

お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

PDxxSIM I/O DLLキット

ユーザーズマニュアル

Active X、Microsoft、MS-DOS、Visual Basic、Visual C++、Windows および Windows NT は、米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。

IBM および AT は、米国 International Business Machines Corporation の登録商標です。

Intel、Pentium は、米国 Intel Corporation の登録商標です。

Adobe および Acrobat は、Adobe Systems Incorporated（アドビシステムズ社）の登録商標です。

その他すべてのブランド名および製品名は個々の所有者の登録商標もしくは商標です。

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

- 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは責任を負いません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、予告なしに、本資料に記載した製品又は仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前に株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (<http://www.renesas.com>) などを通じて公開される情報に常にご注意ください。
- 本資料に記載した情報は、正確を期すため、慎重に制作したものです。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を負いません。
- 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な事項を記入の上、ツール技術サポート窓口 support_tool@renesas.com まで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

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

はじめに

PDxxSIM I/O DLLキットは、シミュレータデバッガM3T-PDxxSIMの機能を拡張するためのキットです。I/O DLLを作成するには、市販のWindowsアプリケーション開発環境であるMicrosoft Visual C++が必要です。

本ユーザーズマニュアルには、PDxxSIM I/O DLL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Visual C++の言語仕様、操作方法などについては、Visual C++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。

対応シミュレータデバッガ

PDxxSIM I/O DLLキットは、全てのシミュレータデバッガM3T-PDxxSIMで利用できるものではありません。PDxxSIM I/O DLLキットと連携できるシミュレータデバッガ、およびそのバージョンについては、PDxxSIM I/O DLL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

PDxxSIM I/O DLLキットの使用権は、使用するシミュレータデバッガ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、PDxxSIM I/O DLL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

PDxxSIM I/O DLL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のみ対応させていただきますので、あらかじめご了承ください。

[MEMO]

目次

1. 概要.....	1
2. 構成.....	1
3. I/O DLLの作成方法.....	2
3.1. IODLLTEMPLATEプロジェクトを使用する方法.....	2
3.2. 新規にI/O DLLを作成する方法.....	3
4. I/O DLLの使用方法.....	5
5. I/O DLLのデバッグ方法.....	6
6. I/O DLL側に情報を通知する関数の仕様.....	7
7. SIMXX.EXE側の情報を取得する関数の仕様.....	11
8. I/O DLLでターゲットプログラム内のシンボルを指定するには.....	18
9. 制限事項.....	24

[MEMO]

1. 概要

I/O DLL とは、シミュレータデバッガ PDxxSIM (xx には、機種名 308、30 等が入ります) のシミュレータエンジンと連携して動作する DLL (ダイナミックリンクライブラリ) のことです。

シミュレータデバッガは、所定の設定を行うことで I/O DLL をロードし、1 命令実行やメモリの Read/Write、割り込み発生等のタイミングに同期させて I/O DLL を動作させることができます。

これにより、マイコンの入出力ポートや、内蔵周辺機能の動作をシミュレートしてターゲットプログラムのデバッグができます。また、外部ツールとの間でデータを入出力するような連携が可能になります。

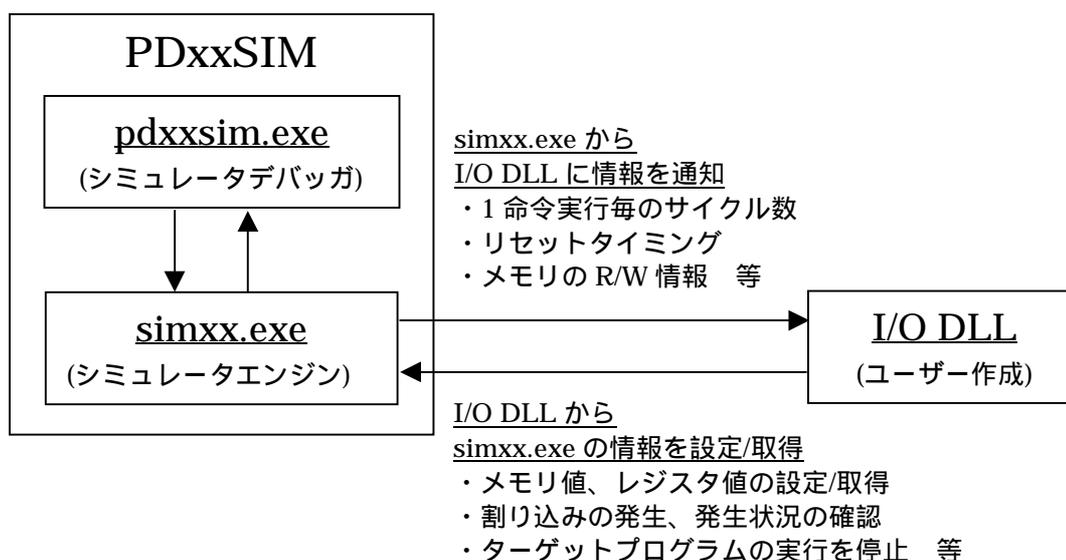
I/O DLL は、C/C++言語を使用して任意に作成することができます。I/O DLL を作成するには、市販の Windows アプリケーション開発環境である Microsoft Visual C++が必要です。

2. 構成

I/O DLL は、シミュレータエンジン simxx.exe (xx には、機種名 308、30 等が入ります) にアクセスしてメモリ値やレジスタ値等の情報を取得します。

逆に simxx.exe は、メモリを読み書きした情報や 1 命令実行時のサイクル数等の情報を I/O DLL へ通知します。

構成は、以下のようになります。



3. I/O DLL の作成方法

本章では、Microsoft Visual C++ 6.0 (以下 VC++とする) を用いた I/O DLL の作成方法を説明します。VC++の使用方法については、VC++ のマニュアル、ヘルプ等を参照ください。

I/O DLL を作成するには、I/O DLL キットに添付されている I/O DLL のサンプルプロジェクト IodllTemplate プロジェクトを使用する方法と、新規に作成する方法があります。IodllTemplate プロジェクトは、I/O DLL を作成するための VC++の雛型のプロジェクトです。

3.1. IodllTemplate プロジェクトを使用する方法

以下に、IodllTemplate プロジェクトを用いて I/O DLL を作成する方法を PD308SIM 用の I/O DLL を作成する場合を例にとって説明します。他の機種の場合は、“308”の記述を対応する機種名 (“30”、“32R”) に読み替えてください。

1. VC++ のメニュー「ファイル」 「ワークスペースを開く」を選択します。I/O DLL をインストールしたディレクトリ(以下、C:\MTOOL\Iodll とする)の以下のディレクトリにある IodllTemplate プロジェクトのプロジェクトファイル IodllTemplate.dsp をオープンしてください。

“Samples\Pdxxsim\IodllTemplate”

2. VC++ のメニュー「プロジェクト」 「設定」を選択します。リンクタブのカテゴリ「一般」のオブジェクト/ライブラリモジュール欄に指定されている I/O DLL 用のライブラリファイル sim308.lib と IodllExpLib(d).lib のパスの設定を確認してください(絶対パス、または相対パス付きで指定してください)。ライブラリファイルは、C:\MTOOL\Iodll の“Library”ディレクトリに格納されています。

- 設定の対象欄が Win32 Debug の場合は、IodllExpLibd.lib を指定してください。

オブジェクト/ライブラリ モジュール(L):

- 設定の対象欄が Win32 Release の場合は、IodllExpLib.lib を指定してください。

オブジェクト/ライブラリ モジュール(L):

3. VC++ のメニュー「プロジェクト」 「設定」を選択します。リンクタブのカテゴリ「一般」の出力ファイル名欄に指定されている I/O DLL ファイル名“IodllTemplate.dll”を任意のファイル名(拡張子“.dll”)に変更して、I/O DLL の出力先が PD308SIM をインストールしたディレクトリ(以下、C:\MTOOL\PD308SIM とする)になるように設定してください。

出力ファイル名(N):

4. iofunc\iofunc.cpp をオープンして、シミュレーション用のコードを記述してください。iofunc.cpp には、sim308.exe から呼び出される関数が実装されています。目的の関数内で

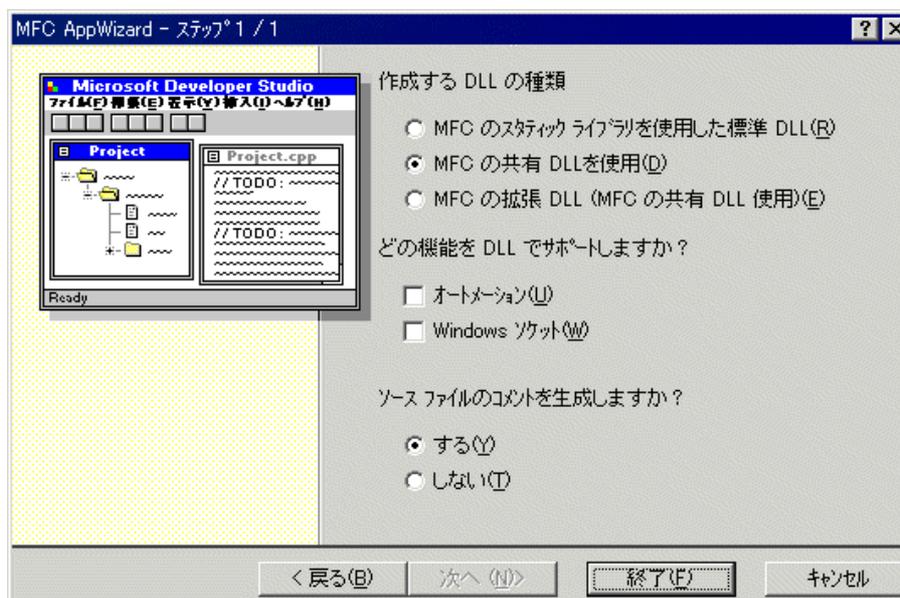
sim308.exe へアクセスする関数を使用して、シミュレーション用のコードを記述してください。

- ビルドすると、C:\¥MTOOL¥PD308SIM の下に I/O DLL (sample.dll) が作成されます。

3.2. 新規に I/O DLL を作成する方法

IodllTemplate プロジェクトを使用せず、新規に I/O DLL を作成するには、以下の方法で作成してください。

- VC++のメニュー「ファイル」 「新規作成」を選択して、以下の設定でプロジェクトを新規に作成してください。
 - プロジェクトタブで MFC AppWizard(dll)を選択してください。
 - 任意のプロジェクト名と位置を設定してください (例：sample プロジェクト)。
 - 「新規にワークスペースを作成」を選択してください。
 - プラットフォームで Win32 をチェックしてください。
 - [OK] [終了]をクリックします



2. VC++ のメニュー「プロジェクト」 「設定」を選択して、以下の設定をしてください。
 - C/C++タブを選択してカテゴリ「一般」のプリプロセッサの定義欄に "IODLL_EXPORTS" 定義を追加してください。

プリプロセッサの定義(O):
_WINDOWS_WINDLL_AFXDLL_MBCS_USRDLL_IODLL_EXPORTS

- リンクタブを選択してカテゴリ「一般」のオブジェクト/ライブラリモジュール欄に I/O DLL 用のライブラリファイル sim308.lib と IodllExpLib(d).lib を絶対パス、または相対パス付きで指定してください。ライブラリファイルは、C:\MTOOL\Iodll の"Library"ディレクトリに格納されています。

- 設定の対象欄が Win32 Debug の場合は、IodllExpLibd.lib を指定してください。

オブジェクト/ライブラリ モジュール(L):
..\..\Library\Sim308.lib\Library\IodllExpLibd.lib

- 設定の対象欄が Win32 Release の場合は、IodllExpLib.lib を指定してください。

オブジェクト/ライブラリ モジュール(L):
..\..\Library\Sim308.lib\Library\IodllExpLib.lib

3. VC++ のメニュー「プロジェクト」 「設定」を選択します。リンクタブのカテゴリ「一般」の出力ファイル名欄に I/O DLL の出力先が C:\MTOOL\PD308SIM になるように設定します。

出力ファイル名(N):
C:\MTOOL\PD308SIM\sample.dll

4. IodllTemplate プロジェクトの iofunc ディレクトリにある iofunc.cpp と iofunc.h を、新規に作成したプロジェクトの任意のディレクトリにコピーして、以下の方法でプロジェクトに追加してください。
 - VC++ のメニュー「プロジェクト」 「プロジェクトへ追加」 「ファイル」で iofunc.cpp を選択して[OK]を押してください。
5. iofunc.h をオープンして、iofunc.h にインクルードしているアプリケーションのメインヘッダファイルの指定を新規に作成したメインヘッダファイル名に変更してください。
(例) #include "..\IodllTemplate.h" "#include "..\sample.h"
6. iofunc.cpp をオープンして、シミュレーション用のコードを記述してください。
7. ビルドすると、C:\MTOOL\PD308SIM の下に I/O DLL (sample.dll) が作成されます。

4. I/O DLL の使用方法

本章では、PDxxSIM で I/O DLL をロードして使用方法を説明します。

I/O DLL を使用するには、I/O DLL を simxx.exe へ登録する必要があります。

simxx.exe は、起動時に I/O DLL が登録されていた場合、登録された I/O DLL をロードしてシミュレーションを開始します。

以下に、PDxxSIM に I/O DLL を登録して使用方法を PD308SIM の場合を例にとって説明します。他の機種の場合は、“308”の記述を対応する機種名（“30”、“32R”）に読み替えてください。

1. 使用する I/O DLL (.dll ファイル) を C:\¥MTOOL¥PD308SIM へコピーしてください。
2. I/O DLL を sim308.exe へ登録します。登録するには、sim308.exe の環境設定ファイル sim308.ini ファイルに I/O DLL ファイル名を記述します。
sim308.ini ファイルは、C:\¥MTOOL¥PD308SIM に存在します。ただし、PD308SIM をインストールして一度も起動していない場合は作成されていませんので、別途エディタ等で作成してください。
3. sim308.ini ファイルでは、以下のように[DLLNAME] セクションを作成して、I/O DLL ファイル名を“IODLL=” の後に拡張子“.dll”を取って記述してください。

例) I/O DLL ファイル名が “Sample.dll” のとき

```
[DLLNAME]
IODLL=Sample
```

4. PD308SIM を起動すると I/O DLL がロードされます。

なお、I/O DLL を使用しない場合は、sim308.ini ファイルに作成した[DLLNAME] セクションの記述を削除して PD308SIM を起動してください。

```
[DLLNAME]           削除
IODLL=Sample        削除
```

5. I/O DLL のデバッグ方法

本章では、作成した I/O DLL をデバッグする方法を説明します。

I/O DLL を VC++ でデバッグするには、I/O DLL のプロジェクトの設定を変更する必要があります。以下に、変更内容を PD308SIM 用の I/O DLL をデバッグする場合を例にとって説明します。他の機種の場合は、「308」の記述を対応する機種名（「30」、「32R」）に読み替えてください。

1. 前章の I/O DLL の使用方法にしたがい、I/O DLL を使用するように登録します。
2. VC++ のメニュー「プロジェクト」「設定」を選択します。リンクタブのカテゴリ「一般」の出力ファイル名欄に I/O DLL の出力先が C:\MTOOL\PD308SIM になるように設定されているか確認します。



3. VC++ のメニュー「プロジェクト」「設定」を選択します。デバッグタブのデバッグセッションの実行可能ファイル欄に、デバッグ時の実行ファイルが C:\MTOOL\PD308SIM にあるシミュレータエンジンの実行ファイル sim308.exe となるように設定します。



4. 設定後、VC++ のメニュー「ビルド」「デバッグの開始」「実行」を選択して、デバッグを開始すると以下のダイアログが表示されます。

I/O DLL のデバッグは可能ですので OK ボタンを押して続行してください。



5. シミュレータエンジン sim308.exe の起動後、PD308SIM を起動すると I/O DLL をデバッグできます (sim308.exe は、起動後 Windows のシステムトレイに登録 (表示) されますので、それを確認後に PD308SIM を起動してください)。

なお、VC++ のデバッグ機能については、VC++ のマニュアル、ヘルプを参照ください。

6. I/O DLL 側に情報を通知する関数の仕様

本章では、I/O DLL 側にシミュレータエンジン側の情報を通知する関数の仕様を説明します。

本関数は、I/O DLL プロジェクトのソースファイル `iofunc\iofunc.cpp` に実装している関数です。`simxx.exe` 側では、本関数を呼び出して I/O DLL 側に情報を通知します。シミュレーション用のコードは、本関数内に記述してください。

関数名	概要
NotifyStepCycle	1 命令実行毎に実行したサイクル数を通知します。
NotifyPreExecutionPC	実行直前に現在の PC 値を通知します。
NotifyReset	ターゲットプログラムがリセットされたことを通知します。
NotifyStart	シミュレータエンジンが起動されたことを通知します。
NotifyEnd	シミュレータエンジンが終了されたことを通知します。
NotifyInterrupt	割り込みが発生したときに発生した割り込みのベクタ番号(ベクタアドレス)を通知します。
NotifyPreReadMemory	ターゲットプログラムからメモリの読み込みが発生したとき、メモリの値を読み込む直前にアドレスとデータ長を通知します。
NotifyPostWriteMemory	ターゲットプログラムからメモリにデータが書き込まれた後に書き込みがあったアドレス、データ長、およびデータ値を通知します。

以下に関数の仕様を示します。

● 実行した 1 命令のサイクル数を通知

関数名： `void NotifyStepCycle(int cycle)`

引数： `int cycle` 実行したサイクル数

戻り値： なし

機能： 1 命令実行毎に実行したサイクル数を通知します。

● 実行直前のプログラムカウンタ(PC)値を通知

関数名： `void NotifyPreExecutionPC(unsigned long address)`

引数： `unsigned long address` 実行直前の PC 値

戻り値： なし

機能： 実行直前に現在の PC 値を通知します。

● リセットを通知

関数名： `void NotifyReset(void)`

引数： なし

戻り値： なし

機能： ターゲットプログラムがリセットされたことを通知します。

- シミュレータエンジンの起動を通知

関数名： void NotifyStart(void)

引数： なし

戻り値： なし

機能： シミュレータエンジンが起動されたことを通知します。

- シミュレータエンジンの終了を通知

関数名： void NotifyEnd(void)

引数： なし

戻り値： なし

機能： シミュレータエンジンが終了されたことを通知します。

- 割り込みの発生を通知

関数名： void NotifyInterrupt(unsigned long vec)

引数： unsigned long vec 発生した割り込みのベクタ番号 (ベクタアドレス)

PD308SIM, PD30SIM はベクタ番号。

PD32RSIM は EIT ベクタエントリ (0x00000080 のみ有効)。

戻り値： なし

機能： 割り込みが発生したときに発生した割り込みのベクタ番号 (ベクタアドレス) を通知します。

- メモリをリード(データリード)する直前に通知

関数名： void NotifyPreReadMemory(unsigned long address, int length)

引数： unsigned long address リードするメモリアドレス

int length リードするメモリデータのデータ長

1 1 バイト

2 2 バイト

3 3 バイト

4 4 バイト

戻り値： なし

機能： ターゲットプログラムからメモリの読み込みが発生したとき、メモリの値を読み込む直前にアドレスとデータ長を通知します。

- **メモリをライトした直後に通知**

関数名： void NotifyPostWriteMemory(unsigned long address, int length, unsigned long data)

引数： unsigned long address ライトされたメモリアドレス
 int length ライトされたメモリデータのデータ長

 1 1 バイト

 2 2 バイト

 3 3 バイト

 4 4 バイト

 unsigned long data ライトされたメモリデータ値

戻り値： なし

機能： ターゲットプログラムからメモリにデータが書き込まれた後に、書き込みがあったアドレス、データ長、およびデータ値を通知します。

以下に上記関数を使用した記述例を示します。この例の網掛けの部分は、iofunc.cpp ファイルにテンプレートとして実装されている部分です。

```
void NotifyStepCycle(int cycle)
{
    unsigned long tabsrData, ta0Data, ta0icData;

    if (sCountFlag == FALSE) { // カウント開始フラグのチェック
        return;
    }
    RequestGetMemory(TABSR, 1, &tabsrData);
    if ((tabsrData & 0x01) == 0x01) { // カウント開始フラグのチェック
        sCountCycle += cycle;
        RequestGetMemory(TA0, 2, &ta0Data);
        if (sCountCycle >= ta0Data + 1) { // 分周比分カウントダウン
            RequestGetMemory(TA0IC, 1, &ta0icData);
            RequestInterrupt(TA0INT, ta0icData & 0x7);
            // タイマ A0 割り込みの発生

            sCountCycle = 0;
        }
    }
}
return;

void NotifyPreExecutionPC(unsigned long address)
{
    return;
}
:
:
:
void NotifyPostWriteMemory(unsigned long address, int length)
{
    if (address == TABSR) {
        if ((data & 0x01) == 0x01) { // カウント開始フラグのチェック
            sCountFlag = TRUE;
        }
    }
}
return;
}
```

7. simxx.exe 側の情報を取得する関数の仕様

本章では、I/O DLL 側からシミュレータエンジン側の情報を取得する関数の仕様を説明します。

本関数は、simxx.exe 側に実装されている関数です。I/O DLL 側では、本関数を呼び出して simxx.exe 側から情報を取得できます。本関数は、I/O DLL プロジェクトのソースファイル iofunc\iofunc.cpp に実装している関数内で、シミュレーション用のコードを記述するときに使用します。

関数名	概要
RequestGetMemory	指定したアドレスのメモリデータを取得します。
RequestPutMemory	指定したアドレスにメモリデータを設定します。
RequestGetRegister	指定したレジスタの値を取得します。
RequestPutRegister	指定したレジスタの値を設定します。
RequestInterrupt	指定した割り込みを発生します。
RequestInterruptStatus	割り込みの発生状況を取得します。
RequestTotalCycle	現在の総実行サイクル数を参照します。
RequestInstructionNum	現在の総実行命令数を参照します。
RequestStop	ターゲットプログラムの実行を停止します。
RequestErrorNum	直前に実行した関数でエラーが発生した場合、そのエラー番号を取得します。

以下に関数の仕様を示します。

● メモリ値の取得

関数名： int RequestGetMemory(unsigned long address, int length, unsigned long * data)

引数： unsigned long address メモリデータを取得するアドレス
int length 取得するメモリデータのデータ長

1 1 バイト

2 2 バイト

3 3 バイト

4 4 バイト

unsigned long * data 取得するメモリデータの格納先

戻り値： int status

TRUE 正常終了

FALSE エラー発生

機能： 指定したアドレスのメモリデータを取得します。

本関数でのリードアクセス情報は、仮想ポート入力機能、I/O スクリプト機能には反映されません。

● メモリ値の設定

関数名： int RequestPutMemory(unsigned long address, int length, unsigned long data)

引数： unsigned long address メモリデータを設定するアドレス

int length 設定するメモリデータのデータ長

1 1 バイト

2 2 バイト

3 3 バイト

4 4 バイト

unsigned long data 設定するメモリデータ

戻り値： int status

TRUE 正常終了

FALSE エラー発生

機能： 指定したアドレスにメモリデータを設定します。

本関数でのライトアクセス情報は、GUI 出力機能、仮想ポート出力機能、および I/O スクリプト機能には反映されません。

- レジスタ値の取得

関数名： int RequestGetRegister(int regNo, unsigned long * regValue)

引数： int regNo 値を取得するレジスタの番号

レジスタの番号は、I/O DLL のサンプルプログラムに含まれているヘッダファイル `iofunc\iofunc.h` 内で定義していますのでご参照ください。

(例) PD308SIM での定義

regNo	レジスタ
REG_Rx_F	バンク 0Rx レジスタ(x は 0 ~ 3)
REG_RxH_F	バンク 0Rx レジスタの上位 8 ビット(x は 0 ~ 1)
REG_RxL_F	バンク 0Rx レジスタの下位 8 ビット(x は 0 ~ 1)
REG_Ax_F	バンク 0Ax レジスタ(x は 0 ~ 1)
REG_FB_F	バンク 0FB レジスタ
REG_SB_F	バンク 0SB レジスタ
REG_Rx_R	バンク 1Rx レジスタ(x は 0 ~ 3)
REG_RxH_R	バンク 1Rx レジスタの上位 8 ビット(x は 0 ~ 1)
REG_RxL_R	バンク 1Rx レジスタの下位 8 ビット(x は 0 ~ 1)
REG_Ax_R	バンク 1Ax レジスタ(x は 0 ~ 1)
REG_FB_R	バンク 1FB レジスタ
REG_SB_R	バンク 1SB レジスタ
REG_Rx	B フラグの示すバンクの Rx レジスタ(x は 0 ~ 3)
REG_RxH	B フラグの示すバンクの Rx レジスタの上位 8 ビット(x は 0 ~ 1)
REG_RxL	B フラグの示すバンクの Rx レジスタの下位 8 ビット(x は 0 ~ 1)
REG_Ax	B フラグの示すバンクの Ax レジスタ(x は 0 ~ 1)
REG_FB	B フラグの示すバンクの FB レジスタ
REG_SB	B フラグの示すバンクの SB レジスタ
REG_USP	USP レジスタ
REG_ISP	ISP レジスタ
REG_FLG	FLG レジスタ
REG_PC	プログラムカウンタ
REG_INTB	INTB レジスタ
REG_SVF	SVF レジスタ
REG_SVP	SVP レジスタ
REG_VCT	VCT レジスタ
REG_DMDx	DMDx レジスタ(x は 0 ~ 1)
REG_DCTx	DCTx レジスタ(x は 0 ~ 1)
REG_DRCx	DRCx レジスタ(x は 0 ~ 1)
REG_DMAx	DMAx レジスタ(x は 0 ~ 1)
REG_DSAx	DSAx レジスタ(x は 0 ~ 1)
REG_DRAx	DRAx レジスタ(x は 0 ~ 1)

(記述例) バンク 0 の R0 レジスタの値を取得する場合

```
RequestGetRegister( REG_R0_F, &regValue );
```

unsigned long * regValue 取得したレジスタ値の格納先

戻り値： int status
TRUE 正常終了
FALSE エラー発生
機能： 指定したレジスタの値を取得します。

● レジスタ値の設定

関数名： int RequestPutRegister(int regNo, unsigned long regValue)
引数： int regNo 値を設定するレジスタの番号
regNo は、RequestGetRegister 関数を参照ください。
unsigned long regValue 設定するレジスタ値
戻り値： int status
TRUE 正常終了
FALSE エラー発生
機能： 指定したレジスタの値を設定します。

● 割り込みの発生

関数名： int RequestInterrupt(unsigned long vec, int ipl)
引数： unsigned long vec 発生させる割り込みのベクタ番号（ベクタアドレス）
PD308SIM, PD30SIM はベクタ番号。
PD32RSIM は EIT ベクタエントリ（0x00000080 のみ指定可）。
int ipl 発生させる割り込みの優先順位
PD32RSIM の場合は、優先順位はありませんので”0”を指定してください。
戻り値： int status
TRUE 正常終了
FALSE エラー発生
機能： 指定した割り込みを発生します。

● 割り込みの発生状況を確認

関数名： int RequestInterruptStatus(unsigned long * vec)
引数： unsigned long * vec 発生した割り込みのベクタ番号（ベクタアドレス）
PD308SIM, PD30SIM はベクタ番号。
PD32RSIM は EIT ベクタエントリ（0x00000080 のみ有効）。
戻り値： int happen 発生状況
TRUE 発生。 vec には、発生した割り込みのベクタ番号（ベクタアドレス）が格納されます。
FALSE 未発生。 vec は不定。
機能： 割り込みの発生状況を取得します。

- **総実行サイクル数の参照**

関数名： void RequestTotalCycle(unsigned long * cycle)

引数： unsigned long * cycle 取得した総実行サイクル数の格納先

戻り値： なし

機能： 現在の総実行サイクル数を参照します。

- **総実行命令数の参照**

関数名： void RequestInstructionNum(unsigned long * inst)

引数： unsigned long * inst 取得した総実行命令数の格納先

戻り値： なし

機能： 現在の総実行命令数を参照します。

- **ターゲットプログラムの実行停止**

関数名： void RequestStop(void)

引数： なし

戻り値： なし

機能： ターゲットプログラムの実行を停止します。

● エラー情報の取得

関数名： int RequestErrorNum(void)

引数： なし

戻り値： int errNum 発生したエラーの番号

errNom	エラー内容
000	エラー発生なし
001	指定したアドレス値が範囲外です。
002	指定した領域にメモリがないので、参照/書き込みができません。
003	必要なメモリが確保できません。
004	指定したデータサイズが範囲外です。
005	指定したアドレスにアクセスできません。
100	レジスタの記述に誤りがあります。
101	指定したデータ値が不正です。
200	指定したベクタが範囲外です。
201	指定した優先度が範囲外です。
202	必要なメモリが確保できません。

機能： 直前に実行した関数で発生したエラーの番号を取得します。

この関数は、以下の関数を呼び出して戻り値が FALSE の場合に発生しているエラーを取得できます。

- RequestGetMemory 関数
- RequestPutMemory 関数
- RequestGetRegister 関数
- RequestPutRegister 関数
- RequestInterrupt 関数

(使用例)

```

unsigned long data;
char    str[5];
if ( RequestGetMemory( 0x800, 4, data ) == FALSE ) {
    sprintf( str, "%d", RequestErrorNum() );
    MessageBox( NULL, str, "エラー番号", MB_OK );
                // メッセージボックスにエラー番号を表示
}

```

以下に上記関数を使用した記述例を示します。この例の網掛けの部分は、iofunc.cpp ファイルにテンプレートとして実装されている部分です。

```
void NotifyPostWriteMemory(unsigned long address, int length, unsigned long data)
{
    if (address == 0x3e0) {
        RequestGetRegister(REG_PC, &val);           // PC 値を取得
        RequestPutMemory(0x800, 4, val);           // 0x800 番地に PC 値を格納
    } else if (address == 0x3e1) {
        RequestPutRegister(REG_R0_F, data);        // バンク 0R0 レジスタに値を格納
        RequestInterrupt(21, 7);                   // タイマ A0 割り込みの発生
    }
}

return;
```

8. I/O DLL でターゲットプログラム内のシンボルを指定するには

本章では、I/O DLL でターゲットプログラム内のシンボルを指定する方法について説明します。

ターゲットプログラムで定義されているシンボルを I/O DLL で指定するには、PDxxSIM の拡張ウィンドウであるシンボルウィンドウを使用してシンボル情報を I/O DLL へ読み込む必要があります。

以下に、I/O DLL でターゲットプログラム内のシンボルを指定するための手順を、PD308SIM の場合を例にとって説明します。他の機種の場合は、“308”の記述を対応する機種名 (“30”, “32R”) に読み替えてください。

1. シンボルウィンドウを PD308SIM へ登録します。
2. シンボルウィンドウでシンボル情報をファイルに出力します。
3. 出力したファイルを I/O DLL で読み込んでシンボル情報を登録します。
4. I/O DLL でシンボルを指定します。

各手順の詳細を説明します。

1. シンボルウィンドウを PD308SIM へ登録します。

シンボルウィンドウは、以下の方法で PD308SIM へ登録してください。

- (1) C:\¥MTOOLS¥Iodll にある “SymbolWindow¥Pd308sim” ディレクトリのシンボルウィンドウ用 DLL ファイル “SymbolWindow.dll” を、C:\¥MTOOLS¥PD308SIM へコピーしてください。
- (2) C:\¥MTOOLS¥PD308SIM にある PD308SIMDLL.DEF ファイルをエディタでオープンし、[GENERAL] セクションの “WindowCount” のカウント数をインクリメントしてください。

```
[GENERAL]
ProductName=PD308SIM
WindowCount=9      WindowCount=10 に変更
```

- (3) 以下の内容を PD308SIMDLL.DEF の最後に追加してください。
(追加内容)

```
[Window9]
Title=Symbol Window
ID=50000
Module=SymbolWindow
Initialize=InitializeMDIChildFrame
Create=CreateMDIChildFrame
IsEnableToOpen=IsEnableToOpenMDIChildFrame
GetOptionMenuID=GetOptionMenuID
Append=1
```

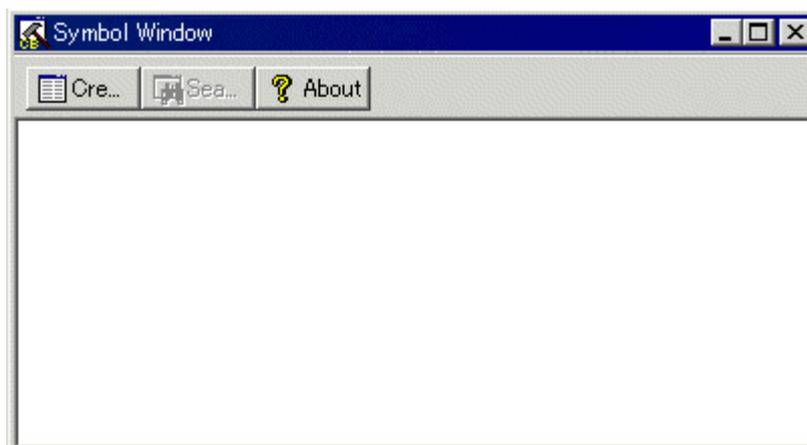
追加したセクションのセクション名 ([Window9]) の番号は、他のウィンドウの続きになるように設定してください。

なお、シンボルウィンドウの登録を解除するには、PD308SIMDLL.DEF で変更した内容を元に戻してください。

2. シンボルウィンドウでシンボル情報をファイルに出力します。

以下に手順を示します。

- (1) PD308SIM を起動して、メニュー [OptionalWindows] [Add-In Windows] [Symbol Window]を選択するとシンボルウィンドウがオープンします。

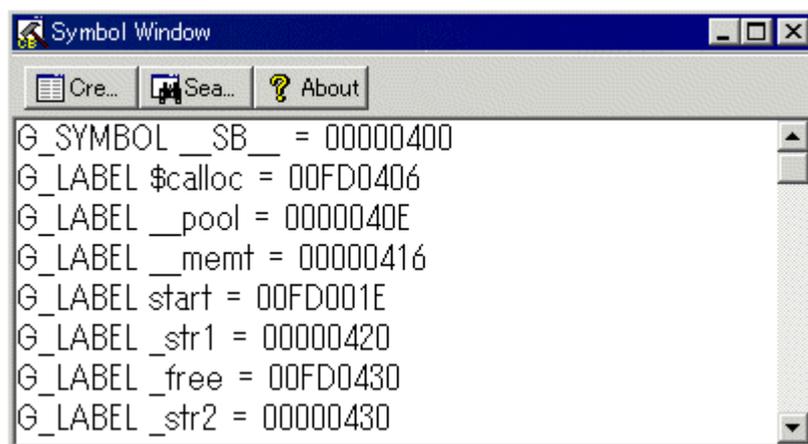


- (2) PD308SIM ヘターゲットプログラムをダウンロードします。ダウンロード後、シンボルウィンドウの Create symbol list ボタンを押して、シンボル情報を出力するファイルを選択してください。

(注意)

出力されるシンボル情報のうちローカルなシンボルについては、PD308SIM でスコープが設定されているモジュールファイルの情報です（スコープとは、ローカルシンボル/ローカルラベルの有効範囲のことです）。別のモジュールファイルに含まれるローカルシンボル/ローカルラベルの情報を出力する場合は、PD308SIM でスコープの設定を変更してください。詳細は、PD308SIM のオンラインヘルプ「PD308SIM ヘルプ」の scope コマンドについての説明を参照ください。

- (3) シンボル情報をファイルに出力すると、シンボルウィンドウにシンボル情報が表示されます。



- 表示内容は、以下のようになります。

シンボル種別 シンボル名 = シンボルアドレス

シンボル種別は、G_SYMBOL (グローバルシンボル)、L_SYMBOL (ローカルシンボル)、G_LABEL (グローバルラベル)、L_LABEL (ローカルシンボル) の順に表示されます。ターゲットプログラムに存在しないシンボルの種別は表示されません。
- シンボルを検索する場合は、Search symbol ボタンを押して検索するシンボル名を入力してください。
- ターゲットプログラムを再ダウンロードすると、シンボルウィンドウに表示されているシンボル情報は更新されます。このとき、シンボル情報を出力したファイルは、前回指定したファイルに上書きされます。シンボル情報を出力するファイルを変更する場合は、Create symbol list ボタンを押して再度シンボルファイルを指定してください。

(補足)

一度シンボル情報を出力するファイルを選択している (シンボルウィンドウにシンボル情報が表示されている) 場合は、それ以後ターゲットプログラムをダウンロードしたタイミングでシンボル情報が自動的に更新されます。

シンボルウィンドウにシンボル情報が表示されていない状態でターゲットプログラムをダウンロードした場合は、シンボル情報は更新されません。Create symbol list ボタンを押して、シンボル情報を出力してください。

I/O DLL での使用例を示します。

```
void NotifyReset(void)
{
    // リセットしたときにシンボル情報を読み込んで登録します。
    RegistrationSymbolList( "d:/sample/test.sym" );

    return;
}
void NotifyEnd(void)
{
    // I/O DLL で読み込んだシンボル情報を PDxxSIM の終了時に削除します。
    FreeSymbolList();

    return;
}
```

(補足)

RegistrationSymbolList 関数は、1つのシンボルファイルに含まれるシンボル情報のみ登録します。複数回呼び出した場合は、前回のシンボル情報を削除（メモリ開放）後、新しいシンボル情報を登録します。なお、同じシンボルファイルを指定した場合は、ファイルのタイムスタンプをチェックして、更新されている場合のみシンボル情報を再登録します。

4. I/O DLL でシンボルを指定します。

以下に手順を示します。

- I/O DLL でシンボルを値に変換するには、ChangeSymToVal 関数を使用します。関数の仕様を以下に示します。

(シンボルの値変換)

関数名： BOOL ConvertSymToVal(const char * pszSym, unsigned long * ulAddr,
int nType)

引数： const char * pszSym 変換するシンボル名
unsigned long * ulAddr 変換したシンボル値を格納するポインタ
int nType 検索する際に優先するシンボル種別
 TYPE_SYMBOL シンボル優先
 TYPE_LABEL ラベル優先

戻り値： BOOL status

TRUE 成功
FALSE シンボルが見つからない

機能： シンボルに対応する値を検索して ulAddr に格納します。検索する際に優先するシンボル種別は、nType に指定するシンボル種別によって以下のようになります。

優先順位	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	ローカルラベル
2	グローバルシンボル	グローバルラベル
3	ローカルラベル	ローカルシンボル
4	グローバルラベル	グローバルシンボル

(注意)

ConvertSymToVal 関数の引数 pszSym に指定するシンボル名は、シンボルウィンドウに表示されているシンボル名で指定してください。例えば、シンボルウィンドウに "_str1" とアンダーバー付きでシンボル名が表示されている場合は、引数 pszSym にも "_str1" とアンダーバー付きで指定してください。

I/O DLL での使用例を示します。

```
void NotifyPreReadMemory(unsigned long address, int length)
{
    unsigned long s_addr;
    unsigned long data;

    // アドレス 0x3e0 がリードされたときにシンボル "str1" の値を
    // 0x3e0 に代入します
    if ( address != 0x3e0 ) {
        return;
    }
    if ( ConvertSymToVal("_str1", &s_addr, TYPE_LABEL ) == FALSE ) {
        return;
    }
    RequestGetMemory(s_addr, length, &data);
    RequestPutMemory(address, length, data);
}
```

9. 制限事項

1. I/O DLL を使用してメモリへ入出力した値の変化は、PDxxSIM の GUI 出力機能、仮想ポート入力機能、仮想ポート出力機能、および I/O スクリプト機能を利用して参照することはできません。
2. PDxxSIM へ指定できる I/O DLL は、1 つのみです。複数の I/O DLL を指定することはできません。

以上

PDxxSIM I/O DLL キットユーザーズマニュアル

Rev. 1.00
03.05.01
RJJ10J0095-0100Z

COPYRIGHT ©2003 RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

PDxxSIM I/O DLL キット 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

RJJ10J0095-0100Z