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。

I/O スクリプトサンプルプログラムのご紹介

1. I/O スクリプトサンプルプログラムの概要

M16C/60 シリーズの内蔵デバイスをシミュレーションするための I/O スクリプトのサンプルを用意しています。サンプルプログラムには、以下のものが含まれています。

1. タイマのサンプル
2. AD 変換のサンプル
3. シリアル I/O のサンプル
4. CRC 演算回路のサンプル
5. DMAC のサンプル
6. MR30 用のタイマのサンプル

これらのサンプルプログラムの詳細に関しては、「I/O スクリプトサンプルプログラムの詳細」の章で説明します。

2. I/O スクリプトサンプルプログラムの使用方法

I/O スクリプトサンプルプログラムは、以下に示す手順に従ってシミュレータデバッガに組み込んでいただくことによって使用できます。

なお、I/O スクリプトサンプルプログラムは以下のディレクトリに格納されています。

"インストールディレクトリ
¥Tools¥Renesas¥DebugComp¥Platform¥PDTarget¥PD30SIM¥Samples"

- シミュレータデバッガを起動し、I/O タイミング設定ウィンドウをオープンします。
- I/O タイミング設定ウィンドウの読み込みメニューを選択します。選択するとファイルセレクションダイアログがオープンします。
- ここで、組み込みたい I/O スクリプトサンプルプログラムを選択します。

3. I/O スクリプトサンプルプログラムの詳細

3.1. タイマ A0（タイマモード）のサンプルプログラム（timerA0_TMode.ios）

このサンプルプログラムでは、タイマ A0 のタイマモードの動作をシミュレーションします。

「タイマの機能」

カウントソース：サイクル数をカウント
パルス出力機能：なし
ゲート機能　　：なし

「動作」

1. カウント開始フラグを 1 にすると 16 サイクル毎に、タイマ A0 レジスタの内容を 16 ずつダウンカウントします。
2. アンダフローすると、タイマ A0 レジスタにリロードレジスタの内容をリロードしてカウントを続けます。
   同時に、タイマ A0 割り込み要求ビットが 1 になります。
3. カウント開始フラグを 0 にすると、カウント値を保持してダウンカウントを停止します。
4. 割り込みが受け付けられると、タイマ A0 割り込み要求ビットが 0 になります。

「実チップのタイマとの違い」

1. プログラムでタイマ A0 レジスタの値を変更すると、直ちにタイマ A0 レジスタとリロードレジスタに書き込まれます。
   →実チップでは、直ちに書き込まれずリロードレジスタに保持されます。
2. プログラムでタイマ A0 割り込み要求ビットを 0 にしても、タイマ割り込みの要求をキャンセルできません。
3. カウントソースは、f1（1 分周）に固定です。

3.2.タイマ A0（タイマモード、ゲート機能選択時）のサンプルプログラム（timerA0_TMod_Gate.ios）

このサンプルプログラムでは、タイマ A0 のタイマモード（ゲート選択時）の動作をシミュレーションします。

「タイマの機能」

カウントソース：サイクル数をカウント
パルス出力機能：なし
ゲート機能　　：TA0IN 端子が 1（"H"レベル）の期間だけカウントを行う。

「動作」

1. カウント開始フラグが 1 で TA0IN 端子の入力が 1（"H"レベル）の時、16 サイクル毎にタイマ A0 レジスタの内容を 16 ずつダウンカウントします。
2. TA0IN 端子の入力が 0（"L"レベル）の時、カウント値を保持してダウンカウントを停止します。
3. アンダフローすると、タイマ A0 レジスタにリロードレジスタの内容をリロードしてカウントを続けます。
   同時に、タイマ A0 割り込み要求ビットが 1 になります。
4. カウント開始フラグを 0 にすると、カウント値を保持してダウンカウントを停止します。
5. 割り込みが受け付けられると、タイマ A0 割り込み要求ビットが 0 になります。

「TA0IN 端子の入力信号の作成方法」

timerA0_TMod_Gate.ios ファイルの下記の部分の I/O スクリプトを変更して入力信号を作成します。この例では、10000 サイクル毎に TA0IN 端子の入力信号の H/L を切り替えています。

```
{
  while(1){
    set [0x3ed] = [0x3ed] & 0xfd    ;TA0IN 端子を"L"にする
    waitc 10000                     ;L レベルを 10000 サイクル持続する
    set [0x3ed] = [0x3ed] | 0x02    ;TA0IN 端子を"H"にする
    waitc 10000                     ;H レベルを 10000 サイクル持続する
  }
}
```

「実チップのタイマとの違い」

1. プログラムでタイマ A0 レジスタの値を変更すると、直ちにタイマ A0 レジスタとリロードレジスタに書き込まれます。
   →実チップでは、直ちに書き込まれずリロードレジスタに保持されます。
2. プログラムでタイマ A0 割り込み要求ビットを 0 にしても、タイマ割り込みの要求をキャンセルできません。
3. カウントソースは、f1（1 分周）に固定です。

3.3.タイマ A0（タイマモード、パルス出力機能選択時）のサンプルプログラム（timerA0_TMod_Pulse.ios）

このサンプルプログラムでは、タイマ A0 のタイマモード（パルス出力機能選択時）の動作をシミュレーションします。

「タイマの機能」

カウントソース：サイクル数をカウント
パルス出力機能：あり
ゲート機能　　：なし

「動作」

1. カウント開始フラグを 1 にすると 16 サイクル毎に、タイマ A0 レジスタの内容を 16 ずつダウンカウントします。
2. アンダフローすると、タイマ A0 レジスタにリロードレジスタの内容をリロードしてカウントを続けます。
   同時に、タイマ A0 割り込み要求ビットが 1 になります。また、TA0OUT 端子の出力極性が反転します。
3. カウント開始フラグを 0 にすると、カウント値を保持してダウンカウントを停止します。また、TA0OUT 端子は 0（"L"レベル）を出力します。
4. 割り込みが受け付けられると、タイマ A0 割り込み要求ビットが 0 になります。

「実チップのタイマとの違い」

1. プログラムでタイマ A0 レジスタの値を変更すると、直ちにタイマ A0 レジスタとリロードレジスタに書き込まれます。
   →実チップでは、直ちに書き込まれずリロードレジスタに保持されます。
2. プログラムでタイマ A0 割り込み要求ビットを 0 にしても、タイマ割り込みの要求をキャンセルできません。
3. カウントソースは、f1（1 分周）に固定です。

「注意」

TA0OUT 端子の出力の変化を I/O タイミング設定ウィンドウの仮想ポート出力機能を利用して参照することはできません。

3.4.タイマ A0（ワンショットタイマモード）のサンプルプログラム (timerA0_OneShotTM.ios)

このサンプルプログラムでは、タイマ A0 のワンショットタイマモードの動作をシミュレーションします。

「タイマの機能」

カウントソース　：サイクル数をカウント
パルス出力機能　：なし
カウント開始条件：ワンショット開始フラグへの 1 の書き込み

「動作」

1. カウント開始フラグが 1 の状態でワンショット開始フラグを 1 にすると 16 サイクル毎に、タイマ A0 レジスタの内容を 16 ずつダウンカウントします。
2. アンダフローすると、タイマ A0 レジスタにリロードレジスタの内容をリロードしてカウントを停止します。
   同時に、タイマ A0 割り込み要求ビットが 1 になります。
3. カウント中にワンショット開始フラグに 1 が書き込まれた場合、再度リロードレジスタの値をリロードしてカウントを続けます。
4. カウント開始フラグを 0 にすると、ダウンカウントを停止し、リロードレジスタの内容をリロードします。
5. 割り込みが受け付けられると、タイマ A0 割り込み要求ビットが 0 になります。

「実チップのタイマとの違い」

1. プログラムでタイマ A0 レジスタの値を変更すると、直ちにタイマ A0 レジスタとリロードレジスタに書き込まれます。
   →実チップでは、直ちに書き込まれずリロードレジスタに保持されます。
2. プログラムでタイマ A0 割り込み要求ビットを 0 にしても、タイマ割り込みの要求をキャンセルできません。
3. カウントソースは、f1（1 分周）に固定です。

3.5.A-D 変換器（単発モード）のサンプルプログラム（AD.ios）

このサンプルプログラムでは、A-D 変換器の単発モードの動作をシミュレーションします。なお、サンプルプログラムでは A-D レジスタ 0 への入力をシミュレーションします。

「シミュレータデバッガにおける A-D 変換器の考え方（実チップとの動作の違い）」

シミュレータデバッガでは、アナログ入力のシミュレーションをサポートしていません。そのため、A-D 変換後の入力値を A-D レジスタに入力することで、A-D 変換器の機能を擬似的に実現します。

「動作」

1. A-D 変換開始フラグを 1 にすると、A-D 変換割り込みが発生します。同時に A-D レジスタ 0 にデータを入力します。
2. その時、A-D 変換割り込み要求ビットが 1 になります。また、A-D 変換開始フラグが 0 になり、A-D 変換器の動作を停止します。
3. 割り込みが受け付けられると、A-D 変換割り込み要求ビットが 0 になります。
4. その後、再び A-D 変換開始フラグを 1 にすると、入力データがなくなるまで上記の 1～3 の動作を行います。

「A-D 入力データの作成方法」

AD.ios ファイルの下記の部分の I/O スクリプトを変更して A-D 入力データを作成します。

```
{
  ;AD データの設定
  ;A-D 変換割り込みが発生する毎に、0x0、0x1、0x2 の順にデータを入力します。

  set #isint:14, [0x3c0] = 0x0, 0x1, 0x2, 0x3, →ここに入力データを定義します

  set %data_exist = 0       ;AD データの入力終了
}
```

「注意」

プログラムで A-D 変換割り込み要求ビットを 0 にしても、A-D 変換割り込みの要求をキャンセルできません。

3.6.シリアル I/0（受信）のサンプルプログラム（SI0.ios）

このサンプルプログラムでは、シリアル I/O（受信）の動作をシミュレーションします。なお、サンプルプログラムでは UART0 への入力をシミュレーションします。

「サンプルプログラムにおけるシリアル I/O の考え方（実チップとの動作の違い）」

サンプルプログラムでは、シリアル I/O の入力として RxD から入力信号を取り込むのではなく、UART 受信バッファレジスタに直接データを入力することで簡易的にシリアル I/O の動作をシミュレーションしています。

なお、サンプルプログラムはクロック同期型シリアルモード、クロック非同期型シリアルモード共通で使用できます。但し、クロック非同期型シリアルモードでの受信データ長は 8 ビットです。

「動作」

1. 受信許可ビットを 1 にすると、UART0 受信割り込みが発生します。同時に UART0 受信バッファにデータを入力します。
2. その時、受信完了フラグと UART0 割り込み要求ビットが 1 になります。
3. 受信完了フラグは、UART0 受信バッファレジスタの下位バイトを読み出したとき 0 になります。
4. 割り込みが受け付けられると、UART0 割り込み要求ビットが 0 になります。
5. その後、10000 サイクル毎に入力データがなくなるまで上記の 1～4 の動作を行います。

「受信データの作成方法」

SIO.ios ファイルの下記の部分の I/O スクリプトを変更して受信データを作成します。

```
{
  ;受信データの設定
  ; UART0 割り込みが発生する毎に、0x0、0x1、0x2 の順にデータを入力します。

  set #isint:18, [0x3a6] = 0x0, 0x1, 0x2, 0x3, →ここに受信データを定義します

  set %data_exist = 0       ;受信データの入力終了
}
```

「受信データ入力タイミングの変更方法」

SIO.ios ファイルの下記の部分の I/O スクリプトを変更して受信データ入力タイミングを変更します。サンプルプログラムでは、10000 サイクル毎にデータを入力します。

```
if(%data_exist == 1){       ;受信データ有無
        waitc 10000   →ここを変更して、任意サイクル毎にデータを入力するように
                        変更できます。
```

「注意」

1. プログラムで UART0 割り込み要求ビットを 0 にしても、UART0 割り込みの要求をキャンセルできません。
2. エラー検知は、オーバランエラーだけを検知できます。

3.7.CRC 演算回路のサンプルプログラム（CRC.ios）

このサンプルプログラムでは、CRC 演算回路の動作をシミュレーションします。

「動作」

1. CRC データレジスタに初期値 0 を設定します。
2. CRC インプットレジスタに 1 バイトのデータを書き込むと、書き込んだデータと CRC レジスタの内容に基づいて、CRC コードが CRC データレジスタに生成されます。
3. 連続数バイト CRC 演算を行う場合には、続けて次のデータを CRC インプットレジスタに書き込んで下さい。
4. 全データを書き終えた後の CRC データレジスタの内容が CRC 符号となります。

「実チップの CRC 演算回路との違い」

1. 実チップでは、CRC コードの生成に 2 マシンサイクル要しますが、サンプルプログラムではサイクルは要しません（加算されません）。

3.8.DMAC のサンプルプログラム（DMAC.ios）

このサンプルプログラムでは、DMAC のリピート転送の動作をシミュレーションします。
DMA0 と DMA1 の動作を共にシミュレーションできます。

「DMA0、DMA1 の機能」

転送空間　：　以下の 3 種類

(1) 1M バイトの任意の空間から固定アドレス
(2) 固定アドレスから 1M バイトの任意空間
(3) 固定アドレスから固定アドレス

DMA 転送要因　　：　タイマ A0
チャネルの優先順位：　DMA0 の転送要求と DMA1 の転送要求が同時に発生した場合、DMA0 の転送が優先して行われます。
転送モード　　：　リピート転送のみ
転送単位　　：　8 ビット/16 ビット

「動作」

1. DMA 許可ビットが 1 のときタイマ A0 割り込みが発生すると、DMA 要求が受け付けられます。
2. DMA 要求が受け付けられると、指定された転送空間、転送単位で DMA 転送を行います。
3. DMA0/1 転送カウンタがアンダフローしても DMA 許可ビットは 1 のままです。DMA0/1 転送カウンタがアンダフローしたとき DMA0/1 割り込み要求ビットが 1 になります。
4. DMA0/1 割り込みが受け付けられると、DMA0/1 割り込み要求ビットが 0 になります。
5. DMA0/1 転送カウンタがアンダフローした後、次の DMA 要求が発生すると 1.に戻り、DMA 転送を繰り返します。

「実チップの DMAC との違い」

1. プログラムで DMA0/1 割り込み要求ビットを 0 にしても、DMA0/1 割り込みの要求をキャンセルできません。
2. 転送モードには、リピート転送のみ指定できます。
3. DMA 要求ビットのシミュレーションを行っていないため、DMA 許可ビットが 1 のとき、タイマ A0 割り込みが発生すると DMA 要求が必ず受け付けられます。

「DMA 転送要因として他の割り込みを使用する場合の変更方法」
タイマA1に変更する場合を例にとって説明します。DMAC.iosの以下の2箇所を変更します。

●DMA0 の変更箇所

```
if(%dma0_start == 1){          ;DMA0 開始
        pass #isint:21, 1   →   ここを pass #isint:22, 1 に変更します。
```

●DMA1 の変更箇所

```
if(%dma1_start == 1){          ;DMA1 開始
        pass #isint:21, 1   →   ここを pass #isint:22, 1 に変更します。
```

上記のように、DMA 転送要因となる割り込みのベクタ番号を#isint 文で指定して下さい。

3.9. MR30 用のタイマのサンプルプログラム（MR.ios）

このサンプルプログラムでは、MR30 でタイマ A0 を使用する場合の動作をシミュレーションします。

シミュレータデバッガで MR30 を使用したアプリケーションプログラムを実行する場合、以下の設定を行う必要があります。

MR30 では、周期ハンドラやアラームハンドラ、get_tim システムコール、dly_tsk システムコール等が参照するシステムクロックを設定するために、以下の例のようにコンフィグレーションファイル中で MR30 が使用するタイマとタイマ割り込みが発生する時間間隔を指定します。

```
clock{
        mpu_clock       = 10MHz;
        timer           = A0;
        IPL             = 4;
        unit_time       = 100ms;        // ms
        initial_time    = 0:0:0;
};
```

この例の場合、システムのタイマ割り込み用にタイマ A0 を使用し、タイマ A0 の割り込み発生間隔を 100ms に設定しています。

シミュレータデバッガで MR30 を用いたアプリケーションプログラムを実行する場合、MR30 で使用するタイマの設定（この場合タイマ A0）を行う必要があります。

以下にサンプルプログラム（mr.ios）を示します。

```
;タイマ A0 の定義例
{
    while(1){                                   →  while 文
        if( ([0x380].b & 0x01) == 0x01){        →  タイマ A0 のカウント開始フラグのチェック
            waitc [0x386].w + 1                 →  タイマ A0 に設定された分周比のサイクル数分
                                                   I/O スクリプトの実行をウェイトする
            int 21, [0x55] & 0x7                →  タイマ A0 の仮想割り込みを発生
                                                   （優先順位は、割り込み制御レジスタを参照）
        }else{
            waiti 100                           →  100 命令分 I/O スクリプトの実行をウェイト
        }
    }
}
```

この I/O スクリプトを I/O タイミング設定ウィンドウの読み込みメニューでシミュレータデバッガに登録することにより MR30 のアプリケーションのシミュレートが行えます。

「他のタイマを使用する場合の変更方法」
タイマ A3 に変更する場合を例にとって説明します。

●タイマ A0 のサンプルプログラム

```
{
        while(1){
          if( ([0x380].b & 0x01) == 0x01){
             waitc [0x386].w + 1
              int 21, [0x55] & 0x07
          }else{
             waiti 100
          }
        }
}
```

↓　タイマ A3 を使用するように以下の箇所を変更します。

●タイマ A3 のサンプルプログラム

```
{
        while(1){
          if( ([0x380].b & 0x08) == 0x08){   →  タイマ A3 のカウント開始フラグを参照するように変更
             waitc   [0x38C].w + 1            →  タイマ A3 レジスタの分周比を参照するように変更
              int 21,   [0x58] & 0x07         →  タイマ A3 割り込み制御レジスタの割り込み優先レベル選択ビットを参照するように変更
          }else{
             waiti 100
          }
        }
}
```