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。

E10A-USB エミュレータ用 フラッシュメモリ ダウンロードプログラム アプリケーションノート

要旨

E10A－USB には「フラッシュメモリへのダウンロード機能」があります。この機能を使用する場合ユーザにダウンロードプログラムご用意いただく必要があります。本アプリケーションノートではこのダウンロードプログラムについてサンプルプログラムを紹介し、カスタマイズ方法及び H10A-USB を使用したダウンロード手順を紹介しています。

目次

1. 重要事項

本資料の作成にあたりましては、弊社にて動作確認を行っておりますが、弊社にて動作を保証するものではありませんので、あらかじめご了解のほどお願い申し上げます。

【注】 本アプリケーションノートをお読みになる前に必ず、各マイコンファミリ用 E10A-USB エミュレータ ユーザーズマニュアル（HS0005KCU01HJ）デバッガ編「フラッシュメモリへのダウンロード機能」をご一読ください。

本アプリケーションノートではアプリケーション例として SH7727 SolutionEngine（（株）日立超 LSI システムズ製）に接続された外部フラッシュメモリにユーザプログラムをダウンロードする例を紹介しています。

SH7727 SolutionEngine とユーザシステムの仕様が異なる場合、サンプルとして提供していますライト/消去モジュールをカスタマイズする必要があります。

ご使用にあたりましては、本資料をよくお読みになり内容をご理解いただいた上でカスタマイズし、お客様ご自身で動作確認を行っていただきますよう、宜しくお願い申し上げます。

2. 機能

本サンプルプログラムは E10A-USB エミュレータの「フラッシュメモリダウンロード機能」を使い、外付けフラッシュメモリ領域にユーザプログラムをダウンロードすることができます。

ユーザプログラムをダウンロードするためには、以下が必要です。

- 本サンプルプログラム
  - フラッシュメモリにライトするプログラム
    （以後、ライトモジュールと呼びます）
  - フラッシュメモリを消去するプログラム
    （以後、消去モジュールと呼びます）
- ライト/消去モジュールをダウンロードし実行する RAM 領域

E10A-USB エミュレータの「フラッシュメモリダウンロード」機能の詳細は、各マイコンファミリ用 E10A-USB エミュレータ ユーザーズマニュアル（HS0005KCU01HJ）デバッガ編「フラッシュメモリへのダウンロード機能」を参照してください。

本資料は、マイコンと接続されたフラッシュメモリにプログラムをダウンロードするためのライト/消去モジュールを、ユーザシステム仕様（対象マイコンとの接続形態）に合わせてカスタマイズしていただく際の修正内容を示しております。

3. ライト/消去モジュールの作成

「7 サンプルプログラム修正の詳細」に掲載されたサンプルプログラム（fmtool.src）は、ユーザシステム仕様に合わせてカスタマイズしていただく必要があります。

カスタマイズしていただく箇所は、ライト/消去モジュールが使用するユーザシステムの RAM アドレスおよび、ユーザプログラムをダウンロードするフラッシュメモリのアドレスです。

さらに、CUI コマンド方式のフラッシュメモリの場合は、各セクタブロックの先頭アドレスの合わせ込みが必要です。

3.1 カスタマイズの詳細内容

ユーザシステム上の RAM のアドレス（ライト/消去モジュールをダウンロードし実行する RAM のアドレス）に合わせて、fmtool.src の下記に示す箇所の確認および修正を行っていただく必要があります。fmtool.src 上の位置についての詳細は、「7 サンプルプログラム修正の詳細」をご参照ください。下記に示すアドレスは、「5.3 [Configuration]ダイアログボックスの設定」でも一部使用します。

```
O_FMErase          .equ   H'0C001000  … 消去モジュールの先頭アドレス
O_FMWrite          .equ   H'0C001100  … ライトモジュールの先頭アドレス
FM_TOOL_STACK      .equ   H'0C002000  … ライト/消去モジュール用スタックエリア先頭アドレス
FM_TOP_ADDRESS     .equ   H'00000000  … ユーザシステムのフラッシュメモリの先頭アドレス
```

また、フラッシュメモリが「CUI コマンド方式」の場合は、下記に示す各セクタブロックの先頭アドレスを、ご使用になるフラッシュメモリに合わせていただく必要があります。

```
FM_ERASE_ADDRESS:                  … 各セクタブロックの先頭アドレス
                   .data.l   H'00000000, H'00080000, H'00100000, H'00180000
                   .data.l   H'00200000, H'00280000, H'00300000, H'00380000
                   .data.l   H'00400000, H'00480000, H'00500000, H'00580000
                   .data.l   H'00600000, H'00680000, H'00700000, H'00780000
                      :                                  :
                   .data.l   H'FFFFFFFF
```

4. 実行ファイルの作成

「3 ライト/消去モジュールの作成」に従い、必要に応じて fmtool.src を修正後、実行ファイルを作成します。

HEW の[オールビルド]ボタンを押してビルドを実行します。これによりアセンブルおよびリンクが実行され実行ファイルを作成します。

ビルドが正常に終了しますと、ビルドコンフィグレーションと同じ名前のフォルダ（通常は release フォルダ）の下に fmtool.mot が作成されます。

5. ユーザプログラムダウンロードの手順

「4 実行ファイルの作成」において作成された fmtool.mot を使用してユーザプログラムをフラッシュメモリにダウンロードする手順を以下に示します。以下では SH7727 SolutionEngine（（株）日立超 LSI システムズ製）を使用した例を示します。

5.1 ダウンロード環境の準備

(1) PC に接続した E10A-USB と、ユーザシステムを接続します。
(2) HEW を起動し、ユーザプログラムのワークスペースを開きます。
(3) 図 1 に示す[CPU select]ダイアログボックスが表示されます。
ご使用の CPU をドロップダウンリストボックスより選択し、[OK]ボタンを押してください。

図 1 [CPU select]ダイアログボックス

(4) [Connecting]ダイアログボックスが表示され、エミュレータの接続を開始します。
図 2 に示すダイアログボックスが表示されます。

図 2 信号入力要求メッセージのダイアログボックス

ユーザシステムの電源を入れます。
ユーザシステムから RESET 信号を入力し、[OK]ボタンをクリックします。

HEW の[Output]ウィンドウに”Connected”と表示されたら、E10A-USB エミュレータの起動は完了です。

5.2 バスステートコントローラの設定

CPU から RAM 及びフラッシュメモリへアクセス出来るように、マイコンの設定を行います。HEW で[I/O]ウィンドウを表示します。図 3 に示す[I/O]ウィンドウでユーザシステムに合わせた I/O レジスタの設定を行います。

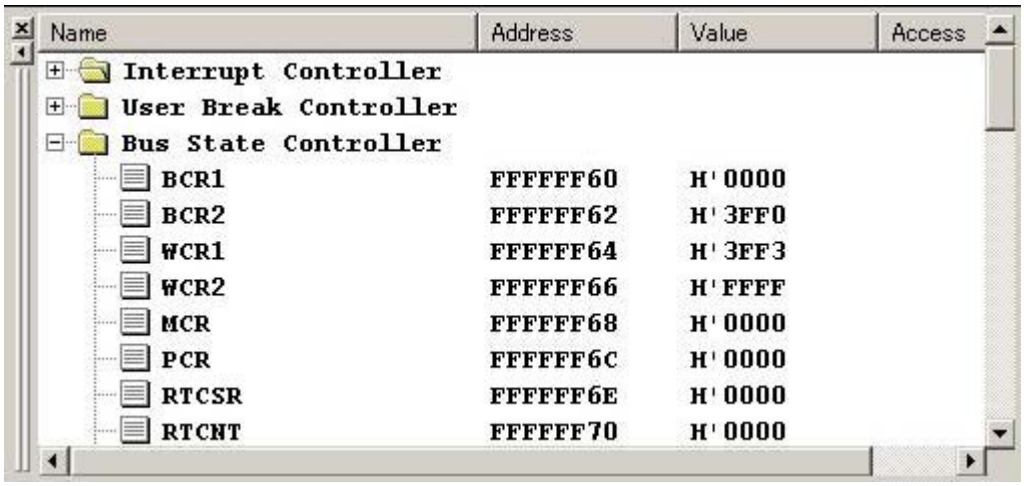

図 3 [I/O]ウィンドウ

SH7727 SolutionEngine の場合の設定例を以下に示します。

BCR1	FFFFFF60	H’0008
BCR2	FFFFFF62	H’2FF3
MCR	FFFFFF68	H’512C
RTCNT	FFFFFF70	H’a500
RTCOR	FFFFFF72	H’a50C
RFCR	FFFFFF74	H’a400
RTCSR	FFFFFF6E	H’a518
SDMR	FFFFE880	H’0

5.3 [Configuration]ダイアログボックスの設定

図 4 に示す[Configuration]ダイアログボックスの[Loading flash memory]ページで、E10A-USB エミュレータを使用して外部フラッシュメモリにユーザプログラムをダウンロードするための設定を行います。

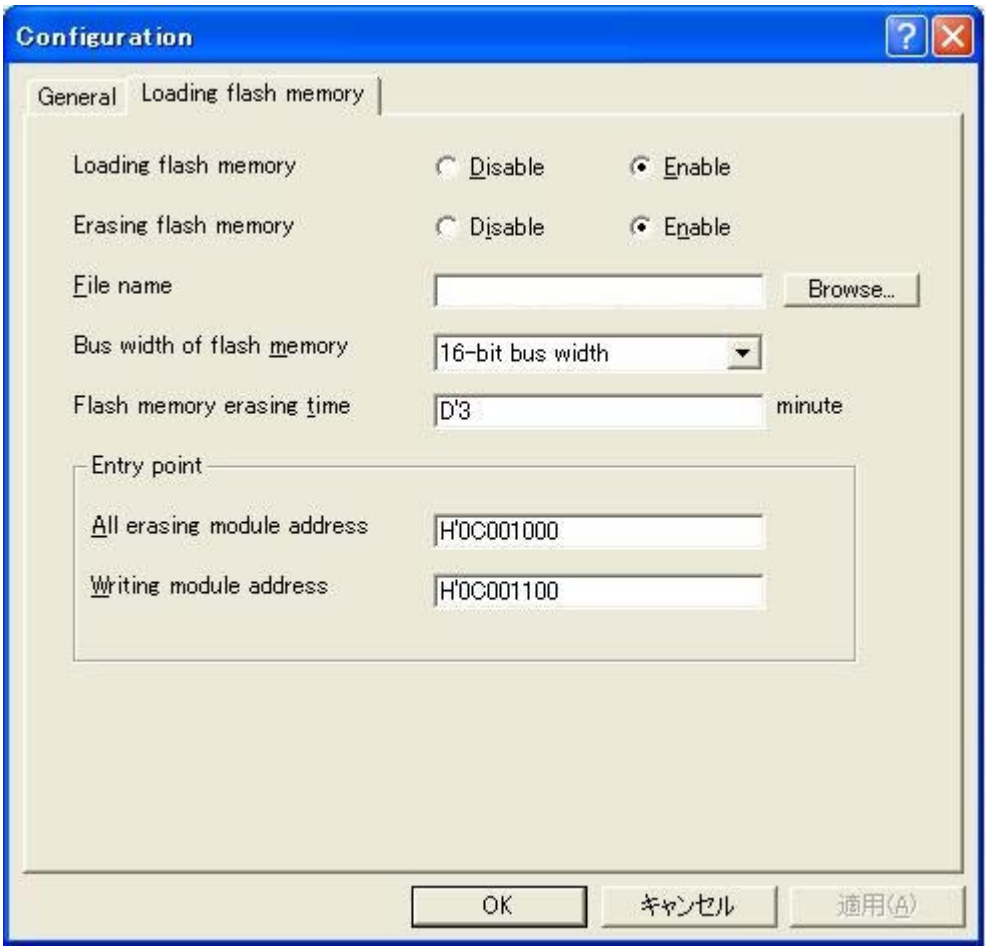

図 4 [Configuration]ダイアログボックス（[Loading flash memory]ページ）

各項目の設定を以下に示します。

Loading flash memory	：	Enable をチェックします。
Erasing flash memory	：	Enable をチェックします。
File Name	：	「4 実行ファイルの作成」で作成した fmtool.mot を指定します。
Bus width of flash memory	：	対象マイコンとフラッシュメモリの接続バス幅を指定します。
All erasing module address	：	消去モジュールの先頭アドレスを指定します。
Writing module address	：	ライトモジュールの先頭アドレスを指定します。

設定が終了したら[OK]ボタンをクリックします。

5.4 ユーザプログラムのダウンロード

「5.3 [Configuration]ダイアログボックスの設定」が完了したら、図 5 に示す通常のプログラムロード機能にて、ユーザプログラム（フラッシュメモリにロードしたいプログラム）をダウンロードします。

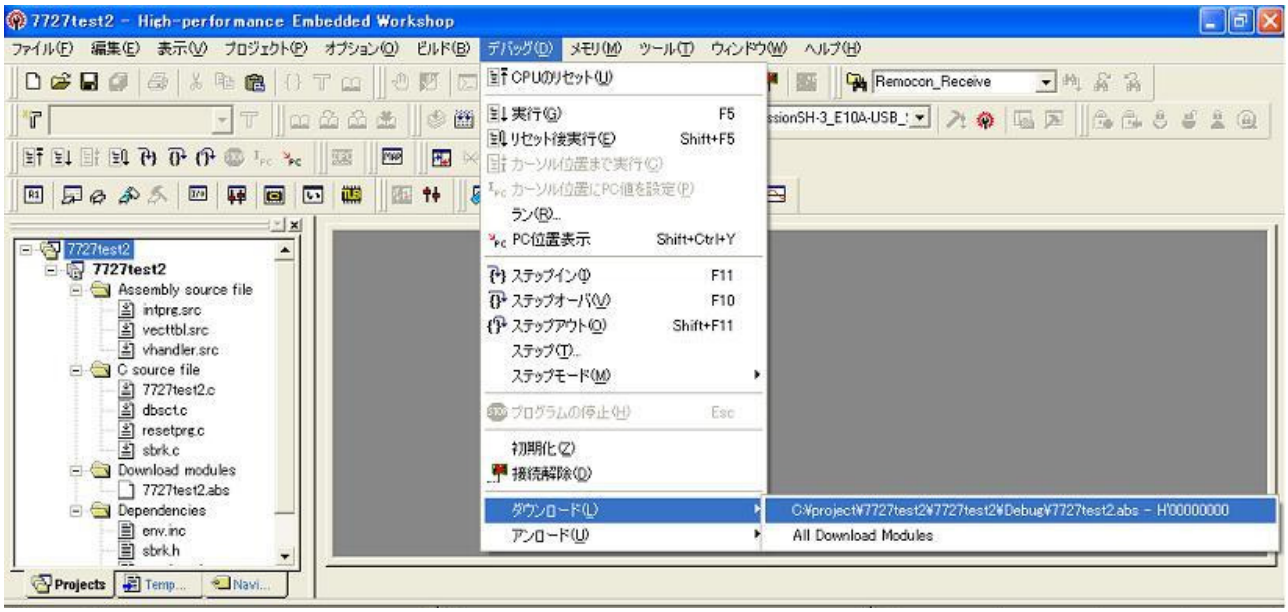

図 5 ユーザプログラムのダウンロード

5.5 ダウンロード後の処理について

図5に示すプログラムロード機能にてプログラムをダウンロードする場合は、直前に「5.3 [Configuration] ダイアログボックスの設定」で設定した内容にしたがって動作します。したがって、フラッシュメモリへプログラムロードを行った後、周辺フラッシュメモリ以外のメモリへのプログラムロードを行う場合は、図6に示すように[Configuration]ダイアログボックスの設定の[Loading flash memory]設定を「Disable」にしてください。

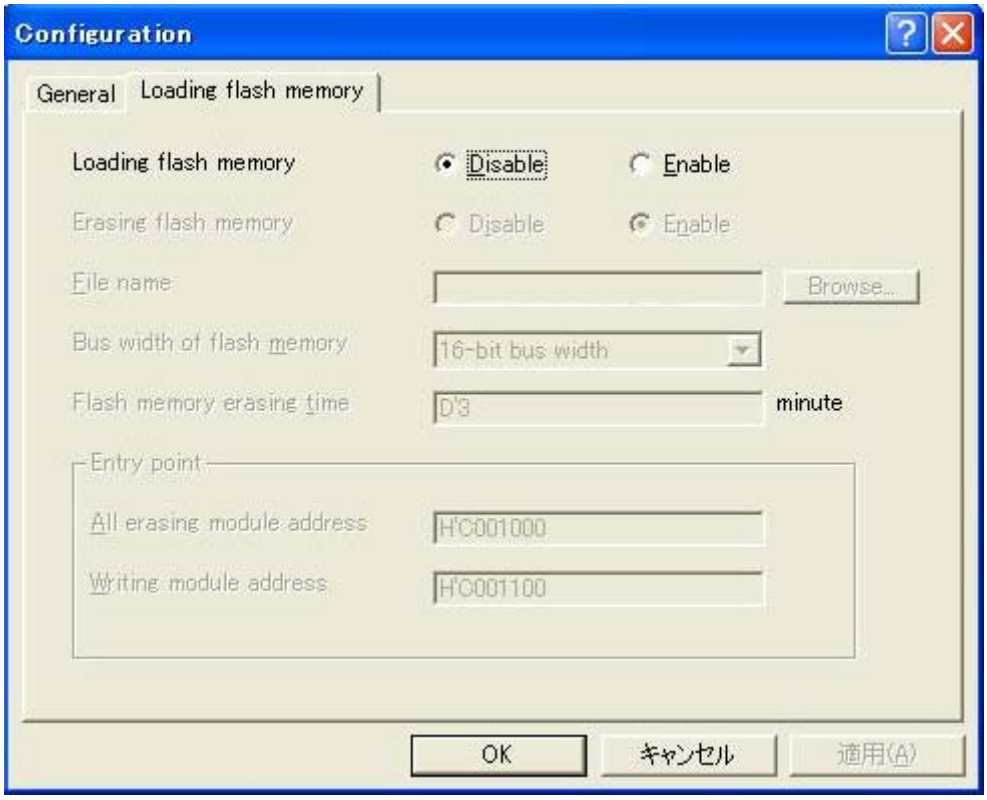

図 6 周辺フラッシュメモリ以外のメモリへのプログラムロード

6. Q&A

(1) 操作の手順についてですが、まずユーザシステムと正常に接続してから、［オプション］−［エミュレータ］−［システム］の Configuration 画面で「Loading flash memory」を設定し、ユーザプログラムのダウンロードを実行しています。この手順で正しいのでしょうか。

— 問題ありません。

(2) FLASH ライト／消去モジュールの S ファイルに含まれるのは、FLASH ライト／消去モジュールのみで、CPU の初期化処理などは必要ないのでしょうか。

— CPU の初期化処理は、FLASH ライト／消去モジュールの S ファイルには必要ありません。
CPU の初期化処理は、別途ダウンロード実施前に、実施してください。
（（5）の回答を参照ください。）

(3) RAM へ転送されるのは、FLASH ライト／消去モジュールだけで、ユーザプログラムは、E10A-USB エミュレータから RAM の FLASH ライト／消去モジュールを介して直接 FLASH に書かれるのでしょうか。

— 必要な RAM のサイズは FLASH ライト／消去モジュールの分だけです。
RAM へ転送されるのは、FLASH ライト／消去モジュールのみとなります。

(4) FLASH ライト／消去モジュールの転送先は、S ファイル中のアドレスなのでしょうか。

— FLASH ライト／消去モジュールの転送先は、S ファイル作成時に指定されているアドレスとなります。
FLASH ライト／消去モジュール内で、.section にてアドレスを割り付けている場合、FLASH ライト／消去モジュールの転送先は、.section にて設定しているアドレスとなります。

(5) SDRAM の設定について

各マイコンファミリ用 E10A-USB エミュレータ ユーザーズマニュアル（HS0005KCU01HJ）デバッガ編「プログラムをダウンロードする」に、注記として「外部 RAM にプログラムをダウンロードする場合は、必ずダウンロード対象領域のバスステートコントローラの設定を行ってから、ダウンロードを実行してください。特に SDRAM の初期化、バス幅の設定がユーザシステムに合っているかを十分にご確認ください。」とありますが、SDRAM の初期化は E10A-USB エミュレータをどのように操作して行ったらよいのでしょうか。

— SDRAM の初期化は E10A-USB エミュレータの IO レジスタウィンドウより、対象のレジスタへ値を直接入力し設定します。
冒頭でも記述いたしましたが、FLASH ライト／消去モジュールの転送先のメモリが Read/Write 可能となるように設定後、［オプション］−［エミュレータ］−［システム］の Configuration 画面の「Loading flash memory」設定を確認し、ダウンロードを実施してください。

7. サンプルプログラム修正の詳細

以下に添付したサンプルプログラムは RENESAS 製フラッシュメモリのワードモード接続用、および FUJITSU 製フラッシュメモリのワードモード接続用のものです。

各フラッシュメーカ共通で修正が必要な個所は、EQU 部にある①～④の 4 箇所です。

「CUI コマンド方式」のフラッシュメモリの場合はさらに⑤の部分も修正してください。

バイトモード/ワードモードどちらも修正個所は同じです。バイトモードの場合はバス幅 8 ビット(フラッシュメモリ 1 個使用)/16 ビット(フラッシュメモリ 2 個使用)/32 ビット(フラッシュメモリ 4 個使用)のすべてに対応した作りとなっています。同様にワードモードの場合は 16 ビット(フラッシュメモリ 1 個使用)/32 ビット(フラッシュメモリ 2 個使用)のすべてに対応しています。

(a) RENESAS 製フラッシュメモリ

ファイル名：FMTOOL.src

```
;+=========================================================================+
;|                                                                         |
;| Flash memory tool program sample                                        |
;|    SuperH Family Flash memory load is supported                         |
;|    Copyright (C) 2004 Renesas Technology Corp. All rights reserved.     |
;|    Licensed Material of Renesas Technology Corp.                        |
;|    Erasing flash memory routine : O_FMErase                             |
;|    Writing flash memory routine : O_FMWrite                             |
;|    Stack pointer : FM_TOOL_STACK                                        |
;|    Flash memory top address : FM_TOP_ADDRESS                            |
;|                                                                         |
;| Target flash memory : RENESAS [Word mode]                               |
;|                                                                         |
;+=========================================================================+
;
;+=========================================================================+
;|      EQU                                                                |
;+=========================================================================+
O_FMErase       .equ  H'0C001000
O_FMWrite       .equ  H'0C001100
FM_TOOL_STACK   .equ  H'0C002000
FM_TOP_ADDRESS  .equ  H'00000000
;
FM_CMD_ERASE    .equ  H'00000020
FM_CMD_WRITE    .equ  H'00000040
FM_CMD_READ     .equ  H'ffffffff
FM_CMD_STSCLR   .equ  H'00000050
FM_CMD_CONFIRM  .equ  H'000000d0
;
FM_CHK_SR7      .equ  H'00000080
FM_CHK_SR5      .equ  H'00000020
FM_CHK_SR4      .equ  H'00000010
FM_CHK_SR3      .equ  H'00000008
;
FM_OK           .equ  H'0000
FM_BT           .equ  H'4254
;
TYPE_BYTE       .equ  H'4220
TYPE_WORD       .equ  H'5720
TYPE_LONG       .equ  H'4c20
;
;
;
   .align      4
```

①

消去モジュールの先頭アドレス

ユーザシステムの RAM 領域を使用します。サイズは H’100 バイト以上確保してください。

②

ライトモジュールの先頭アドレス

ユーザシステムの RAM 領域を使用します。サイズは H’800 バイト以上確保してください。

③

ライト/消去モジュール用スタックエリア先頭アドレス

ユーザシステムの RAM 領域を使用します。サイズは H’100 バイト以上確保してください。

④

ユーザシステムのフラッシュメモリの先頭アドレス

ユーザシステムのフラッシュメモリ領域の先頭アドレスを設定します。

```
;=====================================================================
;
; NAME = FM_TOOL_ERASE;
; FUNC = The routine of erasing all flash memories.
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  : access size = 0x4220("B ");
;                            = 0x5720("W ");
;                            = 0x4c20("L ");
; OUTP = R0.L  : status      = 0 (OK);
;                            !=0 (ERROR);
;
;=====================================================================
;  .org     O_FMErase
   .section fm_erase,CODE,LOCATE=O_FMErase
FM_TOOL_ERASE:
   mov.l    #FM_TOOL_STACK,r15      ; Set stack address to R15 register
   sts.l    pr,@-r15                ;
   mov.l    r3,@-r15                ;
;
   mov      r4,r0                   ;>>> Check an argument
   shlr8    r0                      ;
   cmp/eq   #H'42,r0                ; 'B' ? -> Illegal
   bt       FM_ERASE_BYTE           ;
   cmp/eq   #H'57,r0                ; 'W' ?
   bt       FM_ERASE_WORD           ;
   bf       FM_ERASE_LONG           ; 'L' jump
;
;
;>>> Call a module for the bus width of 8 bits --> Illegal call
FM_ERASE_BYTE:
   bra      FM_ERASE_END            ;
   nop                              ;
;
;
;>>> Call a module for the bus width of 16 bits
FM_ERASE_WORD:
   bsr      ClearAllStatusOfWord    ;>>> Clear the status of flash memory
   nop                              ;
;
   bsr      FmEraseWord             ;>>> Erase flash memory
   nop                              ; .. The routine has no parameter
;
   mov.l #CheckStatusWord,r3        ;
   jsr      @r3                     ;>>> Check the status of flash memory
   nop                              ; .. A status returns to R0
;
   bsr      ClearAllStatusOfWord    ;
   nop                              ;
;
   bra      FM_ERASE_END            ;
   nop                              ;
;
```

```
;>>> Call a module for the bus width of 32 bits
FM_ERASE_LONG:
   bsr      ClearAllStatusOfLong     ;>>> Clear the status of flash memory
   nop                               ;
;
   bsr      FmEraseLong              ;>>> Erase flash memory
   nop                               ; .. The routine has no parameter
;
   mov.l    #CheckStatusLong,r3
   jsr      @r3                      ;>>> Check the status of flash memory
   nop                               ; .. A status returns to R0
;
   bsr      ClearAllStatusOfLong     ;
   nop                               ;
;
FM_ERASE_END:
;
   mov.l @r15+,r3                    ;
   lds.l @r15+,pr                    ;
   rts                               ;
   nop                               ;
;
;
;
;
;=====================================================================
;
; NAME = FM_TOOL_WRITE;
; FUNC = The routine of writing data in flash memory.
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R4.L  : write address;
;        R5.L  : access size        = 0x4220("B ");
;                                   = 0x5720("W ");
;                                   = 0x4c20("L ");
;        R6.L  : write data;
;        R7.L  : verify flag        = 0 (non verify);
;                                   = 1 (verify);
; OUTP = R0.L  : status              = 0 (OK);
;                                   !=0 (ERROR);
;
;=====================================================================
;; .org     O_FMWrite
   .section fm_write,CODE,LOCATE=O_FMWrite
   .align      4
FM_TOOL_WRITE:
   mov.l    #FM_TOOL_STACK,r15       ; Set stack address to R15 register
   sts.l    pr,@-r15                 ;
   mov.l    r1,@-r15                 ;
   mov.l    r4,@-r15                 ;
   mov.l    r5,@-r15                 ;
   mov.l    r6,@-r15                 ;
   mov.l    r7,@-r15                 ;
;
```

```
    mov     r5,r0                   ;>>> Check an argument
    shlr8   r0                      ;
    cmp/eq  #H'42,r0                ; 'B' ? -> Illegal
    bt      FM_WRITE_BYTE           ;
    cmp/eq  #H'57,r0                ; 'W' ?
    bt      FM_WRITE_WORD           ;
    bf      FM_WRITE_LONG           ; 'L' jump
;
;
;>>> Call a module for the bus width of 8 bits --> Illegal call
FM_WRITE_BYTE:
    bra     FM_WRITE_END            ;
    nop                             ;
;
;
;>>> Call a module for the bus width of 16 bits
FM_WRITE_WORD:
    extu.w  r6,r6                   ;
;
    bsr     ClearAllStatusOfWord    ;>>> Clear the status of flash memory
    nop                             ;
;
    bsr     FmWriteWord             ;>>> Write a data to flash memory
    nop                             ; .. The routine has R4 and R6 (address and
                                    ;data) parameters
;
    bsr     CheckStatusWord         ;>>> Check the status of flash memory
    nop                             ; ... A status returns to R0
;
    cmp/eq  #0,r0                   ; if return == NG
    bf      FM_WRITE_WORD_E         ; then End
;
    cmp/pl  r7                      ; if verify flag is OFF
    bf      FM_WRITE_WORD_E         ; then end
;
    bsr     CheckVerifyWord         ;>>> Call a verify routine
    nop                             ;
;

FM_WRITE_WORD_E:
    bsr     ClearAllStatusOfWord    ;>>> Clear the status of flash memory
    nop                             ;
;
    bra     FM_WRITE_END            ;
    nop                             ;
;
;
```

```
;>>> Call a module for the bus width of 32 bits
FM_WRITE_LONG:
   bsr      ClearAllStatusOfLong     ;>>> Clear the status of flash memory
   nop                               ;
;
   bsr      FmWriteLong              ;>>> Write a data to flash memory
   nop                               ; .. The routine has R4 and R6 (address and
                                     ;data) parameters
;
;  bsr      CheckStatusLong          ;>>> Check the status of flash memory
   nop                               ; ... A status returns to R0
;
   cmp/eq   #0,r0                    ; if return == NG
   bf       FM_WRITE_LONG_E          ; then End
;
   cmp/pl   r7                       ; if verify flag is OFF
   bf       FM_WRITE_LONG_E          ; then end
;
   bsr      CheckVerifyLong          ;>>> Call a verify routine
   nop                               ;
;

FM_WRITE_LONG_E:
   bsr      ClearAllStatusOfLong     ;>>> Clear the status of flash memory
   nop                               ;
;
;
;
FM_WRITE_END:
   mov.l    @r15+,r7                 ;
   mov.l    @r15+,r6                 ;
   mov.l    @r15+,r5                 ;
   mov.l    @r15+,r4                 ;
   mov.l    @r15+,r1                 ;
   lds.l    @r15+,pr                 ;
;
   rts                               ;
   nop                               ;
;
;
;
```

```
;=====================================================================
;
; NAME = FmEraseWord;
; FUNC = The routine of erasing flash memory(Bus width is 16 bits).
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = none;
; OUTP = R0.L  = status;
;
;=====================================================================
FmEraseWord:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
   mov.l    r3,@-r15                ;
   mov.l    r4,@-r15                ;
   mov.l    r8,@-r15                ;
;
   mov.l    #FM_TOP_ADDRESS,r3      ; R5 <- Top address of flash memory
   mov.l    #FM_ERASE_ADDRESS,r4    ; R4 <- Table of flash memory entry address
   mov.l    @r4,r8
   ;
;
FmEraseWord_Main:
   add      r3,r8                   ; R8 <- add flash top address (offset)
   mov.l    #FM_CMD_ERASE,r1        ;>>> Write the Erase command to flash memory
   mov.w    r1,@r8                  ;
;
   mov.l    #FM_CMD_CONFIRM,r1      ;>>> Write the Write confirm command to flash
                                    ;memory
   mov.w    r1,@r8                  ;
;
FmEraseWord_Loop:
   mov.w    @r8,r0                  ; Read status
   mov.l    #FM_CHK_SR7,r1          ;
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status.7 == 0
   bf       FmEraseWord_Loop        ; then loop
;
   mov.w    @r8,r0                  ; Set a return code to R8
   extu.w   r0,r0                   ;
;
   mov.l    #FM_CMD_READ,r1         ;>>> Write the Read command to flash memory
   mov.w    r1,@r8                  ;
;
   add      #4,r4                   ;>>> R8 <- Next entry address
   mov.l    @r4,r8                  ;
   mov      #-1,r1                  ;
   cmp/eq   r1,r8                   ;
   bf       FmEraseWord_Main        ;
;
```

```
FmEraseWord_End:
   mov.l    @r15+,r8                ;
   mov.l    @r15+,r4                ;
   mov.l    @r15+,r3                ;
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;=====================================================================
;
; NAME = FmEraseLong;
; FUNC = The routine of erasing flash memory(Bus width is 32 bits).
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = none;
; OUTP = R0.L  = status;
;
;=====================================================================
FmEraseLong:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
   mov.l    r3,@-r15                ;
   mov.l    r4,@-r15                ;
   mov.l    r8,@-r15                ;
;
   mov.l    #FM_TOP_ADDRESS,r3      ; R5 <- Top address of flash memory
   mov.l    #FM_ERASE_ADDRESS,r4    ; R4 <- Table of flash memory entry address
   mov.l    @r4,r8                  ;
;
FmEraseLong_Main:
   add      r3,r8                   ; R8 <- add flash top address (offset)
   mov.l    #FM_CMD_ERASE,r1        ;>>> Write the Erase command to flash memory
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   mov.l    r1,@r8                  ;
;
   mov.l    #FM_CMD_CONFIRM,r1      ;>>> Write the Write confirm command to flash
                                    ;memory
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   mov.l    r1,@r8                  ;
;
```

```
FmEraseLong_Loop:
   mov.l    @r8,r0                  ; Read status
   mov.l    #FM_CHK_SR7,r1          ;
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status.7 == 0
   bf       FmEraseLong_Loop        ; then loop
;
   mov.l    @r8,r0                  ; Set a return code to R8
;
   mov.l    #FM_CMD_READ,r1         ;>>> Write the Read command to flash memory
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   mov.l    r1,@r8                  ;
;
   add      #4,r4                   ;>>> R8 <- Next entry address
   mov.l    @r4,r8                  ;
   mov      #-1,r1                  ;
   cmp/eq   r1,r8                   ;
   bf       FmEraseLong_Main        ;
;
FmEraseLong_End:
   mov.l    @r15+,r8                ;
   mov.l    @r15+,r4                ;
   mov.l    @r15+,r3                ;
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = FmWriteWord
; FUNC = It is the routine of writing data to flash memory
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R4.L  = write address;
;        R6.L  = write data;
; OUTP = R0.L  = status;
;
;=====================================================================
FmWriteWord:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
;
   mov.l    #FM_CMD_WRITE,r1        ;>>> Write the Write command to flash memory
   mov.w    r1,@r4                  ;
;
   mov.w    r6,@r4                  ; Write data to flash memory
;
FmWriteWord_Check:
   mov.w    @r4,r0                  ;>>> Check status of flash memory
   extu.w   r0,r0                   ;
   mov.l    #FM_CHK_SR7,r1          ;
;
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status & 0x8080 != 0x8080
   bf       FmWriteWord_Check       ; then loop
;
   mov.w    @r4,r0                  ; Set a return code to R4
   extu.w   r0,r0                   ;
;
   mov.l    #FM_CMD_READ,r1         ;>>> Write the Read command to flash memory
   mov.w    r1,@r4                  ;
;
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;======================================================================
;
; NAME = CheckStatusWord
; FUNC = It is the routine of checking the status of flash memory
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R0.L  = check status value;
; OUTP = R0.L  = return code        = 0 (OK);
;                                   !=0 (NG);
;
;======================================================================
CheckStatusWord:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
;
   mov.l    #FM_CHK_SR3,r1          ;>>> Check SR.3, SR.5, and SR.4 of status
                                    ;register
   mov.l    #FM_CHK_SR5,r2          ;
   or       r2,r1                   ;
   mov.l    #FM_CHK_SR4,r2          ;
   or       r2,r1                   ;
;
   extu.w   r0,r0                   ;
   and      r1,r0                   ;
   cmp/eq   #0,r0                   ; if SR != 0
   bf       CSW_Error               ; then error
;
   bra      CSW_END                 ;
   mov      #0,r0                   ; Set the OK code
CSW_Error:
   mov      #-1,r0                  ; Set an error code
CSW_END:
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = CheckVerifyWord
; FUNC = It is the routine of checking the written data
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R4.L  = check data address;
;        R6.L  = write data
;
;=====================================================================
CheckVerifyWord:
   mov.l    r1,@-r15                ;
;
   extu.w   r6,r6                   ; Verify check
   mov.w    @r4,r1                  ;
;
   extu.w   r1,r1                   ;
;
   cmp/eq   r6,r1                   ; if read == write data
   bt       CheckVeriW_OK           ; then OK
;
   mov.w    #FM_BT,r0               ; Set an error code
   bra      CheckVeriW_End          ;
   nop                              ;
;
CheckVeriW_OK:
   mov      #0,r0                   ; Set the OK code
CheckVeriW_End:
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = FmWriteLong
; FUNC = It is the routine of writing data to flash memory
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R4.L  = write address;
;        R6.L  = write data;
; OUTP = R0.L  = status;
;
;=====================================================================
FmWriteLong:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
;
   mov.l    #FM_CMD_WRITE,r1        ;>>> Write the Write command to flash memory
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   mov.l    r1,@r4                  ;
;
   mov.l    r6,@r4                  ; Write data to flash memory
;
FmWriteLong_Check:
   mov.l    @r4,r0                  ;>>> Check status of flash memory
   mov.l    #FM_CHK_SR7,r1          ;
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status & 0x80808080 != 0x80808080
   bf       FmWriteLong_Check       ; then loop
;
   mov.l    @r4,r0                  ; Set a return code to R4
;
   mov.l    #FM_CMD_READ,r1         ;>>> Write the Read command to flash memory
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status
   mov.l    r1,@r4                  ;
;
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = CheckStatusLong
; FUNC = It is the routine of checking the status of flash memory
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R0.L  = check status value;
; OUTP = R0.L  = return code        = 0 (OK);
;                                   !=0 (NG);
;
;=====================================================================
CheckStatusLong:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
;
   mov.l    #FM_CHK_SR3,r1          ;>>> Check SR.3, SR.5, and SR.4 of status
                                    ;register
   mov.l    #FM_CHK_SR5,r2          ;
   or       r2,r1                   ;
   mov.l    #FM_CHK_SR4,r2          ;
   or       r2,r1                   ;
   mov      r1,r2                   ;
   shll16   r2                      ;
   or       r2,r1                   ; R1 <- 32-bit status

   and      r1,r0                   ;
   cmp/eq   #0,r0                   ; if SR != 0
   bf       CSL_Error               ; then error
;
   bra      CSL_END                 ;
   mov      #0,r0                   ; Set the OK code
CSL_Error:
   mov      #-1,r0                  ; Set an error code
CSL_END:
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = CheckVerifyLong
; FUNC = It is the routine of checking the written data
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R4.L  = check data address;
;        R6.L  = write data
;
;=====================================================================
CheckVerifyLong:
   mov.l    r1,@-r15                ;
;
   mov.l    @r4,r1                  ; Verify check
   cmp/eq   r6,r1                   ; if read == write data
   bt       CheckVeriL_OK           ; then OK
;
   mov.w    #FM_BT,r0               ; Set an error code
   bra      CheckVeriL_End          ;
   nop                              ;
;
CheckVeriL_OK:
   mov      #0,r0                   ; Set the OK code
CheckVeriL_End:
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
;=====================================================================
;
; NAME = ClearStatus
; FUNC = It is the routine which clears the status of flash memory.
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = R4.L  : access address;
;        R5.L  : access size        = 0x4220("B ");
;                                   = 0x5720("W ");
;                                   = 0x4c20("L ");
;
;=====================================================================
ClearStatus:
   mov.l    r0,@-r15                ;
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
   mov.l    r3,@-r15                ;
;
   mov.l    #FM_TOP_ADDRESS,r3      ; R7 <- Top address of flash memory
   add      r3,r4                   ; R4 <- add flash top address (offset)
   mov.l    #FM_CMD_STSCLR,r1       ; R1 <- Status clear command
;
```

```
    mov      r5,r0                  ;
    shlr8    r0                     ;
    cmp/eq   #H'42,r0               ; if access size == 'B ' -> Illegal
    bt       ClearByte              ; then ClearByte
;
    cmp/eq   #H'57,r0               ; if access size == 'W '
    bt       ClearWord              ; then ClearWord
;
    bf       ClearLong              ; then ClearLong
;
;>>> Write the status clear command to flash memory
ClearByte:                          ;>>> flash memory of 8-bit bus width -> Illegal
    bra      ClearEnd               ;
    nop                             ;
;
ClearWord:
    ;>>> flash memory of 16-bit bus width
    mov.w    r1,@r4                 ;
    bra      ClearEnd               ;
    nop
;
ClearLong:                          ;>>> flash memory of 32-bit bus width
    mov      r1,r2                  ;
    shll16   r2                     ;
    or       r2,r1                  ; R1 <- 32-bit status clear command
    mov.l    r1,@r4
    ;
;
ClearEnd:
    mov.l    @r15+,r3               ;
    mov.l    @r15+,r2               ;
    mov.l    @r15+,r1               ;
    mov.l    @r15+,r0               ;
    rts                             ;
    nop                             ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = ClearAllStatusOfWord
; FUNC = It is the routine which clears all the status of flash memory.
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = none;
;
;=====================================================================
ClearAllStatusOfWord:
   sts.l    pr,@-r15                ;
   mov.l    r0,@-r15                ;
   mov.l    r4,@-r15                ;
   mov.l    r5,@-r15                ;
   mov.l    r8,@-r15                ;
;
   mov.l    #FM_ERASE_ADDRESS,r8    ;>>> Clear all the status register
;
ClearAllStatusW_LOOP:
   mov.l    @r8,r4                  ;
   mov.l    #TYPE_WORD,r5           ;
   bsr      ClearStatus             ;
   nop                              ;
;
   add      #4,r8                   ;>>> Check finished
   mov.l    @r8,r4                  ;
   mov      #-1,r0                  ;
   cmp/eq   r0,r4                   ;
   bf       ClearAllStatusW_LOOP    ;
;
   mov.l    @r15+,r8                ;
   mov.l    @r15+,r5                ;
   mov.l    @r15+,r4                ;
   mov.l    @r15+,r0                ;
   lds.l    @r15+,pr                ;
   rts                              ;
   nop                              ;
;
;
;
;=====================================================================
;
; NAME = ClearAllStatusOfLong
; FUNC = It is the routine which clears all the status of flash memory.
; NOTE = NEW;
; HIST = 2004.09.01
; INPU = none;
;
;=====================================================================
ClearAllStatusOfLong:
   sts.l    pr,@-r15                ;
   mov.l    r0,@-r15                ;
   mov.l    r4,@-r15                ;
   mov.l    r5,@-r15                ;
   mov.l    r8,@-r15                ;
;
```

```
   mov.l    #FM_ERASE_ADDRESS,r8     ;
;
ClearAllStatusL_LOOP:
   mov.l    @r8,r4                   ;>>> Clear all the status register
   mov.l    #TYPE_LONG,r5            ;
   bsr      ClearStatus              ;
   nop
;
   add      #4,r8                    ;>>> Check finished
   mov.l    @r8,r4                   ;
   mov      #-1,r0                   ;
   cmp/eq   r0,r4                    ;
   bf       ClearAllStatusL_LOOP     ;
;
   mov.l    @r15+,r8                 ;
   mov.l    @r15+,r5                 ;
   mov.l    @r15+,r4                 ;
   mov.l    @r15+,r0                 ;
   lds.l    @r15+,pr                 ;
   rts                               ;
   nop                               ;
;
;
;
;+=========================================================================+
;|      FM DATA TABLE                                                      |
;+=========================================================================+
;  .align   4
;
FM_ERASE_ADDRESS:
   .data.l  H'00000000, H'00080000, H'00100000, H'00180000
   .data.l  H'00200000, H'00280000, H'00300000, H'00380000
   .data.l  H'00400000, H'00480000, H'00500000, H'00580000
   .data.l  H'00600000, H'00680000, H'00700000, H'00780000
   .data.l  H'FFFFFFFF
;
;
;
;
;
   .end
```

CUI コマンド方式のみ

各セクタブロックの先頭アドレス

フラッシュメモリの各セクタブロックの先頭アドレスを列挙します。

最後の H’FFFFFFFF は変更しないでください。

⑤

(b) FUJITSU 製フラッシュメモリ

ファイル名：FMTOOL.src

```
;+=========================================================================+
;|                                                                         |
;| Flash memory tool program sample                                        |
;|    SuperH Family Flash memory load is supported                         |
;|    Copyright (C) 2004 Renesas Technology Corp. All rights reserved.     |
;|    Licensed Material of Renesas Technology Corp.                        |
;|                                                                         |
;|    Erasing flash memory routine top address : O_FMErase                 |
;|    Writing flash memory routine top address : O_FMWrite                 |
;|    Stack pointer address : FM_TOOL_STACK                                |
;|    Flash memory top address : FM_TOP_ADDRESS                            |
;|                                                                         |
;| Target flash memory : FUJITSU [Word mode]                               |
;|                                                                         |
;+=========================================================================+
;
;+=========================================================================+
;|      EQU                                                                |
;+=========================================================
;
O_FMErase       .equ  H'2c001000
O_FMWrite       .equ  H'2c001100
FM_TOOL_STACK   .equ  H'2c002000
FM_TOP_ADDRESS  .equ  H'20000000
;
FM_CMD_RESET    .equ  H'000000F0
FM_CHK_DQ7      .equ  H'00000080
FM_CHK_DQ5      .equ  H'00000020
;
FM_OK           .equ  H'0000
FM_BT           .equ  H'4254
;
TYPE_BYTE       .equ  H'4220
TYPE_WORD       .equ  H'5720
TYPE_LONG       .equ  H'4c20
;
;
;
   .align      4
```

① **消去モジュールの先頭アドレス**

ユーザシステムの RAM 領域を使用します。サイズは H’100 バイト以上確保してください。

② **ライトモジュールの先頭アドレス**

ユーザシステムの RAM 領域を使用します。サイズは H’800 バイト以上確保してください。

③ **ライト/消去モジュール用スタックエリア先頭アドレス**

ユーザシステムの RAM 領域を使用します。サイズは H’100 バイト以上確保してください。

④ **ユーザシステムのフラッシュメモリの先頭アドレス**

ユーザシステムのフラッシュメモリ領域の先頭アドレスを設定します。

```
;=====================================================================
;
; NAME = FM_TOOL_ERASE;
; FUNC = The routine of erasing all flash memories.
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  : access size = 0x4220("B ");
;                            = 0x5720("W ");
;                            = 0x4C20("L ");
; OUTP = R0.L  : status      = 0 (OK);
;                            !=0 (ERROR);
;
;=====================================================================
;  .org     O_FMErase
   .section fm_erase,CODE,LOCATE=O_FMErase
;
FM_TOOL_ERASE:
   mov.l    #FM_TOOL_STACK,r15      ; Set stack address to R15 register
   sts.l    pr,@-r15                ;
   mov.l    r3,@-r15                ;
;
   mov      r4,r0                   ; Check an argument
   shlr8    r0                      ;
   cmp/eq   #H'42,r0                ; 'B' ? -> Illegal
   bt       FM_ERASE_BYTE           ;
   cmp/eq   #H'57,r0                ; 'W' ?
   bt       FM_ERASE_WORD           ;
   bf       FM_ERASE_LONG           ; 'L' jump
;
;
;>>> Call a module for the bus width of 8 bits  --> Illegal call
FM_ERASE_BYTE:
;
   bra      FM_ERASE_END            ;
   nop                              ;
;
;
;>>> Call a module for the bus width of 16 bits
FM_ERASE_WORD:
;
   bsr      ClearAllStatusWord      ; Clear the status of flash memory
   nop                              ;
;
   bsr      FmEraseWord             ; Erase flash memory
   nop                              ; .. The routine has no parameter
;
   bsr      ClearAllStatusWord      ; Clear the status of flash memory
   nop                              ;
;
   bra      FM_ERASE_END            ;
   nop                              ;
;
;
```

```
;>>> Call a module for the bus width of 32 bits
FM_ERASE_LONG:
;
   bsr      ClearAllStatusLong      ; Clear the status of flash memory
   nop                              ;
;
   bsr      FmEraseLong             ; Erase flash memory
   nop                              ; .. The routine has no parameter
;
   bsr      ClearAllStatusLong      ; Clear the status of flash memory
   nop                              ;
;
FM_ERASE_END:
   mov.l    @r15+,r3                ;
   lds.l    @r15+,pr                ;
   rts                              ;
   nop                              ;
;
;
;=====================================================================
;
; NAME = FM_TOOL_WRITE;
; FUNC = The routine of writing data in flash memory.
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  : write address;
;        R5.L  : access size        = 0x4220("B ");
;                                   = 0x5720("W ");
;                                   = 0x4c20("L ");
;        R6.L  : write data;
;        R7.L  : verify flag        = 0 (non verify);
;                                   = 1 (verify);
; OUTP = R0.L  : status             = 0 (OK);
;                                   !=0 (ERROR);
;
;=====================================================================
;  .org  O_FMWrite
   .section fm_write,CODE,LOCATE=O_FMWrite
;
FM_TOOL_WRITE:
   mov.l    #FM_TOOL_STACK,r15      ; Set stack address to R15 register
   sts.l    pr,@-r15                ;
   mov.l    r1,@-r15                ;
   mov.l    r4,@-r15                ;
   mov.l    r5,@-r15                ;
   mov.l    r6,@-r15                ;
   mov.l    r7,@-r15                ;
;
```

```
    mov      r5,r0                  ; Check an argument
    shlr8    r0                     ;
    cmp/eq   #H'42,r0               ; 'B' ? Illegal
    bt       FM_WRITE_BYTE          ;
    cmp/eq   #H'57,r0               ; 'W' ?
    bt       FM_WRITE_WORD          ;
    bf       FM_WRITE_LONG          ; 'L' jump
;
;
;>>> Call a module for the bus width of 8 bits  --> Illegal call
FM_WRITE_BYTE:
;
    mov.l #1,r0                     ; Error
    bra      FM_WRITE_END           ;
    nop                             ;
;
;
;>>> Call a module for the bus width of 16 bits
FM_WRITE_WORD:
    bsr      ClearAllStatusWord     ; Clear the status of flash memory
    nop                             ;
;
    extu.w   r6,r6                  ;
;
    bsr      FmWriteWord            ; Write a data to flash memory
    nop                             ; .. The routine has R4 and R6 (address and
                                    ; data) parameters
;
    cmp/eq   #0,r0                  ; if return == NG
    bf       FM_WRITE_END_WORD      ; then End
;
    cmp/pl   r7                     ; if verify flag is OFF
    bf       FM_WRITE_END_WORD      ; then end
;
    bsr      CheckVerifyWord        ; Call a verify routine
    nop                             ;
;
FM_WRITE_END_WORD:
    bsr      ClearAllStatusWord     ; Clear the status of flash memory
    nop                             ;
;
    bra      FM_WRITE_END           ;
    nop                             ;
;
;
;>>> Call a module for the bus width of 32 bits
FM_WRITE_LONG:
    bsr      ClearAllStatusLong     ; Clear the status of flash memory
    nop                             ;
;
    bsr      FmWriteLong            ; Write a data to flash memory
    nop                             ; .. The routine has R4 and R6 (address and
                                    ; data) parameters
;
```

```
    cmp/eq    #0,r0                    ; if return == NG
    bf        FM_WRITE_END_LONG        ; then End
;
    cmp/pl    r7                       ; if verify flag is OFF
    bf        FM_WRITE_END_LONG        ; then end
;
    bsr       CheckVerifyLong          ; Call a verify routine
    nop                                ;
;
FM_WRITE_END_LONG:
    bsr       ClearAllStatusLong       ; Clear the status of flash memory
    nop                                ;
;
    bra       FM_WRITE_END             ;
    nop                                ;
;
FM_WRITE_END:

    mov.l     @r15+,r7                 ;
    mov.l     @r15+,r6                 ;
    mov.l     @r15+,r5                 ;
    mov.l     @r15+,r4                 ;
    mov.l     @r15+,r1                 ;
    lds.l     @r15+,pr                 ;
    rts                                ;
    nop                                ;
;
;
;
;=====================================================================
;
; NAME = FmEraseWord;
; FUNC = The routine of erasing flash memory(Bus width is 16 bits).
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = none;
; OUTP = R0.L  = status;
;
;=====================================================================
;
FmEraseWord:
    mov.l     r1,@-r15                 ;
    mov.l     r2,@-r15                 ;
    mov.l     r4,@-r15                 ;
    mov.l     r5,@-r15                 ;
    mov.l     r8,@-r15                 ;
;
    mov.l     #FM_TOP_ADDRESS,r5       ; R5 <- Top address of flash memory
    mov.l     #FM_ERASE_DATA,r4        ; R4 <- Table of flash memory command data
    mov.l     #H'FFFFFFFF,r0           ; R0 <- Table end code
;
```

```
FmEraseWord_Main:
   mov.l    @r4+,r8                 ; R8 <- command address
   add      r5,r8                   ; R8 <- add flash top address (offset)
   mov.l    @r4+,r1                 ; R1 <- command data
   cmp/eq   r0,r1                   ; if Table end code
   bt       FmEraseWord_Loop        ; then next
;
   mov.w    r1,@r8                  ; Write the Erase command to flash memory
   bra      FmEraseWord_Main        ; loop
   nop                              ;
;
FmEraseWord_Loop:
   mov.w    @r8,r0                  ; Read status
   mov      r0,r2                   ;
   mov.l    #FM_CHK_DQ7,r1          ;
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status.DQ7 == 1 (0 = Busy / 1 = End)
   bt       FmEraseWord_Loop_Next   ; then exit
   nop
;
   mov      r2,r0                   ;
   mov.l    #FM_CHK_DQ5,r1          ;
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status.DQ5 == 0 (0 = OK / 1 = Fail)
   bf       FmEraseWord_Loop        ; then loop
;
   bra      FmEraseWord_End         ; error end
   nop                              ;
;
FmEraseWord_Loop_Next:
   mov.l    #0,r0                   ; set OK
;
FmEraseWord_End:
   mov.l    @r15+,r8                ;
   mov.l    @r15+,r5                ;
   mov.l    @r15+,r4                ;
   mov.l    @r15+,r2                ;
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
```

```
;=====================================================================
;
; NAME = FmEraseLong;
; FUNC = The routine of erasing flash memory(Bus width is 32 bits).
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = none;
; OUTP = R0.L  = status;
;
;=====================================================================
;
FmEraseLong:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
   mov.l    r4,@-r15                ;
   mov.l    r5,@-r15                ;
   mov.l    r8,@-r15                ;
;
   mov.l    #FM_TOP_ADDRESS,r5      ; R5 <- Top address of flash memory
   mov.l    #FM_ERASE_DATA,r4       ; R4 <- Table of flash memory command data
   mov.l    #H'FFFFFFFF,r0          ; R0 <- Table end code
;
FmEraseLong_Main:
   mov.l    @r4+,r8                 ; R8 <- command address
   shll     r8                      ; adjust address
   add      r5,r8                   ; R8 <- add flash top address (offset)
   mov.l    @r4+,r1                 ; R1 <- command data
   cmp/eq   r0,r1                   ; if Table end code
   bt       FmEraseLong_Loop        ; then next
   mov      r1,r2                   ;
   shll16   r2                      ; command for upper word
   or       r2,r1                   ; R1 <- 32-bit command
;
   mov.l r1,@r8                     ; Write the Erase command to flash memory
   bra      FmEraseLong_Main        ; loop
   nop                              ;
;
FmEraseLong_Loop:
   mov.l    @r8,r0                  ; Read status
   mov      r0,r2                   ;
   mov.l    #FM_CHK_DQ7,r1          ;
   shll16   r1                      ;
   mov.l    #FM_CHK_DQ7,r5          ;
   or       r5,r1                   ; R1 <- 32-bit status
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status.DQ7 == 1 (0 = Busy / 1 = End)
   bt       FmEraseLong_Loop_Next   ; then exit
;
   mov      r2,r0
   and      r5,r0                   ;
   cmp/eq   #H'0,r0                 ; if status.DQ7 == 1 (0 = Busy / 1 = End)
   bf       FmEraseLong_Loop_u      ; then next
```

```
    mov      r2,r0                  ;
    mov.l    #FM_CHK_DQ5,r1         ;
    and      r1,r0                  ;
    cmp/eq   r1,r0                  ; if status.DQ5 == 0 (0 = OK / 1 = Fail)
    bf       FmEraseLong_Loop_u     ; then next
;
    bra      FmEraseLong_End        ; error end
    nop                             ;
;
FmEraseLong_Loop_u:
    mov      r2,r0                  ;
    shlr16   r0                     ;
    and      r5,r0                  ;
    cmp/eq   #H'0,r0                ; if status.DQ7 == 1 (0 = Busy / 1 = End)
    bf       FmEraseLong_Loop       ; then loop
;
    mov      r2,r0                  ;
    mov.l    #FM_CHK_DQ5,r1         ;
    shll16   r1                     ;
    and      r1,r0                  ;
    cmp/eq   r1,r0                  ; if status.DQ5 == 0 (0 = OK / 1 = Fail)
    bf       FmEraseLong_Loop       ; then loop
;
    bra      FmEraseLong_End        ; error end
    nop                             ;
;
FmEraseLong_Loop_Next:
    mov.l    #0,r0                  ; set OK
;
FmEraseLong_End:
    mov.l    @r15+,r8               ;
    mov.l    @r15+,r5               ;
    mov.l    @r15+,r4               ;
    mov.l    @r15+,r2               ;
    mov.l    @r15+,r1               ;
    rts                             ;
    nop                             ;
;
;
;
```

```
;=====================================================================
;
; NAME = FmWriteWord
; FUNC = The routine of writing data to flash memory
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  = write address;
;        R6.L  = write data;
; OUTP = R0.L  = status;
;
;=====================================================================
FmWriteWord:
   mov.l    r1,@-r15                ;
   mov.l    r2,@-r15                ;
   mov.l    r5,@-r15                ;
   mov.l    r6,@-r15                ;
   mov.l    r7,@-r15                ;
   mov.l    r8,@-r15                ;
;
   mov.l    #FM_TOP_ADDRESS,r7      ; R7 <- Top address of flash memory
   mov.l    #FM_WRITE_DATA,r5       ; R5 <- Table of flash memory command data
   mov.l    #H'FFFFFFFF,r0          ; R0 <- Table end code
;
FmWriteWord_Main:
   mov.l    @r5+,r8                 ; R8 <- command address
   add      r7,r8                   ; R8 <- add flash top address (offset)
   mov.l    @r5+,r1                 ; R1 <- command data
   cmp/eq   r0,r1                   ; if Table end code
   bt       FmWriteWord_Write       ; then next
;
   mov.w    r1,@r8                  ; Write the Write command to flash memory
   bra      FmWriteWord_Main        ; loop
   nop                              ;
;
FmWriteWord_Write:
   mov.w    r6,@r4                  ; Write data to flash memory
   mov.l    #FM_CHK_DQ7,r7          ;
   and      r6,r7                   ;
;
FmWriteWord_Loop:
   mov.w    @r4,r0                  ; Read memory
   mov      r0,r2                   ;
   mov.l    #FM_CHK_DQ7,r1          ;
   and      r1,r0                   ;
   cmp/eq   r7,r0                   ; if end the write
   bt       FmWriteWord_Loop_Next   ; then exit
   nop
;
   mov      r2,r0                   ;
   mov.l    #FM_CHK_DQ5,r1          ;
   and      r1,r0                   ;
   cmp/eq   r1,r0                   ; if status.DQ5 == 0 (0 = OK / 1 = Fail)
   bf       FmWriteWord_Loop        ; then loop
;
```

```
    bra      FmWriteWord_End          ; error end
    nop                               ;
;
FmWriteWord_Loop_Next:
    mov.l    #0,r0                    ;
;
FmWriteWord_End:
    mov.l    @r15+,r8                 ;
    mov.l    @r15+,r7                 ;
    mov.l    @r15+,r6                 ;
    mov.l    @r15+,r5                 ;
    mov.l    @r15+,r2                 ;
    mov.l    @r15+,r1                 ;
    rts                               ;
    nop                               ;
;
;
;
;=====================================================================
;
; NAME = FmWriteLong
; FUNC = The routine of writing data to flash memory
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  = write address;
;        R6.L  = write data;
; OUTP = R0.L  = status;
;
;=====================================================================
FmWriteLong:
    mov.l    r1,@-r15                 ;
    mov.l    r2,@-r15                 ;
    mov.l    r5,@-r15                 ;
    mov.l    r6,@-r15                 ;
    mov.l    r7,@-r15                 ;
    mov.l    r8,@-r15                 ;
;
    mov.l    #FM_TOP_ADDRESS,r7       ; R7 <- Top address of flash memory
    mov.l    #FM_WRITE_DATA,r5        ; R5 <- Table of flash memory command data
    mov.l    #H'FFFFFFFF,r0           ; R0 <- Table end code
;
FmWriteLong_Main:
    mov.l    @r5+,r8                  ; R8 <- command address
    shll     r8                       ; adjust address (address signal connecting
                                      ; A2)
    add      r7,r8                    ; R8 <- add flash top address (offset)
    mov.l    @r5+,r1                  ; R1 <- command data
    mov      r1,r2                    ;
    shll16   r2                       ;
    or       r2,r1                    ;
    cmp/eq   r0,r1                    ; if Table end code
    bt       FmWriteLong_Write        ; then next
;
```

```
    mov.l    r1,@r8                 ; Write the Write command to flash memory
    bra      FmWriteLong_Main       ; loop
    nop                             ;
;
FmWriteLong_Write:
    mov.l    r6,@r4                 ; Write data to flash memory
    mov.l    #FM_CHK_DQ7,r7         ;
    shll16   r7                     ;
    mov.l    #FM_CHK_DQ7,r5         ;
    or       r5,r7                  ;
    and      r6,r7                  ;
;
FmWriteLong_Loop:
    mov.l    @r4,r0                 ; Read memory
    mov      r0,r2                  ;
    mov.l    #FM_CHK_DQ7,r1         ;
    shll16   r1                     ;
    mov.l    #FM_CHK_DQ7,r5         ;
    or       r5,r1                  ;
    and      r1,r0                  ;
    cmp/eq   r7,r0                  ; if status.DQ7 == 1 (0 = Busy / 1 = End)
    bt       FmWriteLong_Loop_Next  ; then exit
;
    mov      r7,r1                  ;
    and      r5,r1                  ;
    mov      r2,r0                  ;
    and      r5,r0                  ;
    cmp/eq   r1,r0                  ; if status.DQ7 == write data
    bt       FmWriteLong_Loop_u     ; then next
;
    mov      r2,r0                  ;
    mov.l    #FM_CHK_DQ5,r1         ;
    and      r1,r0                  ;
    cmp/eq   r1,r0                  ; if status.DQ5 == 0 (0 = OK / 1 = Fail)
    bf       FmWriteLong_Loop_u     ; then next
;
    bra      FmWriteLong_End        ; error end
    nop                             ;
;
FmWriteLong_Loop_u:
    shll16   r5                     ;
    mov      r7,r1                  ;
    and      r5,r1                  ;
    mov      r2,r0                  ;
    and      r5,r0                  ;
    cmp/eq   r1,r0                  ; if status.DQ7 == write data
    bt       FmWriteLong_Loop       ; then loop
    nop                             ;
;
    mov      r2,r0                  ;
    mov.l    #FM_CHK_DQ5,r1         ;
    shll16   r1                     ;
    and      r1,r0                  ;
    cmp/eq   r1,r0                  ; if status.DQ5 == 0 (0 = OK / 1 = Fail)
    bf       FmWriteLong_Loop       ; then loop
;
```

```
    bra      FmWriteLong_End          ; error end
    nop                               ;
;
FmWriteLong_Loop_Next:
    mov.l    #0,r0                    ;
;
FmWriteLong_End:
    mov.l    @r15+,r8                 ;
    mov.l    @r15+,r7                 ;
    mov.l    @r15+,r6                 ;
    mov.l    @r15+,r5                 ;
    mov.l    @r15+,r2                 ;
    mov.l    @r15+,r1                 ;
    rts                               ;
    nop                               ;
;
;
;
;=====================================================================
;
; NAME = CheckVerifyWord
; FUNC = The routine of checking the written data
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  = check data address;
;        R6.L  = write data
;
;=====================================================================
CheckVerifyWord:
    mov.l    r1,@-r15                 ;
;
    extu.w   r6,r6                    ; Verify check
    mov.w    @r4,r1                   ;
    extu.w   r1,r1                    ;
    cmp/eq   r6,r1                    ; if read == write data
    bt       CheckVeriW_OK            ; then OK
;
    mov.w    #FM_BT,r0                ; Set an error code
    bra      CheckVeriW_End           ;
    nop                               ;
;
CheckVeriW_OK:
    mov      #0,r0                    ; Set the OK code
CheckVeriW_End:
    mov.l    @r15+,r1                 ;
    rts                               ;
    nop                               ;
;
;
;
;
```

```
;======================================================================
;
; NAME = CheckVerifyLong
; FUNC = The routine of checking the written data
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = R4.L  = check data address;
;        R6.L  = write data
;
;======================================================================
CheckVerifyLong:
   mov.l    r1,@-r15                ;
;
   mov.l    @r4,r1                  ; Verify check
   cmp/eq   r6,r1                   ; if read == write data
   bt       CheckVeriL_OK           ; then OK
   mov.w    #FM_BT,r0               ; Set an error code
   bra      CheckVeriL_End          ;
   nop                              ;
;
CheckVeriL_OK:
   mov      #0,r0                   ; Set the OK code
CheckVeriL_End:
   mov.l    @r15+,r1                ;
   rts                              ;
   nop                              ;
;
;
;
;
;======================================================================
;
; NAME = ClearAllStatusWord
; FUNC = The routine which clears the status of flash memory.
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = none;
;
;======================================================================
ClearAllStatusWord:
   mov.l    r0,@-r15                ;
   mov.l    r1,@-r15                ;
;
   mov.l    #FM_CMD_RESET,r1        ; R1 <- Status clear command
   mov.l    #FM_TOP_ADDRESS,r0      ; R0 <- Flash memory top address
   mov.w    r1,@r0                  ;
   nop                              ;
   nop                              ;
   nop                              ;
   nop                              ;
   nop                              ;
   nop                              ;
;
```

```
   mov.l    @r15+,r1                ;
   mov.l    @r15+,r0                ;
   rts                              ;
   nop                              ;
;
;
;
;=====================================================================
;
; NAME = ClearAllStatusLong
; FUNC = The routine which clears the status of flash memory.
; NOTE = NEW;
; HIST = 2004.09.01;
; INPU = none;
;
;=====================================================================
ClearAllStatusLong:
   mov.l    r0,@-r15                ;
   mov.l    r1,@-r15                ;
;
   mov.l    #FM_CMD_RESET,r0        ;
   mov.l    #FM_CMD_RESET,r1        ;
   shll16   r1                      ;
   or       r0,r1                   ; R1 <- 32-bit status clear command
   mov.l    #FM_TOP_ADDRESS,r0      ; R0 <- Flash memory top address
   mov.l    r1,@r0                  ;
   nop                              ;
   nop                              ;
   nop                              ;
   nop                              ;
   nop                              ;
   nop                              ;
;
   mov.l @r15+,r1                   ;
   mov.l @r15+,r0                   ;
   rts                              ;
   nop                              ;
;
;
;
;+=====================================================================+
;|      FM DATA TABLE                                                  |
;+=====================================================================+
   .align   4
;
;       [ADDRESS(W)],[DATA(W)]
FM_ERASE_DATA:
   .data.l  H'00000AAA,H'000000AA    ; 1
   .data.l  H'00000554,H'00000055    ; 2
   .data.l  H'00000AAA,H'00000080    ; 3
   .data.l  H'00000AAA,H'000000AA    ; 4
   .data.l  H'00000554,H'00000055    ; 5
   .data.l  H'00000AAA,H'00000010    ; 6
   .data.l  H'00000000,H'FFFFFFFF    ; .. END ID.
;
;
```

```
;       [ADDRESS(W)],[DATA(W)]
FM_WRITE_DATA:
   .data.l  H'00000AAA,H'000000AA    ; 1
   .data.l  H'00000554,H'00000055    ; 2
   .data.l  H'00000AAA,H'000000A0    ; 3
   .data.l  H'00000000,H'FFFFFFFF    ; .. END ID.
;
;
   .end
```

安全設計に関するお願い

1. 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

1. 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ルネサス テクノロジ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。
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、ルネサス テクノロジは責任を負いません。
3. 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、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。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、事前にルネサス テクノロジ、ルネサス販売また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。
4. 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、ルネサス テクノロジはその責任を負いません。
5. 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。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 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、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 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ルネサス テクノロジ、ルネサス販売または特約店までご照会ください。