お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

User's Manual

RENESAS

PDSDK COMキット

リファレンスマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00　2006.09

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

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

製品の内容及び本書についてのお問い合わせ先

インストーラが生成する以下のテキストファイルに必要事項を記入の上、コンタクトセンタ csc@renesas.comまで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス テクノロジ
コンタクトセンタ　csc@renesas.com
ユーザ登録窓口　regist_tool@renesas.com
ホームページ　http://japan.renesas.com/tools

はじめに

PDSDK COM キットは、デバッガ M3T-PDxx/M3T-PDxxSIM の機能を拡張するためのキットです。市販の Windows アプリケーション開発ツールを利用して、M3T-PDxx/M3T-PDxxSIM 用のカスタマイズウィンドウ(アプリケーション)作成や他のアプリケーションとの連携ができます。

本リファレンスマニュアルには、PDSDK COM キットをご使用いただくための基本的な情報を掲載しています。利用する Windows アプリケーション開発ツールの言語仕様、操作方法 などについては、それぞれの製品のマニュアルやオンラインヘルプなどをご参照ください。

対応デバッガ

PDSDK COM キットは、全てのデバッガ M3T-PDxx/M3T-PDxxSIM で利用できるものではありません。PDSDK COM キットを利用し、M3T-PDxx/M3T-PDxxSIM と連携できるデバッガ及びそのバージョンについては、PDSDK COM キットのリリースノートに記述していますので、そちらをご参照ください。

使用権

PDSDK COM キットの使用権は、使用するデバッガ M3T-PDxx/M3T-PDxxSIM の「ソフトウェア使用権許諾契約書」に基づきます。また、PDSDK COM キットは、お客様の製品開発の目的でのみ使用できます。その他の目的では使用できませんのでご注意ください。

技術サポート

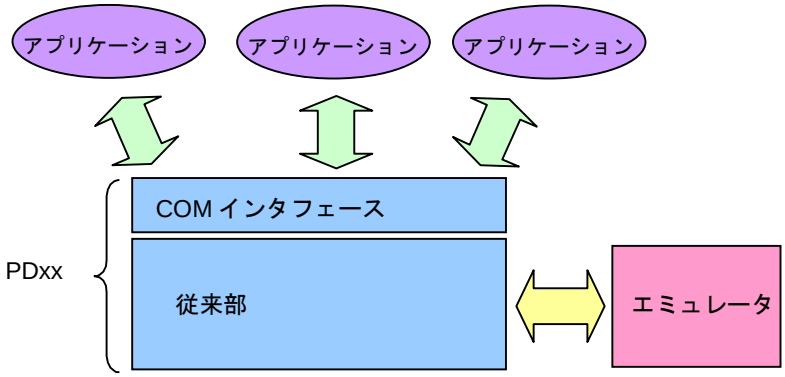
PDSDK COMキットに関する技術サポートは、ホームページ(URL: http://www.renesas.com/jp/tools)に最新情報を掲載する事によって対応させていただきます。お客様からPDSDK COMキットに関してご質問やご要望いただいてもお答えできない場合もありますので、あらかじめご了承願います。

目次

[Memo]

1. 概要

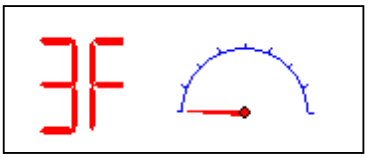
PDSDK COMキットは、デバッガM3T-PDxx/M3T-PDxxSIM(以下、PDxxと記述)の機能を拡張するためのキットです。このキットを使用することにより、PDxx用のカスタマイズウィンドウを作成したり、他のアプリケーションと連携することができます。PDxxには、COM[1]というインタフェースを追加しています。このインタフェースを通して、PDxxの機能を使用することができます。



1.1 利用する開発ツール

カスタマイズウィンドウの作成や他のアプリケーションと連携するには、Microsoft 社の Visual Basic や Visual C++などの COM をサポートした Windows アプリケーション開発ツールを使用します。

- 参考書籍が多数販売されています。アプリケーション作成のための情報も入手しやすくなっています。
- 豊富な GUI 部品を標準で備えています。この GUI 部品をユーザーシステムのシミュレート部品として使用することが可能です。フリーウェアやシェアウェアのコントロール部品(ActiveX コントロール)の利用も可能です。Visual Basic や Visual C++などで独自に作成することも可能です。



- COM インタフェースを持ったアプリケーションと連携することができます。表計算ソフト Microsoft (R) Excel(以下、Excel と記述) などが COM インタフェースを持っており、RAM モニタの結果を Excel に送り、その結果をグラフ化するといった従来の PDxx では実現できなかったことが実現できます。

Visual Basic は、RAD(Rapid Application Development)ツールと呼ばれ、プログラム初心者でも簡単にアプリケーションを作成することができます。

1.2 呼び出し可能なメソッド

PDxx の COM インタフェースを通して呼び出しできるメソッドは、マイコンの実行制御、メモリやレジスタ内容の設定/参照、ソフトウェアブレークポイントの設定などです。独自のカスタマイズツールである CBxx と同じようにデバッガを機能拡張することができます。

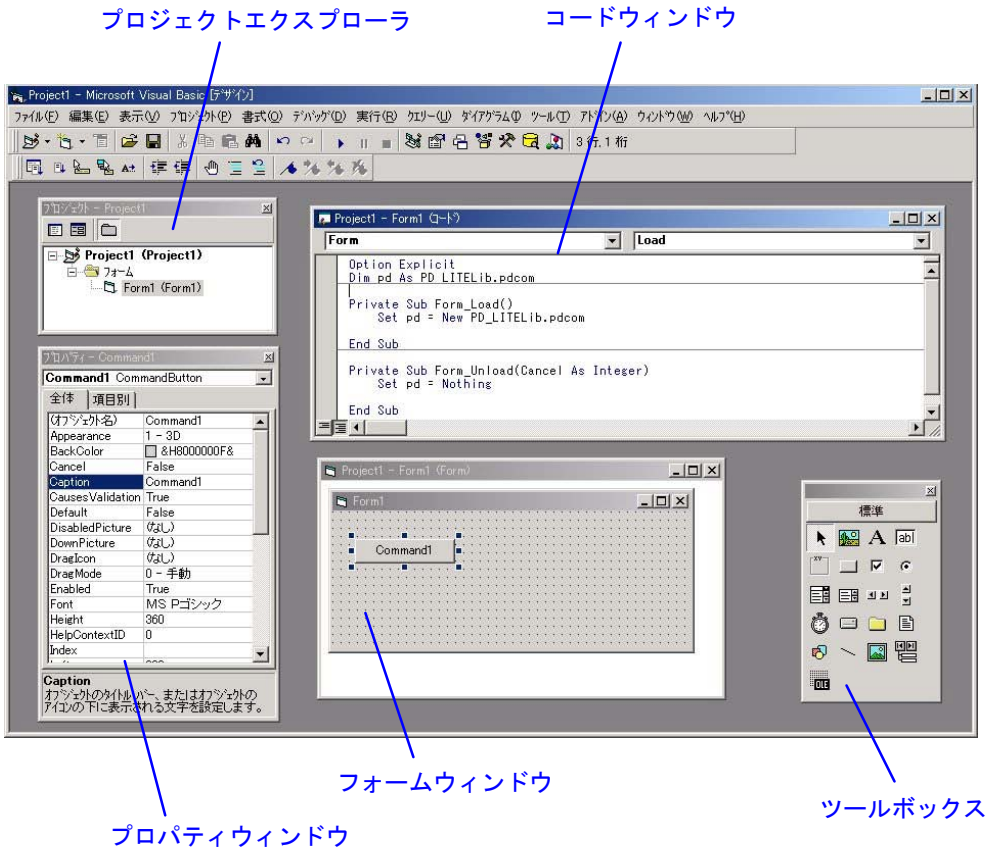
2. セットアップ

PDSDK COM キットのセットアップ手順については、PDSDK COM キットのリリースノートを参照してください。

[1] マイクロソフト社が提唱しているOSやアプリケーション間で連携を行うための規格

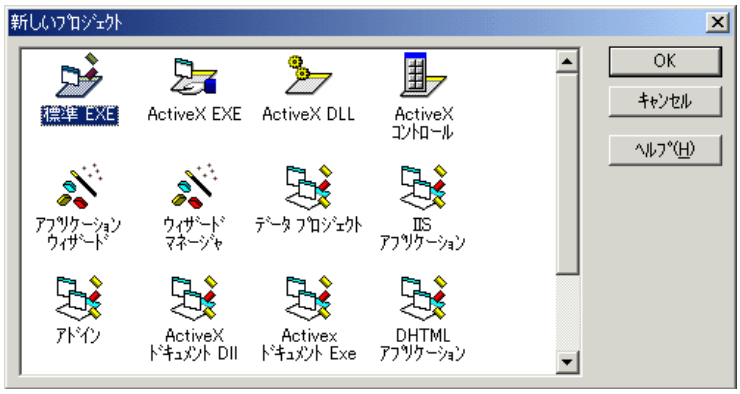
3. ウィンドウの作成方法(Visual Basic)

本章では、Visual Basic 6.0(以下、Visual Basic と記述)を使用した PDxx 用のカスタマイズウィンドウの作成方法について説明します。基本的な言語仕様や操作方法については、Visual Basic のオンラインヘルプや市販の書籍などであらかじめ習得してください。



3.1 プロジェクトの生成

新しいプロジェクトを生成します。Visual Basic のメニュー[ファイル]→[新しいプロジェクト]を選択してください。"新しいプロジェクト"ダイアログがオープンします。そのダイアログで"標準 EXE"を選択し、OK ボタンをクリックしてください。

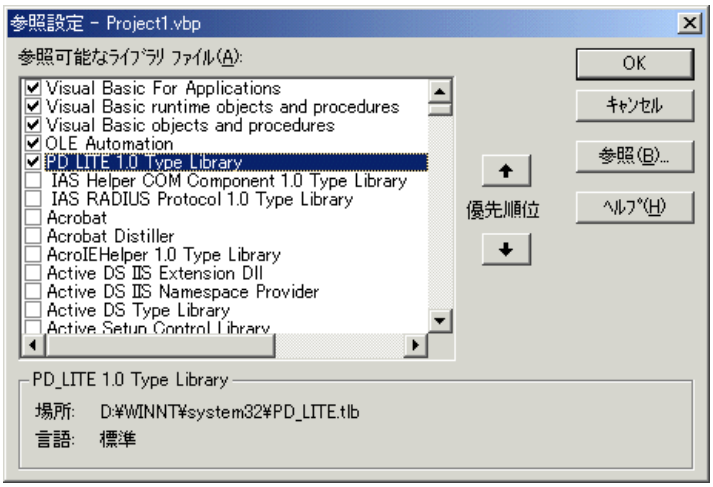


OK ボタンをクリックするとプロジェクトの生成が完了します。

3.2 タイプライブラリの指定

PDxx の COM インタフェースを使用するために、PDxx 用のタイプライブラリファイルを指定します。タイプライブラリファイルとは、COM コンポーネントが公開しているメソッドのメソッド名やそのパラメータを格納したライブラリファイルです。PDxx 用のタイプライブラリファイル名は、pd_lite.tlb です。このファイルは、PDSDK COM キットのインストール時に Windows の SYSTEM ディレクトリへコピーされます(Windows 2000 などでは SYSTEM32 ディレクトリ)。

タイプライブラリを指定するには、Visual Basic のメニュー[プロジェクト]→[参照設定]を選択してください。"参照設定"ダイアログがオープンします。PDxx 用のタイプライブラリは、"PD_LITE 1.0 Type Library"と表示されます。リストボックスから PDxx 用のタイプライブラリを探し、その行のチェックボックスをチェックして、OK ボタンをクリックしてください。



タイプライブラリの指定は、Visual Basic のプロジェクトごとに設定する必要があります。PDxx の COM インタフェースを呼び出すプロジェクトを新規作成する場合は、この作業を必ず実施してください。

3.3 オブジェクトの生成

VB のコードウィンドウで以下のように記述してください。このコードが PDxx の COM インタフェースを呼び出すための基本コードとなります。

```
1:  Dim WithEvents pd As PD_LITELib.pdcom
2:
3:  Private Sub Form_Load()
4:          Set pd = New PD_LITELib.pdcom
5:  End Sub
6:
7:  Private Sub Form_Unload(Cancel As Integer)
8:          Set pd = Nothing
9:  End Sub
```

(各行の説明)

1 行目： ここでは、オブジェクト変数 pd が”PD_LITELib.pdcom”という型であることを宣言しています。この”PD_LITELib.pdcom”は、COM のインタフェース名です。なお、エミュレータ PC7501 対応の PDxx 用メソッドを呼び出す場合は、インタフェース名を”PD_LITELib.Ipdcom7501”としてください。また、WithEvents の記述は、プログラム実行開始やプログラム停止などの PDxx 側で発生したイベントを取得するために指定します。変数名 pd は任意です。

3~5 行目： Form_Load は、アプリケーション起動時(フォームオープン時)に呼び出されるプロシージャ(関数)です。ここでは、変数 pd に”PD_LITELib.pdcom”というオブジェクトを代入しています。PDxx のメソッドは、この変数 pd を通して、呼び出します。

7～9 行目： Form_Unload は、アプリケーション終了時(フォームクローズ時)に呼び出されるプロシージャ(関数)です。ここでは、変数 pd のオブジェクトを破棄しています。オブジェクトを破棄すると、PDxx のメソッドを呼び出すことはできなくなります。

このコードは、Visual Basicのプロジェクトごとに記述する必要があります。PDxxのCOMインタフェースを呼び出すプログラムを新規作成する場合は、「3.2 タイプライブラリの指定」と同様にこの作業を必ず実施してください。

3.4 メソッドの呼び出し

あらかじめ、「3.1、3.2、3.3」の内容に従い、COMインタフェース呼び出しの準備を行ってください。PDxxが公開しているメソッドは、「3.3」で定義したPDxxのオブジェクト変数pdの次に"."を記述し、その後にメソッド名を指定することによって、呼び出すことができます。そのメソッドにパラメータがある場合は、そのパラメータをメソッド名の後に記述します。

pd . xxxxx （パラメータ, ・・・）

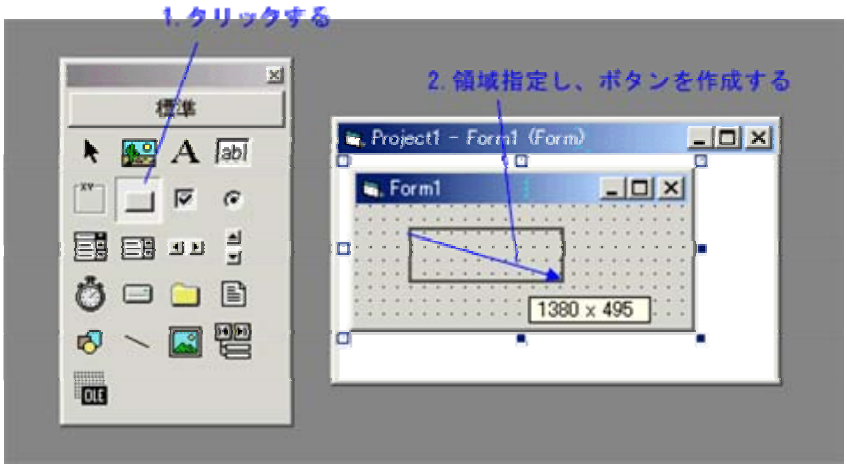
各メソッドの詳細については、「4.」を参照してください。以下、メソッドを呼び出すための具体例な手順を説明します。

ユーザターゲットをリセットする

ユーザターゲットをリセットするカスタマイズウィンドウの作成方法を以下に示します(本キットをインストールしたディレクトリにサンプルを格納しています)。このアプリケーションでは、ボタンコントロールを 1 個使用します。

1. ボタンの追加

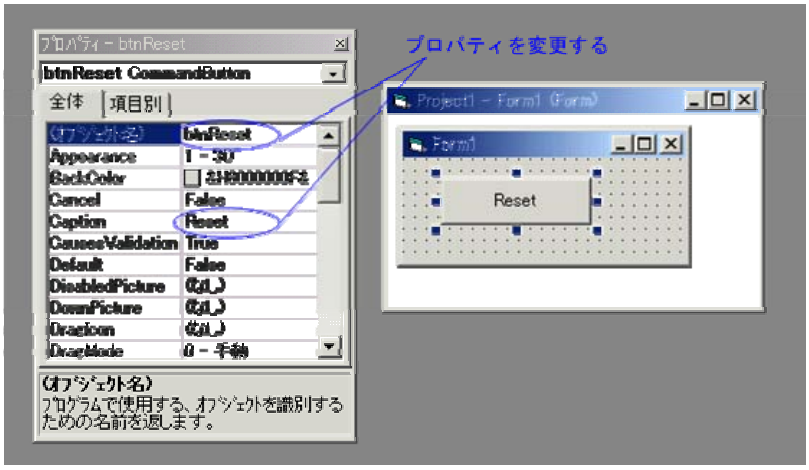
ツールボックスの **CommandButton** をクリックし、フォーム上にボタンコントロールを 1 つ作成してください。



2. ボタンのプロパティ変更

作成したボタンコントロールのプロパティをプロパティウィンドウで変更します。

プロパティ	内容
(オブジェクト名)	btnReset
Caption	Reset



3. ボタンの動作を記述

コードウィンドウで以下のように記述してください。太字箇所が追加した部分です。

```
Dim WithEvents pd As PD_LITELib.pdcom

Private Sub Form_Load()
    Set pd = New PD_LITELib.pdcom
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Set pd = Nothing
End Sub

Private Sub btnReset_Click()
    ret = pd.cpu_reset
    If ret = 0 Then
        pd.err_disp_message
    End If
End Sub
```

4. 動作確認

アプリケーションの動作を確認します。まず、PDxx を起動してください。その後、Visual Basic のメニュー[実行]→[開始]を選択し、アプリケーションを実行してください。アプリケーション上の Reset ボタンをクリックすると、ユーザターゲットがリセットされます。 動作を確認してください。このアプリケーションの実行ファイル(EXE ファイル)を作成するには、Visual Basic のメニュー[ファイル]→[xxxxx.exe の作成]を選択します。保存フォルダと EXE ファイル名を指定してください。このアプリケーションは、エクスプローラなどで直接起動することができます。

3.5 イベントの取得

あらかじめ、「3.1、3.2、3.3」の内容に従い、COMインタフェース呼び出しの準備を行ってください。PDxx側で発生するイベント(プログラム実行開始など)を取得するには、プロシージャ**pd_GotEventMessage**を使用します。 このプロシージャは、PDxx側で発生したイベントを受け取った際に呼び出されるプロシージャです(プロシージャ名の先頭の"pd"はプログラムの先頭で指定したオブジェクトの変数名です)。

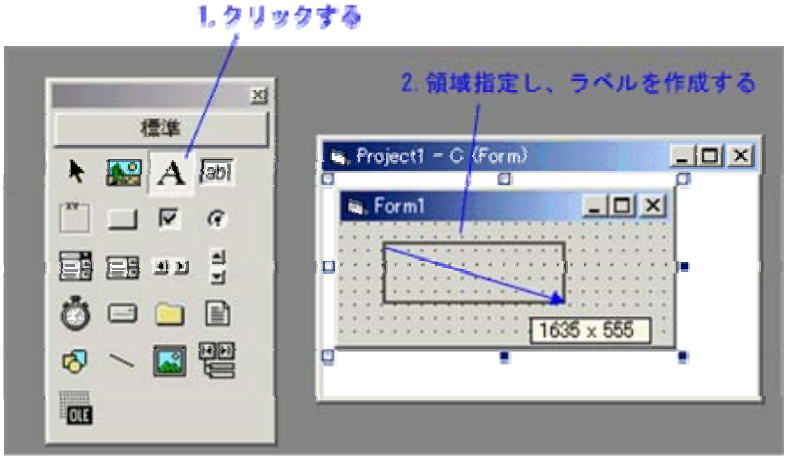
```
Private Sub pd_GotEventMessage(ByVal action As Long)
        :
End Sub
```

パラメータactionにPDxx側で発生したイベント番号が格納されています。このプロシージャでPDxxからイベントを受け取った場合に実行する処理を記述してください。発生したイベント番号については、「5.3 イベント番号一覧」を参照してください。

PDxx側で発生したイベントを取得する

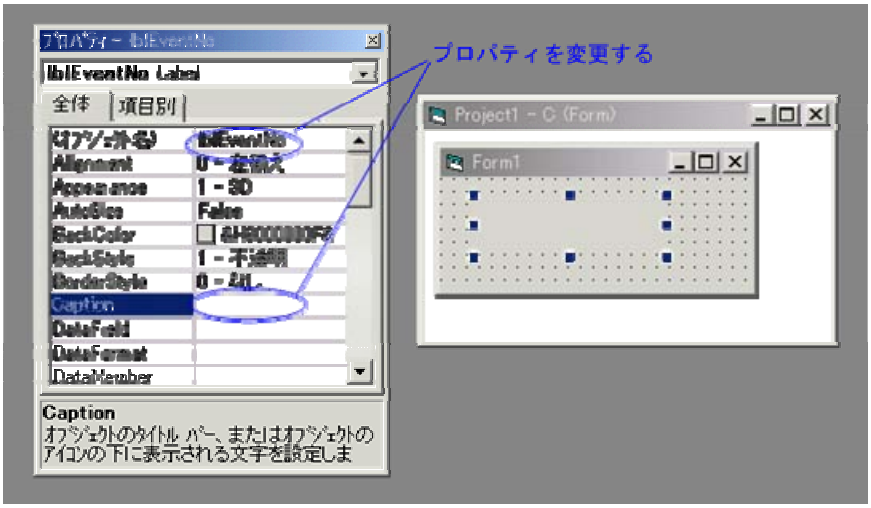
PDxx 側で発生したイベントを取得し、その番号を表示するカスタマイズウィンドウの作成方法を以下に示します(本キットをインストールしたディレクトリにサンプルを格納しています)。このアプリケーションでは、ラベルコントロールを 1 個使用します。

1. ラベルコントロールの追加
   ツールボックスの **Label** をクリックし、フォーム上にラベルコントロールを 1 つ作成してください。



2. プロパティの変更
   作成したラベルコントロールのプロパティをプロパティウィンドウで変更します。

プロパティ	内容
(オブジェクト名)	lblEventNo
Caption	(空欄)



3. イベント発生時の動作を記述
   コードウィンドウで以下のように記述してください。太字箇所が追加した部分です。

```
Dim WithEvents pd As PD_LITELib.pdcom

Private Sub Form_Load()
   Set pd = New PD_LITELib.pdcom
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Set pd = Nothing
End Sub

Private Sub pd_GotEventMessage(ByVal action As Long)
   lblEventNo.Caption = action
End Sub
```

4. 動作確認
   アプリケーションの動作を確認します。まず、PDxx を起動してください。 その後、Visual Basic のメニュー[実行]→[開始]を選択し、アプリケーションを実行してください。 PDxx を操作(Go,Stop など)すると、その PDxx で発生したイベントの番号が表示されます。 動作を確認してください。

4. ウィンドウの作成方法(Visual C++)

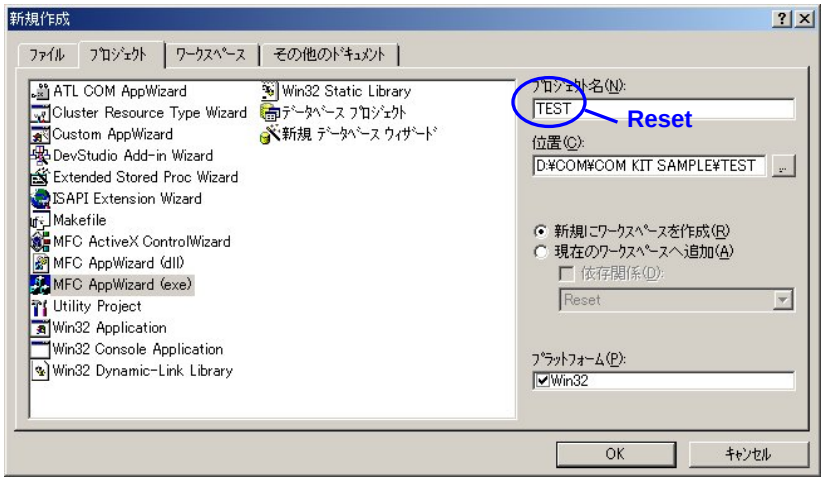
本章では、Visual C++6.0(以下、VC++と記述)を使用した PDxx 用のカスタマイズウィンドウの作成方法について説明します。基本的な言語仕様や操作方法については、VC++のオンラインヘルプや市販の書籍などであらかじめ習得してください。

この章では PDxx を RESET するサンプルを作成します。

4.1 プロジェクトの生成

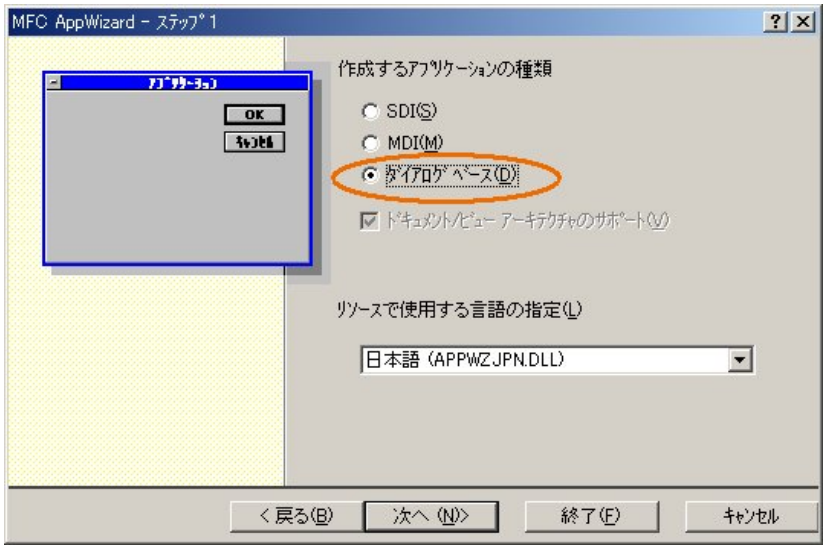
新しいプロジェクトを生成します。VC++のメニュー[ファイル]→[新規作成]を選択してください。

「新規作成」ウィザードがオープンします。そのウィザードで"MFC AppWizard (exe)"を選択し、OK ボタンをクリックしてください。



この例では、プロジェクト名を”Reset”としています。

「ステップ１」で作成するアプリケーションの種類を指定します。今回はダイアログベースのアプリケーションを作成します。次へボタンをクリックして下さい。



ステップ 2 で「オートメーション」をチェックしてください。その他の設定はデフォルトのままでかまいません。

これ以降の手順はデフォルトのまま、進めてください。

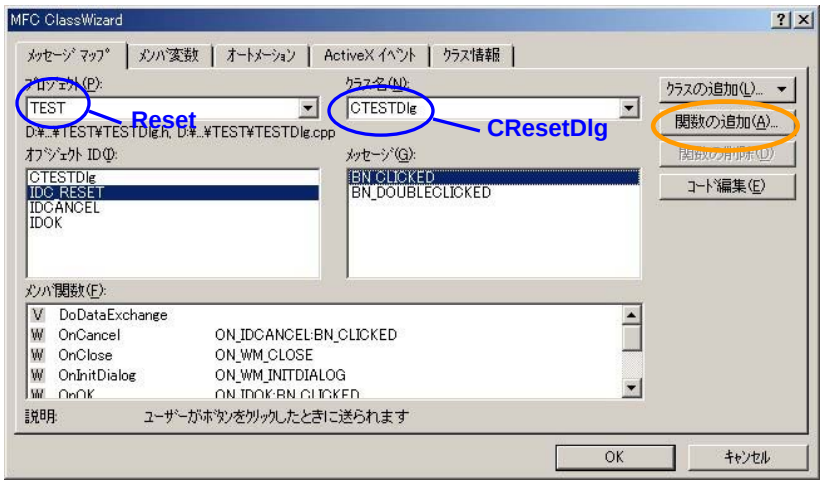
4.2 ボタンの作成

プロジェクトが作成できたら、ダイアログ上にボタンを作成します。ボタンの ID とキャプションは以下のとおりです。

ID	キャプション	コントロール
ID_RESET	Reset	ボタン
IDC_EVENT_NO	(なし)	スタティックテキスト

4.2.1 メッセージの追加

次にボタンが押されたときに呼び出される関数を ClassWizard で追加します。



「関数の追加ボタン」を押すと、CResetDlg.cpp に以下のような空の関数が挿入されます。

```
void CResetDlg::OnReset()
{
    // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください

}
```

この関数を実装します。

```
#import "PD_LITE.TLB" no_namespace.....(1)

        …
        …
        …

void CResetDlg::OnReset()
{
    // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
    IpdcomPtr p( __uuidof(pdcom) ); .....(2)
    p->cpu_reset(); .....(3)
}
```

(1)PDCOM のタイプライブラリをインポートします。PDCOM のインターフェース、メソッドを使用するために必要です。

(2)p は PDCOM のインターフェースのスマートポインタです。スマートポインタ名は、I インターフェース名 Ptr です。PDCOM のインターフェース名は pdcom なので、スマートポインタは IpdcomPTR となります。

(3)メソッド cpu_reset()を呼び出します。

以上の作業が終了したらプロジェクトをビルドしてください。

4.3 PDxx からのイベントの取得

ターゲットプログラムの実行、停止やメモリ、レジスタ値の変更に連動してカスタマイズウィンドウの表示を変更するには、PDxx からイベントを取得する必要があります。

以下、項目4.1~4.2.1で作成したリセットウィンドウにイベントの取得機能を実装する方法を説明します。

4.3.1 シンクオブジェクトの作成の準備（ATL サポート）

サーバー（PDxx）から通知を受け取るオブジェクト（シンクオブジェクト）を作成します。シンクオブジェクトは、ATL の機能を利用して作成するので、プロジェクトに ATL サポートを追加する必要があります。VC++のメニュー[挿入]→[ATL オブジェクトの新規作成]を選択します。以下のダイアログがオープンするので、「はい」を選択します。

次に以下の「ATL オブジェクトウィザード」ダイアログがオープンしますが、ここでは「キャンセル」ボタンを押し、終了します。

4.3.2 シンクオブジェクトの作成

シンクオブジェクトを手作業で作成します。シンクオブジェクトのヘッダファイルは以下のとおりです（ファイル名：ResetEvents.h）。

```
#ifndef       __RESETEVENTS_H_
#define       __RESETEVENTS_H_

#include "resource.h"

class CpdcomEvents :
    public CCcomObjectRoot,
    public IDispatchImpl<_IpdcomEvents, &DIID__IpdcomEvents, &LIBID_PD_LITELib>…(1)
{
public:
    CpdcomEvents() : m_hWnd(NULL) {}

BEGIN_COM_MAP(CpdcomEvents)
    COM_INTERFACE_ENTRY(IDispatch)
    COM_INTERFACE_ENTRY_IID(DIID__IpdcomEvents, IDispatch)
END_COM_MAP()

// Idispatch
    STDMETHOD(Invoke)(DISPID dispid, REFIID riid, LCID lcid,……(2)
        WORD wFlags, DISPPARAMS* pdispParams, VARIANT* pvarResult,
        EXCEPINFO* pexcepinfo, UINT* puArgErr);

void SetHWND(HWND hWnd) { m_hWnd = hWnd; }……(3)

private:
    HWND m_hWnd;……(4)

public:
};

#endif
```

(1) _IpdcomEvents はサーバー（PDxx）で実装しているシンクオブジェクトです。

(2) Invoke()はサーバーから呼び出される関数です。この関数に、イベントが通知された場合の処理を実装します。

(3) SetHWND（）でメンバ変数 m_hWnd にダイアログのハンドルを設定します。

(4) メンバ変数 m_hWnd は今回作成したダイアログのハンドルを保持する変数です。COM、ATL とは直接関係ありません。

シンクオブジェクトの実装は以下のとおりです（ファイル名：ResetEvents.cpp）。

```
#include "stdafx.h"

#import "pd_lite.tlb" no_namespace, named_guids.....(1)


#include "resetEvents.h"
#include "ResetDlg.h"


STDMETHODIMP CpdcomEvents::Invoke(DISPID dispid, REFIID riid, LCID lcid,
                        WORD wFlags, DISPPARAMS* pdispParams,
                        VARIANT* pvarResult, EXCEPINFO* pexcepinfo, UINT* puArgErr )
{
    if(riid!= IID_NULL)
        return DISP_E_UNKNOWNINTERFACE;

    HRESULT hr;
    CComVariant varMessage;
    USES_CONVERSION;
    switch (dispid ) {
    case 1:
        if( !(wFlags & DISPATCH_METHOD))
            return DISP_E_MEMBERNOTFOUND;
        if(pdispParams->cArgs != 1)
            return DISP_E_BADPARAMCOUNT;
        hr = varMessage.ChangeType(VT_BSTR, (&pdispParams->rgvarg[0])); .....(2)
        if(FAILED(hr))
            return DISP_E_TYPEMISMATCH;
        SetDlgItemText(m_hWnd, IDC_EVENT_NO, OLE2T(varMessage.bstrVal)); .....(3)(4)

        return S_OK;
    default:
        return DISP_E_MEMBERNOTFOUND;
    }
}
```

(1) named_guids 属性を指定すると、コンパイラは、LIBID_MyLib、CLSID_MyCoClass、IID_MyInterface、および DIID_MyDispInterface の形式の GUID 変数を古いスタイルで定義し初期化します。

(2) pdispParams->rgvarg[0]に PDxx のイベント番号が格納されています。この例では、ダイアログに文字を表示するため、ChangeType を用いて VT_BSTR 型に変換しています。数値に変換するには

```
hr = varMessage.ChangeType(VT_I4, (&pdispParams->rgvarg[0]));
```

のように記述します（VT_I4、long に変換する場合）。

(3) OLE2T は LPOLESTR を LPTSTR に変換するマクロです。

(4) この例では、SetDlgItemText()を用いて、イベントを文字列としてダイアログに表示しています。

作成した ResetEvents.cpp、ResetEvents.h をプロジェクトに追加してください。

4.3.3 ダイアログの変更

シンクオブジェクトを使用するために、ダイアログのソースを変更します。

ヘッダファイル（ResetDlg.h）に以下の宣言を追加します（下線部）。

```
            …
class CresetDlgAutoProxy;
interface Ipdcom;
class CpdcomEvents;
            …
```

次にメンバ変数を追加します。

```
private:
    CComPtr <Ipdcom> m_ppdcom;
    CComObject<CpdcomEvents>* m_ppdcomEvents;
    DWORD m_dwCookie;
```

m_ppdcom はインターフェースのポインタ、m_ppdcomEvents はシンクオブジェクト、m_dwCookie はシンクオブジェクトとサーバーの接続を一意に決める識別子（クッキー）です。

次にソースファイル（ResetDlg.cpp）を変更します。
まず、ダイアログのコンストラクタに、シンクオブジェクトを初期化するコードを追加します（下線部）。

```
#import "pd_lite.tlb" no_namespace, named_guids
#include "resetEvents.h"

                ……
                ……
                ……
                ……

CResetDlg::CResetDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CResetDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CResetDlg)
        // メモ: この位置に ClassWizard によってメンバの初期化が追加されます。
    //}}AFX_DATA_INIT
    // メモ: LoadIcon は Win32 の DestroyIcon のサブシーケンスを要求しません。
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    m_pAutoProxy = NULL;
    m_ppdcomEvents = NULL;
}
```

OnInitDialog()にインターフェースオブジェクトの作成、シンクオブジェクトの作成、Iunknown インターフェースの取得のコードを挿入します。

```
    // TODO: 特別な初期化を行う時はこの場所に追加してください。
    // インターフェースオブジェクトの作成
    HRESULT hr = CoCreateInstance(CLSID_pdcom, NULL, CLSCTX_ALL,
                                  IID_Ipdcom, (void**) &m_ppdcom);

    if (FAILED(hr)) {
        PostMessage(WM_CLOSE, 0, 0L);
        return TRUE;
    }

    // シンクオブジェクトの作成
    CComObject<CpdcomEvents>::CreateInstance(&m_ppdcomEvents);
    if (!m_ppdcomEvents) {
        m_ppdcom = NULL;
        PostMessage(WM_CLOSE, 0, 0L);
        return TRUE;
    }

    // IUnknown インターフェースの取得
    CComPtr<IUnknown> pEventUnk = m_ppdcomEvents;
    // シンクオブジェクトの通知
    hr = AtlAdvise(m_ppdcom, pEventUnk, DIID__IpdcomEvents, &m_dwCookie);
    if (FAILED(hr)) {
        m_ppdcom = NULL;
        pEventUnk = NULL;
        PostMessage(WM_CLOSE, 0, 0L);
        return TRUE;
    }

    m_ppdcomEvents->SetHWND(m_hWnd);
```

m_ppdcomEvents->SetHWND(m_hWnd)はダイアログに文字列を表示するため、ダイアログのハンドルを設定しています。COM、ATL とは直接関係ありません。

最後に、デストラクタに接続を解除するコードを挿入します(下線部)。

```
CResetDlg::~CResetDlg()
{
    // このダイアログ用のオートメーション プロキシがある場合は、このダイアログ
    // へのポインタを NULL に戻します、それによってダイアログが削除されたこと
    // がわかります。
    if (m_pAutoProxy != NULL)
        m_pAutoProxy->m_pDialog = NULL;

    if (m_ppdcom != NULL) {
        AtlUnadvise(m_ppdcom, DIID__IpdcomEvents, m_dwCookie);
    }
    m_ppdcomEvents = NULL;
    m_ppdcom = NULL;
}
```

5. メソッド一覧

PDxxは、以下のメソッドを公開しています。メソッドの詳細は「5.2 メソッド詳細」を参照してください。各メソッドで記述している使用例は、Microsoft社のVisual Basic V.6.0 を使用した場合の例です。

5.1 メソッド概要

実行制御

メソッド名	パラメータ	説明	ページ
cpu_go	(なし)	プログラムを実行する(ブレークポイント無効)	32
cpu_gb	(なし)	プログラムを実行する(ブレークポイント有効)	31
cpu_wait	(なし)	プログラムの停止するまで待つ	40
cpu_stop	(なし)	プログラムを停止する	39
cpu_reset	(なし)	プログラムをリセットする	34
cpu_step	(なし)	プログラムをステップ実行する(機械語単位)	38
cpu_src_step	(なし)	プログラムをステップ実行する(ソース行単位)	37
cpu_over	(なし)	プログラムをオーバーステップ実行する(命令単位)	33
cpu_src_over	(なし)	プログラムをオーバーステップ実行する(ソース行単位)	36
cpu_return	(なし)	プログラムをリターン実行する	35
cpu_check_run	[out] long *status	プログラムが実行中かチェックする	30

レジスタ操作

メソッド名	パラメータ	説明	ページ
reg_get_pc	[out] long* pc	プログラムカウンタの値を取得する	77
reg_set_pc	[in] long pc	プログラムカウンタの値を設定する	79
reg_get_reg	[in] long regNo [out] long* regVal	レジスタの値を取得する	78
reg_set_reg	[in] long regNo [in] long regVal	レジスタの値を設定する	80

メモリ操作

メソッド名	パラメータ	説明	ページ
mem_get	[in] long addr [in] long length [out] long* data	メモリの値を取得する(1 データ)	62
mem_get_multi	[in] long addr [in] long num [in] long length [out] VARIANT* data	メモリの値を取得する(データ個数指定あり)	64
mem_set	[in] long addr [in] long length [in] long data	メモリの値を設定する(1 データ)	67
mem_set_multi	[in] long addr [in] long num [in] long length [in] VARIANT data	メモリの値を設定する(データ個数指定あり)	69
mem_get_bit	[in] long addr [in] long bit	ビット値を取得する	63

	[out] long* data		
mem_set_bit	[in] long addr [in] long bit	ビットセットする	68
mem_clear_bit	[in] long addr [in] long bit	ビットクリアする	61
mem_fill	[in] long addr [in] long num [in] long length [in] long data	メモリブロックを任意の値で充填する	65
mem_move	[in] long addr [in] long num [in] long top [in] long lenght	メモリブロックを別のブロックにコピーする	66

RAMモニタ操作

メソッド名	パラメータ	説明	ページ
rram_get_area	[out] long* addr	RAM モニタ領域の先頭アドレスを取得する	72
rram_set_area	[in] long addr	RAM モニタ領域の先頭アドレスを設定する	75
rram_get_data	[in] long addr [in] long num [out] VARIANT* data [out] VARIANT* attr	RAM モニタデータを取得する	74
rram_clear	(なし)	RAM モニタの内容をクリアする	70
rram_get_area_blk*	[in] long blkno [out] long* addr	指定したブロック番号の RAM モニタの先頭アドレスを取得する	73
rram_set_area_blk*	[in] long blkno [in] long addr	指定したブロック番号の RAM モニタの先頭アドレスを設定する	76
rram_clear_blk*	[in] long blkno	指定したブロック番号の RAM モニタの内容をクリアする	71

*このメソッドは、連携する PDxx によっては呼び出しできない場合があります。詳細は、「PDSDK COM キット リリースノート」をご参照ください。

ソフトウェアブレーク

メソッド名	パラメータ	説明	ページ
break_set	[in] long addr	ブレークポイントを設定する	29
break_reset	[in] long addr	ブレークポイントの設定を解除する	26
break_reset_all	(なし)	すべてのブレークポイントの設定を解除する	27
break_disable	[in] long addr	ブレークポイントを一時無効にする	22
break_disable_all	(なし)	すべてのブレークポイントを一時無効にする	23
break_enable_all	(なし)	すべてのブレークポイントを有効にする	24
break_get	[out] long* addr [out] long* attr [in] long mode	ブレークポイントを検索する	25
break_search	[in] long addr [out] long* attr	ブレークポイントの設定状態をチェックする	28

リアルタイムトレース

メソッド名	パラメータ	説明	ページ
rtt_get_range	[out] long *s_cycle [out] long *e_cycle	トレースサイクル範囲を取得する	85

rtt_get_bus	[in] long cycle [out] long *addr [out] BSTR *buffer	バス情報文字列を取得する	83
rtt_get_disasm	[in,out] long *cycle [out] long* next_cycle [out] BSTR *buffer [out] long* count	逆アセンブル文字列を取得する	84
rtt_check_isfetch	[in] long cycle [out] long* addr1 [out] long* addr2 [out] long* count	フェッチサイクルであるか、検査する	81
rtt_clear	(なし)	トレース結果をクリアする	82

カバレッジ計測

メソッド名	パラメータ	説明	ページ
cv_get_base*	[out] long *base	カバレッジベースアドレスを取得する	45
cv_set_base*	[in] long base	カバレッジベースアドレスを設定する	51
cv_get_data	[in] long st_addr [in] long en_addr [out] long *rs_addr [out] long *re_addr [out] VARIANT *data	カバレッジデータを取得する	49
cv_clear*	(なし)	カバレッジデータをクリアする	43
cv_get_blkcnt*	[out] long *cnt	有効なカバレッジブロック数を取得する	48
cv_clear_blk*	[in] long blkno	指定したブロックのカバレッジデータをクリアする	44
cv_get_base_all*	[out] VARIANT *base	全ブロックのカバレッジベースアドレスを取得する	46
cv_get_base_blk*	[in] long blkno [out] long *base	指定したブロックのカバレッジベースアドレスを取得する	47

*このメソッドは、連携する PDxx によっては呼び出しできない場合があります。詳細は、「PDSDK COM キット リリースノート」をご参照ください。

シンボル操作

メソッド名	パラメータ	説明	ページ
exp_eval	[in] BSTR str [in] long radix [in] long mode [out] long* data	アセンブラ/C 式に該当する数値を返す	56
sym_sym2val	[in] long mode [in] BSTR symbol [out] long* value	ラベル/シンボルの値を取得する	104
sym_val2sym	[in] long mode [in] long value [out] BSTR* symbol	指定値に該当するラベル/シンボルを取得する	106
sym_bit2val	[in] BSTR symbol [out] long* addr [out] long* bit	ビットシンボルのアドレスとビットを取得する	92
sym_val2bit	[in] long addr [in] long bit [out] BSTR *symbol	指定ビットに該当するビットシンボルを取得する	105
sym_addr2line	[in] long addr	指定アドレスに該当する行番号を取得する	91

	[out] long* line [out] BSTR* filename		
sym_line2addr	[in] BSTR filename [in] long line [out] long* addr	指定行番号に該当するアドレスを取得する	100
sym_add_label	[in] BSTR name [in] long value	ラベルを登録する	89
sym_add_symbol	[in] BSTR name [in] long value	シンボルを登録する	90
sym_add_bitsymbol	[in] BSTR name [in] long addr [in] long bit	ビットシンボルを登録する	88
sym_get_disp_src	[out] BSTR *filename	表示中のソースファイル名を取得する	93
sym_set_disp_src	[in] BSTR filename [in] long line [in] long addr	表示するソースファイルを変更する	101
sym_get_scope	[in] long addr [out] BSTR *filename	現在のスコープを取得する	97
sym_set_scope_addr	[in] long addr	アドレス指定によってスコープを設定する	102
sym_set_scope_obj	[in] BSTR obj	オブジェクト指定によってスコープを設定する	103
sym_get_obj_name	[out] BSTR *obj [in] long mode	ロードモジュール中のオブジェクト名を取得する	96
sym_get_src_name	[in] BSTR obj [out] BSTR *src [in] long mode	オブジェクト中のソースファイルの名前を取得する	98
sym_get_func_name	[in] BSTR obj [out] BSTR *func [in] long mode	オブジェクト中の関数の名前を取得する	95
sym_get_func_info	[in] BSTR func [out] BSTR *type [out] long *start [out] long *end [out] BSTR *str	指定した関数の情報を取得する	94
sym_get_variable_info	[in] BSTR var [out] BSTR *type [out] long *l_data [out] long *h_data [out] BSTR *str1 [out] BSTR *str2	指定した C 変数の情報を取得する	99

ダウンロード

メソッド名	パラメータ	説明	ページ
down_load	[in] BSTR filename [in] long mode	ターゲットプログラムをダウンロードする	52

MCU/製品依存情報

メソッド名	パラメータ	説明	ページ
info_cpu	[in] long flag [out] long* status	MCU 依存情報を取得する	57
info_service	[in] long flag	製品依存情報を取得する	60

	[out] long* status		
info_get_debugger	[out] BSTR *name [out] BSTR *dir [out] BSTR *ini	製品名などのデバッガ情報を取得する	58
info_get_mcufile_name	[out] BSTR *name	INIT ダイアログで選択された MCU ファイル名を取得する	59

その他

メソッド名	パラメータ	説明	ページ
com_send	[in] BSTR data [in] long size	エミュレータへバイト列を送信する	42
com_receive	[out] BSTR* data [in] long size	エミュレータからバイト列を受信する	41
err_disp_message	(なし)	エラーメッセージを表示する	53
err_get_message	[out] BSTR *msg	メソッド呼び出しで発生したエラー文字列を取得する	54
event_set_request_mode*	[in] long mode	COM クライアントへのイベント出力可否を指定する	55
scri_command*	[in] BSTR string	スクリプトコマンドを実行する	86
scri_print*	[in] BSTR string	スクリプトウィンドウに文字列を表示する	87

*このメソッドは、連携する PDxx によっては呼び出しできない場合があります。詳細は、「PDSDK COM キット リリースノート」をご参照ください。

5.2 メソッド詳細

break_disable

内容

指定したアドレスのソフトウェアブレークポイントを一時的に無効にします。

パラメータ

ret = pd.break_disable (addr)

属性	型	内容
[in]	long addr	無効にするブレークポイントのアドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。
エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス F0000h 番地のソフトウェアブレークポイントを一時的に無効化
Dim ret as Long
Dim addr as Long
    addr = &hF0000
    ret = pd.break_disable ( addr )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

break_disable_all

内容

すべてのソフトウェアブレークポイントを一時的に無効にします。

パラメータ

ret = pd.break_disable_all ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'すべてのソフトウェアブレークポイントを一時的に無効化
Dim ret as Long
   ret = pd.break_disable_all
   If ret = 0 Then
      pd.err_disp_message
   End If
```

break_enable_all

内容

すべてのソフトウェアブレークポイントを有効にします。

パラメータ

ret = pd.break_enable_all ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'すべてのソフトウェアブレークポイントを有効化
Dim ret as Long
   ret = pd.break_enable_all
   If ret = 0 Then
      pd.err_disp_message
   End If
```

break_get

内容

ソフトウェアブレークポイントに設定されているアドレスを検索し、そのアドレスと設定状態(有効または無効)を返します。

パラメータ

ret = pd.break_get (addr, attr, mode)

属性	型	内容
[out]	long *addr	ブレークポイントに設定されているアドレス
[out]	long *attr	ブレークポイントの設定状態 **256**：IN1_ENABLE_SBRK　有効 **257**：IN1_DISABLE_SBRK　無効
[in]	long mode	ブレークポイントの検索モード **258**：(IN1_FIRST)　1 点目 **259**：(IN1_NEXT)　2 点目以降

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。
エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Public Const IN1_ENABLE_SBRK = 256
Public Const IN1_DISABLE_SBRK = 257
Public Const IN1_FIRST = 258
Public Const IN1_NEXT = 259
Dim ret as Long
Dim addr(64) as Long
Dim attr(64) as Long
Dim mode as Long
Dim i as Integer
   'ソフトウェアブレークポイントの 1 点目を取得
   ret = pd.break_get ( addr(0), attr(0), IN1_FIRST )
   If ret = 0 Then
      pd.err_disp_message
   End If
   'ソフトウェアブレークポイントの 2 点目以降を取得
   For i = 1 To 63
      ret = hc.break_get(addr(i), attr(i), IN1_NEXT)
      If ret = 0 Then
         GoTo next_step
   End If
   Next i%
   next_step:
```

break_reset

内容

指定したアドレスのソフトウェアブレークポイントを解除します。

パラメータ

ret = pd.break_reset (addr)

属性	型	内容
[in]	long addr	解除するブレークポイントのアドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス F0000h 番地のソフトウェアブレークポイントを解除
Dim ret as Long
Dim addr as Long
    addr = &hF0000
    ret = pd.break_reset ( addr )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

break_reset_all

内容

すべてのソフトウェアブレークポイントを解除します。

パラメータ

ret = pd.break_reset_all ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。
エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'すべてのソフトウェアブレークポイントを解除
Dim ret as Long
   ret = pd.break_reset_all
   If ret = 0 Then
      pd.err_disp_message
   End If
```

break_search

内容

指定したソフトウェアブレークポイントアドレスの設定状態(有効または無効)を返します。

パラメータ

ret = pd.break_search (addr, attr)

属性	型	内容
[in]	long addr	参照するブレークポイントのアドレス
[out]	long *attr	ブレークポイントの設定状態 **256 :** (IN1_ENABLE_SBRK)　有効 **257 :** (IN1_DISABLE_SBRK)　無効

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Public Const IN1_ENABLE_SBRK = 256
Public Const IN1_DISABLE_SBRK = 257
Dim ret as Long
Dim addr as Long
Dim attr as Long
   'アドレス F0000h 番地のソフトウェアブレークポイントの設定状態を参照
   addr = &hF0000
   ret = pd.break_search ( addr, attr )
   If ret = 0 Then
      pd.err_disp_message
   End If
   If attr = IN1_ENABLE_SBRK Then
   '設定有効
   Else
   '設定無効
   End If
```

break_set

内容

指定したアドレスをソフトウェアブレークポイントに設定します。ソフトウェアブレークポイントは、63 点まで指定できます。

パラメータ

ret = pd.break_set (addr)

属性	型	内容
[in]	long addr	ブレークポイントに設定するアドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス F0000h 番地をソフトウェアブレークポイントに設定
Dim ret as Long
Dim addr as Long
    addr = &hF0000
    ret = pd.break_set ( addr )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

cpu_check_run

内容

現在のプログラムの実行状態(実行あるいは停止)を返します。

パラメータ

ret = pd.cpu_check_run (status)

属性	型	内容
[out]	long *status	動作状態 **260**：(IN1_RUN_CPU) 実行状態 **261**：(IN1_STOP_CPU) 停止状態

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
Dim status as Long
   ret = pd.cpu_check_run (status)
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_gb

内容

現プログラムカウンタ位置より、プログラムをブレーク付きで実行します。

パラメータ

ret = pd.cpu_gb ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_gb
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_go

内容

現プログラムカウンタ位置より、プログラムをフリーラン実行します。

パラメータ

ret = pd.cpu_go ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_go
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_over

内容

現プログラムカウンタ位置より、プログラムをオーバーステップ実行します(1 命令)。

パラメータ

ret = pd.cpu_over ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_over
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_reset

内容

プログラムをリセットします。

パラメータ

ret = pd.cpu_reset ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_reset
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_return

内容

現プログラムカウンタ位置より、プログラムをリターン実行します。リターン実行は、サブルーチン/割り込みからの復帰命令(以下の命令)を実行するまでオーバーステップ実行する機能です。

製品名	命令名
PD32R(SIM)	JMP R14,RTE
PD308(SIM), PD30(SIM)	REIT,RTS,FREIT,EXITD
PD79(SIM)	RTS,RTI,RTL,RTSD,RTLD
PD77(SIM)	RTS,RTI,RTL
PD38(SIM)	RTS,RTI

パラメータ

ret = pd.cpu_return ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_return
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_src_over

内容

現プログラムカウンタ位置より、プログラムをオーバーステップ実行します(1 ソース行)。

パラメータ

ret = pd.cpu_src_over ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_src_over
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_src_step

内容

現プログラムカウンタ位置より、プログラムをステップ実行します(1 ソース行)。

パラメータ

ret = pd.cpu_src_step ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_src_step
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_step

内容

現プログラムカウンタ位置より、プログラムをステップ実行します(1 命令)。

パラメータ

ret = pd.cpu_step ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_step
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_stop

内容

プログラムを停止します。

パラメータ

ret = pd.cpu_stop ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
   ret = pd.cpu_stop
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cpu_wait

内容

プログラムの実行が停止するまで、処理を待ちます。

パラメータ

ret = pd.cpu_wait ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
Dim ret as Long
    ret = pd.cpu_gb
    ret = pd.cpu_wait
    If ret = 0 Then
       pd.err_disp_message
    End If
```

com_receive

内容

エミュレータからバイト列を受信します。

パラメータ

ret = pd.com_receive (data, size)

属性	型	内容
[out]	BSTR *data	受信するバイト列
[in]	long size	受信バイト数

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'エミュレータから 3 バイト分を受信
Dim ret as Long
Dim data as String
Dim size as Long
    data = "(07)"
    size = 4
    ret = pd.com_send ( data, size )
    If ret = 0 Then
       pd.err_disp_message
    End If
    size = 3
    ret = pd.com_receive ( data, size )
    If ret = 0 Then
       pd.err_disp_message
    End If
```

com_send

内容

エミュレータへバイト列を送信します。

パラメータ

ret = pd.com_send (data, size)

属性	型	内容
[in]	BSTR data	送信するバイト列
[in]	long size	送信バイト数

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'エミュレータへ"(07)"のバイト列を送信
Dim ret as Long
Dim data as String
Dim size as Long
    data = "(07)"
    size = 4
    ret = pd.com_send ( data, size )
    If ret = 0 Then
        pd.err_disp_message
    End If
    size = 3
    ret = pd.com_receive ( data, size )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

cv_clear

内容

カバレッジ計測結果をクリアします。

パラメータ

ret = pd.cv_clear ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'カバレッジ計測結果をクリア
Dim ret as Long
   ret = pd.cv_clear
    If ret = 0 Then
      pd.err_disp_message
   End If
```

cv_clear_blk

内容

指定したブロックのカバレッジ計測結果をクリアします。
(このメソッドは、エミュレータ PC7501 を使用した場合のみ,有効です。)

パラメータ

ret = pd.cv_clear_blk (blkno)

属性	型	内容
[in]	long blkno	ブロック番号

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim blkno as Long

blkno = 0
ret = pd.cv_clear_blk ( blkno )
If ret = 0 Then
   pd.err_disp_message
End If
```

cv_get_base

内容

カバレッジ計測領域の先頭アドレスを取得します。

パラメータ

ret = pd.cv_get_base (base)

属性	型	内容
[out]	long *base	カバレッジ計測領域の先頭アドレス

base にカバレッジ計測領域の先頭アドレスが格納されます。 取得した先頭アドレスから 256K バイトがカバレッジ計測領域です。 先頭アドレスが 80000h の場合は、80000h～AFFFFh がカバレッジ計測領域となります。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'カバレッジ計測領域の先頭アドレスを取得
Dim ret as Long
Dim base as Long
   ret = pd.cv_get_base ( base )
   If ret = 0 Then
      pd.err_disp_message
   End If
```

cv_get_base_all

内容

すべてのブロックのカバレッジ計測領域の先頭アドレスを取得します。
(このメソッドは、エミュレータ PC7501 を使用した場合のみ有効です。)

パラメータ

ret = pd.cv_get_base_all (base)

属性	型	内容
[out]	VARIANT *base	開始アドレス

base にカバレッジ計測領域の先頭アドレスが格納されます。 取得した先頭アドレスから 256K バイトがカバレッジ計測領域です。 先頭アドレスが 80000h の場合は、80000h～AFFFFh がカバレッジ計測領域となります。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim base as Variant

ret = pd.cv_get_base ( base )
If ret = 0 Then
   pd.err_disp_message
End If
```

cv_get_base_blk

内容

指定したブロック番号のカバレッジ計測開始アドレスを取得します。
(このメソッドは、エミュレータ PC7501 を使用した場合のみ有効です。)

パラメータ

ret = pd.cv_get_base_blk (blkno, base)

属性	型	内容
[in]	long blkno	ブロック番号
[out]	long *base	開始アドレス

base にカバレッジ計測領域の先頭アドレスが格納されます。 取得した先頭アドレスから 256K バイトがカバレッジ計測領域です。 先頭アドレスが 80000h の場合は、80000h～AFFFFh がカバレッジ計測領域となります。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim base as Long
Dim blkno as Long

blkno = 0
ret = pd.cv_get_base_blk ( blkno, base )
If ret = 0 Then
   pd.err_disp_message
End If
```

cv_get_blkcnt

内容

有効なカバレッジブロック数を取得します。
(このメソッドは、エミュレータ PC7501 を使用した場合のみ有効です。)

パラメータ

ret = pd.cv_get_blkcnt (cnt)

属性	型	内容
[out]	long *cnt	ブロック数

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim cnt as Long

ret = pd.cv_get_blkcnt ( cnt )
If ret = 0 Then
   pd.err_disp_message
End If
```

cv_get_data

内容

指定アドレス範囲のカバレッジデータを取得します。

パラメータ

ret = pd.cv_get_data (st_addr, en_addr, rs_addr, re_addr, data)

st_addr に取得するカバレッジデータの先頭アドレス、en_addr に最終アドレスを指定します。

属性	型	内容
[in]	long st_addr	先頭アドレス
[in]	long en_addr	最終アドレス
[out]	long *rs_addr	取得したカバレッジデータの実先頭アドレス
[out]	long *re_addr	取得したカバレッジデータの実最終アドレス
[out]	VARIANT *data	カバレッジ計測結果

data にカバレッジデータが格納されます。 カバレッジデータは、8 バイトアライメントからの 8 バイト分が 1 つのデータに格納されます。 st_addr に C003h、en_addr に C00Eh を指定した場合、rs_addr に C000h、re_addr に C00Fh、data には C000h～C00Fh の カバレッジデータが格納されます(st_addr,en_addr で指定した値によっては、指定アドレス範囲以上のカバレッジデータが格納されます)。

rs_addr に C000h、data(0)のデータが A3h(2 進では 10100011b)であった場合、ターゲットプログラムが アクセスしたアドレスは、C000h,C001h,C005h,C007h です。C002h,C003h,C004h,C006h にはアクセス していません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス C000h～C007h のカバレッジ計測結果を取得
Dim ret as Long
Dim st_addr as Long
Dim en_addr as Long
Dim rs_addr as Long
Dim re_addr as Long
Dim data as Variant
Dim LoDataByte as Long
Dim HiDataByte as Long
   st_addr = &hC000
   en_addr = &hC007
   ret = pd.cv_get_data ( st_addr, en_addr, rs_addr, re_addr, data )
   If ret = 0 Then
      pd.err_disp_message
   End If
   LoDataByte = data(0) And &HF
   Select Case LoDataByte Mod 16
      Case 0
         buff1 = "    "
      Case 1
         buff1 = "*   "
      Case 2
         buff1 = " *  "
      Case 3
         buff1 = "**  "
      Case 4
         buff1 = "  * "
      Case 5
         buff1 = "* * "
      Case 6
         buff1 = " ** "
      Case 7
         buff1 = "*** "
      Case 8
         buff1 = "   *"
      Case 9
         buff1 = "*  *"
      Case 10
         buff1 = " * *"
      Case 11
         buff1 = "** *"
      Case 12
         buff1 = "  **"
      Case 13
         buff1 = "* **"
      Case 14
         buff1 = " ***"
      Case 15
         buff1 = "****"
      End Select

   HiDataByte = (data(0) And &HF0) ¥ &H10
    :
```

cv_set_base

内容

カバレッジ計測領域の先頭アドレスを設定します。

パラメータ

ret = pd.cv_set_base (base)

base にカバレッジ計測の先頭アドレスを指定します。 カバレッジ計測領域は、16 バイト境界から始まる連続 256K バイトです。 指定したアドレスの下位 16 ビットを 0 でマスクしたアドレスがカバレッジ計測の先頭アドレスになります。 指定したアドレスが C0000h の場合、C0000h～FFFFFh がカバレッジ計測領域です。

属性	型	内容
[in]	long base	カバレッジ計測領域の先頭アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'カバレッジ計測領域の先頭アドレスを C0000h 番地に設定
Dim ret as Long
Dim base as Long
    base = &hC0000
    ret = pd.cv_set_base ( base )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

down_load

内容

ターゲットプログラムをダウンロードします。

パラメータ

ret = pd.down_load (filename, mode)

属性	型	内容
[in]	BSTR filename	ダウンロードするファイル名
[in]	long mode	ダウンロードモード **4**：(LOAD_LOAD)　ロードモジュール **5**：(LOAD_SYM)　ロードモジュールのシンボル情報のみ **6**：(LOAD_ROM)　ロードモジュールの機械語情報のみ

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'ロードモジュールファイル"sample.x30"をダウンロード
Const LOAD_LOAD = 4
Dim ret as Long
Dim filename as String
    filename = "sample.x30"
    ret = pd.down_load ( filename, LOAD_LOAD )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

err_disp_message

内容

メソッド呼び出しで発生したエラーを表示します。

パラメータ

ret = pd.err_disp_message ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。

記述例

```
'エラーメッセージを表示
Dim ret as Long
    ret = pd.cpu_reset
    If ret = 0 Then
       pd.err_disp_message
 End If
```

err_get_message

内容

メソッド呼び出しで発生したエラー文字列を取得します。

パラメータ

ret = pd.err_get_message (msg)

属性	型	内容
[out]	BSTR *msg	エラーメッセージ

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。

記述例

```
'エラーメッセージを取得
Dim ret as Long
Dim msg as String

    ret = pd.cpu_reset
    If ret = 0 Then
        ret = pd.err_get_message (msg)
        MsgBox msg
    End If
```

event_set_request_mode

内容

PDxx 側で発生するイベントの COM クライアントへの出力有無を指定します。

パラメータ

ret = pd.event_set_request_mode (mode)

属性	型	内容
[in]	long mode	出力モード **0:** (EVT_DISABLE)　出力なし **1:** (EVT_ENABLE)　出力あり

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Const EVT_DISABLE = 0
Const EVT_ENABLE = 1
Dim ret as Long

ret = pd.event_request_mode( EVT_DISABLE)
If ret = 0 Then
   pd.err_disp_message
End If
```

exp_eval

内容

指定した式(ラベル/シンボル，メモリ変数，レジスタ変数 など)に該当する数値を返します。

パラメータ

ret = pd.exp_eval (str, radix, mode, data)

str に変換する式、radix に変換する際に有効な基数、mode に変換優先モードを指定します。式の仕様については、PDxx のオンラインヘルプを参照してください。

属性	型	内容
[in]	BSTR str	変換する文字列
[in]	long radix	基数 **16**：(EXP_HEX) 16 進数 **10**：(EXP_DEX) 10 進数 **0**：(EXP_DEFAULT) RADIX コマンドでの設定値
[in]	long mode	変換優先モード **0**: (EXP_LABEL) ラベル優先 **1**: (EXP_SYMBOL) シンボル優先
[out]	long *data	該当する数値

data に該当する値が格納されます。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'ラベル"start"に該当するアドレスを取得
Const EXP_LABEL = 0
Const EXP_HEX = 16
Dim ret as Long
Dim str as String
Dim data as Long
   ret = pd.exp_eval ( str, EXP_HEX, EXP_LABEL, data )
   If ret = 0 Then
      pd.err_disp_message
   End If
```

info_cpu

内容

MCU に依存する情報(アドレスの最大値など)を取得します。

パラメータ

pd.info_cpu (flag, status)

属性	型	内容
[in]	long flag	情報番号 **262** : (IN1_ADDRSIZE)　アドレス値の格納に必要なバイト数 **263** : (IN1_MAXADDR)　アドレスの最大値 **264** : (IN1_ADDRCOLM)　アドレス表示に必要な桁数 **265** : (IN1_ENDIAN)　MCU のエンディアン **267** : (IN1_WORD_SIZE)　ワードのバイト長 **270** : (IN1_MAX_OBJ)　1 命令の最大バイト数 **271** : (IN1_MAXDATA)　データの最大値 **272** : (IN1_MAXSTACK)　スタックの最大値 **298** : (IN1_BUS_SIZE)　データバス幅
[out]	long *status	CPU 情報（指定した flag に応じた値)

flag に 265(IN1_ENDIAN)を指定した場合、status にはビッグエンディアンであれば 273、スモールエンディアンであれば 274 が格納されます。

戻り値

戻り値はありません。

記述例

```
'アドレスの最大値を取得
Const IN1_MAXADDR = 263
Dim ret as Long
Dim maxAddr as Long
    flag = IN1_MAXADDR
    pd.info_cpu    flag, maxAddr
    lblMaxAddr = maxAddr
```

info_get_debugger

内容

製品名などのデバッガ情報を取得します。

パラメータ

ret = pd.info_get_debugger (name, dir, ini)

属性	型	内容
[out]	BSTR *name	デバッガ名+製品バージョン
[out]	BSTR *dir	EXE が格納されているディレクトリ
[out]	BSTR *ini	INI ファイル名

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
‘デバッガ情報の取得
Dim ret As Long
Dim name as String
Dim dir As String
Dim ini As String

    ret = pd.info_get_debugger(name, dir, ini)
    If ret = 0 Then
        pd.err_disp_message
    End If
```

info_get_mcufile_name

内容

INIT ダイアログで選択された MCU ファイル名を取得します。

パラメータ

ret = pd.info_get_mcufile_name (name)

属性	型	内容
[out]	BSTR *name	MCU ファイル名

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
‘デバッガ情報の取得
Dim ret As Long
Dim name as String

    ret = pd.info_get_mcufile_name(name)
    If ret = 0 Then
        pd.err_disp_message
    End If
```

info_service

内容

使用している PDxx(SIM)がサポートしているサービスの情報を取得します。

パラメータ

pd.info_service (flag, status)

属性	型	内容	
[in]	long flag	情報番号	
		275 : (IN1_SUPPORT_BITSYM)	ビットシンボルのサポート
		276 : (IN1_SUPPORT_C)	C 言語のサポート
		277 : (IN1_SUPPORT_RAMMONITOR)	RAM モニタのサポート
		278 : (IN1_SUPPORT_RTT)	リアルタイムトレースのサポート
		279 : (IN1_SUPPORT_CV)	カバレッジ計測のサポート
		280 : (IN1_SUPPORT_PROTCT)	プロテクトブレークのサポート
[out]	long *status	サポート情報（指定した flag に応じた値）	

戻り値

戻り値はありません。

記述例

```
'アドレスの最大値を取得
Const IN1_SUPPORT_RAMMONITOR = 277
Dim ret as Long
Dim status as Long
    flag = IN1_SUPPORT_RAMMONITOR
    pd.info_service    flag, status
   If status = 1 Then
       MsgBox("Support RAM Monitor.")
    End If
```

mem_clear_bit

内容

指定したビットをクリア(0)します。

パラメータ

ret = pd.mem_clear_bit (addr, bit)

属性	型	内容
[in]	long addr	アドレス
[in]	long bit	ビット位置

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地のビット 1 のクリア
Dim ret as Long
Dim addr as Long
Dim bit as Long
    addr = &h400
    length = 1
    ret = pd.mem_clear_bit ( addr, bit )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

mem_get

内容

指定したアドレスのデータを取得します。複数のアドレスのデータを取得する場合は、mem_get_multi を使用してください。

パラメータ

ret = pd.mem_get (addr, length, data)

属性	型	内容
[in]	long addr	アドレス
[in]	long length	データ長 1: 1 バイト 2: 2 バイト* 4: 4 バイト*
[out]	long data	データ

*製品バージョンによってはサポートされません(2,4 を指定しても 1 として扱われます)。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地のメモリ値を取得、データ長は 1 バイト
Dim ret as Long
Dim addr as Long
Dim length as Long
Dim data as Long
    addr = &h400
    length = 1
    ret = pd.mem_get ( addr, length, data )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

mem_get_bit

内容

指定したビットの値を取得します。

パラメータ

ret = pd.mem_get_bit (addr, bit, data)

属性	型	内容
[in]	long addr	アドレス
[in]	long bit	ビット位置
[out]	long *data	ビット値

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地のビット 1 の値を取得
Dim ret as Long
Dim addr as Long
Dim bit as Long
Dim data as Long
    addr = &h400
    bit = 1
    ret = pd.mem_get_bit ( addr, bit, data )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

mem_get_multi

内容

指定したアドレス範囲のメモリ値を取得します。

パラメータ

ret = pd.mem_get_multi (addr, num, length, data)

属性	型	内容
[in]	long addr	アドレス
[in]	long num	取得
[in]	long length	サイズデータ長 1: 1 バイト 2: 2 バイト* 4: 4 バイト*
[out]	VARIANT *data	データ

*製品バージョンによってはサポートされません(2,4 を指定しても 1 として扱われます)。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地～403h 番地のメモリ値を取得、データ長は 1 バイト
Dim ret as Long
Dim addr as Long
Dim length as Long
Dim size as Long
Dim data as Variant
Dim a as Long
    addr = &h400
    num = 4
    length = 1
    ret = pd.mem_get_multi ( addr, num, length, data )
    If ret = 0 Then
       pd.err_disp_message
    End If
    a = data(0)
```

mem_fill

内容

指定したアドレス範囲のメモリ値を任意の値で充填します。

パラメータ

ret = pd.mem_fill (addr, num, length, data)

属性	型	内容
[in]	long addr	アドレス
[in]	long num	アドレスサイズ
[in]	long length	データ長 1: 1 バイト 2: 2 バイト* 4: 4 バイト*
[in]	long data	充填する値

*製品バージョンによってはサポートされません(2,4 を指定しても 1 として扱われます)。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地～403h 番地のメモリ値をそれぞれデータ 10h で充填、データ長は 1 バイト
Dim ret as Long
Dim addr as Long
Dim length as Long
Dim size as Long
Dim data as Variant
    addr = &h400
    num = 4
    length = 1
    data = &h10
    ret = pd.mem_fill ( addr, num, length, data )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

mem_move

内容

指定したアドレス範囲のメモリ値を任意のアドレス以降にコピーします。

パラメータ

ret = pd.mem_move (addr, length, top, size)

属性	型	内容
[in]	long addr	コピー元アドレス
[in]	long num	コピー元アドレスのサイズ
[in]	long top	コピー先アドレス
[in]	long length	データ長 1: 1 バイト 2: 2 バイト* 4: 4 バイト*

*製品バージョンによってはサポートされません(2,4 を指定しても 1 として扱われます)。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地～403h 番地のメモリ値をそれぞれデータ 500h 以降にコピー、データ長は 1 バイト
Dim ret as Long
Dim addr as Long
Dim length as Long
Dim top as Long
Dim size as Long
    addr = &h400
    num = 4
    top = &h500
    size = 1
    ret = pd.mem_move ( addr, num, top, size )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

mem_set

内容

指定したアドレスのメモリ値を設定します(1 データ)。

パラメータ

ret = pd.mem_set (addr, length, data)

属性	型	内容
[in]	long addr	アドレス
[in]	long length	データ長 1: 1 バイト 2: 2 バイト* 4: 4 バイト*
[in]	long data	メモリ値

*製品バージョンによってはサポートされません(2,4 を指定しても 1 として扱われます)。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地のメモリ値を 0x12 に設定、データ長は 1 バイト
Dim ret as Long
Dim addr as Long
Dim length as Long
Dim data as Long
   addr = &h400
   length = 1
   data = &h12
   ret = pd.mem_set ( addr, length, data )    If ret = 0 Then
      pd.err_disp_message
   End If
```

mem_set_bit

内容

指定したビットをセット(1)します。

パラメータ

ret = pd.mem_set_bit (addr, bit)

属性	型	内容
[in]	long addr	アドレス
[in]	long bit	ビット位置

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地のビット 1 のセット
Dim ret as Long
Dim addr as Long
Dim bit as Long
    addr = &h400
    length = 1
    ret = pd.mem_set_bit ( addr, bit )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

mem_set_multi

内容

指定したアドレス範囲のメモリ値を設定します。

パラメータ

ret = pd.mem_set_multi (addr, num, length, data)

属性	型	内容
[in]	long addr	アドレス
[in]	long num	取得サイズ
[in]	long length	サイズデータ長 1: 1 バイト 2: 2 バイト* 4: 4 バイト*
[in]	VARIANT data	データ

*製品バージョンによってはサポートされません(2,4 を指定しても 1 として扱われます)。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 400h 番地～403h 番地のメモリ値をそれぞれ 1,2,3,4 に設定、データ長は 1 バイト
Dim ret as Long
Dim addr as Long
Dim num as Long
Dim length as Long
Dim data as Variant
    data(0) = 1
    data(1) = 2
    data(2) = 3
    data(3) = 4
    addr = &h400
    num = 4
    length = 1
    ret = pd.mem_set_multi ( addr, num, length, data )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

rram_clear

内容

RAM モニタのデータをクリアします。

パラメータ

ret = pd.rram_clear ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は 1、エラーが発生した場合は 0 を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'RAM モニタのデータをクリア
Dim ret as Long
    ret = pd.rram_clear
    If ret = 0 Then
        pd.err_disp_message
    End If
```

rram_clear_blk

内容

指定したブロックの RAM モニタのデータをクリアします。
(このメソッドは、エミュレータ PC7501 を使用した場合のみ、有効です。)

パラメータ

ret = pd.rram_clear_blk (blkno)

属性	型	内容
[in]	long blkno	ブロック番号

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim blkno as Long

blkno = 0
ret = pd.rram_clear_blk ( blkno )
If ret = 0 Then
   pd.err_disp_message
End If
```

rram_get_area

内容

RAMモニタ領域の先頭アドレスを取得します。取得した先頭アドレスから1K(1024)バイトがRAMモニタ領域です。先頭アドレスが 400h の場合は、400h～7FFh が RAM モニタ領域です。

パラメータ

ret = pd.rram_get_area (addr)

属性	型	内容
[out]	long *addr	RAM モニタ領域の先頭アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'RAM モニタ領域の先頭アドレスを取得
Dim ret as Long
Dim addr as Long
   ret = pd.rram_get_area ( addr )
   If ret = 0 Then
      pd.err_disp_message
   End If
```

rram_get_area_blk

内容

指定したブロックの RAM モニタ領域の先頭アドレスを取得します。
（このメソッドは、エミュレータ PC7501 を使用した場合のみ有効です。）

パラメータ

ret = pd.rram_get_area (blkno, addr)

属性	型	内容
[in]	long blkno	RAM monitor block number
[out]	long *addr	Top address of RAM monitor

取得した先頭アドレスから 1K(1024)バイトが RAM モニタ領域です。先頭アドレスが 400h の場合は、400h～7FFh が RAM モニタ領域です。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim addr as Long
Dim blkno as Long

blkno = 1
ret = pd.rram_get_area_blk ( blkno, addr )
If ret = 0 Then
   pd.err_disp_message
End If
```

rram_get_data

内容

指定したアドレス領域に該当する RAM モニタのデータとアクセス属性を取得します。 RAM モニタ領域外のアドレスを指定することはできません。

パラメータ

ret = pd.rram_get_data (addr, num, data, attr)

属性	型	内容
[in]	long addr	先頭アドレス
[in]	long num	取得バイト数
[out]	VARIANT *data	RAM モニタのデータ
[out]	VARIANT *attr	アクセス属性 **0**： 変化なし **1**： 書き込み **2**： 読み込み

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'RAM モニタに格納されたアドレス 400h 番地～403h 番地のデータとアクセス属性を取得
Dim ret as Long
Dim addr as Long
Dim num as Long
Dim data as Variant
Dim attr as Variant
dim aaa as Long
dim bbb as Long
    addr = &h400
    num = 4
    ret = pd.rram_get_data ( addr, num, data, attr )
    If ret = 0 Then
        pd.err_disp_message
    End If
    aaa = data(0)
    bbb = attr(0)
```

rram_set_area

内容

RAM モニタ領域の先頭アドレスを設定します。指定したアドレスの下位 4 ビットを 0 でマスクしたアドレスが RAM モニタの先頭アドレスになります。RAM モニタ領域は、16 バイト境界から始まる連続 1K(1024)バイトです。 指定したアドレスが 408h の場合、RAM モニタ領域は 400h から 7FFh となります。

パラメータ

ret = pd.rram_get_area (addr)

属性	型	内容
[in]	long addr	RAM モニタ領域の先頭アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'RAM モニタ領域の先頭アドレスを 400h 番地に設定
Dim ret as Long
Dim addr as Long
    addr = &h400
    ret = pd.rram_set_area ( addr )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

rram_set_area_blk

内容

指定したブロックの RAM モニタ領域の先頭アドレスを設定します。
(このメソッドは、エミュレータ PC7501 を使用した場合のみ有効です。)

パラメータ

ret = pd.rram_get_area_blk (blkno, addr)

属性	型	内容
[in]	long blkno	ブロック番号
[in]	long addr	先頭アドレス

指定したアドレスの下位 4 ビットを 0 でマスクしたアドレスが RAM モニタの先頭アドレスになります。RAM モニタ領域は、16 バイト境界から始まる連続 1K(1024)バイトです。 指定したアドレスが 408h の場合、RAM モニタ領域は 400h から 7FFh となります。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim addr as Long
Dim blkno as Long

blkno = 0
addr = &h400
ret = pd.rram_set_area_blk ( blkno, addr )
If ret = 0 Then
  pd.err_disp_message
End If
```

reg_get_pc

内容

現在のプログラムカウンタの値を返します。

パラメータ

ret = pd.reg_get_pc (pc)

属性	型	内容
[out]	long *pc	プログラムカウンタ値

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'プログラムカウンタ値を取得
Dim ret as Long
Dim pc as Long
   ret = pd.reg_get_pc ( pc )
   If ret = 0 Then
      pd.err_disp_message
   End If
```

reg_get_reg

内容

指定したレジスタ番号に該当するレジスタの値を返します。

パラメータ

ret = pd.reg_get_reg (regNo, regVal)

属性	型	内容
[in]	long regNo	レジスタ番号
[out]	long *regVal	レジスタ値

レジスタ番号は、製品によって異なります。

PD32R/ PD32RSIM	0:R0, 1:R1, 2:R2, 3:R3, 4:R4, 5:R5, 6:R6, 7:R7, 8:R8, 9:R9, 10:R10, 11:R11, 12:R12, 13:R13, 14:R14, 15:R15, 16:PSW, 17:SPI, 18:SPU, 19:BPC, 20:PC, 21:ACC0H, 22:ACC0L, 23:ACC1H, 24:ACC1L
PD308/ PD308SIM	0:0R0, 1:0R1, 2:0R2, 3:0R3, 4:0A0, 5:0A1, 6:0FB, 7:0SB, 8:1R0, 9:1R1, 10:1R2, 11:1R3, 12:1A0, 13:1A1, 14:1FB, 15:1SB, 16:USP, 17:ISP, 18:FLG, 19:PC, 20:INTB, 21:SVF, 22:SVP, 23:VCT, 24:DMD0, 25:DMD1, 26:DCT0, 27:DCT1, 28:DRC0, 29:DRC1, 30:DMA0, 31:DMA1, 32:DCA0, 33:DCA1, 34:DRA0, 35:DRA1
PD30/ PD30SIM	0:0R0, 1:0R1, 2:0R2, 3:0R3, 4:0A0, 5:0A1, 6:0FB, 7:1R0, 8:1R1, 9:1R2, 10:1R3, 11:1A0, 12:1A1, 13:1FB, 14:USP, 15:ISP, 16:FLG, 17:PC, 18:SB, 19:INTB
PD79/ PD79SIM	0:A, 1:B, 2:X, 3:Y, 4:S, 5:PC, 6:DT, 7:PS, 8:PG, 9:DP0, 10:DP1, 11:DP2, 12:DP3, 13:E, 14:PGPC
PD77/ PD77SIM	0:A, 1:B, 2:X, 3:Y, 4:S, 5:PC, 6:DT, 7:PS, 8:PG, 9:DPR, 10:PGPC
PD38/ PD38SIM	0:A, 1:X, 2:Y, 3:S, 4:PC, 5:F

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'レジスタ番号 0 のレジスタ値を取得
Dim ret as Long
Dim regNo as Long
Dim regVal as Long
    regNo = 0
    ret = pd.reg_get_reg ( regNo, regVal )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

reg_set_pc

内容

プログラムカウンタの値を設定します。

パラメータ

ret = pd.reg_set_pc (pc)

属性	型	内容
[in]	long pc	プログラムカウンタ値

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'プログラムカウンタ値を 0xF0000 に設定
Dim ret as Long
Dim pc as Long
    pc = &hF0000
    ret = pd.reg_set_pc ( pc )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

reg_set_reg

内容

指定したレジスタ番号に該当するレジスタの値を設定します。

パラメータ

ret = pd.reg_set_reg (regNo, regVal)

属性	型	内容
[in]	long regNo	レジスタ番号
[in]	long regVal	レジスタ値

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'レジスタ番号 0 のレジスタ値を 3 に設定
Dim ret as Long
Dim regNo as Long
Dim regVal as Long
    regNo = 0
    regVal = 3
    ret = pd.reg_set_reg ( regNo, regVal )
 If ret = 0 Then
        pd.err_disp_message
    End If
```

rtt_check_isfetch

内容

指定サイクルがフェッチサイクルか検査します。

パラメータ

ret = pd.rtt_check_isfetch (cycle, addr1, addr2, count)

cycle に検査するサイクルを指定します。

属性	型	内容
[in]	long cycle	サイクル
[out]	long *addr1	1 点目のフェッチアドレス
[out]	long *addr2	2 点めのフェッチアドレス
[out]	long *count	フェッチ命令数

指定したサイクルがフェッチサイクルの場合、count にはフェッチ命令数、 addr1 には 1 点めの命令のフェッチアドレス、addr2 には 2 点めの命令のフェッチアドレスが格納されます。 指定したサイクルがフェッチサイクルでない場合、count には 0 が格納されます。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'サイクル-10 がフェッチサイクルであるか検査
Dim ret as Long
Dim cycle as Long
Dim addr1 as Long
Dim addr2 as Long
Dim count as Long
    cycle = -10
    ret = pd.rtt_check_isfetch ( cycle, addr1, addr2, count )
    If ret = 0 Then
        pd.err_disp_message
    End If
    If count <> 0 Then
    label.Caption = "Fetch cycle."
    End If
```

rtt_clear

内容

リアルタイムトレース結果をクリアします。

パラメータ

ret = pd.rtt_clear ( )

パラメータはありません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'リアルタイムトレース結果をクリア
Dim ret as Long
    ret = pd.rtt_clear
    If ret = 0 Then
        pd.err_disp_message
    End If
```

rtt_get_bus

内容

指定サイクルのバス情報文字列を取得します。

パラメータ

ret = pd.rtt_get_bus (cycle, addr, buffer)

cycle に取得するサイクルを指定します。

属性	型	内容
[in]	long cycle	サイクル
[out]	long *addr	アドレス
[out]	BSTR *buffer	バス情報文字列

addr には指定サイクルのアドレスバスの内容、buffer にはバス情報文字列(トレースウィンドウのバスモードで表示される内容と同等)が格納されます。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'サイクル-10 のバス情報文字列を取得
Dim ret as Long
Dim cycle as Long
Dim addr as Long
Dim buffer as String
    cycle = -10
    ret = pd.rtt_get_bus ( cycle, addr, buffer )
    If ret = 0 Then
        pd.err_disp_message
    End If
    label.Caption = buffer
```

rtt_get_disasm

内容

指定サイクルからフェッチサイクルを検索し、その命令の逆アセンブル文字列を取得します。

パラメータ

ret = pd.rtt_get_disasm (cycle, next_cycle, buffer, count)

cycle に検索するサイクルを指定します。

属性	型	内容
[in,out]	long *cycle	サイクル
[out]	long *next_cycle	次のフェッチサイクル
[out]	BSTR *buffer	逆アセンブル文字列
[out]	long *count	逆アセンブル命令数

cycle には検索したフェッチサイクル、next_cycle にはその次のフェッチサイクル、buffer には逆アセンブル文字列、count には逆アセンブルした命令数が格納されます。 逆アセンブルした命令数が 2 以上の場合、その区切りには¥n が格納されます。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'サイクル-10 の逆アセンブル文字列を取得
Dim ret as Long
Dim cycle as Long
Dim next_cycle as Long
Dim buffer as String
Dim count as Long
    cycle = -10
    ret = pd.rtt_get_disasm ( cycle, next_cycle, buffer, count )
    If ret = 0 Then
        pd.err_disp_message
    End If
    label.Caption = buffer
```

rtt_get_range

内容

参照可能なトレースサイクルの範囲を取得します。

パラメータ

ret = pd.rtt_get_range (s_cycle, e_cycle)

属性	型	内容
[out]	long *s_cycle	先頭サイクル
[out]	long *e_cycle	最終サイクル

s_cycle に参照可能なトレースサイクルの先頭サイクル、e_cycle に最終サイクルが格納されます。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'トレース範囲のサイクルを取得
Dim ret as Long
Dim s_cycle as Long
Dim e_cycle as Long
   ret = pd.rtt_get_range ( s_cycle, e_cycle )
   If ret = 0 Then
      pd.err_disp_message
   End If
```

scri_command

内容

PDxx のスクリプトコマンドを実行します。

パラメータ

ret = pd.scri_command (string)

属性	型	内容
[in]	BSTR string	スクリプトコマンド文字列

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim buff as String

buff = “DumpByte 00000,00FFF”
ret = pd.scri_command( buff )
If ret = 0 Then
   pd.err_disp_message
End If
```

scri_print

内容

スクリプトウィンドウに任意の文字列を表示します。

パラメータ

ret = pd.scri_print (string)

属性	型	内容
[in]	BSTR string	表示文字列

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'
Dim ret as Long
Dim buff as String

buff = “2002/05/31”
ret = pd.scri_print( buff )
If ret = 0 Then
   pd.err_disp_message
End If
```

sym_add_bitsymbol

内容

ビットシンボルを新規に登録します。登録したビットシンボルは、グローバルビットシンボルとして扱われます。すでに同名のビットシンボルが存在する場合は、エラーとなります。

パラメータ

ret = pd.sym_add_bitsymbol (name, addr, bit)

属性	型	内容
[in]	BSTR name	登録ビットシンボル名
[in]	long addr	アドレス
[in]	long bit	ビット番号

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス"0x401"のビット番号 0 にビットシンボル"bitsym1"を割り付けます。
Dim ret as Long
Dim name as String
Dim addr as Long
Dim bit as Long

    name = "bitsym1"
    addr = &h401
    bit   = 0
    ret = pd.sym_add_bitsymbol ( name, addr, bit )
    If ret = 0 Then
       pd.err_disp_message
    End If
```

sym_add_label

内容

ラベルを新規に登録します。登録したラベルは、グローバルラベルとして扱われます。すでに同名のラベルが存在する場合は、エラーとなります。

パラメータ

ret = pd.sym_add_label (name, value)

属性	型	内容
[in]	BSTR name	登録ラベル名
[in]	long value	アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス"0xF0001"のアドレスにラベル"label1"を割り付けます。
Dim ret as Long
Dim name as String
Dim value as Long

    name = &hF0001
    value = "label1"
    ret = pd.sym_add_label ( name, value )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_add_symbol

内容

シンボルを新規に登録します。登録したシンボルは、グローバルシンボルとして扱われます。すでに同名のシンボルが存在する場合は、エラーとなります。

パラメータ

ret = pd.sym_add_symbol (name, value)

属性	型	内容
[in]	BSTR name	登録シンボル名
[in]	long value	アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス"0x401"にシンボル"sym1"を割り付けます。
Dim ret as Long
Dim name as String
Dim value as Long

    name = &h401
    value = "sym1"
    ret = pd.sym_add_symbol ( name, value )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_addr2line

内容

指定したアドレスに対応するファイル名、行番号を返します。

パラメータ

ret = pd.sym_addr2line (addr, line, filaname)

属性	型	内容
[in]	long addr	アドレス
[out]	long *line	行番号
[out]	BSTR *filename	ファイル名

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 0xF0000 に対応するファイルの行番号とファイル名を取得
Dim ret as Long
Dim addr as Long
Dim line as Long
Dim filename as String

    addr = &hF0000
    ret = pd.sym_addr2line ( addr, line, filename )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_bit2val

内容

指定した文字列のビットシンボルに該当するアドレス、ビット番号を返します。

パラメータ

ret = pd.sym_bit2val (symbol, addr, bit)

属性	型	内容
[in]	BSTR symbol	変換する文字列
[out]	long *addr	アドレス
[out]	long *bit	ビット番号

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'ビットシンボル"bitsym1"に該当する数値を取得
Dim ret as Long
Dim symbol as String
Dim addr as Long
Dim bit as Long

    symbol = "bitsym1"
    ret = pd.sym_bit2val ( symbol, addr, bit )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_get_disp_src

内容

プログラムウィンドウに表示しているソースファイル名を取得します。

パラメータ

ret = pd.sym_get_disp_src (filaname)

属性	型	内容
[out]	BSTR *filename	ソースファイル名

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'プログラムウィンドウに表示しているソースファイル名を取得
Dim ret as Long
Dim filename as String

    ret = pd.sym_get_disp_src ( filename )
    If ret = 0 Then
       pd.err_disp_message
    End If
```

sym_get_func_info

内容

指定した関数の情報を取得します。

パラメータ

ret = pd.sym_get_func_info (func, type, start, end, str)

属性	型	内容
[in]	BSTR func	関数名
[out]	BSTR *type	関数タイプ
[out]	long *start	開始アドレス
[out]	long *end	最終アドレス
[out]	BSTR *str	関数範囲文字列

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'関数 main の情報を取得
Dim ret As Long
Dim func as String
Dim data As Long
Dim type As String
Dim sAddr As Long
Dim eAddr As Long
Dim str As String

    func=”main”
    ret = pd.sym_get_func_info(func, type, sAddr, eAddr, str)
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_get_func_name

内容

指定したオブジェクトの中の関数の名前を取得します。

パラメータ

ret = pd.sym_get_func_name (obj, func, mode)

属性	型	内容
[in]	BSTR obj	オブジェクト名
[out]	BSTR *func	関数名
[in]	long mode	検索モード **2** : (LOAD_FIRST)　1 点目 **3** : (LOAD_NEXT)　2 点目以降

オブジェクト名を省略することはできません。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。すべての関数名を返した場合、戻り値は 0 となります。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'オブジェクト"main"の中の関数名を取得
Const LOAD_FIRST = 2
Const LOAD_NEXT = 3
Dim ret As Long
Dim obj As String
Dim func As String
Dim sw As Long

    sw = LOAD_FIRST
    obj = "main"

    Do
        ret = pd.sym_get_func_name(obj, func, sw)
        If ret = 0 Then
            Exit Do
        Else
            MsgBox func
            sw = LOAD_NEXT
        End If
    Loop
```

sym_get_obj_name

内容

ロードモジュールの中のオブジェクト名を取得します。

パラメータ

ret = pd.sym_get_obj_name (obj, mode)

属性	型	内容
[out]	BSTR *obj	オブジェクト名
[in]	long mode	検索モード **2** : (LOAD_FIRST)　1 点目 **3** : (LOAD_NEXT)　2 点目以降

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。すべてのオブジェクト名を返した場合、戻り値は 0 となります。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'すべてのオブジェクト名を取得
Const LOAD_FIRST = 2
Const LOAD_NEXT = 3
Dim ret As Long
Dim obj As String
Dim sw As Long

    sw = LOAD_FIRST
    Do
        ret = pd.sym_get_obj_name(obj, sw)
        If ret = 0 Then
            Exit Do
        Else
            MsgBox obj
            sw = LOAD_NEXT
        End If
    Loop
```

sym_get_scope

内容

指定アドレスを含むスコープ(オブジェクトファイル名)を取得します。

パラメータ

ret = pd.sym_get_scope (addr, filaname)

属性	型	内容
[in]	long addr	アドレス
[out]	BSTR *filename	オブジェクトファイル名

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 0xF0000 のスコープ(オブジェクトファイル名)を取得
Dim ret as Long
Dim addr as Long
Dim filename as String
    addr = &hF0000
    ret = pd.sym_get_scope ( addr, filename )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_get_src_name

内容

指定したオブジェクトの中のソースファイルの名前を取得します。

パラメータ

ret = pd.sym_get_src_name (obj, src, mode)

属性	型	内容
[in]	BSTR obj	オブジェクト名
[out]	BSTR *src	ソースファイル名
[in]	long mode	検索モード **2** : (LOAD_FIRST)　1 点目 **3** : (LOAD_NEXT)　2 点目以降

オブジェクト名に NULL を与えた場合、すべてのオブジェクトが検索対象となります。

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。すべてのソースファイル名を返した場合、戻り値は 0 となります。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'オブジェクト”main”の中のソースファイル名を取得
Const LOAD_FIRST = 2
Const LOAD_NEXT = 3
Dim ret As Long
Dim obj As String
Dim src As String
Dim sw As Long

    sw = LOAD_FIRST
    obj = “main”

    Do
        ret = pd.sym_get_src_name(obj, src, sw)
        If ret = 0 Then
            Exit Do
        Else
            MsgBox src
            sw = LOAD_NEXT
        End If
    Loop
```

sym_get_variable_info

内容

指定した C 変数の情報を取得します。

パラメータ

ret = pd.sym_get_variable_info (var, type, l_data, h_data, str1, str2)

属性	型	内容
[in]	BSTR var	C 変数名
[out]	BSTR *type	変数タイプ
[out]	long *l_data	変数値(下位 32 ビット)
[out]	long *h_data	変数値(上位 32 ビット)
[out]	BSTR *str1	変数情報 1
[out]	BSTR *str2	変数情報 2

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーエラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'C 変数 aaa の情報を取得
Dim ret As Long
Dim var as String
Dim type As String
Dim lData As Long
Dim hData As Long
Dim str1 As String
Dim str2 As String

    var=”aaa”
    ret = pd.sym_get_func_info(var, type, l_data, h_data, str1, str2)
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_line2addr

内容

指定した行番号,ファイル名に対応するアドレスを返します。

パラメータ

ret = pd.sym_line2addr (line, filaname, addr)

属性	型	内容
[in]	BSTR filename	ファイル名
[in]	long line	行番号
[out]	long *addr	アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'ファイル名"samp.c"、行番号"10"に対応するアドレスを取得
Dim ret as Long
Dim addr as Long
Dim line as Long
Dim filename as String

    filename = "samp.c"
    line = 10
    ret = pd.sym_line2addr ( filename, line, addr )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_set_disp_src

内容

プログラムウィンドウに表示するソースファイルを指定します。 該当するソースファイルが存在しない場合は、指定アドレス位置をプログラムウィンドウに表示します。

パラメータ

ret = pd.sym_set_disp_src (filaname, line, addr)

属性	型	内容
[in]	BSTR filename	ソースファイル名
[in]	long line	行番号
[in]	long addr	アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'プログラムウィンドウにソースファイル"main.c"の 5 行目を表示
Dim ret as Long
Dim filename as String
Dim line as Long
dim addr as Long

    filename = "main.c"
    line = 5
    addr = &hF0000
    ret = pd.sym_set_disp_src ( filename, line, addr )
    If ret = 0 Then
       pd.err_disp_message
    End If
```

sym_set_scope_addr

内容

指定アドレスを含むスコープに有効なスコープを切り替えます。

パラメータ

ret = pd.sym_set_scope_addr (addr)

属性	型	内容
[in]	long addr	アドレス

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス 0xF0000 を含むスコープに切り替え
Dim ret as Long
Dim addr as Long
Dim filename as String
    addr = &hF0000
    ret = pd.sym_set_scope_addr ( addr )
  If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_set_scope_obj

内容

指定したオブジェクトファイルにスコープを切り替えます。

パラメータ

ret = pd.sym_set_scope_obj (obj)

属性	型	内容
[in]	BSTR obj	オブジェクトファイル名

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'オブジェクト"main.r30"にスコープを切り替え
Dim ret as Long
Dim obj as String
    obj = "main.r30"
    ret = pd.sym_set_scope_obj ( obj )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_sym2val

内容

指定したラベル/シンボルに対応する数値を返します。

パラメータ

ret = pd.sym_sym2val (mode, symbol, value)

属性	型	内容
[in]	long mode	検索モード **0**：(LOAD_LABEL)　ラベル優先 **1**：(LOAD_SYMBOL)　シンボル優先
[in]	BSTR symbol	変換する文字列
[out]	long *value	該当する数値

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'シンボル"data1"に該当する数値を取得
Const LOAD_SYMBOL = 1
Dim ret as Long
Dim symbol as String
Dim value as Long

    symbol = "data1"
    ret = pd.sym_sym2val ( LOAD_SYMBOL, str, value )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

sym_val2bit

内容

指定したアドレス,ビット番号に対応するビットシンボルを返します。

パラメータ

ret = pd.sym_val2bit (addr, bit, symbol)

属性	型	内容
[in]	long addr	アドレス
[in]	long bit	ビット番号
[out]	BSTR *symbol	ビットシンボル

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'アドレス"0x400"、ビット番号"0"に該当するビットシンボルを取得
Const LOAD_LABEL = 0
Dim ret as Long
Dim addr as Long
Dim bit as Long
Dim symbol as String

    addr = &h400
    bit = 0
    ret = pd.sym_val2bit ( addr, bit, symbol )
    If ret = 0 Then
       pd.err_disp_message
    End If
```

sym_val2sym

内容

指定した数値に対応するラベル/シンボルを返します。

パラメータ

ret = pd.sym_val2sym (mode, value, symbol)

属性	型	内容
[in]	long mode	検索モード **0**：(LOAD_LABEL)　ラベル優先 **1**：(LOAD_SYMBOL)　シンボル優先
[in]	long value	変換する数値
[out]	BSTR *symbol	文字列

戻り値

メソッドが正常に終了した場合は **1**、エラーが発生した場合は **0** を返します。エラーメッセージは、メソッド err_disp_message を呼び出すことによって表示できます。

記述例

```
'数値"0xF0000"に該当するラベル/シンボルを取得
Const LOAD_LABEL = 0
Dim ret as Long
Dim symbol as String
Dim value as Long

    value = &hF0000
    ret = pd.sym_val2sym ( LOAD_SYMBOL, value, symbol )
    If ret = 0 Then
        pd.err_disp_message
    End If
```

5.3 イベント番号一覧

PDxx 側で発生するイベントは、以下の通りです。

イベント番号		発生イベント内容
1001	(EVENT_GO)	プログラム実行開始
1002	(EVENT_STOP)	プログラム停止
1003	(EVENT_RESET)	プログラムリセット
1004	(EVENT_STEP)	ステップ実行
1005	(EVENT_OVER)	オーバーステップ実行
1006	(EVENT_RETURN)	リターン実行
1007	(EVENT_PUT_REG)	レジスタ値の変更
1008	(EVENT_REG_PC)	PC 値の変更
1009	(EVENT_PUT_MEM)	データの変更
1010	(EVENT_LOAD)	プログラムのダウンロード
1013	(EVENT_SBRK)	ソフトウェアブレークポイント設定の変更
1014	(EVENT_TRACE_START)	リアルタイムトレースの開始
1015	(EVENT_TRACE_END)	リアルタイムトレースの終了
1016	(EVENT_TRACE_PASS)	リアルタイムトレースポイントの通過
1021	(EVENT_MAP)	メモリマッピングの変更

[Memo]

PDSDK COMキット
リファレンスマニュアル

発行年月日　2006年09月16日　　Rev.2.00

発行　　株式会社 ルネサス テクノロジ　営業企画統括部
〒100-0004　東京都千代田区大手町2-6-2

編集　　株式会社 ルネサス ソリューションズ　ツール開発部

PDSDK COM キット
リファレンスマニュアル

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

RJJ10J0111-0200