AN204424

FM3 RAM コードの生成方法

本アプリケーションノートは、Cypress 製マイコン FM3 ファミリを用いて、製品へ RAM で実行するコード (RAM コード) を組み込むことを検討されている方を対象としています。

Contents

1 はじめに

本アプリケーションノートは、Cypress 製マイコン FM3 ファミリを用いて、製品へ RAM で実行するコード (RAM コード) を組み込むことを検討されている方を対象としています。

FM3 において RAM コードを製品へ組み込む用途としては、シングルオペレーション Flash の消去や書込みを実行するプログラムがあります。本プログラムは消去および書込みを Flash で実行できないために RAM コードにします。RAM コードの用途と特長および注意を表 1 に示します。

表 1. RAM コードの用途と特長および注意

用途	特長	注意
シングルオペレーション Flash の消去，書込み用	プログラムにより Flash の書換えが可能	Flash へアクセス中は割込みを禁止すること。また、RAM コードでのみ実行し、Flash では実行しないように実装すること。
		消去，書込みを実行する Flash のセクタにはコード，データなどを一切配置しないこと。
		RAM コードとデータの合計サイズが RAM のサイズを超えないこと。

本アプリケーションノートは RAM コードを生成する方法について説明します。

2 対象製品

本アプリケーションノートに記載されている内容の対象製品は、下記のとおりです。

(TYPE0)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B100A	MB9BF102NA,MB9BF104NA, MB9BF105NA, MB9BF106NA, MB9BF102RA, MB9BF104RA, MB9BF105RA, MB9BF106RA
MB9B300A	MB9BF304NA, MB9BF305NA, MB9BF306NA, MB9BF304RA, MB9BF305RA, MB9BF306RA
MB9B300B	MB9BF304NB, MB9BF305NB, MB9BF306NB, MB9BF304RB, MB9BF305RB, MB9BF306RB
MB9B400A	MB9BF404NA, MB9BF405NA, MB9BF406NA, MB9BF404RA, MB9BF405RA, MB9BF406RA
MB9B500A	MB9BF504NA, MB9BF505NA, MB9BF506NA, MB9BF504RA, MB9BF505RA, MB9BF506RA
MB9B500B	MB9BF504NB, MB9BF505NB, MB9BF506NB, MB9BF504RB, MB9BF505RB, MB9BF506RB

(TYPE1)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A110	MB9AF111L, MB9AF112L, MB9AF114L, MB9AF111M, MB9AF112M, MB9AF114M, MB9AF115M, MB9AF116M, MB9AF111N, MB9AF112N, MB9AF114N, MB9AF115N, MB9AF116N
MB9A110A	MB9AF111LA, MB9AF112LA, MB9AF114LA, MB9AF111MA, MB9AF112MA, MB9AF114MA, MB9AF115MA, MB9AF116MA, MB9AF111NA, MB9AF112NA, MB9AF114NA, MB9AF115NA, MB9AF116NA
MB9A310	MB9AF311L, MB9AF312L, MB9AF314L, MB9AF311M, MB9AF312M, MB9AF314M, MB9AF315M, MB9AF316M, MB9AF311N, MB9AF312N, MB9AF314N, MB9AF315N, MB9AF316N
MB9A310A	MB9AF311LA, MB9AF312LA, MB9AF314LA, MB9AF311MA, MB9AF312MA, MB9AF314MA, MB9AF315MA, MB9AF316MA, MB9AF311NA, MB9AF312NA, MB9AF314NA, MB9AF315NA, MB9AF316NA

(TYPE2)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B110T	MB9BF116S, MB9BF117S, MB9BF118S, MB9BF116T, MB9BF117T, MB9BF118T
MB9B210T	MB9BF216S, MB9BF217S, MB9BF218S, MB9BF216T, MB9BF217T, MB9BF218T
MB9B310T	MB9BF316S, MB9BF317S, MB9BF318S, MB9BF316T, MB9BF317T, MB9BF318T
MB9B410T	MB9BF416S, MB9BF417S, MB9BF418S, MB9BF416T, MB9BF417T, MB9BF418T
MB9B510T	MB9BF516S, MB9BF517S, MB9BF518S, MB9BF516T, MB9BF517T, MB9BF518T
MB9B610T	MB9BF616S, MB9BF617S, MB9BF618S, MB9BF616T, MB9BF617T, MB9BF618T
MB9BD10T	MB9BFD16S, MB9BFD17S, MB9BFD18S, MB9BFD16T, MB9BFD17T, MB9BFD18T

(TYPE3)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A130L	MB9AF131K, MB9AF132K, MB9AF131L, MB9AF132L
MB9A130LA	MB9AF131KA, MB9AF132KA, MB9AF131LA, MB9AF132LA

(TYPE4)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B110R	MB9BF112N, MB9BF114N, MB9BF115N, MB9BF116N, MB9BF112R, MB9BF114R, MB9BF115R, MB9BF116R
MB9B310R	MB9BF312N, MB9BF314N, MB9BF315N, MB9BF316N, MB9BF312R, MB9BF314R, MB9BF315R, MB9BF316R
MB9B410R	MB9BF412N, MB9BF414N, MB9BF415N, MB9BF416N, MB9BF412R, MB9BF414R, MB9BF415R, MB9BF416R
MB9B510R	MB9BF512N, MB9BF514N, MB9BF515N, MB9BF516N, MB9BF512R, MB9BF514R, MB9BF515R, MB9BF516R

(TYPE5)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A110K	MB9AF111K, MB9AF112K
MB9A310K	MB9AF311K, MB9AF312K

(TYPE6)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A140N	MB9AF141L, MB9AF142L, MB9AF144L, MB9AF141M, MB9AF142M, MB9AF144M, MB9AF141N, MB9AF142N, MB9AF144N
MB9A140NA	MB9AF141LA, MB9AF142LA, MB9AF144LA, MB9AF141MA, MB9AF142MA, MB9AF144MA, MB9AF141NA, MB9AF142NA, MB9AF144NA
MB9A340N	MB9AF341L, MB9AF342L, MB9AF344L, MB9AF341M, MB9AF342M, MB9AF344M, MB9AF341N, MB9AF342N, MB9AF344N
MB9A340NA	MB9AF341LA, MB9AF342LA, MB9AF344LA, MB9AF341MA, MB9AF342M, MB9AF344MA, MB9AF341NA, MB9AF342N, MB9AF344NA
MB9AA40N	MB9AFA41L, MB9AFA42L, MB9AFA44L, MB9AFA41M, MB9AFA42M, MB9AFA44M, MB9AFA41N, MB9AFA42N, MB9AFA44N
MB9AA40NA	MB9AFA41LA, MB9AFA42LA, MB9AFA44LA, MB9AFA41MA, MB9AFA42MA, MB9AFA44MA, MB9AFA41NA, MB9AFA42NA, MB9AFA44NA
MB9AB40N	MB9AFB41L, MB9AFB42L, MB9AFB44L, MB9AFB41M, MB9AFB42M, MB9AFB44M, MB9AFB41N, MB9AFB42N, MB9AFB44N
MB9AB40NA	MB9AFB41LA, MB9AFB42LA, MB9AFB44LA, MB9AFB41MA, MB9AFB42MA, MB9AFB44MA, MB9AFB41NA, MB9AFB42NA, MB9AFB44NA

(TYPE7)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A130N	MB9AF131M, MB9AF132M, MB9AF131N, MB9AF132N
MB9AA30N	MB9AFA31L, MB9AFA32L, MB9AFA31M, MB9AFA32M, MB9AFA31N, MB9AFA32N

(TYPE8)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A150R	MB9AF154M, MB9AF155M, MB9AF156M, MB9AF154N, MB9AF155N, MB9AF156N, MB9AF154R, MB9AF155R, MB9AF156R

(TYPE9)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B120M	MB9BF121K, MB9BF122K, MB9BF124K, MB9BF121L, MB9BF122L, MB9BF124L, MB9BF121M, MB9BF122M, MB9BF124M
MB9B320M	MB9BF321K, MB9BF322K, MB9BF324K, MB9BF321L, MB9BF322L, MB9BF324L, MB9BF321M, MB9BF322M, MB9BF324M
MB9B520M	MB9BF521K, MB9BF522K, MB9BF524K, MB9BF521L, MB9BF522L, MB9BF524L, MB9BF521M, MB9BF522M, MB9BF524M

3 RAM コードについて

3.1 概要

本アプリケーションノートにおいては、コードを ROM から RAM へコピーして配置し、RAM コードを生成する方法を説明します。

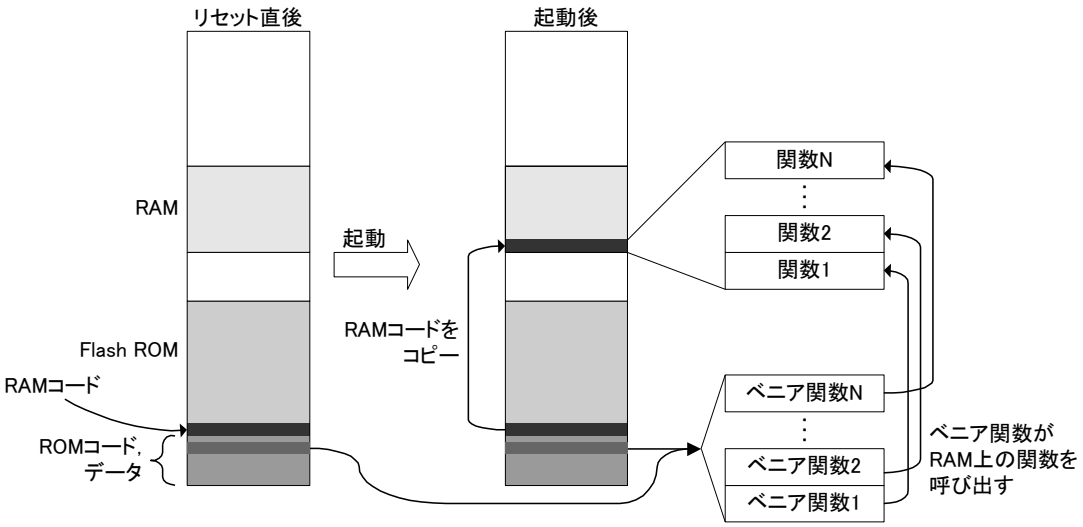
本方法は、製品へ組み込む場合などに使用します。

なお、デバッグ用途として、コードをデバッガにより直接 RAM へ配置する方法もありますが、製品への組込み用途を対象とするため、本アプリケーションノートでは説明しません。

3.2 RAM コードの仕組み

RAM コードの仕組みを図 1 に示します。

図 1. RAM コードの仕組み



リセット直後は Flash ROM にすべてコードがあり、起動時に RAM コードを Flash ROM から RAM へコピーします。その後、RAM コードの関数がベニア関数を介して呼び出されます。

3.3 ベニア関数とは

ベニア関数とはある関数への分岐が規定の範囲を超えるような場合にリンカにより ROM に生成される関数です (詳細は「6 RAM コードの実行」を参照)。

ベニア関数については、開発環境が IAR Embedded Workbench for ARM(EWARM) の場合は「IAR C/C++ 開発ガイド」[1] における「パート 1.ビルドツール」の「アプリケーションのリンク」-「リンクについて」-「ベニア」を参照してください。また、KEIL MDK-ARM の場合は「Linker User Guide」[2] における「Image structure and generation」の「Overview of veneers」を参照してください。

4 動作確認環境

本アプリケーションノートに記載した内容の動作確認環境を表 2 に示します。

表 2. 動作確認環境

項目	内容
使用マイコン	MB9BF506R
動作周波数	コア：80MHz 周辺：40MHz
動作電圧	+3.3V
OS	使用しない
統合開発環境	IAR IAR Embedded Workbench for ARM 6.30.4
	KEIL MDK-Lite Version 4.22a (※)
コンパイル最適化	なし (ありに設定しても動作可能)

(※) MDK-Lite は MDK-ARM のエディション名

5 RAM コードの生成

本章では IAR EWARM および KEIL MDK-ARM のそれぞれの開発環境で RAM コードを生成する方法を説明します。

5.1 サンプルソフトウェアのプロジェクト

本アプリケーションノートに添付するサンプルソフトウェアのプロジェクトは IAR EWARM が 1 通り、KEIL MDK-ARM が 2 通りです。なお、5.2 章および 5.3 章において説明する内容は、各プロジェクトにおいて設定されています。

サンプルソフトウェアの各プロジェクトの設定の参照を表 3 に示します。

表 3. サンプルソフトウェアの各プロジェクトの設定の参照

開発環境	プロジェクト名	RAM コードの生成方法	参照
IAR EWARM	Ramcode	関数毎の RAM コード生成	5.2
KEIL MDK-ARM	Ramcode_source	ファイル毎の RAM コード生成	5.3.1
	Ramcode_section	関数毎の RAM コード生成	5.3.2

5.2 IAR EWARM における RAM コードの生成方法

IAR EWARM において、RAM コードの生成は関数毎に行います。

5.2.1 ソースコードへの記述

RAM コードとして生成する関数に対して、図 2 に示すように関数の定義に「__ramfunc」の記述を追加します。本記述は RAM コードとするすべての関数に対して行う必要があります。

図 2. RAM コードとして使用する関数の記述

```
__ramfunc
en_result_t Ramcode_FlashErase(  uint32_t   u32SectorEraseAddress,
                                 boolean_t bWaitUntilFinished )
{
    ...
}
```

IAR EWARM の場合は本設定のみで、開発環境への設定は必要ありません。

5.3 KEIL MDK-ARM における RAM コードの生成方法

KEIL MDK-ARM の場合には、表 3 に示したとおり 2 通りの RAM コードの生成方法があります。

ファイル毎に RAM コードを生成する場合は、指定した C ソースファイルの関数が全て RAM コードとして設定されます(「5.3.1 ファイルごとに RAM コードを設定する場合の設定」を参照)。

関数毎に RAM コードを生成する場合は、関数にセクションを指定することにより、RAM コードとして設定します。セクションを関数単位に指定するため、同じ C ソースファイル内の特定の関数を RAM コードに設定できます(「5.3.2 関数ごとに RAM コードを生成する場合の設定」を参照)。

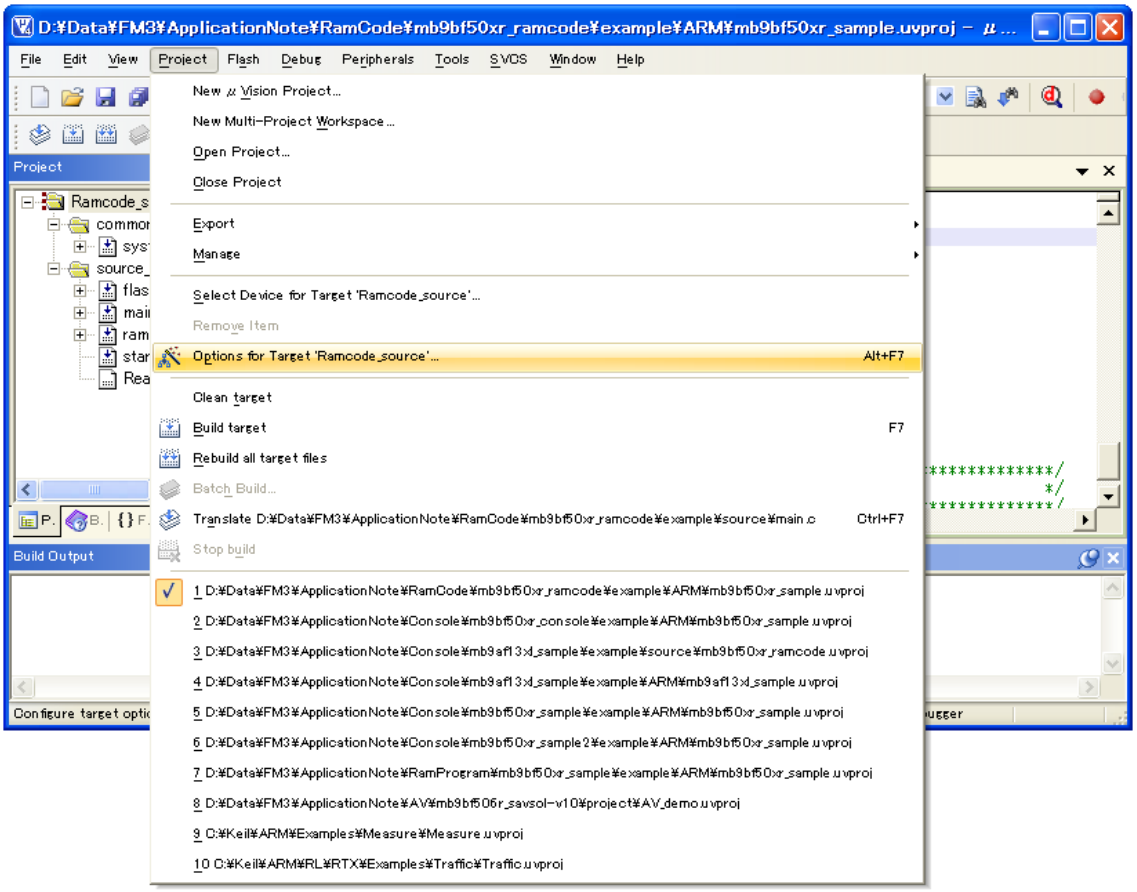
5.3.1 ファイルごとに RAM コードを設定する場合の設定

ファイルごとに RAM コードを設定する場合は、MDK-ARM の環境設定のみを行います。

5.3.1.1 MDK-ARM の環境設定 (1)

MDK-ARM を起動しプロジェクトファイルを開いたら、[Project] メニューの [Option for Target 'Ramcode_source'...] を選択します (図 3)。

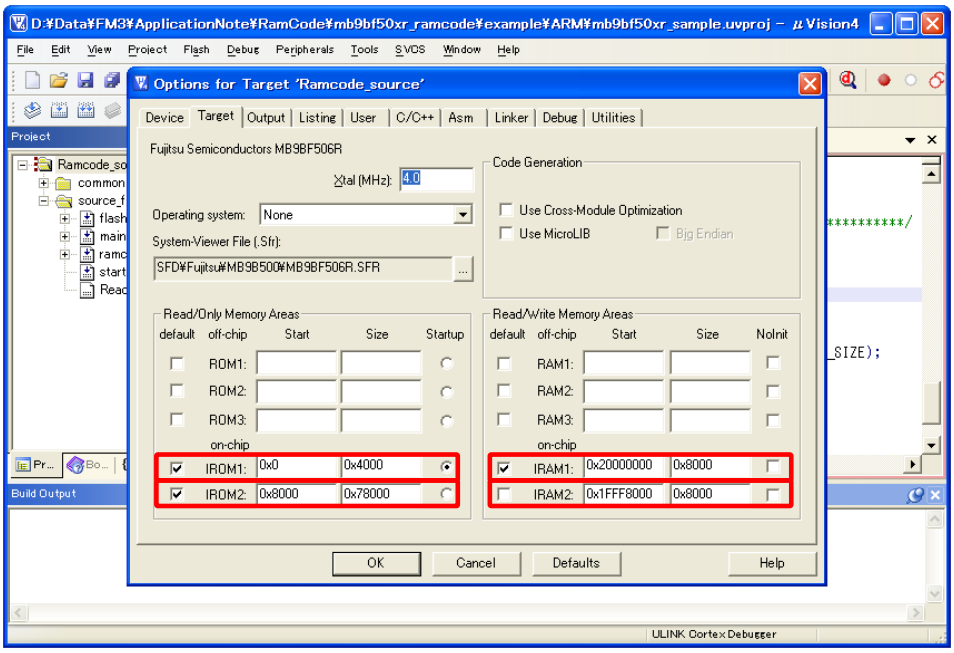
図 3. MDK-ARM の環境設定 (1)



5.3.1.2 MDK-ARM の環境設定 (2)

Option for Target 'Ramcode_source' ウィンドウの [Target] タグの [Read/Only Memory Areas] の [IROM1], [IROM2] および [Read/Write Memory Areas] の [IRAM1], [IRAM2] を図 4 のように設定します。

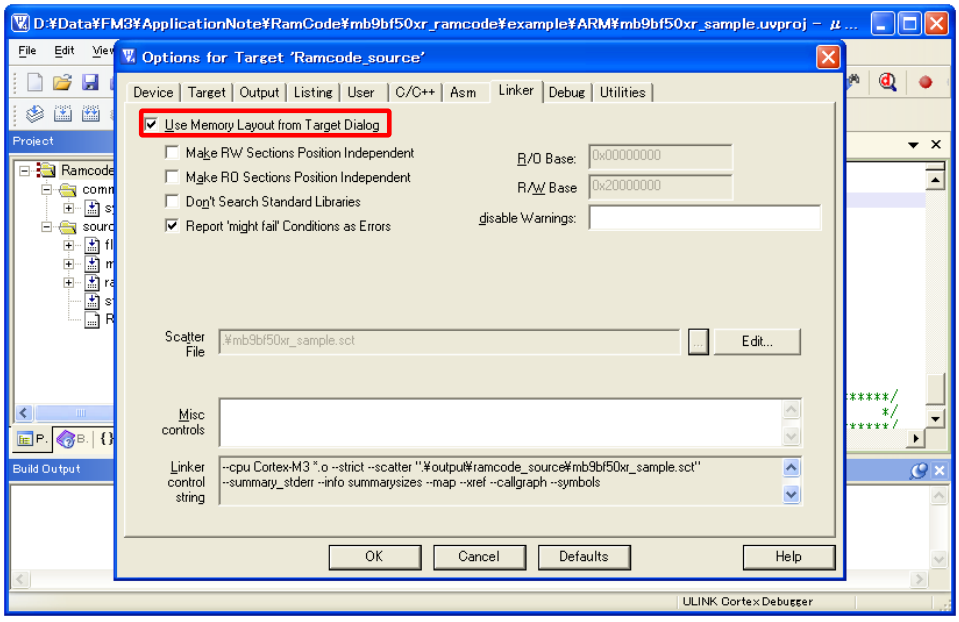
図 4. MDK-ARM の環境設定 (2)



5.3.1.3 MDK-ARM の環境設定 (3)

Option for Target 'Ramcode_source' ウィンドウの [Linker] タグの [Use Memory Layout from Target Dialog] をチェックします (図 5)。なお、本設定はデフォルトの状態です。

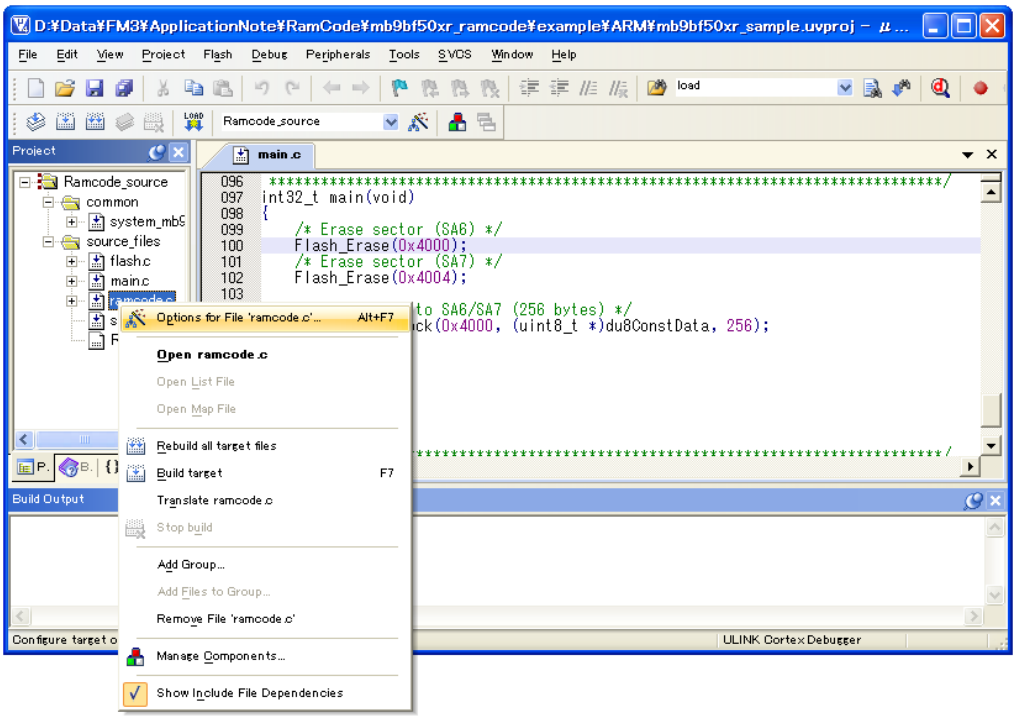
図 5. MDK-ARM の環境設定 (3)



5.3.1.4 MDK-ARM の環境設定 (4)

Project ウィンドウの RAM コードを生成したい C ソースファイル (ramcode.c) を右クリックし、「Option for File 'ramcode.c' ...」を選択します (図 6)。

図 6. MDK-ARM の環境設定 (4)



5.3.1.5 MDK-ARM の環境設定 (5)

Options for File 'ramcode.c' ウィンドウの [Properties] タグの [Memory Assignment] の [Code / Const] において IRAM を選択します (図 7, 図 8)。なお、図 7, 図 8 は IRAM2 (0x1FFF8000~0x1FFFFFFF) を設定した場合の例 (サンプルソフトウェアと同様) です。以上で設定が完了し、コンパイルすると指定した C ソースファイルの関数すべてが RAM コードとして設定されます。

図 7. MDK-ARM の環境設定 (5-1)

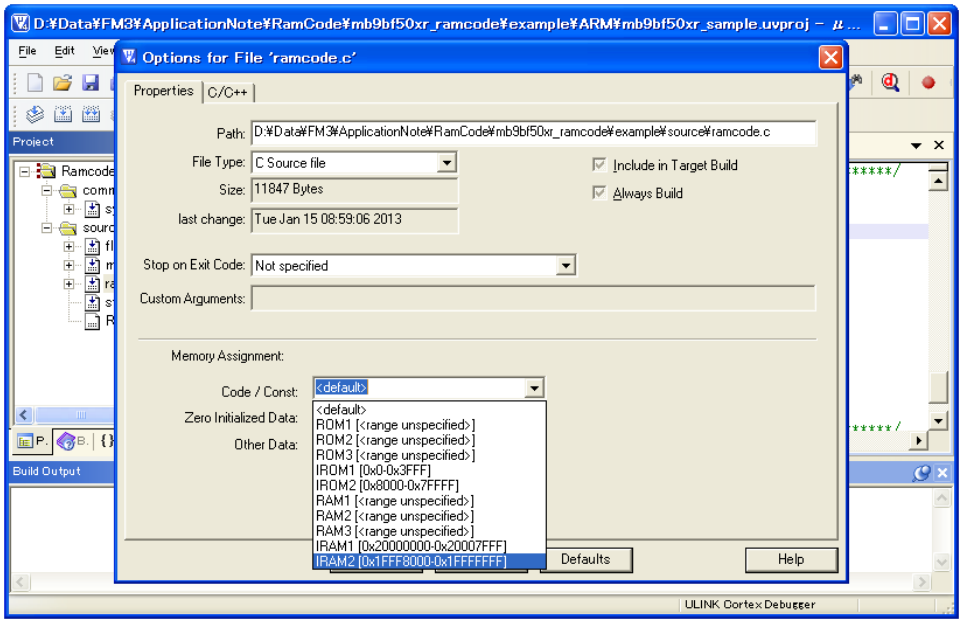
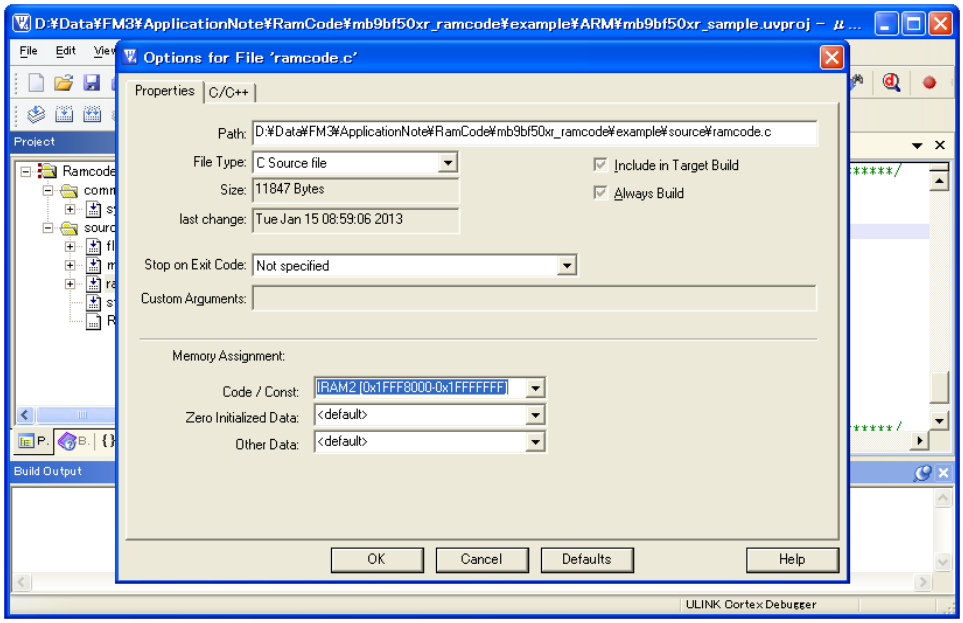


図 8. MDK-ARM の環境設定 (5-2)



5.3.2 関数ごとに RAM コードを生成する場合の設定

関数ごとに RAM コードを設定する場合は、ソースコードへの記述，スキャッタロード記述ファイルの設定および MDK-ARM の環境設定を行います。

5.3.2.1 ソースコードへの記述

RAM コードとして生成する関数に対して、図 9 に示すように関数の定義に「__attribute__ ((section (“セクション名”)))」のようにセクションの記述を追加します。本記述は RAM コードとするすべての関数に対して行う必要があります。

なお、サンプルソフトウェアにおいてはファイルごとに RAM コードを生成する場合（プロジェクト名: Ramcode_source、表 3 参照) も本記述をしています。

図 9. RAM コードとして使用する関数の記述

```
_attribute_ ((section (".ramfunc")))
en_result_t Ramcode_FlashErase(  uint32_t  u32SectorEraseAddress,
                                 boolean_t bWaitUntilFinished )
{
    ...
}
```

5.3.2.2 スキャッタロード記述ファイルの設定

スキャッタロード記述ファイルはリンカに渡すファイルで、メモリマップをテキスト形式で記述したファイルです。

スキャッタロード記述ファイルに RAM コードを生成したいセクションを設定します。図 10 は ramcode.c で設定したコードのセクション「.ramfunc」を RAM の 0x1FFF8000~0x1FFFFFFF に配置し (赤字部分)、ベクタテーブル，リセットハンドラ，RAM 初期化データを 0x00000000～0x00003FFF、プログラムコードを 0x00008000 以降へ配置する例で、サンプルソフトウェアにおける設定と同様です（「8.3 ファイル構成」における ARM フォルダ下にあるスキャッタロード記述ファイル「mb9bf50xr_sample.sct」の内容です）。なお、スキャッタロード記述ファイルは拡張子を「.sct」としてエディタなどで作成します。

図 10. スキャッタロード記述ファイルへの RAM コード生成のための設定例

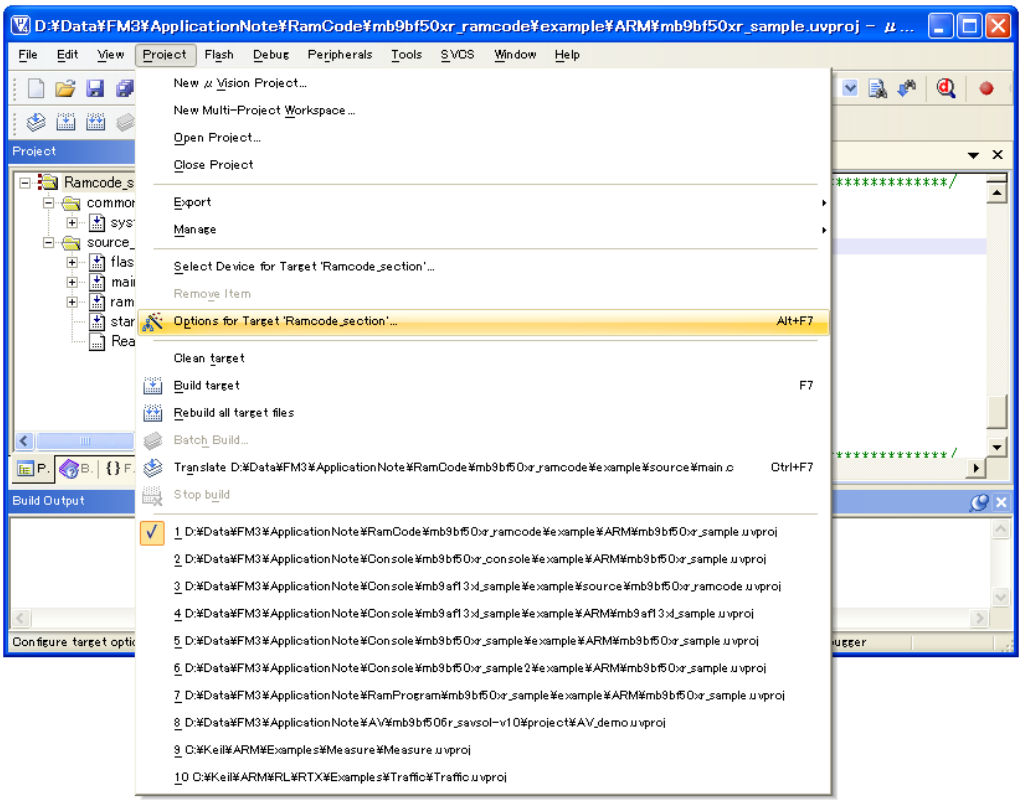
```
; *************************************************************
; ***       Scatter-Loading Description File generated by uVision       ***
; *************************************************************

LR_IROM1 0x00000000 0x00004000  {    ; load region size_region
  ER_IROM1 0x00000000 0x00004000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
  }
  RW_IRAM1 0x20000000 0x00008000  {  ; RW data
   .ANY (+RW +ZI)
  }
  RW_IRAM2 0x1FFF8000 0x00008000  {  ; RAM code
    ramcode.o (.ramfunc)
  }
}

LR_IROM2 0x00008000 0x00078000  {    ; load region size_region
  ER_IROM2 0x00008000 0x00078000  {  ; load address = execution address
   .ANY (+RO)
  }
}
```

スキャッタロード記述ファイルについては「Linker User Guide」[2] における「Using scatter files」を参照してください。

5.3.2.3 MDK-ARM の環境設定 (1)

MDK-ARM を起動しプロジェクトファイルを開いたら、[Project] メニューの [Option for Target 'Ramcode_section'...] を選択します (図 11)。

図 11. MDK-ARM の環境設定 (1)



5.3.2.4 MDK-ARM の環境設定 (2)

Option for Target 'Ramcode_Section' ウィンドウの [Linker] タグの [Use Memory Layout from Target Dialog] のチェックを外し (図 12 の 1)、[Scatter File] に作成したスキャッタロード記述ファイルを指定します (図 12 の 2, 図 13 の 3, 4, 図 14 の 5)。

図 12. MDK-ARM の環境設定 (2-1)

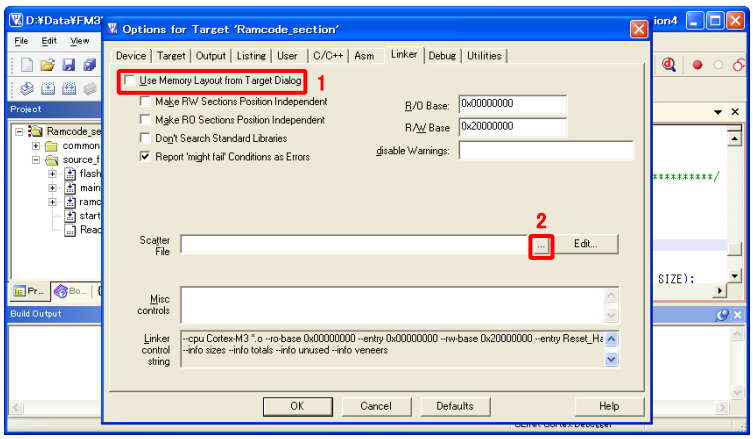


図 13. MDK-ARM の環境設定 (2-2)

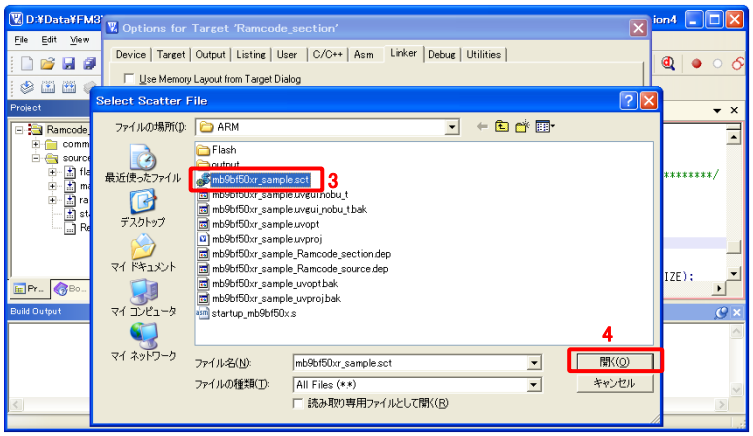
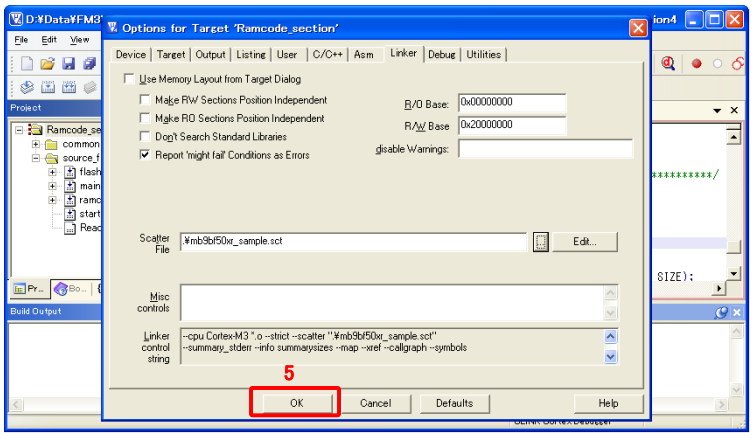


図 14. MDK-ARM の環境設定 (2-3)



以上で設定が完了し、コンパイルするとスキャッタロード記述ファイルで RAM への配置を指定したセクションが RAM コードとして設定されます。

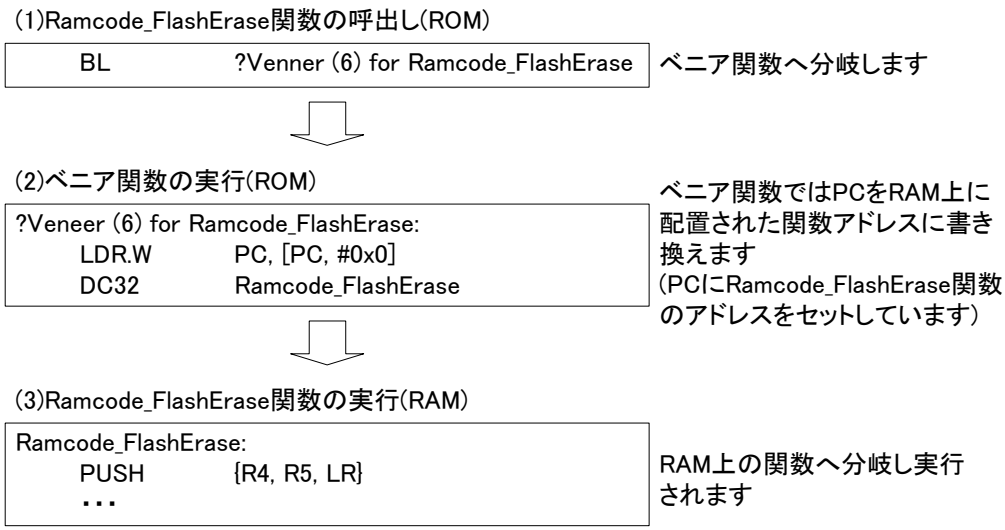
6 RAM コードの実行

RAM コードの関数 (例. Ramcode_FlashErase) を呼び出した場合、まず、BL 命令によりベニア関数へ分岐し、続いてベニア関数から RAM に配置された関数へ分岐します。本章では RAM コードを呼び出したときの実行内容をアセンブラコードにより示します。

6.1 IAR EWARM における RAM コードの実行

ベニア関数においてプログラムカウンタ (PC) を RAM に配置された関数のアドレスに書き換えます。RAM コードの関数呼出し時の実行内容を図 15 に示します。

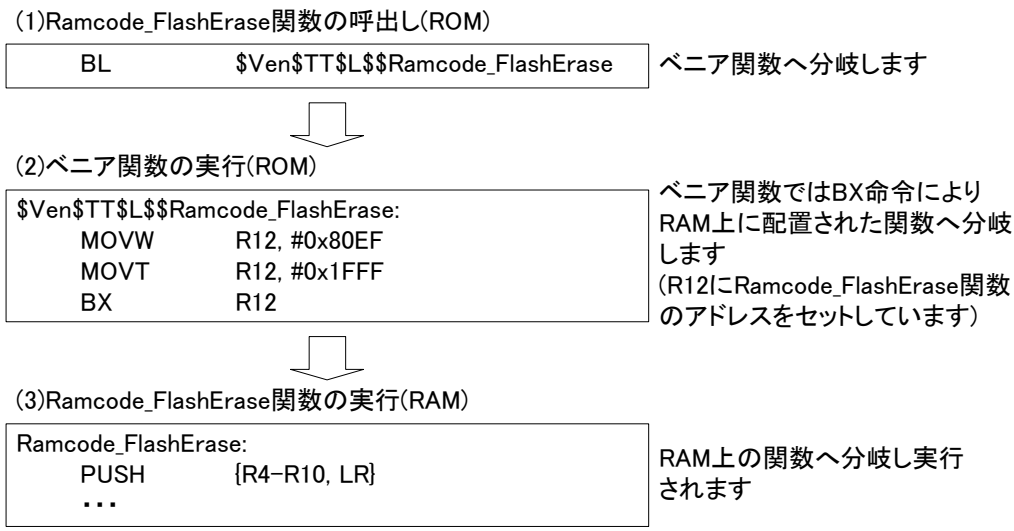
図 15. RAM コードの関数呼出し時の実行内容 (IAR EWARM)



6.2 KEIL MDK-ARM における RAM コードの実行

ベニア関数において BX 命令により RAM に配置された関数のアドレスに分岐します。RAM コードの関数呼出し時の実行内容を図 16 に示します。

図 16. RAM コードの関数呼出し時の実行内容 (KEIL MDK-ARM)



7 各環境および方法による特徴および差異

サンプルソフトウェアの各環境 (IAR EWARM および KEIL MDK-ARM) および方法による差異を表 4 にまとめます。

表 4. 各環境および方法による特徴および差異

環境および方法	特徴	容量 (バイト)		備考
IAR EWARM 関数毎	指定した関数のみが RAM コードとして生成可能 ただし、RAM コードとする関数すべてに「__ramfunc」の記述を追加する必要あり	ベニア関数	16	2 関数分 8 バイト×2 関数
		RAM コード	426	2 関数分 セクタ消去：234 バイト 書込み：192 バイト
KEIL MDK-ARM ファイル毎 (※1)	指定した C ソースファイル内のすべての関数を RAM コードとして生成可能	ベニア関数	20	2 関数分 10 バイト×2 関数
		RAM コード	382	2 関数分 セクタ消去：216 バイト 書込み：166 バイト
KEIL MDK-ARM 関数毎 (※1)	指定した関数のみが RAM コードとして生成可能 ただし、RAM コードとする関数すべてにセクションの記述を追加する必要あり	ベニア関数	20	2 関数分 10 バイト×2 関数
		RAM コード	382	2 関数分 セクタ消去：216 バイト 書込み：166 バイト

(※1) KEIL MDK-ARM においてはファイル毎，関数毎どちらの場合も容量は同じ

8 サンプルソフトウェア

8.1 概要

本アプリケーションノートに添付するサンプルソフトウェアでは、FM3 マイコンの MB9BF506R において Flash のセクタ消去用関数とデータ書込み用関数を RAM コードとして生成します。サンプルソフトウェアの動作を表 5 に、サンプルソフトウェア動作時の Flash メモリの状態を図 17 に示します (あらかじめ、あるプログラムが書き込まれていた場合の例)。なお、Flash へのアクセスについては「FM332 ビット・マイクロコントローラ MB9B500/400/300/100/MB9A100 Series フラッシュプログラミングマニュアル」[3] を参照してください。また、「8.3 ファイル構成」に示す flash.c, ramcode.c は MB9BF506R 用のため、ほかの品種の FM3 マイコンを使用する場合は、該当する品種のフラッシュプログラミングマニュアルを参照し、修正してください。なお、MB9BF506R の Flash のセクタ構成を表 6 に、サンプルソフトウェアのメモリマップを図 18 に示します。

表 5. サンプルソフトウェアの動作

No.	処理	備考
1	SA6 をセクタ消去	0x0000nXX0, 0x0000nXX8 (n:4～7, X:0～F) の 4 バイト毎を消去
2	SA7 をセクタ消去	0x0000nXX4, 0x0000nXXC (n:4～7, X:0～F) の 4 バイト毎を消去
3	データ書込み	0x00004000 から 256 バイトのデータ書込み

図 17. サンプルソフトウェア動作時の Flash メモリの状態

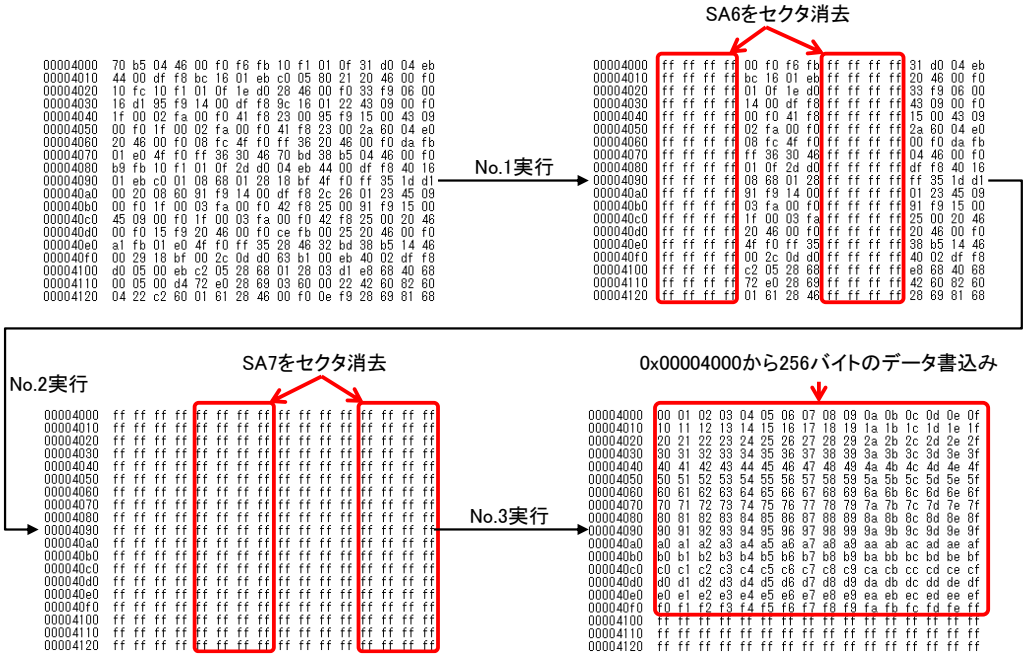


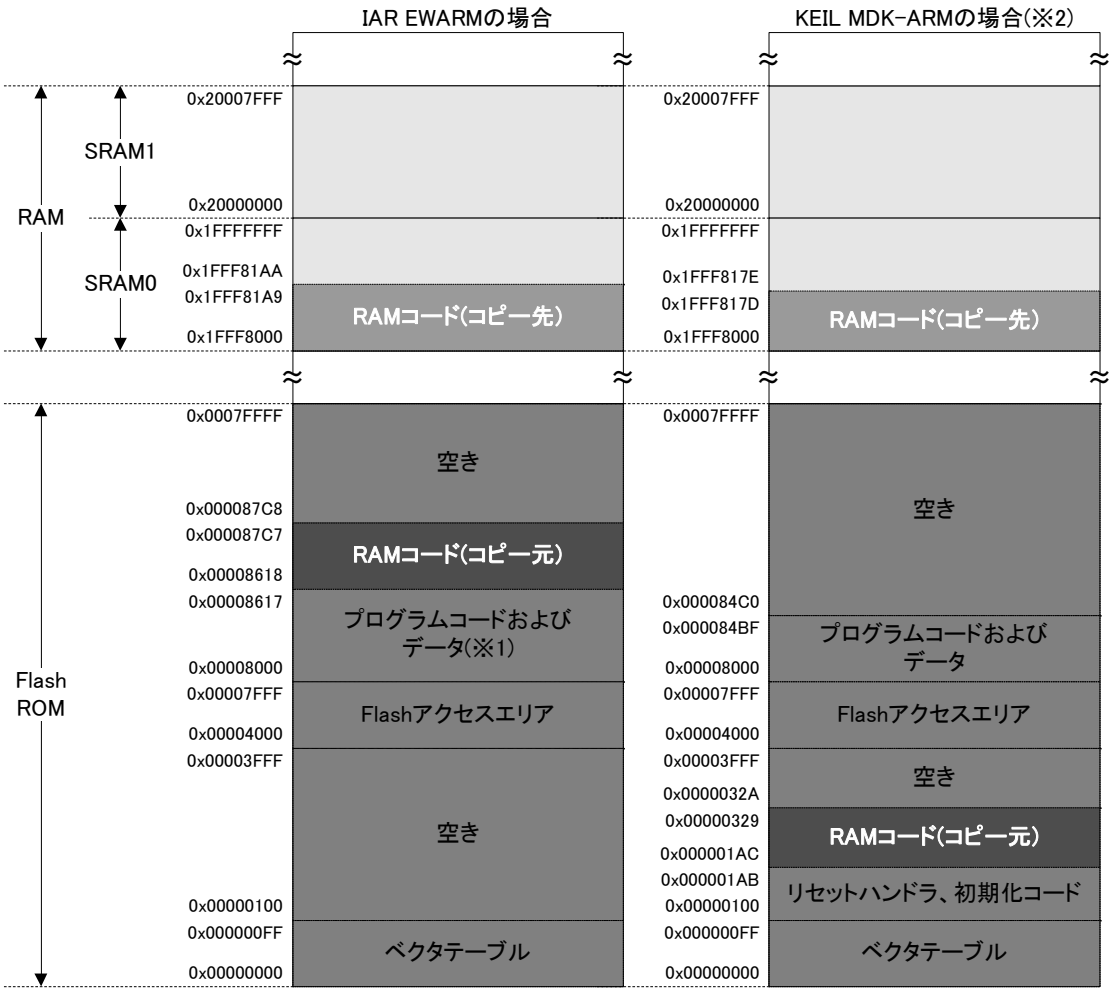
表 6. MB9BF506R の Flash のセクタ構成

アドレス	bit0 – 31 (※1)	bit32 – 63 (※2)
0x00000000～0x00003FFF	SA4 (8K バイト)	SA5 (8K バイト)
0x00004000～0x00007FFF	SA6 (8K バイト)	SA7 (8K バイト)
0x00008000～0x0001FFFF	SA8 (48K バイト)	SA9 (48K バイト)
0x00020000～0x0003FFFF	SA10 (64K バイト)	SA11 (64K バイト)
0x00040000～0x0005FFFF	SA12 (64K バイト)	SA13 (64K バイト)
0x00060000～0x0007FFFF	SA14 (64K バイト)	SA15 (64K バイト)

(※1) 0x000nXXX0 および 0x000nXXX8 から 4 バイト (n:0～7, X:0～F)

(※2) 0x000nXXX4 および 0x000nXXXC から 4 バイト (n:0～7, X:0～F)

図 18. サンプルソフトウェアのメモリマップ



(※1) リセットハンドラ、初期化コードを含む
(※2) Ramcode_source, Ramcode_senctionのどちらのプロジェクトも同様

8.2 ソースコードの記述

本アプリケーションノートに添付するサンプルソフトウェアにおける RAM コード用のソースコードは、図 2 および図 9 に示す記述をまとめています(図 19)。なお、「__ICCARM__」は IAR EWARM のコンパイラ、「__CC_ARM」は KEIL MDK-ARM のコンパイラ用の定義です (※)。

図 19. サンプルソフトウェアにおける RAM コード用ソースコードの記述

```
#ifdef _ICCARM_
  _ramfunc
#elif  _CC_ARM
  _attribute_ ((section (".ramfunc")))
#else
  #error Please check compiler and linker settings for RAM code
#endif
en_result_t Ramcode_FlashErase(  uint32_t  u32SectorEraseAddress,
                                 boolean_t bWaitUntilFinished )
{
    ...
}
```

(※) サンプルソフトウェアにおいては IAR EWARM のコンパイラおよび KEIL MDK-ARM のコンパイラにのみ対応しています。ほかのコンパイラを使用する場合にはエラーになりコンパイルできませんので、使用するコンパイラに合わせて記述を追加してください。

8.3 ファイル構成

本アプリケーションノートに添付するサンプルソフトウェアのファイル構成は下記のとおりです。

```
+---common                          :共用ヘッダファイルディレクトリ
  +---mb9b500r.h                    : ペリフェラル定義ヘッダファイル
  +---system_mb9bf50x.c             : システム設定ソースファイル
  +---system_mb9bf50x.h             : システム設定定義ヘッダファイル
+---example                         : サンプルディレクトリ
  +---ARM                           : KEIL 用プロジェクトディレクトリ
    +---Flash                       : Flash ロード用ファイルディレクトリ
      +---FLASH_DEBUG_README.txt    : Flash ロード Readme テキスト
      +---MB9BFx06_512.FLM          : Flash ロード用ファイル
    +---mb9bf50xr_sample.sct        : スキャッタロード記述ファイル
    +---mb9bf50xr_sample.uvopt      : オプションファイル
    +---mb9bf50xr_sample.uvproj     : プロジェクトファイル
    +---startup_mb9bf50x.s          : スタート UP アセンブラファイル
  +---IAR                           : IAR 用プロジェクトディレクトリ
    +---config                      : コンフィグレーションファイルディレクトリ
      +---flashloader               : Flash ローダディレクトリ
        +---FlashMB9BF50x.mac       : Flash ローダ起動マクロファイル
        +---FlashMB9BF506.flash     : Flash ローダ設定ファイル
        +---FlashMB9BF506.out       : Flash ローダ本体
        +---MB9BF506.board          : Flash ローダ設定指定ファイル
      +---mb9bf506.icf              : リンカ設定ファイル
      +---reset.mac                 : 起動マクロファイル
    +---mb9bf50xr_sample.dep        : 依存関係情報ファイル
    +---mb9bf50xr_sample.ewd        : プロジェクト設定ファイル
    +---mb9bf50xr_sample.ewp        : プロジェクトファイル
    +---mb9bf50xr_sample.eww        : ワークスペースファイル
    +---startup_mb9bf50x.s          : スタート UP アセンブラファイル
```

```
 +---source                    : サンプルソースディレクトリ
   +---base_type.h             : タイプ定義ヘッダファイル
   +---flash.c                 : Flash ドライバサンプルソースファイル
   +---flash.h                 : Flash ドライバサンプルヘッダファイル
   +---main.c                  : サンプルメインルーチンソースファイル
   +---mcu.h                   : MCU 依存ヘッダファイル
   +---ramcode.c               : RAM コードサンプルソースファイル
   +---ramcode.h               : RAM コードサンプルソースファイル
+---Readme.txt                 : サンプル Readme テキスト
```

9 参考ドキュメント

[1] IAR C/C++ 開発ガイド
IAR EWARM のヘルプより参照してください。

[2] Linker User Guide
KEIL MDK-ARM のヘルプにおける µVision_Help より参照してください。

[3] FM332 ビット・マイクロコントローラ MB9B500/400/300/100/MB9A100 Series フラッシュプログラミングマニュアル
http://www.cypress.com/ (本 URL のキーワード検索でキーワード「MN706-00004」にて検索してください)

※URL は予告なく変更される可能性があります。

10 改訂履歴

文書名: AN204424 - FM3 RAM コードの生成方法

文書番号: 002-04424

版	ECN 番号	変更者	発行日	変更内容
**	-	NNAK	01/16/2013	スパンションアプリケーションノート AN706-00063-1v1-J をサイプレスとして登録したものです。
			01/31/2014	社名変更および記述フォーマットの変換 サイプレスとして Spansion アプリケーションノート AN706-00063-1v1-J をドキュメントコード 002-04424 に登録しました。
*A	5640317	NNAK	02/23/2017	最新のテンプレートへ更新しました。
*B	5899064	NNAK	09/28/2017	Cypress の新ロゴを適用。

セールス, ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709