
M16C/Tiny シリーズ

EW1 モードを使用した ユーザ ROM 領域の書き換えについて

RJJ06B0505-0101
Rev.1.01
2010.04.01

1. 要約

この資料では、M16C/Tiny シリーズの EW1 モードを使用したフラッシュメモリの書き換え方法を紹介します。

添付されたプログラムは標準化されたドライバプログラムです。

2. はじめに

この資料で説明する応用例は、次のマイコン、条件での利用に適用されます。

- ・マイコン : M16C/Tiny シリーズ

M16C/Tiny シリーズと同様の SFR (周辺装置制御レジスタ) を持つ他の M16C ファミリでも本プログラムを使用することができます。ただし、一部の機能を機能追加等に変更している場合がありますのでマニュアルで確認してください。このアプリケーションノートをご使用に際しては十分な評価を行ってください。

<目次>

3.	使用例の説明.....	3
3.1	概要.....	3
3.2	フラッシュメモリとは.....	4
3.2.1	フラッシュメモリの動作説明.....	4
3.3	EW1 モードを用いたフラッシュメモリ書き換え方法.....	5
3.3.1	EW1 モード.....	5
3.3.2	フラッシュメモリのモード遷移.....	6
3.3.3	フラッシュメモリの書き換え中の割り込み.....	7
3.4	EW1 自動書き込み、自動消去手順.....	8
3.4.1	自動書き込み、自動消去.....	8
3.5	読み出し手順.....	17
4.	ドライバプログラム.....	18
4.1	ファイル構成.....	19
4.2	動作説明.....	20
4.2.1	自動書き込み、自動消去動作.....	20
4.2.2	イレーズサスペンド処理.....	21
4.3	ソフトウェアインタフェース.....	23
4.3.1	ドライバプログラムのインタフェース.....	23
4.4	カスタマイズ.....	29
4.4.1	CPUクロック設定カスタマイズ.....	29
4.4.2	ドライバプログラム動作のカスタマイズ.....	29
5.	ドライバプログラム.....	30
5.1	ソースコード.....	31
5.1.1	flashdevconf.h.....	31
5.1.2	flashdevdrv.h.....	32
5.1.3	flashm16c.h.....	37
5.1.4	flashdrvdev_ew1.c.....	40
5.1.5	depend_m16c.c.....	46
5.1.6	ncrt0_EW1.a30.....	49
5.1.7	sect30_EW1.inc.....	55
5.1.8	main_m16c.c.....	67
5.1.9	M16C_EW1.tmk.....	73
6.	参考ドキュメント.....	75
7.	ホームページとサポート窓口.....	76

3. 使用例の説明

3.1 概要

M16C/Tiny シリーズでは、CPU がソフトウェアコマンドを実行することにより、ユーザ ROM 領域、データ領域を書き換えることができる CPU 書き換えモードがあります。このモードには EW0 モード、EW1 モードの 2 つがあります。

- EW0 モード
 - 利点 : 書き込み、消去中でも、CPU が動作する。
 - : 書き込み、消去中の割り込み応答が速い
(割り込みプログラムを RAM に配置した場合)。
 - 欠点 : 書き込み消去中でも CPU はフラッシュメモリのプログラムが実行できない。
(フラッシュメモリから読み出しできない)
 - : RAM の使用量が多い (書き込み消去のプログラムを RAM に配置する必要がある)。

- EW1 モード
 - 利点 : RAM の使用量が少ない。(書き込み、消去のプログラムをフラッシュメモリ上に配置できる。)
 - 欠点 : 書き込み消去中、CPU は停止状態 (ホールド状態) となる。
 - : 書き込み消去中の割り込み応答に一定の時間が必要。
(M16C/26 において、書き込み時 標準 100 μ s、消去時 最大 8ms)

- **書き換え動作仕様 (M16C/26 の場合)**

書き換えモード	EW1
書き換え時動作 CPU 周波数	10MHz 注
書き込み時間 (2byte)	標準 75 μ s
消去時間 2KByte ブロック	標準 0.2s
消去動作→イレーズサスペンド遷移時間	最大 8ms

注：オンチップオシレータ、PLLの制限事項は、「3.4.1 自動書き込み、自動消去」を参照ください。

CPU書き込みモードにてフラッシュメモリ書き換え時の制限で、通常動作の制限ではありません。

本アプリケーションノートでは、EW1 モードの書き込み手順について説明を行い、フラッシュメモリに対して読み書き、消去を行う手段を提供します。

この項目を理解することで M16C/Tiny シリーズの EW1 モードを利用した CPU 書き換えについて理解することができます。

3.2 フラッシュメモリとは

フラッシュメモリは、電氣的に書き込み、消去ができる不揮発性メモリです。

M16C/Tiny シリーズのフラッシュメモリへのアクセスは以下の通りです。

- 書き込みは1バイト単位で可能です。
- 消去はブロック単位で行います。
- 書き込み、消去時はフラッシュメモリに対して読み出しができません。

3.2.1 フラッシュメモリの動作説明

フラッシュメモリの動作を以下に示します。

表 3-1 フラッシュメモリの動作と制限事項

動作名	動作説明	制限事項
読み出し	書き込まれたデータを読み出すこと	書き込み、ブロック消去中は全領域からデータを読み出すことができない
書き込み	ビットの値を“1”から“0”にすること	既に書き込まれている番地に対する追加書き込みの禁止。
消去	ブロック全体のビット値を“1”にすること (オール FF ₁₆ となる)	ブロック単位で行う必要がある

フラッシュメモリのデータ書き換え制限事項を以下の方法で解決します。

表 3-2 フラッシュメモリ制限事項と解決方法

制限事項	解決方法
書き込み、消去中はフラッシュメモリから読み出しができない。(フラッシュメモリ上で動作できない)	書き込み、消去プログラムをフラッシュメモリ以外に配置する。(EW0 モード)
	書き込み、消去中は自動的に CPU が停止状態となり読み出しを行わない。(EW1 モード)
ブロック単位でしか消去できない	データ保持方法を工夫し、消去回数を減らす。(注)

注：本アプリケーションノートでは扱いません。

本アプリケーションノートは、EW1 モードを用いてフラッシュメモリ上にある書き換えプログラムからフラッシュメモリの別ブロック(書き換えプログラムを配置していないブロック)に対して書き換え処理を行います。

3.3 EW1 モードを用いたフラッシュメモリ書き換え方法

3.3.1 EW1 モード

EW1 モードは、フラッシュメモリ上に配置した書き換えプログラムからプログラム（自動書き込み）／ブロックイレーズ（自動消去）コマンドを発行することにより、ユーザ ROM 領域の書き換えを行なうことができるモードです。自動書き込み/自動消去中の CPU はホールド状態（入出力ポートはコマンド実行前の状態を保持）です。

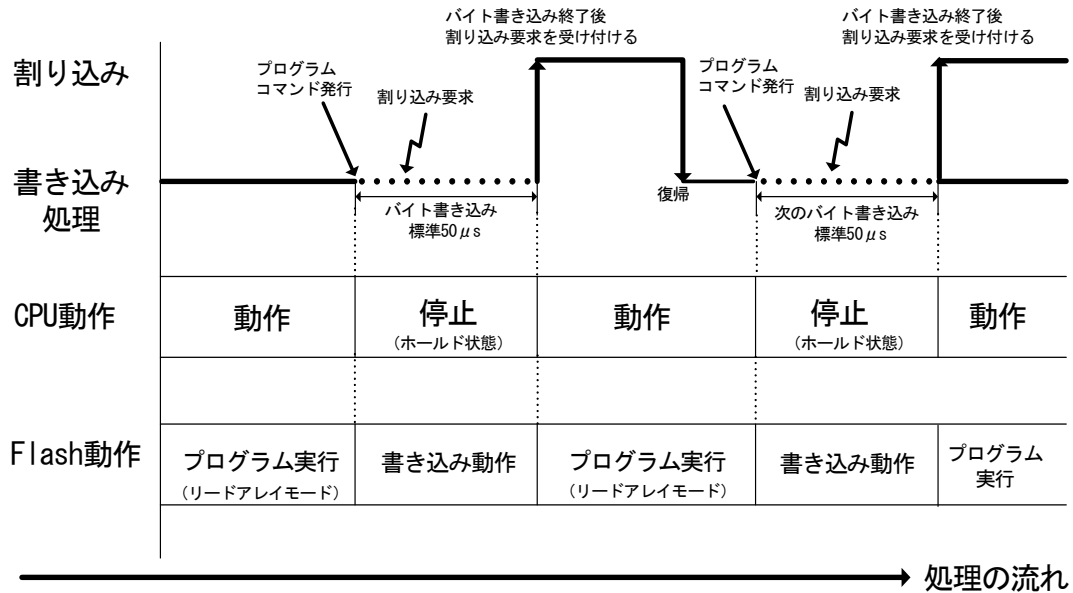


図 3-1 自動書き込み動作概念図

自動消去にはイレーズサスペンド機能があります。イレーズサスペンド機能とは、自動消去中に自動消去処理を中断してフラッシュメモリからデータを読み出す機能です。イレーズサスペンド機能を使用するには、ソフトウェアにてレジスタの設定が必要です。詳細は「4.2.2 イレーズサスペンド処理」をご確認ください。イレーズサスペンドモード中は、フラッシュメモリ上の処理を呼び出すことや、フラッシュメモリからの読み出しができません。

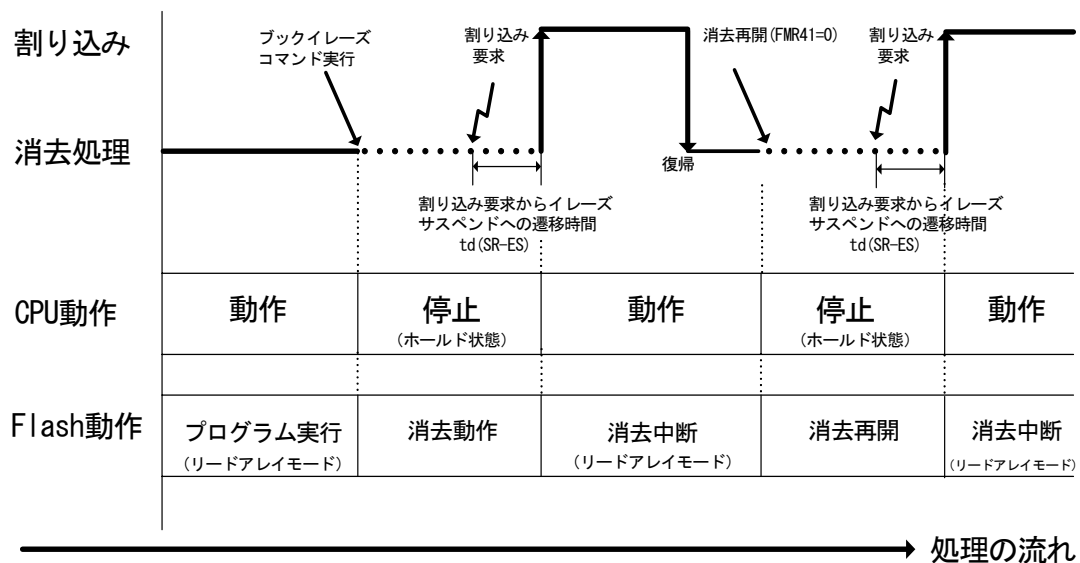


図 3-2 自動消去動作概念図

3.3.2 フラッシュメモリのモード遷移

M16C/Tiny シリーズのフラッシュメモリを制御するには、フラッシュメモリ制御レジスタとソフトウェアコマンドを用います。

ソフトウェアコマンドは、フラッシュメモリに書き込みを行うことで発生します。

以下に EW1 モードにおけるフラッシュメモリのモード遷移図を示します。

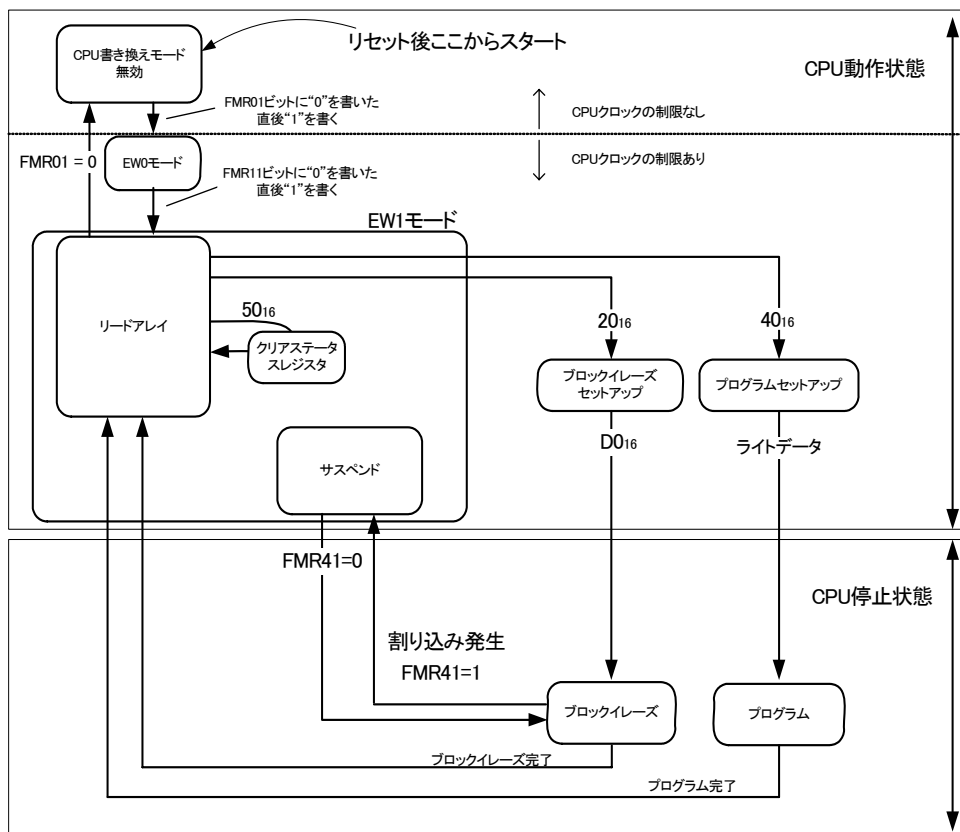


図 3-3 フラッシュメモリのモード遷移図

図 3-3 フラッシュメモリのモード遷移図において、「40₁₆」などの遷移は、ソフトウェアコマンドです。「FMR41 = 0」はレジスタへの設定を示しています。

EW1 モードでは自動消去、自動書き込みが完了するとフラッシュメモリの内容が読み出せます。

3.3.3 フラッシュメモリの書き換え中の割り込み

EW1 モードでイレーズサスペンド機能を使用する場合、使用する割り込みの種類(マスクブル割り込み/ノンマスクブル割り込み)によって消去動作が異なります。

以下に割り込みの種類による動作違いを示します。

表 3-3 自動消去時/自動書き込みの割り込み動作

モード	状態	マスクブル割り込み受付時	ノンマスクブル割り込み受付時	
			NMI 割り込み	ウォッチドッグタイマ
EW1	自動消去中 (イレーズサスペンド機能有効)	自動消去を中断し、割り込み処理を実行します。割り込み処理終了後にイレーズサスペンドリクエストビットを”0”にクリア(イレーズリスタート)することにより、自動消去を再開することができます。	割り込み要求を受け付けると自動消去または自動書き込みは強制停止し、フラッシュメモリをリセットします。一定時間後にフラッシュメモリが再起動した後、割り込み処理を開始します。自動消去中のブロックまたは自動書き込み中のアドレスは強制停止されるために正常値が読み出せなくなる場合がありますので、フラッシュメモリが再起動した後、再度自動消去/書き込みを実行し、正常終了することを確認してください。	ウォッチドッグタイマはコマンド動作中は停止します。
	自動消去中 (イレーズサスペンド機能無効)	自動消去が優先され、割り込み要求が待たされません。自動消去が終了した後、割り込み処理を実行します。		
	自動書き込み	自動書き込みが優先され、割り込み要求が待たされます。自動書き込みが終了した後、割り込み処理を実行します。		

3.4 EW1 自動書き込み、自動消去手順

3.4.1 自動書き込み、自動消去

自動書き込みのフローチャートを以下に示します

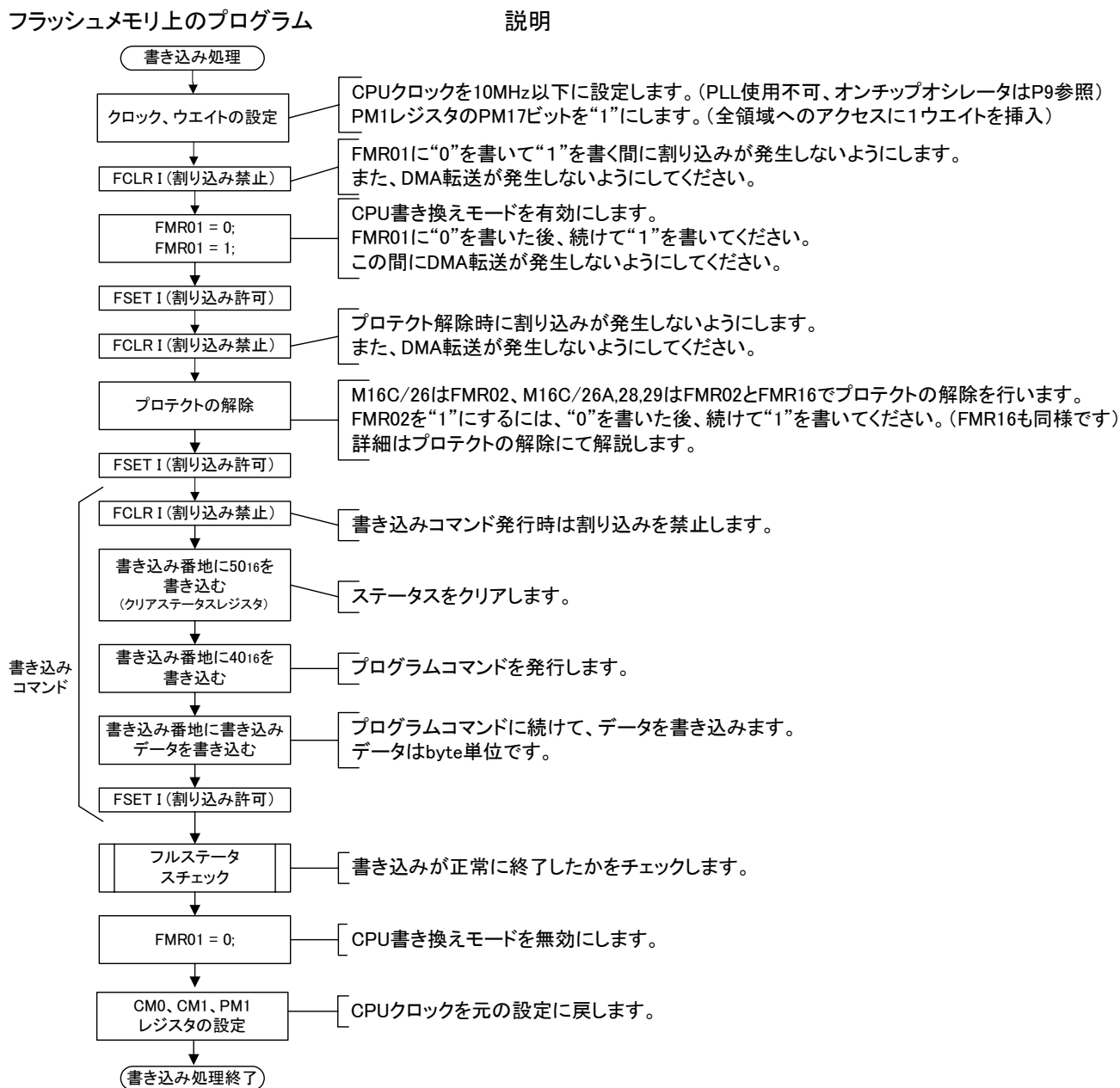


図 3-4 自動書き込みフローチャート

自動消去のフローチャートを以下に示します。

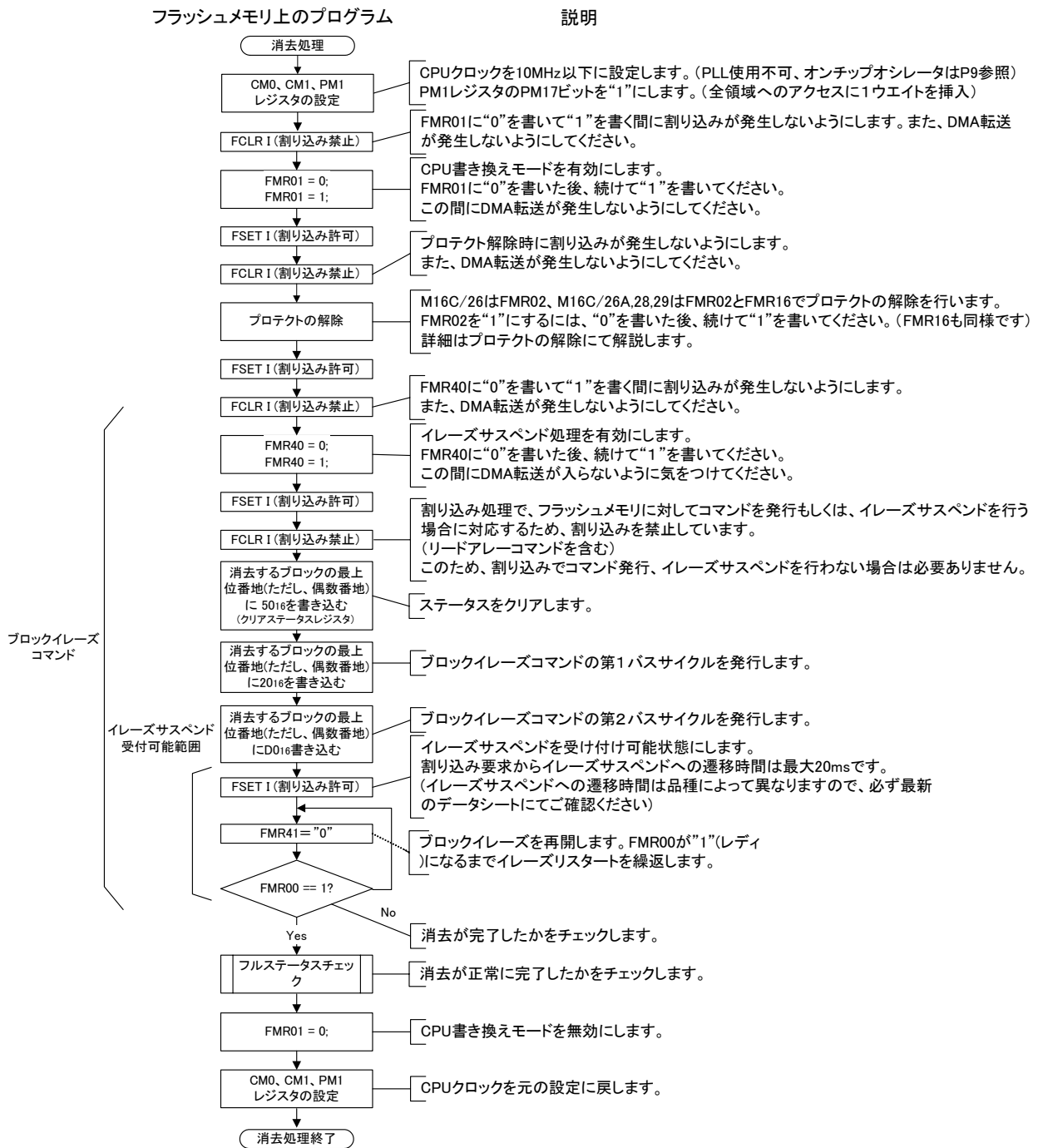


図 3-5 自動消去フローチャート

フローチャートの詳細説明を以下に示します。

● クロック、ウェイトの設定

CPU 書き換えモードに設定(FMR01 に“1”を設定)する時に、以下の制限事項があります。

- ROM、RAM の全領域へのウェイト設定 (PM17 に“1”を設定)
- CPU クロックが以下の条件であること

表 3-4 CPU 書き換えモードのクロック制限事項

動作クロック	制限事項	備考
メインクロック	10MHz 以下	CM0、CM1 にて設定
オンチップオシレータ (M16C/26A, 28, 29)	f1 (ROC)、 f2 (ROC)、 f3 (ROC) でかつ ROCR レジスタ が 4 分周または 8 分周の設定	ROCR にて設定
PLL クロック (M16C/26A, 28, 29)	使用不可	システムクロック制御レジスタ 1 (CM1) のシステムクロック制御ビット (CM11) を使 用して PLL クロックから、メインクロック に変更してください。 PLL 周波数シンセサイザの動作をとめる 必要はありません。

さらに、ブロック A, B を 100 回以上書き換える場合は、CPU 書き換えモード以外でも FMR17 に“1”を設定し、ブロック A, B の読み出しを 1 ウェイトで行う必要があります。

これをまとめると以下のようになります。

表 3-5 CPU 書き換えモードと制限事項

状態		クロック制限	PLL の使用	ウェイト
CPU 書き換えモード 有効	全ブロックの読み出し ソフトウェアコマンド発行	有	不可	必要
CPU 書き換えモード 無効	ブロック A, B の読み出し (書き換え 100 回以内)	無	可	不要
	ブロック A, B の読み出し (書き換え 100 回以上)	無	可	必要
	ブロック A, B 以外の読み出し	無	可	不要

イレーズサスペンド中に CPU クロック制限の解除ができます。イレーズサスペンド中に CPU 書き換えモード選択ビット (FMR01) を “0” (無効) に設定し、CPU クロック制限を解除してください。CPU 書き換えモードを無効にしてもイレーズサスペンド状態は保持されます。

自動消去を再開させるには、CPU クロックを 5MHz 以下にし、CPU 書き換えモードを有効にした後、イレーズリスタートビット (FMR41) を “0” に設定してください。

動作例を下図に示します。

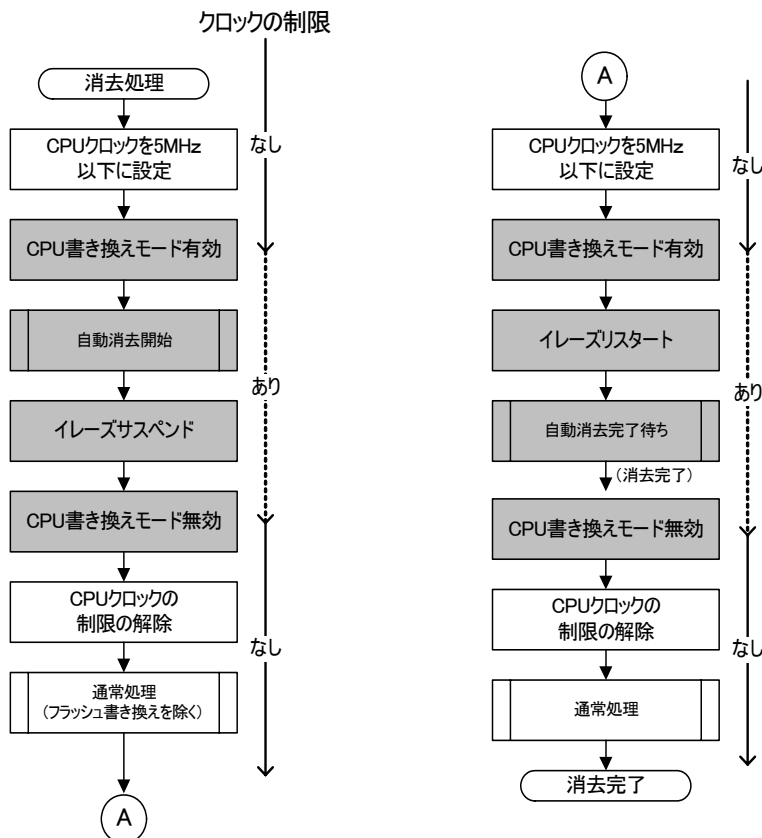


図 3-6 イレーズサスペンド中のクロック制御

● CPU 書き換えモードへの移行

フラッシュメモリ制御レジスタ 0 (FMR0) のビット 1 (FMR01) を設定します。FMR01 ビットを “1” にするときは、“0” を書いた後、続けて “1” を書いてください。“0” を書いた後、“1” を書くまでに割り込みが入らないようにしてください。

● プロテクトの解除

書き換えエリアによっては、プロテクトの解除が必要です。

プロテクトを解除するには、FMR02 ビット及び FMR16 ビット (M16C/26A, 28, 29) の設定が必要です。

機種ごとのプロテクト設定を以下に示します。

表 3-6 M16C/26 のプロテクト設定

レジスタ設定	書き換えエリア		
FMR02	ブロック A, B	ブロック 0, 1	その他のブロック
0	○	×	○
1	○	○	○

○ : 書き換え可 × : 書き換え不可

表 3-7 M16C/26A, 28, 29 のプロテクト設定

レジスタ設定		書き換えエリア		
FMR16	FMR02	ブロック A, B	ブロック 0, 1	その他のブロック
0	0	○	×	×
0	1	○	×	×
1	0	○	×	○
1	1	○	○	○

○ : 書き換え可 × : 書き換え不可

FMR02 及び FMR16 の設定方法は、以下の通りです。

表 3-8 FMR02 及び FMR16 の設定方法

ビット名	FMR01 ビット値が “0”	FMR01 ビット値が “1”
FMR02	常に “0”	設定可 “1” 値に設定するには “0” を書いた後、続けて “1” を書く。 注
FMR16	設定不可 (値を保持)	設定可 “1” 値に設定するには “0” を書いた後、続けて “1” を書く。 注

注：“0”を書いた後、“1”を書くまでに割り込み、DMA 転送が入らないようにしてください。

データ領域 (ブロック A、ブロック B) にはアクセス許可ビット (PM10) があります。データ領域を読み出すとき PM10 ビットを “1” に設定してください。CPU 書き換えモード有効 (FMR01= “1”) のとき、PM10 ビットは自動的に “1” になります。このためブロック A、B にはプロテクトの機能はありません。

PM10 ビットの詳細は「3.5 読み出し手順」をご覧ください。

表 3-8 FMR02 及び FMR16 の設定方法 で示したように、FMR02 ビットは CPU 書き換えモード有効 (FMR01 ビットが “1”) 時のみ値を保持しますが、FMR16 ビットでは、CPU 書き換えモードの有効無効にかかわらず値を保持します。

このため、ドライバプログラムでは FMR02 と FMR16 を以下の処理で設定しています。

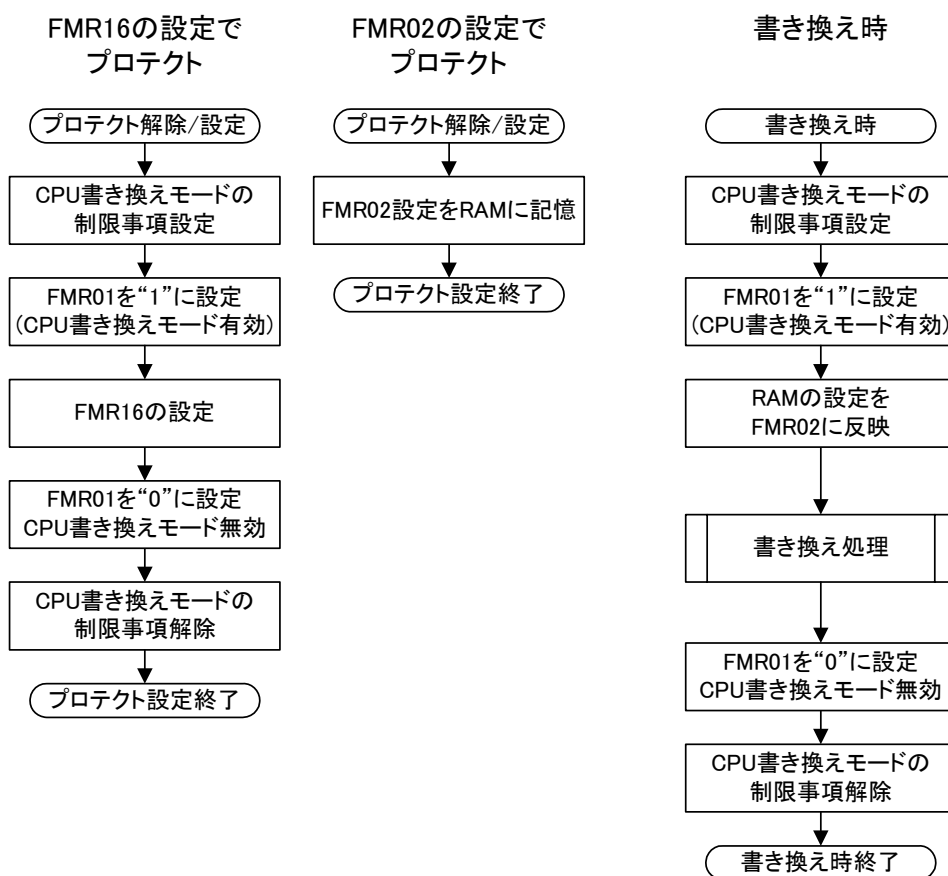


図 3-7 ドライバプログラムでのプロテクト方法

ドライバプログラムでは「プロテクトを設定する」インタフェース関数と、「書き換えを行う」インタフェース関数を分離しています。

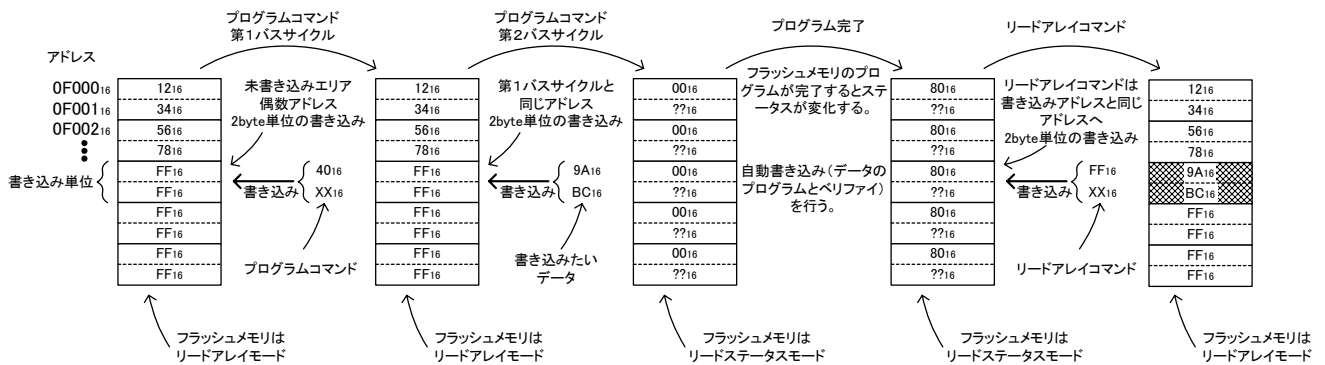
FMR16 ビットは CPU 書き換えモード無効 (FMR01 ビットが “0”) にしても値は保持されます。一方 FMR02 ビットは CPU 書き換えモード無効 (FMR01 ビットが “0”) にした場合、値は保持されず、“0” に変更されます。

FMR16 ビットは、「プロテクトを設定する」インタフェース関数中で設定を行います。FMR02 ビットは書き換え時に最終設定内容を再設定します。

「書き換えを行う」インタフェース関数をコールする前に「プロテクトを設定する」インタフェース関数をコールしてプロテクトの解除を行います。再度、「プロテクトを設定する」インタフェース関数にて設定しない限り、一度設定したプロテクト情報を反映します。

● プログラムコマンド

1ワード(2バイト)単位でフラッシュメモリにデータを書くコマンドです。
プログラムコマンドが発行されると自動書き込み(書き込みとベリファイ)を行います。
自動書き込みは下図のように行います。

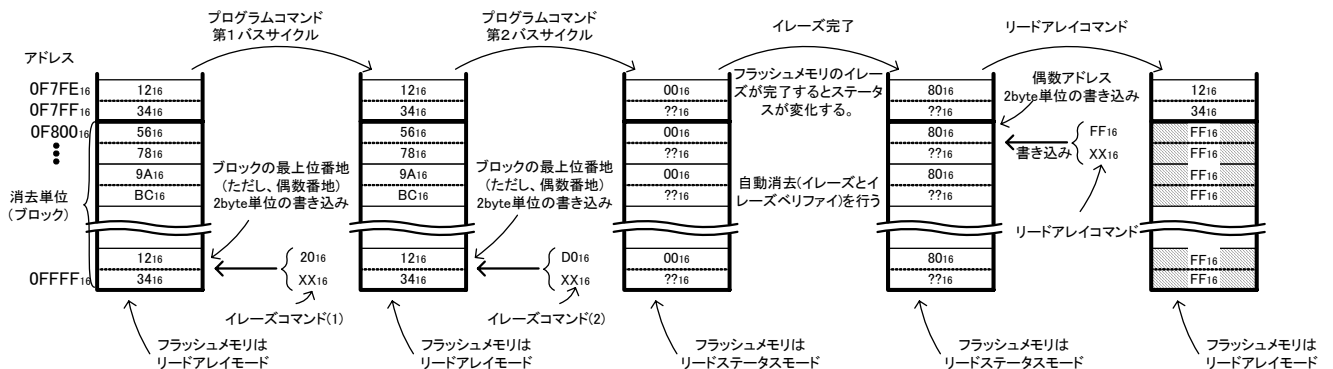


コマンド、書き込みは 16 ビット単位で、ユーザ ROM 領域内の偶数番地に行ってください。

プログラム開始からのフラッシュメモリのモード変化は図 3-3 フラッシュメモリのモード遷移図をご覧ください。

● イレーズコマンド

ブロック単位でフラッシュメモリのデータを消去するコマンドです。イレーズコマンドが発行されると自動消去(消去とベリファイ)を行います。自動消去は下図のように行います。



コマンド、書き込みは 16 ビット単位で、ユーザ ROM 領域内の偶数番地に行ってください。

自動消去開始からのフラッシュメモリのモード変化は図 3-3 フラッシュメモリのモード遷移図をご覧ください。

● フルステータスチェック

自動書き込み、自動消去の実行結果を確認するには、FMR06 ビット及び FMR07 ビットを参照します。以下にステータスレジスタの状態とエラー対応表を示します。

表 3-9 エラーと FMR0 レジスタの状態

FMR00 レジスタ (ステータスレジスタ) の状態		エラー	エラー発生条件
FMR07	FMR06		
1	1	コマンドシーケンスエラー	<ul style="list-style-type: none"> ・コマンドを正しく書かなかったとき ・ブロックイレーズコマンドの第 2 バスサイクルのデータに書いてもよい値(“xxD0₁₆”または“xxFF₁₆”)以外のデータを書いたとき(注 1) ・プロテクトされたブロックにブロックイレーズコマンドを実行したとき ・プロテクトされたブロックにプログラムコマンドを実行したとき
1	0	イレーズエラー	<ul style="list-style-type: none"> ・プロテクトされていないブロックにブロックイレーズコマンドを実行し、正しく自動消去されなかつたとき
0	1	プログラムエラー	<ul style="list-style-type: none"> ・プロテクトされていないブロックにプログラムコマンドを実行し、正しく自動書き込みされなかつたとき
0	0	エラー無し	<ul style="list-style-type: none"> ・成功 ・PM10 を禁止に設定した状態でブロック A、ブロック B に書き込み/消去のソフトウェアコマンドを実行したとき

注 1. これらのコマンドの第 2 バスサイクルで “xxFF₁₆” を書くと、リードアレイモードになり、同時に、第 1 バスサイクルで書いたコマンドコードは無効になります。

● リードアレイコマンド

フラッシュメモリをリードアレイモードにするためのコマンドです。

リードアレイモードでは、フラッシュメモリに記録された内容を読み出すことができます。

第 1 バスサイクルで “xxFF₁₆” を書くと、リードアレイモードになります。次のバスサイクル以降で読む番地を入力すると、指定した番地の内容が 16 ビット単位で読めます。

リードアレイモードは、他のコマンドが書かれるまで保持されるので、複数の番地の内容を続けて読めます。

● CPU 書き換えモード無効

フラッシュメモリ制御レジスタ 0 (FMR0) のビット 1 (FMR01) を “0” に設定します。

● イレーズサスペンド

フラッシュメモリ制御レジスタ (FMR4 レジスタ) の FMR40 ビットを “1” に設定することでイレーズサスペンド機能を使用することができます。

イレーズサスペンド機能を使用することで、イレーズ中に割り込み要求を受け付けることができます。(但し、イレーズコマンド発行後に割り込み要求が受け付けられるまでに最大 20ms (M16C/26 の場合) かかります。その他の品種の時間に関してはデータシートを参照してください。)

フラッシュメモリ制御レジスタ (FMR4 レジスタ) の FMR41 ビットを “0” に設定することで消去を再開します。

● M16C/Tiny において各機種間での違いについて

M16C/26, 26A, 28, 29 ではフラッシュメモリ関連において、以下の内容が異なります。

表 3-10 フラッシュメモリ関連において M16C/26, 26A, 28, 29 の違い

項目	M16C/26	M16C/26A	M16C/28	M16C/29
プロテクト方式	FMR02 による プロテクト	FMR16, FMR02 による プロテクト	←	←
FMR16 の有無	なし	あり	←	←
PLL の有無と PLL での書き換え禁止	PLL 無し	PLL を使用しての書 き換え禁止	←	←

3.5 読み出し手順

フラッシュメモリは、リードアレイモード時にデータを読み出すことができます。

第1バスサイクルで“xxFF 16”を書くと、リードアレイモードになります。(プログラムがフラッシュメモリ上で動作しているとき、すでにリードアレイモードになっています。本アプリケーションノート記載の書き換え手順を行っている場合は、書き込み処理完了時にリードアレイモードとなります。)

さらに、データ領域 (ブロック A、ブロック B) からデータを読み出すには、プロセッサモードレジスタ 1 (PM1) のデータ領域アクセス許可ビット (PM10) を“1”に設定してください。

PM10 ビットについて

- PRCR レジスタの PRC1 ビットを“1” (書き込み許可) にした後で書き換えてください。
- CPU 書き換えモード有効 (FMR01=“1”) のとき、PM10 ビットは自動的に“1”になります。

表 3-11 PM10 ビットとブロック A、B の状態表

PM10	ブロック A、B 状態
0	読み出し不可 (書き換え時は PM10 が必ず“1”となります)
1	読み出し可 書き換え可

ドライバプログラムにおいては、PM10 はデバイスの初期設定で“1”に設定しています。

4. ドライバプログラム

ドライバプログラムについて説明します。ドライバプログラムはフラッシュメモリのデバイスドライバとして記述されています。ドライバのインタフェースとして以下のものを定義します。

このプログラムではフラッシュメモリへの自動書き込み/自動消去時にエラーが発生した場合、エラーコードを返します。エラーが発生した場合、マニュアルに記載されている処理を行ってください。

表 4-1 関数表

関数名	機能	備考
StartEraseFlash()	フラッシュメモリの消去を開始する。	割り込み内使用不可。 内部で割り込みの制御を“1”フラグを使用しています。割り込みを禁止して本関数を使用したい場合は IPL にて禁止してください。
RestartEraseFlash()	中断している消去を再開する。	割り込み内使用不可。 内部で割り込みの制御を“1”フラグを使用しています。割り込みを禁止して本関数を使用したい場合は IPL にて禁止してください。
WriteFlash()	フラッシュメモリへの書き込みを行う。	割り込み内使用不可。 内部で割り込みの制御を“1”フラグを使用しています。割り込みを禁止して本関数を使用したい場合は IPL にて禁止してください。
ReadFlash()	フラッシュメモリの読み出しを行う。	
UnlockBlockFlash()	フラッシュメモリのプロテクトの解除を行う。	
LockBlockFlash()	プロテクトの設定を行う。	
SuspendErase()	フラッシュメモリの消去中断要求のイベントを発行する。	StartEraseFlash()の関数コール後に、本関数がコールされると、StartEraseFlash()は消去を中断し、F_SUSPEND で返る。以降は RestartEraseFlash()で消去を再開する。(このときに SuspendErase()で再度消去が中断される)
ResumErase()	フラッシュメモリの消去中断要求のイベントを取り消す。	SuspendErase()がコールされた後に本関数をコールすると、SuspendErase()で要求した消去中断要求が取り消される。

4.1 ファイル構成

ドライバプログラムは以下のファイル構成となっています。

表 4-2 ファイル構成表

ファイル名	解説
flashdevdrv.h	ドライバを使用する際にインクルードするヘッダーファイルです。
flashdevconf.h	ドライバの設定を行うファイルです。
Flashm16c.h	M16C/Tiny の機種依存部分のフラッシュメモリドライバ内部インクルードファイルです。
flashdrvdev_ew1.c	EW1 モードのフラッシュメモリドライバのファイルです。
depend_m16c.c	フラッシュメモリドライバの M16C/Tiny 機種依存部分のファイルです。
ncrt0_EW1.a30	C 言語初期設定ファイルです。 スタートアップ時の RAM 転送処理が標準ファイル (ncrt0.a30) のものから追加になっています。
sect30_EW1.inc	C 言語セクションファイルです。 新たに、RAM 上で動作するプログラムセクションが追加になっています。
sfr_r26.h sfr_r26a.h sfr_r28.h sfr_r29.h	M16C/26, 26A, 28, 29 用インクルードファイルです。 最新のファイルを入手してご使用ください。
M16C_EW1.tmk	Makefile です。(make - f M16C_EW1.tmk としてコンパイルしてください)

4.2 動作説明

ドライバプログラムのドライバ動作について解説します。
フラッシュメモリの読み込み、書き込み、消去を行うには、必ずドライバを使用します。
以下に消去時のドライバのシーケンスフローを示します。

4.2.1 自動書き込み、自動消去動作

以下に自動書き込み、自動消去 API を使用した動作を示します。

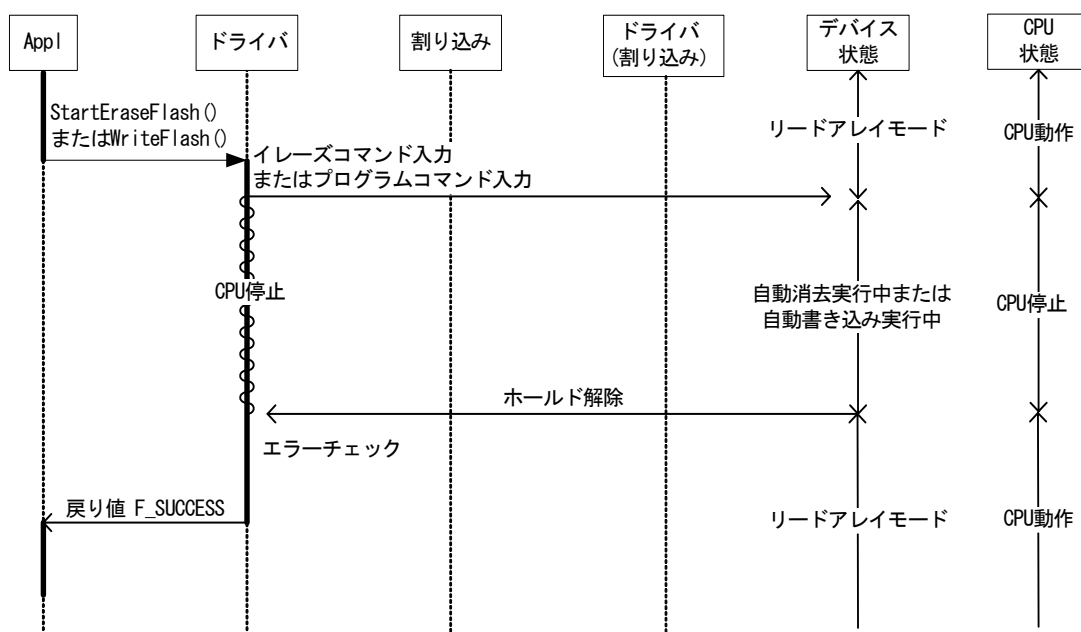


図 4-1 自動書き込み、自動消去 API 動作説明 1

自動書き込みおよび自動消去 API は動作が完了すると戻り値 F_SUCCESS を返します。

4.2.2 イレーズサスペンド処理

SuspendErase()関数は、StartEraseFlash()もしくは、RestartEraseFlash()で行っているイレーズ処理を中断します。この動作シーケンスを示します。

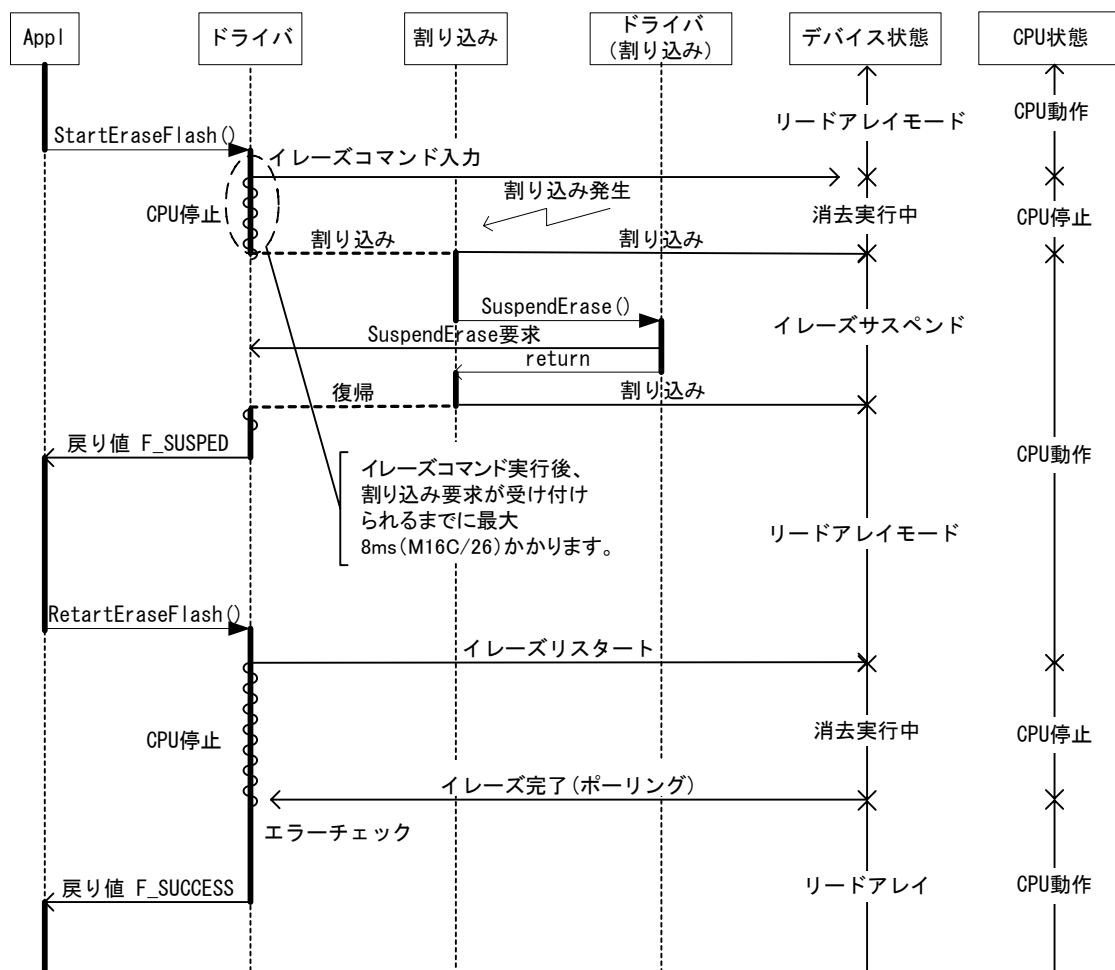


図 4-2 イレーズサスペンド処理シーケンスフロー1

消去インタフェースは同期型の関数ですが、SuspendErase()関数を使用することで、消去インタフェースから復帰することができます。

SuspendErase() 処理からの要求は、イレーズサスペンド中のコールも保持されます。それを回避するには ResumeErase() を使用します。シーケンスを以下に示します。

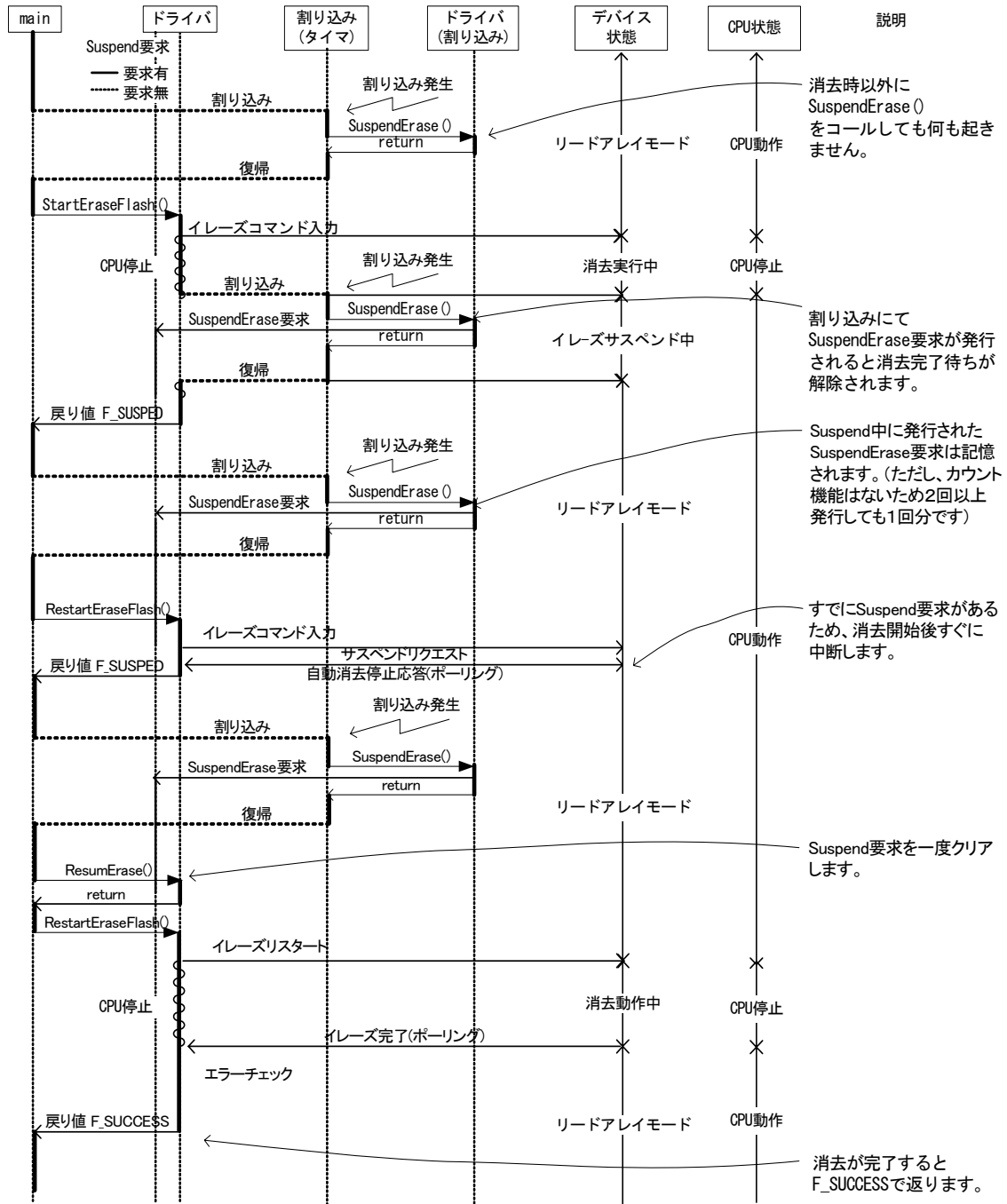


図 4-3 イレーズサスペンド処理シーケンスフロー2

図 4-3 のように、タイマにて SuspendErase() 処理を入れることで、StartFlashErase() や RestartFlashErase() が CPU を占有するのを防止することができます。

4.3 ソフトウェアインタフェース

4.3.1 ドライバプログラムのインタフェース

ドライバプログラムのインタフェースを以下に示します。

表 4-3 StartEraseFlash()

概要	デバイスの消去を行う。		
宣言	FlashResult StartEraseFlash(F_ADR flashAddress);		
インクルード	flash_ew1drv.h		
引数	意味	備考	
F_ADR flashAddress	消去を行うフラッシュメモリの最上位番地(ただし、偶数番地)		
戻り値	意味	値	
FlashResult	エラーコード	F_SUCCESS 成功 他 エラーコード 表 4-11 参照	
機能説明			
フラッシュメモリの消去(ALL 0xFF)を行う。 消去時の割り込み処理で SuspendErase() をコールし割り込み処理が完了すると、本関数がエラーコード F_SUSPEND を伴って途中終了される。このときは、次で示す、RestartEraseFlash() をコールして消去を再開し、完了させること。			
エラーコード			
F_SUCCESS 成功 他 エラーコード 表 4-11 参照			
備考			
エラー発生時(フラッシュメモリデバイスからのエラー応答時)のリトライ動作は行わない。 エラー発生時にはマニュアルに記載してある消去リトライを行うこと。 割り込み内使用不可 内部で割り込みの制御を“1”フラグを使用しています。割り込みを禁止して本関数を使用したい場合は IPL にて禁止してください。			

表 4-4 RestartEraseFlash()

概要	デバイスの消去を再開する。	
宣言	FlashResult RestartEraseFlash(void);	
インクルード	flash_ew1drv.h	
引数	意味	備考
なし		
戻り値	意味	値
FlashResult	エラーコード	F_SUCCESS 成功 他 エラーコード 表 4-11 参照
機能説明		
<p>フラッシュメモリの消去 (ALL 0xFF) を再開する。 StartEraseFlash()/RestartEraseFlash() でエラーコード F_SUSPEND を伴った場合、(イレーズサスペンドとなった場合) に本関数を呼び出して、消去を再開する。再開後消去完了にて F_SUCCESS となる。StartEraseFlash() と同様に SuspendErase() にて消去の中断が可能である。</p>		
エラーコード		
<p>F_SUCCESS 成功 他 エラーコード 表 4-11 参照</p>		
備考		
<p>エラー発生時(フラッシュメモリデバイスからのエラー応答時)のリトライ動作は行わない。 エラー発生時にはマニュアルに記載してある消去リトライを行うこと。 割り込み内使用不可 内部で割り込みの制御を“1”フラグを使用して行っています。割り込みを禁止して本関数を使用したい場合は IPL にて禁止してください。</p>		

表 4-5 WriteFlash()

概要	フラッシュメモリに対してデータを書き込む	
宣言	FlashResult WriteFlash(F_ADR flashAddress, const void * buffer, unsigned short size);	
インクルード	flash_ew1drv.h	
引数	意味	備考
F_ADR flashAddress,	データを書き込む先頭アドレス	2byte 単位の書き込みのため、偶数アドレスを指定すること。
const void * buffer	書き込むデータの先頭アドレス	
unsigned short size	書き込むデータサイズ	2byte 単位で指定すること
戻り値	意味	値
FlashResult	エラーコード	F_SUCCESS 成功 他 エラーコード 表 4-11 参照
機能説明		
フラッシュメモリに対してデータを書き込む。フラッシュメモリの書き込み単位は byte である。		
エラーコード		
F_SUCCESS 成功 他 エラーコード 表 4-11 参照		
備考		
割り込み内使用不可 内部で割り込みの制御を“1”フラグを使用しています。割り込みを禁止して本関数を使用したい場合は IPL にて禁止してください。		

表 4-6 ReadFlash()

概要	フラッシュメモリよりデータの読み込み	
宣言	FlashResult ReadFlash(F_ADR flashAddress, void * buffer, unsigned short size);	
インクルード	flash_ew1drv. h	
引数	意味	備考
F_ADR flashAddress	読み出しデータの先頭アドレス	
void * buffer	読込先アドレス	
unsigned short size	書き込むデータサイズ	
戻り値	意味	値
FlashResult(EW1 モードでは戻り値はない)	エラーコード	F_SUCCESS 成功 他 エラーコード 表 4-11 参照
機能説明		
フラッシュメモリよりデータを読み出す。		
エラーコード		
F_SUCCESS 成功 他 エラーコード 表 4-11 参照		
備考		

表 4-7 UnlockBlockFlash()

概要	フラッシュメモリのライトプロテクトの解除	
宣言	void UnlockBlockFlash(enum FlashBlock blockNumber);	
インクルード	flash_ew1drv. h	
引数	意味	備考
enum FlashBlock blockNumber	プロテクトを解除するレベル	F_ALLBLOCK で全領域解除
戻り値	意味	値
なし		
機能説明		
フラッシュメモリのライトプロテクトを解除する関数である。 F_BLOCK_0 :ブロック 0 の書き込みが可能となる。F_BLOCK_1 :ブロック 1 の書き込みが可能となる。 F_ALLBLOCK : ブロック 0, 1 の書き込みが可能となる。		
エラーコード		
なし		
備考		

表 4-8 LockBlockFlash()

概要	フラッシュメモリのライトプロテクトの設定	
宣言	void LockBlockFlash(enum FlashBlock blockNumber);	
インクルード	flash_ew1drv. h	
引数	意味	備考
enum FlashBlock blockNumber	プロテクトを設定するレベル	F_ALLBLOCK で全領域設定
戻り値	意味	値
なし		
機能説明		
フラッシュメモリのライトプロテクトを設定する関数である。 F_BLOCK_0 : ブロック 0 の書込みが不可となる。F_BLOCK_1 : ブロック 1 の書込みが不可となる。 F_ALLBLOCK : ブロック 0, 1 の書き込みが不可となる。		
エラーコード		
なし		
備考		

表 4-9 SuspendErase()

概要	消去完了待ちの中断要求を発行する	
宣言	void SuspendErase(void);	
インクルード	flash_ew1drv. h	
引数	意味	備考
なし		
戻り値	意味	値
なし		
機能説明		
StartEraseFlash()/RestartEraseFlash() の消去完了待ちに対して、処理を中断する要求を発行する関数である。 フラッシュメモリの状態における処理内容は以下のようになる。 書き込み中 : なにもしない 通常時 : なにもしない イレーズサスペンド中 : イレーズリスタート中断要求を発行する。		
エラーコード		
なし		
備考		
StartEraseFlash() の関数コール後に、本関数がコールされると、StartEraseFlash() は消去を中断し、F_SUSPEND で返る。以降は RestartEraseFlash() で消去を再開する。(このときに SuspendErase() で再度消去が中断される)		

表 4-10 ResumErase()

概要	消去完了待ちの中断要求を取り消す。	
宣言	void ResumErase(void);	
インクルード	flash_ew1drv.h	
引数	意味	備考
なし		
戻り値	意味	値
なし		
機能説明		
SuspendErase() で発行した消去要求を解除する 要求がない場合は何も行わない		
エラーコード		
備考		
SuspendErase() がコールされた後に本関数をコールすると、SuspendErase() で要求した消去中断要求が取り消される。ただし、一度 SuspendErase() を受付後に本関数をコールしても受付取り消しとはならない。		

表 4-11 エラーコード(enum FlashResult)

名称	値	意味
F_SUCCESS	0	成功
F_WRITE_ERROR	1	書き込みエラー。フラッシュメモリより書き込みエラーが通知された。
F_ERASE_ERROR	2	消去エラー
F_CMD_SEQUENCE_ERROR	3	シーケンスエラー。フラッシュメモリよりシーケンスエラーが通知された。
F_WRITE_ADDRESS_ERROR	4	書き込みアドレスアライメントエラー(2Byte アライメント)
F_WRITE_SIZE_ERROR	5	書き込みサイズ未対応エラー(2Byte 単位)
F_DEVICE_BUSY	6	デバイスビジー(書き込み/消去中の読み出し)
F_SUSPEND	7	Write 途中で suspend 要求により中断
F_RESUM_ERROR	8	ResumFlash() で Suspend していないときにコール
F_RESET_OCCURRED	9	フラッシュメモリがリセットされた場合

4.4 カスタマイズ

ドライバプログラムには、システムごとに設定を必要とする部分があります。
以下に、各システムごとに必要なカスタマイズについて記載します。

4.4.1 CPU クロック設定カスタマイズ

CPU 書き換えモードでは、CPU のクロック設定に制限事項があります。
この条件を満たすために以下の関数内の処理をカスタマイズしてください。
ドライバプログラムでは M16C/28 $X_{in}=20\text{MHz}$ にて作成されています。

表 4-12 カスタマイズが必要な関数

関数名	機能
void SlowMCU (ProcessorMode * save);	CPU 動作を制限事項に合うように修正し、修正前の設定を save に保存する
void RestoreMCU (ProcessorMode * save);	save データを CPU クロックの設定として戻す

クロックの制限事項については、3.4.1の内容を参照してください。

4.4.2 ドライバプログラム動作のカスタマイズ

ドライバプログラムのカスタマイズは flashdevconf.h にて行います。

表 4-13 ドライバプログラムオプション定義表

設定内容	デファイン名	デフォルト	設定される内容、説明
CPU シリーズの設定	M16C_SERIES	M16C_SERIES	M16C の機種依存部分をソースに追加します。
CPU グループの設定	M16C_26 M16C_26A M16C_28 M16C_29	M16C_28	CPU のグループを設定します。 機種毎のヘッダをインクルードします。 ロックビットの設定の機種依存部分を追加します。
書き込みモードの設定	FLASH_MODE_EW1	FLASH_MODE_EW1	使用する書き込みモードを選択します。
ウォッチドッグタイマのクリア	ENABLE_WATCHDOG_RESET	未指定	自動消去、自動書き込みの完了待ちのときに、ウォッチドッグタイマをクリアするかを指定します。 通常は、この設定を使用せず、ユーザのメイン処理で行ってください。
アドレスエラーチェックの指定	ENABLE_FLASH_ERR_CHECK	ENABLE_FLASH_ERR_CHECK	アドレスのアライメント、2byte 単位の書き込みのチェックを行います。
フラッシュメモリサスペンド関数使用の定義	USE_SUSPEND_FLASH_FUNCTION	未指定	SuspendFlash() 関数を使用する場合に定義します。 通常は SuspendErase() 関数を使用してください。
フラッシュメモリアドレスの定義	F_ADR_SIZE	Far	書き込み先のフラッシュメモリを指す書き込みポインタの定義を指定してください。データ領域のみの書き込みのときは near に設定できます。

5. ドライバプログラム

ドライバプログラムの使用例を、main_m16c.c 内の main に記載します。
使用方法は、main_m16c.c のファイルを参照ください。

main 処理仕様を以下に示します。

- M16C/28 Xin=20MHz にて動作。
- 20ms 毎のメインループをタイマ A の割り込み処理にて作成。
- 1 秒毎にブロック B を消去。
- 消去は 20ms 毎に中断されメインループに復帰。この処理はタイマ A1 の割り込み処理内で行います。
- 消去完了後 0F000₁₆ から 32byte 毎に書き込みとそのデータの読み出しを行う。

5.1 ソースコード

5.1.1 flashdevconf.h

```
/*
*****
*/
/* FILE NAME : flashdevconf.h */
/* Ver : 1.00 */
/* CPU : M16C/Tiny R8C/Tiny */
/* FUNCTION : Flash erase/read/write driver. */
/* by EWO or EW1 mode operation */
/*-----*/
/* Copyright(C)2004, Renesas Technology Corp. */
/* Copyright(C)2004, Renesas Solutions Corp. */
/* All rights reserved. */
*****
*/
// $Id: flashdevconf.h,v 1.12 2004/08/30 05:16:21 kato Exp $
*/

#ifndef __FLASHDEVCONF_H__
#define __FLASHDEVCONF_H__

/** Select CPU TYPE *****/
// #define R8C_SERIES
// #define R8C_10
// #define R8C_11
// #define R8C_12
// #define R8C_13
#define M16C_SERIES
// #define M16_26
// #define M16_26A
#define M16_28
// #define M16_29

/** Write Mode Define (EWO or EW1) *****/
// #define FLASH_MODE_EWO
#define FLASH_MODE_EW1

// Watch Dog Reset Enable
// #define ENABLE_WATCHDOG_RESET

// M16C only//
// #define ENABLE_FLASH_ERR_CHECK

// USE SuspendFlash function (EWO mode)
// #define USE_SUSPEND_FLASH_FUNCTION

#if defined(M16C_SERIES)
#define F_ADR_SIZE far
#elif defined(R8C_SERIES)
#define F_ADR_SIZE
#undef ENABLE_FLASH_ERR_CHECK
#else
#error "Please choose either R8C_SERIES or M16C_SERIES. "
#endif

/** Check !! *****/
#if (!defined(FLASH_MODE_EWO) && !defined(FLASH_MODE_EW1))
#error "Please choose FLASH_MODE_EWO , FLASH_MODE_EW1 or both. "
#endif

#endif /* #ifndef __FLASHDEVCONF_H__ */
```

5.1.2 flashdevdrv.h

```

/*****/
/* FILE NAME : flashdevdrv.h */
/* Ver : 1.00 */
/* CPU : R8C/Tiny */
/* FUNCTION : Flash erase/read/write driver. */
/* by EWO or EW1 mode operation */
/*-----*/
/* Copyright(C)2004, Renesas Technology Corp. */
/* Copyright(C)2004, Renesas Solutions Corp. */
/* All rights reserved. */
/*****/
/*
// $Id: flashdevdrv.h, v 1.9 2004/09/09 01:41:28 kato Exp $
*/

#include "flashdevconf.h"

#ifndef __FLASHDEVDRV_H__
#define __FLASHDEVDRV_H__

/** Write Mode Define (EWO or EW1) *****/
#if (defined(FLASH_MODE_EWO) && !defined(FLASH_MODE_EW1))
#define WriteFlash(a, b, c) WriteFlashEWO(a, b, c)
#define ReadFlash(a, b, c) ReadFlashEWO(a, b, c)
#define StartEraseFlash(a) StartEraseFlashEWO(a)
#define RestartEraseFlash() RestartEraseFlashEWO()
#define SuspendErase() SuspendEraseEWO()
#define ResumErase() ResumEraseEWO()
#define GetFlashStatus() GetFlashStatusEWO()
#endif /* #if (defined(FLASH_MODE_EWO) && !defined(FLASH_MODE_EW1)) */

#if (!defined(FLASH_MODE_EWO) && defined(FLASH_MODE_EW1))
#define WriteFlash(a, b, c) WriteFlashEW1(a, b, c)
#define ReadFlash(a, b, c) ReadFlashEW1(a, b, c)
#define StartEraseFlash(a) StartEraseFlashEW1(a)
#define RestartEraseFlash() RestartEraseFlashEW1()
#define SuspendErase() SuspendEraseEW1()
#define ResumErase() ResumEraseEW1()
#define GetFlashStatus() GetFlashStatusEW1()
#endif /* #if (!defined(FLASH_MODE_EWO) && defined(FLASH_MODE_EW1)) */

/*=====*/
/*! The address definition of the flash memory */
typedef void F_ADR_SIZE * F_ADR;

/*=====*/
/*! Error code */
typedef enum FlashResult{
    F_SUCCESS, /*! Success */
    F_WRITE_ERROR, /*! Program error (from the flash device) */
    F_ERASE_ERROR, /*! Block erase error (from the Flash device) */
    F_CMD_SEQUENCE_ERROR, /*! Command sequence error (from the Flash device) */
    F_WRITE_ADDRESS_ERROR, /*! The address alignment error of the argument */
    F_WRITE_SIZE_ERROR, /*! The size error of the argument */
    F_DEVICE_BUSY, /*! When a Read/Programing/Erasing requirement */
    /* in Programing/Erasing occurs. */
    F_SUSPEND, /*! Suspend requirement acceptance. */
    F_RESUM_ERROR, /*! When Suspend isn't being done with */
    /* ResumFlash(), call */
} FlashResult;

/*=====*/
/*!
* Erase flash memory in the EWO mode.
* @param flashAddress [in] physical address of the head of
* the block on flash.

```



```

* @return Success or Error code.
* @retval F_SUCCESS          Success
* @retval F_ERASE_ERROR      Block erase error (from the Flash device)
* @retval F_CMD_SEQUENCE_ERROR Command sequence error (from the Flash device)
* @retval F_WRITE_ADDRESS_ERROR The address alignment error of the argument.
* @retval F_DEVICE_BUSY      When a Read/Programing/Erasing requirement
*                             in Programing/Erasing occurs.
* @retval F_SUSPEND          Suspend requirement acceptance.
*/

```

```
FlashResult StartEraseFlashEWO(F_ADR flashAddress);
```

```
/*=====*/
```

```

/*!
* Erase flash memory in the EW1 mode.
* @param flashAddress      [in] physical address of the head of
*                           the block on flash.
* @return Success or Error code.
* @retval F_SUCCESS          Success
* @retval F_ERASE_ERROR      Block erase error (from the Flash device)
* @retval F_CMD_SEQUENCE_ERROR Command sequence error (from the Flash device)
* @retval F_WRITE_ADDRESS_ERROR The address alignment error of the argument.
* @retval F_DEVICE_BUSY      When a Read/Programing/Erasing requirement
*                             in Programing/Erasing occurs.
* @retval F_SUSPEND          Suspend requirement acceptance.
*/

```

```
FlashResult StartEraseFlashEW1(F_ADR flashAddress);
```

```
/*=====*/
```

```

/*!
* Restart to erase a block of the flash memory in the EWO mode.
* @return Success or Error code.
* @retval F_SUCCESS          Success
* @retval F_ERASE_ERROR      Block erase error (from the Flash device)
* @retval F_CMD_SEQUENCE_ERROR Command sequence error (from the Flash device)
* @retval F_WRITE_ADDRESS_ERROR The address alignment error of the argument.
* @retval F_DEVICE_BUSY      When a Read/Programing/Erasing requirement
*                             in Programing/Erasing occurs.
* @retval F_SUSPEND          Suspend requirement acceptance.
*/

```

```
FlashResult RestartEraseFlashEWO(void);
```

```
/*=====*/
```

```

/*!
* Restart to erase a block of the flash memory in the EW1 mode.
* @return Success or Error code.
* @retval F_SUCCESS          Success
* @retval F_ERASE_ERROR      Block erase error (from the Flash device)
* @retval F_CMD_SEQUENCE_ERROR Command sequence error (from the Flash device)
* @retval F_WRITE_ADDRESS_ERROR The address alignment error of the argument.
* @retval F_DEVICE_BUSY      When a Read/Programing/Erasing requirement
*                             in Programing/Erasing occurs.
* @retval F_SUSPEND          Suspend requirement acceptance.
*/

```

```
FlashResult RestartEraseFlashEW1(void);
```

```
/*=====*/
```

```

/*!
* Write data to flash memory in the EWO mode.
* @param flashAddress      [in] physical address on flash to begin write
* @param buffer            [in] address in buffer to write from
* @param size              [in] number of byte to write.
* @return Success or Error code.
* @retval F_SUCCESS          Success
* @retval F_WRITE_ERROR      Program error (from the flash device)
*/

```

```

* @retval F_CMD_SEQUENCE_ERROR    Command sequence error (from the Flash device)
* @retval F_WRITE_ADDRESS_ERROR    The address alignment error of the argument.
* @retval F_WRITE_SIZE_ERROR,      The size error of the argument
* @retval F_DEVICE_BUSY            When a Read/Programing/Erasing requirement
*                                  in Programing/Erasing occurs.
*/
FlashResult WriteFlashEWO(F_ADR flashAddress,
                          const void * buffer,
                          unsigned short size);

/*=====*/
/*!
* Write data to flash memory in the EW1 mode.
* @param flashAddress    [in] physical address on flash to begin write.
* @param buffer          [in] address in buffer to write from.
* @param size            [in] number of byte to write.
* @return Success or Error code.
* @retval F_SUCCESS      Success
* @retval F_WRITE_ERROR  Program error (from the flash device)
* @retval F_CMD_SEQUENCE_ERROR    Command sequence error (from the Flash device)
* @retval F_WRITE_ADDRESS_ERROR    The address alignment error of the argument.
* @retval F_WRITE_SIZE_ERROR,      The size error of the argument
* @retval F_DEVICE_BUSY            When a Read/Programing/Erasing requirement
*                                  in Programing/Erasing occurs.
*/
FlashResult WriteFlashEW1(F_ADR flashAddress,
                          const void * buffer,
                          unsigned short size);

/*=====*/
/*!
* Read data from flash memory for the EW1 mode.
* @param flashAddress    [in] physical address on flash to begin read.
* @param buffer          [out] address in buffer to read to.
* @param size            [in] number of byte to read.
* @return Success or Error code.
* @retval F_SUCCESS      Success
* @retval F_DEVICE_BUSY  When a Read/Programing/Erasing requirement
*                                  in Programing/Erasing occurs.
*/
FlashResult ReadFlashEWO(F_ADR flashAddress,
                        void * buffer,
                        unsigned short size);

/*=====*/
/*!
* Read data from flash memory for the EW1 mode.
* @param flashAddress    [in] physical address on flash to begin read.
* @param buffer          [out] address in buffer to read to.
* @param size            [in] number of byte to read.
* @return Success or Error code.
* @retval F_SUCCESS      Success
* @retval F_DEVICE_BUSY  When a Read/Programing/Erasing requirement
*                                  in Programing/Erasing occurs.
*/
FlashResult ReadFlashEW1(F_ADR flashAddress,
                        void * buffer,
                        unsigned short size);

/*=====*/
/*! Block number */
enum FlashBlock{
    F_BLOCK_0,
    F_BLOCK_1,
    F_BLOCK_2,
    F_BLOCK_3,
    F_BLOCK_4,
    F_BLOCK_5,
    F_ALLBLOCK = -1,
    F_BLOCK_A = -2,

```

```

    F_BLOCK_B = -3,
};

/*=====*/
/*!
 * Disable write protect.
 * @param blockNumber      [in] The block number to protect it.
 */
void UnlockBlockFlash(enum FlashBlock blockNumber);

/*=====*/
/*!
 * Enable write protect.
 * @param blockNumber      [in] The block number to unprotect it.
 */
void LockBlockFlash(enum FlashBlock blockNumber);

/*=====*/
/*!
 * Send the suspension request of a flash memory in the EWO mode.
 */
void SuspendEraseEWO(void);

/*=====*/
/*!
 * Send the suspension request of a flash memory in the EW1 mode.
 */
void SuspendEraseEW1(void);

/*=====*/
/*!
 * Clear the suspension request of a flash memory in the EWO mode.
 */
void ResumEraseEWO(void);

/*=====*/
/*!
 * Clear the suspension request of a flash memory in the EW1 mode.
 */
void ResumEraseEW1(void);

/*=====*/
/*! Status of flash memory or this driver. */
typedef enum FlashStatus{
    FLASH_READY,           /*! ready                */
    FLASH_WRITE,          /*! programing           */
    FLASH_ERASE,          /*! erasing              */
    FLASH_SUSPEND,        /*! suspend erasing     */
    FLASH_INT_SUSPEND,    /*! suspend erasing during the interruption */
}FlashStatus;

/*=====*/

#ifdef FLASH_MODE_EWO

/*=====*/
/*!
 * Suspend at interrupt.
 */
FlashResult SuspendFlashEWO(void);

/*=====*/
/*!
 * This function resumes the erasure processing suspended by SuspendFlashEWO().
 */

```

```
FlashResult ResumFlashEW0(void);
```

```
#endif
```

```
#endif /* #ifndef __FLASHDEVDRV_H__ */
```

5.1.3 flashm16c.h

```

/*****/
/* FILE NAME : flashm16c.h */
/* Ver : 1.00 */
/* CPU : M16C */
/* FUNCTION : Flash erase/read/write driver. */
/* by EW0 or EW1 mode operation */
/*-----*/
/* Copyright(C)2004, Renesas Technology Corp. */
/* Copyright(C)2004, Renesas Solutions Corp. */
/* All rights reserved. */
/*****/
/*
// $Id: flashm16c.h,v 1.6 2004/09/09 01:32:51 kato Exp $
*/

#include "flashdevconf.h"

#if defined( M16_26 )
#include "sfr26.h"
#elif defined( M16_26A )
#include "sfr26a.h"
#elif defined( M16_28 )
#include "sfr28.h"
#elif defined( M16_29 )
#include "sfr29.h"
#else
#error "This cpu type not support."
#endif

#ifndef __FLASHM16C_H__
#define __FLASHM16C_H__

//typedef volatile unsigned short F_ADR_SIZE * DEPEND_F_ADR;

typedef volatile unsigned short DEPEND_FSIZE;

#define FLASH_READARRAY_CMD ((unsigned short)0xff)
#define FLASH_STSREGS_CMD ((unsigned short)0x70)
#define FLASH_CLEAR_STSREGS_CMD ((unsigned short)0x50)
#define FLASH_PRG_CMD ((unsigned short)0x40)
#define FLASH_BLOCK_ERASE_1_CMD ((unsigned short)0x20)
#define FLASH_BLOCK_ERASE_2_CMD ((unsigned short)0xd0)

#define _FLASH_E_EW() {¥
    fmr01 = 0; ¥
    asm(""); ¥
    fmr01 = 1; ¥
    asm(""); ¥
}

#define _FLASH_DIS_EW() {¥
    fmr01 = 0; ¥
}

#define _FLASH_E_SUSPEND() {¥
    fmr40 = 0; ¥
    asm(""); ¥
    fmr40 = 1; ¥
    asm(""); ¥
}

#define _FLASH_DIS_SUSPEND() {¥
    fmr40 = 0; ¥
}

#define _FLASH_SUSPEND_ERASE() {¥

```

```

    fmr41 = 1;¥
}
#define _FLASH_RESUME_ERASE() {¥
    fmr41 = 0;¥
}

#define _FLASH_RESUME_ERASE_EW1(a) {¥
    fmr41 = a;¥
}

#define _FLASH_EWO_MODE() {¥
    fmr11 = 0;¥
}

#define _FLASH_EW1_MODE() {¥
    fmr11 = 0;¥
    fmr11 = 1;¥
}

#define _CLEAR_WATCHDOG() {¥
    wdts = 0x7fff;¥
}

#define _FLASH_BUSY() (fmr00 == 0)

#define _FLASH_READY() (fmr00 == 1)

#define _DATA_FLASH_ENA() {¥
    prcr = 0x3;    /* Unlock CMO, CM1, PM1 */¥
    pm10 = 1;     /* enable flash data block (4KB Virtual EEPROM) access */¥
    prcr = 0;     /* Lock the System Clock Control Register */¥
}

#define _DATA_FLASH_DIS() {¥
    prcr = 0x3;    /* Unlock CMO, CM1, PM1 */¥
    pm10 = 0;     /* disable flash data block (4KB Virtual EEPROM) access */¥
    prcr = 0;     /* Lock the System Clock Control Register */¥
}

#define ERASE_ERR 0x40
#define PRGRAM_ERR 0x80
#define FMRO7_06 (PRGRAM_ERR | ERASE_ERR)

#define _FLASH_GET_STAT_FLG() (fmr0 & FMRO7_06)

#define DEBUG_OUT_ERASE_START 0x04 /* xxxx100 */
#define DEBUG_OUT_SUSPEND 0x02 /* xxxx010 */
#define DEBUG_OUT_READY 0x07 /* xxxx111 */
#define DEBUG_OUT_WAIT_ERASE 0xFF /* not use */

// #define DEBUG_M16C
#ifdef DEBUG_M16C
#define DEBUG_OUT(a) {¥
    if(a != 0xFF) { ¥
        p7_0 = (0x01 & a); /* 0:Eraseing 1:Eraseed */ ¥
        p7_1 = (0x01 & (a>>1)); /* 0 Erase 1:suspend or erase complete */ ¥
        p7_2 = (0x01 & (a>>2)); /* 0:Suspend 1: not Suspend */ ¥
    } ¥
}
#else
#define DEBUG_OUT(a)
#endif /* #ifdef DEBUG_M16C */

struct LockBitStatus{
    unsigned char s_fmr16:1;
    unsigned char s_fmr02:1;
};

/*! for SlowMCU/RestoreMCU */

```

```
#define USE_CLOCKGEAR
typedef struct ProcessorMode
{
    unsigned char p_pm1;
    unsigned char p_cm0;
    unsigned char p_cm1;
} ProcessorMode;
void SlowMCU(ProcessorMode * save);
void RestoreMCU(ProcessorMode * save);
/*=====*/
/*!
 * Initialize flash device.
 */
void FlashInitialize(void);

#endif /* #ifndef __FLASHM16C_H__ */
```

5.1.4 flashdrvdev_ew1.c

```

/*****/
/* FILE NAME : flashdrvdev_ew1.c */
/* Ver : 1.00 */
/* CPU : R8C/Tiny M16C/26, 26A, 28, 29 */
/* FUNCTION : Flash low level (erase/read/write) driver. */
/* by EW1 mode operation */
/*-----*/
/* Copyright(C)2004, Renesas Technology Corp. */
/* Copyright(C)2004, Renesas Solutions Corp. */
/* All rights reserved. */
/*****/
/*
// $Id: flashdrvdev_ew1.c,v 1.22 2004/09/10 04:05:05 kato Exp $
*/
#include <string.h>
#include "flashdevdrv.h"

#if defined(M16C_SERIES)
#include "flashm16c.h"
#endif
#if defined(R8C_SERIES)
#include "flashr8c.h"
#endif

#ifndef FLASH_MODE_EW1

/*===== RAM =====*/
/*! Flash status */
enum FlashStatus stat = FLASH_READY;

/*! Flash status */
unsigned char suspendReq;

/*! Command target address. */
DEPEND_FSIZE F_ADR_SIZE * com_adr;

/*=== Prototype ===*/
FlashResult EraseCheckInternal(void);
FlashResult FullStatusCheck(void);
void SetUpLockBit(void);

void WaitEraseComplete(void);

/*****
Name :ReadFlashEW1()
Purpose :Read data from flash memory.
Arguments :
[in] flashAddress physical add on flash to begin read
[out] buf address in buffer to read to
[in] size number of byte to read
Return :Result
Notice :None
*****/
FlashResult ReadFlashEW1(F_ADR flashAddress,
void * buf,
unsigned short size)
{
if(stat != FLASH_READY && stat != FLASH_SUSPEND)return F_DEVICE_BUSY;
memcpy( buf , flashAddress , size);
return F_SUCCESS;
}

/*****
Name :StartEraseFlashEW1()
Purpose :Start to erase 1 block of the flash memory.

```



```

Arguments :physical address of the head of the block on flash.
Return    :Result
Notice    :This function erase and return result only.
           This function does not perform retry at the time of error.
*****/
FlashResult StartEraseFlashEW1(F_ADR flashAddress)
{
    FlashResult ret;

#if defined(USE_CLOCKGEAR)
    ProcessorMode save_dat;
#endif

    DEPEND_FSIZE F_ADR_SIZE * adr = flashAddress;

#ifdef ENABLE_FLASH_ERR_CHECK
    if(0x01 & ((unsigned long)adr))return F_WRITE_ADDRESS_ERROR;
#endif

    // disable interrupt.
    asm("FCLR I");
    if(stat != FLASH_READY){
        asm("FSET I");
        return F_DEVICE_BUSY;
    }
    asm("FSET I");

    suspendReq = 0;        /* initialize request */

#if defined(USE_CLOCKGEAR)
    SlowMCU(&save_dat);    /* Must change main clock speed to meet flash */
#endif

    asm("FCLR I");
    _FLASH_E_EW();
    asm("FSET I");

    asm("FCLR I");
    _FLASH_EW1_MODE();
    asm("FSET I");

    asm("FCLR I");
    SetUpLockBit();
    asm("FSET I");

    asm("FCLR I");
    _FLASH_E_SUSPEND();
    asm("FSET I");

    //////////////////////////////////////
    // Clear status register.
    //////////////////////////////////////
    asm("FCLR I");
    *adr = FLASH_CLEAR_STSREGS_CMD;    /* Clear status register */
    com_adr = adr;
    stat = FLASH_ERASE;
    //////////////////////////////////////
    // Flash command (Erase) entry.
    //////////////////////////////////////
    *adr = FLASH_BLOCK_ERASE_1_CMD;    /* Erase command 1 */
    *adr = FLASH_BLOCK_ERASE_2_CMD;    /* Erase command 2 */
    asm("FSET I");

    WaitEraseComplete();

    ret = EraseCheckInternal();

    if(_FLASH_READY())
    {

```

```

    _FLASH_DIS_SUSPEND();
}
    _FLASH_DIS_EW();

#if defined(USE_CLOCKGEAR)
    RestoreMCU(&save_dat);          /* Restore clock back to original speed */
#endif

    return ret;
}

/*****
Name      :RestartEraseFlashEW1()
Purpose   :Restart to erase 1 block of the flash memory.
Return    :Result
Notice    :This function erases flash memory with operating erase command.
           It does not perform retry at the time of error.
*****/
FlashResult RestartEraseFlashEW1(void)
{
    FlashResult ret;
#if defined(USE_CLOCKGEAR)
    ProcessorMode save_dat;
#endif

    asm("FCLR I");
    if(stat != FLASH_SUSPEND)
    {
        asm("FSET I");
        return F_DEVICE_BUSY;
    }
    asm("FSET I");

#if defined(USE_CLOCKGEAR)
    SlowMCU(&save_dat);          /* Must change main clock speed to meet flash */
#endif

    asm("FCLR I");
    _FLASH_E_EW();
    asm("FSET I");

    asm("FCLR I");
    SetUpLockBit();
    asm("FSET I");

    stat = FLASH_ERASE;

    WaitEraseComplete();
    ret = EraseCheckInternal();

    if(_FLASH_READY())
    {
        _FLASH_DIS_SUSPEND();
    }
    _FLASH_DIS_EW();

#if defined(USE_CLOCKGEAR)
    RestoreMCU(&save_dat);          /* Restore clock back to original speed */
#endif

    return ret;
}

/*****
Name      :EraseCheckInternal()
Purpose   :The erased result is judged.
*****/

```

```

Return      :Result.
Notice     :
*****/
FlashResult EraseCheckInternal(void)
{
    FlashResult ret;
    if(_FLASH_BUSY()){

        ret = F_SUSPEND;
        stat = FLASH_SUSPEND;
    }
    else{

        ret = FullStatusCheck();      /* Erasing error? */
        stat = FLASH_READY;
    }
    return ret;
}

/*****
Name       :WriteFlashEW1()
Purpose    :Write data to flash memory.
Arguments  :Address
    [in] flashAddress  physical add on flash to begin write
    [in] buf           address in buffer to write from
    [in] size          number of byte to write
Return     :Result
Notice     :This function writes flash memory with operating write command.
            It does not perform retry at the time of error.
*****/
FlashResult WriteFlashEW1(F_ADR flashAddress,
                          const void * buffer,
                          unsigned short size)
{
    #if defined(USE_CLOCKGEAR)
        ProcessorMode save_dat;
    #endif
    FlashResult ret;
    unsigned short data;
    DEPEND_FSIZE F_ADR_SIZE * adr = flashAddress;
    const DEPEND_FSIZE * buf = buffer;

    #ifndef ENABLE_FLASH_ERR_CHECK
        if(0x01 & ((unsigned long)adr))return F_WRITE_ADDRESS_ERROR;
        if(0x01 & size)return F_WRITE_SIZE_ERROR;
    #endif

    /* disable interrupt. */
    asm("FCLR I");
    if(stat != FLASH_READY) {
        asm("FSET I");
        return F_DEVICE_BUSY;
    }
    asm("FSET I");

    #if defined(USE_CLOCKGEAR)
        SlowMCU(&save_dat);          /* Must change main clock speed to meet flash */
    #endif

    asm("FCLR I");
    _FLASH_E_EW();
    asm("FSET I");

    asm("FCLR I");
    _FLASH_EW1_MODE();
    asm("FSET I");

    asm("FCLR I");

```

```

SetupLockBit();
asm("FSET 1");

ret = F_SUCCESS;
while(size > 0)
{
    asm("FCLR 1");
    stat = FLASH_WRITE;
    * adr = FLASH_CLEAR_STSREGS_CMD; /* Clear status register */
    data = *buf;
    * adr = FLASH_PRG_CMD; /* Send write command */
    * adr = data; /* Write next word of data */
    asm("FSET 1");

#ifdef ENABLE_WATCHDOG_RESET
    _CLEAR_WATCHDOG();
#endif

    ret = FullStatusCheck(); /* Write error? */
    stat = FLASH_READY;

    if(ret != F_SUCCESS) break;
    size -= sizeof(DEPEND_FSIZE); /* subtract 2 from byte counter */
    buf++; /* increase to next data index */
    adr++; /* increase to next flash index */
}

_FLASH_DIS_EW();

#ifdef USE_CLOCKGEAR
    RestoreMCU(&save_dat); /* Restore clock back to original speed */
#endif

return ret; /* Write Pass */
}

/*****
Name :FullStatusCheck()
Purpose :Check the status of flash memory.
Arguments :None
Return :Result
*****/
FlashResult FullStatusCheck(void)
{
    unsigned char reg;
    reg = _FLASH_GET_STAT_FLG();
    if(reg == 0) return F_SUCCESS;
    if(reg == PRGRAM_ERR) return F_WRITE_ERROR;
    if(reg == ERASE_ERR) return F_ERASE_ERROR;
    return F_CMD_SEQUENCE_ERROR;
}

/*****
Name :SuspendEraseEW1()
Purpose :Send the suspension request of a flash memory.
Return :None
*****/
void SuspendEraseEW1(void)
{
    if(stat == FLASH_ERASE || stat==FLASH_SUSPEND)
    {
        /* send event to wait roop */
        suspendReq = 1;
    }
}

/*****
Name :ResumEraseEW1()
Purpose :This function clears suspension request.
Return :None
*****/

```

```
*****/
void ResumEraseEW1(void)
{
    if(stat==FLASH_ERASE || stat==FLASH_SUSPEND)
    {
        /* send event to wait roop */
        suspendReq = 0;
    }
}

/*****
Name      :WaitEraseComplete
Purpose   :Wait until complete or suspended.
           Flash memory will be set "Read Array" mode (you can read data on
           flash memory) ,if this function return.
Return    :None
Notice    :None
*****/
void WaitEraseComplete (void)
{
    while(_FLASH_BUSY())
    {
        stat = FLASH_SUSPEND;
#ifdef ENABLE_WATCHDOG_RESET
        _CLEAR_WATCHDOG();
#endif
        if(suspendReq)
        {
            suspendReq = 0;
            return;
        }
        else
        {
            asm("FCLR I");
            _FLASH_RESUME_ERASE_EW1 (suspendReq);
            asm("FSET I");
        }
    }
    return ;
}

#endif /* #ifdef FLASH_MODE_EW1 */
```

5.1.5 depend_m16c.c

```

/*****/
/* FILE NAME : depend_m16c.c */
/* Ver : 1.00 */
/* CPU : M16C/26, 26A, 28, 29 */
/* FUNCTION : Flash low level (erase/read/write) driver for M16C */
/*-----*/
/* Copyright(C)2004, Renesas Technology Corp. */
/* Copyright(C)2004, Renesas Solutions Corp. */
/* All rights reserved. */
/*****/
/*
// $Id: depend_m16c.c,v 1.3 2004/08/18 08:04:03 ikari Exp $
*/

#include "flashdevdrv.h"
#include "flashm16c.h"

#ifdef M16C_SERIES
#error "This file is only for M16C."
#endif

/*! Protect flash status */
struct LockBitStatus fmr_status = {0,0};

/*****/
Name: FlashInitialize()
Purpose :Initial the flash memory
Arguments :none
Return :nonr
Notice :None
/*****/
void FlashInitialize(void)
{
    _DATA_FLASH_ENA();
}

/*****/
Name :SlowMCU()
Purpose :In order to rewrite a flash memory, the clock of MCU operation is made late.
        (Flash Access time nomal -> slow)
Arguments :address in area to save CPU clock setting.
Return :None
Notice :It depends on your system.
        You must modify this function for your system.
        This function is the example of "M16C/26 Xin=20MHz."
/*****/
void SlowMCU(ProcessorMode * save)
{
    asm("FCLR I");
    save->p_cm0 = cm0; /* Save current CPU clock setting */
    save->p_cm1 = cm1;
    save->p_pm1 = pm1;

/* "Modify for your system. This code is sample for M16C/26 Xin=20MHz." */

    fmr17 = 1; /* Set lwait to Block A/B access */
    prcr = 3; /* Unprotect registers CMO and PMO */
    cm16 = 1; /* Use Xin, Xin drive HIGH, Xin/2 (f2) */
    cm17 = 0; /* Use Xin, Xin drive HIGH, Xin/2 (f2) */
    cm06 = 0; /* CM16 and CM17 are valid */
    pm17 = 1; /* enable flash data block 1 wait access */
    prcr = 0; /* Protection register back on */
    asm("FSET I");
}
/*****/
Name :RestoreMCU()

```

```

Purpose      :Restore MCU clock.
Arguments    :address in area to load CPU clock setting.
Return       :None
Notice       :It depends on your system.
              You must modify this function for your system.
              This function is the example of "M16C/26 Xin=20MHz."
*****/
void RestoreMCU(ProcessorMode * save)
{
    asm("FCLR I");
    fmr17 = 0;          /* Set none wait to Block A/B access */
    prcr = 3;          /* Unprotect registers CMO and PMO */
    pm1 = save->p_pm1;
    cm1 = save->p_cm1;
    cm0 = save->p_cm0;

    prcr = 0;          /* Protection register back on */
    asm("FSET I");
}

/*****
Name          :UnlockBlockFlash()
Purpose       :Disable write protect.
Arguments     :F_BLOCK_0,F_BLOCK_1,F_ALLBLOCK    All area writing is possible.
              F_BLOCK_2,F_BLOCK_3,F_BLOCK_4,F_BLOCK_5
              Block2 to 4 area writing is possible. (only M16C/26A, 28, 29)
Return        :None
Notice        :It depends on your system.
*****/
void UnlockBlockFlash(enum FlashBlock blockNumber)
{
    #if (defined(M16_26A) || defined(M16_28) || defined(M16_29))
        ProcessorMode save_dat;
    #endif
    asm("FCLR I");
    switch(blockNumber) {
        case F_BLOCK_0: case F_BLOCK_1: case F_ALLBLOCK:
            fmr_status.s_fmr02 = 1;
            /* Not break */
            #if (defined(M16_26A) || defined(M16_28) || defined(M16_29))
                case F_BLOCK_2: case F_BLOCK_3: case F_BLOCK_4: case F_BLOCK_5:
                    SlowMCU(&save_dat);    // Must change main clock speed to meet flash
                    _FLASH_E_EW();
                    fmr16 = 0;
                    fmr16 = 1;
                    _FLASH_DIS_EW();
                    RestoreMCU(&save_dat);    // Restore clock back to original speed
            #endif
            break;
        // case F_ALLBLOCK: case F_BLOCK_A : case F_BLOCK_B :
        default:
            break;
    }
    asm("FSET I");
}

/*****
Name          :LockBlockFlash()
Purpose       :Enable write protect.
Arguments     :F_BLOCK_0,F_BLOCK_1                Block0,1 are locked.
              F_BLOCK_2,F_BLOCK_3,F_BLOCK_4,F_BLOCK_5
              all area is locked.
              F_BLOCK_A,F_BLOCK_B don't have bit for write to lock.
Return        :None
Notice        :It depends on your system.
*****/
void LockBlockFlash(enum FlashBlock blockNumber)
{
    #if (defined(M16_26A) || defined(M16_28) || defined(M16_29))
        ProcessorMode save_dat;
    #endif

```

```
#endif
asm("FCLR I");
switch(blockNumber) {
#if (defined(M16_26A) || defined(M16_28) || defined(M16_29))
case F_BLOCK_2: case F_BLOCK_3: case F_BLOCK_4: case F_BLOCK_5: case F_ALLBLOCK:
    SlowMCU(&save_dat);    // Must change main clock speed to meet flash
    _FLASH_E_EW();
    fmr16 = 0;
    _FLASH_DIS_EW();
    RestoreMCU(&save_dat);    // Restore clock back to original speed
    /* not break */
#endif
case F_BLOCK_0: case F_BLOCK_1:
    fmr_status.s_fmr02 = 0;
    break;
// case F_ALLBLOCK: case F_BLOCK_A : case F_BLOCK_B :
default:
    break;
}
asm("FSET I");
}
/*****
Name      :SetupLockBit()
Purpose   :set up lock bit.
Notice    :It depends on your system.
*****/
void SetupLockBit(void)
{
    if(fmr_status.s_fmr02) {
        fmr02 = 0;
        fmr02 = 1;
    }
}
```


5.1.6 ncrto_EW1.a30

```

;***** ;
; C COMPILER for R8C/Tiny, M16C/60,30,20,10
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;
; ncrto.a30 : NC30 startup program
;
; This program is applicable when using the basic I/O library
;
; $Id: ncrto_EW1.a30,v 1.1 2004/06/25 02:25:54 nagayoshi Exp $
;
;*****

;-----
; HEAP SIZE definition
;-----
.if __HEAP__ == 1
HEAPSIZE .equ 0H
.else
.if __HEAPSIZE__ == 0
HEAPSIZE .equ 50H
.else
HEAPSIZE .equ __HEAPSIZE__
.endif
.endif

;-----
; STACK SIZE definition
;-----
.if __USTACKSIZE__ == 0
STACKSIZE .equ 300h
.else
STACKSIZE .equ __USTACKSIZE__
.endif

;-----
; INTERRUPT STACK SIZE definition
;-----
.if __ISTACKSIZE__ == 0
ISTACKSIZE .equ 300h
.else
ISTACKSIZE .equ __ISTACKSIZE__
.endif

;-----
; INTERRUPT VECTOR ADDRESS definition
;-----
.if __R8C__ != 1
;VECTOR_ADR .equ 0ffd00h
VECTOR_ADR .equ 0ff700h
.else
VECTOR_ADR .equ 0fedch
.endif

;-----
; Section allocation
;-----
.list OFF
.include sect30_EW1.inc
.list ON

;-----
; SBDATA area definition

```

```

;-----
; .glb    __SB__
__SB__ .equ    data_SE_top
;=====
; Initialize Macro declaration
;-----
N_BZERO    .macro    TOP_ , SECT_
    mov.b    #00H, R0L
    mov.w    #(TOP_ & 0FFFFH), A1
    mov.w    #sizeof SECT_ , R3
    sstr.b
    .endm

N_BCOPY .macro    FROM_ , TO_ , SECT_
    mov.w    #(FROM_ & 0FFFFH), A0
    mov.b    #(FROM_ >>16), R1H
    mov.w    #TO_ , A1
    mov.w    #sizeof SECT_ , R3
    smovf.b
    .endm

BZERO .macro    TOP_ , SECT_
    push.w   #sizeof SECT_ >> 16
    push.w   #sizeof SECT_ & 0ffffh
    pusha    TOP_ >>16
    pusha    TOP_ & 0ffffh
    .stk     8
    .glb    _bzero
    .call   _bzero,G
    jsr.a   _bzero
    .endm

BCOPY .macro    FROM_ , TO_ , SECT_
    push.w   #sizeof SECT_ >> 16
    push.w   #sizeof SECT_ & 0ffffh
    pusha    TO_ >>16
    pusha    TO_ & 0ffffh
    pusha    FROM_ >>16
    pusha    FROM_ & 0ffffh
    .stk     12
    .glb    _bcopy
    .call   _bcopy,G
    jsr.a   _bcopy
    .endm

.if    __R8C__ != 1
;
; for M16C/60,30,20,10 series
;
;
; .glb    __BankSelect
__BankSelect .equ    OBH
;-----
; special page definition
;-----
; macro define for special page
;
;Format:
; SPECIAL number
;
SPECIAL    .macro    NUM
    .org    OFFFEEH-(NUM*2)
    .glb    __SPECIAL_@NUM
    .word   __SPECIAL_@NUM & 0FFFFH
    .endm
;=====
; Interrupt section start
;-----

```

```

.insf start, S, 0
.glob start
.section interrupt
start:
;-----
; after reset, this program will start
;-----
    ldc    #istack_top,    isp    ;set istack pointer
    mov.b  #03h, 0ah
    mov.b  #00h, 04h        ;set processor mode
    mov.b  #00100000B, 07h  ;set CM1
    mov.b  #00000000B, 06h  ;set CM0
    mov.b  #00000000B, 0Ch  ;set CM2
    mov.b  #00h, 0ah
    ldc    #0080h, flg
    ldc    #stack_top,    sp    ;set stack pointer
    ldc    #data_SE_top,  sb    ;set sb register
    ldintb #VECTOR_ADR
;=====
; Program RAM Initialize
;-----
; program area
;-----
    N_BCOPY _from_addr, _program_ram_top, program_ram
;=====
; NEAR area initialize.
;-----
; bss zero clear
;-----
    N_BZERO bss_SE_top, bss_SE
    N_BZERO bss_SO_top, bss_SO
    N_BZERO bss_NE_top, bss_NE
    N_BZERO bss_NO_top, bss_NO
;-----
; initialize data section
;-----
    N_BCOPY data_SEI_top, data_SE_top, data_SE
    N_BCOPY data_SOI_top, data_SO_top, data_SO
    N_BCOPY data_NEI_top, data_NE_top, data_NE
    N_BCOPY data_NOI_top, data_NO_top, data_NO
;=====
; FAR area initialize.
;-----
; bss zero clear
;-----
;
; BZERO    bss_FE_top, bss_FE
; BZERO    bss_F0_top, bss_F0
;-----
; Copy edata_E(0) section from edata_EI(0I) section
;-----
;
; BCOPY    data_FEI_top, data_FE_top, data_FE
; BCOPY    data_F0I_top, data_F0_top, data_F0
;
    ldc    #stack_top, sp
    .stk   -40
;=====
; heap area initialize
;-----
.if __HEAP__ != 1
    .glob  __mbase
    .glob  __mnext
    .glob  __msize
    mov.w  #(heap_top&0FFFFH), __mbase
    mov.w  #(heap_top>>16), __mbase+2
    mov.w  #(heap_top&0FFFFH), __mnext

```

```

    mov.w    #(heap_top>>16), __mnext+2
    mov.w    #(HEAPSIZE&0FFFFH), __msize
    mov.w    #(HEAPSIZE>>16), __msize+2
.endif

;=====
; Initialize standard I/O
;-----

.if __STANDARD_IO__ != 1
    .glb    _init
    .call   _init,G
    jsr.a   _init
.endif

;=====
; Call main() function
;-----

    ldc     #0h,fb    ; for debugger

    .glb    _main
    jsr.a   _main

.else ; __R8C__

;-----
; for R8C/Tiny
;-----

;=====
; Interrupt section start
;-----

    .insf   start,S,0
    .glb    start
    .section interrupt
start:
;-----
; after reset, this program will start
;-----

    ldc     #istack_top,    isp    ;set istack pointer
    mov.b   #02h,0ah
    mov.b   #00h,04h        ;set processor mode
    mov.b   #00h,0ah
    ldc     #0080h, flg
    ldc     #stack_top,    sp     ;set stack pointer
    ldc     #data_SE_top,  sb     ;set sb register
    ldintb  #VECTOR_ADR

;-----
; NEAR area initialize.
;-----

; bss zero clear
;-----

    N_BZERO bss_SE_top,bss_SE
    N_BZERO bss_S0_top,bss_S0
    N_BZERO bss_NE_top,bss_NE
    N_BZERO bss_NO_top,bss_NO

;-----
; initialize data section
;-----

    N_BCOPY data_SEI_top,data_SE_top,data_SE
    N_BCOPY data_S0I_top,data_S0_top,data_S0
    N_BCOPY data_NEI_top,data_NE_top,data_NE
    N_BCOPY data_NOI_top,data_NO_top,data_NO

;-----
; FAR area initialize.
;-----

; bss zero clear
;-----

```

```

; BZERO    bss_FE_top, bss_FE
; BZERO    bss_F0_top, bss_F0

;-----
; Copy edata_E(0) section from edata_EI(01) section
;-----
; BCOPY    data_FEI_top, data_FE_top, data_FE
; BCOPY    data_F0I_top, data_F0_top, data_F0

        ldc    #stack_top, sp
; .stk     -40

;=====
; heap area initialize
;-----
.if __HEAP__ != 1
        .glb    __mbase
        .glb    __mnext
        .glb    __msize
        mov.w   #(heap_top&0FFFFH), __mbase
        mov.w   #(heap_top&0FFFFH), __mnext
        mov.w   #(HEAPSIZE&0FFFFH), __msize
.endif

;=====
; Program RAM Initialize
;-----
        N_BCOPY _from_addr, _program_ram_top, program_ram

;=====
; Initialize standard I/O
;-----
.if __STANDARD_IO__ != 1
        .glb    _init
        .call   _init, G
        jsr.a   _init
.endif

;=====
; Call main() function
;-----
        ldc    #0h, fb    ; for debugger

        .glb    _main
        jsr.a   _main

.endif: __R8C__

;=====
; exit() function
;-----
        .glb    _exit
        .glb    $exit
_exit:                                ; End program
$exit:
        jmp    _exit
        .einsf

;=====
; dummy interrupt function
;-----
dummy_int:
        reit

        .end

;*****
;
; C COMPILER for R8C/Tiny, M16C/60, 30, 20, 10
; COPYRIGHT (C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION

```

; AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

;

5.1.7 sect30_EW1.inc

```

:*****
:
: C Compiler for R8C/Tiny, M16C/60, 30, 20, 10
: COPYRIGHT (C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
: AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
:
:
: Written by T.Aoyama
:
: sect30.inc      : section definition
: This program is applicable when using the basic I/O library
:
: $Id: sect30_EW1.inc,v 1.1 2004/06/25 02:25:54 nagayoshi Exp $
:*****

.if   __R8C__ != 1
:
:   for M16C/60, 30, 20, 10
:
:-----
:
:   Arrangement of section
:
:-----
: Near RAM data area
:-----
: SBDATA area
:   .section data_SE, DATA
:   .org      400H
data_SE_top:

:   .section bss_SE, DATA, ALIGN
bss_SE_top:

:   .section data_S0, DATA
data_S0_top:

:   .section bss_S0, DATA
bss_S0_top:

: near RAM area
:   .section data_NE, DATA, ALIGN
data_NE_top:

:   .section bss_NE, DATA, ALIGN
bss_NE_top:

:   .section data_NO, DATA
data_NO_top:

:   .section bss_NO, DATA
bss_NO_top:

:-----
: Stack area
:-----
:   .section stack, DATA
:   .blkb    STACKSIZE
stack_top:

:   .blkb    ISTACKSIZE
istack_top:

:-----
: heap section

```

```
-----  
; .section heap, DATA  
heap_top:  
    .blkb    HEAPSIZE  
  
-----  
; RAM program area  
-----  
; .section program_ram, ALIGN  
_program_ram_top:  
.glb  _program_ram_top  
  
-----  
; Near ROM data area  
-----  
; .section rom_NE, ROMDATA  
    .org    0f000H  
rom_NE_top:  
  
    .section rom_NO, ROMDATA  
rom_NO_top:  
  
-----  
; Far RAM data area  
-----  
; .section data_FE, DATA  
    .org    10000H  
data_FE_top:  
  
    .section bss_FE, DATA, ALIGN  
bss_FE_top:  
  
    .section data_F0, DATA  
data_F0_top:  
  
    .section bss_F0, DATA  
bss_F0_top:  
  
-----  
; Far ROM data area  
-----  
; .section rom_FE, ROMDATA  
    .org    0f8000H  
rom_FE_top:  
  
    .section rom_F0, ROMDATA  
rom_F0_top:  
  
-----  
; Initial data of 'data' section  
-----  
; .section data_SEI, ROMDATA  
data_SEI_top:  
  
    .section data_S0I, ROMDATA  
data_S0I_top:  
  
    .section data_NEI, ROMDATA  
data_NEI_top:  
  
    .section data_NOI, ROMDATA  
data_NOI_top:  
  
    .section data_FEI, ROMDATA  
data_FEI_top:  
  
    .section data_F0I, ROMDATA  
data_F0I_top:
```



```

;-----
; Switch Table Section
;-----
        .section      switch_table,ROMDATA
switch_table_top:

;-----
; code area
;-----

        .section program

        .section interrupt
        .org          ;must be set internal ROM area
        .section program_S

;-----
; variable vector section
;-----
        .section vector,ROMDATA      ; variable vector table
        .org          VECTOR_ADR

.if     M60TYPE == 1
        .lword  dummy_int           ; vector 0 (BRK)
        .lword  dummy_int           ; vector 1
        .lword  dummy_int           ; vector 2
        .lword  dummy_int           ; vector 3
        .lword  dummy_int           ; vector 4
        .lword  dummy_int           ; vector 5
        .lword  dummy_int           ; vector 6
        .lword  dummy_int           ; vector 7
        .lword  dummy_int           ; vector 8
        .lword  dummy_int           ; vector 9
        .lword  dummy_int           ; vector 10
        .lword  dummy_int           ; DMA0 (for user) (vector 11)
        .lword  dummy_int           ; DMA1 2 (for user) (vector 12)
        .lword  dummy_int           ; input key (for user) (vector 13)
        .lword  dummy_int           ; AD Convert (for user) (vector 14)
        .lword  dummy_int           ; vector 15
        .lword  dummy_int           ; vector 16
        .lword  dummy_int           ; uart0 trance (for user) (vector 17)
        .lword  dummy_int           ; uart0 receive (for user) (vector 18)
        .lword  dummy_int           ; uart1 trance (for user) (vector 19)
        .lword  dummy_int           ; uart1 receive (for user) (vector 20)
        .lword  dummy_int           ; TIMER A0 (for user) (vector 21)
        .lword  dummy_int           ; TIMER A1 (for user) (vector 22)
        .lword  dummy_int           ; TIMER A2 (for user) (vector 23)
        .lword  dummy_int           ; TIMER A3 (for user) (vector 24)
        .lword  dummy_int           ; TIMER A4 (for user) (vector 25)
        .lword  dummy_int           ; TIMER B0 (for user) (vector 26)
        .lword  dummy_int           ; TIMER B1 (for user) (vector 27)
        .lword  dummy_int           ; TIMER B2 (for user) (vector 28)
        .lword  dummy_int           ; INTO (for user) (vector 29)
        .lword  dummy_int           ; INT1 (for user) (vector 30)
        .lword  dummy_int           ; INT2 (for user) (vector 31)
.else
        .glb      _timerA1_int
        .glb      _vec_dummy_int
        .lword  _vec_dummy_int      ; BRK      (vector 0)
        .lword  _vec_dummy_int      ;          (vector 1)
        .lword  _vec_dummy_int      ;          (vector 2)
        .lword  _vec_dummy_int      ;          (vector 3)
        .lword  _vec_dummy_int      ; int3(for user)(vector 4)
        .lword  _vec_dummy_int      ; timerB5(for user)(vector 5)
        .lword  _vec_dummy_int      ; timerB4(for user)(vector 6)
        .lword  _vec_dummy_int      ; timerB3(for user)(vector 7)
        .lword  _vec_dummy_int      ; si/o4 /int5(for user)(vector 8)
        .lword  _vec_dummy_int      ; si/o3 /int4(for user)(vector 9)
        .lword  _vec_dummy_int      ; Bus collision detection(for user)(v10)

```

```

.lword _vec_dummy_int      ; DMA0(for user)(vector 11)
.lword _vec_dummy_int      ; DMA1(for user)(vector 12)
.lword _vec_dummy_int      ; Key input interrupt(for user)(vect 13)
.lword _vec_dummy_int      ; A-D(for user)(vector 14)
.lword _vec_dummy_int      ; uart2 transmit(for user)(vector 15)
.lword _vec_dummy_int      ; uart2 receive(for user)(vector 16)
.lword _vec_dummy_int      ; uart0 transmit(for user)(vector 17)
.lword _vec_dummy_int      ; uart0 receive(for user)(vector 18)
.lword _vec_dummy_int      ; uart1 transmit(for user)(vector 19)
.lword _vec_dummy_int      ; uart1 receive(for user)(vector 20)
.lword _vec_dummy_int      ; timer A0(for user)(vector 21)
.lword _timerA1_int        ; timer A1(for user)(vector 22)
.lword _vec_dummy_int      ; timer A2(for user)(vector 23)
.lword _vec_dummy_int      ; timer A3(for user)(vector 24)
.lword _vec_dummy_int      ; timer A4(for user)(vector 25)
.lword _vec_dummy_int      ; timer B0(for user)(vector 26)
.lword _vec_dummy_int      ; timer B1(for user)(vector 27)
.lword _vec_dummy_int      ; timer B2(for user)(vector 28)
.lword _vec_dummy_int      ; int0 (for user)(vector 29)
.lword _vec_dummy_int      ; int1 (for user)(vector 30)
.lword _vec_dummy_int      ; int2 (for user)(vector 31)
.endif
.lword dummy_int           ; vector 32 (for user or MR30)
.lword dummy_int           ; vector 33 (for user or MR30)
.lword dummy_int           ; vector 34 (for user or MR30)
.lword dummy_int           ; vector 35 (for user or MR30)
.lword dummy_int           ; vector 36 (for user or MR30)
.lword dummy_int           ; vector 37 (for user or MR30)
.lword dummy_int           ; vector 38 (for user or MR30)
.lword dummy_int           ; vector 39 (for user or MR30)
.lword dummy_int           ; vector 40 (for user or MR30)
.lword dummy_int           ; vector 41 (for user or MR30)
.lword dummy_int           ; vector 42 (for user or MR30)
.lword dummy_int           ; vector 43 (for user or MR30)
.lword dummy_int           ; vector 44 (for user or MR30)
.lword dummy_int           ; vector 45 (for user or MR30)
.lword dummy_int           ; vector 46 (for user or MR30)
.lword dummy_int           ; vector 47 (for user or MR30)
.lword dummy_int           ; vector 48
.lword dummy_int           ; vector 49
.lword dummy_int           ; vector 50
.lword dummy_int           ; vector 51
.lword dummy_int           ; vector 52
.lword dummy_int           ; vector 53
.lword dummy_int           ; vector 54
.lword dummy_int           ; vector 55
.lword dummy_int           ; vector 56
.lword dummy_int           ; vector 57
.lword dummy_int           ; vector 58
.lword dummy_int           ; vector 59
.lword dummy_int           ; vector 60
.lword dummy_int           ; vector 61
.lword dummy_int           ; vector 62
.lword dummy_int           ; vector 63

```

```

;=====
; fixed vector section
;-----
.section fvector,ROMDATA           ; fixed vector table
;=====
; special page definition
;-----
; macro is defined in ncrto.a30
; Format: SPECIAL number
;
;-----
; SPECIAL 255
; SPECIAL 254
; SPECIAL 253
; SPECIAL 252

```

;
; SPECIAL 251
;
; SPECIAL 250
;
; SPECIAL 249
;
; SPECIAL 248
;
; SPECIAL 247
;
; SPECIAL 246
;
; SPECIAL 245
;
; SPECIAL 244
;
; SPECIAL 243
;
; SPECIAL 242
;
; SPECIAL 241
;
; SPECIAL 240
;
; SPECIAL 239
;
; SPECIAL 238
;
; SPECIAL 237
;
; SPECIAL 236
;
; SPECIAL 235
;
; SPECIAL 234
;
; SPECIAL 233
;
; SPECIAL 232
;
; SPECIAL 231
;
; SPECIAL 230
;
; SPECIAL 229
;
; SPECIAL 228
;
; SPECIAL 227
;
; SPECIAL 226
;
; SPECIAL 225
;
; SPECIAL 224
;
; SPECIAL 223
;
; SPECIAL 222
;
; SPECIAL 221
;
; SPECIAL 220
;
; SPECIAL 219
;
; SPECIAL 218
;
; SPECIAL 217
;
; SPECIAL 216
;
; SPECIAL 215
;
; SPECIAL 214
;
; SPECIAL 213
;
; SPECIAL 212
;
; SPECIAL 211
;
; SPECIAL 210
;
; SPECIAL 209
;
; SPECIAL 208
;
; SPECIAL 207
;
; SPECIAL 206
;
; SPECIAL 205
;
; SPECIAL 204
;
; SPECIAL 203
;
; SPECIAL 202
;
; SPECIAL 201
;
; SPECIAL 200
;
; SPECIAL 199
;
; SPECIAL 198
;
; SPECIAL 197
;
; SPECIAL 196
;
; SPECIAL 195
;
; SPECIAL 194
;
; SPECIAL 193
;
; SPECIAL 192
;
; SPECIAL 191
;
; SPECIAL 190
;
; SPECIAL 189
;
; SPECIAL 188
;
; SPECIAL 187
;
; SPECIAL 186
;
; SPECIAL 185
;
; SPECIAL 184
;
; SPECIAL 183
;
; SPECIAL 182

;
; SPECIAL 181
;
; SPECIAL 180
;
; SPECIAL 179
;
; SPECIAL 178
;
; SPECIAL 177
;
; SPECIAL 176
;
; SPECIAL 175
;
; SPECIAL 174
;
; SPECIAL 173
;
; SPECIAL 172
;
; SPECIAL 171
;
; SPECIAL 170
;
; SPECIAL 169
;
; SPECIAL 168
;
; SPECIAL 167
;
; SPECIAL 166
;
; SPECIAL 165
;
; SPECIAL 164
;
; SPECIAL 163
;
; SPECIAL 162
;
; SPECIAL 161
;
; SPECIAL 160
;
; SPECIAL 159
;
; SPECIAL 158
;
; SPECIAL 157
;
; SPECIAL 156
;
; SPECIAL 155
;
; SPECIAL 154
;
; SPECIAL 153
;
; SPECIAL 152
;
; SPECIAL 151
;
; SPECIAL 150
;
; SPECIAL 149
;
; SPECIAL 148
;
; SPECIAL 147
;
; SPECIAL 146
;
; SPECIAL 145
;
; SPECIAL 144
;
; SPECIAL 143
;
; SPECIAL 142
;
; SPECIAL 141
;
; SPECIAL 140
;
; SPECIAL 139
;
; SPECIAL 138
;
; SPECIAL 137
;
; SPECIAL 136
;
; SPECIAL 135
;
; SPECIAL 134
;
; SPECIAL 133
;
; SPECIAL 132
;
; SPECIAL 131
;
; SPECIAL 130
;
; SPECIAL 129
;
; SPECIAL 128
;
; SPECIAL 127
;
; SPECIAL 126
;
; SPECIAL 125
;
; SPECIAL 124
;
; SPECIAL 123
;
; SPECIAL 122
;
; SPECIAL 121
;
; SPECIAL 120
;
; SPECIAL 119
;
; SPECIAL 118
;
; SPECIAL 117
;
; SPECIAL 116
;
; SPECIAL 115
;
; SPECIAL 114
;
; SPECIAL 113
;
; SPECIAL 112

;
; SPECIAL 111
;
; SPECIAL 110
;
; SPECIAL 109
;
; SPECIAL 108
;
; SPECIAL 107
;
; SPECIAL 106
;
; SPECIAL 105
;
; SPECIAL 104
;
; SPECIAL 103
;
; SPECIAL 102
;
; SPECIAL 101
;
; SPECIAL 100
;
; SPECIAL 99
;
; SPECIAL 98
;
; SPECIAL 97
;
; SPECIAL 96
;
; SPECIAL 95
;
; SPECIAL 94
;
; SPECIAL 93
;
; SPECIAL 92
;
; SPECIAL 91
;
; SPECIAL 90
;
; SPECIAL 89
;
; SPECIAL 88
;
; SPECIAL 87
;
; SPECIAL 86
;
; SPECIAL 85
;
; SPECIAL 84
;
; SPECIAL 83
;
; SPECIAL 82
;
; SPECIAL 81
;
; SPECIAL 80
;
; SPECIAL 79
;
; SPECIAL 78
;
; SPECIAL 77
;
; SPECIAL 76
;
; SPECIAL 75
;
; SPECIAL 74
;
; SPECIAL 73
;
; SPECIAL 72
;
; SPECIAL 71
;
; SPECIAL 70
;
; SPECIAL 69
;
; SPECIAL 68
;
; SPECIAL 67
;
; SPECIAL 66
;
; SPECIAL 65
;
; SPECIAL 64
;
; SPECIAL 63
;
; SPECIAL 62
;
; SPECIAL 61
;
; SPECIAL 60
;
; SPECIAL 59
;
; SPECIAL 58
;
; SPECIAL 57
;
; SPECIAL 56
;
; SPECIAL 55
;
; SPECIAL 54
;
; SPECIAL 53
;
; SPECIAL 52
;
; SPECIAL 51
;
; SPECIAL 50
;
; SPECIAL 49
;
; SPECIAL 48
;
; SPECIAL 47
;
; SPECIAL 46
;
; SPECIAL 45
;
; SPECIAL 44
;
; SPECIAL 43
;
; SPECIAL 42

```

; SPECIAL 41
; SPECIAL 40
; SPECIAL 39
; SPECIAL 38
; SPECIAL 37
; SPECIAL 36
; SPECIAL 35
; SPECIAL 34
; SPECIAL 33
; SPECIAL 32
; SPECIAL 31
; SPECIAL 30
; SPECIAL 29
; SPECIAL 28
; SPECIAL 27
; SPECIAL 26
; SPECIAL 25
; SPECIAL 24
; SPECIAL 23
; SPECIAL 22
; SPECIAL 21
; SPECIAL 20
; SPECIAL 19
; SPECIAL 18
;
;=====
; fixed vector section
;=====
        .org    0ffffdch
UDI:
        .lword  dummy_int
OVER_FLOW:
        .lword  dummy_int
BRKI:
        .lword  dummy_int
ADDRESS_MATCH:
        .lword  dummy_int
SINGLE_STEP:
        .lword  dummy_int
WDT:
        .lword  dummy_int
DBC:
        .lword  dummy_int
NMI:
        .lword  dummy_int
        .org    0ffffcH
RESET:
        .lword  start

;else : __R8C__
;
; for R8C/Tiny
;
;=====
;
; Arrangement of section
;
;=====
; Near RAM data area
;=====
; SBDATA area
        .section data_SE, DATA
        .org    400H
data_SE_top:

        .section bss_SE, DATA, ALIGN
bss_SE_top:

        .section data_S0, DATA

```

```
data_S0_top:
    .section bss_S0, DATA
bss_S0_top:

; near RAM area
    .section data_NE, DATA, ALIGN
data_NE_top:

    .section bss_NE, DATA, ALIGN
bss_NE_top:

    .section data_NO, DATA
data_NO_top:

    .section bss_NO, DATA
bss_NO_top:

;-----
; Stack area
;-----
    .section stack, DATA, ALIGN
    .blkb    STACKSIZE
stack_top:

    .blkb    ISTACKSIZE
istack_top:

;-----
; heap section
;-----
    .section heap, DATA
heap_top:
    .blkb    HEAPSIZE

;-----
; RAM program area
;-----
    .section program_ram, ALIGN
_program_ram_top:
    .glb     _program_ram_top

;-----
; Near ROM data area
;-----
    .section rom_NE, ROMDATA
    .org     0e000H
rom_NE_top:

    .section rom_NO, ROMDATA
rom_NO_top:
```

```

;-----
; Initial data of 'data' section
;-----
        .section data_SEI, ROMDATA, ALIGN
data_SEI_top:

        .section data_S0I, ROMDATA
data_S0I_top:

        .section data_NEI, ROMDATA, ALIGN
data_NEI_top:

        .section data_NOI, ROMDATA
data_NOI_top:

;-----
; Switch Table Section
;-----
        .section      switch_table, ROMDATA
switch_table_top:

;-----
; code area
;-----

        .section program, CODE, ALIGN

        .section interrupt, CODE, ALIGN

;-----
; variable vector section
;-----
        .section vector, ROMDATA      ; variable vector table
        .org      VECTOR_ADR
        .glb      _int_timerx

        .lword   dummy_int           ; vector 0
        .lword   dummy_int           ; vector 1
        .lword   dummy_int           ; vector 2
        .lword   dummy_int           ; vector 3
        .lword   dummy_int           ; vector 4
        .lword   dummy_int           ; vector 5
        .lword   dummy_int           ; vector 6
        .lword   dummy_int           ; vector 7
        .lword   dummy_int           ; vector 8
        .lword   dummy_int           ; vector 9
        .lword   dummy_int           ; vector 10
        .lword   dummy_int           ; vector 11
        .lword   dummy_int           ; vector 12
        .lword   dummy_int           ; vector 13
        .lword   dummy_int           ; vector 14
        .lword   dummy_int           ; vector 15
        .lword   dummy_int           ; vector 16
        .lword   dummy_int           ; vector 17
        .lword   dummy_int           ; vector 18
        .lword   dummy_int           ; vector 19
        .lword   dummy_int           ; vector 20
        .lword   dummy_int           ; vector 21
        .lword   _int_timerx         ; timerx(vector 22)
        .lword   dummy_int           ; vector 23
        .lword   dummy_int           ; vector 24
        .lword   dummy_int           ; vector 25
        .lword   dummy_int           ; vector 26
        .lword   dummy_int           ; vector 27
        .lword   dummy_int           ; vector 28

```



```

.lword dummy_int          ; vector 29
.lword dummy_int          ; vector 30
.lword dummy_int          ; vector 31
.lword dummy_int          ; vector 32
.lword dummy_int          ; vector 33
.lword dummy_int          ; vector 34
.lword dummy_int          ; vector 35
.lword dummy_int          ; vector 36
.lword dummy_int          ; vector 37
.lword dummy_int          ; vector 38
.lword dummy_int          ; vector 39
.lword dummy_int          ; vector 40
.lword dummy_int          ; vector 41
.lword dummy_int          ; vector 42
.lword dummy_int          ; vector 43
.lword dummy_int          ; vector 44
.lword dummy_int          ; vector 45
.lword dummy_int          ; vector 46
.lword dummy_int          ; vector 47
.lword dummy_int          ; vector 48
.lword dummy_int          ; vector 49
.lword dummy_int          ; vector 50
.lword dummy_int          ; vector 51
.lword dummy_int          ; vector 52
.lword dummy_int          ; vector 53
.lword dummy_int          ; vector 54
.lword dummy_int          ; vector 55
.lword dummy_int          ; vector 56
.lword dummy_int          ; vector 57
.lword dummy_int          ; vector 58
.lword dummy_int          ; vector 59
.lword dummy_int          ; vector 60
.lword dummy_int          ; vector 61
.lword dummy_int          ; vector 62
.lword dummy_int          ; vector 63

;=====
; fixed vector section
;-----
        .section fvector,ROMDATA          ; fixed vector table
;       .org      OffdcH
;UDI:
;       .lword   dummy_int
;OVER_FLOW:
;       .lword   dummy_int
;BRKI:
;       .lword   dummy_int
;ADDRESS_MATCH:
;       .lword   dummy_int
;SINGLE_STEP:
;       .lword   dummy_int
;WDT:
;       .lword   dummy_int
;DBC:
;       .lword   dummy_int
;NMI:
;       .lword   dummy_int
;       .org      OfffcH
RESET:
        .lword   start | 0FF00000H

;-----
; far ROM data area
;-----
;
;
;       .section rom_FE,ROMDATA

```

```
;      .org          10000H
;
;      .section rom_F0,ROMDATA
;
;      .section data_FEI,ROMDATA,ALIGN
;data_FEI_top:
;
;      .section data_F0I,ROMDATA
;data_F0I_top:
;
;*****
;
;      C Compiler for R8C/Tiny, M16C/60,30,20,10
; COPYRIGHT (C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;*****
```

5.1.8 main_m16c.c

```

/*****/
/* FILE NAME : main_m16c.c */
/* Ver : 1.00 */
/* CPU : M16C/26 */
/* FUNCTION : Data Flash rewrite Application Note sample program */
/* for EW0 mode operation */
/*-----*/
/* Copyright(C)2004, Renesas Technology Corp. */
/* Copyright(C)2004, Renesas Solutions Corp. */
/* All rights reserved. */
/*****/
// $Id: main_m16c.c,v 1.7 2004/08/18 04:37:15 ikari Exp $
#include <string.h>
#include <stdio.h>
// #include "sfr28.h"
#include "flashdevdrv.h"
#include "flashm16c.h"

#ifndef FALSE
#define FALSE 0
#endif
#ifndef TRUE
#define TRUE 1
#endif

typedef unsigned char BOOL;

#define BLOCK_SIZE 2048

#pragma INTERRUPT timerA1_int;
void timerA1_int(void);

/**/ Interrupt dummy (running on ram) /**/
#pragma INTERRUPT vec_dummy_int;
void far vec_dummy_int(void);

void ErrorDisp(const char * dt);

void mcu_init(void);

int CheckErasedBlank(void F_ADR_SIZE * f_addr, short size);
int CmpBlank(unsigned char F_ADR_SIZE * buf, short size);

/******/
/** Timer *****/
/******/
enum TimerSource{
    TIMER_DEV_1 = 0x00,
    TIMER_DEV_8 = 0x40,
    TIMER_DEV_32 = 0x80,
    TIMER_SUB_32 = 0xC0,
};
#define TIMER_CLOCK_Hz 20000000
#define TIMER_SUBCLOCK_Hz 32768
#define TA1_DEV32_MS(a) ((TIMER_CLOCK_Hz / 32000) * (a) - 1)
#define TA1_DEV8_MS(a) ((TIMER_CLOCK_Hz / 8000) * (a) - 1)
#define TA1_DEV1_MS(a) ((TIMER_CLOCK_Hz / 1000) * (a) - 1)

BOOL CheckTa1Passed(void);

void SetTickTimer(unsigned char tm);

void TimerA1InitTimerMode(enum TimerSource source,
                          unsigned short timer);

inline void StartTimerA1(void);

```

```

inline void StartTimerA1(void)
{
    ta1s = 1;          // TimerA1 start
}

void ClearTotalTimer(void);

void SystemTimerInc(void);
/** RAM for Timer **/
BOOL tm_ps = FALSE;
unsigned char tm_ms = 0;
unsigned long totalTimer = 0; // ms

inline unsigned long GetTotalTimer(void);
inline unsigned long GetTotalTimer(void)
{
    return totalTimer;
}
/*=====*/
/** Timer End ****/
/*=====*/
enum MainMode{
    ERASE_TEST_START,
    ERASE_TEST_RESTART,
    ERASE_TEST_CHECK,
    PROGRAM_TEST,
    OTHER,
} mode = ERASE_TEST_START;
const char TestData[33] = "0123456789ABCDEF0123456789ABCDEF";
/*****
// Main loop
*****/
void main(void)
{
    unsigned char buffer_addr[32];
    FlashResult err_code = F_SUCCESS;

    mcu_init();      /* initialize MCU */

    FlashInitialize(); /* FlashMemory Initialize */

    StartTimerA1();

    asm("fset i");

    /* Unlock gives a Flash block to write. (This example is unnecessary.) */
    UnlockBlockFlash(F_BLOCK_3);

    /* example : When a main loop is done in 20mS and made to work.

The turn of the movement
mode          Address and contents of a test
<< start >>
ERASE_TEST_START    0xF000-0xF7FF Start Erasing
ERASE_TEST_RESTART  0xF000-0xF7FF Restart Erasing
ERASE_TEST_CHECK    0xF000-0xF7FF Erasing confirmation
PROGRAM_TEST        0xF000-0xF01F Write and confirmation
OTHER               Wait until it passes from the erasing start for one second.
                   Judge it as the error, and reset a flash memory
                   when erasing isn't completed in one second.

    << Repetition >>
*/

    for(;;){

        /* Waiting 20ms passed */
        while(!CheckTa1Passed());

        switch(mode){
            case ERASE_TEST_START:

```

```

case ERASE_TEST_RESTART:
    /* Start/Restart erasing */
    /* So that it may clear a suspend requirement to do ResumErase()
       before resuming elimination. */
    if(mode == ERASE_TEST_START) {
        ClearTotalTimer(); /* Make totalTimer 0 for the erasing time acquisition. */
        err_code = StartEraseFlash((void F_ADR_SIZE *)0xF7FE);
    }
    else {
        ResumErase();
        err_code = RestartEraseFlash();
    }
    /* Check err code. */
    switch(err_code) {
    case F_SUSPEND: mode = ERASE_TEST_RESTART; break;
    case F_SUCCESS: mode = ERASE_TEST_CHECK; break;
    default: ErrorDisp("EraseER1"); break;
    }
    /* TimeOut Check */
    /* Take an error when F_SUSPEND occurs for one second after you start elimination. */
    if(GetTotalTimer() >= 1000) {
        if(mode == ERASE_TEST_RESTART) {
            ErrorDisp("Err Tout");
        }
    }
    break;
case ERASE_TEST_CHECK:
    /* Check whether even an erase block (0xF000-0xF7FFF) is being erased. */
    if(0 == CheckErasedBlank((void F_ADR_SIZE *)0xF000, BLOCK_SIZE)) {
        ErrorDisp("EraseER2");
    }
    mode = PROGRAM_TEST;
    break;
case PROGRAM_TEST:
    /* Writing 32byte and error code check */
    err_code = WriteFlash((void F_ADR_SIZE *)0xF000, TestData, 32);
    if(err_code != F_SUCCESS) ErrorDisp("WriteERR");
    /* The data being written check whether it begins to read it. */
    err_code = ReadFlash((void F_ADR_SIZE *)0xF000, buffer_addr, 32);
    if(err_code != F_SUCCESS) ErrorDisp("Read ERR");
    /* Compare data of write and data of read. */
    if(memcmp(buffer_addr, TestData, 32)) {
        ErrorDisp("Comp ERR");
    }
    mode = OTHER;
    break;
default:
    if(GetTotalTimer() >= 1000) {
        mode = ERASE_TEST_START;
    }
}
}

}

/*=====
 * Confirm whether a flash memory is in the blank.
 * f_addr : physical address on flash memory to confirm.
 * size : Number of bytes to confirm.
 * return : 1:blank
 *         : 0:not blank
 *=====*/
int CheckErasedBlank(void F_ADR_SIZE * f_addr, short size)
{
    unsigned char F_ADR_SIZE * faddr = f_addr;
    FlashResult err_code = F_SUCCESS;
    unsigned short r_buf[16];
    unsigned short r_size;
    for(; size > 0 ;){

```

```

    r_size = (size > sizeof(r_buf)) ? sizeof(r_buf) : size;
    err_code = ReadFlash(faddr, r_buf, r_size);
    if(CmpBlank((unsigned char F_ADR_SIZE *)r_buf, r_size)){
        return 0;
    }
    faddr += r_size;
    size -= r_size;
}
return 1;
}

/*=====
 * Compare the matter whether designated data are "BLANK_PATTERN".
 * f_addr : physical address on flash memory to confirm.
 * size : Number of bytes to confirm.
 * return : 1:blank
 *         : 0:not blank
 *=====*/
#define BLANK_PATTERN 0xff
int CmpBlank(unsigned char F_ADR_SIZE * buf, short size)
{
    while(size --){
        if(* buf ++ != BLANK_PATTERN) return -1;
    }
    return 0;
}

/*=====
 Error display and cancellation of a movement.
 *=====*/
void ErrorDisp(const char * dt)
{
    // DISPLAY(1, dt);
    while(1);
}

/*=====
 initialize MCU
 *=====*/
void mcu_init( void )
{
    /* Select full speed operation */
    /* Switch port initialization */
    pd10_5 = 0; // change switch ports to inputs
    pd10_6 = 0;
    pd10_7 = 0;

    /* LED initialization */
    pd7_0 = 1; // Change LED ports to outputs (connected to LEDs)
    pd7_1 = 1;
    pd7_2 = 1;

    /* unused pins - configure as outputs to decrease power consumption */
    pd6 = 0x90;

    pd8_0 = 1;
    pd8_1 = 1;
    pd8_2 = 1;
    pd8_3 = 1;

    prc2 = 1; // P9 is write protected - disable protection before writing to P9
    pd9_0 = 1;
    pd9_1 = 1;
    pd9_2 = 1;
    pd9_3 = 1;
    prc2 = 0; // Write protect P9

    pd10_0 = 1;
    pd10_1 = 1;

```

```

pd10_2 = 1;
pd10_3 = 1;
pd10_4 = 1;

// Set up a Timer A1
TimerA1InitTimerMode(TIMER_DEV_32, TA1_DEV32_MS(20));
SetTickTimer(20);

    ta1ic = 0x07;                // Set Timer-A1 Interrupt-Priority-Level

// Port Initialize
    p6 = 0x00;                // Port-6 clear
//    pd6 = 0xff;            // Port-6 is output port

    p7 = 0x07;                // p7_0 - p7_2 LED off
    pd7 = 0x07;                // p7_0 to p7_2 output select

    pu25 = 1;                // p10_4 to P10_7 pull-up
    p10 = 0x00;            // Port-10 clear
    pd10 = 0x1f;            // p10_5 to p10_7 is Key-in port
}

/*****
*/ Timer */
/*****
/*=====
Check tick time.
=====*/
BOOL CheckTa1Passed(void)
{
    if(tm_ps){
        tm_ps = FALSE;
        return TRUE;
    }
    return FALSE;
}
/*=====
Set up value of tick time.
=====*/
void SetTickTimer(unsigned char tm)
{
    tm_ms = tm;
}
/*=====
Initialize Timer A1.
=====*/
void TimerA1InitTimerMode(enum TimerSource source,
                          unsigned short timer)
{
// Set up a Timer A1
    ta1s = 0;                // Timer-A1 Stopped
    ta1ic = 0x07;            // Set Timer-A1 Interrupt-Priority-Level

    ta1mr = source;          // Set Timer-A1 mode register
                            // Mode=Timer-mode, Count-src=f1
    ta1 = timer;            // Set Timer-A1 timer-value (50ms)
}
/*=====
Clear totaltimer
=====*/
void ClearTotalTimer(void)
{
    totalTimer = 0;
}
/*****
The following is a program to work by the RAM. (EWO only)
*****/
#ifdef FLASH_MODE_EWO

```

```
#pragma SECTION program program_ram
#endif
/*=====
Timer A1 Interrupt function
=====*/
void timerA1_int(void)
{
    /* Suspend erasing and advance timer. */
    SuspendErase();
    SystemTimerInc();
}

/*=====
This is function for Timer A1 interrupt.
=====*/
void SystemTimerInc(void)
{
    // 1ms
    tm_ps = TRUE;
    totalTimer += tm_ms;
}

/*=====
dummy Interrupt
=====*/
void vec_dummy_int(void)
{
}
```


5.1.9 M16C_EW1.tmk

```
#####
# Makefile for TM V.3.20A
# COPYRIGHT (C) 1998(1998-2003)
# RENESAS TECHNOLOGY CORPORATION ALL RIGHTS RESERVED
# AND
# RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
#
# Notice : Don't edit.
# Date : 2004 06(June) 25(Friday) AM.11.28
# Project : M16C_EW1
#####

DELETE = @-del
LNLIST = $(PROJECT).cmd
FROM_ADDR = 0FB000
LMC = LMC30
CC = NC30
AR = LB30
UTL = utl30
AS = AS30
LIBFILE = $(PROJECT).lib
OUTDIR = M16CEW1
MKDIR = @-mkdir
ABSFILE = $(PROJECT).x30
ODINCMD = $(OUTDIR)
LN = LN30
TARGET = $(ABSFILE)
ECHO = @-echo
MKFILE = $(PROJECT).tmk
PROJECT = M16C_EW1
TYPE = @-type
LFLAGS = -MS -L nc30lib -G -LOC program_ram=$(FROM_ADDR) -O $(OUTDIR)$(TARGET)
UTLFLAGS =
CFLAGS = -c -dir $(OUTDIR) -g -gbool_to_char -OR -04 -OSA -finfo -fUD -fNA -fSA -WNP -WUP -WNC -Wall -WUV -WNUA
LMCFLAGS = -L
LIBFLAGS = -C
AFLAGS = -LM -D__HEAP__=1 -D__STANDARD_IO__=1 -D_from_addr=$(FROM_ADDR)h:__USTACKSIZE__=160h:__ISTACKSIZE__=160h
-finfo -O$(OUTDIR)
.SUFFIXES: .a30 .r30 .c .x30 .lib
.PHONY: all
all: ¥
    $(OUTDIR)¥$(TARGET)
.PHONY: clean
clean:
    $(DELETE) $(OUTDIR)¥$(TARGET)
    $(DELETE) $(ODINCMD)¥$(LNLIST)
    $(DELETE) $(OUTDIR)¥ncrt0_EW1.r30
    $(DELETE) $(OUTDIR)¥depend_m16c.r30
    $(DELETE) $(OUTDIR)¥flashdrvdev_ew1.r30
    $(DELETE) $(OUTDIR)¥main_m16c.r30
$(ODINCMD)¥$(LNLIST): ¥
    .¥$(MKFILE)
    $(ECHO)¥$(MRLFLAGS) $(LFLAGS) > $(ODINCMD)¥$(LNLIST)
    $(ECHO)¥$(OUTDIR)¥ncrt0_EW1.r30 >> $(ODINCMD)¥$(LNLIST)
    $(ECHO)¥$(OUTDIR)¥depend_m16c.r30 >> $(ODINCMD)¥$(LNLIST)
    $(ECHO)¥$(OUTDIR)¥flashdrvdev_ew1.r30 >> $(ODINCMD)¥$(LNLIST)
    $(ECHO)¥$(OUTDIR)¥main_m16c.r30 >> $(ODINCMD)¥$(LNLIST)
$(OUTDIR)¥$(TARGET): ¥
    $(ODINCMD)¥$(LNLIST) ¥
    $(OUTDIR)¥ncrt0_EW1.r30 ¥
    $(OUTDIR)¥depend_m16c.r30 ¥
    $(OUTDIR)¥flashdrvdev_ew1.r30 ¥
    $(OUTDIR)¥main_m16c.r30
    $(LN) @$$(ODINCMD)¥$(LNLIST)
$(OUTDIR)¥depend_m16c.r30: ¥
    .¥depend_m16c.c ¥
```

```
.¥flashdevdrv.h ¥
.¥flashdevconf.h ¥
.¥flashm16c.h
$(CC) $(MRCFLAGS) $(CFLAGS) depend_m16c.c
$(OUTDIR)¥flashdrvdev_ew1.r30: ¥
.¥flashdrvdev_ew1.c ¥
.¥flashdevdrv.h ¥
.¥flashdevconf.h ¥
.¥flashm16c.h
$(CC) $(MRCFLAGS) $(CFLAGS) flashdrvdev_ew1.c
$(OUTDIR)¥main_m16c.r30: ¥
.¥main_m16c.c ¥
.¥flashdevdrv.h ¥
.¥flashdevconf.h
$(CC) $(MRCFLAGS) $(CFLAGS) main_m16c.c
$(OUTDIR)¥ncrt0_EW1.r30: ¥
.¥ncrt0_EW1.a30 ¥
.¥sect30_EW1.inc
$(AS) $(MRAFLAGS) $(AFLAGS) ncrt0_EW1.a30
#####
# End of makefile for TM V.3.20A
# COPYRIGHT (C) 1998(1998-2003)
# RENESAS TECHNOLOGY CORPORATION ALL RIGHTS RESERVED
# AND
# RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
#####
```

6. 参考ドキュメント

ハードウェアマニュアル

M16C/26 グループハードウェアマニュアル Rev. 1.11

M16C/28 グループハードウェアマニュアル Rev. 2.00

(最新版をルネサスエレクトロニクスホームページから入手してください。)

7. ホームページとサポート窓口

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

ルネサス製品全般に関するお問い合わせ先
<http://japan.renesas.com/inquiry>
E-mail : csc@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.09.30	-	初版発行
1.01	2010.04.01	9	「図 3-5 自動消去フローチャート(参考)」一部変更
		14	「図 3-9 自動消去手順」一部変更
		23	「表 4-3 StartEraseFlash()」引数の意味 先頭アドレス → 最上位番地(ただし、偶数番地)
		69	7 行目 0xF000 → 0xF7FE

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

【注意】電源投入時は、製品の状態は不定です。

電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。

外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレス（予約領域）のアクセス禁止

【注意】リザーブアドレス（予約領域）のアクセスを禁止します。

アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

【注意】リセット時は、クロックが安定した後、リセットを解除してください。

プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

同じグループのマイコンでも型名が違っていると、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が異なる製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りが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、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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



ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。

総合お問合せ窓口：<http://japan.renesas.com/inquiry>