お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

User's Manual

M3T-PD38SIM V.2.10

ユーザーズマニュアル

740 ファミリ用シミュレータデバッガ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.05

Microsoft 、MS- DOS 、Windows およびWindows NT は、米国Microsoft Corporation の米国およびその他の国における登録商標です。
IBM およびAT は、米国International Business Machines Corporation の登録商標です。
Intel,Pentium は、米国Intel Corporation の登録商標です。
Adobe およびAcrobat は、Adobe Systems Incorporated （アドビシステムズ社）の登録商標です。
その他すべてのブランド名および製品名は個々の所有者の登録商標もしくは商標です。

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

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

製品の内容及び本書についてのお問い合わせ先

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

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス ソリューションズ マイコンツール部
ツール技術サポート窓口　support_tool@renesas.com
ユーザ登録窓口　regist_tool@renesas.com
ホームページ　http://www.renesas.com/jp/tools

はじめに

PD38SIMは、8 ビットマイクロコンピュータ 740 ファミリの動作をシミュレートして、そのターゲットプログラムの評価を行うためのシミュレータデバッガ［Windows 対応版］です。本ユーザーズマニュアルは、PD38SIMの特長、機能、セットアップ方法、操作方法等について説明しています。

プログラムの使用権

本製品に含まれるプログラムの使用権は、「ソフトウェア使用権許諾契約書」に基づきます。PD38SIMのプログラムは、お客様の製品開発の目的でのみ使用できます。その他の目的では使用できませんのでご注意ください。

また、本マニュアルによってソフトウェアの使用権の実施に対する保証及び使用権の実施の許諾を行うものではありません。

このページは白紙です。

目次

概要編 1

1PD38SIM の概要 3

2PD38SIM の特長 4

2.1 マルチウィンドウ機能....4
2.2 I/O シミュレーション機能....4
2.3 割り込みシミュレーション機能....4
2.4 簡易システムシミュレーション機能....4
2.5 RAM モニタ機能....4
2.6 ブレーク機能....5
2.7 ソースレベルデバッグ機能....5
2.8 オンデマンド方式....5

3 PD38SIM のシミュレーション仕様 6

3.1 実チップとの主な相違点....6
3.2 命令動作....7
3.3 リセット動作....7
3.4 メモリ....8
3.5 仮想ポート入力機能....8
3.6 仮想ポート出力機能....8
3.7 仮想割り込み機能....8
3.8 GUI 入力機能....9
3.9 GUI 出力機能....9
3.10 I/O スクリプト機能....9
3.11 シミュレータ固有の機能....10

4 PD38SIM の入出力ファイル 11

4.1 入力ファイル....11
4.2 出力ファイル....13
4.3 テンポラリファイル....14

セットアップ編 15

1 セットアップ 17

1.1 インストール 17
1.2 PD38SIM の起動 17
1.3 pd38sim の動作環境の設定 18
1.4 sim38 の動作環境の設定 20

ウィンドウ機能編 23

1 PD38SIM のウィンドウ機能 25
1.1 PD38SIM ウィンドウ 25
1.2 プログラムウィンドウ 30
1.3 ソースウィンドウ 35
1.4 レジスタウィンドウ 38
1.5 メモリウィンドウ 40
1.6 ダンプウィンドウ 42
1.7 RAM モニタウィンドウ 44
1.8 ASM ウォッチウィンドウ 47
1.9 C ウォッチウィンドウ 50
1.10 ローカルウィンドウ 53
1.11 ファイルローカルウィンドウ 55
1.12 グローバルウィンドウ 57
1.13 スクリプトウィンドウ 58
1.14 I/O ウィンドウ 60
1.15 GUI 入力ウィンドウ 77
1.16 GUI 出力ウィンドウ 79
1.17 カバレッジウィンドウ 81
1.18 S/W ブレークポイント設定ダイアログ 84
1.19 H/W ブレークポイント設定ダイアログ 86

基本操作方法編 87

1 ターゲットプログラムの読み込み・表示 89
1.1 ダウンロードするには 89
1.2 最近ダウンロードしたファイルを再ダウンロードするには 91
1.3 ロードモジュール更新時に自動ダウンロードするには 92
1.4 ダウンロード直後のプログラム表示位置を変更するには 92
1.5 アップロードするには 93

1.6 逆アセンブル結果を保存するには 93
1.7 プログラムの任意位置を常に表示するには 94
1.8 プログラムの表示位置を変更するには 94
1.9 他ディレクトリに存在するソースプログラムを参照するには 98
1.10 ソースと逆アセンブル結果を MIX 表示するには 99
1.11 逆アセンブル結果を表示するには 100
1.12 表示色のカスタマイズ 101

2 ターゲットプログラムの実行／停止 102

2.1 実行・停止するには 102
2.2 ステップ実行するには 103
2.3 現ルーチンから上位ルーチンへ戻るには 104
2.4 指定位置までプログラムを実行するには 105
2.5 プログラムをリセットするには 105
3 レジスタ情報、メモリ内容の参照・設定 106
3.1 レジスタの内容を参照するには 106
3.2 レジスタの内容を変更するには 106
3.3 プログラム実行中に RAM の変化を参照するには 109
3.4 任意アドレスの値を参照するには 109
3.5 スコープを切り替えるには 113
3.6 指定アドレスにデータを設定するには 113
3.7 メモリ内容の表示を更新するには 114
3.8 メモリの取得モードを切換えるには 115

4 ソフトウェアブレーク 116

4.1 S/W ブレークポイント設定ダイアログをオープンするには 116
4.2 ブレークポイントを設定するには 117
4.3 ブレークポイントを解除するには 118
4.4 ブレークポイントを一時的に無効化するには 118
4.5 ブレークポイントを一時的に有効化するには 119
4.6 ウィンドウ上からブレークポイントを設定するには 119
4.7 ツールバーからブレークポイントを設定するには 120
4.8 ブレークポイントを保存するには 120
4.9 ブレークポイントを読み込むには 120

5 ハードウェアブレーク 121

5.1 H/W ブレークポイント設定ダイアログをオープンするには 121
5.2 ハードウェアブレークポイントを設定するには 122
5.3 ハードウェアブレークポイントを解除するには 125

6 C 変数の参照・変更 126

6.1 C 変数の値を参照するには 126

6.2 C 変数の値を変更するには 129

7 スクリプトコマンド 130

7.1 スクリプトコマンドを実行するには....130

7.2 スクリプトコマンドの実行結果を記録するには 131

7.3 スクリプトコマンドを一括して実行するには....134

8 PD38SIM の終了 136

8.1 PD38SIM を終了するには....136

9 その他 137

9.1 ラインアセンブルするには....137

9.2 Make を起動するには....139

9.3 ターゲットプログラムの文字列を検索するには 140

9.4 ウィンドウの表示領域の割合を変更するには....140

9.5 アクティブウィンドウを切換えるには 141

9.6 PD38SIM のバージョンを表示するには 141

9.7 PD38SIM の動作をカスタマイズするには....142

9.8 エディタをオープンするには 142

より高度なデバッグ編 143

1 I/O ウィンドウでの仮想ポート入力の設定 145

1.1 概要 145

1.2 サイクル同期入力の設定 145

1.3 リードアクセス同期入力の設定....148

1.4 割り込み同期入力の設定 150

2 I/O ウィンドウでの仮想ポート出力の設定 153

2.1 概要 153

2.2 仮想ポート出力の設定....153

3 I/O ウィンドウでの仮想割り込みの設定 155

3.1 概要 155

3.2 サイクル同期割り込みの設定 155

3.3 実行アドレス同期割り込みの設定 157

4 I/O ウィンドウのその他の機能 160

4.1 仮想ポート入力、仮想割り込みの設定データを変更するには 160
4.2 設定した仮想ポート入力、仮想ポート出力、仮想割り込み、I/O スクリプトファイルを削除するには 165
4.3 仮想ポート入力、仮想ポート出力、仮想割り込みの表示形式を変更するには 169
4.4 表示画面のスケールを変更するには 170
4.5 表示画面のカラーを変更するには 171
4.6 表示データを検索するには 172
4.7 登録した I/O スクリプトファイルの一覧表示 173
4.8 設定した仮想ポート入力、仮想割り込み、I/O スクリプトファイルの評価タイミングについて 173

5 GUI 入力ウィンドウの設定 174

5.1 概要 174
5.2 ボタンの作成方法 174
5.3 作成したボタンを保存するには 176
5.4 ボタン作成後にボタンの配置、設定を変更するには 177
5.5 ボタンをコピーするには 178
5.6 ボタンを削除するには 178
5.7 グリッド線を表示するには 178

6 GUI 出力ウィンドウの設定 179

6.1 概要 179
6.2 ラベルの作成方法 180
6.3 LED の作成方法 182
6.4 作成したパーツを保存するには 184
6.5 パーツ作成後にパーツの配置、設定を変更するには 185
6.6 パーツをコピーするには 185
6.7 パーツを削除するには 186
6.8 グリッド線を表示するには 186

7 I/O スクリプト機能 187

7.1 概要 187
7.2 I/O スクリプトの記述方法 187
7.3 I/O スクリプトの構成要素 188
7.4 右辺式の記述方法 192
7.5 左辺式の記述方法 195

8 カバッレッジ情報 197

8.1 カバレッジを参照するには 197

8.2 カバレッジの表示を更新するには 197
8.3 カバレッジを初期化するには 197
8.4 カバレッジ計測情報を保存／読み込みするには 198

9 カスタマイズ機能 199

9.1 カスタマイズ機能とは 199

リファレンス編 203

1 スクリプトコマンド一覧 205

1.1 入力書式 205
1.2 コマンド一覧 206

2 スクリプトファイルの記述方法 215

2.1 スクリプトファイルの構成要素 215
2.2 式の記述方法 217

3 C言語式について 221

3.1 C言語式の記述方法 221
3.2 C言語式の表示形式 224

4 エラーメッセージ一覧 229

索引 241

概要編

このページは白紙です。

1 PD38SIMの概要

PD38SIMは、8 ビットマイクロコンピュータ 740 ファミリの動作をシミュレートして、そのターゲットプログラムの評価を行うためのシミュレータデバッガ［Windows 対応版］です。

PD38SIMは、以下のソフトウェアから構成されています。

1. pd38sim（シミュレータデバッガフロントエンド）
2. sim38（シミュレータエンジン）

2 PD38SIMの特長

2.1 マルチウィンドウ機能

PD38SIMは、オーバーラッピング形式のマルチウィンドウ機能をサポートし、様々な情報を同時に表示します。各ウィンドウは、メニュー、ボタン等を備えており、これらをマウスで操作することによってコマンドを実行できます。

2.2 I/O シミュレーション機能

PD38SIMは、以下の I/O シミュレーション機能を用意しています。

●仮想ポート入力機能
外部からメモリに入力するデータの変化を I/O ウィンドウやファイルで定義できます。
●仮想ポート出力機能
プログラムがメモリに出力するデータを記録できます。
記録したデータの変化をグラフ表示できます。

2.3 割り込みシミュレーション機能

ソフトウェア割り込みを定義することができます。発生させる割り込みは、I/O ウィンドウやファイルで定義できます。

2.4 簡易システムシミュレーション機能

●GUI 入力機能
GUI でキーマトリクスを定義できます。
●GUI 出力機能
GUI で LED 等を定義できます。

2.5 RAM モニタ機能

ターゲットプログラム実行中のメモリ内容が参照できる RAM モニタ機能をサポートしています。この機能を用いて参照できるメモリ領域を RAM モニタ領域と呼びます。PD38SIMは、1K バイトの RAM モニタ領域を備えており、この RAM モニタ領域は、任意のアドレスに配置できます。RAM モニタ領域の参照は、RAM モニタウィンドウで行うことができます。

2.6 ブレーク機能

PD38SIMは、以下の 2 種類のブレーク機能を用意しています。

2.6.1 ソフトウェアブレーク

ソフトウェアブレークとは、指定アドレスの命令を実行する手前でブレークする機能のことです。このブレークが行われるポイントを、ソフトウェアブレークポイントといいます。ソフトウェアブレークポイントの設定は、S/W ブレークポイント設定ダイアログで行います。また、プログラムウィンドウやソースウィンドウからも簡単にソフトウェアブレークポイントが設定できます。PD38SIMでは、64点のソフトウェアブレークポイントが設定できます。

S/W ブレークポイント設定ダイアログから設定したソフトウェアブレークポイントの保存や読み込みが可能です。

2.6.2 ハードウェアブレーク

ハードウェアブレークとは、メモリのデータ書き込み／読み込み／命令フェッチを検出したときにブレークする機能のことです。このブレークが行われるポイントを、ハードウェアブレークポイントといいます。ハードウェアブレークポイントの設定は、H/W ブレークポイント設定ダイアログで行います。PD38SIMでは、64点のハードウェアブレークポイントが設定できます。

2.7 ソースレベルデバッグ機能

ソースファイルを表示し、ソース行でのブレークポイント指定やステップ実行など、ソースレベルのデバッグが行えます。PD38SIMでは、C 言語及びアセンブリ言語レベルでのデバッグが可能です。

- ソースファイルは、プログラムウィンドウ及びソースウィンドウで参照できます。
- C 言語ソースファイルに記述した C 変数や C 言語式は、C ウォッチウィンドウ、ローカルウィンドウ、ファイルローカルウィンドウ、グローバルウィンドウで参照できます。
- アセンブラソースファイルに記述したラベルやシンボルは、ASM ウォッチウィンドウで参照できます。
- ブレークポイント等のアドレス指定には、ラベル名やシンボル名が指定できます。これらから値への変換は、ローカル、グローバルの順に行います。C 言語ソースファイルで定義した変数（関数）名を指定する場合は、変数（関数）名の前にアンダーバー“_”を追加します。

2.8 オンデマンド方式

PD38SIMでは、ターゲットプログラムをダウンロードした際に、テンポラリファイルを作成し、必要なデバッグ情報を必要になった時にメモリ上に読み込む「オンデマンド方式」をサポートしています。これにより、メモリの使用量を削減することができます。なおデフォルトは、メモリ上に全てのデバッグ情報を保持する「オンメモリ方式」です。

「オンデマンド方式」と「オンメモリ方式」のどちらを使用するかは、pd38simの動作環境の設定で選択することができます。pd38simの動作環境の設定については、本マニュアル セットアップ編の項目「1.3 pd38simの動作環境の設定」をご参照下さい。

テンポラリファイルは、pdb_xxxx.tmp (xxxx は 16 進 4 桁の数値）というファイル名で生成します。このファイルは、ダウンロードの直前およびPD38SIM終了時に削除されます。

テンポラリファイルを作成するディレクトリは、Init ダイアログで指定したディレクトリに作成します。指定したディレクトリにテンポラリファイルと同名のファイルがある場合は、テンポラリファイルを作成するディレクトリを変更、または「オンメモリ方式」を選択してください。

3 PD38SIMのシミュレーション仕様

3.1 実チップとの主な相違点

以下に、PD38SIMと実チップとの主な相違点を述べます。
詳細については後述の各項目をご参照下さい。

3.1.1 実時間のタイミング

PD38SIMの時間管理は、サイクル単位で行われます。但し、実チップとは、次の点が異なります。サイクル数は「740 ファミリソフトウエアマニュアル」に記載されている値を使用しています。

- サイクル数の計測は、リセット直後から開始し（リセット直後のサイクル数は 0 になります）機械語 1 命令の実行に要したサイクル数を命令実行後に加算していきます（下図 3.1 を参照下さい）。

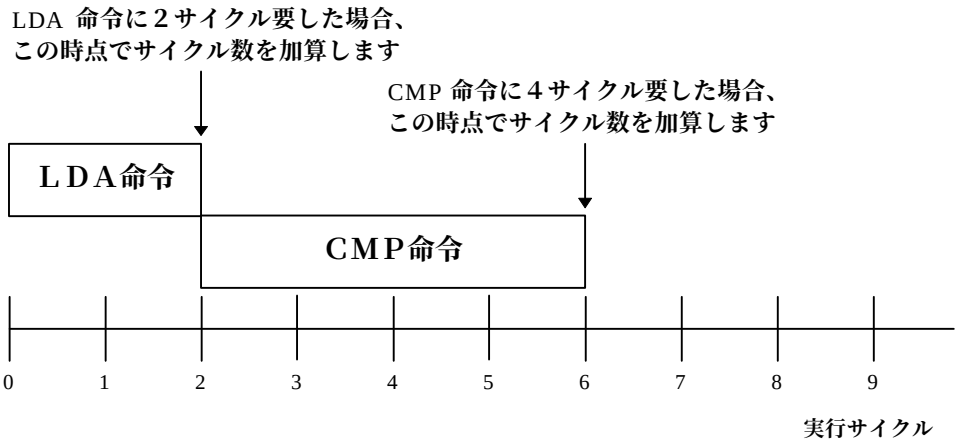


図３．１　サイクルの計測方法

上記の例では、LDA 命令や CMP 命令等が実行されている間は、サイクル数は加算されません。命令の実行後にサイクル数が加算されます。
なお、仮想ポート入出力機能や仮想割り込み機能は命令実行後のタイミングで処理されます。

3.1.2 周辺I／O

実チップの CPU コア以外の、タイマ、シリアル I/O などの周辺 I/O はサポートしていません。周辺 I/O が接続される SFR 領域も PD38SIM では RAM として扱います。

ただし CPU モードレジスタのスタックページ選択ビットおよび割り込み制御レジスタは、それぞれ SFR として扱います(各レジスタの配置は、チップユーザーズマニュアルを参照下さい)。

スタックページ選択ビットを 1 にした時、1 ページ内の RAM をスタック領域として使用することができます。

割り込み制御レジスタのビットを 1 にした時、そのビットに対応する割り込み発生が許可されます。

また、タイマ割り込みなどの割り込みや、SFR 等のメモリへのデータ入力を擬似的に実現する方法を用意しています。この方法に関しては、後述の仮想ポート入出力機能、仮想割り込み機能を参照下さい。

3.1.3 メモリ空間

プロセッサモードはありません。0000_{16}〜$FFFF_{16}$の 64KB のメモリ空間全てが RAM として割り当てられています。

3.1.4割り込み

実チップでは、周辺 I／O（外部の割り込み信号も含む）が割り込みの発生要因となりますが、PD38SIM では周辺 I／O に該当するものがありません。

PD38SIMではこれに代わるものとして、擬似的に割り込みをかける機能（仮想割り込み）を用意しています。仮想割り込みは、指定されたサイクルや実行アドレス等のタイミングで発生させることができます。

3.2 命令動作

3.2.1 WIT、STP 命令

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

3.3 リセット動作

- S レジスタは $0FF_{16}$、PS レジスタは I フラグのみ"1"となり、プログラムカウンタ値はリセットベクタの値になります。それ以外のレジスタは 0 で初期化されます。
- SFR 領域はPD38SIMにはありませんので、実チップのような初期化は行われません。
- 実行サイクル数は0になります。

またPD38SIM起動時にもリセット動作を行います。

3.4 メモリ

3.4.1 メモリの種類

メモリ空間全域（0000_{16} ～ $FFFF_{16}$）は、すべて RAM として動作し、起動時 全ての領域にメモリが割り当てられています。

3.4.2 起動直後のメモリ構成とその初期値

起動直後は、次のようにメモリが設定されています。

0000_{16} ～ $FFFF_{16}$　　00_{16} で埋められています。

3.5 仮想ポート入力機能

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

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

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

3.6 仮想ポート出力機能

仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。
記録されたデータは、I/O ウィンドウでグラフや数値形式で確認できます。
本機能で記録できるデータの個数は、プログラム実行開始時から最大 30000 データです。

3.7 仮想割り込み機能

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

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

上記のタイミングにおける仮想割り込みの定義は、I/O ウィンドウで行うことができます。
なお、I/O スクリプト機能（仮想ポート入力や仮想割り込みをユーザが定義できる機能。後述の「より高度なデバッグ編」を参照下さい）を利用すると、プログラムがメモリをリードあるいはライトした時、命令を指定回数実行した時等、さらに詳細な割り込みの発生タイミングを指定することができます。

3.7.1 仮想割り込みと実チップの割り込みの相違点

仮想割り込みは、実チップの割り込みと以下の点が異なります。

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

2. リセット割り込みは、発生させることができません。

3.8 GUI 入力機能

GUI 入力機能とは、ユーザターゲットシステムの簡単なキー入力パネル（ボタン）をウィンドウ上で実現する機能です。キー入力パネルの作成は、GUI 入力ウィンドウで行います。

GUI 入力ウィンドウに作成したボタンを押すことにより、仮想ポート入力や仮想割り込みを行うことができます。ボタンには以下のアクションが設定できます。

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

3.9 GUI 出力機能

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

出力パネルには、以下のパーツが配置できます。

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

3.10 I/O スクリプト機能

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

I/O スクリプトの詳細な仕様に関しては、「より高度なデバッグ編」を参照下さい。

3.11 シミュレータ固有の機能

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

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

スタック使用量計測は、Go あるいは GoFree コマンド発行から中断までに行われ、スタックポインタ（S レジスタ）の最大値・最小値が記録されます。

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

3.11.2 サイクル数計測機能･･･ CYcle（CY）コマンド

CYcle コマンドを使用することで、実行したプログラムのおおよそのサイクル数を知ることができます。サイクル数は「740 ファミリソフトウエアマニュアル」に記載されている値を使用しています。

4 PD38SIMの入出力ファイル

4.1 入力ファイル

PD38SIMが取り扱う入力ファイルは、以下の通りです。

4.1.1 IEEE-695 アブソリュート形式ファイル

IEEE-695 アブソリュート形式ファイルは、ソースファイルで使用している C 言語変数情報や行情報等のデバッグ情報及び機械語情報を格納しているファイルです。ファイル属性は、".695"です。IAR 社製 C コンパイラicc740により生成されます。IEEE-695 アブソリュート形式ファイルの作成方法につきましては、icc740 のユーザーズマニュアルを御参照ください。

4.1.2 インテルHEX フォーマットファイル

インテル HEX フォーマットファイルは、機械語情報を格納しているファイルです。ファイル属性は、".hex"です。アセンブラsra74用のリンケージエディタ**link74**により生成されます。

4.1.3 シンボルファイル

シンボルファイルは、ソースファイルで使用しているシンボル情報、行番号情報等のデバッグ情報を格納しているファイルです。ファイル属性は、".sym"です。アセンブラsra74用のリンケージエディタ**link74**により生成されます。

4.1.4 レジスタ情報ファイル

ターゲット MCU のレジスタに関する情報（レジスタ名・サイズ等）を持つ情報ファイルです。ファイル名は"PD38SIM.rdf"です。PD38SIMは本ファイルを自動的に読み込み、その情報をレジスタウィンドウの表示に使用します（本ファイルがないとレジスタウィンドウを開くことはできません）。本ファイルはPD38SIMに添付されています。ユーザが編集することはできません。

4.1.5 スクリプトファイル

スクリプトファイルは、スクリプトコマンドを自動実行するためのファイルです。スクリプトファイルは、スクリプトウィンドウから読み込みます。ファイル属性は、".scr"です。

4.1.6 ヘルプファイル

ヘルプファイルは、PD38SIMのヘルプメッセージを含んだファイルです。ファイル属性は、".hlp"です。ヘルプファイルは、PD38SIMに添付されています。

4.1.7 環境設定ファイル

環境設定ファイルは、PD38SIMの環境設定に関する情報を保持するファイルです。環境設定ファイルは、PD38SIMが自動的に生成します。ファイル名は、pd38sim.ini です。このファイルは、Windows ディレクトリ（ご使用の Windows がインストールされたディレクトリ）に保存され、ユーザ自身が作成／編集することはできません。

4.1.8 SFR ファイル

ターゲット MCU の固有情報を持つ情報ファイルです。SFR ファイルは、製品に付属されています。ファイル名は、M3xxxx.SFR です。

4.1.9 カバレッジ計測情報ファイル

カバレッジ計測情報ファイルは、カバレッジ計測の結果を格納するバイナリファイルであり、ファイル属性は、“.cov”です。このファイルは、カバレッジウィンドウで保存／読み込みできます。

4.1.10 ＡＳＭウォッチポイント情報格納ファイル

ＡＳＭウォッチポイント情報格納ファイルは、ＡＳＭウォッチウィンドウに登録されている、ＡＳＭウォッチポイントの情報を保持するファイルです。ファイル属性は”.wpt”です。本ファイルは、ＡＳＭウォッチウィンドウから読み込みます。

4.1.11 Ｃウォッチポイント情報格納ファイル

Ｃウォッチポイント情報格納ファイルは、Ｃウォッチウィンドウに登録されているＣウォッチポイントの情報を保持するファイルです。 Ｃウォッチポイント情報格納ファイルは、PD38SIMが自動的に生成します。ファイル属性は、“.cwp”です。このファイルは、Windows ディレクトリ（ご使用の Windows がインストールされたディレクトリ）に保存され、ユーザ自身が作成／編集することはできません。

4.1.12 ソフトウェアブレークポイント保存ファイル

ソフトウェアブレークポイント保存ファイルは、ソフトウェアブレークポイントの設定内容を保存しているファイルです。ファイル属性は、“.brk”です。

本ファイルを読み込むことにより、ソフトウェアブレークポイントを設定することができます。本ファイルは、ソフトウェアブレークポイント設定ダイアログから読み込みます。

4.1.13 I/O スクリプトファイル

I/O スクリプトファイルは、仮想ポート入力、仮想割り込みを記述したファイルです。作成した I/O スクリプトファイルは、I/O ウィンドウから読み込みます。ファイル属性は、“.scr”です。

4.1.14 GUI 入力ファイル

GUI 入力ファイルは、GUI 入力ウィンドウで作成したキーパネルの定義を保存しているファイルです。ファイル属性は、“.btn”です。

本ファイルを GUI 入力ウィンドウから読み込むことにより、作成したキーパネルを再設定することができます。

4.1.15 GUI 出力ファイル

GUI 出力ファイルは、GUI 出力ウィンドウで作成した出力パネルの定義を保存しているファイルです。ファイル属性は、“.gof”です。

本ファイルを GUI 出力ウィンドウから読み込むことにより、作成した出力パネルを再設定することができます。

4.2 出力ファイル

PD38SIMが取り扱う出力ファイルは、以下の通りです。

4.2.1 インテルHEX フォーマットファイル

インテル HEX フォーマットファイルは、機械語情報を格納しているファイルです。ファイル属性は、“.hex”です。PD38SIMのアップロード機能により保存できます。保存したインテル HEX フォーマットファイルは、PD38SIMで再ダウンロードすることもできます。

4.2.2 逆アセンブルファイル

逆アセンブルファイルは、プログラムメモリの逆アセンブル結果を保存したファイルです。ファイル属性は、“.txt”です。逆アセンブルファイルは、参照用のテキストファイルです。逆アセンブルファイルの再アセンブル／再ダウンロードはできません。

4.2.3 ログファイル

スクリプトコマンドの実行結果を保存するテキストファイルです。ログファイルは、ログオンからログオフするまでのコマンド実行結果を格納しています。ファイル属性は、“.log”です。

4.2.4 ビューファイル

スクリプトウィンドウの表示内容を保存したテキストファイルです。PD38SIMでは、スクリプトコマンド実行結果の最新 1000 行分をビューバッファといわれる領域に格納しています。ビューファイルは、このビューバッファの内容を格納するためのファイルです。ファイル属性は、「.viw」です。

4.2.5 ＡＳＭウォッチポイント情報格納ファイル

ＡＳＭウォッチポイント情報格納ファイルは、ＡＳＭウォッチウィンドウに登録されている、ＡＳＭウォッチポイントの情報を保持するファイルです。ファイル属性は”.wpt”です。本ファイルは、ＡＳＭウォッチウィンドウから保存します。

4.2.6 カバレッジ計測情報ファイル

カバレッジ計測情報ファイルは、カバレッジ計測の結果を格納するバイナリファイルであり、ファイル属性は、“.cov”です。このファイルは、カバレッジウィンドウで保存／読み込みできます。

4.2.7 ソフトウェアブレークポイント保存ファイル

ソフトウェアブレークポイント保存ファイルは、ソフトウェアブレークポイントの設定内容を保存しているファイルです。ファイル属性は、“.brk”です。

本ファイルは、ソフトウェアブレークポイント設定ダイアログから保存します。

4.2.8 I/O スクリプトファイル

I/O スクリプトファイルは、I/O ウィンドウで設定、あるいはユーザが作成した仮想ポート入力、仮想割り込みの定義内容を保存しているファイルです。

4.2.9 GUI 入力ファイル

GUI 入力ファイルは、GUI 入力ウィンドウで作成したキーパネルの定義を保存しているファイルです。ファイル属性は、".btn"です。
本ファイルを GUI 入力ウィンドウから保存します。

4.2.10 GUI 出力ファイル

GUI 出力ファイルは、GUI 出力ウィンドウで作成した出力パネルの定義を保存しているファイルです。ファイル属性は、".gof"です。
本ファイルを GUI 出力ウィンドウから保存します。

4.2.11 仮想ポート出力ファイル

仮想ポート出力ファイルは、I/O ウィンドウで指定した仮想ポート出力の結果を保存したファイルです。
本ファイルは、I/O ウィンドウが仮想ポート出力の結果を表示する際に参照します。

4.3 テンポラリファイル

4.3.1 オンデマンド方式指定時に作成するファイル

ターゲットプログラムをダウンロードする際に、デバッグ情報をオンデマンド方式で読み込むように指定していた場合、テンポラリファイルが作成されます。テンポラリファイルは、Init ダイアログで指定されたディレクトリに作成します。ディレクトリの指定がない場合は、ダウンロードしたファイルがあるディレクトリに作成します。
テンポラリファイルはファイル名 pdb_xxxx.tmp (xxxx は 16 進 4 桁の数値になります）です。このファイルは、ダウンロードの直前およびPD38SIM終了時に削除されます。

4.3.2 I/O ウィンドウが作成するファイル

I/O ウィンドウで仮想ポート入力や仮想割り込み、及び I/O スクリプトファイルの設定を行う際にテンポラリファイルが作成されます。テンポラリファイルは、仮想ポート入力や仮想割り込みの設定を保存したファイルを読み込んだディレクトリ、あるいは I/O スクリプトファイルを読み込んだディレクトリに作成します。従って、それらのディレクトリにファイルを書き込むアクセス権がないと、I/O ウィンドウの [Option]→[Load]メニューでファイルを読み込む際にエラーとなります。

セットアップ編

このページは白紙です。

1. セットアップ

1.1. インストール

PD38SIMのインストール方法については、製品に付属しているリリースノートをご参照下さい。

1.2. PD38SIMの起動

PD38SIMを起動するには、以下の操作を行ってください。

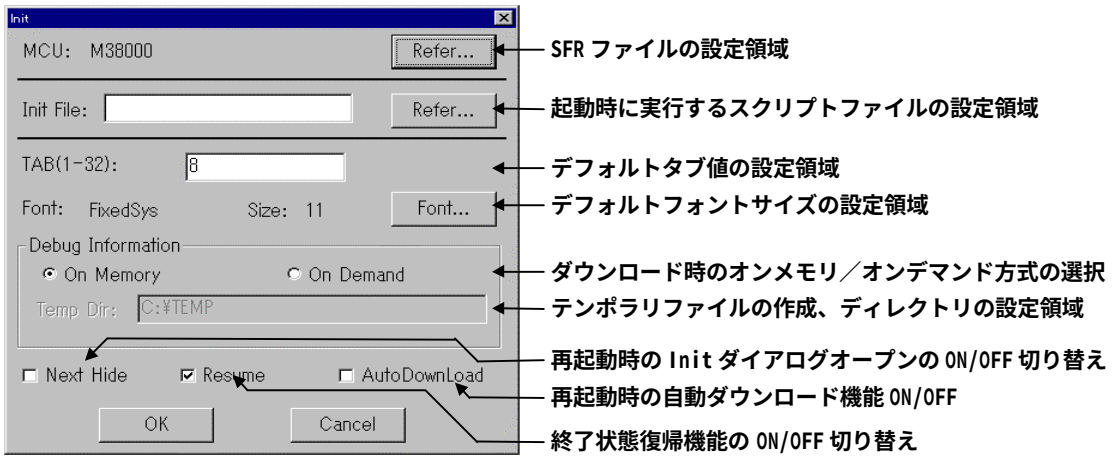
スタートボタンをクリックし、
→ プログラム(P)
→［RENESAS - TOOLS］
→［PD38SIM　V.X.XX ReleaseX］
→［PD38SIM］　を選択してください。

上記操作を行うとpd38sim（シミュレータデバッガフロントエンド）が起動し、同時にsim38（シミュレータエンジン）も起動します。すでにsim38が起動している場合は、そのままpd38simを起動します。

1.3. pd38simの動作環境の設定

pd38simを起動すると、Initダイアログがオープンします。pd38simの動作環境は、このInitダイアログで設定します。

1.3.1. Initダイアログの構成



1.3.2. 動作環境の設定

SFRファイルの設定

ターゲットMCUに対応するSFRファイルを指定します。”Refer”ボタンをクリックすると、ファイルセレクションダイアログがオープンします。対応するSFRファイルを指定してください。対応するSFRファイルが存在しない場合、新規に作成する必要があります。SFRファイルの作成方法については、本製品パッケージに添付されているリリースノートをご参照ください。

起動時に実行するスクリプトファイルの設定

起動時、スクリプトコマンドを実行する場合に設定します。実行するスクリプトコマンドは、あらかじめスクリプトファイルに記述してください。“Refer”ボタンをクリックするとファイルセレクションダイアログがオープンします。起動時に実行するスクリプトファイル名をマウスで選択してください。選択されたスクリプトファイルは、Initダイアログの Init File:領域に表示されます。

デフォルトタブ値の設定

プログラムウィンドウ、ソースウィンドウのデフォルトタブ値を設定します。タブ値には、1～32までの数値が指定できます。

補足

タブ値は、ウィンドウ毎に設定することもできます。この場合、対象ウィンドウがアクティブな状態でPD38SIMウィンドウのメニュー［Option］→［TAB］を選択してください。TABダイアログがオープンしますのでタブ値を指定してください。

デフォルトフォントサイズの設定

pd38simで表示する文字のデフォルトフォントを指定します。“Font”ボタンをクリックするとフォント指定ダイアログがオープンしますのでフォントとフォントサイズを指定してください。

補足

フォントサイズは、ウィンドウ毎に設定することもできます。この場合、対象ウィンドウがアクティブな状態でPD38SIMウィンドウのメニュー［Option］→［Font］を選択してください。フォント指定ダイアログがオープンしますのでフォントとフォントサイズを指定してください。

ダウンロード時のオンメモリ／オンデマンド方式の選択

ターゲットプログラムをダウンロードした際に、デバッグ情報をオンメモリ方式で読み込むか、オンデマンド方式で読み込むかを選択します。オンメモリ方式とは、メモリ上に全てのデバッグ情報を保持する方式です。オンデマンド方式とは、ターゲットプログラムをダウンロードした際に、テンポラリファイルを作成し、必要なデバッグ情報を必要になった時にメモリ上に読み込む方式です。

なお、PD38SIMウィンドウのメニュー［Environ］→［Init］を選択して Init ダイアログから設定した場合、その設定は次回のダウンロード時に有効になります。

テンポラリファイル作成ディレクトリの設定

PD38SIMでは、オンデマンド方式を選択してダウンロードした際、テンポラリファイルを作成します。PD38SIMでは、指定したディレクトリにテンポラリファイルを作成します。ディレクトリの指定が無い場合は、ダウンロードしたファイルがあるディレクトリに作成します。

再起動時の Init ダイアログオープンの ON/OFF

PD38SIM再起動時に Init ダイアログをオープンするか否かを指定します。Next Hide をチェックした場合、次回から Init ダイアログがオープンしません。Init ダイアログを再オープンさせるには、PD38SIMウィンドウのメニュー［Environ］→［Init］を選択し、Init ダイアログから Next Hide のチェックを解除してください。または、Ctrl キーを押しながらpd38simを起動すると、起動時に強制的に Init ダイアログを表示することができます。

終了状態復帰機能の ON/OFF

前回終了したときのウィンドウの表示状態でPD38SIMを起動するか否かを指定します。Resume をチェックした場合、前回終了したときとおなじウィンドウの表示状態でPD38SIMが起動します。

再起動時の自動ダウンロード機能

最後に読み込んだターゲットプログラムをPD38SIM起動時に読み込むか否かを指定します。AutoDownLoad をチェックした場合、PD38SIM起動時にターゲットプログラムを読み込みます。

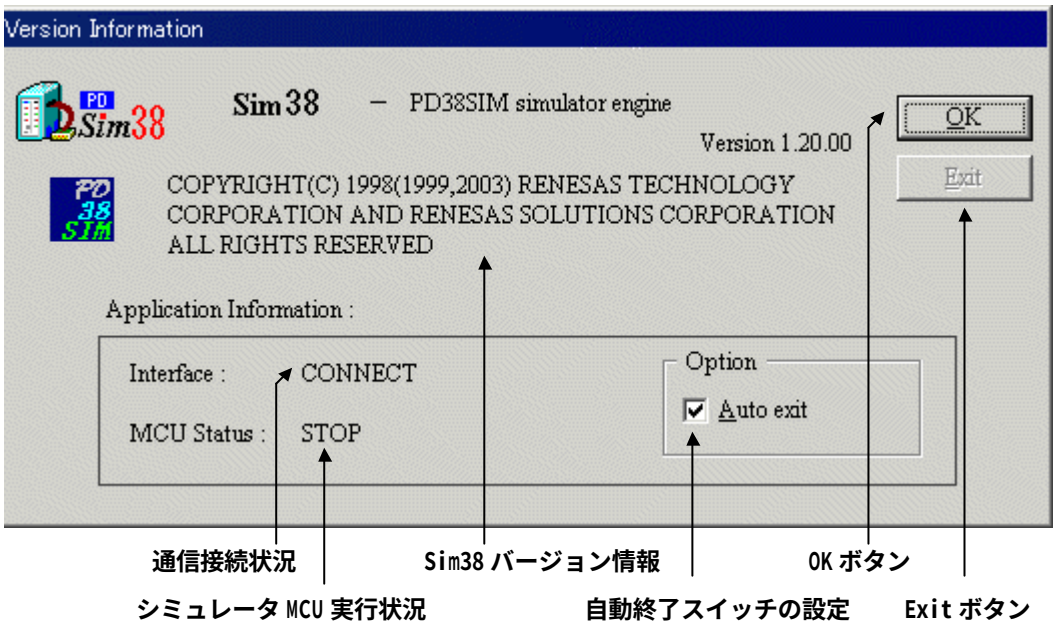
1.4. sim38の動作環境の設定

sim38は、起動すると、システムトレイに登録されます。

1.4.1. 起動中のsim38

起動中のsim38のアイコンを右クリックしてメニュー[Version...]を選択すると、Version Information ダイアログがオープンします。sim38の動作環境は、この Version Information ダイアログで設定します。

1.4.2. Version Information ダイアログの構成



1.4.3. 動作環境の設定

自動終了スイッチの設定

Auto exit チェックボックスをチェックすることにより、pd38simの終了と同時にsim38を終了させることができます。

通信接続状況

pd38simとの接続時には CONNECT を、切断時には CUT を表示します。

シミュレータ MCU 実行状況

シミュレータ MCU の実行時には RUN を、停止時には STOP を表示します。

OK ボタン

Version Information ダイアログをクローズします。

Exit ボタン

sim38を終了します。なおsim38は、pd38simと接続時には終了することができません。

【 MEMO 】

ウィンドウ機能編

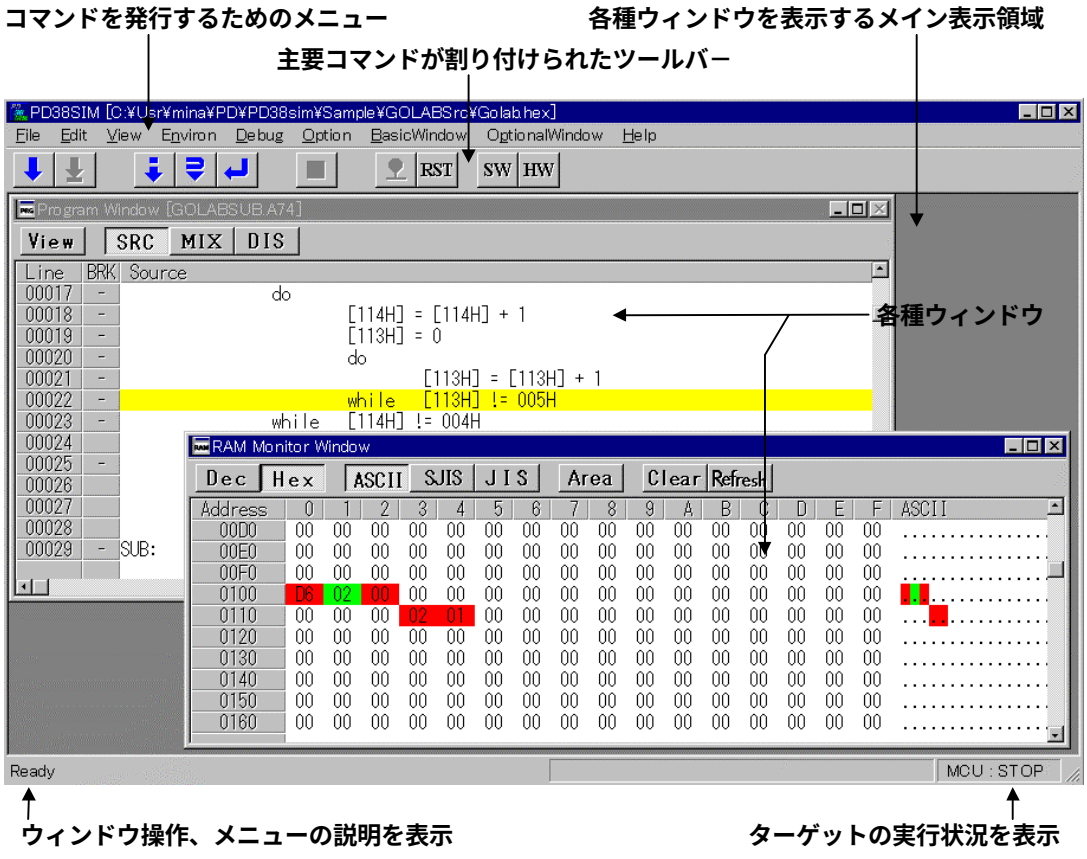
このページは白紙です。

1 PD38SIMのウィンドウ機能

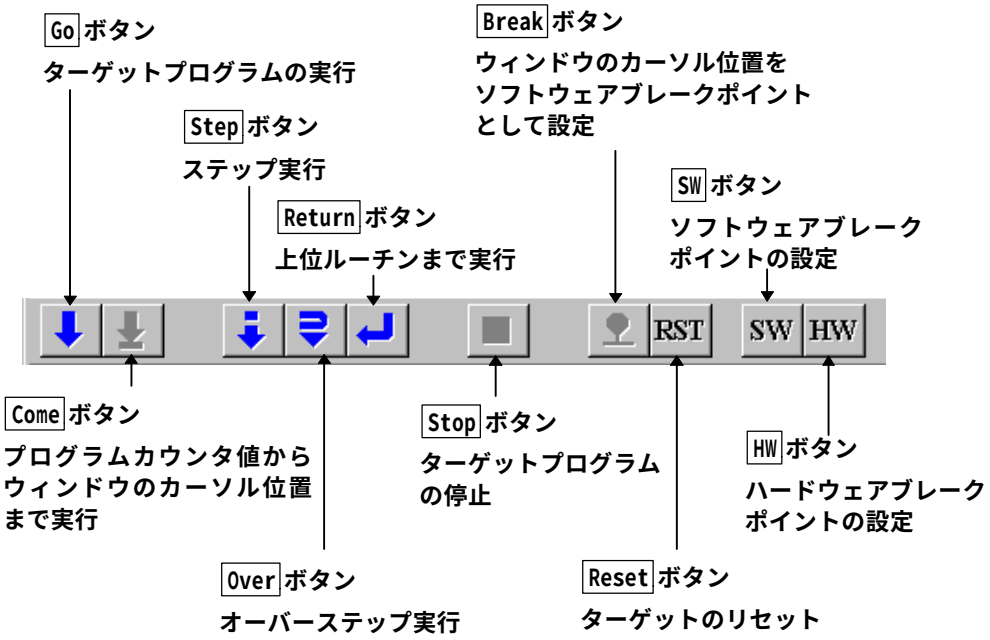
1.1 PD38SIMウィンドウ

PD38SIMウィンドウは、PD38SIMのメインウィンドウです。PD38SIMを起動した際に、最初にオープンします。PD38SIMウィンドウでは、主要コマンドをツールバーに割り付けています。ツールバーのボタンをクリックすることにより、ターゲットプログラムの実行、ステップ実行等が容易に操作できます。また、メイン表示領域には、ターゲットプログラム表示用ウィンドウ等の各種ウィンドウを表示します。

1.1.1 PD38SIMウィンドウの画面構成

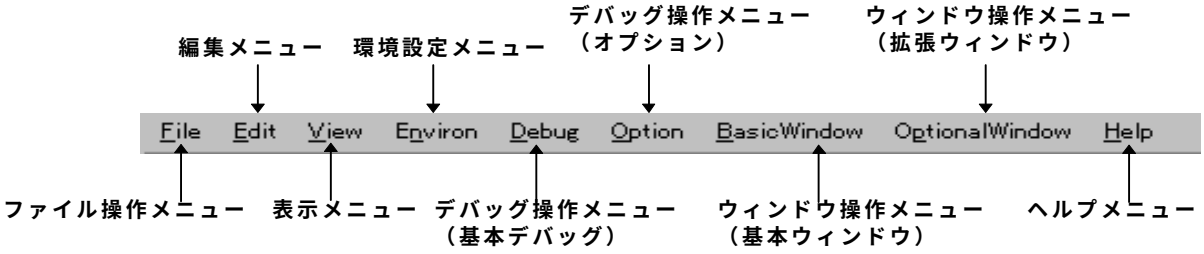


1.1.2 PD38SIMウィンドウのツールバー



1.1.3 PD38SIMウィンドウのメニュー

PD38SIMウィンドウのメニューは、基本メニューと拡張メニューに分類することができます。



「基本メニュー」と「拡張メニュー」について

PD38SIMのメニュー項目のうち Option メニューについては、PD38SIMウィンドウのメイン表示領域に表示されたアクティブなウィンドウに応じてサブメニュー項目が自動的に切り替わります。この Option メニューのことを「拡張メニュー」と呼びます。

これに対し、Option メニュー以外のすべてのメニューは、アクティブなウィンドウの変化によらずメニュー項目が常に一定です。これらを「基本メニュー」と呼びます。基本メニューは、PD38SIMの基本操作およびデバッグ操作を行うためのメニューを備えています。

以下に、基本メニューの各項目の機能について説明します。なお、拡張メニューの各項目の機能については、各ウィンドウで説明します。

1.1.3.1 ファイル操作

［File］メニューには、PD38SIMの機能のうち、ファイルの読み込みや保存、PD38SIMの終了等、ファイル操作に関するメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
File	Download	ターゲットプログラムをダウンロード	—
	Load Module...	機械語データとデバッグ情報をダウンロード	Shift + F.1
	Memory Image...	機械語データのみダウンロード	—
	Symbol...	デバッグ情報のみダウンロード	—
	Reload...	ターゲットプログラムの再ダウンロード	—
	Upload...	ターゲットプログラムのアップロード	—
	AutoDownLoad…	ロードモジュール更新時の自動ダウンロード	—
	Save Disasm...	逆アセンブル結果の保存	—
	File Name	最近ダウンロードしたファイル	—
	Exit	PD38SIMを終了	—

1.1.3.2 編集

［Edit］メニューには、PD38SIMの機能のうち、文字列のコピー、ペースト、検索等、編集操作に関するメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
Edit	Copy	選択した文字列をクリップボードにコピー	Ctrl + C
	Paste	クリップボードの文字列を貼り付け	Ctrl + V
	Find...	文字列の検索	—

1.1.3.3 表示

［View］メニューには、PD38SIMの機能のうち、ツールバーやステータスバーの表示切り替えに関するメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
View	Tool Bar	ツールバーの表示／非表示の切り替え	—
	Status Bar	ステータスバーの表示／非表示の切り替え	—

1.1.3.4 環境設定

［Environ］メニューには、PD38SIMの機能のうち、環境設定に関するメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
Environ	Init...	環境設定	—
	Path...	ソースファイルのサーチパス	—
	Start Up...	スタートアップ関数の設定	—
	Customize…	カスタマイズダイアログオープン	—

1.1.3.5 デバッグ操作（基本デバッグ）

［Debug］メニューには、PD38SIMの機能のうち、ターゲットの実行／停止やステップ実行等、基本的なデバッグ操作に関するメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
Debug	Go	ターゲットプログラムの実行	—
	Go	現PCからの実行	F.1
	Go Option...	指定アドレスからの実行	—
	GoFree	ターゲットプログラムのフリーラン実行	—
	Come	カーソル位置までの実行	F.2
	Step	ステップ実行	—
	Step	一回のステップ実行	F.3
	Step Option...	指定回数のステップ実行	—
	Over	オーバーステップ実行	—
	Over	一回のオーバーステップ実行	F.4
	Over Option...	指定回数のオーバーステップ実行	—
	Return	現サブルーチンの復帰まで実行	F.5
	Break Point	ブレークポイントの設定	—
	S/W Break Point...	S/W ブレークポイント設定ダイアログをオープン	F.7
	H/W Break Point...	H/W ブレークポイント設定ダイアログをオープン	Shift + F.7
	Break	カーソル位置にソフトウェアブレークを設定／解除	—
	Reset	ターゲットのリセット	F.8
	Stop	ターゲットプログラムの実行停止	—
	Scope…	スコープ設定ダイアログオープン	—
	Entry…	メイクファイルの登録	—
	Make	ターゲットプログラムのメイク	—

1.1.3.6 デバッグ操作（オプション）

拡張メニューには、PD38SIMが表示する各ウィンドウを操作するためのメニューが割り当てられます。拡張メニューの下は、アクティブなウィンドウの変化によって、メニュー項目が変化します。拡張メニューの各項目の機能については、各ウィンドウの説明をご参照下さい。

メニュー	メニュー項目	機能	ショートカットキー
Option		（各ウィンドウの拡張メニューが追加されます。）	—

1.1.3.7 ウィンドウ操作

［Basic Window］メニューには、PD38SIMの機能のうち、PD38SIMが表示する各ウィンドウの表示形態を操作するメニューおよび基本的なウィンドウをオープンするメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
Basic Window	Cascade	ウィンドウを重ねて表示	—
	Tile	ウィンドウを並べて表示	—
	Arrange Icon	アイコンを整列	—
	Program Window	プログラムウィンドウをアクティブ	—
	Source Window	ソースウィンドウをオープン	—
	Register Window	レジスタウィンドウをオープン	—
	Memory Window	メモリウィンドウをオープン	—
	Dump Window	ダンプウィンドウをオープン	—
	RAM Monitor Window	RAM モニタウィンドウをオープン	—
	ASM Watch Window	ASM ウォッチウィンドウをオープン	—
	C Watch Window	C 言語レベルのウォッチウィンドウをオープン	—
	C Watch Window	C ウォッチウィンドウをオープン	—
	Local Window	ローカルウィンドウをオープン	—
	File Local Window	ファイルローカルウィンドウをオープン	—
	Global Window	グローバルウィンドウをオープン	—
	Script Window	スクリプトウィンドウをオープン	—

［Optional Window］メニューには、PD38SIMの機能のうち、より高機能なウィンドウをオープンするメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
Optional Window	IO Window	I/O ウィンドウをオープン	—
	GUI Input Window	GUI 入力ウィンドウをオープン	—
	GUI Output Window	GUI 出力ウィンドウをオープン	—
	Coverage Window	カバレッジウィンドウをオープン	—
	Custom Window		—
	Option	カスタムウィンドウの登録	—
	User definition menu	カスタムウィンドウのオープン	—

1.1.3.8 ヘルプ

［Help］メニューには、PD38SIMの機能のうち、PD38SIMのヘルプやバージョンを表示するためのメニューが割り当てられています。

メニュー	メニュー項目	機能	ショートカットキー
Help	Index	ヘルプ表示	—
	About...	PD38SIM のバージョン情報表示	—

1.2 プログラムウィンドウ

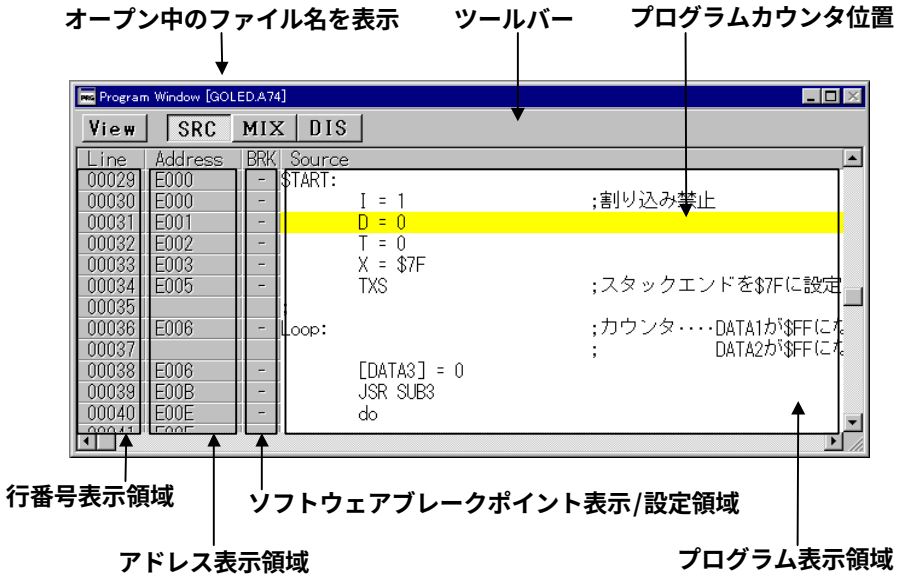
プログラムウィンドウは、現在のプログラムカウンタに相当するプログラムを常に表示するウィンドウです。プログラムカウンタに相当する行は、黄色の反転ラインで表示します。プログラムウィンドウは、PD38SIMを起動した際、PD38SIMウィンドウのメイン表示領域内に自動的にオープンします。プログラムウィンドウでは、カーソル位置までの実行、マウスによるソフトウェアブレークポイントの設定／解除、逆アセンブル表示等が行えます。ソフトウェアブレークポイントの設定／解除は、ソフトウェアブレークポイント表示／設定領域をダブルクリックすることによって行えます。プログラム表示領域をクリックしてメニュー［Option］→［Line Assemble］を選択することによりクリック位置からラインアセンブルを行うことができます。

1.2.1 プログラムウィンドウの画面構成

プログラムウィンドウには、ソース、逆アセンブルと両者を同時に表示する 3 種類の表示モードがあります。以下に３種類の表示モードの場合のプログラムウィンドウの画面構成を示します。

1.2.1.1 ソース表示モード時のプログラムウィンドウの画面構成

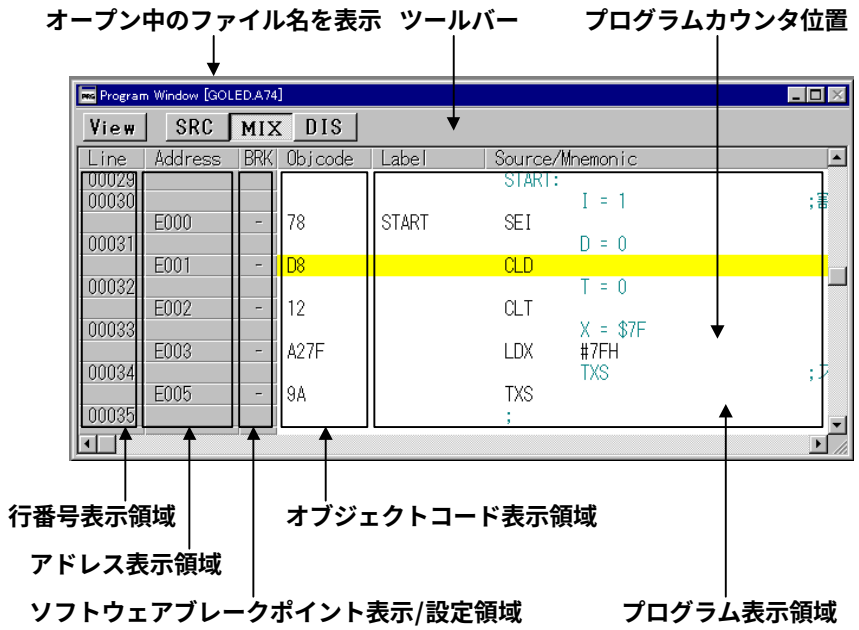
ソース表示モードは、ソースレベルでデバッグするためのモードです。ソース表示モードでは、ターゲットプログラムのソースファイルが参照できます。



- 行番号表示領域、アドレス表示領域は、それぞれメニュー［Option］→［Layout］→［Line Area］、［Option］→［Layout］→［Address Area］の選択／解除によって、表示／非表示にすることができます。なお、アドレス表示領域は、デフォルトでは非表示になっています。
- 行番号表示領域をダブルクリックすることで、表示するソースファイルを変更することができます。
- アドレス表示領域をダブルクリックすることで、表示開始アドレス、または表示開始行を変更することができます。
- プログラム表示領域で C 変数上にマウスカーソルを一定時間（約 0.5 秒）停止させると、変数の値を表示します。
- メニュー［Option］→［Coverage］を［On］にすると、カバレッジ計測結果を表示します。

1.2.1.2 MIX 表示モード時のプログラムウィンドウの画面構成

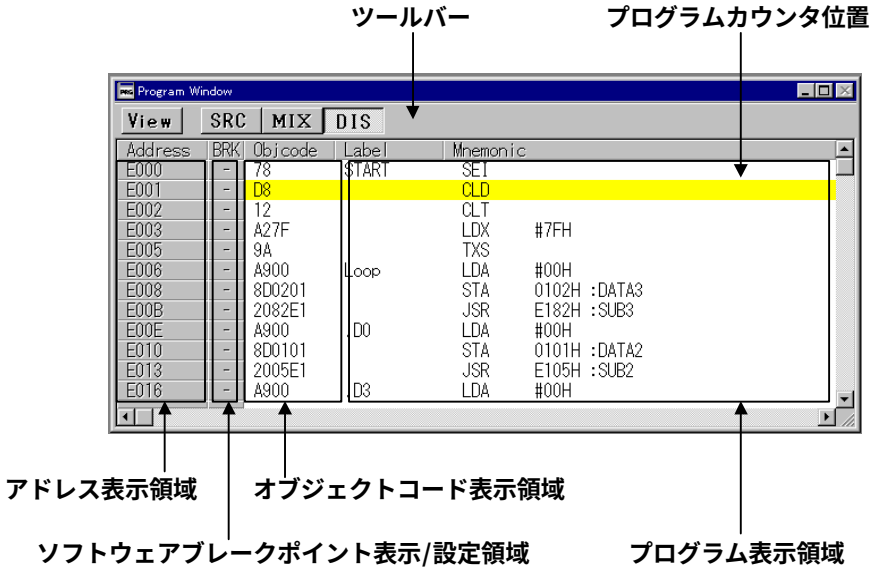
MIX 表示モードは、ソースプログラムとその部分の逆アセンブル結果を混合して表示します。ソースプログラムと逆アセンブル結果は異なった色で表示します。



- 行番号表示領域、アドレス表示領域、オブジェクトコード表示領域は、それぞれメニュー［Option］→［Layout］→［Line Area］、［Option］→［Layout］→［Address Area］、［Option］→［Layout］→［Code Area］の選択／解除によって、表示／非表示にすることができます。
- 行番号表示領域をダブルクリックすることで、表示するソースファイルを変更することができます。
- アドレス表示領域をダブルクリックすることで、表示開始アドレス、または表示開始行を変更することができます。
- メニュー［Option］→［Coverage］を［On］にすると、カバレッジ計測結果を表示します。

1.2.1.3 逆アセンブル表示モード時のプログラムウィンドウの画面構成

逆アセンブル表示モードは、命令レベルでデバッグするためのモードです。逆アセンブル表示モードでは、ターゲットプログラムの逆アセンブル結果が参照できます。



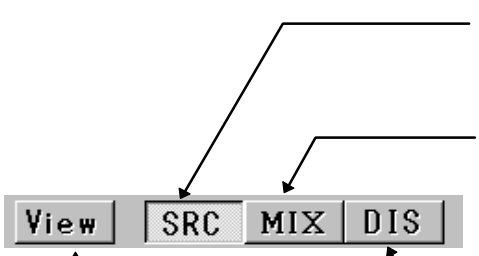
- アドレス表示領域、オブジェクトコード表示領域は、それぞれメニュー［Option］→［Layout］→［Address Area］、［Option］→［Layout］→［Code Area］の選択／解除によって、表示／非表示にすることができます。
- 垂直方向のスクロールをする場合、逆方向のスクロールは、順方向のスクロールの後でないと使用できません。順方向へのスクロールでは、以前の表示アドレスを内部バッファに保存します。逆方向へのスクロールは、このアドレス情報を利用して実現しています。
  なお、コマンド実行によって、先頭行アドレスが変更された場合、内部バッファの内容はクリアされます。
- プログラム以外の領域（データ・空き領域など）を逆アセンブルした場合、そのメモリ内容を命令コードとみなして逆アセンブル表示します。その際、未定義命令・未定義オペランドとなった場合、“ **???** ”を表示します。
- アドレス表示領域をダブルクリックすることで、表示開始アドレスを変更することができます。
- メニュー［Option］→［Coverage］を［On］にすると、カバレッジ計測結果を表示します。

1.2.2 プログラムウィンドウのショートカットメニュー

プログラムウィンドウ上で右クリックすると、ショートカットメニューがオープンし、選択関数へのジャンプや選択変数のＣウォッチウィンドウ登録等ができます。

メニュー	メニュー項目	機能	ショートカットキー
右クリック	Jump to function	選択した関数の表示	—
	Open Source Window	選択した関数の表示（新規にソースウィンドウを開く）	—
	Add C Watch…	マウスで選択した変数をＣウォッチに登録	—
	Add C Watch Pointer…	マウスで選択したポインタ変数をＣウォッチに登録	—
	Add ASM Watch…	マウスで選択した変数をASM ウォッチに登録	—
	Bit Add ASM Watch…	マウスで選択したビットシンボルをASM ウォッチに登録	—
	Open Editor	エディタのオープン	—
	Entry Editor…	エディタの登録	—
	Line Assemble…	ラインアセンブルダイアログのオープン	—

1.2.3 プログラムウィンドウのツールバー



SRC ボタン
プログラム表示をソース表示モードに変更します。

MIX ボタン
プログラム表示を MIX 表示モードに変更します。

DIS ボタン
プログラム表示を逆アセンブル表示モードに
変更します。

View ボタン
プログラムの表示箇所を変更します。ソースファイル名、関数名、
アドレス、プログラムカウンタ位置の何れかで指定できます。

1.2.4 プログラムウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、プログラムウィンドウがアクティブな場合は、［Option］メニューには以下の拡張メニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	TAB...	ソースファイル表示のタブ設定	—
	Color	表示色の変更	—
	View	表示内容の変更	—
	Source...	指定ソースファイル・関数からの表示	—
	Address...	指定アドレスまたは行番号からの表示	—
	Program Counter	現在のプログラムカウンタからの表示	—
	Mode	表示モードの変更	—
	Source mode	ソース表示モードへ変更	Ctrl + R
	Mix mode	MIX表示モードへ変更	Ctrl + R
	Disasm mode	逆アセンブル表示モードへ変更	Ctrl + R
	Layout	レイアウト設定	—
	Line Area	行番号表示領域の表示／非表示	—
	Address Area	アドレス表示領域の表示／非表示	—
	Code Area	オブジェクトコード表示領域の表示／非表示	—
	Line Assemble..	ラインアセンブルダイアログのオープン	Ctrl + L
	Coverage	カバレッジ表示設定	—
	On/Off	計測結果の表示／非表示	—
	Clear	計測結果のクリア	—
	Refresh	計測結果の再取得	—

1.3 ソースウィンドウ

ソースウィンドウは、特定のプログラムを継続して参照するためのウィンドウです。プログラムカウンタに相当する行は、黄色の反転ラインで表示します。前記のプログラムウィンドウがプログラムカウンタに追従して表示内容を更新するのに対し、ソースウィンドウは、ユーザが指定しない限り表示内容を更新しません。したがって、ソースウィンドウは、特定のサブルーチンやタスクを継続して参照される際にご使用下さい。ソースウィンドウは、計 10 枚までオープンできます。その他の機能は、プログラムウィンドウと同等です。

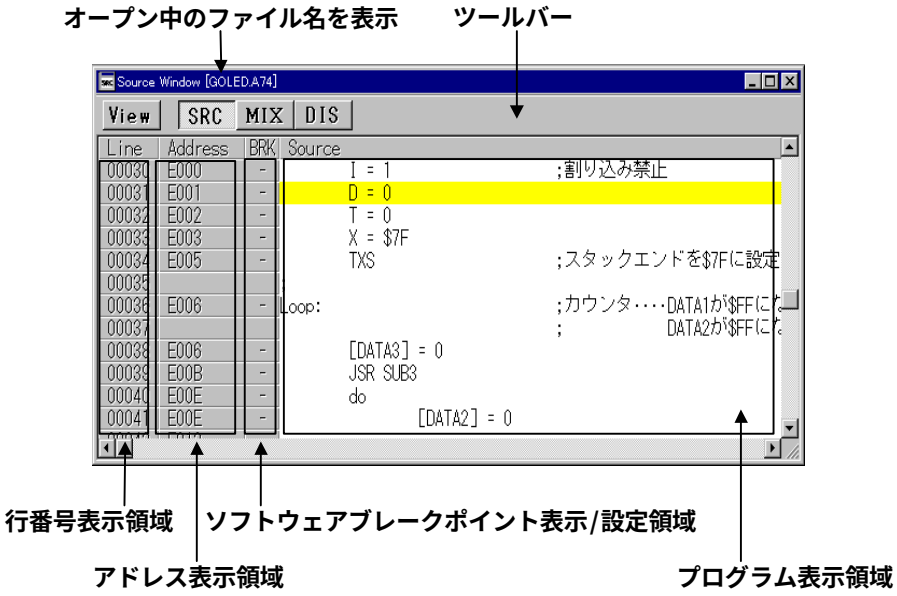
1.3.1 ソースウィンドウの画面構成

ソースウィンドウには、ソース、逆アセンブルと両者を同時に表示する 3 種類の表示モードがあります。以下に 3 種類の表示モードの場合のプログラムウィンドウの画面構成を示します。

なお、ソースウィンドウの画面構成は、プログラムウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.2.1 プログラムウィンドウの画面構成」をご参照下さい。

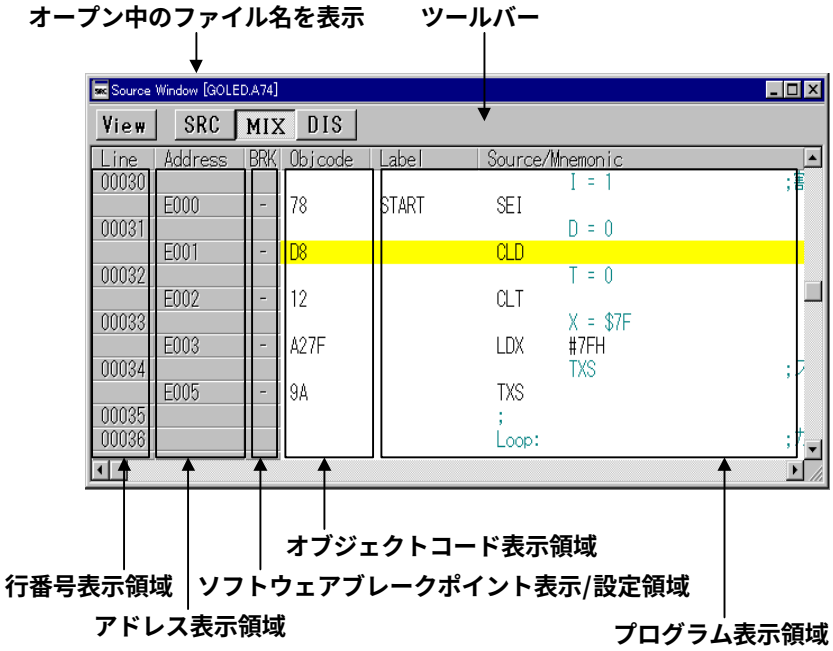
1.3.1.1 ソース表示モード時のソースウィンドウの画面構成

ソース表示モードは、ソースレベルでデバッグするためのモードです。ソース表示モードでは、ターゲットプログラムのソースファイルが参照できます。



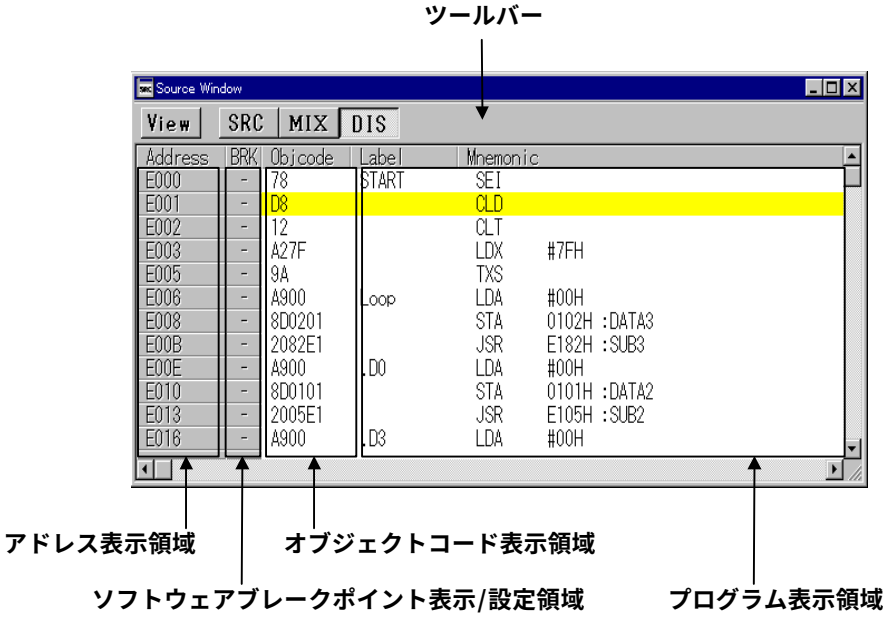
1.3.1.2 MIX 表示モード時のプログラムウィンドウの画面構成

MIX 表示モードは、ソースプログラムとその部分の逆アセンブル結果を混合して表示します。ソースプログラムと逆アセンブル結果は異なった色で表示します。



1.3.1.3 逆アセンブル表示モード時のソースウィンドウの画面構成

逆アセンブル表示モードは、命令レベルでデバッグするためのモードです。逆アセンブル表示モードでは、ターゲットプログラムの逆アセンブル結果が参照できます。



1.3.2 ソースウィンドウのショートカットメニュー

ソースウィンドウのショートカットメニューは、プログラムウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.2.2　プログラムウィンドウのショートカットメニュー」をご参照下さい。

1.3.3 ソースウィンドウのツールバー

ソースウィンドウのツールバーは、プログラムウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.2.3 プログラムウィンドウのツールバー」をご参照下さい。

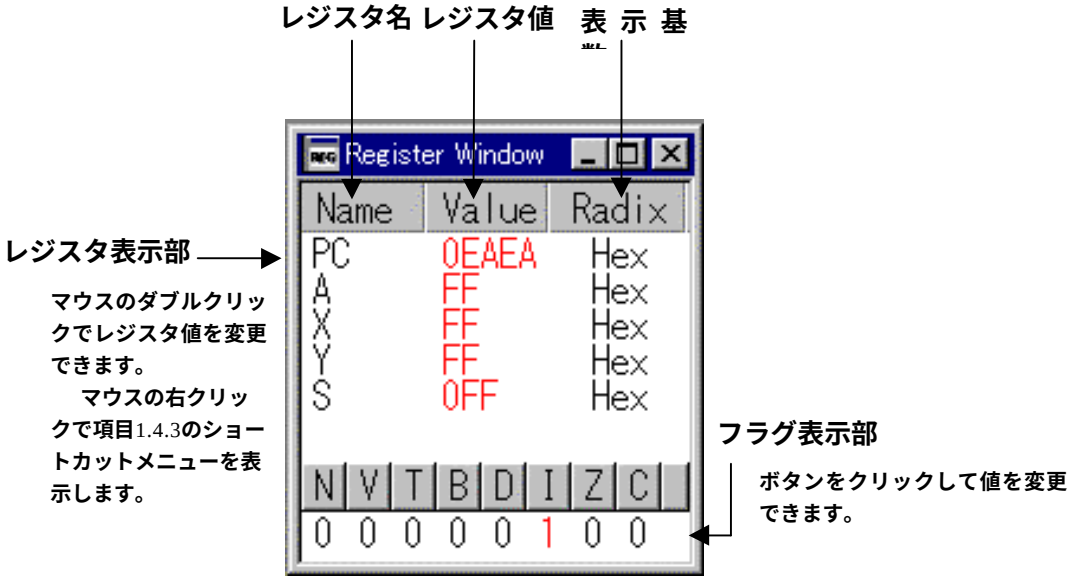
1.3.4 ソースウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、ソースウィンドウがアクティブな場合は、［Option］メニューにはソースウィンドウの拡張メニューが割り当てられます。ソースウィンドウの拡張メニューは、プログラムウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.2.4 プログラムウィンドウの拡張メニュー」をご参照下さい。

1.4 レジスタウィンドウ

レジスタウィンドウは、レジスタの内容やフラグの内容を表示するウィンドウです。表示内容は、各コマンド実行後に更新します。レジスタ値は、レジスタが表示されている行をダブルクリックすることで変更できます。フラグ値は、フラグに対応したボタンをクリックすることで変更できます。

1.4.1 レジスタウィンドウの画面構成



1.4.2 レジスタウィンドウの拡張メニュー

PD38SIMのメイン領域に表示されたウィンドウのうち、レジスタウィンドウがアクティブな場合は、[Option]メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Layout	レイアウト設定	—
	Hide Radix	基数表示列の表示 / 非表示	—
	Hide FLAGs	フラグ表示部の表示/ 非表示	—
	Font...	フォントの変更	—

1.4.3 レジスタウィンドウのショートカットメニュー

レジスタウィンドウのレジスタ表示部でマウスを右クリックすると、以下のショートカットメニューを表示します。

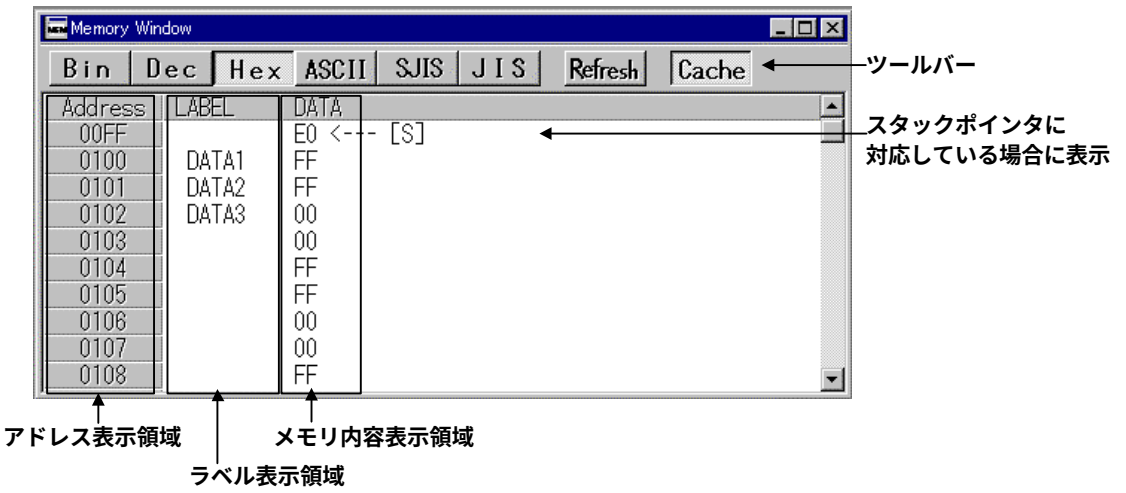
メニュー	メニュー項目	機能	ショートカットキー
右クリック	Hex	選択したレジスタ値を16進表示	—
	Dec	選択したレジスタ値を10進表示	—
	Bin	選択したレジスタ値を2進表示	—
	Layout	レイアウト設定	—
	Hide Radix	基数表示列の表示 / 非表示	—
	Hide FLAGs	フラグ表示部の表示/ 非表示	—
	Font...	フォントの変更	—

- レジスタ値更新前と更新後で値が異なるものについては、値を赤色表示します。

1.5 メモリウィンドウ

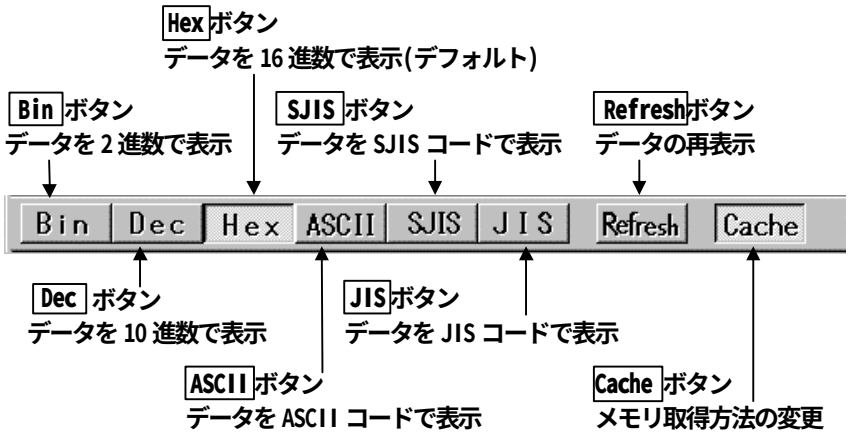
メモリウィンドウは、連続したメモリ内容を「アドレス」「ラベル」「データ（メモリ内容）」の書式で表示するウィンドウです。表示内容は、各コマンド実行後に更新します。データの表示は、2 進数、10 進数、16 進数、ASCII 表示が可能です。メモリウィンドウは、計 10 枚までオープンできます。メモリウィンドウでは、メモリ内容の変更や指定メモリ領域の充填／移動が容易に行えます。

1.5.1 メモリウィンドウの画面構成



- アドレス表示領域をダブルクリックすることで、表示開始アドレスを変更することができます。
- ラベル、またはメモリ内容表示領域をダブルクリックすることで、メモリの内容を変更することができます。
- メモリ内容表示領域をマウスでドラッグすることにより、選択領域のアドレス（Start，End）を FILL、MOVE ダイアログの開始、終了アドレスに設定できます。領域選択後にメニュー［Option］→［Debug］→［Move］、［Option］→［Debug］→［Fill］を選択するとオープンするダイアログの開始アドレス、終了アドレスに選択された領域が設定されます。

1.5.2 メモリウィンドウのツールバー



1.5.3 メモリウィンドウの拡張メニュー

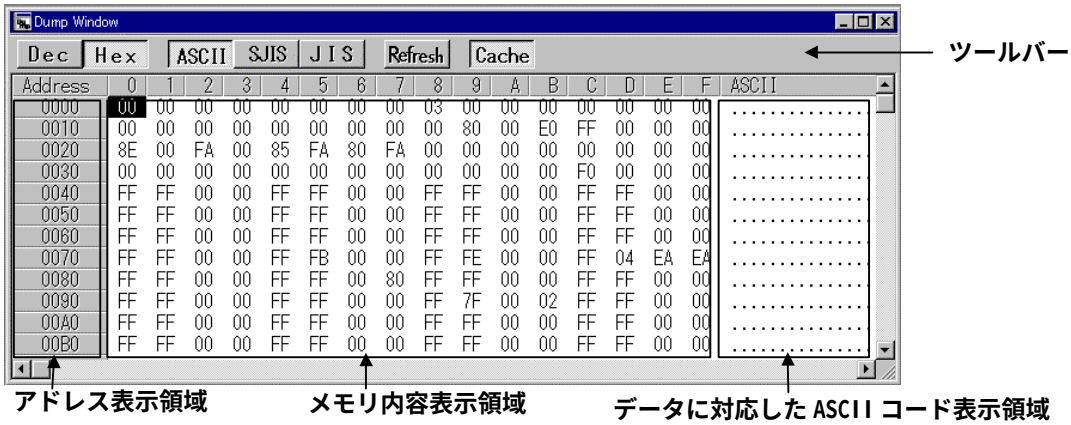
PD38SIMのメイン表示領域に表示されたウィンドウのうち、メモリウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	View	表示内容の変更	—
	Scroll Area...	スクロール範囲の指定	—
	Address...	表示開始アドレスの指定	—
	S	表示開始アドレスをS レジスタ値に変更	—
	Data Length	表示データ長の指定	—
	Byte	1 バイト単位で表示	—
	Word	2 バイト単位で表示	—
	Radix	表示基数の指定	—
	Bin	2 進数で表示	—
	Dec	10 進数で表示	—
	Hex	16 進数で表示	—
	Ascii	ASCII 文字で表示	—
	SJIS	SJIS コードで表示	—
	JIS	JIS コードで表示	—
	Refresh	データの再表示	—
	Debug	メモリ内容の設定	—
	Set...	指定アドレスにデータを設定	—
	Fill...	指定したメモリブロックにデータ充填	—
	Move...	指定したメモリブロックを指定アドレスに移動	—
	Cache On	メモリ取得方法の変更	—

1.6 ダンプウィンドウ

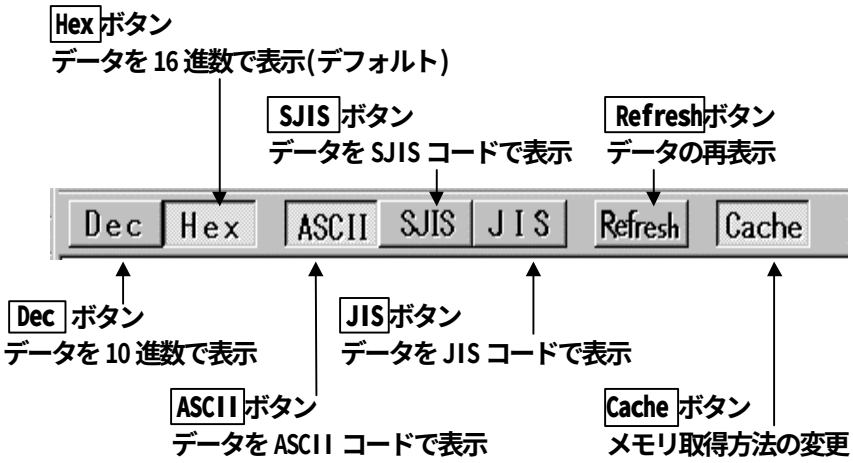
ダンプウィンドウは、連続したメモリ内容をダンプ形式で表示するウィンドウです。表示内容は、各コマンド実行後に更新します。ダンプウィンドウは、計 10 枚までオープンできます。ダンプウィンドウでは、メモリウィンドウと同様にメモリ内容の変更や指定メモリ領域の充填／移動が容易に行えます。

1.6.1 ダンプウィンドウの画面構成



- アドレス表示領域をダブルクリックすることで、表示開始アドレスを変更することができます。
- メモリ内容表示領域をダブルクリックすることで、メモリの内容を変更することができます。
- データ表示領域をマウスでドラッグすることにより、選択領域のアドレス（Start, End）を FILL、MOVE ダイアログの開始、終了アドレスに設定できます。領域選択後にメニュー［Option］→［Debug］→［Move］、［Option］→［Debug］→［Fill］を選択するとオープンするダイアログの開始アドレス、終了アドレスに選択された領域が設定されます。

1.6.2 ダンプウィンドウのツールバー



1.6.3 ダンプウィンドウの拡張メニュー

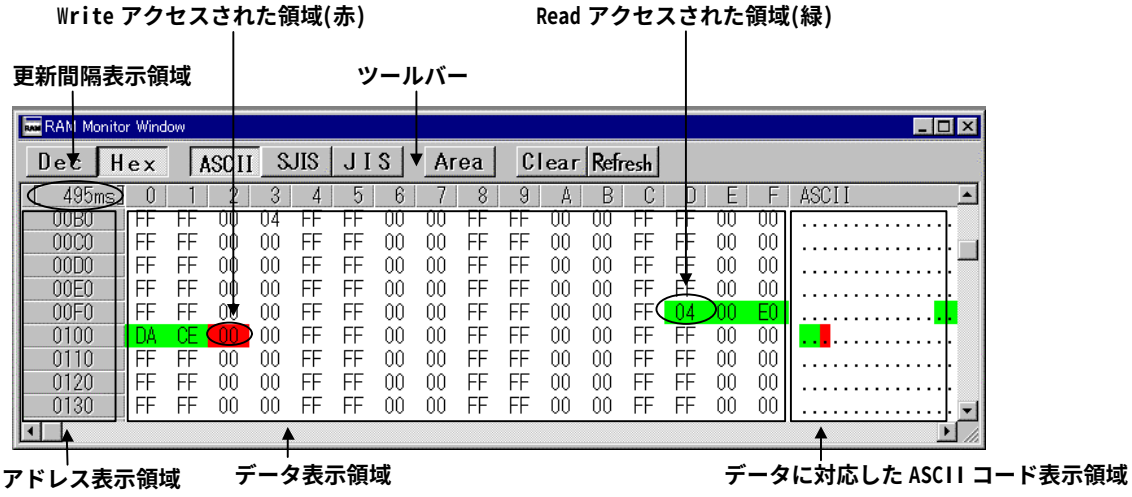
PD38SIMのメイン表示領域に表示されたウィンドウのうち、ダンプウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	View	表示内容の変更	—
	Scroll Area...	スクロール範囲の指定	—
	Address...	表示開始アドレスの指定	—
	Data Length	表示データ長の指定	—
	Byte	1バイト単位で表示	—
	Word	2バイト単位で表示	—
	Radix	表示基数の指定	—
	Dec	10進数で表示	—
	Hex	16進数で表示	—
	ASCII	ASCII コードで表示	—
	SJIS	SJIS コードで表示	—
	JIS	JIS コードで表示	—
	Refresh	データの再表示	—
	Debug	メモリ内容の設定	—
	Set...	指定アドレスにデータを設定	—
	Fill...	指定したメモリブロックにデータ充填	—
	Move...	指定したメモリブロックを指定アドレスに移動	—
	Cache On	メモリ取得方法の変更	—

1.7 RAM モニタウィンドウ

RAM モニタウィンドウは、RAM モニタ領域に該当するメモリ内容をダンプ形式で表示するウィンドウです。表示内容は、ターゲットプログラム実行中に一定間隔（デフォルトは 100ms）で更新します。PD38SIMは、1K バイトの RAM モニタ領域用のメモリ空間を備えており、任意の連続したアドレス空間を RAM モニタ領域として指定することができます。

1.7.1 RAM モニタウィンドウの画面構成

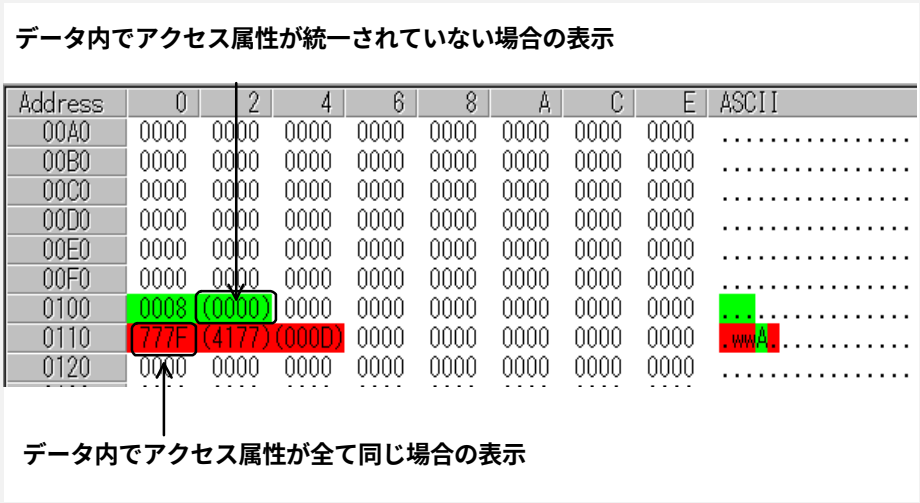


- アドレス表示領域をダブルクリックすることで、表示開始アドレスを変更することができます。なお、指定した表示開始アドレスが現在の RAM モニタ領域の範囲外のときは、RAM モニタ領域も変更されます。
- ターゲットプログラム実行中の更新間隔は、更新間隔表示領域に表示されます（ターゲット停止中は、”Address”文字列表示）。ただし更新間隔は、動作状況によって指定した更新間隔より遅くなる場合があります。特に以下の要因が更新間隔に大きく影響します。
  - ホストコンピュータの性能・負荷状況
  - ウィンドウのサイズ（メモリ表示量）
  - 書き換えが必要な個数（値が変化したメモリの個数）
- データ、及び ASCII コード表示領域のメモリ表示の背景色は、アクセス属性によって以下のように変化します（アクセスなしの場合、背景色は白色）。
  - READ アクセスされた領域
    緑色になります。
  - WRITE アクセスされた領域
    赤色になります。

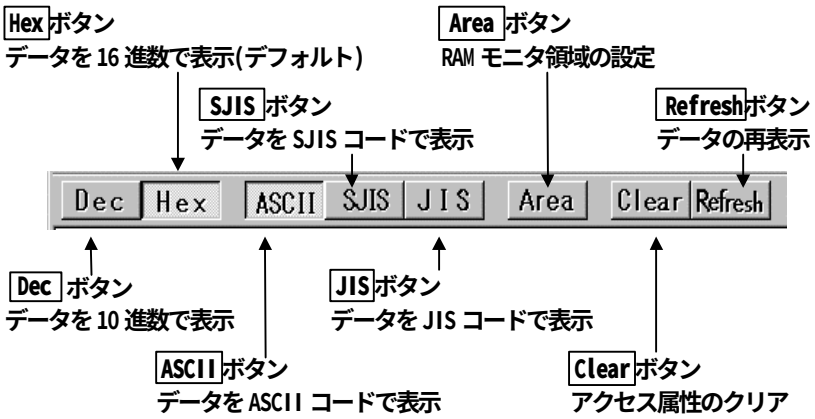
  なお、この背景色の設定は、メニュー［Option］→［Color］で変更することができます。
  また、このアクセス属性の表示は、メニュー［Option］→［View］→［Clear］を選択した場合、およびダウンロード実行後にクリアします（アクセスされていない状態の表示に戻します）。

注意事項

- RAM モニタ表示領域の表示データのデータ長が 2 バイトまたは 4 バイト単位の場合（メニュー［Option］→［View］→［Data Length］でWordメニューまたはDwordメニューを選択している場合）、そのデータの 1 バイト単位でメモリに対するアクセス属性が異なる場合があります。このように 1 つのデータの中でアクセス属性が統一していない場合は、以下に示すように、そのデータが括弧で囲まれて表示されます。ただし、この時のメモリ表示の背景色は、そのデータの 1 バイト目のアクセス属性色に統一されています。



1.7.2 RAM モニタウィンドウのツールバー



1.7.3 RAM モニタウィンドウの拡張メニュー

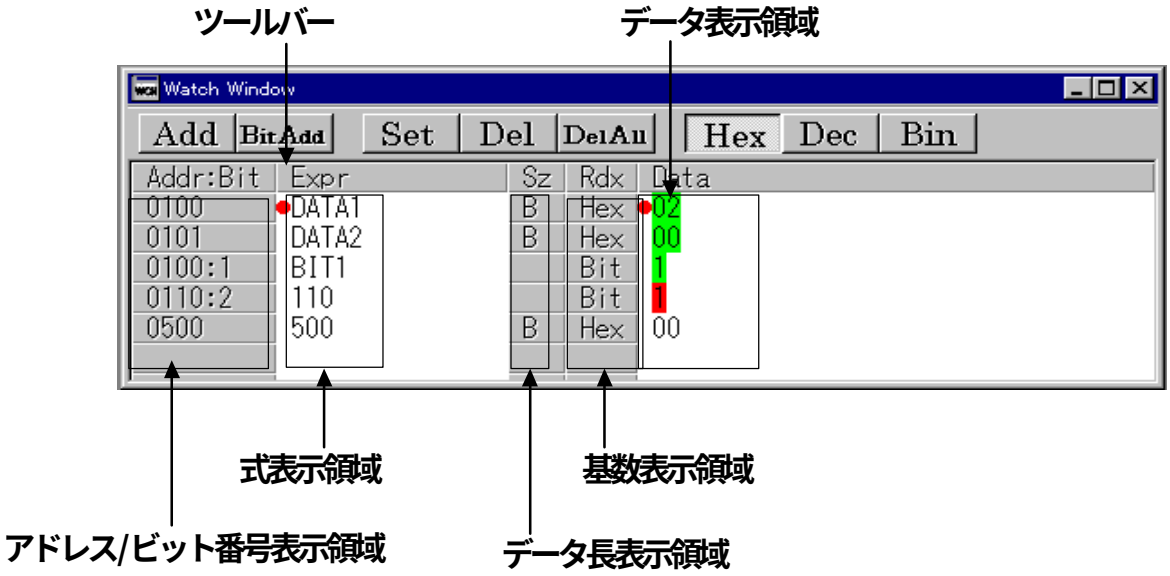
PD38SIMのメイン表示領域に表示されたウィンドウのうち、RAM モニタウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	View	表示内容の変更	—
	Address...	指定アドレスからの表示	—
	Data Length	表示データ長の指定	—
	Byte	1 バイト単位で表示	—
	Word	2 バイト単位で表示	—
	Radix	表示基数の指定	—
	Dec	10 進数で表示	—
	Hex	16 進数で表示	—
	ASCII	ASCII コードで表示	—
	SJIS	SJIS コードで表示	—
	JIS	JIS コードで表示	—
	Refresh	データの再表示	—
	Clear	アクセス属性のクリア	—
	Layout	レイアウト設定	—
	Ascii	ASCII 文字列の表示／非表示	—
	RAM Monitor Area...	RAM モニタ領域の設定	—
	Color...	アクセス属性の表示色の設定	—
	Sampling period...	RAM モニタの表示更新間隔の設定	—

1.8 ASM ウォッチウィンドウ

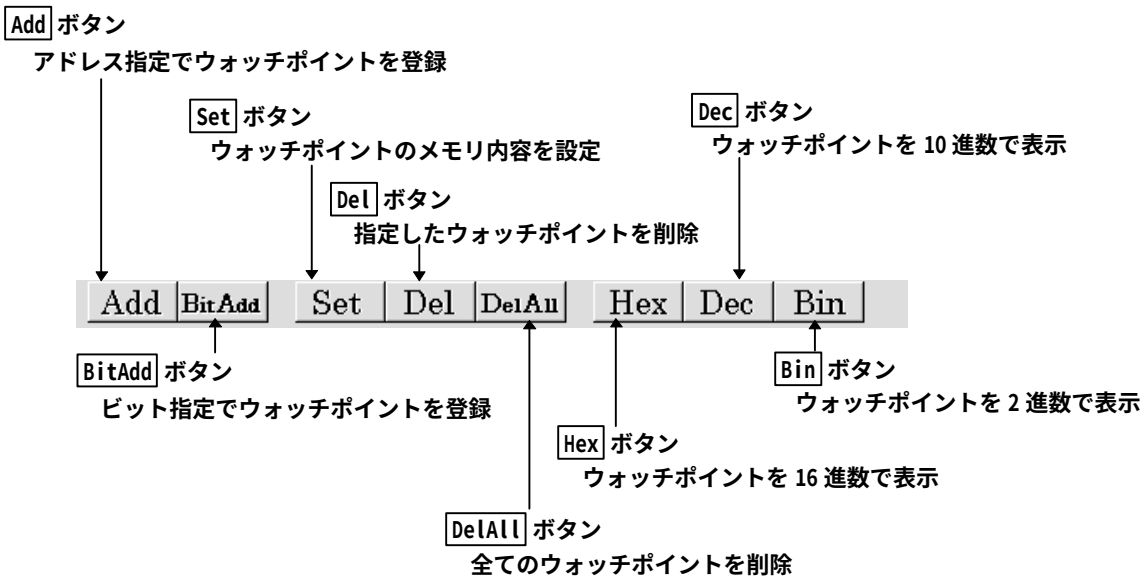
ASM ウォッチウィンドウは、任意アドレスの内容を参照するウィンドウです。この参照する任意アドレスをウォッチポイントと呼びます。ウォッチポイントは、アドレス（シンボルの記述可能）、アドレス＋ビット番号、ビットシンボルのいずれかで指定することができます。表示内容は、各コマンド実行後に更新されます。

1.8.1 ASM ウォッチウィンドウの画面構成



- アドレス式表示領域とデータ表示領域には、赤いマークでカーソル位置が表示されます。カーソル位置は、どちらかの領域をクリックするか、または↑ ↓キーで移動できます。
- 基数表示領域をダブルクリックすると、その位置のデータ表示の基数が、現在の表示基数から、
  ・・・ → 16 進数 → 10 進数 → 2 進数 → 16 進数 → ・・・
  のローテーションで変更されます。
- ウォッチポイントが RAM モニタ領域の範囲内であれば、表示内容はターゲットプログラム実行中に一定間隔で更新します。
- 設定したウォッチポイントの情報は、ASM ウォッチウィンドウをクローズする際、または PD38SIMを終了する際に初期化ファイルへ保存します。これにより、ASM ウォッチウィンドウを再オープンした際に、以前のウォッチポイントを自動的に登録します。
- ASM ウォッチウィンドウは、ターゲットプログラムをダウンロードした際、既に登録済みのウォッチポイントのアドレス式を再計算し、新たなアドレスでメモリ内容を参照します。これにより、プログラムの変更に伴ってウォッチポイントのアドレスが変化した場合でも、アドレスを再設定する必要がありません。
  - 無効なウォッチポイント（メモリ値が“ --<not active>-- ”と表示）については、再計算の結果、アドレス式が正しく計算できた場合、有効なウォッチポイントになります。

1.8.2 ASM ウォッチウィンドウのツールバー



1.8.3 ASM ウォッチウィンドウの拡張メニュー

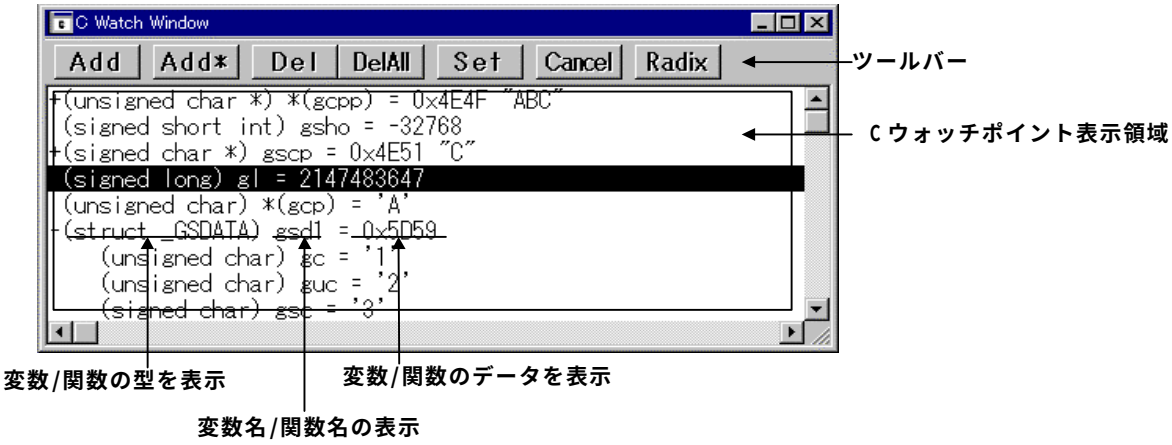
PD38SIMのメイン表示領域に表示されたウィンドウのうち、ASM ウォッチウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	Watch	ウォッチポイントの登録／削除	—
	Add...	ウォッチポイントの登録	Ctrl + A
	Bitadd...	ビットレベルのウォッチポイントの登録	Ctrl + B
	Set...	選択位置のウォッチポイントのメモリ内容の設定	Ctrl + S
	Del	選択位置のウォッチポイントの削除	Ctrl + D
	DelAll	全ウォッチポイントの削除	—
	Radix	表示基数の変更	—
	Bin	選択位置のウォッチポイントの値を 2 進数で表示	—
	Dec	選択位置のウォッチポイントの値を 10 進数で表示	—
	Hex	選択位置のウォッチポイントの値を 16 進数で表示	—
	Layout	レイアウト設定	—
	Address Area	アドレス／ビット表示領域の表示／非表示	—
	Size Area	データ長表示領域の表示／非表示	—
	RAM Monitor	RAM モニタ表示	—
	Sampling period...	RAM モニタの表示更新間隔の設定	—
	Color...	アクセス属性の表示色の設定	—
	File	ウォッチポイントの保存／読み込み	—
	Save…	ウォッチポイントの保存	—
	Load…	ウォッチポイントの読み込み	—

1.9 C ウォッチウィンドウ

C ウォッチウィンドウは、C 言語の式（以下、C 言語式）とその値（計算結果）を表示するウィンドウです。この C ウォッチウィンドウの表示対象となる C 言語式のことを C ウォッチポイントと呼びます。C ウォッチポイントの計算結果の表示は、各コマンド実行後に更新されます。

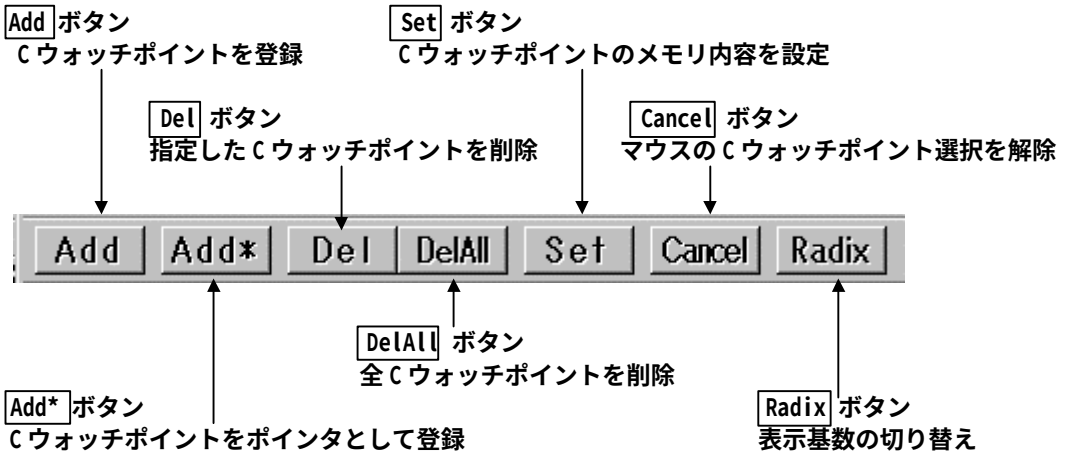
1.9.1 C ウォッチウィンドウの画面構成



- C ウォッチポイントには、以下のものが使用できます。
  - Cシンボル
    C言語ソースプログラムで定義されている変数名および関数名。
  - Cシンボルを含む式（C 言語式）
    C 言語式に使用できる字句（トークン）については、本マニュアル リファレンス編の項目「3.1 C言語式の記述方法」をご参照下さい。
- C 言語式が正しく計算できなかった場合、（例えばシンボル未定義の場合）でも、無効な C ウォッチポイントとして登録します。無効な C ウォッチポイントは、計算結果表示の対象になりません。ただし、無効な C ウォッチポイントは再計算を行った結果、正しく計算できた場合には有効な C ウォッチポイントに変わります。
- 以下に示す C ウォッチポイントには、値を代入することはできません。
  - 浮動小数点型変数
  - ビットフィールド型変数
  - レジスタ変数
  - その他、アドレスを示さない C ウォッチポイント
- 任意の変数をダブルクリックすると、その位置のデータ表示の基数が、現在の表示基数から、
  ・・・ → 16 進数 → 10 進数 →2 進数 → 16 進数 → ・・・
  のローテーションで変更されます。
- RAM モニタ機能が有効の場合に、C ウォッチポイントが RAM モニタ領域の範囲内であれば、ターゲットプログラム実行中（一定間隔）に表示内容を更新します。また、アクセスがあった変数は、変数の値を色付で表示します。RAM モニタ機能は、メニュー [Option] → [RAM Monitor] → [Enable] で有効／無効を設定します。

- 設定したＣウォッチポイントの情報は、Ｃウォッチウィンドウをクローズする際、またはPD38SIMを終了する際にＣウォッチポイント情報格納ファイルへ保存します。これにより、Ｃウォッチウィンドウを再オープンした際に、以前のＣウォッチポイントを自動的に登録します。
  - Ｃウォッチポイント情報格納ファイルは、ロードしたオブジェクトファイルごとに別々に作成されます（ロードしたオブジェクトファイル名の情報が含まれます）。
    Ｃウォッチポイントを再登録する際は、まず現在ロードされているオブジェクトファイルと同じオブジェクトファイル名の情報を持つＣウォッチポイント情報格納ファイルを検索します。検索の結果、該当するファイルが見つかれば、そのファイルの情報から得たＣウォッチポイントを再登録します。

1.9.2 C ウォッチウィンドウのツールバー



- Ｃウォッチポイント表示領域上でクリックして選択したＣウォッチポイントに対して、削除や値の設定が行えます。なお、構造体の内容（メンバ）などＣウォッチポイントを登録する際に付加情報として表示されたものは、その付加情報だけを削除することはできません。
- ポインタなどのアドレス表示は、表示基数にかかわらず 16 進表示で固定です。なお、表示形式の詳細については、本マニュアル リファレンス編の項目「3.2 Ｃ言語式の表示形式」をご参照下さい。

1.9.3 Cウォッチウィンドウの拡張メニュー

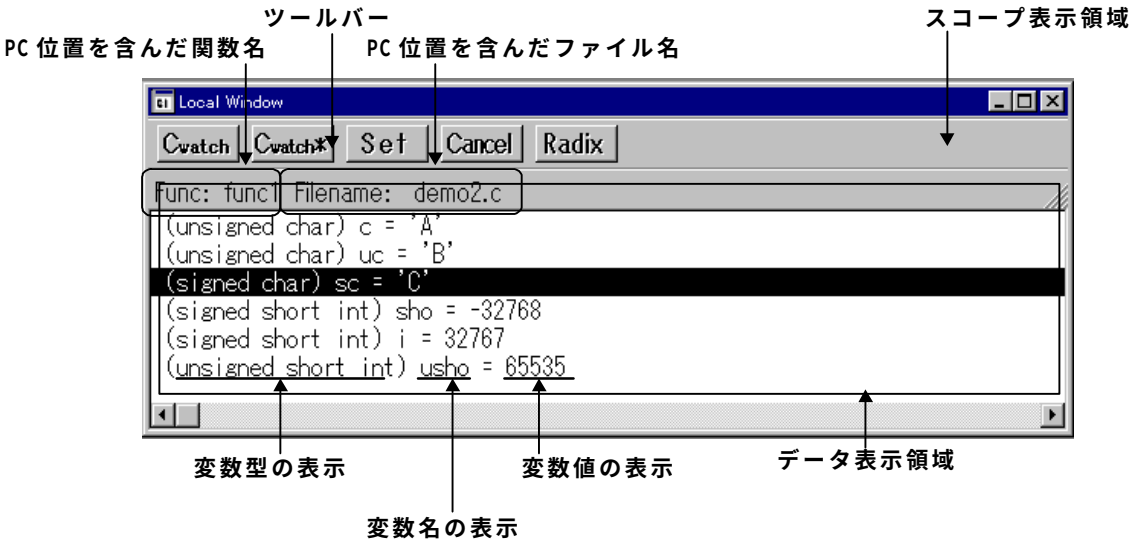
PD38SIMのメイン表示領域に表示されたウィンドウのうち、C ウォッチウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	Watch	C ウォッチポイントの登録／削除	—
	Add...	C ウォッチポイントの登録	—
	Add Pointer...	C ウォッチポイントの登録（ポインタ）	—
	Del	選択位置のC ウォッチポイントの削除	—
	Set...	選択位置のC ウォッチポイントの値を設定	—
	Cancel	C ウォッチポイントの選択の解除	—
	Del All	全C ウォッチポイントの削除	—
	View	表示内容の変更	—
	Radix	表示基数の変更	—
	Layout	型名の表示／非表示	—
	Sort	アルファベット順に並べ替え	—
	Display String	文字列表示／文字表示	—
	RAM Monitor	RAM モニタ表示	—
	Enable	RAM モニタ機能の有効／無効	—
	RAM Monitor Area...	RAM モニタの領域の設定	—
	Color...	アクセス属性の表示色の設定	—
	Sampling period...	RAM モニタの表示更新間隔の設定	—
	Clear	アクセス属性のクリア	—

1.10 ローカルウィンドウ

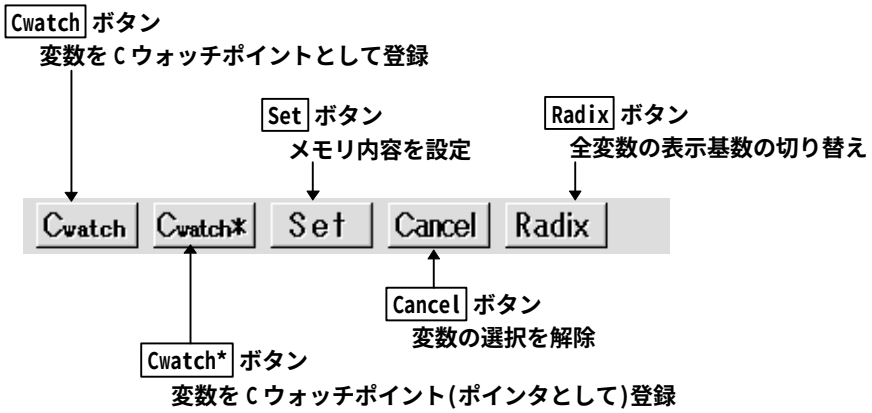
ローカルウィンドウは、C 言語の関数内ローカル変数の一覧とその値を表示するウィンドウです。ローカルウィンドウで表示される変数は、各コマンド実行後に更新されます。

1.10.1 ローカルウィンドウの画面構成



- ステップ実行等を行った際に、プログラムカウンタが含まれる関数が変更された場合は、それまで表示していた変数を削除し、新しい関数の関数内ローカル変数を自動登録します。

1.10.2 ローカルウィンドウのツールバー



- データ表示領域上でクリックして選択した変数に対して、Ｃウォッチポイントとして Ｃウォッチウィンドウに登録、及び値の設定が行えます。なお、値の設定には、Ｃ言語式が使用できます。C 言語式に使用できる字句（トークン）については、本マニュアル リファレンス編の項目「3.1 Ｃ言語式の記述方法」をご参照下さい。
- ポインタなどのアドレス表示は、表示基数にかかわらず 16 進表示で固定です。なお、表示形式の詳細については、本マニュアル リファレンス編の項目「3.2 Ｃ言語式の表示形式」をご参照下さい。

1.10.3 ローカルウィンドウの拡張メニュー

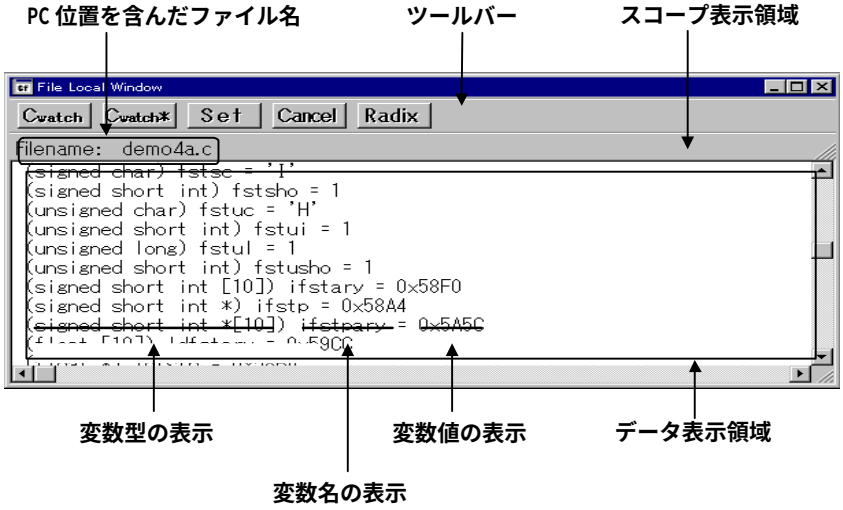
PD38SIMのメイン表示領域に表示されたウィンドウのうち、ローカルウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	Watch	Ｃ変数に関する操作	—
	Cwatch	選択位置のＣ変数をＣウォッチポイントとして登録	—
	Cwatch Pointer	選択位置のＣ変数のポインタをＣウォッチポイントとして登録	—
	Set...	選択位置のＣ変数の値を設定	—
	Cancel	Ｃ変数の選択を解除	—
	View	表示内容の変更	—
	Radix	表示基数の変更	—
	Layout	型名の表示／非表示	—
	Sort	アルファベット順に並べ替え	—
	Display String	文字列表示／文字表示	—

1.11 ファイルローカルウィンドウ

ファイルローカルウィンドウは、C 言語のファイル内ローカル変数の一覧とその値を表示するウィンドウです。ファイルローカルウィンドウで表示される変数は、各コマンド実行後に更新されます。

1.11.1 ファイルローカルウィンドウの画面構成



- ステップ実行等を行った際に、プログラムカウンタが含まれるファイルが変更された場合は、それまで表示していた変数を削除し、新しいソースファイルのファイル内ローカル変数を自動登録します。

1.11.2 ファイルローカルウィンドウのツールバー

ファイルローカルウィンドウのツールバーは、ローカルウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.10.2 ローカルウィンドウのツールバー」をご参照下さい。

1.11.3 ファイルローカルウィンドウの拡張メニュー

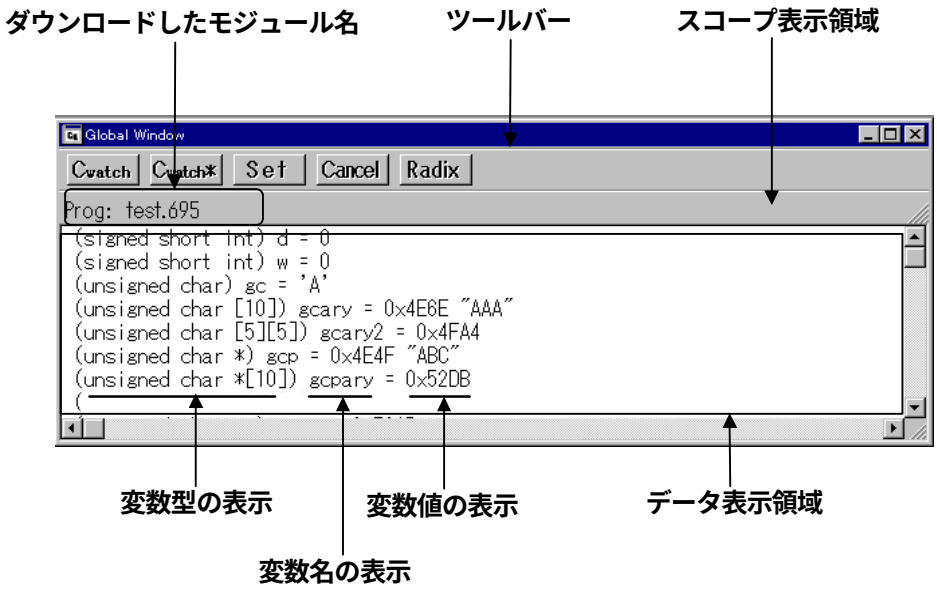
PD38SIMのメイン表示領域に表示されたウィンドウのうち、ファイルローカルウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	Watch	C変数に関する操作	—
	Cwatch	選択位置のC変数をCウォッチポイントとして登録	—
	Cwatch Pointer	選択位置のC変数のポインタをCウォッチポイントとして登録	—
	Set...	選択位置のC変数の値を設定	—
	Cancel	C変数の選択を解除	—
	View	表示内容の変更	—
	Radix	表示基数の変更	—
	Layout	型名の表示／非表示	—
	Sort	アルファベット順に並べ替え	—
	Display String	文字列表示／文字表示	—
	RAM Monitor	RAMモニタ表示	—
	Enable	RAMモニタ機能の有効／無効	—
	RAM Monitor Area...	RAMモニタの領域の設定	—
	Color...	アクセス属性の表示色の設定	—
	Sampling period...	RAMモニタの表示更新間隔の設定	—
	Clear	アクセス属性のクリア	—

1.12 グローバルウィンドウ

グローバルウィンドウは、C 言語のグローバル変数の一覧とその値を表示するウィンドウです。グローバルウィンドウで表示される変数は、各コマンド実行後に更新されます。

1.12.1 グローバルウィンドウの画面構成



1.12.2 グローバルウィンドウのツールバー

グローバルウィンドウのツールバーは、ローカルウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.10.2 ローカルウィンドウのツールバー」をご参照下さい。

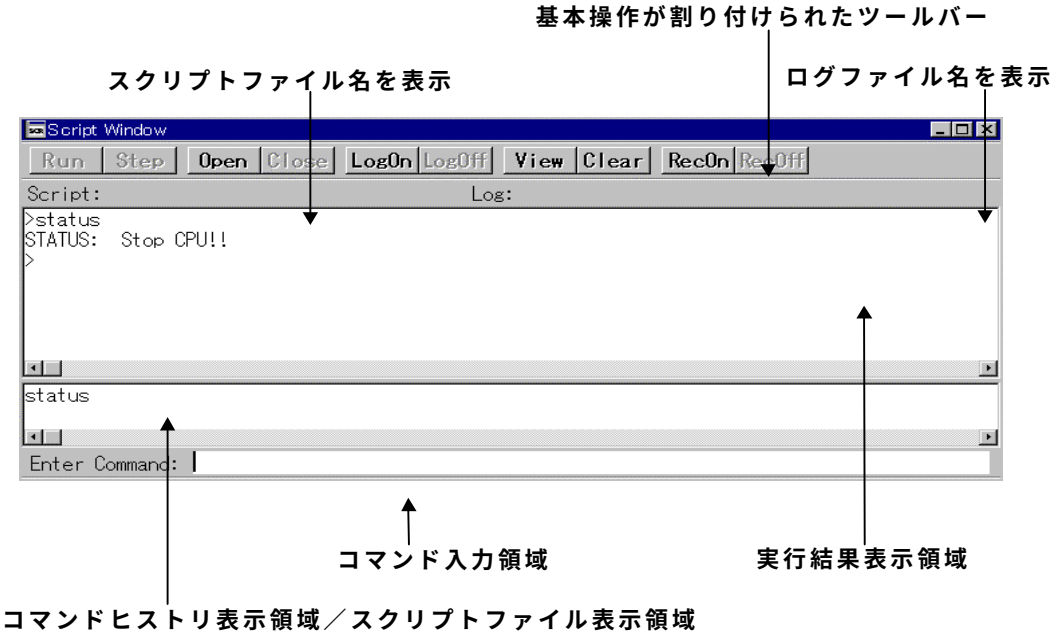
1.12.3 グローバルウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、グローバルウィンドウがアクティブな場合は、［Option］メニューにはグローバルウィンドウの拡張メニューが割り当てられます。グローバルウィンドウの拡張メニューは、ファイルローカルウィンドウと同じです。詳細な説明については、本マニュアル ウィンドウ機能編の項目「1.11.3 ファイルローカルウィンドウの拡張メニュー」をご参照下さい。

1.13 スクリプトウィンドウ

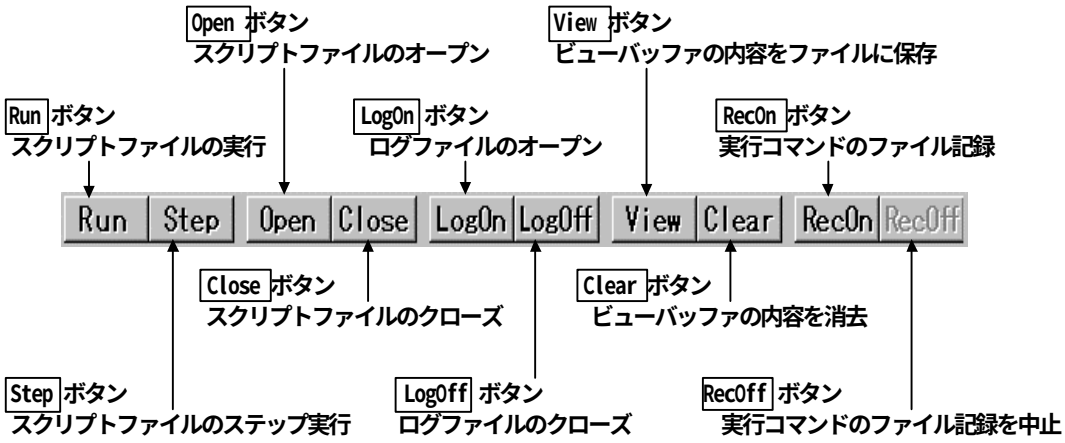
スクリプトウィンドウは、テキスト形式のスクリプトコマンドの実行、及び実行結果を表示するウィンドウです。スクリプトコマンドは、スクリプトファイルまたは対話入力によって実行できます。スクリプトコマンドをあらかじめスクリプトファイルに記述しておくことにより、スクリプトコマンドを自動実行できます。また、スクリプトコマンドの実行結果は、あらかじめ指定したファイル（ログファイル）に保存することができます。

1.13.1 スクリプトウィンドウの画面構成



- スクリプトウィンドウは、最新 1000 行分の実行結果を保存するビューバッファを持っており、ログファイルを指定していなくても実行結果をファイル（ビューファイル）に保存することができます。
- スクリプトファイルをオープンすると、コマンドヒストリ領域はスクリプトファイル表示領域に切り替わり、スクリプトファイルの内容を表示します。スクリプトファイルをネストオープンしている場合は、一番最後にオープンしたスクリプトファイルの内容を表示します。また、スクリプトファイル表示領域では、現在実行しているスクリプトファイル行を反転表示します。
- スクリプトファイルオープン時も、スクリプトファイルの実行が停止している時のみ、コマンド入力領域からスクリプトコマンドを発行することができます。
- スクリプトウィンドウは、実行したコマンドをファイルに記録することができます。この機能はログ機能とは異なり、コマンドのみを記録するため、保存したファイルをスクリプトファイルとして使用することができます。

1.13.2 スクリプトウィンドウのツールバー



1.13.3 スクリプトウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、スクリプトウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font....	フォントの変更	—
	Script	スクリプトファイルの操作	—
	Open...	スクリプトファイルのオープン	—
	Run	スクリプトファイルの実行	—
	Stop	スクリプトファイルの実行停止	—
	Step	スクリプトファイルのステップ実行	—
	Close	スクリプトファイルのクローズ	—
	View	ビューバッファの操作	—
	Save...	ビューバッファのファイル保存	—
	Clear	ビューバッファのクリア	—
	Log	ログファイルの操作	—
	On...	ログファイルのオープン（出力開始）	—
	Off	ログファイルのクローズ（出力終了）	—
	Record	コマンドの記録	—
	On...	コマンドのファイル記録	—
	Off	コマンドのファイル記録の中止	—

1.14 I/O ウィンドウ

I/O ウィンドウは、仮想ポート入力や仮想ポート出力、仮想割り込みの設定、表示を行うウィンドウです。仮想ポート入力、仮想割り込みの設定及び、仮想ポート出力の結果をチャート、数値、グラフ形式の表示で参照できます。

仮想ポート入力、仮想ポート出力、仮想割り込みの設定方法の詳細に関しては、後述の「より高度なデバッグ編」を参照下さい。

ここでは、I/O ウィンドウの機能、画面構成、ツールバー、メニューについて説明します。

- 設定できる仮想ポート入力/仮想割り込みは、合わせて 20 個までです。
  仮想ポート出力は 20 個まで設定できます。

1.14.1 仮想ポート入力

仮想ポート入力とは、外部から指定アドレスのメモリに入力されるデータの変化を定義する機能です。この機能を利用すると、SFR に定義されているポートに対するデータ入力等のシミュレートが行えます。

定義した入力データは、チャート、16 進数値、グラフ形式で参照できます。

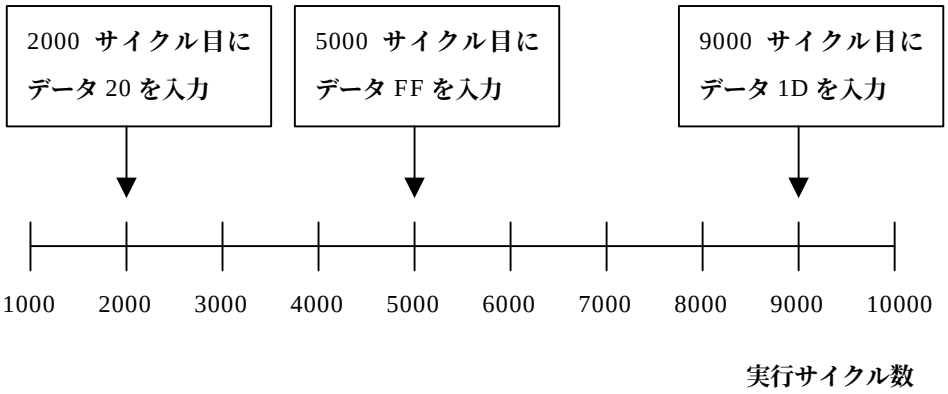
仮想ポート入力には、以下の 3 種類の入力があります。

(1) サイクル同期入力

プログラムの実行が指定サイクルになった時に、入力データをメモリに書き込むことができます。入力するデータのサイズは 1 バイトです。

サイクルに同期した仮想ポート入力の例を以下に示します。

「アドレス 0 番地にデータを入力する場合の例」



上記のように、ユーザが指定した任意のサイクルにおいて 0 番地のメモリにデータを入力するように設定することができます。

(2) リードアクセス同期入力

指定されたメモリをプログラムがリードアクセスした時にデータを入力することができます。入力するデータのサイズは１バイトです。
メモリのリードアクセスに同期した仮想ポート入力の例を以下に示します。

下記のプログラム例は、ポート 0（0 番地）の読み出しを行う関数の例です。

```
T_PORT0:            .byte           0000H
                    .section        Z
                    .org            0040H
                          :
                          :
READ_PORT0:
                    A   =   [T_PORT0]
                          :
                          :
```

この関数では、アキュムレータにポート 0 の値を代入しようとしています。このような場合、ポート 0（0 番地）をプログラムがリードアクセスした時に、ポート 0 に対して値を入力すればアキュムレータに値を代入することができます。

このような関数の処理を支援する機能として、指定したアドレスのメモリをリードした回数に応じて入力するデータを定義する機能（メモリのリードアクセスに同期した仮想ポート入力）を用意しています。この機能を利用すると、次のように 0 番地を 1 回目にリードした時に 0x10、2 回目にリードした時に 0x20 を 0 番地のメモリに入力する処理が行えます。

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

(3) 割り込み同期入力

仮想割り込みが発生した時に、指定されたメモリにデータを入力することができます。入力するデータのサイズは１バイトです。
仮想割り込みに同期した仮想ポート入力の例を以下に示します。

下記のプログラムの例は、ポート 1（2 番地）の読み出しを割り込みハンドラルーチンでおこなっている場合の例です。

```
                    .section        DATA
                    .org            0080H
T_PORT1:            .byte           0002H

                    .section        PROGRAM
                    .org            8000H

; 割り込みハンドラ
INT_2:
                    A   =   [T_PORT0]
                            :
                            :
                    RTI
```

この割り込みハンドラルーチンでは、仮想割り込みが発生した時にアキュムレータにポート 1 の値を代入しようとしています。このような場合、仮想割り込みが発生した時に、ポート 1 に対して値を入力すればアキュムレータに値を代入することができます。
なお、割り込みは、別途仮想割り込みの機能を利用して発生させているものとします（後述の仮想割り込みの項を参照下さい）。

このような割り込みハンドラの処理を支援する機能として、仮想割り込みが発生した回数に応じて入力するデータを定義する機能（仮想割り込みに同期した仮想ポート入力）を用意しています。この機能を利用すると、次のように仮想割り込みが 1 回目に発生した時に 0xFF、2 回目に発生した時に 0xFE を 2 番地のメモリに入力する処理が行えます。

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

1.14.2 仮想ポート出力

仮想ポート出力機能とは、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録する機能です。
記録されたデータは、チャート形式、数値形式やグラフ形式で確認できます。
なお、記録されるデータの個数は、プログラム実行開始時から最大 30000 データです。

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

```
T_PORT0:        .byte           0000H

                .section        Z
                .org            0040H
DATABUFF:       .blkb           1
                        :
                        :
OUT_PORT0:
                [T_PORT0]  =  [DATABUFF]
```

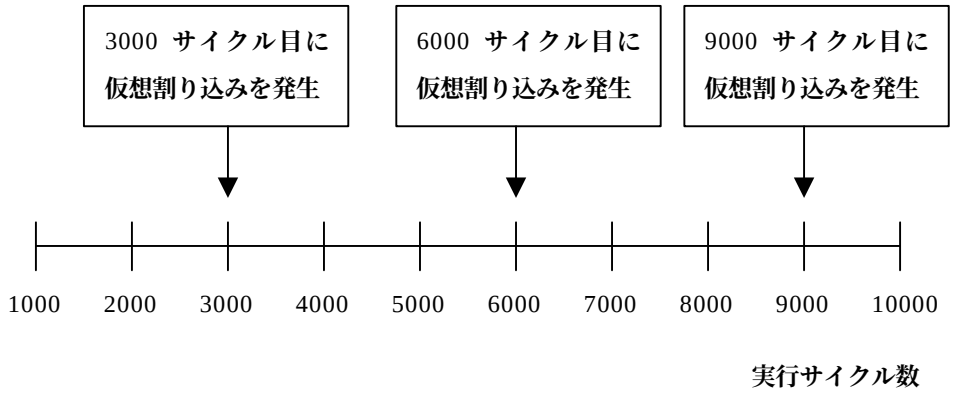
0 番地に書き込まれたデータとその時のサイクル数を記録します。

1.14.3 仮想割り込み

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

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

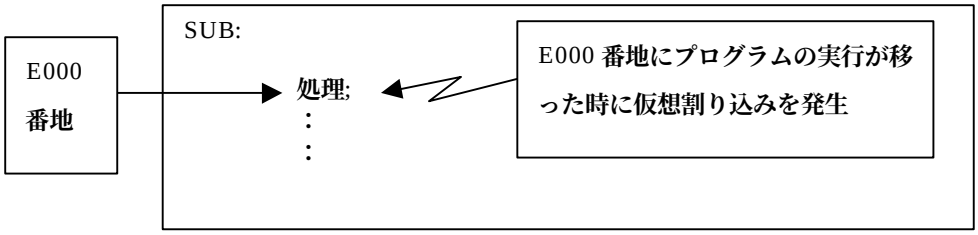
「ベクトル番地 0xffec（タイマ 2）の仮想割り込みの定義例」



上記のように、仮想割り込み（この場合タイマ 2 の割り込み）を任意のサイクルで発生させることができます。

(2) 実行アドレス同期割り込み

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



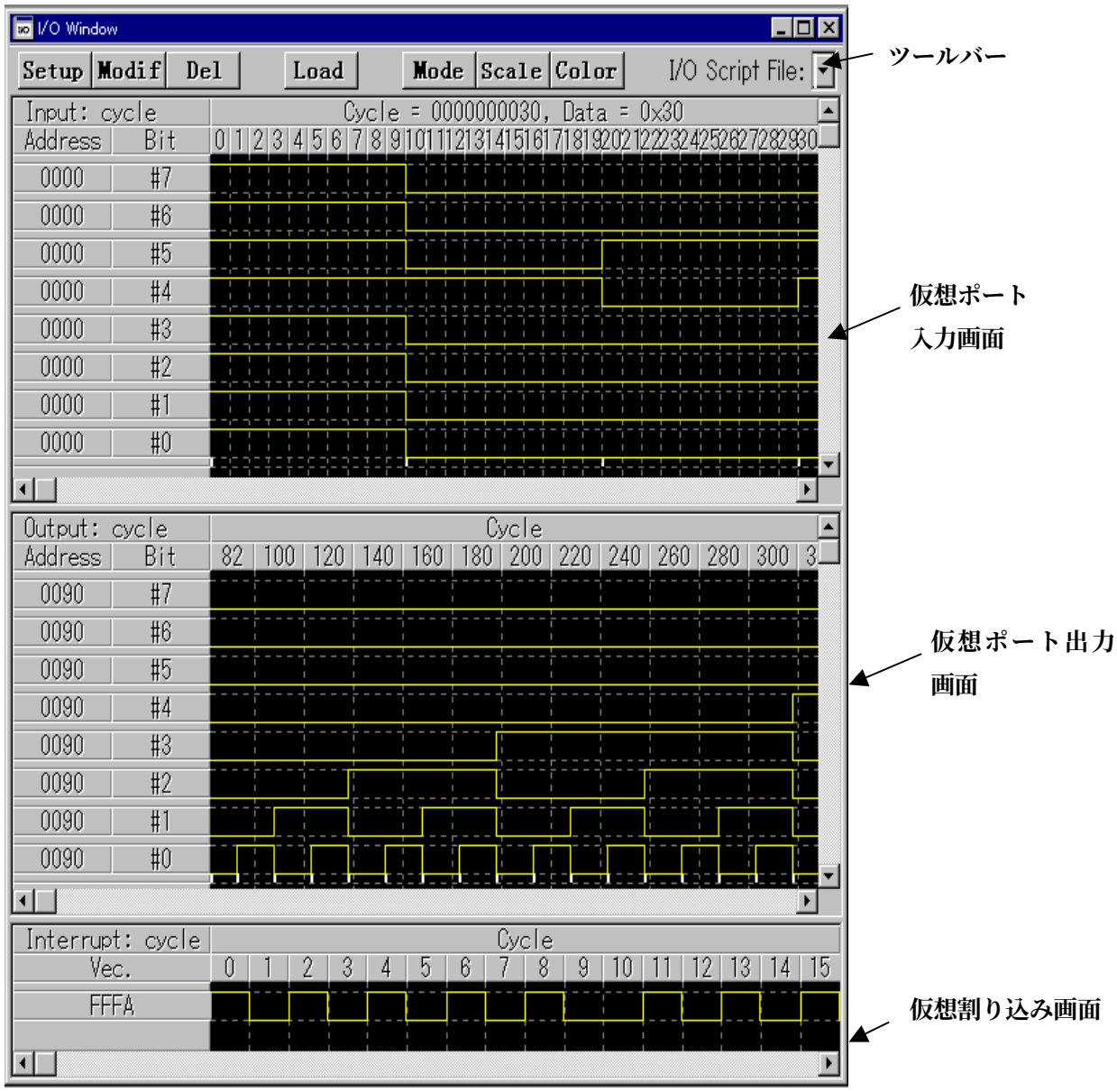
上記のように、プログラムの実行が E000 番地に移った時に指定した仮想割り込みを発生するように定義することができます。
この機能を利用すると、次のようにプログラムが E000 番地を 1 回目に実行した時に仮想割り込みを発生、2 回目に実行した時は割り込みを発生させないというような指定が行えます。

E000 番地を実行した回数	仮想割り込み発生の有無
1 回目	仮想割り込みを発生
2 回目	仮想割り込みを発生しない
3 回目	仮想割り込みを発生
⋮	⋮
⋮	⋮

1.14.4 I/O ウィンドウの画面構成

I/O ウィンドウは、仮想ポート入力の設定内容、仮想ポート出力の出力結果、及び仮想割り込みの設定内容をそれぞれ個別の画面で表示します。

I/O ウィンドウは、下図に示す３画面で構成されています。



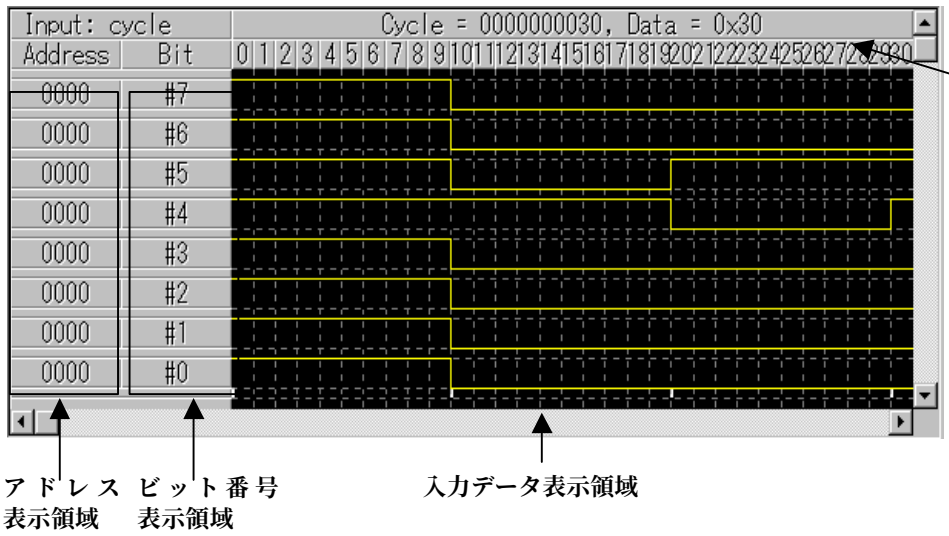
各画面については、次節以降で詳細に説明します。

1.14.5 仮想ポート入力画面の構成

1.14.5.1 サイクル同期入力の画面構成

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

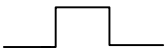
(1) チャート形式（ビット単位）
設定された仮想ポート入力をビット単位のチャート形式で表示します。

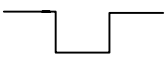


アドレス表示領域
仮想ポート入力を行うメモリのアドレスを表示します。

ビット番号表示領域
仮想ポート入力を行うメモリのビット番号を表示します。

入力データ表示領域
設定された仮想ポート入力のデータをビットごとにチャート形式で表示します。


この表示は、メモリのビットが１の状態です。

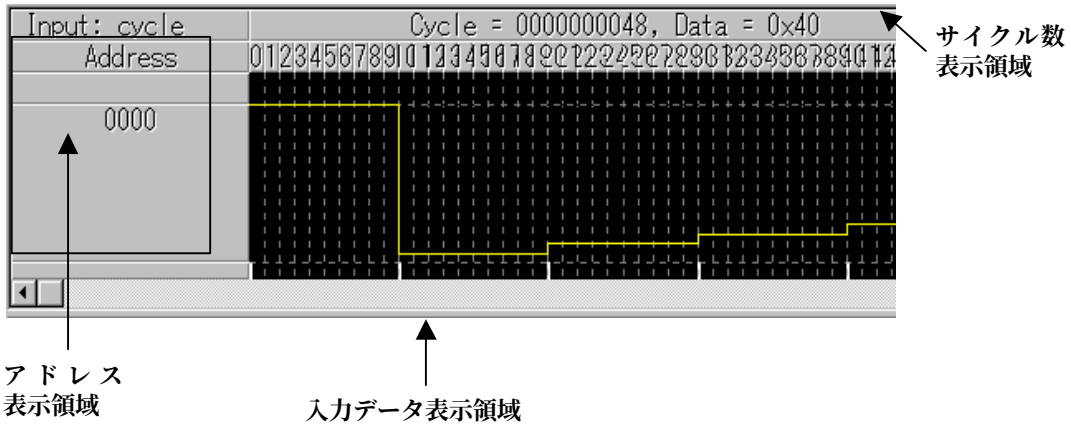

この表示は、メモリのビットが０の状態です。

入力データ表示領域の一番下に表示されている短い白線は、データが入力されるポイントを示しています。

「データ値を参照するには」
この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

サイクル数表示領域
サイクル数を表示します。

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



アドレス表示領域

仮想ポート入力を行うメモリのアドレスを表示します。

入力データ表示領域

設定された仮想ポート入力のデータをグラフ形式で表示します。

表示されるグラフの凸は、データを表示している領域の高さを 255（1 バイトデータの最大値）等分してデータ値を表示しています。

入力データ表示領域の一番下に表示されている短い白線は、データが入力されるポイントを示しています。

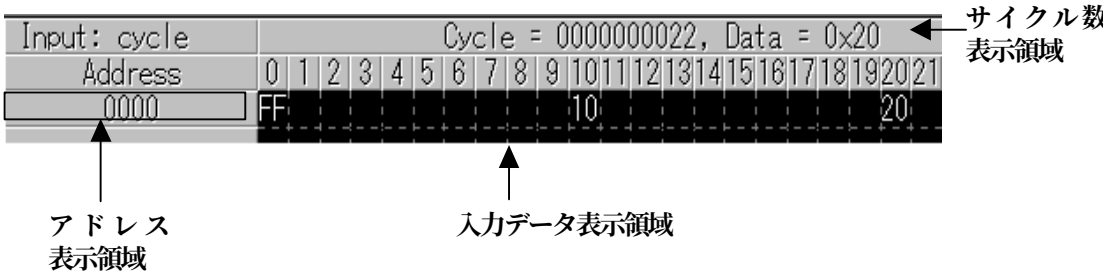
「データ値を参照するには」

この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

サイクル数表示領域

サイクル数を表示します。

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



アドレス表示領域

仮想ポート入力を行うメモリのアドレスを表示します。

入力データ表示領域

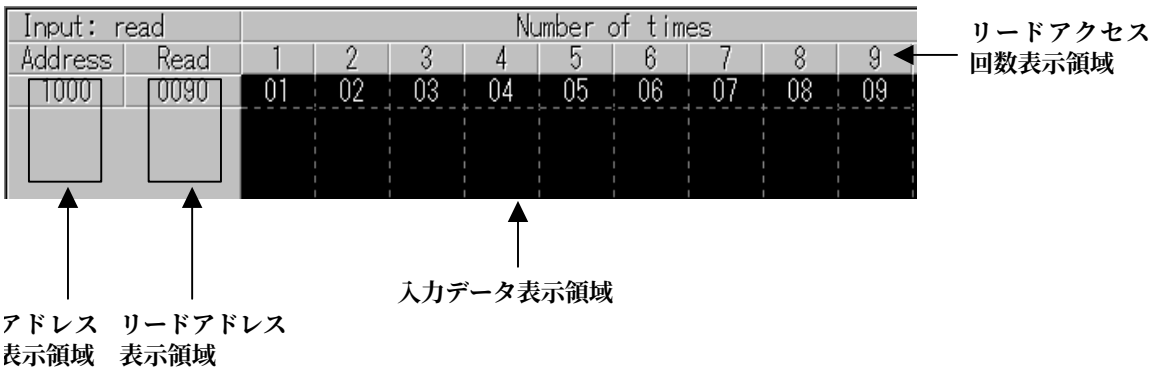
設定された仮想ポート入力のデータを 16 進数値で表示します。
「データ値を参照するには」
この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

サイクル数表示領域

サイクル数を表示します。

1.14.5.2 リードアクセス同期入力の画面構成

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



アドレス表示領域

仮想ポート入力を行うメモリのアドレスを表示します。

リードアドレス表示領域

リードアクセスの監視を行うアドレスを表示します。

入力データ表示領域

設定された仮想ポート入力のデータを 16 進数値で表示します。

「データ値を参照するには」

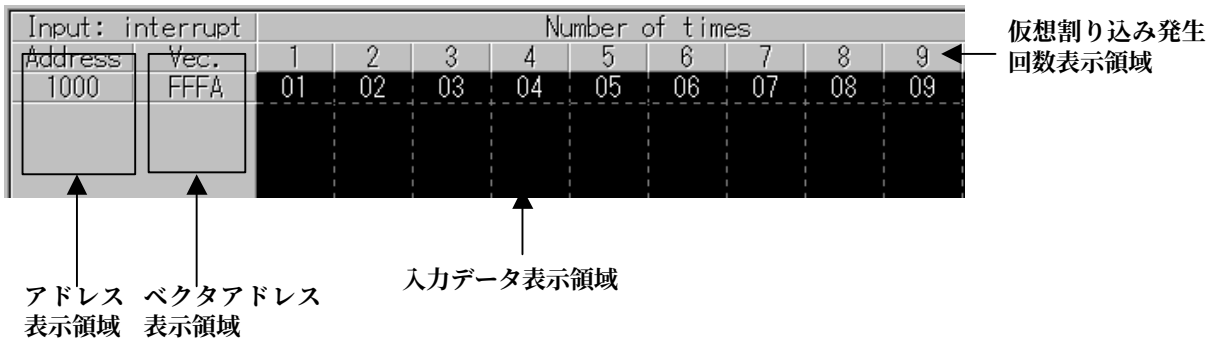
この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とリードアクセス回数をリードアクセス回数表示領域に表示します。

リードアクセス回数表示領域

リードアクセス回数を表示します。

1.14.5.3 割り込み同期入力の画面構成

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



アドレス表示領域

仮想ポート入力を行うメモリのアドレスを表示します。

ベクタアドレス表示領域

監視する仮想割り込みのベクタアドレスを表示します。

入力データ表示領域

設定された仮想ポート入力のデータを 16 進数値で表示します。

「データ値を参照するには」

この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値と仮想割り込み発生回数を仮想割り込み発生回数表示領域に表示します。

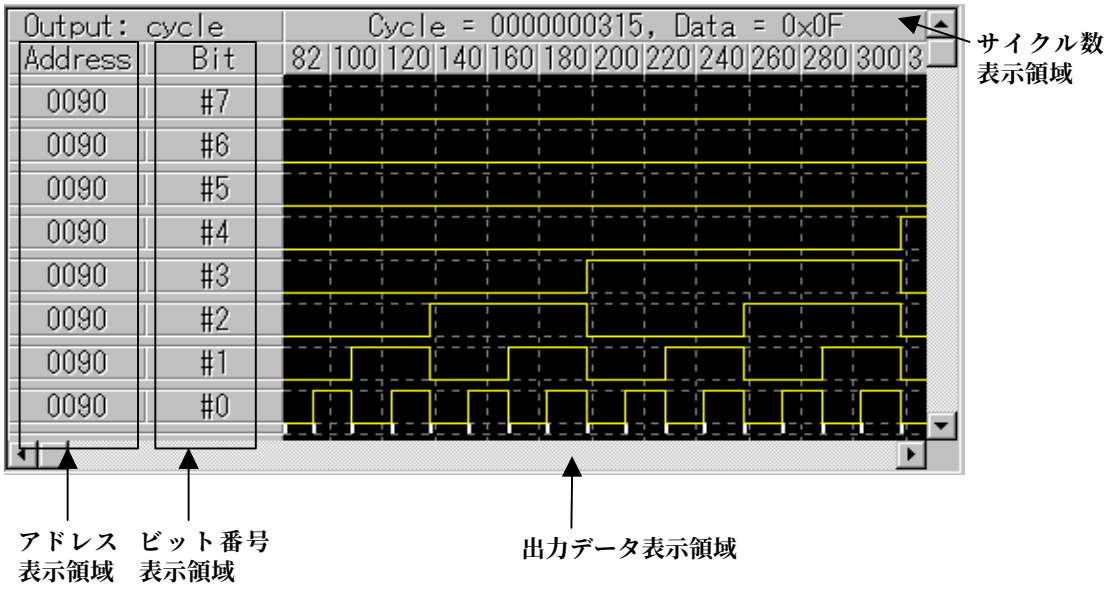
仮想割り込み発生回数表示領域

仮想割り込み発生回数を表示します。

1.14.6 仮想ポート出力画面の構成

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

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



アドレス表示領域
仮想ポート出力の監視を行うアドレスを表示します。

ビット番号表示領域
仮想ポート出力の監視を行っているメモリのビット番号を表示します。

出力データ表示領域
仮想ポート出力結果のデータをビットごとにチャート形式で表示します。


この表示は、メモリのビットが１の状態です。

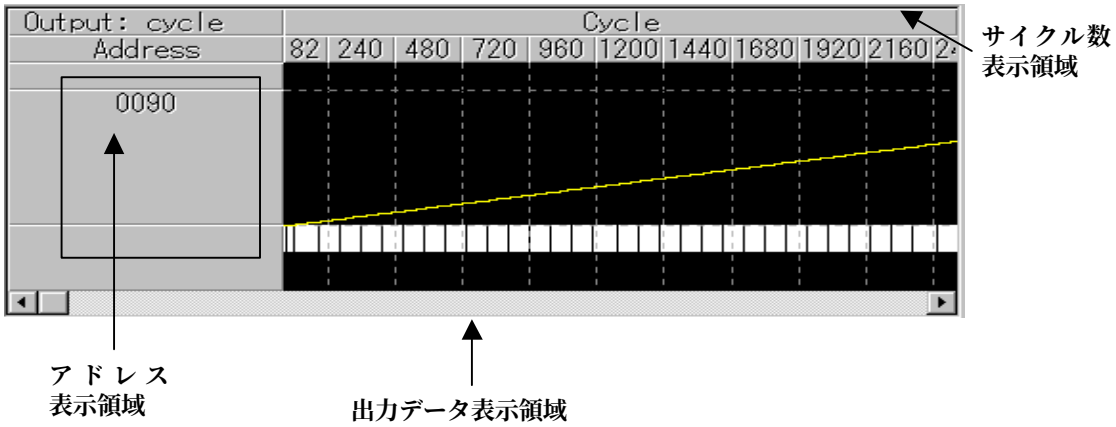

この表示は、メモリのビットが０の状態です。

出力データ表示領域の一番下に表示されている短い白線は、データが出力されたポイントを示しています。

「データ値を参照するには」
この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

サイクル数表示領域
サイクル数を表示します。

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



アドレス表示領域

仮想ポート出力の監視を行うアドレスを表示します。

出力データ表示領域

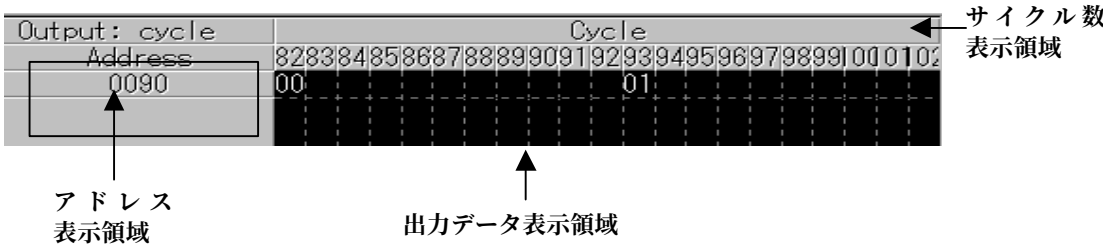
仮想ポート出力結果のデータをバイト単位のグラフ形式で表示します。
表示されるグラフの凸は、データを表示している領域の高さを 255（1 バイトデータの最大値）等分してデータ値を表示しています。
出力データ表示領域の一番下に表示されている短い白線は、データが出力されたポイントを示しています。

「データ値を参照するには」
この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

サイクル数表示領域

サイクル数を表示します。

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



アドレス表示領域

仮想ポート出力の監視を行うアドレスを表示します。

出力データ表示領域

仮想ポート出力結果のデータを 16 進数値で表示します。
「データ値を参照するには」
この領域にマウスカーソルを移動させると、カーソル位置に表示されているデータの値とサイクル数をサイクル数表示領域に表示します。

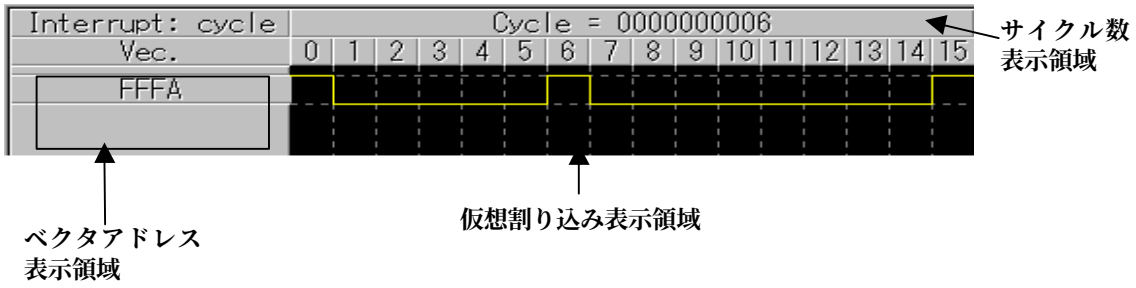
サイクル数表示領域

サイクル数を表示します。

1.14.7 仮想割り込み画面の構成

1.14.7.1 サイクル同期割り込みの画面構成

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

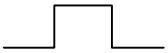


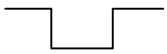
ベクタアドレス表示領域

仮想割り込みのベクタアドレスを表示します。

仮想割り込み表示領域

設定された仮想割り込みの発生タイミングをグラフ形式で表示します。


この表示は、仮想割り込みを発生することを意味します。

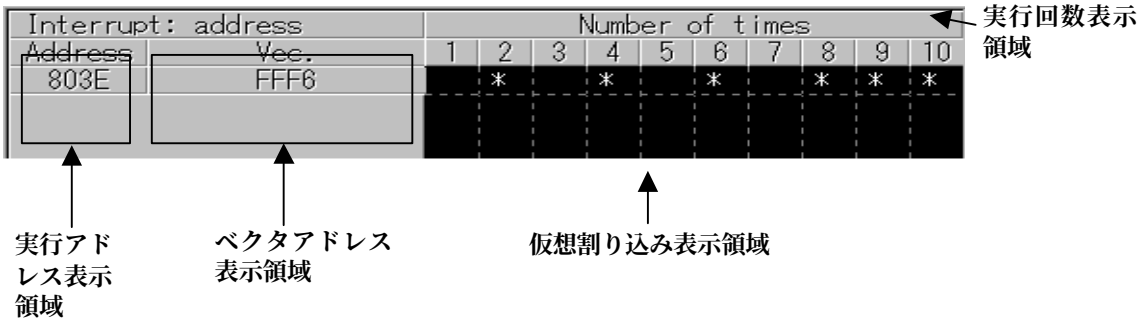

この表示は、仮想割り込みを発生させないことを意味します。

サイクル数表示領域

サイクル数を表示します。

1.14.7.2 実行アドレス同期割り込みの画面構成

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



実行アドレス表示領域-

仮想割り込みの発生タイミングとなるフェッチアドレス（プログラムが実行されるアドレス）を表示します。

ベクタアドレス表示領域

仮想割り込みのベクタアドレスを表示します。

仮想割り込み表示領域

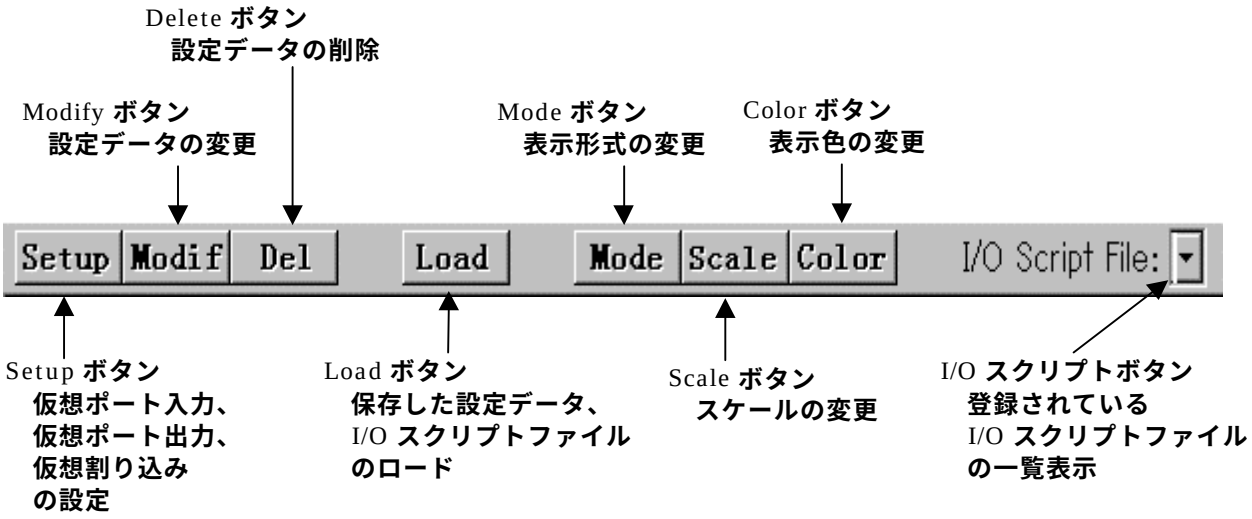
設定された仮想割り込みの発生タイミングをアスタリスク（＊）で表示します。

アスタリスク（＊）が表示されていると、仮想割り込みを発生することを意味します。表示されていない時は、仮想割り込みが発生しないことを意味します。

実行回数表示領域

実行回数（プログラムが指定アドレスを実行した回数）を表示します。

1.14.8 I/O ウィンドウのツールバー



1.14.9 I/O ウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、I/O ウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font.....	フォントの変更	—
	Setup...	仮想ポート入力、仮想ポート出力、仮想割り込みの設定	—
	Modify...	設定した仮想ポート入力、仮想割り込みの変更	—
	Delete...	設定した仮想ポート入力、仮想ポート出力、仮想割り込み、及びユーザが作成したI/O スクリプトファイルの削除	—
	Load...	保存した仮想ポート入力、仮想ポート出力、仮想割り込みファイル、及びユーザが作成したI/O スクリプトファイルの読み込み	—
	Mode...	表示形式の変更	—
	Scale...	表示スケールの変更	—
	Color...	表示色の変更	—

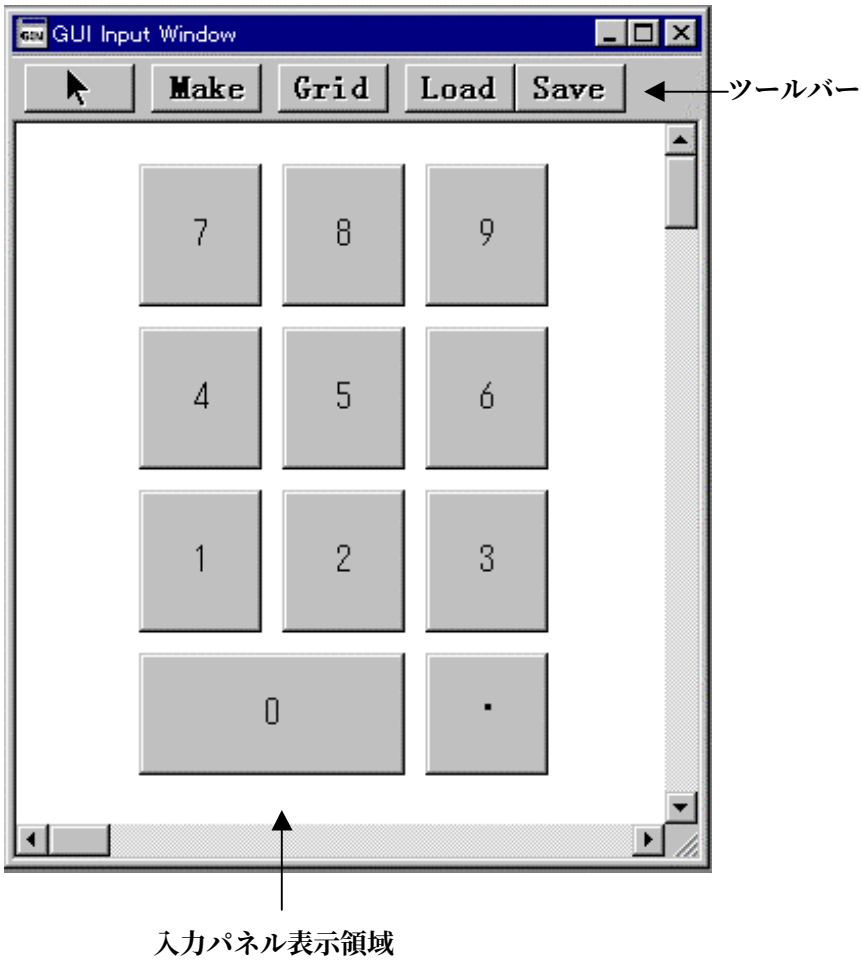
1.15 GUI 入力ウィンドウ

GUI 入力ウィンドウでは、ユーザターゲットシステムの簡単なキー入力パネル（ボタン）をウィンドウ上で作成し、作成したボタンを押すことにより仮想ポート入力や仮想割り込みを行うことができます。
キー入力パネルの作成、設定方法の詳細に関しては、後述の「より高度なデバッグ編」を参照下さい。
ここでは、GUI 入力ウィンドウの画面構成、ツールバー、メニューについて説明します。

GUI 入力ウィンドウのキー入力パネルに作成したボタンを押すことにより、以下の３通りのアクションを行うことができます。

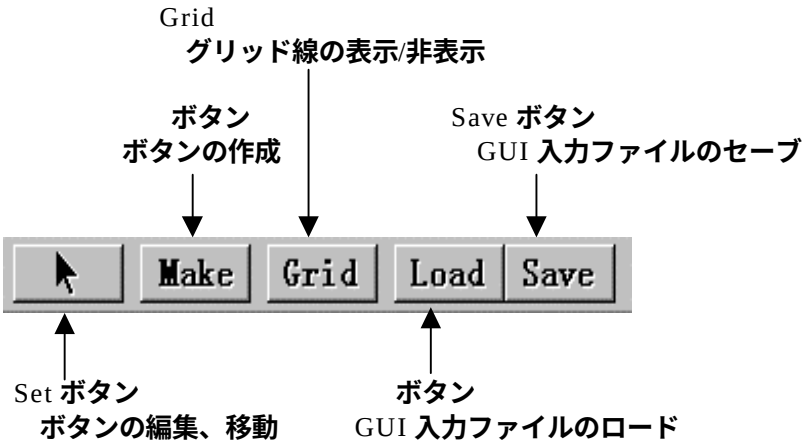
- 仮想ポート入力
- 仮想割り込みの発生
- 仮想割り込みと仮想ポート入力を同時に発生

1.15.1 GUI 入力ウィンドウの画面構成



- 入力パネル表示領域で、ボタンの作成、編集、移動が行えます。
- 作成したボタンには、ラベル（ボタン名）をつけることができます。
- 作成したボタンを押すことで、仮想ポート入力、仮想割り込み、仮想ポート入力＋仮想割り込みを発生させることができます。
- 作成した入力パネルをファイル（GUI 入力ファイル）に保存することができます。

1.15.2 GUI 入力ウィンドウのツールバー



1.15.3 GUI 入力ウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、GUI 入力ウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Set	ボタンの編集、移動	—
	Del	ボタンの削除	—
	Copy	ボタンのコピー	—
	Paste	ボタンのペースト	—
	Make Button	ボタンの作成	—
	Display Grid Line	グリッド線の表示/非表示	—
	Load…	GUI 入力ファイルのロード	—
	Save…	GUI 入力ファイルの保存	—

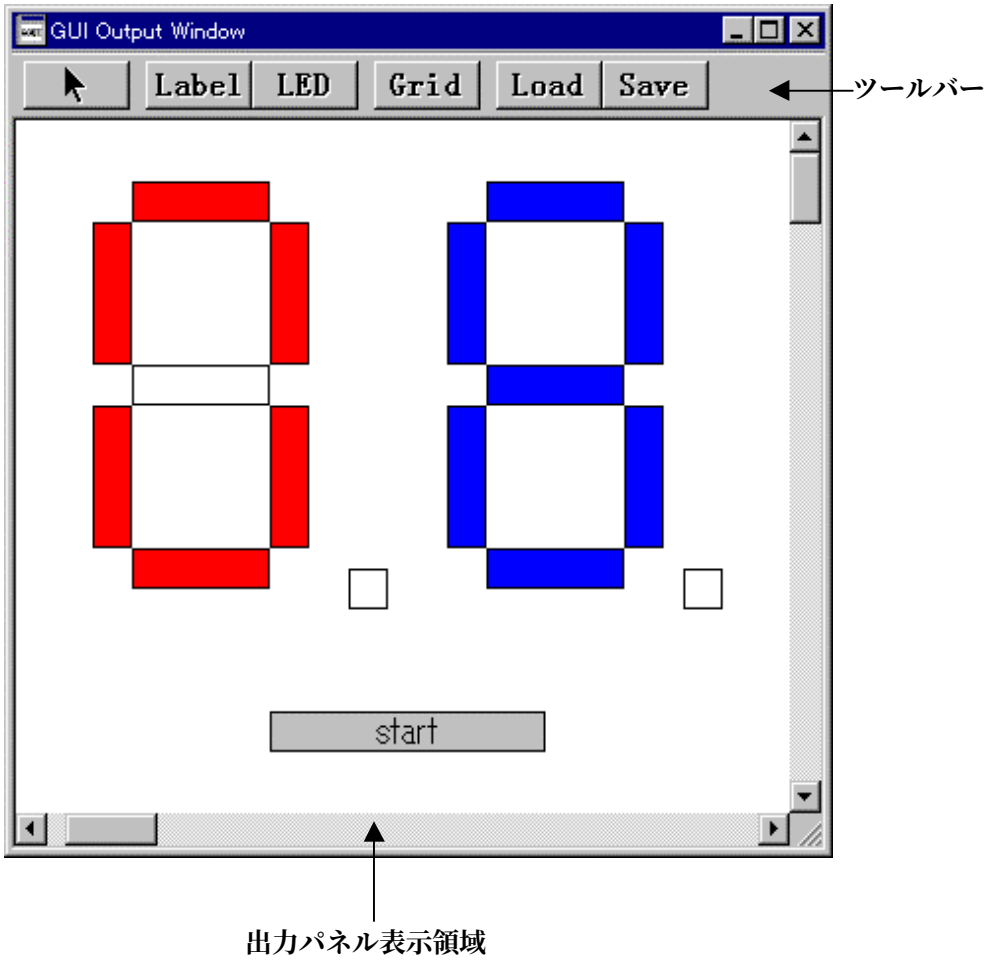
1.16 GUI 出力ウィンドウ

GUI 出力ウィンドウでは、ユーザターゲットシステムの簡単な出力パネルをウィンドウ上で実現できます。
出力パネルには、以下のパーツが配置できます。

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

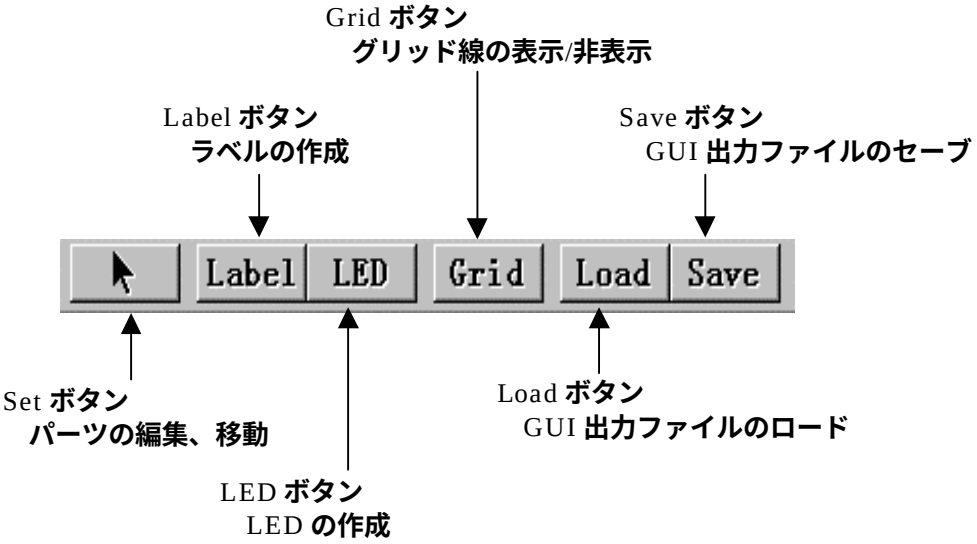
出力パネルの作成、設定方法の詳細に関しては、後述の「より高度なデバッグ編」を参照下さい。
ここでは、GUI 出力ウィンドウの画面構成、ツールバー、メニューについて説明します。

1.16.1 GUI 出力ウィンドウの画面構成



- 出力パネル表示領域で、ラベル・LED の作成、編集、移動が行えます。
- 作成した出力パネルをファイル（GUI 出力ファイル）に保存することができます。

1.16.2 GUI 出力ウィンドウのツールバー



1.16.3 GUI 出力ウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、GUI 出力ウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

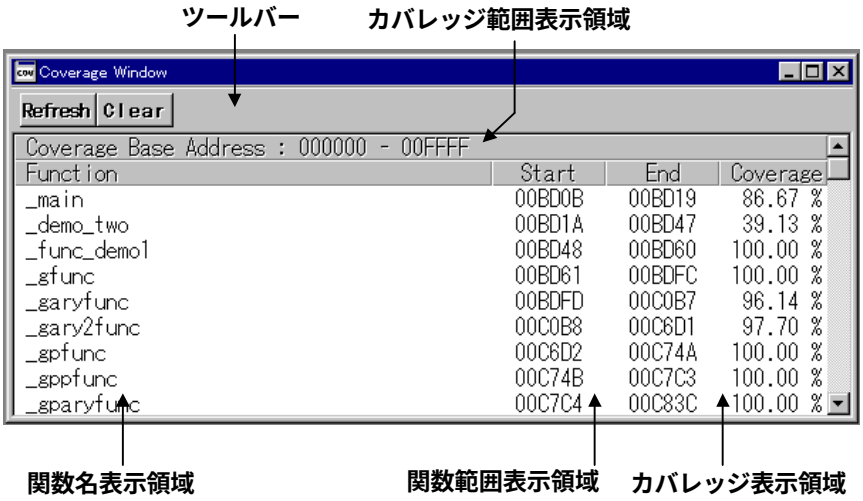
メニュー	メニュー項目	機能	ショートカットキー
Option	Set	パーツの編集、移動	—
	Del	パーツの削除	—
	Copy	ボタンのコピー	—
	Paste	ボタンのペースト	—
	Make Label	ラベルの作成	—
	Make LED	LED の作成	—
	Display Grid Line	グリッド線の表示非表示	—
	Load…	GUI 出力ファイルのロード	—
	Save…	GUI 出力ファイルの保存	—

1.17 カバレッジウィンドウ

カバレッジウィンドウは、現在ダウンロードされているC言語プログラムの各関数のカバレッジ（C0カバレッジ）を測定するためのウィンドウです。各関数の開始/終了アドレスとカバレッジを確認できるカバレッジウィンドウと、ソース行単位で実行/非実行が確認できるカバレッジソースウィンドウの２つのウィンドウが用意されています。

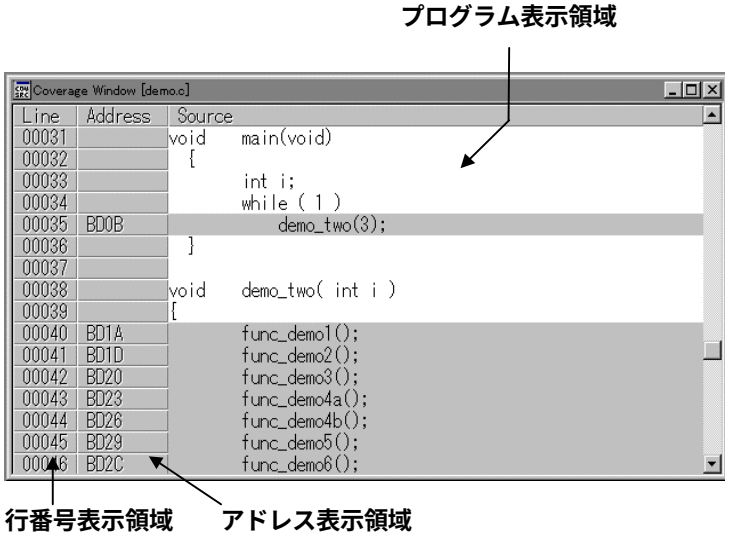
1.17.1 カバレッジウィンドウの画面構成

1.17.1.1 カバレッジウィンドウの画面構成



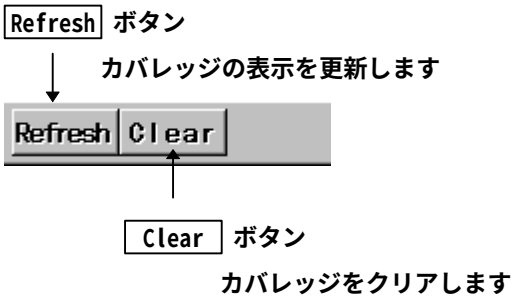
- GO,STEP 等で、ターゲットプログラムを実行した場合は、カバレッジの表示が‘—’に変わります。表示を更新したい時は、ツールバーの Refresh ボタン（またはメニュー [Option] → [Refresh]）を押して下さい。
- 関数の任意の行をダブルクリックする事により、ソース行単位で実行/非実行が確認できるカバレッジソースウィンドウを開く事ができます。
- 関数範囲表示領域は、メニュー [Option] → [Layout] → [Address Area] の選択/解除によって、表示/非表示にすることができます。

1.17.1.2 カバレッジソースウィンドウの画面構成



- 既に実行された行は水色で、実行されていない行は灰色で表示されます。コードの存在しない行（例えばコメント行）は、白色で表示されます。またターゲットプログラムの実行中は、コードの存在する行が灰色表示になります。
- プログラム表示領域の実行/非実行の情報は、プログラムが停止した時点で自動的に表示更新されます。また別の関数を見たい場合は、カバレッジウィンドウで目的の関数をダブルクリックして下さい。（同ソースファイル内であればスクロールでも可能）
- 行番号表示領域、アドレス表示領域は、それぞれメニュー [Option] → [Layout] → [Line Area] 、 [Option] → [Layout] → [Address Area] の選択/解除によって、表示/非表示にすることができます。なお、デフォルトで行番号表示領域は、表示有り、アドレス表示領域はデフォルトで非表示になっています。

1.17.2 カバレッジウィンドウのツールバー



1.17.3 カバレッジウィンドウの拡張メニュー

1.17.3.1 カバレッジウィンドウの拡張メニュー

PD38SIMのメイン表示領域に表示されたウィンドウのうち、カバレッジウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	Refresh	カバレッジ計測結果の表示更新	—
	Clear	カバレッジ計測結果の初期化	—
	File	カバレッジ計測結果のファイル入出力	—
	Save...	カバレッジ計測結果のファイルセーブ	—
	Load...	カバレッジ計測結果のファイルロード	—
	Layout	レイアウト設定	—
	Address Area	アドレス範囲表示領域の表示/非表示	—

1.17.3.2 カバレッジソースウィンドウの拡張メニュー

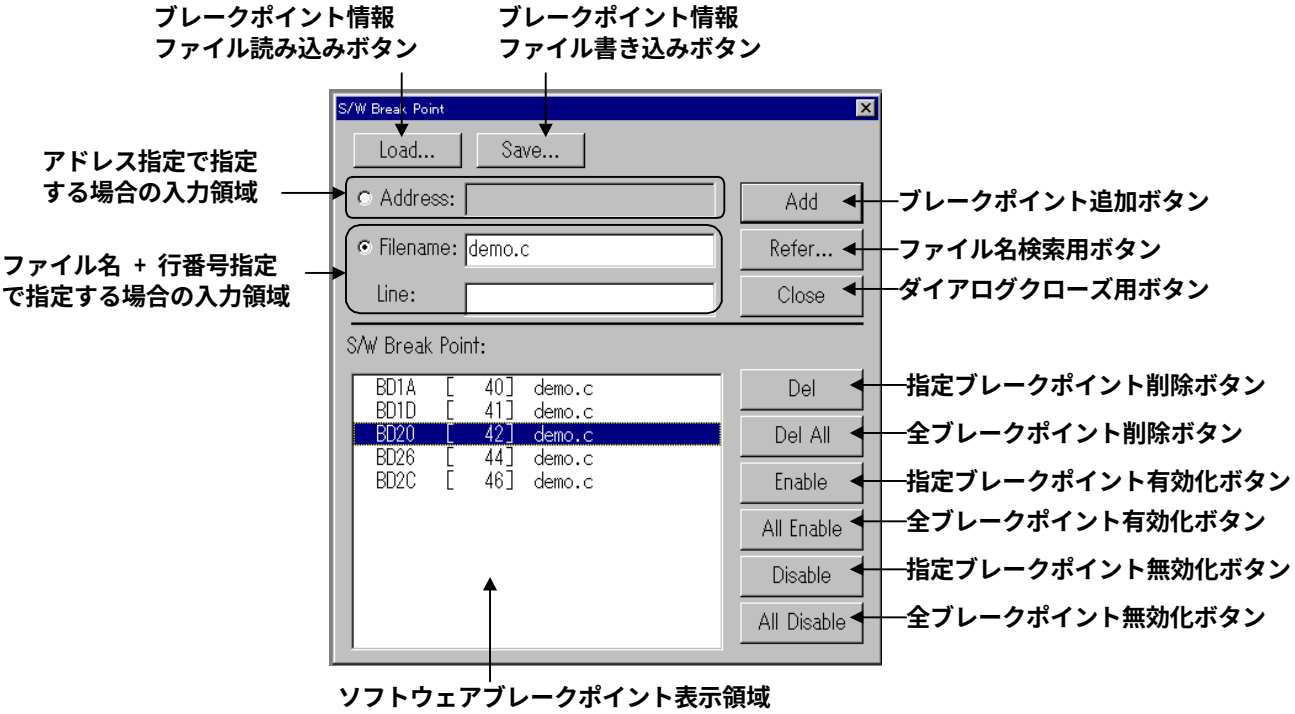
PD38SIMのメイン表示領域に表示されたウィンドウのうち、カバレッジソースウィンドウがアクティブな場合は、［Option］メニューには以下のメニューが割り当てられます。

メニュー	メニュー項目	機能	ショートカットキー
Option	Font...	フォントの変更	—
	TAB...	ソースファイル表示のタブ設定	—
	Layout	レイアウト設定	—
	Line Area	行番号表示領域の表示/非表示	—
	Address Area	アドレス表示領域の表示/非表示	—

1.18 S/W ブレークポイント設定ダイアログ

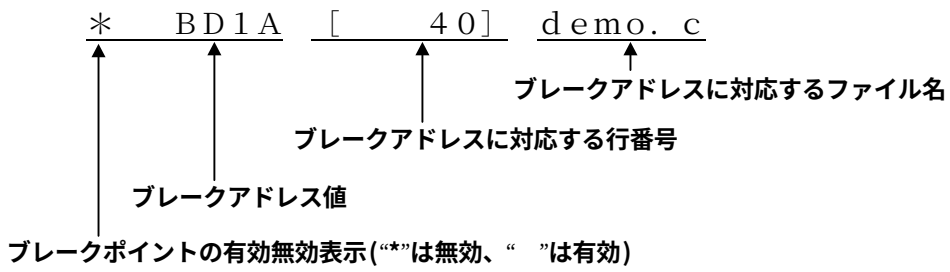
S/W ブレークポイント設定ダイアログは、ソフトウェアブレークポイントを設定するためのダイアログです。ソフトウェアブレークは、設定したブレークポイントの命令を実行する手前でブレークします。それぞれのブレークポイントには、一時的に有効化／無効化を設定することができます。

1.18.1 S/W ブレークポイント設定ダイアログの画面構成



- PD38SIMでは、64 点のソフトウェアブレークポイントが設定できます。
- ソフトウェアブレークポイントを複数設定した場合、組み合わせは OR 条件になります。つまり、いずれか 1 点のソフトウェアブレークアドレスに到達するとプログラム実行を中止します。
- ソフトウェアブレークポイントの設定は、〈Close〉ボタンをクリックして S/W ブレーク設定ダイアログをクローズするまで、連続して指定することができます。
- ソフトウェアブレークポイント表示領域上でクリックして選択したソフトウェアブレークポイントに対して、削除、有効化／無効化が行えます。また、有効化／無効化については、ソフトウェアブレークポイントをダブルクリックすることで、変更することができます。

1.18.2 ソフトウェアブレークポイント一覧の記述

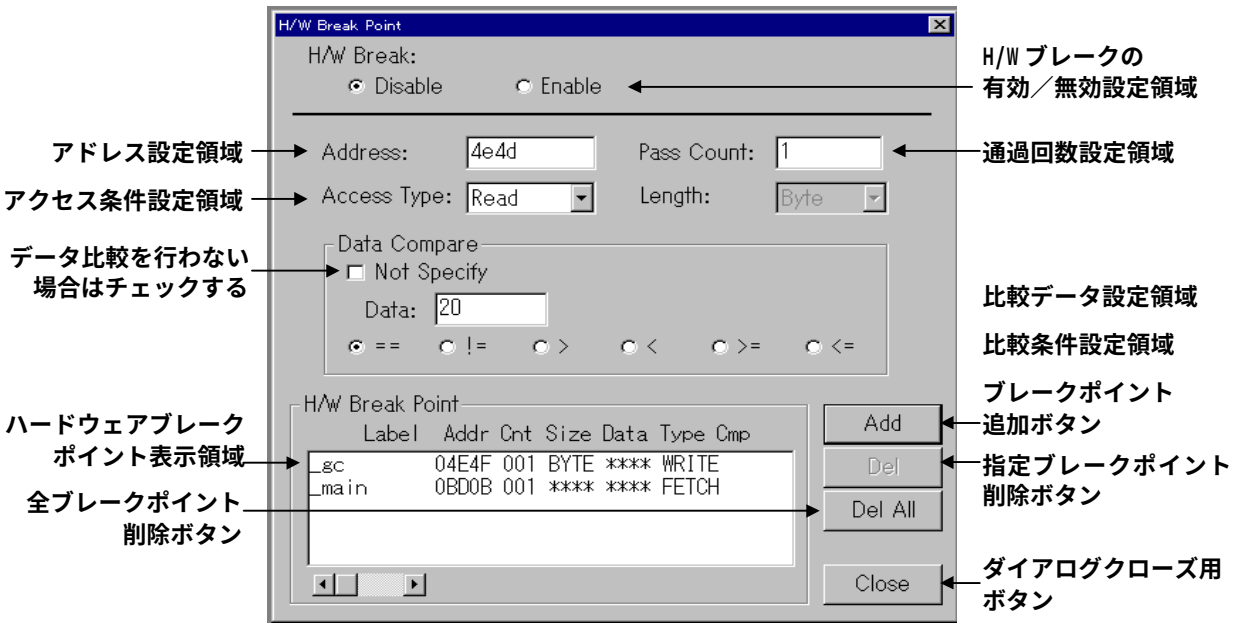


- アドレスに相当するソース行頭がない場合は、アドレス値のみ表示します。

1.19 H/W ブレークポイント設定ダイアログ

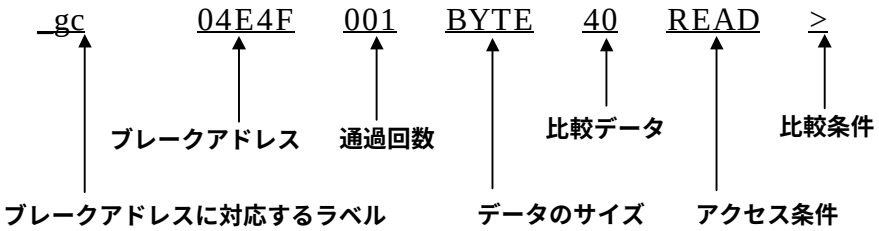
H/W ブレークポイント設定ダイアログは、ハードウェアブレークポイントを設定するためのダイアログです。ハードウェアブレークは、メモリのデータ書き込み／読み込み／命令フェッチを検出したときにブレークします。

1.19.1 H/W ブレークポイント設定ダイアログの画面構成



- PD38SIMでは、64 点のハードウェアブレークポイントが設定できます。
- ハードウェアブレークポイントを複数設定した場合、組み合わせは OR 条件になります。つまり、いずれか 1 点のハードウェアブレークポイントに到達するとプログラム実行を中止します。
- ハードウェアブレークポイントの設定は、〈Close〉ボタンをクリックして H/W ブレーク設定ダイアログをクローズするまで、連続して指定することができます。
- ハードウェアブレークポイント表示領域上でクリックして選択したハードウェアブレークポイントに対して、削除が行えます。

1.19.2 ハードウェアブレークポイント一覧の記述



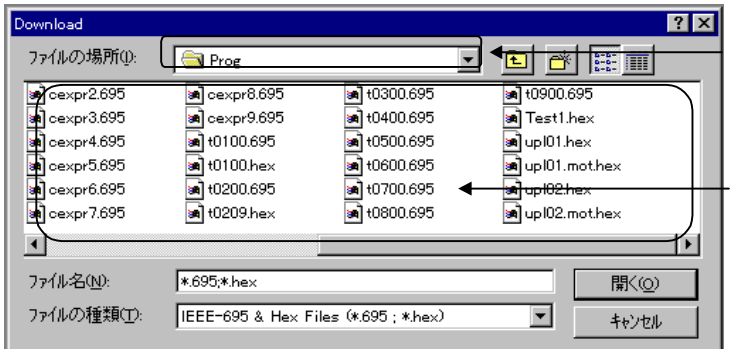
基本操作方法編

このページは白紙です。

1 ターゲットプログラムの読み込み・表示

1.1 ダウンロードするには

ターゲットプログラムをダウンロードするには、PD38SIMウィンドウのメニュー
［File］→［Download］→［Load module...］
を選択してください。ファイルセレクションダイアログがオープンします。ファイルセレクションダイアログからターゲットプログラムを選択してください。ダウンロードの対象ファイルは、インテル HEX フォーマットファイル（以下、HEX ファイルと記述）または IEEE-695 アブソリュート形式ファイル（以下、695 ファイルと記述）です。HEX ファイルのファイル属性は".hex"、695 ファイルのファイル属性は、".695"です。HEX ファイルをダウンロードファイルに指定した場合、HEX ファイルのダウンロードをした後に同名のシンボルファイルを読み込みます。シンボルファイルのファイル属性は、".sym"です。
ダウンロード用ファイルセレクションダイアログは、Shift + F1 キー入力でもオープンします。



1.ファイルの格納ディレクトリを探す

2.ダウンロードするファイルをダブルクリックする

注意事項

ターゲットプログラムのダウンロードが完了するとプログラムウィンドウにターゲットリセット後のプログラムカウンタ位置のソースプログラムを表示します。このプログラムカウンタ位置にソース行情報が存在しない場合（スタートアッププログラムにソース行情報がない場合等）、逆アセンブル表示モードで表示されます。プログラムウィンドウがソース表示モードに切り替わらない場合は、シンボルファイルにソース行情報が存在しない可能性があります。コンパイル・アセンブル・リンク時のオプションをご確認ください。

機械語情報のみをダウンロードするには…

機械語情報のみをダウンロードするには、PD38SIMウィンドウのメニュー

［File］→［Download］→［Memory Image...］

を選択してください。ファイルセレクションダイアログがオープンします。ファイルセレクションダイアログから HEX ファイルまたは 695 ファイルを選択し、機械語情報のダウンロードを行ってください。

シンボル情報のみダウンロードするには…

シンボル情報のみをダウンロードするには、PD38SIMウィンドウのメニュー

［File］→［Download］→［Symbol...］

を選択してください。ファイルセレクションダイアログからシンボルファイルまたは 695 ファイルを選択し、シンボル情報のダウンロードを行ってください。695 ファイルを指定した場合、695 ファイルからシンボル情報だけを読み込みます。

再ダウンロードするには・・・

PD38SIMでは、前回ダウンロードしたファイルを再度ダウンロードする際に、ファイル名を指定せずにダウンロードすることができます。再ダウンロードする際には、PD38SIMウィンドウのメニュー

［File］→［Reload］

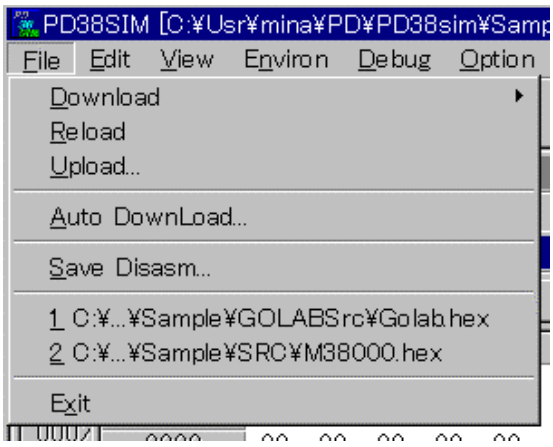
を選択してください。本メニューを選択するとダウンロードが実行されます。

IAR 社製の C コンパイラ ICC740 をご使用の場合は・・・

IAR 社製の C コンパイラ ICC740 をご使用の場合は、コンパイル・リンク時にオプションを指定し、IEEE-695 アブソリュート形式ファイルを作成する必要があります。詳細はPD38SIMのリリースノートをご参照ください。

1.2 最近ダウンロードしたファイルを再ダウンロードするには

PD38SIMウィンドウの[File]メニューに最近ダウンロードしたファイルが表示されます（最大４個まで）。再ダウンロードするにはファイル名を選択してください。選択後、再ダウンロードされます。



注意事項

ファイル名（パス付き）が２５文字を超える場合、上位ディレクトリを省略した形式で表示します。

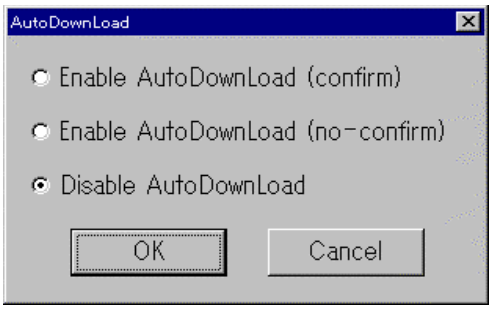
1.3 ロードモジュール更新時に自動ダウンロードするには

PD38SIMでは、実行系のコマンド（Go、Step、Over、Return 等）実行時に、ダウンロードしたターゲットプログラムが更新されている場合、自動的にダウンロードすることができます。

PD38SIMウィンドウのメニュー

［File］→［AutoDownLoad...］

を選択してください。AutoDownLoad ダイアログがオープンします。このダイアログで機能設定することができます。



・Enable AutoDownLoad（confirm）
　ロードモジュール更新時に自動ダウンロードします（確認あり）。
・Enable AutoDownLoad（no-confirm）
　ロードモジュール更新時に自動ダウンロードします（確認なし）。
・Disable AutoDownLoad
　ロードモジュール更新時に自動ダウンロードしません。

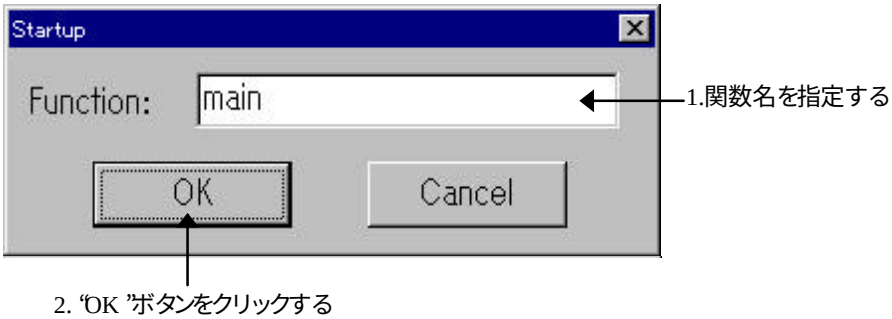
1.4 ダウンロード直後のプログラム表示位置を変更するには

PD38SIMでは、ターゲットプログラムをダウンロードするとターゲットリセット後のプログラムカウンタ位置のソースプログラムをプログラムウィンドウに表示します。このプログラムカウンタ位置にソース行情報が存在しない場合(スタートアッププログラムにソース行情報がない場合等)、逆アセンブルモードで表示されます。

ターゲットプログラムのダウンロード後、自動的に main 関数等のソースプログラムを表示したい場合は、予め関数名を指定しておく必要があります。PD38SIMウィンドウのメニュー

［Environ］→［StartUp...］

を選択してください。StartUp ダイアログがオープンします。StartUp ダイアログで表示する関数名を指定してください。



StartUp ダイアログで指定した関数がプログラムウィンドウに表示されてもプログラムカウンタ値はダウンロード直後のプログラムカウンタ値のままです。StartUp ダイアログで指定した関数までプログラムカウンタ値を進める場合は、カム実行を行ってください。カム実行については、本マニュアル基本操作編の項目「2.4 指定位置までプログラムを実行するには」をご参照下さい。

注意事項

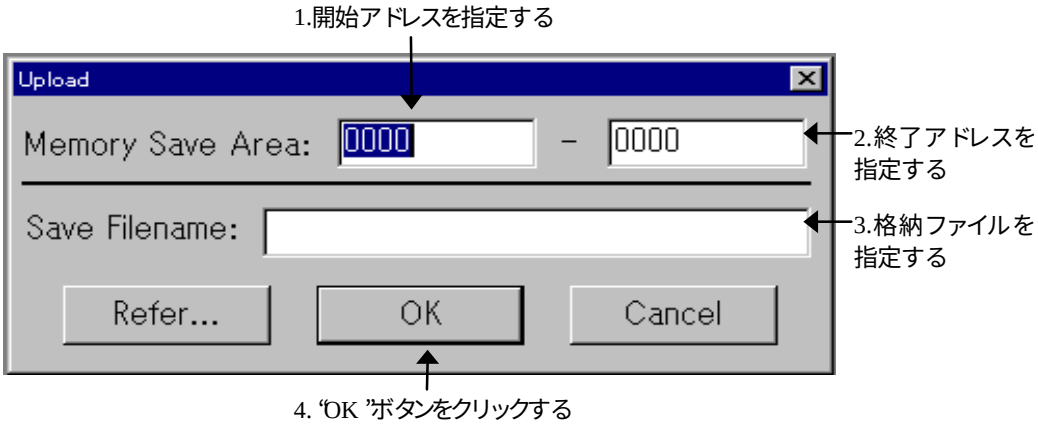
ダウンロード直後のプログラムカウンタ位置にソース行情報が存在する場合、この設定は無視されます。

1.5 アップロードするには

PD38SIM ウィンドウのメニュー

［File］→［Upload...］

を選択してください。アップロードダイアログがオープンします。アップロードする領域と保存するファイル名を入力してください。保存ファイルは、インテル HEX フォーマットファイルが指定できます。インテル HEX フォーマットファイル形式で保存する場合は、拡張子に“.hex”を指定してください。なお、ファイル名に既存のファイル名を指定した場合は、上書きします。

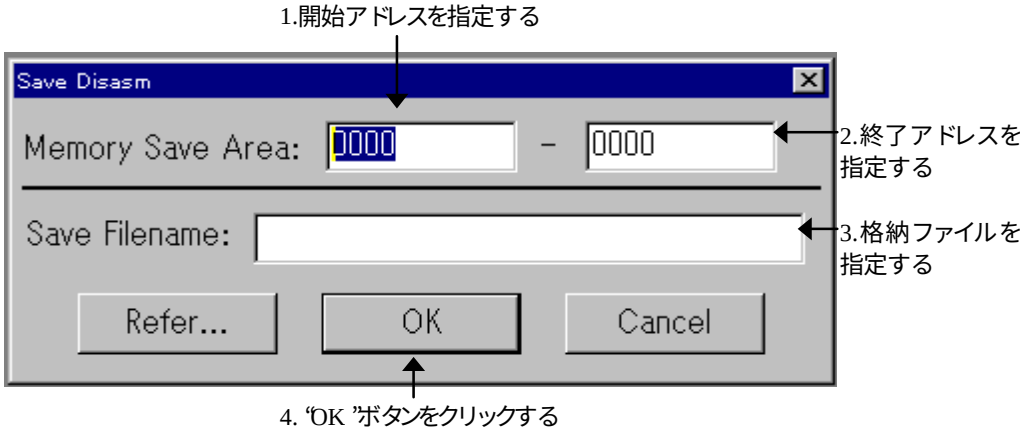


1.6 逆アセンブル結果を保存するには

PD38SIM ウィンドウのメニュー

［File］→［Save Disasm...］

を選択してください。逆アセンブル結果保存ダイアログがオープンします。保存する領域とファイル名を入力してください。保存ファイル名には、任意のファイル名、ファイル属性が指定できます。なお、ファイル名に既存のファイル名を指定した場合は、上書きします。



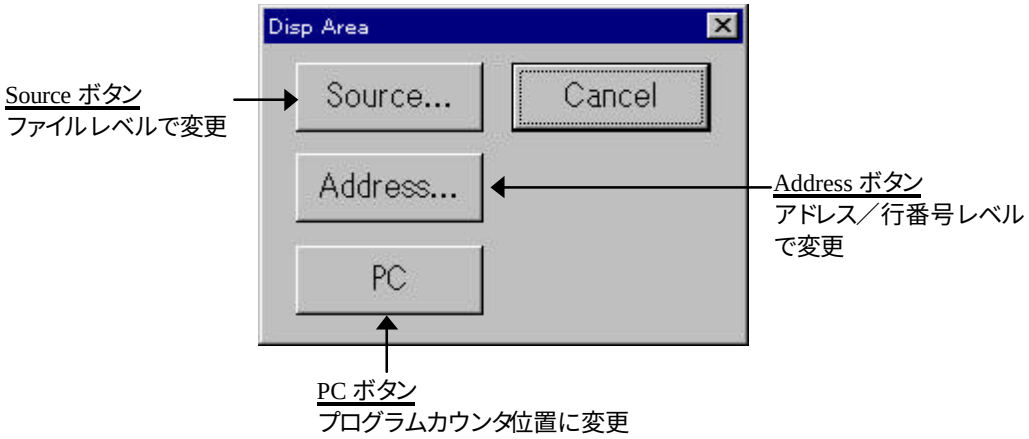
1.7 プログラムの任意位置を常に表示するには

プログラムウィンドウは、プログラムカウンタ位置を追従して表示位置を切り替えますのでターゲットプログラムの任意位置を常に表示することはできません。ターゲットプログラムの任意位置を常に表示するには、ソースウィンドウをご使用ください。ソースウィンドウは、PD38SIMウィンドウのメニュー
［Basic Window］→［Source Window］
の選択でオープンします。

1.8 プログラムの表示位置を変更するには

ソースプログラムの内容は、プログラムウィンドウ及びソースウィンドウで表示できます。プログラム(ソース)ウィンドウの表示位置を変更するには、プログラム(ソース)ウィンドウのツールバーから“View”ボタンをクリックしてください。表示位置の変更は、アクティブウィンドウのみ有効です。

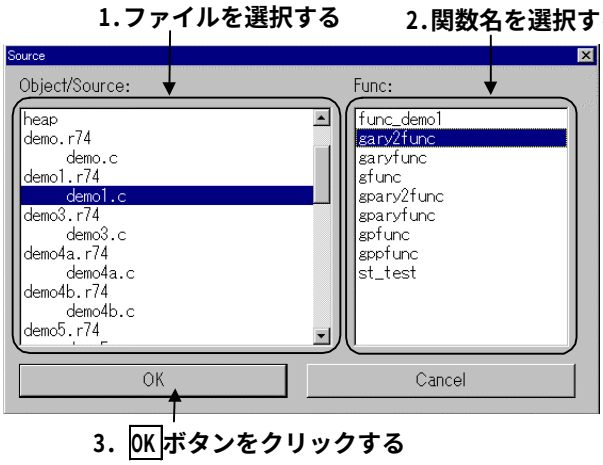
“View”ボタンをクリックすると以下の Disp Area ダイアログがオープンします



なお、デバッグ情報を読み込んでいない場合は、ファイルレベル、および行番号レベルでの変更は行えません。また、プログラム（ソース）ウィンドウの表示モードが逆アセンブル表示モードの場合、行番号レベルでの変更は行えません。

ファイルレベルで表示位置を変更するには・・・

Disp Area ダイアログから Source ボタンをクリックしてください（Source ボタンは、デバッグ情報を読み込んでいない場合はクリックできません）。以下の Source ダイアログがオープンします。Source ダイアログでは、読み込んだターゲットプログラムのファイル構成、及び関数情報をを表示しています。変更するファイル名及び関数名をクリックしてください。



Source ダイアログは、プログラム（ソース）ウィンドウの行番号表示領域のダブルクリックまたは、PD38SIMウィンドウのメニュー

［Option］→［View］→［Source...］

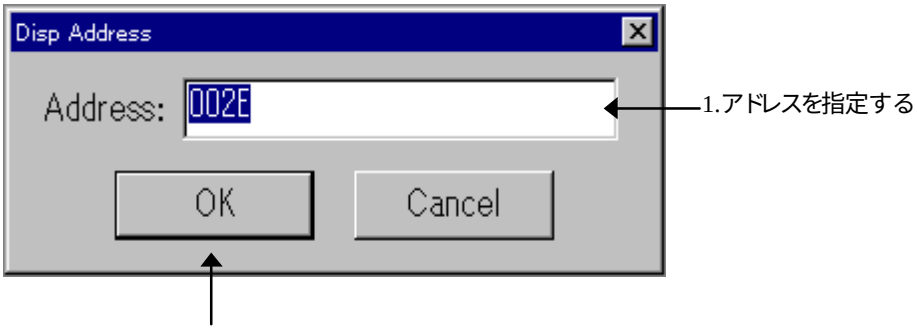
の選択でもオープンします。

注意事項

695 ファイルを読み込んでない場合は、関数名は表示されません。ファイル名のみを選択して下さい。

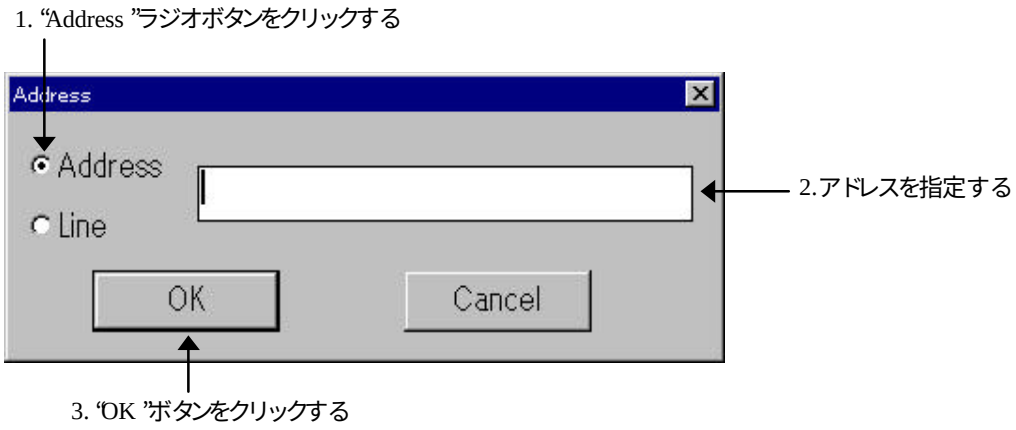
アドレスレベルで表示位置を変更するには…

プログラム（ソース）ウィンドウの表示モードが逆アセンブル表示モードのとき、Disp Area ダイアログから“Address”ボタンをクリックしてください。“Address”ボタンをクリックすると以下の Disp Address ダイアログがオープンします。



注意事項

ソース表示モードのときは、以下の Address ダイアログがオープンします。Address ダイアログの Address ラジオボタンをクリックして、変更アドレスを入力してください。

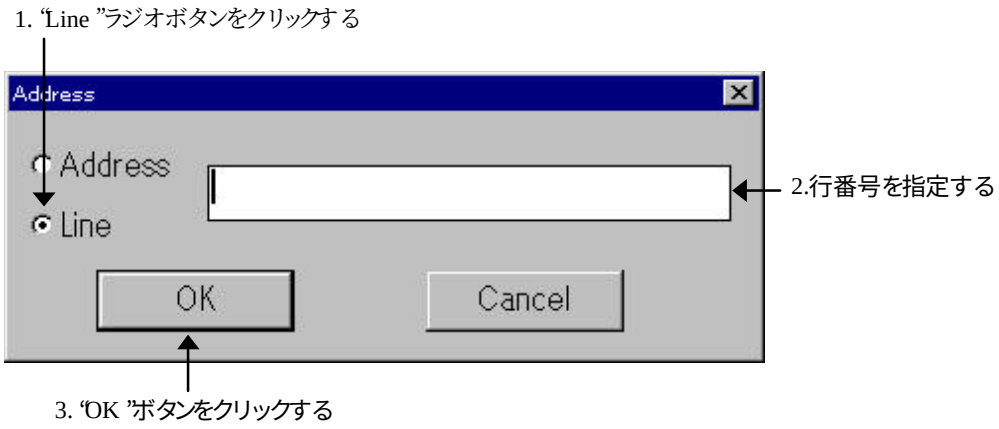


Disp Address（Address）ダイアログは、PD38SIMウィンドウのメニュー
［Option］→［View］→［Address...］
の選択でもオープンします。

また、Disp Address（Address）ダイアログは、プログラム（ソース）ウィンドウのアドレス表示領域をダブルクリックすることでもオープンします。

行番号レベルで表示位置を変更するには…

プログラム（ソース）ウィンドウの表示モードがソース表示モードのとき、Disp Area ダイアログから“ Address ”ボタンをクリックしてください。“ Address ”ボタンをクリックすると以下の Address ダイアログがオープンします。Address ダイアログの Line ラジオボタンをクリックして、変更する行番号を入力してください。



Address ダイアログは、**PD38SIM** ウィンドウのメニュー

［Option］→［View］→［Address...］

の選択でもオープンします。

また、Address ダイアログは、プログラム（ソース）ウィンドウのアドレス表示領域をダブルクリックすることでもオープンします。

プログラムカウンタ位置に変更するには…

Disp Area ダイアログから“PC”ボタンをクリックしてください。“PC”ボタンをクリックするとウィンドウの表示位置がプログラムカウンタ位置に変更します。**PD38SIM** ウィンドウのメニュー

［Option］→［View］→［Program Counter］

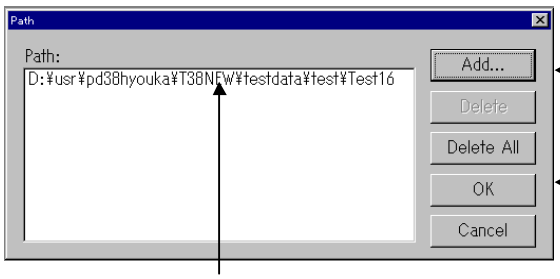
の選択でも変更できます。

1.9 他ディレクトリに存在するソースプログラムを参照するには

ソースファイルのサーチパスを指定します。ターゲットプログラムのソースファイルがカレントディレクトリにない場合、複数のサブディレクトリに別れて格納されているといった場合に有効です。これにより、ソースプログラムの参照やウィンドウ上からのソフトウェアブレークポイント設定等ができます。サーチパスを指定するには、**PD38SIM**ウィンドウのメニュー

［Environ］→［Path...］

を選択してください。Path ダイアログがオープンします。サーチパスを追加する場合は、Path ダイアログの“Add”ボタンをクリックしてください。ファイルセレクションダイアログがオープンしますので参照するファイル名をマウスで選択してください。



注意事項

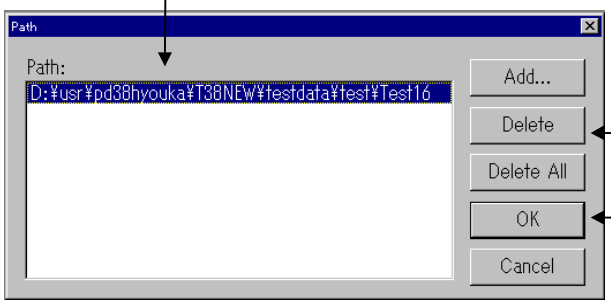
PD38SIMは、プログラムウィンドウおよびソースウィンドウにソースファイルを表示する際、下記の優先順位でディレクトリを検索します。

1. デバッグ情報内に記述されたパス
2. ターゲットプログラムのパス
3. サーチパスで指定したパス（指定順）

サーチパスを削除するには…

Path ダイアログから以下の操作を行ってください。

1. 削除するサーチパスをクリックする



2. ”Delete“ボタンをクリックする

3. “OK”ボタンをクリックする

＊　　全サーチパスを削除するには、”Delete All”ボタンをクリックする。

1.10 ソースと逆アセンブル結果をMIX 表示するには

ソースファイルとその逆アセンブル結果を混合して表示するには、プログラム（ソース）ウィンドウのツールバーから“MIX”ボタンをクリックしてください。MIX 表示モードの切り替えは、**PD38SIM**ウィンドウのメニュー

［Option］→［Mode］→［MIX Mode］

の選択でも行うことができます。表示モードの変更は、アクティブウィンドウのみ有効です。

ターゲットプログラム停止後のプログラムカウンタの停止位置が、ソース行情報が出力されている領域でかつソースの行頭アドレスに一致しない場合、プログラムウィンドウの表示モードが自動的に MIX 表示モードとなります。

MIX 表示をソースプログラム表示に戻すには・・・

プログラム（ソース）ウィンドウのツールバーから“SRC”ボタンをクリックしてください。

PD38SIMウィンドウのメニュー

［Option］→［Mode］→［Source Mode］

の指定でもソースプログラム表示となります。

注意事項

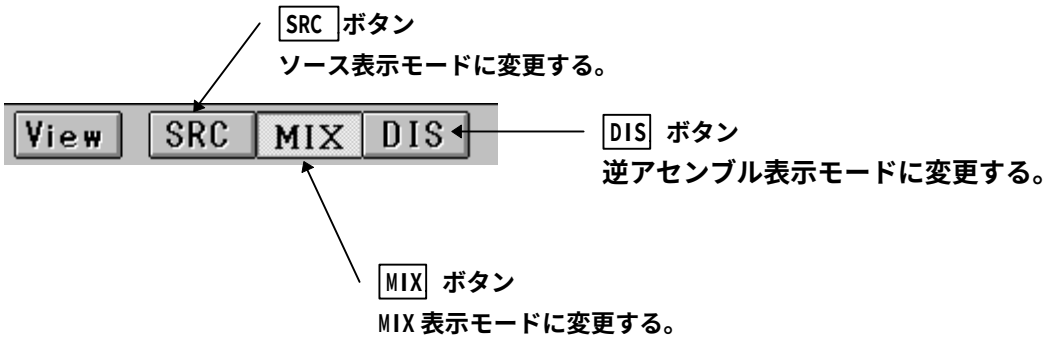
- マクロの定義部に PC 値がある場合、PC 値を示す黄色のラインが表示されない場合があります。

1.11 逆アセンブル結果を表示するには

逆アセンブル結果を表示するには、プログラム（ソース）ウィンドウのツールバーから“DIS”ボタンをクリックしてください。逆アセンブル結果は、**PD38SIM**ウィンドウのメニュー

［Option］→［Mode］→［Disasm Mode］

の選択でも行うことができます。表示モードの変更は、アクティブウィンドウのみ有効です。



ターゲットプログラム停止後のプログラムカウンタ位置が、ソース行情報の存在しない領域であった場合、プログラムウィンドウの表示モードが自動的に逆アセンブル表示モードとなります。

逆アセンブル表示をソースプログラム表示に戻すには・・・

プログラム（ソース）ウィンドウのツールバーから“SRC”ボタンをクリックしてください。

PD38SIMウィンドウのメニュー

［Option］→［Mode］→［Source Mode］

の指定でもソースプログラム表示となります。

逆アセンブル表示を MIX 表示に戻すには・・・

プログラム（ソース）ウィンドウのツールバーから“MIX”ボタンをクリックしてください。

PD38SIMウィンドウのメニュー

［Option］→［Mode］→［MIX Mode］

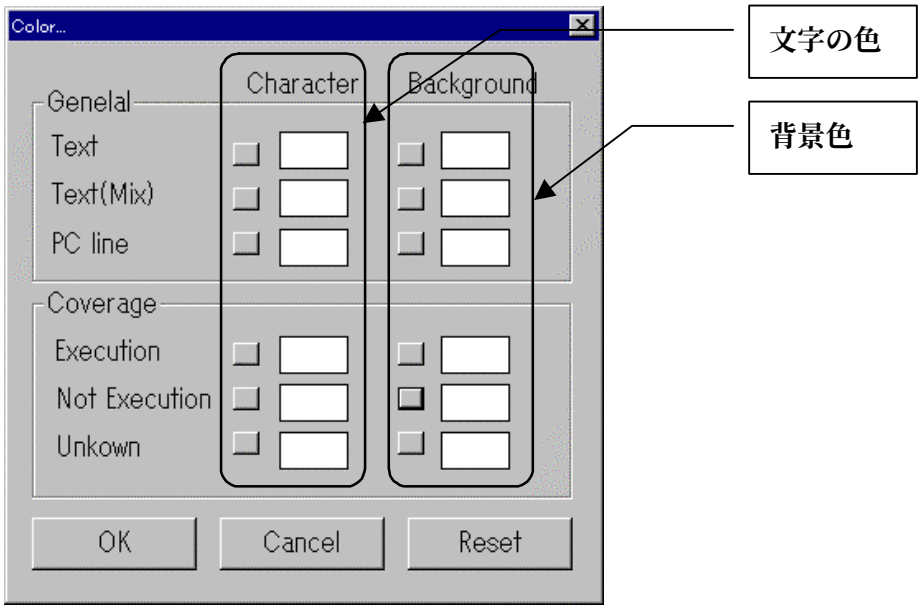
の指定でもソースプログラム表示となります。

注意事項

プログラム（ソース）ウィンドウのプログラム表示領域の先頭行に、ソース行情報が存在しない場合､逆アセンブル表示モードからソース表示モードおよび MIX 表示モードに変更することができません（“SRC”、“ MIX ”ボタンが無効となります）。この場合、垂直スクロールバー、上下矢印キーを用いてプログラム表示領域の先頭アドレスをソース行情報が存在する位置に変更してください。

1.12 表示色のカスタマイズ

プログラムウィンドウの表示色をカスタマイズするには、メニュー［Option］→［Color…］を選択してください。カラー設定ダイアログがオープンします。なお、表示色の設定は、すべてのソースウィンドウに反映されます。



1.12.1 カラー設定ダイアログの機能

- 各表示色の左横のボタンを押下することにより、“色の設定”ダイアログがオープンし、表示色が変更できます。
- Reset ボタンを押下することにより、デフォルトの設定色に戻すことができます

2 ターゲットプログラムの実行/停止

2.1 実行・停止するには

ターゲットプログラムを実行するには・・・

PD38SIMウィンドウのツールバーから“Go”ボタンをクリックしてください。ターゲットプログラムは、F1 キーの入力でも実行できます。

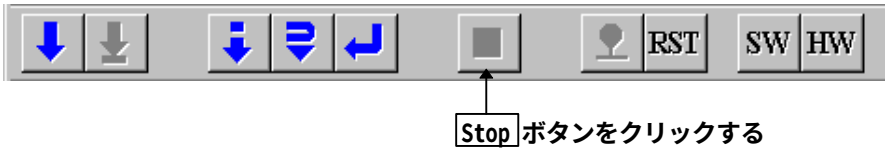
また、**PD38SIM**ウィンドウのメニュー
［Debug］→［Go］
の選択でもターゲットプログラムを実行できます。
実行したターゲットプログラムは、ブレークポイントに到達すると停止します。

ターゲットプログラムをフリーランするには・・・

PD38SIMウィンドウのメニュー
［Debug］→［GoFree］
の選択でターゲットプログラムをフリーラン実行できます。
フリーラン実行では、全てのブレークポイント（S/W,H/W）が無効になります。

ターゲットプログラムを停止するには・・・

PD38SIMウィンドウのツールバーから“Stop”ボタンをクリックしてください。



また、**PD38SIM**ウィンドウのメニュー
［Debug］→［Stop］
の選択でもターゲットプログラムを停止できます。

注意事項

ターゲットプログラム停止後のプログラムカウンタ位置が、ソース行情報の存在しない領域であった場合、プログラムウィンドウの表示モードが MIX 表示モードとなります。

ターゲットプログラムを任意アドレスから実行するには・・・

PD38SIMウィンドウのメニュー
［Debug］→［Go］→［Go Option...］
を選択してください。Go ダイアログがオープンしますので開始アドレスを入力してください。

ターゲットプログラムの実行状態（実行中か否か）を参照するには・・・

PD38SIMウィンドウのステータスバー右部を参照してください。現在の実行状態を表示しています。

MCU : RUN　ターゲットプログラムが実行中の場合

MCU : STOP　ターゲットプログラムが停止中の場合

2.2 ステップ実行するには

PD38SIMウィンドウのツールバーから“Step”ボタンをクリックしてください。ステップ実行は、F3 キー入力でも実行できます。

プログラムウィンドウがソース表示モードの場合は、ソース行単位でステップ実行します。プログラムウィンドウが逆アセンブル表示モードの場合は、命令単位でステップ実行します。
また、**PD38SIM**ウィンドウのメニュー
［Debug］→［Step］→［Step］
の選択でもステップ実行が行えます。

サブルーチンを 1 命令としてステップ実行するには・・・

サブルーチンを 1 命令としてステップ実行することをオーバーステップ実行と呼びます。
PD38SIMウィンドウのツールバーから“Over”ボタンをクリックしてください。オーバーステップ実行は、F4 キー入力でも実行できます。

また、PD38SIMウィンドウのメニュー
［Debug］→［Over］→［Over］
の選択でもオーバーステップ実行が行えます。

ステップ実行の回数を指定したい場合は・・・

PD38SIMウィンドウのメニュー
［Debug］→［Step］→［Step Option...］
（オーバーステップ実行の場合、［Debug］→［Over］→［Over Option...］）
を選択してください。オープンしたStepダイアログ（Overダイアログ）でステップ回数を指定してください。

ステップ実行を停止したい場合は・・・

ツールバーの“Stop”ボタンをクリックしてください。PD38SIMウィンドウのメニュー
［Debug］→［Stop］
を選択することにより、ステップ実行を途中で停止させることもできます。オーバーステップ実行の場合も同様です。

2.3 現ルーチンから上位ルーチンへ戻るには

PD38SIMウィンドウのツールバーから“Return”ボタンをクリックしてください（リターン実行と呼びます）。リターン実行は、F5キーの入力でも実行できます。

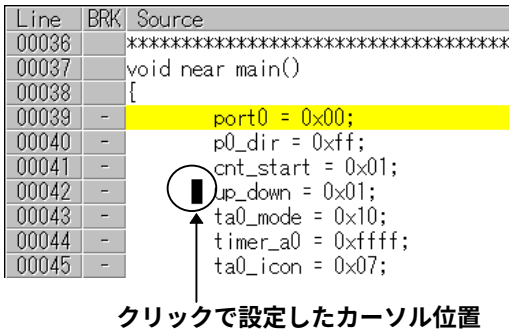
また、PD38SIMウィンドウのメニュー
［Debug］→［Return］
の選択でもリターン実行が行えます。

リターン実行を途中で停止するには・・・

ツールバーの“Stop”ボタンをクリックしてください。
また、PD38SIMウィンドウのメニュー
［Debug］→［Stop］
の選択でもリターン実行を停止させることができます。

2.4 指定位置までプログラムを実行するには

指定位置までターゲットプログラムを実行する（カム実行と呼びます）には、まず、プログラム（ソース）ウィンドウのプログラム表示領域の停止させたい行をクリックして、カーソル位置を指定してください。ただし、ソフトウェアブレークポイントが設定できない行（コメント行、データ定義行等）にカーソル位置を指定した場合は、カム実行は行えません。



カム実行を開始するには、**PD38SIM**ウィンドウのツールバーから“Come”ボタンをクリックしてください。カム実行は、F2 キーの入力でも実行できます。

また、**PD38SIM**ウィンドウのメニュー
［Debug］→［Come］
の選択でもカム実行が行えます。

カム実行を途中で停止するには・・・

ツールバーの“Stop”ボタンをクリックしてください。
また、**PD38SIM**ウィンドウのメニュー
［Debug］→［Stop］
の選択でもカム実行を停止させることができます。

2.5 プログラムをリセットするには

ターゲットプログラムをリセットするには、**PD38SIM**のツールバーから“Reset”ボタンをクリックしてください。リセットは、F8 キーの入力でも実行できます。

また、**PD38SIM**ウィンドウのメニュー
［Debug］→［Reset］
の選択でもターゲットプログラムをリセットできます。

3 レジスタ情報、メモリ内容の参照・設定

3.1 レジスタの内容を参照するには

レジスタの内容を参照するには、レジスタウィンドウをオープンします。レジスタウィンドウでは、CPU が持つレジスタの一覧を表示しています。レジスタウィンドウは、PD38SIMウィンドウのメニュー

［BasicWindow］→［Register Window］

の選択でオープンします。

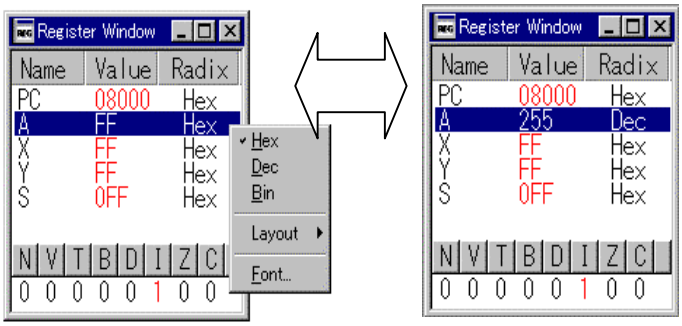
レジスタウィンドウについては、本マニュアル ウィンドウ機能編の項目「1.4 レジスタウィンドウ」をご参照下さい。

レジスタの表示基数を変更するには・・・

レジスタの表示基数は、レジスタ毎に変更が可能です。表示基数は、以下の操作で変更できます。

1. 基数を変更したいレジスタ上でマウスを右クリックして下さい。
2. メニューから変更したい基数を選択して下さい。

A レジスタを 16 進表示



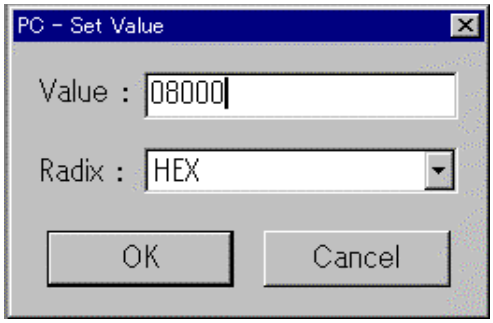
A レジスタを 10 進表示

3.2 レジスタの内容を変更するには

レジスタの内容を変更するには・・・

レジスタの内容は、以下の操作で変更できます。

1. 変更するレジスタが表示されている行をダブルクリックするか、行を選択した状態で Enter を入力して下さい。
2. レジスタの値を設定するダイアログがオープンするので、変更する値と基数を入力してください。



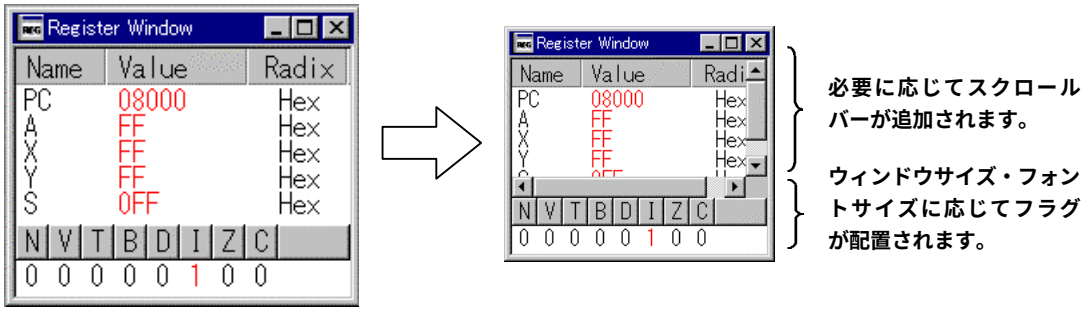
フラグの値を変更するには・・・

- フラグ表示部を表示している場合
  変更するフラグのボタンをクリックして下さい。クリックするたびにフラグの状態が切り替わります。

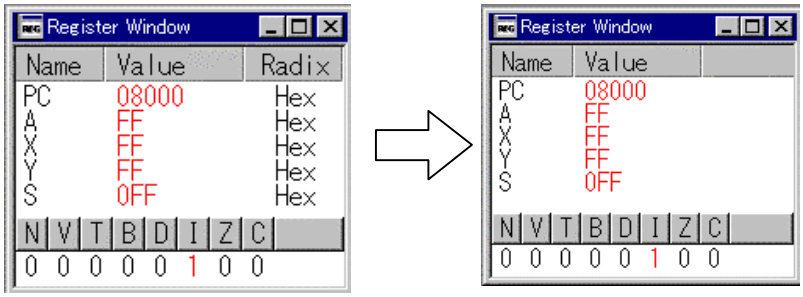
- フラグ表示部が非表示の場合
  レジスタの内容と同じ方法で変更できます。FLG レジスタが表示されている行をダブルクリックするか、行を選択した状態で Enter を入力して下さい。

レジスタウィンドウのレイアウトを変更するには・・・

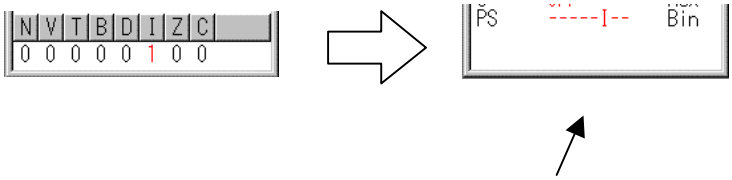
- ウィンドウのサイズを変更すると・・・
  ウィンドウサイズを変更すると、変更後のウィンドウサイズに応じてウィンドウ内のレイアウトが変わります。



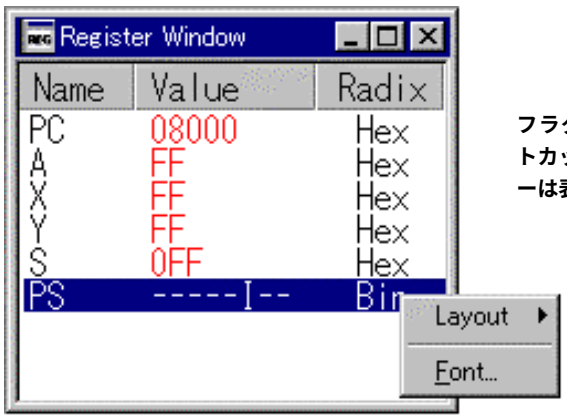
- 基数を非表示にする。
  レジスタウィンドウをアクティブにした状態で、メニュー[Option]→[Layout]→[Hide Radix]を選択するか、レジスタ表示部でマウスを右クリックして、[Layout]→[Hide Radix]を選択して下さい。



- フラグ表示部を非表示にする。
  レジスタウィンドウをアクティブにした状態で、メニュー[Option]→[Layout]→[Hide FLAGs]を選択するか、レジスタ表示部でマウスを右クリックして、[Layout]→[Hide FLAGs]を選択して下さい。
  なお、フラグレジスタの表示基数は変更できません。

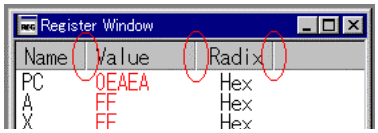


フラグレジスタの表示。現在セットされているフラグをフラグ名で表示します。



フラグレジスタ上で表示されるショートカットメニューには、基数変更メニューは表示されません。

- 各列の幅を調節する。
  レジスタウィンドウの各列（Name, Value, Radix）幅を調節できます。セパレータの部分をドラッグして下さい。また、セパレータの部分をダブルクリックすると、文字列長に応じて自動調整されます。



3.3 プログラム実行中にRAM の変化を参照するには

ターゲットプログラム実行中にメモリ内容の変化を参照するには、RAM モニタ機能を使用します。PD38SIMでは、RAM モニタ領域を 1K バイト用意しています。デフォルトの RAM モニタ領域は、0～$3FF_{16}$です。実行時のメモリ内容の変化は、RAM モニタウィンドウで参照します。RAM モニタウィンドウをオープンするには、PD38SIMウィンドウのメニュー

［BasicWindow］→［RAM Monitor Window］

の選択でオープンします。

RAM モニタウィンドウについては、本マニュアル ウィンドウ機能編の項目「1.7 RAM モニタウィンドウ」をご参照下さい。

RAM モニタ領域を変更するには・・・

RAM モニタウィンドウのツールバーの Area ボタンをクリック、もしくは RAM モニタウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［RAM Monitor Area...］

を選択してください。オープンした Ram Monitor Area ダイアログで RAM モニタ領域の先頭アドレスを入力します。

また、RAM モニタウィンドウのアドレス表示領域をダブルクリックして、表示開始アドレスの変更を行うことも可能です。指定した表示開始アドレスが現在の RAM モニタ領域の範囲外の場合は、RAM モニタ領域が変更されます。このとき、RAM モニタ領域を変更するかどうかの確認のダイアログが表示されるので、OK ボタンをクリックしてください。

サンプリング周期を変更するには・・・

RAM モニタウィンドウに表示されるメモリ内容は、一定間隔で自動的に更新します。デフォルトのサンプリング周期は、100[ms]です。サンプリング周期を変更するには、RAM モニタウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［Sampling Period...］

を選択してください。サンプリング周期は、オープンした Sampling Period ダイアログで入力します。但し、動作状況によっては指定した更新間隔よりも遅くなる場合があります。

3.4 任意アドレスの値を参照するには

任意アドレスの値を参照するには、ウォッチ機能を使用します。任意アドレスの値は、ASM ウォッチウィンドウで参照します。この参照するアドレスをウォッチポイントと呼びます。ASM ウォッチウィンドウは、PD38SIMウィンドウのメニュー

［BasicWindow］→［ASM Watch Window］

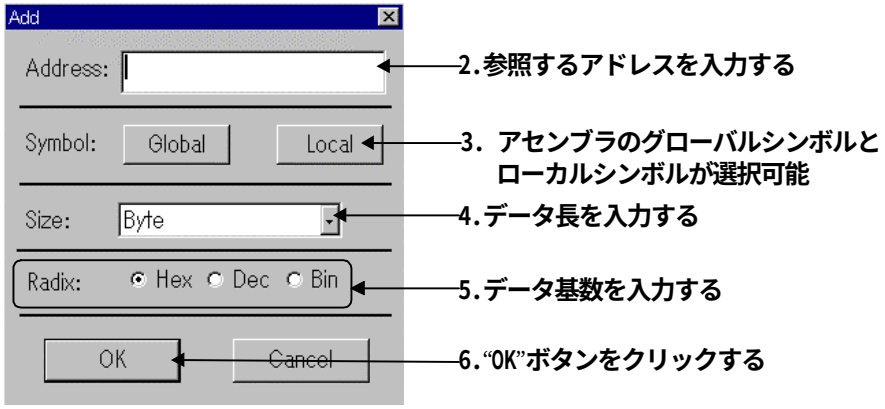
の選択でオープンします。指定したウォッチポイントが RAM モニタ領域内であれば、プログラム実行中の変化を ASM ウォッチウィンドウから監視することができます。

ASM ウォッチウィンドウについては、本マニュアル ウィンドウ機能編の項目「1.8 ASM ウォッチウィンドウ」をご参照下さい。

ウォッチポイントを登録するには・・・

ASM ウォッチウィンドウのメニューバーから Add ボタンをクリックしてください。Add ボタンをクリックすると Add ダイアログがオープンします。参照するアドレスを入力してください。また、プログラムウィンドウ上のポップアップメニュー（Add ASM Watch）でも登録できます。

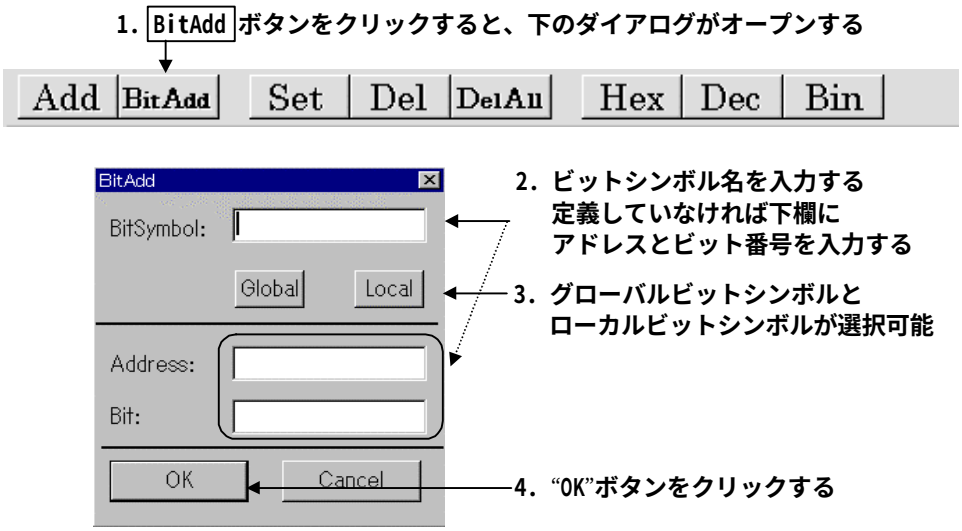
1. Add ボタンをクリックすると、下のダイアログがオープンする



登録したウォッチポイントは、ASM ウォッチウィンドウの現在のカーソル位置に追加されます。カーソル位置は、アドレス表示領域とデータ表示領域に、赤いマークで表示されます。また、カーソル位置は、どちらかの領域をクリックするか、または ↑ ↓ キーで移動できます。

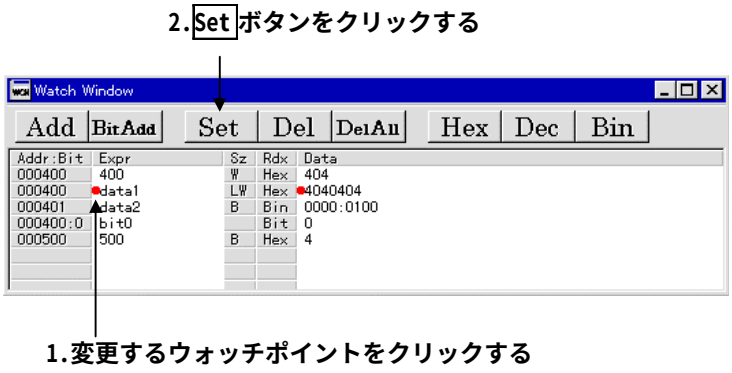
任意ビットをウォッチポイントとして登録するには・・・

ASM ウォッチウィンドウのメニューバーから BitAdd ボタンをクリックしてください。BitAdd ボタンをクリックすると BitAdd ダイアログがオープンします。参照するビットシンボル名、またはアドレスとビット番号を入力してください。また、プログラムウィンドウ上のポップアップメニュー（BitAdd ASM Watch）でも登録できます。



任意アドレスの値を変更するには・・・

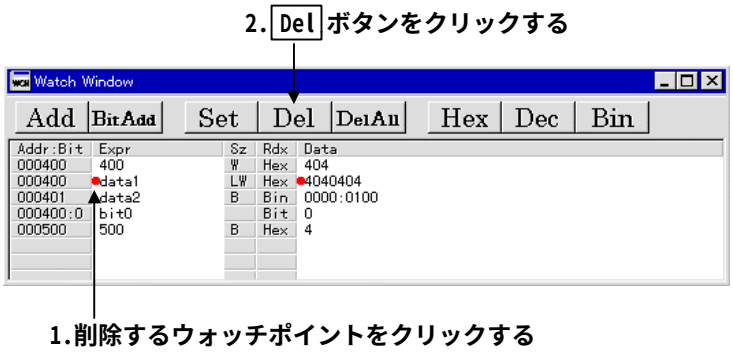
ASM ウォッチウィンドウから変更するウォッチポイントを選択し、ASM ウォッチウィンドウのツールバー Set ボタンをクリックしてください。



Set ダイアログがオープンしますので変更値を入力してください。

ウォッチポイントを削除するには・・・

ASM ウォッチウィンドウから削除するウォッチポイントを選択し、ASM ウォッチウィンドウのツールバー Del ボタンをクリックしてください。

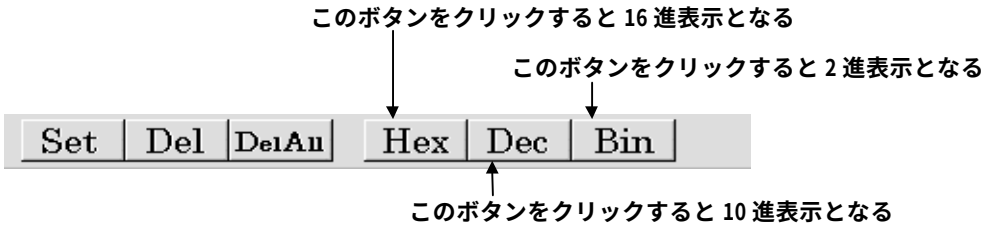


＊ 全ウォッチポイントを削除する場合は、DelAll ボタンをクリックする

または、削除するウォッチポイントをクリックして、Delete キーを入力することでも行えます。

メモリの参照形式を変更するには・・・

表示基数を変更することができます。ASM ウォッチウィンドウから変更するウォッチポイントを選択し、ASM ウォッチウィンドウのツールバー Hex、Dec、Bin ボタンのいずれかをクリックしてください。



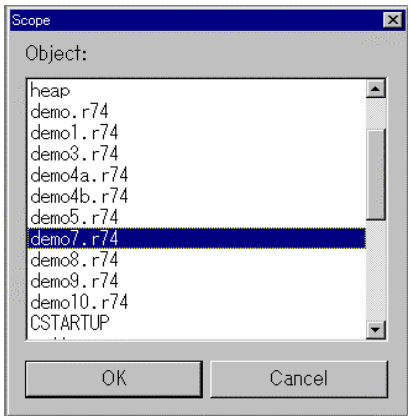
また、ASM ウォッチウィンドウの基数表示領域をダブルクリックすることでも、変更が行えます。

3.5 スコープを切り替えるには

スコープを切り替えるには、PD38SIMウィンドウのメニュー

［Debug］→［Scope...］

を選択すると、以下に示すダイアログがオープンします。同ダイアログに表示されたオブジェクトを選択するとスコープが切り替わります。



3.6 指定アドレスにデータを設定するには

指定アドレスへのデータ設定は、メモリウィンドウ、及びダンプウィンドウ上から設定できます。メモリウィンドウは、PD38SIMウィンドウのメニュー

［BasicWindow］→［Memory Window］

の選択でオープンします。ダンプウィンドウは、PD38SIMウィンドウのメニュー

［BasicWindow］→［Dump Window］

の選択でオープンします。

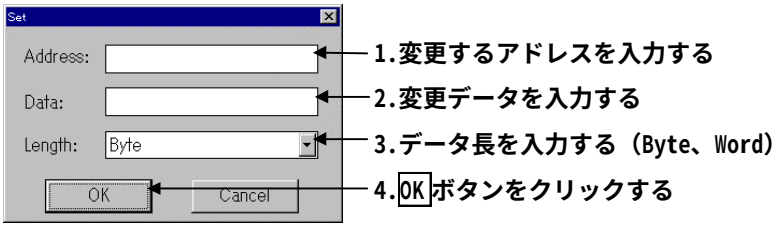
メモリウィンドウについては、本マニュアル ウィンドウ機能編の項目「1.5 メモリウィンドウ」、ダンプウィンドウについては、本マニュアル ウィンドウ機能編の項目「1.6 ダンプウィンドウ」をご参照下さい。

指定アドレスのデータを変更するには・・・

メモリウィンドウ、あるいはダンプウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［Debug］→［Set...］

を選択してください。選択すると Set ダイアログがオープンします。変更するアドレスとそのデータを入力してください。



また、Set ダイアログは、メモリウィンドウ、あるいはダンプウィンドウのデータ表示領域をダブルクリックすることでもオープンします。

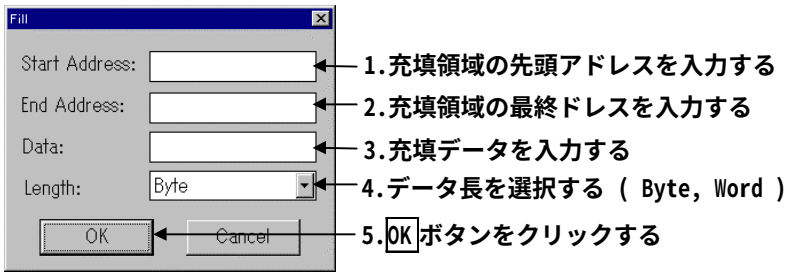
指定領域を一定データで充填するには・・・

メモリウィンドウ、あるいはダンプウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［Debug］→［Fill...］

を選択してください。選択すると Fill ダイアログがオープンします。充填するアドレス範囲と充填データを入力してください。

また、指定領域をマウスでドラッグ後に Fill ダイアログをオープンすると、先頭アドレス、最終アドレスに選択された領域が設定されます。



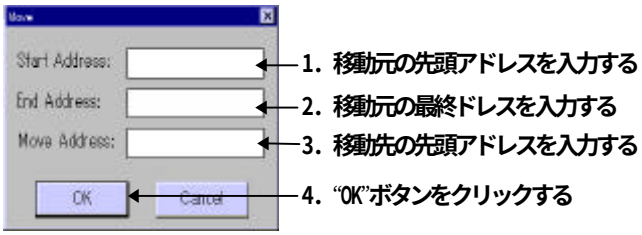
指定領域を他の領域へ移動するには・・・

メモリウィンドウ、あるいはダンプウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［Debug］→［Move...］

を選択してください。選択すると Move ダイアログがオープンします。移動するアドレス範囲と移動先のアドレスを入力してください。

また、指定領域をマウスでドラッグ後に Move ダイアログをオープンすると、先頭アドレス、最終アドレスに選択された領域が設定されます。



3.7 メモリ内容の表示を更新するには

メモリウィンドウ、及びダンプウィンドウでは、メモリ内容の変更を伴うコマンド（メモリの設定・充填・転送、実行停止、ステップ実行など）を実行した際、メモリ内容の表示を自動的に更新します。しかし、I/O 領域など、MCU の実行とは無関係に変更される領域を表示している際に、表示しているデータと実際のメモリ内容が異なることが起こる場合があります。

この場合、メモリウィンドウで最新のメモリ内容を表示するには、メモリウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［View］→［Refresh］

を選択してください（ツールバーの Refresh ボタンでも可能）。

また、ダンプウィンドウで最新のメモリ内容を表示するには、ダンプウィンドウがアクティブな状態では、PD38SIMウィンドウのメニュー

［Option］→［View］→［Refresh］

を選択してください（ツールバーの Refresh ボタンでも可能）。

3.8 メモリの取得モードを切換えるには

PD38SIMのダンプウィンドウ、メモリウィンドウでは表示の高速化のために512バイト単位でメモリを取得し内部バッファ（メモリキャッシュ）に貯えています。ウィンドウをスクロール、リサイズした場合でもこの512バイト内であればメモリは取得しません。

メモリキャッシュを使用しないようにするには、Cacheボタンを押します（またはメニューのCacheOnを選択します）。メモリキャシュを使用しない場合、表示している領域以外のメモリは取得されません。また、スクロール、リサイズ等で表示アドレスが変化した場合、メモリが取得されます。

4 ソフトウェアブレーク

ターゲットプログラムを指定行（アドレス）で停止するには、ソフトウェアブレークを使用します。ソフトウェアブレークポイントを設定するには、S/W ブレークポイント設定ダイアログを使用します。ソフトウェアブレークは、指定したソフトウェアブレークポイントの実行直前でターゲットプログラムを停止します。

- ソフトウェアブレークポイントは、64 点まで設定可能です。
- ソフトウェアブレークポイントを複数設定した場合、いずれか 1 点のソフトウェアブレークポイントに到達するとターゲットプログラムは停止します。

4.1 S/W ブレークポイント設定ダイアログをオープンするには

PD38SIM ウィンドウのツールバーから“SW”ボタンをクリックしてください。 S/W ブレークポイント設定ダイアログは、F7 キー入力でもオープンできます。

また、PD38SIM ウィンドウのメニュー

［Debug］→［Break Point］→［S/W Break Point...］

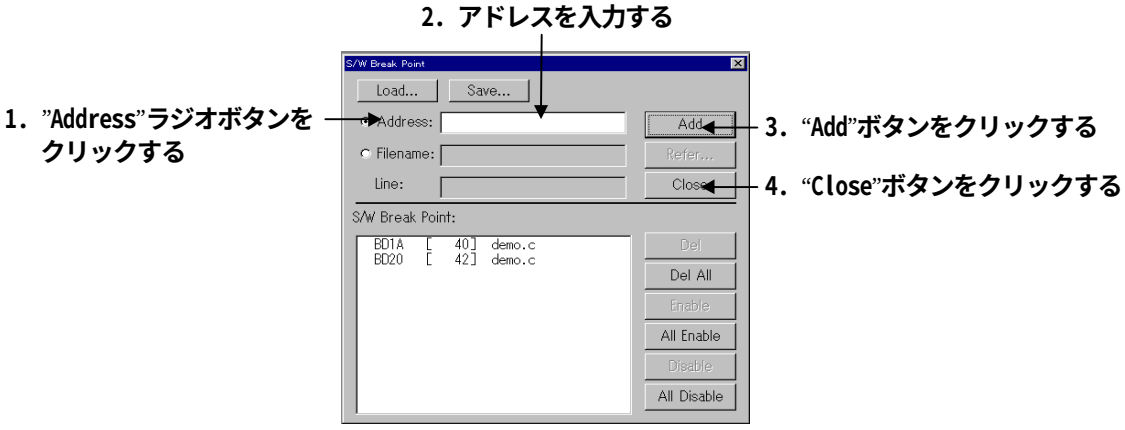
の選択でも S/W ブレークポイント設定ダイアログをオープンすることができます。

S/W ブレークポイント設定ダイアログの構成については、本マニュアル ウィンドウ機能編の項目「1.19 S/W ブレークポイント設定ダイアログ」をご参照下さい。

4.2 ブレークポイントを設定するには

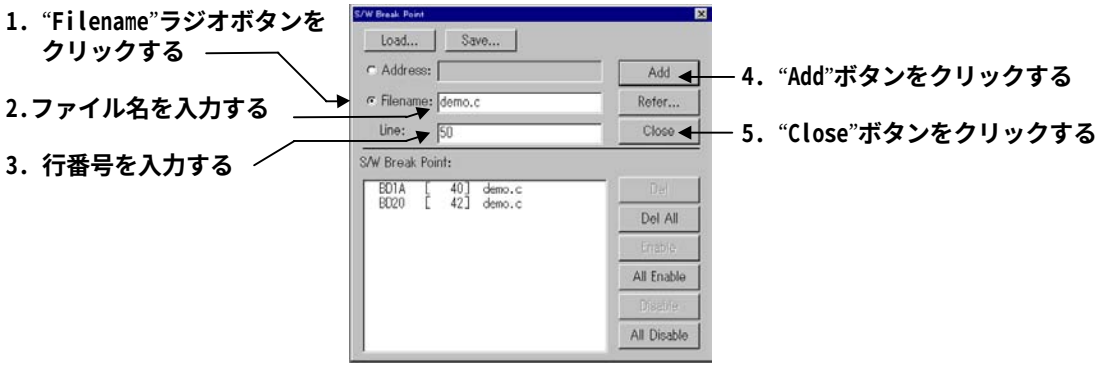
ブレークポイントをアドレスで指定するには・・・

S/W ブレークポイント設定ダイアログから以下の操作を行ってください。アドレス入力には、ラベルを指定することも可能です。



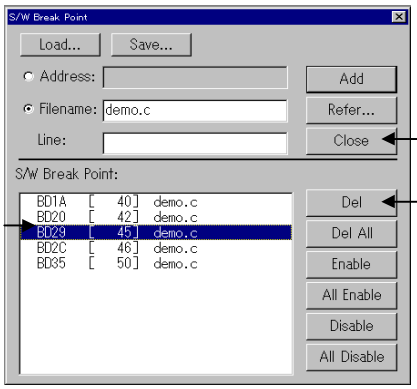
ブレークポイントを行番号で指定するには・・・

S/W ブレークポイント設定ダイアログから以下の操作を行ってください。



4.3 ブレークポイントを解除するには

S/W ブレークポイント設定ダイアログから以下の操作を行ってください。

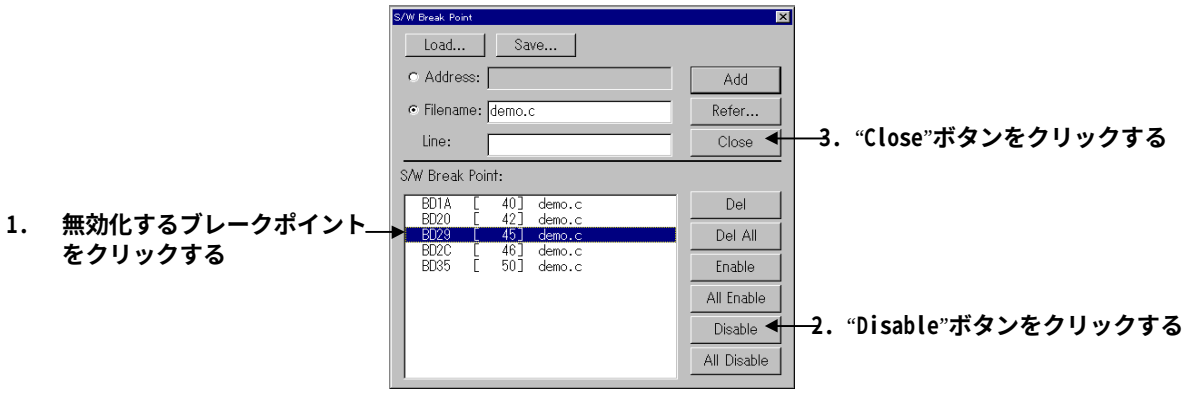


※全ブレークポイントを解除する場合は、、"Del All"ボタンをクリックする

または、解除するブレークポイントをクリックして、Delete キーを入力することでも行えます。

4.4 ブレークポイントを一時的に無効化するには

S/W ブレークポイント設定ダイアログから以下の操作を行います。

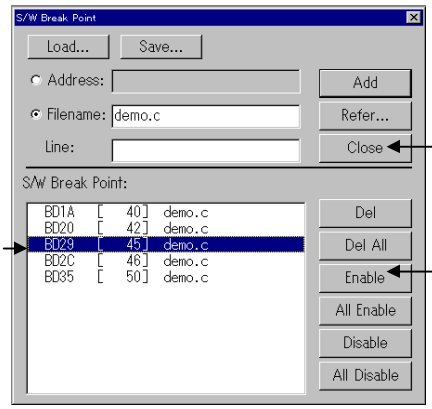


※全ブレークポイントを無効化する場合は、"All Disable"ボタンをクリックする

または、無効化するブレークポイントをダブルクリックすることでも行えます（"*"が表示されます）。

4.5 ブレークポイントを一時的に有効化するには

S/W ブレークポイント設定ダイアログから以下の操作を行ってください。



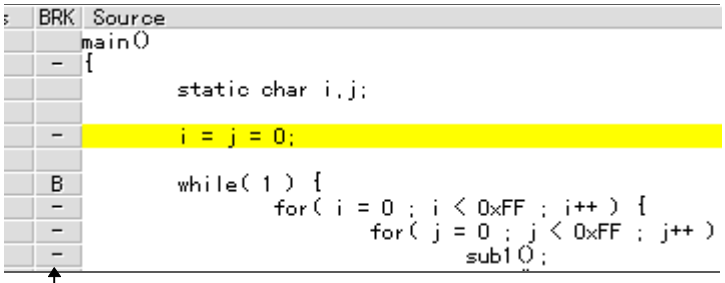
1. 有効化するブレークポイントをクリックする
2. “Enable”ボタンをクリックする
3. “Close”ボタンをクリックする

※全ブレークポイントを無効化する場合は、”All Enable”ボタンをクリックする

または、有効化するブレークポイントをダブルクリックすることでも行えます（“”＊”の表示が削除されます）。

4.6 ウィンドウ上からブレークポイントを設定するには

プログラム（ソース）ウィンドウからブレークポイントを設定することもできます。プログラム（ソース）ウィンドウから、ブレークさせたい行のブレークポイント表示領域（“－”の表示部分）をダブルクリックしてください。



ここをダブルクリックする

ソフトウェアブレークポイントとして設定された行は、ウィンドウのブレークポイント表示領域が“－”から“B”に変化します。“B”と表示された部分を再度ダブルクリックすると、ソフトウェアブレークポイントが解除されます。

4.7 ツールバーからブレークポイントを設定するには

PD38SIMウィンドウのツールバーからブレークポイントを設定することもできます。プログラム（ソース）ウィンドウから、ブレークさせたい行のプログラム表示領域（対応するブレークポイント表示領域に“—”の表示がある行）をクリックしてください。

ブレークポイントを設定するには、PD38SIMウィンドウのツールバーから”Break”ボタンをクリックしてください。

また、PD38SIMウィンドウのメニュー

［Debug］→［Break Point］→［Break］

の選択でもブレークポイントの設定が行えます。

ソフトウェアブレークポイントとして設定された行は、ウィンドウのブレークポイント表示領域が“—”から“B”に変化します。“B”と表示された行を再度クリックして”Break”ボタンをクリックすると、ソフトウェアブレークポイントが解除されます。

4.8 ブレークポイントを保存するには

S/W ブレークポイント設定ダイアログの Save ボタンをクリックするとファイル選択ダイアログがオープンしますので、保存するファイル名を指定してください。拡張子を省略した場合、拡張子”.brk”が付加されます。

4.9 ブレークポイントを読み込むには

保存したファイルからブレークポイントを読み込むには、S/W ブレークポイント設定ダイアログの Load ボタンをクリックするとファイル選択ダイアログがオープンしますので、読み込むファイルを指定してください。現在設定されているブレークポイントにファイルから読み込んだブレークポイントを追加します。ソフトウェアブレークポイントが 64 点を超える場合は、65 点目以降は無視します。

5 ハードウェアブレーク

メモリへのアクセスによりターゲットプログラムを停止させるには、ハードウェアブレークを使用します。ハードウェアブレークポイントを設定するには、H/W ブレークポイント設定ダイアログを使用します。ハードウェアブレークは、メモリのデータ書き込み／読み込み／命令フェッチを検出したときにブレークします。

- ハードウェアブレークポイントは、64 点まで設定可能です。
- ハードウェアブレークポイントを複数設定した場合、いずれか 1 点のハードウェアブレークポイントに到達するとターゲットプログラムは停止します。

5.1 H/W ブレークポイント設定ダイアログをオープンするには

PD38SIMウィンドウのツールバーから“HW”ボタンをクリックしてください。 H/W ブレークポイント設定ダイアログは、Shift + F7 キー入力でもオープンできます。

HW ボタンをクリックする

また、**PD38SIM**ウィンドウのメニュー

［Debug］→［Break Point］→［H/W Break Point...］

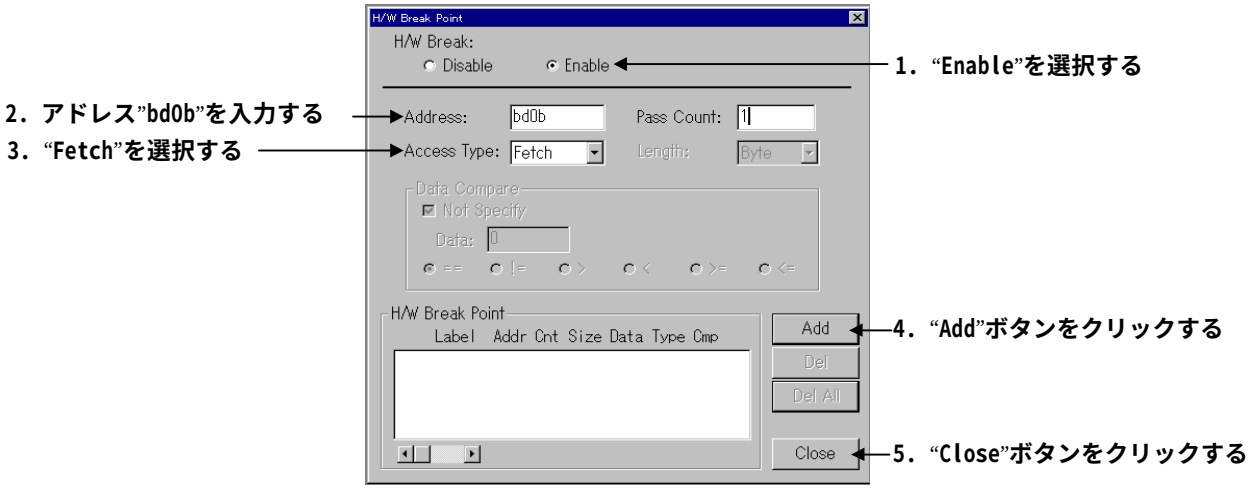
の選択でも H/W ブレークポイント設定ダイアログをオープンすることができます。

H/W ブレークポイント設定ダイアログの構成については、本マニュアル ウィンドウ機能編の項目「1.20 H/W ブレークポイント設定ダイアログ」をご参照下さい。

5.2 ハードウェアブレークポイントを設定するには

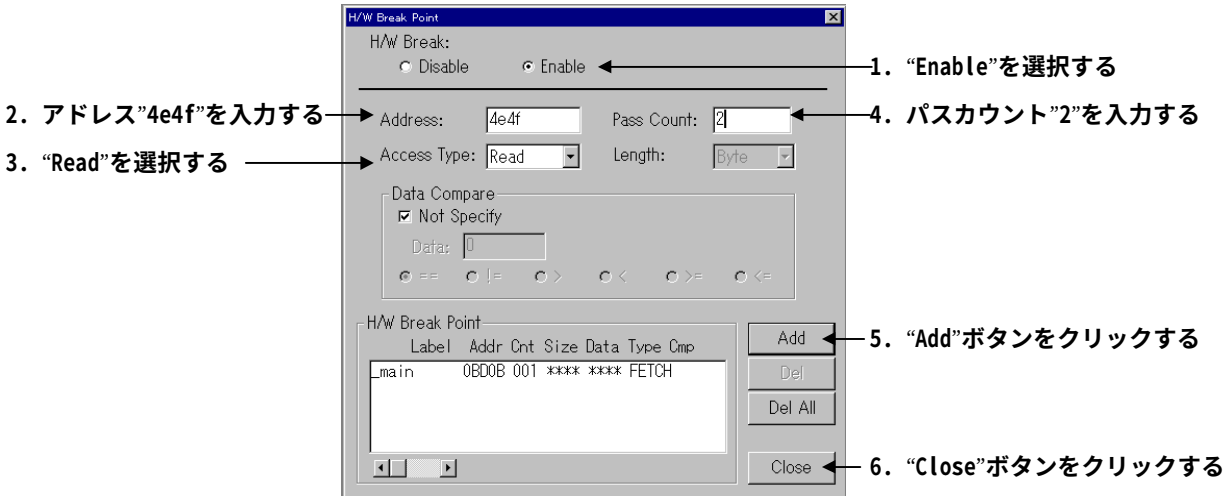
指定アドレスの命令を実行した際にブレークさせるには・・・

アドレス $BD0B_{16}$ 番地の命令を実行した際にブレークさせる設定を以下に示します。H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



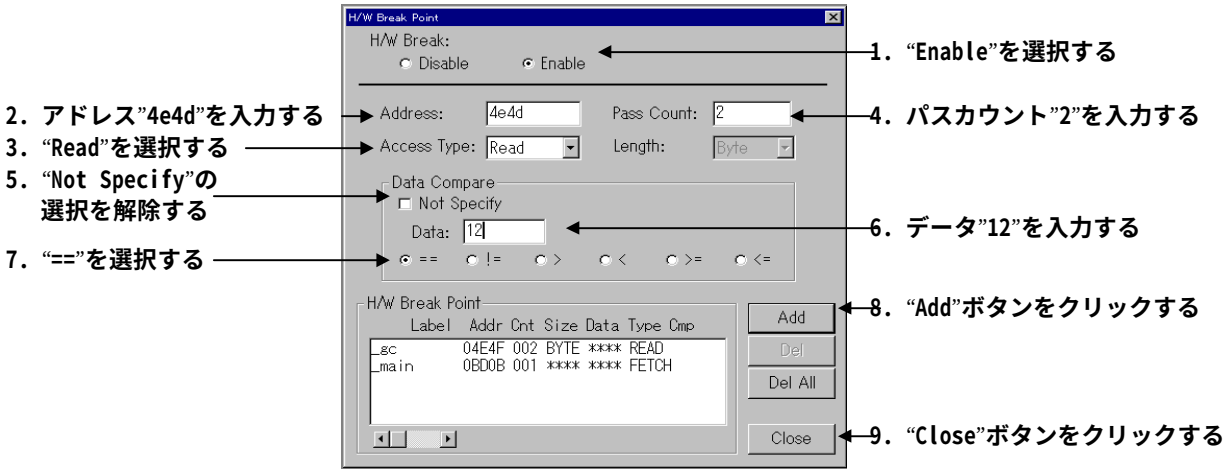
指定アドレスから読み込みを行った際にブレークさせるには・・・

アドレス $4E4F_{16}$ 番地からデータが 2 回読み込まれた際にブレークさせる設定を以下に示します。H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



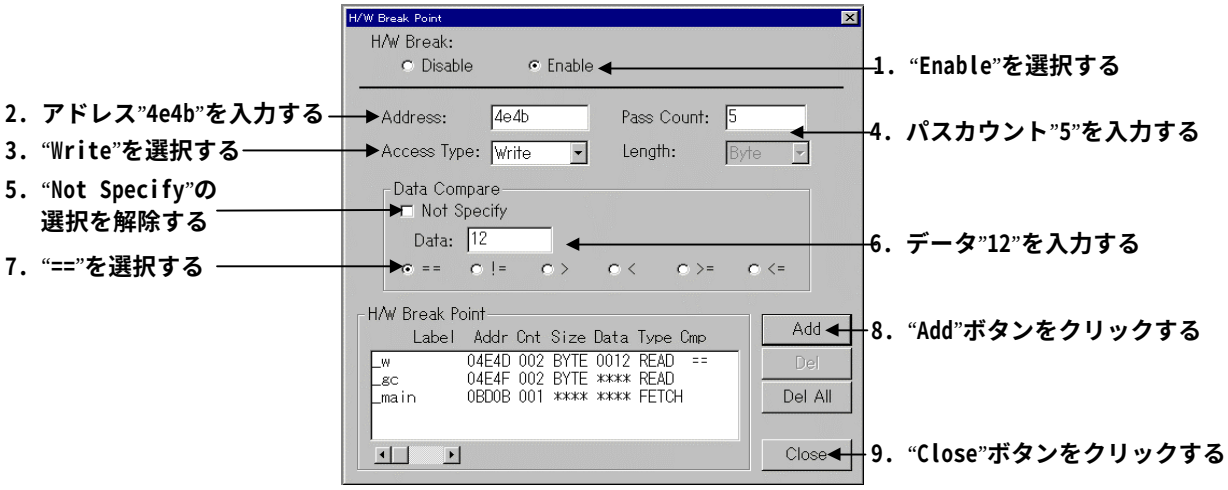
指定アドレスから指定データの読み込みを行った際にブレークさせるには・・・

アドレス $4E4D_{16}$ 番地からデータ 12_{16} が2回読み込まれた際にブレークさせる設定を以下に示します。H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



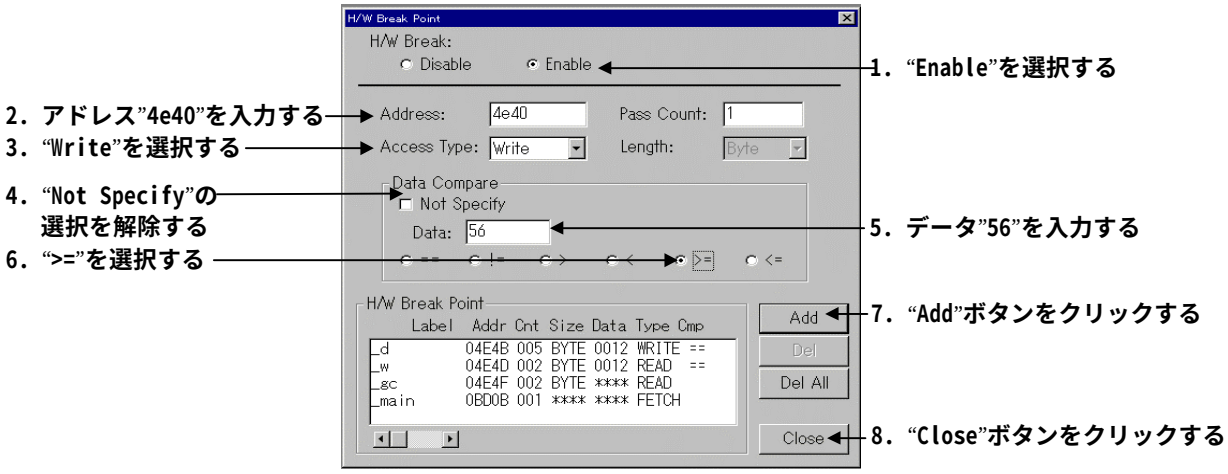
指定アドレスへ書き込みを行った際にブレークさせるには・・・

アドレス $4E4B_{16}$ 番地にデータ 12_{16} が5回書き込まれた際にブレークさせる設定を以下に示します。H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



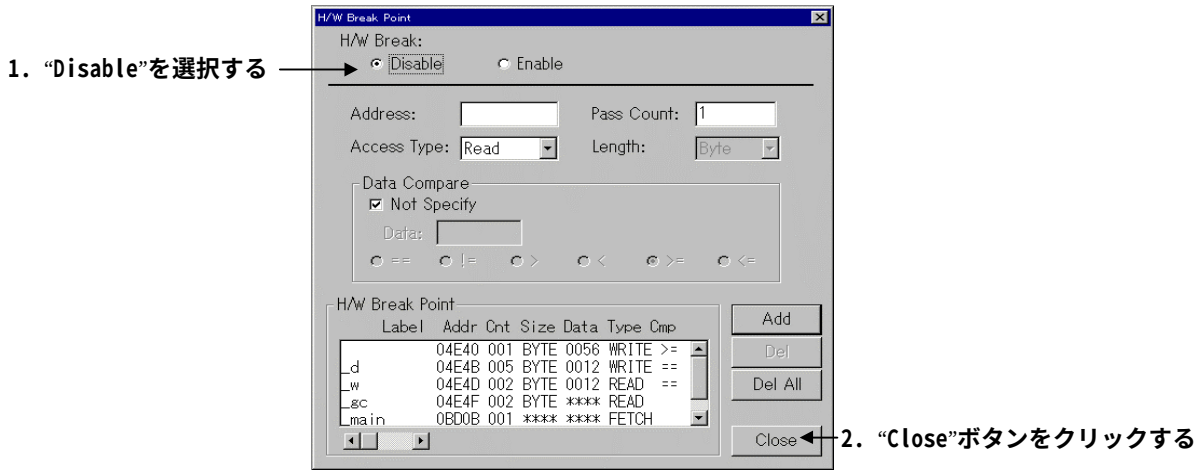
指定アドレスへ指定データ以上のデータの書き込みを行った際にブレークさせるには・・・

アドレス $4E40_{16}$ 番地にデータ 56_{16} 以上のデータが書き込まれた際にブレークさせる設定を以下に示します。H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



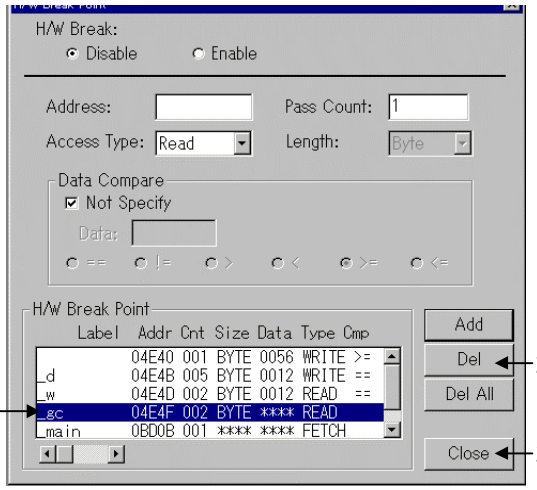
ハードウェアブレークを無効化するには・・・

ハードウェアブレークそのものを使用しない設定を以下に示します。H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



5.3 ハードウェアブレークポイントを解除するには

H/W ブレークポイント設定ダイアログから以下の操作を行ってください。



1．解除するブレークポイントをクリックする

2．“Del”ボタンをクリックする

3．“Close”ボタンをクリックする

※全ブレークポイントを解除するには、”Del All”ボタンをクリックする

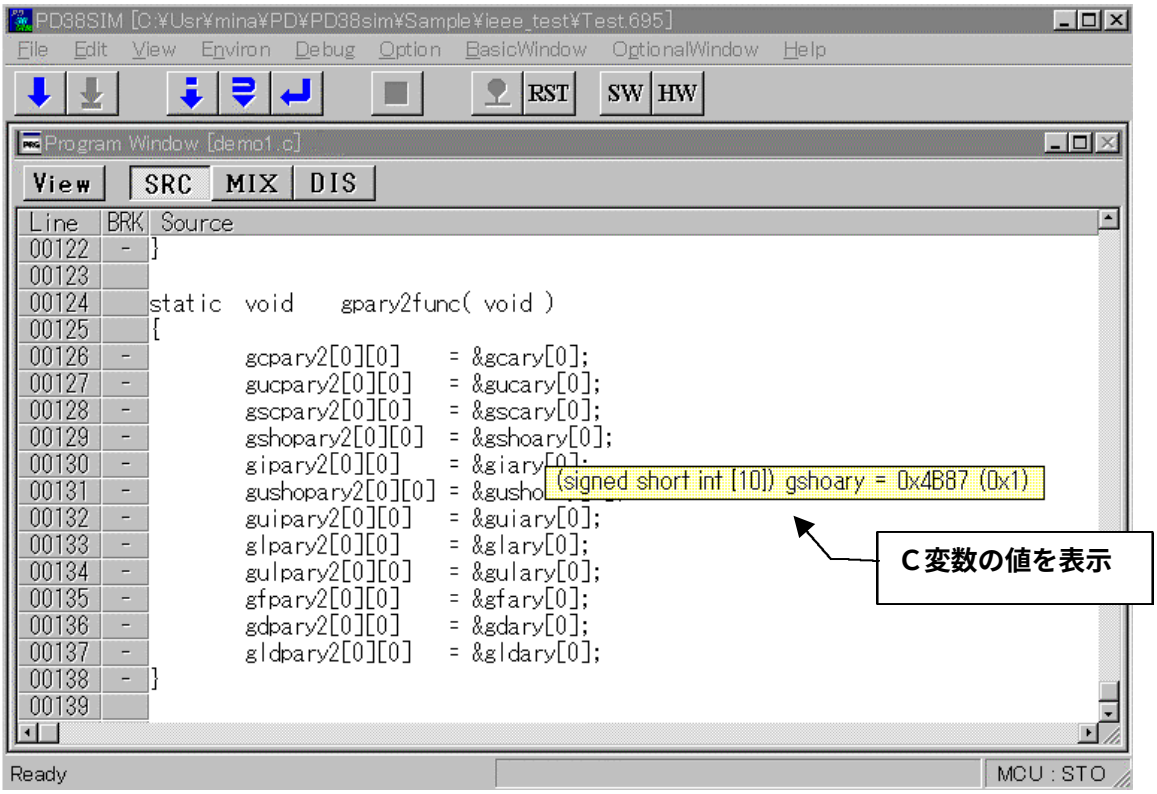
または、解除するブレークポイントをクリックして、Delete キーを入力することでも行えます。

6 C 変数の参照・変更

6.1 C 変数の値を参照するには

6.1.1 ソース／プログラムウィンドウでC 変数の値を参照するには

ソース／プログラムウィンドウのプログラム表示領域で C 変数上にマウスカーソルを一定時間（約 0.5 秒）静止させると、変数の値を表示します。



6.1.2 CWatch ウィンドウでC 変数の値を参照するには

ターゲットプログラム中で宣言されている C 言語の変数（以下 C 変数と記述）を参照するために、PD38SIMでは 4 つのウィンドウを用意しています。

- ローカルウィンドウ
- ファイルローカルウィンドウ
- グローバルウィンドウ
- C ウォッチウィンドウ

ローカルウィンドウは、関数内でのみ有効な変数の値を表示するウィンドウです。ローカルウィンドウは、PD38SIMウィンドウのメニュー

［Basic Window］→［C Watch Window］→［Local Window］

の選択でオープンします。

ファイルローカルウィンドウは、ファイル内でのみ有効な変数の値を表示するウィンドウです。ファイルローカルウィンドウは、PD38SIMウィンドウのメニュー

［Basic Window］→［C Watch Window］→［File Local Window］

の選択でオープンします。

グローバルウィンドウは、グローバルな変数の値を表示するウィンドウです。グローバルウィンドウは、PD38SIMウィンドウのメニュー

［Basic Window］→［C Watch Window］→［Global Window］

の選択でオープンします。

C ウォッチウィンドウは、任意変数の値を表示するウィンドウです。C ウォッチウィンドウは、PD38SIMウィンドウのメニュー

［Basic Window］→［C Watch Window］→［C Watch Window］

の選択でオープンします。

ローカルウィンドウ、ファイルローカルウィンドウ、グローバルウィンドウは、参照する C 変数を選択することができません。各ウィンドウは、現在のターゲットプログラムの実行位置に応じて、表示する変数が変化します。

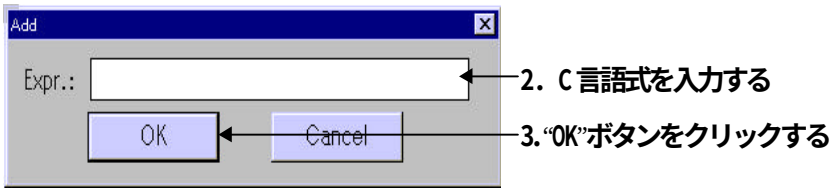
- ローカルウィンドウ
  現在実行されている関数に応じて、表示する変数が変化する。
- ファイルローカルウィンドウ
  現在実行されているソースファイルに応じて、表示する変数が変化する。
- グローバルウィンドウ
  実行位置に関係なく、C 言語のグローバル変数の一覧を表示する。

なお、任意の C 変数を参照する場合は、C ウォッチウィンドウを使用してください。

Cウォッチポイントを登録するには・・・

（方法1　C ウォッチウィンドウから登録）

C ウォッチウィンドウのツールバーから“Add”ボタンをクリックしてください。“Add”ボタンをクリックすると Add ダイアログがオープンします。Add ダイアログから C 言語式（C 言語の変数、式 等）を指定してください。ポインタとして C 言語式を登録する場合は、ツールバーから“Add*”ボタンをクリックしてください。これにより、C 言語式が C ウォッチポイントとして登録されます。また、プログラムウィンドウ上のポップアップメニュー（Add C Watch／Add C Watch Pointer）でも登録できます。

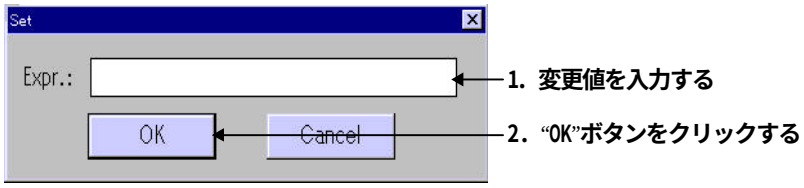


（方法2　ローカル／ファイルローカル／グローバルウィンドウから登録）

ローカルウィンドウ、ファイルローカルウィンドウ、グローバルウィンドウに表示されている変数を C ウォッチポイントとして登録することができます。登録したい変数をクリックしてください。その後、そのウィンドウのツールバーから“Cwatch”ボタンをクリックしてください。ポインタとして登録する場合は、“Cwatch*”ボタンをクリックしてください。

6.2 C 変数の値を変更するには

ローカルウィンドウ、ファイルローカルウィンドウ、グローバルウィンドウ、C ウォッチウィンドウのいずれからも C 変数の値を変更することが可能です。変更したい C 変数をクリックし、ウィンドウのツールバーから“Set”ボタンをクリックしてください。“Set”ボタンをクリックすると Set ダイアログがオープンします。変更値を入力してください。



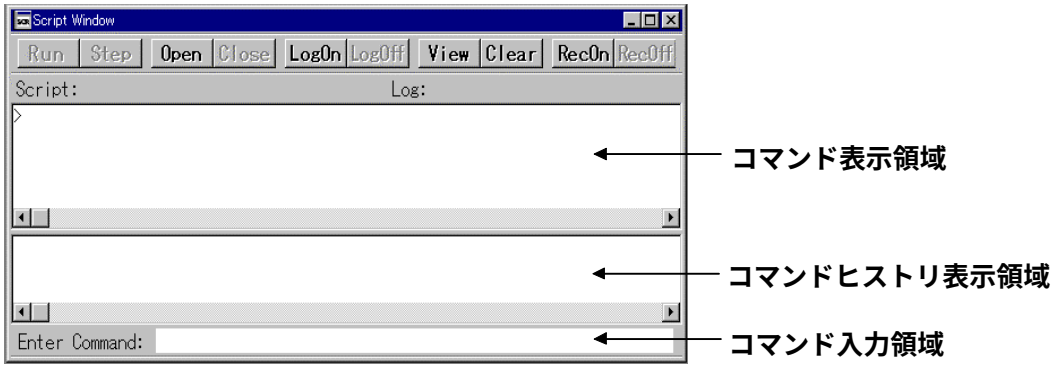
7 スクリプトコマンド

7.1 スクリプトコマンドを実行するには

スクリプトコマンドは、スクリプトウィンドウから実行します。スクリプトウィンドウは、PD38SIM ウィンドウのメニュー

［Basic Window］→［Script Window］

の選択でオープンします。スクリプトコマンドは、スクリプトウィンドウのコマンド入力領域から入力します。コマンド入力領域をクリックし、フォーカスを与えてスクリプトコマンドを入力してください。コマンド実行結果は、コマンド表示領域に出力されます。



以前実行したスクリプトコマンドを再実行するには・・・

コマンドヒストリ表示領域にこれまでの実行履歴が表示されています。再実行するコマンドをダブルクリックしてください。

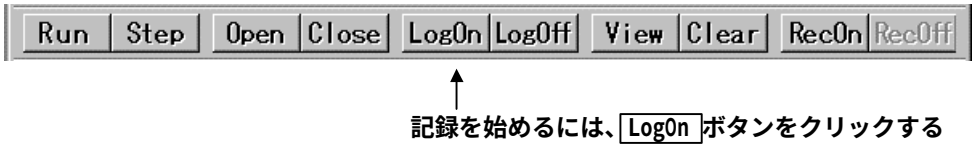
スクリプトコマンドを途中終了するには・・・

スクリプトコマンド実行中のみ表示されるダイアログの STOP ボタンをクリックしてください。

7.2 スクリプトコマンドの実行結果を記録するには

スクリプトコマンドの実行結果を記録するには、ロギング機能を使用します。スクリプトコマンドを実行する前に、スクリプトウィンドウのツールバーから“LogOn”ボタンをクリックしてください。

“LogOn”ボタンをクリックするとファイルセレクションダイアログがオープンしますので、記録するファイル名を指定してください。ファイルの拡張子を省略した場合は、“.log”となります。

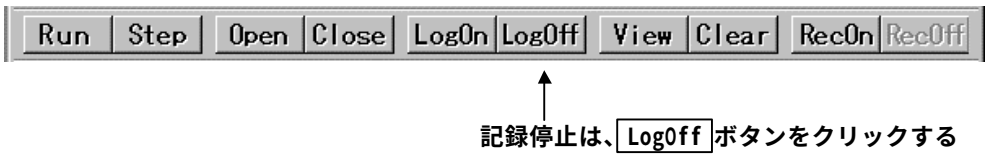


PD38SIMウィンドウのメニュー
［Option］→［Log］→［On...］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

- PD38SIMの起動後に一度オープン／クローズして保存したログファイルを再びオープンした場合は、既存のログファイルの最後に新たな内容を追加します。ただし、今回のPD38SIM起動以前に既に作成されていたログファイルを再びオープンした場合は、上書きになります。
- ファイルセレクションダイアログのファイルリストには、拡張子が”.log”のファイル名の一覧を優先して表示します。しかし、ファイル名を入力する領域に直接、フルネームでファイル名を入力することにより、ファイル属性”.log”以外のファイルをログファイルとしてオープンすることも可能です。
- ログファイルは、最大、8段までネストしてオープンできます。

実行結果の記録を終了するには・・・

スクリプトウィンドウのツールバーから“LogOff”ボタンをクリックしてください。



PD38SIMウィンドウのメニュー
［Option］→［Log］→［Off］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

- ログファイルがネストしている場合、現在のログファイルへの出力は、終了しますが、1つ前のログファイルへの出力を再開します。

実行結果をコマンド実行後に記録するには・・・

PD38SIMは、最新の 1000 行分のコマンド実行結果を保存するビューバッファを持っています。ビューバッファの内容を保存するには、スクリプトウィンドウのツールバーから“View”ボタンをクリックしてください。

“View”ボタンをクリックするとファイルセレクションダイアログがオープンしますので、記録するファイル名を指定してください。ファイルの拡張子を省略した場合は、“.viw”となります。

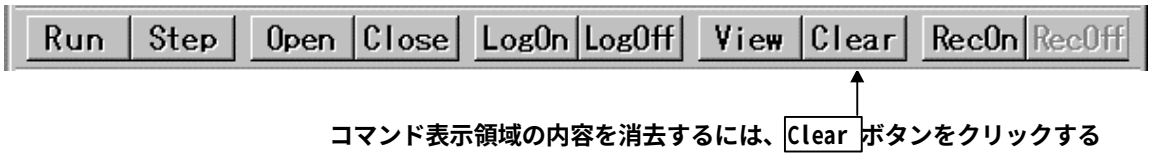
PD38SIMウィンドウのメニュー
［Option］→［View］→［Save...］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

- ファイル名に既存のファイル名を指定した場合は、既存ファイルの最後にビューバッファ内容を追加します。
- ファイルセレクションダイアログのファイルリストには、拡張子が”.viw”のファイル名の一覧を優先して表示します。しかし、ファイル名を入力する領域に直接、フルネームでファイル名を入力することにより、ファイル属性”.viw”以外のファイルをビューファイルとしてオープンすることも可能です。

実行結果の表示を消去するには・・・

コマンド表示領域の内容を消去するには、スクリプトウィンドウのメニューから“Clear”ボタンをクリックしてください。

“Clear”ボタンをクリックするとコマンド表示領域の内容が消去されます。また、ビューバッファの内容も消去されます。



PD38SIMウィンドウのメニュー
［Option］→［View］→［Clear］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

実行したコマンドを保存するには・・・

PD38SIMは、実行したコマンドをファイルに記録する機能を持っています。この機能はlogと異なり、コマンドのみを記録するため保存したファイルをスクリプトファイルとして使用することができます。実行したコマンドを保存するには、スクリプトウィンドウのツールバーから“RecOn”ボタンをクリックしてください。

“RecOn”ボタンをクリックするとファイルセレクションダイアログがオープンしますので、記録するファイル名を指定してください。ファイルの拡張子を省略した場合は、“.scr”となります。

実行コマンドを保存するには、RecOn ボタンをクリックする

PD38SIMウィンドウのメニュー
［Option］→［Record］→［On...］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

- ファイルセレクションダイアログのファイルリストには、拡張子が”.scr”のファイル名の一覧を優先して表示します。しかし、ファイル名を入力する領域に直接、フルネームでファイル名を入力することにより、ファイル属性”.scr”以外のファイルを保存ファイルとしてオープンすることも可能です。

実行したコマンドの保存を中止するには・・・

実行したコマンドの保存を中止するには、スクリプトウィンドウのメニューから“RecOff”ボタンをクリックしてください。

“RecOff”ボタンをクリックするとコマンド保存ファイルがクローズされます。

実行コマンドの保存を中止するには、RecOff ボタンをクリックする

PD38SIMウィンドウのメニュー
［Option］→［Record］→［Off］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

7.3 スクリプトコマンドを一括して実行するには

スクリプトコマンドは、一括して実行することもできます。あらかじめ、実行するコマンドをお手持ちのエディタでファイルに記述してください。ファイルの拡張子は、".scr"としてください。

次にスクリプトウィンドウからスクリプトファイルを読み込みます。スクリプトウィンドウのツールバーから"Open"ボタンをクリックしてください。

"Open"ボタンをクリックするとファイルセレクションダイアログがオープンしますので、実行するスクリプトファイルを選択してください。

スクリプトファイルをオープンするには、Open ボタンをクリックする

PD38SIMウィンドウのメニュー
［Option］→［Script］→［Open］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

- ファイルセレクションダイアログのファイルリストには、拡張子が".scr"のファイル名の一覧を優先して表示します。しかし、ファイル名を入力する領域に直接、フルネームでファイル名を入力することにより、ファイル属性".scr"以外のファイルをスクリプトファイルとしてオープンすることも可能です。
- スクリプトファイルは、最大、5 段までネストしてオープンできます。

スクリプトファイルを読み込むとスクリプトウィンドウのコマンドヒストリ表示領域がスクリプトファイル表示領域に変化します。

スクリプトファイルの内容を一括実行するには、スクリプトウィンドウのツールバーから"Run"ボタンをクリックしてください。

"Run"ボタンをクリックすると一括実行を始めます。スクリプトファイルは、記述されたすべてのコマンド実行後にクローズします。

スクリプトファイルを一括実行するには、Run ボタンをクリックする

PD38SIMウィンドウのメニュー
［Option］→［Script］→［Run］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

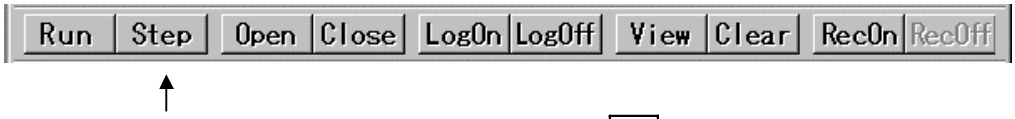
スクリプトファイル実行を中断するには・・・

スクリプトファイル実行中に表示されるダイアログの STOP ボタンをクリックしてください。

スクリプトファイルの実行は、その次の行が実行される手前で停止します。

スクリプトファイルのコマンドを一コマンドづつ実行するには・・・

スクリプトウィンドウのツールバーから“Step”ボタンをクリックしてください（スクリプトのステップ実行）。“Step”ボタンをクリックする度に一コマンドづつ実行します。



PD38SIMウィンドウのメニュー
［Option］→［Script］→［Step］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

スクリプトファイルをクローズするには・・・

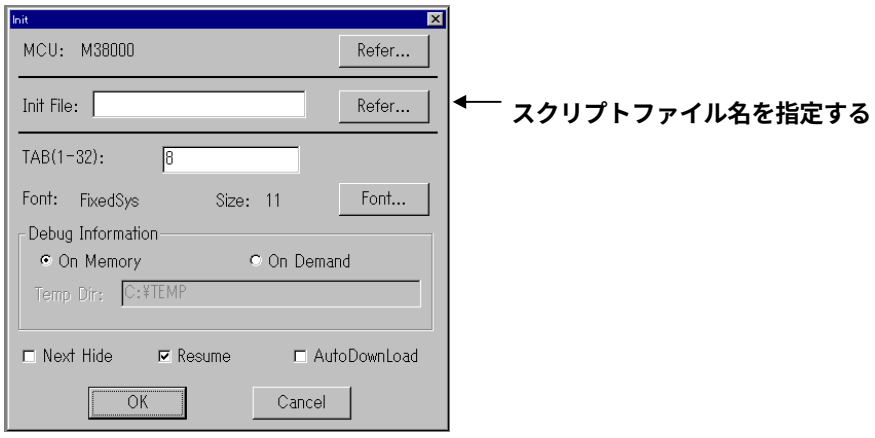
スクリプトウィンドウのツールバーから“Close”ボタンをクリックしてください。

PD38SIMウィンドウのメニュー
［Option］→［Script］→［Close］
の指定でも同じ動作を行います（スクリプトウィンドウがアクティブ時）。

- スクリプトファイルがネストしている場合、現在のスクリプトファイルはクローズしますが、1 つ前のスクリプトファイルがオープンします。

スクリプトコマンドをPD38SIM起動時に実行するには・・・

起動時に表示する Init ダイアログにおいて起動時に実行するスクリプトファイル名を指定してください。



8 PD38SIMの終了

8.1 PD38SIMを終了するには

PD38SIMウィンドウのメニュー
［File］→［Exit］
を選択してください。終了確認のためのダイアログがオープンしますのでOKボタンをクリックしてください。

“OK”ボタンをクリックする

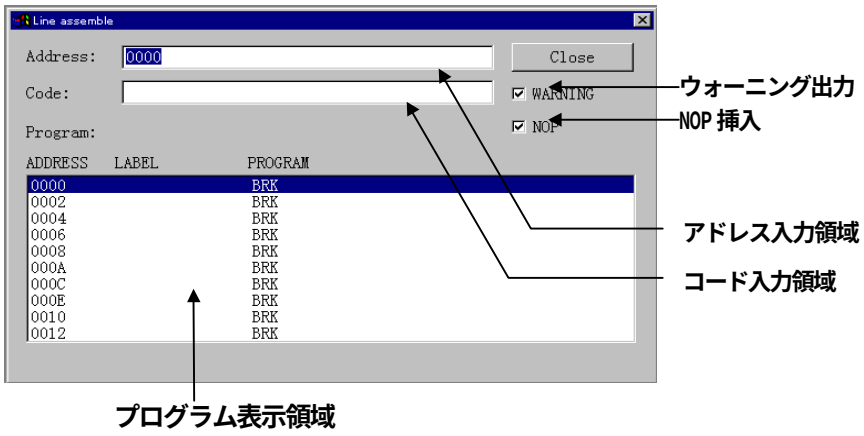
9 その他

9.1 ラインアセンブルするには

ラインアセンブルするには、ラインアセンブルダイアログで行う方法とスクリプトコマンドで行う 2 種類の方法があります。

9.1.1 ラインアセンブルダイアログで行う方法

プログラム（ソース）ウィンドウでラインアセンブルを行いたい位置をクリックし、メニュー［Option］→［Line Assemble］を選択すると以下に示すラインアセンブルダイアログがオープンします（位置を指定しない場合は、ダイアログをオープンした後にアドレスを入力してください）。



- アドレス入力領域、コード入力領域にラインアセンブルを行うアドレスとアセンブル命令を入力し、リターンキーを押します。本ダイアログのプログラム表示領域で、反転表示している行がラインアセンブルの対象となるアドレスです。
  - 入力した命令バイト数が変更前の命令バイト数より少ない場合は、NOP 命令を挿入してバイト数を合わせます。
  - 入力した命令バイト数が変更前の命令バイト数より多い場合、ウォーニングダイアログを表示します。OK ボタンを押すと、入力した命令を書き込みます。Cancel ボタンを押すと、書き込みを中止します
- NOP 挿入を OFF にすると、入力した命令バイト数が変更前の命令バイト数より短い場合でも NOP 命令を挿入しません。
- ウォーニング出力を OFF にすると、ウォーニングダイアログを表示せずに、強制的に入力した命令を書き込みます。
- 本ダイアログのプログラム表示領域の任意の行をクリックすることにより、クリックした行をラインアセンブルの対象にすることができます。

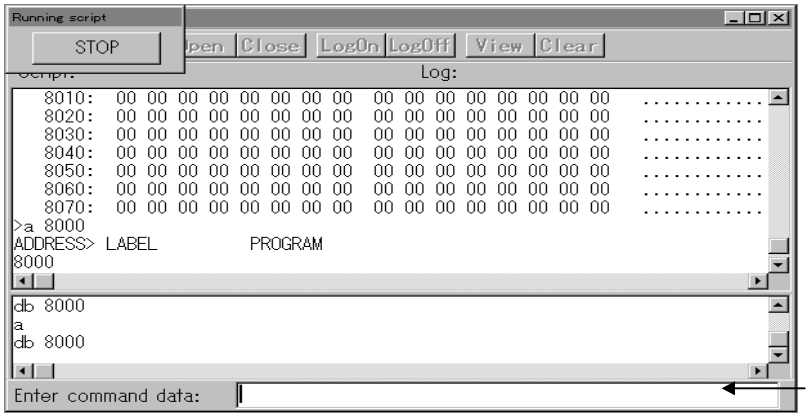
9.1.2 Assemble コマンドで行う方法

ラインアセンブルするには、スクリプトウィンドウからアセンブルコマンドを実行します。コマンド名は、Assemble（短縮名 A）です。コマンドの後にラインアセンブルするアドレスを指定してください。

>Assemble　アセンブルアドレス

Assemble コマンドを実行するとスクリプトウィンドウのコマンド入力領域がニーモニック入力領域となります。

スクリプトコマンドの使用方法については、本マニュアル 基本操作方法編の項目「7.1 スクリプトコマンドを実行するには」をご参照下さい。ラインアセンブルするニーモニックを入力してください。以下にスクリプトウィンドウのニーモニック入力待ち状態を示します。



ニーモニック入力領域

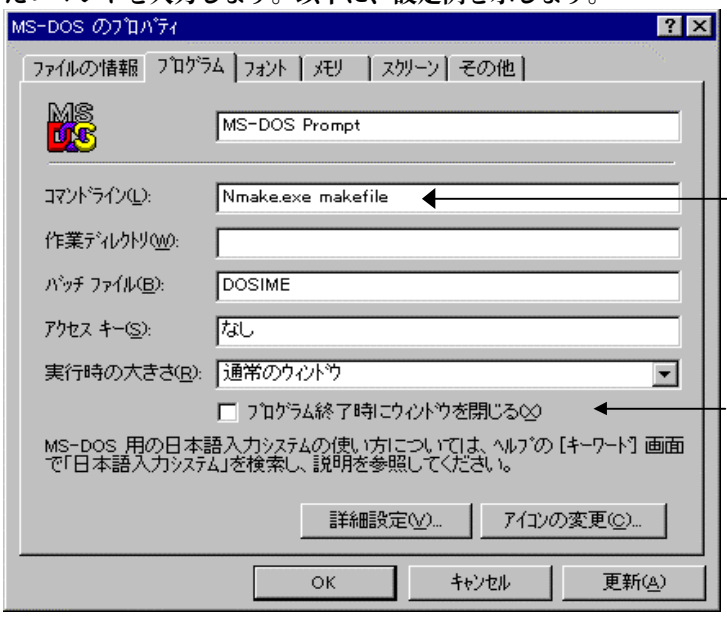
ラインアセンブルを終了するには・・・

ニーモニック入力領域に“Enter”のみを入力してください。スクリプトウィンドウがスクリプトコマンド入力待ち状態となります。

9.2 Make を起動するには

従来、DOS 窓からコマンドを入力してターゲットプログラムを Make していた作業をPD38SIMから行うことができます。
PD38SIMで Make を行う手順を以下に示します。

1 pif ファイルを作成します。
以下の手順で pif ファイルを作成してください。
(a) windows のディレクトリにある command.com のショートカットキーを作成します。
(b) 作成したショートカットキーにファイル名を付けてください（なお、**拡張子は付けないで下さい**）。その後、メイクファイルのあるディレクトリにコピーします。これが pif ファイルとなります。
(c) このファイルのプロパティダイアログをオープンし、コマンドラインに DOS 窓で入力していたコマンドを入力します。以下に、設定例を示します。

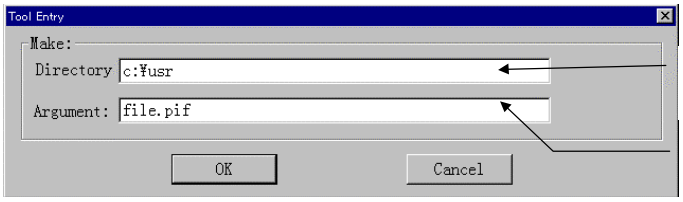


コマンドラインを入力（実行ファイル名には、拡張子を指定して下さい）

チェックを解除

プロパティダイアログは、エクスプローラなどで pif ファイルを選択してマウスの右ボタンをクリックした際に表示されるメニューの［プロパティ］を選択することによりオープンします。

2 pif ファイルをPD38SIMに登録します。
メニュー［Debug］→［Entry］を選択し、以下のダイアログをオープンし、pif ファイルを登録します。



pif ファイルをコピーしたディレクトリを入力

pif ファイル名を入力（拡張子に.pif を指定して下さい）。

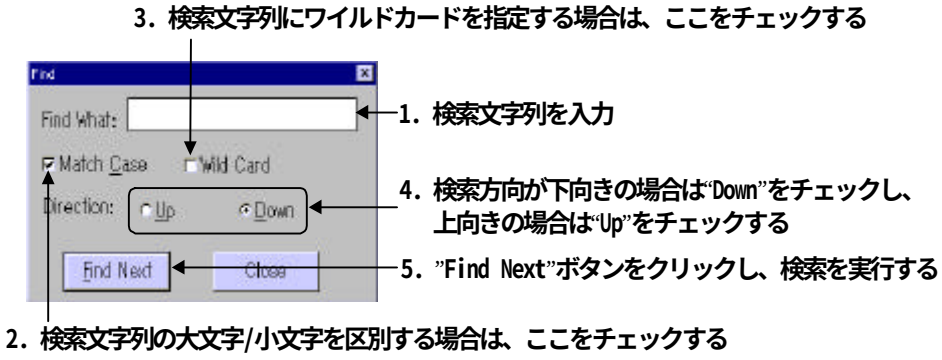
3 Make を起動します。
メニュー［Debug］→［Make］を選択すると、pif ファイルで指定した内容を実行します。

9.3 ターゲットプログラムの文字列を検索するには

プログラムウィンドウあるいはソースウィンドウがアクティブな状態の時、ターゲットプログラム内の文字列を検索することができます。対象とするプログラム（ソース）ウィンドウをアクティブにし、PD38SIMウィンドウのメニュー

［Edit］→［Find...］

を選択してください。選択すると以下の Find ダイアログがオープンします。検索文字列を入力して検索を開始してください。

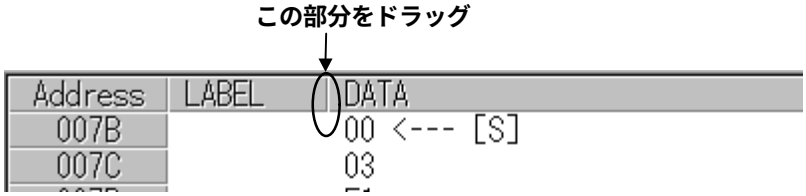


9.4 ウィンドウの表示領域の割合を変更するには

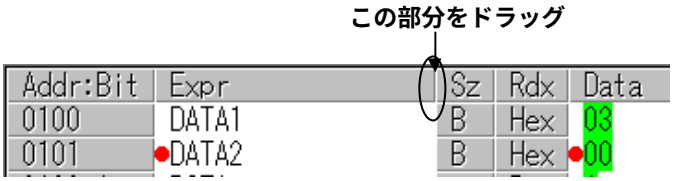
プログラムウィンドウ、ソースウィンドウ、メモリウィンドウ、及び ASM ウォッチウィンドウでは、各項目の表示領域の広さの割合をマウスで調節することができます。以下に各ウィンドウごとに、その方法を示します。

- プログラムウィンドウ、及びソースウィンドウ
  逆アセンブル表示モードのときに、オブジェクトコード表示領域（〈Objcode〉）とプログラム表示領域の 2 つの領域（〈Label〉と〈Mnemonic〉）の広さの割合が調節できます。

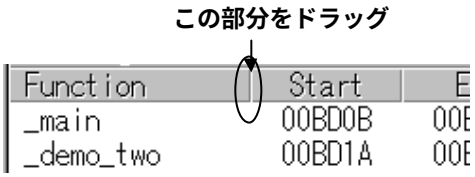
- メモリウィンドウ
  ラベル表示領域（〈LABEL〉）とメモリ内容表示領域の（〈DATA〉）の広さの割合が調節できます。



- ASM ウォッチウィンドウ
  式表示領域（〈Expr〉）の広さの割合が調節できます。



- カバレッジウィンドウ
  関数名表示領域（〈Function〉）の広さの割合が調節できます。



9.5 アクティブウィンドウを切換えるには

PD38SIMでは、操作対象となるウィンドウがアクティブになっている必要があります。マウスでウィンドウをクリックしてアクティブウィンドウを切換える他に、キーボード操作でも切換えることができます。

キーボードからアクティブウィンドウを切換えるには以下のように入力します。

［Ctrl］＋［TAB］

Shift キーを押しながら上記操作を行うと、逆順にアクティブウィンドウが切換わります。

9.6 PD38SIMのバージョンを表示するには

PD38SIM ウィンドウのメニュー

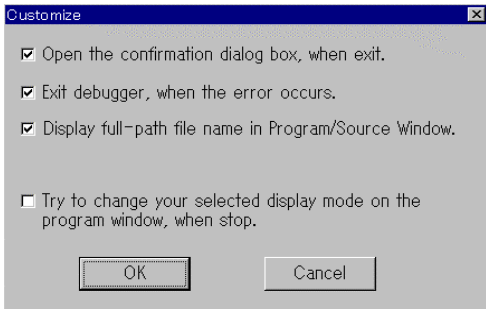
［Help］→［About...］

を選択してください。About ダイアログがオープンします。About ダイアログでは、PD38SIM のバージョンを表示しています。

OK ボタンを押すと About ダイアログは、クローズします。

9.7 PD38SIMの動作をカスタマイズするには

PD38SIMの動作をカスタマイズするには、**PD38SIM**ウィンドウのメニュー
［Environ］→［Customize...］
を選択すると、以下に示すカスタマイズダイアログがオープンします。カスタマイズダイアログで
PD38SIMの動作を設定できます。

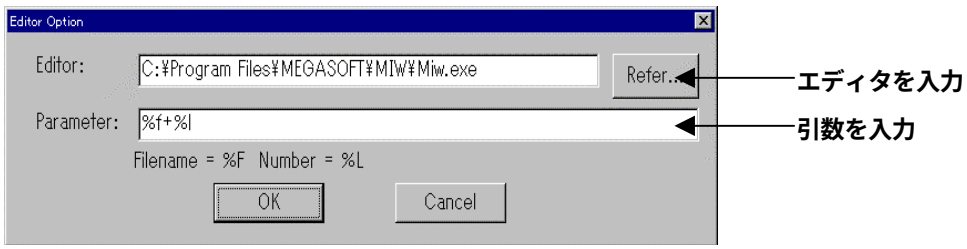


・Open the confirmation dialog box, when exit.
PD38SIM終了時に終了確認ダイアログを表示します。
・Exit debugger, when the error occurs.
エラーが生じた場合、デバッガを終了します。
・Display full-path name in Program/Source Window.
Program/Source Window **にパス付でファイル名を表示します。**
・Try to cange your selected display mode on the program window, when stop.
ターゲットプログラム停止時に、プログラムウィンドウの表示モードをユーザーが指定した表示モードに更新します(停止位置によっては MIX **表示や逆アセンブル表示モードになる場合があります)。**

9.8 エディタをオープンするには

エディタをオープンするための手順を以下に示します。

1．エディタを登録します。
プログラムウィンドウのポップアップメニュー[Entry Editor]**を選択し、以下のダイアログでエディタ及び引数を登録します。エディタ起動時のオプションに、ファイル名や行番号を指定可能な場合、引数を指定してください。ファイル名の項目には、**%F**、行番号の項目には**%L**を指定してください。**
起動時のオプションの指定方法はご使用になるエディタのマニュアルを参照ください。



2．エディタをオープンします。
プログラムウィンドウのポップアップメニュー[Open Editor]**を選択します。**

より高度なデバッグ編

このページは白紙です。

1 I/O ウィンドウでの仮想ポート入力の設定

1.1 概要

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

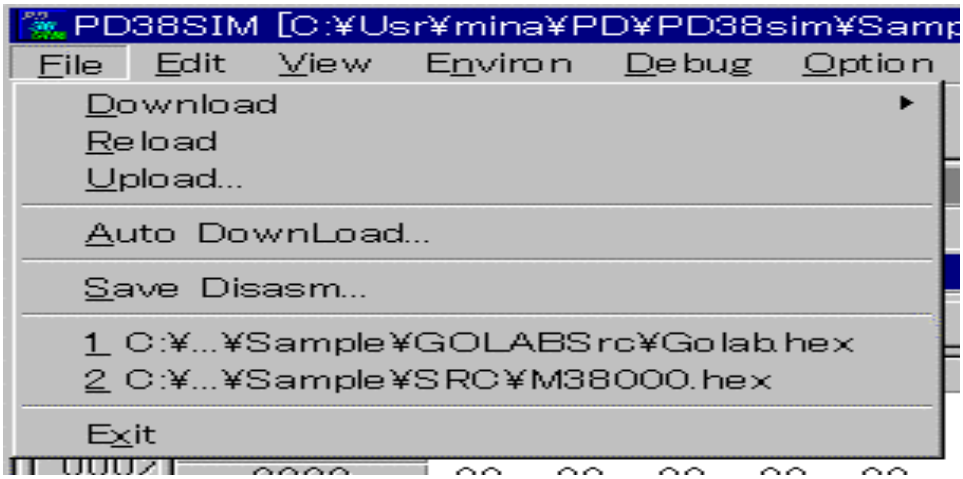
1. あるメモリに対して時間の変化とともにデータを入力したい場合
   プログラムの実行が指定サイクルになった時に、データを入力することができます。
   この場合、サイクル同期入力の設定を行って下さい。

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

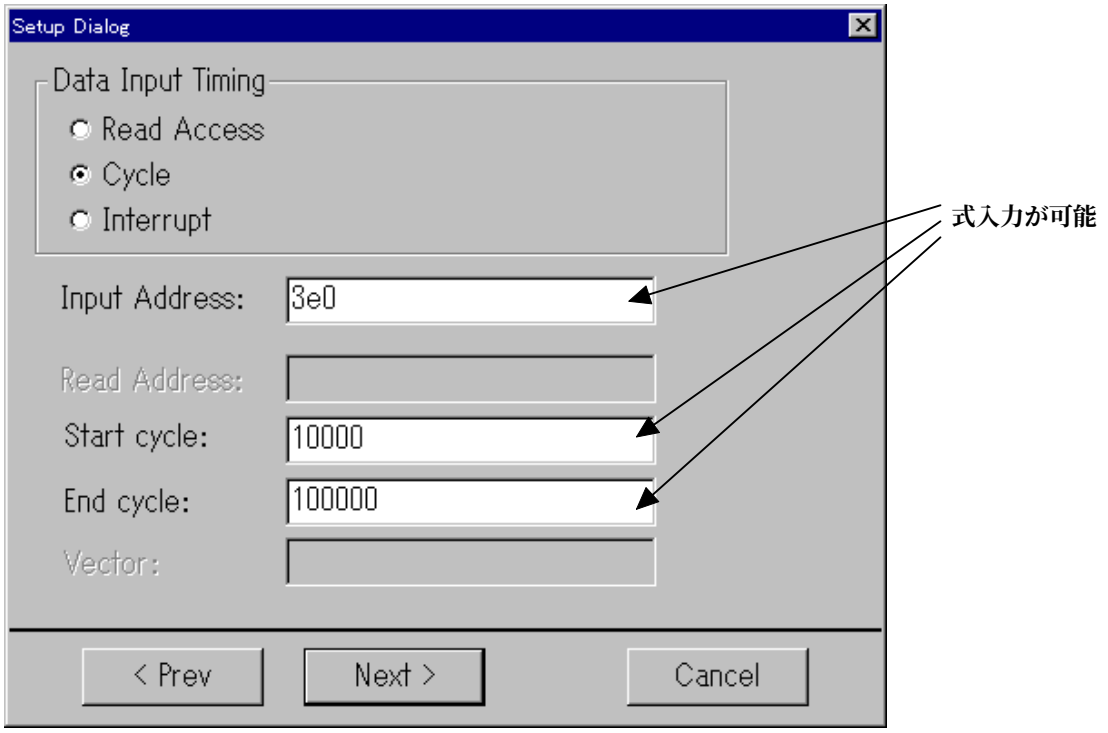
3. ある仮想割り込みが発生したタイミングで、データを入力したい場合
   指定された仮想割り込みが発生した時に、データを入力することができます。
   例えば、割り込みハンドラの中で SFR のメモリを参照している場合などに利用します。
   この場合、割り込み同期入力の設定を行って下さい。

1.2 サイクル同期入力の設定

サイクルに同期した仮想ポート入力を設定するには、I/O ウィンドウの[Option]→[Setup]メニュー（あるいは Setup ボタン）を選択して下さい。選択すると次のダイアログがオープンします。

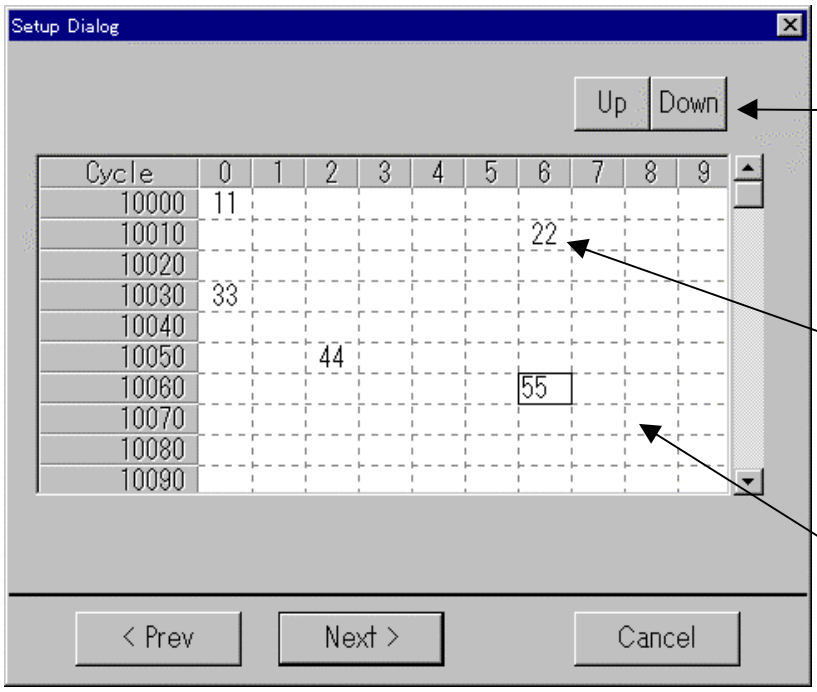


ここで、Set Virtual Port Input の欄を選択し Next ボタンを押して下さい（Cancel ボタンを押すと設定を取りやめてダイアログをクローズします）。仮想ポート入力のタイミングを設定するダイアログがオープンします。



まず、Data Input Timing の欄で Cycle を指定します。次に、Input Address 欄に仮想ポート入力を行いたいアドレス（データを入力するアドレス）を 16 進数値で指定します。そして、仮想ポート入力を開始するサイクルと終了サイクルを 10 進数値で、Start Cycle,End Cycle にそれぞれ指定します。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。

仮想ポート入力のデータを設定するマトリクスダイアログがオープンします。



設定したデータの前データ(UP)、次データ(DOWN)の検索を行います。

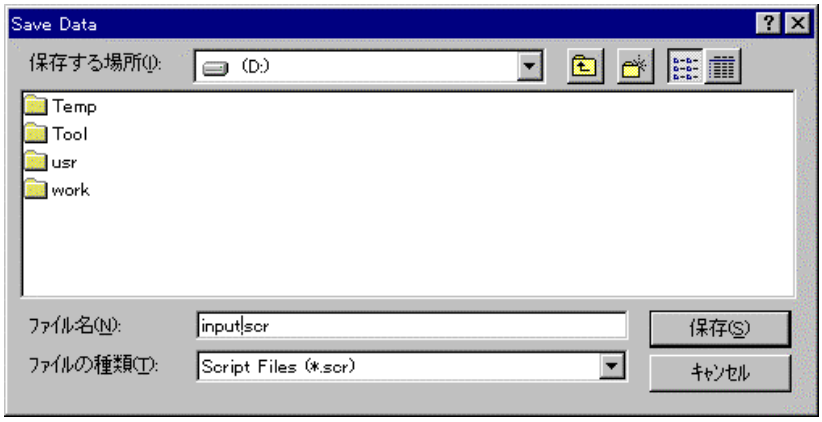
このエレメントの設定例は、「10016 サイクル目で 0x22 を設定」したことを意味します。

目的のエレメントをマウスでダブルクリックし、入力値を設定します。

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

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

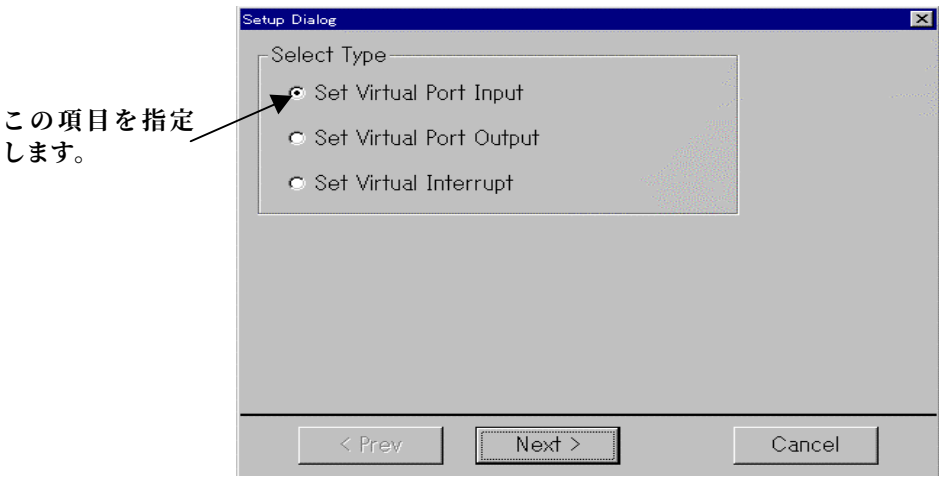
すべてのデータの入力後、Next ボタンを押してください。
設定した仮想ポート入力のデータをファイル（仮想ポート入力ファイル）に保存するためのダイアログがオープンします。



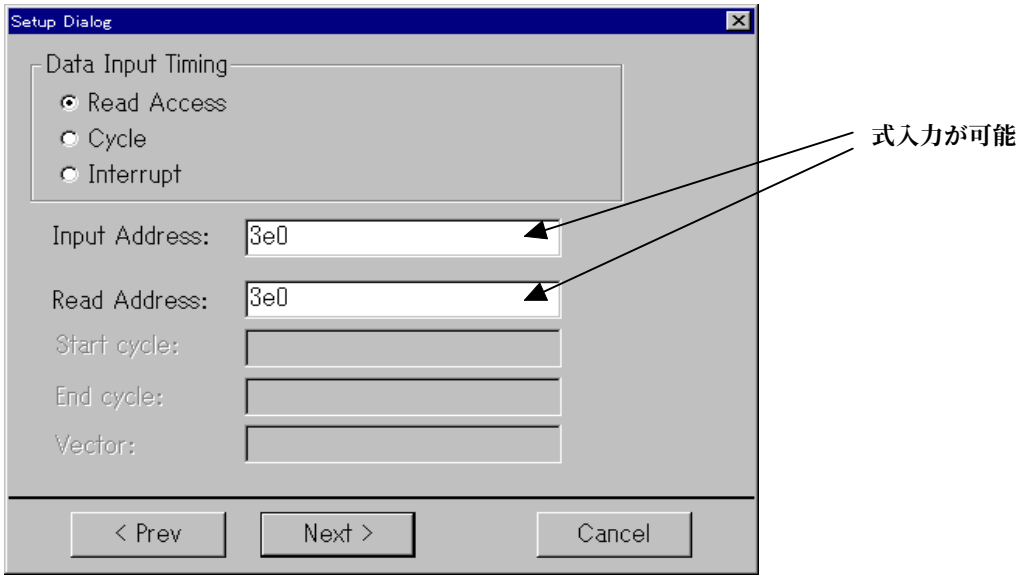
ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O ウィンドウの[Option]→[Load]メニュー（あるいは Load ボタン）で再度読み込むことができます。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、サイクルに同期した仮想ポート入力の設定が完了します。

1.3 リードアクセス同期入力の設定

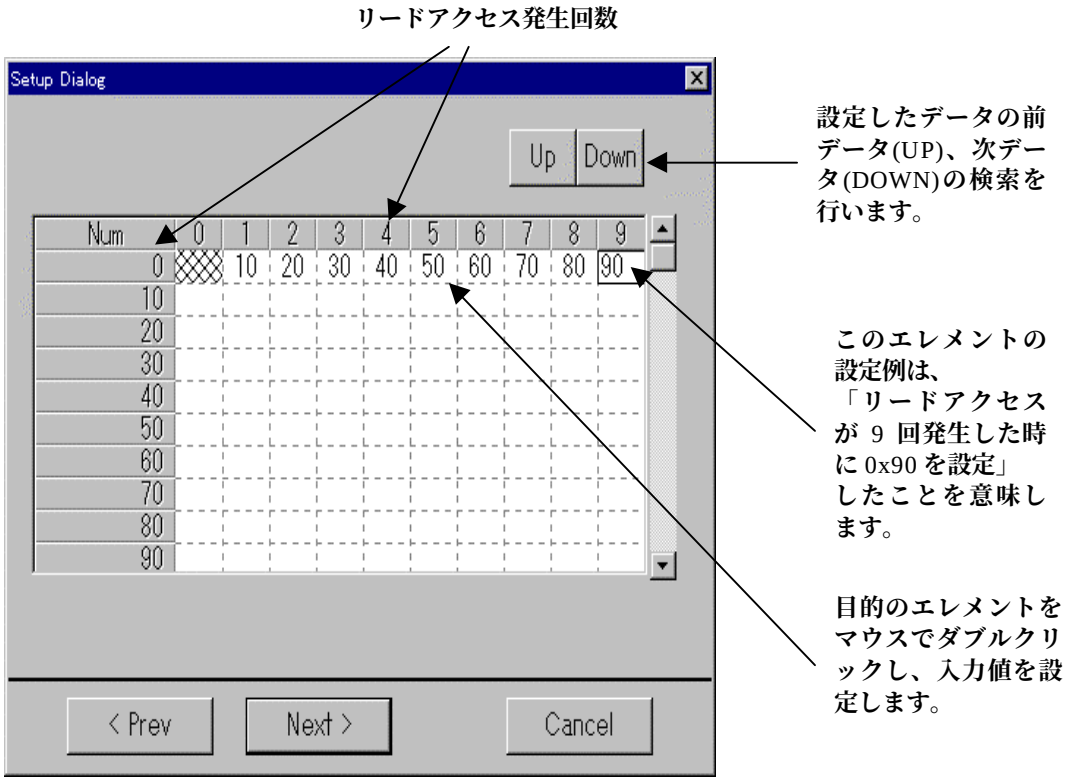
リードアクセスに同期した仮想ポート入力を設定するには、I/O ウィンドウの[Option]→[Setup]メニュー （あるいは Setup ボタン）を選択して下さい。選択すると次のダイアログがオープンします。



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



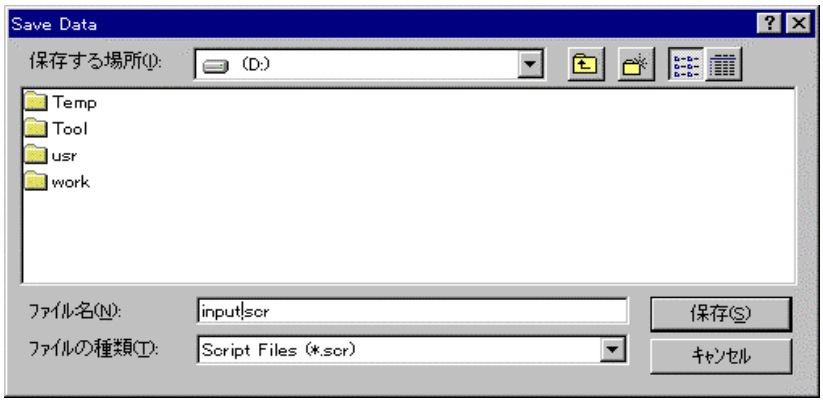
まず、Data Input Timing の欄で Read Access を指定します。次に、Input Address 欄に仮想ポート入力を行いたいアドレス（データを入力するアドレス）を 16 進数値で指定します。そして、Read Address 欄にリードアクセス(メモリの読み込み)が行われるアドレスを入力して下さい（ここで指定したメモリにリードアクセスが発生した時に仮想ポート入力を行います）。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想ポート入力のデータを設定するマトリクスダイアログがオープンします。



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

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

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

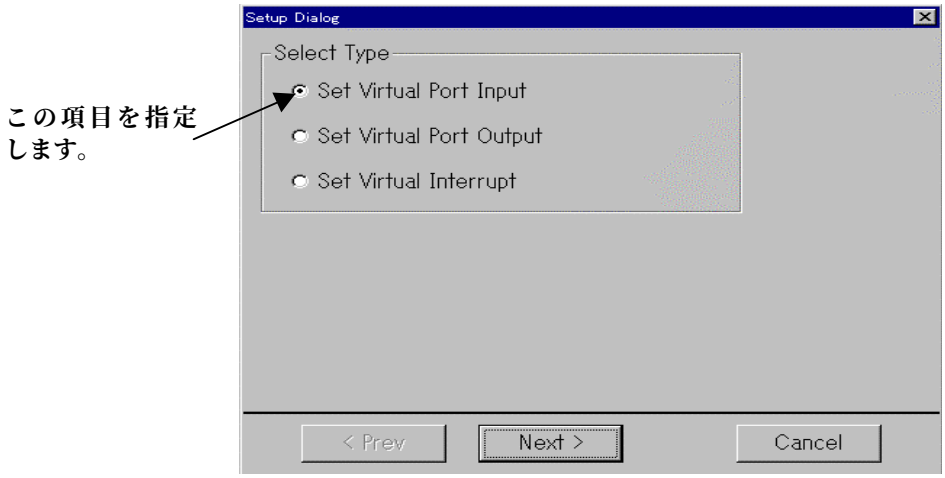


ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。

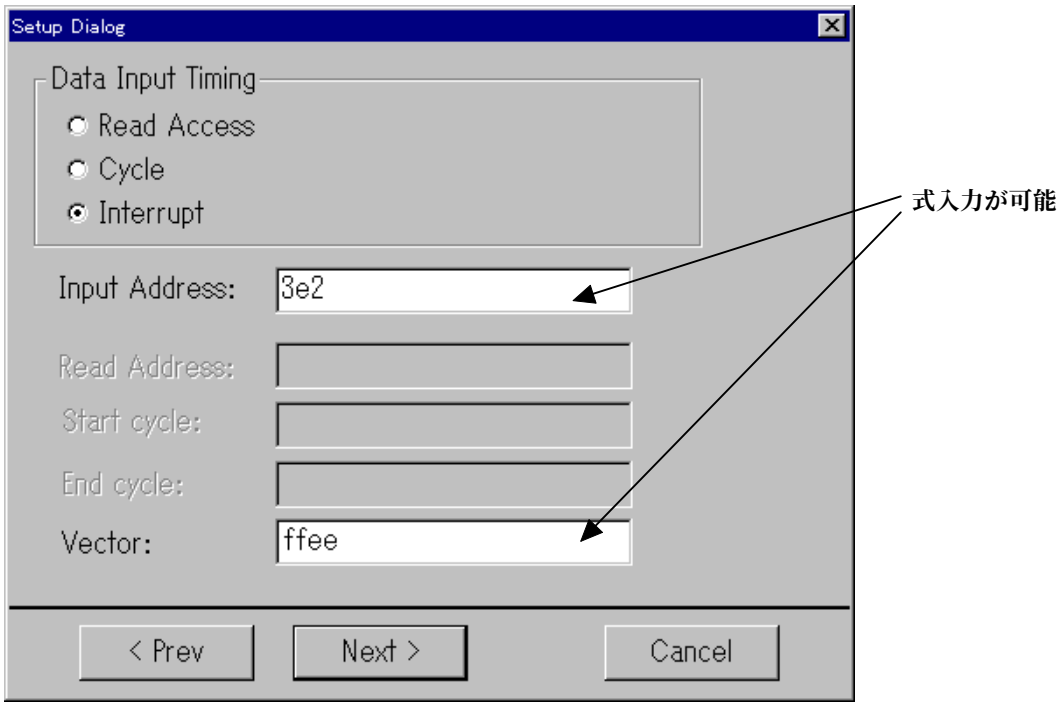
保存したファイルは、I/O ウィンドウの[Option]→[Load]メニュー（あるいは Load ボタン）で再度読み込むことができます。ファイル名を入力したら、保存ボタンを押して下さい。
これで、リードアクセスに同期した仮想ポート入力の設定が完了します。

1.4 割り込み同期入力の設定

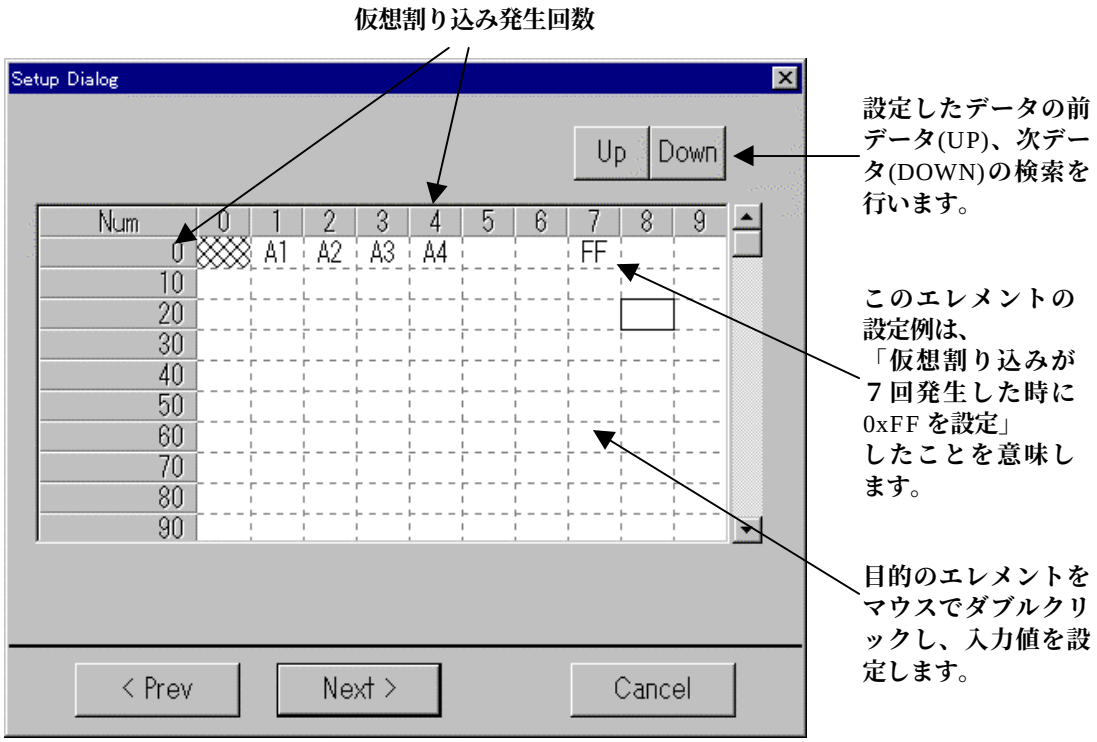
仮想割り込みに同期した仮想ポート入力を設定するには、I/O ウィンドウの[Option]→[Setup]メニュー （あるいは Setup ボタン）を選択して下さい。選択すると次のダイアログがオープンします。



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



まず、Data Input Timing の欄で Interrupt を指定します。次に、Input Address 欄に仮想ポート入力を行いたいアドレス（データを入力するアドレス）を 16 進数値で指定します。そして、Vector 欄に仮想ポート入力のタイミングとなる仮想割り込みのベクタアドレスを入力して下さい。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想ポート入力のデータを設定するマトリクスダイアログがオープンします。

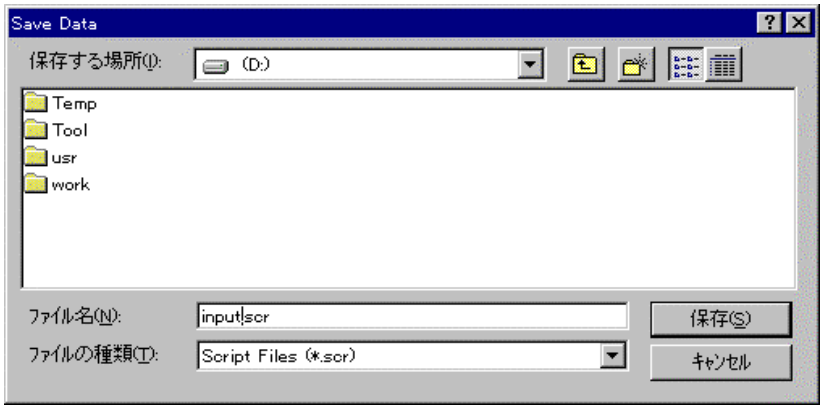


このダイアログでは、実際にメモリに入力するデータを設定します。
データの設定手順は以下のように行って下さい。

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

すべてのデータの入力後、Next ボタンを押してください。

設定した仮想ポート入力のデータをファイル（仮想ポート入力ファイル）に保存するためのダイアログがオープンします。



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

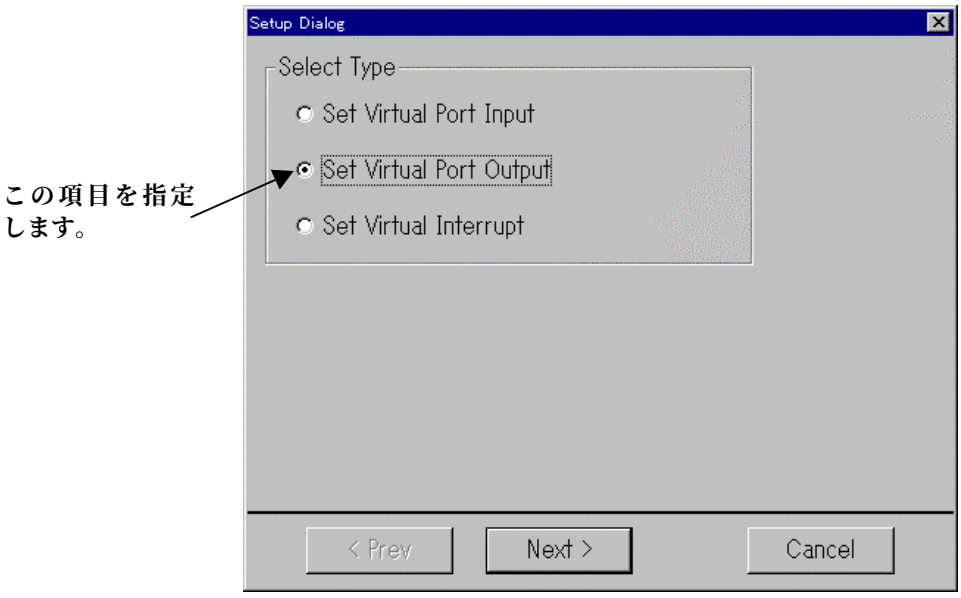
2 I/O ウィンドウでの仮想ポート出力の設定

2.1 概要

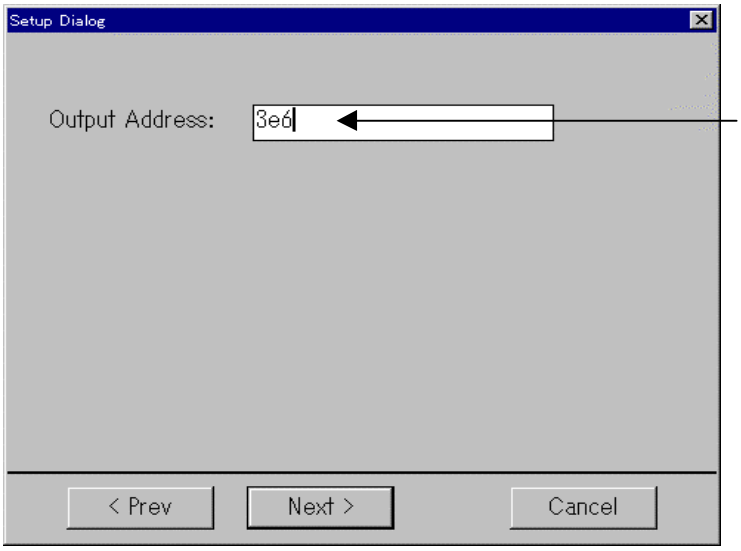
仮想ポート出力機能を利用すると、プログラムによりあるメモリアドレスにデータの書き込みが発生した時に、その書き込まれたデータ値とその時のサイクルを記録することができます。
記録されたデータは、グラフや数値形式で確認できます。
なお、記録されるデータの個数は、プログラム実行開始時から最大 30000 データです。

2.2 仮想ポート出力の設定

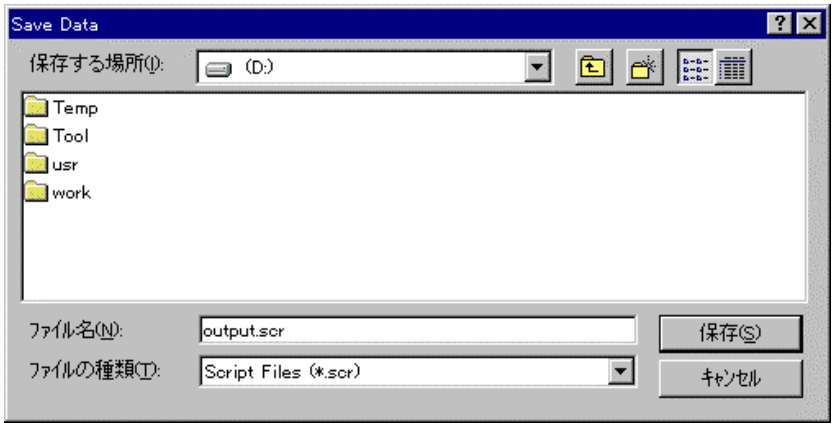
仮想ポート出力を設定するには、I/O ウィンドウの[Option]→[Setup]メニュー （あるいは Setup ボタン）を選択して下さい。選択すると次のダイアログがオープンします。



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



仮想ポート出力の監視を行うアドレスを Output Address 欄に入力して下さい。
入力後、Next ボタンを押してください。
仮想ポート出力の結果を保存(記録)するファイル（仮想ポート出力ファイル）を指定するためのダイアログがオープンします。（PD38SIMは、プログラム実行中に発生した仮想ポート出力をファイルに保存し、プログラム停止時にそのファイルを参照します）。



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

3 I/O ウィンドウでの仮想割り込みの設定

3.1 概要

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

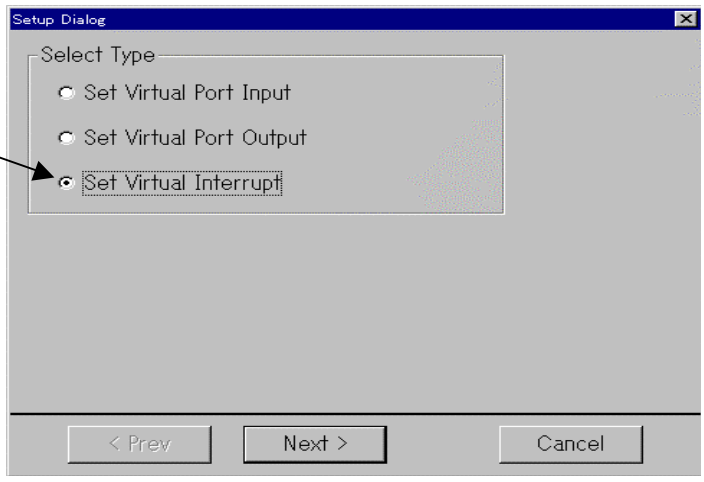
1. 時間の変化とともに、仮想割り込みを発生させる場合
   プログラムの実行が指定サイクルになった時に、仮想割り込みを発生させることができます。
   この場合、サイクル同期割り込みの設定を行って下さい。

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

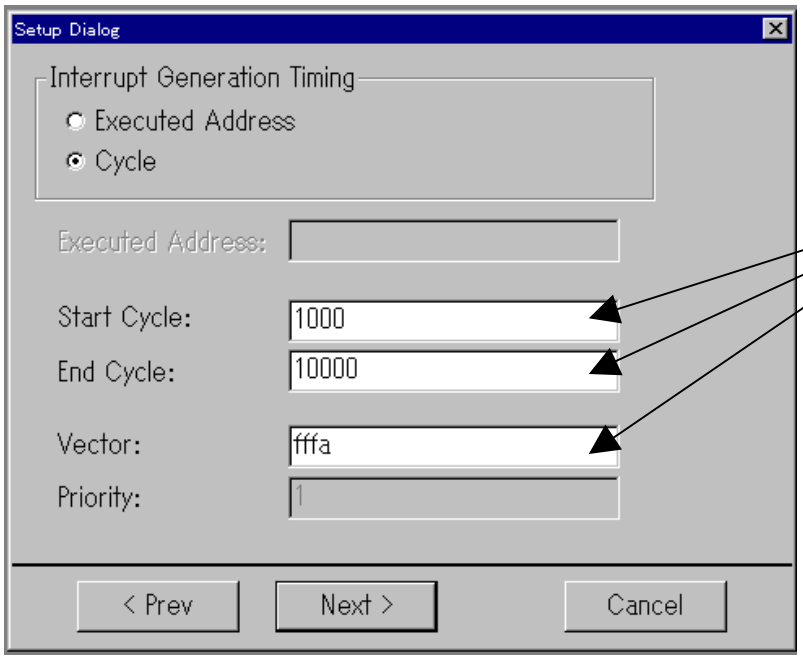
3.2 サイクル同期割り込みの設定

サイクルに同期した仮想割り込みを設定するには、I/O ウィンドウの[Option]→[Setup]メニュー （あるいは Setup ボタン）を選択して下さい。選択すると次のダイアログがオープンします。

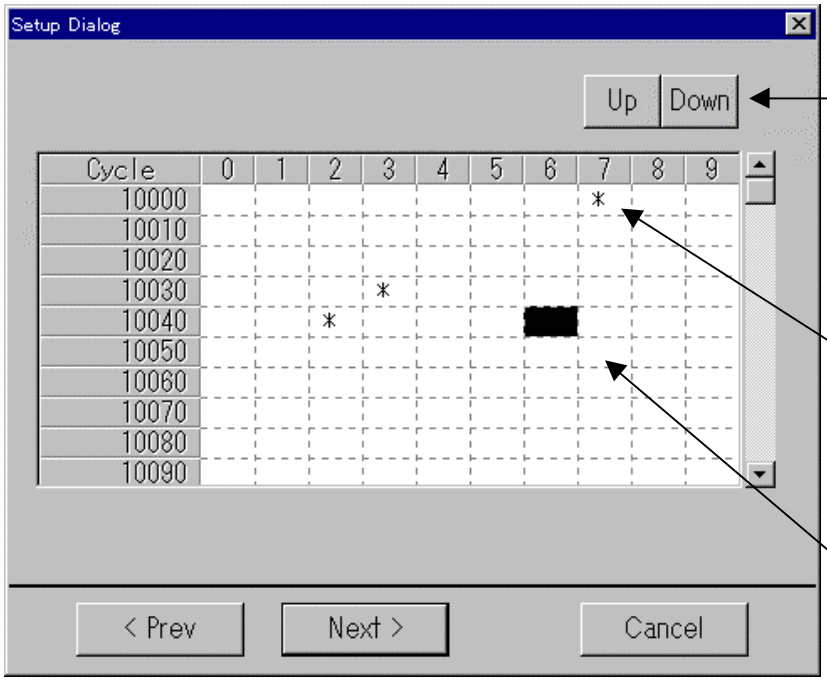
この項目を指定します。



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



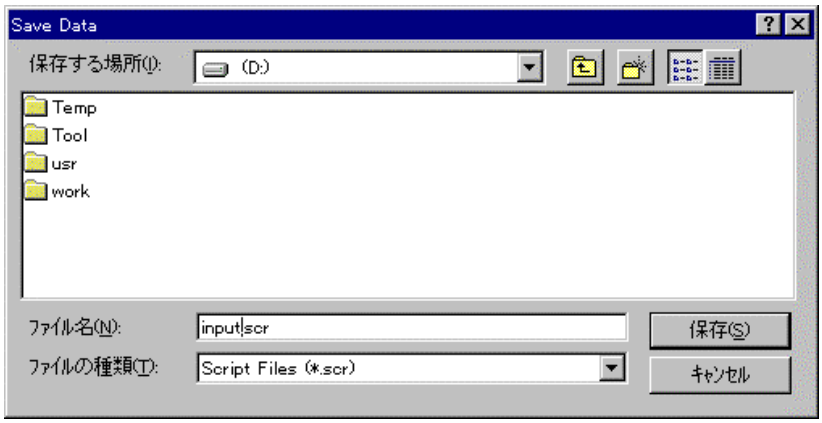
まず、Interrupt Generation Timing の欄で Cycle を指定します。次に、仮想割り込みを開始するサイクルと終了サイクルを 10 進数値で、Start Cycle,End Cycle にそれぞれ指定します。そして、発生させる仮想割り込みのベクタアドレスを 16 進数値で、Vector に指定します。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。
仮想割り込みを設定するマトリクスダイアログがオープンします。



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

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

すべての仮想割り込みの設定後、Next ボタンを押してください。
設定した仮想割り込みのデータをファイル（仮想割り込みファイル）に保存するためのダイアログがオープンします。

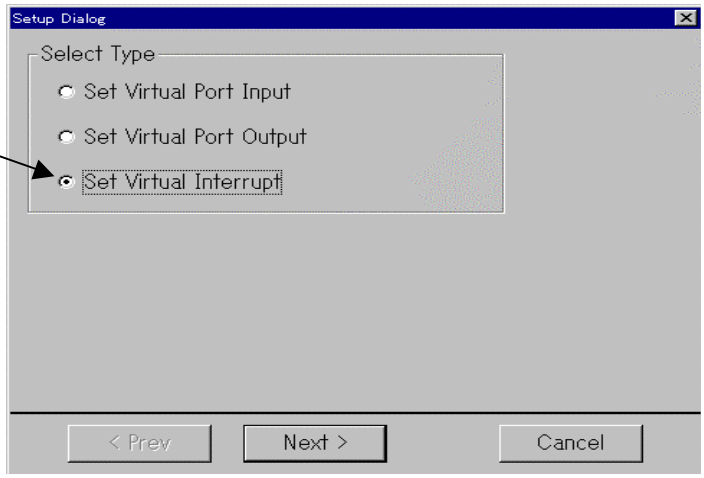


ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O ウィンドウの[Option]→[Load]メニュー（あるいは Load ボタン）で再度読み込むことができます。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、サイクルに同期した仮想割り込みの設定が完了します。

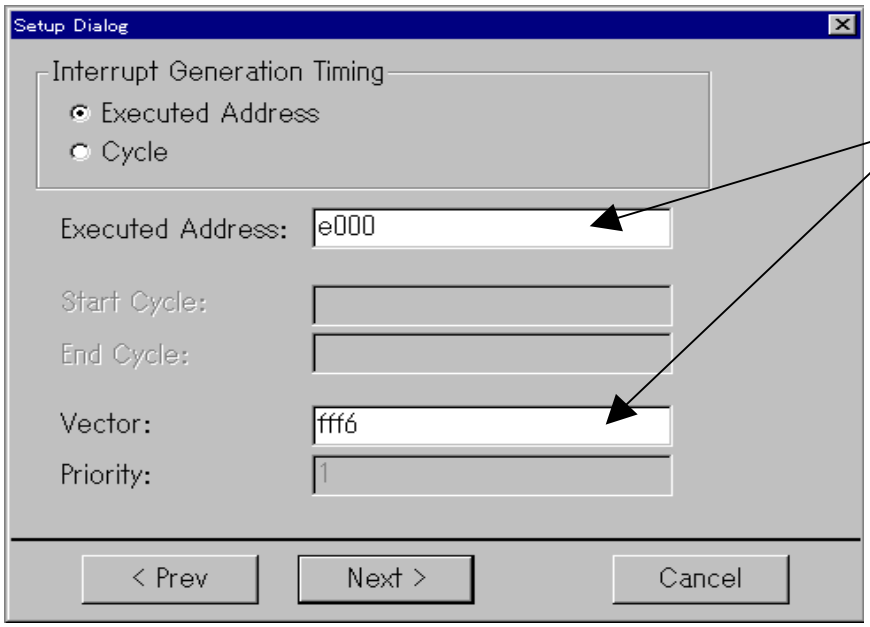
3.3 実行アドレス同期割り込みの設定

アドレスに同期した仮想割り込みを設定するには、I/O ウィンドウの[Option]→[Setup]メニュー （あるいは Setup ボタン）を選択して下さい。選択すると次のダイアログがオープンします。

この項目を指定します。

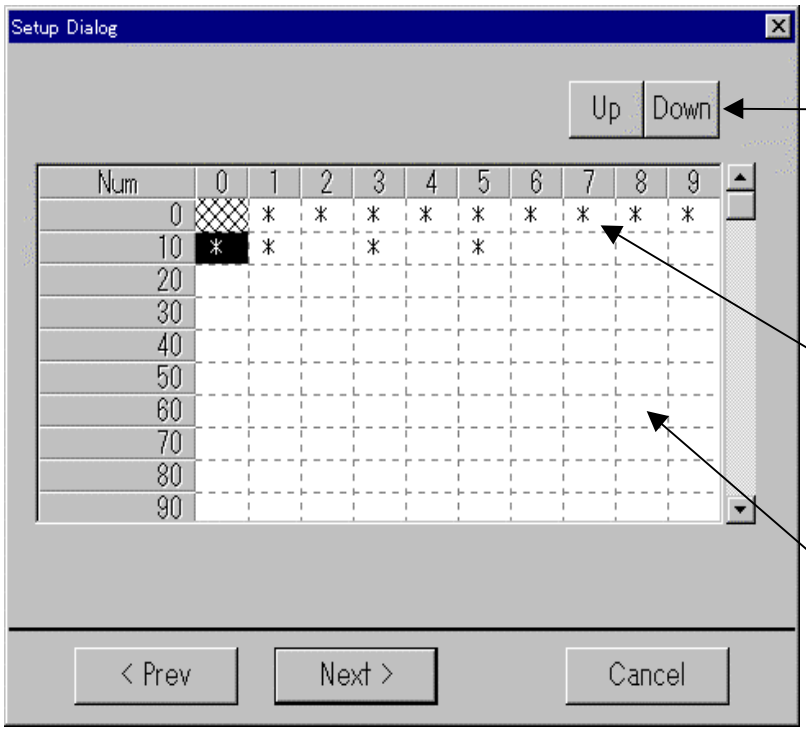


ここで、Set Virtual Interrupt の欄を選択し Next ボタンを押して下さい（Cancel ボタンを押すと設定を取りやめてダイアログをクローズします）。仮想割り込みのタイミングを設定するダイアログがオープンします。



まず、Interrupt Generation Timing の欄で Executed Address を指定します。次に、仮想割り込みを行うタイミングとなる実行アドレス（ここで指定したアドレスを実行した時に仮想割り込みを行います）を Executed Address に指定します。そして、発生させる仮想割り込みのベクタアドレスを 16 進数値で、Vector に指定します。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。

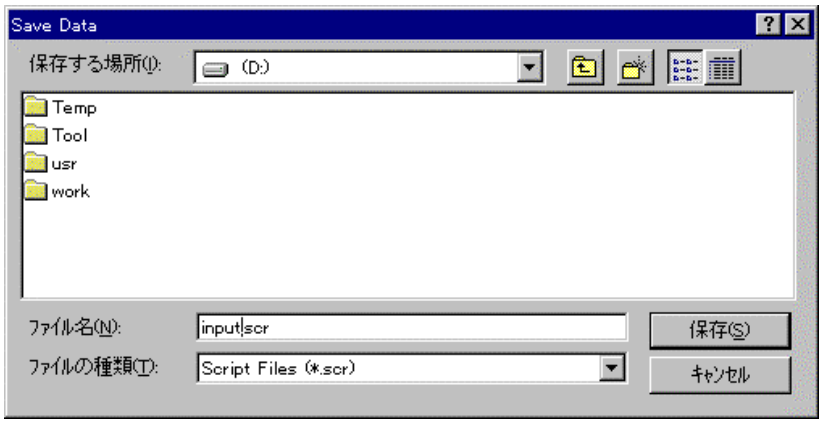
仮想割り込みを設定するマトリクスダイアログがオープンします。



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

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

すべての仮想割り込みの設定後、Next ボタンを押してください。
設定した仮想割り込みのデータをファイル（仮想割り込みファイル）に保存するためのダイアログがオープンします。



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

4 I/O ウィンドウのその他の機能

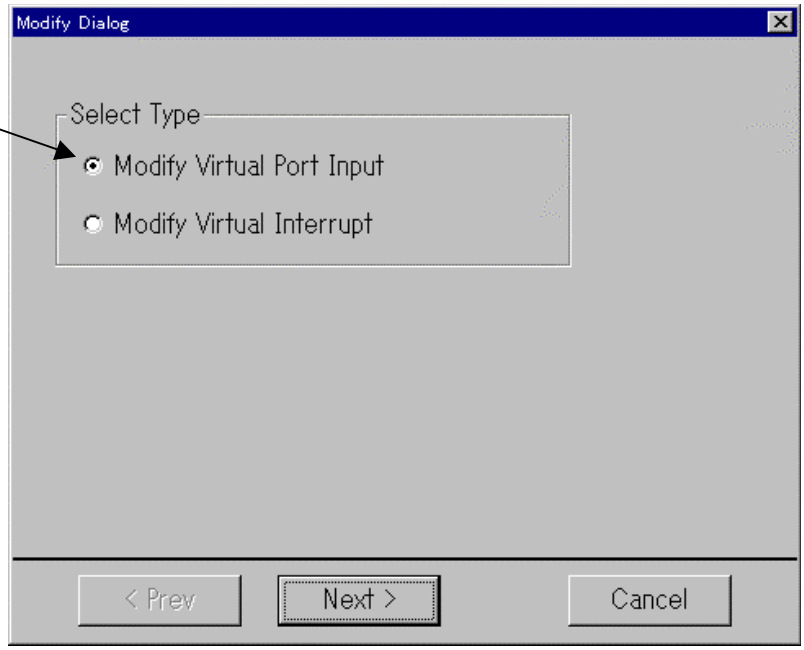
4.1 仮想ポート入力、仮想割り込みの設定データを変更するには

Setup メニューで設定した仮想ポート入力や仮想割り込みの設定データを変更することができます。

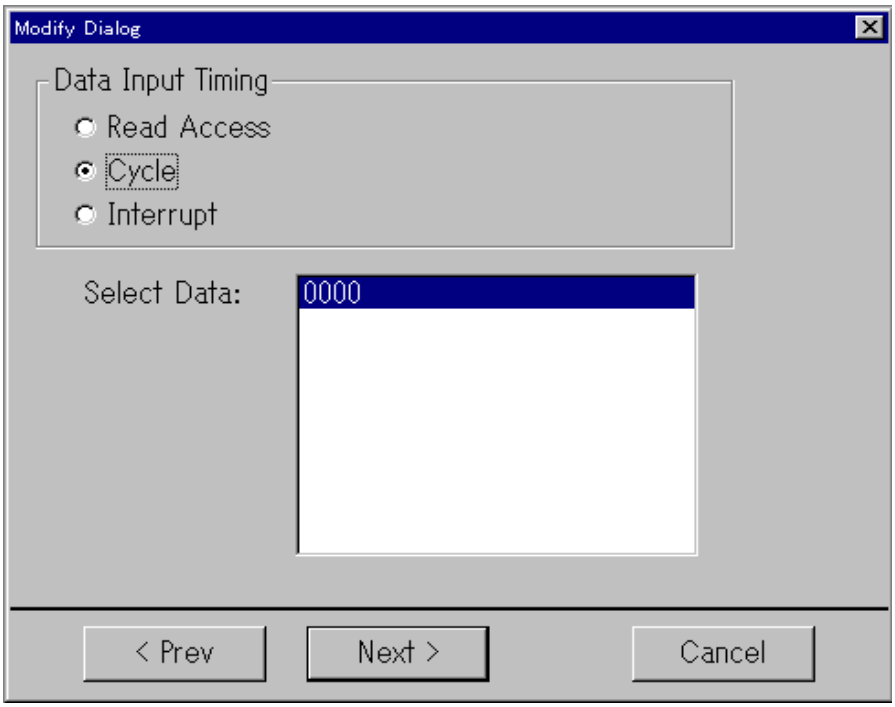
4.1.1 仮想ポート入力の設定データの変更

設定データを変更するには、[Option]→[Modify]メニュー（あるいは Modify ボタン）を選択して下さい。選択すると以下のようなダイアログが表示されます。

この項目を指定します。

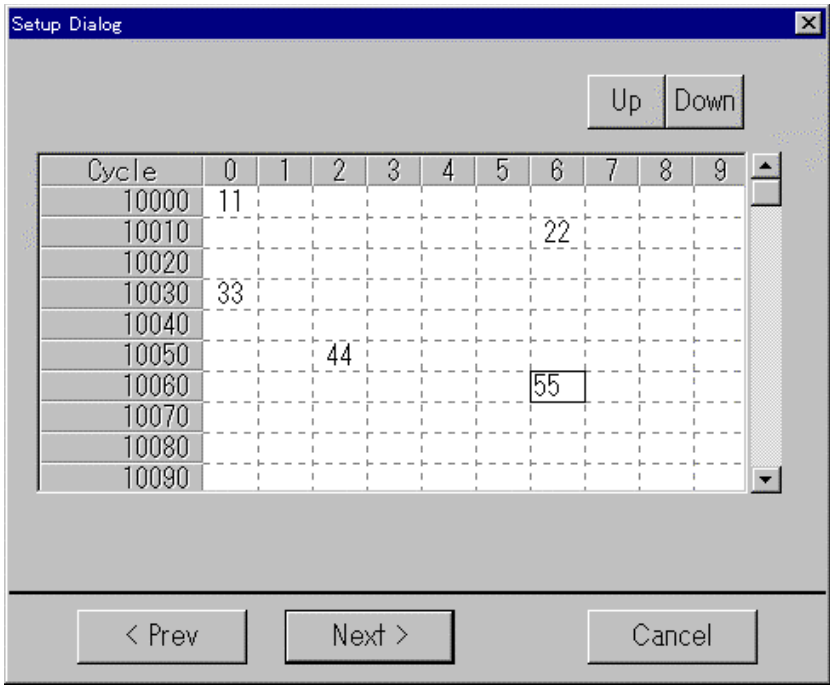


ここで、Modify Virtual Port Input の欄を選択し Next ボタンを押して下さい（Cancel ボタンを押すと設定を取りやめてダイアログをクローズします）。設定を変更したい仮想ポート入力を選択するダイアログがオープンします。

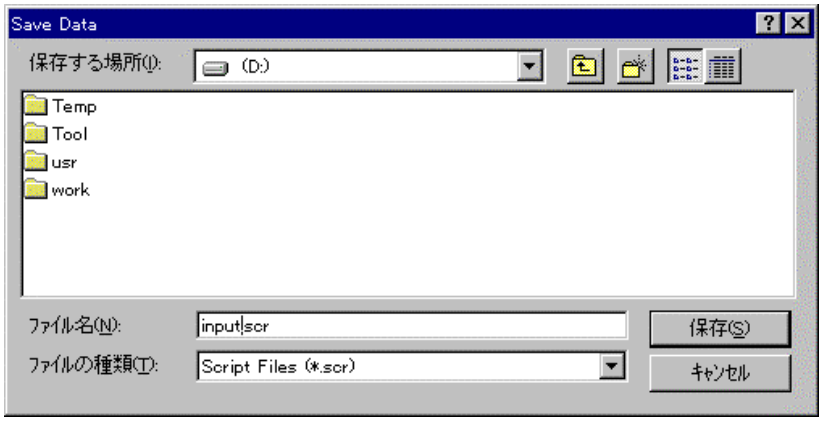


まず、Data Input Timing の欄で変更したい仮想ポート入力の種類を選択します。選択すると、現在設定されている仮想ポート入力の一覧が Select Data 欄に表示されます。ここで、変更したい仮想ポート入力を選択して下さい。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。

仮想ポート入力を変更するためのマトリクスダイアログがオープンします。



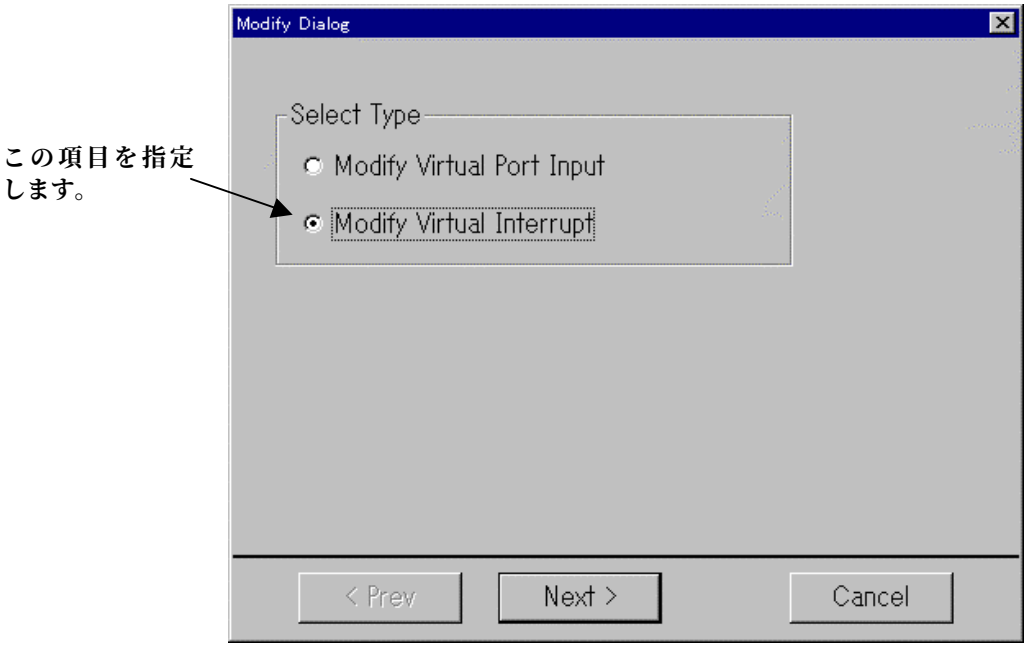
ここで、データの変更を行って下さい。データ変更方法は、仮想ポート入力の設定方法と同様です。（1．I/O ウィンドウでの仮想ポート入力の設定方法を参照下さい）。
データの変更後、Next ボタンを押してください。
設定した仮想ポート入力のデータをファイル（仮想ポート入力ファイル）に保存するためのダイアログがオープンします。



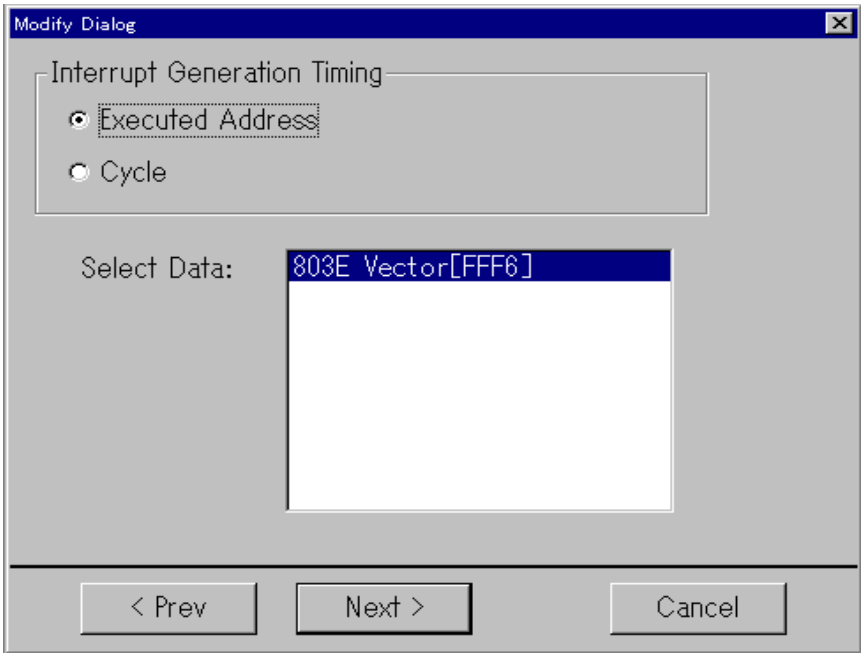
ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O ウィンドウの[Option]→[Load]メニュー（あるいは Load ボタン）で再度読み込むことができます。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、仮想ポート入力のデータの変更が完了します。

4.1.2 仮想割り込みの設定データの変更

設定データを変更するには、[Option]→[Modify]メニュー（あるいは Modify ボタン）を選択して下さい。選択すると以下のようなダイアログが表示されます。

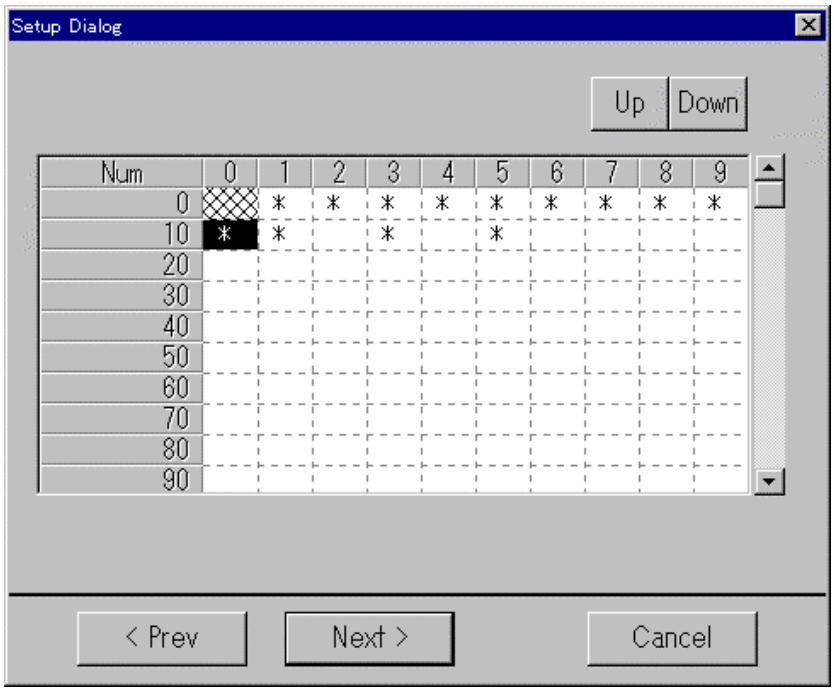


ここで、Modify Virtual Interrupt の欄を選択し Next ボタンを押して下さい（Cancel ボタンを押すと設定を取りやめてダイアログをクローズします）。設定を変更したい仮想割り込みを選択するダイアログがオープンします。

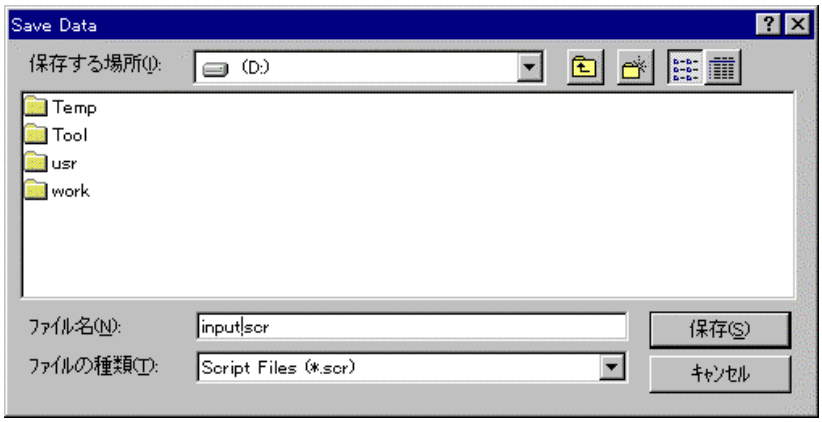


まず、Interrupt Generation Timing の欄で変更したい仮想割り込みの種類を選択します。選択すると、現在設定されている仮想割り込みの一覧が Select Data 欄に表示されます。ここで、変更したい仮想割り込みを選択して下さい。その後、Next ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。

仮想割り込みを変更するためのマトリクスダイアログがオープンします。



ここで、データの変更を行って下さい。データ変更方法は、仮想割り込みの設定方法と同様です。（1. I/O ウィンドウでの仮想割り込みの設定方法を参照下さい）。
データの変更後、Next ボタンを押してください。
設定した仮想割り込みのデータをファイル（仮想割り込みファイル）に保存するためのダイアログがオープンします。

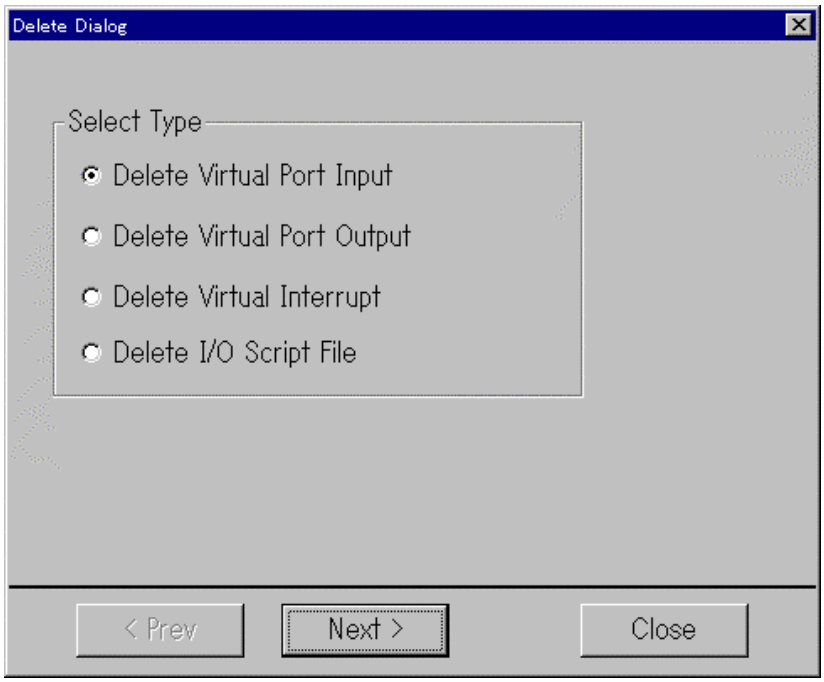


ここで、設定したデータを保存するディレクトリとファイル名を入力して下さい。
保存したファイルは、I/O ウィンドウの[Option]→[Load]メニュー（あるいは Load ボタン）で再度読み込むことができます。
ファイル名を入力したら、保存ボタンを押して下さい。
これで、仮想割り込みのデータの変更が完了します。

4.2 設定した仮想ポート入力、仮想ポート出力、仮想割り込み、I/O スクリプトファイルを削除するには

Setup メニューで設定した仮想ポート入力、仮想ポート出力、仮想割り込み、及び I/O スクリプトファイルを削除できます。

削除するには、[Option]→[Delete]メニュー（あるいは Delete ボタン）を選択して下さい。選択すると以下のようなダイアログが表示されます。

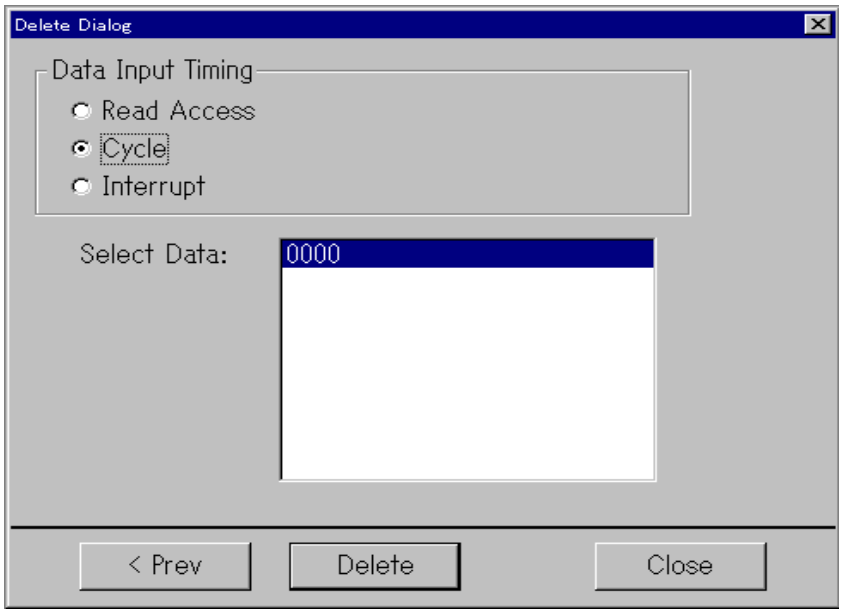


- 仮想ポート入力の設定を削除する場合
  Delete Virtual Port Input を選択して下さい。
- 仮想ポート出力の設定を削除する場合
  Delete Virtual Port Output を選択して下さい。
- 仮想割り込みの設定を削除する場合
  Delete Virtual Interrupt を選択して下さい。
- I/O スクリプトファイルの登録を削除する場合
  Delete I/O Script File を選択して下さい。

選択後の操作に関しては、以下の手順で行って下さい。

4.2.1 仮想ポート入力の削除

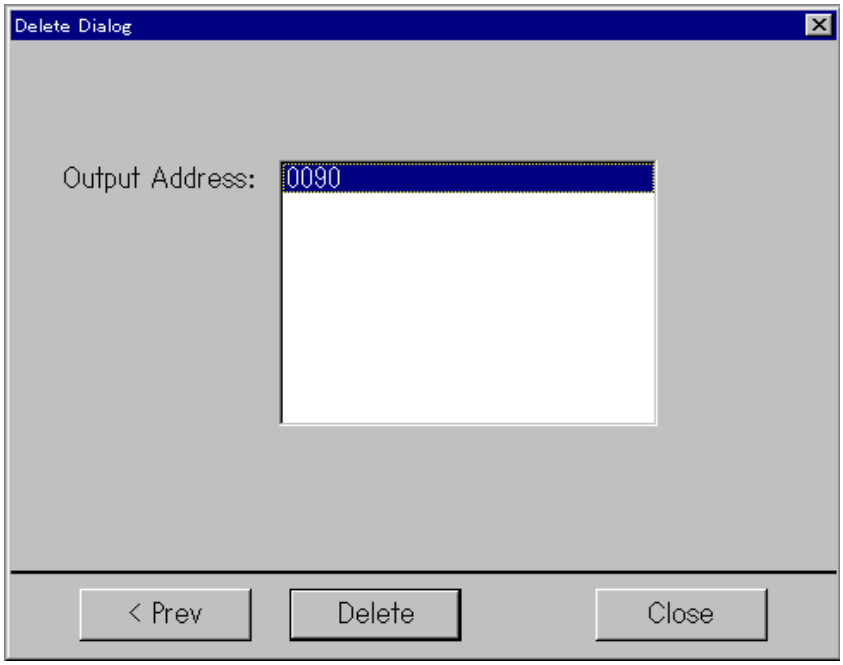
Delete Virtual Port Input を選択後、Next ボタンを押すと次のダイアログがオープンします。



まず、Data Input Timing の欄で削除したい仮想ポート入力の種類を選択します。選択すると、現在設定されている仮想ポート入力の一覧が Select Data 欄に表示されます。ここで、削除したい仮想ポート入力を選択して下さい。その後、Delete ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。これで、設定が削除されます。
Close ボタンを押すとダイアログがクローズされます。

4.2.2 仮想ポート出力の削除

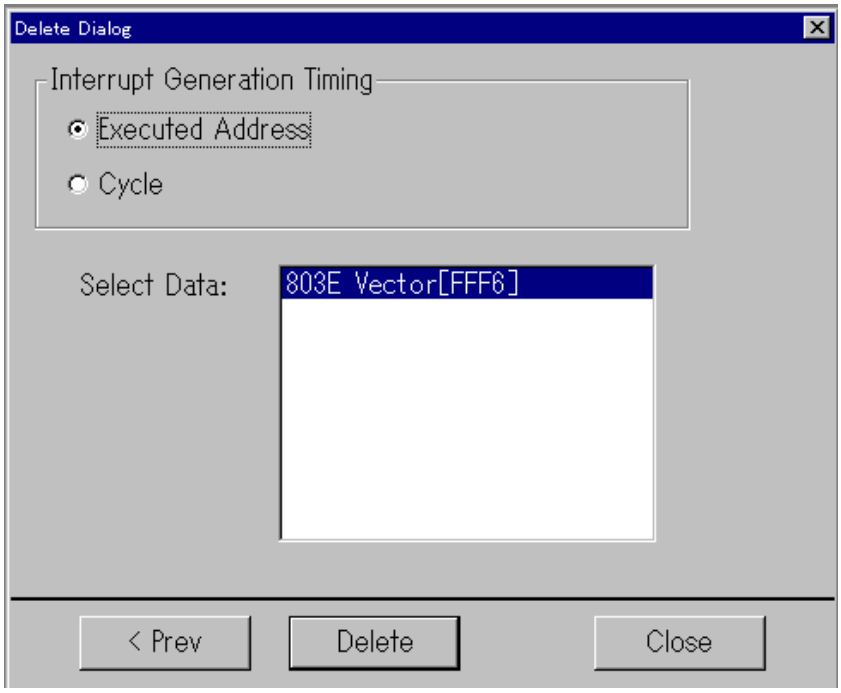
Delete Virtual Output を選択後、Next ボタンを押すと次のダイアログがオープンします。



ここで、削除したい仮想ポート出力を選択して下さい。その後、Delete ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。これで、設定が削除されます。
Close ボタンを押すとダイアログがクローズされます。

4.2.3 仮想割り込みの削除

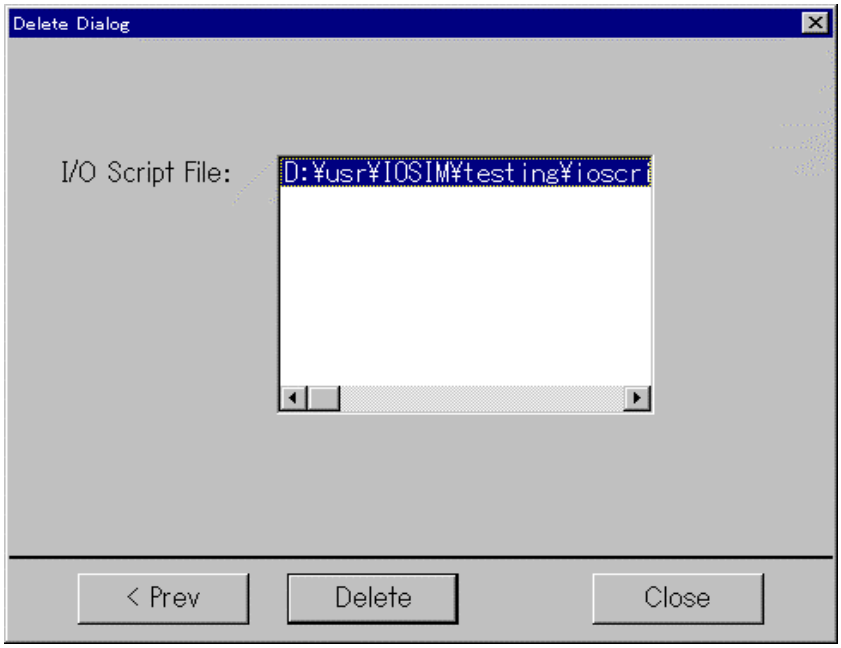
Delete Virtual Interrupt を選択後、Next ボタンを押すと次のダイアログがオープンします。



まず、Interrupt Genaration Timing の欄で削除したい仮想割り込みの種類を選択します。選択すると、現在設定されている仮想割り込みの一覧が Select Data 欄に表示されます。ここで、削除したい仮想割り込みを選択して下さい。その後、Delete ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。これで、設定が削除されます。
Close ボタンを押すとダイアログがクローズされます。

4.2.4 I/O スクリプトファイルの登録の削除

Delete I/O Script File を選択後、Next ボタンを押すと次のダイアログがオープンします。



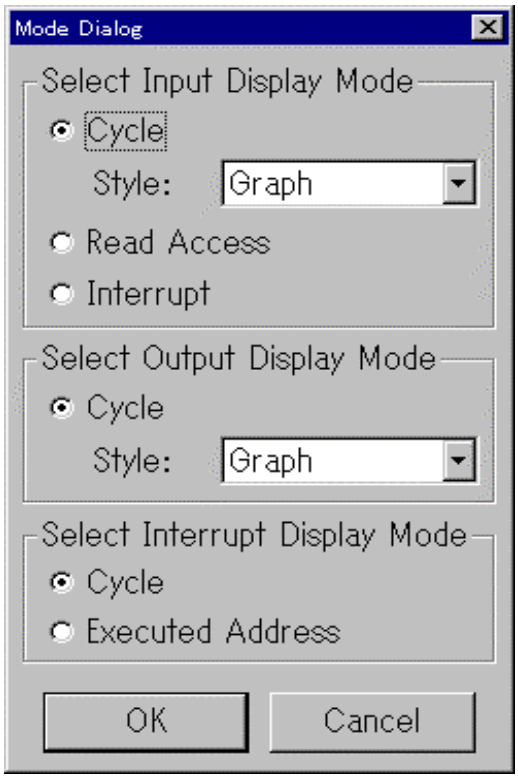
ここで、登録を削除したい I/O スクリプトファイルを選択して下さい。その後、Delete ボタンを押して下さい（ここで Prev ボタンを押すと前の設定ダイアログに戻ることができます）。これで、設定が削除されます。

Close ボタンを押すとダイアログがクローズされます。

4.3 仮想ポート入力、仮想ポート出力、仮想割り込みの表示形式を変更するには

Setup メニューで設定した仮想ポート入力、仮想ポート出力、仮想割り込みの表示形式を変更することができます。

変更するには、[Option]→[Mode]メニュー（あるいは Mode ボタン）を選択して下さい。選択すると以下のようなダイアログが表示されます。



各表示形式の変更方法を以下に説明します。

4.3.1 仮想ポート入力の表示形式を変更する場合

1. サイクル同期入力の表示を行う場合
   Select Input Display Mode 欄の Cycle を選択して下さい。その後、Style 欄で表示形式を選択します。
   - チャート形式で表示する場合、Chart を選択して下さい。
   - 16 進数値形式で表示する場合、Hex を選択して下さい。
   - グラフ形式で表示する場合、 Gragh を選択して下さい。
2. リードアクセス同期入力の表示を行う場合
   Select Input Display Mode 欄の Read Access を選択して下さい。
3. 割り込み同期入力の表示を行う場合
   Select Input Display Mode 欄の Interrupt を選択して下さい。

4.3.2 仮想ポート出力の表示形式を変更する場合

Select Output Display Mode 欄の Style 欄で表示形式を選択します。

- チャート形式で表示する場合、Chart を選択して下さい。
- 16 進数値形式で表示する場合、Hex を選択して下さい。
- グラフ形式で表示する場合、Gragh を選択して下さい。

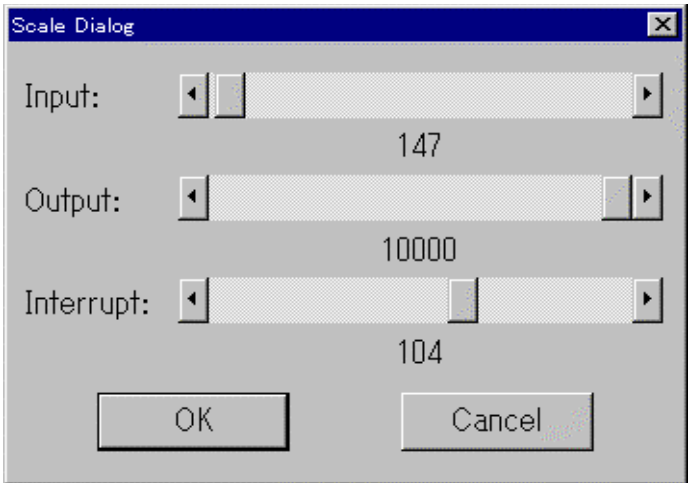
4.3.3 仮想割り込みの表示形式を変更する場合

1. サイクルに同期した仮想割り込みの表示を行う場合
   Select Interrupt Display Mode 欄の Cycle を選択して下さい。
2. アドレスに同期した仮想割り込みの表示を行う場合
   Select Interrupt Display Mode 欄の Executed Address を選択して下さい。

4.4 表示画面のスケールを変更するには

仮想ポート入力、仮想ポート出力、仮想割り込みの表示画面のスケールを変更することができます。スケールとは、１画面に表示できるサイクル数等を変更する機能です。

変更するには、[Option]→[Scale]メニュー（あるいは Scale ボタン）を選択して下さい。選択すると以下のようなダイアログが表示されます。



各表示画面のスケールの変更方法を以下に説明します。

1. 仮想ポート入力の表示画面のスケールを変更する場合
   Input 欄のスクロールバーをスライドさせてスケールを変更して下さい。
2. 仮想ポート出力の表示画面のスケールを変更する場合
   Output 欄のスクロールバーをスライドさせてスケールを変更して下さい。
3. 仮想割り込みの表示画面のスケールを変更する場合
   Interrupt 欄のスクロールバーをスライドさせてスケールを変更して下さい。

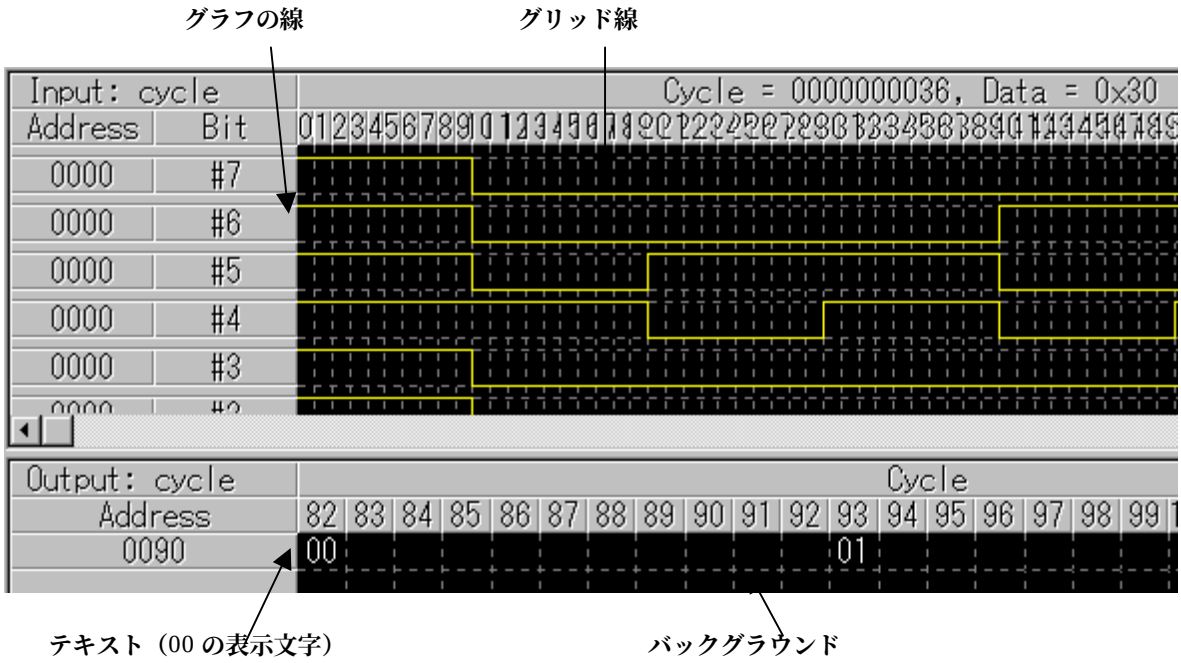
スケールは１から最大 10000 までの間で変更することができます。

例えば、仮想ポート入力をチャート形式で表示している場合、スケールを 50 にすると、1 画面に 50 サイクル分表示されます。

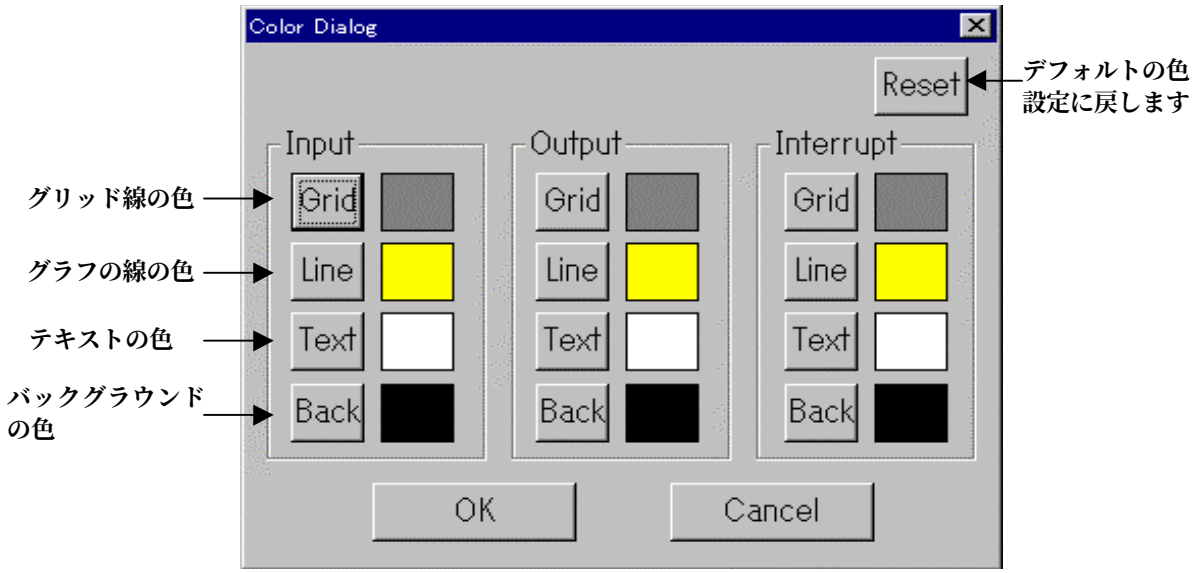
但し、数値形式表示のとき、スケールの最大は 50 です。

4.5 表示画面のカラーを変更するには

仮想ポート入力、仮想ポート出力、仮想割り込みの表示画面のカラーを変更することができます。
カラーを変更できるのは、各データ表示領域のグリッド線、グラフの線、表示文字、背景の４つです。



カラーを変更するには、[Option]→[Color]メニュー（あるいは Color ボタン）を選択して下さい。選択すると以下のようなダイアログが表示されます。

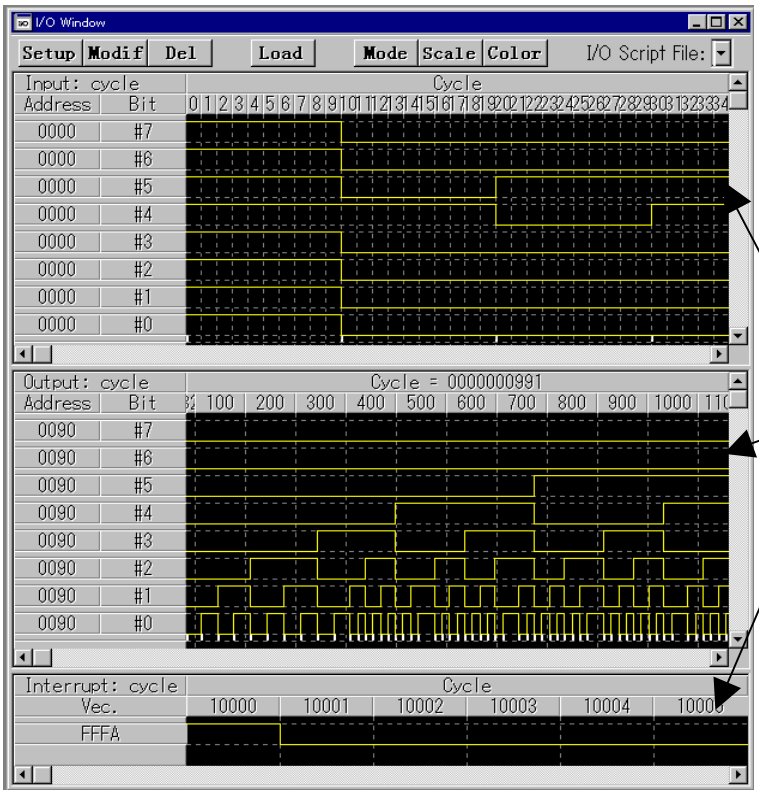


各表示画面のカラーの変更方法を以下に説明します。

1. 仮想ポート入力の表示画面のカラーを変更する場合
   Input 欄でカラーを変更したい項目のボタンを押して下さい。
   押すとカラーを選択するカラーダイアログがオープンします。そこでカラーを指定して下さい。
2. 仮想ポート出力の表示画面のカラーを変更する場合
   Output 欄でカラーを変更したい項目のボタンを押して下さい。
   押すとカラーを選択するカラーダイアログがオープンします。そこでカラーを指定して下さい。
3. 仮想割り込みの表示画面のカラーを変更する場合
   Interrupt 欄でカラーを変更したい項目のボタンを押して下さい。
   押すとカラーを選択するカラーダイアログがオープンします。そこでカラーを指定して下さい。

4.6 表示データを検索するには

設定した仮想ポート入力や仮想割り込みデータ、及び仮想ポート出力の出力データを検索し、ウィンドウの左端に表示します。但し、最後尾のデータは右端に表示します。
検索方法を以下に示します。

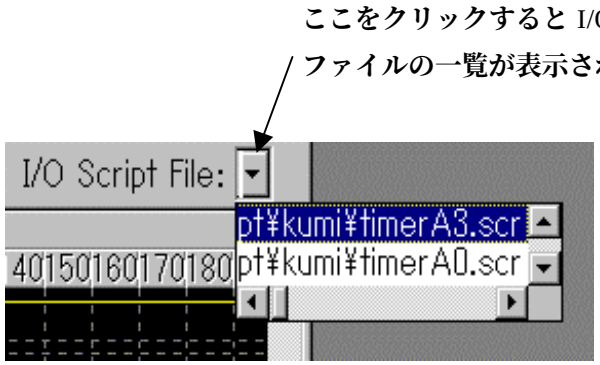


データの検索を開始したい位置にマウスカーソルを移動し、マウスの右ボタンをクリックします。クリックすると次のような、ポップアップメニューが表示されます。

ここで、Before data を選択すると、マウスで指定した位置から前データを検索します。
Next data を選択すると、マウスで指定した位置から次データを検索します。

4.7 登録したI/O スクリプトファイルの一覧表示

登録した I/O スクリプトファイル（I/O スクリプトファイルの詳細に関しては 7 章　I/O スクリプト機能を参照下さい）の一覧を表示することができます。
一覧の表示方法を以下に示します。



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

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

「評価タイミング」

- プログラム実行時、カム実行時
- ステップ実行時
- オーバステップ実行時
- リターン実行時

「プログラムがリセットされた場合の処理」

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

「I/O ウィンドウをクローズした場合の処理」

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

5 GUI 入力ウィンドウの設定

5.1 概要

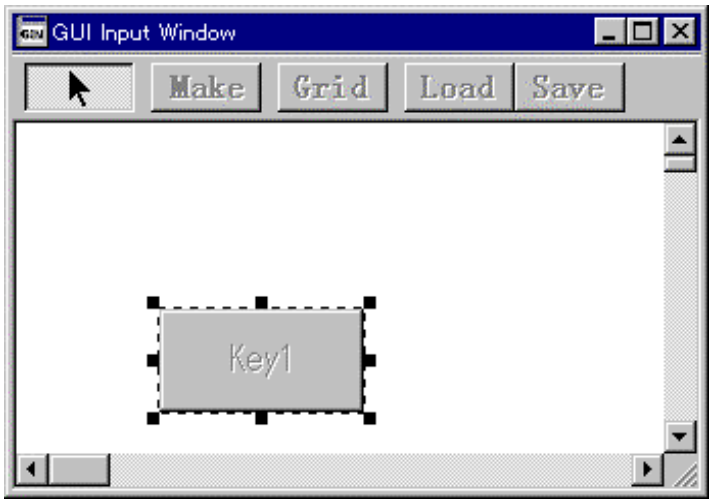
GUI 入力ウィンドウでは、ユーザターゲットシステムの簡単なキー入力パネル（ボタン）をウィンドウ上で作成し、作成したボタンを押すことにより仮想ポート入力や仮想割り込みを行うことができます。
ここでは、ボタンの作成方法について説明します。

5.2 ボタンの作成方法

ボタンの作成は、以下の手順で行って下さい。

1. GUI 入力ウィンドウの[Option]→[Make]メニュー（あるいは Make ボタン）を選択します。
2. 選択した後、マウスを GUI 入力ウィンドウの入力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、ボタンを作成したい位置でマウスの左ボタンをクリックし、クリックしたまま作成したい大きさに広げ、左ボタンを離して下さい。

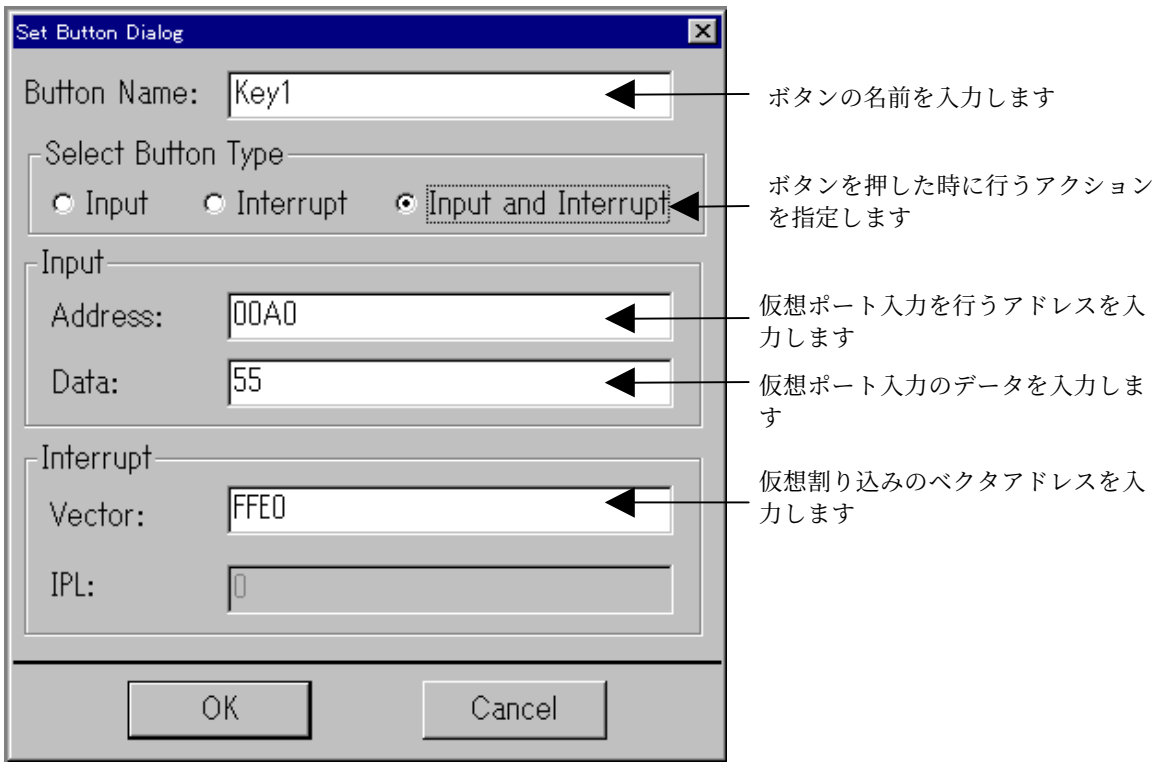
4. 次のようにボタンが作成されます。



5. 次に、作成したボタンをダブルクリックして下さい。

「注意」
（作成したボタンをダブルクリックする前に、ツールバーの↑ボタンが選択されていることを確認して下さい。選択されていない時は、↑ボタンを選択してから作成したボタンをダブルクリックしてください。）

6. 次のようなボタンのアクションを設定するためのダイアログがオープンします。
ボタンのアクションには、仮想ポート入力、仮想割り込み、仮想ポート入力＋仮想割り込みのいずれかを設定します。



(1)ボタンに名前を付けます。（Button Name の欄に名前を入力します）

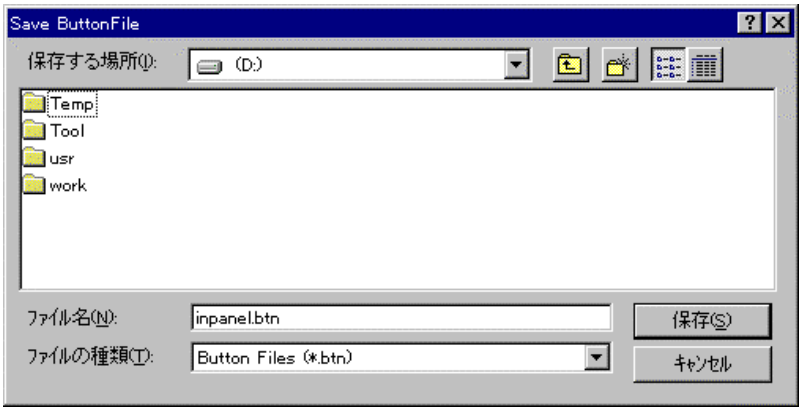
(2)Select Button Type の欄でボタンを押した時に行うアクションを指定します。
●仮想ポート入力を行う場合、Input を選択します。
●仮想割り込みを行う場合、Interrupt を選択します。
●仮想ポート入力と仮想割り込みを同時に発生させる場合、Input and interrupt を選択します。
(3)仮想ポート入力を行う場合、Input の欄にデータを入力したいアドレスと、入力データを入力して下さい。
仮想割り込みを行う場合、Interrupt の欄にベクタアドレスを入力して下さい。

7. ダイアログの OK ボタンを押して下さい。これで、ボタンの作成と設定が完了します。
8. さらにボタンを作成する場合は、上記の１～７の操作を繰り返し行って下さい。

5.3 作成したボタンを保存するには

ボタンの作成が終了した時に、作成したボタンのデータ（設定内容、レイアウト）をファイル（GUI 入力ファイル）に保存できます。保存した GUI 入力ファイルを[Option]→[Load]メニューで再度読み込むことにより、保存したボタンを再現することができます。
ボタンデータの保存は、以下の手順で行って下さい。

GUI 入力ウィンドウの[Option]→[Save]メニュー（あるいは Save ボタン）を選択してください。選択すると以下のダイアログがオープンします。

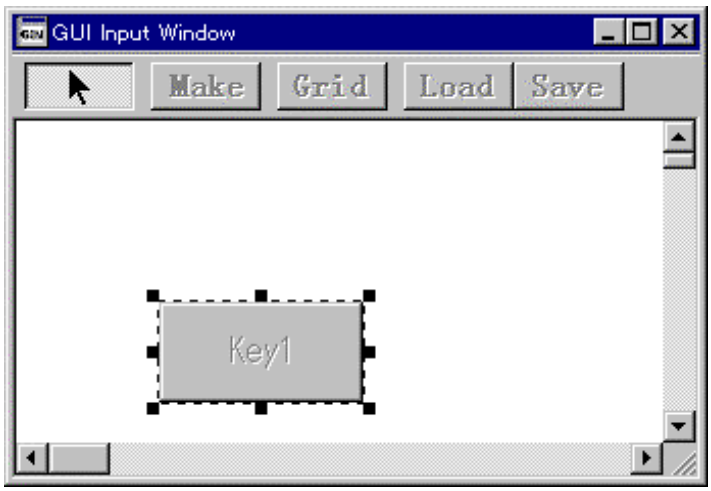


ここで、ボタンデータを保存するディレクトリとファイル名を入力して下さい。
ファイル名を入力したら、保存ボタンを押して下さい。

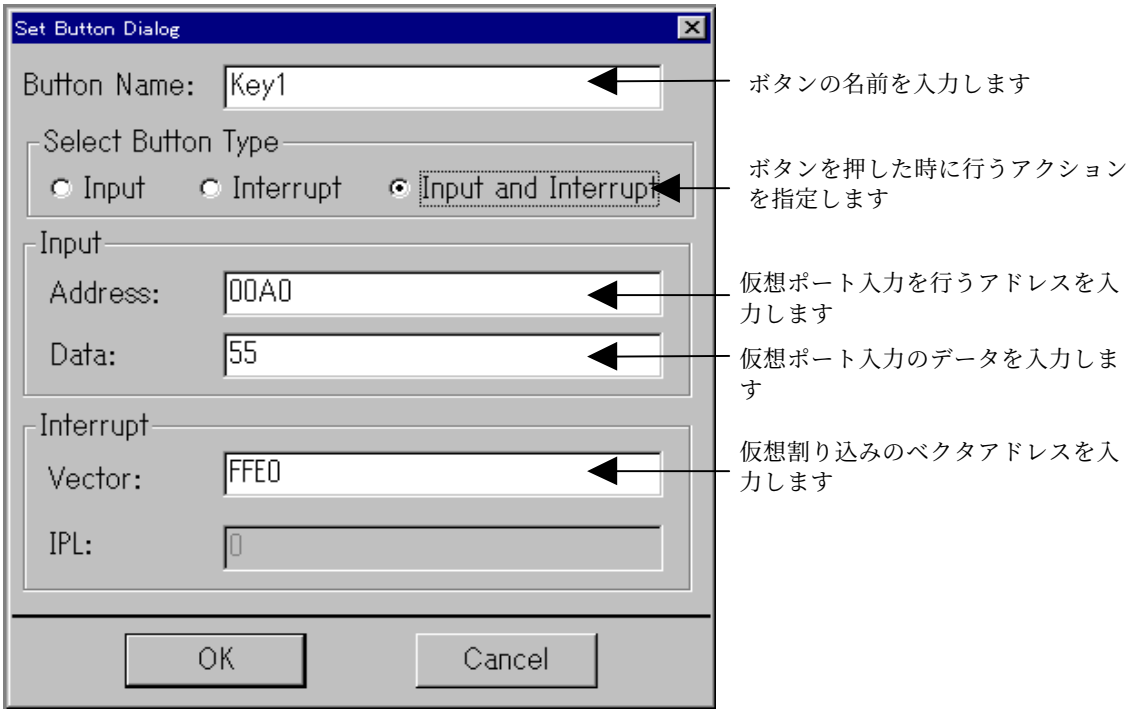
5.4 ボタン作成後にボタンの配置、設定を変更するには

ボタンを作成した後でも、ボタンの配置や設定内容を変更することができます。

1. ボタンの配置を変更するには
   GUI 入力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して下さい。
   選択後、配置を変更したいボタンをマウスの左ボタンでクリックして下さい。



   クリックすると上記のようにボタンが選択された状態になります。この時に、マウスの左ボタンで、ボタンを移動したい位置までドラッグして下さい。
2. ボタンの設定内容を変更するには
   上記１と同様に GUI 入力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して、設定内容を変更したいボタンをマウスの左ボタンでダブルクリックして下さい。
   以下のダイアログがオープンします。



   ここで、設定内容を変更して下さい。

5.5 ボタンをコピーするには

ボタンのコピーは、以下の手順で行って下さい。

1. GUI 入力ウィンドウの[Option]→[Copy]メニューを選択します。
2. 選択した後、マウスを GUI 入力ウィンドウの入力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、コピーしたいボタン上でマウスの左ボタンをクリックして下さい。
4. 次に、GUI 入力ウィンドウの[Option]→[Paste]メニューを選択すると、マウスの左ボタンで選択したボタン上に、新しいボタンがコピーされます。

または、GUI 入力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して下さい。選択後、削除したいボタンをマウスの左ボタンでクリックして選択状態にし、Ctrl+C、Ctrl+V キーを入力して下さい。

5.6 ボタンを削除するには

ボタンの削除は、以下の手順で行って下さい。

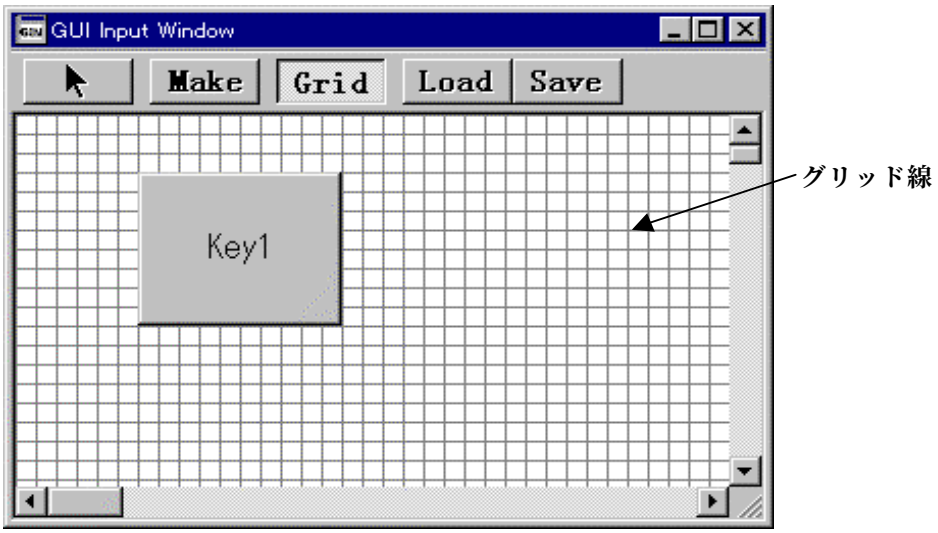
1. GUI 入力ウィンドウの[Option]→[Del]メニューを選択します。
2. 選択した後、マウスを GUI 入力ウィンドウの入力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、削除したいボタン上でマウスの左ボタンをクリックして下さい。

または、GUI 入力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して下さい。選択後、削除したいボタンをマウスの左ボタンでクリックして選択状態にし、Delete キーを入力して下さい。

5.7 グリッド線を表示するには

GUI 入力ウィンドウには、ボタンを配置することのできる位置にグリッド線を表示する機能があります。ボタンを配置する際に利用して下さい。

グリッド線を表示するには[Option]→[Display Grid Line]メニュー（あるいは Grid ボタン）を選択して下さい。選択すると以下のようなグリッド線が表示されます。



6 GUI 出力ウィンドウの設定

6.1 概要

GUI 出力ウィンドウでは、ユーザターゲットシステムの簡単な出力パネルをウィンドウ上で実現できます。

出力パネルには、以下のパーツが配置できます。

- ラベル（文字列）
  指定アドレスのメモリにある値が書き込まれた時、あるいは、ビットの 1/0 に応じてユーザが指定した文字列を表示／消去します。
- LED
  指定アドレスのメモリにある値が書き込まれた時、あるいは、ビットの 1/0 に応じて LED の点灯を行います。

配置できるパーツの最大数について

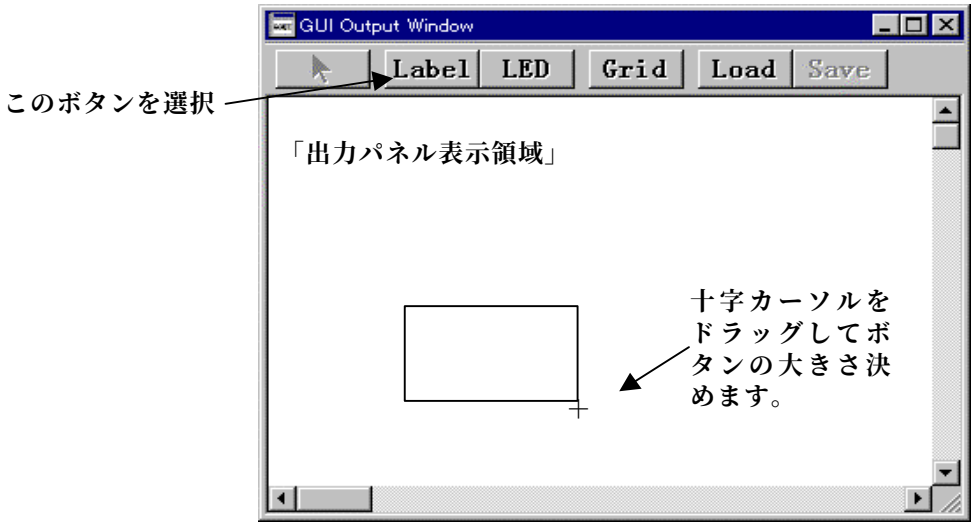
- 作成したパーツに設定できるアドレスは、すべてのパーツをあわせて 20 点までです。
  そのため、各パーツに設定したアドレスが全て異なる場合、配置できるパーツの最大数は 20 個です。
- 各パーツに設定したアドレスが 20 点以下の場合、配置できるパーツの最大数に、事実上制限はありません。

ここでは、GUI 出力ウィンドウのパーツの作成方法について説明します。

6.2 ラベルの作成方法

ラベルの作成は、以下の手順で行って下さい。

1. GUI 出力ウィンドウの[Option]→[Make Label]メニュー（あるいは Label ボタン）を選択します。
2. 選択した後、マウスを GUI 出力ウィンドウの出力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、ラベルを作成したい位置でマウスの左ボタンをクリックし、クリックしたまま作成したい大きさに広げ、左ボタンを離して下さい。

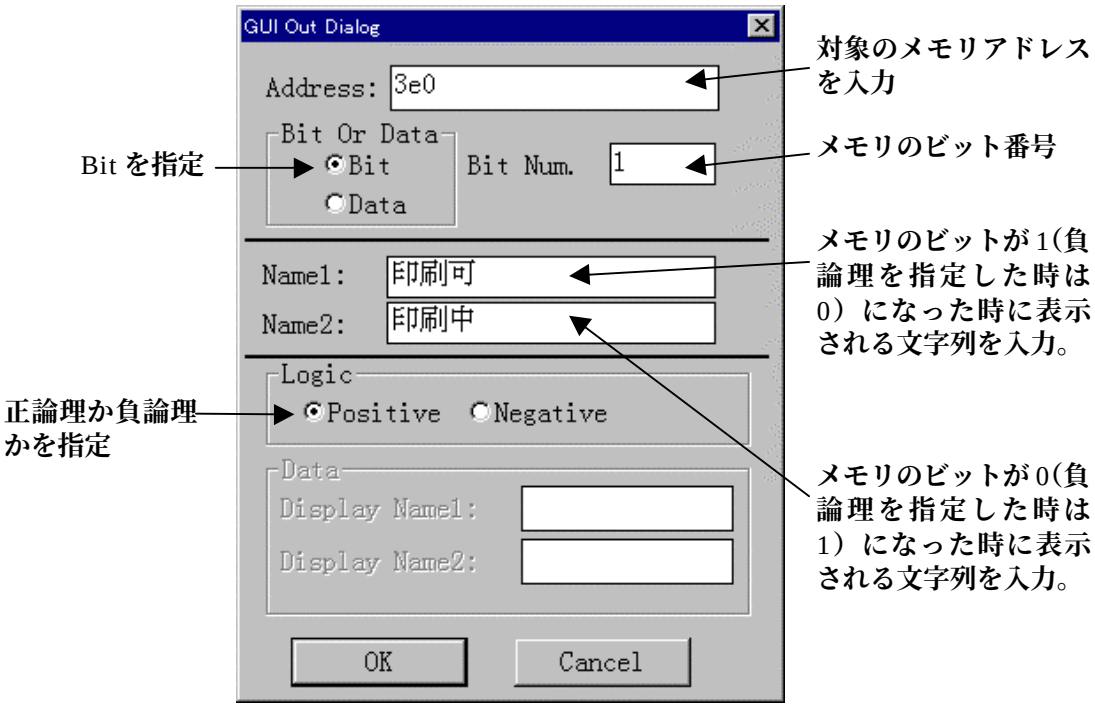


4. ラベルを表示するための枠が表示されますので、枠内の領域をダブルクリックして下さい。

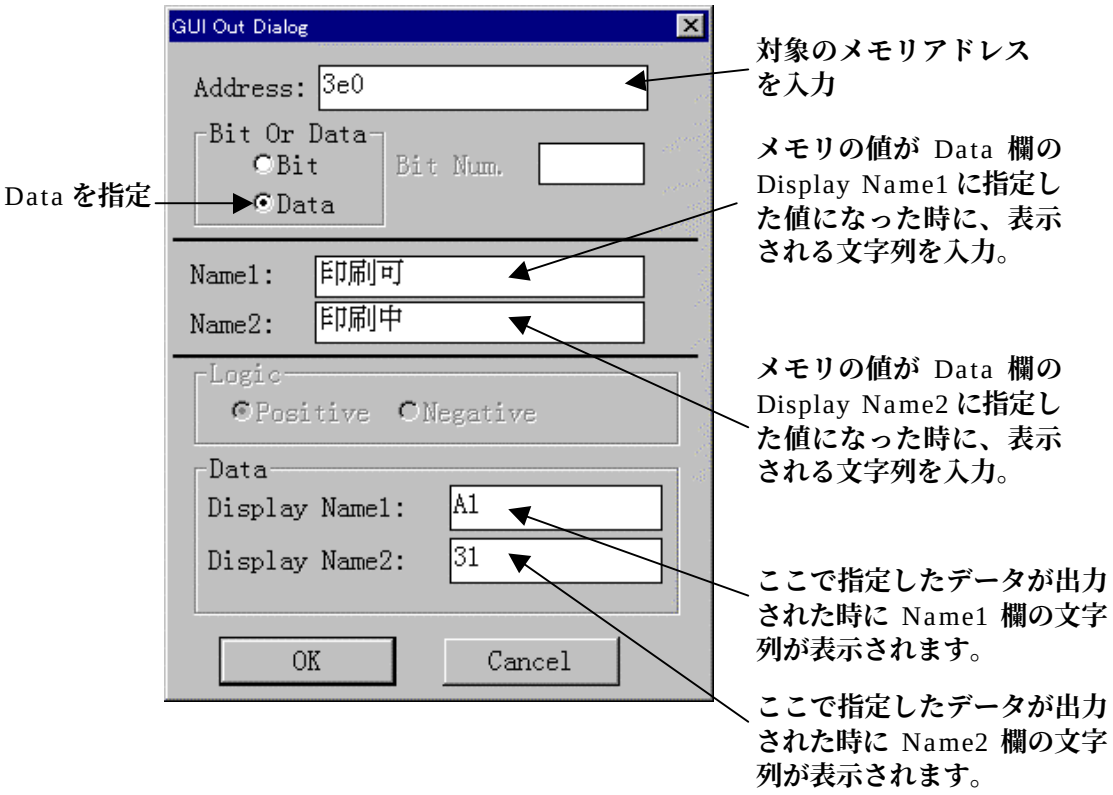
「注意」
（作成したラベルをダブルクリックする前に、ツールバーの↑ボタンが選択されていることを確認して下さい。選択されていない時は、↑ボタンを選択してから作成したボタンをダブルクリックしてください。）

5. 次のようなラベルの設定を行うためのダイアログがオープンします。
ここでは、次の２通りの出力の監視の指定が行えます。
●メモリのビットの 1／0 に応じてユーザが指定した文字列を表示／消去
●メモリにある値が書き込まれた時に、文字列を表示／消去

(1) メモリのビットの 1／0 に応じて文字列を表示／消去する場合



(2)メモリに指定した値が出力された時に、文字列を表示／消去する場合

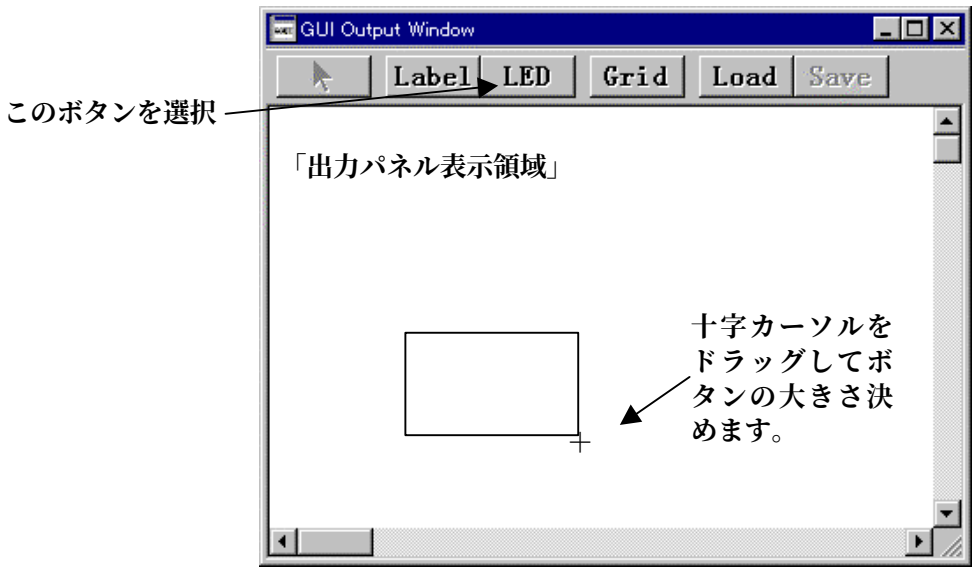


6. ダイアログの OK ボタンを押して下さい。これで、ラベルの作成と設定が完了します。
7. さらにラベルを作成する場合は、上記の１～6 の操作を繰り返し行って下さい。

6.3 LED の作成方法

LED の作成は、以下の手順で行って下さい。

1. GUI 出力ウィンドウの[Option]→[Make LED]メニュー（あるいは LED ボタン）を選択します。
2. 選択した後、マウスを GUI 出力ウィンドウの出力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、LED を作成したい位置でマウスの左ボタンをクリックし、クリックしたまま作成したい大きさに広げ、左ボタンを離して下さい。

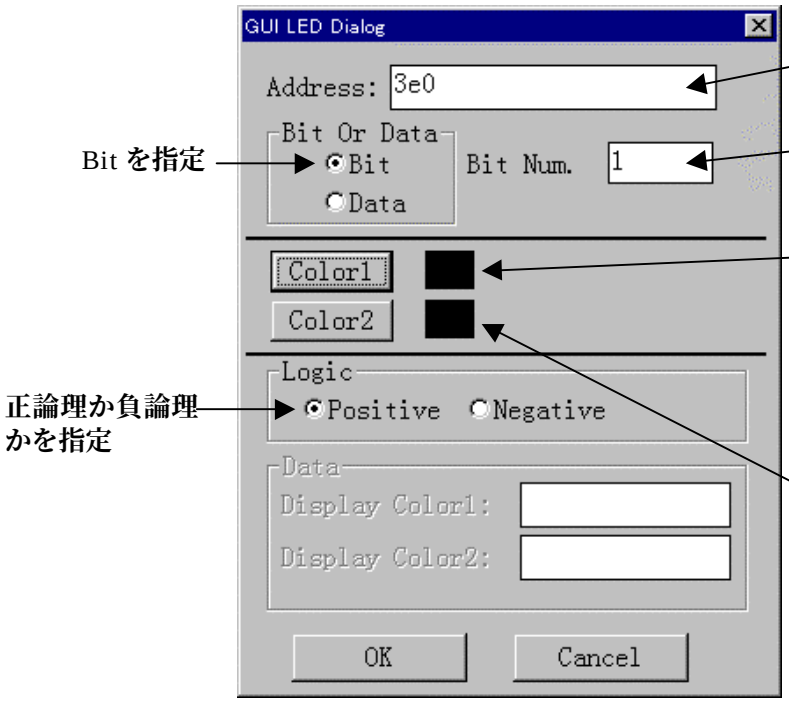


4. LED を表示するための枠が表示されので、枠内の領域をダブルクリックして下さい。

「注意」
（作成した LED をダブルクリックする前に、ツールバーの↑ボタンが選択されていることを確認して下さい。選択されていない時は、↑ボタンを選択してから作成したボタンをダブルクリックしてください。）

5. 次のような LED の設定を行うためのダイアログがオープンします。
ここでは、次の２通りの出力の監視の指定が行えます。
●メモリのビットの 1／0 に応じて LED を点灯
●メモリにある値が出力された時に、LED を点灯

(1)メモリのビットの 1／0 に応じて LED を点灯する場合



Bit を指定

対象のメモリアドレスを入力

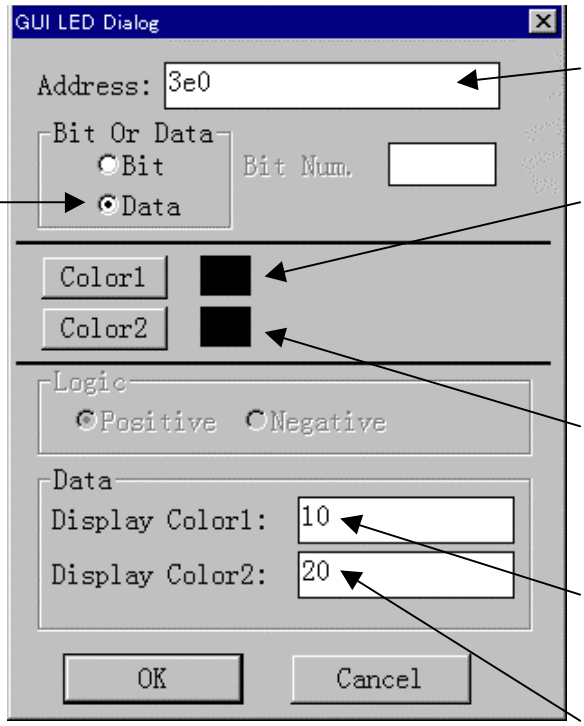
メモリのビット番号

メモリのビットが 1(負論理を指定した時は 0）になった時に表示される LED の色を選択。

正論理か負論理かを指定

メモリのビットが 0(負論理を指定した時は 1）になった時に表示される LED の色を選択。

(2)メモリに指定した値が出力された時に、文字列を表示／消去する場合



Data を指定

対象のメモリアドレスを入力

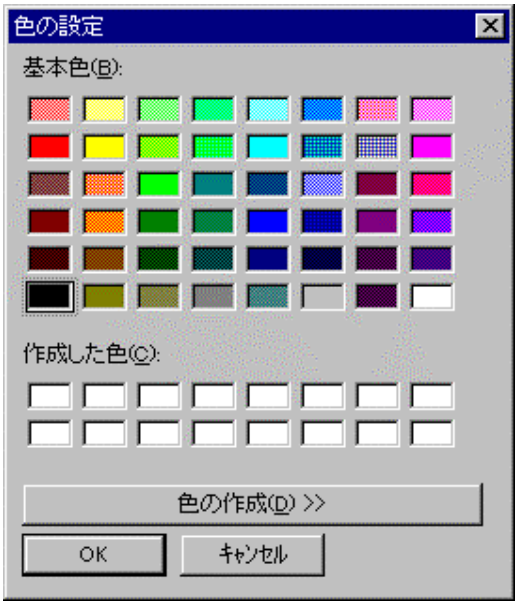
メモリの値が Data 欄の Display Color1 に指定した値になった時に、表示される LED の色を選択。

メモリの値が Data 欄の Display Color2 に指定した値になった時に、表示される LED の色を選択。

ここで指定したデータが出力された時に Color1 欄の色で LED が表示されます。

ここで指定したデータが出力された時に Color2 欄の色で LED が表示されます。

6. Color1、Color2 のボタンを押すと LED のカラーを選択するダイアログがオープンします。



ここで、LED に表示したい色を選択して、OK ボタンを押して下さい。

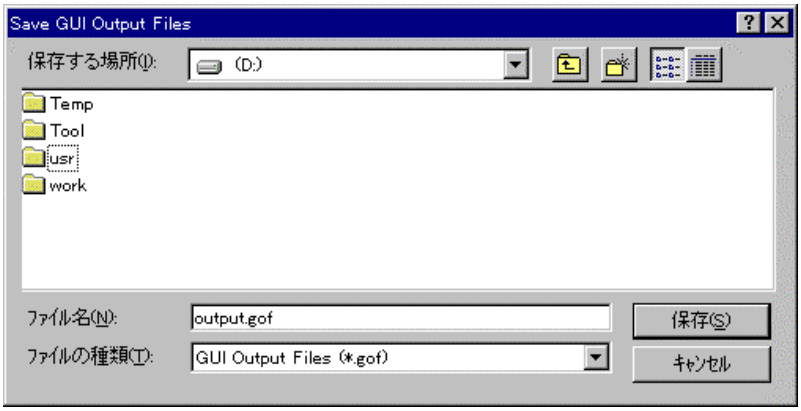
7. これで、LED の作成と設定が完了します。
8. さらに LED を作成する場合は、上記の１～７の操作を繰り返し行って下さい。

6.4 作成したパーツを保存するには

パーツの作成が終了した時に、作成したパーツのデータ（設定内容、レイアウト）をファイル（GUI 出力ファイル）に保存できます。保存した GUI 出力ファイルを[Option]→[Load]メニューで再度読み込むことにより、保存したパーツを再現することができます。

パーツデータの保存は、以下の手順で行って下さい。

GUI 出力ウィンドウの[Option]→[Save]メニュー（あるいは Save ボタン）を選択してください。選択すると以下のダイアログがオープンします。

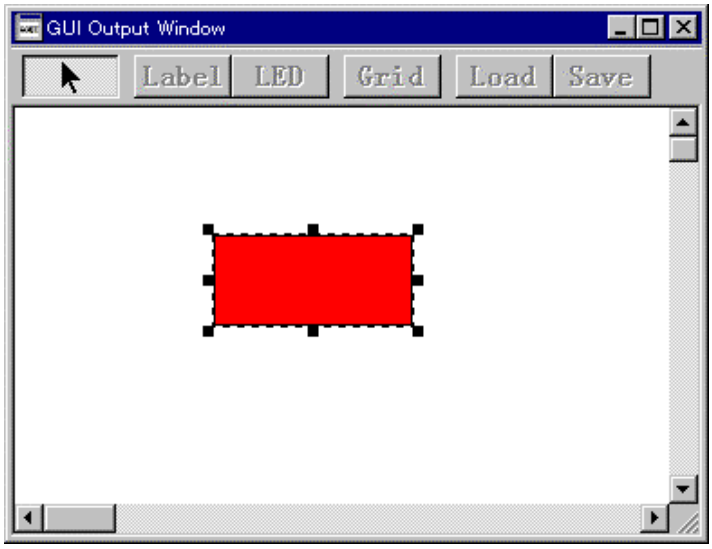


ここで、パーツデータを保存するディレクトリとファイル名を入力して下さい。
ファイル名を入力したら、保存ボタンを押して下さい。

6.5 パーツ作成後にパーツの配置、設定を変更するには

パーツを作成した後でも、パーツの配置や設定内容を変更することができます。

1. パーツの配置を変更するには
   GUI 出力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して下さい。
   選択後、配置を変更したいパーツをマウスの左ボタンでクリックして下さい。



   クリックすると上記のようにパーツが選択された状態になります。この時に、マウスの左ボタンで、パーツを移動したい位置までドラッグして下さい。
2. パーツの設定内容を変更するには
   上記 1 と同様に GUI 出力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して、設定内容を変更したいパーツをマウスの左ボタンでダブルクリックして下さい。

   パーツ作成の際に表示されたダイアログがオープンしますので、そのダイアログで設定を変更して下さい。

6.6 パーツをコピーするには

パーツのコピーは、以下の手順で行って下さい。

1. GUI 出力ウィンドウの[Option]→[Copy]メニューを選択します。
2. 選択した後、マウスを GUI 出力ウィンドウの出力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、コピーしたいパーツ上でマウスの左ボタンをクリックして下さい。
4. 次に、GUI 出力ウィンドウの[Option]→[Paste]メニューを選択すると、マウスの左ボタンで選択したパーツ上に、新しいパーツがコピーされます。

または、GUI 出力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して下さい。選択後、コピーしたいパーツをマウスの左ボタンでクリックして選択状態にし、Ctrl+C、Ctrl+V キーを入力して下さい。

6.7 パーツを削除するには

パーツの削除は、以下の手順で行って下さい。

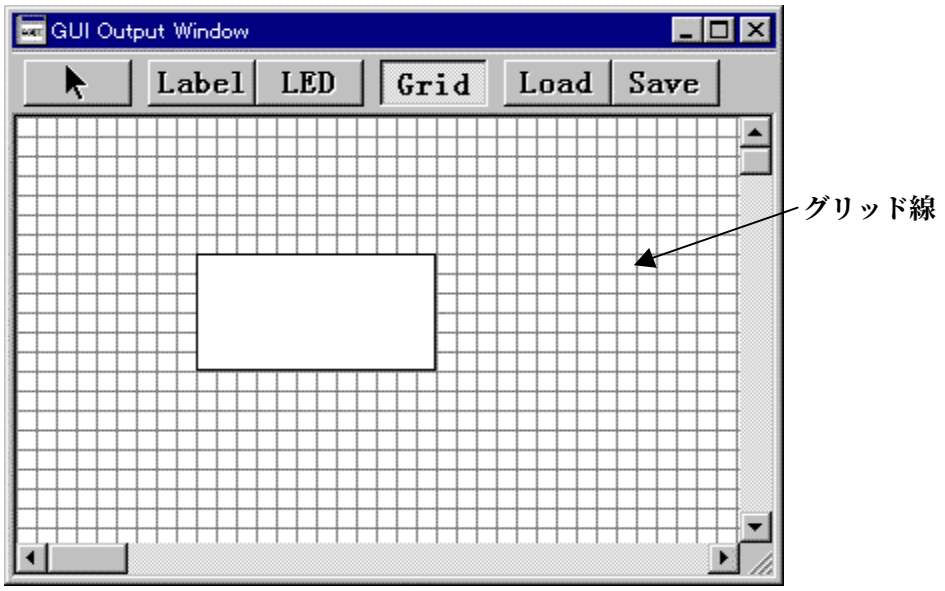
1. GUI 出力ウィンドウの[Option]→[Del]メニューを選択します。
2. 選択した後、マウスを GUI 出力ウィンドウの出力パネル表示領域に移動するとマウスカーソルの形状が＋（十字）になります。
3. この状態で、削除したいパーツ上でマウスの左ボタンをクリックして下さい。

または、GUI 出力ウィンドウの[Option]→[Set]メニュー（あるいは↑ボタン）を選択して下さい。選択後、削除したいパーツをマウスの左ボタンでクリックして選択状態にし、Delete キーを入力して下さい。

6.8 グリッド線を表示するには

GUI 出力ウィンドウには、パーツを配置することのできる位置にグリッド線を表示する機能があります。パーツを配置する際に利用して下さい。

グリッド線を表示するには[Option]→[Display Grid Line]メニュー（あるいは Grid ボタン）を選択して下さい。選択すると以下のようなグリッド線が表示されます。



7 I/O スクリプト機能

7.1 概要

仮想ポート入力や仮想割り込みの設定を、ファイルにスクリプト形式で記述できます。このスクリプトのことを I/O スクリプトといいます。また、I/O スクリプトを記述したファイルのことを I/O スクリプトファイルといいます。

I/O スクリプトを利用することにより、I/O ウィンドウでの仮想ポート入力や仮想割り込みの設定よりさらに柔軟な設定を行うことができます。例えば、I/O ウィンドウでは設定できない次のような設定が行えます。

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

その他にも様々な I/O の設定が可能です。

7.2 I/O スクリプトの記述方法

I/O スクリプトに記述する仮想ポート入力や仮想割り込み等の I/O の定義方法について定義例を用いて説明します。

I/O スクリプトを定義するには、プロシジャーを記述します。プロシジャーは{}で囲んで記述します。プロシジャーは、1 つのファイルに複数記述できます。

各プロシジャーには、仮想ポート入力や仮想割り込みの設定、タイミング等を記述します。

定義された複数の各プロシジャーは、プログラム実行中に並列に処理されます。但し、各プロシジャーが評価される順序は不定です。

I/O スクリプトファイルは、I/O ウィンドウの[Option]→[Load]メニュー（あるいは Load ボタン）で PD38SIMに登録します。I/O スクリプトファイルは複数登録できます。但し、登録できるプロシジャーの数は全て合わせて 20 プロシジャーまでです。

- 下記例のプロシジャー1 は、タイマ１のタイマモードの定義例です。
  タイマ１に指定されたサイクル数毎に、タイマ１割り込みを発生させる例です。

- 下記例のプロシジャー2 は、サイクルに同期した仮想ポート入力の定義例です。
  プログラムが 10000 サイクル実行された時に仮想ポート入力を行う例です。I/O ウィンドウでは、バイト単位の仮想ポート入力しかできませんが、I/O スクリプトではワード単位での仮想ポート入力が可能です。

```
; I/O スクリプトファイルの定義例                    → コメント

; プロシジャー 1 の定義（仮想割り込みの例）
{                                                  → プロシジャー 1 の開始
    while(1){                                      → while 文
            ;タイマ 1 カウンタ×プリスケーラ 12 カウンタ×分周
            waitc (([0x0021]+1) * ([0x0020]+1) * 16)
                                                   → タイマ 1 に設定されたサイクル数分
                                                        I/O スクリプトの実行をウェイトする
            int   0xffee                           → タイマ 1 の割り込み発生
    }
}                                                  → プロシジャー 1 の終了
```

```
;プロシジャー2 の定義（仮想ポート入力の例）
{                                                  →  プロシジャー2 の開始
    waitc 10000                                    →  10000 サイクル分 I/O スクリプトの実行を
                                                        ウェイト
    set [0x0] = 0x20                               →  0x0 番地に 0x20 を入力
    waitc 10000
    set [0x0] = 0x30                               →  0x0 番地に 0x30 を入力
}                                                  →  プロシジャー2 の終了
```

7.3 I/O スクリプトの構成要素

I/O スクリプトは以下の文が記述できます。

- **プロシジャー**
- **I/O スクリプト文**
- **判断文（if,else)**
  式の結果を判断して、実行する文を分岐します。
- **繰り返し文（while)と Break 文**
  式の結果を判断して、文を繰り返し実行します。
- **コメント文**
  I/O スクリプトにコメント（注釈）を記述できます。I/O スクリプト実行の際、コメント文は無視されます。

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

7.3.1 プロシジャー

プロシジャーは、I/O スクリプトの定義ブロックを指定します。プロシジャーは 1 つのファイルに複数指定できます。但し、プロシジャーの定義は 20 個までです（複数のファイルにプロシジャーを定義した場合、そのすべてのプロシジャーの定義数が 20 個までとなります）。以下に記述書式を示します。

```
{
    文
}

{
    文
}

以下、複数定義可能
      :
```

7.3.2 I/O スクリプト文

I/O スクリプト文には、以下の５つの文が指定できます。

(1)waiti 文

「書式」　waiti　機械語命令数

「機能」

指定された機械語命令数分、次の文の実行を抑止します。

機械語命令数には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。

例えば、次のような記述の場合、

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

set 文の実行は、機械語命令が 100 命令実行された後に行われます。

(2)waitc 文

「書式」　waitc　サイクル数

「機能」

指定されたサイクル数分、次の文の実行を抑止します。

サイクル数には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。

例えば、次のような記述の場合、

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

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

(3)int 文

「書式」　int　ベクタアドレス

「機能」

指定されたベクタアドレスの仮想割り込みを発生させます。

ベクタアドレスには、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。

例えば、次のような記述の場合、

```
int 0xffec
```

タイマ２（ベクタアドレス 0xffec）割り込みを発生させます。

(4)set 文

set 文には、3 通りの書式があります。

「書式１」　set　メモリアドレス = 入力値

「機能」

指定したメモリアドレスに入力値を入力します（仮想ポート入力を行います）。

メモリアドレスには左辺式が、入力値には右辺式が指定できます（左辺式、右辺式の仕様は、後述を参照下さい）。

例えば、次のような記述の場合、

```
set [0x2] = 0x1d
```

0x2 番地のメモリに 0x1d を入力します。

「書式２」　set　条件式　, メモリアドレス = 入力値１, 入力値２, ...

「機能」

条件式が成立するごとに、指定したメモリアドレスに入力値１、入力値２を順次入力します。

メモリアドレスには左辺式が、条件式, 入力値には右辺式が指定できます（左辺式、右辺式の仕様は、後述を参照下さい）。

例えば、次のような記述の場合、

```
set #isfetch:0xf000 , [0x3] = 0x10 , 0x20
;#isfetch はプログラムが指定番地を実行した時に真（成立）になります。
```

プログラムが 0xf000 番地を実行するごとに、0x3 番地のメモリにデータ 0x10,0x20 を順次入力します。

つまり、0xf000 を１回目に実行した時には、0x3 番地のメモリにデータ 0x10 が、2 回目に実行した時は 0x20 が入力されます。

「書式３」　set %マクロ変数 = 右辺式

指定したマクロ変数に右辺式を代入します（マクロ変数の仕様は、後述を参照下さい）。

例えば、次のようなマクロ変数の記述が行えます。

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

(5)pass 文

「書式」　　pass　条件式 , パス回数

「機能」

パス回数で指定された回数分、条件式が成立するまで、次の文の実行を抑止します。

条件式、パス回数には、右辺式が指定できます（右辺式の仕様は、後述を参照下さい）。

例えば、次のような記述の場合、

```
pass #isint:0xffec , 3
  ; #isint は指定された仮想割り込みが発生すれば真（成立）になります。
set [0x800] = 0x10
```

set 文の実行は、タイマ２割り込み（ベクタアドレス 0xffec）が３回発生した後に行われます。

7.3.3 判断文（if, else)

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

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

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

7.3.4 繰り返し文（while)とbreak 文

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

```
while ( 条件式 ) {
    文  または  break 文
}
```

- 条件式が真の場合、文を繰り返し実行します。条件式が偽の場合、ループから抜けます。
- while 文は、32 段までネストすることができます。
- while 文を強制的の抜ける場合は、break 文を使用します。while 文がネストしている場合は、最も内側のループから抜けます。
- 条件式には右辺式が使用できます。
- I/O スクリプトに記述する条件式は、unsigned 型で計算します。したがって、while 文の式で負の値を比較した場合の動作は不定になります。

7.3.5 コメント文

コメント文は、I/O スクリプトにコメント（注釈）を記述する場合に使用します。以下に記述書式を示します。

```
; 文字列
```

- セミコロン（';'）から文を記述します。
- ;以降、行の最後までをコメントとして扱います。
- コメント文の行は、I/O スクリプト実行時に無視されます。

7.4 右辺式の記述方法

I/O スクリプト文の機械語命令数、サイクル数、ベクタアドレス、優先順位、入力値、条件式、パス回数や if,while 文の式の指定に右辺式を記述することができます。以下に右辺式を使用した I/O スクリプト文の例を示します。

```
waitc LABEL
waiti [0x800] + 20
if( [0x1ff] == 0x30 )
while( #isfetch:0xf000 )
```

7.4.1 右辺式の構成要素

右辺式の構成要素として、以下のものが使用できます。

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

以下に、各構成要素について説明します。

7.4.2 定数

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

	16 進数	10 進数	2 進数※1
先頭	0x,0X	なし	%
例	0xAB24	1234	%10010

7.4.3 シンボル、ラベル

ターゲットプログラムで定義しているグローバルシンボル／グローバルラベルが使用できます。

- シンボル／ラベル名には、英数字、アンダスコア(‘ _ ’)、ピリオド(‘ . ’)、クエスチョンマーク(‘ ? ’)が使用可能です。ただし、先頭文字に数字は使用できません。
- シンボル／ラベル名は、255 文字まで記述できます。
- 大文字／小文字は区別します。
- アセンブラsra74 の構造化命令、擬似命令、マクロ命令、オペコード予約語は使用できません(.SECTION, .BYTE, switch, if など)。
- “ .. ”で始まる文字列は、シンボル／ラベル名には使用できません。

7.4.4 マクロ変数

マクロ変数は、変数名の先頭に' % 'を付加して使用します。

- パーセント文字（' % '）の後の変数名には、英数字と'_'が使用可能です。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- 変数名には、レジスタ名は使用できません。
- 変数名の大文字／小文字を区別します。
- マクロ変数は、32 個まで定義できます。一度定義したマクロ変数は、PD38SIMが終了するまで有効です。

7.4.5 メモリ変数

メモリの値を式中で利用する際に使用します。メモリ変数の書式を以下に示します。

[アドレス］.データサイズ

- アドレスには、式が記述できます（メモリ変数も指定可能）。
- データサイズは、以下のように指定します。

バイト長の場合	B または b
ワード（2 バイト）長の場合	W または w

- データサイズの指定を省略した場合、バイト長を指定したことになります。
  例１：8000_{16}番地のメモリ内容をバイトで参照する場合
  [0x8000].B　または　[0x8000]
  例２：8000_{16}番地のメモリ内容をワードで参照する場合
  [0x8000].w

7.4.6 文字定数

指定された文字または文字列を ASCII コードに変換し、定数として扱います。

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

7.4.7 演算子

式に記述可能な演算子を以下に示します。

- 演算子の優先度は、レベル１が最も高く、レベル１２が最も低くなります。
  優先順位が同じ場合は、式の左から順番に計算します。

演算子	意味	優先度
( )	括弧	レベル１
+, -, ~	単項正、単項負、単項論理否定	レベル２
*, /	二項乗算、二項除算	レベル３
+, -	二項加算、二項減算	レベル４
>>, <<	右シフト、左シフト	レベル５
<, <=, >, >=	二項比較	レベル６
==, !=	二項比較	レベル７
&	二項論理積	レベル８
^	二項排他的論理和	レベル９
|	二項論理和	レベル１０
&&	積結合	レベル１１
||	和結合	レベル１２

7.4.8 #isfetch, #isint, #isread, #iswrite

#isfetch,#isint,#isread,#iswrite 式は、 I/O スクリプト文や if 文、while 文の条件式に使用します。

(1) #isfetch 式

「書式」 #isfetch: アドレス

「機能」

指定されたアドレスにプログラムの PC 値が移った時に、式の値が真(1)になります。
それ以外は偽（0）となります。
例えば、次の if 文は、

```
if ( #isfetch:0xfc00)
```

プログラムのアドレス（PC 値）が 0xfc00 になった時に真(1)になります。

(2) #isint 式

「書式」 #isint: ベクタアドレス

「機能」

指定されたベクタアドレスの仮想割り込みが発生した直後に式の値が真(1)になります。
それ以外は偽(0)となります。
例えば、次の if 文は、

```
if ( #isint:0xffee)
```

この if 文が評価された時（評価される直前）に、ベクタアドレス 0xffee の仮想割り込みが発生していたら真(1)になります。

(3) #isread 式

「書式」 #isread: アドレス

「機能」

指定されたアドレスのメモリがリードアクセス（メモリの読み込み）された直後に式の値が真(1)となります。それ以外は偽(0)となります。
例えば、次の if 文は、

```
if ( #isread:0x800)
```

この if 文が評価された時（評価される直前）に、0x800 番地のメモリにリードアクセスがあった場合、真(1)となります。

(4) #iswrite 式

「書式」 #iswrite: アドレス

「機能」

指定されたアドレスのメモリがライトアクセス（メモリへの書き込み）された直後に式の値が真(1)となります。それ以外は偽(0)となります。

例えば、次の if 文は、

```
if ( #iswirte:0x800)
```

この if 文が評価された時（評価される直前）に、0x800 番地のメモリにライトアクセスがあった場合、真(1)となります。

7.5 左辺式の記述方法

左辺式は、I/O スクリプト文の set 文のメモリアドレス、マクロ変数に記述することができます。以下に左辺式を使用した I/O スクリプト文の例を示します。

```
set [0x2] = 0x1a
set %val = 10
```

7.5.1 左辺式の構成要素

左辺式の構成要素として、以下のものが使用できます。

- マクロ変数
- メモリ変数

以下に、各構成要素について説明します。

7.5.2 マクロ変数

マクロ変数は、変数名の先頭に' % 'を付加して使用します。

- マクロ変数名には、英数字と'_'が使用できます。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- マクロ変数に代入する式が扱える値の範囲は、0_{16} から $FFFFFFFF_{16}$ までの整数です。負の数を指定した場合は 2 の補数として扱います。

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

```
set %val = 0                    ;マクロ変数%val に 0 を代入します。
while(%val < 10){               ;%val が 10 になるまで while 文を繰り返します。
        waitc 10000
        int 0xffee
        set %val = %val + 1     ;%val の値に 1 を足します。
}
```

7.5.3 メモリ変数

メモリに値を書き込む際に使用します。メモリ変数の書式を以下に示します。

［アドレス］.データサイズ

- アドレスには、式が記述できます（メモリ変数は記述できません）。
- データサイズは、以下のように指定します。

バイト長の場合	B または b
ワード（2 バイト）長の場合	W または w

- データサイズの指定を省略した場合、バイト長を指定したことになります。
  例１：8000_{16}番地のメモリにバイトで書き込む場合
  `set [0x8000].B = 0x10　または　set [0x8000] = 0x10`
  例２：8000_{16}番地のメモリにワードで書き込む場合
  `set [0x8000].w = 0x1234`

8 カバッレッジ情報

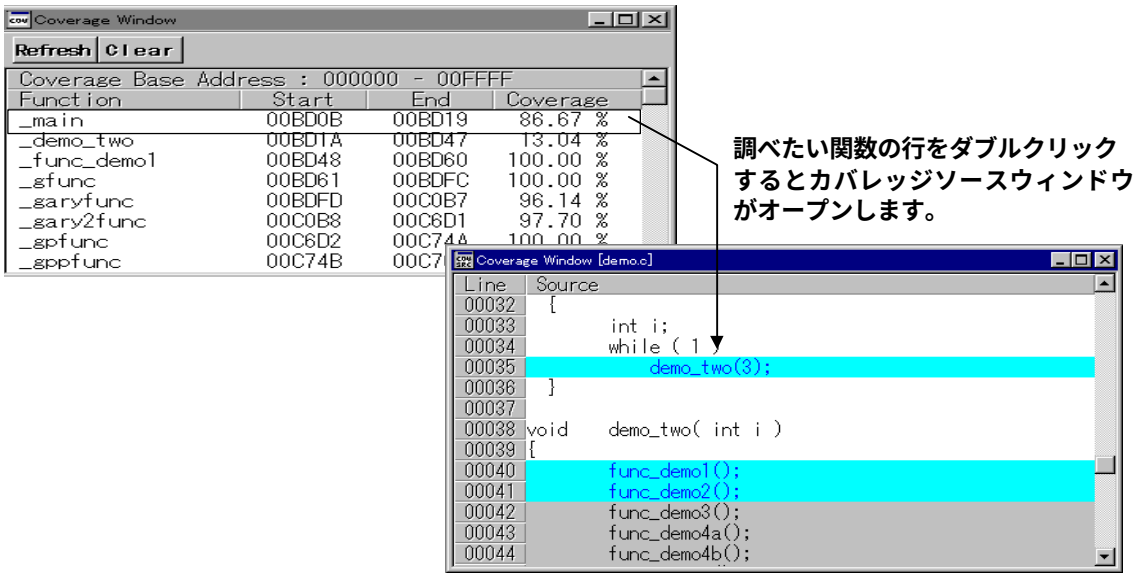
8.1 カバレッジを参照するには

ダウンロードされているC言語プログラムの各関数のカバレッジ（C0 カバレッジ）を参照するには、カバレッジウィンドウをオープンします。カバレッジウィンドウは、PD38SIMウィンドウのメニュー

［Optional Window］→［Coverage Window］

の選択でオープンします。

各関数の実行率が参照できるカバレッジウィンドウ



調べたい関数の行をダブルクリックするとカバレッジソースウィンドウがオープンします。

行単位に実行／非実行が参照できるカバレッジソースウィンドウ

8.2 カバレッジの表示を更新するには

GO,STEP 等で、ターゲットプログラムを実行した場合は、カバレッジウィンドウのカバレッジ表示領域が‘—’に変わります。表示の自動更新はありません。表示を更新したい時は、ツールバーの Refresh ボタン（またはメニュー [Option] → [Refresh]）を押して下さい。

またカバレッジソースウィンドウの表示は、プログラム停止時に自動更新します。

8.3 カバレッジを初期化するには

カバレッジ計測情報を初期化する場合は、ツールバーの Clear ボタン（またはメニュー [Option] → [Clear]）を押して下さい。カバレッジは、全て 0%になります。

8.4 カバレッジ計測情報を保存／読み込みするには

カバレッジ計測情報をファイルに保存したり、保存したファイルから読み込んで、続けて計測を開始する事ができます。

カバレッジ計測情報を保存するには・・・

カバレッジ情報を保存するには、カバレッジウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［File］→［Save...］

を選択してください。メニューを選択すると、カバレッジセーブダイアログが表示されます。

- ファイル名の指定には、パスを付加する事ができます。
- ファイルの拡張子を省略した場合は、デフォルトのファイル拡張子.cov を付加します。
- ファイル名に既存のファイル名を指定した場合は、上書きします。
- <Refer>ボタンをクリックすると、ファイルセレクションダイアログがオープンします。ファイルセレクションダイアログによるファイル名指定も可能です。
- <OK>ボタンをクリックすると、指定ファイルにセーブされます。

カバレッジ計測情報を読み込むには・・・

同機能の Save で書き込まれたカバレッジ計測結果の保存ファイルから、データを読み込むには、カバレッジウィンドウがアクティブな状態で、PD38SIMウィンドウのメニュー

［Option］→［File］→［Load...］

を選択してください。メニューを選択すると、ファイルセレクションダイアログが表示されます。

- ファイル名の指定には、パスを付加する事ができます。
- ファイルの拡張子を省略した場合は、デフォルトのファイル拡張子.cov が付加されます。
- ファイル名を入力後（またはファイル一覧でシングルクリック）、<開く>ボタンを押す、またはファイル一覧で、ファイル名をダブルクリックすると指定ファイルからロードされます。

9 カスタマイズ機能

9.1 カスタマイズ機能とは

カスタマイズ機能とは、**PD38SIM**にユーザ独自の機能を追加する機能です。**PD38SIM**にカスタムコマンドプログラムおよびカスタムウィンドウプログラムをロードすることにより、**PD38SIM**の機能を拡張することが可能です。

カスタムコマンドプログラム/カスタムウィンドウプログラムを作成するには・・・

カスタムコマンドプログラム/カスタムウィンドウプログラムは、**PD38SIM**に付属の**CB38SIM** (Custom Builder for **PD38SIM**)を用いて作成します。**CB38SIM**で作成したカスタムコマンドプログラムおよびカスタムウィンドウプログラムは、MACRO コマンドを用いて**PD38SIM**に登録することにより使用可能となります。カスタムコマンドプログラム/カスタムウィンドウプログラムの作成方法は、「**CB38SIM ユーザーズマニュアル**」を参照ください。

カスタムウィンドウプログラムを使用するには・・・

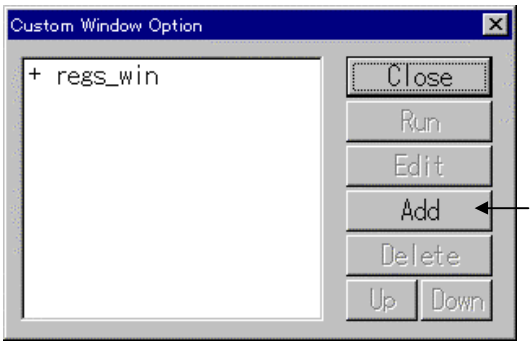
CB38SIMで作成したカスタムウィンドウプログラムを使用するには、以下に示す２つの方法があります。

1. カスタムウィンドウをメニューに登録し、メニューから起動する。
2. スクリプトウィンドウから MACRO コマンドで登録および起動する。

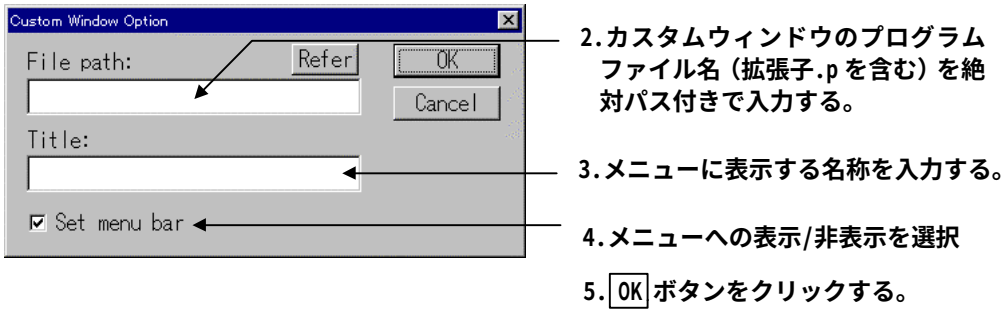
メニューへの登録方法および起動方法を以下に示します。MACRO コマンドでの登録および起動方法は、後述の「カスタムコマンドプログラム/カスタムウィンドウプログラムを使用するには・・・」を参照ください。

メニューへの登録方法

カスタムウィンドウをメニューに登録するには、メニュー
［Optional Window］ → [Custom Window] → [Option]
を選択し、以下のダイアログをオープンします。



カスタムウィンドウランチャー



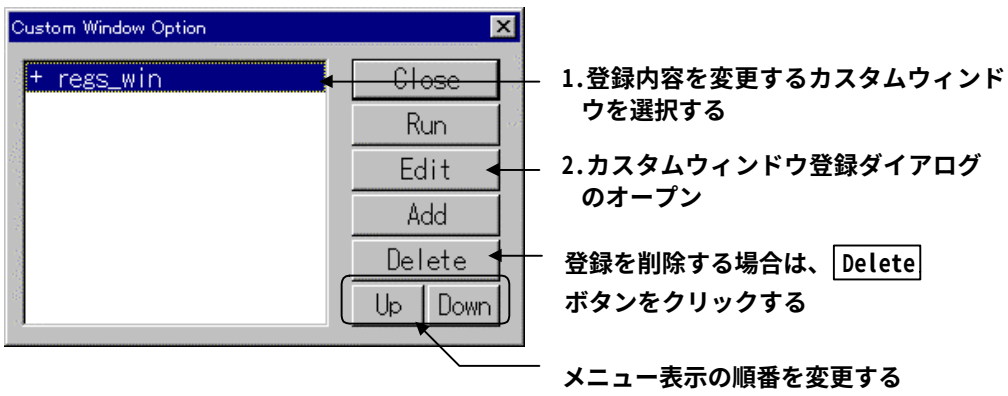
カスタムウィンドウ登録ダイアログ

注意

カスタムウィンドウをスクリプトウィンドウから起動する場合は、MACRO コマンドでカスタムウィンドウを登録してください。

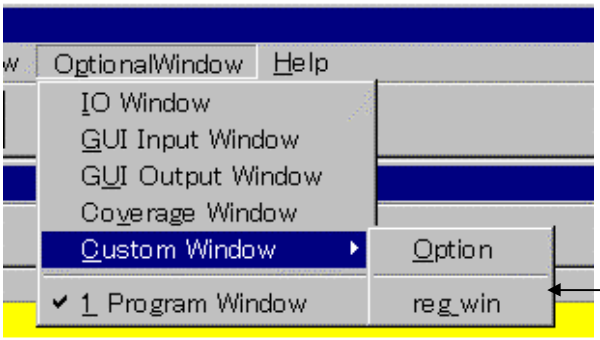
登録内容の変更

登録内容を変更するには、下記手順でカスタムウィンドウ登録ダイアログをオープンし、登録内容を変更してください。



カスタムウィンドウの起動

カスタムウィンドウを登録すると、カスタムウィンドウを起動するためのメニューが追加されます。



１．起動するカスタムウィンドウを選択する

メニューを非表示にしているカスタムウィンドウを起動する場合は、カスタムウィンドウランチャーで起動するカスタムウィンドウを選択して、Run ボタンをクリックします。

カスタムコマンドプログラム/カスタムウィンドウプログラムを使用するには・・・

CB38SIMで作成したカスタムコマンドプログラム/カスタムウィンドウプログラムは、スクリプトウィンドウから MACRO コマンドを使用して登録します。パラメータとしてカスタムコマンド/ウィンドウのプログラムファイル名（拡張子.p は省略する）を指定して**PD38SIM**に登録することにより使用可能となります。

(例)　　>MACRO custom<RET>

MACRO コマンドで登録した、カスタムコマンドプログラムの実行およびカスタムウィンドウプログラムのウィンドウオープンは、スクリプトコマンドの実行と同様の手続きで行えます。MACRO コマンドで登録したプログラムファイル名をコマンド名として実行します。

(例)　　>custom<RET>

MACRO コマンドで登録した、カスタムコマンドプログラムおよびカスタムウィンドウプログラムは、DELMACRO コマンドおよび DELMACROALL コマンドを用いて削除されるまで、削除されません。（**PD38SIM**を終了しても、保持されます。）

なお、**PD38SIM**は、カスタムコマンドプログラム/カスタムウィンドウプログラムを**PD38SIM**のカレントディレクトリ（スクリプトコマンド CD で設定したディレクトリ）から検索します。別ディレクトリに格納する場合は、後記の検索ディレクトリの説明に従って、カスタムコマンド/ウィンドウプログラムの格納ディレクトリを指定してください。

カスタムコマンドプログラム/カスタムウィンドウプログラムを削除するには・・・

MACRO コマンドで登録した、カスタムコマンドプログラムおよびカスタムウィンドウプログラムの削除は、DELMACRO コマンドに MACRO コマンドで登録したファイル名（拡張子は除く）を指定するか、DELMACROALL コマンドを用います。DELMACRO コマンドでは、指定されたカスタムコマンドプログラムまたはカスタムウィンドウプログラムのみを削除します。DELMACROALL コマンドでは、登録されている全カスタムコマンドプログラムおよびカスタムウィンドウプログラムを削除します。

(例)　>DELMACRO custom<RET>
　　>DELMACROALL<RET>

カスタムコマンドプログラム/カスタムウィンドウプログラムの検索ディレクトリを設定するには・・・

MACRO コマンドで登録した、カスタムコマンドプログラムおよびカスタムウィンドウプログラムは、実行される際に**PD38SIM**にロードされます。ロードするカスタムコマンドプログラムおよびカスタムウィンドウプログラムは、MACROPATH(MPATH)コマンドで設定されたディレクトリから検索されます。MACROPATH(MPATH)で指定可能なディレクトリは１つです。

(例)　>MACROPATH　c:¥usr¥project¥custom<RET>

リファレンス編

このページは白紙です。

1 スクリプトコマンド一覧

1.1 入力書式

- PD38SIMのスクリプトコマンドの入力形式
  1. コマンド名とパラメータの間には、1 文字以上の空白文字またはタブが必要です。
  2. コマンド名は、大小の英数字が使用できます。

- コマンド書式の記号説明

パラメータ	コマンド書式
XXXX	XXXX を入力する必要があります。
[XXXX]	入力を省略するか、又は XXXX が入力可能です。
{ X1 | X2 | X3 }	X1,X2,X3 の内、いずれか１つを入力する必要があります。
[{ X1 | X2 | X3 }]	入力を省略するか、X1,X2,X3 の内、いずれか１つが入力可能です。

1.2 コマンド一覧

以下にコマンド一覧を示します。網掛けがされているコマンドはランタイム実行可能です。
コマンド名欄の( )内はコマンド名の省略形です。
なお、各コマンドの詳細な説明は、PD38SIMのヘルプをご参照下さい。ヘルプを表示するには、
PD38SIMウィンドウのメニュー
［Help］→［Index］
を選択してください。

1.2.1 実行関連

コマンド名	書式	機能概要
Go(G)	Go ［開始アドレス］	ターゲットプログラムを実行します。
GoFree(GF)	GoFree［開始アドレス］	ターゲットプログラムをフリーラン実行します。
STOP	STOP	ターゲットプログラムを停止します。
STATUS	STATUS	ターゲットプログラムの実行状態を表示します。
Step(S)	Step［実行回数］	ソースレベルでステップ実行します。
StepInstruction(SI)	StepInstruction［実行回数］	機械語レベルでステップ実行します。
OverStep(O)	OverStep［実行回数］	ソースレベルでオーバステップ実行します。
OverStepInstruction (OI)	OverStepInstruction ［実行回数］	機械語レベルでオーバーステップ実行します。
RETurn(RET)	RETurn	ソースレベルでリターン実行します。
RETurnInstruction (RETI)	RETurnInstruction	機械語レベルでリターン実行します。
RESET	RESET	ターゲットプログラムをリセットします。

1.2.2 ファイル操作関連

コマンド名	書式	機能概要
Load(L)	Load ファイル名[.hex|.sym］	hex/sym ファイルをダウンロードします。
LoadHex(LH)	LoadHex ファイル名[.hex］	hex ファイルをダウンロードします。
LoadiE(LE)	LoadiE ファイル名[.695］	695 ファイルをダウンロードします。
LoadSymbol(LS)	LoadSymbol ファイル名[.695|.sym］	デバッグ情報のみをダウンロードします。
UploadHex(UH)	UploadHex 開始アドレス，終了アドレス，ファイル名	指定範囲のメモリ内容を hex ファイルに出力します。

1.2.3 レジスタ操作関連

コマンド名	書式	機能概要
Register(R)	① Register［ レジスタ名 ］ ② Register レジスタ名, 設定値	① 指定レジスタの値を参照します。 ② 指定レジスタに値を設定します。

1.2.4 メモリ操作関連

コマンド名	書式	機能概要
DumpByte(DB)	DumpByte ［ 開始アドレス［, 終了アドレス ］］	メモリ内容を 1 バイト単位で表示します。
SetMemoryByte(MB)	setMemoryByte アドレス［, データ［, ...］］	指定アドレスのメモリ内容を 1 バイト単位で変更します。終了するには、“.”を入力してください。
FillByte(FB)	FillByte 開始アドレス, 終了アドレス, データ	指定したアドレス範囲に指定したデータを 1 バイト単位で書き込みます。
MOVE	MOVE 開始アドレス, 終了アドレス, 転送先アドレス	指定範囲のメモリ内容を転送先アドレス以降に転送します。
MOVEWord(MOVEW)	MOVEWord 開始アドレス, 終了アドレス, 転送先アドレス	指定範囲のメモリ内容を転送先アドレス以降に 2 バイト単位で転送します。

1.2.5 RAM モニタ関連

コマンド名	書式	機能概要
RRAM	①RRAM ②RRAM　SET ,アドレス ③RRAM　CLEAR	①RAM モニタ領域を表示します。 ②RAM モニタ領域を設定します。 ③RAM モニタ領域のアクセス状態を初期化します。

1.2.6 アセンブル/逆アセンブル関連

コマンド名	書式	機能概要
Assemble(A)	Assemble [アドレス]	指定したアドレスから 1 行単位でアセンブルします。
DisAssemble(DA)	DisAssemble [開始アドレス[,終了アドレス[,m,x]]]	指定した範囲の逆アセンブル結果を表示します。
MODule(MOD)	MODule	全モジュール（オブジェクト名）を表示します。
SCOPE	①SCOPE ② SCOPE モジュール名	① 現在のスコープを表示します。 ② 指定モジュールにスコープを設定します。
SECtion(SEC)	SECTION	セクション情報を表示します。
BIT	① BIT ② BIT{GLOBAL｜G}[, ビットシンボル名[, 設定値]] ③ BIT{LOCAL　｜L}[, ビットシンボル名 [, 設定値]]	① 全ビットシンボルを表示します。 ② グローバルビットシンボルを表示／設定します。 ③ ローカルビットシンボルを表示／設定します。
SYMbol(SYM)	①SYMbol ② SYMbol{GLOBAL｜G}[, シンボル名] ③ SYMbol{LOCAL｜L}[, シンボル名]	① 全シンボルを表示します。 ② グローバルシンボルを表示します。 ③ ローカルシンボルを表示します。
EXPress(EXP)	EXPress アセンブラ式	指定したアセンブラ式の値を表示します。

1.2.7 ソフトウェアブレーク設定関連

コマンド名	書式	機能概要
SoftwareBreak(SB)	① SoftwareBreak ② SoftwareBreak アドレス	① 現在設定されているソフトウェアブレークポイントを表示します。 ② 指定したアドレスにソフトウェアブレークポイントを設定します。
SoftwareBreakClear(SBC)	SoftwareBreakClear アドレス	指定アドレスのソフトウェアブレークポイントを削除します。
SoftwareBreakClearAll (SBCA)	SoftwareBreakClearAll	全ソフトウェアブレークポイントを削除します。
SoftwareBreakDisable(SBD)	SoftwareBreakDisable アドレス	指定アドレスのソフトウェアブレークポイントを無効にします。
SoftwareBreakDisableAll (SBDA)	SoftwareBreakDisableAll	全ソフトウェアブレークポイントを無効にします。
SoftwareBreakEnable(SBE)	SoftwareBreakEnable アドレス	指定アドレスのソフトウェアブレークポイントを有効にします。
SoftwareBreakEnableAll (SBEA)	SoftwareBreakEnableAll	全ソフトウェアブレークポイントを有効にします。
BREAKAT	BREAKAT 行番号［, ソースファイル名 ］	指定した行番号にソフトウェアブレークポイントを設定します。
BREAKIN	BREAKIN 関数名［, モジュール名 ］	指定した関数の先頭にソフトウェアブレークポイントを設定します。

1.2.8 ハードウェアブレーク設定関連

コマンド名	書式	機能概要
HardwareBreak(HB)	① HardwareBreak ② HardwareBreak **アドレス**, { READ | WRITE | RW | FETCH } [, [通過回数] [, [BYTE] [, 比較データ [, 比較条件]]]]	① ハードウェアブレークポイントを参照します。 ② ハードウェアブレークポイントを設定します。 詳細は、211ページをご参照下さい。
HardwareBreakClear (HBC)	HardwareBreakClear アドレス	指定アドレスのハードウェアブレークポイントを削除します。
HardwareBreakClearAll (HBCA)	HardwareBreakClearAll	全ハードウェアブレークポイントを削除します。
BreakMode(BM)	① BreakMode ② BreakMode { ON | OFF }	① ハードウェアブレークのモードを参照します。 ② ハードウェアブレークのモードを設定します。

1.2.9 カバレッジ計測関連

コマンド名	書式	機能概要
CoVerage(CV)	① Coverage ② Coverage LOCAL［, 開始アドレス, 終了アドレス ］ ③ Coverage GLOBAL［, 開始アドレス, 終了アドレス ］ ④ Coverage TOTAL［, 開始アドレス, 終了アドレス ］ ⑤ Coverage FUNC ⑥ Coverage CLEAR ⑦ Coverage DISP, 表示開始アドレス	① カバレッジ計測結果の表示開始アドレスを参照します。 ② カバレッジ計測結果を 1 バイト単位で表示します。 ③ カバレッジ計測結果を 4 バイト単位で表示します。 ④ カバレッジ計測結果をパーセントで表示します。 ⑤ サブルーチン単位のカバレッジ計測結果をパーセントで表示します。 ⑥ カバレッジ計測用メモリを初期化します。 ⑦ カバレッジ計測結果の表示開始アドレスを設定します。 詳細は、213ページをご参照下さい。

1.2.10 スタック使用量計測関連

コマンド名	書式	機能概要
StackMonitor(SM)	① StackMonitor ② StackMonitor { ON | OFF }	① スタック使用量計測のモードを参照します。 ② スタック使用量を計測するか否かを設定します。

1.2.11 サイクル数計測関連

コマンド名	書式	機能概要
CYcle(CY)	① Cycle ② CYcle { ON | OFF }	① サイクル数計測のモードを参照します。 cycle OFF 時（デフォルト）、総実行サイクル数を表示します。 cycle ON 時、cycle ON 指定時からの累積実行サイクル数を表示します。 ② サイクル数を計測するか否かを設定します。

1.2.12 スクリプト/ログファイル関連

コマンド名	書式	機能概要
SCRIPT	SCRIPT ファイル名	スクリプトファイルをオープンします。
EXIT	EXIT	スクリプトファイルをクローズします。
WAIT	WAIT [BREAK]	ターゲットが停止するまでコマンド入力を待機します。
PAUSE	PAUSE “メッセージ”	指定したメッセージを Pause ダイアログに表示し、ユーザーのボタン入力待ちになります。
SLEEP	SLEEP 秒数	指定した秒数だけコマンド入力を待機します。
LOGON	LOGON [ファイル名[.ファイル属性]]	ログファイルをオープンします。
LOGOFF	LOGGOFF	ログファイルをクローズします。

1.2.13 プログラムウィンドウ制御関連

コマンド名	書式	機能概要
PATH	PATH [サーチパス [;サーチパス ;....]]	ソースファイルが存在するパスを設定します。
FILE	① FILE ② FILE ソースファイル名	① ソースファイルの一覧を表示します。 ② 指定したソースファイルを表示します。

1.2.14 C言語デバッグ関連

コマンド名	書式	機能概要
PRINT	PRINT 変数式	指定したC言語変数式の値を参照します。
SET	SET 変数式, データ	指定したC言語変数式に指定データを設定します。

1.2.15 カスタムコマンド/カスタムウィンドウ関連

コマンド名	書式	機能概要
MACRO	① MACRO ② MACRO カスタムコマンド/カスタムウィンドウのプログラム名	① 登録されているカスタムコマンド/カスタムウィンドウの一覧表示を行います。 ② カスタムコマンド/カスタムウィンドウプログラムを登録します。
DELMACRO	DELMACLRO カスタムコマンド/カスタムウィンドウのプログラム名	登録されているカスタムコマンド/カスタムウィンドウの削除します。
DELMACROALL	DELMACROALL	登録されているカスタムコマンド/カスタムウィンドウの全てを削除します。
MACROPATH(MPATH)	① MACROPATH ② MACROPATH ディレクトリパス名	① 設定されているディレクトリを表示します。 ② カスタムコマンド/カスタムウィンドウプログラムが格納されているディレクトリを設定します。

1.2.16 ユーティリティ関連

コマンド名	書式	機能概要
RADIX	①RADIX ②RADIX { 2 | 8 | 10 | 16 }	① 定数入力の既定値を参照します。 ② 定数入力の既定値を設定します。
ALIAS	①ALIAS ② ALIAS 別名, コマンド名	① コマンドの別名定義を参照します。 ② コマンドに別名を定義します。
UNALIAS	UNALIAS 別名	指定した別名定義を削除します。
UNALIASALL	UNALIASALL	すべての別名定義を削除します。
Help(H)	① Help ② Help コマンド名	① スクリプトコマンド一覧を表示します。 ② 指定したコマンドの説明を表示します。
VERsion(VER)	VERsion	PD38SIMのバージョンを表示します。
DATE	DATE	現在の日時(年月日、曜日、時間)を表示します。
ECHO	ECHO “メッセージ”	指定したパラメータを表示します。
QUIT	QUIT	PD38SIMを終了します。
CD	① CD ② CD ディレクトリ名	① 現在のカレントディレクトリを参照します。 ② カレントディレクトリを設定します。

1.2.17 スクリプトコマンドの補足説明

HardwareBreak(HB)

ハードウェアブレークの設定/参照

［入力書式］

（書式 1） HardwareBreak
（書式 2） HardwareBreak アドレス, FETCH [, 通過回数]
（書式 3） HardwareBreak アドレス, アクセス条件 [, 通過回数]
（書式 4） HardwareBreak アドレス, アクセス条件, [通過回数], サイズ
（書式 5） HardwareBreak アドレス, アクセス条件, [通過回数], [サイズ]
, 比較データ [, 比較条件]

アクセス条件、サイズ、比較条件に指定できる値を以下に示します。

アクセス条件	READ, R, WRITE, W, RW
サイズ	BYTE, B
比較条件	<, >, <=, >=, !=, ==

［機能］

- ハードウェアブレークとは、メモリに対する読み込み／書き込みがあったとき、命令フェッチを検出したときにブレークする機能です。
- PD38SIMでは、ハードウェアブレークポイントとしてアドレス 64 点が使用できます。
- ハードウェアブレークを使用するには、以下に示すように BreakMode コマンドでハードウェアブレークを有効にしてください。

BreakMode ON

ハードウェアブレークの設定内容を参照するには・・・

書式 1 を使用します。ハードウェアブレークの設定内容を表示するには、以下のように入力します。

HardwareBreak

指定アドレスを実行したときに止めるには・・・

書式２を使用します。

- 8000_{16} 番地の命令を実行したときに止めるには、以下のように入力します。

HardwareBreak 8000, FETCH

- 8000_{16} 番地の命令を 10 回実行したときに止めるには、以下のように入力します。

HardwareBreak 8000, FETCH, 10

通過回数は省略すると 1 を指定したことになります（他の書式も同様です)。

指定アドレスにデータをアクセスしたとき止めるには・・・

書式３～５を使用します。以下に各書式の相違点を示します。

書式３	アクセスするデータの値に関係なく止めたいときに使用します。
書式４	アクセスするデータのサイズによって止めるか否かを指定したい場合に使用します。
書式５	アクセスするデータの値によって止めるか否かを指定したい場合に使用します。

- 30_{16}番地にデータを書き込んだときに止める場合は、以下のように入力します（書式３）。

  HardwareBreak 30, WRITE

  WRITE は W と省略できます。データを読み込んだときに止める場合は、READ または R を指定します。読み込みまたは書き込んだときに止める場合は、RW を指定します。

- 30_{16}番地に１バイトのデータを書き込んだときに止める場合は、以下のように入力します（書式４）。

  HardwareBreak 30, WRITE, , BYTE

  BYTE は B と省略できます。
  サイズは省略すると BYTE を指定したことになります。

- 30_{16} 番地に 50_{16} の値を書き込んだときに止める場合は、以下のように入力します（書式５）。

  HardwareBreak 30, WRITE, , , 50

  比較データを指定して、比較条件を省略すると==を指定したことになります（他の書式も同様です)。
- 30_{16} 番地に 12_{16} の値を書き込んだときに止める場合は、以下のように入力します（書式５）。

  HardwareBreak 30, WRITE, , BYTE, 12

- 30_{16}番地に 50_{16}以上の値を書き込んだときに止める場合は、以下のように入力します（書式５）。

  HardwareBreak 30, WRITE, , , 50, >=

CoVerage(CV)

カバレッジの計測

実行時にアクセスしたアドレスを参照するには、カバレッジ計測機能を使用します。カバレッジ計測機能は、スクリプトコマンドで実現しています。コマンド名は CoVerage（短縮名 CV）です。スクリプトコマンドの使用方法については、本マニュアル 基本操作方法編の項目「7.1 スクリプトコマンドを実行するには」をご参照下さい。

［入力書式］

（書式 1） CoVerage
（書式 2） CoVerage { LOCAL | GLOBAL | TOTAL } [, 開始アドレス, 終了アドレス]
（書式 3） CoVerage FUNC
（書式 4） CoVerage CLEAR
（書式 5） CoVerage DISP, カバレッジ結果の表示開始アドレス

［機能］

- カバレッジとは、ターゲットプログラムが実行（アクセス）したアドレスを記録する機能です（C0 カバレッジ）。

カバレッジ計測を行い、アクセス状態を格納するには・・・

ターゲットプログラムを実行します。実行の方法は、本マニュアル 基本操作方法編の項目「2.1 実行・停止するには」をご参照下さい。

アクセスしたアドレスを参照するには・・・

書式２～３を使用します。参照できるアドレスの範囲は、0000_{16} ～ $FFFF_{16}$ までです。

- カバレッジ計測結果をアドレス 8000_{16} ～ $8FFF_{16}$ まで 1 バイト単位で参照するには、以下のように入力します（書式２）。

  CoVerage LOCAL, 8000, 8FFF

- カバレッジ計測結果をアドレス 8000_{16} ～ $8FFF_{16}$ まで 4 バイト単位で参照するには、以下のように入力します（書式２）。

  CoVerage GLOBAL, 8000, 8FFF

- カバレッジ計測結果をアドレス 8000_{16} ～ $8FFF_{16}$ までパーセント表示で参照するには、以下のように入力します（書式２）。

  CoVerage TOTAL, 8000, 8FFF

- カバレッジ計測結果をサブルーチン単位のパーセント表示で参照するには、以下のように入力します（書式３）。

  CoVerage FUNC

前回のカバレッジ計測結果を初期化するには・・・

書式４を使用します。

- カバレッジ計測結果を初期化するには、以下のように入力します。

  CoVerage CLEAR

カバレッジ計測結果の表示開始アドレスを参照／設定するには・・・

書式１／書式５を使用します。カバレッジ計測結果の参照で開始／終了アドレス省略した場合は、カバレッジ計測結果の表示開始アドレスから 1K バイト分の計測結果を表示します。

- カバレッジ計測結果の表示開始アドレスを参照するには、以下のように入力します。

  CoVerage

- カバレッジ計測結果の表示開始アドレスを $C000_{16}$ にするには、以下のように入力します。

  CoVerage DISP, C000

  カバレッジ計測結果の表示開始アドレスを $C000_{16}$ と設定した後に、開始／終了アドレスを省略してカバレッジ計測結果を参照すると、表示領域は $C000_{16}$ ～ $C400_{16}$ になります。

2 スクリプトファイルの記述方法

PD38SIMでは、スクリプトウィンドウでスクリプトファイルを実行することができます。スクリプトファイルは、スクリプトコマンドを自動実行するために、その制御などを記述したファイルです。

2.1 スクリプトファイルの構成要素

スクリプトファイルに以下の文が記述できます。

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

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

2.1.1 スクリプトコマンド

スクリプトウィンドウで入力するコマンドを、そのまま記述することができます。またスクリプトファイルからスクリプトファイルを呼び出すこともできます（ネストは５段まで）。

2.1.2 代入文

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

```
%マクロ変数名 = 式
```

- マクロ変数名には、英数字と'_'が使用できます。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- マクロ変数に代入する式が扱える値の範囲は、0_{16} から $FFFFFFFF_{16}$ までの整数です。負の数を指定した場合は 2 の補数として扱います。
- マクロ変数は、式の中で使用することができます。
- マクロ変数は、先頭に'%'を付加して使用します。

2.1.3 判断文（if,endi,else)

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

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

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

2.1.4 繰り返し文（while,endw)とbreak 文

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

```
while ( 式 )
    文
endw
```

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

2.1.5 コメント文

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

```
;文字列
```

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

注意事項

- スクリプトコマンドのコメントとして同一行に記述することはできません。
- スクリプトファイルのネストは５段までです。
- if 文と while 文のネストはそれぞれ 32 段までです。
- 一つのスクリプトファイルで if と endi 文、while と endw が対になっていなければいけません。
- スクリプトファイルに記述する式は、unsigned 型で計算します。したがって、if 文、while 文の式で負の値を比較した場合の動作は不定になります。
- １行に記述できる文字数は、4096 文字までです。これを越える行を実行した場合、エラーになります。
- 不適当な記述のあるスクリプトファイルを自動実行した場合（スクリプトウィンドウでスクリプトファイルをオープン後メニュー［Option］→［Script］→［Run］（またはスクリプトウィンドウの〈Run〉ボタン）を選択した場合）、スクリプト行自身が読み込めない場合を除いて、エラー検出後もスクリプトファイルの終わりまで実行処理は続けられます。
  ただしこの場合、エラー検出後の動作は不定であり、したがってエラー検出後の実行結果は信頼性がありません。

2.2 式の記述方法

PD38SIMでは、アドレス、データ、通過回数などの指定に式を記述することができます。以下に式を使用したコマンド例を示します。

```
>DB TABLE1
>DB TABLE1+20
```

2.2.1 式の構成要素

式の構成要素として、以下のものが使用できます。

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

以下に、各構成要素について説明します。

2.2.2 定数

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

	16 進数	10 進数	8 進数	2 進数※1
先頭	0x,0X	@	なし	%
末尾	h,H	なし	o,O	b,B
例	0xAB24 AB24h	@1234	1234o	%10010 10010b

※1　基数の既定値が 16 進数のときは、' % 'のみ指定可能

- 既定値と同じ基数で入力する場合は、基数を示す記号は省略可能です（2 進数は除く）
- 基数の既定値は、RADIX コマンドで設定します。ただし、以下のデータに関する入力を行う場合は、RADIX コマンドの設定に関係なく、基数は固定です。

種別	基数
アドレス	16 進
行番号 実行回数 通過回数 限定回数	10 進

2.2.3 シンボル、ラベル

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

- シンボル／ラベル名には、英数字、アンダスコア(‘ _ ’)、ピリオド(‘ . ’)、クエスチョンマーク(‘ ? ’)が使用可能です。ただし、先頭文字に数字は使用できません。
- シンボル／ラベル名は、255 文字まで記述できます。
- 大文字／小文字は区別します。
- アセンブラsra74の構造化命令、擬似命令、マクロ命令、オペコード予約語は使用できません(.SECTION, .BYTE, switch, if など)。
- “ .. ”で始まる文字列は、シンボル／ラベル名には使用できません。

［参照１］ローカルラベルシンボルとスコープ

PD38SIMでは、プログラムの全領域から参照可能なグローバルラベルシンボルと、宣言したファイル内でのみ参照可能なローカルラベルシンボルの 2 種類をサポートしています。

ローカルラベルシンボルの有効範囲をスコープといいます。スコープの単位は、オブジェクトファイル（”.r74”ファイル）です。PD38SIMでは、下記の場合に応じて、スコープを切り替えます。

- コマンド入力時
  プログラムカウンタが示すアドレスを含むオブジェクトファイルが、現在のスコープとなります。また SCOPE コマンドでスコープを設定した場合、設定したスコープが有効になります。

- コマンド実行中
  コマンドが扱うプログラムアドレスによって現在のスコープを自動的に切り替えます。

［参照２］ラベル／シンボルの優先順位

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

- アドレス値を変換する場合
  1. ローカルラベル
  2. グローバルラベル
  3. ローカルシンボル
  4. グローバルシンボル
  5. スコープ範囲外のローカルラベル
  6. スコープ範囲外のローカルシンボル

- データ値を変換する場合
  1. ローカルシンボル
  2. グローバルシンボル
  3. ローカルラベル
  4. グローバルラベル
  5. スコープ範囲外のローカルシンボル
  6. スコープ範囲外のローカルラベル

- ビット値を変換する場合
  1. ローカルビットシンボル
  2. グローバルビットシンボル
  3. スコープ範囲外のローカルビットシンボル

2.2.4 マクロ変数

マクロ変数は、スクリプトファイル中の代入文で定義します。詳細については、本マニュアル リファレンス編の項目「2.1 スクリプトファイルの構成要素」をご参照下さい。
マクロ変数は、変数名の先頭に'％'を付加して使用します。

- パーセント文字（'％'）の後の変数名には、英数字と'_'が使用可能です。ただし、マクロ変数名の先頭には、数字を記述することはできません。
- 変数名には、レジスタ名は使用できません。
- 変数名の大文字／小文字を区別します。
- マクロ変数は、32 個まで定義できます。一度定義したマクロ変数は、PD38SIMが終了するまで有効です。

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

2.2.5 レジスタ変数

レジスタの値を式中で利用する場合に使用します。レジスタ変数は、レジスタ名の前に'_'を付加します。以下に記述形式を示します。

_レジスタ名

使用できるレジスタ名を以下に示します。

A， X， Y， S， PC， F

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

2.2.6 メモリ変数

メモリの値を式中で利用する際に使用します。メモリ変数の書式を以下に示します。

[アドレス］.データサイズ

- アドレスには、式が記述できます（メモリ変数も指定可能）。
- データサイズは、以下のように指定します。

バイト長の場合	B または b
ワード（2 バイト）長の場合	W または w

- データサイズの指定を省略した場合、ワード長を指定したことになります。
  例１：8000_{16} 番地のメモリ内容をバイトで参照する場合
  [8000h].B
  例２：8000_{16} 番地のメモリ内容をワードで参照する場合
  [8000h].w

2.2.7 行番号

ソースファイルの行番号です。行番号の書式を以下に示します。

#行番号
#行番号."ソースファイル名"

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

2.2.8 文字定数

指定された文字または文字列を ASCII コードに変換し、定数として扱います。

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

2.2.9 演算子

式に記述可能な演算子を以下に示します。

- 演算子の優先度は、レベル１が最も高く、レベル８が最も低くなります。
  優先順位が同じ場合は、式の左から順番に計算します。

演算子	意味	優先度
( )	括弧	レベル１
+, =, ~	単項正、単項負、単項論理否定	レベル２
*, /	二項乗算、二項除算	レベル３
+, -	二項加算、二項減算	レベル４
>>, <<	右シフト、左シフト	レベル５
&	二項論理積	レベル６
|, ^	二項論理和、二項排他的論理和	レベル７
<, <=, >, >=, ==, !=	二項比較	レベル８

3 C言語式について

3.1 C言語式の記述方法

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

字句 (トークン)	例
即値	10, 0x0a, 012
四則演算子	+, -, *, /
ポインタ	*, **,...
参照	&
符号反転	-
コンマによるメンバ参照	Struct.Member
矢印によるメンバ参照	Struct->Member
括弧	(,)
配列	Array[2], DArray[2][3],...
基本型へのキャスト	(int), (char*), (unsigned long *),...
typedef された型へのキャスト	(DWORD), (ENUM),...
変数名および関数名	var, i, j, func,...
文字定数	'A', 'b',...
文字列リテラル	"abcdef", "I am a boy.",...

3.1.1 即値

即値としては、16 進数、10 進数、および 8 進数が使用できます。0x で始まれば 16 進数、0 で始まれば 8 進数として認識します。それ以外の数値は、10 進数として認識します。

注意事項

- 即値のみの C ウォッチポイントへの登録はできません。
- 即値は、C ウォッチポイントを指定する C 言語式の中に用いる場合、および代入する値を指定する場合にのみ有効です。

3.1.2 四則演算子

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

（*），（/） ＞ （+），（-）

注意事項

- 浮動小数点に対する四則計算は、現在サポートしておりません。

3.1.3 ポインタ

ポインタは、 * で表され、ポインタのポインタ ** 、ポインタのポインタのポインタ *** 、・・・が使用できます。
「*変数名」、「**変数名」、・・・という記述で使用します。

注意事項

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

3.1.4 参照

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

3.1.5 符号反転

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

注意事項

- 浮動小数点に対する符号反転は、現在サポートしておりません。

3.1.6 コンマによるメンバ参照

コンマによる構造体、共用体のメンバ参照は、「変数名.メンバ名」のみが使用できます。
（例）

```
struct S{
        int     member1;
        char    member2;
};

struct S        STRUCT;
struct S        *STRUCT_P;
```

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

3.1.7 矢印によるメンバ参照

矢印による構造体、共用体のメンバ参照は、「変数名->メンバ名」のみが使用できます。
（例）

```
struct S{
        int     member1;
        char    member2;
};

struct S        STRUCT;
struct S        *STRUCT_P;
```

この場合、(&STRUCT)->member1、STRUCT_P->member2は、正しくメンバを参照することができます。

3.1.8 括弧

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

3.1.9 配列

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

3.1.10 基本型へのキャスト

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

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

注意事項

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

3.1.11 typedef された型へのキャスト

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

注意事項

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

3.1.12 変数名および関数名

変数名および関数名は、C の規約通りアルファベットで始まる文字列が使用できます。最大文字数は、255 文字です。

3.1.13 文字定数

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

注意事項

- 文字定数のみの C ウォッチポイントへの登録はできません。
- C ウォッチポイントを指定する C 言語式の中に用いる場合、および代入する値を指定する場合にのみ有効です（文字定数は即値と同じ扱いになります）。

3.1.14 文字列リテラル

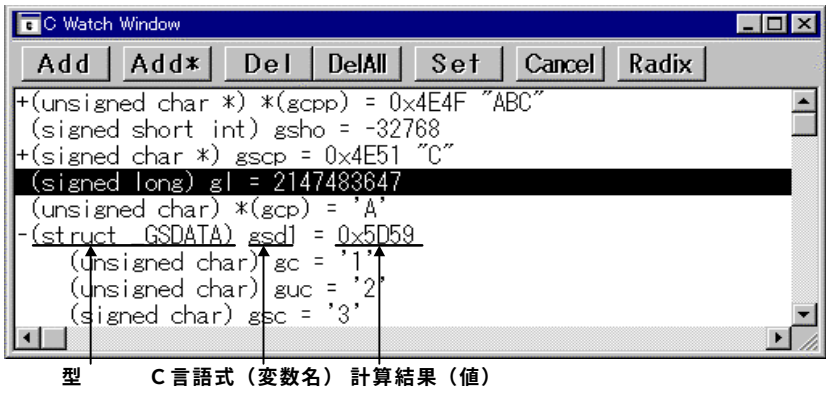
文字列リテラルとして、ダブルクォーテーション（"）で囲まれた文字列が使用できます。
例えば、"abcde"、"I am a boy." 等です。

注意事項

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

3.2 C言語式の表示形式

Cウォッチ、ファイル、ファイルローカル、及びグローバルウィンドウのデータ表示領域における C 言語式の表示は、以下に示すように、その型名、C 言語式（変数名）、計算結果（値）から構成されています。



以下に、型別に表示形式を説明します。

3.2.1 列挙型の場合

- 計算結果の値が定義されているものであれば、その名前で表示します。
  (DATE) date = Sunday　（全 Radix）

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

3.2.2 基本型の場合

- 計算結果が char 型および浮動小数点以外の基本型の場合には、以下のように表示します。
  (unsigned int) i = 65280　（Radix が初期状態の場合）
  (unsigned int) i = 0xFF00　（Radix が 16 進数の場合）
  (unsigned int) i = 1111111100000000B　（Radix が 2 進数の場合）

- 計算結果が char 型の場合には、以下のように表示します。
  (unsigned char) c = 'J'　（Radix が初期状態の場合）
  (unsigned char) c = 0x4A　（Radix が 16 進数の場合）
  (unsigned char) c = 10100100B　（Radix が 2 進数の場合）

- 計算結果が浮動小数点の場合には、以下のように表示します。
  (double) d = 8.207880399131839E-304　（Radix が初期状態の場合）
  (double) d = 0x10203045060708　（Radix が 16 進数の場合）
  (double) d = 0000000010.....1000B　（Radix が 2 進数の場合）
  （...は省略を表す）

3.2.3 ポインタ型の場合

- 計算結果が char*型以外のポインタ型の場合には、以下のように内容を 16 進数表示します。
  (unsigned int *) p = 0x1234　（全 Radix）

- 計算結果が char*型の場合には、メニュー［Option］→［View］→［Display String］で文字列／文字の表示が指定できます。
  表示例を以下に示します。

  －文字列表示の場合
  (unsigned char *) str = 0x1234 "Japan"　（全 Radix）

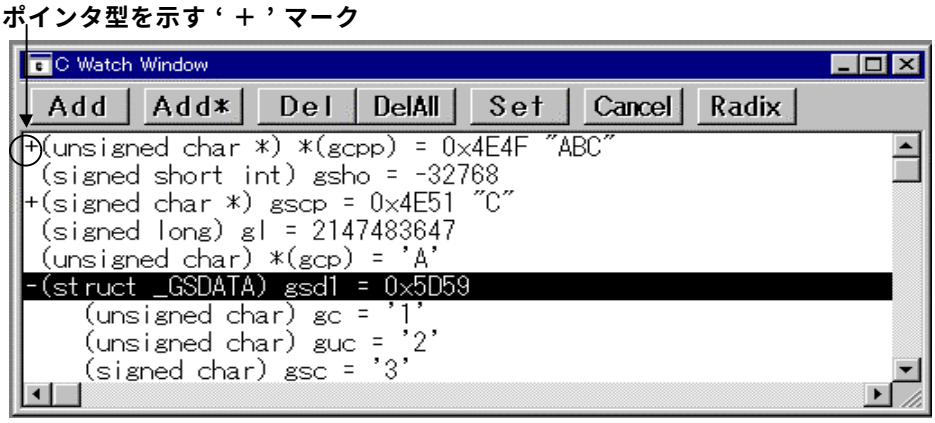
  ー文字表示の場合
  (unsigned char *) str = 0x1234 (74 'J')　（全 Radix）

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

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

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

なお、C言語式がポインタ型の場合は、以下に示すように、型名の左側に‘＋’マークが現れます。



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

3.2.4 配列型の場合

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

  (signed int [10]) z = 0x1234　　（全Radix）

- 計算結果がchar[]型の場合には、以下のように表示します。

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

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

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

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

なお、C言語式が配列型の場合は、ポインタ型と同様、型名の左側に‘＋’マークが現れます。展開方法は、ポインタ型と同じです。詳細な説明については、「3.2.4 ポインタ型の場合」をご参照下さい。

配列のサイズが1000以上の場合、下記ダイアログがオープンするので、展開する要素数を指定してください。

Start で指定した要素から End で指定した要素までを表示します。配列の要素数の最大値を超える値を指定した場合は、配列の最大値を指定した事になります。なお、Cancel ボタンを押下した場合、配列は展開しません。

3.2.5 関数型の場合

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

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

3.2.6 参照型の場合

- 計算結果が参照型の場合には、以下のように参照するアドレスを 16 進数表示します。

```
(signed int &) ref = 0xD038
```
（全 Radix）

3.2.7 ビットフィールド型の場合

- 計算結果がビットフィールド型の場合には、以下のように表示します。

```
(unsigned int :13) s.f = 8191
```
（Radix が初期状態の場合）

```
(unsigned int :13) s.f = 0x1FFF
```
（Radix が 16 進数の場合）

```
(unsigned int :13) s.f = 1111111111111B
```
（Radix が 2 進数の場合）

3.2.8 C シンボルが見つからなかった場合

- 計算した式の中に発見できなかった C シンボルがあった場合には、以下のように表示します。

```
() x = <not active>
```
（全 Radix）

3.2.9 文法エラーの場合

- 計算した式が文法的に間違っていた場合には、以下のように表示します。

```
() str*(p = <syntax error>
```
（全 Radix）

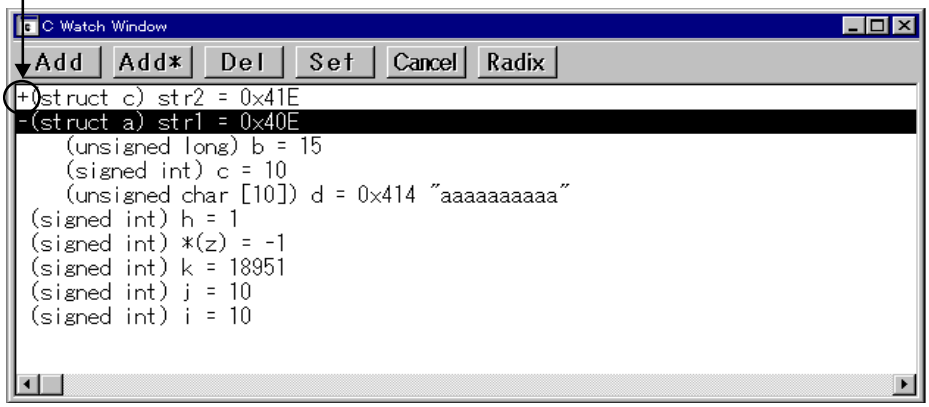
（str*(p は間違った記述）

3.2.10 構造体・共用体型の場合

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

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

構造体・共用体を示す‘ ＋ ’マーク



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

3.2.11 レジスタ変数の場合

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

4 エラーメッセージ一覧

以下に、**PD38SIM**のエラーメッセージ一覧を示します。

番号	エラーメッセージ	補足・対応
150	これ以上 (name) Window はオープンできません。	指定したウィンドウは既に最大枚数オープンしています。
151	(name) Window のオープンに失敗しました。	指定したウィンドウを開くことができません。メモリ不足が原因と考えられます。他のアプリケーションを終了するか、メモリを増設してください。
152	ターゲット実行中のため (name) window はオープンできません。	ターゲットプログラムを停止してからウィンドウをオープンしてください。
153	指定した値が範囲外です。	指定したアドレスが MCU の最大アドレスを越えています。
154	既にPD38SIMは起動しています。	
156	指定されたファイル (filename)が見つかりません。	
157	指定されたパス(pathname)が見つかりません。	
158	メモリが不足しているため実行できませんでした。	
159	実行できませんでした。	

番号	エラーメッセージ	補足・対応
200	表示モードを切り替えることができません。	表示開始アドレスがソースの行頭アドレスと一致していない、または該当するソースファイルが見つかりません。
201	ソースファイル (filename) が見つかりません。	指定されたソースファイルが見つかりません。PATH コマンドまたは、メニュー[Environ]→[Path]でソースファイルがあるディレクトリを指定してください。
202	検索文字列 (name) が見つかりません。	検索開始位置から最後まで指定文字列を検索しましたが見つかりませんでした。
203	ソースファイル (filename) の行数が (line) 行を超えています。	ソースファイルが表示可能な行数を越えているために、ソース表示できません。表示モードを逆アセンブル表示モードに切り替えて表示します。

番号	エラーメッセージ	補足・対応
300	endi が多すぎます (filename lin e) 。	endi に対応する if がありません。
301	endw が多すぎます (filename line）。	endw に対応する while がありません。
303	既にスクリプトファイルは起動しています。	
304	endi が足りません (filename line) 。	if に対応する endi がありません。
305	一行が長すぎます (filename line) 。	一行に記述できる文字数の制限を超えました。
306	ネストが深すぎます (filename line) 。	
307	スクリプトファイルが見つかりません (%s) 。	
308	スクリプトファイルが読み込めません (filename) 。	スクリプトファイルの続きが読み込めません。
309	スクリプト文法エラー (filename line) 。	
310	endw が見つかりません (filename line) 。	while に対応する endw がありません。
311	スクリプトファイルのネストが制限 (num) を越えました。	
313	Break が多すぎます (filename) 。	

番号	エラーメッセージ	補足・対応
400	スクロール範囲外のアドレスが指定されました。	

番号	エラーメッセージ	補足・対応
600	ウォッチポイントの個数が制限数 (num) を超えるので追加できません。	
601	指定したアドレスが範囲外です。	
602	指定した値が範囲外です。	
603	指定したビット値が範囲外です。	

番号	エラーメッセージ	補足・対応
650	シンボル情報がロードされていません。	ロードモジュールファイルがロードされていません。
651	文字列が長すぎます。	

番号	エラーメッセージ	補足・対応
900	シンボルファイルフォーマットが異常です。	ロードモジュールファイルのフォーマットに誤りがあります。
901	シンボルファイルのロードを中断しました。	
902	シンボルファイル (filename) が見つかりません。	ロードモジュールファイルが存在しません。
903	必要なメモリが確保できません。	メモリが不足しています。他のアプリケーションを終了するか、メモリを増設してください。
904	テンポラリファイルがオープンできません。	オンデマンド方式でダウンロードする際のテンポラリファイルがオープンできません。。

番号	エラーメッセージ	補足・対応
1001	シンボルが見つかりません。	指定したシンボルは、存在しません。
1002	指定した式は、ウォッチポイントとして登録できません。	
1004	文法エラーです。	式の記述に誤りがあります。
1005	スコープが見つかりません。	指定された変数は、スコープ内にありません。
1006	シンボルが見つかりません。	
1007	関数が見つかりません。	指定された関数はありません。
1008	右辺式が不適切です。	
1009	型の異なる構造体 (共用体) をコピーしようとしました。	
1010	代入できません。	
1011	型が見つかりません。	指定された型はありません。
1012	浮動小数点型の演算はサポートしておりません。	
1013	指定の演算はポインタ型同士に対してはできません。	
1014	指定の演算はポインタ型に対してはできません。	
1015	ポインタ変数によって減算しようとしました。	
1016	0 で除算しようとしました。	
1017	不正な演算子を用いています。	
1018	型情報が壊れています。	ロードモジュールファイルのシンボル情報に誤りがあります。
1019	左辺値は、ポインタ変数でなければなりません。	
1020	左辺値は、構造体 (共用体) 型でなければなりません。	

番号	エラーメッセージ	補足・対応
1021	メンバが見つかりません。	
1022	左辺値は、構造体 (共用体) 型への参照でなければなりません。	
1023	左辺値が不適切です。	
1024	被演算子は数値でなければなりません。	
1025	指定の被演算子は符号反転できません。	
1026	アドレス値を求めることができません。	
1027	配列変数が不適切です。	
1028	配列の要素番号が不適切です。	
1029	被演算子がアドレスではありません。	
1030	レジスタ変数に対するキャスト演算はサポートしておりません。	
1031	キャストする型の指定が不適切です。	
1032	指定の型に対するキャスト演算はサポートしておりません。	
1033	アドレスに変換できる C 式ではありません。	

番号	エラーメッセージ	補足・対応
1100	指定したアドレス値が範囲外です。	指定したアドレスが MCU の扱える最大値 FFFFFFh を越えています。
1101	アセンブリ言語の記述に誤りがあります。	
1102	ジャンプ先のアドレスが範囲外です。	
1103	指定したオペランドの値が範囲外です。	
1104	式の記述に誤りがあります。	

番号	エラーメッセージ	補足・対応
1200	コマンド行の文法エラーです。	
1201	コマンド名に誤りがあります。	
1202	alias の登録が多すぎます。	alias の最大登録数は、256 個です。
1203	alias にはコマンド名のみ登録できます。	
1204	ターゲットプログラム実行中のため、指定したコマンドは使用できません。	指定されたコマンドは、プログラムを実行しているときには使用できません。
1205	これ以上 up できません。	
1206	これ以上 down できません。	
1207	この関数にブレークをかけることはできません。	
1208	開始アドレスが終了アドレスよりも大きいアドレス値になっています。	
1209	別名にコマンド名および予約語は指定できません。	
1211	現在このコマンドはサポートされていません。	
1212	ファイル (filename) が見つかりません。	指定したファイルを保存することができません。
1213	範囲外のデータ値が指定されました。	

番号	エラーメッセージ	補足・対応
1300	行番号の指定に誤りがあります。	
1301	右括弧 ')' が見つかりません。	
1302	マクロ定数の個数が制限数 (limit) を越えています。	
1303	指定した定数値が範囲外です。	
1304	定数の基数を示すプレフィクスの記述に誤りがあります。	
1305	間接参照の記述に誤りがあります。	
1306	文字列の終わりを示す (str) が見つかりません。	
1307	式の記述に誤りがあります。	
1308	マクロ定数 (macro) が定義されていません。	
1309	シンボル (symbol) が定義されていません。	
1310	定数値の記述に誤りがあります。	
1311	0 で除算を行いました。	
1313	解析結果が MCU の扱える最大値を越えています。	
1314	マクロ変数名にレジスタ名を使用しています。	

番号	エラーメッセージ	補足・対応
1400	指定したアドレス値が範囲外です。	指定したアドレスが MCU の扱える最大値　を越えています。
1401	既にターゲットプログラムは停止しています。	
1402	ブレークポイントの個数が制限数 (limit)を超えています。	
1403	ブレークポイントが設定されていません。	
1404	指定したデータ値が範囲外です。	
1406	指定した領域にメモリがないので、参照／書き込みができません。	メモリの存在しないアドレスに対しての参照／書き込みは行なえません。
1407	必要なメモリが確保できません。	メモリが不足しています。他のアプリケーションを終了するか、メモリを増設してください。
1408	指定したレジスタ値が範囲外です。	
1409	ターゲットプログラム実行中のため、指定したコマンドは使用できません。	
1410	開始アドレスが終了アドレスよりも大きいアドレス値になっています。	
1411	実行を中断しました。	
1412	指定したアドレスを含むソース行が見つかりません。	指定したアドレスには、ソース行情報がありません。
1413	本コマンドは現在サポートされていません。	
1417	これ以上のスタックの検索はできません。	
1418	指定回数が 65535 回を越えています。	
1450	指定したアドレス値が範囲外です。	

番号	エラーメッセージ	補足・対応
1500	シミュレータから未定義のステータスが送信されました。	
1501	指定した領域にメモリがないので、参照／書き込みができません。	
1502	ポイントの個数が制限数(num)を超えています。	これ以上ポイントは設定できません。
1503	すでにポイントが設定されています。	
1504	すでに別の種類のブレークポイントが設定されています。	
1505	指定したアドレスにはハードウェアブレークポイントが設定されていません。	
1506	必要なメモリが確保できません。	
1507	これ以上 I/O スクリプトファイルを登録できません。	I/O ウィンドウに登録できる I/O スクリプトファイルのプロシジャー、仮想ポート入力、及び仮想割り込みは全て合わせて 20 個までです。
1508	これ以上仮想ポート出力を登録できません。	仮想ポート出力に登録できる最大数は 20 です。
1509	指定したベクタアドレスが範囲外です。	
1510	指定した優先度が範囲外です。	
1550	指定したアドレス値が範囲外です。	指定したアドレスが MCU の扱える最大値 を越えています。
1551	指定したビット番号が範囲外です。	
1552	ファイル (filename) が壊れています。	
1553	ファイル (filenam) が見つかりません。	
1554	関数/サブルーチン情報が見つかりません。	デバッグ情報を出力するオプションをつけて、ターゲットプログラムを再度作成して下さい。
1555	シンボル／ラベルとして記述できない文字が文字列中にあります。	
1557	指定した行番号が見つかりません。	
1558	既に同名のシンボル／ラベルが登録されています。	
1559	指定した行番号には、機械語が生成されていません。	指定した行番号に対応するアドレスには機械語が生成されていません。
1560	必要なメモリが確保できません。	
1561	スコープが見つかりません。	
1562	セクション情報が見つかりません。	デバッグ情報を出力するオプションをつけて、ターゲットプログラムを再度作成して下さい。
1563	指定したアドレスに該当するソース行が見つかりません。	
1564	シンボル (symbol) が見つかりません。	
1565	指定したアドレスを含むスコープが見つかりません。	
1566	ロードがキャンセルされました。	
1569	レジスタ名に誤りがあります。	

番号	エラーメッセージ	補足・対応
1704	ターゲットと接続されていません。	
1705	ターゲットに接続できません。	
1707	タイムアウトエラーが発生しました。	ターゲットとの通信中にタイムアウトエラーが発生しました。
1712	通信エラーが発生しました。ターゲットとの接続が切断されました。	ターゲットとの通信中にターゲットとの接続が切断されました。
1713	通信エラーが発生しました。ターゲットにデータを転送できません。	ターゲットへのデータ転送中に通信エラーが発生しました。
1714	通信エラーが発生しました。ターゲットよりデータを受信できません。	ターゲットからのデータ受信中に通信エラーが発生しました。
1717	シミュレータエンジンが見つかりません。	

番号	エラーメッセージ	補足・対応
2400	指定したアドレスが不正です。	
2401	範囲外のデータ値が指定されました。	
2402	指定したアドレスが不正です。	開始アドレスより小さい値を終了アドレスに指定しています。
2403	指定回数は（num）以上を指定してください。	num 以上の値を指定してください。
2404	範囲外の値が指定されました。	

番号	エラーメッセージ	補足・対応
5700	範囲外のデータ値が指定されました。	
5701	指定したアドレスが不正です。	
5702	スクロール範囲外のアドレスが指定されました。	スクロール範囲に指定したアドレスが、MCU の最大アドレス FFFFFFh を越えています。

番号	エラーメッセージ	補足・対応
5800	サンプリング周期の値が範囲外です。	
5801	範囲外のアドレスが指定されました。	
5802	ターゲットプログラム実行中のため、RAM モニタ領域を変更できません。	ターゲットプログラムを停止してから、RAM モニタ領域を変更してください。

番号	エラーメッセージ	補足・対応
5900	指定されたスクリプトファイル (filename) がオープンできません。	
5901	指定されたスクリプトファイル (filename) は既にオープンされています。	
5902	スクリプトファイルがオープンされていません。	
5903	指定されたログファイル (filename) がオープンできません。	
5904	これ以上ログファイルをオープンすることができません。	
5905	ログファイルがオープンされていません。	
5906	指定されたログファイル (filename) は既にオープンされています。	
5907	ビューファイル (filename) がオープンできません。	

番号	エラーメッセージ	補足・対応
6000	ソースファイル (filename) が見つかりません。	
6001	ソースファイル (filename) の行数が (line) 行を超えています。	
6002	範囲外のアドレスが指定されました。	
6003	ファイル (filename) がオープンできません。	
6004	ファイルフォーマットが不正です。	
6006	エミュレータでセーブしたファイルは読み込みできません。	

番号	エラーメッセージ	補足・対応
6401	すでにハードウェアブレークポイントが設定されています。	

番号	エラーメッセージ	補足・対応
10800	指定した値が範囲外です。	
10801	レジスタ情報ファイルが見つかりません。	
10802	レジスタ情報ファイルの記述が間違っています。	
10803	メモリを割り当てることができません。	

番号	エラーメッセージ	補足･対応
11000	セーブファイル名 (filename)が不正です。	
11001	指定したアドレス値が範囲外です。	
11003	ファイル名が不適当です。	

番号	エラーメッセージ	補足・対応
20000	指定したファイルのフォーマットが不正です。	GUI 出力ファイルが壊れています。

番号	エラーメッセージ	補足・対応
21000	ボタンファイル(filename)がオープンできません。	GUI 入力ファイルがオープンできません。
21001	ボタンファイルフォーマットが以上です。	GUI 入力ファイルが壊れています。

番号	エラーメッセージ	補足・対応
22000	テンポラリファイルがオープンできません。	
22001	テンポラリファイルが削除できません。	
22002	データファイル(filename)がオープンできません。	
22003	データが設定されていません。	
22004	既に同一の出力ファイルが設定されています。	
22005	データが見つかりません。	
22006	開始サイクルが終了サイクルよりも大きいサイクル値になっています。	
22007	既に出力ポートが設定されています。	
22008	入力ファイルにデータがありません。	
22009	ファイル形式が不正です。	

番号	エラーメッセージ	補足・対応
25000	{がありません (line)。	
25001	}がありません (line)。	
25002	(がありません(line)。	
25003	シンボルが見つかりません(line , token)。	
25004	)がありません(line)。	
25005	文法エラー(line , token)。	
25006	if 文のネストが深すぎます(line)。	
25007	while 文のネストが深すぎます(line)。	
25008	break 文のネストが深すぎます(line)。	
25009	else 文に対応する if 文がありません(line)。	
25010	使用できない文字が使用されています(line , token)。	
25011	(filename)ファイルがオープンできません。	
25012	(filename)ファイルは、I/O ウィンドウで作成されたファイルではありません。	
25013	メモリ変数の記述に誤りがあります (line)。	

番号	エラーメッセージ	補足・対応
30000	入力されたファイル (filename) が存在しません。	
30001	ファイル (filename) が作成出来ません。	
30002	ファイル (filename) がクローズできません。	
30003	シークエラーが発生しました。 (in filename)。	
30004	ディスク容量が不足しています。	
30005	ファイル (filename) に規定されていないデータ (data) が存在します。(filename)	
30006	動作するためのメモリが不足しています。	
30007	このデータ (data) はサポートされていません。	

【 MEMO 】

索引

このページは白紙です。

索引

―記号―

#isfetch 式.....194
#isint 式.....194
#isread 式.....194
#iswrite 式.....195

- A-

ASM ウォッチウィンドウ..... 47, 49
ASM ウォッチポイント情報格納ファイル 12, 13

- B-

break 文.....191, 215, 216

- C-

CYcle コマンド 10
C ウォッチウィンドウ 50, 52, 127
C ウォッチポイント.....50
Cウォッチポイント情報格納ファイル 12
C 言語式.....50,l 128, 221, 224

- E-

else 文..... 188, 191, 215, 216
endi 文..... 215, 216
endw 文 215, 216

- G-

GUI 出力ウィンドウ..... 79, 179
GUI 出力ファイル..... 9
GUI 出力機能..... 13
GUI 入力ウィンドウ..... 77, 174
GUI 入力ファイル..... 9
GUI 入力機能..... 12

-H-

H/W ブレークポイント設定ダイアログ.....86

- I -

I/O ウィンドウ 60
I/O シミュレーション機能 4
I/O スクリプト 9
I/O スクリプトファイル..... 12
I/O スクリプト文..... 189
IEEE-695 アブソリュート形式ファイル . 11, 89
if 文..... 188, 191, 215, 216
Init ダイアログ 18, 135
int 文..... 189

-M-

Make..... 139
MIX 表示モード..... 99

- P-

pass 文..... 190
PD38SIM ウィンドウ 25

- R-

RAM モニタ..... 4
RAM モニタウィンドウ..... 44, 46
RAM モニタ領域..... 4, 44, 109

- S-

S/W ブレークポイント設定ダイアログ 84
set 文..... 190
StackMonitor コマンド..... 10

- V-

Version Information ダイアログ.....20

-W-

waitc 文..... 189
waiti 文..... 189
while 文..... 188, 191, 215, 216

-あ-

アップロード 93
アドレスに同期した仮想割り込み 157

-い-

インストール 17
インテル HEX フォーマットファイル 11, 13, 93

-う-

ウォッチポイント 47, 109
右辺式 192

-え-

演算子 194, 220

-お-

オーバーステップ実行 103
オンデマンド方式 5, 19
オンメモリ方式 5, 19

-か-

カスタマイズ機能 199
仮想ポート出力 63, 153
仮想ポート出力機能 8, 153
仮想ポート出力ファイル 14
仮想ポート入力 60
仮想ポート入力機能 8, 145
仮想割り込み 63
仮想割り込み機能 8, 155
仮想割り込みに同期した仮想ポート入力 150
カバレッジウィンドウ 81
カバレッジ計測機能 213
カバレッジ計測情報ファイル 12
カバレッジソースウィンドウ 81
カム実行 105
環境設定 27
関数内ローカル変数 53

-き-

起動 17
逆アセンブルファイル 13
逆アセンブル結果保存 93
行番号 220

-く-

繰り返し文 188, 191, 215, 216
グローバルウィンドウ 57, 127
グローバル変数 57
グローバルラベルシンボル 218

-こ-

コメント文 188, 191, 215, 216

-さ-

サイクル数 10
サイクルに同期した仮想ポート入力 145
サイクルに同期した仮想割り込み 155
再ダウンロード 91
左辺式 195
サンプリング周期 109

-し-

時間管理 6
自動ダウンロード 92
ショートカットメニュー 33, 37
初期化ファイル 12
シンボル 192, 218

-す-

スクリプトウィンドウ 58, 59
スクリプトコマンド 58, 130, 205, 215
スクリプトファイル 11, 53, 134, 215
スコープ 218
スコープダイアログ 113
スタック使用量計測 10
ステップ実行 103

-そ-

ソースウィンドウ 35, 37, 137
ソフトウェアブレーク 5, 116
ソフトウェアブレークポイント 5, 84, 116
ソフトウェアブレークポイント保存ファイル 13

-た-

代入文 215
ダウンロード 89
ダンプウィンドウ 42, 43

-て-

定数 192, 217
テンポラリファイル 14, 19

-と-

動作環境 18, 20

-は-

ハードウェアブレーク 5, 121
ハードウェアブレークポイント 5, 86, 121
ハードウェアブレークポイント保存ファイル 14
判断文 188, 191, 215, 216

-ひ-

ビューバッファ 13, 58, 132, 133
ビューファイル 13, 58
表示色のカスタマイズ 101

-ふ-

ファイル内ローカル変数 55
ファイルローカルウィンドウ 55, 56, 127
プログラムウィンドウ 30, 34, 137
プロシジャー 189

-へ-

ヘルプファイル 11

-ま-

マクロ変数 183, 185, 210

-め-

メモリウィンドウ 40, 41
メモリ変数 193, 195, 219

-も-

文字定数 193, 220
モトローラ S フォーマットファイル .. 11, 90, 93

-ら-

ラインアセンブル 137, 138
ラインアセンブルダイアログ 137
ラベル 40, 192, 218

-り-

リードアクセスに同期した仮想ポート入力 .. 148
リセット 105
リターン実行 104

-れ-

レジスタウィンドウ 38
レジスタ情報ファイル 11
レジスタ変数 219

-ろ-

ローカルウィンドウ 53, 54, 127
ローカルラベルシンボル 218
ロギング機能 131
ログファイル 13, 58

【 MEMO 】

M3T-PD38SIM V.2.10 ユーザーズマニュアル

Rev. 1.00
03.05.01
RJJ10J0077-0100Z

M3T-PD38SIM V.2.10
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

RJJ10J0077-0100Z