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

I/O DLLキット

ユーザーズマニュアル

シミュレータデバッガ拡張キット

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2006.11

本資料ご利用に際しての留意事項

１．本資料は、お客様に用途に応じた適切な弊社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弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
２．本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
３．本資料に記載の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、あるいはその他軍事用途の目的で使用しないでください。また、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、それらの定めるところにより必要な手続を行ってください。
４．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などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認頂きますとともに、弊社ホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意下さい。
５．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、弊社はその責任を負いません。
６．本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独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下さい。弊社は、適用可否に対する責任は負いません。
７．本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会下さい。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
８．第７項にかかわらず、本資料に記載された製品は、下記の用途には使用しないで下さい。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
　１）生命維持装置。
　２）人体に埋め込み使用するもの。
　３）治療行為（患部切り出し、薬剤投与等）を行なうもの。
　４）その他、直接人命に影響を与えるもの。
９．本資料に記載された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、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて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、弊社はその責任を負いません。
１０．弊社は製品の品質および信頼性の向上に努めておりますが、特に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。弊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、誤動作防止設計などの安全設計（含むハードウエアおよびソフトウエア）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エアは、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致します。
１１．本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
１２．本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します。
１３．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弊社営業窓口までご照会下さい。

はじめに

I/O DLLキットは、シミュレータデバッガ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、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

I/O DLLキットは、全てのシミュレータデバッガで利用できるものではありません。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、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

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の「ソフトウェア使用権許諾契約書」に基づきます。また、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

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://japan.renesas.com/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、あらかじめご了承願います。

[MEMO]

目次

[MEMO]

1. 概要

I/O DLL とは、シミュレータデバッガ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。

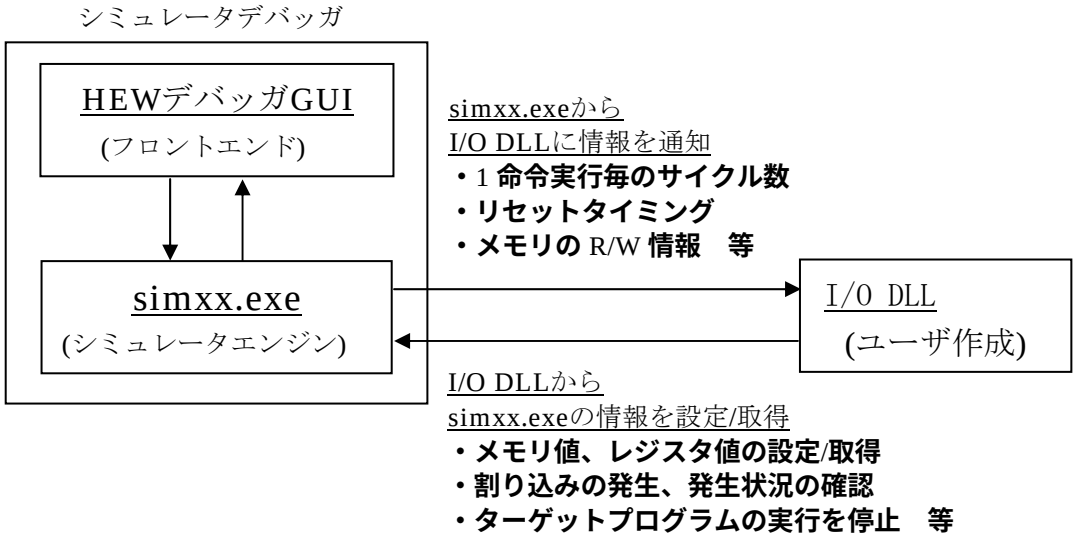
I/O DLL は、C/C++言語を使用して任意に作成することができます。I/O DLL を作成するには、市販の Windows アプリケーション開発環境である Microsoft Visual C++が必要です。

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 **M32C シリーズ用シミュレータデバッガ**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（“100”、“30”等）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:¥Renesas¥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:¥Renesas¥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。

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

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 の出力先がシミュレータデバッガをインストールしたディレクトリになるように設定してください。

   シミュレータデバッガは以下のディレクトリに格納されています。

   "HEW インストールディレクトリ
   ¥Tools¥Renesas¥DebugComp¥Platform¥PDTarget¥PD308SIM"

   （例）出力ファイル名：

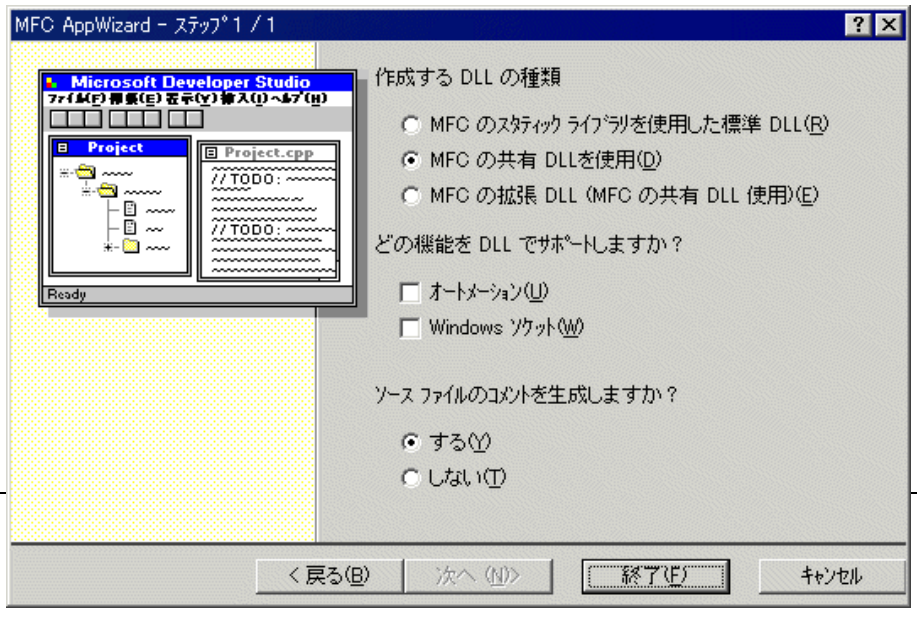
   C:¥Program Files¥Renesas¥HEW¥Tools¥Renesas¥DebugComp¥Platform¥
   PDTarget¥PD308SIM¥sample.dll

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。
5. ビルドすると、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。

1. 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。
   - プラットホームで 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。

   - 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:¥Renesas¥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。

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

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 の出力先がシミュレータデバッガをインストールしたディレクトリになるように設定します。
   （例）出力ファイル名：
   C:¥Program Files¥Renesas¥HEW¥Tools¥Renesas¥DebugComp¥Platform¥PDTarget¥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ー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、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、シミュレータデバッガで I/O DLL をロードして使用する方法を説明します。

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、シミュレータデバッガに I/O DLL を登録して使用する方法を **M32C シリーズ用シミュレータデバッガ**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（“100”、“30”等）に読み替えてください。

1. シミュレータデバッガをインストールしたディレクトリへ使用する I/O DLL（.dll ファイル）をコピーしてください。

   シミュレータデバッガは以下のディレクトリに格納されています。

   "HEW インストールディレクトリ
   ¥Tools¥Renesas¥DebugComp¥Platform¥PDTarget¥PD308SIM"

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、シミュレータデバッガをインストールしたディレクトリに存在します。ただし、シミュレータデバッガ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. シミュレータデバッガを起動すると 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シミュレータデバッガ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 **M32C シリーズ用シミュレータデバッガ**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（“100”、“30”等）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 の出力先がシミュレータデバッガをインストールしたディレクトリになるように設定されているか確認します。

   （例）出力ファイル名：

   C:¥Program Files¥Renesas¥HEW¥Tools¥Renesas¥DebugComp¥Platform¥PDTarget¥PD308SIM¥sample.dll

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シミュレータデバッガをインストールしたディレクトリ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。

   （例）デバッグセッションの実行可能ファイル：

   C:¥Program Files¥Renesas¥HEW¥Tools¥Renesas¥DebugComp¥Platform¥PDTarget¥PD308SIM¥sim308.exe

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メッセージが表示されます。

   “sim308.exe にはデバッグ情報がありません。続行する場合は[OK]を押してください。”

   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、シミュレータデバッガを起動すると 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、それを確認後にシミュレータデバッガ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

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 longtabsrData, 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(unsigend long address, int length, unsigend 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(unsigend long address, int length, unsigend 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
int ipl 発生させる割り込みの優先順位

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

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. 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、シミュレータデバッガの 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. シミュレータデバッガへ指定できる 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

[MEMO]

I/O DLL**キット**
ユーザーズマニュアル

発行年月日　2006**年**11**月**01**日**　Rev.1.00

発行　　**株式会社 ルネサス テクノロジ　営業統括部**
〒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**

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

RJJ10J1810-0100