お客様各位

# カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

# ご注意書き

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

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

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

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

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

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

お客様各位

# 資料中の「三菱電機」、「三菱XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って株式会社日立製作所及び三菱電機株式会社のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。
従いまして、本資料中には「三菱電機」、「三菱電機株式会社」、「三菱半導体」、「三菱XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス　テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

注:「高周波・光素子事業、パワーデバイス事業については三菱電機にて引き続き事業運営を行います。」

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

User's Manual

# エミュレータデバッガ
# PD30 用　カスタムビルダ

## CB30 V.1.00　ユーザーズマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　1998.02

《安全設計に関するお願い》

三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は品質、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三菱半導体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三菱電機株式会社・三菱電機セミコンダクタシステム株式会社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、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社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、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は、予告なしに、本資料に記載した製品または仕様を変更することがあります。三菱半導体製品のご購入に当たりましては、事前に三菱電機株式会社・三菱電機セミコンダクタシステム株式会社または特約店へ最新の情報をご確認頂きますとともに、三菱電機半導体情報ホームページ（http://www.semicon.melco.co.jp/）および三菱開発ツールホームページ（http://www.tool-spt.mesc.co.jp/）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、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社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。三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は、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、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社また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、文書による三菱電機株式会社・三菱電機セミコンダクタシステム株式会社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三菱電機株式会社・三菱電機セミコンダクタシステム株式会社また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

電子メールの場合：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、開発ツールサポート窓口support@tool.mesc.co.jpまで送信ください。

Windows 98/95/Windows NT 4.0版：\SUPPORT\製品名\SUPPORT.TXT
EWS版：/support/製品名/toolinfo.txt

FAXの場合：各ユーザーズマニュアル及びガイドブックの最後に添付されている「技術サポート連絡書」に必要事項を記入の上、開発ツールサポート窓口まで送信ください。FAX送信先は「技術サポート連絡書」に記載してあります。

# 1.概要

## 1.1.セットアップ

CB30 のセットアップは、PD30 のセットアップと同じです。PD30 のセットアップの詳細は、「PD30 V.3.00 ユーザズマニュアル<<セットアップ/機能概要編>>」を参照してください。

## 1.2.特長

CB30 は、PD30 のカスタマイズ機能を利用して、独自のスクリプトコマンド（以下、カスタムコマンドプログラムと呼びます）、または独自のウィンドウ（以下、カスタムウィンドウプログラムと呼びます）を作成する環境です。CB30 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を PD30 に登録することにより、PD30 の機能を拡張することができます。

以下に CB30 の特長を示します。

1. PD30 と同様のユーザインターフェースをサポート
2. プログラミング、ビルド、デバッグを統合した開発環境を提供
3. カスタムコマンドプログラムとカスタムウィンドウプログラムの作成をサポート
4. PD30 のレジスタウィンドウ、メモリウィンドウ、ダンプウィンドウ、スクリプトウィンドウをサポート

次節より、各特長について説明します。

### 1.2.1.PD30 と同様のユーザインターフェースをサポート

PD30 と同様のデザインを採用し、PD30 と使用感の統一を図っています。

### 1.2.2.プログラミング、ビルド、デバッグを統合した開発環境を提供

ソースファイル作成からビルド、デバッグまでのコントロールが可能です。CB30 でサポートするウィンドウには、プロジェクトウィンドウ、メッセージウィンドウ、エディタウィンドウ、ローカルウィンドウ、グローバルウィンドウがあり、それぞれ、プロジェクトの管理、ビルド結果などのステータス表示、ソースファイルの編集、ローカルシンボルの表示、グローバルシンボルの表示が行えます。

### 1.2.3.カスタムコマンドプログラムとカスタムウィンドウプログラムの作成をサポート

CB30 では、プロジェクト作成時にオープンするダイアログで、作成するプログラムの種別を指定することにより、カスタムコマンドプログラムとカスタムウィンドウプログラムの作成を切換えることができます。

### 1.2.4.PD30 のレジスタウィンドウ、メモリウィンドウ、ダンプウィンドウ、スクリプトウィンドウをサポート

PD30 がサポートするウィンドウのうち、レジスタウィンドウ、メモリウィンドウ、ダンプウィンドウ、スクリプトウィンドウをサポートし、カスタムコマンドプログラムとカスタムウインドウプログラムの作成時に使用することができます。

**（注意）スクリプトウィンドウでは、macro スクリプトコマンドは使用できません。**

# 2.各ウィンドウの機能紹介

CB30 のウィンドウ構成を図１に示します。

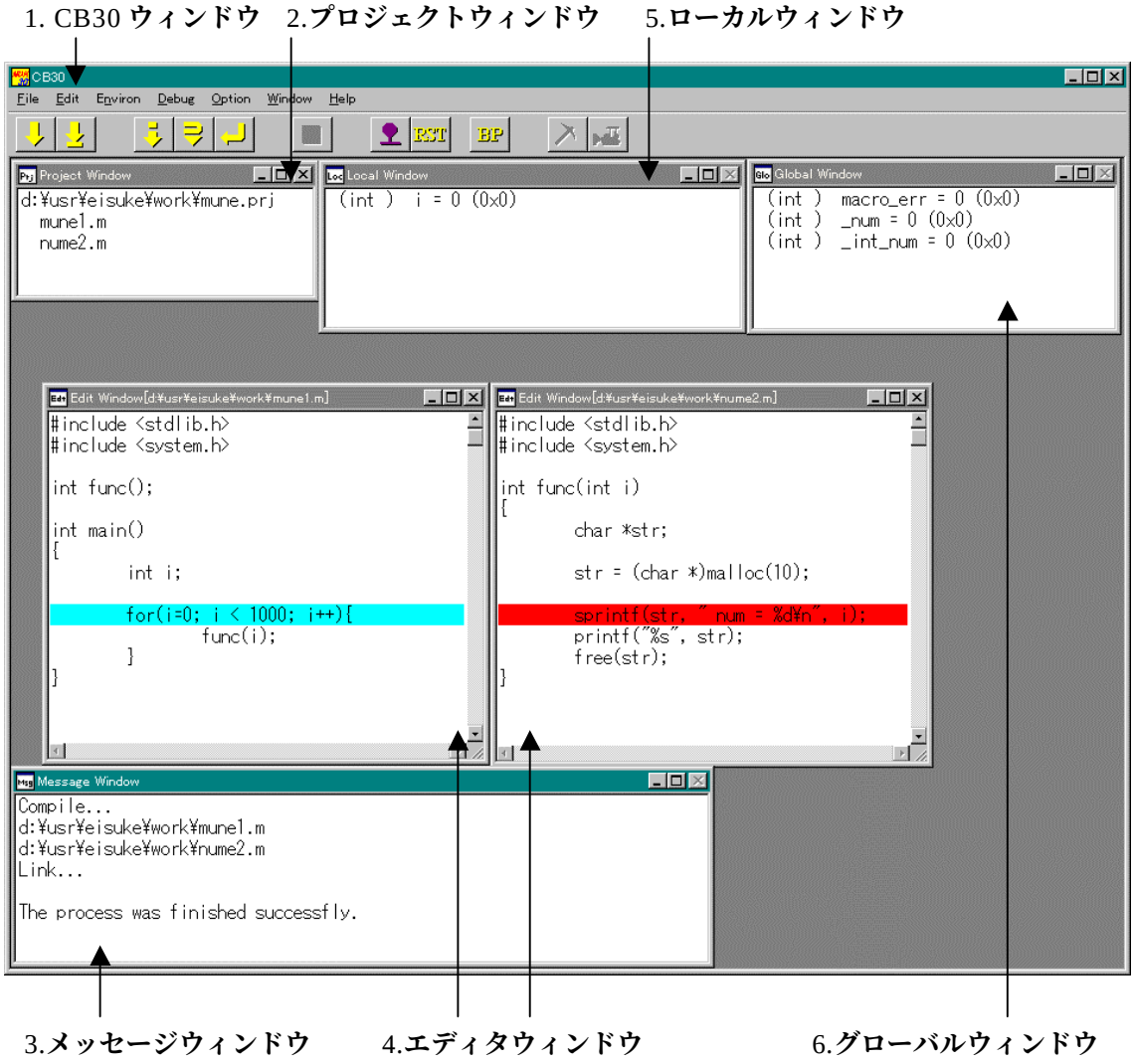


**図１： CB30 のウィンドウ構成図**

CB30 の各ウィンドウの概要とその機能について、以下で説明します。

## 2.1.CB30 ウィンドウ

CB30 ウィンドウは、CB30 のメインウィンドウです。CB30 を起動した際に、最初にオープンします。

### 2.1.1.メニューバー

メニューバーの構成を表１、表２に示します。

表１：メニューバーの構成（CB30 ウィンドウ）（その１）

| メニュー項目 | プルダウンメニュー項目 | 機能 |
|---|---|---|
| [F]ile | [N]ew | |
| | [S]ource/Header... | 新規ソース/ヘッダファイルの作成 |
| | [P]roject... | 新規プロジェクトの作成 |
| | [O]pen... | ソース/プロジェクトのオープン |
| | [S]ave | ソースファイルの保存 |
| | Save [A]s... | 名前をつけて保存 |
| | [C]lose | ソースファイルのクローズ |
| | E[x]it | CB30 の終了 |
| [E]dit | C[u]t | 指定範囲を削除 |
| | [C]opy | 指定範囲をクリップボードにコピー |
| | [P]aste | クリップボードの内容をコピー |
| | [F]ind | 指定文字列の検索 |
| [E]nviron | [I]nit... | Init ダイアログのオープン |
| | [P]ath... | Path ダイアログのオープン |
| [D]ebug | [G]o | プログラムの実行 |
| | [C]ome | プログラムの Come 実行 |
| | [S]tep | プログラムの Step 実行 |
| | [O]ver | プログラムの Over 実行 |
| | Retur[n] | プログラムの Return 実行 |
| | [A]nimate | プログラムの Animate 実行 |
| | [B]reak Point... | Break ダイアログのオープン |
| | Break Point | |
| | [S]et | ブレークポイントの設定・解除 |
| | [L]ist... | Break ダイアログのオープン |
| | [R]eset | リセット |
| | [S]top | プログラムの実行停止 |
| | B[u]ild | 現プロジェクトの構築 |
| | R[e]Build | 現プロジェクトの再構築 |
| [O]ption | フォーカスを持つウィンドウによって変更される（３．２節以降を参照） | |

表２：メニューバーの構成（CB30 ウィンドウ）（その２）

| メニュー項目 | プルダウンメニュー項目 | 機能 |
|---|---|---|
| [W]indow | [C]ascade | ウィンドウを重ねて表示 |
| | [T]ile | ウィンドウを並べて表示 |
| | [A]rrange Icon | アイコンの整列 |
| | [R]egister Window | PD30 のレジスタウィンドウをオープン |
| | M[e]mory Window | PD30 のメモリウィンドウをオープン |
| | [D]ump Window | PD30 のダンプウィンドウをオープン |
| | Scr[i]pt Window | PD30 のスクリプトウィンドウをオープン |
| [H]elp | [I]ndex | オンラインヘルプ目次のオープン |
| | [A]bout... | CB30 のバージョンを表示 |

### 2.1.2.ツールバー

ツールバーの構成を表３に示します。

表３：ツールバーの構成（CB30 ウィンドウ）

| ボタン | 機能 | 対応メニュー |
|---|---|---|
|  | 実行 | [Debug]→[Go] |
|  | カーソル位置まで実行 | [Debug]→[Come] |
|  | ステップ実行 | [Debug]→[Step] |
|  | オーバー実行 | [Debug]→[Over] |
|  | リターン実行 | [Debug]→[Return] |
|  | 実行停止 | [Debug]→[Stop] |
|  | ブレークポイントの設定・解除 | [Debug]→[Break Point]→[Set] |
|  | リセット | [Debug]→[Reset] |
|  | ブレークダイアログのオープン | [Debug]→[Break Point...] |
|  | ビルド | [Debug]→[Build] |
|  | リビルド | [Debug]→[ReBuild] |

## 2.2.プロジェクトウィンドウ

プロジェクトウィンドウは、CB30 で作成するカスタムコマンドプログラムとカスタムウィンドウプログラムのソースファイルを管理するウィンドウです。プロジェクトウィンドウに表示されたソースファイルはマウスのダブルクリックなどによりエディタウィンドウ上にオープンする事ができます。

### 2.2.1.メニューバー

メニューバーの Option メニューの構成を表４に示します。

表４：メニューバー Option メニューの構成（プロジェクトウィンドウ）

| メニュー項目 | プルダウンメニュー項目 | 機能 |
|---|---|---|
| [O]ption | [S]et up...<br>[A]dd File...<br>[D]el File | セットアップダイアログのオープン<br>プロジェクトにソースファイルを追加<br>プロジェクトよりソースファイルを削除 |

## 2.3.メッセージウィンドウ

メッセージウィンドウは、ビルド時のコンパイルエラーやリンクエラー、その他デバッグ中のメッセージを表示するウィンドウです。これらのメッセージはビルド開始時に初期化されます。コンパイルエラーが表示された行をマウスでダブルクリック、もしくはシングルクリックで選択した後にメニューバーの[Option]→[Jump]メニューを選択すると、エディタウィンドウに該当するソースファイルを表示し、該当する行にカーソルを移動します。

### 2.3.1.メニューバー

メニューバーの Option メニューの構成を表５に示します。

表５：メニューバー Option メニューの構成（メッセージウィンドウ）

| メニュー項目 | プルダウンメニュー項目 | 機能 |
|---|---|---|
| [O]ption | [J]ump | エラー発生行を表示 |

## 2.4.エディタウィンドウ

エディタウィンドウはソースファイルを編集するウィンドウです。このウィンドウは複数のオープンが可能であり、タイトルバーにはソースファイル名が表示されます。エディタウィンドウは、文字入力／削除、クリップボードのカット／ペースト、ファイルのロード／セーブなどの編集機能を提供します。また、デバッグ時には、ブレークポイント行が赤色で示され、次の実行行が青色で示されます。ブレークポイント行と次の実行行が重なった場合には黄色で示されます。

### 2.4.1.メニューバー

エディタウィンドウの Option メニューにはサブメニューがありません。

## 2.5.ローカルウィンドウ

デバッグ時にプログラムカウンタに該当する関数のローカル変数とその値を表示するウィンドウです。このウィンドウはデバッグ開始時にオープンされ、デバッグ終了時にクローズされます。

### 2.5.1.メニューバー

ローカルウィンドウの Option メニューにはサブメニューがありません。

## 2.6.グローバルウィンドウ

デバッグ時にグローバル変数とその値を表示するウィンドウです。このウィンドウはデバッグ開始時にオープンされ、デバッグ終了時にクローズされます。

### 2.6.1.メニューバー

グローバルウィンドウの Option メニューにはサブメニューがありません。

# 3.プログラムの作成方法

CB30 を用いた、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作成方法について、簡単なサンプルを用いて説明します。

## 3.1.カスタムコマンドプログラムの作成方法

CB30 を用いて、カスタムコマンドプログラムを作成する手順は、以下の通りです。

1. カスタムコマンドプログラム用新規プロジェクトを作成する
2. 新規ソースファイルを記述する
3. ソースファイルをプロジェクトに登録する
4. ビルドする
5. 必要に応じてデバッグ、ソースファイルの修正をおこなう
6. プログラムが正常に動作するまで、４～５を繰り返す

本節で作成するカスタムコマンドプログラムの仕様は以下の通りです。

| | |
|---|---|
| プログラム名 | m_reset |
| パラメタ | なし |
| 機能 | リセット前のプログラムカウンタ値を表示する。<br>ターゲット MCU をリセットする。<br>リセット後のプログラムカウンタ値を表示する。 |

### 3.1.1.カスタムコマンドプログラム用新規プロジェクトの作成方法

・CB30 ウィンドウの[File]→[New]→[Project...]メニューを選択すると、以下のダイアログがオープンします。

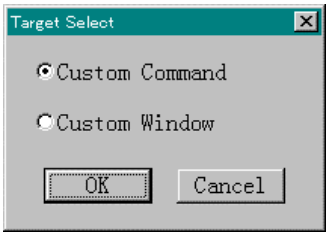


図２: ターゲット選択ダイアログ

Custom Command を選択し、OK ボタンを入力します。

・ファイルセレクションダイアログがオープンするので、プロジェクト名を入力し、保存ボタンを入力します（拡張子は指定しなくても構いません）。本節で作成するサンプルカスタムコマンドプログラム名の m_reset を入力した図を以下に示します。

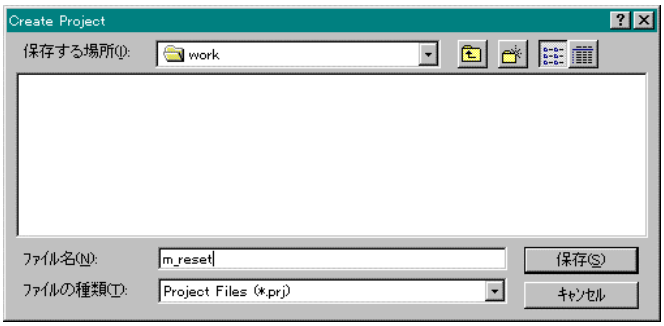


図３：作成プロジェクト名選択ダイアログ

・作成されたプロジェクトファイル名を表示したプロジェクトウィンドウと、プロジェクトのセットアップダイアログがオープンします。

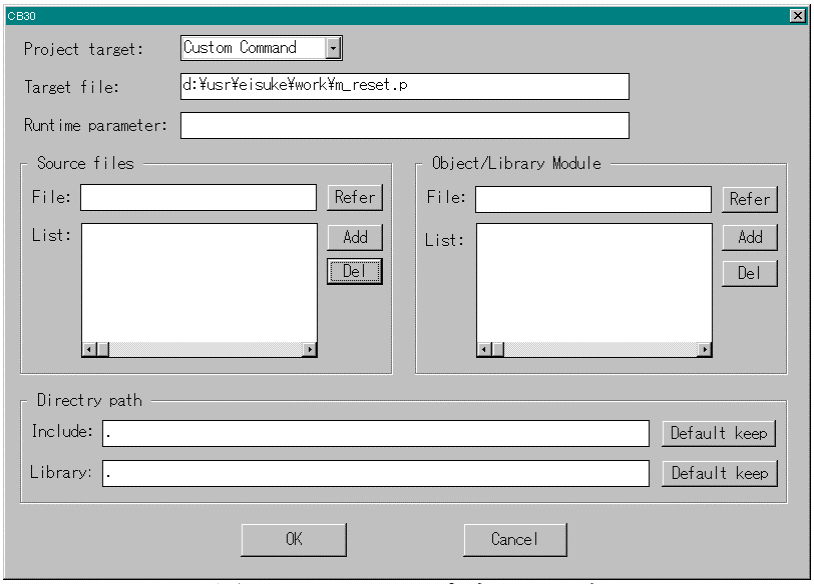


図４:セットアップダイアログ

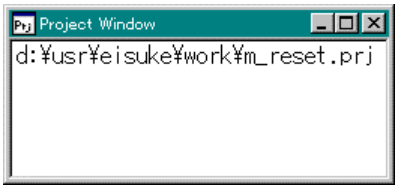


図５：プロジェクトウィンドウ

・セットアップダイアログは、プロジェクトウィンドウの Option メニューから随時オープンでき、設定の変更が可能です。この例では、単に Cancel ボタンを入力しておくことにします。セットアップダイアログの使用方法の詳細は24ページの「セットアップダイアログの使用方法セットアップダイアログの使用方法」を参照してください。

以上の操作によりプロジェクトファイル m_reset.prj が作成されます。

### 3.1.2.新規ソースファイルの作成方法

CB30 ウィンドウの[File]→[New]→[Source/Header...]メニューを選択すると、以下のエディタウィンドウがオープンします。

図６： 空のエディタウィンドウ

このエディタウィンドウにフォーカスを移動し、CB30 の[File]→[Save As...]メニューを選択すると、Save As ダイアログがオープンするので、ファイル名を入力し、保存ボタンを入力します。ソースファイル名の拡張子として、".m"を指定します。

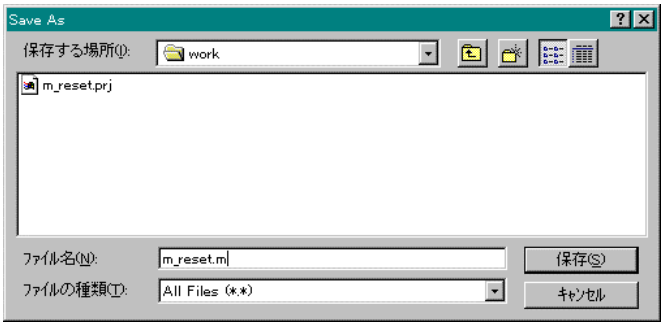


図７： Save As ダイアログ

タイトルバーに、Save As ダイアログで付けられた名前が表示されます。

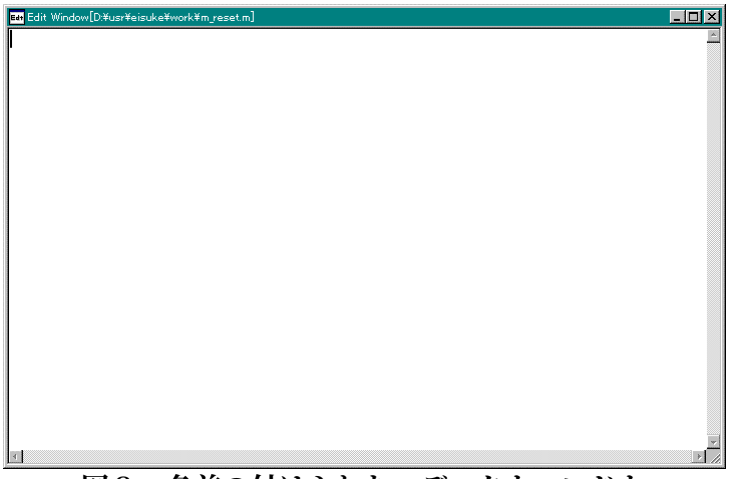


図８： 名前の付けられたエディタウィンドウ

エディタウィンドウでカスタムコマンドソースプログラムを記述します。

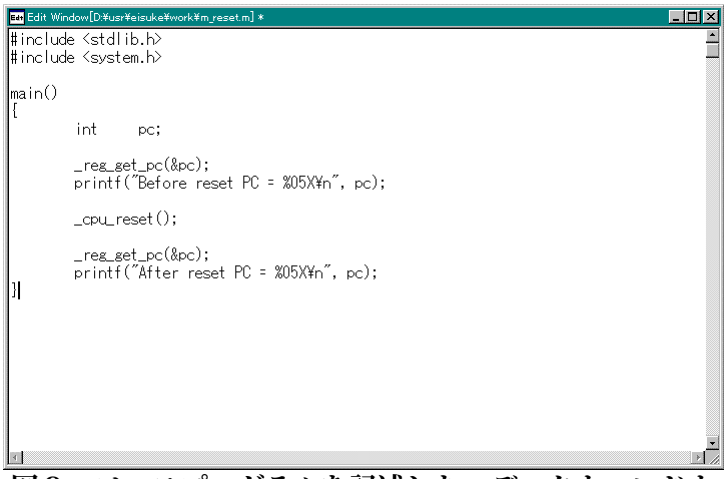


図９：ソースプログラムを記述したエディタウィンドウ

プログラム言語仕様は、29ページの「プログラム言語仕様プログラム言語仕様」を参照してください。

ライブラリ関数仕様は、30ページの「リファレンスリファレンス」を参照してください。

タイトルバーのファイル名の最後に追加されたアスタリスク'*'は、このファイルに変更があったことを示しています。

以上の操作で、カスタムコマンドソースファイル m_reset.m が作成されます。

### 3.1.3.ソースファイルのプロジェクトへの登録方法

前節で作成したソースファイルをビルドするためには、プロジェクトへ登録する必要があります。プロジェクトウィンドウの[Option]→[Add File...]メニューを選択すると Add in source ダイアログがオープンするので、プロジェクトへ登録するファイル名を選択し、開くボタンを入力します。プロジェクトウィドウに登録されたソースファイル名が表示されます。

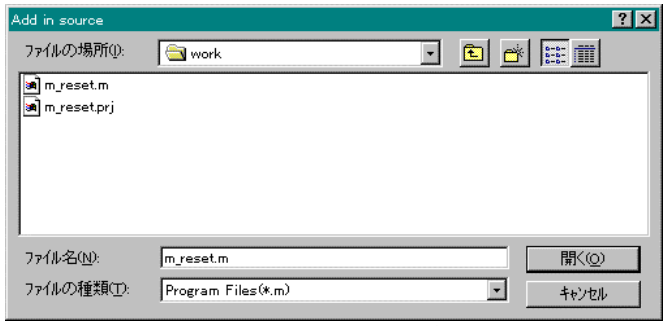


図１０: Add in source ダイアログ

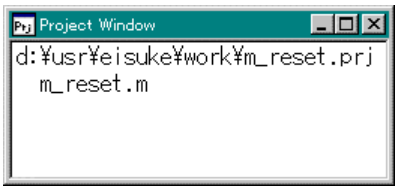


図１１: ソースファイルが登録されたプロジェクトウィンドウ

以上の操作で、プロジェクトに m_reset.m が登録されます。

ソースファイルのプロジェクトへの登録は、セットアップダイアログで行うことも可能です。セットアップダイアログの使用方法は24ページの「セットアップダイアログの使用方法セットアップダイアログの使用方法」を参照してください。

### 3.1.4.ビルドの方法

プロジェクトに登録されたソースファイルを処理し、カスタムコマンドプログラムファイル、およびカスタムウィンドウプログラムファイルを生成することをビルド、およびリビルドと呼びます。ビルドは、プロジェクトに登録されているソースファイルのうち、以前にプログラムファイルを生成してから変更のあったソースファイルのみを処理の対象にします。リビルドは、プロジェクトに登録されているソースファイル全てを処理の対象にします。

ビルドは、CB30 の[Debug]→[Build]メニューを選択するか、ツールバーのビルドボタンを入力することで実行されます。
リビルドは、CB30 の[Debug]→[ReBuild]メニューを選択するか、ツールバーのリビルドボタンを入力することで実行されます。

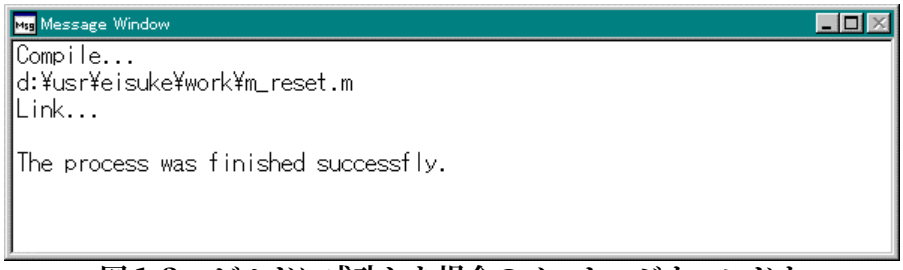


図１２：ビルドに成功した場合のメッセージウィンドウ

以上の操作で、ソースプログラムおよびSet upダイアログの設定に間違いが無ければ、カスタムコマンドプログラムファイルが生成されます。

この例では、プロジェクトの作成時にオープンしたSet upダイアログで、単にCancelボタンを入力を行ったために、インクルードファイル、およびライブラリファイルの検索パスが、デフォルト値（カレントディレクトリ）になっています。したがって、これまでの操作を行って、ビルドした場合には、インクルードファイルがオープンできない旨のメッセージがメッセージウィンドウに表示されます。

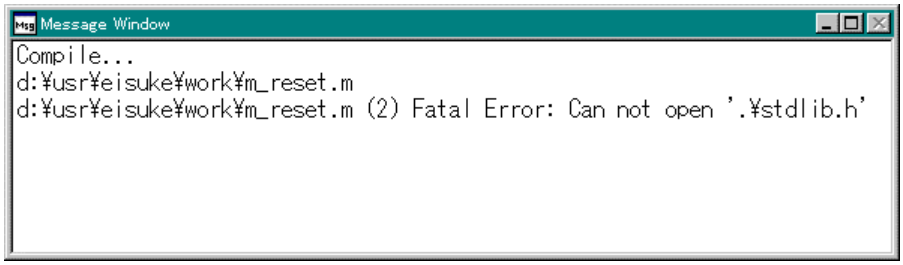


図１３：ビルド時にエラーが発生した場合のメッセージウィンドウ

このとき、メッセージウィンドウに表示されたエラーメッセージ行をシングルクリックして、[Option]→[Jump]メニューを選択するか、エラーメッセージ行をダブルクリックすると、該当するソース行がエディタウィンドウに表示され、カーソルが移動します。

この例の場合では、セットアップダイアログで、インクルードファイル、およびライブラリファイルの検索パスを適切に設定することにより、ビルドに成功するようになります。

セットアップダイアログの使用方法の詳細は24ページの「セットアップダイアログの使用方法セットアップダイアログの使用方法」を参照してください。

### 3.1.5.カスタムコマンドプログラムの実行例

この例で作成した、m_reset コマンドの実行例を以下に示します。
実行は、CB30 ウィンドウのツールバーにある実行ボタンを入力します。

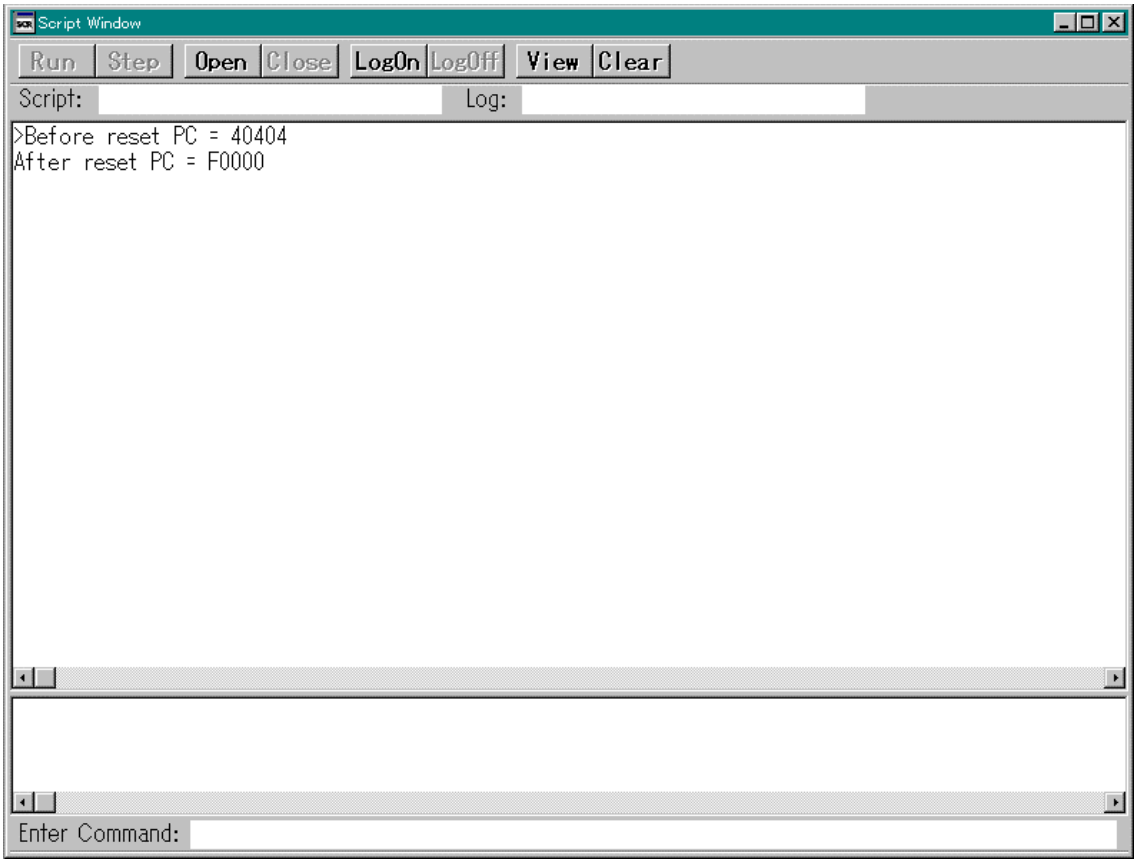


図１４：カスタムコマンドプログラム m_reset.p の実行例

リセット前のアドレスが 40404H で、リセット後のアドレスが F0000 Hであることが表示されています。

カスタムコマンドプログラムからの出力は、スクリプトウィンドウに送られます。したがって、スクリプトウィンドウがオープンされていない場合には、カスタムコマンドプログラムからの出力は、確認できません。

## 3.2.カスタムウィンドウプログラムの作成方法

CB30 を用いて、カスタムウィンドウプログラムを作成する手順は、以下の通りです。

1. カスタムウィンドウプログラム用新規プロジェクトを作成する
2. CB30 によって生成された、フレームワークソースファイルを編集する
3. ビルドする
4. 必要に応じてデバッグ、ソースファイルの修正をおこなう
5. プログラムが正常に動作するまで、３～４を繰り返す

本節で作成するカスタムウィンドウプログラムの仕様は以下の通りです。

| | |
|---|---|
| プログラム名 | dump1000 |
| 機能 | １０００Ｈ番地からの１２８バイトをダンプする |

### 3.2.1.カスタムウィンドウプログラム用新規プロジェクトの作成方法

・CB30 ウィンドウの[File]→[New]→[Project...]メニューを選択すると、以下のダイアログがオープンします。

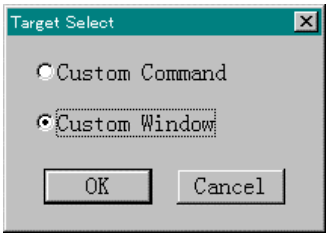


図１５：ターゲット選択ダイアログ

Custom Window を選択し、OK ボタンを入力します。

・ファイルセレクションダイアログがオープンするので、プロジェクト名を入力し、保存ボタンを入力します（拡張子は指定しなくても構いません）。本節で作成するサンプルカスタムウィンドウプログラム名の dump1000 を入力した図を以下に示します。

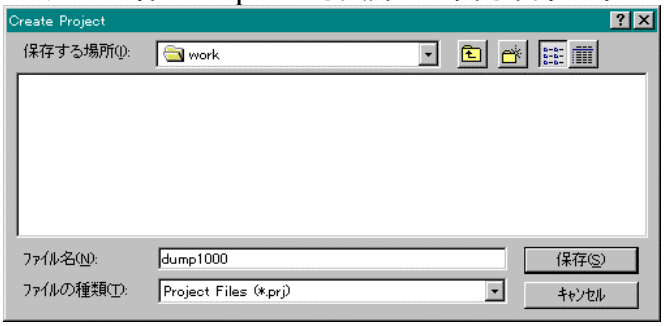


図１６：作成プロジェクト名選択ダイアログ

・以下のフレームワーク生成の確認ダイアログがオープンしますので、はいを入力します。

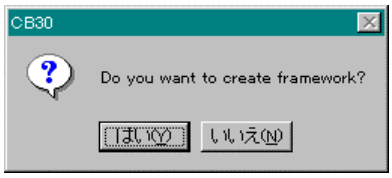


図１７：フレームワーク生成確認ダイアログ

ここで、いいえを入力すると、フレームワークの自動生成は行われません。

・作成されたプロジェクトファイル名を表示したプロジェクトウィンドウと、プロジェクトのセットアップダイアログがオープンします。

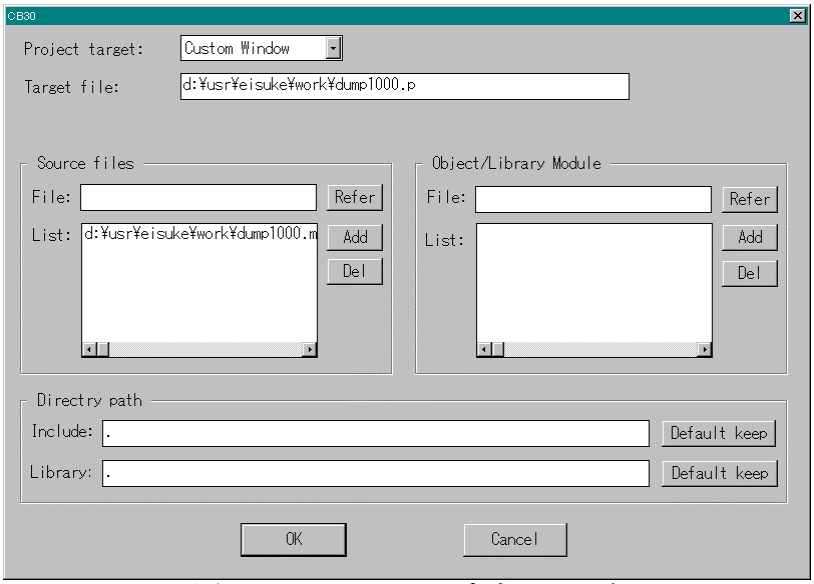


図１８：セットアップダイアログ

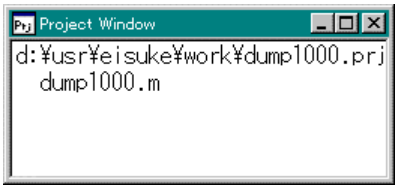


図１９：プロジェクトウィンドウ

・セットアップダイアログは、プロジェクトウィンドウの Option メニューから随時オープンでき、設定の変更が可能です。この例では、単に Cancel ボタンを入力しておくことにします。セットアップダイアログの使用方法の詳細は24ページの「セットアップダイアログの使用方法セットアップダイアログの使用方法」を参照してください。

カスタムウィンドウプログラム用のプロジェクトを生成する際には、CB30 によってフレームワークソースファイルが自動生成されます。この例では、dump1000.m が自動生成されます。カスタムウィンドウプログラムのプログラミングは、このフレームワークソースファイルを編集することにより行います。

以上の操作によりプロジェクトファイル dump1000.prj、およびフレームワークソースファイル dump1000.m が作成されます。

### 3.2.2.CB30 によって自動生成された、フレームワークソースファイルの編集方法

CB30 によって自動生成されたフレームワークソースファイルには、ウィンドウのイベントに対応したハンドル関数が記述されています。

ハンドル関数の詳細は、87ページの「カスタムウィンドウ用ハンドル関数カスタムウィンドウ用ハンドル関数」を参照してください。

この例で取り扱うハンドル関数は、OnDraw 関数と OnEvent 関数です。OnDraw 関数は、他のウィンドウに隠れていた部分が表示された場合に呼び出されます。OnEvent 関数は、ターゲットのメモリ値が変更になった場合など、デバッガの状態を変更する必要がある場合に呼び出されます。

dump1000 では、OnDraw 関数が呼び出された際に、1000H 番地から 128 バイトのメモリ値をターゲットから入手し、文字列に変換してウィンドウに表示します。これらの処理は、OnDraw 関数の内部を編集して記述します。また、OnEvent 関数が呼び出された際に、OnDraw 関数を呼び出し、ウィンドウの表示を更新します。

**（注意）フレームワークソースファイルに記述されている関数を削除しないでください。**
**正常にビルドできなくなります。関数の追加には制限はありません。**

カスタムウィンドウプログラム dump1000 用に編集した、OnDraw 関数を表示した、エディタウィンドウを以下に示します。

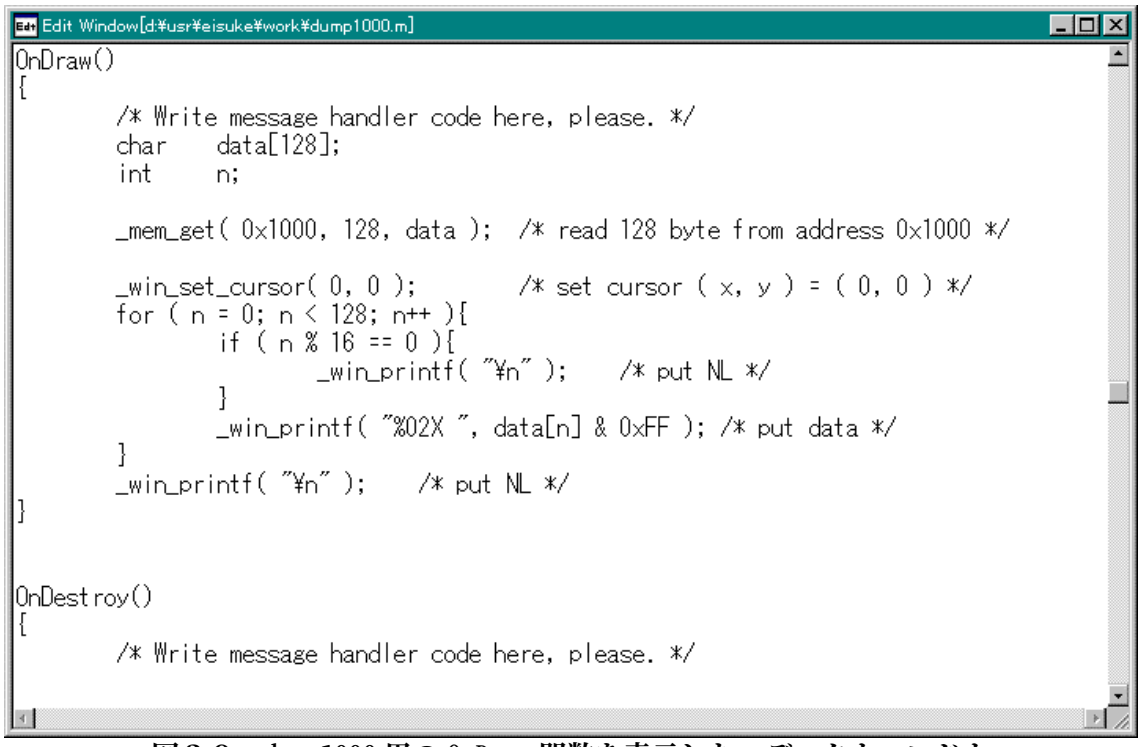


図２０：dump1000 用の OnDraw 関数を表示したエディタウィンドウ

ビルドの方法は、カスタムコマンドプログラムを作成する場合と同様です。17ページの「ビルドの方法ビルドの方法」を参照してください。

### 3.2.3.カスタムウィンドウプログラムの実行例

この例で作成した、dump1000 ウィンドウの実行例を以下に示します。
実行は、CB30 ウィンドウのツールバーにある実行ボタンを入力します。

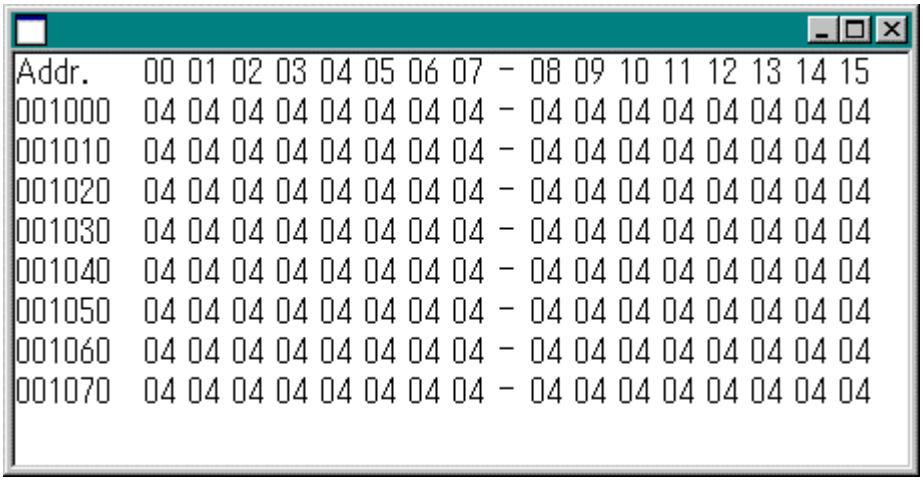


図２１： カスタムウィンドウプログラム dump1000.p の実行例

1000 H番地からの 128 バイトがダンプ形式で表示されています。

カスタムウィンドウプログラムは、他のウィンドウによって隠されていた領域が表示される場合に OnDraw 関数を呼び出し、ターゲットメモリの内容を変更した場合など、デバッガの状態を更新する必要がある場合に OnEvent 関数を呼び出します。したがって、dump1000 カスタムウィンドウプログラムは、表示されていなかった部分が表示された場合や、ターゲットメモリの内容が変更した場合などに、自動的に表示が更新されます。

## 3.3.セットアップダイアログの使用方法

セットアップダイアログは、プロジェクトの設定を行うダイアログです。プロジェクトウィンドウの[Option]→[Set up...]メニューを選択するか、プロジェクトウィンドウに表示されているプロジェクトファイル名をダブルクリックすることによりオープンします。

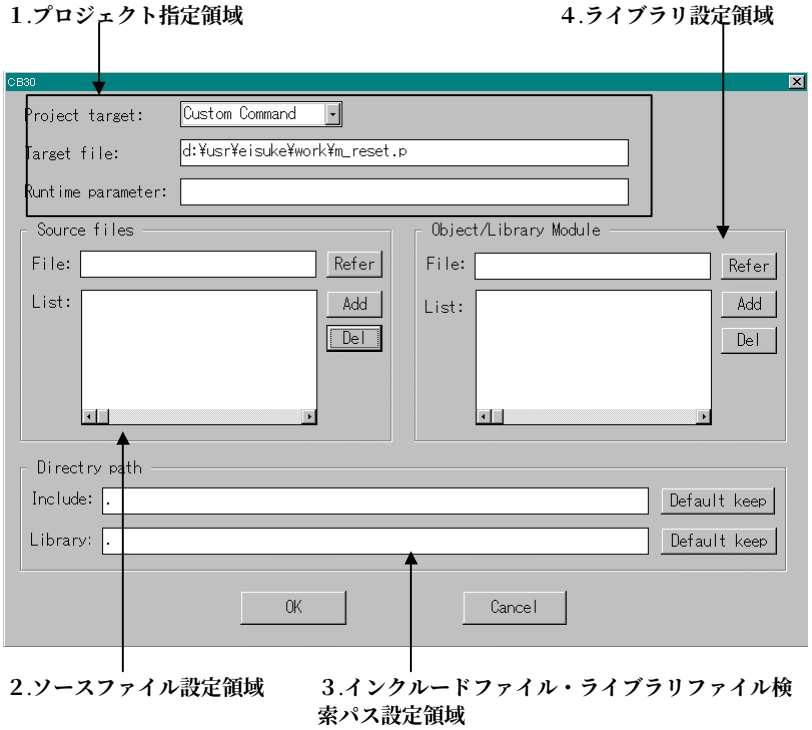


図２２：セットアップダイアログの構成図

### 3.3.1.プロジェクト設定領域

プロジェクト設定領域は、以下の３つの領域から構成されます。

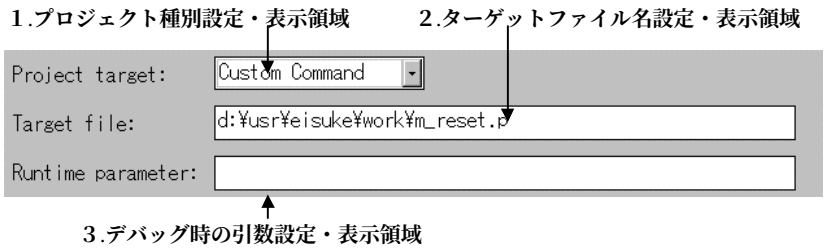


図２３：プロジェクト設定領域の構成図

#### 3.3.1.1.プロジェクト種別設定・表示領域

プロジェクト種別は、以下の２種類が設定可能です。

| | |
|---|---|
| Custom Command | カスタムコマンドプログラムを作成します |
| Custom Window | カスタムウィンドウプログラムを作成します |

設定されたプロジェクト種別を表示します。

作成するプログラムに応じた種別を選択することにより、ビルド時に結合するスタートアップルーチン、ライブラリを切換えます。

プロジェクト種別の変更は、ビルド時に結合するスタートアップルーチン、ライブラリの切換えのみに影響します。

#### 3.3.1.2.ターゲットファイル名設定・表示領域

ビルド時に生成するプログラムファイル名を設定します。

設定されたファイル名を表示します。

#### 3.3.1.3.デバッグ時の引数設定・表示領域

プロジェクト種別に Custom Command を指定した場合に現われ、カスタムコマンドプログラムのデバッグ時の引数を設定します。設定された引数は、main()関数の引数、argc、argv に以下の内容で渡されます。

| | |
|---|---|
| argc | 引数の個数 |
| argv | 引数に指定された文字列が格納されている領域へのポインタが格納されているポインタ配列のアドレス |

設定された引数を表示します。

### 3.3.2.ソースファイル設定領域

ソースファイル設定領域は、以下の５つの領域から構成されます。

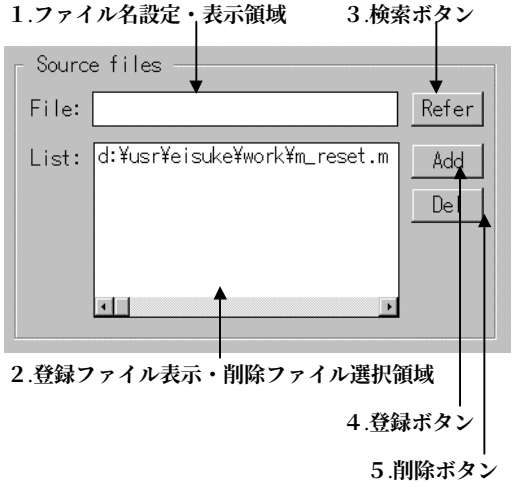


図２４: ソースファイル設定領域

3.3.2.1.**ファイル名設定・表示領域**

プロジェクトに登録するソースファイル名を設定します。

本領域に設定したソースファイルは、登録ボタンを入力することによりプロジェクトに登録され、登録ファイル表示・削除ファイル選択領域にソースファイル名が表示されます。

登録ボタンを入力することによりプロジェクトに登録されるソースファイル名を表示します。

3.3.2.2.**登録ファイル表示・削除ファイル選択領域**

プロジェクトに登録されているソースファイル名を表示します。

表示されているソースファイル名をマウスのシングルクリックで選択し、削除ボタンを入力することにより、プロジェクトから削除することができます。

3.3.2.3.**検索ボタン**

検索ボタンを入力することにより、ファイルセレクションダイアログがオープンします。

オープンしたダイアログで選択したソースファイル名がファイル名設定・表示領域に入力されますので、続いて登録ボタンを入力することにより、直接ソースファイル名を入力すること無く、プロジェクトへのソースファイルの登録が行えます。

3.3.2.4.**登録ボタン**

ファイル名設定・表示領域に入力されているソースファイルを、プロジェクトに登録します。

登録時にソースファイルの存在を確認しますので、指定したソースファイルが存在しない場合、または既にプロジェクトに登録されている場合には、登録を行いません。

3.3.2.5.**削除ボタン**

登録ファイル表示・削除ファイル選択領域で、マウスのシングルクリックで選択されたソースファイルをプロジェクトから削除します。

ソースファイルが選択されていない場合には、削除を行いません。

### 3.3.3.インクルードファイル・ライブラリファイル検索パス設定領域

インクルードファイル・ライブラリファイル検索パス設定領域は、以下の４つの領域から構成されます。

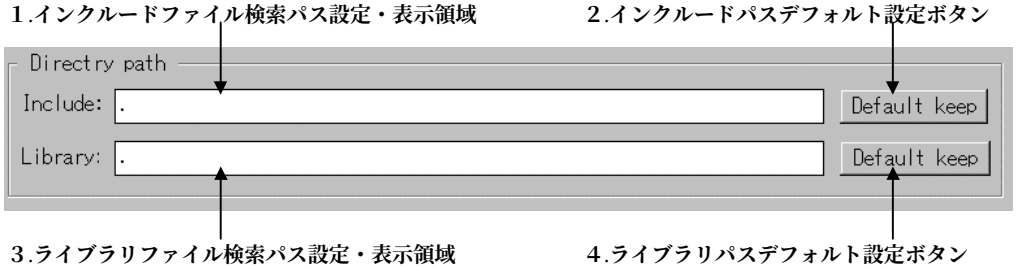


図２５：インクルードファイル・ライブラリファイル検索パス設定領域の構成図

#### 3.3.3.1.インクルードファイル検索パス設定・表示領域

ソースファイル中に、#include <*filename*>でファイルのインクルードを指定された場合に、ファイルを検索するディレクトリを設定します。

通常システムインクルードファイルが格納されているディレクトリを設定します。

インストーラを使用してインストールした場合には、システムインクルードファイルは、C:\MTOOL\PD30\INCLUDE にインストールされます。

設定されたインクルードファイル検索パスを表示します。

#### 3.3.3.2.インクルードパスデフォルト設定ボタン

インクルードファイル検索パス設定・表示領域に設定されているディレクトリを、CB30 で新規にプロジェクトを生成する際のデフォルトに設定します。

本ボタンで設定した後に CB30 で新規プロジェクトを生成した際に、設定されたディレクトリがインクルードファイル検索パスとして設定されます。

#### 3.3.3.3.ライブラリファイル検索パス設定・表示領域

ビルド時にリンクするライブラリファイルを検索するディレクトリを設定します。

通常システムライブラリファイルが格納されているディレクトリを設定します。

インストーラを使用してインストールした場合には、システムライブラリファイルは、C:\MTOOL\PD30\LIB にインストールされます。

設定されたライブラリファイル検索パスを表示します。

#### 3.3.3.4.ライブラリパスデフォルト設定ボタン

ライブラリファイル検索パス設定・表示領域に設定されているディレクトリを、CB30 で新規にプロジェクトを生成する際のデフォルトに設定します。

本ボタンで設定した後に CB30 で新規プロジェクトを生成した際に、設定されたディレクトリがライブラリファイル検索パスとして設定されます。

### 3.3.4.ライブラリ設定領域

ライブラリ設定領域は、以下の５つの領域から構成されます。

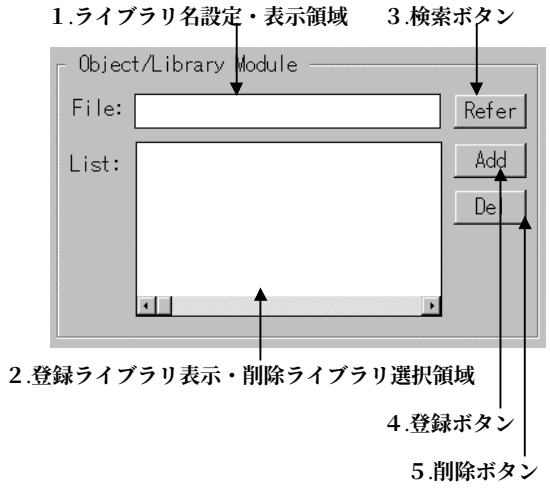


図２６: ライブラリ設定領域

#### 3.3.4.1.ライブラリ名設定・表示領域

プロジェクトに登録する、ビルド時にシステムライブラリ以外にリンクしたいライブラリファイル名を設定します。

本領域に設定したライブラリファイルは、登録ボタンを入力することによりプロジェクトに登録され、登録ライブラリファイル表示・削除ライブラリファイル選択領域にライブラリファイル名が表示されます。

登録ボタンを入力することによりプロジェクトに登録されるライブラリファイル名を表示します。

#### 3.3.4.2.登録ライブラリ表示・削除ライブラリ選択領域

プロジェクトに登録されているライブラリファイル名を表示します。

表示されているライブラリファイル名をマウスのシングルクリックで選択し、削除ボタンを入力することにより、プロジェクトから削除することができます。

#### 3.3.4.3.検索ボタン

検索ボタンを入力することにより、ファイルセレクションダイアログがオープンします。

オープンしたダイアログで選択したライブラリファイル名がライブラリ名設定・表示領域に入力されますので、続いて登録ボタンを入力することにより、直接ライブラリファイル名を入力すること無く、プロジェクトへのライブラリファイルの登録が行えます。

#### 3.3.4.4.登録ボタン

ライブラリ名設定・表示領域に入力されているライブラリファイルを、プロジェクトに登録します。

登録時にライブラリファイルの存在を確認しますので、指定したライブラリファイルが存在しない場合、または既にプロジェクトに登録されている場合（システムライブラリを含む）には、登録を行いません。

#### 3.3.4.5.削除ボタン

登録ライブラリ表示・削除ライブラリ選択領域で、マウスのシングルクリックで選択されたライブラリファイルをプロジェクトから削除します。

ライブラリファイルが選択されていない場合には、削除を行いません。

# 4.プログラム言語仕様

CB30 で記述可能なプログラム記述言語は、C 言語のサブセットになっており、一般的な C 言語と比較して、以下の様な制限事項があります。

- struct, union, enum がありません。
- 初期化を伴う変数の宣言ができません。
  ```
  int         a = 10;          など
  ```
- static 記憶クラスがありません。
- 記憶クラス指定子は、extern のみが使用できます。
- 型は、char 型、int 型、pointer 型、配列型のみが使用できます。
  ```
  char        a;        /* 1Byte */
  int         b;        /* 4Byte */
  char        *str;     /* 4Byte */
  int         *p;       /* 4Byte */      など
  ```
- char 型、int 型は符号付です(signed、unsigned 指定子は使用できません)
- 関数のプロトタイプ宣言では引数リストを記述できません
  ```
  int         foo(int);                 /* ← エラー */
  int         foo2(char *str);          /* ← エラー */
  ```
- 関数定義の引数は ANSI 風に定義します。
  ```
  int func( int a, int b )
  {
      ...
  }
  ```
  関数呼び出しの際の引数の型のチェックを行いませんが、関数の戻り値の型はチェックします。
- 関数内ローカルブロック内では、変数を宣言できません。
  ```
  int func()
  {
      ...
      {
          int x;     /* ← エラー */
      }
  }
  ```
- プリプロセッサでは、引数付きのマクロを展開できません。また、式を定義できません。
  ```
  #define   FUNC(A) A++               /* ← エラー */
  #define   EXP     label + 1         /* ← エラー */
  ```
- プリプロセッサ疑似命令の#if では、オペランドに 0 または 1 のみが指定可能です。

# 5.リファレンス

## 5.1.標準関数(stdlib.lib)

stdlib.lib では、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で使用できる標準的な関数を提供します。

各関数のプロトタイプ宣言は、stdlib.h に記述されています。

| 関数名 | 説明 |
|---|---|
| malloc | ヒープ領域からのメモリの確保 |
| free | malloc で確保された領域の解放 |
| strlen | 文字列の長さの取得 |
| strcat | 文字列の連結 |
| strcmp | 文字列の比較 |
| strcpy | 文字列の複写 |
| strtoi | 文字列の数値への変換 |
| gets | 文字列の入力（Script Window からの入力） |
| exit | プログラム実行の終了 |
| fopen | ファイルのオープン |
| fclose | ファイルのクローズ |
| fseek | ファイルポインタの移動 |
| fgetc | 文字の入力（ファイルからの入力） |
| fputc | 文字の出力（ファイルへの出力） |
| fgets | 文字列の入力（ファイルからの入力） |
| fputs | 文字列の出力（ファイルへの出力） |
| printf | 書式付き出力（Script Window への出力） |
| sprintf | 書式付き出力（メモリへの出力） |
| fprintf | 書式付き出力（ファイルへの出力） |

### 5.1.1.malloc ヒープ領域からのメモリの確保

関数名　char *malloc(int size)
引数　int　size　確保するバイト数
戻り値　char *　確保した領域
　　NULL　エラー
機能　ヒープ領域から size バイトの領域を確保し、先頭アドレスを返します。
　　確保する領域がない場合には、NULL を返します。

### 5.1.2.free: malloc()関数で確保された領域の解放

関数名　int free(char *p)
引数　char　*p　解放する領域
戻り値　0　成功
　　1　エラー
機能　malloc()関数で確保された領域を解放します。

### 5.1.3.strlen: 文字列の長さの取得

関数名　int strlen(char *s)
引数　　char　　*s　　文字列
戻り値　int　　文字列の長さ
機能　　s の長さを返します。

### 5.1.4.strcat: 文字列の連結

関数名　char *strcat(char *s1, char *s2)
引数　　char　　*s1　　追加先
　　char　　*s2　　追加する文字列
戻り値　char *　追加先
機能　　文字列 s2 を s1 の終りに連結し、s1 を返します。

### 5.1.5.strcmp: 文字列の比較

関数名　int strcmp(char *s1, char *s2)
引数　　char　　*s1　　文字列 1
　　char　　*s2　　文字列 2
戻り値　正数　　s1 > s2
　　0　　s1 == s2
　　負数　　s1 < s2
機能　　文字列 s1 と文字列 s1 を比較します。s1 > s2 ならば正数、s1 == s2 ならば 0、s1 < s2 ならば負数を返します。

### 5.1.6.strcpy: 文字列の複写

関数名　char *strcpy(char *s1, char *s2)
引数　　char　　*s1　　複写先
　　char　　*s2　　複写元
戻り値　char *　複写先
機能　　'\0'を含めて文字列 s2 を s1 に複写し、s1 を返します。

### 5.1.7.strtoi: 文字列の数値への変換

関数名　int strtoi(char *str, int radix, int *value)
引数　　char　*str　　文字列
　　int　radix　　変換基数
　　int　*value　　変換後の数値
戻り値　TURE　成功
　　FALSE　エラー
機能　　str で指定された文字列を radix で指定された変換基数の数値として変換します。変換に成功した場合には、変換した値が*value に格納されます。radix には、以下に示す数値が指定可能です。

| radix の値 | 説明 |
|---|---|
| ０ | str が 0x で始まる場合には１６進数として変換し、0 で始まる場合には８進数として変換し、それ以外の場合には１０進数として変換する |
| ８ | str を８進数の数値として変換する |
| １０ | str を１０進数の数値として変換する |
| １６ | str を１６進数の数値として変換する |

### 5.1.8.gets: 文字列の入力(Script Window からの入力)

関数名　char *gets(char *s)
引数　　char　*s　　格納先
戻り値　char *　格納先
　　NULL　エラー
機能　　スクリプトウィンドウの入力領域から一行読み込み s に格納します。行末の改行記号は、'\0'に置換されます。戻り値は格納先 s で、エラーが発生した場合には NULL を返します。

### 5.1.9.exit: 実行の終了

関数名　int exit(int stat)
引数　　int　stat　プログラムの戻り値
戻り値　0　常に 0
機能　　実行を終了し、PD30 に処理を戻します。stat が 0 の場合には、正常に処理されたと解釈されます。stat が非 0 の場合には、何らかの問題が発生したと解釈し、macro_err に設定されている番号のエラーメッセージがスクリプトウィンドウに表示されます。

### 5.1.10.fopen: ファイルのオープン

関数名　int fopen(char *filename, char *attr)
引数　　char　*filename　　ファイル名
　　char　*attr　　オープンモード
戻り値　int　ファイルディスクプリタ
　　NULL　エラー
機能　　filename で指定されたファイルを attr で指定されたモードでオープンします。成功した場合は、ファイルディスクプリタ値を返します。

### 5.1.11.fclose: ファイルのクローズ

関数名　int fclose(int fd)
引数　int　fd　ファイルディスクプリタ
戻り値　TRUE　成功
FALSE　エラー
機能　fd で指定されたファイルをクローズします。

### 5.1.12.fseek: ファイルポインタの移動

関数名　int fseek(int fd, int pos, int org)
引数　int　fd　ファイルディスクプリタ
int　pos　ファイルポインタの移動量
int　org　pos の基点
戻り値　TRUE　成功
FALSE　エラー
機能　fd で指定されたファイルの読み書き対象の現在位置を移動します。移動量 pos には、基点 org(０: ファイル先頭、１: 現在位置、２: ファイル末尾)からのオフセットを指定します。

### 5.1.13.fgetc: 文字の入力(ファイルからの入力)

関数名　int fgetc(int fd)
引数　int　fd　ファイルディスクプリタ
戻り値　int　入力文字
機能　fd で指定されたファイルのファイルポインタの現在位置から一バイトを読み込みます。

### 5.1.14.fputc: 文字の出力(ファイルへの出力)

関数名　int fputc(char c, int fd)
引数　char　c　出力文字
int　fd　ファイルディスクプリタ
戻り値　TURE　成功
FALSE　エラー
機能　fd で指定されたファイルのファイルポインタの現在位置に c で指定された一バイトを出力します。

### 5.1.15.fgets: 文字列の入力(ファイルからの入力)

関数名　char *fgets (char *str, int n, int fd)
引数　char　*str　入力文字列格納領域
int　n　入力最大文字数
int　fd　ファイルディスクプリタ
戻り値　char*　入力文字列格納領域
NULL　エラー
機能　fd で指定されたファイルのファイルポインタの現在位置から一行を読み込み str からの領域に格納します。

### 5.1.16.fputs: 文字列の出力(ファイルへの出力)

関数名　int fputs (char *str, int fd)
引数　char　*str　出力文字列格納領域
　　int　fd　ファイルディスクプリタ
戻り値　TURE　成功
　　FALSE　エラー
機能　fd で指定されたファイルのファイルポインタの現在位置から str からの域に格納された文字列を出力します。

### 5.1.17.printf: 書式付き出力(Script Window への出力)

関数名　int printf(char *format, ...)
引数　char　*format フォーマット
　　...　可変引数
戻り値　正数　出力された文字数
　　負数　エラー
機能　format の制御のもとに出力を変換し、スクリプトウィンドウへ出力します。戻り値は書き出された文字の数で、エラーが発生した場合には負の数を返します。

### 5.1.18.sprintf: 書式付き出力(メモリへの出力)

関数名　int sprintf(char *s, char *format, ...)
引数　char　*s　出力先
　　char　*format フォーマット
　　...　可変引数
戻り値　正数　出力された文字数
　　負数　エラー
機能　format の制御のもとに出力を変換し、s で指定されたアドレスからへ出力します。出力の最後に'\0'が付伽されます戻り値は書き出された文字の数で('\0'は含まない)、エラーが発生した場合には負の数を返します。

### 5.1.19.fprintf: 書式付き出力(ファイルへの出力)

関数名　int fprintf(int fd, char *format, ...)
引数　int　fd　ファイルディスクプリタ
　　char　*format フォーマット
　　...　可変引数
戻り値　正数　出力された文字数
　　負数　エラー
機能　format の制御のもとに出力を換し fd で指定されたファイルへ出力します。戻り値は書き出された文字の数で、エラーが発生した場合には負の数を返します。

## 5.2.デバッガ操作系システムコール関数(system.lib)

system.lib には、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で使用できるシステムコール関数を提供します。

各関数のプロトタイプ宣言は、system.h に記述されています。

| 関数名 | 説明 |
|---|---|
| _cpu_go | フリーラン実行 |
| _cpu_gb | ブレーク付き実行 |
| _cpu_stop | 実行停止 |
| _cpu_reset | リセット |
| _cpu_src_step | ソース行ステップ実行 |
| _cpu_step | １命令ステップ実行 |
| _cpu_src_over | ソース行オーバー実行 |
| _cpu_over | １命令オーバー実行 |
| _cpu_src_return | ソース行リターン実行 |
| _cpu_return | １命令リターン実行 |
| _cpu_wait | 実行停止待ち |
| _reg_get_reg | レジスタ値取得 |
| _reg_put_reg | レジスタ値設定 |
| _reg_get_pc | プログラムカウンタ値取得 |
| _reg_put_pc | プログラムカウンタ値設定 |
| _reg_clear_cache | レジスタキャッシュのクリア |
| _mem_get | メモリ値取得 |
| _mem_put | メモリ値設定 |
| _mem_get_endian | エンディアン付きメモリ値取得 |
| _mem_put_endian | エンディアン付きメモリ値設定 |
| _mem_fill | メモリ充填 |
| _mem_move | メモリブロック転送 |
| _mem_clear_cache | メモリキャッシュのクリア |
| _break_set | ソフトウェアブレーク設定/有効化 |
| _break_get | ソフトウェアブレーク設定取得 |
| _break_reset | ソフトウェアブレーク解除 |
| _break_reset_all | 全ソフトウェアブレーク解除 |
| _break_disable | ソフトウェアブレーク無効化 |
| _break_disable_all | 全ソフトウェアブレーク無効化 |
| _break_enable_all | 全ソフトウェアブレーク有効化 |
| _break_search | ソフトウェアブレーク設定属性の取得 |
| _rram_clear | RAM モニタメモリクリア |
| _rram_get_area | RAM モニタ領域取得 |
| _rram_set_area | RAM モニタ領域設定 |
| _rram_get_size | RAM モニタ領域サイズ取得 |
| _rram_get_data | RAM モニタデータ取得 |
| _info_check_run | 実行検出 |
| _info_service | サービス内容取得 |

| 関数名(つづき) | 説明 |
|---|---|
| _info_cpu | CPU 情報取得 |
| _info_get_map | マップ情報取得 |
| _info_check_map | マップ内検査 |
| _info_get_suffix | ロードファイル拡張子取得 |
| _info_set_suffix | ロードファイル拡張子設定 |
| _scope_set_obj | オブジェクトファイル名によるスコープ設定 |
| _scope_set_addr | アドレスによるスコープ設定 |
| _sym_addr_sym | シンボル登録 |
| _sym_val2sym | 値のシンボル取得 |
| _sym_sym2val | シンボルの値取得 |
| _sym_addr_bit | ビットシンボル登録 |
| _sym_val2bit | アドレス:ビット値のビットシンボル取得 |
| _sym_bit2val | ビットシンボルのアドレス:ビット値取得 |
| _line_addr2line | アドレスのソース行取得 |
| _line_line2addr | ソース行のアドレス取得 |
| _src_get_name | ソースファイル名一覧取得 |
| _obj_get_name | オブジェクトファイル名一覧取得 |
| _obj_addr2obj | アドレスによるオブジェクトファイル名取得 |
| _func_get_name | 関数名取得 |
| _exp_eval | アセンブラ式解析 |
| _com_send | エミュレータへのバイト列転送 |
| _com_receive | エミュレータからのバイト列受信 |
| _scri_echo_on | スクリプトウィンドウへの出力許可 |
| _scri_echo_off | スクリプトウィンドウへの出力不許可 |
| _c_exp_eval | C 言語式解析 |
| _get_shared_mem | 共有変数の取得 |
| _set_shared_mem | 共有変数の設定 |
| _delete_shared_mem | 共有変数の削除 |
| _get_err_msg | PD30 のエラーメッセージ文の取得 |
| _get_tick_count | Windows 起動からの経過した時間の取得 |
| _get_time | システムの現在の日付と時刻の取得 |
| _disp_src_line | プログラムウィンドウの表示内容変更 |
| _rtt_get_range | RTT データ範囲の取得 |
| _rtt_get_disasm | RTT データの逆アセンブル解析結果の取得 |
| _rtt_get_bus | RTT データのバスモード表示文字列の取得 |
| _rtt_check_isfetch | RTT データのフェッチサイクルの検査 |
| _rtt_get_data | RTT データの取得 |
| _rtt_clear_cache | リアルタイムトレース(RTT)キャッシュのクリア |
| _cv_get_data | カバレッジデータの取得 |
| _cv_set_data | カバレッジデータの設定 |
| _cv_get_base | カバレッジ範囲のベースアドレスの取得 |
| _cv_set_base | カバレッジ範囲のベースアドレスの設定 |

| 関数名(つづき) | 説明 |
|---|---|
| _cv_clear_data | カバレッジデータのクリア |
| _cv_clear_cache | カバレッジキャッシュのクリア |
| _syscom | PD30 のスクリプトコマンドの実行 |
| _doscom | DOS コマンドの実行 |

エラーが発生した場合には、グローバル変数 macro_err に「エラー」の項目に記述されているエラー番号が設定されます。エミュレータエラーについては、66ページの「5.2.86 エミュレータエラー一覧」を参照してください。カスタムコマンドプログラムの場合には、main()関数から FALSE を戻り値として返した場合には、macro_err に設定されているエラー番号に対応するエラーメッセージがスクリプトウィンドウ(PD30 の場合)、またはエラーダイアログ(CB30 の場合)に表示されます。

### 5.2.1._cpu_go: フリーラン実行

関数名　int _cpu_go()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　現 PC よりターゲットプログラムのフリーラン実行を開始します。
エラー　エミュレータエラー

### 5.2.2._cpu_gb: ブレーク付き実行

関数名　int _cpu_gb()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　現 PC よりターゲットプログラムのブレーク付き実行を開始します。
エラー　エミュレータエラー

### 5.2.3._cpu_stop: 実行停止

関数名　int _cpu_stop()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　ターゲットプログラムの実行を停止します。
エラー　エミュレータエラー

### 5.2.4._cpu_reset: リセット

関数名　int _cpu_reset()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　ターゲット CPU をリセットします。
エラー　ER_IN1_RUNNING　実行中であるためにリセットできない
　　その他　エミュレータエラー

### 5.2.5._cpu_src_step: ソース行ステップ実行

関数名　int _cpu_src_step()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　現 PC よりソース行ステップ実行を開始します。
エラー　エミュレータエラー

### 5.2.6._cpu_step: 1 命令ステップ実行

関数名　int _cpu_step()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　現 PC より 1 命令ステップ実行を開始します。
エラー　エミュレータエラー

### 5.2.7._cpu_src_over: ソース行オーバー実行

関数名　int _cpu_src_over()
引数　なし
戻り値　TRUE　成功
　　FALSE　エラー
機能　現 PC よりソース行オーバー実行を開始します。
エラー　ER_IN1_RUNNING　すでに実行中である
　　ER_IN1_CANCEL　実行中断
　　その他　エミュレータエラー

### 5.2.8._cpu_over: 1命令オーバー実行

関数名　int _cpu_over()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　現PCより1命令オーバー実行を開始します。
エラー　エミュレータエラー

### 5.2.9._cpu_src_return: ソース行リターン実行

関数名　int _cpu_src_return()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　現PCよりソース行リターン実行を開始します。
エラー　エミュレータエラー

### 5.2.10._cpu_return: 1命令リターン実行

関数名　int _cpu_return()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　現PCより1命令オーバー実行を開始します。
エラー　エミュレータエラー

### 5.2.11._cpu_wait: 実行停止待ち

関数名　int _cpu_wait()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　ターゲットプログラムが停止するまで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実行を停止します。
エラー　エミュレータエラー

### 5.2.12._reg_get_reg: レジスタ値取得

関数名　int _reg_get_reg(int *reg, int regno)
引数　　int　　*reg　　レジスタ値
int　　regno　　レジスタ番号
戻り値　TRUE　　成功
FALSE　　エラー
機能　　regno で指定されたレジスタの値を取得します。M16C では、regno は以下のように定義されています。

| regno | レジスタ |
|---|---|
| IN1_REG_0_Rx | 表 Rx レジスタ(x は 0～3) |
| IN1_REG_0_Ax | 表 Ax レジスタ(x は 0～1) |
| IN1_REG_0_FB | 表 FB レジスタ |
| IN1_REG_1_Rx | 裏 Rx レジスタ(x は 0～3) |
| IN1_REG_1_Ax | 裏 Ax レジスタ(x は 0～1) |
| IN1_REG_1_FB | 裏 FB レジスタ |
| IN1_REG_USP | USP レジスタ |
| IN1_REG_ISP | ISP レジスタ |
| IN1_REG_FLG | FLG レジスタ |
| IN1_REG_PC | プログラムカウンタ |
| IN1_REG_SB | SB レジスタ |
| IN1_REG_INTB | INTB レジスタ |

エラー　エミュレータエラー

### 5.2.13._reg_put_reg: レジスタ値設定

関数名　int _reg_put_reg(int reg, int regno)
引数　　int　　reg　　レジスタ値
戻り値　TRUE　　成功
FALSE　　エラー
機能　　regno で指定されたレジスタの値を設定します。regno の定義は、_reg_get_reg()関数と同様です。
エラー　ER_IN1_DATA_OUTRANGE　　データ範囲が不正
その他　　エミュレータエラー

### 5.2.14._reg_get_pc: プログラムカウンタ値取得

関数名　int _reg_get_pc(int *pc)
引数　　int　　*pc　　プログラムカウンタ値
戻り値　TRUE　　成功
FALSE　　エラー
機能　　プログラムカウンタ値を取得します。
エラー　エミュレータエラー

### 5.2.15._reg_put_pc: プログラムカウンタ値設定

関数名　int _reg_put_pc(int pc)
引数　　int　　pc　　プログラムカウンタ値
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　プログラムカウンタ値を設定します。
エラー　ER_IN1_ADDR_OUTRANGE　　アドレス範囲が不正
　　その他　　エミュレータエラー

### 5.2.16._reg_clear_cache: レジスタキャッシュのクリア

関数名　int _reg_clear_cache()
引数　　なし
戻り値　TRUE　　常に TRUE を返す
機能　　レジスタキャッシュをクリアします。

### 5.2.17._mem_get: メモリ値取得

関数名　int _mem_get(int addr, int size, char *data)
引数　　int　　addr　　アドレス
　　int　　size　　取得バイト数
　　char　　*data　　取得データ格納先
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　addr から size バイトのデータを data に格納します。
エラー　ER_IN1_ADDR_OUTRANGE　　アドレス範囲が不正
　　その他　　エミュレータエラー

### 5.2.18._mem_put: メモリ値設定

関数名　int _mem_put(int addr, int size, char *data)
引数　　int　　addr　　アドレス
　　int　　size　　設定バイト数
　　char　　*data　　設定データ
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　データ data を addr から size バイトのメモリに設定します。
エラー　ER_IN1_ADDR_OUTRANGE　　アドレス範囲が不正
　　その他　　エミュレータエラー

### 5.2.19._mem_get_endian: エンディアン付きメモリ値取得

関数名　int _mem_get_endian(int addr, int num, int size, int *data)
引数　int　addr　アドレス
int　num　データ数
int　size　1データのサイズ
int　*data　取得データ格納先
戻り値　TRUE　成功
FALSE　エラー
機能　addrからデータサイズsizeバイトのnum個のデータをCPUのエンディアンに従ってdata[]に格納します。sizeには1～4が指定可能です。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
ER_IN1_DATA_RANGE　sizeが1～4以外
その他　エミュレータエラー

### 5.2.20._mem_put_endian: エンディアン付きメモリ値設定

関数名　int _mem_put_endian(int addr, int num, int size, int *data)
引数　int　addr　アドレス
int　num　データ数
int　size　1データのサイズ
int　*data　設定データ
戻り値　TRUE　成功
FALSE　エラー
機能　データdata[]に格納されているデータサイズsizeバイトのnum個のデータをCPUのエンディアンに従ってaddrからのメモリに設定します。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
ER_IN1_DATA_RANGE　sizeが1～4以外
その他　エミュレータエラー

### 5.2.21._mem_fill: メモリ充填

関数名　int _mem_fill(int start, int end, int data, int size)
引数　int　start　開始アドレス
int　end　終了アドレス
int　data　充填データ
int　size　1データのサイズ
戻り値　TRUE　成功
FALSE　エラー
機能　start～endまでをデータサイズsizeバイトのデータdataで充填します。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
ER_IN1_DATA_RANGE　sizeが1～4以外
その他　エミュレータエラー

### 5.2.22._mem_move: メモリブロック転送

関数名　int _mem_move(int start, int end, int top)
引数　int　start　開始アドレス
　　int　end　終了アドレス
　　int　top　転送先先頭アドレス
戻り値　TRUE　成功
　　FALSE　エラー
機能　start～end までを top からの領域にに転送します。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
　　ER_IN1_DATA_RANGE　size が 1～4 以外
　　その他　エミュレータエラー

### 5.2.23._mem_clear_cache: メモリキャッシュのクリア

関数名　int _mem_clear_cache()
引数　なし
戻り値　TRUE　常に TRUE を返す。
機能　キャッシング付きメモリ取得モジュールのキャッシュバッファをクリアします。

### 5.2.24._break_set: ソフトウェアブレーク設定

関数名　int _break_set(int addr)
引数　int　addr　設定アドレス
戻り値　TRUE　成功
　　FALSE　エラー
機能　addr にソフトウェアブレークポイントを設定します。_break_disable()、_break_disable_all()で無効化したブレークポイントの復帰も本関数で行ないます。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
　　ER_IN1_BP_FULL　ブレークポイントが満杯
　　その他　エミュレータエラー

### 5.2.25._break_get: ソフトウェアブレーク設定取得

関数名　int _break_get(int *addr, int *attr, int mode)
引数　　int　　*addr　アドレス
　　int　　*attr　設定属性
　　int　　mode　検索開始モード
　　IN1_FIRST : 1点目のブレークポイント
　　IN1_NEXT : 2点目以降のブレークポイント
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　*addr にブレークポイントのアドレスが格納されます。*attr には、以下に示すブレークポイントの設定属性が格納されます。

| | |
|---|---|
| IN1_ENABLE_SBRK | 有効 |
| IN1_DISABLE_SBRK | 無効 |

エラー　ER_IN1_BP_NOTFOUND　　ブレークポイントが見つからない
　　その他　　エミュレータエラー

### 5.2.26._break_reset: ソフトウェアブレーク解除

関数名　int _break_reset(int addr)
引数　　int　　addr　　アドレス
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　addr のブレークポイントを解除します。
エラー　ER_IN1_ADDR_OUTRANGE　　アドレス範囲が不正
　　ER_IN1_BP_NOTFOUND　　ブレークポイントが見つからない
　　その他　　エミュレータエラー

### 5.2.27._break_reset_all: 全ソフトウェアブレーク解除

関数名　int _break_reset_all()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　エラー
機能　　全ブレークポイントを解除します。
エラー　その他　　エミュレータエラー

### 5.2.28._break_disable: ソフトウェアブレーク無効化

関数名　int _break_disable(int addr)
引数　　int　　addr　　アドレス
戻り値　TRUE　　成功
　　FALSE　　エラー
機能　　addr のブレークポイントを無効化します。
エラー　ER_IN1_ADDR_OUTRANGE　　アドレス範囲が不正
　　ER_IN1_BP_NOTFOUND　　ブレークポイントが見つからない
　　その他　　エミュレータエラー

### 5.2.29._break_disable_all: 全ソフトウェアブレーク無効化

関数名　int _break_disable_all()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　　エラー
機能　　全ブレークポイントを無効化します。
エラー　エミュレータエラー

### 5.2.30._break_enable_all: 全ソフトウェアブレーク有効化

関数名　int _break_enable_all()
引数　　なし
戻り値　TRUE　　成功
　　FALSE　　エラー
機能　　全ブレークポイントを有効化します。
エラー　エミュレータエラー

### 5.2.31._break_search: ソフトウェアブレークポイントの設定属性取得

関数名　int _break_search(int addr, int *attr)
引数　　int　　addr　　アドレス
　　int　　*attr　　設定属性
戻り値　TRUE　　成功
　　FALSE　　エラー
機能　　addr のブレークポイントの設定属性を取得します。*attr に以下のブレークポイントの設定属性が格納されます。

| | |
|---|---|
| IN1_ENABLE_SBRK | 有効 |
| IN1_DISABLE_SBRK | 無効 |

エラー　ER_INR1_ADDR_OUTRANGE　　アドレス範囲が不正
　　その他　　エミュレータエラー

### 5.2.32._rram_clear: RAM モニタメモリクリア

関数名　int _rram_clear()
引数　なし
戻り値　TRUE　成功
FALSE　エラー
機能　RAM モニタメモリのアクセス状態を初期化します。
エラー　ER_IN1_RUNNING　実行中であるため設定できない
その他　エミュレータエラー

### 5.2.33._rram_get_area: RAM モニタ領域取得

関数名　int _rram_get_area(int *addr)
引数　int　*addr　先頭アドレス
戻り値　TRUE　成功
FALSE　エラー
機能　RAM モニタメモリの先頭アドレスを*addr に格納します。
エラー　エミュレータエラー

### 5.2.34._rram_set_area: RAM モニタ領域設定

関数名　int _rram_set_area(int addr)
引数　int　addr　先頭アドレス
戻り値　TRUE　成功
FALSE　エラー
機能　RAM モニタメモリの先頭アドレスを addr に設定します。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
ER_IN1_RUNNING　実行中であるため設定できない
その他　エミュレータエラー

### 5.2.35._rram_get_size: RAM モニタ領域サイズ取得

関数名　int _rram_get_size(int *size)
引数　int　*size　サイズ
戻り値　TRUE　常に TURE を返す
機能　RAM モニタメモリのサイズを*size に設定します。

### 5.2.36._rram_get_data: RAM モニタデータ取得

関数名　int _rram_get_data(int addr, int size, char *data, char *attr)

引数　int　addr　先頭アドレス
　　int　size　バイト数
　　char　*data　データ
　　char　*attr　アクセス状態

戻り値　TRUE　成功
　　FALSE　エラー

機能　addr から size バイトのデータ(*data)、およびアクセス状態(*attr)を RAM モニタから取得します。*attr には、以下に示すアクセス状態が格納されます。

| | |
|---|---|
| IN1_RRAM_READ | リード |
| IN1_RRAM_WRITE | ライト |
| IN1_RRAM_NONE | アクセスなし |

エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
　　その他　エミュレータエラー

### 5.2.37._info_check_run: 実行検出

関数名　int _info_check_run(int *status)

引数　int　*status 実行状態

戻り値　TRUE　成功
　　FALSE　エラー

機能　ターゲットプログラムの実行状態を*status に格納します。*status には、以下に示す実行状態が格納されます。

| | |
|---|---|
| IN1_RUN_CPU | 実行中 |
| IN1_STOP_CPU | 停止中 |

エラー　エミュレータエラー

### 5.2.38._info_service: サービス内容取得

関数名　int _info_service(int flag, int *status)

引数　int　flag　サービス内容
　　int　*status サポートの有無
　　TRUE　サポートあり
　　FALSE　サポートなし

戻り値　TRUE　常に TRUE を返す

機能　PD30 がサポートするサービス内容を取得します。flag には、以下に示すサービス内容を指定します。

| | |
|---|---|
| IN1_SUPPORT_BITSYM | ビットシンボルのサポート |
| IN1_SUPPORT_C | C言語デバッグのサポート |
| IN1_SUPPORT_RAMMONITOR | リアルタイム RAM モニタ機能のサポート |
| IN1_SUPPORT_RTT | リアルタイムトレースのサポート |
| IN1_SUPPORT_CV | カバレッジ機能のサポート |
| IN1_SUPPORT_PROTCT | プロテクトブレークのサポート |
| IN1_SUPPORT_EVENT | ハードウェアイベントのサポート |

### 5.2.39._info_cpu: CPU 情報取得

関数名　int _info_cpu(int flag, int *status)

引数　int　flag　情報内容

int　*status CPU 情報

IN1_BIG_ENDIAN　ビッグエンディアン

IN1_LITTLE_ENDIAN　リトルエンディアン

その他　flag に応じた値

戻り値　TRUE　常に TRUE を返す

機能　ターゲット CPU の情報を取得します。flag には、以下に示す情報内容を指定します。

| | |
|---|---|
| IN1_ADDRSIZE | アドレス値を格納するのに必要なバイト数 |
| IN1_MAXADDR | アドレスの最大値 |
| IN1_ADDRCOLM | アドレス値を 16 進で表示する際の表示桁数 |
| IN1_ENDIAN | ターゲット CPU のエンディアン |
| IN1_HWORD_SIZE | ハーフワードのバイト長 |
| IN1_WORD_SIZE | ワードのバイト長 |
| IN1_DWORD_SIZE | ダブルワードのバイト長 |
| IN1_LWORD_SIZE | ダブルワードのバイト長 |
| IN1_MAXDATA | データの最大値 |
| IN1_MAXSTACK | スタックの最大値 |
| IN1_MAX_OBJ | 1 命令長の最大バイト数 |

### 5.2.40._info_get_map: マップ情報取得

関数名　int _info_get_map(int *start, int *end, int mode)

引数　int　*start　先頭アドレス

int　*end　最終アドレス

int　mode　検索開始モード

IN1_FIRST : 1 つ目のマップ

IN1_NEXT : 2 つ目以降のマップ

戻り値　TRUE　成功

FALSE　エラー

機能　マップ情報を取得する。*start、*end には、マップの先頭アドレス、最終アドレスが格納されます。

エラー　IN1_MAP_END　これ以上マップはない

### 5.2.41._info_check_map: マップ内検査

関数名　int _info_check_map(int start, int end, int *status, int *erradr)

引数　int　start　先頭アドレス
　　int　end　最終アドレス
　　int　*status 検査結果
　　int　*erraddr　エラーアドレス

戻り値　TRUE　成功
　　FALSE　エラー

機能　start～end のアドレス範囲が、マッピングされた領域か否かを検査します。start～end のアドレス範囲が、全てマッピングされた領域である場合には、*status に TRUE を格納します。start～end のアドレス範囲に、マッピングされていない領域が存在する場合には、*status に FALSE を格納し、start から検索して一番最初に見つかたマッピングされていないアドレスを*erraddr に格納します。

エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正
　　その他　エミュレータエラー

### 5.2.42._info_get_suffix: ロードファイル拡張子取得

関数名　int _info_get_suffix(char *suffix, int mode)

引数　char　*suffix 取得した拡張子
　　int　mode　モード

戻り値　TRUE　常に TRUE を返す

機能　mode で指定されたモードでターゲットプログラムをダウンロードしようとした時に付加するファイルセレクションダイアログのフィルターサフィックスを取得します。mode には、以下に示す属性を指定します。

| | |
|---|---|
| IN1_LOAD | シンボルファイルおよびプログラムファイル |
| IN1_SYM | シンボルファイル |
| IN1_ROM | プログラムファイル |

### 5.2.43._info_set_suffix: ロードファイル拡張子設定

関数名　int _info_set_suffix(char *suffix, int mode)

引数　char　*suffix 設定する拡張子
　　int　mode　モード

戻り値　TRUE　常に TRUE を返す

機能　mode で指定されたモードでターゲットプログラムをダウンロードしようとした時に付加するファイルセレクションダイアログのフィルターサフィックスを設定します。mode には、以下に示す属性を指定します。

| | |
|---|---|
| IN1_LOAD | シンボルファイルおよびプログラムファイル |
| IN1_SYM | シンボルファイル |
| IN1_ROM | プログラムファイル |

### 5.2.44._info_ispc4700h: 接続エミュレータの確認

関数名　int _info_ispc4700h()
引数　なし
戻り値　TRUE　接続エミュレータが PC4700H または PC4701HS
FALSE 接続エミュレータが上記以外
機能　接続エミュレータの種類を取得します。カバレッジ機能、リアルタイムトレース機能は、エミュレータ PC4700H または PC4701HS でのみ使用可能です。_cv_で始まる関数と、_rtt_で始まる関数を使用する場合には、本関数を用いて接続エミュレータが PC4700H または PC4701HS であることを確認する必要があります。

### 5.2.45._scope_set_obj: オブジェクトファイル名によるスコープ設定

関数名　int _scope_set_obj(char *name)
引数　char　*name　オブジェクトファイル名
戻り値　TRUE　成功
FALSE　エラー
機能　オブジェクトファイル名によって、現在のスコープを設定します。
エラー　ER_SCOPE_NOTFOUND　指定したオブジェクトファイル名に該当するスコープがない

### 5.2.46._scope_set_addr: アドレスによるスコープ設定

関数名　int _scope_set_addr(int addr)
引数　int　addr　アドレス
戻り値　TRUE　成功
FALSE　エラー
機能　アドレスによって、現在のスコープを設定します。
エラー　ER_IN1_ADDR_OUTRANGE　アドレス範囲が不正

### 5.2.47._sym_add_sym: シンボル登録

関数名　int _sym_add_sym(int mode, char *name, int value)
引数　int　mode　検索モード
char　*name　シンボル
int　value　値
戻り値　TRUE　成功
FALSE　エラー
機能　シンボル(またはラベル)name をグローバルシンボル(またはラベル)として登録します。mode には、以下に示す種別を指定します。

| | |
|---|---|
| LOAD_SYMBOL | シンボル |
| LOAD_LABEL | ラベル |

エラー　ER_LOAD_ILLEGAL_CHAR　シンボルやラベルとして記述できない文字が文字列中に含まれている
ER_LOAD_MULTIDEFINE　すでに同名のグローバルシンボル(またはラベル)が存在する

### 5.2.48._sym_val2sym: 値のシンボル取得

関数名　int _sym_val2sym(int mode, int value, char *symbol)

引数　int　mode　検索モード
　　int　value　値
　　char　*symbol シンボル格納領域

戻り値　TRUE　成功
　　FALSE　対応するシンボルが発見できなかった

機能　値 value に対応するシンボル文字列を検索し、symbol に格納します。mode には、以下に示す検索の優先順位を指定します。

| | |
|---|---|
| LOAD_SYMBOL | シンボル優先 |
| LOAD_LABEL | ラベル優先 |

各モードの検索の優先順位は、以下の通りです。

| | シンボル優先の場合 | | ラベル優先の場合 |
|---|---|---|---|
| 1 | ローカルシンボル<br>(スコープ範囲内) | 1 | ローカルラベル<br>(スコープ範囲内) |
| 2 | グローバルシンボル | 2 | グローバルラベル |
| 3 | ローカルラベル<br>(スコープ範囲内) | 3 | ローカルシンボル<br>(スコープ範囲内) |
| 4 | グローバルラベル | 4 | グローバルシンボル |
| 5 | ローカルシンボル<br>(スコープ範囲外) | 5 | ローカルラベル<br>(スコープ範囲外) |
| 6 | ローカルラベル<br>(スコープ範囲外) | 6 | ローカルシンボル<br>(スコープ範囲外) |

### 5.2.49._sym_sym2val: シンボルの値取得

関数名　int _sym_sym2val(int mode, char *symbol, int *value)

引数　int　mode　検索モード
　　char　*symbol シンボル
　　int　*value　値

戻り値　TRUE　成功
　　FALSE　シンボルが発見できなかった

機能　シンボル文字列 symbol に対応する値を検索し、*value に格納します。mode は、_sym_val2sym()と同様です。

エラー　ER_LOAD_SYMBOL_NOTFOUND　　シンボルが見つからない

### 5.2.50._sym_add_bit: ビットシンボル登録

関数名　int _sym_add_bit(char *bitsym, int addr, int bit)
引数　char　*bitsym ビットシンボル
　　int　addr　アドレス
　　int　bit　ビット
戻り値　TRUE　成功
　　FALSE　エラー
機能　addr で指定されたアドレスの bit で指定されたビットに、bitsym で指定されたビットシンボルを登録します。
エラー　ER_LOAD_ILLEGAL_CHAR　ビットシンボルとして記述できない文字が文字列中に含まれている
　　ER_LOAD_MULTIDEFINE　すでに同名のグローバルビットシンボルが存在する
　　ER_LOAD_ADDR_OUTRANGE　指定したアドレスが範囲外
　　ER_LOAD_BIT_OUTRANGE　指定したビット番号が範囲外

### 5.2.51._sym_val2bit: アドレス:ビット値のビットシンボル取得

関数名　int _sym_val2bit(int addr, int bit, char *bitsym)
引数　int　addr　アドレス
　　int　bit　ビット
　　char　*bitsym ビットシンボル格納領域
戻り値　TRUE　成功
　　FALSE　対応するビットシンボルが発見できなかった
機能　addr で指定されたアドレスの bit で指定されたビットに対応するビットシンボル文字列を検索し、bitsym に格納します。

### 5.2.52._sym_bit2val: ビットシンボルのアドレス:ビット値取得

関数名　int _sym_sym2val(char *bitsym, int *addr, int *bit)
引数　char　*bitsym ビットシンボル
　　int　*addr　アドレス格納領域
　　int　*bit　ビット格納領域
戻り値　TRUE　成功
　　FALSE　ビットシンボルが発見できなかった
機能　bitsym で指定されたビットシンボルに対応するアドレス、ビットを検索し、それぞれ*addr、*bit に格納します。
エラー　ER_LOAD_SYMBOL_NOTFOUND 指定したビットシンボルが見つからない

### 5.2.53._line_addr2line: アドレスのソース行取得

関数名　int _sym_addr2line(int addr, int *line, char *filename)
引数　int　addr　アドレス
int　*line　行番号
char　*filename　ファイル名格納領域
戻り値　TRUE　成功
FALSE　ソース行情報が見つからない
機能　アドレス addr に対応する行番号(*line)、およびそのファイル名(filename)を取得します。
エラー　ER_LOAD_FILE_NOTFOUND　ファイルが見つからない
ER_LOAD_SRCINF_NOTFOUND　ソース情報が見つからない

### 5.2.54._line_line2addr: ソース行のアドレス取得

関数名　int _sym_line2addr(char *filename, int line, int *addr)
引数　char　*filename　ファイル名
int　line　行番号
int　*addr　アドレス
戻り値　TRUE　成功
FALSE　ソース行情報が見つからない
機能　ファイル(filename)の行(line)に対応するアドレス(*addr)を取得します。
エラー　ER_LOAD_LINE_NOTFOUND　行情報が見つからない

### 5.2.55._src_get_name: ソースファイル名一覧取得

関数名　int _src_get_name(char *objname, char *srcname, int mode)
引数　char　*objname　オブジェクトファイル名
char　*srcname　ソースファイル名格納領域
int　mode　検索開始モード
LOAD_FIRST : 1つ目のソースファイル名
LOAD_NEXT : 2つ目以降のソースファイル名
戻り値　TRUE　成功
FALSE　ソースファイル名が見つからない
機能　オブジェクトファイル objname 内のソースファイル名一覧を取得します。objname に NULL を与えた場合には、全てのオブジェクトファイルに対するソースファイル名の一覧を取得します。

### 5.2.56._obj_get_name: オブジェクト名一覧取得

関数名　int _obj_get_name(char *objname, int mode)
引数　char　*objname　オブジェクトファイル名格納領域
int　mode　検索開始モード
LOAD_FIRST : 1つ目のソースファイル名
LOAD_NEXT : 2つ目以降のソースファイル名
戻り値　TRUE　成功
FALSE　オブジェクトファイル名が見つからない
機能　オブジェクトファイル名一覧を取得します。

### 5.2.57._obj_addr2obj: アドレスによるオブジェクトファイル名取得

関数名　int _obj_addr2obj(int addr, char *objname)

引数

| | | |
|---|---|---|
| int | addr | アドレス |
| char | *objname | オブジェクトファイル名格納領域 |

戻り値

| | |
|---|---|
| TRUE | 成功 |
| FALSE | 対応するオブジェクトファイル名が見つからない |

機能　アドレス addr を含むオブジェクトファイル名 objname を取得します。

### 5.2.58._func_get_name: 関数名一覧取得

関数名　int _func_get_name(char *objname, char *funcname, int mode)

引数

| | | |
|---|---|---|
| char | *objname | オブジェクトファイル名 |
| char | *funcname | 関数名格納領域 |
| int | mode | 検索開始モード<br>LOAD_FIRST : 1つ目のソースファイル名<br>LOAD_NEXT : 2つ目以降のソースファイル名 |

戻り値

| | |
|---|---|
| TRUE | 成功 |
| FALSE | 関数名が見つからない |

機能　オブジェクトファイル objname 内の関数名一覧を取得します。objname に NULL を与えた場合には、全てのオブジェクトファイルに対する関数名の一覧を取得します。

### 5.2.59._exp_eval: アセンブラ式解析

関数名　int _exp_eval(char *exp, int radix, int mode, int *value)

引数　char　*exp　アセンブラ式
　　int　radix　基数
　　int　mode　シンボル(ラベル)の評価優先順位
　　int　*value　解析結果格納領域

戻り値　TRUE　成功
　　FALSE　エラー

機能　アセンブラ式(exp)を解析し、その結果を*value に格納します。radix には、以下に示す定数の基数を指定します。

| | |
|---|---|
| EXP_DEC | 10 進数 |
| EXP_HEX | 16 進数 |
| EXP_DEFAULT | RADIX コマンドでの設定値を使用 |

mode には、以下に示すシンボル(ラベル)の評価優先順位を指定します。

| | |
|---|---|
| EXP_SYMBOL | シンボル |
| EXP_LABEL | ラベル |

エラー

| | |
|---|---|
| ER_EXP_SYNTAX | 文法エラー |
| ER_EXP_ZERO | 0除算エラー |
| ER_EXP_LPAR | 左括弧がない |
| ER_EXP_SIZE | サイズ指定子が間違っている |
| ER_EXP_STRING | 文字列が終了していない |
| ER_EXP_LINE | 行番号の指定が間違っている |
| ER_EXP_VALUE | 定数値の指定が間違っている |
| ER_EXP_UNDEF_SYMBOL | シボルが定義されていない |
| ER_EXP_PREFIX | 定数の基数指定が間違っている |
| ER_EXP_OVER | 定数値が範囲外 |
| ER_EXP_UNDEF_MACRO | マクロ定数が定義されていない |
| ER_EXP_ILLEGAL_MACRO | マクロ変数名にレジスタ名を使用している |
| ER_EXP_OUTOF_MACRO | マクロ定数の個数が制限数を超えている |

### 5.2.60._com_send: エミュレータへのバイト列転送

関数名　int _com_send(char *data, int size)

引数　char　*data　バイト列
　　int　size　転送バイト数

戻り値　TRUE　成功
　　FALSE　エラー

機能　バイト列 data を size バイト転送します。

エラー　エミュレータエラー

### 5.2.61._com_receive: エミュレータからのバイト列受信

関数名　int _com_receive(char *data, int size)
引数　　char　　*data　　バイト列
　　int　　size　　受信バイト数
戻り値　TRUE　　成功
　　FALSE　　エラー
機能　　size バイトのデータを受信し、data に格納します。size バイトのデータが受信できない場合は、戻り値として FALSE を返します。
エラー　エミュレータエラー

### 5.2.62._scri_echo_on: スクリプトウィンドウへの出力許可

関数名　int _scri_echo_on()
引数　　なし
戻り値　TRUE　　常に TRUE を返す
機能　　スクリプトウィンドウへの出力を許可します。スクリプトウィンドウは、デフォルトでは出力許可状態です。

### 5.2.63._scri_echo_off: スクリプトウィンドウへの出力不許可

関数名　int _scri_echo_off()
引数　　なし
戻り値　TRUE　　常に TRUE を返す
機能　　スクリプトウィンドウへの出力を不許可にします。

### 5.2.64._c_exp_eval: C 言語式解析

関数名　int _c_exp_eval(char *exp, int *value1, int *value2, char *type, char *str, char *misc)

引数

| | | |
|---|---|---|
| char | *exp | C 言語式 |
| int | *value1 | 解析結果 1 |
| int | *value2 | 解析結果 2 |
| char | *type | 解析結果の型を示す文字列 |
| char | *str | 解析結果を示す文字列 |
| char | *misc | 解析結果の付加情報を示す文字列 |

戻り値

| | |
|---|---|
| TRUE | 成功 |
| FALSE | エラー |

機能　exp で指定された C 言語式を現在のスコープで解析します。解析結果は 64 ビットの値で、下位 32 ビットが*value1、上位 32 ビットが*value2 に格納されます。解析結果の型名が type に格納され、解析結果を文字列に変換したものが str に格納されます。解析結果が関数など、通常の値を示すものでない場合に、付加的な情報が misc に格納されます。type, str, misc に格納される情報を printf()関数を用いて出力することにより、print スクリプトコマンドと同様な表示が行えます。

エラー

| | |
|---|---|
| ER_CEXP_NOT_FOUND | シンボルが見つからない |
| ER_CEXP_SYNTAX_ERROR | 文法エラー |
| ER_CEXP_NO_SCOPE | スコープが見つからない |
| ER_CEXP_NO_SYMBOL | シンボルが見つからない |
| ER_CEXP_NO_FUNC | 関数が見つからない |
| ER_CEXP_RIGHT_WRONG | 右辺式が不適切 |
| ER_CEXP_DEF_TYPE | 型の異なる構造体(共用体)のコピーは禁止 |
| ER_CEXP_CANT_ASSIGN | 代入できない |
| ER_CEXP_NO_TYPE | 型が見つからない |
| ER_CEXP_CANT_FLOAT | 浮動小数点型の演算はサポートしていない |
| ER_CEXP_CANT_P_TO_P | 指定の演算はポインタ型同士に対してはできない |
| ER_CEXP_CANT_SUB_P | 指定の演算はポインタ型に対してはできない |
| ER_CEXP_CANT_P | ポインタ変数による減算はできない |
| ER_CEXP_0_DIV | 0 で除算した |
| ER_CEXP_UNKNOWN_OP | 不正な演算子を用いた |
| ER_CEXP_BROKEN_TYPE | 型情報が壊れている |
| ER_CEXP_LEFT_POINT | 左辺値は、ポインタ変数でなければならない |
| ER_CEXP_LEFT_STRUCT | 左辺値は、構造体(共用体)型でなければならない |
| ER_CEXP_NO_MEMBER | メンバが見つからない |
| ER_CEXP_LEFT_STRUCT_REF | 左辺値は、構造体(共用体)型への参照でなければならない |
| ER_CEXP_LEFT_WRONG | 左辺値が不適切 |
| ER_CEXP_VAL_NUM | 被演算子は数値でなければならない |
| ER_CEXP_CANT_MIN | 指定の被演算子は符号反転できない |
| ER_CEXP_CANT_REF | アドレス値を求めることができない |
| ER_CEXP_LEFT_ARRAY | 配列変数が不適切 |

| | |
|---|---|
| ER_CEXP_RIGHT_NUM | 配列の要素番号が不適切 |
| ER_CEXP_NOT_POINT | 被演算子がアドレスではない |
| ER_CEXP_CANT_CAST_REG | レジスタ変数に対するキャスト演算はサポートしていない |
| ER_CEXP_CANT_CAST | キャストする型の指定が不適切 |
| ER_CEXP_CAST_NOT_SUPPORT | 指定の型に対するキャスト演算はサポートしていない |

### 5.2.65._get_shared_mem: 共有変数の取得

関数名　int _get_shared_mem(char *name, char *value)
引数　char　*name　共有変数名
　　char　*value　共有変数の値
戻り値　TRUE　成功
　　FALSE　共有変数が見つからない
機能　name で指定された共有変数を検索し、その値（文字列）を value に格納します。共有変数は、すべての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で共通に使用可能な変数です。変数名、および変数の値は文字列で、いくつかのオペレーションシステムに見られる環境変数の様に使用できます。変数名、および変数の値は６３バイトまで使用可能です。

### 5.2.66._set_shared_mem: 共有変数の設定

関数名　int _set_shared_mem(char *name, char *value)
引数　char　*name　共有変数名
　　char　*value　共有変数の値
戻り値　TRUE　常に TRUE を返す
機能　name で指定された共有変数を value で指定された値に設定します。既に定義済みの共有変数に対して値を設定した場合には、以前の値を value で指定された値に変更します。定義されていない共有変数の場合には、新たに変数領域を確保します。

### 5.2.67._delete_shared_mem: 共有変数の削除

関数名　int _delete_shared_mem(char *name)
引数　char　*name　共有変数名
戻り値　TRUE　常に TRUE を返す
機能　name で指定された共有変数を削除します。共有変数が定義されていない場合には、何も行いません。

### 5.2.68._get_err_msg: PD30 のエラーメッセージ文の取得

関数名　int _get_err_msg(int err_no, char *msg)
引数　int　err_no　エラー番号
　　char　*msg　エラーメッセージ文
戻り値　TRUE　成功
　　FALSE　エラー番号に対応するエラーメッセージ文が見つからない
機能　err_no で指定されたエラー番号に対応する PD30 のエラーメッセージ文を取得します。

### 5.2.69._get_tick_count: Windows 起動からの経過した時間の取得

関数名　int _get_tick_count()
引数　なし
戻り値　Windows 起動からの経過した時間（ミリ秒単位）
機能　Windows 起動からの経過した時間をミリ秒単位で取得します。

### 5.2.70._get_time: システムの現在の日付と時刻の取得

関数名　int _get_time(int *year, int *month, int *dayOfWeek,int *day,
int *hour, int *minute, int *secont, int *milliseconds)

| 引数 | | | |
|---|---|---|---|
| | int | *year | 現在の年の格納先 |
| | int | *month | 現在の月(1～12)の格納先 |
| | int | *dayOfWeek | 現在の曜日(日曜日=0,・・・)の格納先 |
| | int | *day | 現在の日(1～31)の格納先 |
| | int | *hour | 現在の時刻(1～24)の格納先 |
| | int | *minute | 現在の分(0～59)の格納先 |
| | int | *second | 現在の秒(0～59)の格納先 |
| | int | *milliseconds | 現在のミリ秒(0～999)の格納先 |

戻り値　TRUE　常にTRUEを返す
機能　システムの現在の日付と時刻を取得し、各引数に指定された領域に格納します。引数にNULLを指定した場合には、その情報は格納されません。

### 5.2.71._disp_src_line: プログラムウィンドウの表示内容変更

関数名　int　disp_src_line(int lineno, char *filename, int addr)

| 引数 | | | |
|---|---|---|---|
| | int | lineno | 行番号 |
| | char | *filename | ファイル名 |
| | int | addr | アドレス |

戻り値　TRUE　成功
FALSE　エラー
機能　PD のプログラムウィンドウの表示内容を更新します。lineno および filename で指定されたソースファイルの指定行をプログラムウィンドウ内にソースモードで表示します。指定されたソースファイルの指定行を表示できなかった場合には、addr で指定されたアドレスから逆アセンブルモードで表示します。

### 5.2.72._rtt_get_range: RTT データ範囲の取得

関数名　int _rtt_get_range(int *s_cycle, int *e_cycle)

| | | | |
|---|---|---|---|
| 引数 | int | *s_cycle | 開始サイクル |
| | int | *e_cycle | 終了サイクル |
| 戻り値 | TRUE | 成功 | |
| | FALSE | エラー | |

機能　参照可能なトレースデータの開始、終了サイクルを取得します。

| | | |
|---|---|---|
| エラー | ER_IN2_NONE_RTT | 参照可能なトレースデータが見つからない |
| | その他 | エミュレータエラー |

### 5.2.73._rtt_get_disasm: RTT データの逆アセンブル解析結果の取得

関数名　int _rtt_get_disasm(int *cycle, int *next_cycle, char *buffer,
int *count)

| | | | |
|---|---|---|---|
| 引数 | int | *cycle | 解析開始/解析結果サイクル |
| | int | *next_cycle | 次のフェッチサイクル |
| | char | *buffer | 解析結果文字列格納領域 |
| | int | *count | フェッチ命令数 |
| 戻り値 | TRUE | 成功 | |
| | FALSE | エラー | |

機能　*cycle で指定されたサイクルから、フェッチサイクルを検索し、見つかればそのサイクルを*cycle に格納し、その命令を逆アセンブルして、buffer に格納します。*next_cycle には、次のフェッチサイクルを格納します。*count には、見つかったフェッチサイクルで何命令フェッチしたかを格納します。1 命令より多くの命令をフェッチした場合には、その命令分の逆アセンブルを行い、改行コードで区切って buffer に格納します。

| | | |
|---|---|---|
| エラー | ER_IN2_NONE_RTT | 参照可能なトレースデータが見つからない |
| | ER_IN2_CYCLE_OUTRANGE | 指定したサイクル値が範囲外 |
| | その他 | エミュレータエラー |

### 5.2.74._rtt_get_bus: RTT データのバスモード表示文字列の取得

関数名　int _rtt_get_bus(int cycle, char *buffer)
引数　int　cycle　サイクル
　　char　*buffer 文字列格納領域
戻り値　TRUE　成功
　　FALSE　エラー
機能　cycle で指定されたサイクルのトレースデータを表示用文字列に変換して、buffer に格納します。
エラー　ER_IN2_NONE_RTT　参照可能なトレースデータが見つからない
　　ER_IN2_CYCLE_OUTRANGE　指定したサイクル値が範囲外
　　その他　エミュレータエラー

### 5.2.75._rtt_check_isfetch: RTT データのフェッチサイクルの検査

関数名　int _rtt_check_isfetch(int cycle, int *addr1, int *addr2,
　　int *count)
引数　int　cycle　検査サイクル
　　int　*addr1　フェッチアドレス１
　　int　*addr2　フェッチアドレス２
　　int　*count　フェッチ命令数
戻り値　TRUE　成功
　　FALSE　エラー
機能　cycle で指定されたサイクルがフェッチサイクルであるかを検査します。フェッチサイクルでない場合には、*count に０を格納します。フェッチサイクルの場合には、*count にフェッチ命令数を格納し、*addr1、*addr2 にそれぞれ一つ目の命令をフェッチしたアドレス、二つ目の命令をフェッチしたアドレスを格納します。
エラー　ER_IN2_NONE_RTT　参照可能なトレースデータが見つからない
　　ER_IN2_CYCLE_OUTRANGE　指定したサイクル値が範囲外
　　その他　エミュレータエラー

### 5.2.76._rtt_get_data: RTT データの取得

関数名　_rtt_get_data(int cycle, int *addr, int *data, int *state, int *ext,
int *dn, int *h, int *m, int *s, int *ms, int *us)

引数

| | | |
|---|---|---|
| int | cycle | サイクル |
| int | *addr | アドレスバスの値 |
| int | *data | データバスの値 |
| int | *state | CPU テータス信号の値 |
| int | *ext | 外部トリガ信号の値 |
| int | *dn | キューバッファ占有バイト数 |
| int | *h | 時間 |
| int | *m | 分 |
| int | *s | 秒 |
| int | *ms | ミリ秒 |
| int | *us | マイクロ秒 |

戻り値

| | |
|---|---|
| TRUE | 成功 |
| FALSE | エラー |

機能　cycle で指定されたサイクルの RTT データを取得します。*addr、*data には、それぞれアドレスバスの値、データバスの値が格納されます。*state には、以下のように CPU ステータス信号が格納されます。

CPU ステータス上位

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MBHE* | MBIU2 | MBIU1 | MBIU0 | 0 | BUS16* | ALE | RWT* |

CPU ステータス下位

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MCPU2 | MCPU1 | MCPU0 | 0 | 0 | 0 | WR* | RD* |

（図中の*印は Low アクティブであることを示します）

*ext、*dn には、それぞれ外部トリガ入力値、キューバッファに入っている命令バイト数が格納されます。*h、*m、*s、*ms、*us には、それぞれ実行開始からの時、分、秒、ミリ秒、マイクロ秒が格納されます。

エラー

| | |
|---|---|
| ER_IN2_NONE_RTT | 参照可能なトレースデータが見つからない |
| ER_IN2_CYCLE_OUTRANGE | 指定したサイクル値が範囲外 |
| その他 | エミュレータエラー |

### 5.2.77._rtt_clear_cache: リアルタイムトレース(RTT)キャッシュのクリア

関数名　int _rtt_clear_cache()

引数　なし

戻り値　TRUE　常に TRUE を返す

機能　リアルタイムトレース(RTT)キャッシュをクリアします。

### 5.2.78._cv_get_data: カバレッジデータの取得

関数名　int _cv_get_data(int s_addr, int e_addr, int *rs_addr, int *re_addr, char *data)

引数

| | | |
|---|---|---|
| int | s_addr | 取得開始アドレス |
| int | e_addr | 取得終了アドレス |
| int | *rs_addr | 取得開始実アドレス |
| int | *re_addr | 取得終了実アドレス |
| char | *data | 取得カバレッジデータ |

戻り値

| | |
|---|---|
| TRUE | 成功 |
| FALSE | エラー |

機能　引数 s_addr, e_addr で指定された範囲を含むカバレッジデータを引数 data で指定した領域に格納します。ただし、data1 バイト分には、8 バイトアライメントからの 8 バイト分が格納されるため、実際には s_addr, e_addr で指定したアドレスより多くの範囲のデータが格納されることがあります。たとえば、3h 番地～19h 番地を指定した場合には、0h 番地～1Fh 番地のデータが格納されます。実際に格納された取得開始実アドレス、取得終了実アドレスは、それぞれ引数*rs_addr, *re_addr に格納されます。なお、*rs_addr, *re_addr に格納される値は、以下の計算式で求めることができます。

*rs_addr = s_addr / 8 * 8
*re_addr = e_addr / 8 * 8 + 7

引数 data には、e_addr - s_addr / 8 + 1 より大きい配列を指定してください。data 1 バイトに格納されるカバレッジデータのデータフォーマットは、以下の通りです。

（上段：ビットオフセット、下段：アドレスオフセット）

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ＋7 | ＋6 | ＋5 | ＋4 | ＋3 | ＋2 | ＋1 | ＋0 |

たとえば、s_addr が 0x400 の場合の data[0]には、下記の様にそれぞれのビットに対応するアドレスオフセットのアドレスのカバレッジ結果が格納されます。

（上段：ビットオフセット、下段：アドレス）

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0x407 | 0x406 | 0x405 | 0x404 | 0x403 | 0x402 | 0x401 | 0x400 |

つまり、s_addr から 1 バイトおきにアクセスがあった場合には、下記の様にカバレッジデータが格納され、

（上段：ビットオフセット、下段：カバレッジ計測結果）

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

data[0]には、01010101B すなわち 0x55 が格納されます。

エラー

| | |
|---|---|
| ER_IN2_ADDR_OUTRANGE | 指定アドレスが範囲外 |
| ER_IN2_RUNNING | 実行中であるため取得できない |
| その他 | エミュレータエラー |

### 5.2.79._cv_set_data: カバレッジデータの設定

関数名　int _cv_set_data(int s_addr, int e_addr, char *data)
引数　int　s_addr　設定開始アドレス
int　e_addr　設定終了アドレス
char　*data　設定カバレッジデータ
戻り値　TRUE　成功
FALSE　エラー
機能　引数 data で指定した領域に格納されているカバレッジデータを、引数 s_addr, e_addr で指定された範囲に設定します。ただし、data 1 バイトには、アドレス 8 バイト分のカバレッジデータが格納されるため、s_addr, e_addr は、８バイト単位の値を指定してください。data のフォーマットは、上記の_cv_get_data()関数と同様です。
エラー　ER_IN2_ADDR_OUTRANGE　指定アドレスが範囲外
ER_IN2_RUNNING　実行中であるために設定できない
その他　エミュレータエラー

### 5.2.80._cv_get_base: カバレッジ範囲のベースアドレスの取得

関数名　int _cv_get_base(int *base)
引数　int　*base　取得ベースアドレス
戻り値　TRUE　常に TRUE を返す
機能　カバレッジベースアドレスを*base に格納します。

### 5.2.81._cv_set_base: カバレッジ範囲のベースアドレスの設定

関数名　int _cv_set_base(int base)
引数　int　base　設定ベースアドレス
戻り値　TRUE　成功
FALSE　エラー
機能　カバレッジデータをクリアし、引数 base で指定されたカバレッジベースアドレスを設定します。ベースアドレスは、0x00000～0xfffff までの 64KB 単位のアドレス値が、設定可能です。ただし、0xc0000 以上の値が指定された場合には、0xc0000 に丸められます。
エラー　ER_IN2_RUNNING　実行中であるために設定できない
その他　エミュレータエラー

### 5.2.82._cv_clear_data: カバレッジデータのクリア

関数名　int _cv_clear_data()
引数　なし
戻り値　TRUE　成功
FALSE　エラー
機能　カバレッジデータをクリアします。
エラー　ER_IN2_RUNNING　実行中であるためにクリアできない
その他　エミュレータエラー

### 5.2.83._cv_clear_cache: カバレッジキャッシュのクリア

関数名　int _cv_clear_cache()
引数　なし
戻り値　TRUE　常に TRUE を返す
機能　カバレッジキャッシュをクリアします。

### 5.2.84._syscom: PD30 のスクリプトコマンドの実行

関数名　int _syscom(char *command)
引数　char　*command　PD30 のスクリプトコマンド文字列
戻り値　TRUE　成功
FALSE　エラー
機能　command で指定された文字列を PD30 のスクリプトコマンドとして実行します。たとえば、1000H 番地から 1FFFH 番地までをダンプするスクリプトコマンドは、以下の様に指定します。

```
_syscom("DumpByte 1000, 1FFF");
```

### 5.2.85._doscom: DOS コマンドの実行

関数名　int _doscom(char *command)
引数　char　*command　DOS コマンド文字列
戻り値　TRUE　成功
FALSE　エラー
機能　command で指定された文字列を DOS コマンドとして実行します。たとえば、DOS の DIR コマンドに/W オプションを指定して、出力結果を TMP ファイルにリダイレクトするコマンドは、以下の様に指定します。

```
_doscom("DIR /W > TMP");
```

### 5.2.86.エミュレータエラー一覧

システムコール関数が FALSE を返した場合に、グローバル変数 macro_err に格納されるエラー番号は以下の通りです。

| エラー番号 | 説明 |
|---|---|
| ER_IN2_MCU_RESET | ターゲットリセット |
| ER_IN2_ERROR_2 | 受信データのチェックサムエラー |
| ER_IN2_ERROR_3 | 指定されたデータが存在しない |
| ER_IN2_ERROR_4 | ターゲットプログラム実行中 |
| ER_IN2_ERROR_5 | ターゲットプログラム停止中 |
| ER_IN2_ERROR_6 | すでに測定停止状態 |
| ER_IN2_ERROR_7 | すでに測定実行状態 |
| ER_IN2_ARG_ERROR | 引数エラー |
| ER_IN2_ERROR_9 | 測定が完了していない |
| ER_IN2_ERROR_A | 指定サイクルのトレースデータがない |
| ER_IN2_ERROR_B | トレースデータがない |
| ER_IN2_MCU_DISRESET | ターゲットリセット不可 |
| ER_IN2_MCU_POF | MCU パワー OFF |
| ER_IN2_ERROR_E | 時間計測カウンタオーバーフロー |
| ER_IN2_ERROR_F | POF 状態を強制リセットで解除した |
| ER_IN2_ERROR_G | ポイントの点数が範囲を越えている |
| ER_IN2_ERROR_H | ブレークが設定されていない |
| ER_IN2_ERROR_I | ソース情報がロードされていない |
| ER_IN2_ERROR_J | トリガモードがソフト出力でない |
| ER_IN2_MCU_CLKOFF | ターゲットクロック停止 |
| ER_IN2_ERROR_L | ステップ実行中に例外処理を検出した |
| ER_IN2_ERROR_M | 関数範囲が設定外である |
| ER_IN2_ERROR_N | EEPROM への書き込みエラー |
| ER_IN2_MCU_NOSOURCE | ターゲット電源 OFF |
| ER_IN2_MCU_RUN | ターゲット MCU 実行エラー |

## 5.3.ウィンドウ操作系システムコール関数(winlib.lib)

winlib.lib では、カスタムウィンドウプログラムで使用できるウィンドウ操作関数を提供します。各関数のプロトタイプ宣言は、winlib.h に記述されています。

| 関数名 | 説明 |
|---|---|
| _win_printf | 書式付きテキスト出力 |
| _win_puts | カスタムウィンドウへの文字列の出力 |
| _win_set_cursor | カーソル位置の設定 |
| _win_set_color | テキストの色の設定 |
| _win_set_bkcolor | 背景色の設定 |
| _win_column2dot | カーソル座標のピクセル座標への変換 |
| _draw_text_out | カスタムウィンドウへの文字列の出力 |
| _draw_set_color | テキストの色の設定 |
| _draw_set_bkcolor | 背景色の設定 |
| _draw_set_bkmode | バックグラウンドモードの設定 |
| _draw_set_font | フォントの設定 |
| _draw_get_char_size | フォント文字の大きさの取得 |
| _draw_line | 線の描画 |
| _draw_fill_rect | 四角形の塗りつぶし |
| _draw_frame_rect | 四角形の描画 |
| _draw_invert_rect | 四角形の色の反転 |
| _draw_arc | 楕円の弧の描画 |
| _draw_pie | 扇形の描画 |
| _win_redraw | カスタムウィンドウの再描画 |
| _win_redraw_clear | カスタムウィンドウの再描画 |
| _win_redraw_item | コントロールアイテムの再描画 |
| _win_show_window | コントロールアイテムの表示・非表示の設定 |
| _win_set_window_title | カスタムウィンドウのタイトルの設定 |
| _win_enable_window | コントロールアイテムの有効・無効の設定 |
| _win_button_create | ボタンの作成 |
| _win_button_set_text | ボタンのテキストの変更 |
| _win_hscroll_range | 水平スクロールバーのスクロール範囲の設定 |
| _win_hscroll_pos | 水平スクロールボックスの位置の設定 |
| _win_vscroll_range | 垂直スクロールバーのスクロール範囲の設定 |
| _win_vscroll_pos | 垂直スクロールボックスの位置の設定 |
| _win_statusbar_create | ステータスバーの作成 |
| _win_statusbar_set_pane | ステータスバーの項目の設定 |
| _win_statusbar_set_text | ステータスバーのテキストの設定 |
| _win_dialog | 入力ダイアログの作成 |
| _win_message_box | メッセージボックスの作成 |
| _win_filedialog | ファイル選択ダイアログの作成 |
| _win_set_window_pos | カスタムウィンドウの位置の設定 |
| _win_set_window_size | カスタムウィンドウのサイズの設定 |
| _win_timer_set | システムタイマの設定 |
| _win_timer_kill | システムタイマの解除 |

### 5.3.1._win_printf: 書式付きテキスト出力(カスタムウィンドウへの出力)

関数名　int _win_printf(char *format , ...);
引数　　char　　*forma　フォーマット
　　...　　可変引数
戻り値　int　　出力した文字数
機能　　format の制御のもとに出力が変換されて、_win_set_color()関数で指定されたテキスト色、_win_set_bkcolor()関数で指定された背景色で、カスタムウィンドウのカーソル位置への出力が行なわれます。カーソル位置は、最後に出力された文字の次に設定されます。カーソル位置は、_win_set_cursor()関数によって任意の位置に設定できます。なお、文字のフォントは FIXED_SYS のみが使用されます。

### 5.3.2._win_puts: カスタムウィンドウへの文字列の出力

関数名　int _win_puts(char *str)
引数　　char　　*str　　出力文字列
戻り値　TRUE　　常に TRUE を返す
機能　　_win_set_color()関数で指定されたテキスト色、_win_set_bkcolor()関数で指定された背景色で、カスタムウィンドウのカーソル位置へ str で指定された文字列を出力します。カーソル位置は、最後に出力された文字の次に設定されます。カーソル位置は、_win_set_cursor()関数によって任意の位置に設定できます。なお、文字のフォントは FIXED_SYS のみが使用されます。

### 5.3.3._win_set_cursor: カーソル位置の設定

関数名　int _win_set_cursor(int x, int y)
引数　　int　　x　　カーソルの X カラム位置
　　int　　y　　カーソルの Y カラム位置
戻り値　TRUE　　常に TRUE を返す
機能　　x, y で指定された位置にカーソル位置を移動します。カーソル位置は、カスタムウィンドウのクライアント領域の左上隅を(0, 0)とし、右方向にｘ、下方向にｙとなります。１カラムには、１文字が出力されます。

### 5.3.4._win_set_color: テキストの色の設定

int _win_set_color(int color)

引数　int　color　テキストの色

戻り値　int　事前のテキスト色

機能　テキストに、colorで指定された色を設定します。本関数で指定されたテキストの色は、_win_printf()関数および_win_puts()関数で文字列を出力する際に使用されます。colorには、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.5._win_set_bkcolor: 背景色の設定

関数名　int _win_set_bkcolor(int color)
引数　　int　　color　テキストの背景色
戻り値　int　　事前の背景色
機能　　現在の背景色に、color で指定された色を設定します。本関数で指定されたテキストの色は、_win_printf()関数および_win_puts()関数で文字列を出力する際に使用されます。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.6._win_column2dot: カーソル座標のピクセル座標への変換

関数名　int _win_column2dot(int xcol, int ycol, int *xpix, int *ypix)
引数　　int　　xcol　X カラム位置
　　int　　ycol　Y カラム位置
　　int　　*xpix　X カラム位置の X ピクセル座標
　　int　　*ypix　Y カラム位置の Y ピクセル座標
戻り値　TRUE　常に TRUE を返す
機能　　xcol, ycol で指定されたカーソル位置を対応するピクセル座標に変換し、*xpix, *ypix に格納します。

### 5.3.7._draw_text_out: カスタムウィンドウへの文字列の出力

関数名　int _draw_text_out(int x, int y, char *str)
引数　　int　　x　　テキストの開始位置の X ピクセル座標
　　int　　y　　テキストの開始位置の Y ピクセル座標
　　char　　*str　描画される文字列へのポインタ
戻り値　TRUE　常に TRUE を返す
機能　　現在選択されているフォントを使って、_draw_set_color()関数で指定されたテキスト色、_draw_set_bkcolor()関数で指定された背景色で、指定された位置に文字列を書き込みます。

### 5.3.8._draw_set_color: テキストの色の設定

関数名　int _draw_set_color(int color)
引数　　int　　color　テキストの色
戻り値　int　　事前のテキスト色
機能　　テキストに、color で指定された色を設定します。本関数で指定されたテキストの色は、_draw_text_out()関数で文字列を出力する際に使用されます。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.9._draw_set_bkcolor: 背景色の設定

関数名　int _draw_set_bkcolor(int color)
引数　　int　　color　テキストの背景色
戻り値　int　　事前の背景色
機能　　現在の背景色に、color で指定された色を設定します。本関数で指定された背景色は、_draw_text_out()関数で文字列を出力する際に使用されます。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

バックグラウンドモードが"塗りつぶしモード"のときは、システムはスタイル付きライン間のすきま、ブラシのハッチライン間のすきま、文字セルのバックグラウンドを背景色で塗りつぶします。

### 5.3.10._draw_set_bkmode: バックグラウンドモードの設定

関数名 int _draw_set_bkmode(int flag)
引数　　int　　flag　　設定するモード
戻り値　TRUE　　常に TRUE を返す
機能　　バックグラウンドモードを設定します。テキストの描画前に領域を背景色で塗りつぶすかを指定します。flag に TRUE を指定すると、バックグラウンドを現在の背景色で塗りつぶします(デフォルト)。flag に FALSE を設定すると、描画前にバックグラウンドは変更されません。

### 5.3.11._draw_set_font: フォントの設定

関数名　int _draw_set_font(int size, int font)
引数　　int　　size　　フォントのサイズ
　　int　　font　　フォント定数
戻り値　TRUE　　常に TRUE を返す
機能　　現在の描画フォントのサイズ、種類を指定します。font には、以下に示すフォント定数を｜で組み合わせて指定します。

| フォント定数 | フォントの種類 |
|---|---|
| FONT_FIXED_SYS | "FixedSys" |
| FONT_MINTYO | "ＭＳ 明朝" |
| FONT_GOTHIC | "ＭＳ ゴシック" |
| FONT_TIMESNEWROMAN | "Times New Roman" |
| FONT_CENTURY | "Century" |
| FONT_ARIAL | "Arial" |
| FONT_BOLD | 太字 |
| FONT_ITALIC | 斜体 |

### 5.3.12._draw_get_char_size: フォント文字の大きさの取得

関数名　int _draw_get_char_size(int *pWidth, int *pHeight)
引数　　int　　*pWidth　　文字の幅の格納先
　　int　　*pHeight　　文字の高さの格納先
戻り値　TRUE　　常に TRUE を返す
機能　　現在設定されているフォント文字の大きさを取得します。

### 5.3.13._draw_line: 線の描画

関数名　int _draw_line(int x1, int y1, int x2, int y2, int color)

引数　int　x1　線の開始位置の X ピクセル座標
int　y1　線の開始位置の Y ピクセル座標
int　x2　線の終了位置の X ピクセル座標
int　y2　線の終了位置の Y ピクセル座標
int　color　線の色

戻り値　TRUE　常に TRUE を返す

機能　指定した座標間に、指定した色の線を描画します。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.14._draw_fill_rect: 四角形の塗りつぶし

関数名　int _draw_fill_rect(int x1, int y1, int x2, int y2, int color)

引数　int　x1　四角形の左上の X ピクセル座標
int　y1　四角形の左上の Y ピクセル座標
int　x2　四角形の右下の X ピクセル座標
int　y2　四角形の右下の Y ピクセル座標
int　color　塗りつぶす色

戻り値　TRUE　常に TRUE を返す

機能　指定したピクセル座標を四隅とする、指定した色で塗りつぶされた四角形を描画します。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.15._draw_frame_rect: 四角形の描画

関数名　int _draw_frame_rect(int x1, int y1, int x2, int y2, int color)

引数　int　x1　四角形の左上の X ピクセル座標
int　y1　四角形の左上の Y ピクセル座標
int　x2　四角形の右下の X ピクセル座標
int　y2　四角形の右下の Y ピクセル座標
int　color　線の色

戻り値　TRUE　常に TRUE を返す

機能　指定したピクセル座標を四隅とする、指定した色の四角形の線を描画します。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.16._draw_invert_rect: 四角形の色の反転

関数名　int _draw_invert_rect(int x1, int y1, int x2, int y2)

引数　int　x1　四角形の左上の X ピクセル座標
int　y1　四角形の左上の Y ピクセル座標
int　x2　四角形の右下の X ピクセル座標
int　y2　四角形の右下の Y ピクセル座標

戻り値　TRUE　常に TRUE を返す

機能　指定したピクセル座標を四隅とする四角形の色を反転します。

### 5.3.17._draw_arc: 楕円の弧の描画

**関数名** int _draw_arc(int x1, int y1, int x2, int y2,
int x3, int y3, int x4, int y4, int color)

**引数**

| | | |
|---|---|---|
| int | x1 | 境界矩形の左上隅の X ピクセル座標 |
| int | y1 | 境界矩形の左上隅の Y ピクセル座標 |
| int | x2 | 境界矩形の右下隅の X ピクセル座標 |
| int | y2 | 境界矩形の右下隅の Y ピクセル座標 |
| int | x3 | 弧の描画開始位置の X ピクセル座標 |
| int | y3 | 弧の描画開始位置の Y ピクセル座標 |
| int | x4 | 弧の描画終了位置の X ピクセル座標 |
| int | y4 | 弧の描画終了位置の Y ピクセル座標 |
| int | color | 円弧の色 |

**戻り値** TRUE 成功
FALSE エラー

**機能** 楕円の弧を描画します。境界矩形のピクセル座標(x1, y1)-(x2, y2)と弧の始点のピクセル座標(x3, y3)と終点のピクセル座標(x4, y4)を指定します。弧の始点と終点は弧の線上にある必要はありません。指定された始点と境界矩形の中心を結ぶ直線を計算し、それをもとにして弧の始点を算出します。終点の算出も同様です。color には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.18._draw_pie: 扇形の描画

**関数名** int _draw_pie(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, int framecolor, int paintcolor)

**引数**

| | | |
|---|---|---|
| int | x1 | 境界矩形の左上隅の X ピクセル座標 |
| int | y1 | 境界矩形の左上隅の Y ピクセル座標 |
| int | x2 | 境界矩形の右下隅の X ピクセル座標 |
| int | y2 | 境界矩形の右下隅の Y ピクセル座標 |
| int | x3 | 扇形の始点位置の X ピクセル座標 |
| int | y3 | 扇形の始点位置の Y ピクセル座標 |
| int | x4 | 扇形の終点位置の X ピクセル座標 |
| int | y4 | 扇形の終点位置の Y ピクセル座標 |
| int | framecolor | 扇形の枠線の色 |
| int | paintcolor | 扇形を塗りつぶす色 |

**戻り値** TRUE 成功
FALSE エラー

**機能** 扇形を描画します。扇形の外周の円は楕円の境界矩形のピクセル座標(x1, y1)-(x2, y2)によって定義します。framecolor、paintcolor には、以下に示すカラー定数を指定します。

| カラー定数 | 色 |
|---|---|
| COLOR_BLACK | 黒 |
| COLOR_BLUE | 青 |
| COLOR_GREEN | 緑 |
| COLOR_CYAN | 水色 |
| COLOR_RED | 赤 |
| COLOR_MAGENDA | 紫 |
| COLOR_YELLOW | 黄 |
| COLOR_WHITE | 白 |
| COLOR_GRAY | 灰 |
| COLOR_DKBLUE | 暗い青 |
| COLOR_DKGREEN | 暗い緑 |
| COLOR_DKCYAN | 暗い水色 |
| COLOR_DKRED | 暗い赤 |
| COLOR_DKMAGENDA | 暗い紫 |
| COLOR_DKYELLOW | 暗い黄 |
| COLOR_LTGRAY | 明るい灰 |

### 5.3.19._win_redraw: カスタムウィンドウの再描画

関数名　int _win_redraw()
引数　　なし
戻り値　TRUE　　常に TRUE を返す
機能　　カスタムウィンドウの表示を消去せずに再描画します。

### 5.3.20._win_redraw_clear: カスタムウィンドウの再描画

関数名　int _win_redraw_clear()
引数　　なし
戻り値　TRUE　　常に TRUE を返す
機能　　カスタムウィンドウの表示を消去してから再描画します。

### 5.3.21._win_redraw_item: コントロールアイテムの再描画

関数名　int _win_redraw_item(int handle)
引数　　int　　handle　コントロールアイテムのハンドル
戻り値　TRUE　　常に TRUE を返す
機能　　handle で指定されたコントロールアイテム(ボタン)を再描画します。

### 5.3.22._win_show_window: コントロールアイテムの表示・非表示の設定

関数名　int _win_show_window(int handle, int flag)
引数　　int　　handle　コントロールアイテムのハンドル
　　int　　flag　　TRUE:表示　FALSE:非表示
戻り値　TRUE　　常に TRUE を返す
機能　　handle で指定されたコントロールアイテム(ボタン)の表示/非表示を指定します。flag に TRUE を指定した場合に表示され、FALSE を指定した場合に非表示になります。

### 5.3.23._win_set_window_title: カスタムウィンドウのタイトルの設定

関数名　int _win_set_window_title(char *title)
引数　　char　　*title　ウィドウタイトル
戻り値　TRUE　　常に TRUE を返す
機能　　title で指定された文字列を、カスタムウィンドウのタイトルに設定します。

### 5.3.24._win_enable_window: コントロールアイテムの有効・無効の設定

関数名　int _win_enable_window(int handle, int flag)
引数　　int　　handle　コントロールアイテムのハンドル
　　int　　flag　　TRUE:有効　FALSE:無効
戻り値　TRUE　　常に TRUE を返す
機能　　handle で指定されたコントロールアイテム(ボタン)の状態を指定します。flag に TRUE を指定した場合に有功となり、FALSE を指定した場合に無効となります。無効時コントロールアイテムは灰色表示 されます。

### 5.3.25._win_button_create: ボタンの作成

関数名　int _win_button_create(int x1,int y1,int x2,int y2,
　　char *str,int id)

引数

| | | |
|---|---|---|
| int | x1 | ボタン左上の X ピクセル座標 |
| int | y1 | ボタン左上の Y ピクセル座標 |
| int | x2 | ボタン右下の X ピクセル座標 |
| int | y2 | ボタン右下の Y ピクセル座標 |
| char | *str | ボタンコントロールのテキスト |
| int | id | ボタンのコントロール ID |

戻り値　int　ボタンのハンドル

機能　x1, y1, x2, y2 で指定された領域に、表面に引数 str で指定されたテキストを表示するボタンを作成します。id で指定されたコントロール ID はボタンクリック時に、OnCommand()ハンドル関数の引数 nId として送信されます。

### 5.3.26._win_button_set_text: ボタンのテキストの変更

関数名　int _win_button_set_text(int handle. char *text)

引数

| | | |
|---|---|---|
| int | handle | ボタンのハンドル |
| char | *text | ボタンコントロールのテキスト |

戻り値

| | |
|---|---|
| TRUE | 成功 |
| FALSE | エラー |

機能　handle で指定されたボタンに表示するテキストを text で指定されたテキストに変更します。

### 5.3.27._win_hscroll_range: 水平スクロールバーのスクロール範囲の設定

関数名　int _win_hscroll_range(int min, int max)

引数

| | | |
|---|---|---|
| int | min | 水平スクロールバーの最小スクロール位置 |
| int | max | 水平スクロールバーの最大スクロール位置 |

戻り値　TRUE　常に TRUE を返す

機能　カスタムウィンドウの水平スクロールバーの最小位置と最大位置を指定します。min と max の両方に 0 を指定すると、水平スクロールバーは非表示になります。デフォルトでは両方の引数に 0 が指定された非表示状態になっています。スクロール範囲の奨励は 0 から 100 です。

### 5.3.28._win_hscroll_pos: 水平スクロールボックスの位置の設定

関数名　int _win_hscroll_pos(int pos)

引数

| | | |
|---|---|---|
| int | pos | 水平スクロールボックスの新しい位置 |

戻り値　TRUE　常に TRUE を返す

機能　カスタムウィンドウの水平スクロールボックスの現在位置を設定し、スクロールボックスの新しい位置を反映するためにスクロールバーを再描画します。新しい位置はスクロール範囲内になければなりません。

### 5.3.29._win_vscroll_range: 垂直スクロールバーのスクロール範囲の設定

関数名　int _win_vscroll_range(int min, int max)
引数　int　min　垂直スクロールバーの最小スクロール位置
　　int　max　垂直スクロールバーの最大スクロール位置
戻り値　TRUE　常に TRUE を返す
機能　カスタムウィンドウの垂直スクロールバーの最小位置と最大位置を指定します。min と max の両方に 0 を指定すると、垂直スクロールバーは非表示になります。デフォルトでは両方の引数に 0 が指定された非表示状態になっています。スクロール範囲の奨励は 0 から 100 です。

### 5.3.30._win_vscroll_pos: 垂直スクロールボックスの位置の設定

関数名　int _win_vscroll_pos(int pos)
引数　int　pos　垂直スクロールボックスの新しい位置
戻り値　TRUE　常に TRUE を返す
機能　カスタムウィンドウの垂直スクロールボックスの現在位置を設定し、スクロールボックスの新しい位置を反映するためにスクロールバーを再描画します。新しい位置はスクロール範囲内になければなりません。

### 5.3.31._win_statusbar_create: ステータスバーの作成

関数名　int _win_statusbar_create(int cnt)
引数　int　cnt　ステータスバーの項目数
戻り値　TRUE　常に TRUE を返す
機能　カスタムウィンドウの下端にステータスバーを作成します。cnt にはステータスバーの項目数を設定します。

### 5.3.32._win_statusbar_set_pane: ステータスバーの項目の設定

関数名　int _win_statusbar_set_pane(int index, int style, int size)
引数　int　index　ステータスバーの項目のインデックス番号
　　int　style　項目のスタイル
　　int　size　項目のサイズ(ピクセル値)
戻り値　TRUE　常に TRUE を返す
機能　作成したステータスバーの index で指定された項目に、style で指定されたスタイルと、size で指定されサイズを設定します。style には、以下に示すスタイルを指定します。

| スタイル | 説明 |
|---|---|
| SBPS_NOBORDERS | ペインの周囲に3Dの境界線を持たない。 |
| SBPS_POPOUT | テキストが"浮き出す"反転表示された境界線を持つ。 |
| SBPS_DISABLED | テキストを描画しない。 |
| SBPS_NORMAL | 伸張も反転もせず、境界も持たない。 |
| SBPS_STRETCH | 未使用スペースを埋めるためにペインを伸張する。ステータスバーの中ではこのスタイルを持つペインは1つしか許されない。また、このスタイルは他のスタイルと\|で組み合わせることができる。 |

### 5.3.33._win_statusbar_set_text: ステータスバーのテキストの設定

関数名　int _win_statusbar_set_text(tint index, char *text)
引数　int　index　ステータスバーの項目のインデックス番号
　　char　*text　ステータスバーに表示するテキスト
戻り値　TRUE　常に TRUE を返す
機能　項目に表示するテキストを設定します。

### 5.3.34._win_dialog: 入力ダイアログの作成

関数名　int _win_dialog(char *str, char *buf)
引数　char　*str　表示するメッセージを示す文字列
　　char　*buf　取得した文字列の格納先
戻り値　TRUE　OK ボタンが押された
　　FALSE　キャンセルボタンが押された
機能　入力ダイアログをオープンして１行の文字列を取得します。

### 5.3.35._win_message_box: メッセージボックスの作成

関数名　int _win_message_box(char *str, char *title, int style)
引数　char　*str　表示するメッセージ
　　char　*title　メッセージボックスのタイトル
　　int　style　メッセージボックスの動作と内容の指定
戻り値　int　以下に示す関数の実行結果

| 値 | 意味 |
|---|---|
| 0 | 十分なメモリがない |
| IDABORT | [中止]ボタンが選択されまた |
| IDCANCEL | [キャンセル]ボタンが選択された |
| IDIGNORE | [無視]ボタンが選択された |
| IDNO | [いいえ]ボタンが選択された |
| IDOK | [OK]ボタンが選択された |
| IDRETRY | [再試行]ボタンが選択された |
| IDYES | [はい]ボタンが選択された |

機能　メッセージボックスを作成します。style には、以下に示すスタイルを｜で組み合わせて指定します。

| スタイル | 説明 |
|---|---|
| MB_ABORTRETRYIGNORE | メッセージボックスには、[中止]、[再試行]、[無視]の３つのプッシュボタンが含まれる。 |
| MB_APPLMODAL | メッセージボックスに応答するまで PD30/CB30 の動作を停止する(デフォルト)。 |
| MB_DEFBUTTON1 | 最初のボタンがデフォルトになる。MB_DEFBUTTON2、MB_DEFBUTTON3 を指定しなければ、最初のボタンが常にデフォルトになる。 |
| MB_DEFBUTTON2 | ２番目のボタンがデフォルトになる。 |
| MB_DEFBUTTON3 | ３番目のボタンがデフォルトになる。 |

| スタイル(つづき) | 説明 |
| --- | --- |
| MB_ICONEXCLAMATION | 感嘆符アイコンがメッセージボックスに表示される。 |
| MB_ICONHAND | MB_ICONSTOP と同じである。 |
| MB_ICONINFORMATION | 円の中に小文字の「i」があるアイコンがメッセージボックス内に表示される。 |
| MB_ICONQUESTION | 疑問符(?)のアイコンがメッセージボックス内に表示される。 |
| MB_ICONSTOP | 「STOP」アイコンがメッセージボックスに表示される。 |
| MB_OK | メッセージボックスには、[OK]プッシュボタンが含まれる。 |
| MB_OKCANCEL | メッセージボックスには、[OK]プッシュボタンと[キャンセル]プッシュボタンが含まれる。 |
| MB_RETRYCANCEL | メッセージボックスには、[再試行]プッシュボタンと[キャンセル]プッシュボタンが含まれる。 |
| MB_SYSTEMMODAL | すべてのアプリケーションが、ユーザがメッセージボックスに応答するまで中断される。システムモーダルメッセージボックスは、すぐに対応を求められるような、重大で潜在的な危険性のあるエラー(メモリ不足など)を通知するために使用すること。 |
| MB_YESNO | メッセージボックスには、[はい]と[いいえ]の 2 つのプッシュボタンが含まれる。 |
| MB_YESNOCANCEL | メッセージボックスには、[はい]、[いいえ]、[キャンセル]の 3 つのプッシュボタンが含まれる。 |

### 5.3.36._win_filedialog: ファイル選択ダイアログの作成

関数名　int _win_filedialog(char *title, int openFileDialog, char *defExt, char *defFileName, int flags, char *filter, char *fileName)

引数

| | | |
|---|---|---|
| char | *title | ダイアログのタイトル |
| int | openFileDialog | オープン・セーブの指定 |
| char | *defExt | デフォルトのファイル拡張子 |
| char | *defFileName | デフォルトのファイル名 |
| int | flags | ダイアログをカスタマイズするフラグ |
| char | *filter | フィルタ指定 |
| char | *fileName | 取得したファイル名の格納先 |

戻り値　TRUE　OK ボタンが押された
FALSE　キャンセルボタンが押された

機能　ファイル選択ダイアログを作成し、選択されたファイル名を取得します。title には、ダイアログのタイトルを指定します。openFileDialog には、[ファイルを開く]ダイアログボックスを構築するときは TRUE を、[ファイル名を付けて保存]ダイアログボックス構築するときはFALSEを指定します。defExt には、ファイル名用のエディットボックスに、拡張子を付けずに入力したときに、自動的に付加される拡張子を指定します。NULL が指定された場合には拡張子は付加されません。defFileName には、ファイル名入力用のエディットボックスに最初に表示されるファイル名を指定します。NULL が指定指定された場合には表示されません。flags には、以下に示す、スタイルを｜で組み合わせて指定します。

| フラグ | 説明 |
|---|---|
| OFN_ALLOWMULTISELECT | [ファイル名]リストボックスで複数選択を可能にすることを指定する(プライベートなテンプレートを使ってダイアログボックスを作成する場合は、[ファイル名]リストボックスの定義の中に LBS_EXTENDEDSEL 値を指定しなければならない)。 |
| OFN_CREATEPROMPT | 指定されたファイルが存在しない場合に、ダイアログボックス関数がファイルを新しく作成するかどうかをユーザに問い合わせるようにすることを指定する(このフラグは、OFN_PATHMUSTEXIST フラグと OFN_FILEMUSTEXIST フラグを自動的にセットする)。 |
| OFN_FILEMUSTEXIST | ユーザが[ファイル名]エントリフィールドに既存のファイル名しか入力できないことを指定する。このフラグがセットされているときにユーザが[ファイル名]エントリフィールドに無効なファイル名を入力した場合、ダイログボックス関数はメッセージボックスに警告を表示する。このフラグがセットされると、OFN_PATHMUSTEXIST フラグもセットされる。 |

| フラグ（つづき） | 説明 |
| --- | --- |
| OFN_HIDEREADONLY | [書き込み禁止]チェックボックスを非表示にする。 |
| OFN_NOCHANGEDIR | ダイアログボックスに対して、現在のディレクトリをダイアログボックス呼び出し時のディレクトリにリセットさせる。 |
| OFN_NONETWORKBUTTON | [ネットワーク]ボタンを非表示にして使用不能にする。 |
| OFN_NOREADONLYRETURN | 返されたファイルに対して[書き込み禁止]チェックボックスがチェックされておらず、ファイルが書き込み保護のディレクトリにないことを指定する。 |
| OFN_NOTESTFILECREATE | ダイアログボックスがクローズされる前にファイルが作成されないことを指定する。このフラグは、アプリケーションが「作成しても修正しない」ネットワーク共有ポイントにファイルを保存する場合には、セットしなければならない。アプリケーションがこのフラグをセットすると、ライブラリは、書き込み保護、ディスク容量の有無、ドライブのドアの状態、ネットワーク保護などをチェックしなくなる。この状態でいったんファイルをクローズしてしまうと再びオープンすることができなくなる。このため、このフラグを使うアプリケーションは、ファイル操作を注意して行わなければならない。 |
| OFN_OVERWRITEPROMPT | 選択されたファイルがすでに存在する場合に、[ファイル名を付けて保存]ダイアログボックスにメッセージボックスを生成させる。ユーザは、ファイルを上書きするかどうかを確認しなければならない。 |
| OFN_PATHMUSTEXIST | ユーザが有効なパスしか入力できないことを指定する。このフラグがセットされているときにユーザが[ファイル名]エントリフィールドに無効なパスを入力した場合、ダイアログボックス関数はメッセージボックスに警告を表示する。 |
| OFN_READONLY | ダイアログボックスを作成するときに、初期状態で[書き込み禁止]チェックボックスをチェックされた状態にする。また、ダイアログボックスがクローズされるときの[書き込み禁止]チェックボックスの状態を示す。 |

filter には、以下に示すフォーマットで、ファイルを特定するためのフィルタを指定する文字列のペアを指定します。次の例では、(*.m;*.h)と(*.*)のフィルタを指定しています。

"Files(*.m;*.h)|*.m;*.h|All Files(*.*)|*.*||"

フィルタを指定すると、ファイル用のリスト ボックスには選択されたもののみが表示されるようになります。FileName には、選択されたファイル名が格納されます。複数のファイルの選択を許している場合で、複数のファイルが選択された場合には、スペース文字をデリミタとして格納されます。

### 5.3.37._win_set_window_pos: カスタムウィンドウの位置の設定

関数名　int _win_set_window_pos(int x, int y)
引数　int　x　カスタムウィンドウの新しい左辺の X ピクセル座標
　　int　y　カスタムウィンドウの新しい上辺の Y ピクセル座標
戻り値　TRUE　成功
　　FALSE　エラー
機能　カスタムウィンドウの位置を変更します。

### 5.3.38._win_set_window_size: カスタムウィンドウのサイズの設定

関数名　int _win_set_window_size(int cx, int cy)
引数　int　cx　カスタムウィンドウの新しい幅（ピクセル値）
　　int　cy　カスタムウィンドウの新しい高さ（ピクセル値）
戻り値　TRUE　成功
　　FALSE　エラー
機能　カスタムウィンドウのサイズを変更します。

### 5.3.39._win_timer_set: システムタイマの設定

関数名　int _win_timer_set(int nId, int nElapse)
引数　int　nId　0 以外のタイマ識別子
　　int　nElapse タイムアウト値(ミリ秒単位)
戻り値　TRUE　成功
　　FALSE　エラー
機能　nId で指定されたタイマ識別子を持つシステムタイマをセットします。タイムアウト値が指定され、タイムアウトが発生するたびに、システムは引数 nIDEvent にタイマ識別子の値を格納して OnTimer()ハンドラ関数を呼び出します。タイマの解除を行うには_win_timer_kill()関数を用います。

### 5.3.40._win_timer_kill: システムタイマの解除

関数名　int _win_timer_kill(int nId)
引数　int　nId　0以外のタイマ識別子
戻り値　TRUE　成功
　　FALSE　エラー
機能　nIdで指定されたシステムタイマを解除します。

## 5.4.カスタムウィンドウ用ハンドル関数

カスタムウィンドウ作成モードで新規プロジェクトを作成する際に、CB30 によって自動生成されるフレームワークに記述されるハンドル関数は、カスタムウィンドウが Windows のメッセージを受け取った際に呼び出されます。ハンドル関数の一覧を以下に示します。

| ハンドル関数名 | 説明 |
|---|---|
| OnChar | ASCII 文字コードに変換可能なキーが押された際に OnKeyDonn() ハンドル関数に続いて呼び出されます。 |
| OnCommand | コマンドメッセージが送られた際に呼び出されます。 |
| OnCreate | ウィンドウの作成が要求された際に呼び出されます。 |
| OnDestroy | ウィンドウの破壊が要求された際に呼び出されます。 |
| OnDraw | ウィンドウの再描画が要求された際に呼び出されます。 |
| OnEvent | PD30 のイベントが送られた際に呼び出されます。 |
| OnHScroll | 水平スクロールバーがクリックされた際に呼び出されます。 |
| OnKeyDown | キーが押された際に呼び出されます。 |
| OnKeyUp | キーが離された際に呼び出されます。 |
| OnLButtonDblClk | マウスの左ボタンがダブルクリックされた際に呼び出されます。 |
| OnLButtonDown | マウスの左ボタンが押された際に呼び出されます。 |
| OnLButtonUp | マウスの左ボタンが離された際に呼び出されます。 |
| OnMouseMove | マウスカーソルが移動した際に呼び出されます。 |
| OnRButtonDblClk | マウスの右ボタンがダブルクリックされた際に呼び出されます。 |
| OnRButtonDown | マウスの右ボタンが押された際に呼び出されます。 |
| OnRButtonUp | マウスの右ボタンが離された際に呼び出されます。 |
| OnSize | ウィンドウのサイズが変更された際に呼び出されます。 |
| OnTimer | タイマの時間経過によりタイムアウトの時間間隔が通知された際に呼び出されます。 |
| OnVScroll | 垂直スクロールバーがクリックされた際に呼び出されます。 |

### 5.4.1.ハンドル関数に渡されるデータの仕様

ハンドル関数は、カスタムウィンドウが Windows のメッセージを受け取った際に呼び出されます。カスタムウィンドウは、ハンドル関数の呼び出しの際に、メッセージに付属する情報をグローバル変数_HandleMsgBlock の示す領域に格納し、ハンドル関数から参照できるようにします。

以下に、ハンドル関数とグローバル変数_HandleMsgBlock による情報の受け渡しの例を示します。

```
extern  char    _HandleMsgBlock[32];

OnSize()
{
        int     nType;  /* Message data */
        int     cx:;    /* Message data */
        int     cy;     /* Message data */

        /* Restore message data */
        nType = ((int*)_HandleMsgBlock)[0];
        cx = ((int*)_HandleMsgBlock)[1];
        cy = ((int*)_HandleMsgBlock)[2];

        /* Write message handler code hear, please. */

}
```

ハンドル関数の最初で、_HandleMsgBlock に格納されている情報をハンドル関数のローカル変数に格納します。この処理以降、ハンドル関数に渡された情報は、変数として参照可能となります。ハンドル関数に渡される情報は、ハンドル関数毎に異なっています。これらの処理は、フレームワークにデフォルトで記述されています。

### 5.4.2.OnChar ハンドル関数

関数名　OnChar

説明　ASCII 文字コードに変換可能なキーが押された際に OnKeyDown()ハンドル関数に続いて呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| ASCII 文字コード | ４バイト |
| リピートカウント | ４バイト |
| フラグ(未使用) | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

| | | |
|---|---|---|
| int | nChar | ASCII 文字コード値 |
| int | nRepCnt | キーを押しつづけている間にキーストロークを発生させる回数を表すリピートカウント値 |
| int | nFlags | 本バージョンでは未使用 |

### 5.4.3.OnCommand ハンドル関数

関数名　OnCommand
説明　コマンドメッセージが送られた際に呼び出されます。
データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| コマンドＩＤ | ４バイト |
| 通知メッセージ | ４バイト |
| ハンドル | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

```
int     nId         コントロールアイテムのコマンド ID
int     nMsg        コントロールアイテムの通知メッセージ
int     nHandle     コントロールアイテムのハンドル
```

補足　本ハンドル関数は、主にカスタムウィンドウに設定したコントロールアイテムにイベントが発生した際に呼び出されます。nId にはコントロールアイテムを識別する ID 番号が設定され、nMsg には発生したイベントを識別する通知メッセージが設定され、nHandle にはコントロールアイテムのハンドルが設定されます。これらの変数に設定される値は、コントロールアイテムによって異なりますので、詳しくはコントロールアイテムを操作するシステムコール関数の仕様を参照してください。

### 5.4.4.OnCreate ハンドル関数

関数名　OnCreate
説明　ウィンドウの作成が要求された際に呼び出されます。本関数では、コントロールアイテム等の生成、および変数の初期化などを行います。
データ　なし
変数　なし

### 5.4.5.OnDestroy ハンドル関数

関数名　OnDestroy
説明　ウィンドウの破壊が要求された際に呼び出されます。本関数では、確保したヒープ領域の開放などを行います。
データ　なし
変数　なし

### 5.4.6.OnDraw ハンドル関数

関数名　OnDraw
説明　ウィンドウの再描画が要求された際に呼び出されます。本関数は、他のウィンドウなどによって隠れていた部分が表示されるような場合に呼び出されます。本関数では、カスタムウィンドウの再描画などを行います。
データ　なし
変数　なし

### 5.4.7.OnEvent ハンドル関数

関数名　OnEvent

説明　PD30 のイベントが送られた際に呼び出されます。本関数は、PD30 の状態を変更する必要がある場合に呼び出されます。本関数では、必要に応じて、メモリ値の取得、再描画等を行います。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| PD30 のイベント番号 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nEventID　以下の PD30 のイベント番号

| PD30 のイベント番号 | イベントが送られる場合 |
|---|---|
| EVENT_GO | 実行開始 |
| EVENT_STOP | 実行停止 |
| EVENT_RESET | リセット |
| EVENT_STEP | ステップ実行 |
| EVENT_OVER | オーバー実行 |
| EVENT_RETURN | リターン実行 |
| EVENT_PUT_REG | レジスタ値変更 |
| EVENT_REG_PC | PC 値変更 |
| EVENT_PUT_MEM | メモリ値変更 |
| EVENT_LOAD | プログラムロード |
| EVENT_ADD_SYMBOL | アセンブラシンボル追加 |
| EVENT_DEL_SYMBOL | アセンブラシンボル削除 |
| EVENT_SBRK | ソフトウェアブレークポイント変更 |
| EVENT_TRACE_START | トレース計測開始 |
| EVENT_TRACE_END | トレース計測完了 |
| EVENT_TRACE_PASS | トレースポイント通過 |
| EVENT_FUNC | 表示関数変更 |
| EVENT_FILE | 表示ファイル変更 |
| EVENT_UP | 上位関数にスコープ変更 |
| EVENT_DOWN | 下位関数にスコープ変更 |
| EVENT_MAP | マップ変更 |
| EVENT_PATH | サーチパス変更 |
| EVENT_RAMDISP | リアルタイムラムモニタ再描画 |
| EVENT_RAMINFO | リアルタイムラムモニタ設定変更 |
| EVENT_HWBRK | ハードウェブレーク設定変更 |
| EVENT_EXIT | PD 終了 |
| EVENT_FONT | フォント変更 |
| EVENT_TAB | タブストップ値変更 |
| EVENT_CWATCH_UPDATE | C ウォッチウィンドウ再描画 |
| EVENT_SCRIPT_INIT | スクリプトウィンドウ初期化 |
| EVENT_TIME_10MS | 10 ミリ秒ごとのタイマ割り込み |

### 5.4.8.OnHScroll ハンドル関数

関数名　OnHScroll

説明　水平スクロールバーがクリックされた際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| スクロールバーのコード | ４バイト |
| スクロールボックスの位置 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nSBCode　以下のスクロール要求を示すスクロールバーのコード

| 値 | 説明 |
|---|---|
| SB_LEFT | 左端へスクロール |
| SB_ENDSCROLL | スクロール終了 |
| SB_LINELEFT | 左へスクロール |
| SB_LINERIGHT | 右へスクロール |
| SB_PAGELEFT | １ページ左へスクロール |
| SB_PAGERIGHT | １ページ右へスクロール |
| SB_RIGHT | 右端へスクロール |
| SB_THUMBPOSITION | 絶対位置へスクロール(現在位置は nPos で指定される) |
| SB_THUMBTRACK | スクロールボックスを指定位置へドラッグする(現在位置は nPos で指定される) |

int　nPos　nSBCode が SB_ THUMBPOSITION または SB_THUMBTRACK の時の位置

### 5.4.9.OnKeyDown ハンドル関数

関数名　OnKeyDown

説明　キーが押された際に呼び出されます。ただし「システムキー」に該当するキーは対象外になります。「システムキー」は、パソコンの機種等によって定義が異なりますが、通常 Alt キーと他のキーの同時入力等が該当します。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーコード | ４バイト |
| リピートカウント | ４バイト |
| フラグ | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nChar　キーの仮想キーコード値

int　nRepCnt　キーを押しつづけている間にキーストロークを発生させる回数を表すリピートカウント値

int　nFlags　以下の状態フラグ

| ビット | 説明 |
|---|---|
| 0～7 | 未使用。 |
| 8 | 拡張キー。ファンクションキー、数字キーパッド上のキー。（拡張キーの時は１、それ以外の時は０） |
| 11～12 | 未使用。 |
| 13 | 常に０。 |
| 14 | 直前のキー状態。（呼び出される際にキーが押されている時は１、それ以外の時は０） |
| 15 | 常に０。 |

仮想キーコードについては、次ページの「仮想キーコードについて」をご参照ください。

**[仮想キーコードについて]**

Windows では、あらゆる機種をサポートするために、仮想キーを定義しています。Windows は、F1 キーの押下を検出した際に、F1 キーに対応した仮想キーコードへ変換し、アプリケーションに F1 キーの押下を通知します。仮想キーを用いることで、アプリケーションでは、キーボードの違いを意識しなくてもよくなります。

CB30 では、以下の仮想キーコードを使用することができます。

| 仮想キーコード | 対応するキーボードのキー |
|---|---|
| VK_CANCEL | Ctrl + Break |
| VK_BACK | Backspace |
| VK_TAB | Tab |
| VK_CLEAR | Num Lock オフでのテンキーの 5 |
| VK_RETURN | Enter |
| VK_SHIFT | Shift |
| VK_CONTROL | Ctrl |
| VK_MENU | Alt |
| VK_PAUSE | Pause |
| VK_CAPITAL | Casp Lock |
| VK_ESCAPE | Esc |
| VK_SPACE | Spasebar |
| VK_PRIOR | Page Up |
| VK_NEXT | Page Down |
| VK_END | End |
| VK_HOME | Home |
| VK_LEFT | ← |
| VK_UP | ↑ |
| VK_RIGHT | → |
| VK_DOWN | ↓ |
| VK_SNAPSHOT | Print Screen |
| VK_INSERT | Ins |
| VK_DELETE | Del |
| VK_NUMPAD0 | Num Lock オンでのテンキーの 0 |
| VK_NUMPAD1 | Num Lock オンでのテンキーの 1 |
| VK_NUMPAD2 | Num Lock オンでのテンキーの 2 |
| VK_NUMPAD3 | Num Lock オンでのテンキーの 3 |
| VK_NUMPAD4 | Num Lock オンでのテンキーの 4 |
| VK_NUMPAD5 | Num Lock オンでのテンキーの 5 |
| VK_NUMPAD6 | Num Lock オンでのテンキーの 6 |
| VK_NUMPAD7 | Num Lock オンでのテンキーの 7 |
| VK_NUMPAD8 | Num Lock オンでのテンキーの 8 |
| VK_NUMPAD9 | Num Lock オンでのテンキーの 9 |

| 仮想キーコード | 対応するキーボードのキー |
|---|---|
| VK_MULTIPLY | テンキーの * （拡張キーボード） |
| VK_ADD | テンキーの + （拡張キーボード） |
| VK_SUBTRACT | テンキーの - （拡張キーボード） |
| VK_DIVIDE | テンキーの / （拡張キーボード） |
| VK_F1 | ファンクションキの F1 |
| VK_F2 | ファンクションキの F2 |
| VK_F3 | ファンクションキの F3 |
| VK_F4 | ファンクションキの F4 |
| VK_F5 | ファンクションキの F5 |
| VK_F6 | ファンクションキの F6 |
| VK_F7 | ファンクションキの F7 |
| VK_F8 | ファンクションキの F8 |
| VK_F9 | ファンクションキの F9 |
| VK_F10 | ファンクションキの F10 |
| VK_F11 | ファンクションキの F11 （拡張キーボード） |
| VK_F12 | ファンクションキの F12 （拡張キーボード） |
| VK_NUMLOCK | Num Lock |
| VK_SCROLL | Scroll Lock |

0 ～ 9 および A ～ Z は、仮想キーコード値としてそれぞれ、'0' ～ '9' および'A' ～'Z'を使用します。

### 5.4.10.OnKeyUp ハンドル関数

関数名　OnKeyUp

説明　キーが離された際に呼び出されます。ただし「システムキー」に該当するキーは対象外になります。「システムキー」は、パソコンの機種等によって定義が異なりますが、通常 Alt キーと他のキーの同時入力等が該当します。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーコード | ４バイト |
| リピートカウント | ４バイト |
| フラグ | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nChar　キーの仮想キーコード値

int　nRepCnt　キーを押しつづけている間にキーストロークを発生させる回数を表すリピートカウント値
OnKeyUp ハンドル関数が呼び出される際は、1になります。

int　nFlags　以下の状態フラグ

| ビット | 説明 |
|---|---|
| 0～7 | 未使用。 |
| 8 | 拡張キー。ファンクションキー、数字キーパッド上のキー。（拡張キーの時は１、それ以外の時は０） |
| 11～12 | 未使用。 |
| 13 | 常に０。 |
| 14 | 直前のキー状態。（呼び出される際にキーが押されている時は１、それ以外の時は０） |
| 15 | 常に０。 |

仮想キーコードについては、前節の「仮想キーコードについて」をご参照ください。

### 5.4.11.OnLButtonDblClk ハンドル関数

関数名　OnLButtonDblClk

説明　マウスの左ボタンがダブルクリックされた際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nFlags　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　x　マウスカーソルの X ピクセル座標
int　y　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.12.OnLButtonDown ハンドル関数

関数名　OnLButtonDown

説明　マウスの左ボタンが押された際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nFlags　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　x　マウスカーソルの X ピクセル座標
int　y　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.13.OnLButtonUp ハンドル関数

関数名　OnLButtonUp

説明　マウスの左ボタンが離された際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nFlags　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　x　マウスカーソルの X ピクセル座標
int　y　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.14.OnMouseMove ハンドル関数

関数名　OnMouseMove

説明　マウスカーソルが移動した際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nFlags　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　x　マウスカーソルの X ピクセル座標
int　y　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.15.OnRButtonDblClk ハンドル関数

関数名　OnRButtonDblClk

説明　マウスの右ボタンがダブルクリックされた際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nFlags　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　x　マウスカーソルの X ピクセル座標
int　y　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.16.OnRButtonDown ハンドル関数

関数名　OnRButtonDown

説明　マウスの右ボタンが押された際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nFlags　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　x　マウスカーソルの X ピクセル座標
int　y　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.17.OnRButtonUp ハンドル関数

関数名　OnRButtonUp
説明　　マウスの右ボタンが離された際に呼び出されます。
データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| 仮想キーの種類 | ４バイト |
| マウスの X ピクセル座標 | ４バイト |
| マウスの Y ピクセル座標 | ４バイト |

変数　　_HandleMsgBlock より設定される変数を以下に示します。

int　　nFlags　　押されている仮想キー
格納される値は、以下の仮想キーを表す値の論理和である。

| 値 | 説明 |
|---|---|
| MK_CONTROL | Ctrl キー押下 |
| MK_LBUTTON | マウス左ボタン押下 |
| MK_MBUTTON | マウス中央ボタン押下 |
| MK_RBUTTON | マウス右ボタン押下 |
| MK_SHIFT | Shift キー押下 |

int　　x　　マウスカーソルの X ピクセル座標
int　　y　　マウスカーソルの Y ピクセル座標
座標は、常にウィンドウの左上隅からの相対位置になる。

### 5.4.18.OnSize ハンドル関数

関数名　OnSize
説明　　ウィンドウのサイズが変更された際に呼び出されます。
データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| サイズ変更の種類 | ４バイト |
| 新しい幅(ピクセル値) | ４バイト |
| 新しい高さ(ピクセル値) | ４バイト |

変数　　_HandleMsgBlock より設定される変数を以下に示します。

int　　nType　　以下の要求されるサイズ変更の種類

| 値 | 説明 |
|---|---|
| SIZE_MAXIMIZED | 最大表示 |
| SIZE_MINIMIZED | アイコン化 |
| SIZE_RESTORED | サイズ変更されるが、SIZE_MINIMIZED と SIZE_MAXIMIZED は適用されない |
| SIZE_MAXHIDE | 他のいくつかのウィンドウが最大表示された際に、すべてのポップアップウィンドウにメッセージが送られる |
| SIZE_MAXSHOW | 他のいくつかのウィンドウが元のサイズに復元された際に、すべてのポップアップウィンドウにメッセージが送られる |

int　　cx　　クライアント領域の新しい幅(ピクセル値)
int　　cy　　クライアント領域の新しい高さ(ピクセル値)

### 5.4.19.OnTimer ハンドル関数

関数名　OnTimer

説明　タイマの時間経過によりタイムアウトの時間間隔が通知された際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| タイマの識別子 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nIDEvent　タイマの識別番号

### 5.4.20.OnVScroll ハンドル関数

関数名　OnVScroll

説明　垂直スクロールバーがクリックされた際に呼び出されます。

データ　_HandleMsgBlock に格納される情報を以下に示します。

| | |
|---|---|
| スクロールバーのコード | ４バイト |
| スクロールボックスの位置 | ４バイト |

変数　_HandleMsgBlock より設定される変数を以下に示します。

int　nSBCode　以下のスクロール要求を示すスクロールバーのコード

| 値 | 説明 |
|---|---|
| SB_BOTTOM | 一番下までスクロール |
| SB_ENDSCROLL | スクロール終了 |
| SB_LINEDOWN | １行下へスクロール |
| SB_LINEUP | １行上へスクロール |
| SB_PAGEDOWN | １ページ下へスクロール |
| SB_PAGEUP | １ページ上へスクロール |
| SB_THUMBPOSITION | 絶対位置へスクロール(現在位置は nPos で指定される) |
| SB_THUMBTRACK | スクロールボックスを指定位置へドラッグする(現在位置は nPos で指定される) |
| SB_TOP | 一番上までスクロール |

int　nPos　nSBCode が SB_ THUMBPOSITION または SB_THUMBTRACK の時の位置

# 技術サポート連絡書

年　月　日（合計　　枚）

三菱電機セミコンダクタシステム株式会社
マイコンソフトツール部
# 開発ツールサポート窓口行

［電子メール］ support@tool.mesc.co.jp
［大阪地区］ FAX： 06-6398-6191
［東京地区］ FAX： 03-5783-7339
［中部地区］ FAX： 052-221-7318
［九州地区］ FAX： 092-452-1427

※インストーラが生成する以下のテキストファイルもサポート連絡書としてご利用できます。
Windows 98/95/Windows NT 4.0版の場合：\SUPPORT\製品名\SUPPORT.TXT
EWS版の場合：/support/製品名/toolinfo.txt

| ご連絡先 | 製品情報 |
|---|---|
| 会社名： | ソフトウェア： |
| 部署名： | バージョン番号：V. |
| 担当者名： | ライセンスID： |
| 電話番号： | – – – – |
| FAX番号： | ホストマシン： |
| 電子メール： | OS： V. |
| 通信欄： | |

# MEMO

**CB30 V.1.00ユーザーズマニュアル**
**第1版：1998年2月16日発行**
**資料番号：MSD-CB30-UO-980216**

三菱電機株式会社
三菱電機セミコンダクタシステム株式会社

# エミュレータデバッガ　PD30 用　カスタムビルダ
# CB30 V.1.00　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