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。

M16C/62P グループ
フラッシュメモリ版 CPU 書き換えモード(EW0 モード)サンプル

1. 要約

この資料では、フラッシュメモリ版での CPU 書き換えモード(EW0 モード)の使用例を紹介します。

2. はじめに

この資料で説明する応用例は次のマイコンに適用されます。

・ マイコン：M16C/62P グループ

M16C/62P グループと同様の 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で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。

3. 使用例の説明

EW0 モードの特徴：

EW0 モードでは、CPU 書き換えプログラムを RAM 上に転送し、RAM 上の CPU 書き換えプログラムでプログラムコマンド、イレーズコマンドを発行することで、ユーザ ROM 領域とデータ領域を書き換えることができます。EW0 モードでは、プログラム、イレーズ中でも CPU は動作しているので、周辺機能割り込みはベクタと割り込みプログラムを RAM 上に配置することで、プログラム、イレーズ中に割り込みを受け付けることができます。

3.1 CPU 書き換えモード(EW0 モード)実行フロー

CPU 書き換えモード(EW0 モード)の実行フローを図 1 に示します。

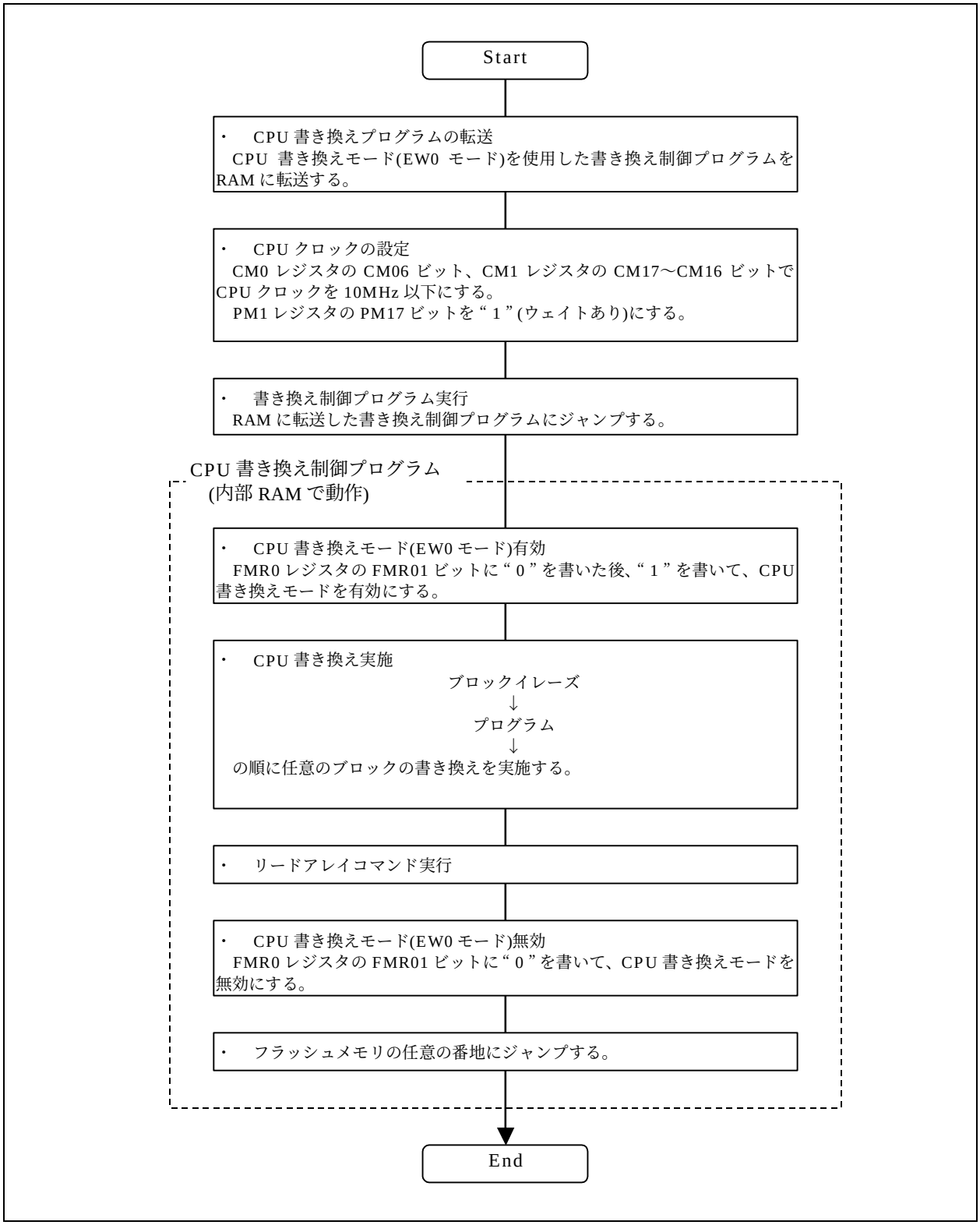
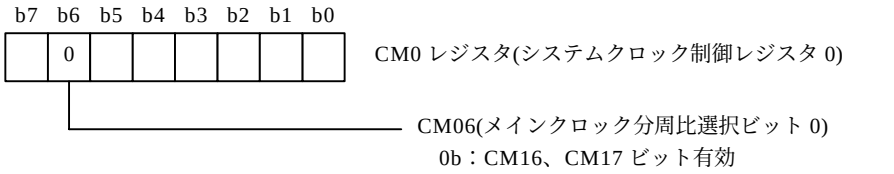


図 1. CPU 書き換えモード(EW0 モード)実行フロー

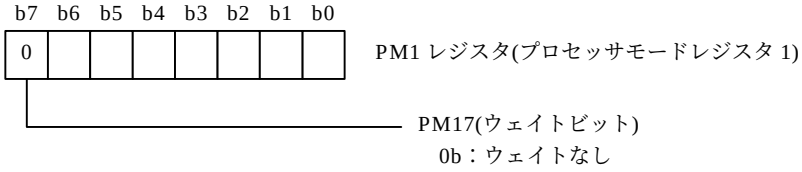
3.2 設定方法

3.2.1 CPU クロックの設定

(1) メインクロック分周比の設定



(2) ウェイトの設定



3.2.2 CPU 書き換え制御プログラムの RAM への転送

CPU 書き換え制御プログラムは RAM 上で動作させる必要があります。ここでは、0FA000h 番地に格納された CPU 書き換え制御プログラムを RAM 上に転送する例を説明します。

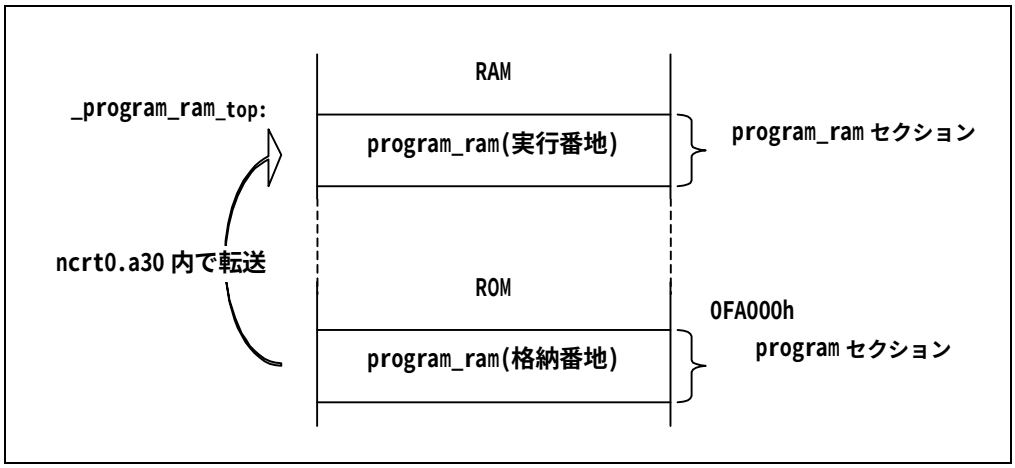


図 2. プログラム配置図

(1) セクション名を変更する。

セクション名として「program_ram」を追加し、このセクションに RAM 上で動作する CPU 書き換え制御プログラムを配置します。CPU 書き換え制御プログラムを、program セクションから program_ram セクションに配置しなおすには、下記の様に記述します。

```
void main(void)
{
      /* このプログラムは program セクション上に配置される */
}
```

```
/* #pragma SECTION  宣言以降のプログラムは program_ram セクション上に配置される */
#pragma SECTION program program_ram
void ew0_mode_program(void)
{
      /* このプログラムは program_ram セクション上に配置される */
}
```

(2) sect30.inc の変更

sect30.inc に program_ram セクションを追加します。ここでは、heap セクションの後ろに配置します。また、_program_ram_top ラベルは、プログラム転送時に使用します。

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

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

ここに追加

(3) CPU 書き換え制御プログラムの転送

スタートアップルーチン(ncrt0.a30)に CPU 書き換え制御プログラムを RAM に転送する処理を追加します。

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

```
;===================================================================
; Program Ram initialize
;  _from_addr is defined by as30 option "-D_from_addr=0fa000h"
;-------------------------------------------------------------------
    N_BCOPY _from_addr,_program_ram_top,program_ram
;
```

ここに追加

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

      .glb      _main
      jsr.a     _main
```

(4)　プログラム格納位置の指定

RAM 上に転送したプログラムを実行するには、プログラムの格納番地(ROM 上)と実行番地(RAM 上)を別々に配置するようにリンカ(ln30)で指定する必要があります。

```
ln30 –LOC program_ram=0FA000
```

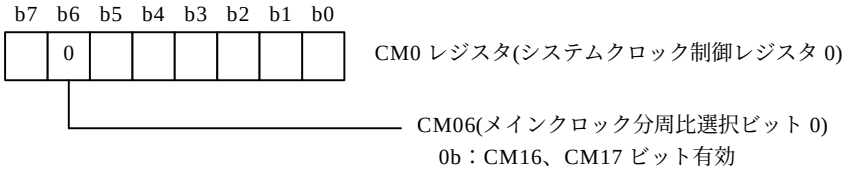
上記オプションでは、program_ram セクションを 0FA000h 番地から格納します。

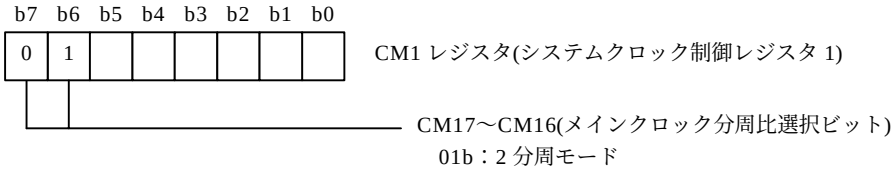
3.2.2 CPU 書き換え制御プログラム内の処理

(1) CPU クロックを 10MHz 以下に設定する。

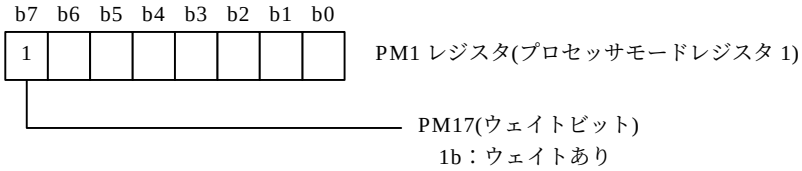
・ メインクロック分周比の設定

XIN=16MHz なので、CPU クロックを 10MHz 以下にするために、メインクロック分周比を 2 分周モードに設定する。

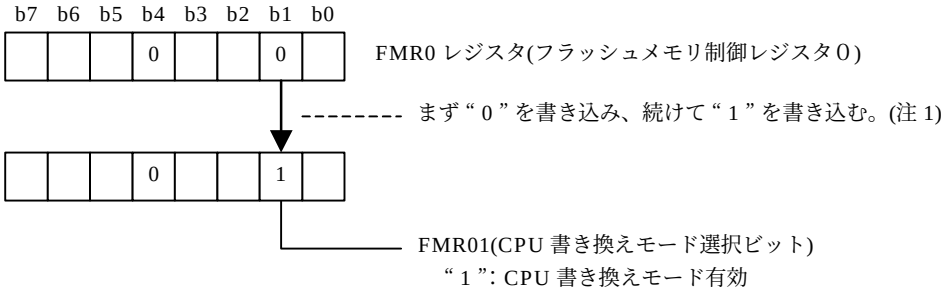




・ ウェイトの設定



(2) CPU 書き換えモード(EW0 モード)有効



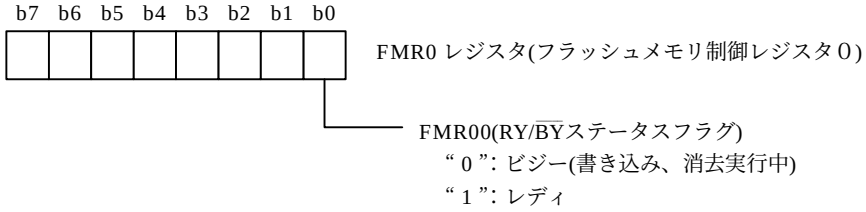
注 1. “ 0 ”を書いた後、“ 1 ”を書くまでに割り込み、DMA 転送が入らないようにしてください。
このビットは、$\overline{\text{NMI}}$端子が“ H ”の状態で書いてください。
また、EW0 モード時はフラッシュメモリ以外の領域で変更してください。
このビットはリードアレイモードにしてから“ 0 ”にしてください。

(3) ブロックイレーズ処理

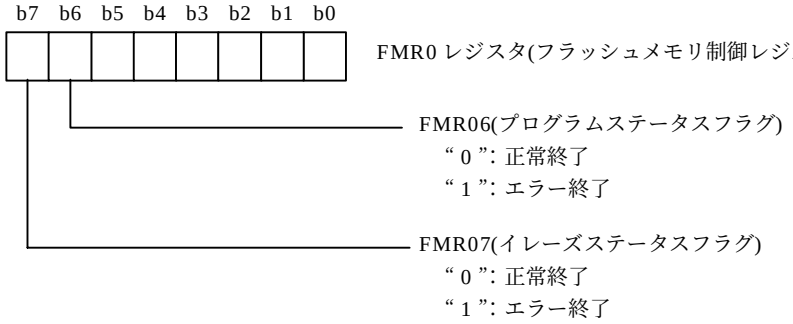
・ ブロックイレーズコマンド発行
ブロックイレーズするブロックの最上位アドレス(注 1)に“ 0020h ”を書き込んだ後、続けて“ 00d0h ”を書き込む。

注 1. ブロック 5 をブロックイレーズする場合は、F7FFEh 番地に書き込む。

・ ブロックイレーズ完了待ち
FMR0 レジスタの FMR00 ビットが“ 1 ”(レディ)になるまで待つ。



・ ステータスチェック
FMR0 レジスタの FMR06 ビット、FMR07 ビットをチェックし、イレーズエラーが発生していないかチェックする。
エラーが発生している場合は、イレーズコマンドを書き込んだアドレスに“ 0050h ”(クリアステータスコマンド)を書き込んだ後、CPU 書き換え処理を中止する。



(4) プログラム処理
該当するブロックの全領域に対し 1word ずつ、以下の手順でプログラムを行う。

・ プログラムコマンド発行
プログラムするアドレスに“ 0040h ”(プログラムコマンド)を書いた後、プログラムするデータを書き込む。

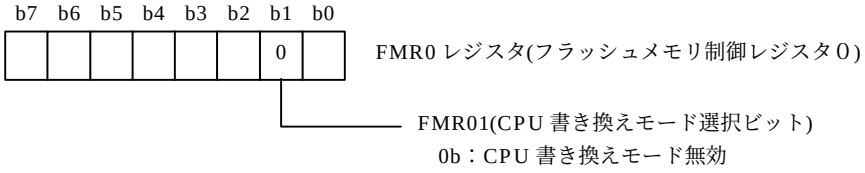
・ プログラム完了待ち
FMR0 レジスタの FMR00 ビットが“ 1 ”(レディ)になるまで待つ。

・ ステータスチェック
FMR0 レジスタの FMR06 ビット、FMR07 ビットをチェックし、プログラムエラーが発生していないかチェックする。
エラーが発生している場合は、プログラムコマンドを書き込んだアドレスに“ 0050h ”(クリアステータスコマンド)を書き込んだ後、CPU 書き換え処理を中止する。

(5) CPU 書き換えモード無効

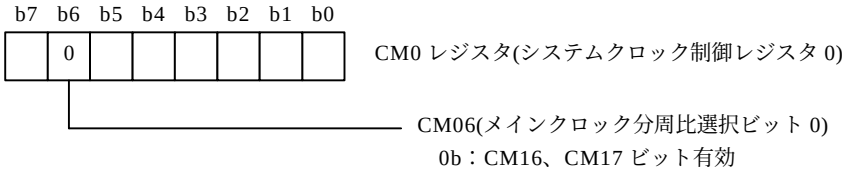
・ リードアレイコマンド発行
該当するブロックの最上位アドレスに“00FFh”(リードアレイコマンド)を書き込む。

・ CPU 書き換えモード無効設定

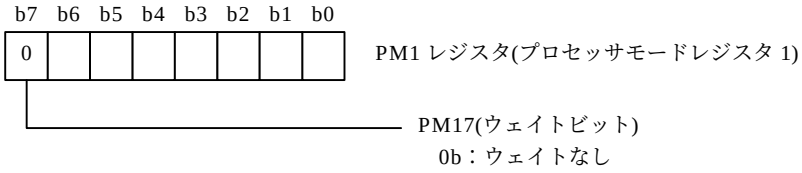


(6) CPU クロックを元に戻す。

・ メインクロック分周比の設定



・ ウェイトの設定



(7) フラッシュメモリ上のプログラムに戻る。

3.3 CPU 書き換えモード(EW0 モード)の注意事項

CPU 書き換えモード(EW0 モード)を実施する場合の注意事項を以下に示します。(最新の情報は、ハードウェアマニュアルで確認してください。)

(1) 動作速度

CPU 書き換えモード(EW0、EW1 モード)に入る前に、CM0 レジスタの CM06 ビット、CM1 レジスタの CM17～CM16 ビットで CPU クロックを 10MHz 以下にしてください。また、PM1 レジスタの PM17 ビットは“1”(ウェイトあり)にしてください。

(2) 使用禁止命令

EW0 モードでは、次の命令はフラッシュメモリ内部のデータを参照するため使用できません。
UND 命令、INTO 命令、JMPS 命令、ISRS 命令、BRK 命令

(3) 割り込み

・ 可変ベクタテーブルにベクタを持つ割り込みは、ベクタを RAM 領域に移すことで使用できます。
・ $\overline{\text{NMI}}$割り込み、ウォッチドッグタイマ割り込みは、割り込み発生時に強制的に FMR0 レジスタ、FMR1 レジスタが初期化されるので使用できます。固定ベクタテーブルに各割り込みルーチンの飛び先番地を設定してください。$\overline{\text{NMI}}$割り込み、ウォッチドッグタイマ割り込み発生時、書き換え動作終了します。割り込みルーチン終了後、書き換えプログラムを再実行してください。
・ アドレス一致割り込みはフラッシュメモリ内部のデータを参照するため使用できません。

(4) アクセス方法

FMR0 レジスタの FMR01 ビット、FMR02 ビット、FMR1 レジスタの FMR11 ビットを“1”にする場合、対象となるビットに“0”を書き込んだ後、続けて“1”を書いてください。なお、“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。また、$\overline{\text{NMI}}$端子に“H”を入力した状態で行ってください。

(5) ユーザ ROM 領域の書き換え

書き換え制御プログラムが格納されているブロックを書き換えている最中に電源電圧が低下すると、書き換え制御プログラムが正常に書き換えられないため、その後フラッシュメモリの書き換えができなくなる可能性があります。この場合、標準シリアル入出力モードまたはパラレル入出力モードを使用してください。

(6) コマンド、データの書き込み

コマンドコード、データは偶数番地に書いてください。

(7) ウェイトモード

ウェイトモードに移行する場合は、FMR0 レジスタの FMR01 ビットを“0”(CPU 書き換えモード無効)にした後、WAIT 命令を実行してください。

(8) ストップモード

ストップモードに移行する場合は、次のようにしてください。

・ FMR0 レジスタの FMR01 ビットを“0”(CPU 書き換えモード無効)にし、DMA 転送を禁止した後で、CM1 レジスタの CM10 ビットを“1”(ストップモード)にする。
・ CM10 ビットを“1”にする命令の次に JMP.B 命令を実行する。

```
        BSET    0, CM1                  ; ストップモードへ移行
        JMP.B   L1
L1:
        ・・・ストップモード復帰後の処理・・・
```

(9) 低消費電力モード、オンチップオシレータ低消費電力モード

CM0 レジスタの CM05 ビットが“1”(メインクロック停止)のときは、次のコマンドを実行しないでください。

- プログラム
- ブロックイレーズ
- イレーズ全アンロックブロック
- ロックビットプログラム
- リードロックビットステータス

4. 参考プログラム例

CPU 書き換えモード(EW0 モード)を使用して、INT0割り込み要求の発生をトリガに、内部 RAM(1800h 番地～2BFFh 番地)をブロック 5(F0000h 番地～F7FFFh 番地)にバックアップするプログラム例を示します。本プログラム例ではブロック 5 に対して、ブロックイレーズ、プログラム(RAM 領域の保存)を順に行います。

メモリマップ：

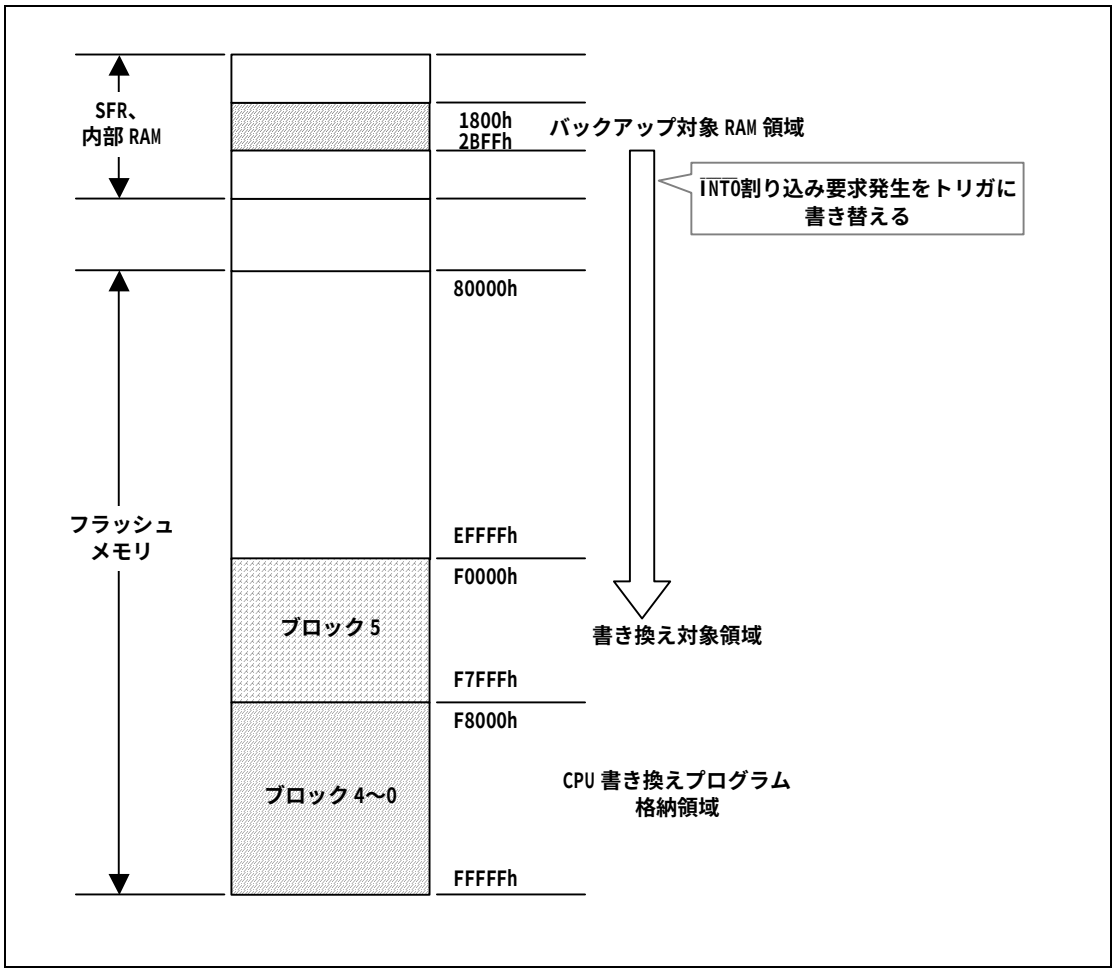


図 3. メモリマップ

動作条件：

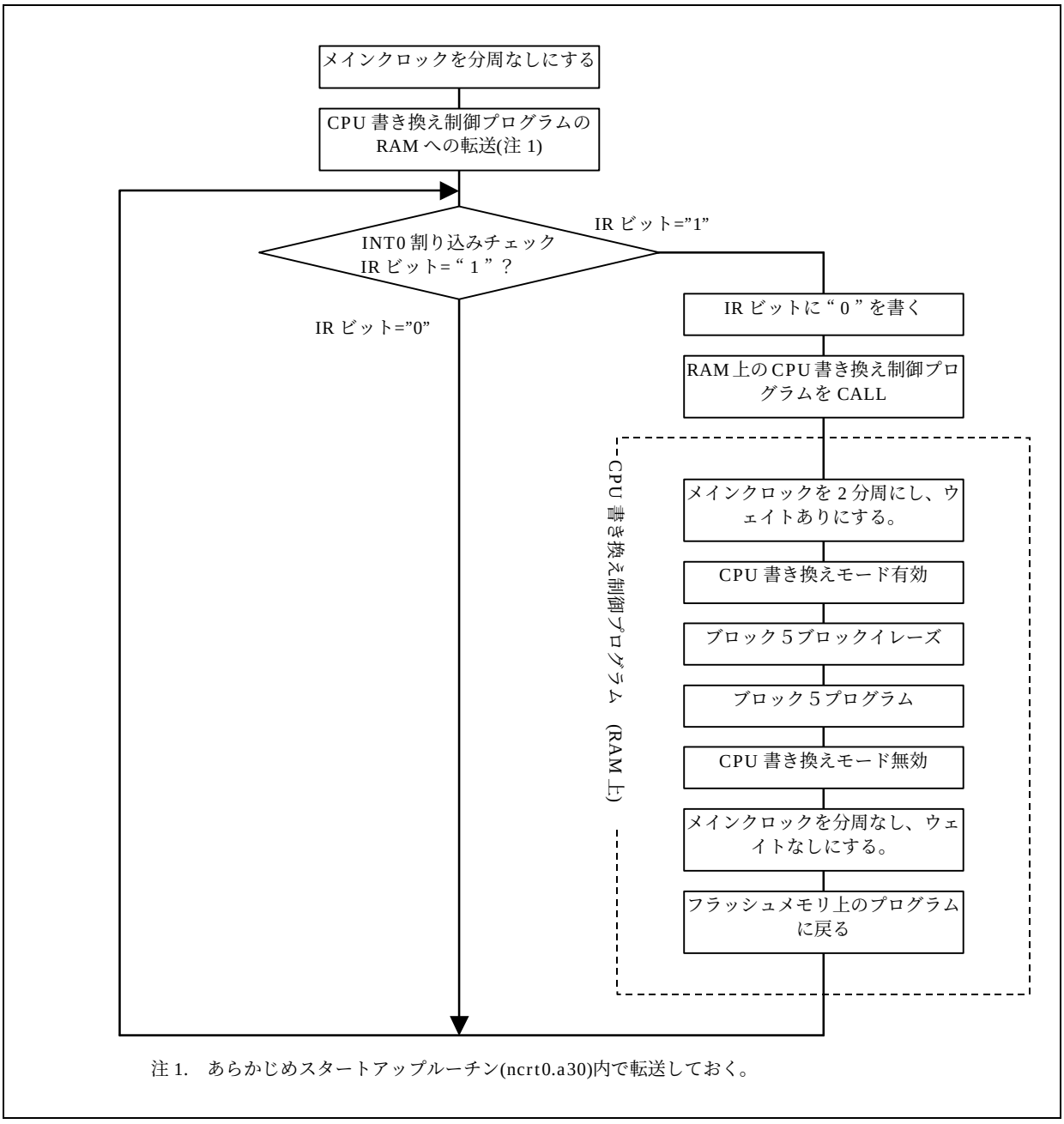
(1) VCC1=VCC2=5V

(2) XIN=16MHz（CPU 書き換えモード時は、CM0 レジスタの CM06 ビットと CM1 レジスタの CM17～CM16 ビットでメインクロック分周比を 2 分周モードに、PM1 レジスタの PM17 ビットをウェイトありにして 4MHz で動作させる）

メモリ使用容量：

RAM 上に転送される CPU 書き換えプログラムのメモリ使用容量は、最適化オプションなしでコンパイルした場合、312 バイト になります。

4.1 処理フロー



注 1. あらかじめスタートアップルーチン(ncrt0.a30)内で転送しておく。

図 4. 参考プログラム処理フロー

4.2 プログラムソース

(1) プログラム本体(rjj05b0641_src.c)

```
/************************************************************************/
/*                                                                      */
/*  M16C/62P Group Program Collection                                   */
/*                                                                      */
/*  FILE NAME : rjj05bxxxx_src.c                                        */
/*  CPU       : This program is the execution sample                    */
/*              in CPU rewriting mode.                                  */
/*  HISTORY   : 2004.11.01 Ver 1.00                                     */
/*                                                                      */
/*  Copyright (C) 2004. Renesas Technology Corp.                        */
/*  Copyright (C) 2004. Renesas Solutions Corp.                         */
/*  All right reserved.                                                 */
/*                                                                      */
/************************************************************************/
/*************************************/
/*    include file                   */
/*************************************/
#include "sfr62p.h"                    //  Special Function Register Header File

/*************************************/
/*    Function declaration           */
/*************************************/
int     ew0_mode_program(void);        // CPU rewrite control routine.
int     full_chk(void);                // full status check routine.

/*************************************/
/*    global variable declaration    */
/*************************************/
unsigned short volatile far *wp;       // Erase/Write address pointer
unsigned short volatile     *ramp;     // Ram save address.

/*************************************/
/*    symbol declaration             */
/*************************************/
#define     OK      0
#define     NG      1

/*************************************/
/*    main function                  */
/*************************************/
void main(void)
{

    // Set up a CPU operation clock.
    prcr = 0x03;                       // protect disabled.
    cm0  = 0x08;
    cm1  = 0x20;                       // main-clock no-divide mode.
                                       //  main-clock = 16MHz(XIN(16MHz))
    pm17 = 0;                          // Set "0" to wait bit.
    prcr = 0;                          // Protect enabled.

    int0ic = 0;                        // Set INT0 interrupt proprity level

    asm("    fclr I");
```

```
    while(1)
    {
                                        // Waiting for an INT0 interruption demand.
        if(ir_int0ic == 1)
        {
            int0ic      = 0;            // An INT0 interruption demand is cleared.

            if (ew0_mode_program()==NG) // CPU rewriting program execution.
            {                           // CPU rewrite error!
                p0  = 0x55;
                pd0 = 0xff;             // error display.
                while(1);
            }
        }
    }

}

#pragma SECTION program program_ram
/*************************************/
/*    CPU rewrite(EW0 mode) program   */
/*************************************/
int ew0_mode_program(void)
{
    // Set up a CPU operation clock.
    prcr = 0x03;                        // protect disabled.
    cm0  = 0x08;
    cm1  = 0x60;                        // main-clock divide-by-2 mode.
                                        //  main-clock = 8MHz(XIN(16MHz))
    pm17 = 1;                           // Set "1" to wait bit.
    prcr = 0;                           // Protect enabled.

    fmr01 = 0;                          // CPU rewrite mode enabled.
    fmr01 = 1;

                                        // Block-5 erased.
    wp = (unsigned short *)0xf7ffe;
    *wp = 0x20;                         // Erase Command(1st bus cycle) write
    *wp = 0xd0;                         // Erase Command(2nd bus cycle) write
    while (fmr00 == 0);                 // Wait for FMR00(RY/BY status) bit on
    if (full_chk() != OK) {             // Full-status check
        fmr01 = 0;
        return(NG);
    }

                                        // Block-5 programed.
    wp = (unsigned short *)0xf0000;
    for (ramp=(unsigned short *)0x01800; ramp<(unsigned short *)0x02c00; ) {
        *wp = 0x40;                     // Program Command(1st bus cycle) weite
        *wp = *ramp;                    // Program Command(2nd bus cycle(DATA)) write.
        while (fmr00 == 0);             // Wait for FMR00(RY/BY status) bit on
        if (full_chk() != OK) {         // Full-status check
            fmr01 = 0;
            return(NG);
        }
        wp++;                           // Program address count up.
        ramp++;                         // Ram address count up.
    }
```

```
    wp = (unsigned short *)0xf0000;
    *wp = 0xff;                         // Read-Array Command Write.
    fmr01 = 0;                          // CPU rewrite mode disabled.

    // Set up a CPU operation clock.
    prcr = 0x03;                        // protect disabled.
    cm0  = 0x08;
    cm1  = 0x20;                        // main-clock no-divide mode.
                                        //  main-clock = 16MHz(XIN(16MHz))
    pm17 = 0;                           // Set "0" to wait bit.
    prcr = 0;                           // Protect enabled.

    return(OK);
}

/*************************************/
/*  Full-status check routine        */
/*                                   */
/*      return value : 0=normal      */
/*                     1=error       */
/*************************************/
int full_chk(void)
{
    if ((fmr0 & 0xc0) == 0xC0) {
                                        // Command sequence error
        *wp = 0x50;                     // Clear status register.
        return(NG);                     // Error return.

    } else if ((fmr0 & 0x80) != 0) {
                                        // Erase error
        *wp = 0x50;                     // Clear status register.
        return(NG);                     // Error return.

    } else if ((fmr0 & 0x40) != 0) {
                                        // Program error
        *wp = 0x50;                     // Clear status register.
        return(NG);                     // Error return.

    }

    return(OK);                         // Normal return.

}
```

(2) スタートアップファイル(ncrt0.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
;
;
; ncrt0.a30 : NC30 startup program
;
; This program is applicable when using the basic I/O library
;
; $Id: ncrt0.a30,v 1.21 2003/07/09 02:28:54 muranaka Exp $
;
;***********************************************************************

;------------------------------------------------------------------
; HEEP SIZE definition
;------------------------------------------------------------------
.if    __HEAP__ == 1
HEAPSIZE        .equ    0H
.else

.if    __HEAPSIZE__ == 0
HEAPSIZE        .equ    300H
.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
.else
VECTOR_ADR      .equ    0fedch
.endif

;------------------------------------------------------------------
; Section allocation
```

```
;-----------------------------------------------------------------------
        .list OFF
        .include sect30.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    0BH
```

```
;---------------------------------------------------------------
; special page definition
;---------------------------------------------------------------
;       macro define for special page
;
;Format:
;       SPECIAL   number
;

SPECIAL.macro   NUM
        .org    0FFFFEH-(NUM*2)
        .glb    __SPECIAL_@NUM
        .word   __SPECIAL_@NUM  & 0FFFFH
.endm
;===============================================================
; 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 processer 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_SO_top,bss_SO
        N_BZERO  bss_NE_top,bss_NE
        N_BZERO  bss_NO_top,bss_NO

        mov.b   #00H, R0L
        mov.w   #0400h, A1
        mov.w   #2800h, R3
        sstr.b

;---------------------------------------------------------------
; 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_FO_top,bss_FO

;-----------------------------------------------------------------------
; Copy edata_E(O) section from edata_EI(OI) section
;-----------------------------------------------------------------------
        BCOPY   data_FEI_top,data_FE_top,data_FE
        BCOPY   data_FOI_top,data_FO_top,data_FO

        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>>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

;=======================================================================
; Program Ram initialize
;  _from_addr is defined by as30 option "-D_from_addr=0fa000h"
;-----------------------------------------------------------------------
    N_BCOPY _from_addr,_program_ram_top,program_ram
;

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

        .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 processer 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_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_FO_top,bss_FO

;-----------------------------------------------------------------------
; Copy edata_E(O) section from edata_EI(OI) section
;-----------------------------------------------------------------------
;       BCOPY    data_FEI_top,data_FE_top,data_FE
;       BCOPY    data_FOI_top,data_FO_top,data_FO

        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

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

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

        .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
;
;*****************************************************************************
```

(3) セクション定義ファイル(sect30.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.inc,v 1.20 2003/08/19 01:43:59 muranaka 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_SO,DATA
data_SO_top:

        .section bss_SO,DATA
bss_SO_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,ALIGN
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_FO,DATA
data_FO_top:

        .section bss_FO,DATA
bss_FO_top:


;------------------------------------------------------------
; Far ROM data area
;------------------------------------------------------------
        .section rom_FE,ROMDATA
        .org             0F8000H
rom_FE_top:

        .section rom_FO,ROMDATA
rom_FO_top:

;------------------------------------------------------------
; Initial data of 'data' section
;------------------------------------------------------------
        .section data_SEI,ROMDATA
```

```
data_SEI_top:

        .section data_SOI,ROMDATA
data_SOI_top:

        .section data_NEI,ROMDATA
data_NEI_top:

        .section data_NOI,ROMDATA
data_NOI_top:

        .section data_FEI,ROMDATA
data_FEI_top:

        .section data_FOI,ROMDATA
data_FOI_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          ; INT0 (for user) (vector 29)
        .lword   dummy_int          ; INT1 (for user) (vector 30)
        .lword   dummy_int          ; INT2 (for user) (vector 31)
.else
        .lword   dummy_int          ; BRK    (vector 0)
        .lword   dummy_int          ;        (vector 1)
        .lword   dummy_int          ;        (vector 2)
        .lword   dummy_int          ;        (vector 3)
        .lword   dummy_int          ; int3(for user)(vector 4)
        .lword   dummy_int          ; timerB5(for user)(vector 5)
        .lword   dummy_int          ; timerB4(for user)(vector 6)
        .lword   dummy_int          ; timerB3(for user)(vector 7)
        .lword   dummy_int          ; si/o4 /int5(for user)(vector 8)
        .lword   dummy_int          ; si/o3 /int4(for user)(vector 9)
        .lword   dummy_int          ; Bus collision detection(for user)(v10)
        .lword   dummy_int          ; DMA0(for user)(vector 11)
        .lword   dummy_int          ; DMA1(for user)(vector 12)
        .lword   dummy_int          ; Key input interrupt(for user)(vect 13)
        .lword   dummy_int          ; A-D(for user)(vector 14)
        .lword   dummy_int          ; uart2 transmit(for user)(vector 15)
        .lword   dummy_int          ; uart2 receive(for user)(vector 16)
        .lword   dummy_int          ; uart0 transmit(for user)(vector 17)
        .lword   dummy_int          ; uart0 receive(for user)(vector 18)
        .lword   dummy_int          ; uart1 transmit(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          ; int0 (for user)(vector 29)
        .lword   dummy_int          ; int1 (for user)(vector 30)
        .lword   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 defination
;------------------------------------------------------
;       macro is defined in ncrt0.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     0FFFDCH
;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_SO,DATA
data_SO_top:

        .section bss_SO,DATA
bss_SO_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

;------------------------------------------------------------
; 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_SOI,ROMDATA
data_SOI_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

        .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   dummy_int          ; 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     0FFDCH
;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     0fffcH
RESET:
        .lword   start

.endif ; __R8C

;------------------------------------------------------------
; far ROM data area
;------------------------------------------------------------
;
;       .section rom_FE,ROMDATA
;       .org              10000H
;
;       .section rom_FO,ROMDATA
;
;       .section data_FEI,ROMDATA,ALIGN
;data_FEI_top:
;
;       .section data_FOI,ROMDATA
;data_FOI_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. 参考ドキュメント

ハードウエアマニュアル

M16C/62P グループ（M16C/62P、M16C/62PT）ハードウエアマニュアル
（最新版をルネサス テクノロジホームページから入手してください。）

6. 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://www.renesas.com/jpn/

M16C ファミリ MCU 技術サポート窓口

E-mail:support_apl@renesas.com

改訂記録

Rev.	発行日		改訂内容
		ページ	ポイント
1.00	2004.04.01	-	初版発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。