52
10.2.1	関数の選択	52
10.2.2	関数の削除	52
10.2.3	関数の設定	52
11	ユーザインタフェースの構成	53
11.1	ウィンドウの配置	53
11.1.1	ウィンドウの最小化	53
11.1.2	アイコンの整列	54
11.1.3	ウィンドウのタイル表示	55
11.1.4	ウィンドウのカスケード表示	55
11.2	現在開いているウィンドウの検索	55
11.2.1	次のウィンドウの検索	55
11.2.2	特定のウィンドウの検索	55
11.3	ステータスバーの表示 / 非表示	56
11.4	ツールバーのカスタマイズ	56
11.4.1	全体概要	57
11.4.2	ツールバーのカスタマイズ	57
11.4.3	ボタンカテゴリ	58
11.4.4	ツールバーへのボタンの追加	58
11.4.5	ツールバーのボタンの位置変更	59
11.4.6	ツールバーからのボタンの削除	59
11.5	フォントのカスタマイズ	59
11.6	ファイルフィルターのカスタマイズ	60
11.7	セッションの保存	61
11.8	セッションのロード	61
11.9	HDI オプションの設定	62
11.10	デフォルト基数の設定	64
12	メニュー	65
12.1	File	65
12.1.1	New Session	65
12.1.2	Load Session	65
12.1.3	Save Session	65
12.1.4	Save Session As	65
12.1.5	Load Program	66
12.1.6	Initialize	66
12.1.7	Exit	66
12.2	Edit	66
12.2.1	Cut	66
12.2.2	Copy	66
12.2.3	Paste	66
12.2.4	Find	66
12.2.5	Evaluate	67
12.3	View	67
12.3.1	Breakpoints	67
12.3.2	Command Line	67
12.3.3	Disassembly	67

12.3.4	I/O Area.....	67
12.3.5	Labels.....	67
12.3.6	Locals.....	67
12.3.7	Memory.....	67
12.3.8	Performance Analysis .....	68
12.3.9	Registers.....	68
12.3.10	Source.....	68
12.3.11	Status.....	68
12.3.12	Trace .....	68
12.3.13	Watch.....	68
12.4	Run.....	68
12.4.1	Reset CPU.....	68
12.4.2	Go.....	68
12.4.3	Reset Go.....	68
12.4.4	Go To Cursor .....	69
12.4.5	Set PC To Cursor .....	69
12.4.6	Run.....	69
12.4.7	Step In.....	69
12.4.8	Step Over.....	69
12.4.9	Step Out.....	69
12.4.10	Step.....	69
12.4.11	Halt.....	69
12.5	Memory .....	69
12.5.1	Refresh .....	69
12.5.2	Load.....	70
12.5.3	Save.....	70
12.5.4	Verify.....	70
12.5.5	Test.....	70
12.5.6	Fill.....	70
12.5.7	Copy.....	70
12.5.8	Compare.....	70
12.5.9	Configure Map .....	70
12.5.10	Configure Overlay.....	70
12.6	Setup.....	71
12.6.1	Status Bar .....	71
12.6.2	Options.....	71
12.6.3	Radix .....	71
12.6.4	Customize.....	71
12.6.5	Configure Platform.....	71
12.7	Window .....	72
12.7.1	Cascade .....	72
12.7.2	Tile .....	72
12.7.3	Arrange Icons.....	72
12.7.4	Close All .....	72
12.8	Help.....	72
12.8.1	Index .....	72
12.8.2	Using Help .....	72
12.8.3	Search for Help on .....	72
12.8.4	About HDI.....	72
13	ウィンドウ .....	73
13.1	Breakpoints.....	73
13.1.1	Add.....	73

---

13.1.2	Edit.....	73
13.1.3	Delete.....	73
13.1.4	Delete All.....	73
13.1.5	Disable/Enable.....	74
13.1.6	Go To Source.....	74
13.2	Command Line.....	74
13.2.1	Run Batch File.....	74
13.2.2	Play.....	75
13.2.3	Set Log File.....	75
13.2.4	Logging.....	75
13.3	Disassembly.....	75
13.3.1	Copy.....	76
13.3.2	Set Address.....	76
13.3.3	Go To Cursor.....	77
13.3.4	Set PC Here.....	77
13.3.5	Instant Watch.....	77
13.3.6	Add Watch.....	77
13.4	I/O Registers.....	77
13.4.1	Copy.....	78
13.4.2	Edit.....	78
13.4.3	Expand/Collapse.....	78
13.5	Labels.....	78
13.5.1	Add.....	79
13.5.2	Edit.....	79
13.5.3	Find.....	80
13.5.4	Delete.....	80
13.5.5	Delete All.....	80
13.5.6	Load.....	81
13.5.7	Save.....	81
13.5.8	Save As.....	81
13.6	Locals.....	82
13.6.1	Copy.....	82
13.6.2	Edit Value.....	82
13.6.3	Radix.....	82
13.7	Memory Mapping.....	83
13.7.1	Add.....	83
13.7.2	Edit.....	83
13.7.3	Reset.....	83
13.7.4	Help.....	83
13.8	Memory.....	84
13.8.1	Set Address.....	84
13.8.2	Load.....	84
13.8.3	Save.....	85
13.8.4	Test.....	85
13.8.5	Fill.....	85
13.8.6	Copy.....	85
13.8.7	Compare.....	85
13.8.8	Search.....	85
13.8.9	ASCII/Byte/Word/Long/Single Float/Double Float.....	85
13.9	Performance Analysis.....	86
13.9.1	Add Range.....	86
13.9.2	Edit Range.....	86
13.9.3	Delete Range.....	86

13.9.4	Reset Counts/Times .....	86
13.9.5	Delete All Ranges .....	86
13.9.6	Analysis Enabled.....	86
13.10	Registers .....	87
13.10.1	Copy .....	87
13.10.2	Edit.....	87
13.10.3	Toggle Bit .....	87
13.11	Source.....	87
13.11.1	Copy .....	89
13.11.2	Find .....	89
13.11.3	Set Address .....	89
13.11.4	Set Line .....	89
13.11.5	Go To Cursor .....	89
13.11.6	Set PC Here.....	89
13.11.7	Instant Watch .....	89
13.11.8	Add Watch .....	89
13.11.9	Go To Disassembly .....	90
13.12	System Status .....	90
13.12.1	Update .....	90
13.12.2	Copy .....	90
13.13	Trace.....	91
13.13.1	Find .....	91
13.13.2	Find Next.....	91
13.13.3	Filter .....	91
13.13.4	Acquisition.....	91
13.13.5	Halt.....	91
13.13.6	Restart .....	91
13.13.7	Snapshot.....	92
13.13.8	Clear .....	92
13.13.9	Save.....	92
13.13.10	View Source .....	92
13.13.11	Trim Source.....	92
13.14	Watch .....	92
13.14.1	Copy .....	93
13.14.2	Delete .....	93
13.14.3	Delete All .....	93
13.14.4	Add Watch .....	93
13.14.5	Edit Value .....	93
13.14.6	Radix .....	93
付録 A	システムモジュール.....	95
付録 B	コマンドラインフォーマット .....	97
付録 C	コマンドライン一覧表 .....	117
付録 D	GUI コマンド一覧.....	119
付録 E	I/O レジスタファイルのフォーマット.....	121
付録 F	シンボルファイルのフォーマット .....	125

## 図版目次

図 3-1	Select Sessionダイアログボックス	エラー! ブックマークが定義されていません。
図 3-2	Memory Mappingウィンドウ	エラー! ブックマークが定義されていません。
図 3-3	Edit Memory Mappingダイアログボックス	10
図 3-4	System Statusウィンドウ	11
図 3-5	Load Programダイアログボックス	12
図 3-6	Browse Program Fileダイアログボックス	12
図 3-7	ファイルタイプの選択	12
図 4-1	Sourceウィンドウ	15
図 4-2	Disassemblyウィンドウ	16
図 4-3	Assemblerダイアログボックス	16
図 4-4	Labelsウィンドウ	18
図 4-5	Labelダイアログボックス	18
図 4-6	Set Addressダイアログボックス	19
図 4-7	Findダイアログボックス	20
図 5-1	Open Memory Windowダイアログボックス	21
図 5-2	Memoryウィンドウ(バイト単位)	21
図 5-3	Set Addressダイアログボックス	23
図 5-4	Editダイアログボックス	24
図 5-5	Search Memoryダイアログボックス	24
図 5-6	Fill Memoryダイアログボックス	25
図 5-7	Copy Memoryダイアログボックス	26
図 5-8	Test Memoryダイアログボックス	27
図 5-9	Save Memoryダイアログボックス	27
図 5-10	Load Memoryダイアログボックス	28
図 5-11	Verify S-Record File with Memoryダイアログボックス	28
図 6-1	PCが示すアドレス値に対応する行の強調表示	29
図 6-2	Step Programダイアログボックス	32
図 7-1	プログラムブレークポイントの設定	33
図 7-2	Breakpointsウィンドウ	35
図 7-3	Run Programダイアログボックス	37
図 8-1	Tooltip Watch	39
図 8-2	Instant Watchダイアログボックス	40
図 8-3	Add Watchダイアログボックス	41
図 8-4	Watchウィンドウ	41
図 8-5	ウォッチ項目の拡張	42
図 8-6	Edit Valueダイアログボックス	42
図 8-7	Localsウィンドウ	43
図 8-8	Registersウィンドウ	44
図 8-9	ビットレジスタの拡張	45
図 8-10	Registerダイアログボックス	46
図 8-11	I/O Registersウィンドウ	47
図 8-12	I/Oレジスタ値変更ダイアログボックス	48
図 9-1	Overlayダイアログボックス(表示時)	49
図 9-2	Overlayダイアログボックス(アドレス範囲選択時)	49
図 9-3	Overlayダイアログボックス(優先セクショングループ選択時)	50
図 10-1	Select Functionダイアログボックス	51
図 11-1	ウィンドウの最小化	53
図 11-2	Disassemblyウィンドウのアイコン	53
図 11-3	整列前	54
図 11-4	整列後	54

☒ 11-5	ウィンドウの選択	56
☒ 11-6	Customizeダイアログボックス(Toolbarsシート)	57
☒ 11-7	Customizeダイアログボックス(Commandsシート)	58
☒ 11-8	Fontダイアログボックス	59
☒ 11-9	Customize File Filterダイアログボックス	60
☒ 11-10	セッション名の表示	61
☒ 11-11	HDI Options ( Session ) ダイアログボックス	62
☒ 11-12	HDIオプション(Confirmation)ダイアログボックス	63
☒ 11-13	HDIオプション(Viewing)ダイアログボックス	63
☒ 11-14	基数の設定	64
☒ 12-1	メニュー	65
☒ 13-1	Breakpointsウィンドウ	73
☒ 13-2	Command Lineウィンドウ	74
☒ 13-3	Disassemblyウィンドウ	75
☒ 13-4	I/O Registersウィンドウ	77
☒ 13-5	Labels ウィンドウ	78
☒ 13-6	Add Labelダイアログボックス	79
☒ 13-7	Edit Labelダイアログボックス	79
☒ 13-8	Find Label Containing ダイアログボックス	80
☒ 13-9	ラベル削除確認メッセージボックス	80
☒ 13-10	Confirming All Label Deletionメッセージボックス	80
☒ 13-11	Load Symbols ダイアログボックス	81
☒ 13-12	Localsウィンドウ	82
☒ 13-13	Memory Mappingウィンドウ	83
☒ 13-14	Memoryウィンドウ	84
☒ 13-15	Performance Analysisウィンドウ	86
☒ 13-16	Registersウィンドウ	87
☒ 13-17	Sourceウィンドウ	88
☒ 13-18	System Statusウィンドウ	90
☒ 13-19	Traceウィンドウ	91
☒ 13-20	Watchウィンドウ	92
☒ A-1	HDIシステムモジュール	95



## 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。

### 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 1 インサーキットエミュレータ、または評価ボードとともにHDIを使用する場合などには、デバッグプラットフォームのハードウェアの詳細について、別冊の『デバッグプラットフォームのマニュアル』を参照してください。

注 2 HDIはWindows®3.1では動作しません。



## 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されます。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

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
- アドレス参照 ('&') の指定は1レベルのみ可能です
- 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++)
&value	// アドレス参照指定(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  
ン情報が表示されます。

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。



### 3 デバッグの準備

本章では、デバッグプラットフォームを、プログラムをデバッグできる状態に設定するため、HDIで使用するすべての機能について説明します。

ここでは、デバッグプラットフォームの選択・構成方法、およびデバッグオブジェクトファイルのダウンロード方法について説明します。

#### 3.1 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。デバッグオプションを指定すれば、コンパイラ、リンケージエディタは、C/C++言語プログラムのデバッグに必要な全情報をアブソリュートファイルまたは管理情報ファイルに格納します。このことから、これらのファイルは通常、デバッグオブジェクトファイルと呼ばれます。

**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ケージエディタの両方で、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。

#### 3.2 デバッグプラットフォームの選択

HDIを起動すると、起動メッセージとメインウィンドウを表示します。その後、起動メッセージの表示をクリアし、Select Sessionダイアログボックスを表示します。

Select SessionダイアログボックスでCreate a new session onオプションを選んだ場合は、リストから、使用するデバッグプラットフォームを選び、[OK] ボタンをクリックしてください。

誤ったプラットフォームを選択してしまった場合は、[File->New Session...]メニューオプションをクリックすることにより、プラットフォームを選択し直すことが出来ます。

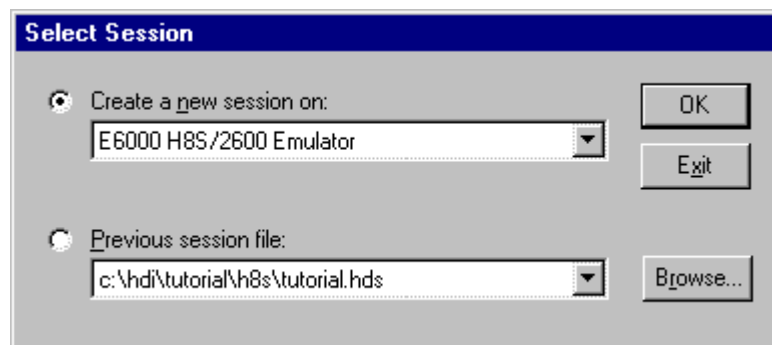


図 3.1 Select Sessionダイアログボックス

HDIはプラットフォームのプラグインモジュールをロードし、デバッグプラットフォームとの連動を開始します。ターゲットモジュールがロードされた

後、各種のハードウェアまたはデータ群を初期化します。初期化の進行にしたがって、ステータスバーにステータスメッセージを表示します。デバッグプラットフォームが正常に初期化を終了すると、HDIはステータスバー上に“Link up”を表示します。

### 3.3 デバッグプラットフォームの構成

プログラムをデバッグプラットフォームにロードする前に、デバッグプラットフォームの構成を、開発するユーザシステムに合わせて設定しなければなりません。設定しなければならない項目は、デバイスの種類、動作モード、クロック速度、およびメモリマップです。プログラムをデバッグプラットフォームにロードするには、ロード先に必ずメモリが必要なので、メモリマップの設定は特に重要です。

#### 3.3.1 設定

デバッグプラットフォームの構成を設定するには、[**Setup->Configure Platform...**]メニューオプションを選択します。Select Sessionダイアログボックスで選択したデバッグプラットフォームに対応する設定ダイアログボックスが表示されます。

注 デバッグプラットフォームで使用できる各機能についての詳細は、別冊の『デバッグプラットフォームのマニュアル』を参照してください。

#### 3.3.2 マッピング

ユーザシステムを正しく反映したデバッグ環境に設定するため、メモリマップを設定する必要があります。デバイスのアドレス空間内のどの領域が、RAM、ROM、オンチップレジスタ、またはメモリなしの領域であるかを認識する必要があります。

Debugging platform configurationダイアログボックスでデバイスタイプおよびモードを選択すると、HDIは自動的に、選択したデバイスおよびプロセッサが稼働するモードに合わせてマップを設定します。たとえば、ROM、RAMを内蔵したデバイスの場合は、デフォルトで、このデバイスのメモリマップ内でROM領域およびRAM領域が設定されます。

内蔵メモリのないデバイスを使用する場合や、内蔵メモリの代わりに、あるいは内蔵メモリに加えて外部メモリを備えたデバイスを使用する場合は、外部メモリがあることをデバッグプラットフォームに設定する必要があります。また、エミュレータを使用してデバッグする場合で、デバイス内部にも、ユーザシステムの外部にも存在しないメモリをアドレスマップに使用したい場合は、エミュレータから一部のエミュレーションメモリを、使用するアプリケーション用のアドレス空間にマップすることが可能です。

メモリマップの構成を編集するには[**Memory->Configure Map...**]メニューを選択します。Select Sessionダイアログボックスで選択したデバッグプラット



フォームにより、表示形態が異なります。デバッグプラットフォームにインサーキットエミュレータを使用した場合は、以下のように表示されます。

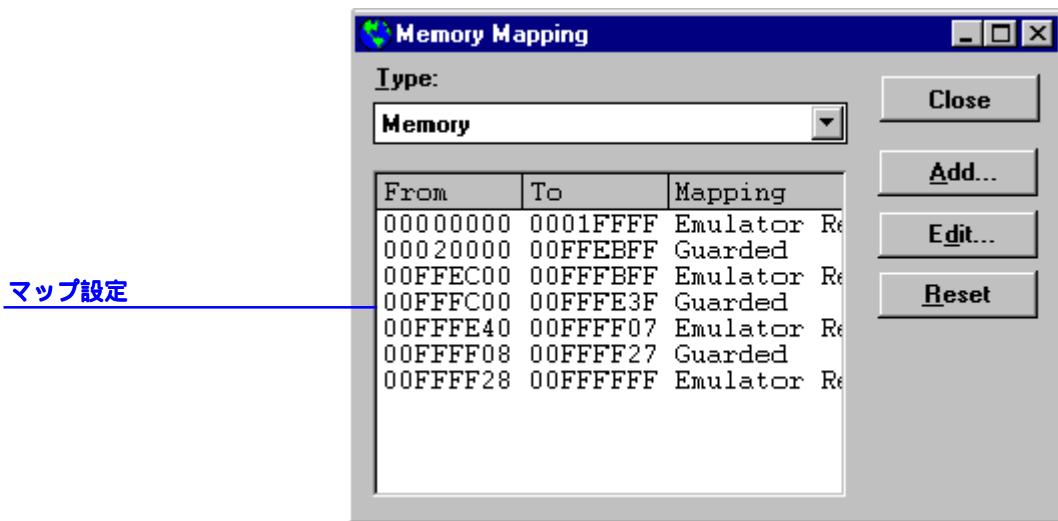


図 3.2 Memory Mappingウインドウ

「マップ設定」領域には、アドレス空間の現在のマッピング状況が表示されます。この領域には、アドレス空間に存在するすべてのアドレス範囲、およびそれらのアドレス範囲が設定されているメモリタイプがリスト表示されます。エミュレータに対して内部であるか外部であるか、およびアドレス範囲がもつアクセス制限(例：読み出し専用や保護(アクセス不可)など)がリスト表示されます。この領域には、HDIが自動的に設定した範囲とユーザ自身が設定または変更した範囲が含まれます。

メモリマップに関する補足情報はSystem StatusウインドウのMemoryシートに表示されます。「デバイス構成」領域には、Configure Platformダイアログボックスで選択したデバイスタイプとモード、およびオンチップメモリの制御設定に応じて自動的に構成された、デバイスのアドレス空間でのメモリ状況が表示されます。「システムリソース」領域には、システムに使用可能なリソースのマッピング状況が表示されます。たとえばエミュレータの場合、この領域には、エミュレータメモリを割り付けたアドレス範囲、および現在使用可能なエミュレータメモリが表示されます。

[Reset]ボタンをクリックすると、システムマップの設定内容が、現在のデバイスタイプおよびモード用のデフォルト設定に戻ります。

マップ設定を変更するには、対象の設定値を選択して、[Edit]ボタンをクリックするか、または対象のマップ設定行をダブルクリックします。

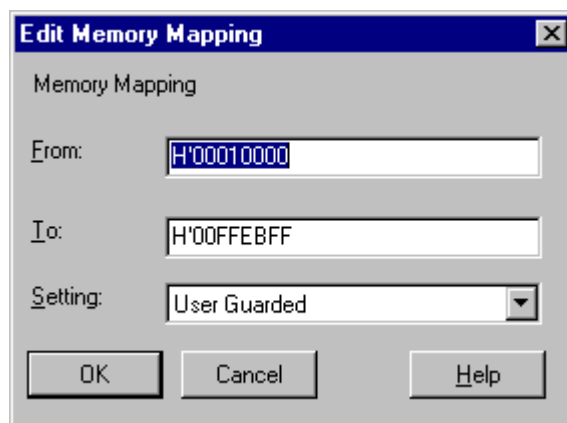


図 3.3 Edit Memory Mappingダイアログボックス

ダイアログボックスで、マップ範囲の開始アドレスと終了アドレス、およびメモリタイプを設定できます。

新規に領域を追加するには、[**A**dd]ボタンをクリックしてAdd Memory Mappingダイアログボックスを開きます(デフォルト値が未設定であることを除いてEdit Memory Mappingダイアログボックスと同じです)。マップ範囲の開始アドレスと終了アドレス、および新規領域のメモリタイプ設定を入力してください。新規領域が既存範囲の間にある場合、HDIは自動的に、新規領域の範囲を調整します。

注 マップ設定時、エミュレータによって設定できる最小のアドレス範囲が異なりますので、アドレス範囲が入力したアドレス値に一致しないことがあります。

### 3.3.3 ステータス

デバッグプラットフォームの構成や設定情報をチェックするには、[**V**iew->**S**tatus]メニューオプションを選択してSystem Statusウィンドウを開きます。

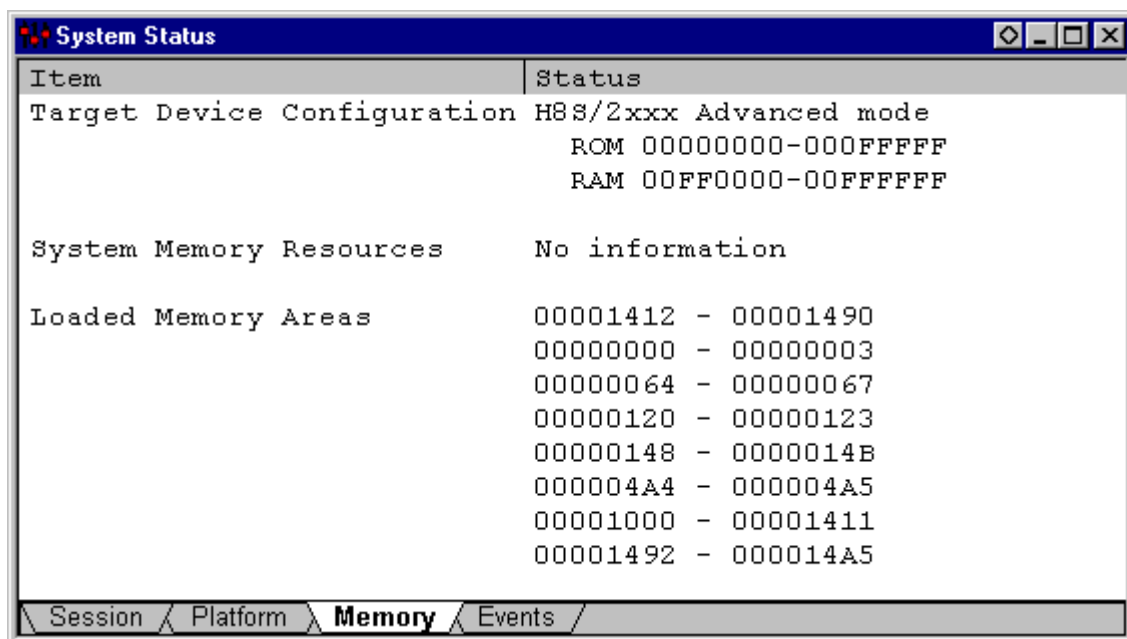


図 3.4 System Statusウインドウ

System Statusウインドウは、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、実行状態およびクロック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エリアなど、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。

ステータス情報を更新するためには、ポップアップメニューの[Update]メニューオプションを選択してください。

### 3.4 プログラムのダウンロード

プログラムをダウンロードするシステム上に、必要なサイズのメモリを確保したら、デバッグするプログラムをダウンロードしてください。[File->Load Program...]メニューオプションを選択すると、Load Programダイアログボックスが開きます。

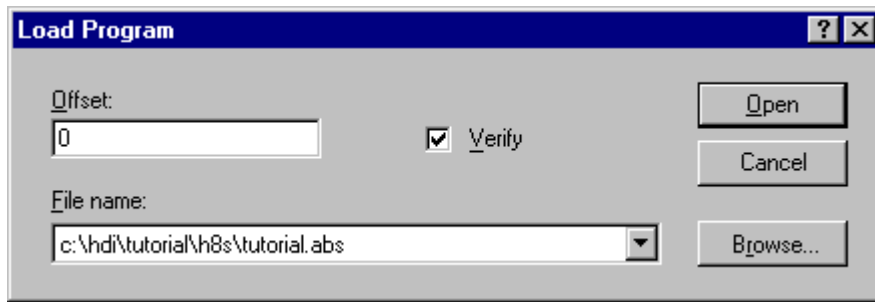


図 3.5 Load Programダイアログボックス

ダイアログボックスには、直近にロードされた4つのファイルのリストを持つコンボボックス、オフセットアドレス（S-Recordのようにオフセット・アドレスが使用できるオブジェクトフォーマットの場合）およびベリファイボタンがあります。ベリファイとは、プラットフォームへダウンロードされたデータが正しいかをリードバックしてチェックします。そのためダウンロード速度が遅くなるので、メモリやリンクファイルに問題があるかを調べるときにベリファイを行なうようにしてください。

コンボボックスにロードしたいファイルが含まれていない場合、編集エリアへファイル名を直接入力するか、Browse ボタンを使用してロードしたいファイルを探査することができます。

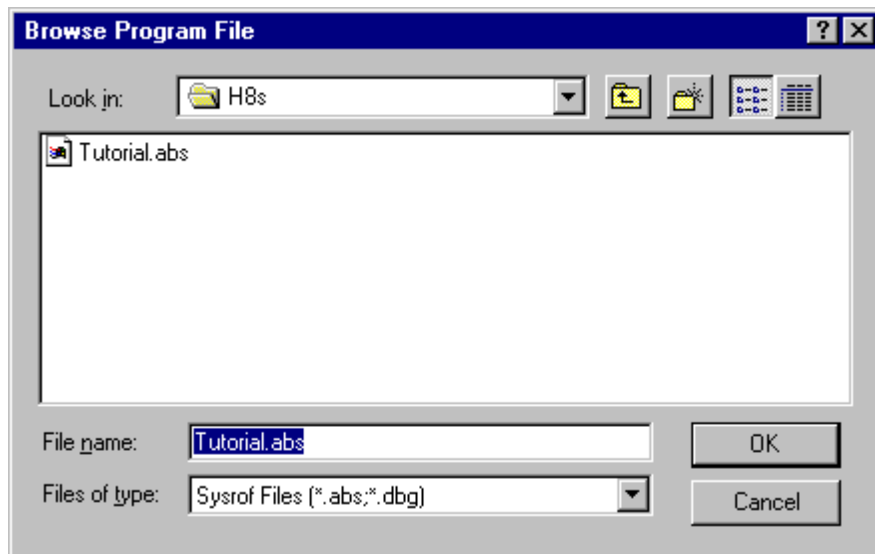


図 3.6 Browse Program Fileダイアログボックス

ロードするファイルを選択するには、まず Browse Program FileダイアログボックスのFiles of typeフィールド内をクリックして、表示するファイルのタイプを選び、そのファイルタイプをクリックします。

Files of typeフィールドについては「11.6 ファイルフィルタのカスタマイズ」を参照してください。

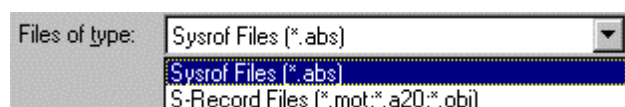


図 3.7 ファイルタイプの選択

ファイルタイプを選択すると、指定したファイルタイプのファイルがファイルリストに表示され、このリストからファイルを選択できます。ファイルリストの右に表示される、標準ウィンドウのファイルオープンダイアログボックスのコントロールを使用して、ディレクトリおよびドライブを変更することができます。また、File nameフィールドに直接ファイル名を入力することもできます。

[OK]ボタンは、Browserダイアログボックスで選択したプログラムのパスおよびファイル名をFile nameフィールドにセットしてLoad Programダイアログボックスに戻ります。

ファイルを選択した後、[Open]ボタンをクリックしてロードを開始します。ダウンロード中は、ステータスバーに進行状況を表示します。




## 4 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モニター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。

注 ブレークが成立時、Sourceウィンドウにはプログラムカウンター(PC)の位置を表示します。しかし、SYSROFオブジェクトフォーマットを使用したプロジェクトの場合、そのオリジナルのパスからソースファイルが移動されていると、自動的に見つけることができません。この場合は、現在のセッションで以前に使用されたパスリストを検索します。それでもソースファイルが見つからない場合、Source File Browserダイアログボックスをオープンするので、ソースファイルの存在するパスを指定してください。以降、ここで指定したパスは自動的に検索します。

### 4.1 コードの表示

#### 4.1.1 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。

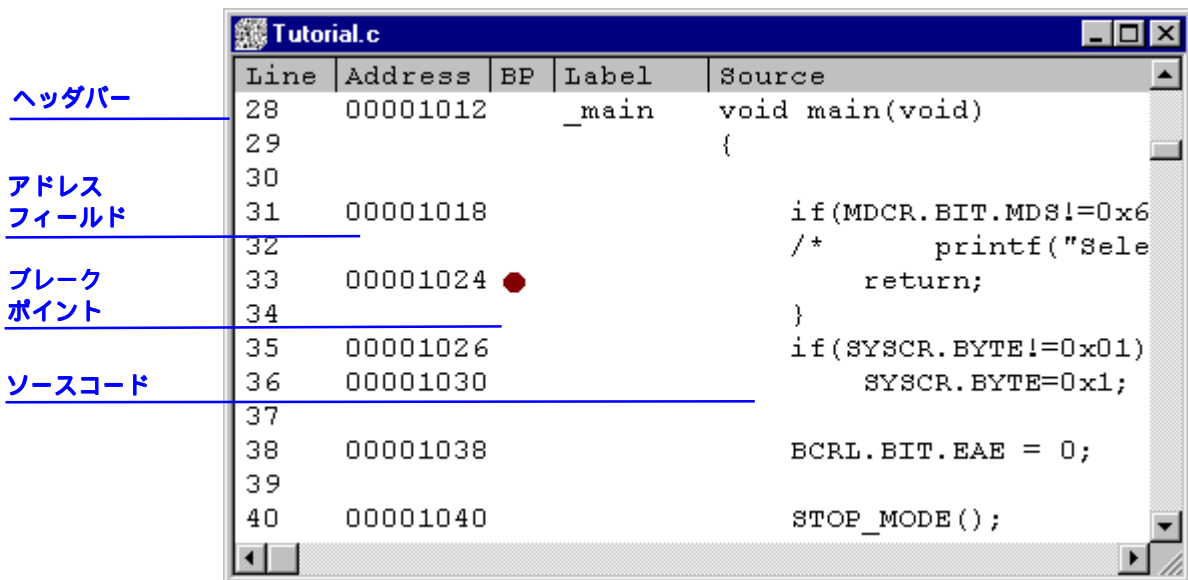




図 4.1 Sourceウィンドウ

Sourceウィンドウは、ヘッダ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、ヘッダ領域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。

#### 4.1.2 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、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)が表示されます。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。

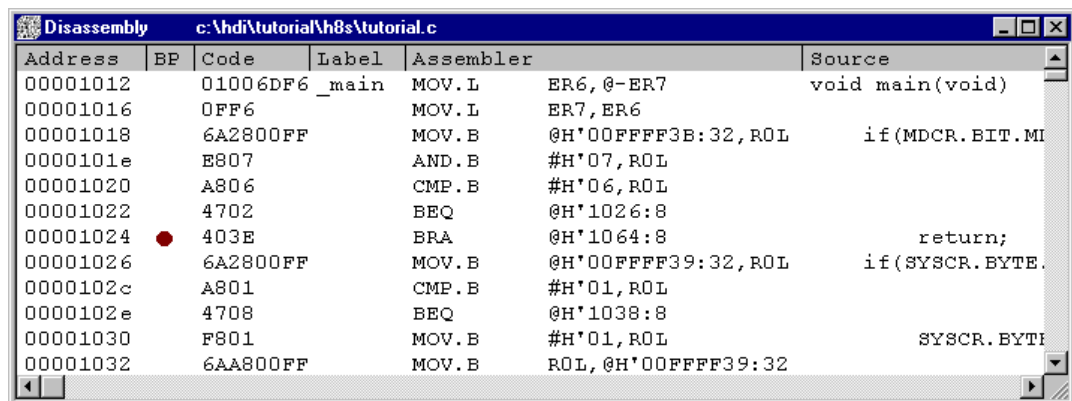


図 4.2 Disassemblyウィンドウ

#### 4.1.3 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。

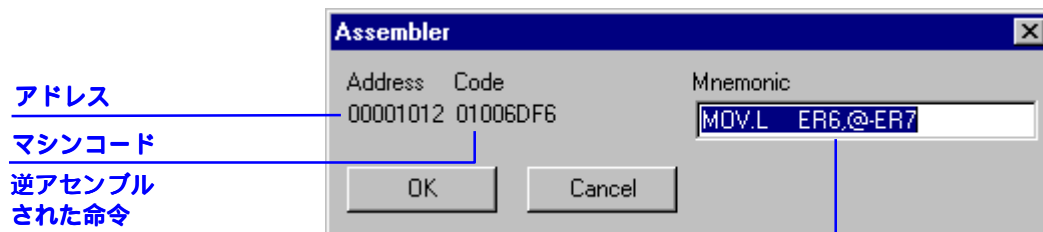


図 4.3 Assemblerダイアログボックス

アドレス、マシンコード、および逆アセンブルされた命令が表示されます。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。[OK]をクリックするか、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。[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されます。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が表示されます。ただし、この内容は、ソース表示とは一致しません。

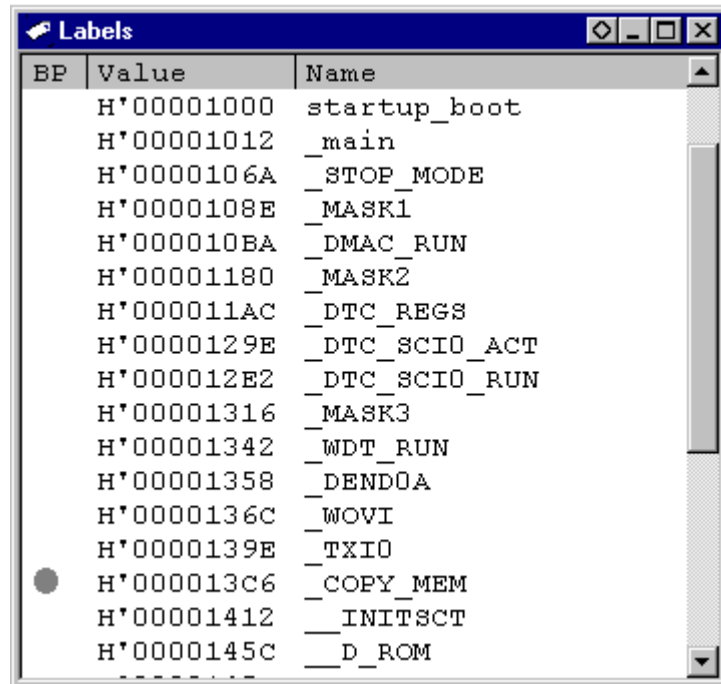
## 4.2 ラベルの表示

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フィールドに表示されます。また、Assemblerフィールドにオペランドの一部として表示されます。

- 注 命令のオペランドとラベルが一致する場合は、そのオペランドがラベルに置き換えられます。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が表示されます。
- 注 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。

### 4.2.1 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。



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 4.4 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。

#### 4.2.2 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。

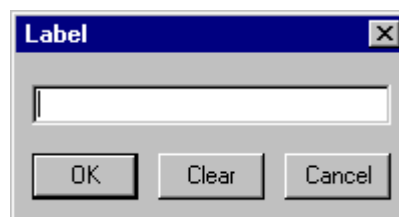


図 4.5 Labelダイアログボックス

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、「13.5.8 Save As ...」を参照してください。

### 4.3 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。

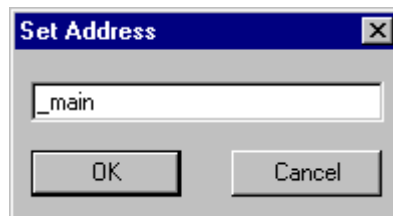


図 4.6 Set Addressダイアログボックス

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。詳細については、「10 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。

#### 4.3.1 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。

#### 4.4 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SHIFT + F10を押します。

Findダイアログボックスが表示されます。

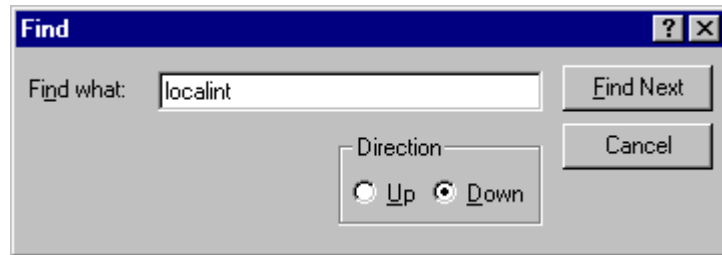



図 4.7 Findダイアログボックス

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。

## 5 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、およびベリファイ方法について述べます。

### 5.1 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。Open Memory Windowダイアログボックスが表示されます。

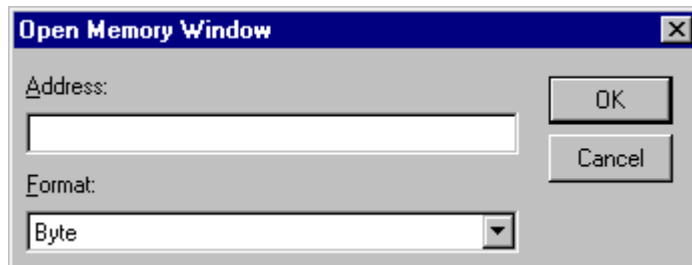


図 5.1 Open Memory Windowダイアログボックス

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。

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 5.2 Memoryウィンドウ(バイト単位)

Memoryウィンドウは、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せん。

- 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の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。

#### 5.1.1 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。

#### 5.1.2 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5-2)

#### 5.1.3 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。

#### 5.1.4 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。

#### 5.1.5 単精度浮動小数点形式でのメモリ表示

メモリを単精度浮動小数点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形式で表示されます。

#### 5.1.6 倍精度浮動小数点形式でのメモリ表示

メモリを倍精度浮動小数点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形式で表示されます。

#### 5.1.7 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。

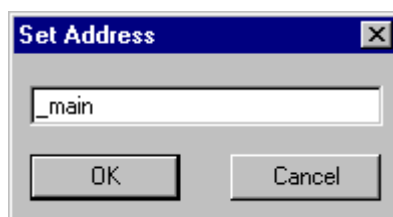


図 5.3 Set Addressダイアログボックス

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、「10 関数の設定」を参照してください。

## 5.2 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。

### 5.2.1 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。値は、0-9およびa-fの範囲内でなければなりません。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。

### 5.2.2 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。

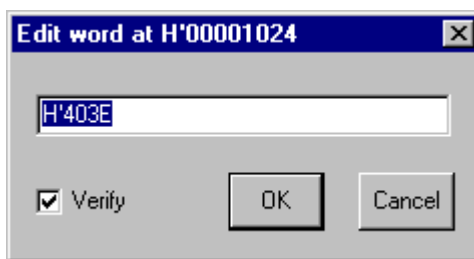


図 5.4 Editダイアログボックス

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。

### 5.2.3 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。

選択する範囲がMemoryウィンドウに表示されている範囲を超える、または、Memoryウィンドウに表示されていない場合、Memoryダイアログボックスに先頭番地およびバイト数を入力して、選択することができます。

## 5.3 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。

Search Memoryダイアログボックスが開きます。

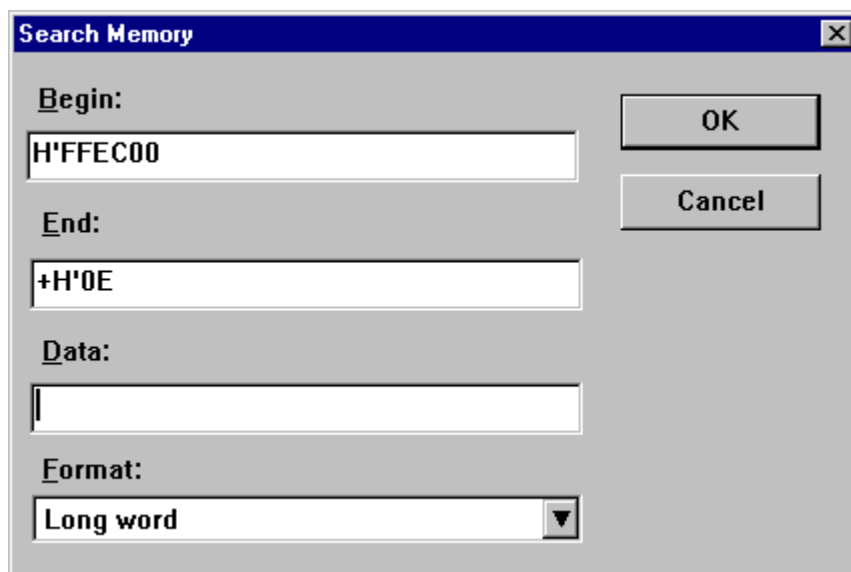


図 5.5 Search Memoryダイアログボックス



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、データが検出されたアドレスを示すメッセージがステータスバーに表示されます。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ステータスバーに表示されます。

## 5.4 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。

### 5.4.1 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。

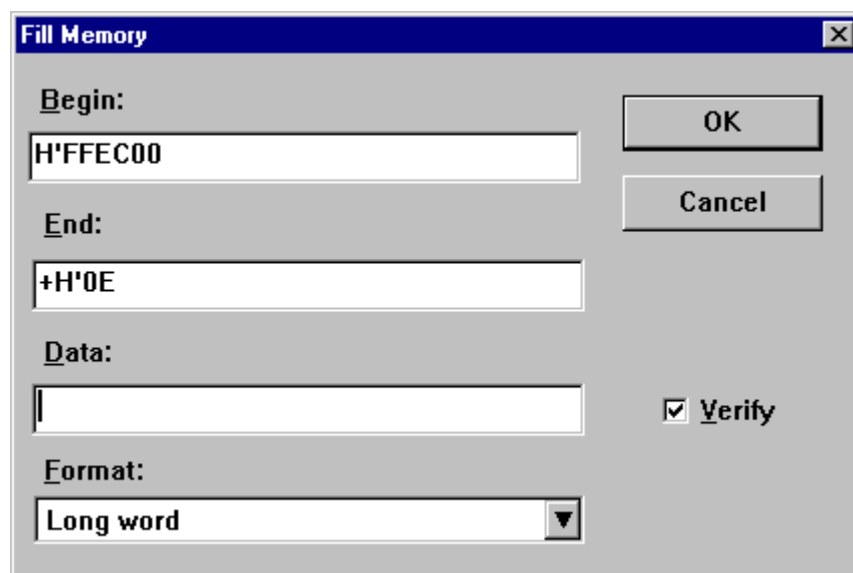


図 5.6 Fill Memoryダイアログボックス

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。

## 5.5 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(「5.2.3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。

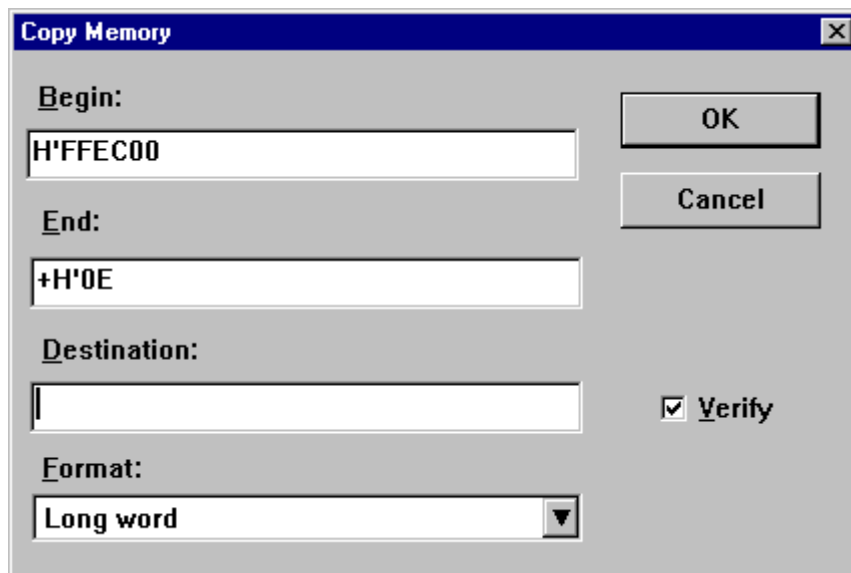


図 5.7 Copy Memoryダイアログボックス

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。

## 5.6 メモリ領域のテスト

**注** このテストは、デバッグプラットフォームに依存します。しかし、どのデバッグプラットフォームでも現在のメモリ内容は上書きされ、ユーザプログラムとデータは削除されます。

メモリテスト機能を使用して、アドレス空間内でメモリ領域をテストできます。メモリ範囲を選択して(「5.2.3メモリ範囲の選択」を参照)、ポップアップメニューの[Test]メニューオプションを選択します。Test Memoryダイアログボックスが開きます。

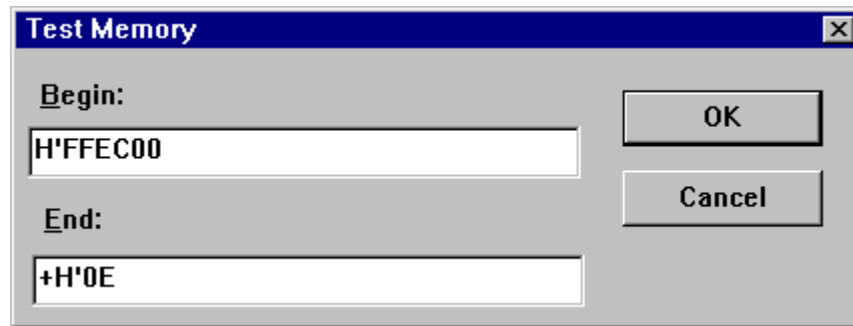


図 5.8 Test Memoryダイアログボックス

Beginおよび Endフィールドには、Memoryウィンドウで選択した開始アドレスと終了アドレスが設定されています。[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、HDIはメモリ範囲のテストを実行します。

## 5.7 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ダイアログボックスを開きます。

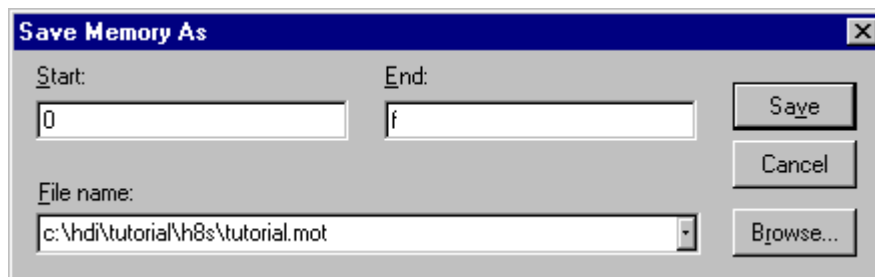


図 5.9 Save Memoryダイアログボックス

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File 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。

## 5.8 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。

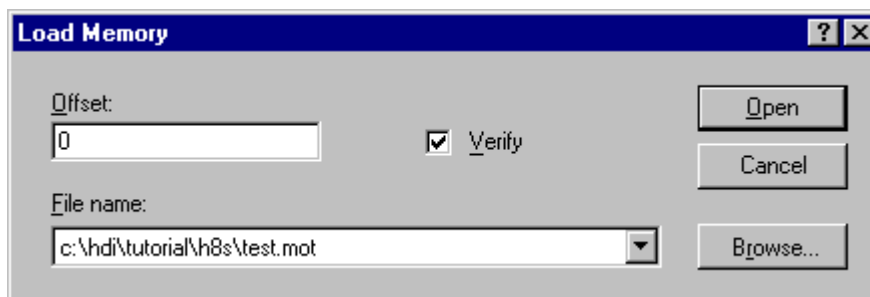


図 5.10 Load Memoryダイアログボックス

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。

ファイルからのメモリ領域のロードが完了するとSystem Statusウィンドウのメモリシートに表示します。

## 5.9 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。

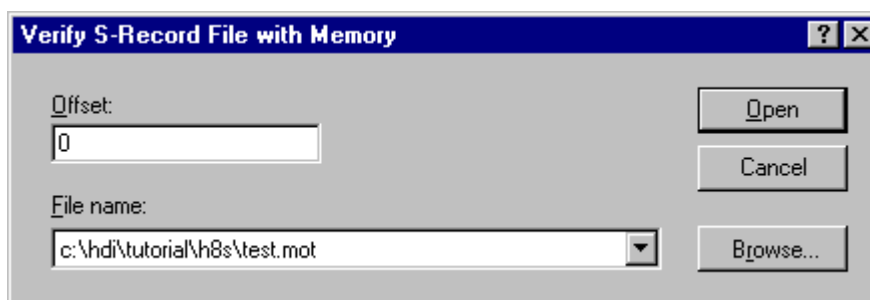



図 5.11 Verify S-Record File with Memoryダイアログボックス


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。

## 6 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。

### 6.1 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。

### 6.2 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。

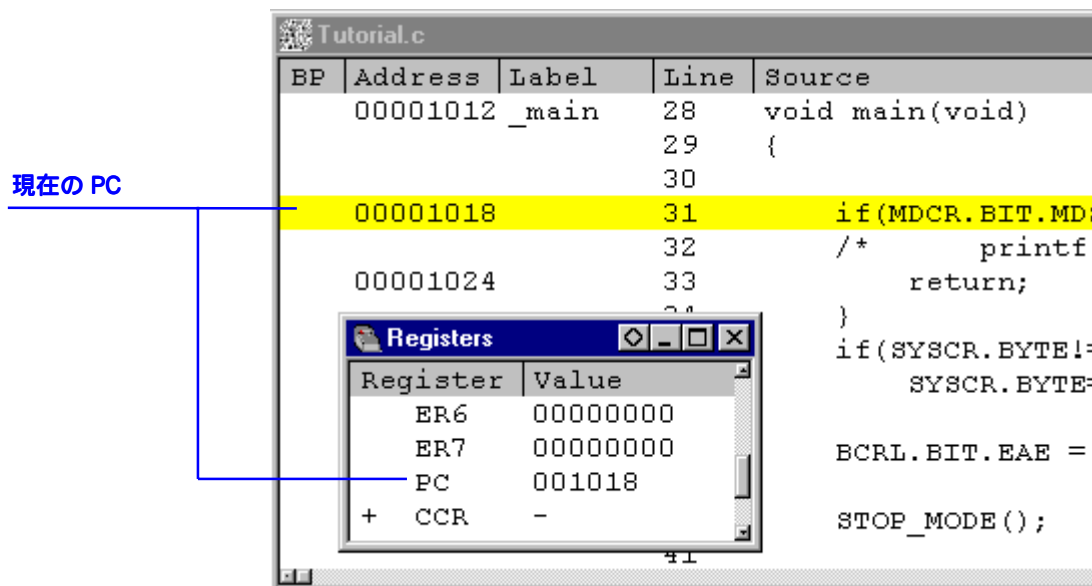



図 6.1 PCが示すアドレス値に対応する行の強調表示

現在のPCのアドレスから連続実行するには、Continue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。


### 6.3 カーソル位置まで実行

プログラムの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 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。

注 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。


### 6.4 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(「7.5 テンポラリブレークポイント」を参照)。


### 6.5 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。

### 6.5.1 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。

### 6.5.2 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。

## 6.6 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。

## 6.7 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が表示されます。



図 6.2 Step Programダイアログボックス

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。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。



## 7 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。

### 7.1 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 [STOP] (赤色のSTOPマーク)、プログラム停止時は無効状態 [STOP] (グレー表示のSTOPマーク) となります。プログラムを停止するには、ツールバーのHaltボタンをクリックするか、メニューで [Run->Halt] を選択します。

プログラムの実行が停止され、メッセージ “ Break = User Break ” 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。

### 7.2 標準ブレークポイント(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”という語と灰色の丸 ( ) が、その列に表示されます。

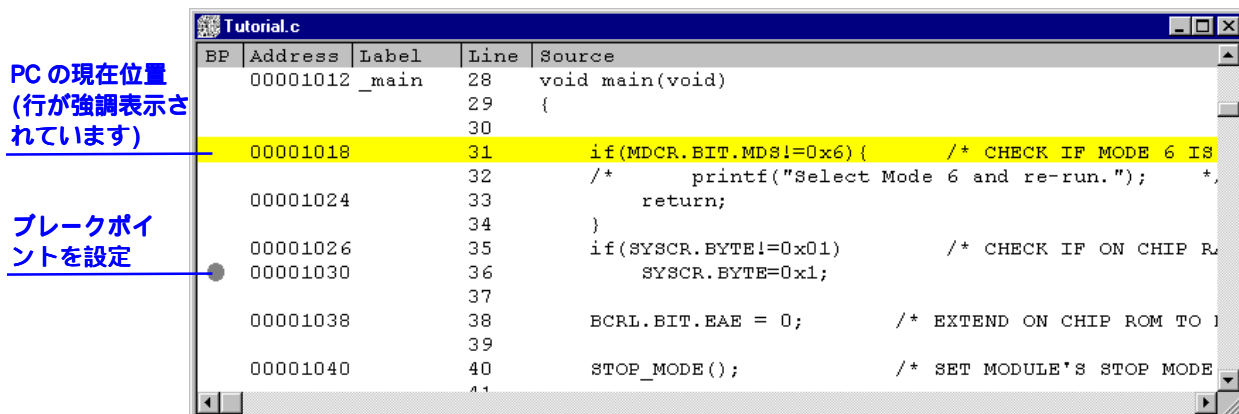


図 7.1 プログラムブレークポイントの設定

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 Break ”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に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。

### 7.2.1 標準ブレークポイントのタイプ切り替え


標準ブレークポイントは通常、PCブレークポイントですが、デバッグプラットフォームによっては、それ以外のタイプのブレークポイントを持つものがあります。プログラム(PC)ブレークポイントが設定されている行のBP列でマウスの左ボタンをダブルクリックするか、またはその行にカーソルを置いてF9キーを押すことにより、ブレークポイントのタイプを切り替える事ができます。利用可能な標準ブレークポイントタイプによって色付きの丸または略文字が、BP列に表示されます。

### 7.2.2 標準ブレークポイントの削除

BP列でマウスの右ボタンをクリックすると、現在のプラットフォームで有効な標準ブレークポイントタイプのリストを表示します。現在のブレークポイントタイプには、チェックマークがついています。ブレークポイントを削除するには[None]オプションを選択します。

標準ブレークポイントのタイプ切り替えの操作で、現在のプラットフォームに有効な標準ブレークポイントタイプに一通り切り替えた後、さらにもう一度繰り返すとブレークポイントが削除されます。

## 7.3 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。

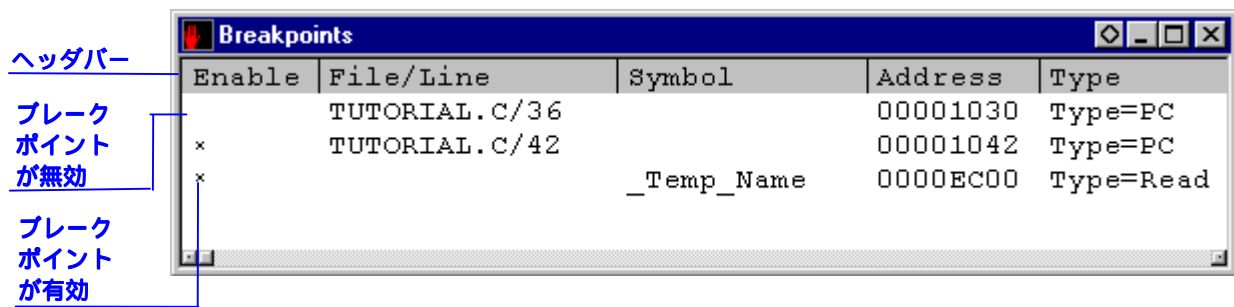


図 7.2 Breakpointsウィンドウ

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 $\text{+}$ 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。

### 7.3.1 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。

Breakpoint/Event Properties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。

**注** Breakpoint/Event Propertiesダイアログボックスは、選択したデバッグプラットフォームに固有のものです。このダイアログボックスの表示内容および動作は、デバッグプラットフォームで使用可能なブレークポイント機能により異なります。デバッグプラットフォーム固有のブレークポイントについての詳細は、別冊の『デバッグプラットフォームのマニュアル』を参照してください。

### 7.3.2 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。

Breakpoint/Event Properties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。

**注** Breakpoint/Event Propertiesダイアログボックスは、選択したデバッグプラットフォームに固有のものです。このダイアログボックスの表示内容および動作は、デバッグプラットフォームで使用可能なブレークポイント機能により異なります。デバッグプラットフォーム固有のブレークポイントについての詳細は、別冊の『デバッグプラットフォームのマニュアル』を参照してください。

### 7.3.3 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。

### 7.3.4 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。

## 7.4 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ないのでは、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。

### 7.4.1 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。

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になったことが示されます。

### 7.4.2 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。

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再度有効になったことが示されます。

## 7.5 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。

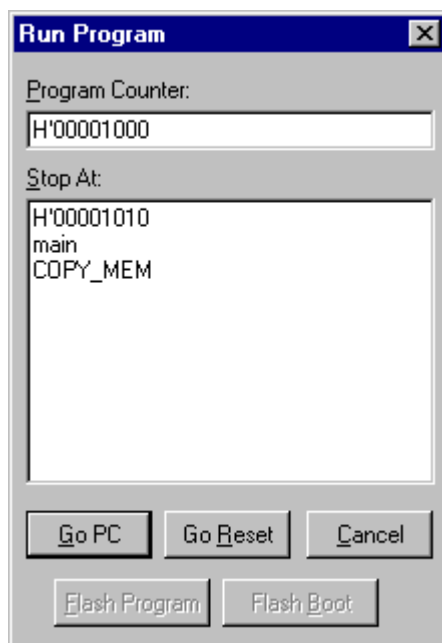


図 7.3 Run Programダイアログボックス

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、「10 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、[Reset Go]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[Reset Go]ボタンをクリックすると、再度設定されます。

## 7.6 ハードウェアブレークポイント(イベント)

注 ハードウェアブレークポイントは、選択したデバッグプラットフォームに固有のものです。各ブレークポイントの動作は、デバッグプラットフォームで使用可能なブレークポイント機能により異なります。デバッグプラットフォーム固有のブレークポイントについての詳細は、別冊の『デバッグプラットフォームのマニュアル』を参照してください。

## 8 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。

### 8.1 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 }
```

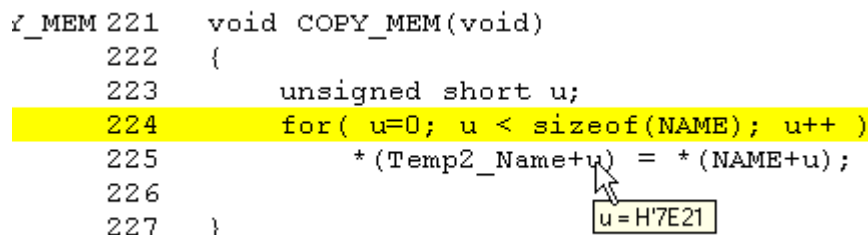


図 8.1 Tooltip Watch

### 8.2 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。

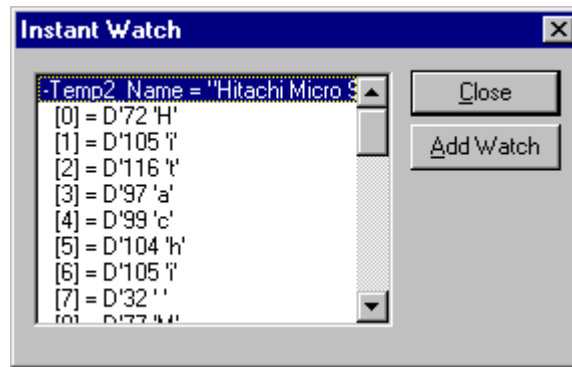



図 8.2 Instant Watchダイアログボックス

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。

### 8.3 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。

#### 8.3.1 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。

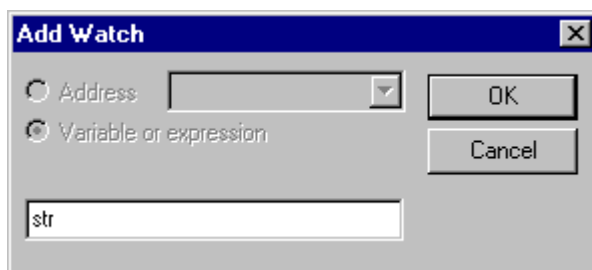


図 8.3 Add Watchダイアログボックス

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。

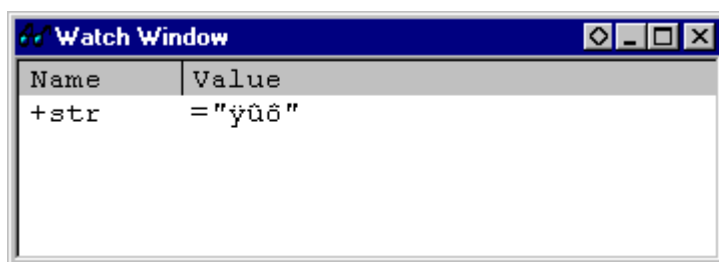


図 8.4 Watchウィンドウ

**注** 追加した変数が、現在の有効範囲外のローカル変数である場合、HDIはその変数をWatchウィンドウに追加しますが、その変数の値は空白になるか”?”を表示します。

#### 8.3.2 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。

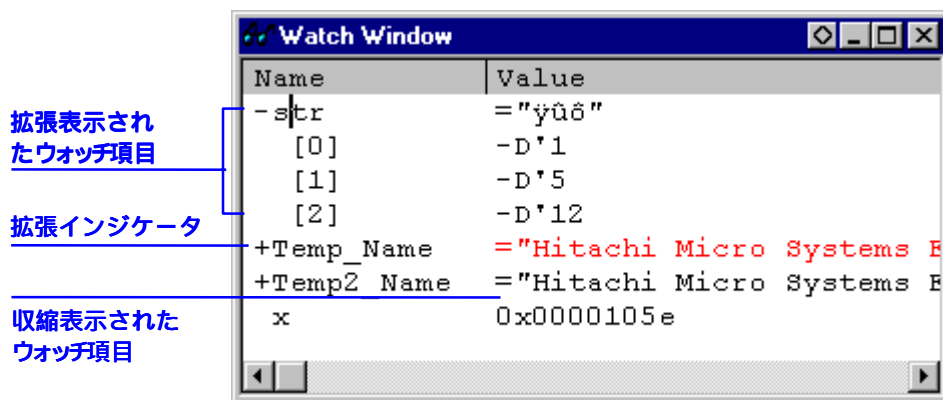


図 8.5 ウォッチ項目の拡張

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。

### 8.3.3 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。(メインメニューの[Setup->Radix]メニューオプションでは変更できません。)

### 8.3.4 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、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。

#### 🔹 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。

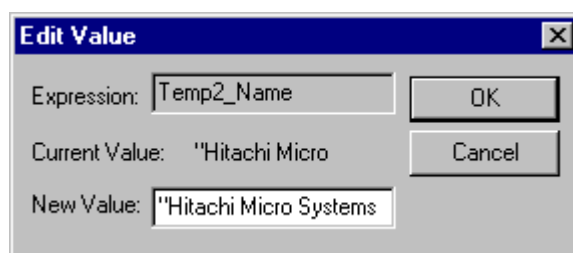


図 8.6 Edit Valueダイアログボックス

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。

### 8.3.5 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 Watch]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せん。詳細については、「11 ユーザインタフェースの構成」を参照してください。

## 8.4 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。

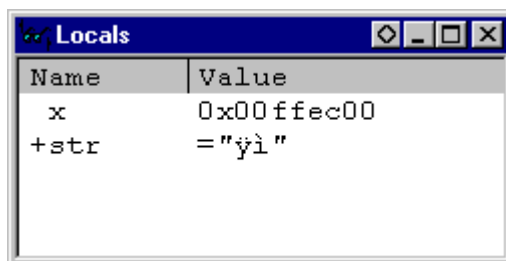


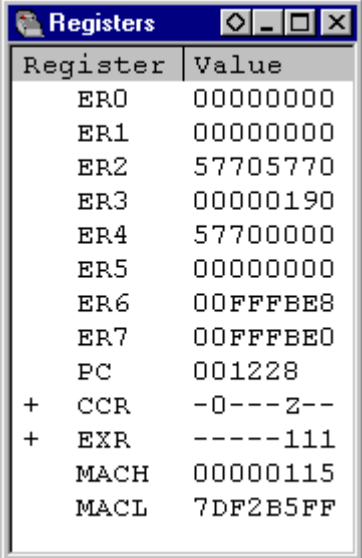
図 8.7 Localsウィンドウ

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。


## 8.5 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---Z--
+ EXR	-----111
MACH	00000115
MACL	7DF2B5FF

図 8.8 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Registersボタンをクリックします。Registersウィンドウが開き、汎用レジスタの値(16進数表記)がすべて表示されます。

### 8.5.1 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 8.9 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。

### 8.5.2 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、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の範囲内でなければなりません。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[Edit...]メニューオプションを選択する。

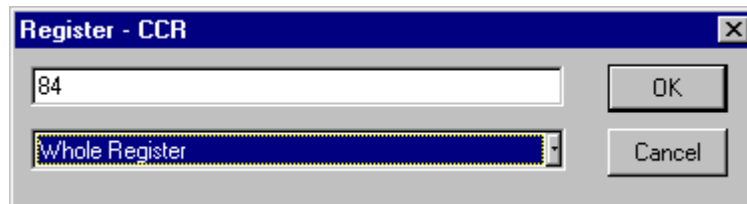


図 8.10 Registerダイアログボックス

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。

### 8.5.3 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)。

## 8.6 I/O レジスタの表示

CPUやROM/RAMに加えて、マイクロコンピュータには内蔵周辺モジュールも組み込まれています。周辺モジュールの数およびタイプはデバイスによって異なりますが、一般的なモジュールは、DMAコントローラ、シリアルインタフェース、A/Dコンバータ、統合タイマユニット、バスステートコントローラ、およびウォッチドッグタイマです。これらの内蔵周辺モジュールをプログラムで制御するには、マイクロコンピュータのアドレス空間へマップされたレジスタへアクセスします。

通常、これらの内蔵周辺レジスタの設定および使用は、埋込み型のマイクロコンピュータアプリケーションには非常に重要なため、これらのレジスタの内容を明確に見ることができると便利です。Memoryウィンドウでは、バイト、ワード、ロングワード、単精度浮動小数点、倍精度浮動小数点、またはASCIIの各値でしかメモリ内のデータを見ることができないため、HDIには、これらのレジスタを簡単に検証および設定できるI/O Registersウィンドウも備わっています。

### 8.6.1 I/O Registers ウィンドウのオープン

I/O Registersウィンドウを開くには、[View->I/O Area]メニューオプションを選択します。あるいは、ツールバーの I/O Registersボタンをクリックして開くこともできます。I/Oレジスタ情報は、内蔵周辺モジュールに対応して、モジュールごとに並べて表示されます。I/O Registersウィンドウを初めて開いた場合には、モジュール名のリストしか表示されません。

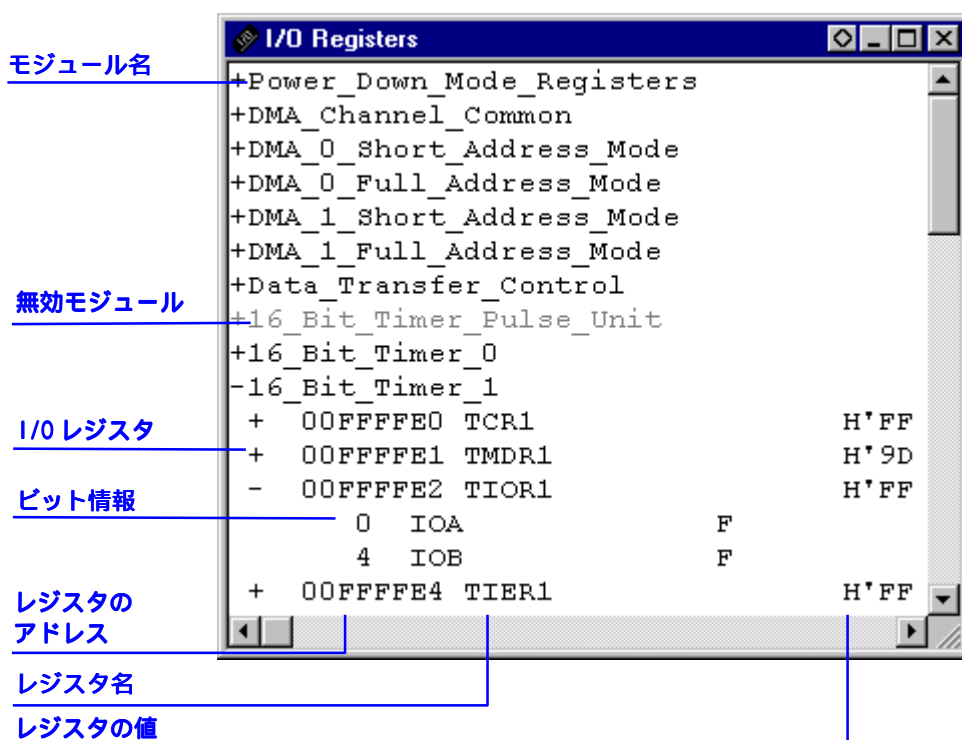


図 8.11 I/O Registersウィンドウ

### 8.6.2 I/O レジスタ表示の拡張

I/Oレジスタの名前、アドレス、および値を表示するには、そのモジュール名をダブルクリックするか、あるいはモジュール名を選択し、ENTERキーを押します。モジュール表示が拡張され、モジュール内の各レジスタの名前、アドレス、および値が表示されます。そのモジュール名を再度ダブルクリックする(またはENTERキーを押す)と、I/Oレジスタの表示が閉じます。また、同様の操作で、I/Oレジスタを拡張してビット単位に表示することができます。

各ビットの状態は以下のように色分けして表示します。

黒	通常の状態
赤	最後に更新された
グレー	無効になった（周辺機器のコントロールレジスタによって）

### 8.6.3 I/O レジスタの変更

I/Oレジスタの値を変更する場合、16進数で値を直接入力することができます。また、16進数以外で入力したい場合は、該当するレジスタをダブルクリックするか、**ENTER**キーを押して、レジスタの内容を変更するダイアログボックスを開きます。

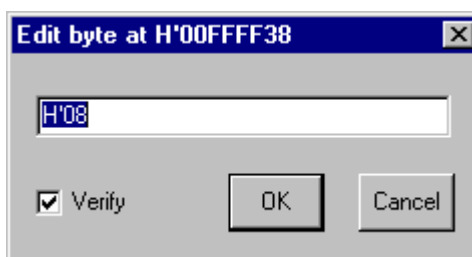


図 8.12 I/Oレジスタ値変更ダイアログボックス

新しい数値または式を入力後、[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。

- 注 エミュレータデバッグプラットフォームを使用する場合は、そのデバッグプラットフォームがI/Oレジスタからデータを読み取ることにより、ユーザプログラムの動作に影響がでる可能性があります。たとえば、データレジスタの読み取りにより、待ち状態の割り込みをキャンセルする可能性があります。データは、I/O Registersウィンドウで展開したI/Oモジュールからしか読みとられません(これによりレジスタの各値を表示されます)。つまり、I/Oモジュールを表示する必要がなければ、それらのI/Oモジュールを閉じている限り、問題は起こりません。プログラムに影響を与えるかどうかをチェックするには、I/O Registersウィンドウなしでプログラムを実行してください。また、I/O領域でMemoryウィンドウ(またはDisassemblyウィンドウ)を開いていると、同じ影響がでる可能性があることに注意してください。



## 9 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。

### 9.1 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。

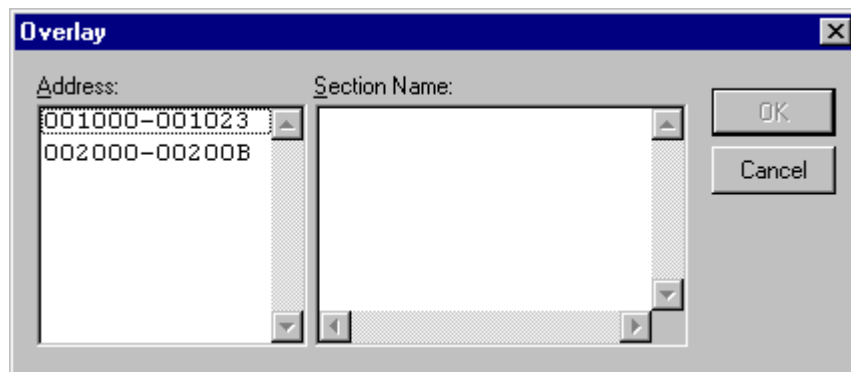


図 9.1 Overlayダイアログボックス（表示時）

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。

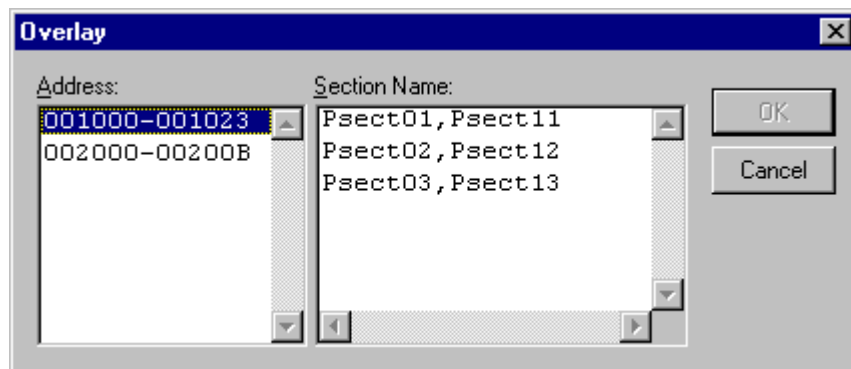


図 9.2 Overlayダイアログボックス（アドレス範囲選択時）

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。

## 9.2 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。

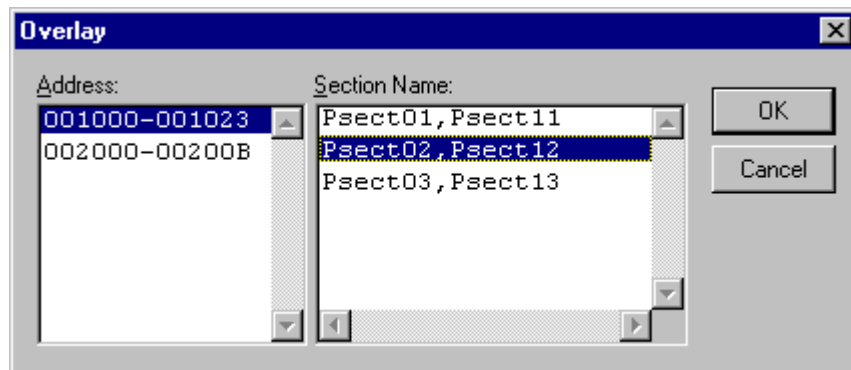


図 9.3 Overlayダイアログボックス（優先セクショングループ選択時）

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。

## 10 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。

### 10.1 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。

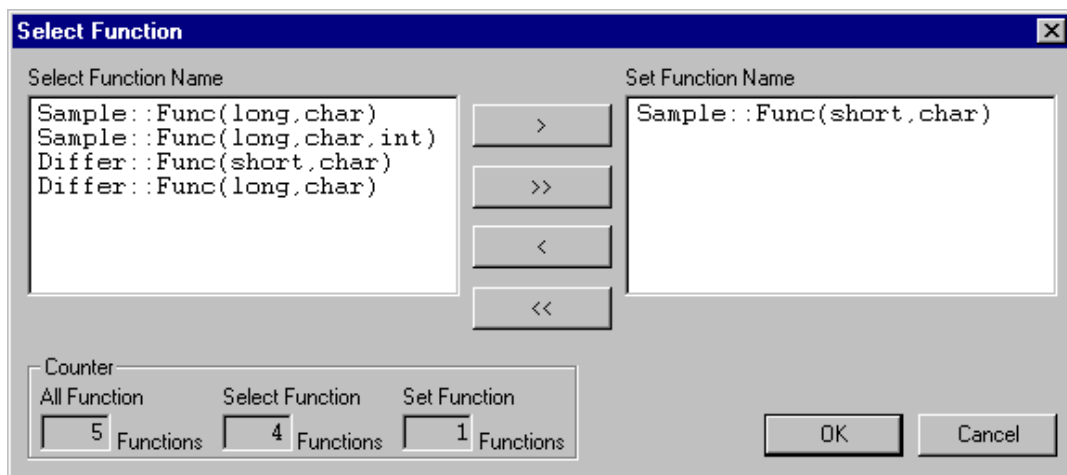


図 10.1 Select Functionダイアログボックス

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。

## 10.2 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のみ複数選択できます。

### 10.2.1 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。

### 10.2.2 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。

### 10.2.3 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。

## 11 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。

### 11.1 ウィンドウの配置

#### 11.1.1 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[] > **Minimize**]メニューオプションを選択します。

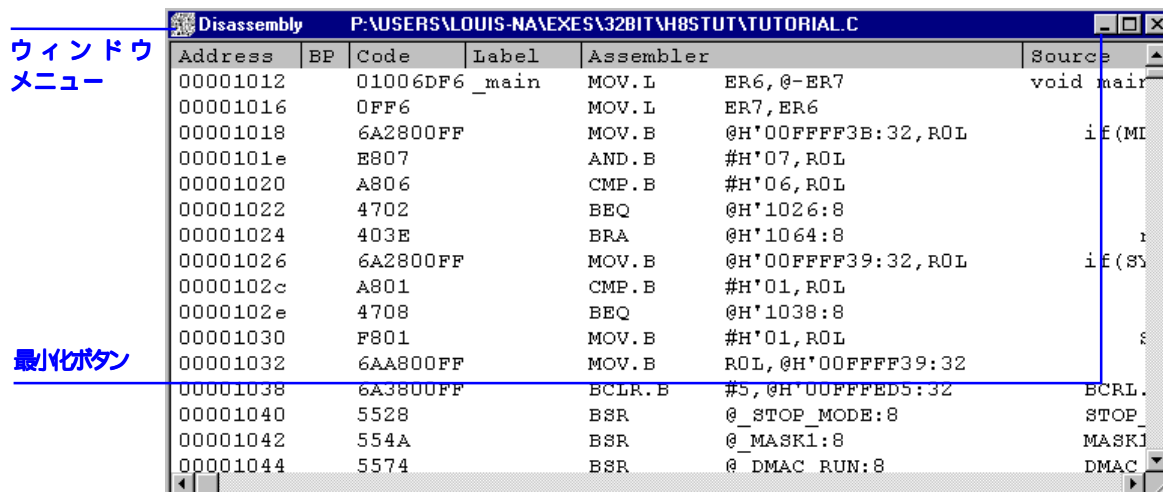


図 11.1 ウィンドウの最小化

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 11.2 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の[**R**estore]を選択します。

### 11.1.2 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。

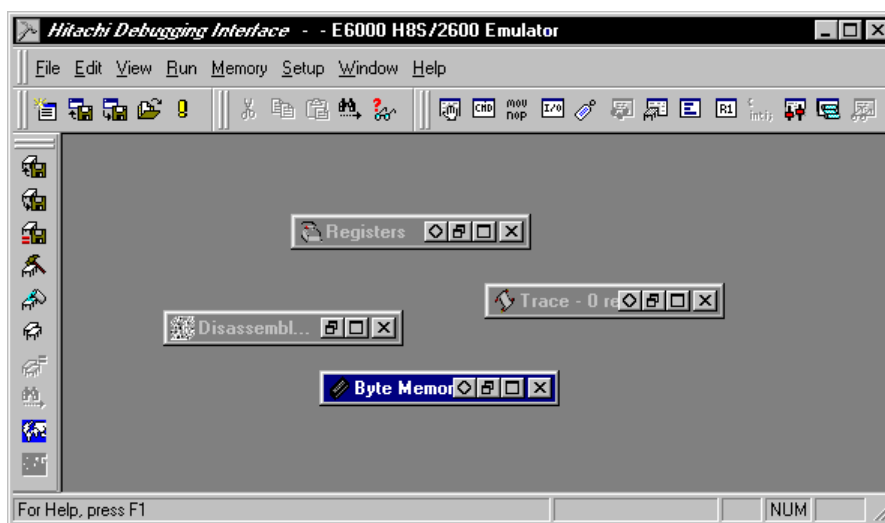


図 11.3 整列前

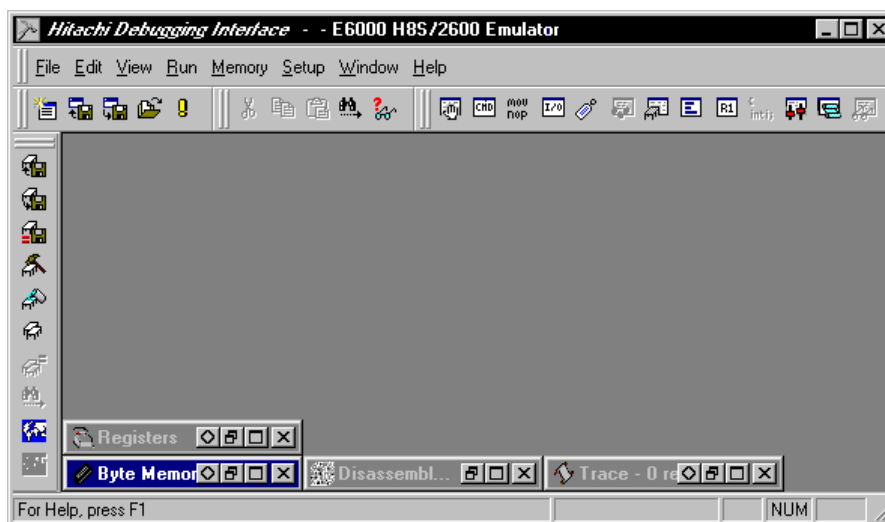


図 11.4 整列後

### 11.1.3 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。

### 11.1.4 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。

## 11.2 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。

### 11.2.1 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。

### 11.2.2 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。

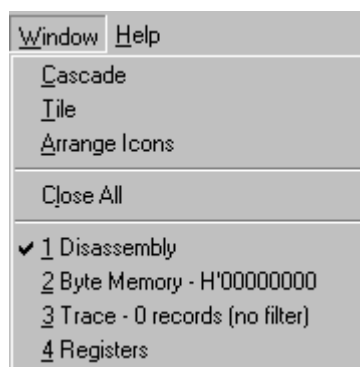


図 11.5 ウィンドウの選択

選択したウィンドウが手前に表示されます。上記の例でTraceウィンドウを選択しています。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。

### 11.3 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。ステータスバーの非表示にするには、[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。

### 11.4 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。

Toolbar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。



### 11.4.1 全体概要

Toolbarsシートで、表示するツールバーを選択します。

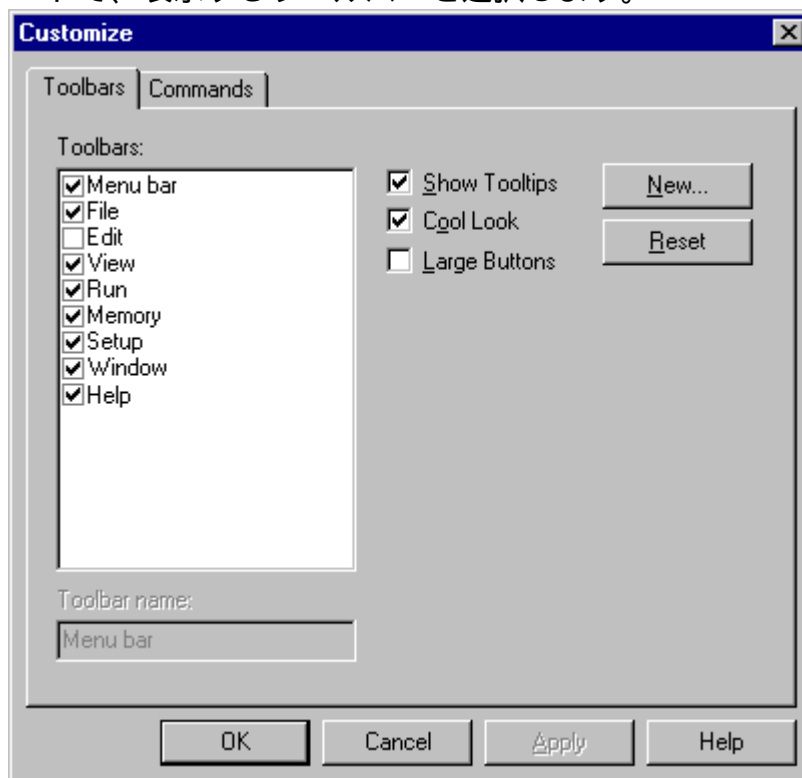


図 11.6 Customizeダイアログボックス(Toolbarsシート)

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。

### 11.4.2 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

マウスできません。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。

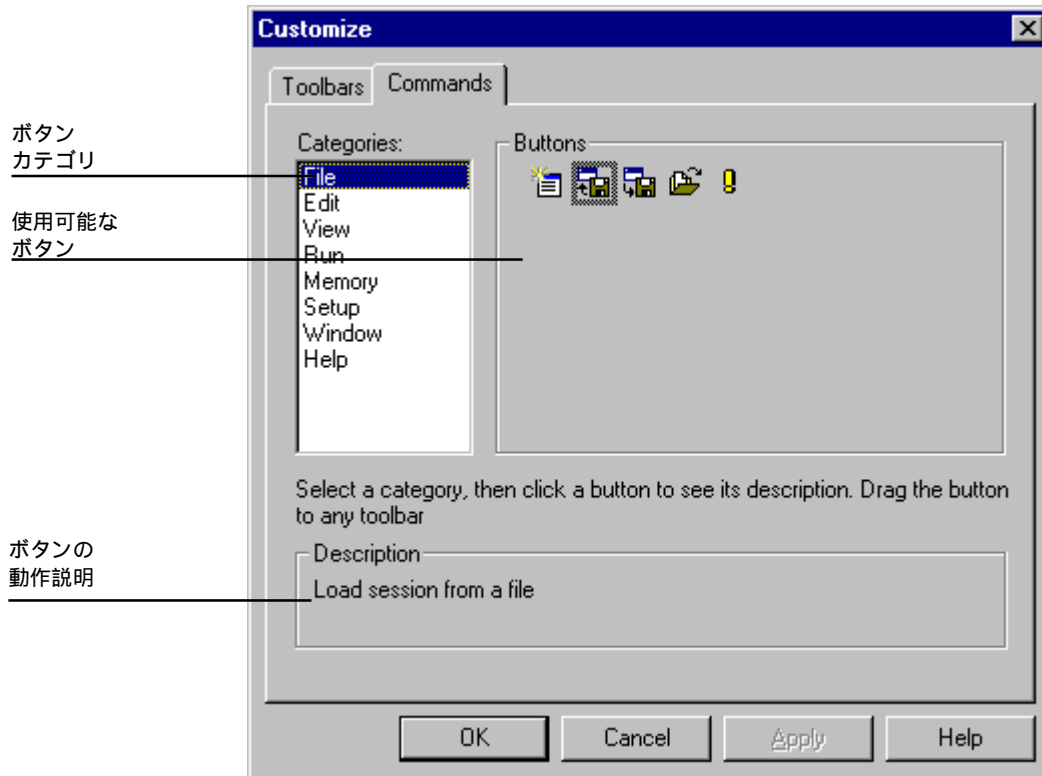


図 11.7 Customizeダイアログボックス(Commandsシート)

#### 11.4.3 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。リスト中のボタンボタン動作をクリックすると、Descriptionにボタンの動作についての説明が表示されます。

#### 11.4.4 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。

注 セパレータ(ボタンとボタンの間の空白)はサポートしていません。

#### 11.4.5 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。

#### 11.4.6 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。

### 11.5 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。

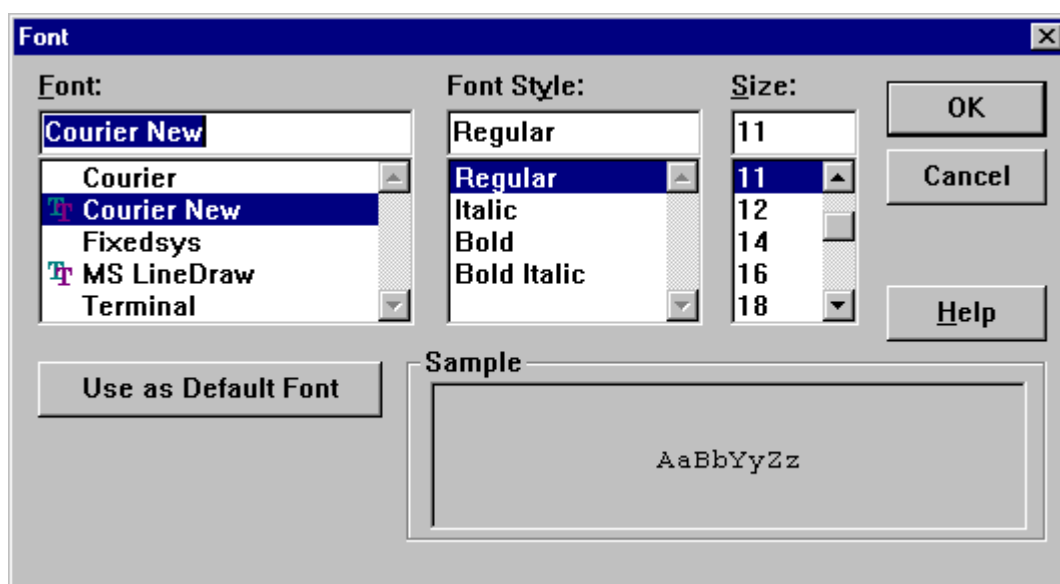


図 11.8 Fontダイアログボックス

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 **D**efault 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。

## 11.6 ファイルフィルターのカスタマイズ

Browserファイルダイアログで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。

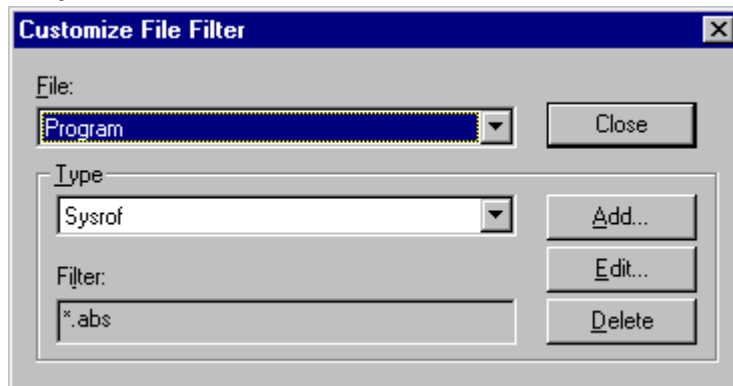


図 11.9 Customize File Filterダイアログボックス

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。

## 11.7 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をする場合は、[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が表示されます。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 11.10 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、「5.7 メモリ領域の保存」、「13.5.8 Save As ...」を参照してください。

## 11.8 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が表示されます。

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。

## 11.9 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が表示されます。

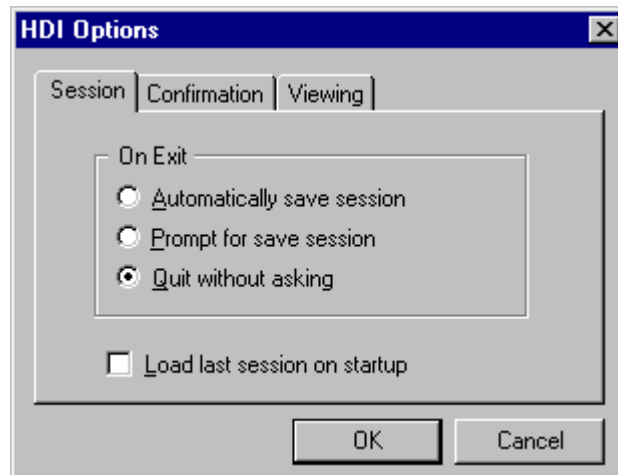


図 11.11 HDI Options ( Session ) ダイアログボックス

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。

- **A**utomatically save session - 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。
- **P**rompt 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。
- **Q**uit 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、**L**oad last session on startupチェックボックスをチェックします。

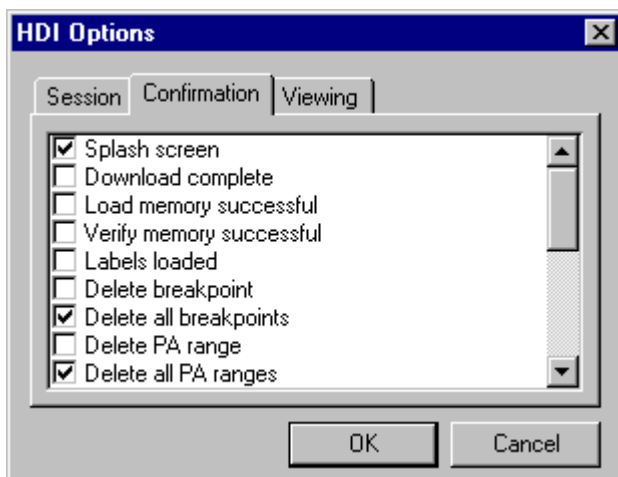


図 11.12 HDIオプション(Confirmation)ダイアログボックス

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。

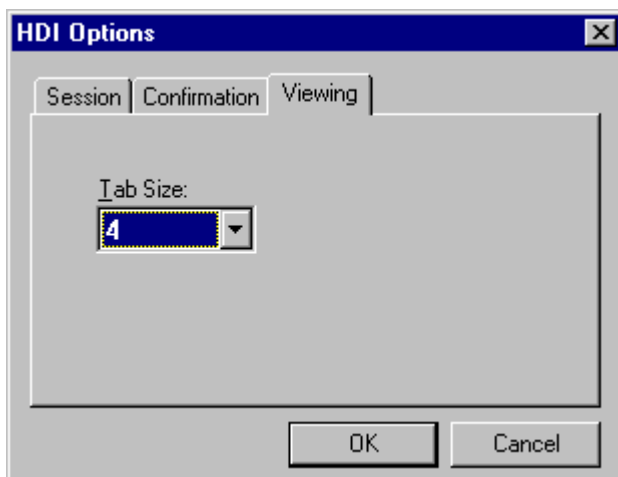


図 11.13 HDIオプション(Viewing)ダイアログボックス

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。

## 11.10 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が表示されます。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。

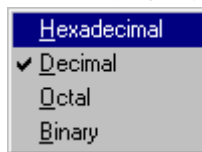


図 11.14 基数の設定



## 12 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。

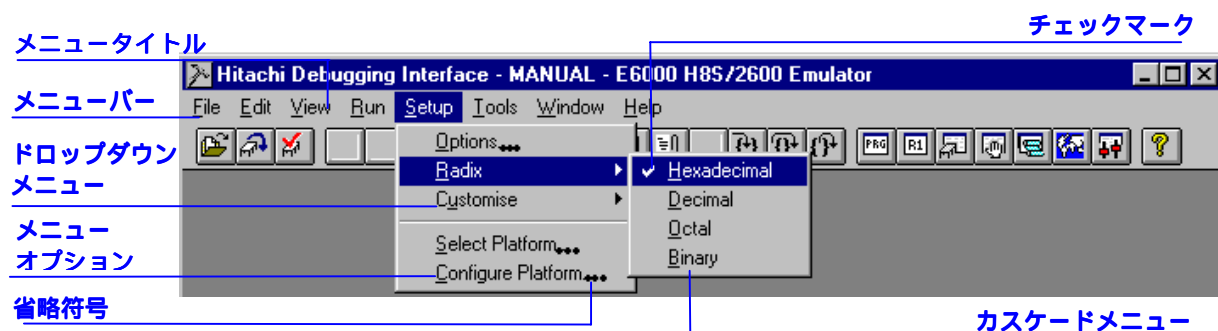


図 12.1 メニュー

チェックマークは、そのメニューオプションにより提供され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が表示されることを示します。

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。


### 12.1 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されます。


#### 12.1.1 New Session...

 新しいデバッグプラットフォームを選択できるようにSession ダイアログボックスを表示します。HDIプログラムファイルとして、同じディレクトリにターゲットDLLが1つしかない場合、このボタンはグレー表示されます。HDIがロードされたとき自動的にデバッグプラットフォームが選択されます。

#### 12.1.2 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Open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詳細、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。

#### 12.1.3 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。

#### 12.1.4 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

ームの詳細、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。

#### 12.1.5 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SYSROFあるいは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、シンボルもロードされます。

#### 12.1.6 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。「12.4.1 Reset CPU」も参照してください。

#### 12.1.7 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。「12.6.2 Options...」も参照してください。

### 12.2 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されます。

#### 12.2.1 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のみ使用できます(図では無効状態になっています)。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。

#### 12.2.2 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のみ使用できます(図では無効状態になっています)。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。

#### 12.2.3 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(図では無効状態になっています)。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。


#### 12.2.4 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(図では無効状態になっています)。

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。


#### 12.2.5 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されているすべての基数で表示します。


### 12.3 View

Viewメニューは、新規に各ウィンドウを開くために使用されます。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され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。


#### 12.3.1 Breakpoints

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


#### 12.3.2 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。


#### 12.3.3 Disassembly...

 表示させるテキストファイル名を入力できるようにOpenダイアログボックスを表示します。


#### 12.3.4 I/O Area

 I/O Registersウィンドウを開きます。ターゲットの内蔵入出力機能(たとえば割り込みコントローラ)を制御することができます。


#### 12.3.5 Labels

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

#### 12.3.6 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ければ、ウィンドウは空白となります。

#### 12.3.7 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。

### 12.3.8 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。

### 12.3.9 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。

### 12.3.10 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"というメッセージが表示されます。

### 12.3.11 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。

### 12.3.12 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。

### 12.3.13 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。

## 12.4 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されます。

### 12.4.1 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 12.1.6 Initializeを参照)。

### 12.4.2 Go



現在のPCからユーザプログラムを実行します。

### 12.4.3 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。

### 12.4.4 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。

#### 12.4.5 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。

#### 12.4.6 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。

#### 12.4.7 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(12.4.10 Step...参照)。サブルーチン呼び出した場合は、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。

#### 12.4.8 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す場合は、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設定されるまでユーザプログラムが実行されます。

#### 12.4.9 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。

#### 12.4.10 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。

#### 12.4.11 Halt



ユーザプログラムの実行を停止します。

### 12.5 Memory

Memoryメニューは、プログラムがアクセスするメモリの設定に使われます。

#### 12.5.1 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。

#### 12.5.2 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 Fileダイアログボックスを表示します。


#### 12.5.3 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 Fileダイアログボックスを表示しま

す。Memoryウィンドウ中の反転表示されたメモリブロックは、ダイアログボックスが表示されたときに自動的に設定される開始、終了アドレスです。


#### 12.5.4 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。


#### 12.5.5 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。


#### 12.5.6 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。


#### 12.5.7 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、Findオプションと同様に設定されます。


#### 12.5.8 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。Memoryウィンドウ中の反転表示されたメモリブロックは、ダイアログボックスが表示されたときに自動的に設定される開始、終了アドレスです。

#### 12.5.9 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、(サポートされていれば)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。

#### 12.5.10 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。




## 12.6 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されます。


### 12.6.1 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が表示されます。

### 12.6.2 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。

### 12.6.3 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。

### 12.6.4 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。


### 12.6.5 Configure Platform...

 set-upダイアログボックスを表示します。新しいデバッグプラットフォームを選択できるようにします。デバッグプラットフォームのユーザーズマニュアルには、ダイアログボックスの指定可能なオプションについてより詳細に記述してあるので参照してください。


## 12.7 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で使用されます。以下のメニューオプションは常に表示されています。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されます。一番手前に表示されているウィンドウにはチェックマークがついています。


### 12.7.1 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。

### 12.7.2 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。

### 12.7.3 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。

### 12.7.4 Close All

すべてのウィンドウをクローズします。

## 12.8 Help

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

### 12.8.1 Index

メインヘルプファイルのインデックスを開きます。

### 12.8.2 Using Help

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

### 12.8.3 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。

### 12.8.4 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。



## 13 ウィンドウ

本章では、各ウィンドウの種類と、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。

### 13.1 Breakpoints

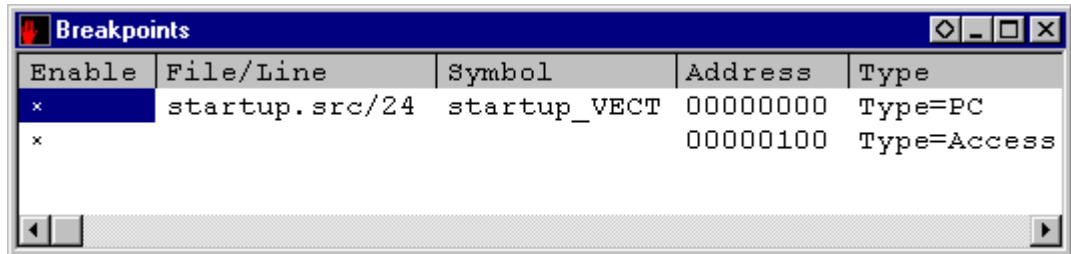


図 13.1 Breakpointsウィンドウ

現在設定されているブレークポイントの表示や制御を行い、ハードウェアのブレークポイントリソースを表示するウィンドウです。サポートされているブレークポイントの種類とリソースについては、別冊の『デバッグプラットフォームのマニュアル』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が表示されます。このメニューには以下のオプションが含まれています。

#### 13.1.1 Add...

Breakpoint/Event Propertiesダイアログボックスを表示して、新しいブレークポイントを入力します。このダイアログボックスは、デバッグプラットフォームに依存します。

#### 13.1.2 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。Breakpoint/Event Propertiesダイアログボックスを表示して、既存のブレークポイントの設定を変更できます。このダイアログボックスは、デバッグプラットフォームに依存します。

#### 13.1.3 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(13.1.5 Disable/Enable参照)。

#### 13.1.4 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。

### 13.1.5 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'x'が表示されます)。

### 13.1.6 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。

## 13.2 Command Line

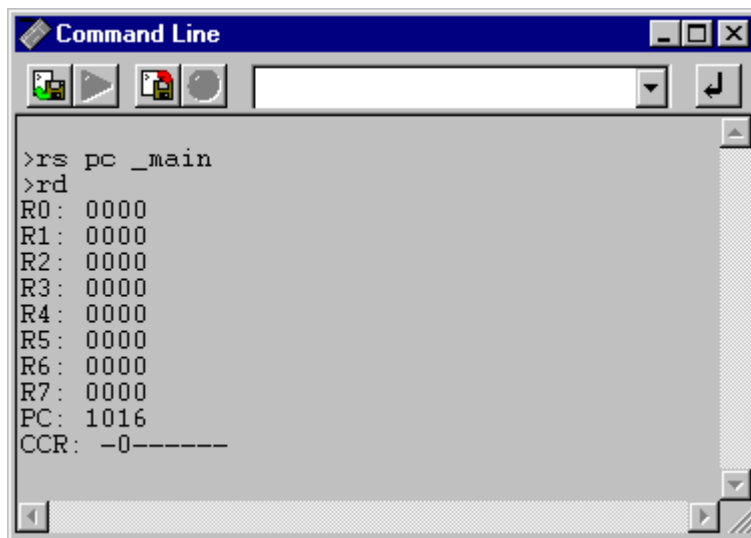



図 13-2 Command Lineウィンドウ

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。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。


### 13.2.1 Run Batch File

 Run 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)を入力できます。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。


## 13.2.2 Play

 最後に実行され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され、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されます。

## 13.2.3 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、ログファイル名がウィンドウ上に表示されます。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。

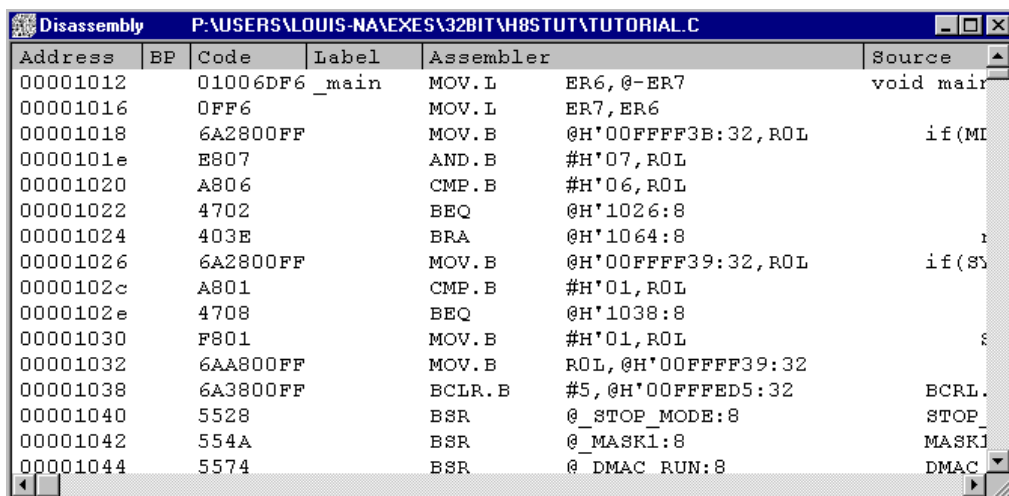
## 13.2.4 Logging

 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、メニューテキストの左にチェックマークが付きます(かつボタンが無効状態になります。)。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されます。

## 13.3 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。

アセンブラフォーマットは他の2つ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。対応するソースファイルがない場合には、“Code”とウィンドウタイトルに表示します。



Address	BP	Code	Label	Assembler	Source
00001012		01006DF6	_main	MOV.L ER6, @-ER7	void main
00001016		0FF6		MOV.L ER7, ER6	
00001018		6A2800FF		MOV.B @H'00FFFF3B:32, R0L	if(MI
0000101e		E807		AND.B #H'07, R0L	
00001020		A806		CMP.B #H'06, R0L	
00001022		4702		BEQ @H'1026:8	
00001024		403E		BRA @H'1064:8	
00001026		6A2800FF		MOV.B @H'00FFFF39:32, R0L	if(S)
0000102c		A801		CMP.B #H'01, R0L	
0000102e		4708		BEQ @H'1038:8	
00001030		F801		MOV.B #H'01, R0L	
00001032		6AA800FF		MOV.B R0L, @H'00FFFF39:32	
00001038		6A3800FF		BCLR.B #5, @H'00FFED5:32	BCLR.
00001040		5528		BSR @_STOP_MODE:8	STOP
00001042		554A		BSR @_MASK1:8	MASK
00001044		5574		BSR @ DMAC RUN:8	DMAC

図 13.3 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、ソースファイル中にある場合は、新しいウィンドウをオープンし、カーソルが該当箇所をさします。もしくは、ソースプログラムの該当箇所を表示します。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オープンするので、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箇所を表示します。

BP列で右クリックすると現在サポートされ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で表示されます。

### 13.3.1 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。

### 13.3.2 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。

### 13.3.3 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されます。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されます。

### 13.3.4 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。

### 13.3.5 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。ソース行が有効なときのみ機能します。

### 13.3.6 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されます。ソース行が有効なときのみ機能します。

## 13.4 I/O Registers

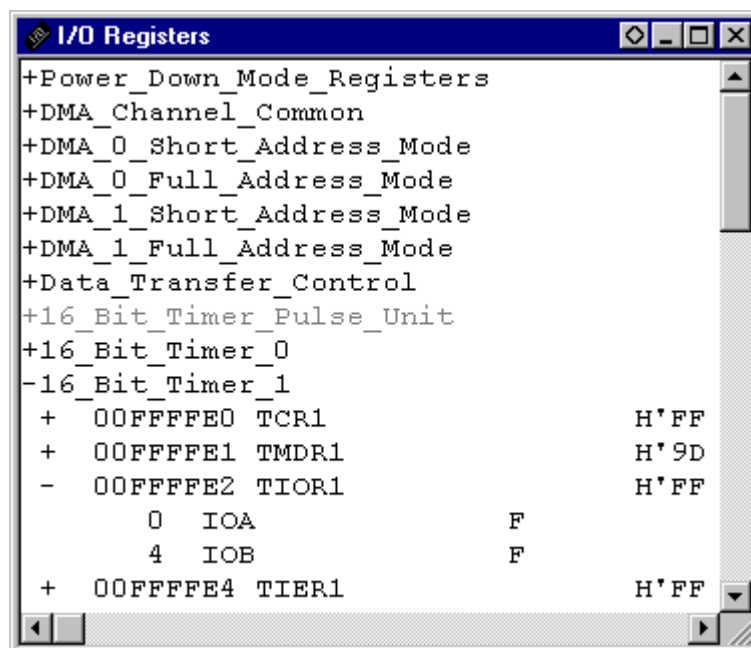


図 13.4 I/O Registersウィンドウ


ユーザハードウェアシステムのオンチップ周辺機器の参照、設定ができます。周辺機器はモジュールによって構成されており、表示される周辺機器の

詳細情報はリスト表示され、 '+'記号は変数名をダブルクリックすれば情報を拡張できることを、 '-'記号は情報を圧縮できることを示します。

'+'記号および '-'記号をダブルクリックするか、 '+'キーおよび '-'キーを入力するとレジスタの情報を拡張 / 収縮表示します。

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が表示されます。このメニューには以下のオプションが含まれています。

#### 13.4.1 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。

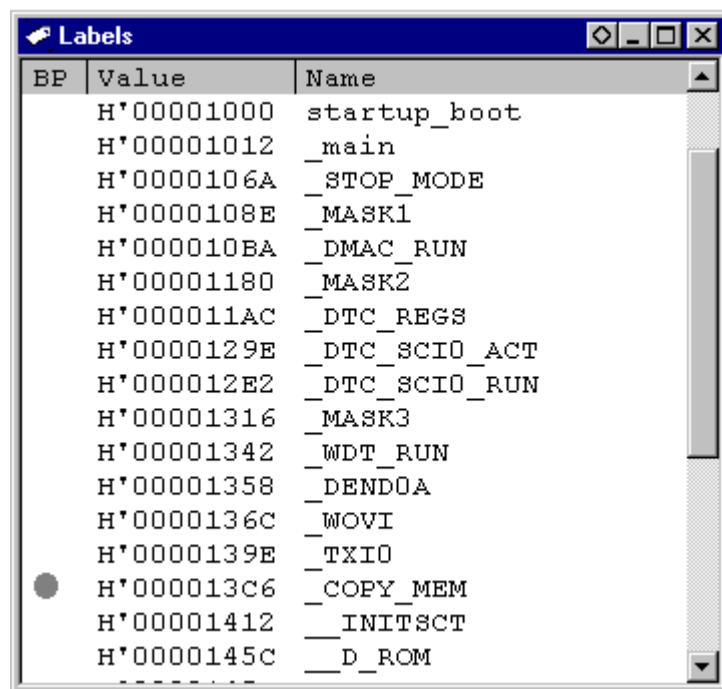
#### 13.4.2 Edit...

選択しているレジスタの値を変更するダイアログボックスを表示します。

#### 13.4.3 Expand/Collapse

選択しているモジュール情報を拡張、または圧縮します。

### 13.5 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_SCIO_ACT
	H'000012E2	_DTC_SCIO_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 13.5 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  
そのアドレスで標準のイベントをトグルします。
- 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 Labels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列に現在サポートされている標準ブレイ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イ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で表示されます。

### 13.5.1 Add...

Add Labelダイアログボックスを表示します。

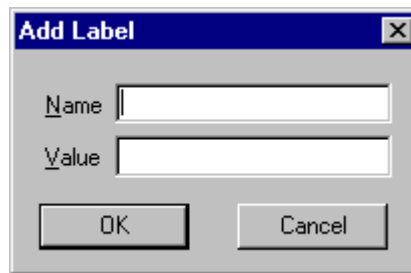


図 13.6 Add Labelダイアログボックス

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オープンされるので、関数を選択してValueフィールドを設定します。詳細は「10 関数の設定」を参照してください。

### 13.5.2 Edit...

Edit Labelダイアログボックスが表示されます。

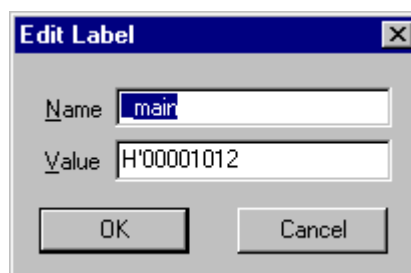


図 13.7 Edit Labelダイアログボックス

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オープンされるので、関数を選択してValueフィールドを設定します。詳細は「10 関数の設定」を参照してください。

### 13.5.3 Find...

Find Label Containingダイアログボックスを表示します。



図 13.8 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サーチします。

注 ラベルのはじめの1024文字でソートします。したがって、最初の文字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。

### 13.5.4 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。確認メッセージが表示されます。



図 13.9 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 Option DialogのConfirm Delete Labelオプションを選択しないでください。

### 13.5.5 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が表示されます。



図 13.10 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不要のときは、HDI Option DialogのConfirm Delete All Labelオプションを選択しないでください。

#### 13.5.6 Load...

現在のHDIのシンボルテーブルに結合します。Load Symbolダイアログボックスがオープンされます。

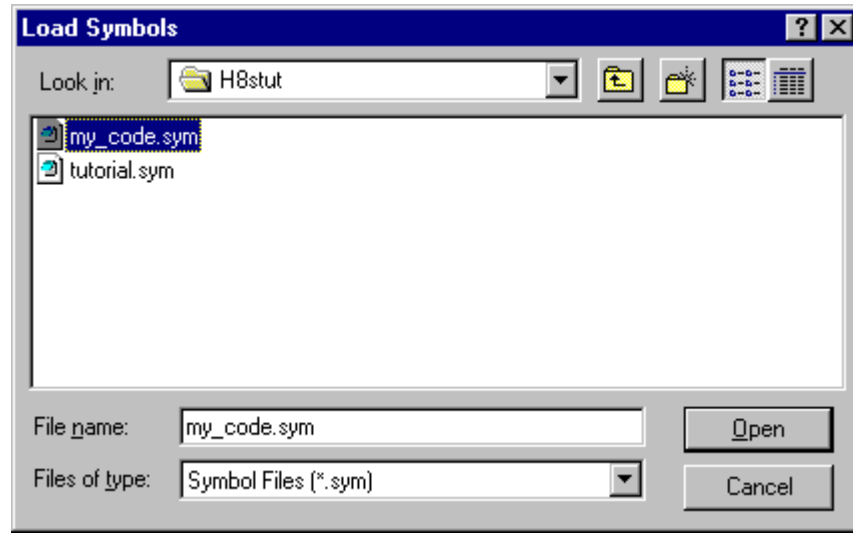


図 13.11 Load Symbols ダイアログボックス

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。

#### 13.5.7 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。

#### 13.5.8 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Fにあるので参照してください。

## 13.6 Locals

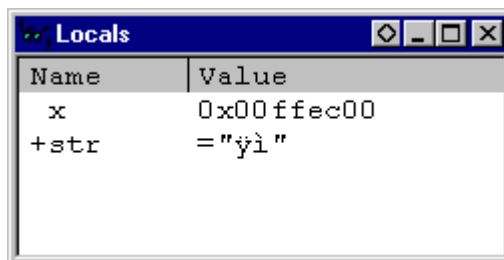



図 13.12 Localsウィンドウ

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、アプ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。

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、「8.3.2 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が表示されます。このメニューには次のオプションが含まれています。

### 13.6.1 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。

### 13.6.2 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ます。

### 13.6.3 Radix

選択しているローカル変数の表示基数を変更します。

## 13.7 Memory Mapping

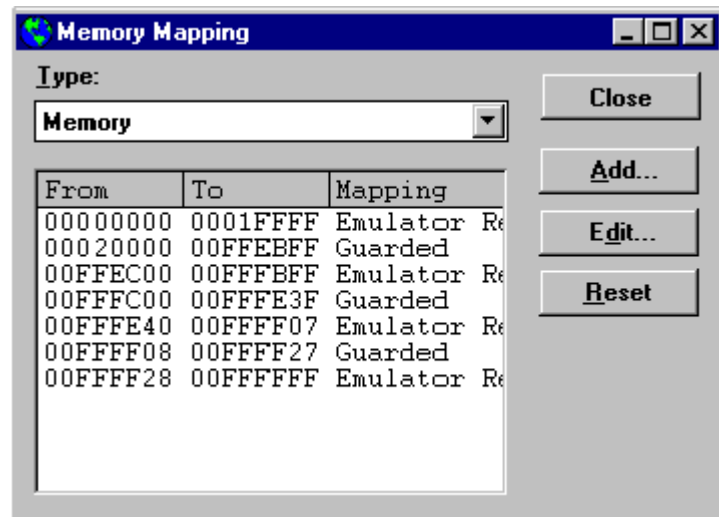


図 13.13 Memory Mappingウィンドウ

デバッグプラットフォームのメモリマップを表示・変更して、デバッグプラットフォームのメモリ構成およびリソースを表示するウィンドウです。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として表示します。また、コマンドボタンの機能は、デバッグプラットフォームにより以下に示したポップアップメニューオプションと異なる場合があります。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が表示されます。このメニューには次のオプションが含まれています。

## 13.7.1 Add

Edit Memory Mappingダイアログボックスを表示します。マップに追加する新規メモリ領域の詳細情報を入力します。デバッグプラットフォームがマップの変更を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されます。

## 13.7.2 Edit

Edit Memory Mappingダイアログボックスを表示します。現在選択されているメモリマップの詳細情報を変更します。デバッグプラットフォームがマップの変更を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されます。

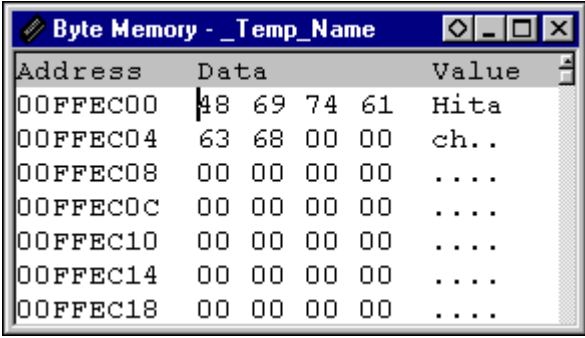
## 13.7.3 Reset

マップ情報をデバッグプラットフォームのデフォルト値に戻します。デバッグプラットフォームがマップの変更を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されます。

## 13.7.4 Help

ヘルプを起動します。

## 13.8 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 13.14 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が表示され、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"というメッセージが表示されます。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されます。

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が表示されます。このメニューには以下のオプションが含まれています。

### 13.8.1 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。

### 13.8.2 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。オプションベ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。

### 13.8.3 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、Findオプションと同様に設定されます。

### 13.8.4 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、Findオプションと同様に設定されます。

### 13.8.5 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、Findオプションと同様に設定されます。

### 13.8.6 Copy...

Move 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、Findオプションと同様に設定されます。

### 13.8.7 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が表示されたときに開始アドレスと終了アドレスが自動的に入力されます。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。

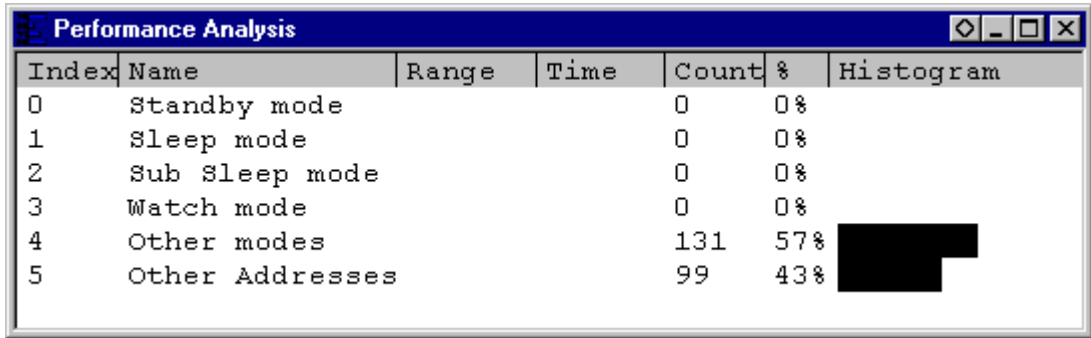
### 13.8.8 Search...

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が自動入力されます。

### 13.8.9 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。

## 13.9 Performance Analysis



Index	Name	Range	Time	Count	%	Histogram
0	Standby mode			0	0%	
1	Sleep mode			0	0%	
2	Sub Sleep mode			0	0%	
3	Watch mode			0	0%	
4	Other modes			131	57%	██████████
5	Other Addresses			99	43%	██████████

図 13.15 Performance Analysisウィンドウ

プログラムの実行効率データを表示して、制御するウィンドウです。デフォルトで表示される項目は、削除または変更することはできません。表示内容および動作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固有のパフォーマンスについては別冊の『デバッグプラットフォームのマニュアル』を参照してください。

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が表示されます。このメニューには以下のオプションが含まれています。

### 13.9.1 Add Range

Add PA Rangeダイアログボックスを表示します。ソース行またはアドレス範囲に基づいて新しいユーザ範囲を追加します。範囲名は編集可能です。

### 13.9.2 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。Edit PA Range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。

### 13.9.3 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。

### 13.9.4 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。

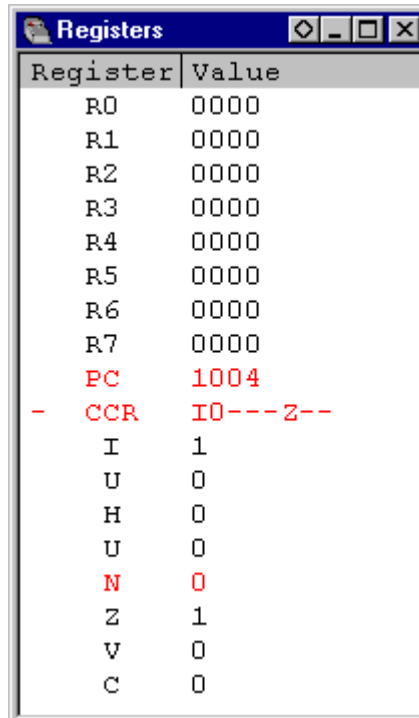
### 13.9.5 Delete All Ranges

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。

### 13.9.6 Analysis Enabled

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が表示されます。

## 13.10 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 13.16 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が表示されます。このメニューには以下のオプションが含まれています。

### 13.10.1 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。

### 13.10.2 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。

### 13.10.3 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。

## 13.11 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を指定されていれば表示できます。

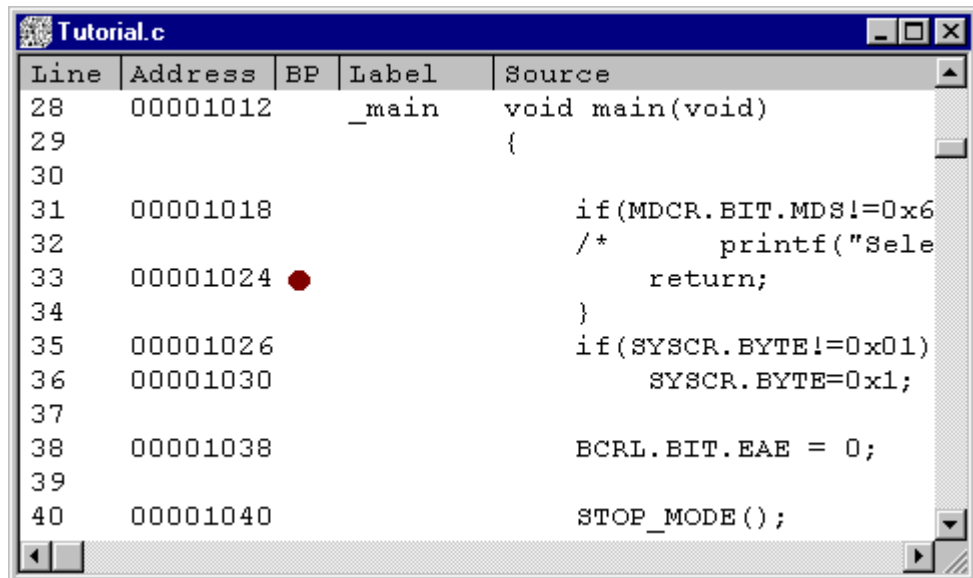


図 13.17 Sourceウィンドウ

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をオープンし関数を選択します。

- 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箇所を表示します。

右クリックするとBP行に現在サポートされ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行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で表示されます。



### 13.11.1 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ます。

### 13.11.2 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。ソース表示フォーマットでのみ使用できます。

### 13.11.3 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。

### 13.11.4 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。ソース表示フォーマットでのみ使用できます。

### 13.11.5 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、別のブレイク条件が成立するまで実行されます。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されます。

### 13.11.6 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。

### 13.11.7 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。

### 13.11.8 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されます。ソース行が有効なときのみ機能します。

## 13.11.9 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。

## 13.12 System Status

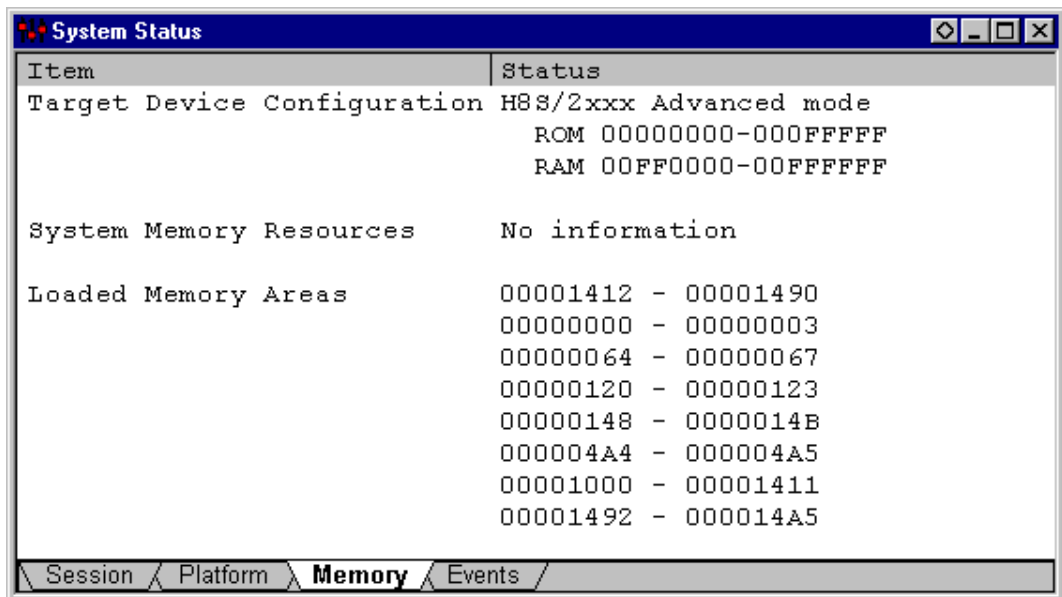


図 13.18 System Statusウィンドウ

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。テキストは、標準セクション(EmulatorからCause of last breakまで)とデバッグプラットフォーム依存セクションという2つのセクションにより構成されています。プラットフォーム依存セクションについては、別冊の『デバッグプラットフォームのマニュアル』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が表示されます。このメニューには以下のオプションが含まれています。

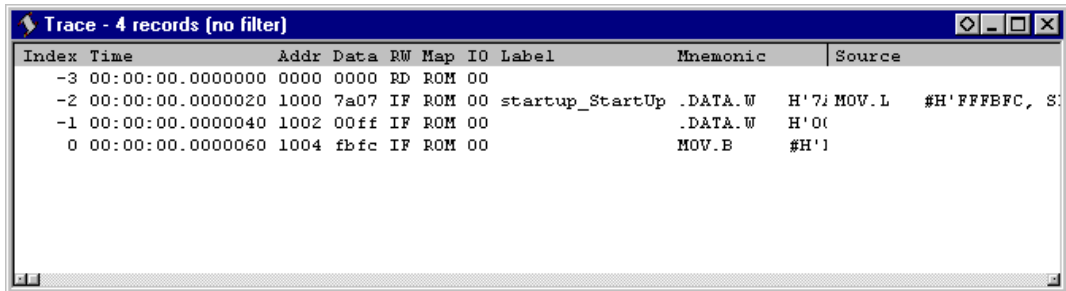
## 13.12.1 Update

表示データを最新情報に更新します。

## 13.12.2 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ます。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。

## 13.13 Trace



Index	Time	Addr	Data	RW	Map	IO	Label	Mnemonic	Source
-3	00:00:00.0000000	0000	0000	RD	ROM	00			
-2	00:00:00.0000020	1000	7a07	IF	ROM	00	startup_StartUp	.DATA.W H'7; MOV.L #H'FFBFC, S:	
-1	00:00:00.0000040	1002	00ff	IF	ROM	00		.DATA.W H'0(	
0	00:00:00.0000060	1004	fbfc	IF	ROM	00		MOV.B #H'1	

図 13.19 Trace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に達するまでに実行された命令シーケンスを表示するウィンドウです。表示は選択したデバッグプラットフォームによって異なります。

列をダブルクリックするとそのアドレスのソースまたは逆アセンブルコードを表示します。

コマンドボタンの機能は、以下に示したポップアップメニューオプションと同じです。

## 13.13.1 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。

## 13.13.2 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。

## 13.13.3 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。

## 13.13.4 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。

## 13.13.5 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。

## 13.13.6 Restart

トレースデータの収集を開始します。

## 13.13.7 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。

## 13.13.8 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。

## 13.13.9 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。

## 13.13.10 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。

## 13.13.11 Trim Source

ソースプログラムの左側の空白を取り除きます。

## 13.14 Watch

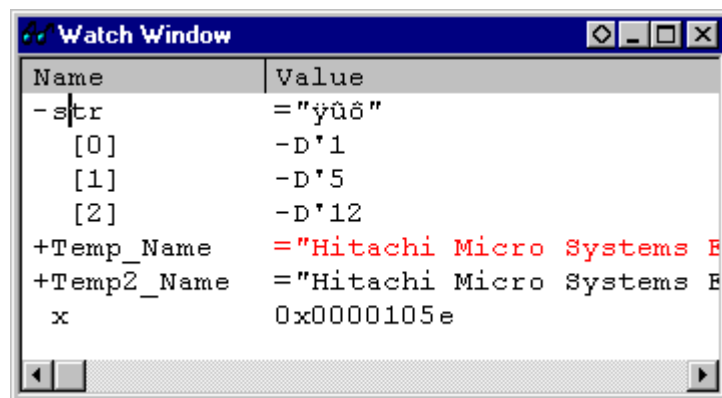



図 13.20 Watchウィンドウ

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。変数はリスト表示され、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示すプラス記号と、情報を収縮表示できることを示すマイナス記号が付きます。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が表示されます。このメニューには以下のオプションが含まれます。

#### 13.14.1 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。

#### 13.14.2 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。

#### 13.14.3 Delete All

すべての変数を、Watchウィンドウから削除します。

#### 13.14.4 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。

#### 13.14.5 Edit Value

Edit Watchダイアログボックスを表示して、変数の値を変更できるようにします。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。

#### 13.14.6 Radix

選択しているウォッチ項目の表示基数を変更します。



## 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。

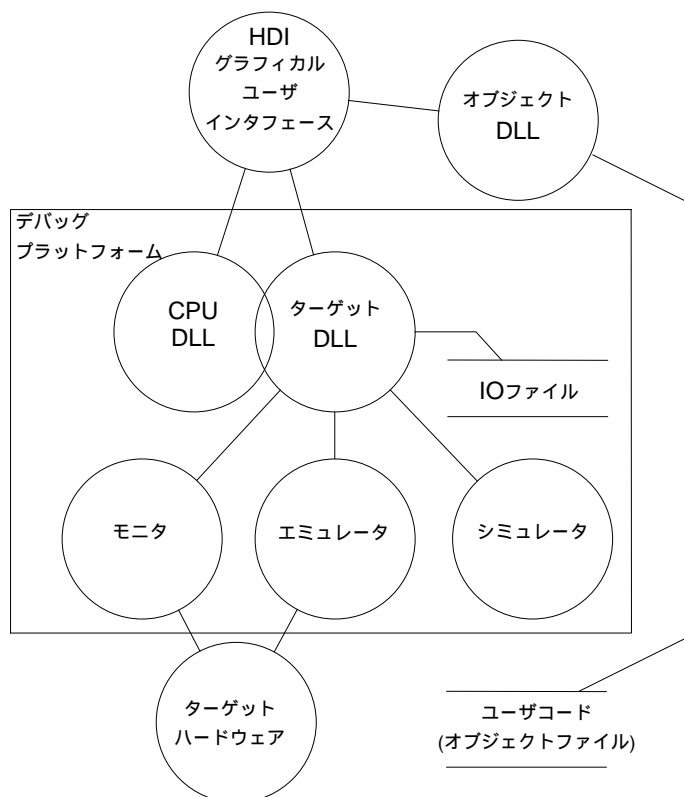


図 A.1 HDIシステムモジュール

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。

### 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。

## 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を見つけるまで、1つ1つを試します。

## 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。

## 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せん。

ターゲットDLLで使用できる機能の詳しい説明については、各『デバッグプラットフォームのマニュアル』を参照してください。



## 付録 B コマンドラインフォーマット

### HDI 搭載コマンド

以下のコマンドは、HDIで使用することができる標準コマンドです。

#### !(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

#### ACCESS

省略形: AC

説明:

不正アクセス時の処理を設定、または表示します。

シンタックス

access [ <state> ]

パラメータ	型	説明
なし		現在の設定を表示します
<state>	キーワード	処理を設定します
	break	ブレークします(デフォルト)
	none	なにもしません

不正アクセスとは、ユーザプログラム実行中にプロテクトエリアへの書き込み、内蔵ROMへの書き込み、またはメモリマップされない領域へのアクセスなどのことです。

例:

ACCESS break      不正アクセスがあったらブレークします(デフォルト設定)  
 AC                  現在設定されている不正アクセス時の処理を表示します  
 AC none            不正アクセスがあっても何もしません

**ANALYSIS**

省略形: AN

説明:

性能分析範囲を有効、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されません。

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:

性能分析範囲を設定するか、またはパラメータなしで性能分析範囲を表示します。シンタックスはデバッグプラットフォームによって異なります。別冊の『デバッグプラットフォームのマニュアル』を参照してください。

シンタックス

ar [ &lt;start&gt; &lt;end&gt; [ &lt;name&gt; ] ]

パラメータ	型	説明
なし		すべての性能分析範囲を表示します
<start>	数値	性能分析範囲の開始アドレス
<end>	数値	性能分析範囲の終了アドレス
<name>	文字列	性能分析範囲の名称

例:

ANALYSIS_RANGE H'0 H'100	アドレス H'0 - H'100 の性能分析範囲を定義します
AR H'1000 H'3FFF	アドレス H'1000 - H'3FFF の性能分析範囲を定義します
AR	性能分析範囲設定を表示します

**ANALYSIS\_RANGE\_DELETE**

省略形: AD

説明:

指定された項目番号の性能分析範囲、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性能分析範囲を削除します(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		すべての性能分析範囲を削除します
<index>	数値	削除する性能分析範囲の番号

例:

ANALYSIS\_RANGE\_DELETE 6 システムから項目番号 6 の性能分析範囲を削除します  
AD すべての性能分析範囲を削除します

**ASSEMBLE**

省略形: AS

説明:

メモリにアセンブルします。

アセンブルモードでは "." は終了、"^" は1バイト戻り、**ENTER**キーを押すと先に進みません。

シンタックス

as &lt;address&gt;

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。式は偽のときエラーが返されます。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 &lt;expression&gt;

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

**DISASSEMBLE**

省略形: DA

説明:

アセンブルコードのメモリ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 &lt;address&gt; [&lt;length&gt;]

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。

シンタックス

er

パラメータ	型	説明
なし		Command Line ウィンドウをクリアします

例:

ER                              Command Line ウィンドウをクリアします

**EVALUATE**

省略形: EV

説明:

簡単な式、そして括弧、混合基数とシンボルを持つ複雑な式を評価することができます。すべての演算子は同じ優先順位を持っていますが、括弧は評価の順序を変更することができます。演算子はC/C++と同じ意味を持っています。式もまた数値が要求されるコマンドで使うことができます。レジスタ名を使うことができませんが、通常文字で前定義されなければいけません。結果は、16進、10進、8進、2進で表示されます。

シンタックス

ev &lt;expression&gt;

パラメータ	型	説明
<expression>	式	評価する式

有効な演算子:

&& 論理 AND	論理 OR	<< 左算術シフト	>> 右算術シフト
+ 加算	- 減算	* 乗算	/ 除算
% 剰余	ビット毎の OR	& ビット毎の AND	~ ビット毎の NOT
^ ビット毎の排他的 OR	! 論理 NOT	== 等しい	!= 等しくない
> より大きい	< より小さい	>= 以上	<= 以下



**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

**GO**

省略形: GO

説明:

ユーザプログラムを実行します。ユーザプログラムが実行されているとき、Performance Analysis ウィンドウは更新されます。ユーザプログラムが停止すると、PC値が表示されません。

シンタックス

go [ &lt;state&gt; ] [ &lt;address&gt; ]

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 (任意、デフォルト= wait )
	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	PC のための開始アドレス (任意、デフォルト=PC 値)

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    現在の PC から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 CONTINUE H'1000                      H'1000 から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す)

**GO\_RESET**

省略形: GR

説明:

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。ユーザプログラムが実行されているとき、Performance Analysisウィンドウは更新されます。

シンタックス

gr [ &lt;state&gt; ]

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 (任意、デフォルト= wait )
	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

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:

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、カレントの 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 [ &lt;state&gt; ] &lt;address&gt;...

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 (リスト)

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

**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。同じことをコマンドラインで HELP、または HE、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 [ &lt;command&gt; ]

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を含む)、ターゲット (ターゲットライブラリを再選択するように) を初期化します。すべてのブレークポイント、メモリマッピングなどもリセットされます。

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



**INTERRUPTS**

省略形: IR

説明:

割り込みを有効、または無効にするか、CPUの割り込み優先度レベルを設定します。このコマンドは CPU Status Register (SR、または CCR) を変更することにより操作します。

**注意:** このコマンドは、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ことがあります。

シンタックス

ir [ &lt;state&gt; | &lt;level&gt; ]

パラメータ	型	説明
なし		CPU 割り込み状態を表示します
<state>	キーワード	割り込みの有効/無効
	enable	すべての割り込みを有効にします
	disable	すべての割り込みを無効にします (NMI を除く)
<level>	数値	割り込み優先度レベルを設定します

例:

IR CPU 割り込み状態を表示します  
 IR ENABLE すべての割り込みを有効にします  
 IR DISABLE すべての割り込みを無効にします (NMI を除く)  
 IR 5 割り込み優先度レベル 5 を設定します

**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、現在のログ状況が表示されます。存在するファイルを指定すると、追加するかユーザに確認され、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されます。

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 のリストボックス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 カレントのロギング状態を表示します

**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

**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、カレント位置は 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)

## MEMORY\_FILL

省略形: MF

説明:

メモリ領域を指定されたデータ値に変更します。

## シンタックス

mf &lt;start&gt; &lt;end&gt; &lt;data&gt; [ &lt;mode&gt; ] [ &lt;state&gt; ]

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
<data>	数値	データ値
<mode>	キーワード	データサイズ (任意、デフォルト=byte)
	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
<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:

MEMORY\_FILL H'C000 H'C100 H'55AA WORD

H'C000 から H'C0FF までワードデータ H'55AA に変更します

MF H'5000 H'7FFF H'21

H'5000 から H'7FFF までデータ H'21 に変更します

**MEMORY\_MOVE**

省略形: MV

説明:

メモリ領域を移動します。

シンタックス

mv &lt;start&gt; &lt;end&gt; &lt;dest&gt; [ &lt;state&gt; ]

パラメータ	型	説明
<start>	数値	ソース開始アドレス
<end>	数値	ソース終了アドレス (この値を含む)
<dest>	数値	移動先開始アドレス
<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:

MEMORY\_MOVE H'1000 H'1FFF H'2000    領域 H'1000 - H'1FFF を H'2000 へ移動します  
 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。テストはマップ環境によりメモリをアクセスします。

シンタックス

mt &lt;start&gt; &lt;end&gt;

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

**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 [&lt;mode&gt;]

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 &lt;register&gt; &lt;value&gt; &lt;mode&gt;

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 \_StartUp

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に設定されます。メモリマッピングとブレークポイントは影響されません。

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

**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 &lt;milliseconds&gt;

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。カレントの 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によって行われます。ステップ数のデフォルトは1です。

シンタックス

st [ &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:

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:

カレント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:

カレントの 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されます。

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

**SUBMIT**

省略形: SU

説明:

エミュレータコマンドの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 &lt;filename&gt;

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 &lt;symbol&gt; &lt;value&gt;

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    カレント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 [ &lt;symbol&gt; ]

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。 シンボルファイル拡張子は .SYM をデフォルトに設定します。

シンタックス

sl &lt;filename&gt;

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 フォーマットである必要があります。 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が表示されます。

シンタックス

ss &lt;filename&gt;

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。

シンタックス

sv [ &lt;pattern&gt; ]

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。遅延トレースが設定されていないとき、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。トレース遅延が設定されているとき、レベル1のブレイクポイントが発生したサイクルはサイクル0で、最も新しいサイクルはトレース遅延値の数を示します。

シンタックス

tr [ &lt;start rec&gt; [ &lt;count&gt; ] ]

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 -10 5

サイクル-10 から 5 行トレースバッファの内容を表示します

## デバッグプラットフォーム特有コマンド

次のコマンドは、デバッグプラットフォーム単位に特有なコマンドです。詳細については、別冊の『デバッグプラットフォームのマニュアル』を参照してください。

ANALYSIS\_RANGE  
BREAKPOINT  
BREAKPOINT\_CLEAR  
BREAKPOINT\_DISPLAY  
BREAKPOINT\_ENABLE  
BREAKPOINT\_SEQUENCE  
BREAK\_ACCESS  
BREAK\_CLEAR  
BREAK\_DATA  
BREAK\_DISPLAY  
BREAK\_ENABLE  
BREAK\_REGISTER  
BREAK\_SEQUENCE  
CLOCK  
DEVICE\_TYPE  
MAP\_SET  
MODE  
REFRESH  
TEST\_EMULATOR  
TIMER  
TRACE\_ACQUISITION  
TRACE\_COMPARE  
TRACE\_SAVE  
TRACE\_SEARCH  
USER\_SIGNAL
























































## 付録 C コマンドライン一覧表

コマンド名	短縮形	説明
!	-	コメント
ACCESS	AC	不正アクセスに対する動作の設定
ANALYSIS	AN	性能分析機能の有効化 / 無効化
ANALYSIS_RANGE	AR	性能分析範囲の設定、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	ブレークポイントの設定
BREAKPOINT_CLEAR	BC	ブレークポイントの削除
BREAKPOINT_DISPLAY	BD	ブレークポイントの表示
BREAKPOINT_ENABLE	BE	ブレークポイントの有効化 / 無効化
BREAKPOINT_SEQUENCE	BS	イベントシーケンスの設定、削除
BREAK_ACCESS	BA	メモリ範囲のアクセスによるブレークポイントの設定
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	ブレークポイントの表示
BREAK_ENABLE	BE	ブレークポイントの有効化 / 無効化
BREAK_REGISTER	BR	レジスタのデータ値によるブレークポイントの設定
BREAK_SEQUENCE	BS	実行順序を指定したブレークポイントの設定
CLOCK	CK	エミュレータの CPU クロック時間の設定
DEVICE_TYPE	DE	エミュレータのデバイスタイプの選択
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	プラットフォームの初期化
INTERRUPTS	IR	プラットフォームの割り込み処理の有効化 / 無効化
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	メモリマッピングの設定
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
MODE	MO	CPU モードの設定、表示
QUIT	QU	HDI の終了
RADIX	RA	入力ラディックス(基数)の設定
REFRESH	RF	メモリ関連ウインドウの更新
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	エミュレータ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
TEST_EMULATOR	TE	エミュレータハードウェアのテスト
TIMER	TI	実行時間測定タイマの分解能の設定、表示
TRACE	TR	トレース情報の表示
TRACE_ACQUISITION	TA	トレース取得条件の設定、表示
TRACE_COMPARE	TC	トレースファイルと現在のトレース情報を比較
TRACE_SAVE	TV	トレース情報をトレースファイルに保存
TRACE_SEARCH	TS	トレース情報の検索
USER_SIGNALS	US	ユーザシグナル情報の有効化 / 無効化

## 付録 D 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	Toolbar		
	Status bar		
	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	
	Registers	Ctrl+R	
	Source...	Ctrl+K	
	Status	Ctrl+U	
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	

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
Memory	<u>Refresh</u> <u>Load...</u> <u>Save...</u> <u>Verify...</u> <hr/> <u>Test...</u> <u>Fill...</u> <u>Copy...</u> <u>Compare...</u> <u>Search...</u> <hr/> <u>Configure Map...</u> <u>Configure Overlay...</u>	F12	          
Setup	<u>Options...</u> <u>Radix (Input) &gt;</u> <u>Hexadecimal</u> <u>Decimal</u> <u>Octal</u> <u>Binary</u> <u>Customize</u> <u>Toolbar...</u> <u>Font...</u> <u>File Filter...</u> <hr/> <u>Configure Platform...</u>		    
Window	<u>Cascade</u> <u>Tile</u> <u>Arrange Icons</u> <hr/> <u>Close All</u>		  
Help	<u>Index</u> <u>Using Help</u> <u>Search for Help on</u> <hr/> <u>About HDI</u>	F1	



## 付録 E I/O レジスタファイルのフォーマット

HDIは、I/Oレジスタ定義ファイルの情報に基づいて、I/O Registersウィンドウをフォーマットします。[Setup->Configure Platform...]メニューオプションを使用してデバッグプラットフォームを選択すると、HDIは、選択したデバイスに対応する“<device>.IO”ファイルを検索し、そのファイルが存在する場合にファイルをロードします。このファイルはフォーマット済みのテキストファイルで、各I/Oモジュールおよびそれらのレジスタのアドレスとサイズが記述されています。テキストエディタを使用してこのファイルを編集し、アプリケーションに固有のメモリマップレジスタまたは周辺レジスタ(例：マイクロコンピュータのアドレス空間にマップされたASICデバイスのレジスタなど)のサポートを追加できます。

### ファイルのフォーマット

各モジュール名は[Modules]定義セクションで定義する必要があり、各モジュールの番号は連続していなければなりません。モジュールごとにレジスタ定義セクションがあり、そのセクション内の各エントリにより、I/Oレジスタが定義されます。

Base Addressは、CPUモードに応じてI/Oレジスタの位置が移動するデバイスで設定します。Base Addressには、あるモードでのI/Oレジスタの基準アドレスを設定し、レジスタ定義には、同一モードでの各レジスタのアドレスを設定してください。I/Oレジスタが実際に使用される際には、レジスタ定義のアドレスからBase Addressの値を差し引き、その結果求められたオフセット値が選択されたモードでの基準アドレスに加算されます。

各モジュールの定義セクションには、依存関係を示すオプションで構成されるレジスタを定義するセクションがあります。依存関係のオプションはモジュールが使用可能かどうかをチェックすることができます。

各レジスタ名は連続な登録番号とともにセクションの中で定義します。

依存関係のオプションは以下のように定義されます。

dep=<reg> <bit> <value>

1. <reg> : オプションの登録IDです。
2. <bit> : レジスタ内のビット位置です。
3. <value> : モジュールがイネーブルになるためのビットの値です。

[Register]定義エントリは次のフォーマットで定義されます。  
id=<name> <address> [<size>[<absolute>[<format>[<bitfield>]]]]

1. <name> : レジスタ名
2. <address> : レジスタの値
3. <size> : B(バイト)、W(ワード)、またはL(ロングワード)のいずれかです(デフォルトはバイトです)。
4. <absolute> : I/Oレジスタが絶対アドレスの場合は<absolute>に'A'を指定できます。'A'を指定した場合、基準アドレスのオフセット計算は行わず、指定されたアドレスを直接使用します。CPUモードによってI/Oレジスタの位置が変わる場合は<absolute>を指定しないでください。
5. <format> : レジスタの表示形式 (H:16進数、D:10進数、B:2進数)
6. <bitfield> : レジスタ内のビットを定義しているセクション名

ビットフィールドはレジスタ内のビットを以下のフォーマットで定義します。

bit<no>=<name>

1. <no> : ビット番号
2. <name> : ビットのシンボル名

コメント行を付けることができますが、その場合は、“;”文字で始めなければなりません。

例:

コメント ; H8S/2655 Series I/O Register Definitions File

モジュール

```
[Modules]
BaseAddress=0
Module1=Power_Down_Mode_Registers
Module2=DMA_Channel_Common
Module3=DMA_0_Short_Address_Mode
...
Module42=Bus_Controller
Module43=System_Control
Module44=Interrupt_Controller
```

...

モジュール  
の定義

```
[DMA_Channel_Common]
reg0=regDMAWER
reg1=regDMATCR
reg2=regDMABCRH/SAM
reg3=regDMABCRL/SAM
reg4=regDMABCRH/FAM
reg5=regDMABCRL/FAM
dep= regMSTPCRH 7 0
```

レジスタ名  
ビット  
値

...

レジスタの  
定義 [regDMAWER]  
id=DMAWER 0xffff00 B A H dmawer\_bitfields

レジスタ名  
アドレス  
サイズ  
絶対アドレスフラグ  
フォーマット  
ビットフィールド

...

ビット  
フィールドの  
定義

```
[dmawer_bitfields]
bit0=WE0A
bit1=WE0B
bit2=WE1A
bit3=WE1B
```



## 付録 F 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

About HDI .....	72
Acquisition .....	91
Add .....	73, 79, 83
Add Range .....	86
Add Watch .....	40, 41, 77, 89, 93
Address .....	76, 79, 88
Analysis Enabled .....	86
Arrange Icons .....	54, 72
ASCII .....	3, 125
Assembler .....	16
BP .....	79, 88
Breakpoint/Event Properties .....	35, 73
Breakpoints .....	33, 34, 35, 36, 67, 73
Cascade .....	55, 72
Clear .....	92
Close All .....	72
Code and Assembler .....	76
Command Line .....	67, 74, 97, 100
Compare .....	70, 85
Configure Map .....	8, 70
Configure Overlay .....	70
Configure Platform .....	8, 9, 71, 121
Copy .....	66, 70, 76, 78, 82, 85, 87, 90, 93
Copy Memory .....	26, 70
Customize .....	59, 60, 71
Cut .....	66
Delete .....	73, 80, 93
Delete All .....	73, 80, 93
Delete All Ranges .....	86
Delete Range .....	86
Description .....	58
Disable/Enable .....	74
Disassembly .....	16, 29, 30, 53, 56, 67, 75
Edit .....	9, 23, 35, 46, 66, 73, 78, 79, 83, 87
Edit Range .....	86
Edit Value .....	42, 82, 93
Evaluate .....	67
Exit .....	62, 66
Expand/Collapse .....	78
File .....	11, 35, 61
File Filter .....	60
Fill .....	70, 85
Fill Memory .....	70, 85
Filter .....	91
Find .....	66, 67, 70, 80, 85, 89, 91
Find Next .....	91
Font .....	59
Go .....	29, 34, 37, 38, 68
Go To Cursor .....	30, 37, 68, 77, 89
Go To Disassembly .....	90
Go To Source .....	74
Halt .....	33, 69, 91
HDI Options .....	62, 71
HDI セッションファイル .....	61
Help .....	4, 72, 83
I/O Area .....	67
I/O Registers .....	47, 48, 67, 77, 121

Index.....	72
Initialize.....	66
Instant Watch.....	39, 40, 77, 89
Label.....	76, 88
Labels.....	17, 67, 78
Line.....	88
Load.....	69, 81, 84
Load Memory.....	28
Load Program.....	11, 12, 66
Load Session.....	61, 65
Locals.....	43, 67, 82
Logging.....	75
Memory.....	8, 21, 23, 25, 26, 27, 47, 51, 59, 67, 69, 84
Memory Mapping.....	9, 10, 70, 83
Minimize.....	53
Name.....	79
New.....	57
New Session.....	65
Next.....	55
Open.....	68
Open Memory Window.....	21, 67
Options.....	62, 71
Overlay.....	49, 50
Paste.....	66
PC.....	19, 29, 37, 38, 46, 67, 68, 69, 77, 89, 103, 111, 118
Performance Analysis.....	68, 86, 102, 103
Play.....	75
Radix.....	3, 42, 64, 71, 82, 93
Refresh.....	69
Register.....	46, 122
Registers.....	44, 68, 87
Reset.....	83
Reset Counts/Times.....	86
Reset CPU.....	68
Reset Go.....	29, 37, 38, 68
Restart.....	91
Restore.....	53
Run.....	29, 30, 31, 33, 68, 69
Run Batch File.....	74
Run Program.....	37, 51, 69
Save.....	27, 70, 81, 85, 92
Save As.....	81
Save Memory.....	27
Save Session.....	61, 65
Save Session As.....	61, 65
Search.....	24, 85
Search for Help on.....	72
Select Function.....	19, 23, 37, 51, 52, 76, 84, 89
Select Session.....	7, 8
Set Address.....	19, 23, 51, 76, 84, 89
Set Line.....	89
Set Log File.....	75
Set PC Here.....	77, 89
Set PC To Cursor.....	69
Setup.....	3, 8, 49, 56, 59, 60, 62, 64, 71, 121
Snapshot.....	92
Source.....	15, 19, 20, 29, 30, 34, 37, 39, 40, 46, 51, 59, 68, 76, 87, 88
Status.....	10, 68
Status Bar.....	56, 71
Step.....	69



---

Step In .....	31, 69
Step Out .....	31, 69
Step Over .....	31, 32, 69
Step Program .....	31, 69
System Status .....	10, 11, 68, 90
Test .....	70, 85
Test Memory .....	26, 27, 70, 85
Tile .....	55, 72
Toggle Bit .....	87
Toolbar .....	56, 71
Tooltip Watch .....	39
Trace .....	68, 91
Trim Source .....	92
Update .....	90
Use as Default Font .....	59
Using Help .....	72
Verify .....	70
View .....	10, 15, 16, 17, 21, 34, 40, 43, 44, 47, 67
View Source .....	92
Watch .....	39, 40, 41, 43, 68, 77, 89, 92, 93
Window .....	72



# ルネサスデバッグインタフェース 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-702-231B