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。

H8S/2378, H8S/2378R グループ

ユーザプログラムモード時のフラッシュ書き込み/消去を効率よくデバッグする方法

要旨

本アプリケーションノートは、H8S/2378R をユーザプログラムモードで動作させて、フラッシュメモリ(0.18μm F-ZTAT 版)の書き込み/消去の各処理の要所要所にエラーチェック手順を埋め込み、エミュレータの画面を使用しながら効率よくデバッグする方法を説明します。

本書の適用範囲は、フラッシュメモリ(0.18μm F-ZTAT 版)内蔵の H8S/2378 グループ、H8S/2378R グループとなっていますが、注意すべき項目ならびに着眼点は他の同類マイコンを使ったデバッグ作業時にも応用が可能です。

目次

1. 仕様

本アプリケーションノートでは、H8S/2378Rをユーザプログラムモードで動作させて、本書で説明するサンプルプログラムを使ってフラッシュメモリ(0.18μm F-ZTAT版)の書き込み/消去を行い、各処理の要所要所に埋め込んだエラーチェック手順をE10A-USBエミュレータの画面を使って確認しながら正常に書き込み/消去が完了するまでを解説します。

フラッシュメモリの書き込み/消去を行なう上で必要な処理として、H8S/2378Rに内蔵されている書き込み/消去プログラムの内蔵RAMへのダウンロード処理、初期化処理、書き込み/消去処理、書き込み終了処理があります。各処理には、エラーチェックのためのパラメータ(DPFR： ダウンロードパス・フェイルリザルトパラメータ、FPFR： フラッシュパス・フェイルパラメータ)があり、そのパラメータの戻り値を参照することで各処理が正常に行なえているか確認できます。

本サンプルプログラムでは、フラッシュメモリの書き込み/消去の各処理後にDPFRもしくはFPFRの戻り値を確認してエラーチェックを行うシーケンスになっています。本書の説明では、E10A-USBエミュレータのブレークコンディション機能を使用して各処理のエラーチェック後にブレークを設定し、ブレーク箇所でのDPFRもしくはFPFRの戻り値を確認してエラーチェックを行います。

また、本サンプルプログラムではプログラム正常終了後、エラー発生後に汎用レジスタに下記パラメータを書き込むシーケンスになっていますので、汎用レジスタによってエラーチェック結果を確認することもできます。

・R0L → 消去プログラムのダウンロード処理エラーチェック結果(DPFR)の値

・R1L → 消去プログラムの初期化・消去処理エラーチェック結果(FPFR)の値

・R2L → 消去プログラムの RTN の値

・R3L → 書き込みプログラムのダウンロード処理エラーチェック結果(DPFR)の値

・R4L → 書き込みプログラムの初期化・消去処理エラーチェック結果(FPFR)の値

・R5L → 書き込みプログラムの RTN の値

RTN とは各処理が行われ、終了すると 1 ずつ加算するパラメータで、どの処理でエラーが発生したか明確にする為に本サンプルプログラムで定義したシンボルです。

【本アプリケーションノート(サンプルプログラム)を使用する上での注意事項】

- **フラッシュメモリの書き込み/消去処理中は高電圧がフラッシュメモリに印加されるため、処理中はフラッシュメモリへのアクセスは禁止となっています。間違って処理中にフラッシュメモリへアクセスすると正常に処理が行われず、フラッシュメモリにダメージを与え破壊する可能性があります。**

  よって、プログラム領域の内容はフラッシュメモリの書き込み/消去を行う前にフラッシュメモリ以外の領域に転送して実行する必要があります。本サンプルプログラムでは、内蔵RAMにプログラム領域の内容を転送して、内蔵RAM上でフラッシュメモリの書き込み/消去を行うプログラムを実行しています。
- **ブレークする箇所は必ず書き込み/消去の各処理後とし、処理中の箇所にブレーク設定を行わないでください。また、書き込み処理と書き込み終了処理は一連の処理となっていますので、書き込み処理と書き込み終了処理の間にブレーク設定を行わないでください。**
- **書き込み/消去の各処理中の箇所でシングルステップ実行を行わないでください。**
- **E10A-USBエミュレータのソフトウェアブレークは使用しないでください。(フラッシュメモリへの書き換えが発生してしまうため)**

2. 動作概要

本章では、フラッシュメモリ書き込み/消去エラーチェックの動作概要を説明します。図 2.1 に動作環境を示します。

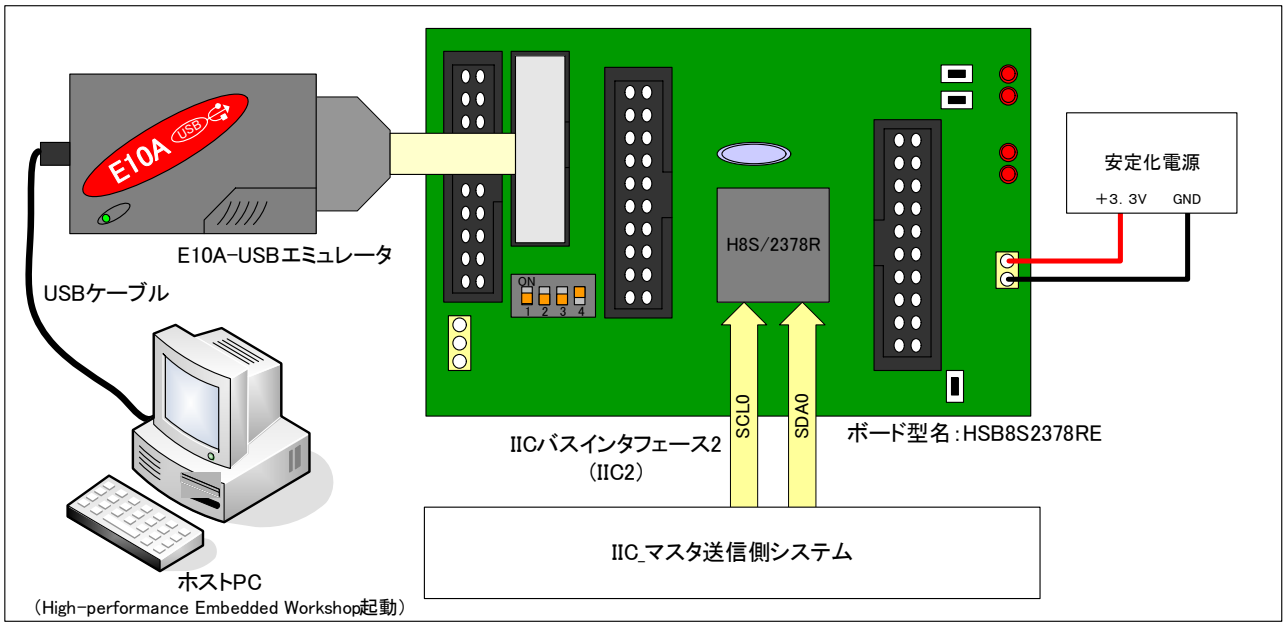


図 2.1　動作環境

① ボード（型名：HSB8S2378RE）上の H-UDI コネクタ（14 ピンタイプ）に E10A-USB エミュレータを接続して、電源コネクタから 3.3V を供給します。

② ホストコンピュータで High-performance Embedded Workshop（HEW）を起動させて、本書で説明するサンプルプログラムをダウンロードして実行します。

③ 最初にサンプルプログラム内で設定したユーザマットのブロック番号に対応するフラッシュメモリのエリアを消去します。

HEW を使って、各処理後（消去プログラムのダウンロード、初期化、消去）の DPFR（ダウンロードパス・フェイルリザルトパラメータ）、FPFR（フラッシュパス・フェイルパラメータ）の戻り値を確認してエラーチェックを行います。

④ フラッシュメモリ書き込みを行うため、書き込みプログラムのダウンロード処理、初期化処理を行います。各処理後、DPFR、FPFR の戻り値を確認してエラーチェックを行います。

⑤ 初期化処理終了後、IIC バスインタフェース 2 のチャネル 0（IIC2_0）を使用して、IIC_マスタ送信側から H8S/2378R（スレーブ受信）側へフラッシュメモリに書き込むデータを送信します。H8S/2378R 側で受信したデータは内蔵 RAM に格納されます。1 回の通信で受信するデータサイズは 4k バイトとなっています。

なお、フラッシュメモリに書き込むデータを送信する（IIC_マスタ送信側）プログラムは、参考例として「付録 IIC_マスタ送信側プログラム」に載せています。

⑥ 受信後、内蔵 RAM に格納されたデータ（4k バイト）をフラッシュメモリに書き込みます。書き込み処理後、FPFR の戻り値を確認してエラーチェックを行います。

⑦ サンプルプログラム内で設定した書き込み回数分、⑤・⑥の処理を繰り返します。最大 508k バイト（※1）のデータをフラッシュメモリに書き込むことができます。

すべての書き込みが終わると、書き込み終了処理を行います。書き込み終了処理後、FPFR の戻り値を確認してエラーチェックを行います。

※1. このサンプルプログラムは、フラッシュメモリの H'000000～H'000FFF 番地をプログラム領域として使用しているため書き込めない仕様となっています。よって、プログラム領域の 4k バイト分を除いた 508k バイトが最大書き込みデータサイズとなります。

3. 開発環境

3.1 ハードウェア開発環境

表 3.1 にハードウェア開発環境の詳細を示します。

表 3.1 ハードウェア開発環境

項目	内容
CPU 種類	H8S/2378R(デバイス型名:HD64F2378R)
メモリ容量	ROM : 512k バイト、RAM : 32k バイト
使用ボード	HSB シリーズ CPU ボード(ボード型名:HSB8S2378RE)
電源電圧	3.3V
動作周波数	外部クロック : 8.25MHz、システムクロック : 33MHz
動作モード	モード 7(シングルチップモード)(※1) MD2 = 1、MD1 = 1、MD0 = 1
エミュレータ	E10A-USB エミュレータ(製品型名:HS0005KCU02H)

※1. ユーザプログラムモードにするため、SYSCR(システムコントロールレジスタ)の FLSHE = 1 にしてください。

3.2 ソフトウェア開発環境

表 3.2 にソフトウェア開発環境の詳細を示します。

表 3.2 ソフトウェア開発環境

項目	内容
開発ツール	High-performance Embedded Workshop Ver. 4.03.00.001
C/C++コンパイラ	H8S,H8/300 Standard Toolchain 6.2.0.0
コンパイラオプション	-cpu=2000A:24 -code=asmcode -object="$(CONFIGDIR)¥$(FILELEAF).src" -debug -nolist -chgincpath -nologo
E10A-USB エミュレータソフトウェア	E10A-USB エミュレータソフトウェア V2.10 Release 01

4. 使用機能説明

4.1　IIC バスインタフェース 2 機能

本書で使用するサンプルプログラムは、フラッシュメモリに書き込むデータを H8S/2378R のオプション機能である IIC バスインタフェース 2 のチャネル 0（IIC2_0）を使用して取得しています。表 4.1 に今回使用した IIC2_0 の機能仕様を示します。

表 4.1 IIC バスインタフェース 2（IIC2）機能仕様

項目	内容
使用シリアルクロック端子	IIC2_0 シリアルクロック入出力端子（SCL0）
使用シリアルデータ端子	IIC2_0 シリアルデータ入出力端子（SDA0）
転送レート	98.2kbps
通信モード	スレーブ受信モード
スレーブアドレス	H’08
受信データサイズ	4k バイト

4.2　フラッシュメモリ書き込み/消去機能

本書で使用するサンプルプログラムは、ユーザプログラムモードでユーザマットの書き込み/消去を行っております。あらかじめマイコン内に内蔵されているプログラムをダウンロードして書き込み/消去を実施します。

なお、書き込み/消去処理中はフラッシュメモリ内部に高電圧が印加されていますので、書き込み/消去処理中にはリセット、ハードウェアスタンバイへの遷移は行わないようにしてください。フラッシュメモリにダメージを与え、破壊する可能性があります。誤ってリセットしてしまった場合は、100μs の通常より長いリセット入力期間のあとにリセットリリースしてください。

4.2.1　フラッシュメモリ書き込み/消去実行時の内蔵 RAM アドレス

ダウンロードの要求、書き込み/消去の手順、結果の判定などのユーザで作成してもらう手続きプログラムの一部は必ず内蔵 RAM 上で実行する必要があります。また、ダウンロードされる内蔵プログラムはすべて内蔵 RAM 上に存在します。これらが重複することのないように、内蔵 RAM 上の領域管理に気を付けてください。図 4.1 にダウンロードされる書き込み/消去プログラムの領域を示します。

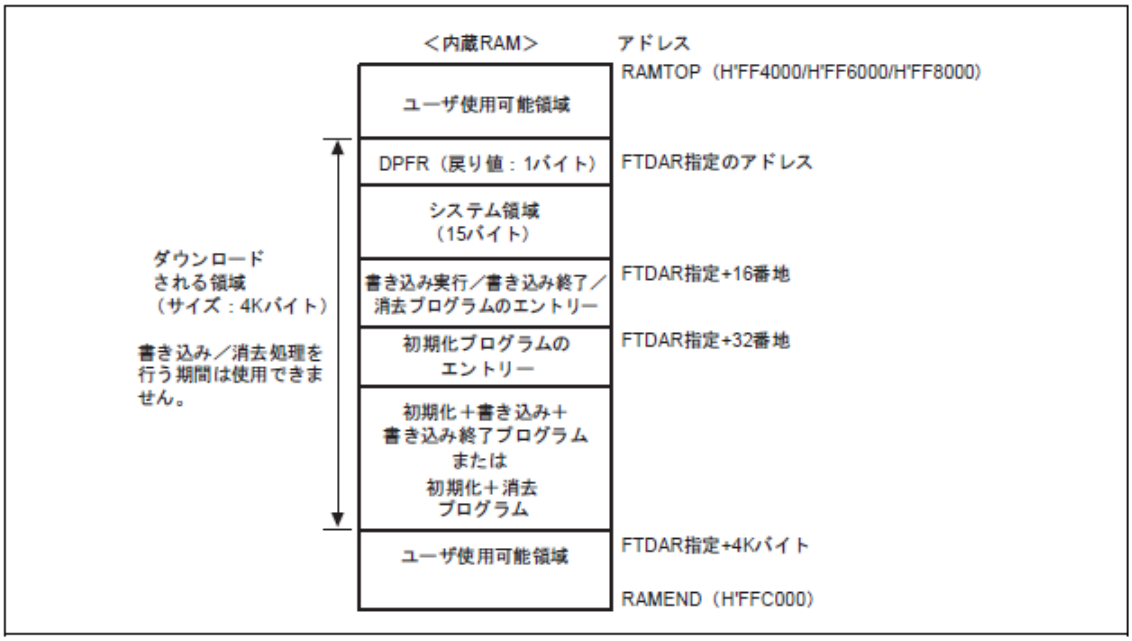


図 4.1　フラッシュメモリ書き込み/消去実行時の内蔵 RAM アドレスマップ

4.2.2 ユーザプログラムモード時のフラッシュメモリ書き込み手順

図 4.2 にフラッシュメモリ書き込み手順を示します。

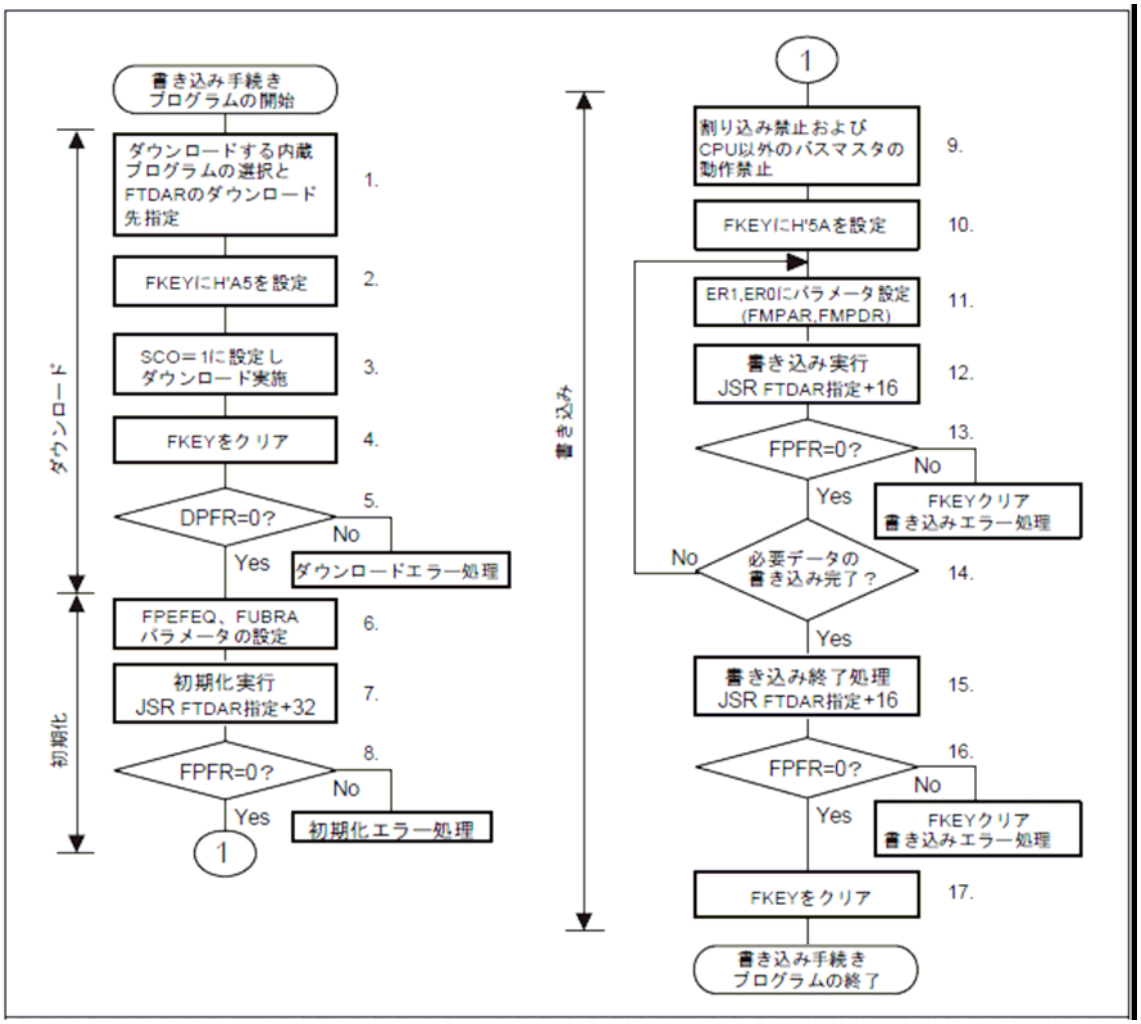


図 4.2 フラッシュメモリ書き込み手順

手続きプログラムは、書き込み対象のフラッシュメモリ以外で実行してください。特に、ダウンロードのために FCCS の SCO ビットを 1 に設定する部分は、必ず内蔵 RAM 上で実行するようにしてください。

1 回の書き込み処理では 128 バイトの書き込みを行います。128 バイトを超える書き込みを行う場合は、書き込み先アドレス/書き込みデータのパラメータを 128 バイト単位で更新して書き込みを繰り返します。

128 バイト未満の書き込みの場合も無効データを埋め込んで 128 バイトにそろえる必要があります。埋め込む無効データを H' FF にすると書き込み処理時間を短縮できます。

4.2.3 ユーザプログラムモード時のフラッシュメモリ消去手順

図 4.3 にフラッシュメモリ消去手順を示します。

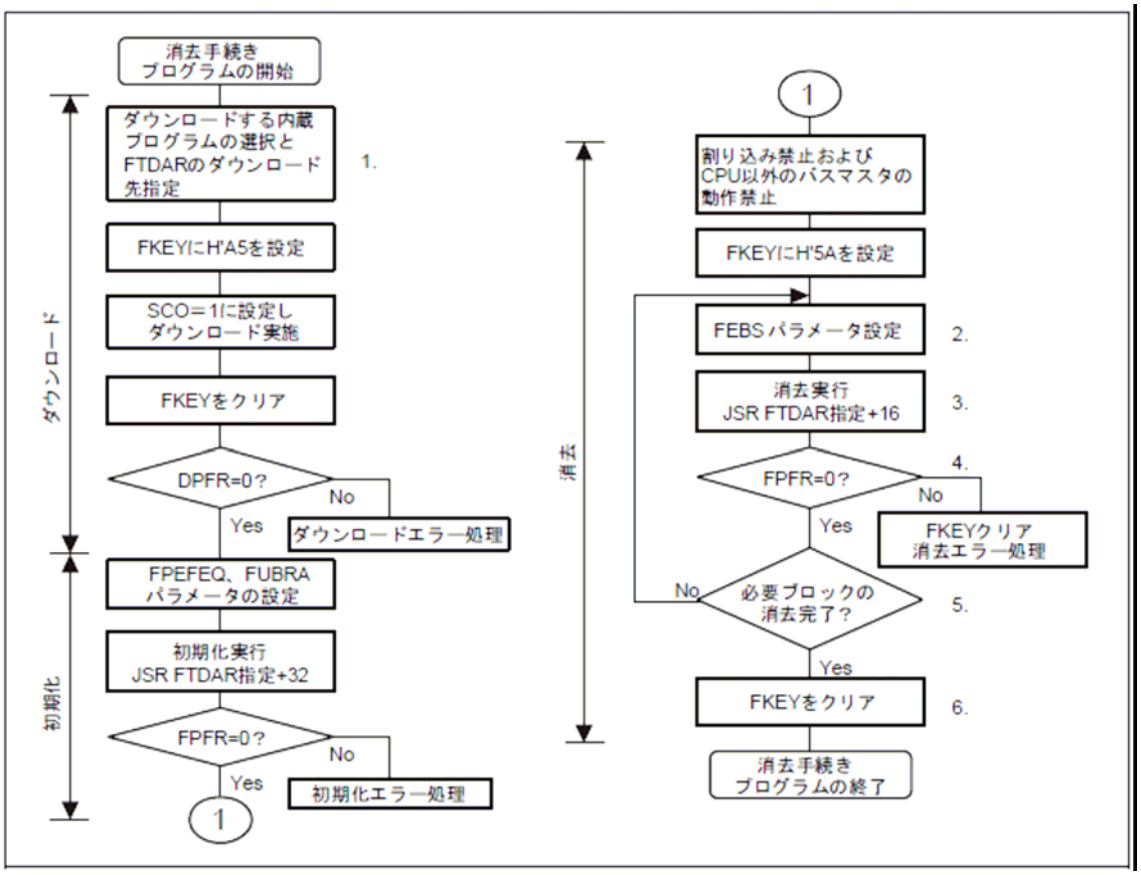


図 4.3 フラッシュメモリ消去手順

手順プログラムは、消去対象のユーザマット以外で実行してください。特に、ダウンロードのために FCCS の SCO ビットを 1 に設定する部分は、必ず内蔵 RAM 上で実行するようにしてください。

1 回の消去処理では 1 分割ブロックの消去を行います。2 ブロック以上の消去を行う場合は、消去ブロック番号を更新して消去を繰り返します。図 4.4 にユーザマットのブロック分割を示します。

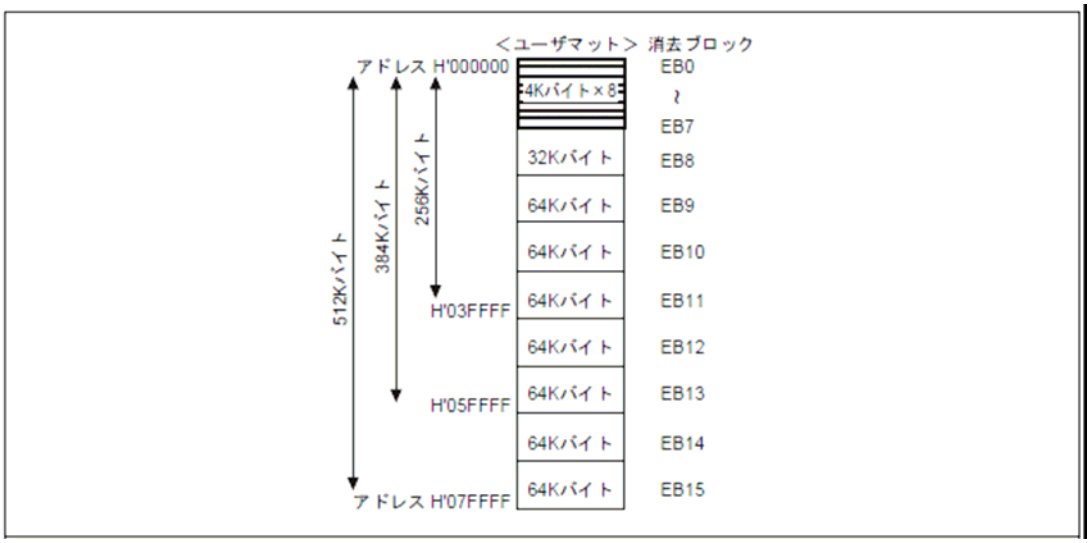


図 4.4 ユーザマットのブロック分割

5. ソフトウェア説明

5.1 関数一覧

本書で使用するサンプルプログラムの関数一覧を表 5.1～表 5.3、階層構造図を図 5.1 に示します。

表 5.1 H8S_2378R_MAIN.c ファイル関数一覧

関数名	内容
main	フラッシュメモリに格納されたサンプルプログラムを内蔵 RAM へ転送する関数（flash_ram_tr）と、フラッシュメモリ書き込み/消去を行う関数をコールする関数(flash_erpr_128)をコールするメイン関数。
init	クロック設定、モジュールストップ解除を行う。
flash_ram_tr	フラッシュメモリに格納されたサンプルプログラムを内蔵 RAM へ転送。
finiprg	フラッシュメモリ書き込み/消去の正常終了時の分岐先。

表 5.2 H8S_FlashPRG.c ファイル関数一覧

関数名	内容
flash_symbol	フラッシュメモリ書き込み/消去で使用するシンボルを定義。 （アセンブリ言語で記述）
flash_erpr_128	フラッシュメモリ書き込み/消去を行う関数(flash_erase、flash_prog)をコール。
flash_erase	フラッシュメモリの消去を行う。 （アセンブリ言語で記述）
flash_prog	IIC バスインタフェース 2 のチャネル 0(IIC2_0)を使用(iic_sr_128 関数をコール)してデータを受信し、受信したデータをフラッシュメモリに書き込む。 （アセンブリ言語で記述）

表 5.3 H8S_iic_sr.c ファイル関数一覧

関数名	内容
iic_sr_128	IIC_マスタ送信側から送信されたデータに対して、スレーブ受信を開始する。
iic_init	IIC バスインタフェース 2 のチャネル 0(IIC2_0)の初期化関数。
iici0_int	IIC バスインタフェース 2 のチャネル 0(IIC2_0)の割り込み関数。
receive_stop_condition	停止条件検出処理関数。
slave_receive	IIC バスインタフェース 2 のチャネル 0(IIC2_0)割り込み処理からコールされるスレーブ受信処理を行う。

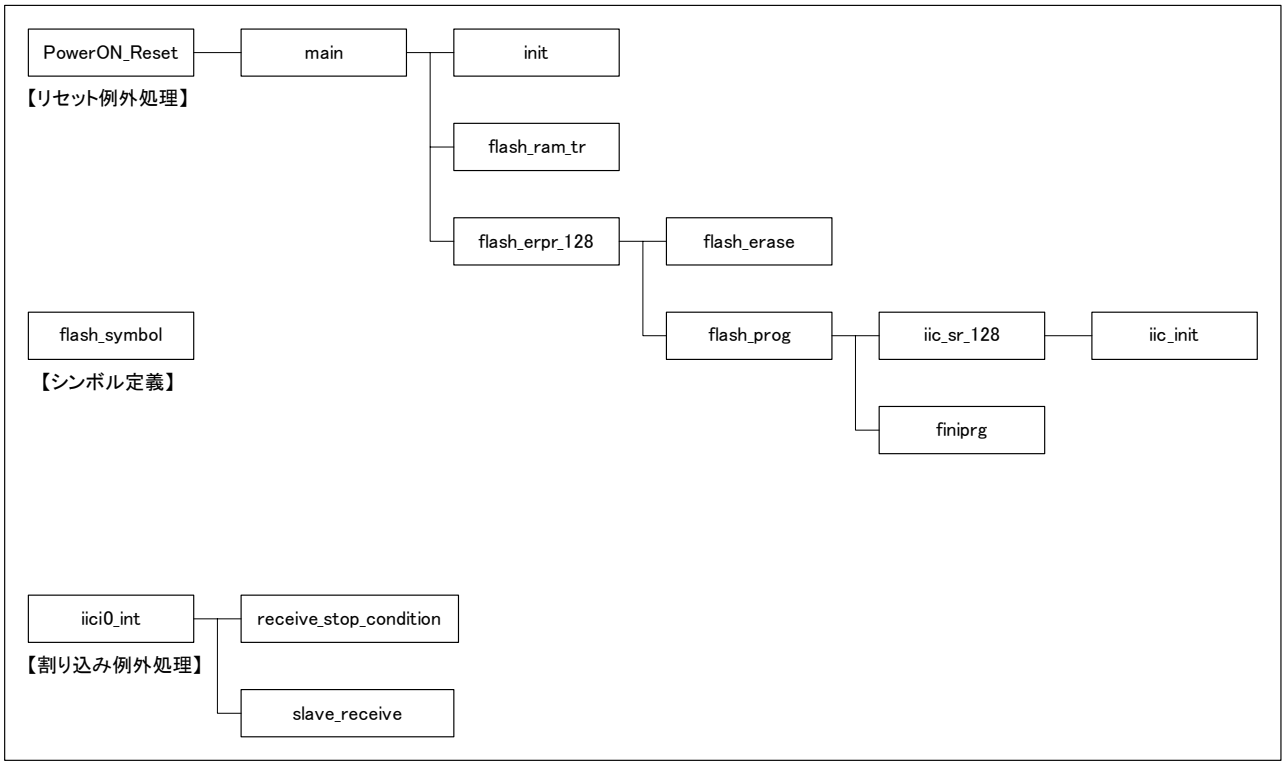


図 5.1　サンプルプログラム階層構造図

5.2　ベクタテーブル

本書で使用するサンプルプログラムで使用している例外処理ベクタテーブルを表 5.4 に示します。

表 5.4　例外処理ベクタテーブル

例外処理要因	ベクタ番号	ベクタテーブルアドレス	割り込み先関数
リセット	0	H’000000	PowerON_Reset
IICI0 割り込み	116	H’0001D0	iici0_int

5.3 セクション

本書で使用するサンプルプログラムのセクション詳細を表 5.5 に示します。

表 5.5 セクション

セクション名	アドレス	内容
PResetPRG	H’000400 ～ H’000415	リセットスタート領域
PIntPRG	H’000416 ～ H’0004C1	割り込み領域
PFiniPRG	H’0004C2 ～ H’0004C3	finiprg 関数領域
P	H’000800 ～ H’0008E3	プログラム領域
PROMPRG	H’0008E4 ～ H’000DE1	内蔵 RAM へ転送するプログラム領域
C$DSEC	H’000DE2 ～ H’000DED	初期化データセクションのアドレス領域
C$BSEC	H’000DEE ～ H’000DF5	未初期化データセクションのアドレス領域
D	H’000DF6 ～ H’000DF9	初期化データ領域
B	H’FF4000 ～ H’FF4427	未初期化データ領域
R	H’FF4428 ～ H’FF442B	セクション“D”の内蔵 RAM へのマップ先領域
PRAMPRG	H’FF5000 ～ H’FF54FD	セクション“PROMPRG”の内蔵 RAM へのマップ先領域
S	H’FFBE00 ～ H’FFBFFF	スタック領域

5.4 マクロ定数

本書で使用するサンプルプログラム内で定義しているマクロ定数を表 5.6、表 5.7 に示します。

表 5.6 マクロ定数（IIC バスインタフェース 2（IIC2）関連）

定数名	設定値	内容	使用関数名
SLAVE_ADDR	H’08	スレーブアドレス	iic_init
MODE_ST	2	サンプルプログラム内での処理状態 スレーブ送信モード	iici0_int
MODE_SR	1	サンプルプログラム内での処理状態 スレーブ受信モード	iici0_int
MODE_IDLE	0	サンプルプログラム内での処理状態 アイドルモード	iic_init、iici0_int、 receive_stop_condition
DTNUM	H’1000	受信データサイズ （1 バイト ＊ H’1000 = 4k バイト）	iic_sr_128

表 5.7 マクロ定数(フラッシュ書き込み/消去関連)

シンボル名	設定値	内容	使用関数名
BLOCK_NUM(※1)	H'00000001	消去開始ブロック番号 設定範囲:H'00000001～H'0000000F	flash_erase
ERASE_N(※2)	H'0F	消去回数 設定範囲:BLOCK_NUM の設定値に依存 (6.3.1 章の表 6.2 をご参照ください)	flash_erase
PR_FL_TOP	H'00001000	フラッシュメモリ書き込み先の先頭アドレス 設定範囲:H'00001000～H'0007F000	flash_prog
PROG_N(※3)	H'01	書き込み回数 測定範囲:PR_FL_TOP の設定値に依存 (6.3.2 章の表 6.3 をご参照ください)	flash_prog
PROG_SIZE	H'20	書き込み回数 1 回あたりのデータサイズ (128 バイト ＊H'20 = 4k バイト)	flash_prog
DATA_RAM_TOP	H'00FF8000	受信データバッファの先頭アドレス	flash_prog
EDL_RAM_TOP	H'00FF9000	消去プログラムダウンロード先の先頭アドレス	flash_erase
PDL_RAM_TOP	H'00FFA000	書き込みプログラムダウンロード先の先頭アドレス	flash_prog
E_DPFR	H'00FFB000	消去時のダウンロードパス・フェイルリザルトパラメータアドレス	flash_erase
E_FPFR	H'00FFB002	消去時のフラッシュパス・フェイルパラメータアドレス	flash_erase
E_RTN	H'00FFB004	消去処理実行結果チェックパラメータアドレス	flash_erase
P_DPFR	H'00FFB006	書き込み時のダウンロードパス・フェイルリザルトパラメータアドレス	flash_prog
P_FPFR	H'00FFB008	書き込み時のフラッシュパス・フェイルパラメータアドレス	flash_prog
P_RTN	H'00FFB00A	書き込み処理実行結果チェックパラメータアドレス	flash_prog

※1. BLOCK_NUM は、フラッシュメモリの消去を開始するブロック番号(EB1～EB15)を指定します。このサンプルプログラムは、EB0 をプログラム領域として使用しているため消去できない仕様となっています。

※2. ERASE_N は、BLOCK_NUM で指定したブロック番号から数えて順番に、ブロック単位で消去を何回行うか指定します。例えば、BLOCK_NUM を“H'00000001”、ERASE_N を“H'0F”と設定すると EB1～EB15 を消去対象ブロックとします。

※3. PROG_N は、4k バイトのデータを繰り返し何回書き込むか指定します。最大で 508kバイトのデータ(設定値“H'7F”)をフラッシュメモリに書き込むことができます。

5.5 H8S_2378R_MAIN.c ファイル内関数の機能説明

本章では、H8S_2378R_MAIN.c ファイル内の関数について機能と使用レジスタを説明します。

5.5.1 init 関数

表 5.8 init 関数の機能説明

機能	戻り値	引数
・分周器の分周比選択 ・PLL 回路の逓倍率設定 ・IIC2_0 を除く全てのモジュールをストップ	なし	なし

【使用レジスタ】

システムクロックコントロールレジスタ（SCKCR） アドレス：H’FFFF3B

ビット	ビット名	初期値	R/W	説明
3	STCS	0	R/W	周波数逓倍率切り替えモード選択 PLL 回路の周波数逓倍率変更時の動作を選択します。 0：変更した逓倍率は、ソフトウェアスタンバイモード遷移後に有効 1：変更した逓倍率は、STC1、STC0 ビット書き換え後に有効
2 1 0	SCK2 SCK1 SCK0	0 0 0	R/W R/W R/W	システムクロックセレクト 2～0 分周比を選択します。 000：1/1 001：1/2 010：1/4 011：1/8 100：設定禁止 101：設定禁止 11X：設定禁止

PLL コントロールレジスタ（PLLCR） アドレス：H’FFFF45

ビット	ビット名	初期値	R/W	説明
1 0	STC1 STC0	0 0	R/W R/W	周波数逓倍率設定 PLL 回路の周波数逓倍率を設定します。 00：×1 01：×2 10：×4 11：設定禁止

モジュールストップコントロールレジスタ H、L（MSTPCRH、MSTPCRL）　　アドレス：H’FFFF40

1 のとき対応するモジュールはモジュールストップモードになり、クリアするとモジュールストップモードは解除されます。

MSTPCRH

ビット	ビット名	初期値	R/W	説明
15	ACSE	0	R/W	全モジュールクロックストップモードイネーブル MSTPCR で制御されるすべての内蔵周辺機能、または TMR 以外の内蔵周辺機能をモジュールストップモードにし、SLEEP 命令実行後の全モジュールクロックストップモードへの遷移を許可または禁止します。 0：全モジュールクロックストップモードを禁止 1：全モジュールクロックストップモードを許可
14	MSTP14	0	R/W	EXDMA コントローラ（EXDMAC）
13	MSTP13	0	R/W	DMA コントローラ（DMAC）
12	MSTP12	0	R/W	データトランスファコントローラ（DTC）
11	MSTP11	1	R/W	16 ビットタイマパルスユニット（TPU）
10	MSTP10	1	R/W	プログラマブルパルスジェネレータ（PPG）
9	MSTP9	1	R/W	D/A 変換器（チャネル 0、1）
8	MSTP8	1	R/W	D/A 変換器（チャネル 2、3）

MSTPCRL

ビット	ビット名	初期値	R/W	説明
7	MSTP7	1	R/W	D/A 変換器（チャネル 4、5）
6	MSTP6	1	R/W	A/D 変換器
5	MSTP5	1	R/W	シリアルコミュニケーションインタフェース 4（SCI_4）
4	MSTP4	1	R/W	シリアルコミュニケーションインタフェース 3（SCI_3）
3	MSTP3	1	R/W	シリアルコミュニケーションインタフェース 2（SCI_2）
2	MSTP2	1	R/W	シリアルコミュニケーションインタフェース 1（SCI_1）
1	MSTP1	1	R/W	シリアルコミュニケーションインタフェース 0（SCI_0）
0	MSTP0	1	R/W	8 ビットタイマ（TMR）

エクステンションモジュールストップコントロールレジスタ H、L（EXMSTPCRH、EXMSTPCRL）　アドレス：H’FFFF42

1 のとき対応するモジュールはモジュールストップモードになり、クリアするとモジュールストップモードは解除されます。

EXMSTPCRH

ビット	ビット名	初期値	R/W	説明
15～12	－	1	R/W	リザーブビット リード/ライト可能ですがライト時は 1 をライトしてください。
11	MSTP27	1	R/W	－
10	MSTP26	1	R/W	－
9	MSTP25	1	R/W	－
8	MSTP24	1	R/W	－

EXMSTPCRL

ビット	ビット名	初期値	R/W	説明
7	MSTP23	1	R/W	－
6	MSTP22	1	R/W	－
5	MSTP21	1	R/W	－
4	MSTP20	1	R/W	IIC バスインタフェース 2_1（IIC2_1）
3	MSTP19	1	R/W	IIC バスインタフェース 2_0（IIC2_0）
2	MSTP18	1	R/W	－
1	MSTP17	0	R/W	－
0	MSTP16	1	R/W	－

5.5.2 main 関数

表 5.9 main 関数の機能説明

機能	戻り値	引数
・RAM 転送関数（flash_ram_tr）のコール ・フラッシュメモリ書き込み/消去関数（flash_erpr_128）のコール	なし	なし

【使用レジスタ】

なし

5.5.3 flash_ram_tr 関数

表 5.10 flash_ram_tr 関数の機能説明

機能	戻り値	引数
・フラッシュメモリに格納されるセクション“PROMPRG”のプログラムを内蔵 RAM 上のセクション“PRAMPRG”に転送	なし	なし

【使用レジスタ】

なし

5.5.4 finiprg 関数

表 5.11 finiprg 関数の機能説明

機能	戻り値	引数
・フラッシュメモリ書き込み/消去が正常に終了した時の戻り（ジャンプ先）関数	なし	なし

【使用レジスタ】

なし

5.6 H8S_FlashPRG.c ファイル内関数の機能説明

本章では、H8S_FlashPRG.c ファイル内の関数について機能と使用レジスタを説明します。

5.6.1 flash_symbol 関数

表 5.12 flash_symbol 関数の機能説明

機能	戻り値	引数
・フラッシュメモリ書き込み/消去で使用するシンボルを定義し、インライン展開している	なし	なし

【使用レジスタ】

なし

5.6.2 flash_erpr_128 関数

表 5.13 flash_erpr_128 関数の機能説明

機能	戻り値	引数
・フラッシュメモリ消去関数（flash_erase）のコール ・フラッシュメモリ書き込み関数（flash_prog）のコール	なし	なし

【使用レジスタ】

なし

5.6.3 flash_erase 関数

表 5.14 flash_erase 関数の機能説明

機能	戻り値	引数
・消去処理で使用するシンボルの設定値チェック ・ユーザプログラムモードへの遷移（FLSHE = 1） ・消去プログラムのダウンロード処理 ・フラッシュメモリの初期化処理 ・フラッシュメモリの消去処理 ・各処理のエラーチェック	なし	なし

【使用レジスタ】

システムコントロールレジスタ（SYSCR）　　アドレス：H’FFFF3D

ビット	ビット名	初期値	R/W	説明
3	FLSHE	0	R/W	フラッシュメモリコントロールレジスタイネーブル フラッシュメモリの制御レジスタの CPU アクセスを制御します。このビットを 1 にセットすると、フラッシュメモリの制御レジスタをリード/ライトすることができます。0 にクリアするとフラッシュメモリの制御レジスタは非選択となります。このとき、フラッシュメモリの制御レジスタの内容は保持されています。フラッシュメモリ版以外は 0 をライトしてください。 0：アドレス H’FFFFC4～H’FFFFCF のエリアはフラッシュメモリのレジスタを非選択 1：アドレス H’FFFFC4～H’FFFFCF のエリアはフラッシュメモリのレジスタを選択

フラッシュイレーズコードセレクトレジスタ（FECS）　　アドレス：H’FFFFC6

ビット	ビット名	初期値	R/W	説明
0	EPVB	0	R/W	イレーズパルスベリファイブロック 消去プログラムを選択します。 0：内蔵消去プログラムを選択しません。 ［クリア条件］ 転送が終了するとクリアされます。 1：内蔵消去プログラムを選択します。

フラッシュトランスファディスティネーションアドレスレジスタ（FTDAR）　　アドレス：H’FFFFCA

ビット	ビット名	初期値	R/W	説明
7	TDER	0	R/W	トランスファディスティネーションアドレス設定エラー TDA6～TDA0 ビットで指定するダウンロード先頭アドレス指定にエラーがあった場合、1 がセットされます。アドレス指定のエラー判定は、FCCS の SCO ビットを 1 にして、ダウンロード処理が実行されたときに、TDA6～TDA0 の値が H’00～H’03 の範囲にあるかどうかを判定します。SCO ビットを 1 に設定する前に、本ビットの値を 0 にすることも含めて、FTDAR の値を H’00～H’03 の範囲に設定してください。 0：TDA6～TDA0 の設定は、正常値です。 1：TDER、TDA6～TDA0 の設定値が H’04～H’FF であり、ダウンロードは中断したことを示します。

フラッシュトランスファディスティネーションアドレスレジスタ（FTDAR）　　アドレス：H’FFFFCA

ビット	ビット名	初期値	R/W	説明
6	TDA6	0	R/W	トランスファディスティネーションアドレス ダウンロード先頭アドレスを指定します。設定可能な値はH’00～H’03で、4Kバイト単位で内蔵RAM 上のダウンロード先頭アドレスを指定できます。 H’00：ダウンロード先頭アドレスをH’FF9000に設定 H’01：ダウンロード先頭アドレスをH’FFA000に設定 H’02：ダウンロード先頭アドレスをH’FFB000に設定 H’03：ダウンロード先頭アドレスをH’FF8000に設定 H’04～H’7F：設定しないでください。この値が設定された場合、ダウンロード処理において、TDER ビットが1になり、内蔵プログラムのダウンロード処理は中断されます。
5	TDA5	0	R/W	
4	TDA4	0	R/W	
3	TDA3	0	R/W	
2	TDA2	0	R/W	
1	TDA1	0	R/W	
0	TDA0	0	R/W	

ダウンロードパス・フェイルリザルトパラメータ（DPFR：FTDAR レジスタで指定した内蔵 RAM の先頭アドレス 1 バイト）

ダウンロード結果の戻り値です。ダウンロードが実行できたかどうかは、本パラメータの値で判断します。

ビット	ビット名	初期値	R/W	説明
7～3	－	－	－	未使用ビット 値 0 が戻されます。
2	SS	－	R/W	ソースセレクトエラー検出ビット ダウンロード可能な内蔵プログラムは1種類のみ指定できます。2種類以上の選択を行った場合、選択されていない場合、およびマッピングされていない選択の場合にはエラーとなります。 0：ダウンロードプログラムの選択関係は正常 1：ダウンロードエラー発生（多重選択または、マッピングされていないプログラム選択）
1	FK	－	R/W	フラッシュキーレジスタエラー検出ビット FKEYの値が、H’A5であるかどうかをチェックした結果を返すビットです。 0：FKEYの設定は正常（FKEY = H’A5） 1：FKEY の設定値エラー（FKEY は、H’A5 以外の値）
0	SF	－	R/W	サクセス／フェイルビット ダウンロードが正常に終了したかどうかを戻すビットです。内蔵RAM上にダウンロードしたプログラムをリードバックし、内蔵RAM上に転送できているかの判定結果です。 0：内蔵プログラムのダウンロードは正常終了（エラーなし） 1：内蔵プログラムのダウンロードが異常終了（エラーが発生している）

フラッシュキーコードレジスタ（FKEY）　　アドレス：H’FFFFC8

ビット	ビット名	初期値	R/W	説明
7	K7	0	R/W	キーコード H’A5を書き込んだ場合にのみ、SCOビットの書き込みが有効になります。H’A5以外の値がFKEYに書かれている場合、SCOビットに1を書き込むことができないため、内蔵RAMへのダウンロードができません。また、H’5Aを書き込んだ場合のみ、書き込み/消去が可能になります。内蔵の書き込み/消去プログラムを実行しても、H’5A以外の値がFKEYレジスタに書かれている場合はフラッシュメモリの書き込み/消去はできません。 H’A5：SCOビットの書き込みを許可します。（H’A5以外ではSCOビットのセットはできません） H’5A：書き込み/消去を許可します。（H’5A以外ではソフトウェアプロテクト状態） H’00：初期値
6	K6	0	R/W	
5	K5	0	R/W	
4	K4	0	R/W	
3	K3	0	R/W	
2	K2	0	R/W	
1	K1	0	R/W	
0	K0	0	R/W	

フラッシュコードコントロール・ステータスレジスタ（FCCS）　　アドレス：H’FFFFC4

ビット	ビット名	初期値	R/W	説明
0	SCO	0	（R）/W	ソースプログラムコピーオペレーション 内蔵書き込み/消去プログラムを内蔵RAMにダウンロードする要求ビットです。本ビットに1を書き込むと、FPCS/FECSレジスタで選択した内蔵プログラムが、FTDARレジスタで指定された内蔵RAMの領域に自動的にダウンロードされます。本ビットに1を書き込むためには、FKEYへのH’A5の書き込み、および内蔵RAM上での実行が必要です。 本ビットに1を書き込んだ直後には、4個のNOP命令を必ず実行するようにしてください。なお、ダウンロード完了時点では本ビットは0クリアされているため、本ビットの1状態を読み出すことはできません。ダウンロード中は、すべての割り込みを禁止する必要があります。ユーザのシステム上で割り込みが入らないようにしてください。 0：内蔵されている書き込み/消去プログラムの内蔵RAMへのダウンロードは行いません ［クリア条件］ ダウンロードが完了するとクリアされます。 1：内蔵されている書き込み/消去プログラムの内蔵RAMへのダウンロードリクエストを発生します。 ［セット条件］ 以下の条件がすべて満足されている状態で、1を書き込んだとき （1）FKEYにH’A5が書かれていること （2）内蔵 RAM 上で実行中であること

フラッシュプログラム/イレーズ周波数パラメータ(FPEFEQ:CPU の汎用レジスタ ER0)

ビット	ビット名	初期値	R/W	説明
31～16	FUBE15～0	−	R/W	フラッシュユーザブランチイネーブルビット ユーザブランチ機能を有効にするときは、H'AA55に設定してください。 それ以外はH'0000に設定してください。
15～0	F15～F0	−	R/W	周波数設定ビット CPUの動作周波数を設定します。設定値は以下のように算出してください。 ・MHz単位で表現した動作周波数を小数点第3位で四捨五入し、小数点第2位までとする。 ・100倍した値を2進数に変換し、FPEFEQパラメータ(汎用レジスタER0)に書き込む。 具体例として、CPUの動作周波数が35.000MHzの場合には、以下のようになります。 ・35.000の小数点第3位を四捨五入し、35.00 ・35.00 × 100 = 3500を2進数変換し、B'0000 1101 1010 1100 (H'0DAC)をR0に設定。

フラッシュユーザブランチアドレス設定パラメータ(FUBRA:CPU の汎用レジスタ ER1)

ビット	ビット名	初期値	R/W	説明
31～0	UA31～UA0	−	R/W	ユーザブランチ先アドレス ユーザブランチ先は、内蔵プログラムが転送されているRAM領域以外のRAM空間または外部バス空間としてください。 実行コードのない領域にブランチして暴走しないように注意し、内蔵プログラムのダウンロード領域やスタック領域を破壊しないようにしてください。暴走やダウンロード領域/スタック領域の破壊が発生した場合、フラッシュメモリの値の保証もできません。 ユーザブランチ先の処理では、内蔵プログラムのダウンロード、初期化、書き込み/消去プログラムを起動しないでください。ユーザブランチ先から復帰時の書き込み/消去の保証ができません。また、すでに準備していた書き込みデータを書き換えないでください。 さらに、ユーザブランチ先の処理で書き込み/消去インタフェースレジスタの書き換えを行わないでください。 ユーザブランチ処理終了後は、RTS命令で書き込み/消去プログラムに戻ってください。

フラッシュパス/フェイルパラメータ（FPFR：CPU の汎用レジスタ R0L）

初期化結果の戻り値です。

ビット	ビット名	初期値	R/W	説明
7～3	－	－	－	未使用ビット 値 0 が戻されます。
2	BR	－	R/W	ユーザブランチエラー検出ビット 指定されたユーザブランチ先アドレスが、ダウンロードされている書き込み/消去関係プログラムの格納領域以外であるかをチェックした結果を戻します。 0：ユーザブランチアドレス設定は正常値 1：ユーザブランチアドレス設定が異常値
1	FQ	－	R/W	周波数エラー検出ビット 指定されたCPU動作周波数が、サポートしている動作周波数の範囲にあるかをチェックした結果を戻します。 0：動作周波数の設定は正常値 1：動作周波数の設定が異常値
0	SF	－	R/W	サクセス/フェイルビット 初期化が正常に終了したかどうかを戻すビットです。 0：初期化は正常終了（エラーなし） 1：初期化が異常終了（エラーが発生している）

フラッシュパス/フェイルパラメータ（FPFR：CPU の汎用レジスタ R0L）

消去処理結果の戻り値です。

ビット	ビット名	初期値	R/W	説明
7	－	－	－	未使用ビット 値 0 が戻されます。
6	MD	－	R/W	消去モード関連設定エラー検出ビット エラープロテクト状態でないことのチェック結果を返します。エラープロテクト状態になっている場合、1が書き込まれます。これらの状態は、FCCSのFLERで確認できます。 0：FLER状態は正常（FLER = 0） 1：FLER = 1 であり、消去できない状態
5	EE	－	R/W	消去実行時エラー検出ビット ユーザマットの消去ができなかったり、ユーザブランチ処理から戻った時点でフラッシュ関連レジスタの一部が書き換えられている場合に、本ビットには1が返されます。これらが原因で、本ビットが1になった場合、ユーザマットは途中まで消去されている可能性が高いため、エラーになる原因を取り除いた後、再度消去を実施し直してください。また、FMATSレジスタの値がH’AAとなっており、ユーザブートマット選択状態のときに消去を実施しても、消去実行時エラーとなります。この場合は、ユーザマット/ユーザブートマットともに、消去されてはいません。ユーザブートマットの消去はブートモードまたはライタモードで実施してください。

フラッシュパス/フェイルパラメータ（FPFR：CPU の汎用レジスタ R0L）

消去処理結果の戻り値です。

ビット	ビット名	初期値	R/W	説明
4	FK	−	R/W	フラッシュキーレジスタエラー検出ビット 消去処理開始前にFKEYの値をチェックした結果を戻します。 0：FKEYの設定は正常（FKEY = H’5A） 1：FKEY の設定値エラー（FKEY は、H’5A 以外の値）
3	EB	−	R/W	イレーズブロックセレクトエラー検出ビット 指定された消去ブロック番号が、ユーザマットのブロック範囲内であるかのチェック結果です。 0：消去ブロック番号の設定は正常値 1：消去ブロック番号の設定が異常値
2 1	− −	− −	− −	未使用ビット 値 0 が戻されます。
0	SF	−	R/W	サクセス/フェイルビット 消去処理が正常に終了したかどうかを戻すビットです。 0：消去は正常終了（エラーなし） 1：消去が異常終了（エラーが発生している）

フラッシュイレーズブロックセレクトパラメータ（FEBS：CPU の汎用レジスタ ER0）

ビット	ビット名	初期値	R/W	説明
31～8	−	−	−	未使用ビット 値 0 を設定してください。
7～0	EBN7～0	−	R/W	イレーズブロック番号 0～15 の範囲で消去ブロック番号を設定します。 EB0 ブロックを選択するときは H’00 を、EB15 ブロックを選択するときは H’0F を設定してください。H’00～H’0F 以外の設定ではエラーになります。

5.6.4 flash_prog 関数

表 5.15 flash_prog 関数の機能説明

機能	戻り値	引数
・書き込み処理で使用するシンボルの設定値チェック ・ユーザプログラムモードへの遷移（FLSHE = 1） ・書き込みプログラムのダウンロード処理 ・フラッシュメモリの初期化処理 ・IIC2_0 によるスレーブ受信処理 ・フラッシュメモリの書き込み処理 ・フラッシュメモリに書き込んだデータのコンペアチェック ・フラッシュメモリの書き込み終了処理 ・各処理のエラーチェック	なし	なし

【使用レジスタ】（flash_erase 関数内でも使用している同じレジスタは省略しています）

フラッシュプログラムコードセレクトレジスタ（FPCS） アドレス：H’FFFFC5

ビット	ビット名	初期値	R/W	説明
0	PPVS	0	R/W	プログラムパルスベリファイ 書き込みプログラムを選択します。 0：内蔵の書き込みプログラムを選択しません。 ［クリア条件］ 転送が終了するとクリアされます。 1：内蔵の書き込みプログラムを選択します。

フラッシュマルチパーパスアドレスエリアパラメータ（FMPAR：CPU の汎用レジスタ ER1）

ビット	ビット名	初期値	R/W	説明
31～0	MOA31 ～ MOA0	－	R/W	ユーザマット上の書き込み先の先頭アドレスを格納します。ここで指定されたユーザマットの先頭アドレスから連続128バイトの書き込みが行われます。 よって、指定する書き込み先の先頭アドレスは128バイト境界となり、MOA6～MOA0は常に0 になります。

フラッシュマルチパーパスデータデスティネーションパラメータ（FMPDR：CPU の汎用レジスタ ER0）

ビット	ビット名	初期値	R/W	説明
31～0	MOD31 ～ MOD0	－	R/W	ユーザマットへの書き込みデータが格納されている領域の先頭アドレスを格納します。ここで指定された先頭アドレスから連続128バイトのデータが、ユーザマットに対して書き込まれます。

フラッシュパス/フェイルパラメータ（FPFR：CPU の汎用レジスタ R0L）

書き込み処理結果の戻り値です。

ビット	ビット名	初期値	R/W	説明
7	−	−	−	未使用ビット 値 0 が戻されます。
6	MD	−	R/W	書き込みモード関連設定エラー検出ビット エラープロテクト状態でないことのチェック結果を返します。エラープロテクト状態になっている場合、1が書き込まれます。これらの状態は、FCCSのFLERビットで確認できます。 0：FLER状態は正常（FLER = 0） 1：FLER = 1 であり、書き込みできない状態
5	EE	−	R/W	書き込み実行時エラー検出ビット ユーザマットが消去されていないために、指定データを書き込めなかったり、ユーザブランチ処理から戻った時点でフラッシュ関連レジスタの一部が書き換えられている場合に、本ビットには1が返されます。これらが原因で、本ビットが1になった場合、ユーザマットは途中まで書き換えられている可能性が高いため、エラーになる原因を取り除いた後、消去から実施し直してくだい。また、FMATS レジスタの値がH ‘AAとなっており、ユーザブートマット選択状態のときに書き込みを実施しても、書き込み実行時エラーとなります。この場合は、ユーザマット/ユーザブートマットともに、書き換えられてはいません。ユーザブートマットの書き込みはブートモードまたはライタモードで実施してください。 0：書き込み処理は正常終了 1：書き込み処理が異常終了し、書き込み結果は保証できない
4	FK	−	R/W	フラッシュキーレジスタエラー検出ビット 書き込み処理開始前にFKEY の値をチェックした結果を戻します。 0：FKEYの設定は正常（FKEY = H’5A） 1：FKEY の設定値エラー（FKEY は、H’5A 以外の値）
3	−	−	R/W	未使用ビット 値 0 が戻されます。
2	WD	−	R/W	ライトデータアドレス検出ビット 書き込みデータの格納先の先頭アドレスとして、フラッシュメモリ領域のアドレスが指定された場合にはエラーとなります。 0：書き込みデータアドレス設定は正常値 1：書き込みデータアドレス設定が異常値
1	WA	−	R/W	ライトアドレスエラー検出ビット 書き込み先先頭アドレスとして、以下が指定された場合にはエラーとなります。 ・フラッシュメモリの領域外が書き込み先アドレスとして指定された場合 ・指定されたアドレスが128バイト境界でない（A6～A0が0でない）場合 0：書き込み先アドレス設定は正常値 1：書き込み先アドレス設定が異常値
0	SF	−	R/W	サクセス／フェイルビット 書き込み処理が正常に終了したかどうかを戻すビットです。 0：書き込みは正常終了（エラーなし） 1：書き込みが異常終了（エラーが発生している）

5.7 H8S_iic_sr.c 関数の機能説明

本章では、H8S_iic_sr.c ファイル内の関数について機能と使用レジスタを説明します。

5.7.1 iic_sr_128 関数

表 5.16 iic_sr_128 関数の機能説明

機能	戻り値	引数
・IIC2_0 初期化関数（iic_init）のコール ・IIC2_0 によるスレーブ受信を開始	なし	なし

【使用レジスタ】

なし

5.7.2 iic_init 関数

表 5.17 iic_init 関数の機能説明

機能	戻り値	引数
・IIC2_0 の初期化	なし	なし

【使用レジスタ】

IIC バスコントロールレジスタ A_0（ICCRA_0） アドレス：H’FFFD58

ビット	ビット名	初期値	R/W	説明
7	ICE	0	R/W	IIC バスインタフェースイネーブル 0：本モジュールは機能停止状態 1：本モジュールは転送動作可能状態（SCL/SDA はバス駆動状態）
6	RCVD	0	R/W	受信ディスエーブル TRS = 0 の状態で ICDRR をリードしたときに次の動作の継続/禁止を設定します。 0：次の受信動作を継続 1：次の受信動作を禁止

IIC バスコントロールレジスタ A_0(ICCRA_0)　　アドレス:H'FFFD58

ビット	ビット名	初期値	R/W	説明
5	MST	0	R/W	マスタ/スレーブ選択
4	TRS	0	R/W	送信/受信選択 マスタモードでバス競合負けをすると、MST、TRS共にハードウェアによってリセットされてスレーブ受信モードに変わります。なおTRSの変更は転送フレーム間で行ってください。また、スレーブ受信モードで開始条件後の7ビットがSARに設定したスレーブアドレスと一致し、8ビット目が1の場合、TRSが自動的に1にセットされます。 MSTとTRSとの組み合わせにより、以下の動作モードになります。 00:スレーブ受信モード 01:スレーブ送信モード 10:マスタ受信モード 11:マスタ送信モード
3		0	R/W	転送クロック選択3～0 マスタモードのとき、必要な転送レート(表5.18参照)にあわせて設定してください。スレーブモードでは、送信モード時のデータセットアップ時間の確保に使用されます。この時間はCKS3 = 0のとき10Tcyc、CKS3 = 1のとき20Tcycとなります。
2		0	R/W	
1		0	R/W	
0		0	R/W	

表 5.18　転送レート

ビット3	ビット2	ビット1	ビット0	クロック	転送レート						
CKS3	CKS2	CKS1	CKS0		φ＝8MHz	φ＝10MHz	φ＝20MHz	φ＝25MHz	φ＝33MHz	φ＝34MHz*1	φ＝35MHz*2
0	0	0	0	φ/28	286kHz	357kHz	714kHz	893kHz	1179kHz	1214kHz	1250kHz
			1	φ/40	200kHz	250kHz	500kHz	625kHz	825kHz	850kHz	875kHz
		1	0	φ/48	167kHz	208kHz	417kHz	521kHz	688kHz	708kHz	729kHz
			1	φ/64	125kHz	156kHz	313kHz	391kHz	516kHz	531kHz	547kHz
	1	0	0	φ/168	47.6kHz	59.5kHz	119kHz	149kHz	196kHz	202kHz	208kHz
			1	φ/100	80.0kHz	100kHz	200kHz	250kHz	330kHz	340kHz	350kHz
		1	0	φ/112	71.4kHz	89.3kHz	179kHz	223kHz	295kHz	304kHz	313kHz
			1	φ/128	62.5kHz	78.1kHz	156kHz	195kHz	258kHz	266kHz	273kHz
1	0	0	0	φ/56	143kHz	179kHz	357kHz	446kHz	589kHz	607kHz	625kHz
			1	φ/80	100kHz	125kHz	250kHz	313kHz	413kHz	425kHz	438kHz
		1	0	φ/96	83.3kHz	104kHz	208kHz	260kHz	344kHz	354kHz	365kHz
			1	φ/128	62.5kHz	78.1kHz	156kHz	195kHz	258kHz	266kHz	273kHz
	1	0	0	φ/336	23.8kHz	29.8kHz	59.5kHz	74.4kHz	98.2kHz	101kHz	104kHz
			1	φ/200	40.0kHz	50.0kHz	100kHz	125kHz	165kHz	170kHz	175kHz
		1	0	φ/224	35.7kHz	44.6kHz	89.3kHz	112kHz	147kHz	152kHz	156kHz
			1	φ/256	31.3kHz	39.1kHz	78.1kHz	97.7kHz	129kHz	133kHz	137kHz

※1. H8S/2378 0.18μm F-ZTAT グループ、H8S/2378R 0.18μm F-ZTAT グループのみサポートしています。

※2. H8S/2378 のみサポートしています。

スレーブアドレスレジスタ_0(SAR_0)　　アドレス:H' FFFD5D

ビット	ビット名	初期値	R/W	説明
7～1	SVA6～0	0	R/W	スレーブアドレス 6～0 IIC バスにつながる他のスレーブと異なるユニークなアドレスを設定します。
0	−	0	R/W	リザーブビット リード/ライト可能ですが、必ず 0 をライトしてください。

IIC バスインタラプトイネーブルレジスタ_0(ICIER_0)　　アドレス:H' FFFD5B

ビット	ビット名	初期値	R/W	説明
7	TIE	0	R/W	トランスミットインタラプトイネーブル ICSRのTDREがセットされたとき、送信データエンプティ割り込み(TXI)を許可/禁止します。 0:送信データエンプティ割り込み要求(TXI)の禁止 1:送信データエンプティ割り込み要求(TXI)の許可
6	TEIE	0	R/W	トランスミットエンドインタラプトイネーブル TEIEは、ICSRのTDREが1の状態で9クロック目が立ち上がったとき、送信終了割り込み(TEI)の許可/禁止を選択します。なおTEIは、TENDを0にクリアするか、TEIEを0にクリアすることで解除できます。 0:送信終了割り込み要求(TEI)の禁止 1:送信終了割り込み要求(TEI)の許可
5	RIE	0	R/W	レシーブインタラプトイネーブル RIEは受信データがICDRSからICDRRに転送され、ICSRのRDRFが1にセットされたとき、受信データフル割り込み要求(RXI)の許可/禁止を選択します。なおRXIは、RDRFを0にクリアするか、またはRIEを0にクリアすることで解除できます。 0:受信データフル割り込み要求(RXI)の禁止 1:受信データフル割り込み要求(RXI)の許可
4	NAKIE	0	R/W	NACK 受信インタラプトイネーブル NAKIEは、ICSRのNACKFおよびALがセットされたとき、NACK受信割り込み要求(NAKI)の許可/禁止を選択します。なおNAKIは、NACKFまたはALを0にクリアするか、またはNAKIEを0にクリアすることで解除できます。 0:NACK 受信割り込み要求(NAKI)の禁止 1:NACK 受信割り込み要求(NAKI)の許可
3	STIE	0	R/W	停止条件検出インタラプトイネーブル 0:停止条件検出割り込み要求(STPI)の禁止 1:停止条件検出割り込み要求(STPI)の許可
2	ACKE	0	R/W	アクノリッジビット判定選択 0:受信アクノリッジの内容を無視して連続的に転送を行う。 1:受信アクノリッジが 1 の場合、転送を中断する。
1	ACKBR	0	R	受信アクノリッジ 送信モード時、受信デバイスから受け取ったアクノリッジビットの内容を格納しておくビットです。ライトは無効です。 0:受信アクノリッジ = 0 1:受信アクノリッジ = 1

IIC バスインタラプトイネーブルレジスタ_0(ICIER_0)　　アドレス:H' FFFD5B

ビット	ビット名	初期値	R/W	説明
0	ACKBT	0	R/W	送信アクノリッジ 受信モード時、アクノリッジのタイミングで送出するビットを設定します。 0:アクノリッジのタイミングで0を送出 1:アクノリッジのタイミングで 1 を送出

IIC バスステータスレジスタ_0(ICSR_0)　　アドレス:H' FFFD5C

ビット	ビット名	初期値	R/W	説明
7	TDRE	0	R/W	トランスミットデータエンプティ [セット条件] ・ICDRTからICDRSにデータ転送が行われ、ICDRTがエンプティになったとき ・TRSをセットしたとき ・開始条件(再送含む)を発行したとき ・スレーブモードで受信モードから送信モードになったとき [クリア条件] ・1の状態をリードした後、0をライトしたとき ・ICDRT へデータをライトしたとき
6	TEND	0	R/W	トランスミットエンド [セット条件] ・TDREが1の状態でSCLの9クロック目が立ち上がったとき [クリア条件] ・1の状態をリードした後、0をライトしたとき ・ICDRT へデータをライトしたとき
5	RDRF	0	R/W	レシーブデータレジスタフル [セット条件] ・ICDRSからICDRRに受信データが転送されたとき [クリア条件] ・1の状態をリードした後、0をライトしたとき ・ICDRR をリードしたとき
4	NACKF	0	R/W	ノーアクノリッジ検出フラグ [セット条件] ・ICIERのACKE = 1の状態で、送信時、受信デバイスからアクノリッジがなかったとき [クリア条件] ・1 の状態をリードした後、0 をライトしたとき
3	STOP	0	R/W	停止条件検出フラグ [セット条件] ・マスタモード時、フレームの転送の完了後に停止条件を検出したとき ・スレーブモード時、ゼネラルコール後および開始条件検出後の第1バイトのスレーブアドレスとSARに設定したアドレスが一致した後、停止条件を検出したとき [クリア条件] ・1 の状態をリードした後、0 をライトしたとき

IIC バスステータスレジスタ_0（ICSR_0）　　アドレス：H’FFFD5C

ビット	ビット名	初期値	R/W	説明
2	AL	0	R/W	アービトレーションロストフラグ ALは、マスタモード時にバス競合負けをしたことを示します。 複数のマスタがほぼ同時にバスを占有しようとしたときにIICバスインタフェースはSDAをモニタし、自分が出したデータと異なった場合、ALフラグを1にセットしてバスが他のマスタによって占有されたことを示します。 [セット条件] ・マスタ送信モードの場合、SCLの立ち上がりで内部SDAとSDA端子のレベルが不一致のとき ・マスタモードの場合、開始条件検出時、SDA端子がハイレベルのとき [クリア条件] ・1 の状態をリードした後、0 をライトしたとき
1	AAS	0	R/W	スレーブアドレス認識フラグ スレーブ受信モードで開始条件直後の第1フレームがSARのSVA6～SVA0と一致した場合にセットされます。 [セット条件] ・スレーブ受信モードでスレーブアドレスを検出したとき ・スレーブ受信モードでゼネラルコールアドレスを検出したとき [クリア条件] ・1 の状態をリードした後、0 をライトしたとき
0	ADZ	0	R/W	ゼネラルコールアドレス認識フラグ スレーブ受信モードのとき有効 [セット条件] ・スレーブ受信モードかつゼネラルコールアドレスを検出したとき [クリア条件] ・1 の状態をリードした後、0 をライトしたとき

5.7.3　iici0_int 関数

表 5.19　iici0_int 関数の機能説明

機能	戻り値	引数
・割り込み例外処理ベクタ IICI0（ベクタ番号 116）を使用した IIC2_0 の割り込み処理 ・IIC2_0 の割り込み処理中に停止条件を検出時に停止処理関数（receive_stop_condition）をコール ・Iic2_0 の割り込み処理中に IIC_マスタ送信側からデータが送信された場合にスレーブ受信処理関数（slave_receive）をコール	なし	なし

【使用レジスタ】

iic_init 関数内で使用している ICSR_0 の STOP ビット、ICCRA_0 の TRS ビットを使用しているため詳細は省略。

5.7.4 receive_stop_condition 関数

表 5.20 receive_stop_condition 関数の機能説明

機能	戻り値	引数
・IIC2_0 割り込み処理で停止条件を検出した場合（ICSR_0 の STOP ビットが 1）にコールされ、停止処理を行う。	なし	なし

【使用レジスタ】

iic_init 関数内で使用している ICSR_0、ICCRA_0、ICIER_0 を使用しているため詳細は省略。

5.7.5 slave_receive 関数

表 5.21 slave_receive 関数の機能説明

機能	戻り値	引数
・IIC2_0 割り込み処理で IIC_マスタ送信側からデータが送信された場合にコールされ、スレーブ受信処理を行う。	なし	なし

【使用レジスタ】

IIC バスインタラプトイネーブルレジスタ_0（ICIER_0） アドレス：H’ FFFD5B

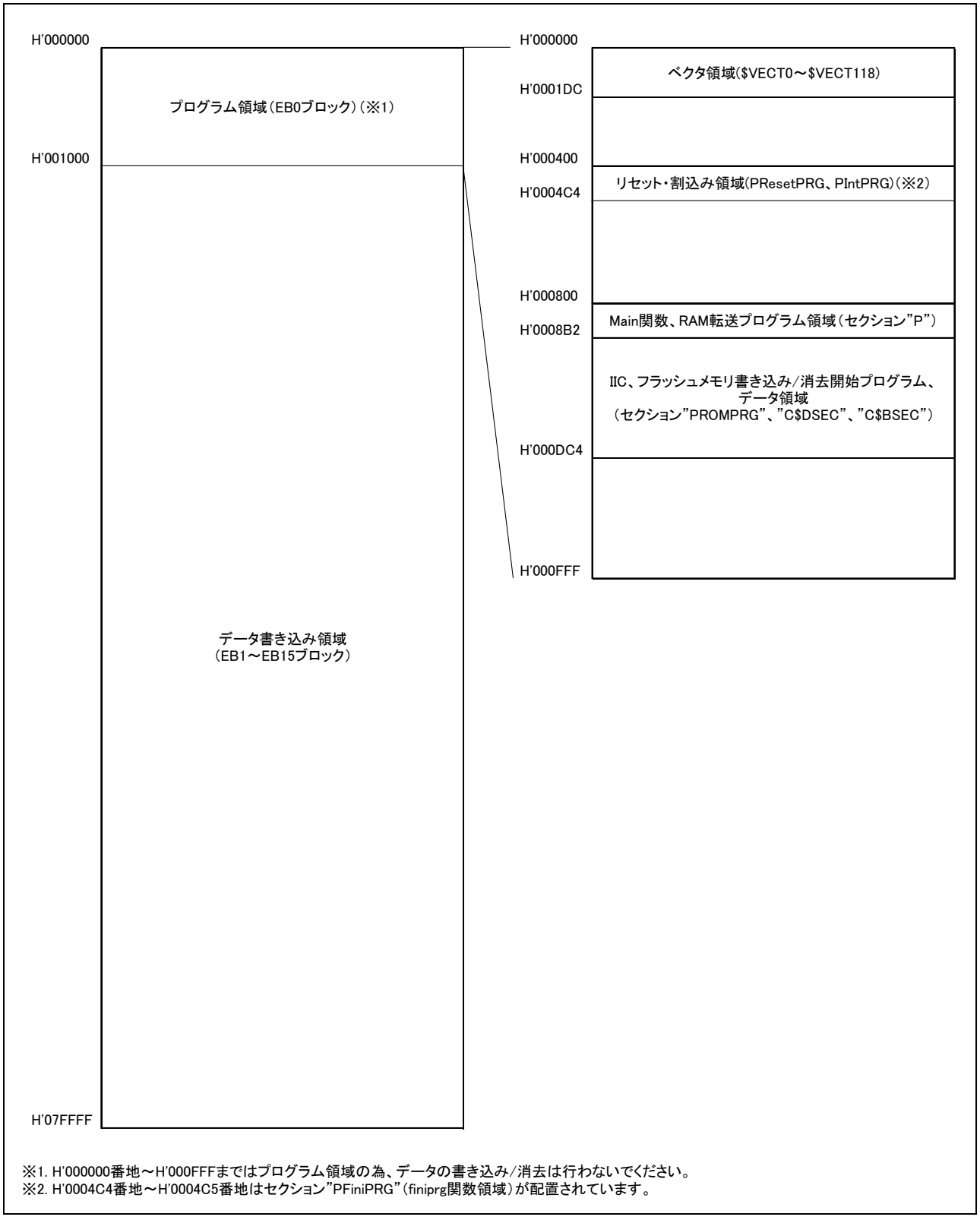
ビット	ビット名	初期値	R/W	説明
0	ACKBT	0	R/W	送信アクノリッジ 受信モード時、アクノリッジのタイミングで送出するビットを設定します。 0：アクノリッジのタイミングで0を送出 1：アクノリッジのタイミングで 1 を送出

IIC バス受信データレジスタ_0（ICDRR_0） アドレス：H’ FFFD5F

ビット	ビット名	初期値	R/W	説明
7～0	ICDRR	FF	R	受信データを格納する 8 ビットのレジスタです。1 バイトのデータの受信が終了すると、受信したデータを ICDRS から ICDRR へ転送し、次のデータを受信可能にします。なお ICDRR は受信専用レジスタですので、CPU からライトできません。

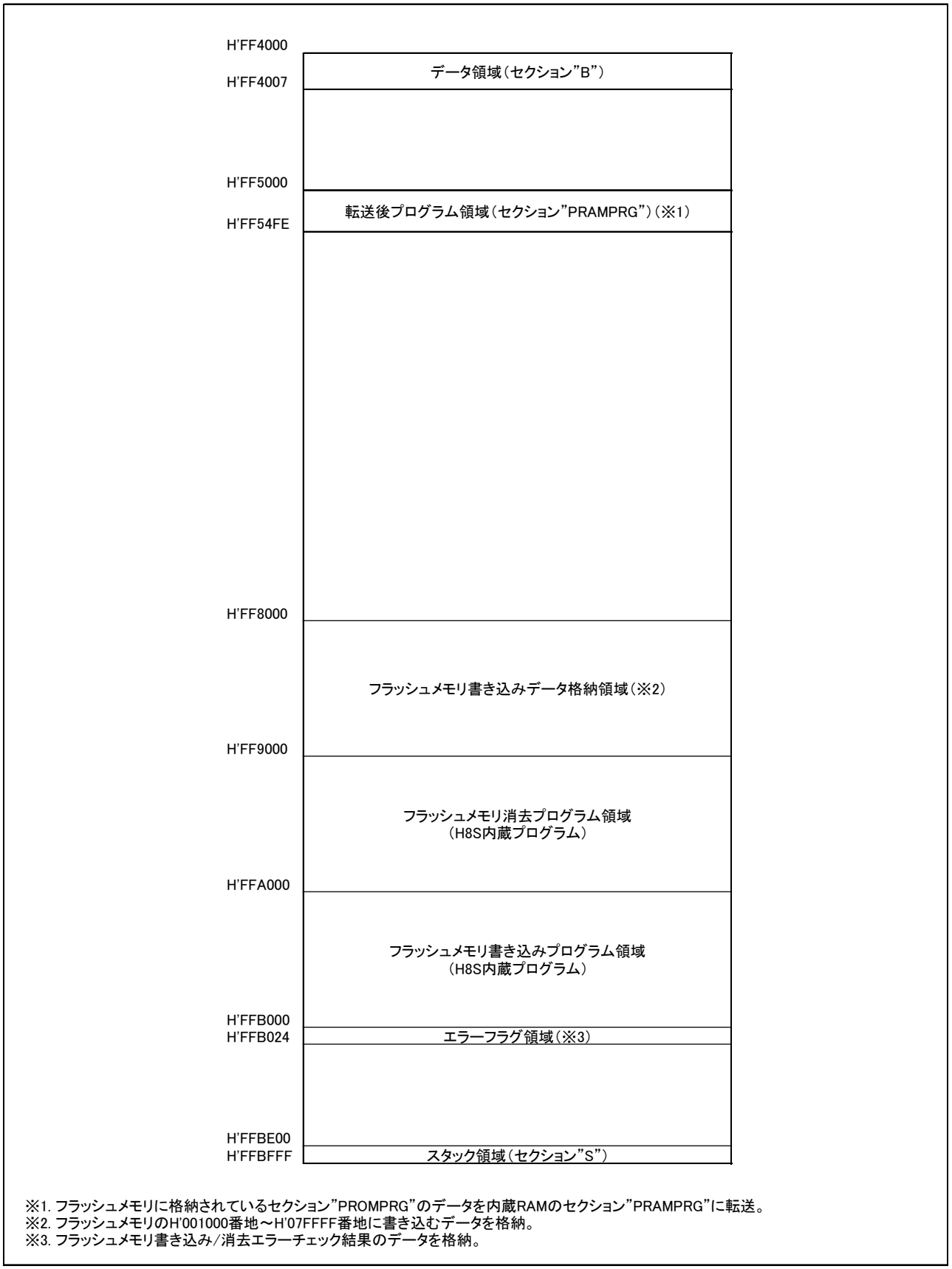
5.8 メモリマップ

図 5.2 にフラッシュメモリ（0.18μm F-ZTAT 版）、図 5.3 に内蔵 RAM のアドレスマップを示します。



※1. H'000000番地～H'000FFFまではプログラム領域の為、データの書き込み/消去は行わないでください。
※2. H'0004C4番地～H'0004C5番地はセクション"PFiniPRG"（finiprg関数領域）が配置されています。

図 5.2 フラッシュメモリ（0.18μm F-ZTAT 版）アドレスマップ



※1. フラッシュメモリに格納されているセクション"PROMPRG"のデータを内蔵RAMのセクション"PRAMPRG"に転送。
※2. フラッシュメモリのH'001000番地～H'07FFFF番地に書き込むデータを格納。
※3. フラッシュメモリ書き込み/消去エラーチェック結果のデータを格納。

図 5.3　内蔵 RAM アドレスマップ

6. 動作説明

本章では、High-performance Embedded Workshop（HEW）を起動し、サンプルプログラムを使ってフラッシュメモリ（0.18μm F-ZTAT 版）の書き込み/消去を行った場合に、正常に書き込み/消去が行なえているか確認する方法を説明します。説明順序としては以下のようになります。

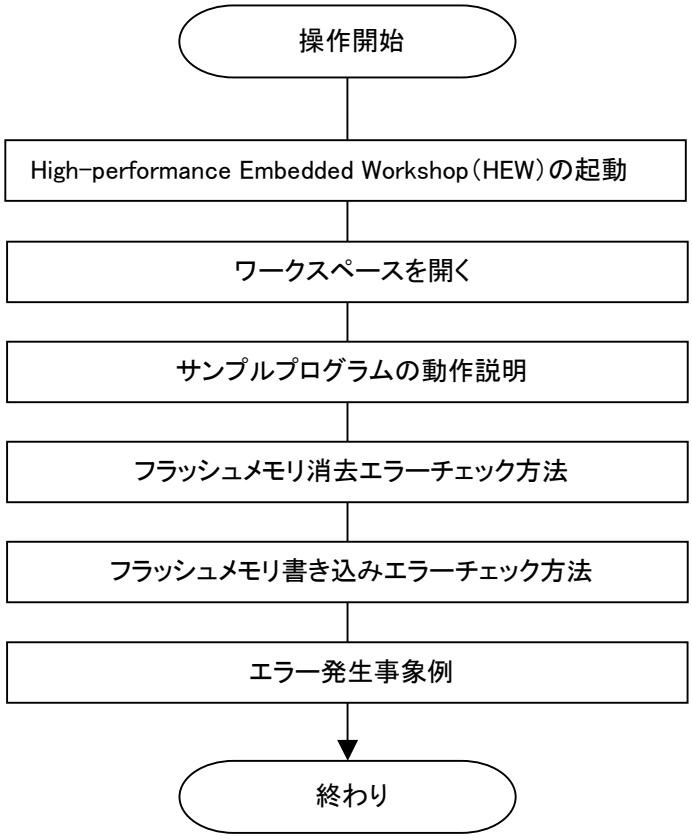


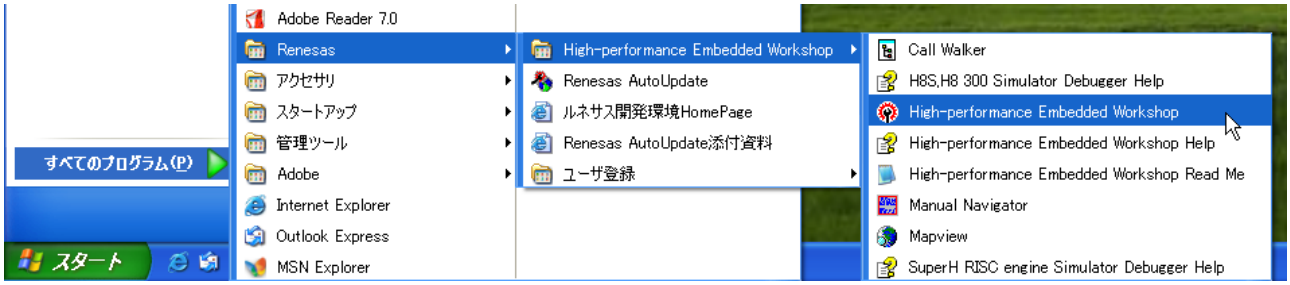
図 6.1　本アプリケーションノートの説明順序

6.1　High-performance Embedded Workshop（HEW）の起動

まず、始めにユーザシステムを接続したE10A-USBエミュレータとホストコンピュータをUSBケーブルで接続し、デバッグ操作が可能であることを確認してください。

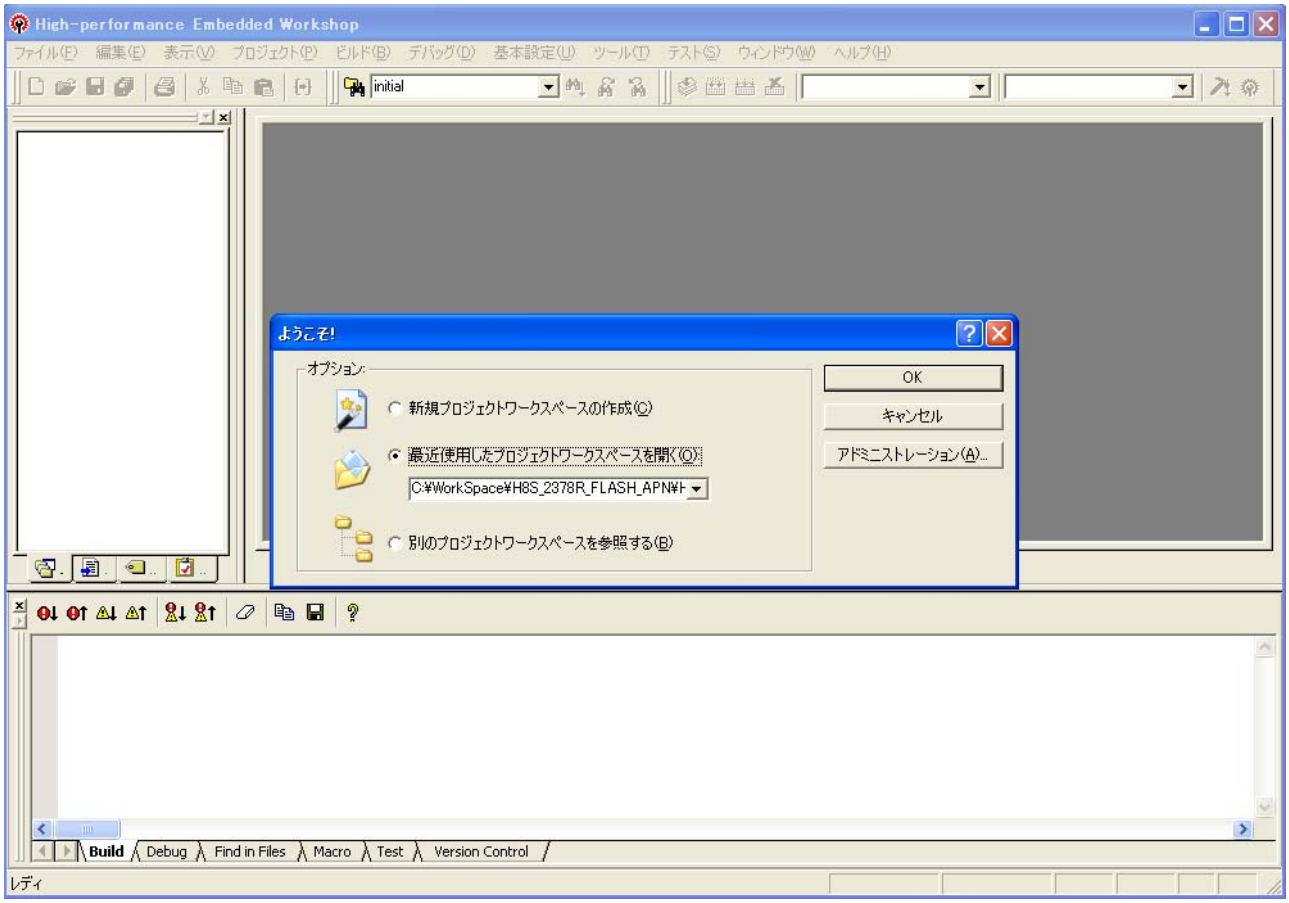
次にHigh-performance Embedded Workshopを起動します。

[スタート]メニューの[すべてのプログラム]から[Renesas]→[High-performance Embedded Workshop]→[High-performance Embedded Workshop]で起動できます。

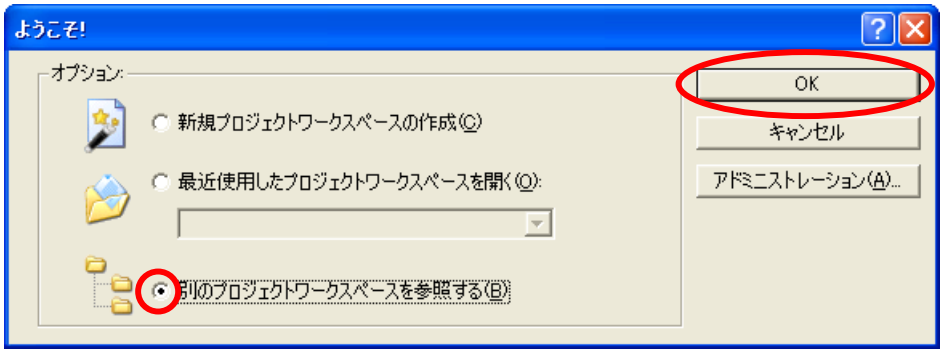


6.2　ワークスペースを開く

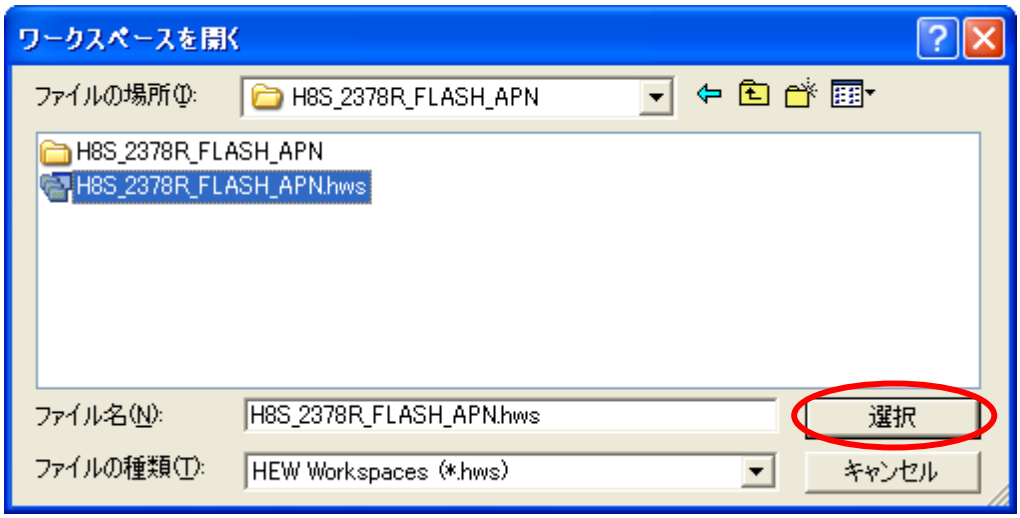
(1)　High-performance Embedded Workshop 上に[ようこそ!]ダイアログボックスが表示されます。



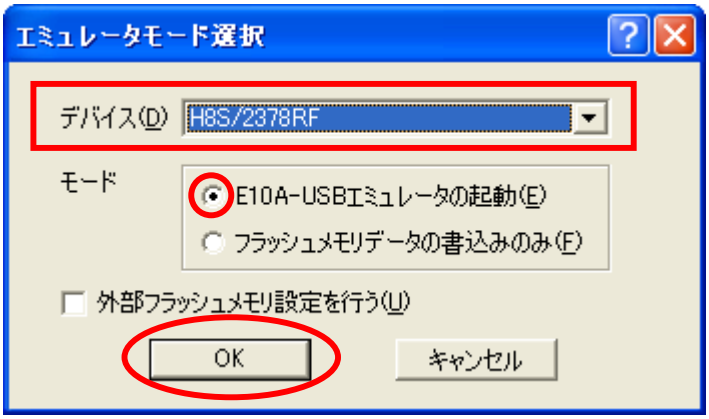
[ようこそ!]ダイアログボックス内の[別のプロジェクトワークスペースを参照する]ラジオボタンを選択して[OK]ボタンを押してください。



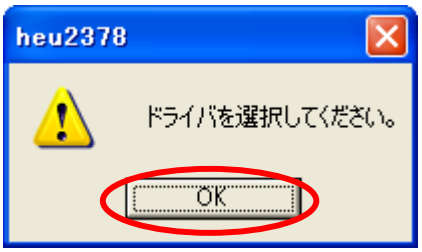
(2) [ワークスペースを開く]ダイアログボックスが表示されます。ファイル“H8S_2378R_FLASH_APN.hws”を指定して[選択]ボタンを押してください。



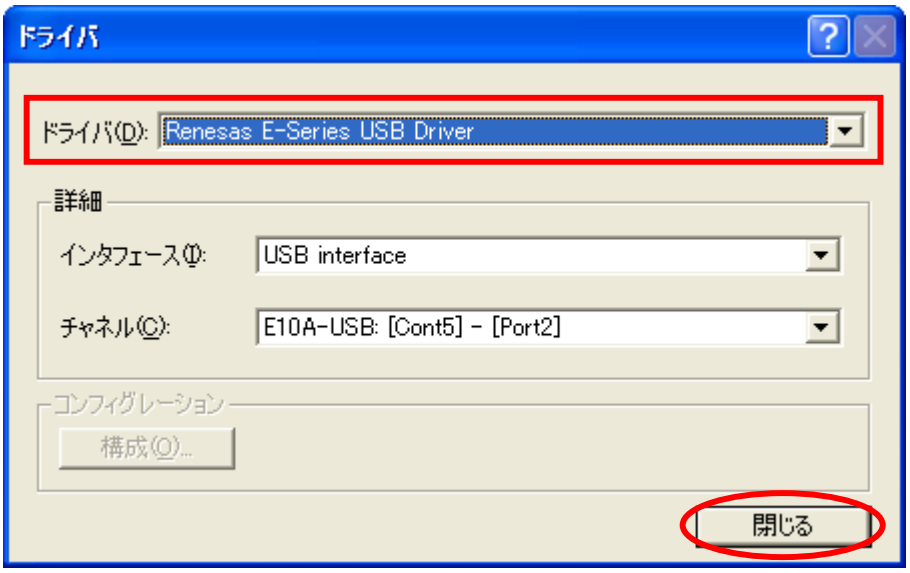
(3) [エミュレータモード選択]ダイアログボックスが表示されるので、[デバイス]欄を“H8S/2378RF”、[モード]欄を“E10A-USB エミュレータの起動”に選択して[OK]ボタンを押してください。



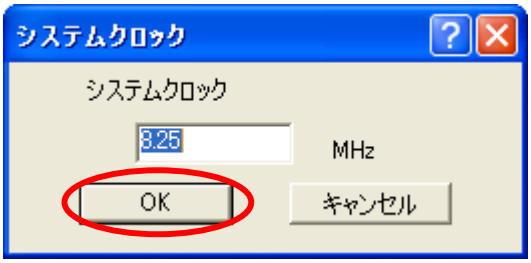
(4) 初回のみ[heu2378]ダイアログボックスが表示されるので、[OK]を押してください。



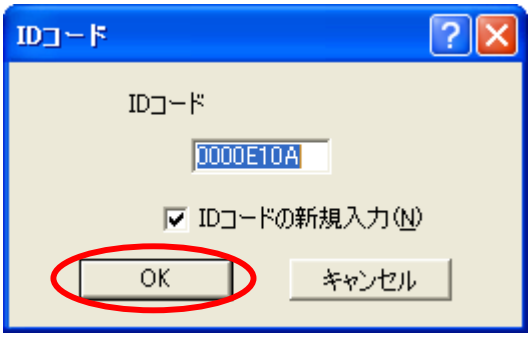
(5) 初回のみ[ドライバ]ダイアログボックスが表示されるので、[ドライバ]欄を“Renesas E-Series USB Driver”に選択して[閉じる]ボタンを押してください。[詳細]は自動的に選択されます。



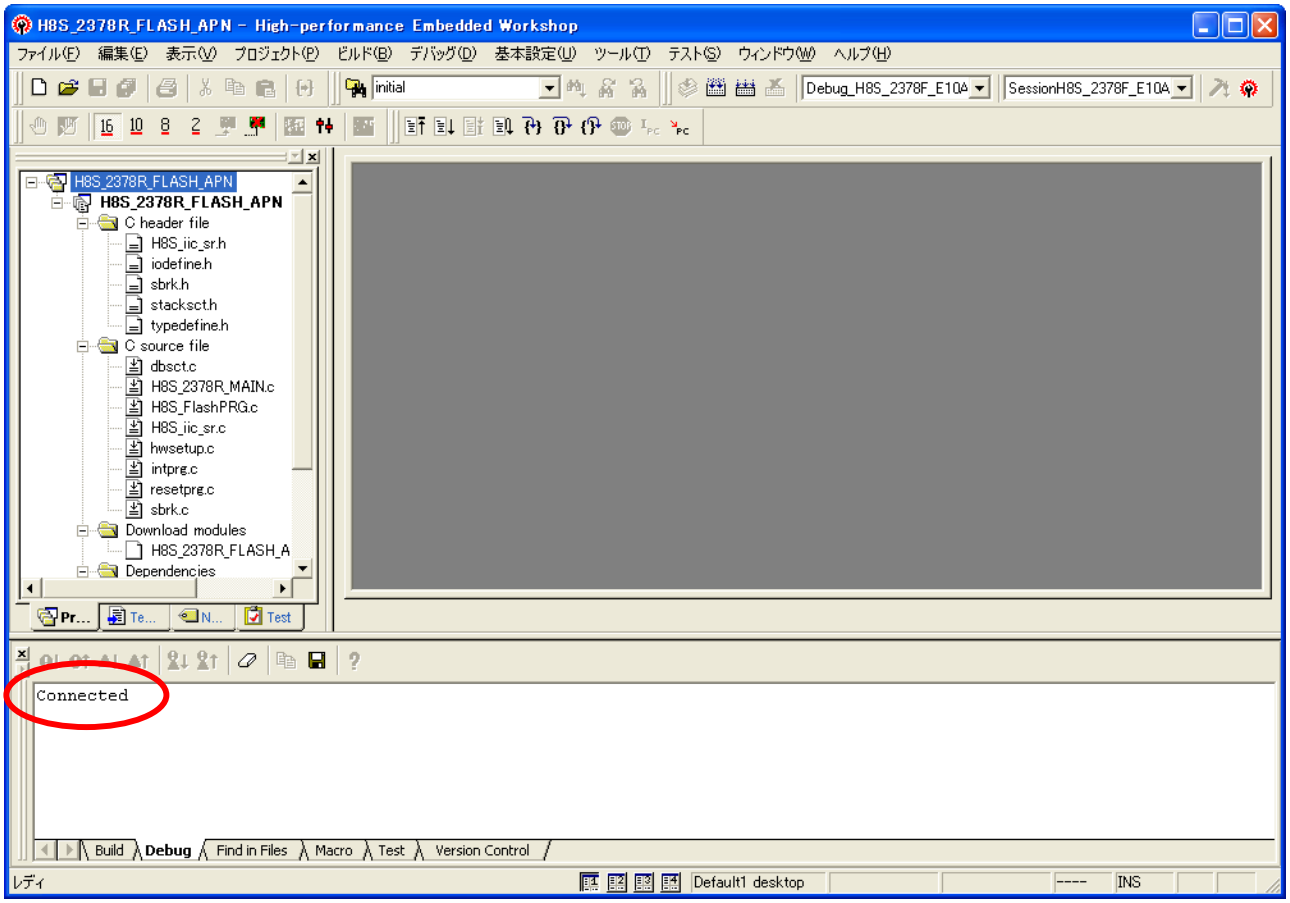
(6) [システムクロック]ダイアログボックスが表示されるので、外部クロック周波数の値を入力して[OK]ボタンを押してください。今回は、本書で使用しているボード“HSB8S2378RE”の外部クロック周波数“8.25MHz”を入力します。



(7) [ID コード]ダイアログボックスが表示されるので、デフォルトのまま変更せずに[OK]ボタンを押してください。



(8) E10A-USB エミュレータの接続が完了して High-performance Embedded Workshop の画面が操作可能になります。接続が完了するとアウトプットウィンドウの[Debug]タブ上に“Connected”と表示されます。



(9) 本サンプルプログラムを動作させる前に以下の変更がされているか確認してください。

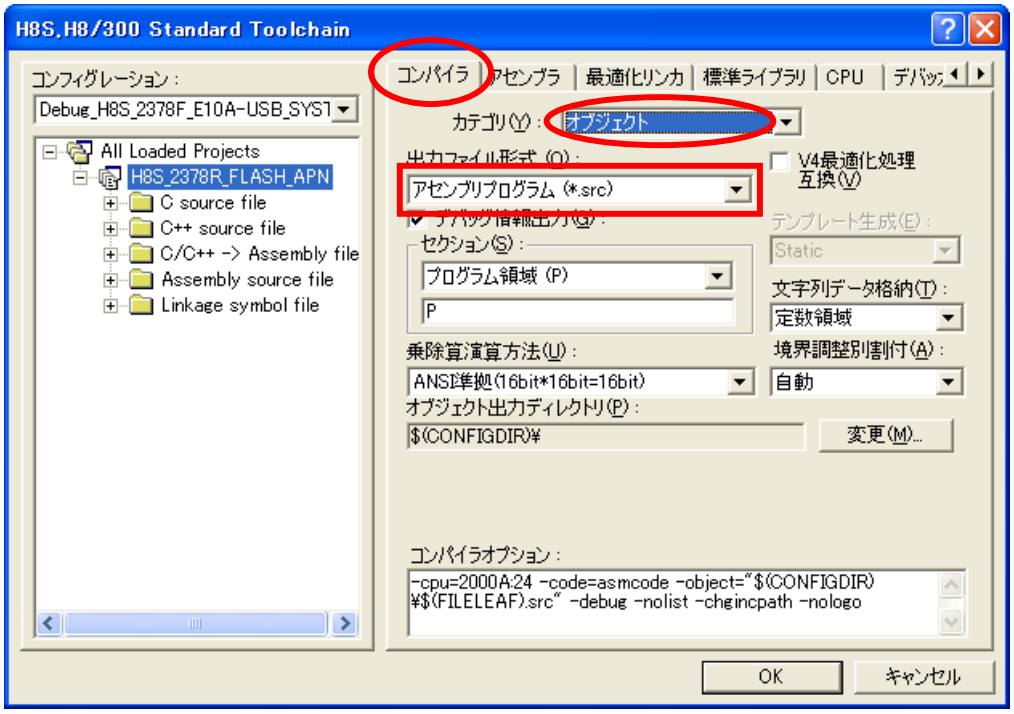
・H8S, H8/300 Standard Toolchain のコンパイラ、セクション設定変更

・割り込みプログラムファイル（intprg.c）の内容変更

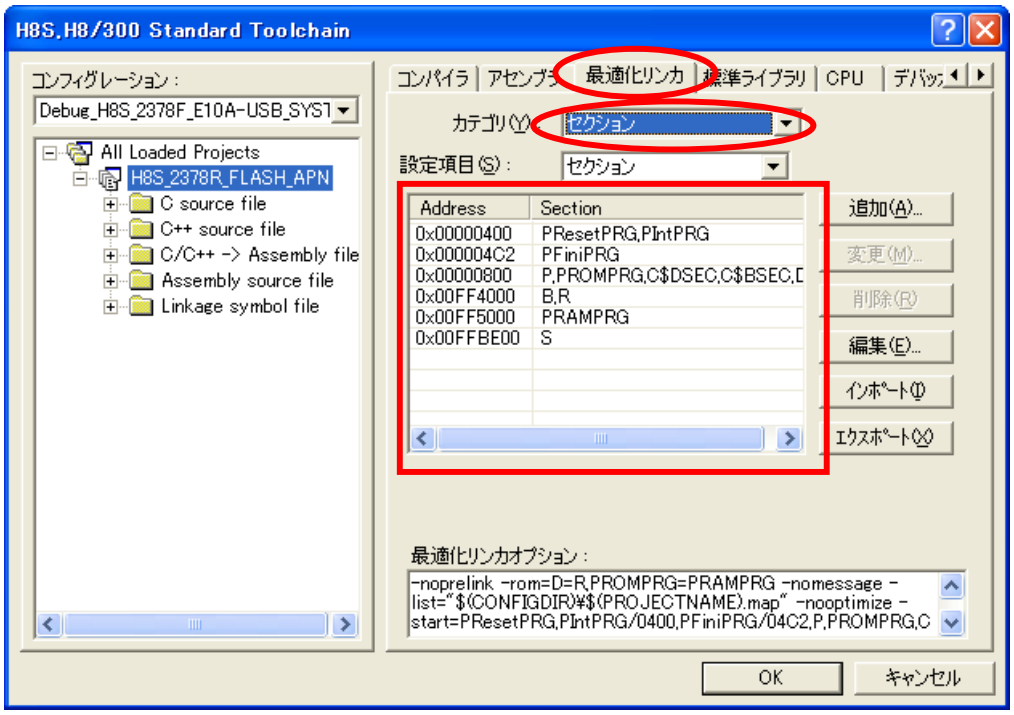
【H8S, H8/300 Standard Toolchain の変更箇所】

[ビルド]メニューの[H8S, H8/300 Standard Toolchain..]を選択してください。

[H8S, H8/300 Standard Toolchain]ダイアログボックスが表示されるので、[コンパイラ]タブの[カテゴリ]欄を“オブジェクト”にして[出力ファイル形式]欄を“アセンブリプログラム（*.src）”に変更します。



[最適化リンカ]タブの[カテゴリ]欄を“セクション”にして、以下のようにセクションを追加・削除します。

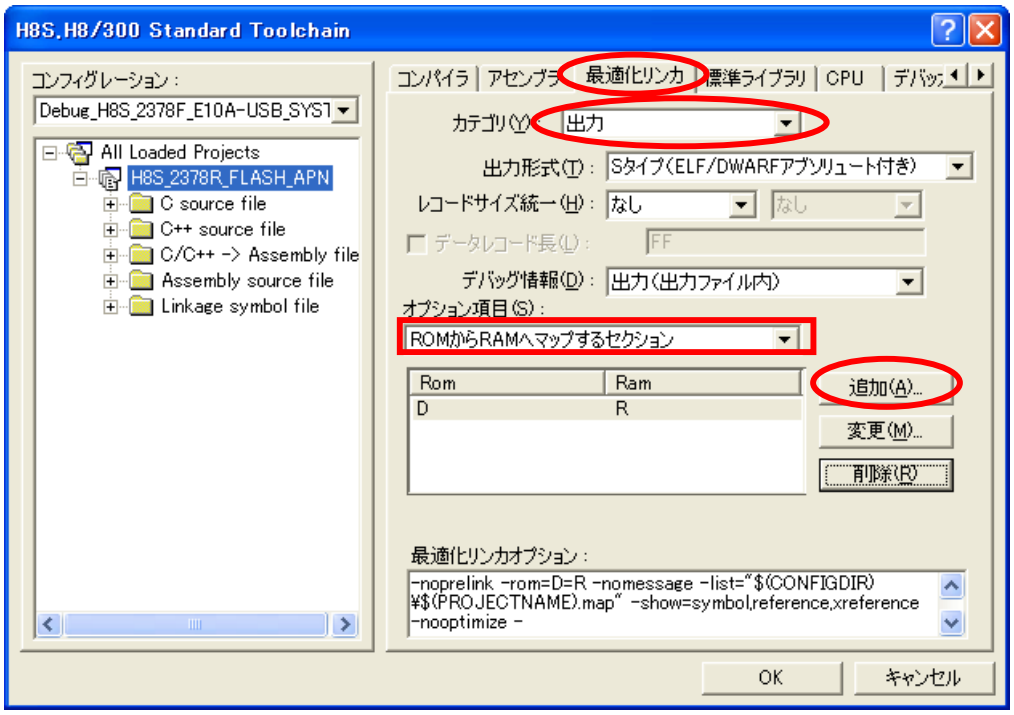


・0x000004C2 番地にセクション“PFiniPRG”を追加。

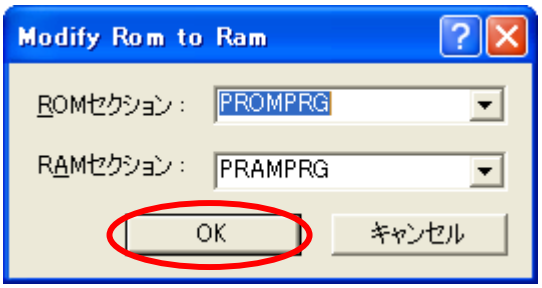
・0x00000800 番地のセクション“C”を削除して、セクション“P”の次にセクション“PROMPRG”を追加。

・0x00FF5000 番地にセクション“PRAMPRG”を追加。

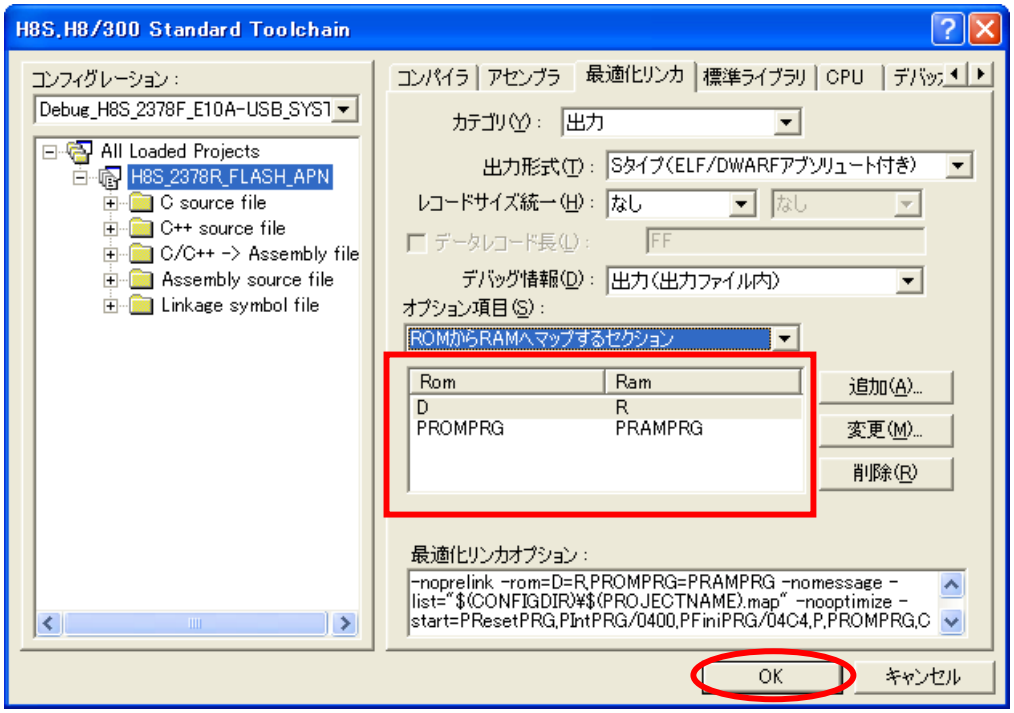
[最適化リンカ]タブの[カテゴリ]欄を“出力”にし、[オプション項目]欄を“ROM から RAM へマップするセクション”にして[追加]ボタンを押してください。



[Modify Rom to Ram]ダイアログボックスが表示されるので、[ROM セクション]欄に“PROMPRG”、[RAM セクション]欄に“PRAMPRG”を入力して[OK]ボタンを押してください。



以下のように表示され、フラッシュメモリ領域にあるセクション“PROMPRG”が、内蔵 RAM 領域にセクション“PRAMPRG”としてマッピングされます。[OK]ボタンを押して[H8S, H8/300 Standard Toolchain]ダイアログボックスを閉じてください。



【割り込みプログラムファイル（intprg.c）の変更箇所】

“vector 116 IICI0”は IIC2_0 スレーブ受信プログラムファイル（H8S_iic_sr.c）内で使用しているため、コメントアウトにしてください。

```
247    // vector 115 Reserved
248
249    // vector 116 IICI0
250    //__interrupt(vect=116) void INT_IICI0(void) {/* sleep(); */}
251    // vector 117 Reserved
252
253    // vector 118 IICI1
254    __interrupt(vect=118) void INT_IICI1(void) {/* sleep(); */}
255    // vector 119 Reserved
256
257    // vector 120 Reserved
258
```

6.3　サンプルプログラムの動作説明

本章では、サンプルプログラムの下記一連の動作について説明します。図 6.2 にサンプルプログラムの流れを示します。

・フラッシュメモリの消去

・フラッシュメモリの書き込み

なお、フラッシュメモリの書き込みを行う場合は、書き込む領域を必ず消去してから実行してください。消去されていない領域にデータを書き込もうとするとエラーになります。

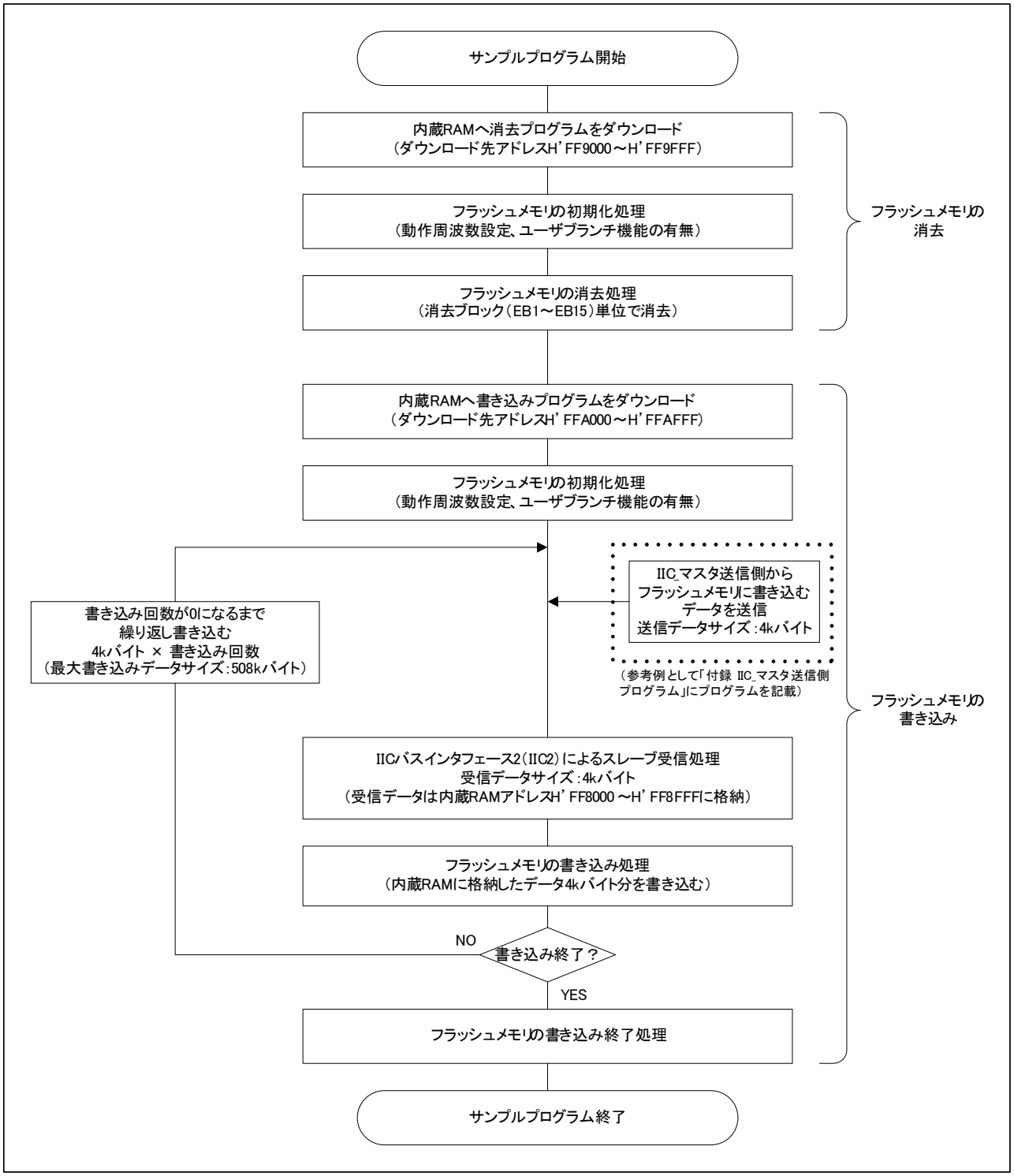
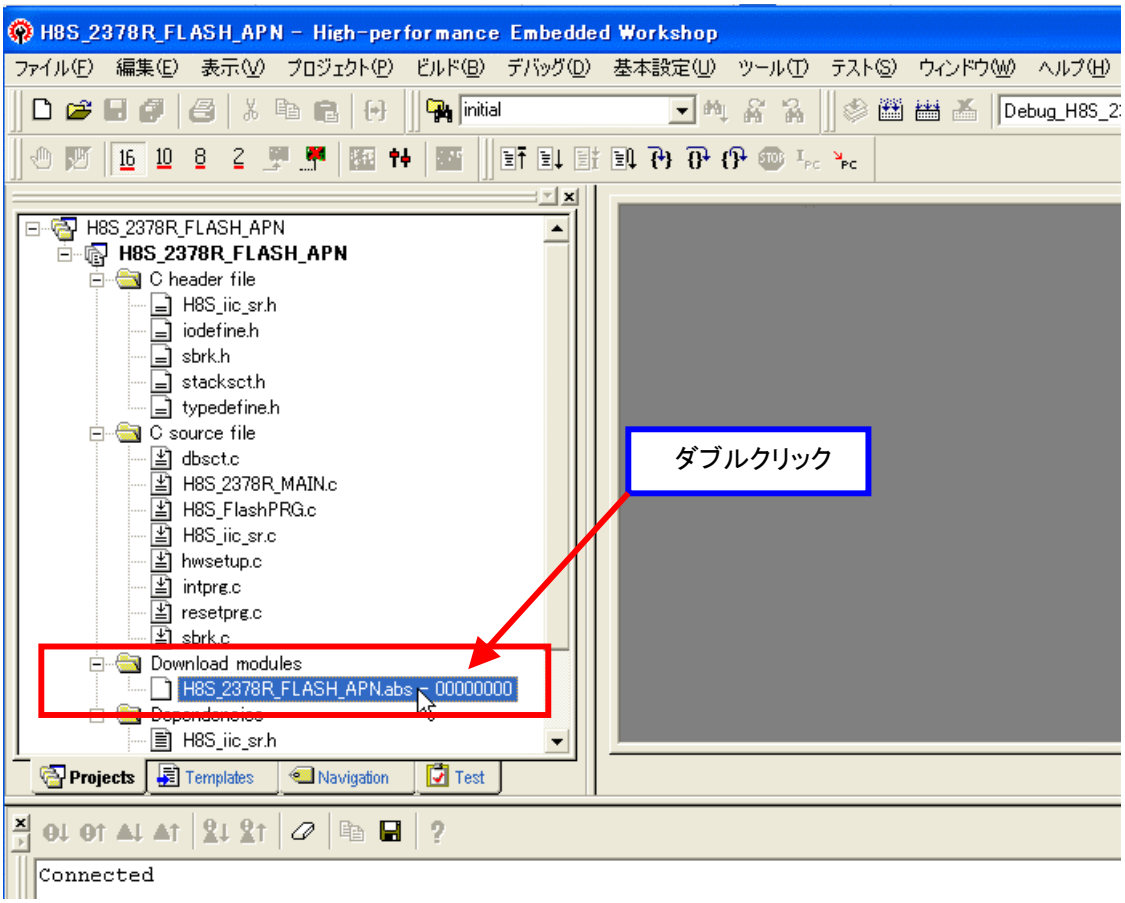


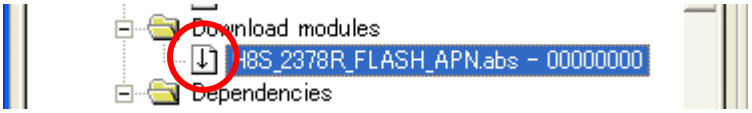
図 6.2　サンプルプログラムの流れ

6.3.1　フラッシュメモリの消去

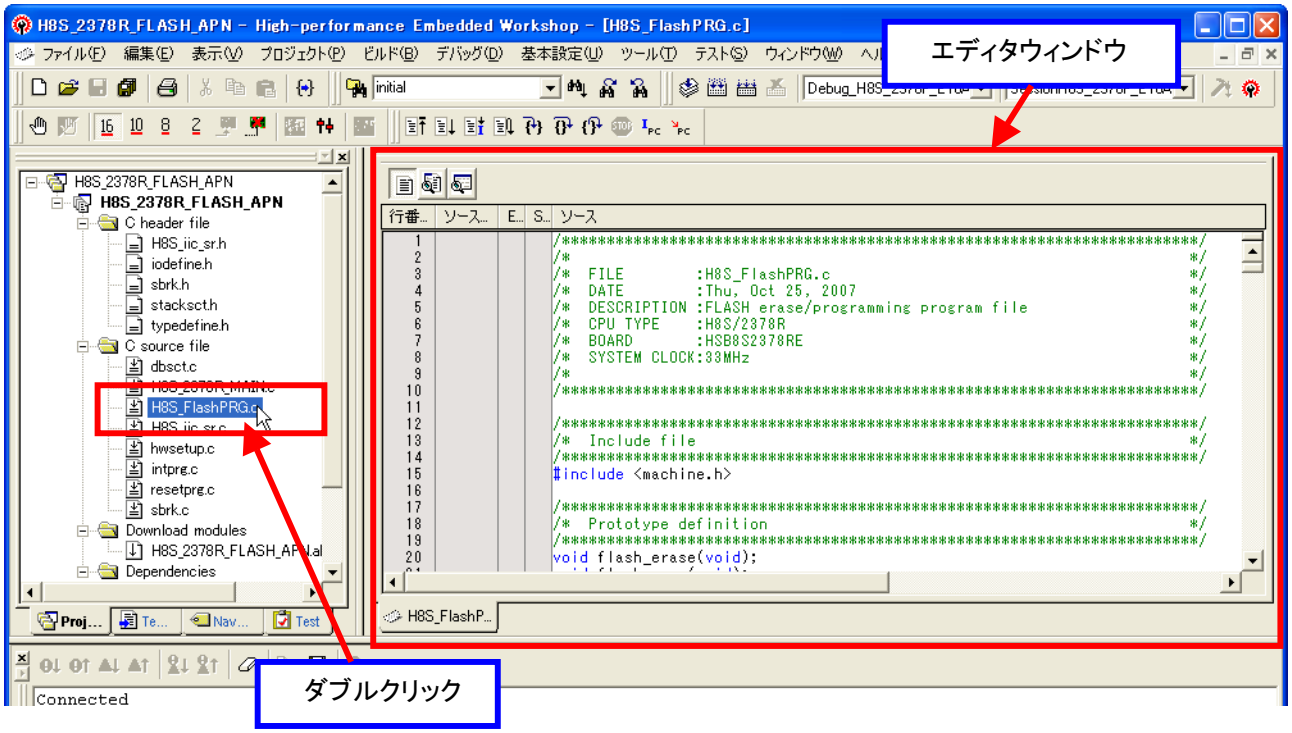
(1)　最初にワークスペースウィンドウ上の[Download modules]フォルダ内のファイル“H8S_2378R_FLASH_APN.abs - 00000000”をダブルクリックして、サンプルプログラムをダウンロードしてください。



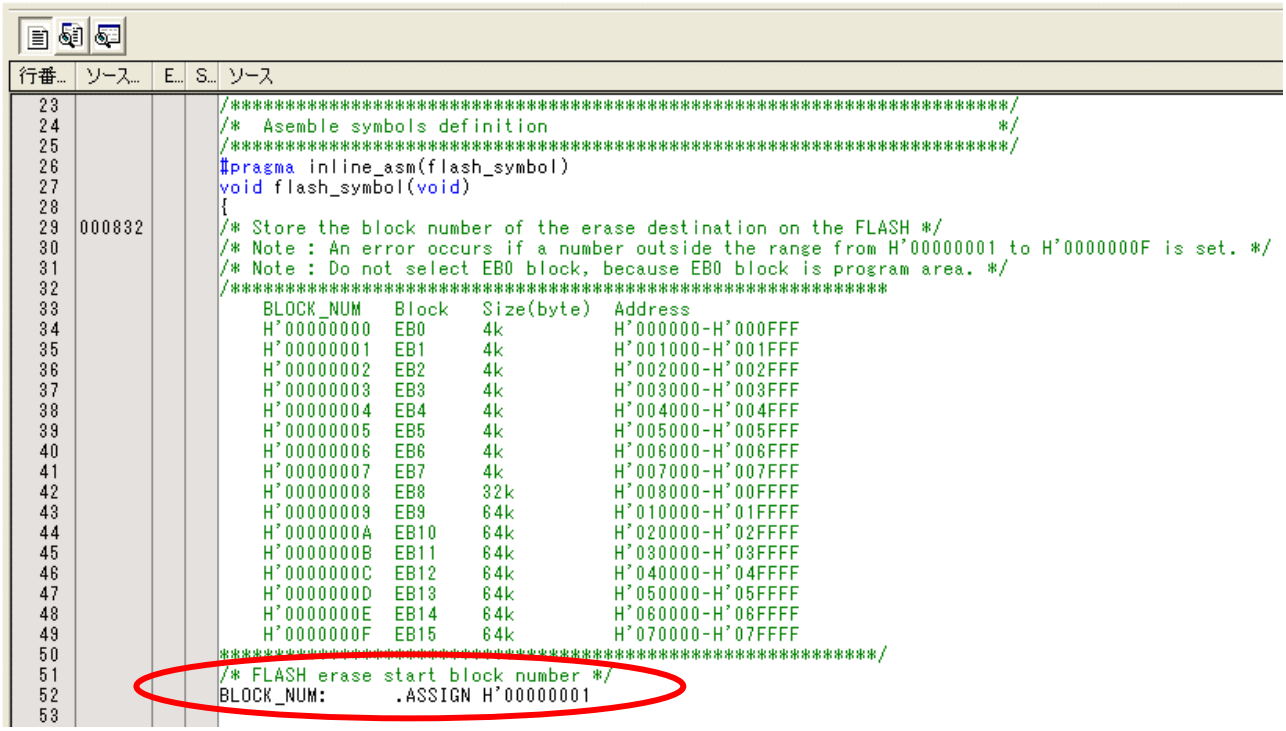
ダウンロードが終了すると以下のように下矢印が表示されます。



(2) ワークスペースウィンドウ上の[C source file]フォルダ内のファイル“H8S_FlashPRG.c”をダブルクリックして、エディタウィンドウ上にソースを表示させてください。



(3) フラッシュメモリの消去を行う上で必要なパラメータ“BLOCK_NUM（消去開始ブロック番号）”、“ERASE_N（消去回数）”の設定を行います。各パラメータは、“H8S_FlashPRG.c”ファイル内の“flash_symbol”関数内にあります。



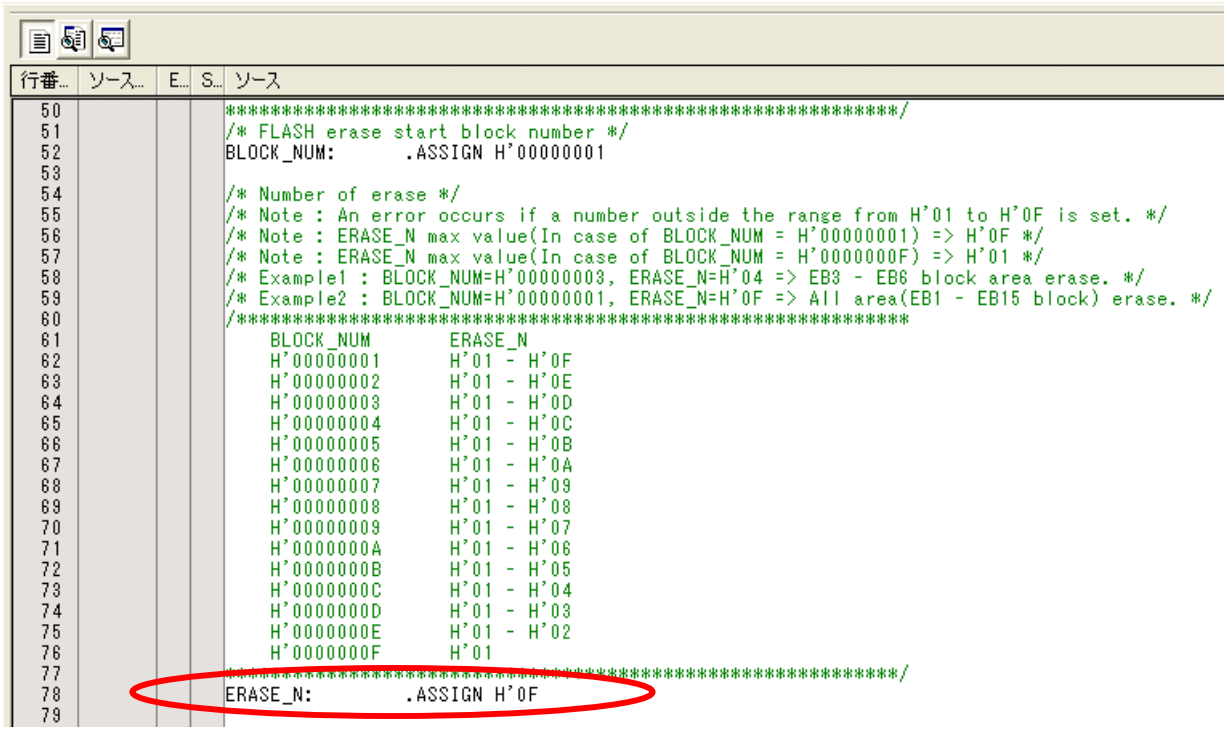
```
/*****************************************************************************/
/*  Asemble symbols definition                                               */
/*****************************************************************************/
#pragma inline_asm(flash_symbol)
void flash_symbol(void)
{
/* Store the block number of the erase destination on the FLASH */
/* Note : An error occurs if a number outside the range from H'00000001 to H'0000000F is set. */
/* Note : Do not select EB0 block, because EB0 block is program area. */
/*************************************************************************
    BLOCK_NUM   Block   Size(byte)  Address
    H'00000000  EB0     4k          H'000000-H'000FFF
    H'00000001  EB1     4k          H'001000-H'001FFF
    H'00000002  EB2     4k          H'002000-H'002FFF
    H'00000003  EB3     4k          H'003000-H'003FFF
    H'00000004  EB4     4k          H'004000-H'004FFF
    H'00000005  EB5     4k          H'005000-H'005FFF
    H'00000006  EB6     4k          H'006000-H'006FFF
    H'00000007  EB7     4k          H'007000-H'007FFF
    H'00000008  EB8     32k         H'008000-H'00FFFF
    H'00000009  EB9     64k         H'010000-H'01FFFF
    H'0000000A  EB10    64k         H'020000-H'02FFFF
    H'0000000B  EB11    64k         H'030000-H'03FFFF
    H'0000000C  EB12    64k         H'040000-H'04FFFF
    H'0000000D  EB13    64k         H'050000-H'05FFFF
    H'0000000E  EB14    64k         H'060000-H'06FFFF
    H'0000000F  EB15    64k         H'070000-H'07FFFF
*************************************************************************/
/* FLASH erase start block number */
BLOCK_NUM:      .ASSIGN H'00000001
```

BLOCK_NUM は、消去を開始するブロック番号（アドレス）を設定するパラメータで、設定範囲は “H’00000001～H’0000000F”です。設定範囲以外の値を設定してサンプルプログラムを実行するとエラーが発生します。また、サンプルプログラムではブロック EB0 の範囲をプログラム領域として使用しているため消去できない仕様となっています。

表 6.1 に BLOCK_NUM の詳細を示します。

表 6.1　BLOCK_NUM パラメータについて

BLOCK_NUM の設定値	消去開始ブロック	メモリ容量	消去開始アドレス
H'00000001	EB1	4k バイト	H'001000
H'00000002	EB2	4k バイト	H'002000
H'00000003	EB3	4k バイト	H'003000
H'00000004	EB4	4k バイト	H'004000
H'00000005	EB5	4k バイト	H'005000
H'00000006	EB6	4k バイト	H'006000
H'00000007	EB7	4k バイト	H'007000
H'00000008	EB8	32k バイト	H'008000
H'00000009	EB9	64k バイト	H'010000
H'0000000A	EB10	64k バイト	H'020000
H'0000000B	EB11	64k バイト	H'030000
H'0000000C	EB12	64k バイト	H'040000
H'0000000D	EB13	64k バイト	H'050000
H'0000000E	EB14	64k バイト	H'060000
H'0000000F	EB15	64k バイト	H'070000



ERASE_N は、BLOCK_NUM で指定したブロック番号から連続して何ブロックまで消去を行うか設定するパラメータで、設定範囲は BLOCK_NUM の値に依存します。詳しくは「表 6.2　BLOCK_NUM と ERASE_N の関係」をご参照ください。

設定範囲以外の値を設定してサンプルプログラムを実行するとエラーが発生します。

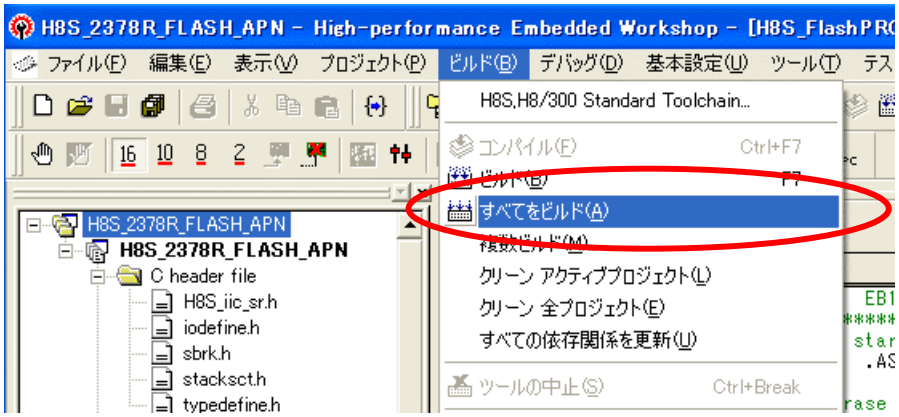
表 6.2　BLOCK_NUM と ERASE_N の関係

BLOCK_NUM の設定値	ERASE_N の設定範囲	消去内容
H’00000001	H’01～H’0F	EB1 の範囲を消去（ERASE_N = H’01 設定時） EB1～EB15 の範囲を消去（ERASE_N = H’0F 設定時）
H’00000002	H’01～H’0E	EB2 の範囲を消去（ERASE_N = H’01 設定時） EB2～EB15 の範囲を消去（ERASE_N = H’0E 設定時）
H’00000003	H’01～H’0D	EB3 の範囲を消去（ERASE_N = H’01 設定時） EB3～EB15 の範囲を消去（ERASE_N = H’0D 設定時）
H’00000004	H’01～H’0C	EB4 の範囲を消去（ERASE_N = H’01 設定時） EB4～EB15 の範囲を消去（ERASE_N = H’0C 設定時）
H’00000005	H’01～H’0B	EB5 の範囲を消去（ERASE_N = H’01 設定時） EB5～EB15 の範囲を消去（ERASE_N = H’0B 設定時）
H’00000006	H’01～H’0A	EB6 の範囲を消去（ERASE_N = H’01 設定時） EB6～EB15 の範囲を消去（ERASE_N = H’0A 設定時）
H’00000007	H’01～H’09	EB7 の範囲を消去（ERASE_N = H’01 設定時） EB7～EB15 の範囲を消去（ERASE_N = H’09 設定時）
H’00000008	H’01～H’08	EB8 の範囲を消去（ERASE_N = H’01 設定時） EB8～EB15 の範囲を消去（ERASE_N = H’08 設定時）
H’00000009	H’01～H’07	EB9 の範囲を消去（ERASE_N = H’01 設定時） EB9～EB15 の範囲を消去（ERASE_N = H’07 設定時）
H’0000000A	H’01～H’06	EB10 の範囲を消去（ERASE_N = H’01 設定時） EB10～EB15 の範囲を消去（ERASE_N = H’06 設定時）
H’0000000B	H’01～H’05	EB11 の範囲を消去（ERASE_N = H’01 設定時） EB11～EB15 の範囲を消去（ERASE_N = H’05 設定時）
H’0000000C	H’01～H’04	EB12 の範囲を消去（ERASE_N = H’01 設定時） EB12～EB15 の範囲を消去（ERASE_N = H’04 設定時）
H’0000000D	H’01～H’03	EB13 の範囲を消去（ERASE_N = H’01 設定時） EB13～EB15 の範囲を消去（ERASE_N = H’03 設定時）
H’0000000E	H’01～H’02	EB14 の範囲を消去（ERASE_N = H’01 設定時） EB14～EB15 の範囲を消去（ERASE_N = H’02 設定時）
H’0000000F	H’01	EB15 の範囲を消去（ERASE_N = H’01 設定時）

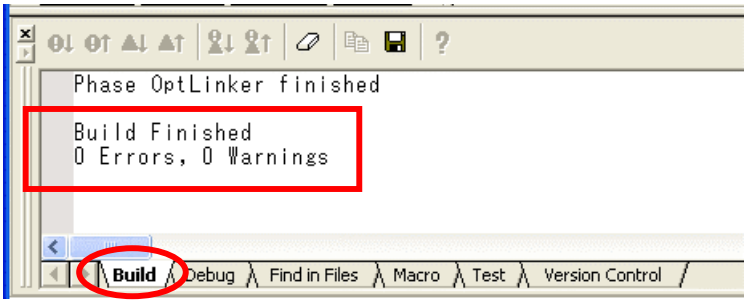
(4) 今回は、ブロック EB1～EB15 の範囲を消去します。BLOCK_NUM を“H’00000001”、ERASE_N を“H’0F”に設定してください。

```
51
52      BLOCK_NUM:      .ASSIGN H'00000001
53
54      /* Number of erase */
55      /* Note : An error occurs if a number ou
56      /* Note : ERASE_N max value(In case of BI
57      /* Note : ERASE_N max value(In case of BI
58      /* Example1 : BLOCK_NUM=H'00000003, ERASI
59      /* Example2 : BLOCK_NUM=H'00000001, ERASI
60      /*****************************************
61          BLOCK_NUM       ERASE_N
62          H'00000001      H'01 - H'0F
63          H'00000002      H'01 - H'0E
64          H'00000003      H'01 - H'0D
65          H'00000004      H'01 - H'0C
66          H'00000005      H'01 - H'0B
67          H'00000006      H'01 - H'0A
68          H'00000007      H'01 - H'09
69          H'00000008      H'01 - H'08
70          H'00000009      H'01 - H'07
71          H'0000000A      H'01 - H'06
72          H'0000000B      H'01 - H'05
73          H'0000000C      H'01 - H'04
74          H'0000000D      H'01 - H'03
75          H'0000000E      H'01 - H'02
76          H'0000000F      H'01
77      ********************************************
78      ERASE_N:        .ASSIGN H'0F
79
```

(5) パラメータを設定後、ビルド作業を行います。[ビルド]メニューの[すべてをビルド]を選択してください。

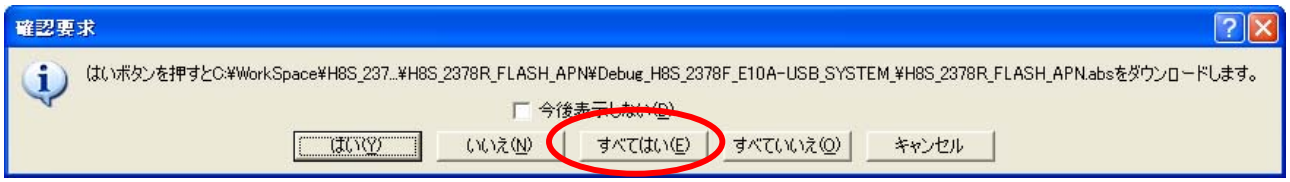


ビルド作業終了後、アウトプットウィンドウの[Build]タブ上の表示が“0 Errors, 0 Warnings”になることを確認してください。

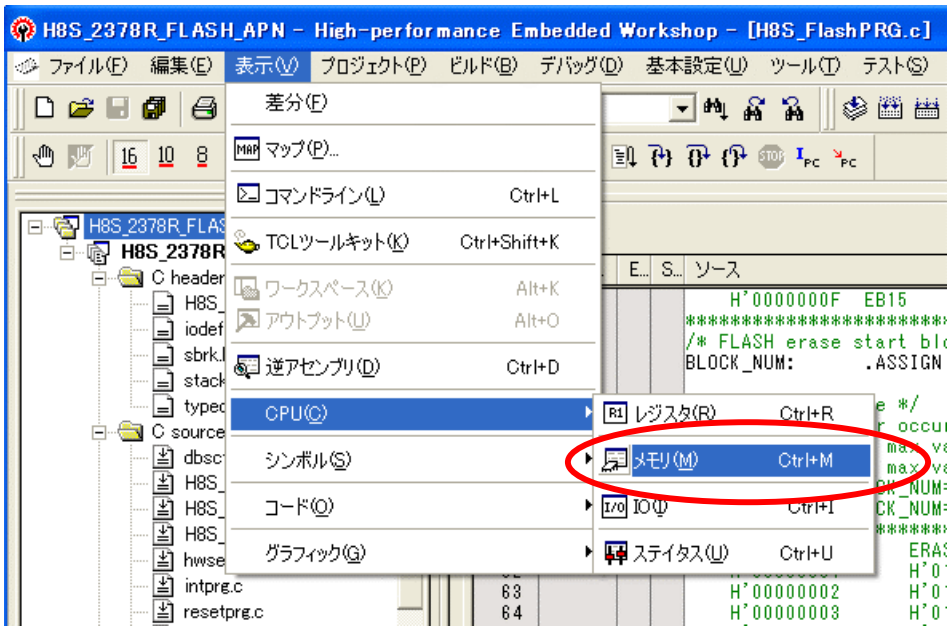


パラメータ変更後は必ずビルド作業をしてください。

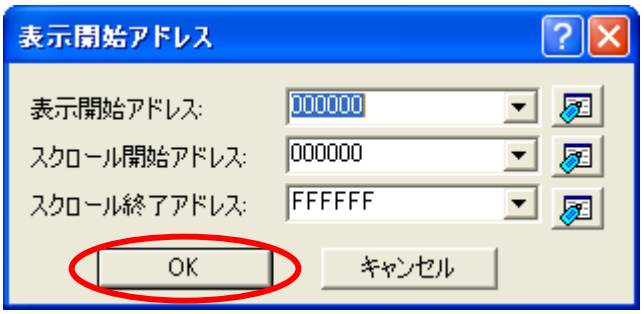
(6) ビルド作業終了後、[確認要求]ダイアログボックスが表示された場合は、[すべてはい]ボタンを押してプログラムをダウンロードしてください。[確認要求]ダイアログボックスが表示されない場合は、ワークスペースウィンドウ上の[Download modules]フォルダ内のファイル“H8S_2378R_FLASH_APN.abs - 00000000”をダブルクリックしてダウンロードしてください（本章の(1)を参照）。



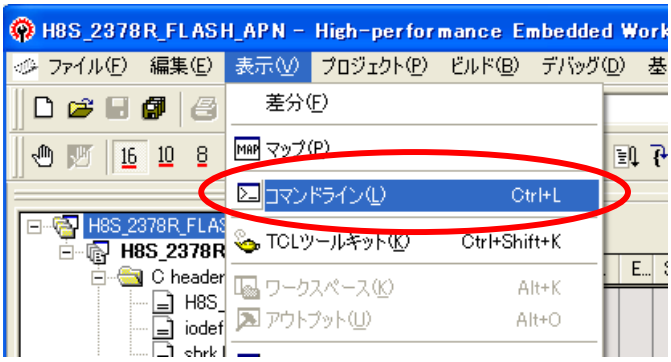
(7) サンプルプログラム実行後、フラッシュメモリの消去が行えているか確認するため、メモリウィンドウを開きます。[表示]メニューの[CPU]の中の[メモリ]を選択してください。



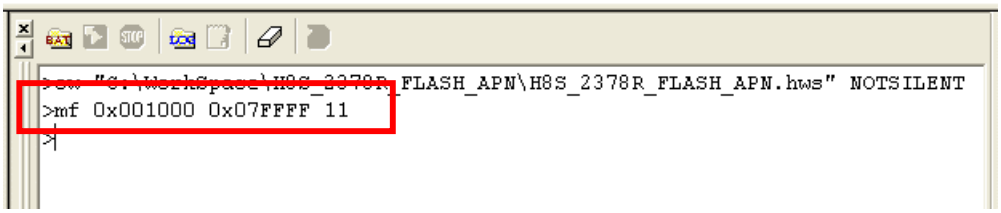
[表示開始アドレス]ダイアログボックスが表示されるので、デフォルトのまま変更せずに[OK]ボタンを押してください。



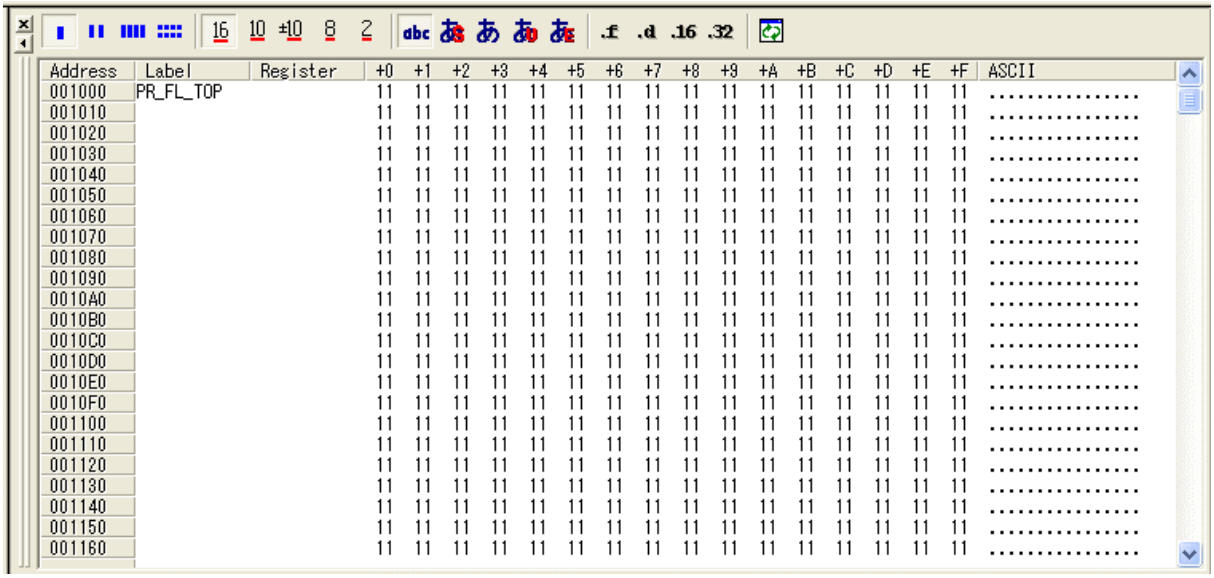
(8) フラッシュメモリの消去結果を分かりやすくするため、フラッシュメモリ内にあらかじめデータを書き込んでおきます。[表示]メニューの[コマンドライン]を選択してコマンドラインウィンドウを開いてください。



コマンドラインウィンドウで以下のように“mf 0x001000 0x07FFFF 11”と入力してコマンド実行してください。
“0x001000～0x07FFFF”は今回の消去対象ブロックのアドレスで、“11”は任意の値です。



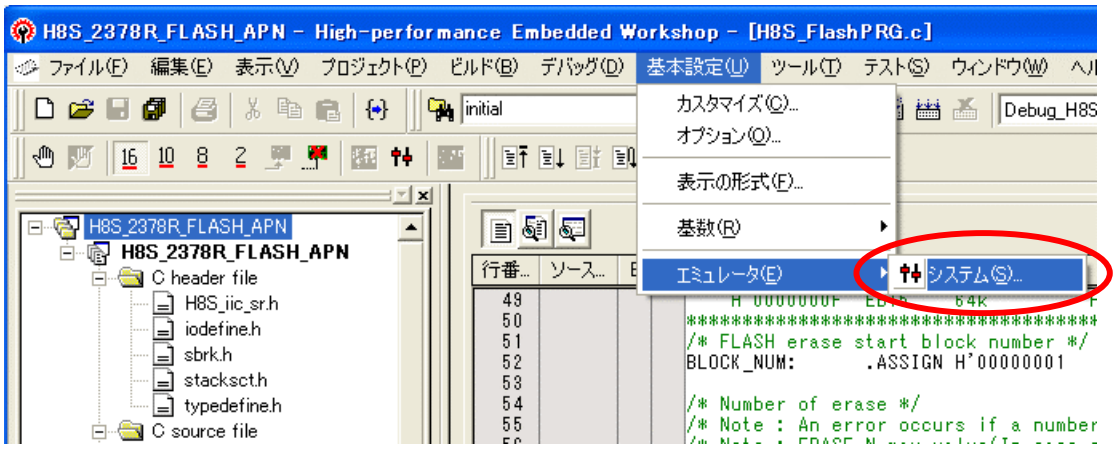
指定したアドレス範囲のデータが“11”に書き換わっていることをメモリウィンドウ上で確認してください。



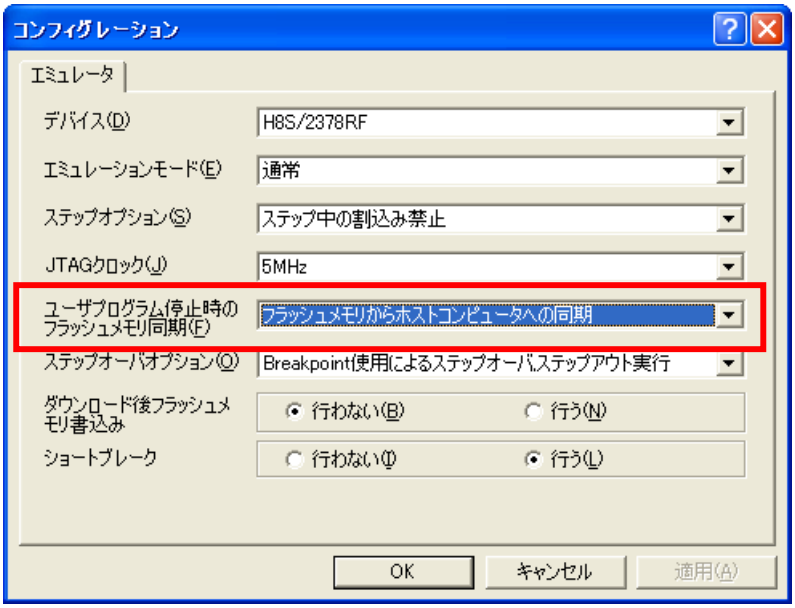
Address	Label	Register	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
001000	PR_FL_TOP		11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001010			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001020			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001030			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001040			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001050			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001060			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001070			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001080			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001090			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
0010A0			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
0010B0			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
0010C0			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
0010D0			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
0010E0			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
0010F0			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001100			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001110			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001120			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001130			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001140			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001150			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....
001160			11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	.....

(9) 次にフラッシュメモリからホストコンピュータへの同期をとります。この設定を行わないと書き込み/消去を行った後のフラッシュメモリの内容がメモリウィンドウ上に表示されません。

[基本設定]メニューの[エミュレータ]の中の[システム]を選択してください。

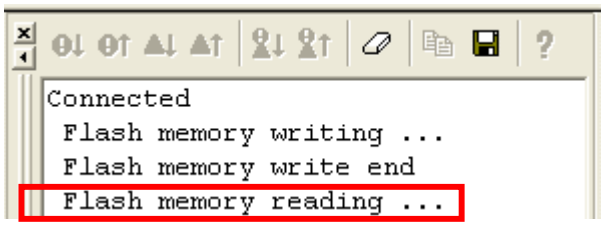


[コンフィグレーション]ダイアログボックスが表示されるので、[ユーザプログラム停止時のフラッシュメモリ同期]欄を“フラッシュメモリからホストコンピュータへの同期”に選択してください。



（参考）

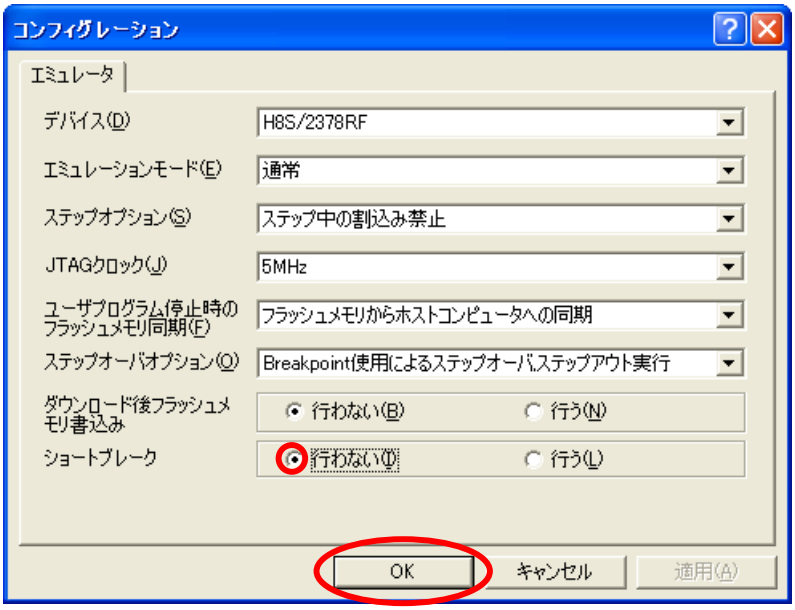
アウトプットウィンドウの[Debug]タブ上に“Flash memory reading ・・・”と表示されている間は、フラッシュメモリの内容がホストコンピュータ側（メモリウィンドウ上）に読み込まれている時です。



(10) 次に E10A-USB エミュレータがエミュレーション実行中にメモリアクセスを行わないように設定します。

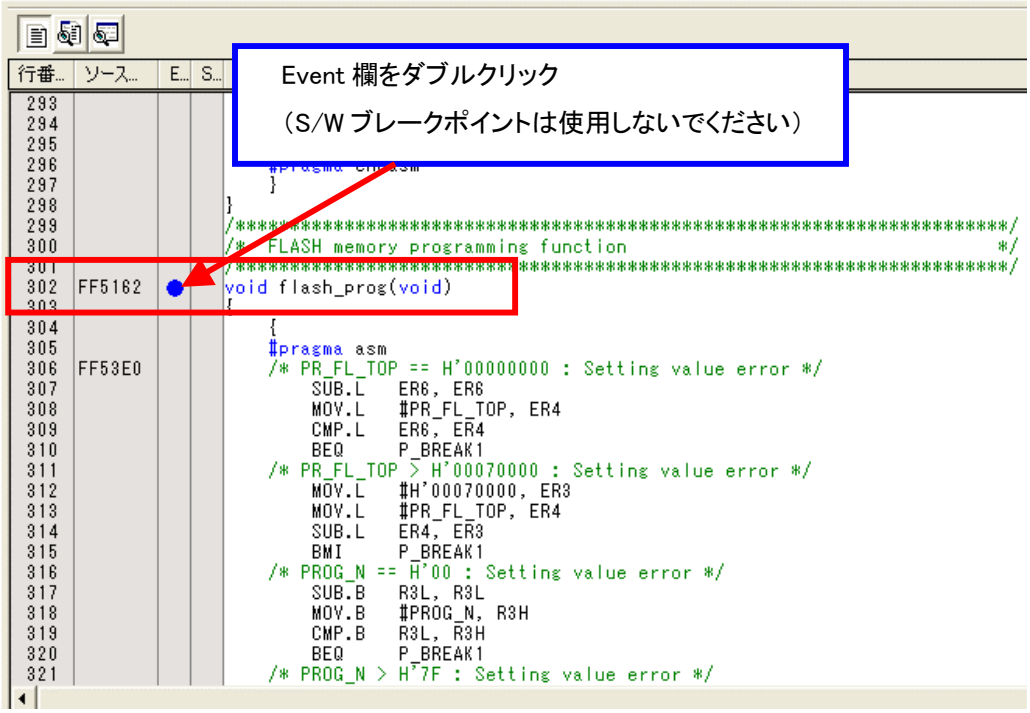
E10A-USB エミュレータは、エミュレーション実行中にメモリにアクセスするとショートブレーク（ユーザプログラムの一時停止）を行ってしまいます。その為、書き込み/消去処理中にフラッシュメモリにアクセスしてしまうような操作（メモリウィンドウのスクロール等）をすると処理に影響を与えてしまう可能性があります。

[コンフィグレーション]ダイアログボックスの[ショートブレーク]欄の[行わない]ラジオボタンにチェックを入れて[OK]ボタンを押してください。

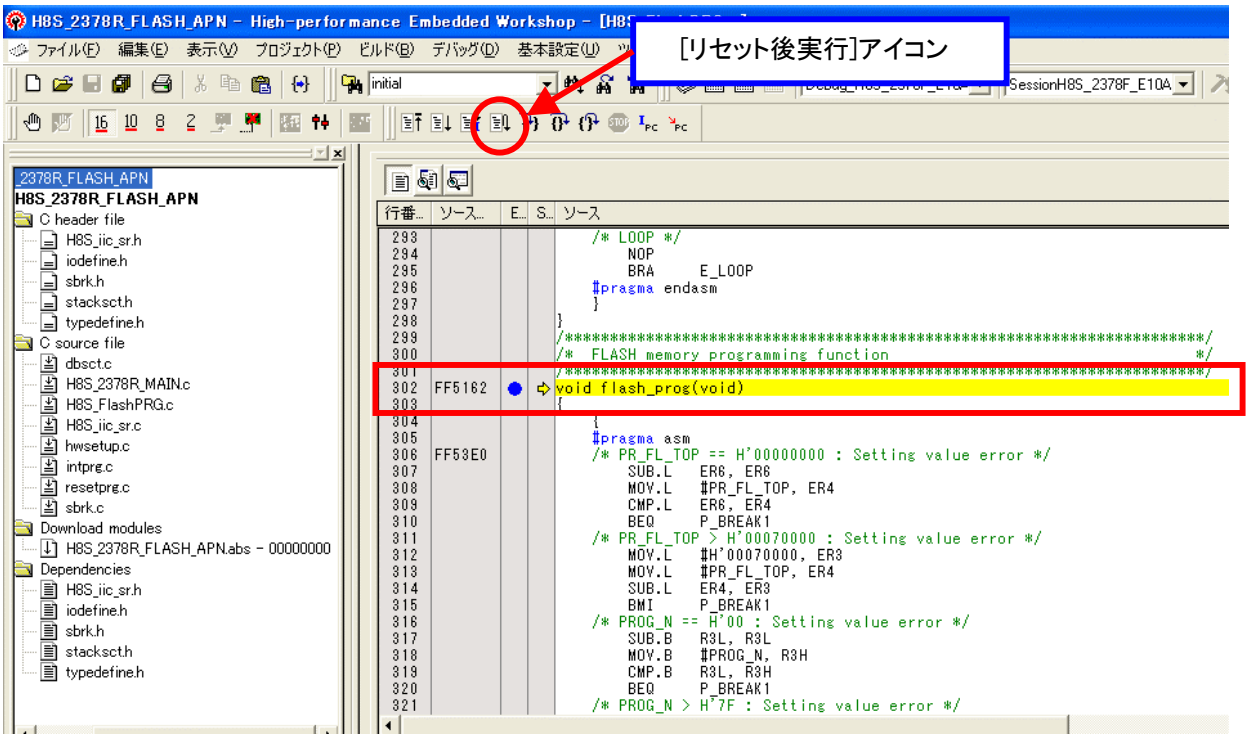


これでE10A-USBエミュレータがエミュレーション実行中にメモリアクセス（ショートブレーク）を行わなくなります。

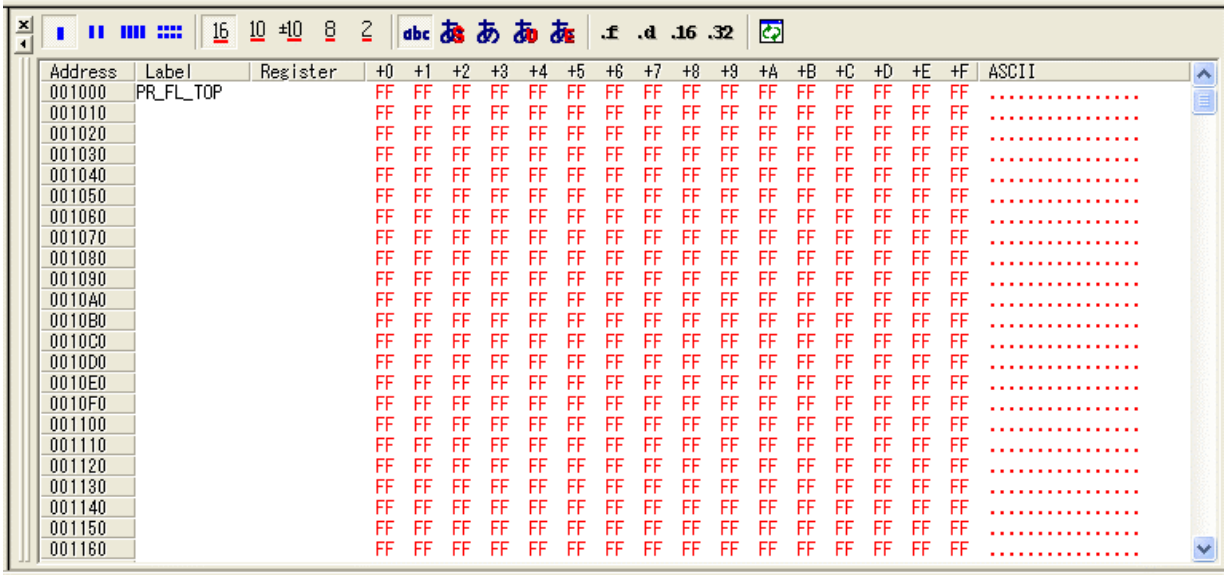
(11) サンプルプログラムを実行させてフラッシュメモリの消去を行います。ソースファイル“H8S_FlashPRG.c”内の以下の箇所（行番号 302）にブレークを設定してください。



(12) [リセット後実行]アイコンをクリックしてサンプルプログラムを実行させてください。しばらく待つとブレークを設定した箇所でプログラムが停止します。1 分程経ってもプログラムが停止しない場合は、エラーが発生している可能性があるので強制停止([停止]アイコンをクリック)してエラー内容を確認してください。エラーについては「6.4 章　フラッシュメモリ消去エラーチェック方法」をご参照ください。



(13) メモリウィンドウを確認すると消去対象に設定したブロック EB1～EB15(H'001000 ～H'07FFFF)の範囲のデータが“FF”に書き換わり、フラッシュメモリの消去が行われたことが確認できます。



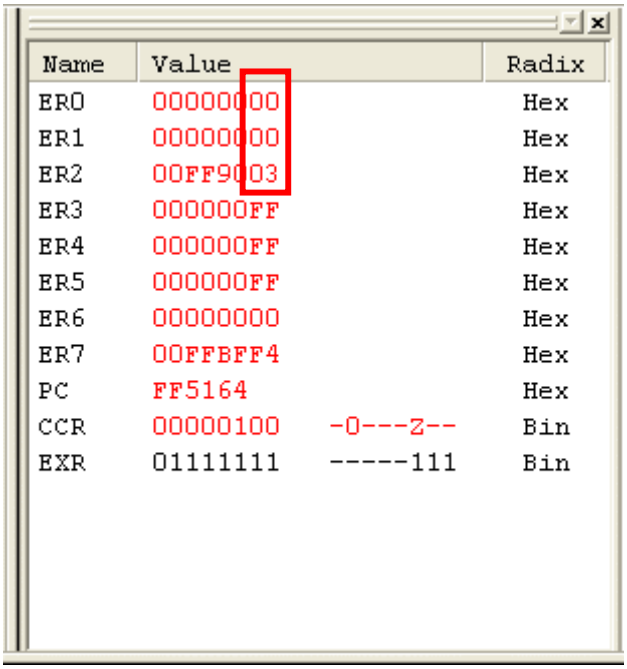
（参考）

フラッシュメモリの消去が正常（エラーなし）に行われた場合には、レジスタウィンドウに以下の値が書き込まれます。

・R0L（DPFR の戻り値） → H’00（消去プログラムが正常にダウンロードされたことを表している）

・R1L（FPFR の戻り値） → H’00（初期化、消去の各処理が正常に行われたことを表している）

・R2L（RTN の値） → H’03（ダウンロード、初期化、消去の各処理が終了したことを表している）



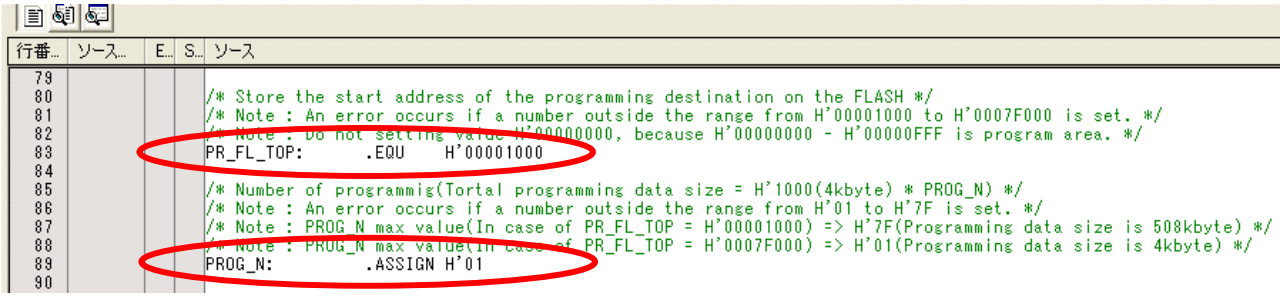
Name	Value		Radix
ER0	00000000		Hex
ER1	00000000		Hex
ER2	00FF9003		Hex
ER3	000000FF		Hex
ER4	000000FF		Hex
ER5	000000FF		Hex
ER6	00000000		Hex
ER7	00FFBFF4		Hex
PC	FF5164		Hex
CCR	00000100	-0---Z--	Bin
EXR	01111111	-----111	Bin

フラッシュメモリ消去時のエラーチェック方法の詳細は、「6.4 章　フラッシュメモリ消去エラーチェック方法」で説明します。

6.3.2 フラッシュメモリの書き込み

(1) 最初にサンプルプログラムをダウンロードし、ソースファイル“H8S_FlashPRG.c”をエディタウィンドウに表示させてください（6.3.1 章の(1)、(2)を参照）。また、あらかじめデータを書き込む領域は消去を行ってください（6.3.1 章フラッシュメモリの消去を参照）。

フラッシュメモリの書き込みを行う上で必要なパラメータ“PR_FL_TOP（書き込み先の先頭アドレス）”、“PROG_N（書き込み回数）”の設定を行います。各パラメータは、“H8S_FlashPRG.c”ファイル内の“flash_symbol”関数内にあります。



PR_FL_TOP は、データを書き始めるフラッシュメモリの先頭アドレスを設定するパラメータです。設定範囲は“H’00001000～H’0007F000”です。このサンプルプログラムでは、“H’00000000～H’00000FFF”をプログラム領域として使用しているため書き込めない仕様となっています。

PROG_N は、IIC バスインタフェース 2 のチャネル 0（IIC2_0）を使用して受信する 4k バイトのデータ（内蔵 RAM のアドレス“H’FF8000～H’FF8FFF”の領域に格納されるデータ）をフラッシュメモリに何回書き込むか設定するパラメータで、最大 508k バイトまで書き込むことができます。設定範囲は PR_FL_TOP の値に依存しており、詳しくは「表 6.3　PR_FL_TOP と PROG_N の関係」をご参照ください。

どちらのパラメータも、設定範囲以外の値を設定してサンプルプログラムを実行するとエラーが発生します。

表 6.3　PR_FL_TOP と PROG_N の関係

PR_FL_TOP の設定値	PROG_N の設定範囲		書き込み内容	
	最小値	最大値	開始アドレス	書き込むデータサイズ
H’00001000	H’01	H’7F	H’001000	4k バイト － 508k バイト
H’00002000	H’01	H’7E	H’002000	4k バイト － 504k バイト
H’00004000	H’01	H’7C	H’004000	4k バイト － 496k バイト
H’00006000	H’01	H’7A	H’006000	4k バイト － 488k バイト
H’00008000	H’01	H’78	H’008000	4k バイト － 480k バイト
H’0000A000	H’01	H’76	H’00A000	4k バイト － 472k バイト
H’0000C000	H’01	H’74	H’00C000	4k バイト － 464k バイト
H’0000E000	H’01	H’72	H’00E000	4k バイト － 456k バイト
H’00010000	H’01	H’70	H’010000	4k バイト － 448k バイト
H’00020000	H’01	H’60	H’020000	4k バイト － 384k バイト
H’00030000	H’01	H’50	H’030000	4k バイト － 320k バイト
H’00040000	H’01	H’40	H’040000	4k バイト － 256k バイト
H’00050000	H’01	H’30	H’050000	4k バイト － 192k バイト
H’00060000	H’01	H’20	H’060000	4k バイト － 128k バイト
H’00070000	H’01	H’10	H’070000	4k バイト － 64k バイト
H’0007F000	－	H’01	H’07F000	4k バイト

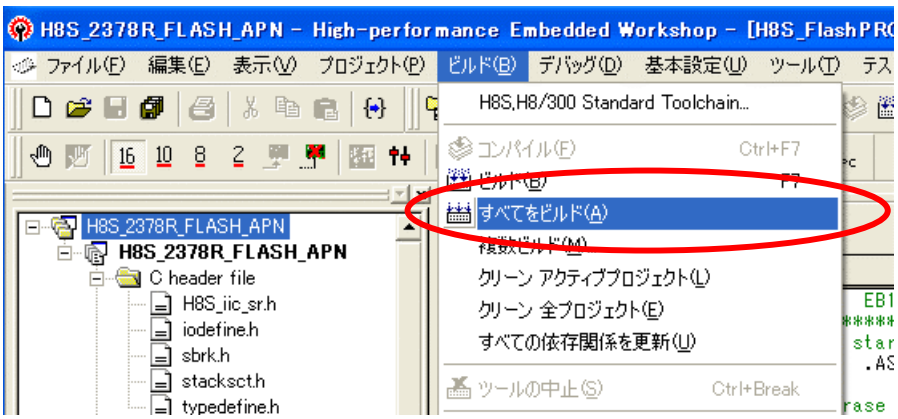
(2) 今回は、H’001000 番地から 4k バイト分のデータ書き込みを行います。PR_FL_TOP を“H’00001000”、PROG_N を“H’01”に設定してください。

```
79
80          /* Store the start address of the programming destination on the FLASH */
81          /* Note : An error occurs if a number outside the range from H'00001000 to H'0007F000 is set. */
82          /* Note : Do not setting value H'00000000, because H'00000000 - H'00000FFF is program area. */
83          PR_FL_TOP:      .EQU    H'00001000
84
85          /* Number of programmig(Tortal programming data size = H'1000(4kbyte) * PROG_N) */
86          /* Note : An error occurs if a number outside the range from H'01 to H'7F is set. */
87          /* Note : PROG_N max value(In case of PR_FL_TOP = H'00001000) => H'7F(Programming data size is 508kbyte) */
88          /* Note : PROG_N max value(In case of PR_FL_TOP = H'0007F000) => H'01(Programming data size is 4kbyte) */
89          PROG_N:         .ASSIGN H'01
90
```

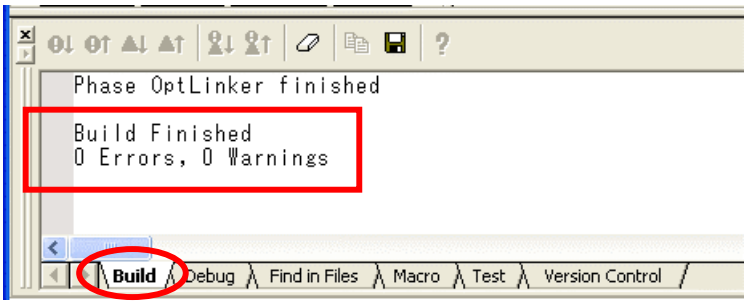
(3) フラッシュメモリの書き込みのみ行うため、“falash_erpr_128”関数内にある“flash_erase( )”（フラッシュメモリ消去関数）をコメントアウトします。

```
行番... ソース... E... S... ソース
124
125         /*  FLASH erase/programming function                                    */
126         /**************************************************************************/
127         #pragma section ROMPRG
128 FF5000  void flash_erpr_128(void)
129         {
130         //  flash_erase();                                  //FLASH erase function
131             flash_prog();                                   //FLASH programming function
132
133             while(1){}
134         }
```

(4) ビルド作業を行います。[ビルド]メニューの[すべてをビルド]を選択してください。

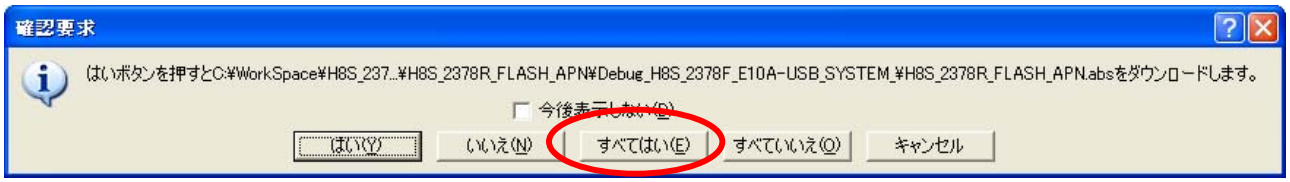


ビルド作業終了後、アウトプットウィンドウの[Build]タブ上の表示が“0 Errors, 0 Warnings”になることを確認してください。

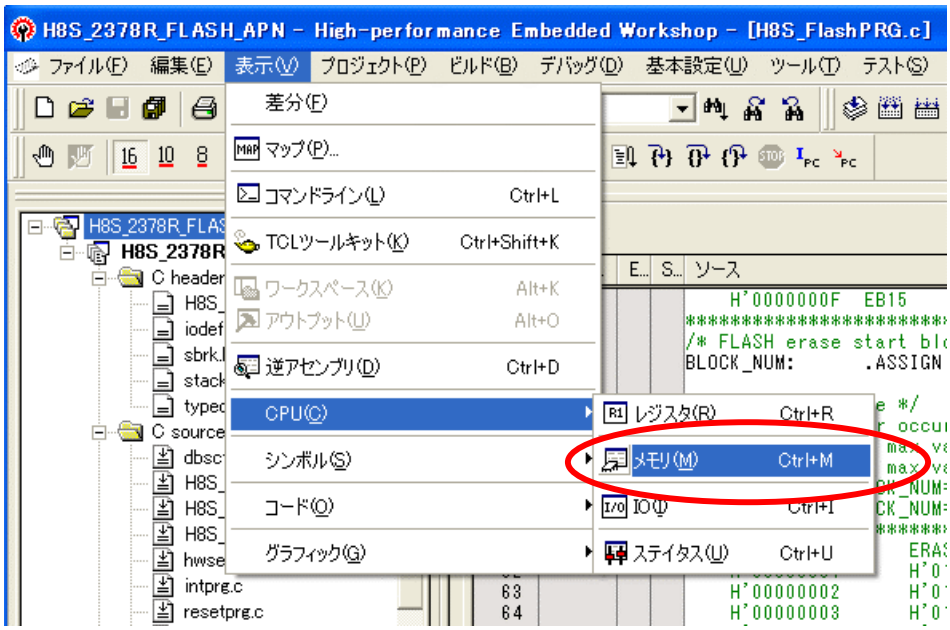


パラメータ変更後もしくはコメントアウト後は必ずビルド作業をしてください。

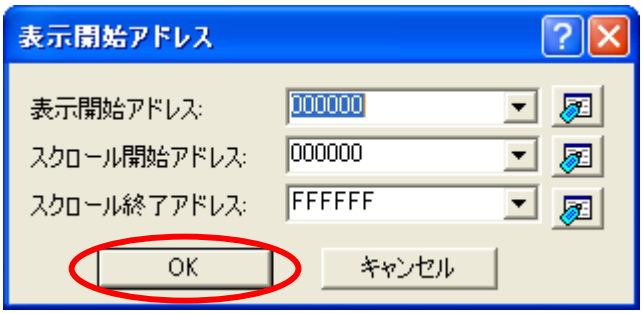
(5) ビルド作業終了後、[確認要求]ダイアログボックスが表示された場合は、[すべてはい]ボタンを押してプログラムをダウンロードしてください。[確認要求]ダイアログボックスが表示されない場合は、ワークスペースウィンドウ上の[Download modules]フォルダ内のファイル“H8S_2378R_FLASH_APN.abs - 00000000”をダブルクリックしてダウンロードしてください（6.3.1 章の(1)を参照）。



(6) サンプルプログラム実行後、フラッシュメモリの書き込みが行えているか確認するため、メモリウィンドウを開きます。[表示]メニューの[CPU]の中の[メモリ]を選択してください。

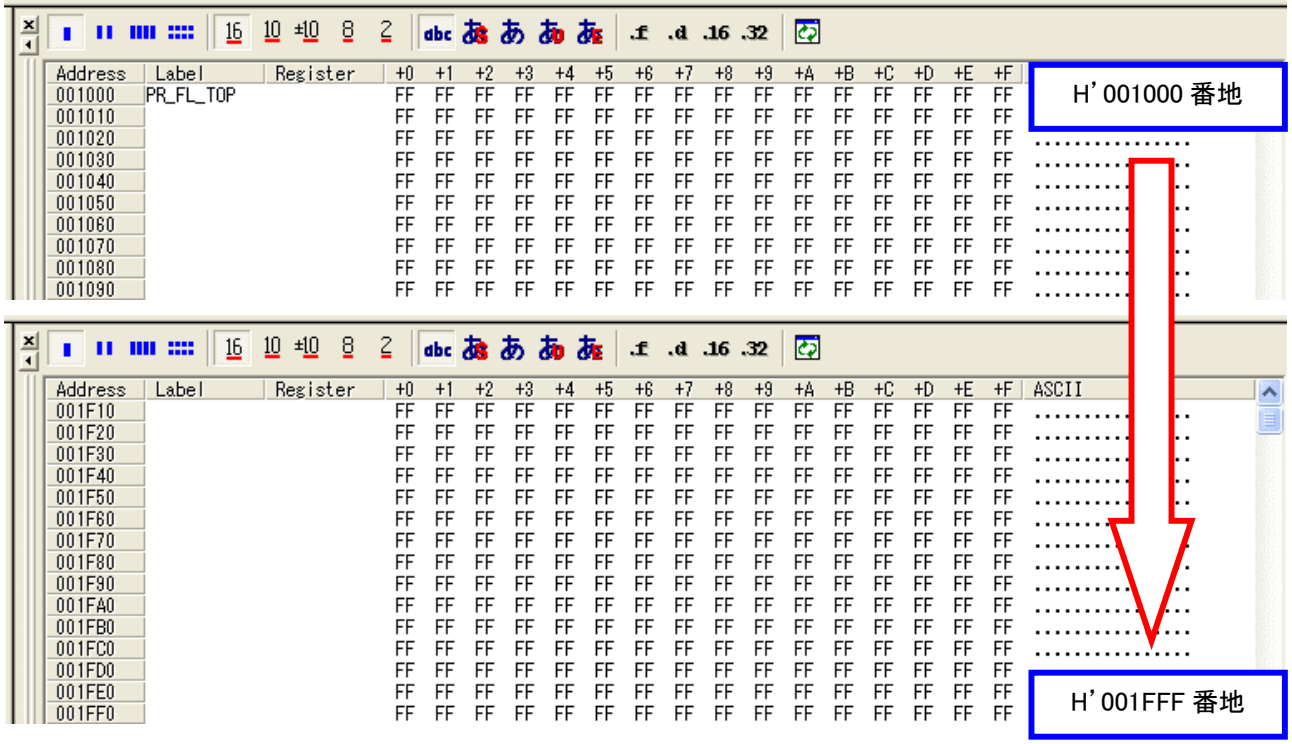


[表示開始アドレス]ダイアログボックスが表示されるので、デフォルトのまま変更せずに[OK]ボタンを押してください。



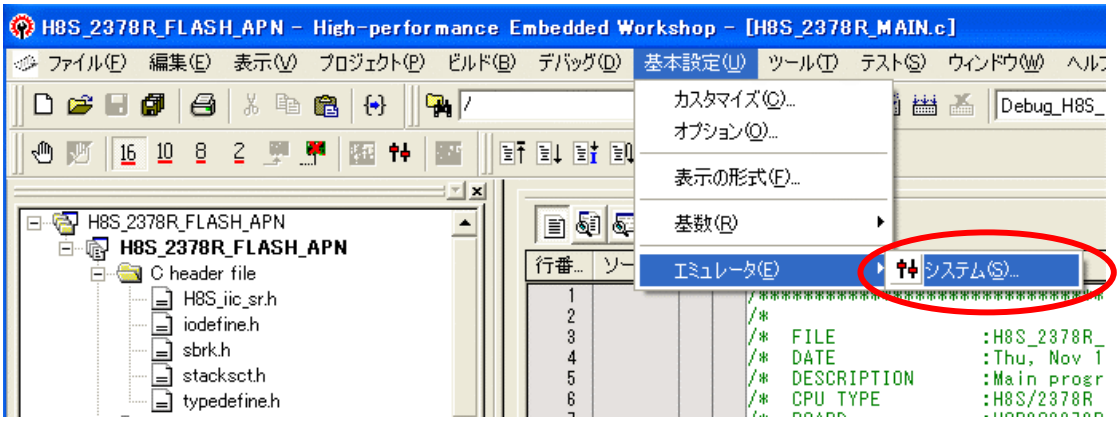
(7) フラッシュメモリの書き込みを始める前に、書き込もうとしている領域が消去（オール“H’FF”）されていることを確認してください。“H’FF”以外の値が書き込まれている場合は、まずフラッシュメモリの消去を行ってください（6.3.1 章 フラッシュメモリの消去を参照）。

今回は、H’001000 番地から 4k バイト分のデータ書き込みを行うので、アドレス“H’001000～H’001FFF”までがオール“H’FF”になっていることをメモリウィンドウで確認します。

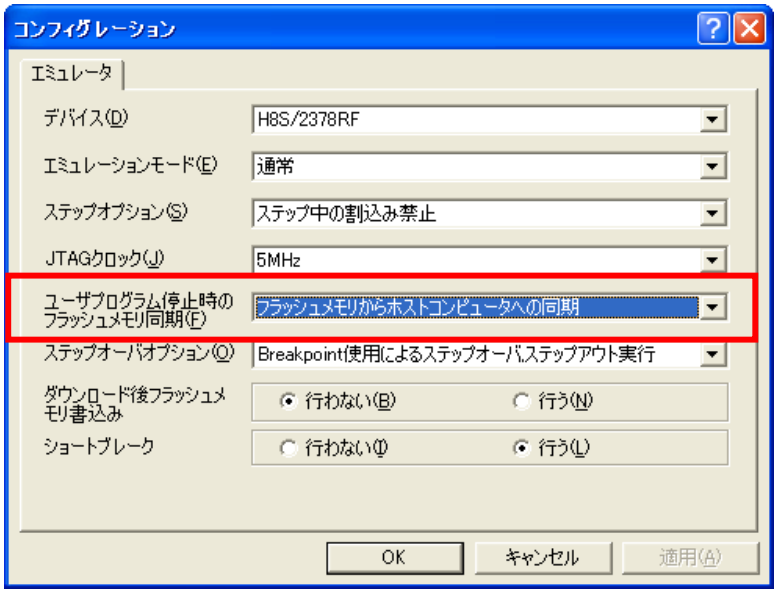


(8) 次にフラッシュメモリからホストコンピュータへの同期をとります。この設定を行わないと書き込み/消去を行った後のフラッシュメモリの内容がメモリウィンドウ上に表示されません。

[基本設定]メニューの[エミュレータ]の中の[システム]を選択してください。

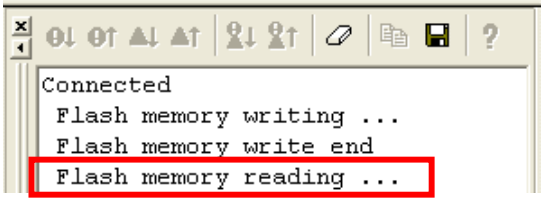


[コンフィグレーション]ダイアログボックスが表示されるので、[ユーザプログラム停止時のフラッシュメモリ同期]欄を“フラッシュメモリからホストコンピュータへの同期”に選択してください。



（参考）

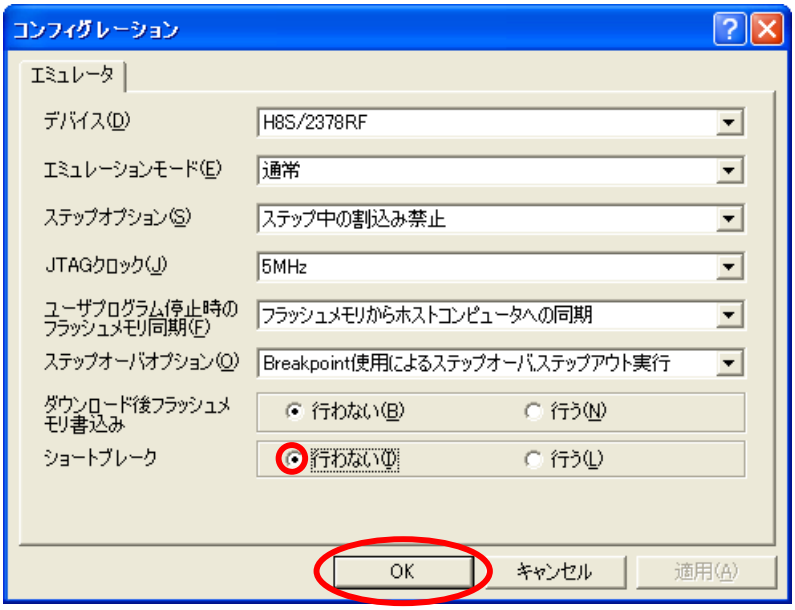
アウトプットウィンドウの[Debug]タブ上に“Flash memory reading ・・・”と表示されている間は、フラッシュメモリの内容がホストコンピュータ側（メモリウィンドウ上）に読み込まれている時です。



(14) 次に E10A-USB エミュレータがエミュレーション実行中にメモリアクセスを行わないように設定します。

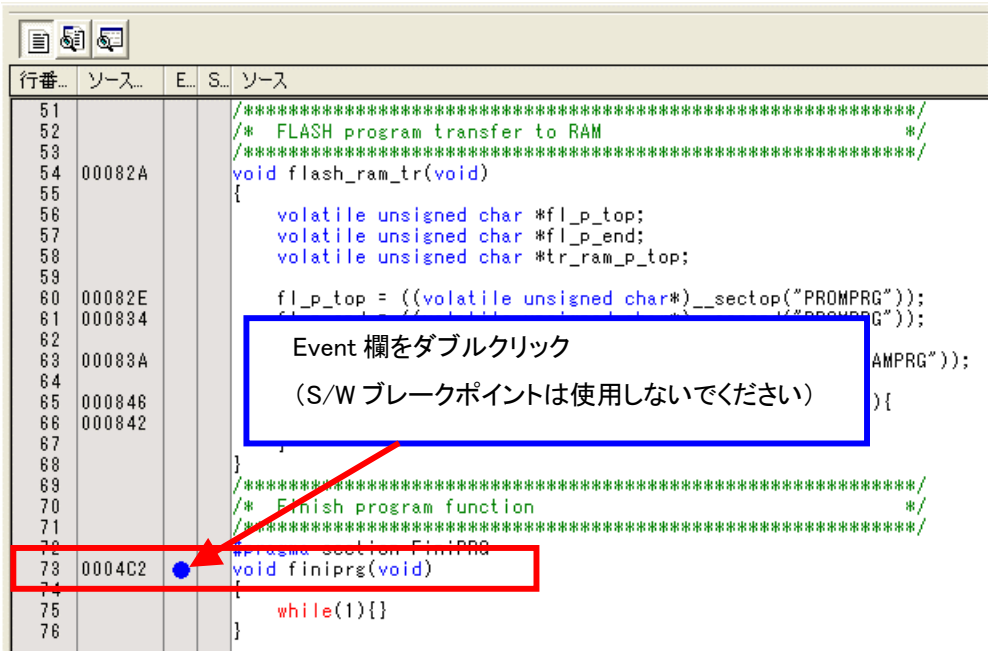
E10A-USB エミュレータは、エミュレーション実行中にメモリにアクセスするとショートブレーク（ユーザプログラムの一時停止）を行ってしまいます。その為、書き込み/消去処理中にフラッシュメモリにアクセスしてしまうような操作（メモリウィンドウのスクロール等）をすると処理に影響を与えてしまう可能性があります。

[コンフィグレーション]ダイアログボックスの[ショートブレーク]欄の[行わない]ラジオボタンにチェックを入れて[OK]ボタンを押してください。

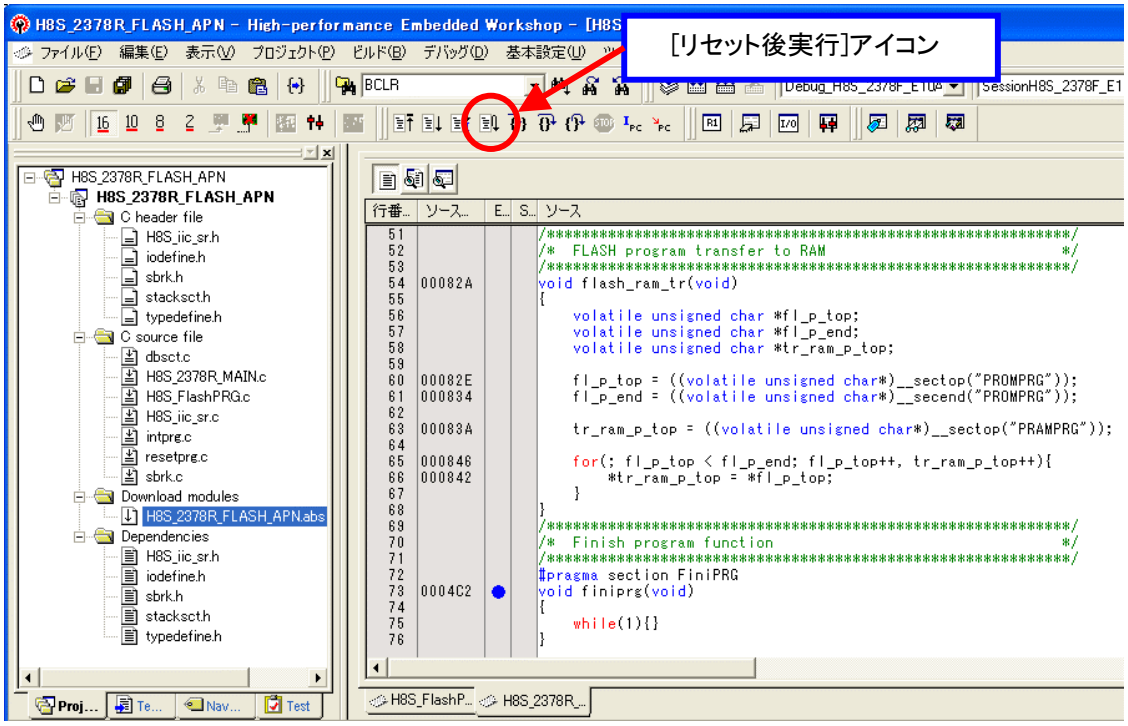


これでE10A-USB エミュレータがエミュレーション実行中にメモリアクセス（ショートブレーク）を行わなくなります。

(9) ソースファイル“H8S_2378R_MAIN.c”をエディタウィンドウに表示させて、以下のように“H8S_2378R_MAIN.c”ファイル内の“finiprg”関数の先頭（行番号 73）にブレークを設定してください。



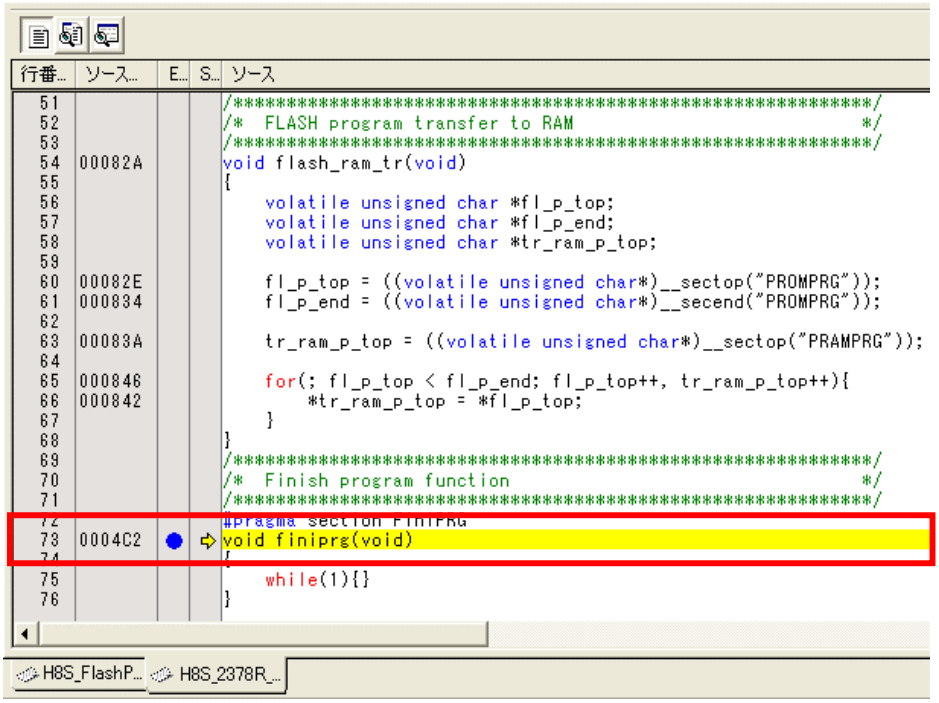
(10) [リセット後実行]アイコンをクリックしてサンプルプログラムを実行させてください。



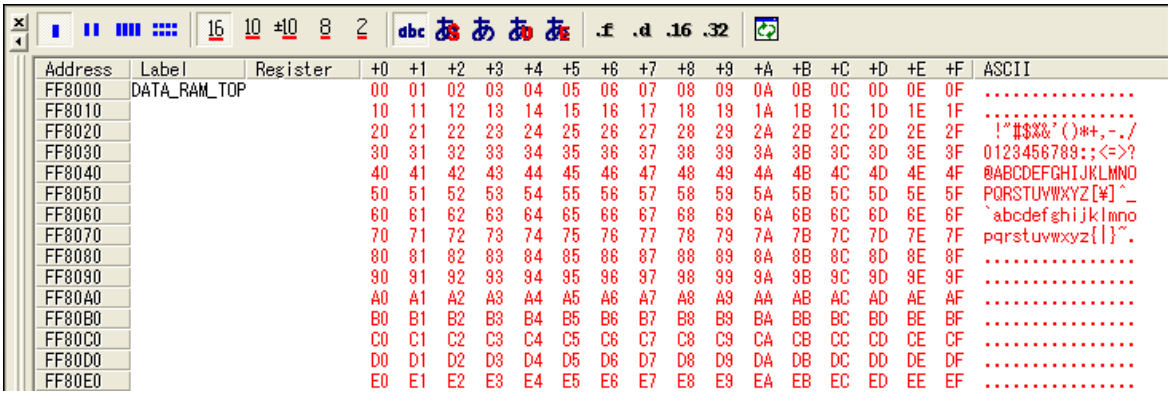
サンプルプログラムの動作として、書き込みプログラムのダウンロード、フラッシュメモリの初期化処理が終わると IIC バスインタフェース 2 のチャネル 0(IIC2_0)によるスレーブ受信モードに入り、フラッシュメモリに書き込むデータが IIC_マスタ送信側から送信されるまで待機状態になります。フラッシュメモリに書き込むデータ(今回は 4k バイト)を送信してください。

なお、フラッシュメモリに書き込むデータを送信する(IIC_マスタ送信側)プログラムは、参考例として「付録 IIC_マスタ送信側プログラム」に記載しています。

(11) IIC_マスタ送信側からデータ送信を行った後、H8S/2378R 側で正常にデータが受信できた場合はブレークを設定した箇所でプログラムが停止します。



受信したデータは、内蔵 RAM 領域（“H’FF8000”～“H’FF8FFF”）に書き込まれます。本書の例では“H’00”～“H’FF”のデータを繰り返し 4k バイト分送信しています。



Address	Label	Register	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
FF8000	DATA_RAM_TOP		00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	.....
FF8010			10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	.....
FF8020			20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	!"#$%&'()*+,-./
FF8030			30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	0123456789:;<=>?
FF8040			40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	@ABCDEFGHIJKLMNO
FF8050			50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	PQRSTUVWXYZ[¥]^_
FF8060			60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	`abcdefghijklmno
FF8070			70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	pqrstuvwxyz{|}~.
FF8080			80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	.....
FF8090			90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	.....
FF80A0			A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	.....
FF80B0			B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	.....
FF80C0			C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	.....
FF80D0			D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	.....
FF80E0			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	.....

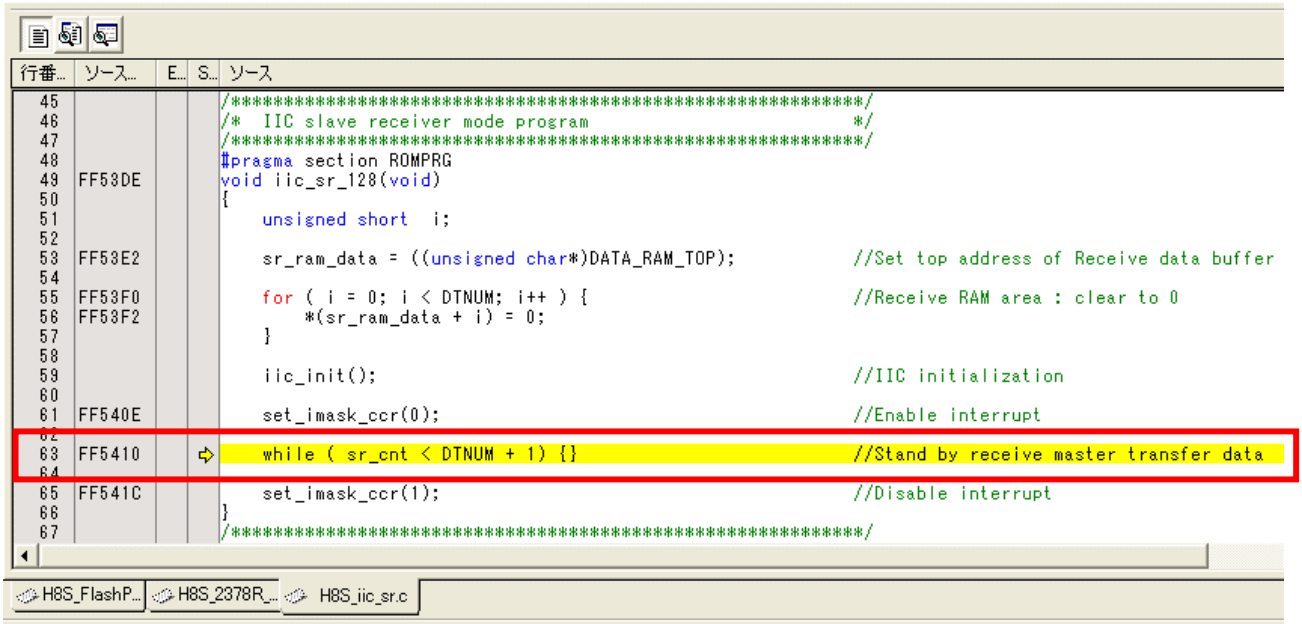
しばらく待ってもプログラムが停止しない場合は、エラーが発生している可能性があるので強制停止（[停止]アイコンをクリック）してエラー内容を確認してください。エラーについては「6.5 章 フラッシュメモリ書き込みエラーチェック方法」をご参照ください。

（書き込みを行った場合の待ち時間目安）

注：IIC スレーブ受信処理時間も含んでいますので、IIC マスタ送信側の性能によって変化します。

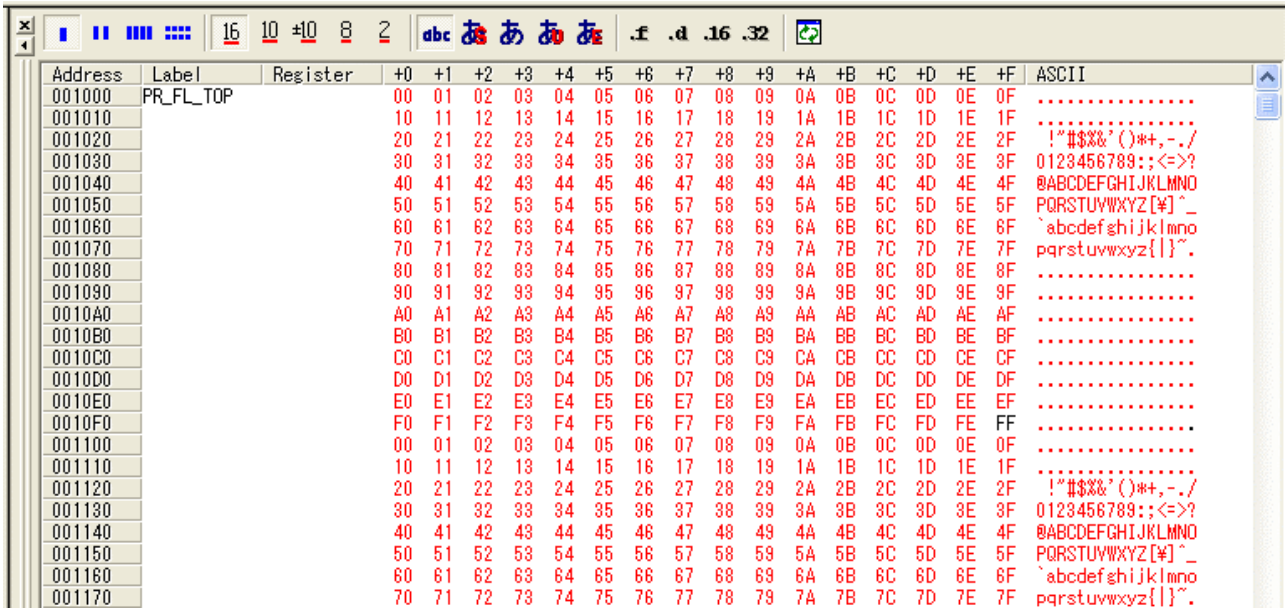
- 4k バイトのデータ書き込み時 → 30 秒程度
- 508k バイトのデータ書き込み時 → 3 分程度

強制停止（[停止]アイコンをクリック）して、以下の箇所（ソースファイル“H8S_iic_sr.c”の“iic_sr_128”関数内の行番号 63）でプログラムが停止した場合は、IIC の送受信が正常に行われていません（送信データ待ち状態のまま）。IIC 関連の設定（スレーブアドレス、データサイズ、転送速度等）を見直してください。



```
行番... ソース... E... S... ソース
45                          /*****************************************************************/
46                          /*  IIC slave receiver mode program                              */
47                          /*****************************************************************/
48                          #pragma section ROMPRG
49  FF53DE                  void iic_sr_128(void)
50                          {
51                              unsigned short  i;
52
53  FF53E2                      sr_ram_data = ((unsigned char*)DATA_RAM_TOP);       //Set top address of Receive data buffer
54
55  FF53F0                      for ( i = 0; i < DTNUM; i++ ) {                    //Receive RAM area : clear to 0
56  FF53F2                          *(sr_ram_data + i) = 0;
57                              }
58
59                              iic_init();                                        //IIC initialization
60
61  FF540E                      set_imask_ccr(0);                                  //Enable interrupt
62
63  FF5410                      while ( sr_cnt < DTNUM + 1) {}                     //Stand by receive master transfer data
64
65  FF541C                      set_imask_ccr(1);                                  //Disable interrupt
66                          }
67                          /*****************************************************************/
```

(12) メモリウィンドウを確認すると IIC_マスタ送信側から送信したデータ（内臓 RAM に格納されたデータ）が“H’001000”番地から“H’001FFF”番地まで 4k バイト分、フラッシュメモリに書き込まれていることが確認できます。



Address	Label	Register	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
001000	PR_FL_TOP		00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	.....
001010			10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	.....
001020			20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	!"#$%&'()*+,-./
001030			30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	0123456789:;<=>?
001040			40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	@ABCDEFGHIJKLMNO
001050			50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	PQRSTUVWXYZ[¥]^_
001060			60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	`abcdefghijklmno
001070			70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	pqrstuvwxyz{|}~.
001080			80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	.....
001090			90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	.....
0010A0			A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	.....
0010B0			B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	.....
0010C0			C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	.....
0010D0			D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	.....
0010E0			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	.....
0010F0			F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	.....
001100			00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	.....
001110			10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	.....
001120			20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	!"#$%&'()*+,-./
001130			30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	0123456789:;<=>?
001140			40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	@ABCDEFGHIJKLMNO
001150			50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	PQRSTUVWXYZ[¥]^_
001160			60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	`abcdefghijklmno
001170			70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	pqrstuvwxyz{|}~.

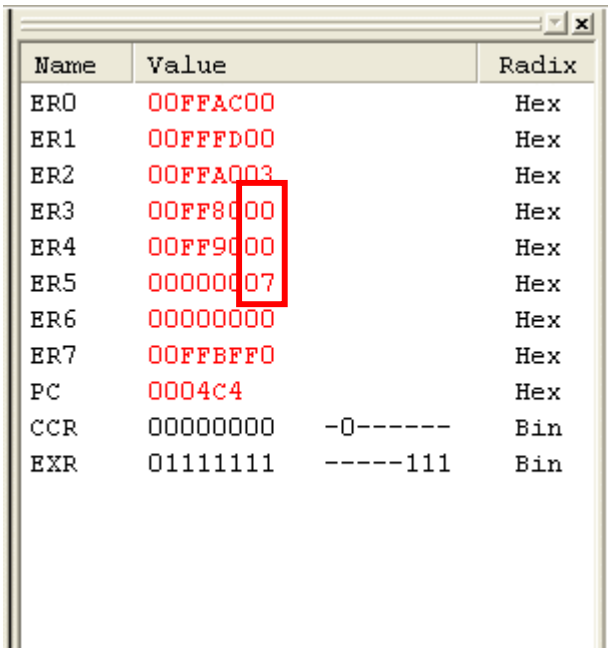
（参考）

フラッシュメモリの書き込みが正常（エラーなし）に行われた場合には、レジスタウィンドウに以下の値が書き込まれます。

・R3L（DPFR の戻り値） → H’00（書き込みプログラムが正常にダウンロードされたことを表している）

・R4L（FPFR の戻り値） → H’00（初期化、書き込み、書き込み終了の各処理が正常に行われたことを表している）

・R5L（RTN の値） → H’07（ダウンロード、初期化、書き込み、書き込み終了の各処理が終了したことを表している）



Name	Value		Radix
ER0	00FFAC00		Hex
ER1	00FFFD00		Hex
ER2	00FFA003		Hex
ER3	00FF8000		Hex
ER4	00FF9000		Hex
ER5	00000007		Hex
ER6	00000000		Hex
ER7	00FFBFF0		Hex
PC	0004C4		Hex
CCR	00000000	-0------	Bin
EXR	01111111	-----111	Bin

フラッシュメモリ書き込み時のエラーチェック方法の詳細は、「6.5 章　フラッシュメモリ書き込みエラーチェック方法」で説明します。

6.4　フラッシュメモリ消去エラーチェック方法

本章では、サンプルプログラムを使用してフラッシュメモリの消去時（消去プログラムのダウンロード処理、初期化処理、消去処理）にエラーが発生した場合のチェック方法を説明します。図 6.3 にエラーチェック方法のフローチャート図を示します。

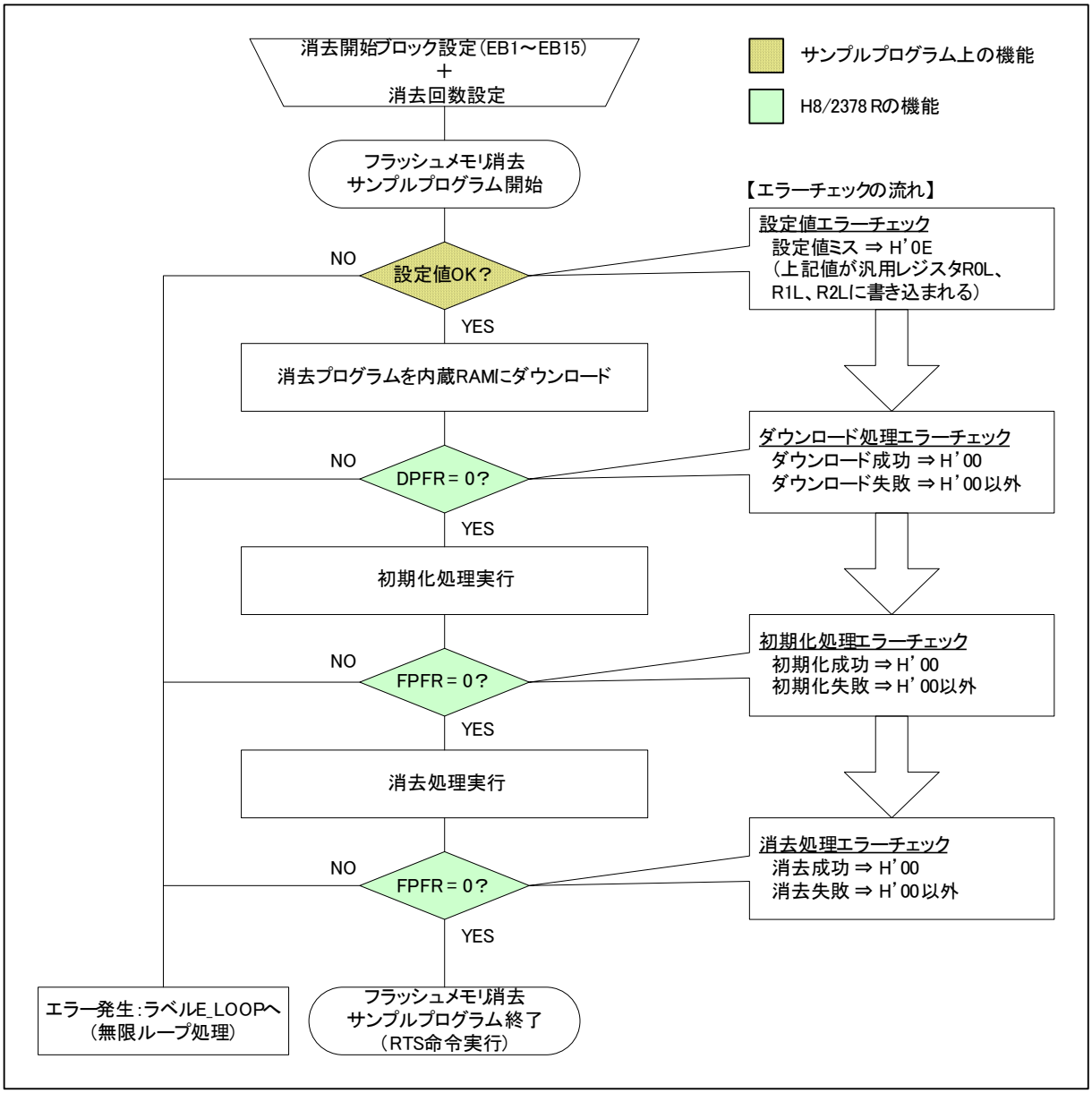


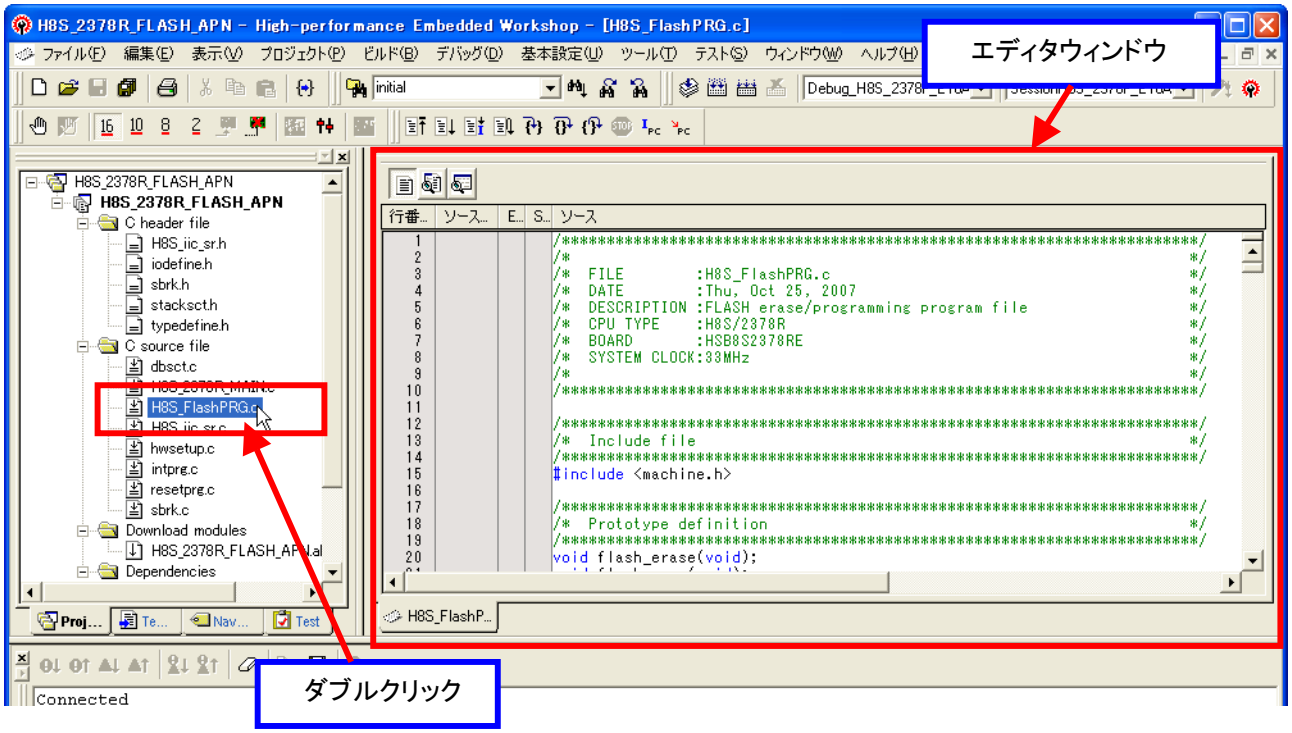
図 6.3　フラッシュメモリ消去時のエラーチェックフローチャート図

本書で記載しているエラーチェックのフローチャート図では、H8/2378R が搭載している機能とサンプルプログラム上の機能とで色分けしてあります。

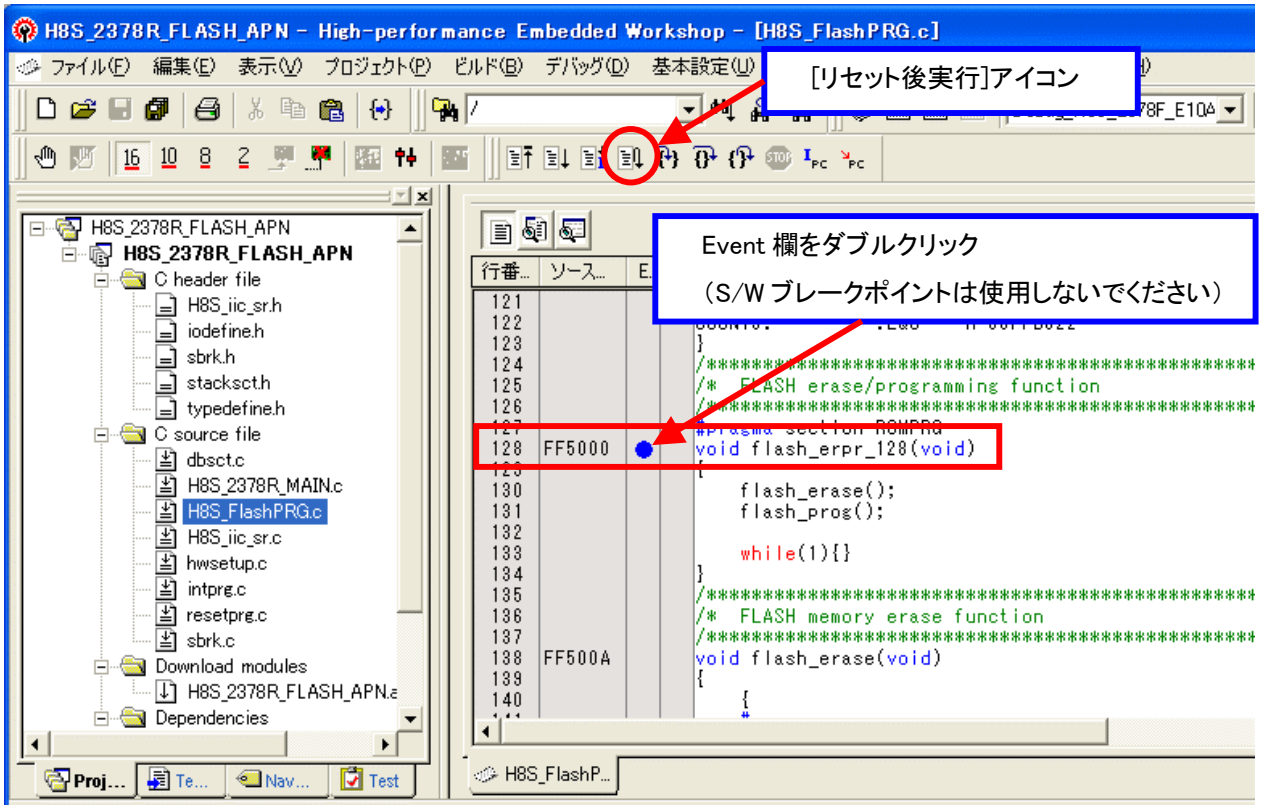
サンプルプログラム上の機能（独自のエラー検出手順）

H8S/2378R の機能（マイコンに内蔵されている書き込み/消去プログラムの戻り値によるエラー検出手順）

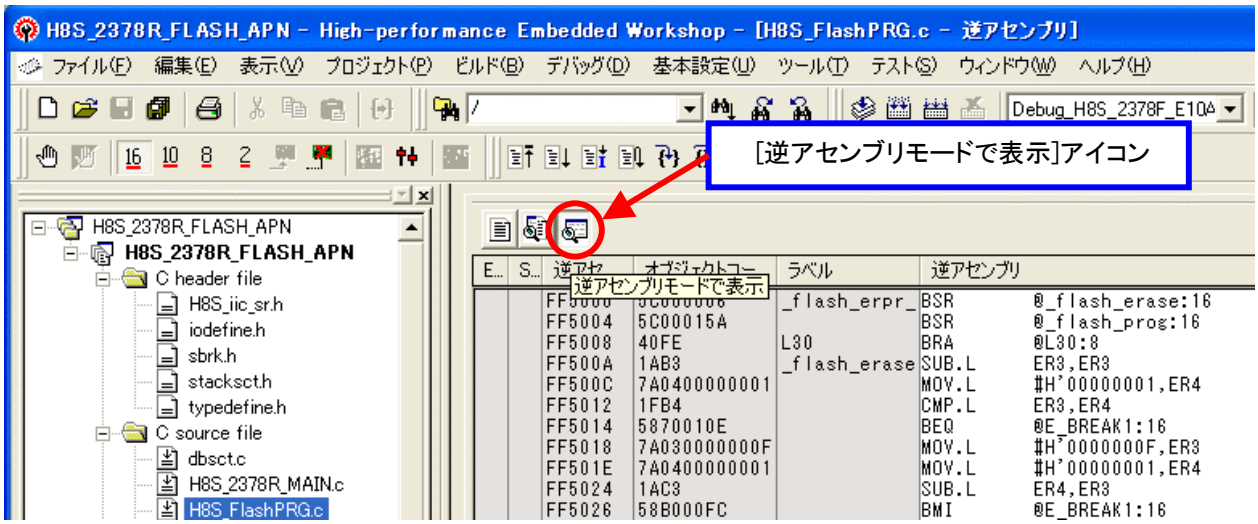
(1) ワークスペースウィンドウ上の[C source file]フォルダ内のファイル“H8S_FlashPRG.c”をダブルクリックして、エディタウィンドウ上にソースを表示させてください。



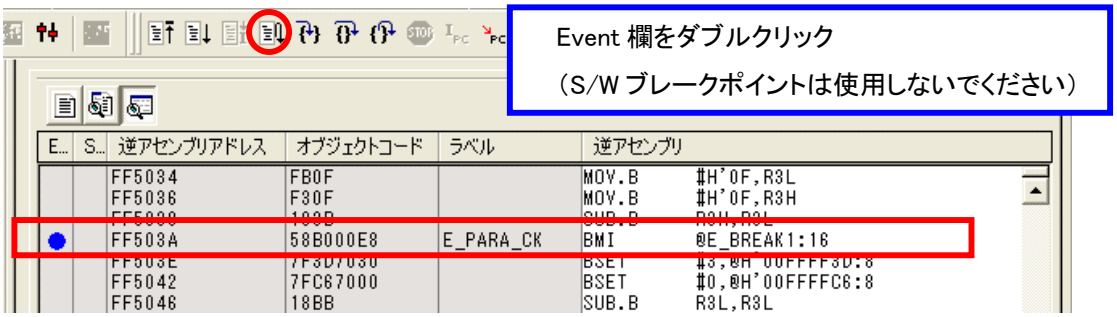
(2) 各処理のエラーチェックを行う前にフラッシュメモリに格納されているサンプルプログラムを内蔵 RAM に転送します。“flash_erpr_128”関数の先頭（行番号 128）にブレークを設定して[リセット後実行]アイコンをクリックしてプログラムを実行してください。



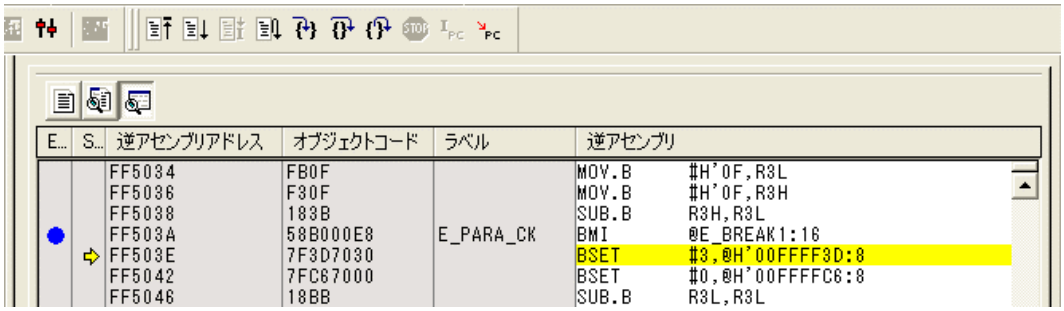
(3) エディタウィンドウの表示を[逆アセンブリモードで表示]アイコンをクリックしてアセンブリ表示にしてください。もしアセンブリ表示内容が本書と異なっている場合は、再度ビルド作業を行った後にサンプルプログラムをダウンロードし直してください。



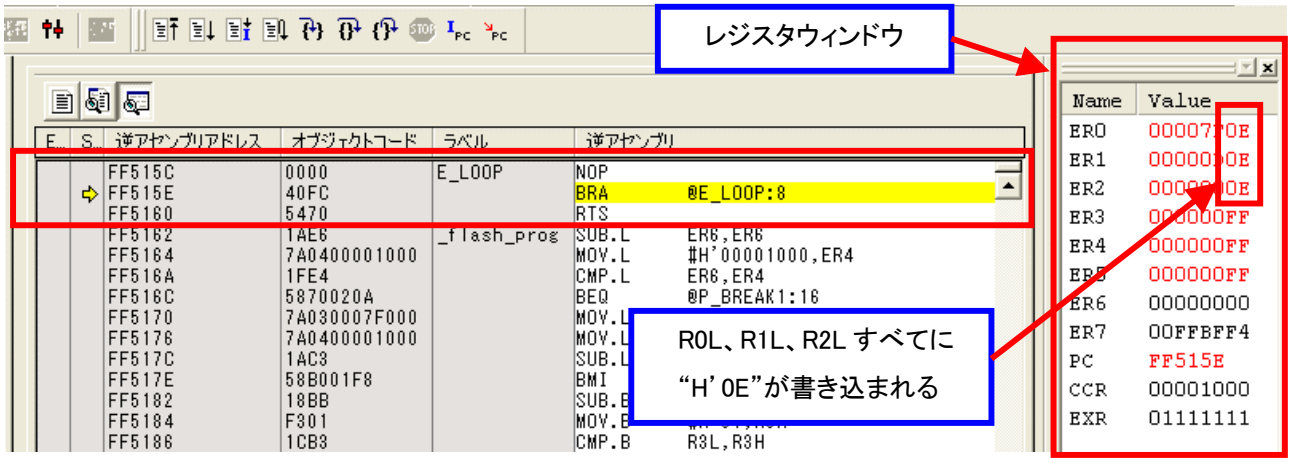
(4) 最初にシンボル（BLOCK_NUM、ERASE_N）の設定値エラーチェック方法について説明します。以下の箇所（ラベル“E_PARA_CK”）にブレークを設定して[リセット後実行]アイコンをクリックしてください。



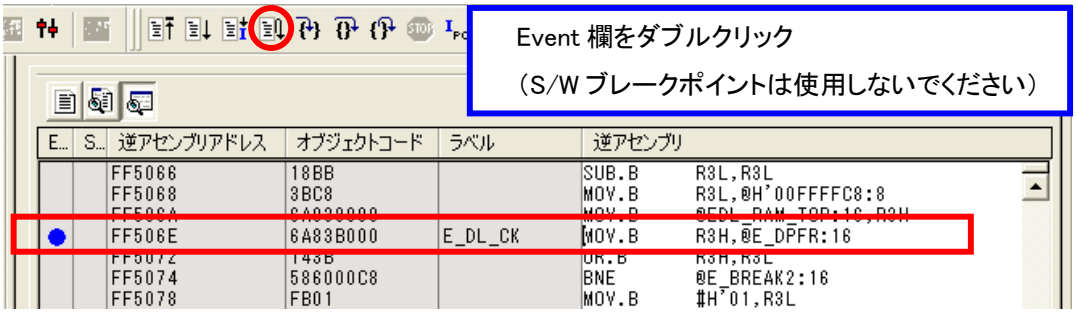
(5) 正しい設定値であれば、ブレークを設定した箇所でプログラムが停止します。この場合は、(6)の内容は読み飛ばしてください。



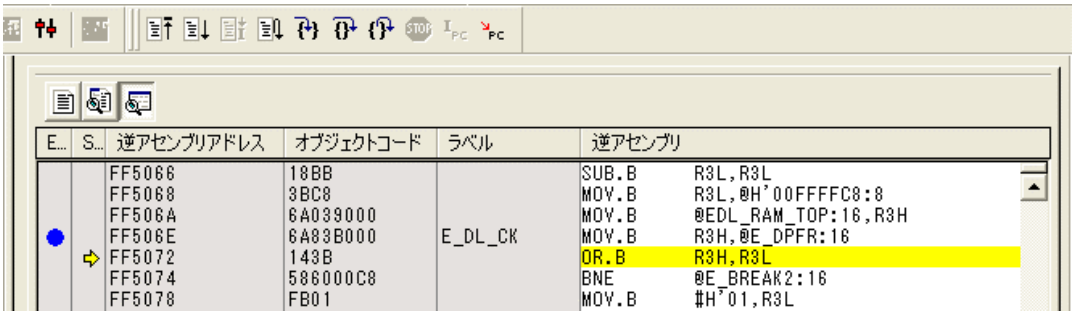
(6) しばらく経ってもプログラムが停止しない場合は間違った設定値になっており、プログラムを強制停止([停止]アイコンをクリック)すると以下の箇所(ラベル“E_LOOP”)でプログラムが停止して、汎用レジスタ R0L、R1L、R2L に“H’0E”が書き込まれます。この場合は、シンボル(BLOCK_NUM、ERASE_N)を正しい値に再設定して(4)からもう一度やり直してください。



(7) 次に消去プログラムのダウンロード処理エラーチェック方法について説明します。以下の箇所(ラベル“E_DL_CK”)にブレークを設定して[リセット後実行]アイコンをクリックしてください。

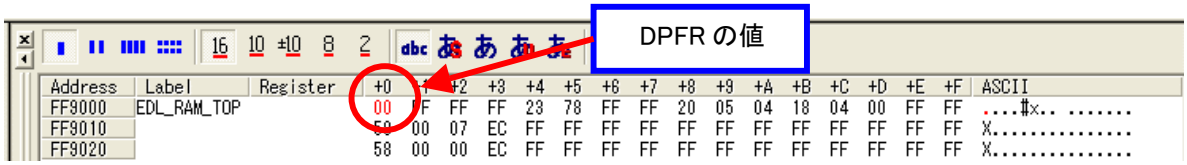


(8) ブレークを設定した箇所でプログラムが停止します。



HEW のメモリウィンドウを表示させて DPFR の値(ダウンロード先の先頭アドレスの値)を確認してください。

サンプルプログラムでは、ダウンロード先の先頭アドレスをシンボル“EDL_RAM_TOP”で設定しているので、“H’FF9000”番地の値が DPFR の値になります。



DPFR の値からダウンロード処理のエラーチェック結果を確認することができます。DPFR = H’00 以外の場合は、ダウンロード処理でエラーが発生しているので、エラー修正後に(7)からもう一度やり直してください。

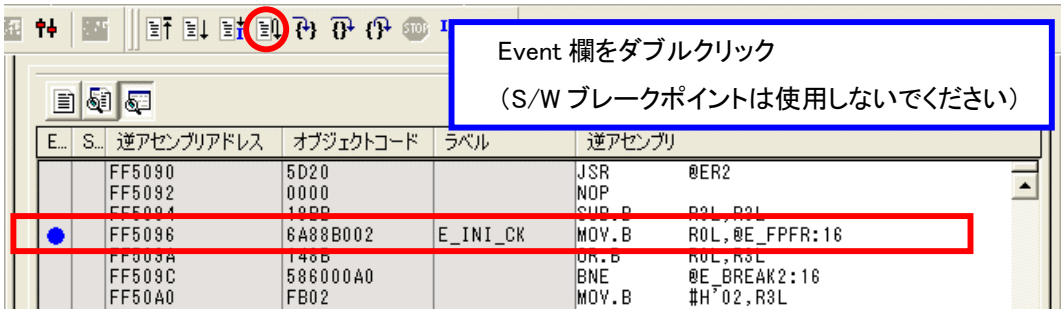
・DPFR = H’00　　→ ダウンロード処理成功(エラーなし)

・DPFR = H’00 以外　　→ ダウンロード処理失敗(エラー内容は表 6.4 を参照)

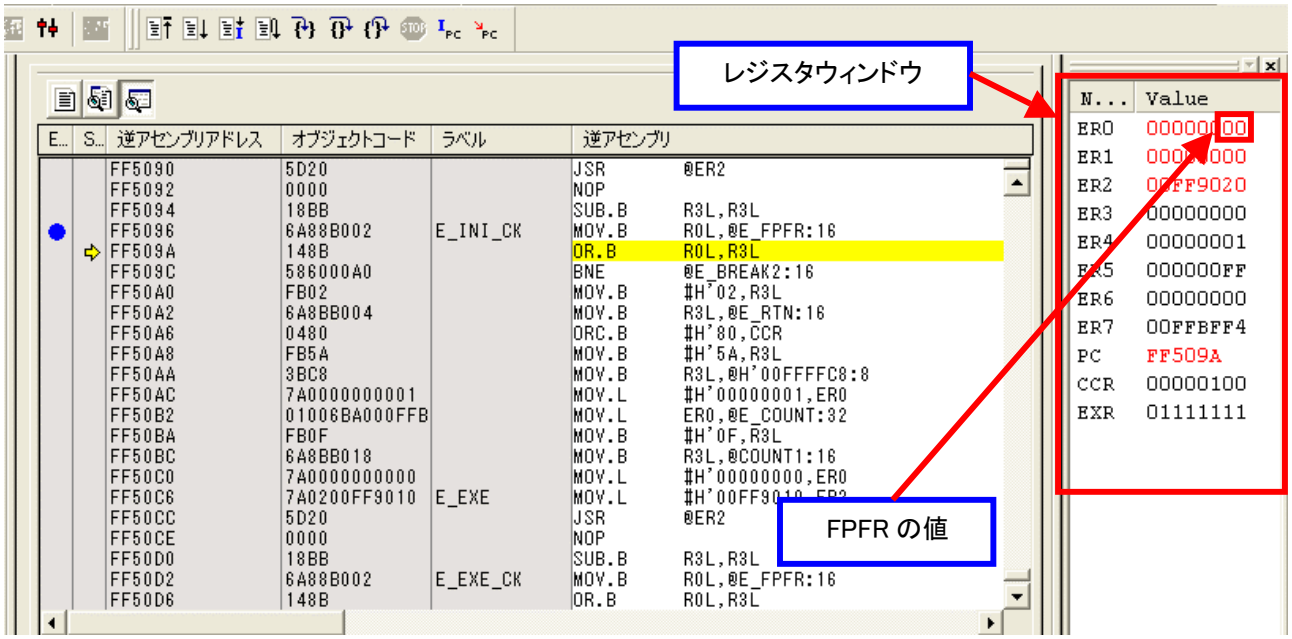
表 6.4 ダウンロードパス・フェイルリザルトパラメータ(DPFR)

ビット	ビット名	初期値	R/W	説明
7～3	−	−	−	未使用ビット 値 0 が戻されます。
2	SS	−	R/W	ソースセレクトエラー検出ビット ダウンロード可能な内蔵プログラムは1種類のみ指定できます。2種類以上の選択を行った場合、選択されていない場合、およびマッピングされていない選択の場合にはエラーとなります。 0:ダウンロードプログラムの選択関係は正常 1:ダウンロードエラー発生(多重選択または、マッピングされていないプログラム選択)
1	FK	−	R/W	フラッシュキーレジスタエラー検出ビット FKEYの値が、H’A5であるかどうかをチェックした結果を返すビットです。 0:FKEYの設定は正常(FKEY = H’A5) 1:FKEY の設定値エラー(FKEY は、H’A5 以外の値)
0	SF	−	R/W	サクセス／フェイルビット ダウンロードが正常に終了したかどうかを戻すビットです。内蔵RAM上にダウンロードしたプログラムをリードバックし、内蔵RAM上に転送できているかの判定結果です。 0:内蔵プログラムのダウンロードは正常終了(エラーなし) 1:内蔵プログラムのダウンロードが異常終了(エラーが発生している)

(9) 次に初期化処理エラーチェック方法について説明します。以下の箇所(ラベル“E_INI_CK”)にブレークを設定して[リセット後実行]アイコンをクリックしてください。



(10) ブレークを設定した箇所でプログラムが停止します。HEW のレジスタウィンドウを表示させて FPFR の値（汎用レジスタ R0L の値）を確認してください。



FPFR の値から、初期化処理のエラーチェック結果を確認することができます。FPFR = H'00 以外の場合は、初期化処理でエラーが発生しているので、エラー修正後に(9)からもう一度やり直してください。

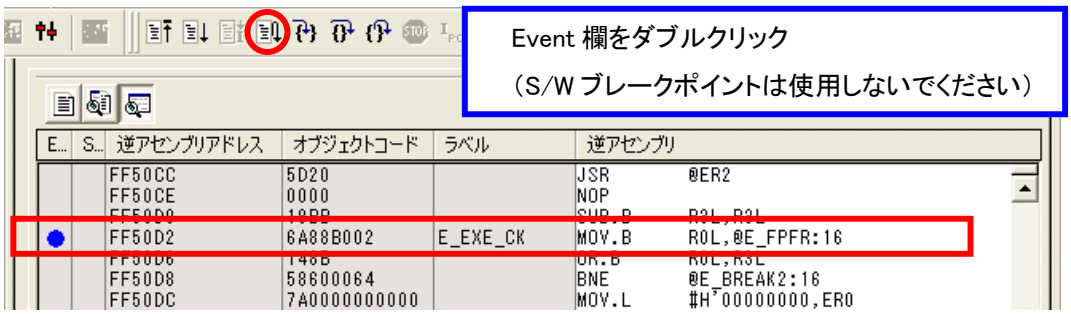
・FPFR = H'00 → 初期化処理成功（エラーなし）

・FPFR = H'00 以外 → 初期化処理失敗（エラー内容は表 6.5 を参照）

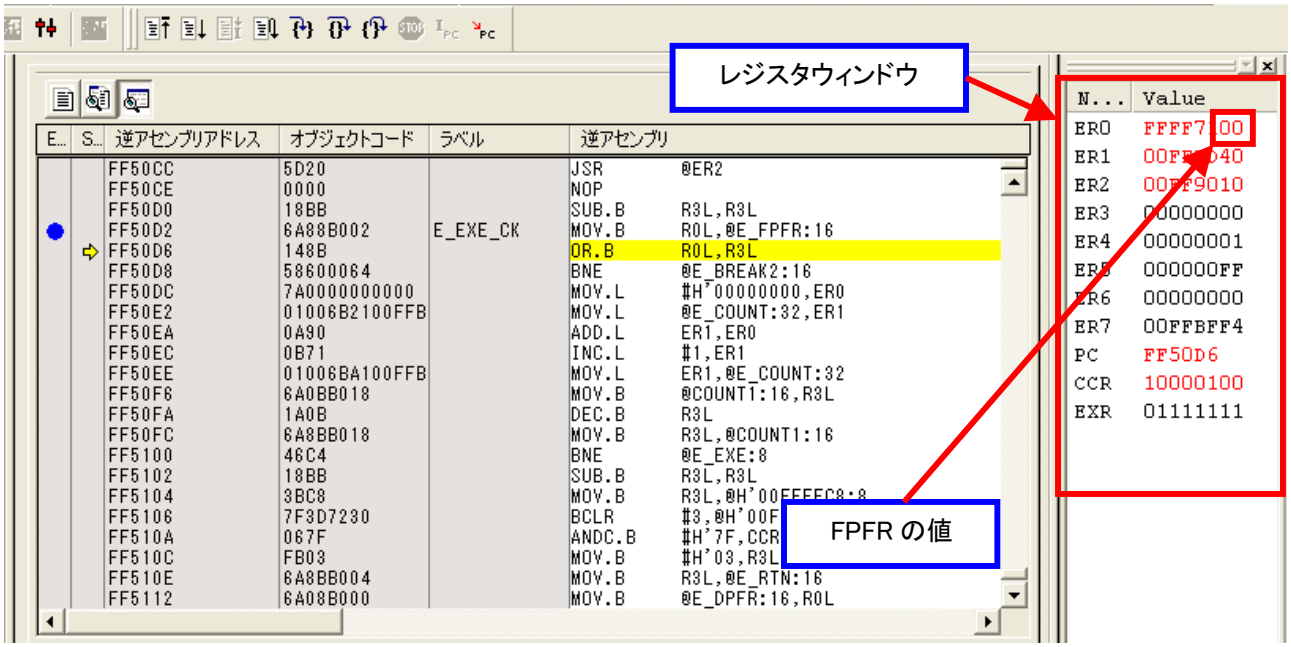
表 6.5 初期化処理時のフラッシュパス/フェイルパラメータ（FPFR）

ビット	ビット名	初期値	R/W	説明
7～3	－	－	－	未使用ビット 値 0 が戻されます。
2	BR	－	R/W	ユーザブランチエラー検出ビット 指定されたユーザブランチ先アドレスが、ダウンロードされている書き込み/消去関係プログラムの格納領域以外であるかをチェックした結果を戻します。 0：ユーザブランチアドレス設定は正常値 1：ユーザブランチアドレス設定が異常値
1	FQ	－	R/W	周波数エラー検出ビット 指定されたCPU動作周波数が、サポートしている動作周波数の範囲にあるかをチェックした結果を戻します。 0：動作周波数の設定は正常値 1：動作周波数の設定が異常値
0	SF	－	R/W	サクセス/フェイルビット 初期化が正常に終了したかどうかを戻すビットです。 0：初期化は正常終了（エラーなし） 1：初期化が異常終了（エラーが発生している）

(11) 次に消去処理エラーチェック方法について説明します。以下の箇所（ラベル“E_EXE_CK”）にブレークを設定して[リセット後実行]アイコンをクリックしてください。



(12) ブレークを設定した箇所でプログラムが停止します。HEW のレジスタウィンドウを表示させて FPFR の値（汎用レジスタ R0L の値）を確認してください。



FPFR の値から、消去処理のエラーチェック結果を確認することができます。FPFR = H’00 以外の場合は、消去処理でエラーが発生しているので、エラー修正後に(11)からもう一度やり直してください。

・FPFR = H’00　　→ 消去処理成功（エラーなし）

・FPFR = H’00 以外　　→ 消去処理失敗（エラー内容は表 6.6 を参照）

表 6.6　消去処理時のフラッシュパス/フェイルパラメータ(FPFR)

ビット	ビット名	初期値	R/W	説明
7	−	−	−	未使用ビット 値 0 が戻されます。
6	MD	−	R/W	消去モード関連設定エラー検出ビット エラープロテクト状態でないことのチェック結果を返します。エラープロテクト状態になっている場合、1が書き込まれます。これらの状態は、FCCSのFLERで確認できます。 0:FLER状態は正常(FLER = 0) 1:FLER = 1 であり、消去できない状態
5	EE	−	R/W	消去実行時エラー検出ビット ユーザマットの消去ができなかったり、ユーザブランチ処理から戻った時点でフラッシュ関連レジスタの一部が書き換えられている場合に、本ビットには1が返されます。これらが原因で、本ビットが1になった場合、ユーザマットは途中まで消去されている可能性が高いため、エラーになる原因を取り除いた後、再度消去を実施し直してください。また、FMATSレジスタの値がH'AAとなっており、ユーザブートマット選択状態のときに消去を実施しても、消去実行時エラーとなります。この場合は、ユーザマット/ユーザブートマットともに、消去されてはいません。ユーザブートマットの消去はブートモードまたはライタモードで実施してください。
4	FK	−	R/W	フラッシュキーレジスタエラー検出ビット 消去処理開始前にFKEYの値をチェックした結果を戻します。 0:FKEYの設定は正常(FKEY = H'5A) 1:FKEY の設定値エラー(FKEY は、H'5A 以外の値)
3	EB	−	R/W	イレーズブロックセレクトエラー検出ビット 指定された消去ブロック番号が、ユーザマットのブロック範囲内であるかのチェック結果です。 0:消去ブロック番号の設定は正常値 1:消去ブロック番号の設定が異常値
2 1	− −	− −	− −	未使用ビット 値 0 が戻されます。
0	SF	−	R/W	サクセス/フェイルビット 消去処理が正常に終了したかどうかを戻すビットです。 0:消去は正常終了(エラーなし) 1:消去が異常終了(エラーが発生している)

(13) 最後に、消去プログラムのダウンロード処理、初期化処理、消去処理をまとめて行った場合のエラーチェック方法について説明します。以下の箇所（ラベル“E_ALL_CK”）にブレークを設定して[リセット後実行]アイコンをクリックしてください。

サンプルプログラムでは、プログラム正常終了後、エラー発生後（ラベル“E_LOOP”へ分岐してループ処理を行う）に汎用レジスタ R0L、R1L、R2L に下記のパラメータを書き込むようになっています。

・R0L → DPFR（ダウンロード処理エラーチェック結果）の値

・R1L → FPFR（初期化・消去処理エラーチェック結果）の値

・R2L → RTN（※1）の値

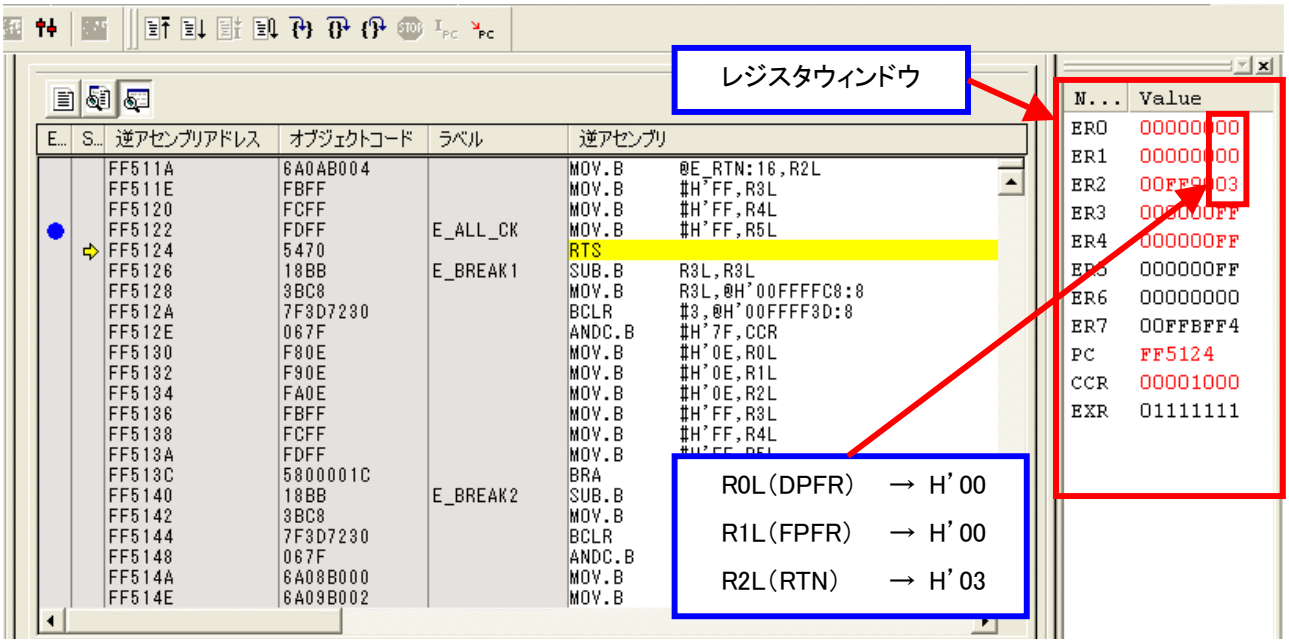
※1. RTN とは各処理が行われ、終了すると 1 ずつ加算するパラメータで、どの処理でエラーが発生したか明確にする為に本サンプルプログラムで定義したシンボルです。RTN の値の意味は下記の通りです。

・RTN = H’01 → 消去プログラムのダウンロード処理が終了

・RTN = H’02 → 初期化処理が終了

・RTN = H’03 → 消去処理が終了

(14) 正常に各処理が行われた場合は、ブレークを設定した箇所でプログラムが停止し、汎用レジスタ R0L、R1L、R2L にそれぞれ“H’00”、“H’00”、“H’03”が書き込まれます。この場合は、(15)の内容は読み飛ばしてください。



(15) しばらく経ってもプログラムが停止しない場合は、どこかの処理でエラーが発生しています。プログラムを強制停止（[停止]アイコンをクリック）すると、エラーの発生した箇所に応じて汎用レジスタ R0L、R1L、R2L に値が書き込まれます。表 6.7 に汎用レジスタによるエラーチェック結果を示します。

表 6.7　消去実行時の汎用レジスタによるエラーチェック結果

R0L（DPFR）	R1L（FPFR）	R2L（RTN）	エラーチェック結果
H’00 以外	−	H’00	消去プログラムのダウンロード処理エラー（表 6.4 を参照）
H’00	H’00 以外	H’01	初期化処理エラー（表 6.5 を参照）
H’00	H’00 以外	H’02	消去処理エラー（表 6.6 を参照）
H’0E	H’0E	H’0E	シンボル（BLOCK_NUM、ERASE_N）設定値エラー

本章で説明した方法でエラーが発生していなければ、フラッシュメモリの消去は正常に行われています。以上で、フラッシュメモリ消去時のエラーチェック方法の説明は終了です。

6.5　フラッシュメモリ書き込みエラーチェック方法

本章では、サンプルプログラムを使用してフラッシュメモリの書き込み時（書き込みプログラムのダウンロード処理、初期化処理、書き込み処理、書き込み終了処理）にエラーが発生した場合のチェック方法を説明します。図 6.4、図 6.5 にエラーチェック方法のフローチャート図を示します。

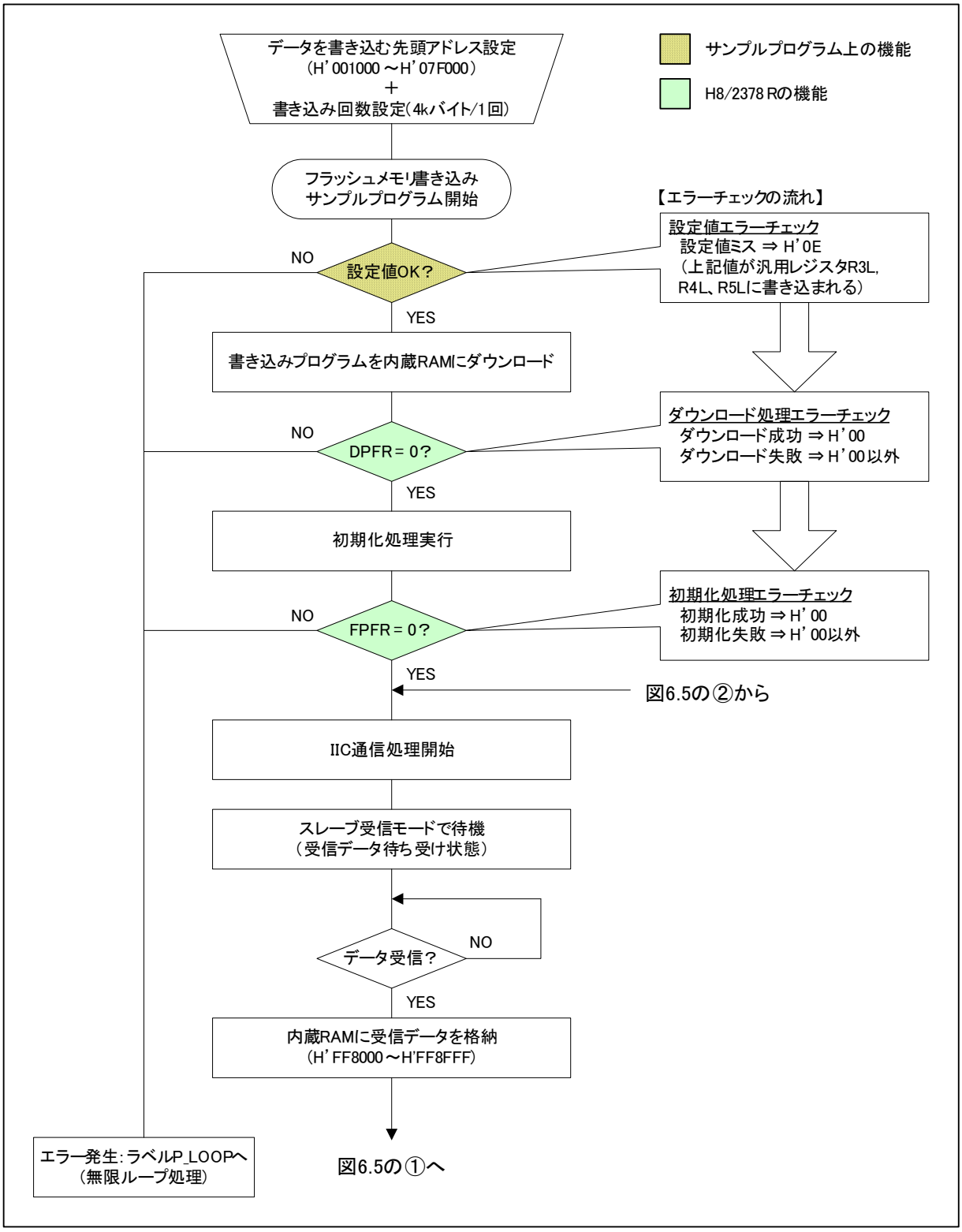


図 6.4　フラッシュメモリ書き込み時のエラーチェックフローチャート図①

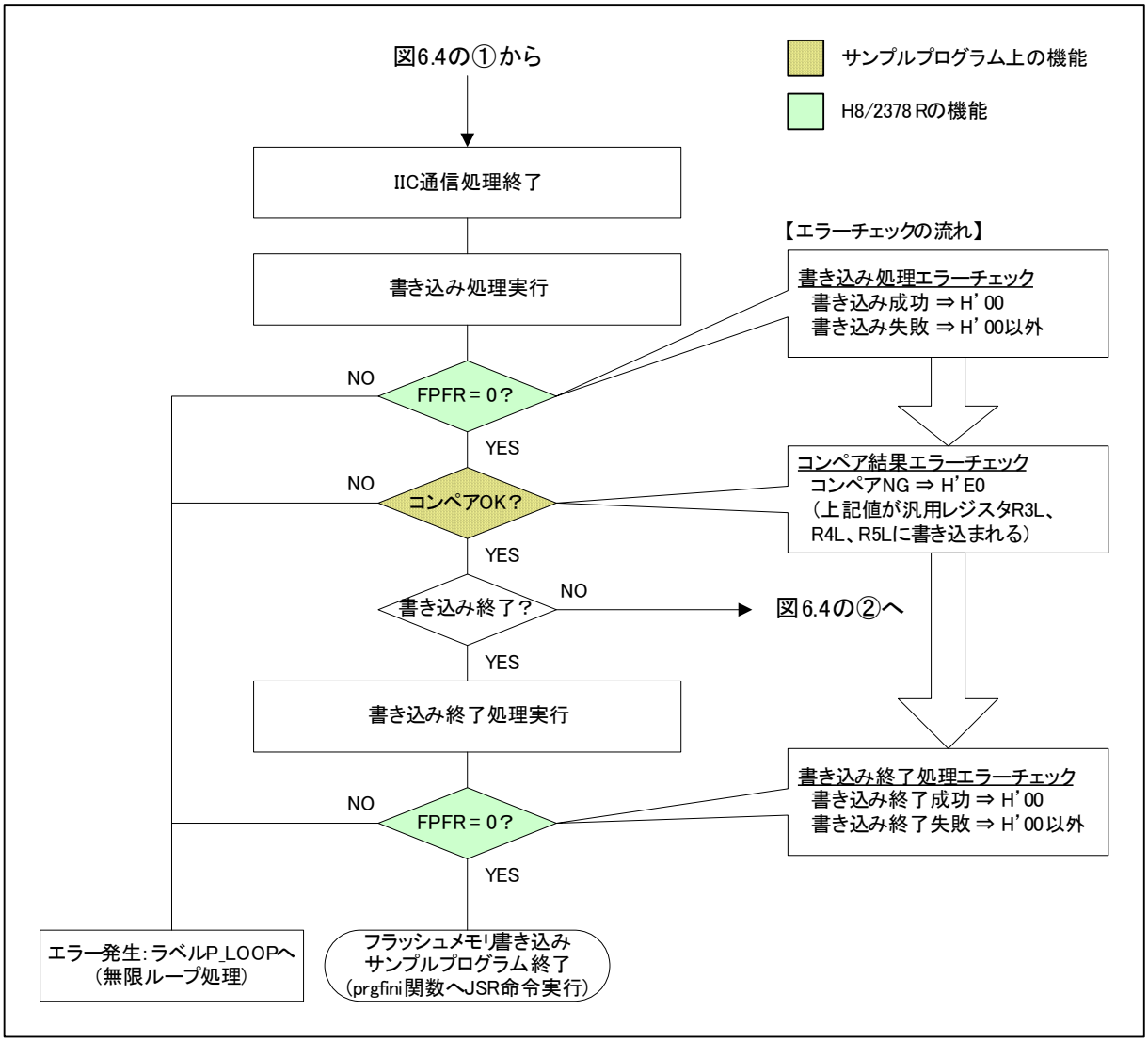
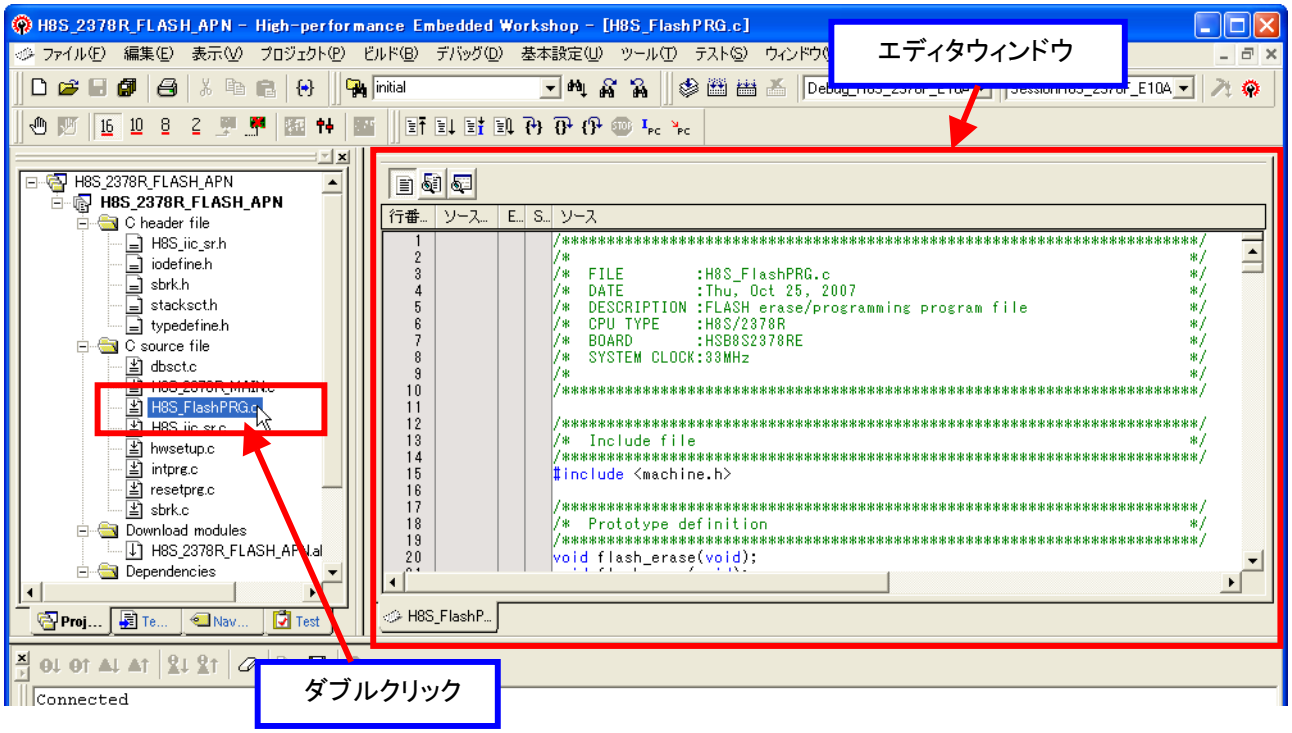
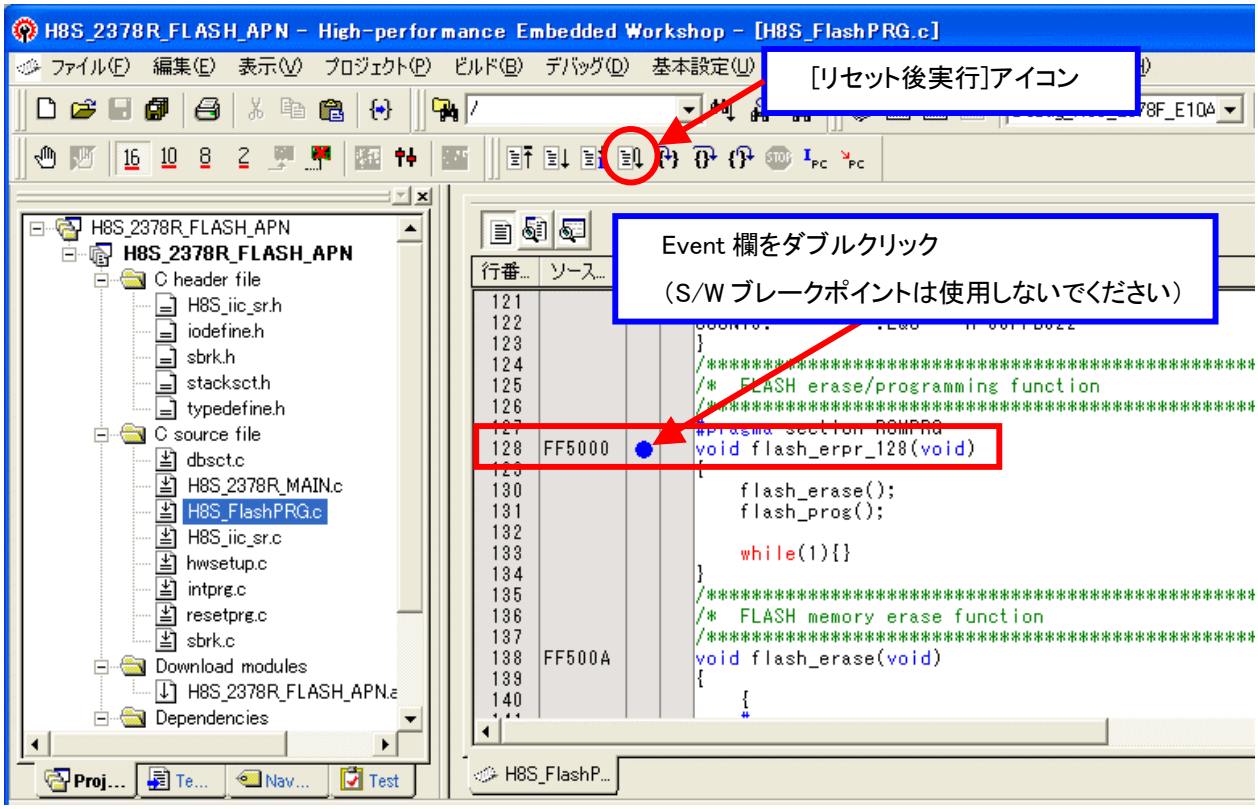


図 6.5　フラッシュメモリ書き込み時のエラーチェックフローチャート図②

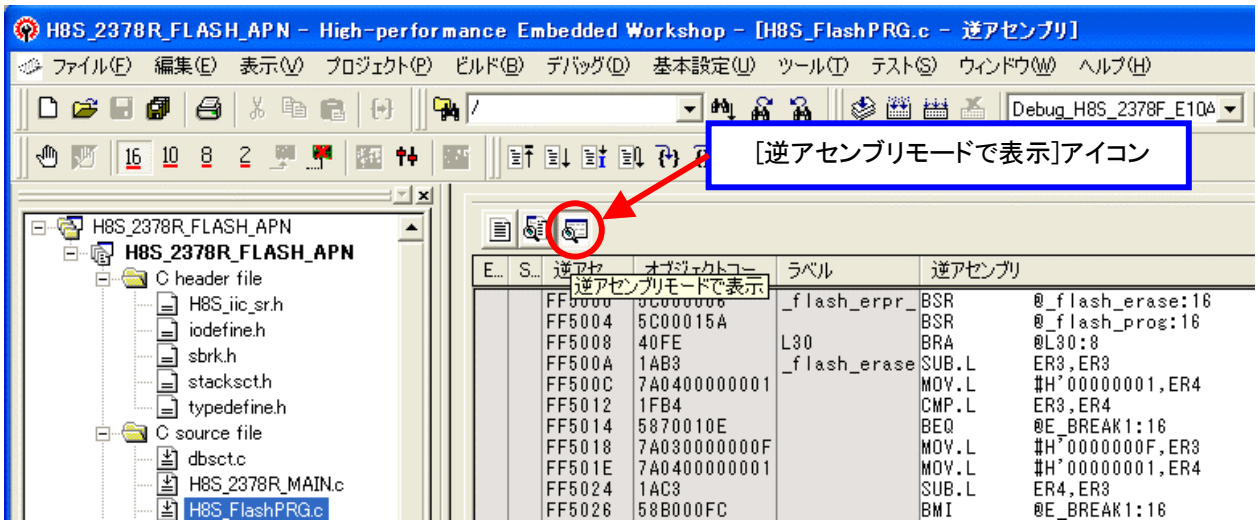
(1) ワークスペースウィンドウ上の[C source file]フォルダ内のファイル“H8S_FlashPRG.c”をダブルクリックして、エディタウィンドウ上にソースを表示させてください。



(2) 各処理のエラーチェックを行う前にフラッシュメモリに格納されているサンプルプログラムを内蔵 RAM に転送します。“flash_erpr_128”関数の先頭（行番号 128）にブレークを設定して[リセット後実行]アイコンをクリックしてプログラムを実行してください。

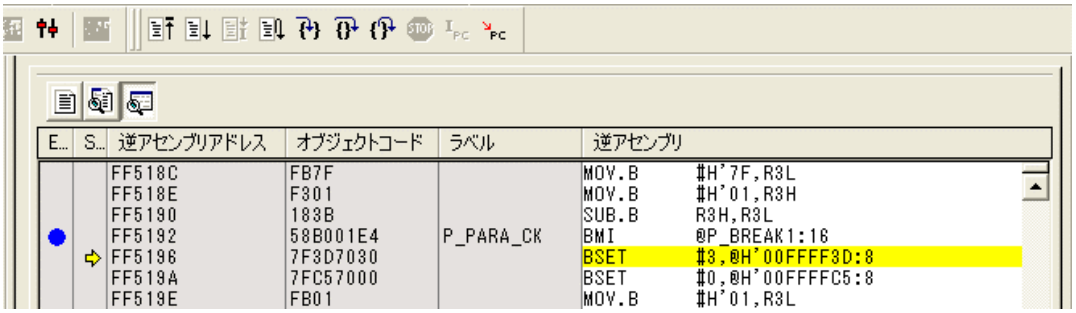


(3) エディタウィンドウの表示を[逆アセンブリモードで表示]アイコンをクリックしてアセンブリ表示にしてください。もし、アセンブリ表示内容が本書と異なっている場合は、再度ビルド作業を行った後にサンプルプログラムをダウンロードし直してください。

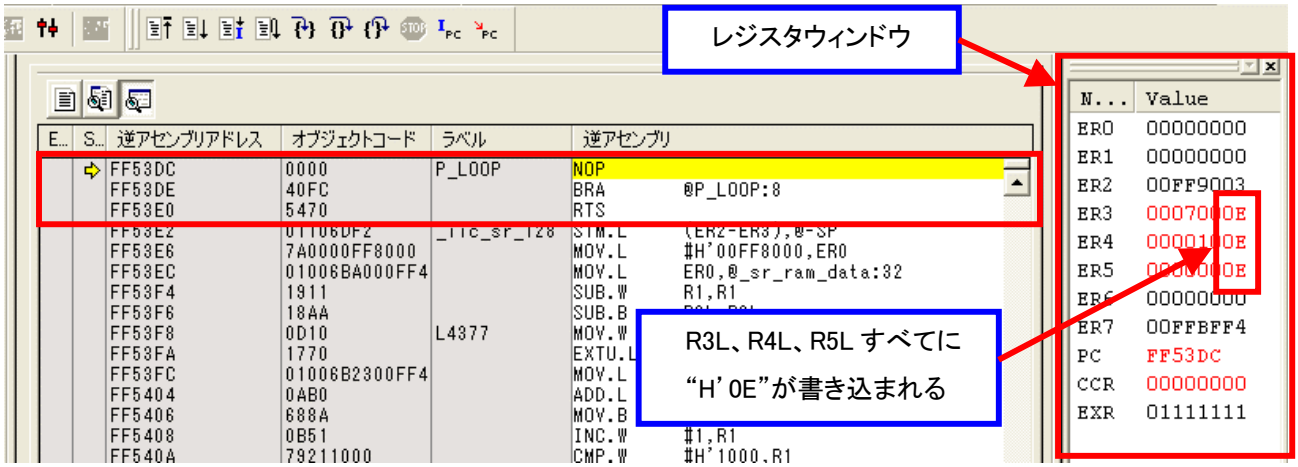


(4) 最初にシンボル（PR_FL_TOP、PROG_N）の設定値エラーチェック方法について説明します。以下の箇所（ラベル“P_PARA_CK”）にブレークを設定して[リセット後実行]アイコンをクリックしてください。

(5) 正しい設定値であれば、ブレークを設定した箇所でプログラムが停止します。この場合は、(6)の内容は読み飛ばしてください。

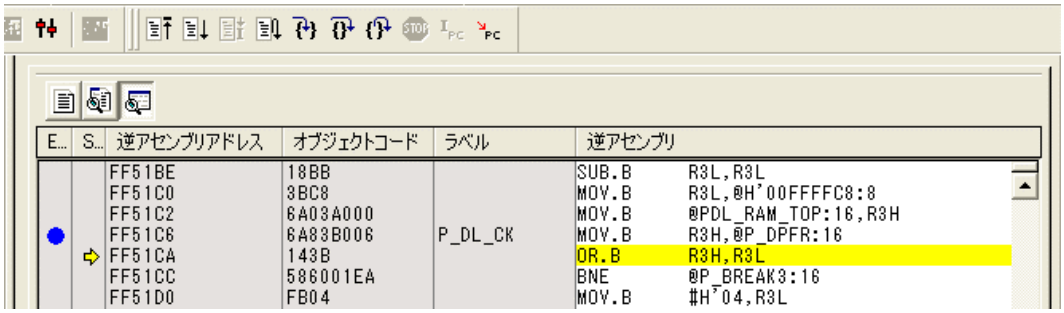


(6) しばらく経ってもプログラムが停止しない場合は間違った設定値になっており、プログラムを強制停止([停止]アイコンをクリック)すると以下の箇所(ラベル“P_LOOP”)でプログラムが停止して、汎用レジスタ R3L、R4L、R5L に“H’0E”が書き込まれます。この場合は、シンボル(PR_FL_TOP、PROG_N)を正しい値に再設定して(4)からもう一度やり直してください。



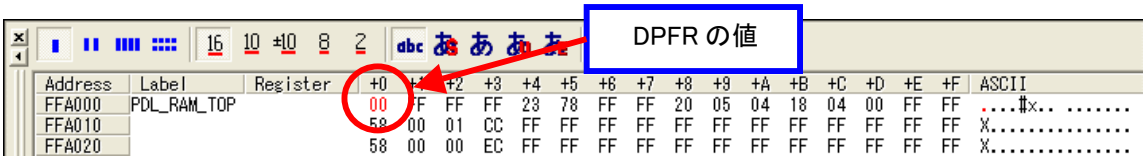
(7) 次に書き込みプログラムのダウンロード処理エラーチェック方法について説明します。以下の箇所(ラベル“P_DL_CK”)にブレークを設定して[リセット後実行]アイコンをクリックしてください。

(8) ブレークを設定した箇所でプログラムが停止します。



HEW のメモリウィンドウを表示させて DPFR の値(ダウンロード先の先頭アドレスの値)を確認してください。

サンプルプログラムでは、ダウンロード先の先頭アドレスをシンボル“PDL_RAM_TOP”で設定しているので、“H’FFA000”番地の値が DPFR の値になります。



DPFR の値からダウンロード処理のエラーチェック結果を確認することができます。DPFR = H’00 以外の場合は、ダウンロード処理でエラーが発生しているので、エラー修正後に(7)からもう一度やり直してください。

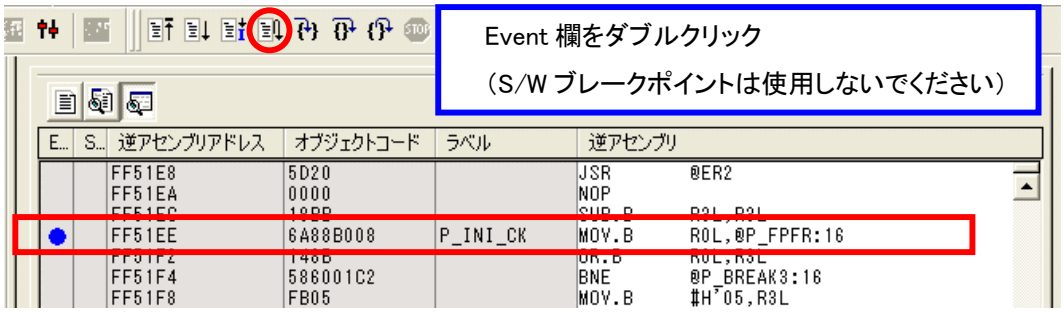
・DPFR = H’00　　→ ダウンロード処理成功(エラーなし)

・DPFR = H’00 以外　　→ ダウンロード処理失敗(エラー内容は表 6.8 を参照)

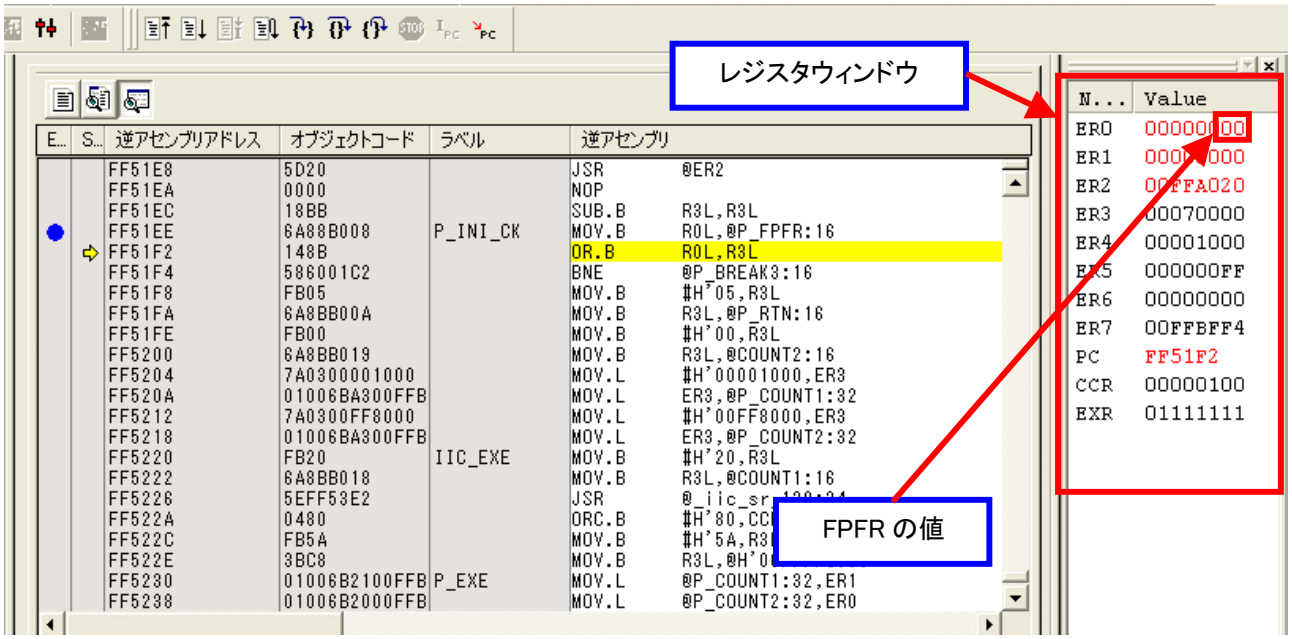
表 6.8 ダウンロードパス・フェイルリザルトパラメータ(DPFR)

ビット	ビット名	初期値	R/W	説明
7～3	−	−	−	未使用ビット 値 0 が戻されます。
2	SS	−	R/W	ソースセレクトエラー検出ビット ダウンロード可能な内蔵プログラムは1種類のみ指定できます。2種類以上の選択を行った場合、選択されていない場合、およびマッピングされていない選択の場合にはエラーとなります。 0:ダウンロードプログラムの選択関係は正常 1:ダウンロードエラー発生(多重選択または、マッピングされていないプログラム選択)
1	FK	−	R/W	フラッシュキーレジスタエラー検出ビット FKEYの値が、H’A5であるかどうかをチェックした結果を返すビットです。 0:FKEYの設定は正常(FKEY = H’A5) 1:FKEY の設定値エラー(FKEY は、H’A5 以外の値)
0	SF	−	R/W	サクセス／フェイルビット ダウンロードが正常に終了したかどうかを戻すビットです。内蔵RAM上にダウンロードしたプログラムをリードバックし、内蔵RAM上に転送できているかの判定結果です。 0:内蔵プログラムのダウンロードは正常終了(エラーなし) 1:内蔵プログラムのダウンロードが異常終了(エラーが発生している)

(9) 次に初期化処理エラーチェック方法について説明します。以下の箇所(ラベル“P_INI_CK”)にブレークを設定して[リセット後実行]アイコンをクリックしてください。



(10) ブレークを設定した箇所でプログラムが停止します。HEW のレジスタウィンドウを表示させて FPFR の値（汎用レジスタ R0L の値）を確認してください。



FPFR の値から、初期化処理のエラーチェック結果を確認することができます。FPFR = H'00 以外の場合は、初期化処理でエラーが発生しているので、エラー修正後に(9)からもう一度やり直してください。

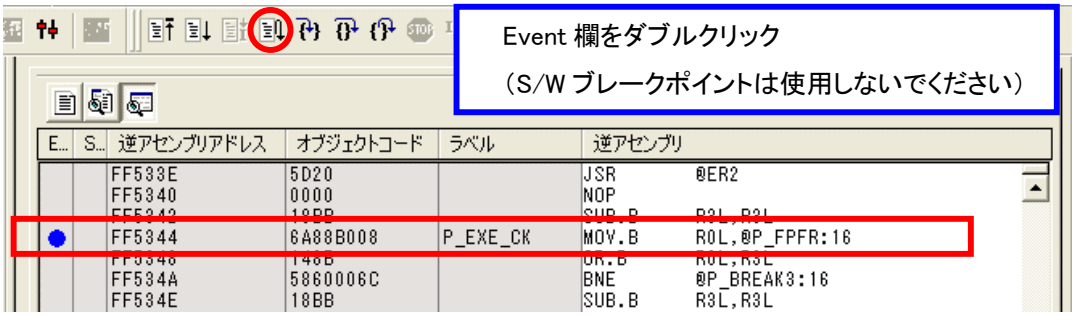
・FPFR = H'00 → 初期化処理成功（エラーなし）

・FPFR = H'00 以外 → 初期化処理失敗（エラー内容は表 6.9 を参照）

表 6.9 初期化処理時のフラッシュパス/フェイルパラメータ（FPFR）

ビット	ビット名	初期値	R/W	説明
7～3	－	－	－	未使用ビット 値 0 が戻されます。
2	BR	－	R/W	ユーザブランチエラー検出ビット 指定されたユーザブランチ先アドレスが、ダウンロードされている書き込み/消去関係プログラムの格納領域以外であるかをチェックした結果を戻します。 0：ユーザブランチアドレス設定は正常値 1：ユーザブランチアドレス設定が異常値
1	FQ	－	R/W	周波数エラー検出ビット 指定されたCPU動作周波数が、サポートしている動作周波数の範囲にあるかをチェックした結果を戻します。 0：動作周波数の設定は正常値 1：動作周波数の設定が異常値
0	SF	－	R/W	サクセス/フェイルビット 初期化が正常に終了したかどうかを戻すビットです。 0：初期化は正常終了（エラーなし） 1：初期化が異常終了（エラーが発生している）

(11) 次に書き込み処理、書き込み終了処理エラーチェック方法について説明します。以下の箇所（ラベル“P_EXE_CK”）にブレークを設定して[リセット後実行]アイコンをクリックしてください。



上記箇所にブレーク設定してプログラムを実行すると、IIC バスインタフェース 2 のチャネル 0（IIC2_0）によるスレーブ受信モードに入り、IIC_マスタ送信側からデータが送信されるまで待機状態となります。プログラム実行後にフラッシュメモリに書き込むデータを送信してください。

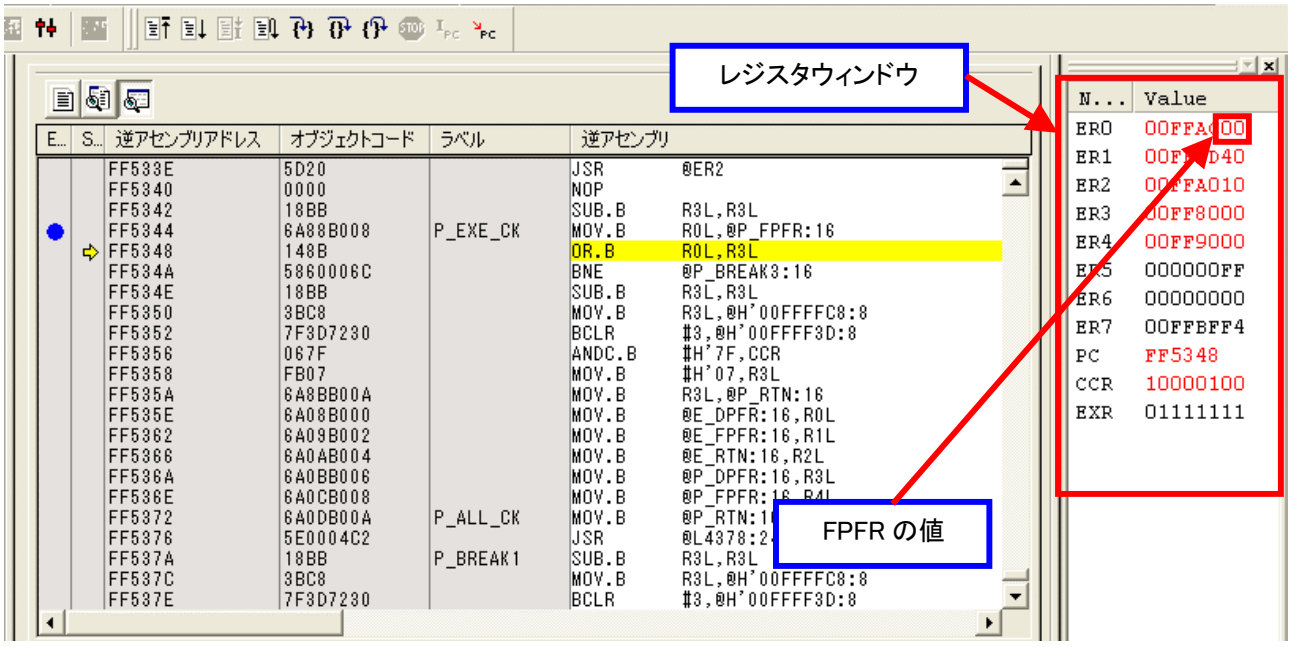
IIC_マスタ送信側プログラムは参考例として「付録 IIC_マスタ送信プログラム」に記載しています。

また、コンペアチェック（フラッシュメモリに書き込むデータとフラッシュメモリに書き込んだデータの比較）は書き込み処理と一緒に行われます。

書き込み処理が正常に終わると書き込み終了処理が行われます。

(12) IIC 通信、書き込み処理、コンペアチェックが正常に行われるとブレークを設定した箇所でプログラムが停止します。この場合は、(13)の内容は読み飛ばしてください。

HEW のレジスタウィンドウを表示させて FPFR の値（汎用レジスタ R0L の値）を確認してください。

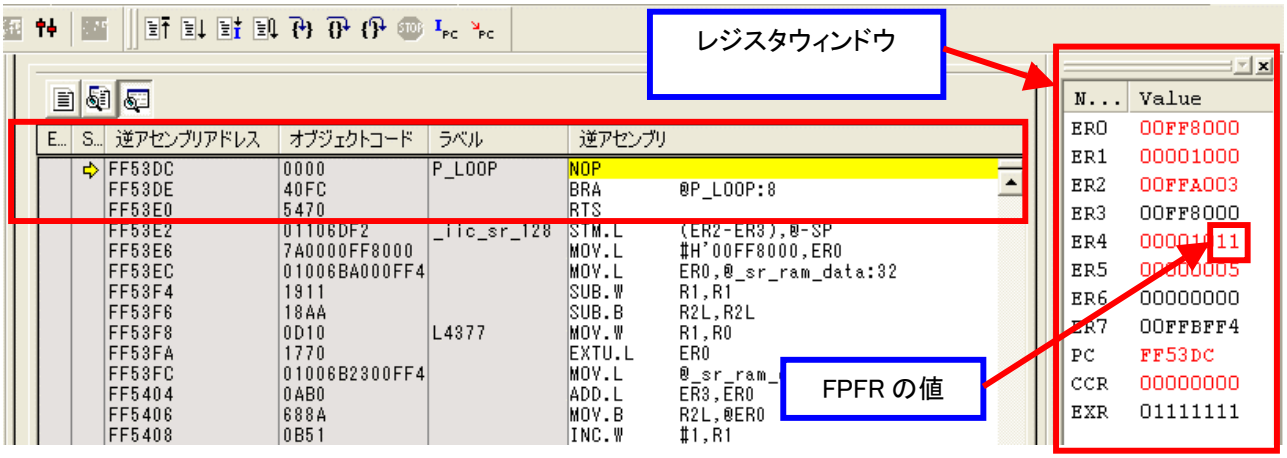


FPFR の値から、書き込み終了処理のエラーチェック結果を確認することができます。FPFR = H’00 以外の場合は、書き込み終了処理でエラーが発生しているので、エラー修正後に(11)からもう一度やり直してください。

・FPFR = H’00 → 書き込み終了処理成功（エラーなし）

・FPFR = H’00 以外 → 書き込み終了処理失敗（エラー内容は表 6.10 を参照）

(13) しばらく経ってもプログラムが停止しない場合は、書き込み処理（コンペアチェックを含む）のところでエラーが発生しています。プログラムを強制停止（[停止]アイコンをクリック）すると以下の箇所（ラベル“P_LOOP”）でプログラムが停止して、汎用レジスタ R4L にエラーチェック結果（FPFR の値）が書き込まれます。



サンプルプログラムでは、プログラム正常終了後、エラー発生後（ラベル“P_LOOP”へ分岐してループ処理を行う）に汎用レジスタ R4L にフラッシュメモリ書き込み処理エラーチェック結果（FPFR の値）を書き込むようになっています。FPFR の値によるエラー内容は表 6.10 をご参照いただき、エラー修正後に（11）からもう一度やり直してください。

また、汎用レジスタ R3L、R4L、R5L すべてに“H’E0”が書き込まれている場合は、コンペアチェックエラーが発生しています。

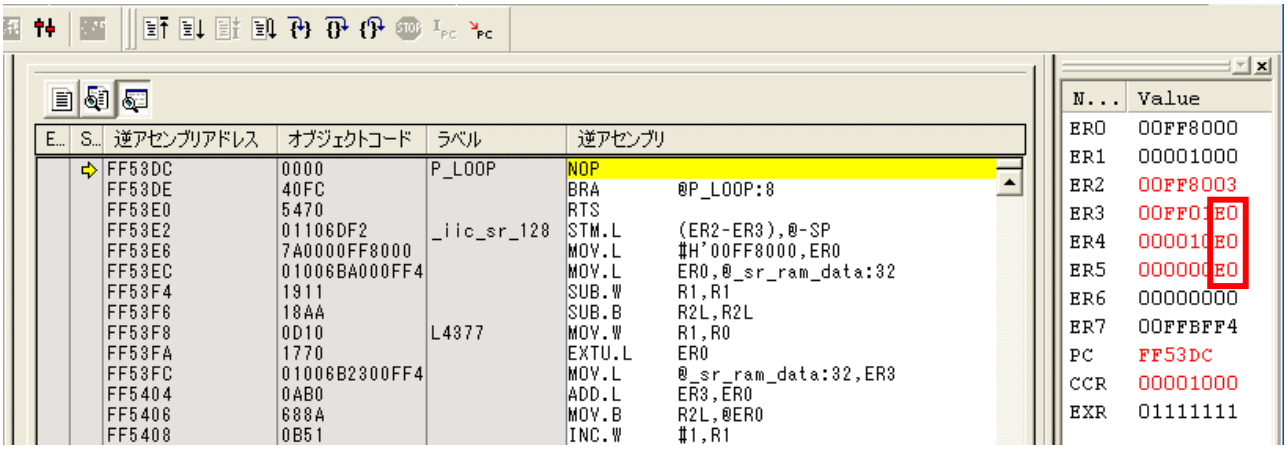


表 6.10 書き込み処理、書き込み終了処理時のフラッシュパス/フェイルパラメータ(FPFR)

ビット	ビット名	初期値	R/W	説明
7	−	−	−	未使用ビット 値 0 が戻されます。
6	MD	−	R/W	書き込みモード関連設定エラー検出ビット エラープロテクト状態でないことのチェック結果を返します。エラープロテクト状態になっている場合、1が書き込まれます。これらの状態は、FCCSのFLERビットで確認できます。 0:FLER状態は正常(FLER = 0) 1:FLER = 1 であり、書き込みできない状態
5	EE	−	R/W	書き込み実行時エラー検出ビット ユーザマットが消去されていないために、指定データを書き込めなかったり、ユーザブランチ処理から戻った時点でフラッシュ関連レジスタの一部が書き換えられている場合に、本ビットには1が返されます。これらが原因で、本ビットが1になった場合、ユーザマットは途中まで書き換えられている可能性が高いため、エラーになる原因を取り除いた後、消去から実施し直してくだい。また、FMATS レジスタの値がH 'AAとなっており、ユーザブートマット選択状態のときに書き込みを実施しても、書き込み実行時エラーとなります。この場合は、ユーザマット/ユーザブートマットともに、書き換えられてはいません。ユーザブートマットの書き込みはブートモードまたはライタモードで実施してください。 0:書き込み処理は正常終了 1:書き込み処理が異常終了し、書き込み結果は保証できない
4	FK	−	R/W	フラッシュキーレジスタエラー検出ビット 書き込み処理開始前にFKEY の値をチェックした結果を戻します。 0:FKEYの設定は正常(FKEY = H' 5A) 1:FKEY の設定値エラー(FKEY は、H' 5A 以外の値)
3	−	−	R/W	未使用ビット 値 0 が戻されます。
2	WD	−	R/W	ライトデータアドレス検出ビット 書き込みデータの格納先の先頭アドレスとして、フラッシュメモリ領域のアドレスが指定された場合にはエラーとなります。 0:書き込みデータアドレス設定は正常値 1:書き込みデータアドレス設定が異常値
1	WA	−	R/W	ライトアドレスエラー検出ビット 書き込み先の先頭アドレスとして、以下が指定された場合にはエラーとなります。 ・フラッシュメモリの領域外が書き込み先アドレスとして指定された場合 ・指定されたアドレスが128バイト境界でない(A6～A0が0でない)場合 0:書き込み先アドレス設定は正常値 1:書き込み先アドレス設定が異常値
0	SF	−	R/W	サクセス／フェイルビット 書き込み処理が正常に終了したかどうかを戻すビットです。 0:書き込みは正常終了(エラーなし) 1:書き込みが異常終了(エラーが発生している)

(16) 最後に、書き込みプログラムのダウンロード処理、初期化処理、書き込み処理、書き込み終了処理をまとめて行った場合のエラーチェック方法について説明します。以下の箇所(ラベル“P_ALL_CK”)にブレークを設定して[リセット後実行]アイコンをクリックしてください。

サンプルプログラムでは、プログラム正常終了後、エラー発生後(ラベル“P_LOOP”へ分岐してループ処理を行う)に汎用レジスタ R3L、R4L、R5L に下記のパラメータを書き込むようになっています。

・R3L → DPFR(ダウンロード処理エラーチェック結果)の値

・R4L → FPFR(初期化処理・書き込み処理・書き込み終了処理エラーチェック結果)の値

・R5L → RTN(※1)の値

※1. RTN とは各処理が行われ、終了すると 1 ずつ加算するパラメータで、どの処理でエラーが発生したか明確にする為に本サンプルプログラムで定義したシンボルです。RTN の値の意味は下記の通りです。

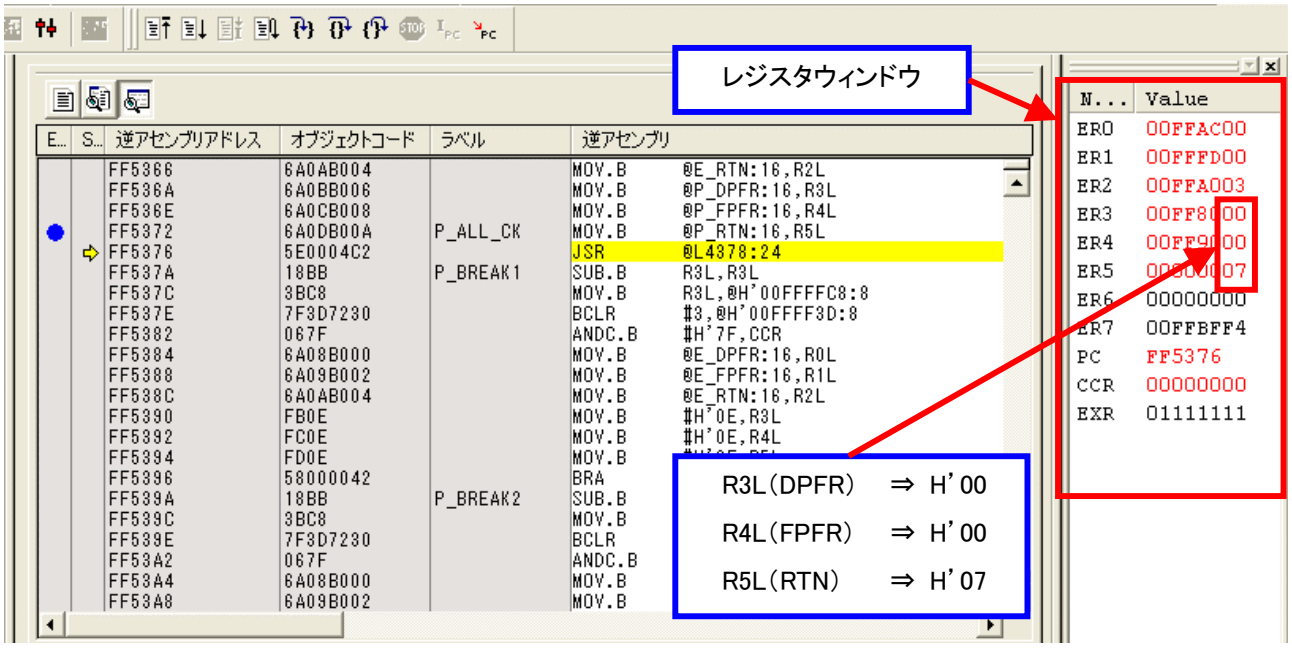
・RTN = H'04 → 書き込みプログラムのダウンロード処理が終了

・RTN = H'05 → 初期化処理が終了

・RTN = H'06 → 書き込み処理が終了

・RTN = H'07 → 書き込み終了処理が終了

(17) 正常に各処理が行われた場合は、ブレークを設定した箇所でプログラムが停止し、汎用レジスタ R3L、R4L、R5L にそれぞれ“H'00”、“H'00”、“H'07”が書き込まれます。この場合は、(18)の内容は読み飛ばしてください。



(18) しばらく経ってもプログラムが停止しない場合は、どこかの処理でエラーが発生しています。プログラムを強制停止（[停止]アイコンをクリック）すると、エラーの発生した箇所に応じて汎用レジスタ R3L、R4L、R5L に値が書き込まれます。表 6.11 に汎用レジスタによるエラーチェック結果を示します。

表 6.11　書き込み実行時の汎用レジスタによるエラーチェック結果

R3L（DPFR）	R4L（FPFR）	R5L（RTN）	エラーチェック結果
H’00 以外	−	H’00	書き込みプログラムのダウンロード処理エラー（表 6.8 を参照）
H’00	H’00 以外	H’04	初期化処理エラー（表 6.9 を参照）
H’00	H’00 以外	H’05	書き込み処理エラー（表 6.10 を参照）
H’00	H’00 以外	H’06	書き込み終了処理エラー（表 6.10 を参照）
H’0E	H’0E	H’0E	シンボル（PR_FL_TOP、PROG_N）設定値エラー
H’E0	H’E0	H’E0	コンペアチェックエラー

本章で説明した方法でエラーが発生していなければ、フラッシュメモリの書き込みは正常に行われています。以上で、フラッシュメモリ書き込み時のエラーチェック方法の説明は終了です。

6.6 エラー発生事象例

本章では、サンプルプログラムでフラッシュメモリの書き込み/消去を行った場合に、エラーが発生する事象をピックアップしました。エラーが発生する事象とその対処法を以下に記します。

(1) ユーザプログラムモードに遷移していない(FLSHE = 1 にしていない)。

FLSHE = 0 の場合、フラッシュメモリコントロールレジスタ(FCCS、FPCS、FECS、FKEY、FMATS、FTDAR、FVACR)を設定しても無効となります。

【サンプルプログラム上のエラー内容】

ダウンロード処理のところでエラーが発生します。

【対処法】

フラッシュメモリコントロールレジスタを使用する前に FLSHE = 1 にしてください。

(2) 消去ブロック番号(FEBS)の設定値ミス、もしくはデータを書き込むフラッシュメモリ領域の設定値ミス。

消去ブロックは、EB0～EB15 までの範囲に設定しないと消去処理エラーになります。

データを書き込む場合は、フラッシュメモリの領域(H'000000～H'07FFFF)を超えて書き込みを行うと書き込み処理エラーになります。

【サンプルプログラム上のエラー内容】

消去ブロック番号(BLOCK_NUM)の設定を間違えるとシンボル設定値ミスとしてエラー処理されます。

フラッシュメモリの領域を超えて書き込み(PR_FL_TOP と PROG_N の設定ミス)を行うとシンボル設定値ミスとしてエラー処理されるか、もしくは書き込み処理のところでエラーが発生します。

なお、サンプルプログラムでは消去ブロック EB0(H'000000～H'000FFF)の範囲はプログラム領域として使用しているため、書き込み/消去できない仕様となっています。間違って設定するとシンボル設定値ミスとしてエラー処理されます。

【対処法】

消去ブロック番号の設定を EB1～EB15 までの範囲にしてください。

データを書き込む場合は、フラッシュメモリの領域(H'001000～H'07FFFF)を超えないようにしてください。

(3) FKEY の設定値ミス。

FKEY は、内蔵プログラムのダウンロードとフラッシュメモリの書き込み/消去を許可するソフトウェアプロテクトのレジスタです。FKEY の設定値を間違えると各処理が行われません。

【サンプルプログラム上のエラー内容】

FKEY の設定値を間違えている処理のところでエラーが発生します。

【対処法】

エラーが発生した処理の FKEY の設定値を確認してください。

・H'A5 → SCO ビットの書き込み(ダウンロード)を許可します(H'A5 以外では SCO ビットのセットは不可)。

・H'5A → 書き込み/消去を許可します(H'5A 以外ではソフトウェアプロテクト状態)。

(4) 内蔵 RAM に書き込み/消去プログラムをダウンロードしていない。

PPVS = 0、EPVB = 0 のまま SCO = 1(ダウンロード要求)にしても書き込み/消去プログラムは内蔵 RAM にダウンロードされません(ダウンロードするプログラムを選択していない状態です)。PPVS = 1 にするとダウンロードするプログラムに書き込みプログラムを選択し、EPVB = 1 にすると消去プログラムを選択します。

【サンプルプログラム上のエラー内容】

・PPVS = 0 のまま SCO = 1 にすると、書き込みプログラムのダウンロード処理のところでエラーが発生します。

・EPVB = 0 のまま SCO = 1 にすると、消去プログラムのダウンロード処理のところでエラーが発生します。

【対処法】

SCO = 1（ダウンロード要求）にする前に PPVS = 1（書き込みプログラムを選択）、もしくは EPVB = 1（消去プログラムを選択）に設定してください。

(5) フラッシュメモリの初期化処理を行わずに消去を行っている。

初期化処理を行わずにフラッシュメモリの書き込み/消去処理を行うと、正常に処理が行われずエラーが発生します。

【サンプルプログラム上のエラー内容】

初期化処理のところでエラーが発生します。

【対処法】

フラッシュメモリの書き込み/消去処理を行う前に、初期化処理を行ってください。

(6) フラッシュメモリの初期化処理のときにフラッシュプログラム/イレーズ周波数パラメータ（FPEFEQ）の設定値が CPU の動作周波数を超えている。

FPEFEQ の設定値が、使用している CPU の動作周波数を超えていると初期化処理でエラーが発生します。

【サンプルプログラム上のエラー内容】

初期化処理のところでエラーが発生します（サンプルプログラムでは、動作周波数 33MHz となっています）。

【対処法】

FPEFEQ の設定値が、使用している CPU の動作周波数を超えないようにしてください。

(7) データを書き込む領域を消去していない状態（“H’FF”以外の値が書き込まれている状態）で書き込みを行う。

フラッシュメモリへ書き込む領域が消去されていない場合は、書き込み処理でエラーが発生します。

【サンプルプログラム上のエラー内容】

書き込み処理のところでエラーが発生します。

【対処法】

フラッシュメモリの書き込みを行う前に、書き込み領域を消去（オール“H’FF”）してください。

(8) 書き込み終了処理を行うときの ER0、ER1 の設定値ミス。

書き込み終了処理を行う直前に ER0 に“H’F0F0F0F0”、ER1 に“H’0F0F0F0F”を設定する必要があります。この設定値を間違えると書き込み終了処理でエラーが発生します。

【サンプルプログラム上のエラー内容】

書き込み終了処理のところでエラーが発生します。

【対処法】

書き込み終了処理を行う直前に ER0 = “H’F0F0F0F0”、ER1 = “H’0F0F0F0F” に設定しているか確認してください。

(9) 書き込み/消去プログラムをダウンロードした RAM 領域（FTDAR で設定したアドレスから 4 k バイト分）に間違ってデータを書き込んでしまっている。

FTDAR で設定したアドレスから 4k バイト分の領域に間違ってデータを書き込んでしまうと、ダウンロードした書き込み/消去プログラムを壊してしまい、フラッシュメモリの書き込み/消去処理を行えません。

【サンプルプログラム上のエラー内容】

FTDAR で設定したアドレスにジャンプしたままリターンせず、暴走します（サンプルプログラムでは、消去プログラムを“H’ FF9000～H’ FF9FFF”、書き込みプログラムを“H’ FFA000～H’ FFAFFF”の RAM 領域にダウンロードしています）。

【対処法】

FTDAR で設定したアドレスから 4k バイト分の領域にデータを書き込まないでください。

(10) ダウンロードした書き込み/消去プログラムの実行アドレス指定ミス。

各処理（初期化、消去、書き込み、書き込み終了）を行うときに指定する実行アドレス（各処理プログラムの先頭アドレス）を間違えると各処理が行われません。

【サンプルプログラム上のエラー内容】

間違えた実行アドレスにジャンプしたまま、暴走します。

【対処法】

実行アドレスを間違えていないか確認してください。下記に各処理の実行アドレスを示します。

・初期化処理　→ ダウンロードした書き込み/消去プログラムの先頭アドレス（FTDAR） + 32
・消去処理　→ ダウンロードした書き込み/消去プログラムの先頭アドレス（FTDAR） + 16
・書き込み処理　→ ダウンロードした書き込み/消去プログラムの先頭アドレス（FTDAR） + 16
・書き込み終了処理 → ダウンロードした書き込み/消去プログラムの先頭アドレス（FTDAR） + 16

7. サンプルプログラムリスト

7.1 リセットプログラム

```
/************************************************************************/
/*                                                                      */
/*  FILE        :resetprg.c                                             */
/*  DATE        :Tue, Jan 22, 2008                                      */
/*  DESCRIPTION :Reset Program                                          */
/*  CPU TYPE    :H8S/2378R                                              */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.4.9).      */
/*                                                                      */
/************************************************************************/

#include      <machine.h>
#include      <_h_c_lib.h>
#include      "typedefine.h"
#include      "stacksct.h"

extern void main(void);
__entry(vect=0) void PowerON_Reset(void);

#pragma section ResetPRG

__entry(vect=0) void PowerON_Reset(void)
{
         set_imask_ccr((_UBYTE)1);
        _INITSCT();

        set_imask_ccr((_UBYTE)0);

        main();

        sleep();
}
```

7.2 メインプログラム

```
/************************************************************************/
/*                                                                      */
/*  FILE            :H8S_2378R_MAIN.c                                   */
/*  DATE            :Thu, Nov 15, 2007                                  */
/*  DESCRIPTION     :Main program file                                  */
/*  CPU TYPE        :H8S/2378R                                          */
/*  BOARD           :HSB8S2378RE                                        */
/*                                                                      */
/************************************************************************/

/************************************************************************/
/*  Include file                                                        */
/************************************************************************/
#include <machine.h>
#include "iodefine.h"

/************************************************************************/
/*  Definition of external reference                                    */
/************************************************************************/
extern void flash_erpr_128(void);
```

```
/*****************************************************************************/
/*  Function prototype declaration                                           */
/*****************************************************************************/
void init(void);
void flash_ram_tr(void);
void finiprg(void);

/*****************************************************************************/
/*  Main program                                                             */
/*****************************************************************************/
void main(void)
{
        init();                                 //Initialize
        flash_ram_tr();                         //RAM transfer function
        flash_erpr_128();                       //FLASH erase/programming function

        while(1){}
}
/*****************************************************************************/
/*  Initialize                                                               */
/*****************************************************************************/
void init(void)
{
        SCKCR.BIT.STCS = 1;                     //Frequency Multiplication Factor Switching Mode Select
        SCKCR.BIT.SCK = 0;                      //Frequency no division
        PLLCR.BIT.STC = 2;                      //Frequency Multiply by 4
        MSTPCR.WORD = 0x7FFF;                   //All module clocks stop mode enabled
        EXMSTPCR.BIT._IIC20 = 0;                //IIC2_0 bus clocks stop mode select disable
}
/*****************************************************************************/
/*  FLASH program transfer to RAM                                            */
/*****************************************************************************/
void flash_ram_tr(void)
{
        volatile unsigned char *fl_p_top;
        volatile unsigned char *fl_p_end;
        volatile unsigned char *tr_ram_p_top;

        fl_p_top = ((volatile unsigned char*)__sectop("PROMPRG"));      //Top address of section"ROMPRG"
        fl_p_end = ((volatile unsigned char*)__secend("PROMPRG"));      //End address of section"ROMPRG"

        tr_ram_p_top = ((volatile unsigned char*)__sectop("PRAMPRG"));  //Top address of section"RAMPRG"

        for(; fl_p_top < fl_p_end; fl_p_top++, tr_ram_p_top++){         //FLASH program transfer to RAM
                *tr_ram_p_top = *fl_p_top;
        }
}
/*****************************************************************************/
/*  Finish program function                                                  */
/*****************************************************************************/
#pragma section FiniPRG
void finiprg(void)
{
        while(1){}
}
```

7.3 フラッシュメモリ書き込み/消去プログラム

```
/*********************************************************************/
/*                                                                   */
/*  FILE        :H8S_FlashPRG.c                                      */
/*  DATE        :Thu, Oct 25, 2007                                   */
/*  DESCRIPTION :FLASH erase/programming program file                */
/*  CPU TYPE    :H8S/2378R                                           */
/*  BOARD       :HSB8S2378RE                                         */
/*  SYSTEM CLOCK:33MHz                                               */
/*                                                                   */
/*********************************************************************/

/*********************************************************************/
/*  Include file                                                     */
/*********************************************************************/
#include <machine.h>

/*********************************************************************/
/*  Prototype definition                                             */
/*********************************************************************/
void flash_erase(void);
void flash_prog(void);

/*********************************************************************/
/*  Asemble symbols definition                                       */
/*********************************************************************/
#pragma inline_asm(flash_symbol)
void flash_symbol(void)
{
/* Store the block number of the erase destination on the FLASH */
/* Note : An error occurs if a number outside the range from H'00000001 to H'0000000F is set. */
/* Note : Do not select EB0 block, because EB0 block is program area. */
/************************************************************
  BLOCK_NUM    Block    Size(byte)    Address
  H'00000000   EB0      4k            H'000000-H'000FFF
  H'00000001   EB1      4k            H'001000-H'001FFF
  H'00000002   EB2      4k            H'002000-H'002FFF
  H'00000003   EB3      4k            H'003000-H'003FFF
  H'00000004   EB4      4k            H'004000-H'004FFF
  H'00000005   EB5      4k            H'005000-H'005FFF
  H'00000006   EB6      4k            H'006000-H'006FFF
  H'00000007   EB7      4k            H'007000-H'007FFF
  H'00000008   EB8      32k           H'008000-H'00FFFF
  H'00000009   EB9      64k           H'010000-H'01FFFF
  H'0000000A   EB10     64k           H'020000-H'02FFFF
  H'0000000B   EB11     64k           H'030000-H'03FFFF
  H'0000000C   EB12     64k           H'040000-H'04FFFF
  H'0000000D   EB13     64k           H'050000-H'05FFFF
  H'0000000E   EB14     64k           H'060000-H'06FFFF
  H'0000000F   EB15     64k           H'070000-H'07FFFF
************************************************************/
/* FLASH erase start block number */
BLOCK_NUM:      .ASSIGN H'00000001

/* Number of erase */
/* Note : An error occurs if a number outside the range from H'01 to H'0F is set. */
/* Note : ERASE_N max value(In case of BLOCK_NUM = H'00000001) => H'0F */
/* Note : ERASE_N max value(In case of BLOCK_NUM = H'0000000F) => H'01 */
```

```
/* Example1 : BLOCK_NUM=H'00000003, ERASE_N=H'04 => EB3 - EB6 block area erase. */
/* Example2 : BLOCK_NUM=H'00000001, ERASE_N=H'0F => All area(EB1 - EB15 block) erase. */
/************************************************************
  BLOCK_NUM    ERASE_N
  H'00000001   H'01 - H'0F
  H'00000002   H'01 - H'0E
  H'00000003   H'01 - H'0D
  H'00000004   H'01 - H'0C
  H'00000005   H'01 - H'0B
  H'00000006   H'01 - H'0A
  H'00000007   H'01 - H'09
  H'00000008   H'01 - H'08
  H'00000009   H'01 - H'07
  H'0000000A   H'01 - H'06
  H'0000000B   H'01 - H'05
  H'0000000C   H'01 - H'04
  H'0000000D   H'01 - H'03
  H'0000000E   H'01 - H'02
  H'0000000F   H'01
************************************************************/
ERASE_N:        .ASSIGN H'0F

/* Store the start address of the programming destination on the FLASH */
/* Note : An error occurs if a number outside the range from H'00001000 to H'0007F000 is set. */
/* Note : Do not setting value H'00000000, because H'00000000 - H'00000FFF is program area. */
PR_FL_TOP:      .EQU    H'00001000

/* Number of programmig(Tortal programming data size = H'1000(4kbyte) * PROG_N) */
/* Note : An error occurs if a number outside the range from H'01 to H'7F is set. */
/* Note : PROG_N max value(In case of PR_FL_TOP = H'00001000) => H'7F(Programming data size is 508kbyte) */
/* Note : PROG_N max value(In case of PR_FL_TOP = H'0007F000) => H'01(Programming data size is 4kbyte) */
PROG_N:         .ASSIGN H'01

/************************************************************************/
/*  Do not modify symbols                                               */
/************************************************************************/
/* FLASH programming data size => 4kbyte */
PROG_SIZE:      .ASSIGN H'20
/* Store the start address of the area which stores the program data for FLASH */
DATA_RAM_TOP:   .EQU    H'00FF8000
/* Store the start address of the area which download internal FLASH erase program */
EDL_RAM_TOP:    .EQU    H'00FF9000
/* Store the start address of the area which download internal FLASH programming program */
PDL_RAM_TOP:    .EQU    H'00FFA000
/* Internal erase program download result pass/fail register */
E_DPFR:         .EQU    H'00FFB000
/* FLASH erase process execute result pass/fail register */
E_FPFR:         .EQU    H'00FFB002
/* FLASH erase program execute result check flag */
E_RTN:          .EQU    H'00FFB004
/* Internal programming program download pass/fail register */
P_DPFR:         .EQU    H'00FFB006
/* FLASH programming process execute result pass/fail register */
P_FPFR:         .EQU    H'00FFB008
/* FLASH programming program execute result check flag */
P_RTN:          .EQU    H'00FFB00A
/* Counter */
E_COUNT:        .EQU    H'00FFB00C
P_COUNT1:       .EQU    H'00FFB010
```

```
P_COUNT2:       .EQU    H'00FFB014
COUNT1:         .EQU    H'00FFB018
COUNT2:         .EQU    H'00FFB019
PCMP_COUNT1:    .EQU    H'00FFB01A
PCMP_COUNT2:    .EQU    H'00FFB01E
COUNT3:         .EQU    H'00FFB022
}
/************************************************************************/
/*  FLASH erase/programming function                                    */
/************************************************************************/
#pragma section ROMPRG
void flash_erpr_128(void)
{
        flash_erase();                          //FLASH erase function
        flash_prog();                           //FLASH programming function

        while(1){}
}
/************************************************************************/
/*  FLASH memory erase function                                         */
/************************************************************************/
void flash_erase(void)
{
        {
        #pragma asm
        /* BLOCK_NUM == H'00000000 : Setting value error */
                SUB.L   ER3, ER3
                MOV.L   #BLOCK_NUM, ER4
                CMP.L   ER3, ER4
                BEQ     E_BREAK1
        /* BLOCK_NUM > H'0000000F : Setting value error */
                MOV.L   #H'0000000F, ER3
                MOV.L   #BLOCK_NUM, ER4
                SUB.L   ER4, ER3
                BMI     E_BREAK1
        /* ERASE_N == H'00 : Setting value error */
                SUB.B   R3L, R3L
                MOV.B   #ERASE_N, R3H
                CMP.B   R3L, R3H
                BEQ     E_BREAK1
        /* ERASE_N > H'0F : Setting value error */
                MOV.B   #H'0F, R3L
                MOV.B   #ERASE_N, R3H
                SUB.B   R3H, R3L
E_PARA_CK:
                BMI     E_BREAK1
        /* FLSHE = 1 : FLASH control register selected */
                BSET.B  #3, @H'00FFFF3D:8
        /* EPVB  = 1 : On-chip erase program is selected */
                BSET.B  #0, @H'00FFFFC6:8
        /* FTDAR = 0x00 : H'FF9000 is specified as a start address to download an on-chip program */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFCA:8
        /* DPFR = 0xFF : DPFR(Downroad Pass/Fail Register) initialization */
        MOV.B   #H'FF, R3L
                MOV.B   R3L, @EDL_RAM_TOP:16
        /* RTN = 0 */
                SUB.B   R3L, R3L
                MOV.B   R3L, @E_RTN:16
```

```
E_DL:
        /* FKEY  = 0xA5 : FLASH key code H'A5(Writing to the SCO bit is enabled) */
                MOV.B   #H'A5, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* SCO   = 1 : On-chip erase program is downloaded to the on-chip RAM is occurred */
                BSET.B  #0, @H'00FFFFC4:8
                NOP
                NOP
                NOP
                NOP
        /* FKEY  = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* DPFR != 0 : Download process is ended abnormally(error occurs) */
                MOV.B   @EDL_RAM_TOP:16, R3H
E_DL_CK:
                MOV.B   R3H, @E_DPFR:16
                OR.B    R3H, R3L
                BNE     E_BREAK2
        /* RTN = 1 */
                MOV.B   #H'01, R3L
                MOV.B   R3L, @E_RTN:16
E_INI:
        /* FPEFEQ : System clock = 33MHz */
                MOV.L   #H'00000CE4, ER0
        /* FUBRA : User branch disable */
                MOV.L   #H'00000000, ER1
        /* FLASH initialization */
                MOV.L   #EDL_RAM_TOP+32, ER2
                JSR     @ER2
                NOP
        /* FPFR != 0 : FLASH initialize is ended abnormally(error occurs) */
                SUB.B   R3L, R3L
E_INI_CK:
                MOV.B   R0L, @E_FPFR:16
                OR.B    R0L, R3L
                BNE     E_BREAK2
        /* RTN = 2 */
                MOV.B   #H'02, R3L
                MOV.B   R3L, @E_RTN:16
        /* Disable interrupt */
                ORC.B   #H'80, CCR
        /* FKEY  = 0x5A : FLASH key code H'5A(Programming is enabled) */
                MOV.B   #H'5A, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* Counter initialization */
                MOV.L   #H'00000001, ER0
                MOV.L   ER0, @E_COUNT:32
                MOV.B   #ERASE_N, R3L
                MOV.B   R3L, @COUNT1:16
        /* FEBS : Store the block number of the erase destination on the FLASH */
                MOV.L   #BLOCK_NUM, ER0
        /* Erase process execution */
E_EXE:
                MOV.L   #EDL_RAM_TOP+16, ER2
                JSR     @ER2
                NOP
        /* FPFR != 0 : Erase process is ended abnormally(error occurs) */
                SUB.B   R3L, R3L
```

```
E_EXE_CK:
                MOV.B   R0L, @E_FPFR:16
                OR.B    R0L, R3L
                BNE     E_BREAK2
        /* Counter */
                MOV.L   #BLOCK_NUM, ER0
                MOV.L   @E_COUNT:32, ER1
                ADD.L   ER1, ER0
                INC.L   #1, ER1
                MOV.L   ER1, @E_COUNT:32
                MOV.B   @COUNT1:16, R3L
                DEC.B   R3L
                MOV.B   R3L, @COUNT1:16
                BNE     E_EXE
        /* FKEY  = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* RTN  = 3 */
                MOV.B   #H'03, R3L
                MOV.B   R3L, @E_RTN:16
        /* Erase error check result for register */
                MOV.B   @E_DPFR:16, R0L
                MOV.B   @E_FPFR:16, R1L
                MOV.B   @E_RTN:16,      R2L
                MOV.B   #H'FF, R3L
                MOV.B   #H'FF, R4L
E_ALL_CK:
                MOV.B   #H'FF, R5L
                RTS
E_BREAK1:
        /* FKEY = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* Setting value error result for register */
                MOV.B   #H'0E, R0L
                MOV.B   #H'0E, R1L
                MOV.B   #H'0E, R2L
                MOV.B   #H'FF, R3L
                MOV.B   #H'FF, R4L
                MOV.B   #H'FF, R5L
                BRA     E_LOOP
E_BREAK2:
        /* FKEY = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* Erase error check result for register */
                MOV.B   @E_DPFR:16, R0L
```

```
                MOV.B   @E_FPFR:16, R1L
                MOV.B   @E_RTN:16, R2L
                MOV.B   #H'FF, R3L
                MOV.B   #H'FF, R4L
                MOV.B   #H'FF, R5L
E_LOOP:
        /* LOOP */
                NOP
                BRA     E_LOOP
        #pragma endasm
        }
}
/************************************************************************/
/*  FLASH memory programming function                                   */
/************************************************************************/
void flash_prog(void)
{
        {
        #pragma asm
        /* PR_FL_TOP == H'00000000 : Setting value error */
                SUB.L   ER6, ER6
                MOV.L   #PR_FL_TOP, ER4
                CMP.L   ER6, ER4
                BEQ     P_BREAK1
        /* PR_FL_TOP > H'0007F000 : Setting value error */
                MOV.L   #H'0007F000, ER3
                MOV.L   #PR_FL_TOP, ER4
                SUB.L   ER4, ER3
                BMI     P_BREAK1
        /* PROG_N == H'00 : Setting value error */
                SUB.B   R3L, R3L
                MOV.B   #PROG_N, R3H
                CMP.B   R3L, R3H
                BEQ     P_BREAK1
        /* PROG_N > H'7F : Setting value error */
                MOV.B   #H'7F, R3L
                MOV.B   #PROG_N, R3H
                SUB.B   R3H, R3L
P_PARA_CK:
                BMI     P_BREAK1
        /* FLSHE = 1 : FLASH control register selected */
                BSET.B  #3, @H'00FFFF3D:8
        /* PPVS  = 1 : On-chip programming program is selected */
                BSET.B  #0, @H'00FFFFC5:8
        /* FTDAR = 0x01 : H'FFA000 is specified as a start address to download an on-chip program */
                MOV.B   #H'01, R3L
        MOV.B   R3L, @H'00FFFFCA:8
        /* DPFR = 0xFF : DPFR(Downroad Pass/Fail Register) initialization */
        MOV.B   #H'FF, R3L
                MOV.B   R3L, @PDL_RAM_TOP:16
        /* RTN = 0 */
                SUB.B   R3L, R3L
                MOV.B   R3L, @P_RTN:16
P_DL:
        /* FKEY  = 0xA5 : FLASH key code H'A5(Writing to the SCO bit is enabled) */
                MOV.B   #H'A5, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* SCO   = 1 : On-chip programming program is downloaded to the on-chip RAM is occurred */
                BSET.B  #0, @H'00FFFFC4:8
```

```
                NOP
                NOP
                NOP
                NOP
        /* FKEY  = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* DPFR != 0 : Download process is ended abnormally(error occurs) */
                MOV.B   @PDL_RAM_TOP:16, R3H
P_DL_CK:
                MOV.B   R3H, @P_DPFR:16
                OR.B    R3H, R3L
                BNE     P_BREAK3
        /* RTN = 4 */
                MOV.B   #H'04, R3L
                MOV.B   R3L, @P_RTN:16
P_INI:
        /* FPEFEQ : System clock = 33MHz */
                MOV.L   #H'00000CE4, ER0
        /* FUBRA : User branch disable */
                MOV.L   #H'00000000, ER1
        /* FLASH initialization */
                MOV.L   #PDL_RAM_TOP+32, ER2
                JSR     @ER2
                NOP
        /* FPFR != 0 : FLASH initialize is ended abnormally(error occurs) */
                SUB.B   R3L, R3L
P_INI_CK:
                MOV.B   R0L, @P_FPFR:16
                OR.B    R0L, R3L
                BNE     P_BREAK3
        /* RTN = 5 */
                MOV.B   #H'05, R3L
                MOV.B   R3L, @P_RTN:16
        /* Counter initialization */
                MOV.B   #PROG_N, R3L
                MOV.B   R3L, @COUNT2:16
                MOV.L   #PR_FL_TOP, ER3
                MOV.L   ER3, @P_COUNT1:32
                MOV.L   #DATA_RAM_TOP, ER3
                MOV.L   ER3, @P_COUNT2:32
IIC_EXE:
        /* Counter initialization */
                MOV.B   #PROG_SIZE, R3L
                MOV.B   R3L, @COUNT1:16
        /* IIC slave receive function execution */
                .IMPORT _iic_sr_128
                JSR     @_iic_sr_128
        /* Disable interrupt */
                ORC.B   #H'80, CCR
        /* FKEY  = 0x5A : FLASH key code H'5A(Programming is enabled) */
                MOV.B   #H'5A, R3L
                MOV.B   R3L, @H'00FFFFC8:8
P_EXE:
        /* FMPAR : Store the start address of the programming destination on the FLASH */
                MOV.L   @P_COUNT1:32, ER1
        /* FMPDR : Store the start address of the area which stores the program data for the FLASH */
                MOV.L   @P_COUNT2:32, ER0
        /* Programming process execution */
```

```
                MOV.L   #PDL_RAM_TOP+16, ER2
                JSR     @ER2
                NOP
        /* FPFR != 0 : Programming process is ended abnormally(error occurs) */
                SUB.B   R3L, R3L
                MOV.B   R0L, @P_FPFR:16
                OR.B    R0L, R3L
                BNE     P_BREAK3
        /* Programming data compare check */
                MOV.L   @P_COUNT1:32, ER1
                MOV.L   ER1, @PCMP_COUNT1:32
                MOV.L   @P_COUNT2:32, ER0
                MOV.L   ER0, @PCMP_COUNT2:32
                MOV.B   #H'80, R3L
                MOV.B   R3L, @COUNT3:16
COMP80:
                MOV.L   @PCMP_COUNT1:32, ER1
                MOV.L   @PCMP_COUNT2:32, ER2
                MOV.B   @ER1, R3L
                MOV.B   @ER2, R3H
                CMP.B   R3L, R3H
                BNE     P_BREAK2
                MOV.L   @PCMP_COUNT1:32, ER3
                INC.L   #1, ER3
                MOV.L   ER3, @PCMP_COUNT1:32
                MOV.L   @PCMP_COUNT2:32, ER3
                INC.L   #1, ER3
                MOV.L   ER3, @PCMP_COUNT2:32
                MOV.B   @COUNT3:16, R3L
                DEC.B   R3L
                MOV.B   R3L, @COUNT3:16
                BNE     COMP80
        /* FMPAR += 0x80, FMPDR += 0x80 */
                MOV.L   #H'00000080, ER3
                MOV.L   @P_COUNT1:32, ER4
                ADD.L   ER3, ER4
                MOV.L   ER4, @P_COUNT1:32
                MOV.L   @P_COUNT2:32, ER4
                ADD.L   ER3, ER4
                MOV.L   ER4, @P_COUNT2:32
                MOV.B   @COUNT1:16, R3L
                DEC.B   R3L
                MOV.B   R3L, @COUNT1:16
                BNE     P_EXE
        /* FKEY  = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* IIC slave receive counter */
                MOV.L   #DATA_RAM_TOP, ER3
                MOV.L   ER3, @P_COUNT2:32
                MOV.B   @COUNT2:16, R3L
                DEC.B   R3L
                MOV.B   R3L, @COUNT2:16
                BNE     IIC_EXE
        /* RTN  = 6 */
                MOV.B   #H'06, R3L
                MOV.B   R3L, @P_RTN:16
```

```
        /* Disable interrupt */
                ORC.B   #H'80, CCR
        /* FKEY  = 0x5A : FLASH key code H'5A(Programming is enabled) */
                MOV.B   #H'5A, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* Programming finish process execution */
                MOV.L   #H'F0F0F0F0, ER0
                MOV.L   #H'0F0F0F0F, ER1
                MOV.L   #PDL_RAM_TOP+16, ER2
                JSR     @ER2
                NOP
        /* FPFR != 0 : Programming finish process is ended abnormally(error occurs) */
                SUB.B   R3L, R3L
P_EXE_CK:
                MOV.B   R0L, @P_FPFR:16
                OR.B    R0L, R3L
                BNE     P_BREAK3
        /* FKEY  = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* RTN  = 7 */
                MOV.B   #H'07, R3L
                MOV.B   R3L, @P_RTN:16
        /* Tortal error check result for register */
                MOV.B   @E_DPFR:16, R0L
                MOV.B   @E_FPFR:16, R1L
                MOV.B   @E_RTN:16,  R2L
                MOV.B   @P_DPFR:16, R3L
                MOV.B   @P_FPFR:16, R4L
P_ALL_CK:
                MOV.B   @P_RTN:16,  R5L
        /* Jump to prgfini function(return to ROM) */
                .IMPORT _finiprg
                JSR     @_finiprg
P_BREAK1:
        /* FKEY = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* Setting value error result for register */
                MOV.B   @E_DPFR:16, R0L
                MOV.B   @E_FPFR:16, R1L
                MOV.B   @E_RTN:16,  R2L
                MOV.B   #H'0E, R3L
                MOV.B   #H'0E, R4L
                MOV.B   #H'0E, R5L
                BRA     P_LOOP
P_BREAK2:
        /* FKEY = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
```

```
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* Programming compare error result for register */
                MOV.B   @E_DPFR:16, R0L
                MOV.B   @E_FPFR:16, R1L
                MOV.B   @E_RTN:16,  R2L
                MOV.B   #H'E0, R3L
                MOV.B   #H'E0, R4L
                MOV.B   #H'E0, R5L
                BRA     P_LOOP
P_BREAK3:
        /* FKEY = 0x00 : FLASH key code H'00(Software protect is enabled) */
                SUB.B   R3L, R3L
                MOV.B   R3L, @H'00FFFFC8:8
        /* FLSHE = 0 : FLASH control register not selected */
                BCLR.B  #3, @H'00FFFF3D:8
        /* Enable interrupt */
                ANDC.B  #H'7F, CCR
        /* Tortal error check result for register */
                MOV.B   @E_DPFR:16, R0L
                MOV.B   @E_FPFR:16, R1L
                MOV.B   @E_RTN:16,  R2L
                MOV.B   @P_DPFR:16, R3L
                MOV.B   @P_FPFR:16, R4L
                MOV.B   @P_RTN:16,  R5L
P_LOOP:
        /* LOOP */
                NOP
                BRA     P_LOOP
        #pragma endasm
        }
}
```

7.4 IIC バスインタフェース 2(IIC2)スレーブ受信ヘッダファイル

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :H8S_iic_sr.h                                          */
/*  DATE        :Thu, Nov 15, 2007                                     */
/*  DESCRIPTION :IIC2 slave mode prameter setting file                 */
/*  CPU TYPE    :H8S/2378R                                             */
/*  BOARD       :HSB8S2378RE                                           */
/*                                                                     */
/***********************************************************************/

/*****************************************************************/
/*  Slave address                                                */
/*****************************************************************/
#define SLAVE_ADDR    0x08          //Slave address

/*****************************************************************/
/*  IIC mode                                                     */
/*****************************************************************/
#define MODE_ST      2              //Slave transfer mode
#define MODE_SR      1              //Slave receive mode
#define MODE_IDLE    0              //IIC idle mode
```

```
/************************************************************/
/*  Receive data size                                       */
/************************************************************/
#define DTNUM    0x1000              //4kbyte
```

7.5　IIC バスインタフェース 2(IIC2)スレーブ受信プログラム

```
/************************************************************************/
/*                                                                      */
/*  FILE             :H8S_iic_sr.c                                      */
/*  DATE             :Thu, Nov 15, 2007                                 */
/*  DESCRIPTION      :IIC2 slave receive mode program file              */
/*  CPU TYPE         :H8S/2378R                                         */
/*  BOARD            :HSB8S2378RE                                       */
/*                                                                      */
/*  SYSTEM CLOCK     :33MHz                                             */
/*  TRANSFER RATE    :98.2kbps                                          */
/*  SLAVE ADDRESS    :0x08                                              */
/*  RECEIVE DATA SIZE:4kbyte                                            */
/*  USED IIC BUS     :UART0                                             */
/*                                                                      */
/************************************************************************/

/************************************************************************/
/*  Include file                                                        */
/************************************************************************/
#include <machine.h>
#include "iodefine.h"
#include "H8S_iic_sr.h"

/************************************************************************/
/*  Global variable declaration                                         */
/************************************************************************/
unsigned char *sr_ram_data;                     //Receive data buffer
unsigned char iic_mode;                         //IIC mode
unsigned short sr_cnt;                          //IIC slave receive data counter

/************************************************************************/
/*  Function prototype declaration                                      */
/************************************************************************/
void iic_sr_128(void);
void iic_init(void);
void iici0_int(void);
void receive_stop_condition(void);
void slave_receive(void);

/************************************************************************/
/*  Vector address                                                      */
/************************************************************************/
#pragma interrupt (iici0_int(vect = 116))       //H'01D0 : IIC interrupt

/************************************************************************/
/*  IIC slave receiver mode program                                     */
/************************************************************************/
#pragma section ROMPRG
void iic_sr_128(void)
{
        unsigned short  i;
```

```
        sr_ram_data = ((unsigned char*)0xFF8000);     //Set top address of Receive data buffer

        for (i = 0; i < DTNUM; i++) {                 //Receive RAM area : clear to 0
                *(sr_ram_data + i) = 0;
        }

        iic_init();                                   //IIC initialization

        set_imask_ccr(0);                             //Enable interrupt

        while (sr_cnt < DTNUM + 1) {}                 //Stand by receive master transfer data

        set_imask_ccr(1);                             //Disable interrupt
}
/************************************************************************/
/*  IIC bus initialization                                              */
/************************************************************************/
void iic_init(void)
{
        unsigned char  tmp;

        IIC20.ICCRA.BIT.ICE = 1;                      //IIC transfer operations : Enable
        IIC20.SAR.BYTE = SLAVE_ADDR;                  //Slave address
        IIC20.ICCRA.BYTE = 0x8C;                      //Transfer rate : 33 MHz,98.2 kbps
        IIC20.ICIER.BYTE = 0xBC;                      //Enable Interrupt source, Ack = 1
        tmp = IIC20.ICSR.BYTE;
        IIC20.ICSR.BYTE = 0x00;                       //All status clear

        sr_cnt = 0;                                   //IIC slave receive data counter
        iic_mode = MODE_IDLE;                         //IIC idle mode
}
/************************************************************************/
/*  IIC2 ch0 Interrupt                                                  */
/************************************************************************/
void iici0_int(void)
{
        if (IIC20.ICSR.BIT.STOP == 1) {
                receive_stop_condition();
        }
        else {
                switch (iic_mode) {                       //Mode check
                case MODE_ST:                             //Slave transfer
                        while(1){}
                case MODE_SR:                             //Slave reception
                        slave_receive();
                        break;
                case MODE_IDLE:
                        if (IIC20.ICCRA.BIT.TRS == 1) {
                                iic_mode = MODE_ST;               //Slave transfer
                                while(1){}
                        }
                        else {
                                iic_mode = MODE_SR;               //Slave reception
                                slave_receive();
                        }
                        break;
                default:
                        break;
```

```
                }
        }
}
/*****************************************************************************/
/*  Receive stop condition                                                   */
/*****************************************************************************/
void receive_stop_condition(void)
{
        unsigned char  tmp;

        IIC20.ICCRA.BIT.RCVD = 0;                   //Enables next reception
        IIC20.ICCRA.BYTE &= 0xCFu;                  //MST,TRS = 00, slave receive mode

        tmp = IIC20.ICSR.BYTE;
        IIC20.ICSR.BYTE  = 0x00;                    //All status clear

        IIC20.ICIER.BYTE = 0xBC;                    //Enable Interrupt source, Ack = 1
        iic_mode = MODE_IDLE;                       //IIC idle mode
}
/*****************************************************************************/
/*  Slave reception                                                          */
/*****************************************************************************/
void slave_receive(void)
{
        unsigned char tmp;

        if (sr_cnt == 0) {
                IIC20.ICIER.BIT.ACKBT = 0;
                tmp = IIC20.ICDRR;                              //Dummy read
        }
        else {
                *(sr_ram_data + (sr_cnt - 1)) = IIC20.ICDRR;    //Store the received data
        }
        sr_cnt++;
}
```

8. よくある質問

8.1　フラッシュメモリの書き込み/消去を行う上で注意すべき点はありますか？

・フラッシュメモリの書き込み/消去処理を行う場合は、処理にかかる時間を考慮してください。書き込み時間は、128 バイト同時書き込みにて 1ms（typ）、1 バイト当たり換算にて 8μs、消去時間は 64k ブロック当たり 750ms（typ）です。

・内蔵プログラムのダウンロード中はフラッシュメモリマットが組み込みプログラム格納領域と入れ替わります。また、書き込み/消去時はフラッシュメモリマットの読み出しはできないため、ダウンロード以降書き込み/消去完了までの一連の手続きプログラムはフラッシュメモリ以外（内蔵 RAM 上など）で実行するようにしてください。

・フラッシュメモリの書き込みにおいては事前に対象領域が消去されている必要があります。

・書き込み/消去処理中は、すべての割り込みを禁止する必要があります。

・書き込み/消去処理中はフラッシュメモリ内部に高電圧が印加されていますので、書き込み/消去処理中にはリセット、ハードウェアスタンバイへの遷移は行わないようにしてください。フラッシュメモリにダメージを与え破壊する可能性があります。誤って、リセットしてしまった場合は、100μs の通常より長いリセット入力期間のあとにリセットリリースしてください。

・初期化ルーチンを含む書き込みプログラム、または初期化ルーチンを含む消去プログラムのコードサイズはそれぞれ 4k バイト以内です。よって、CPU クロック周波数が、35MHz の場合、それぞれ最大で 60μs のダウンロード時間となります。

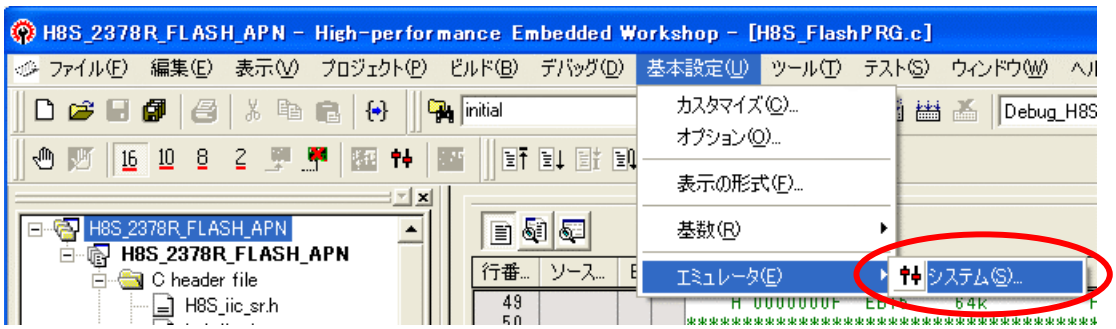
・1 回の書き込み処理では 128 バイトの書き込みを行います。128 バイトを超える書き込みを行う場合は、書き込み先アドレス/書き込みデータのパラメータを 128 バイト単位で更新して書き込みを繰り返します。128 バイト未満の書き込みの場合も無効データを埋め込んで 128 バイトにそろえる必要があります。

8.2　フラッシュメモリの書き込み/消去が正常に終了したはずなのにメモリウィンドウで確認すると値が全く変わっていないのは何故でしょうか？

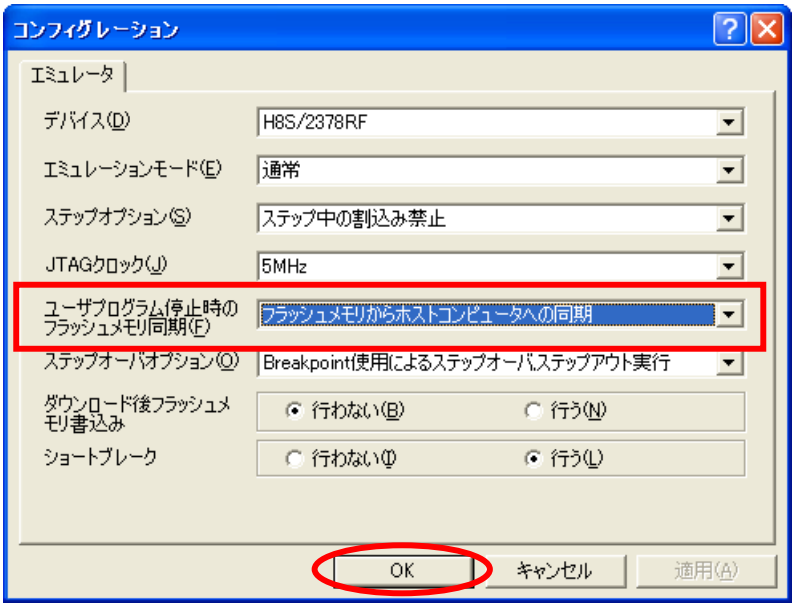
フラッシュメモリの書き込み/消去結果をホストコンピュータ側（メモリウィンドウ上）へ表示させるには、フラッシュメモリからホストコンピュータへの同期設定を行う必要があります。以下に同期設定の手順を示します。

【同期設定の手順】

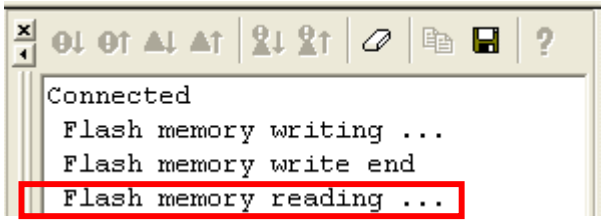
[基本設定]メニューの[エミュレータ]の中の[システム]を選択してください。



[コンフィグレーション]ダイアログボックスが表示されるので、[ユーザプログラム停止時のフラッシュメモリ同期]欄を“フラッシュメモリからホストコンピュータへの同期”に選択して[OK]ボタンを押してください。



なお、フラッシュメモリの内容がホストコンピュータ側へ読み込まれるタイミングはプログラムを停止させたときで、読み込まれている最中はアウトプットウィンドウの[Debug]タブ上に“Flash memory reading ・・・”と表示されます。



8.3 書き込み/消去中にフラッシュメモリにアクセスするような HEW の操作を行うとどうなりますか？

フラッシュメモリの書き込み/消去処理中（高電圧印加中）にフラッシュメモリにアクセスするような HEW の操作（レジスタウィンドウ操作、コマンドラインウィンドウ操作等）を行うと下記のようなエラーメッセージが表示されます。

E10A-USB エミュレータ側で HEW にエラーを返すためフラッシュメモリの書き込み/消去処理には影響を与えませんが、操作によってはフラッシュメモリにダメージを与え破壊する可能性もあります。書き込み/消去処理中はフラッシュメモリへアクセスするような HEW の操作はしないでください。

8.4 書き込み/消去中にメモリウィンドウをスクロールさせると正常に処理が行われない場合があるのは何故でしょうか？

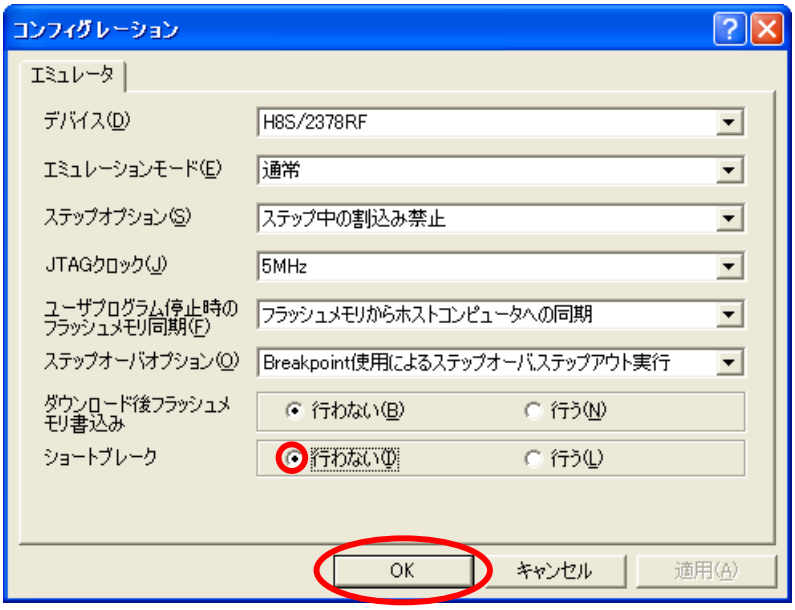
メモリウィンドウをスクロールさせると E10A-USB エミュレータがメモリにアクセスするためにショートブレーク（ユーザプログラムの一時停止）を行ってしまいます。その為、フラッシュメモリの書き込み/消去処理に影響を与えてしまう可能性があります。

下記設定を行うことでエミュレーション実行中にメモリにアクセスしない（ショートブレークを行わない）ようにすることができます。フラッシュメモリの書き込み/消去処理を行う場合はショートブレークを行わないように設定することを推奨いたします。

【ショートブレークを未実行にする設定手順】

[基本設定]メニューの[エミュレータ]の中の[システム]を選択してください。

[コンフィグレーション]ダイアログボックスが表示されるので、[ショートブレーク]欄の[行わない]ラジオボタンにチェックを入れて[OK]ボタンを押してください。



これでフラッシュメモリの書き込み/消去処理中にメモリウィンドウをスクロールさせても E10A-USB エミュレータは、エミュレーション実行中にメモリアクセス（ショートブレーク）を行いません。

9. 関連ドキュメント

E10A-USB エミュレータおよび、HEW には本書で取り上げた機能以外にも便利な機能を豊富に備えています。各製品の仕様の詳細、技術情報、制限事項など有用な情報を記載していますので下記の関連ドキュメントも合わせて参照してください。

【E10A-USB エミュレータ関連ドキュメント】

- H8S、H8SX ファミリ用 E10A-USB エミュレータ ユーザーズマニュアル
- H8S、H8SX ファミリ用 E10A-USB エミュレータ ユーザーズマニュアル 別冊
  (H8S/2378F、H8S/2377F、H8S/2367F、H8S/2368F、H8S/2378RF、H8S/2377RF ご使用時の補足説明)

【High-performance Embedded Workshop 関連ドキュメント】

- High-performance Embedded Workshop ユーザーズマニュアル
- High-performance Embedded Workshop チュートリアル
- High-performance Embedded Workshop リリースノート

【CPU 関連ドキュメント】

- H8S/2378 グループ、H8S/2378R グループ ハードウェアマニュアル
- H8S/2600 シリーズ、H8S/2000 シリーズ ソフトウェアマニュアル

【H8S H8/300 シリーズ C/C++コンパイラ関連ドキュメント】

- H8S H8/300 シリーズ C/C++コンパイラパッケージ ユーザーズマニュアル
- H8SX、H8S、H8/300 ファミリ用 C/C++コンパイラパッケージ 注意事項及びユーザーズマニュアル修正

本製品に関する情報は以下のルネサス・ウェブサイトをご覧下さい:

日本サイト: http://japan.renesas.com/e10a_usb

グロバールサイト: http://www.renesas.com/e10a_usb

付録　IIC_マスタ送信側プログラム

・ハードウェア開発環境

項目	内容
CPU 種類	M16C/62P（デバイス型名：M30626FJPGP）
メモリ容量	ROM：512k バイト、RAM：31k バイト
使用ボード	ルネサス M16C/62P スタータキット（ボード型名：RSKM16C62P）
電源電圧	3.3V
動作周波数	外部クロック ：6MHz、 CPU クロック ：6MHz
動作モード	シングルチップモード
エミュレータ	E8a エミュレータ（製品型名：R0E00008AKCE00）

・ソフトウェア開発環境

項目	内容
開発ツール	High-performance Embedded Workshop Ver. 4.03.00.001
C/C++コンパイラ	Renesas M16C Standard Toolchain 5.43.00
E8a エミュレータ	E8a エミュレータソフトウェア　V1.00　Release　00A

・特殊モード 1（IIC モード）機能仕様

項目	内容
使用シリアルクロック端子	IIC_2 シリアルクロック入出力端子（SCL2）
使用シリアルデータ端子	IIC_2 シリアルデータ入出力端子（SDA2）
転送レート	98.36kbps
通信モード	マスタ送信モード
スレーブアドレス	H’08
送信データサイズ	4k バイト

・マクロ定数

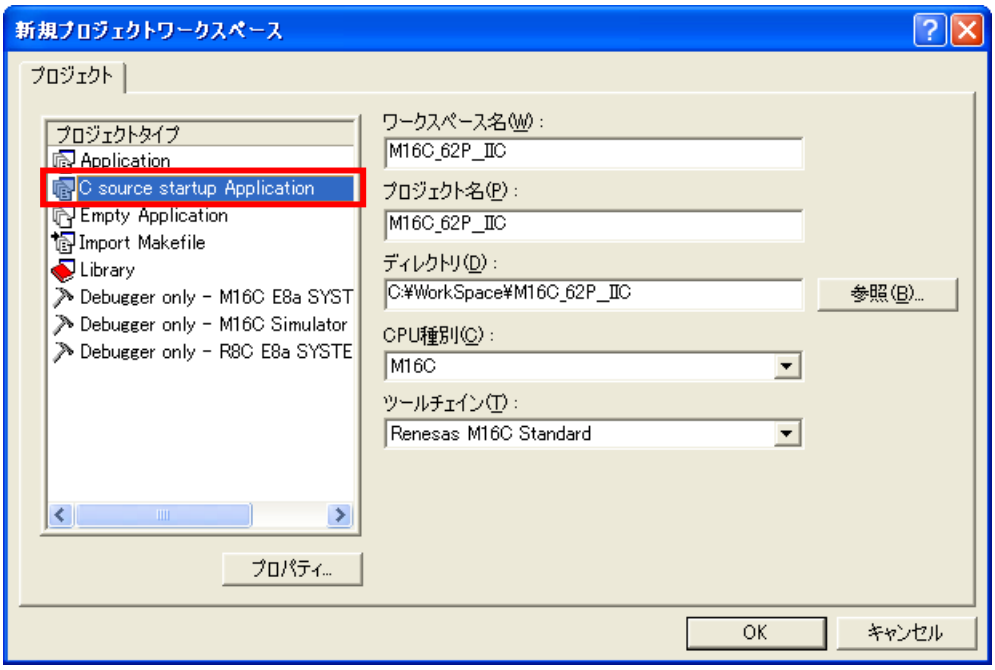
定数名	設定値	内容
MT_DATA_ADD	H’3000	送信データバッファ先頭アドレス
DATA_SIZE	H’1000	1 回あたりの送信データサイズ （1 バイト ＊ H’1000 = 4k バイト）
TRANS_NUM	H’0001	送信回数 総送信データサイズ = DATA_SIZE ＊ TRANS_NUM （最大送信データサイズ ：4k バイト ＊ H’7F = 508k バイト）

・使用される前に

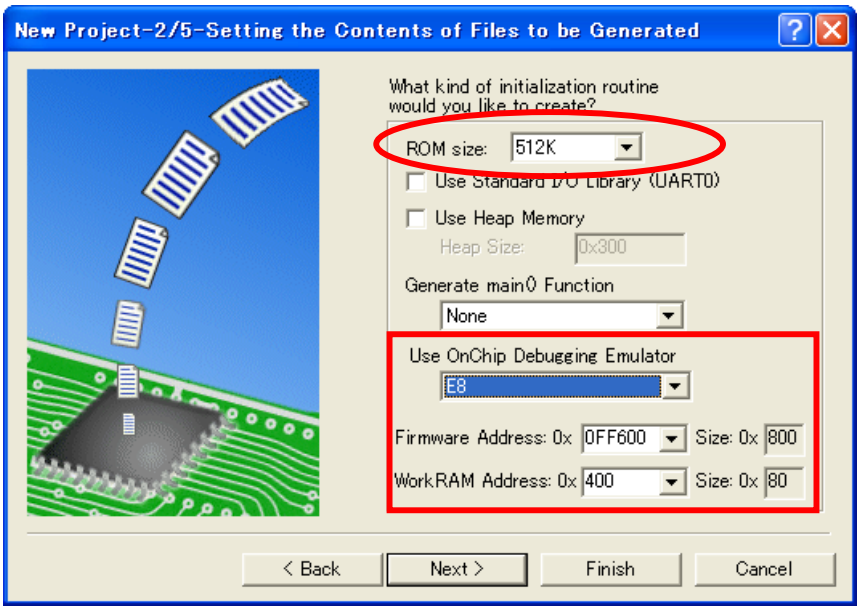
IIC マスタ送信側プログラムを使用される場合は、必ず下記設定変更を行ってください。

(1) プロジェクトワークスペースを作成する際に以下に示す箇所を変更してください。

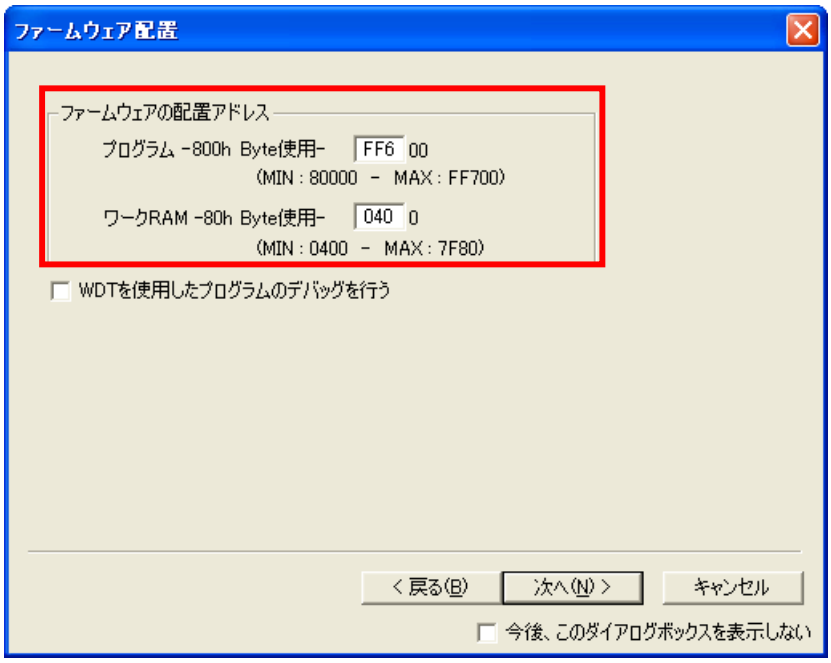
[プロジェクトタイプ]は“C source startup Application”を選択してください。



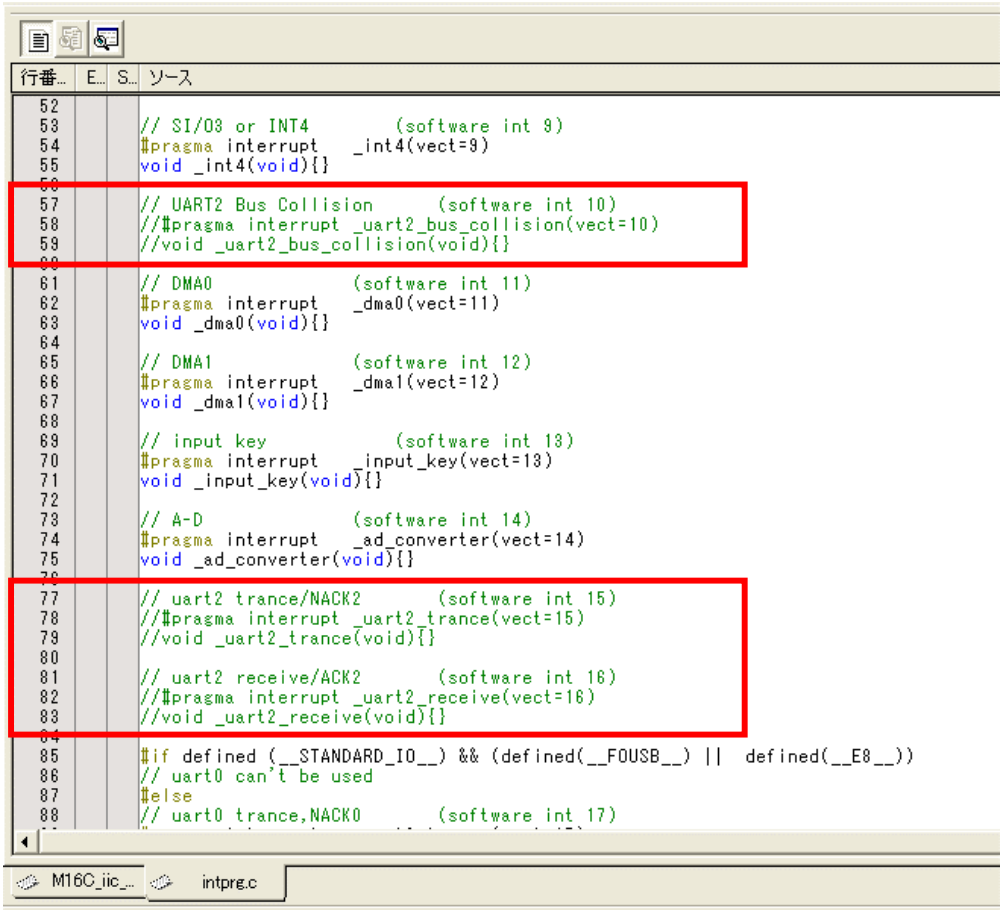
[ROM size]は“512k”を選択し、[Use OnChip Debugging Emulator]は“E8”を選択してください。この設定でエミュレータのファームウェア領域が“H’FF600～H’FFDFF”、ワーク RAM 領域が“H’00400～H’0047F”に確保（セクション設定）されます。



(2) (1)で確保した領域に合わせて[ファームウェアの配置アドレス]を変更してください。今回は[プログラム]を“H’FF600”、[ワーク RAM]を“H’0400”に設定してください。



(3) “intprg.c”ファイル内のソフトウェア割り込み番号 10、15、16 は、IIC マスタ送信側プログラム内で定義していますのでコメントアウトしてください。



・プログラム

```
/************************************************************************/
/*                                                                      */
/*  FILE                :M16C_iic_mt.c                                  */
/*  DATE                :Fri, Oct 26, 2007                              */
/*  DESCRIPTION         :IIC2 Master transfer mode main program file    */
/*  CPU GROUP           :M16C/62P                                       */
/*  BOARD               :RSKM16C62P                                     */
/*                                                                      */
/*  OSCILLATOR CLOCK    :6MHz                                           */
/*  TRANSFER RATE       :98.36kbps                                      */
/*  SLAVE ID            :0x08                                           */
/*  TRANSFER DATA SIZE  :4kbyte                                         */
/*  USED IIC BUS        :UART2                                          */
/*                                                                      */
/************************************************************************/

/************************************************************************/
/*  Include file                                                        */
/************************************************************************/
#include "sfr62P.h"

/************************************************************************/
/*  Prototype definition file                                           */
/************************************************************************/
unsigned char iici_ini(unsigned char);
unsigned char iici_master_start(unsigned char,unsigned char,unsigned char*,unsigned short);
void iici_master_end(unsigned char);
void iici_slave_end(unsigned char,unsigned short);
unsigned char* iici_id_check(unsigned char,unsigned char);
void _sis_int(void);
void _sir_int(void);
void _sit_int(void);
static void sta_int(void);
static void stp_int(void);

/************************************************************************/
/*  Vector definition                                                   */
/************************************************************************/
#pragma interrupt _sis_int(vect=10)
#pragma interrupt _sit_int(vect=15)
#pragma interrupt _sir_int(vect=16)

/************************************************************************/
/*  SFR                                                                 */
/************************************************************************/
#define UiSMR4   u2smr4
#define UiSMR3   u2smr3
#define UiSMR2   u2smr2
#define UiSMR    u2smr
#define UiMR     u2mr
#define UiBRG    u2brg
#define UiTB     u2tb
#define UiC0     u2c0
#define UiC1     u2c1
#define UiRB     u2rb
#define SiSIC    bcnic
#define SiTIC    s2tic
```

```
#define SiRIC    s2ric
#define stareq   stareq_u2smr4                  //Start condition generate bit
#define rstareq  rstareq_u2smr4                 //Restart condition generate bit
#define stpreq   stpreq_u2smr4                  //Stop condition generate bit
#define stspsel2 stspsel_u2smr4                 //SCL,SDA output select bit
#define ackd     ackd_u2smr4                    //ACK data bit
#define ackc     ackc_u2smr4                    //ACK data output enable bit
#define sclhi    sclhi_u2smr4                   //SCL output stop enable bit
#define swc9     swc9_u2smr4                    //SCL wait output bit3(final pulse)
#define ckdir    ckdir_u2mr                     //Internal/external clock select bit
#define csc      csc_u2smr2                     //Clock synchronous bit
#define swc      swc_u2smr2                     //SCL wait output bit(9th pulse)
#define als2     als_u2smr2                     //SDA output stop bit
#define stc      stac_u2smr2                    //UARTi initialize bit
#define abl      abt_u2rb                       //Arbitration lost detecting flag
#define bbs      bbs_u2smr
#define PRCR     prcr

#define p_sda    p7_0                           //SDA port data bit
#define p_scl    p7_1                           //SCL port data bit
#define pd_sda   pd7_0                          //SDA port direction bit
#define pd_scl   pd7_1                          //SCL port direction bit

#define IIC_BRG  61-1                           //98.36Kbps(main clock = 6MHz)

/************************************************************************/
/*  Parameter definition                                                */
/************************************************************************/
#define MT_DATA_ADD 0x003000                    //Top address of master transfer data
#define DATA_SIZE   0x1000                      //One time master transfer data size = 4kbyte
#define TRANS_NUM   0x0001                      //Number of master transfer(max value 0x007F(512kbyte))

/************************************************************************/
/*  Memories definition                                                 */
/************************************************************************/
typedef union{
  struct{
    unsigned char b0:1;
    unsigned char b1:1;
    unsigned char b2:1;
    unsigned char b3:1;
    unsigned char b4:1;
    unsigned char b5:1;
    unsigned char b6:1;
    unsigned char b7:1;
  }bit;
  unsigned char all;
}byte_dt;

typedef union{
  struct{
    unsigned char b0:1;
    unsigned char b1:1;
    unsigned char b2:1;
    unsigned char b3:1;
    unsigned char b4:1;
    unsigned char b5:1;
    unsigned char b6:1;
    unsigned char b7:1;
```

```
    unsigned char b8:1;
    unsigned char b9:1;
    unsigned char b10:1;
    unsigned char b11:1;
    unsigned char b12:1;
    unsigned char b13:1;
    unsigned char b14:1;
    unsigned char b15:1;
  }bit;
  struct{
    unsigned char byte0:8;
    unsigned char byte1:8;
  }byte;
  unsigned int all;
}word_dt;

static byte_dt iic_md;                          //IICbus mode
#define iic_mode iic_md.all
#define f_rw iic_md.bit.b0                      //[0:write, 1:read]
#define f_ms iic_md.bit.b4                      //[0:slave, 1:master]
#define f_ep iic_md.bit.b5                      //[0:slave, 1:master]
#define f_sr iic_md.bit.b7                      //[0:receive, 1:send]
static byte_dt iic_sl;                          //Master 1st byte
#define iic_slave iic_sl.all
#define iic_rw iic_sl.bit.b0                    //0:write 1:read
static unsigned short iic_length;               //Master length
static unsigned short iic_index;
static unsigned char  *iic_pointer;             //pointer

unsigned char mt_result;                        //Master transfer start success/fail result
unsigned char end_result;                       //Master transfer end success/fail result

unsigned char sw_buf[DATA_SIZE];                //Slave transfer buffer
unsigned char sr_buf[DATA_SIZE];                //Slave receive buffer

/*****************************************************************************/
/*  IIC main function                                                        */
/*****************************************************************************/
void main()
{
  unsigned short i;
  unsigned short j;
  unsigned short data_len;
  unsigned char *iic_ram;                       //Master transfer buffer
  unsigned long wait;

  prcr  = 0x03;                                 //Protect disable
  cm0   = 0x08;
  cm1   = 0x20;                                 //Main clock no divide(main clock = 6MHz)
  pm0   = 0;                                    //Single chip mode(PM00 = 0, PM01 = 0)
  prcr  = 0;                                    //Protect enable

  iic_ram = ((unsigned char*)MT_DATA_ADD);      //Top address of master transfer data
  data_len = DATA_SIZE;                         //Transfer data length

  for(i = 0; i < data_len; i++){
    *(iic_ram+i) = i;
  }
```

```
  iici_ini(1);                                  //Initialization [0:disable, 1:enable]
  asm("fset I");                                //Set I flag

  for(wait = 0; wait < 300000; wait++);         //Wait for H8S/2378R F-ZTAT erase processing

/*************************************************************************/
/*  Master proccessing start                                             */
/*  First  number parameter : Address of appoint slave device(0x00～0x7F)  */
/*                            (Setting value must shift 1bit to right)   */
/*  Second number parameter : 0:Master transfer action  1:Master receive action */
/*  Third  number parameter : Transfer buffer pointer or Receive buffer pointer */
/*  Fourth number parameter : Transfer data length                       */
/*************************************************************************/
  for(i = 0; i < TRANS_NUM; i++){
    if(iici_master_start(0x04, 0, iic_ram, data_len) == 0){     //Slave ID = 0x08
      mt_result = 0;                                            //Master transfer start failed
      while(1){}
    }
    else{
      if(iici_master_start(0x04, 0, iic_ram, data_len) == 0){
        mt_result = 1;                                          //Master transfer start successful
        while(bbs == 1){}
        for(wait = 0; wait < 12000; wait++);                    //Wait for H8S/2378R F-ZTAT programming
      }
      else{
        mt_result = 0;                                          //Master transfer start failed
        while(1){}
      }
    }
  }
  while(1){}
}
/*************************************************************************/
/*  IICbus initialize function                                           */
/*************************************************************************/
unsigned char iici_ini(unsigned char ini)
{
  if(ini == 1){                                 //IICbus mode initialize(START)
    UiMR = 0x0a;                                //IIC mode(ext. clock)
    UiBRG = IIC_BRG;                            //98.36KBPS
    UiC0 = 0x90;                                //MSB first,f1,CTS disable
    UiSMR = 0x01;                               //IICbus mode,Arbitration lost flag Update per bit
    UiSMR2 = 0x11;                                   //transfer/receive interrupt,disalbe Clock sync,UART
initialize enable
    UiSMR3 = 0x62;                              //SDA delay = 4-cycle of BRG count source
    UiSMR4 = 0x30;                              //ACK data output(SDA="H")
    pd_sda = 0;                                 //SDA-port input
    pd_scl = 0;                                 //SCL-port input
    UiC1 = 0x15;                                //Transfer/Receive enable
    iic_mode = 0x00;                            //Slave mode
    iic_index = 0x00;
    SiSIC = 0x00;
    SiTIC = 0x00;
    SiRIC = 0x01;                               //Receive int. enable
  }
  else{                                          //Invalidate IICbus(Stop sequence)
    SiSIC = 0x00;
    SiTIC = 0x00;
    SiRIC = 0x00;
```

```
    UiC1 = 0x10;                                //Transfer/Receive disable
    UiSMR2 = 0x01;                              //UART initialize disable
    pd_sda = 0;                                 //SDA-port input
    pd_scl = 0;                                 //SCL-port input
    UiMR = 0x00;                                //selected output Port
  }
  return(1);
}
/************************************************************************/
/*  IICbus master mode starts function                                  */
/************************************************************************/
unsigned char iici_master_start(unsigned char slave,
                                unsigned char rw,
                                unsigned char *buf,
                                unsigned short len)
{
  if(bbs == 1){
    return(0);
  }
  else{
    asm("pushc FLG");
    asm("fclr I");

    UiSMR = 0x01;                               //All bit clear without bit0
    UiSMR4 = 0x70;                              //SCL and SDA is output"H"
    UiMR = 0x00;
    UiC1 = 0x10;                                //Transfer/Receive disable
    UiSMR3 = 0x60;                              //CKPH = 0
    p_sda = 0;
    UiMR = 0x02;
    UiBRG = 0;                                  //TUD No.TN-16C-130A/JA

    asm("or.b #01h,004Ah");                     //Start con. int. enable (SiSIC |= 0x01)

    UiSMR2 = 0x03;                              //UART init. disable,Clock sync. enable
    UiBRG = IIC_BRG;
    UiSMR4 = 0x71;                              //Start condition generate
    UiSMR4 = 0x09;                              //STSP output enable

    iic_slave = slave << 1;
    iic_length = len;
    iic_pointer = buf;
    if(rw == 0){
      iic_mode = 0x10;                           //Master transfer mode
      iic_rw = 0;
    }
    else{
      iic_mode = 0x11;                          //Master receive mode
      iic_rw = 1;
    }
    asm("popc FLG");
  }
  return(1);
}
/************************************************************************/
/*  IICbus master mode end function                                     */
/************************************************************************/
void iici_master_end(unsigned char status)
{
```

```
  if((status & 0xF0) == 0x10){                  //Transfer mode
    switch(status & 0x0F){                      //Transfer status proccessing
      case 0:  end_result = 1; break;           //Normally finish
      case 1:  end_result = 2; break;           //When the 1st byte , NACK found finish
      case 2:  end_result = 3; break;           //When N byte , NACK found finish
      case 3:  end_result = 4; break;           //BUS Arbitration lost
      case 4:  end_result = 5; break;           //Dishonesty start condition
      case 5:  end_result = 6; break;           //Dishonesty stop condition
      default: end_result = 7; break;
    }
  }
  else if((status & 0xF0) == 0x20){             //Receive mode
    switch(status & 0x0F){                      //Receive status proccessing
      case 0:  end_result = 1; break;           //Normally finish
      case 1:  end_result = 2; break;           //When the 1st byte , NACK found finish
      case 2:  end_result = 3; break;           //When N byte , NACK found finish
      case 3:  end_result = 4; break;           //BUS Arbitration lost
      case 4:  end_result = 5; break;           //Dishonesty start condition
      case 5:  end_result = 6; break;           //Dishonesty stop condition
      default: end_result = 7; break;
    }
  }
}
/*****************************************************************************/
/*  IICbus slave mode end function                                           */
/*****************************************************************************/
void iici_slave_end(unsigned char status, unsigned short iic_index)
{
  if((status & 0xF0) == 0x10){                  //Transfer mode
    switch(status & 0x0F){                      //Transfer status proccessing
      case 0:  end_result = 1; break;           //Normally finish
      case 1:  end_result = 2; break;           //When the 1st byte , NACK found finish
      case 2:  end_result = 3; break;           //When N byte , NACK found finish
      case 3:  end_result = 4; break;           //BUS Arbitration lost
      case 4:  end_result = 5; break;           //Dishonesty start condition
      case 5:  end_result = 6; break;           //Dishonesty stop condition
      default: end_result = 7; break;
    }
  }
  else{                                         //Receive mode
    switch(status & 0x0F){                      //Receive status proccessing
      case 0:  end_result = 1; break;           //Normally finish
      case 1:  end_result = 2; break;           //When the 1st byte , NACK found finish
      case 2:  end_result = 3; break;           //When N byte , NACK found finish
      case 3:  end_result = 4; break;           //BUS Arbitration lost
      case 4:  end_result = 5; break;           //Dishonesty start condition
      case 5:  end_result = 6; break;           //Dishonesty stop condition
      default: end_result = 7; break;
    }
  }
}
/*****************************************************************************/
/*  IICbus slave mode ID_check function                                      */
/*****************************************************************************/
unsigned char* iici_id_check(unsigned char ID, unsigned char RW)
{
  if(ID == 0x06){                               //Own device ID
    if(RW == 1){
      return(&sw_buf[0]);                       //Slave transfer
```

```
    }
    else{
      return(&sr_buf[0]);                       //Slave receive
    }
  }
  else{
    return(0);                                  //Not slave operation
  }
}
/************************************************************************/
/*  IIC start/stop condition interrupt function                         */
/************************************************************************/
void _sis_int(void)
{
  int i;

  if(bbs == 1){
    sta_int();                                  // Start condition interrupt
  }
  else{
    stp_int();                                  // Stop condition interrupt
  }
}
/************************************************************************/
/*  IIC start condition function                                        */
/************************************************************************/
static void sta_int(void)
{
  word_dt temp;

  UiSMR3 = 0x62;                                //CKPH = 1
  UiC1 = 0x10;                                  //Transfer/Receive disable
  UiC1 = 0x15;                                  //Transfer/Receive enable
  UiMR = 0x02;
  UiSMR4 = 0x00;                                //STSPSEL = 0 (SI/O output sel)
  temp.byte.byte0 = iic_slave;                  //Slave ID set
  temp.byte.byte1 = 0x01;                       //NACK data set
  UiTB = temp.all;                              //Start 1st byte transfer
  UiRB = 0x00;                                  //Arbitration lost flag clear
  UiSMR2 = 0x1f;                                //UART init. enable,Clock sync. enable
  SiSIC = 0x01;                                 //Stop int. enable
  SiRIC = 0x01;                                 //Receive int. enable
  iic_index = 0x00;
}
/************************************************************************/
/*  IIC stop condition function                                         */
/************************************************************************/
static void stp_int(void)
{
  UiMR = 0x00;                                 //port set(Purpose:TXFUL,TBFUL flag must be cleared when slave
receive)
  UiSMR4 = 0x30;                                //*ACK data output"H"
  UiMR = 0x0a;                                  //ext. clock sel
  UiSMR2 = 0x11;
  UiC1 = 0x15;                                  //Transfer/Receive enable
  SiRIC = 0x01;                                 //receive int. enable
  SiTIC = 0x00;                                 //transfer int. disable
  SiSIC = 0x00;                                 //start/stop int. disable
  if(f_ms == 0 && f_sr == 0){                   //slave receive
```

```
    --iic_index;
    iici_slave_end(0x10,iic_index);              //slave receive complete
  }
  iic_mode = 0x00;                               //slave mode
  iic_index = 0x00;
}
/************************************************************************/
/*  IIC receive(Falling edge of 9th pulse)interrupt function            */
/************************************************************************/
void _sir_int(void)
{
  word_dt temp;

  temp.all = UiRB;                                          //Read receive buffer
  if(iic_index == 0x00){                                    //= On the 1st time =
    f_sr = f_ms ^ temp.bit.b8;                              //Saved transfer/receive flags
    if(f_ms == 1){                                          //= When master=
      if(abl == 1){                                         //= When Arbitration lost =
        iici_master_end(0x03);                              //Arbitration lost error found
        f_ms = 0;                                           //Changed slave mode
        f_sr = ~f_sr;                                       //Reversed transfer/receive mode
        UiMR = 0x0a;
        goto r1_slave;                                      //Go to slave function
      }
      else{                                                 //= When Arbitration win =
        SiTIC = 0x01;                                       //Transfer int. enable
        UiSMR2 = 0x0b;                                      //Released SCL line
      }
    }
    else{                                                   //= When Slave =
r1_slave:
      temp.bit.b7 = 0;                                      //Masked bit7
      if(f_sr == 1){
        iic_pointer = iici_id_check(temp.byte.byte0,1);     //Slave transfer request
      }
      else{
        iic_pointer = iici_id_check(temp.byte.byte0,0);     //Slave receive request
      }
      if(iic_pointer != 0){                                 //Agreed address
        UiSMR4 = 0xa0;                                      //ACK-data output enable
                                                            //When Falling edge of last pulse,
                                                            //"L"hold enable
        SiSIC = 0x01;                                       //Start/stop int. enable
        if(f_sr == 1){
          temp.byte.byte0 = *iic_pointer;                   //send-data set
          temp.byte.byte1 = 0x01;                           //NACK-send set
          UiTB = temp.all;                                  //Data1 transfer start
        }
        else{
          UiTB = 0x00ff;                                    //dummy data (with ACK)transfer start
        }
        UiSMR2 = 0x11;                                      //ALS clear, CSC clear(for Arbitration lost)
        SiTIC = 0x01;                                       //Transfer int. enable
        swc9 = 1;
        swc = 0;
      }
      else{                                                 //disagreed address
        UiSMR4 = 0x30;                                      //NACK-data output enable
        UiMR = 0x0a;
```

```
        stc = 1;                                        //UART initialize enable
        SiRIC = 0x00;                                   //Receive int. disable
        UiSMR2 = 0x11;                                  //ALS clear, CSC clear(for Arbitration lost)
        SiTIC = 0x00;                                   //Transfer int. disable
        swc9 = 0;
        swc = 0;
        SiSIC = 0x01;
        iic_mode = 0x00;
        iic_index = 0x00;
      }
    }
  }
}
/************************************************************************/
/*  IIC transfer(Falling edge of last pulse)interrupt function          */
/************************************************************************/
void _sit_int(void)
{
  word_dt temp;

  temp.all = UiRB;
  if(iic_index == 0x00){                                //= On the 1st time =
    if(f_ms == 1){                                      //= When master =
      if(temp.bit.b8 == 1){                             //= When NACK found =
        if(f_ep == 0){
          if(f_sr == 1){
            iici_master_end(0x03);                      //When the 1st byte , NACK found finish
          }
          else{
            iici_master_end(0x13);                      //When the 1st byte , NACK found finish
          }
        }
        else{
          iici_master_end(0x23);                        //When the 1st byte , NACK found finish
        }
        als2 = 0;                                       //When Arbitration lost,SDA "HiZ" disable
        UiSMR4 = 0x04;                                  //Stop condition generate
        UiSMR4 = 0x3c;                                  //ST/SP output enable
        UiSMR2 = 0x01;
        SiRIC = 0x01;                                   //receive int. enable

      }
      else{                                             //= When ACK found =
        if(f_sr == 0){                                  //= When receive =
          als2 = 0;                                     //When Arbitration lost,SDA "HiZ" disable
          if(iic_length == 1){                          //= When receive at the last byte=
            UiTB = 0x01ff;                              //Send NACK
          }
          else{                                         //= When continuous receive =
            UiTB = 0x00ff;                              //Send ACK
          }
        }
        else{                                           //= When transfer =
          temp.byte.byte0 = *iic_pointer;
          abl = 0;                                      //Arbitration lost flag clear
          als2 = 1;                                     //When Arbitration lost,SDA "HiZ" disable
          UiSMR = 0x07;                                 //Arbitration lost flag Update per Byte
          temp.byte.byte1 = 0x01;                       //NACK-data set
          UiTB = temp.all;
```

```
        }

        SiRIC = 0x00;                                   //receive int. disable
        asm("or.b #01h,$$", SiTIC);                     //transfer int. enable
        if(f_ep == 0 || f_sr == 0){                    //When EEPROM read mode address set,Pointer no touch
          ++iic_pointer;                                //Pointer moved
        }
        ++iic_index;
      }
    }
    else{                                               //= When slave =
      UiMR = 0x0a;                                      //Ext. clock sel.
      stspsel2 = 0;                                     //SI/O output enable
      ackc = 0;                                         //ACKoutput disable
      swc9 = 0;                                         //SCL"L"Hold3 disable
      swc9 = 1;                                         //SCL"L"Hold3 enable
      ++iic_pointer;                                    //Pointer moved
      ++iic_index;
      if(f_sr == 1){                                    //= When transfer =
        temp.byte.byte0 = *iic_pointer;                 //send-data set
        temp.byte.byte1 = 0x01;                         //NACK-data set
        UiTB = temp.all;                                //Data1 transfer start
        ++iic_pointer;                                  //Pointer moved
        ++iic_index;
      }
      else{                                             //= When receive =
        UiTB = 0x00ff;
      }
      SiRIC = 0x00;

      asm("or.b #01h,004Ah");                           //Start con. int. enable (SiSIC |= 0x01)
    }
  }
  else{                                                 //= On and after the 2nd time =
    if(f_ms == 1){                                      //= When master =
      if(iic_length == iic_index){                      //= When last data =
        if(f_sr == 0){                                  //= When receive =
          --iic_pointer;
          *iic_pointer = temp.byte.byte0;
          ++iic_pointer;
          if(temp.bit.b8 == 0){                        //Ack found(Self-uint send NACK < other-master send
ACK)
            ackd = 1;                                     //Other-master still transmitting,So transmit
finish and stop condition no generate
            ackc = 1;
            UiMR = 0x0a;                                //Ext. clock sel
            if(f_ep == 0){
              iici_master_end(0x10);                    //Master normally finish
            }
            else{
              iici_master_end(0x20);                    //Master normally finish
            }
            iic_mode = 0x00;
            iic_index = 0x00;
          }
          else{                                         //Nack found(other-master send to nothing)
            UiSMR4 = 0x04;
            UiSMR4 = 0x3c;                              //Stop condition generate
            if(f_ep == 0){
```

```
            iici_master_end(0x10);                  //Master normally finish
          }
          else{
            iici_master_end(0x20);                  //Master normally finish
          }
        }
      }
      else{                                         //= When transfer =
        if(abl == 1){                               //= When Arbitration lost =
          iici_master_end(0x02);                    //Arbitration lost error found
          UiSMR4 = 0x30;                            //SDA HiZ
          UiMR = 0x0a;                              //Ext. clock sel
          iic_mode = 0x00;                          //Slave mode set
          iic_index = 0x00;
        }
        else{                                       //= When Arbitration win =
          iici_master_end(0x00);
          UiSMR4 = 0x04;                            //Stop condition generate
          UiSMR4 = 0x3c;
        }
      }
      UiSMR2 = 0x03;                                //bit clear without bit0-1
    }
    else{                                           //= When continue =
      if(f_sr == 0){                                //= When receive =
        --iic_pointer;
        *iic_pointer = temp.byte.byte0;
        ++iic_pointer;
        ++iic_pointer;                              //Pointer moved
        ++iic_index;
        if(iic_length == iic_index){
          UiTB = 0x01ff;                            //Send NACK
        }
        else{
          UiTB = 0x00ff;                            //Send ACK
        }
        SiRIC = 0x00;                               //receive int. disable
        asm("or.b #01h,$$", SiTIC);                 //transfer int. enable
      }
      else{                                         //= When transfer =
        if(abl == 1){                               //= When Arbitration lost =
          iici_master_end(0x02);                    //Arbitration lost error found
          UiSMR4 = 0x30;                            //SDA HiZ
          UiMR = 0x0a;                              //Ext. clock sel
          iic_mode = 0x00;                          //Slave mode set
          iic_index = 0x00;
        }
        else{                                       //= When Arbitration win =
          if(temp.bit.b8 == 1){                     //= When NACK found =
            iici_master_end(0x03);                  //When N byte , NACK found finish
            als2 = 0;
            UiSMR4 = 0x04;                          //Stop condition generate
            UiSMR4 = 0x3c;
            UiSMR2 = 0x01;
            SiRIC = 0x01;                           //receive int. enable
          }
          else{                                     //= When ACK found =
            abl = 0;                                //Arbitration lost flag clear
            temp.byte.byte0 = *iic_pointer;
```

```
            temp.byte.byte1 = 0x01;                 //NACK-data set
            UiTB = temp.all;
            SiRIC = 0x00;                           //receive int. disable
            SiTIC = 0x01;                           //transfer int. enable
            ++iic_pointer;                          //Pointer moved
            ++iic_index;
          }
        }
      }
    }
  }
  else{                                             //= When slave =
    UiMR = 0x0a;
    if(f_sr == 1){                                  //= When transfer =
      if(temp.bit.b8 == 1){                         //= When NACK found =
        ackd = 1;                                   //Output NACK-data
        ackc = 1;                                   //NACK-data output enable
        swc9 = 0;                                   //SCL"L"Hold3 disable

        asm("and.b #0F8h,004Ah");                   //Stop con. int. disable (SiSIC &= 0xF8)

        SiRIC = 0x01;                               //Receive int. enable
        iici_slave_end(0x10,iic_index);             //Slave transfer complete
        iic_index = 0x00;
        iic_mode = 0x00;                            //Slave mode set
      }
      else{                                         //= When ACK =
        swc9 = 0;                                   //SCL"L"HOLD3 disable
        swc9 = 1;                                   //SCL"L"HOLD3 enable
        temp.byte.byte0 = *iic_pointer;             //send-data set
        temp.byte.byte1 = 0x01;                     //NACK-data set
        UiTB = temp.all;                            //Data1 transfer start
        ++iic_pointer;                              //Pointer moved
        ++iic_index;

        asm("or.b #01h, 004Ah");                    //Stop con. int. enable (SiSIC |= 0x01)

        SiRIC = 0x00;                               //Receive int. disable
      }
    }
    else{                                           //= When receive =
      --iic_pointer;
      *iic_pointer = temp.byte.byte0;
      ++iic_pointer;
      UiTB = 0x00ff;                                //Send ACK
      swc9 = 0;                                     //SCL"L"HOLD3 disable
      swc9 = 1;                                     //SCL"L"HOLD3 enable
      ++iic_pointer;                                //Pointer moved
      ++iic_index;

      asm("or.b #01h, 004Ah");                      //Stop con. int. enable (SiSIC |= 0x01)

      SiRIC = 0x00;                                 //Receive int. disable
    }
  }
 }
}
```

ホームページとサポート窓口

ルネサステクノロジホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry
csc@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2008.3.17	－	初版発行

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. 弊社は製品の品質及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。