SH7253 SH7256R グループ

CAN を使用したユーザプログラムモードフラッシュ書き換え動作例

R01AN1640JJ0100
Rev.1.00
2013.04.02

要旨

本アプリケーションノートは、SH72544R ユーザプログラムモードを使用した、内蔵フラッシュ書き換えプログラム動作例をまとめたものです。外部制御ツールとのインタフェースとして、CAN を使用します。外部制御ツールは、内蔵 ROM(SH72544R)に書き込むデータを、CAN を介して内蔵 RAM へ格納します。内蔵フラッシュ書き換えプログラムはユーザブートマット上にあるものとします。

尚、本アプリケーションノートに掲載されているタスク例及びアプリケーション例は確認済みですが、実際にご使用になる場合には、必ず動作環境を確認の上ご使用くださいますようにお願いします。

【注】本アプリケーションノートのサンプルコードは SH7254R グループ用に作成しています。SH7253 SH7256R グループでご使用の場合は、本文 6.SH7253 SH7256R グループでご使用の場合の変更点に従い、修正して頂きます様お願い致します。SH7253,SH7256R グループでご使用の場合は、製品ヘッダファイル iodefine.h をグループ用のものに差替えてください。また、iodefine.h を差し替えた場合、レジスタ名をファイルに合わせソースを修正してください。

動作確認デバイス

SH72544R

適用条件

・統合開発環境　：ルネサス エレクトロニクス製
High-performance Embedded Workshop Ver.4.09.00

・C コンパイラ　：ルネサス エレクトロニクス製 SuperH RISC engine ファミリ
C/C++ コンパイラパッケージ Ver.9.04.00 Release 00

・コンパイルオプション　：High-performance Embedded Workshop でのデフォルト設定
-cpu=sh2afpu -object="$(CONFIGDIR)¥$(FILELEAF).obj"
-debug -gbr=auto -chgincpath -errorpath -global_volatile=0
-opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1
-nologo

目次

1. 概要

1.1 仕様

1.1.1 全体仕様

- 本アプリケーションノートは、ユーザプログラムモードでの内蔵フラッシュメモリ（以下 ROM と表記）の書き換えを行います。
- EB1(H’00002000)～EB31(H’0027FFFF)までを ROM 書き換え領域として使用します。
- ROM 書き換え対象デバイスは、内蔵 RAM に ROM 書き込みデータを格納するインタフェースとして CAN を使用しています。
- ROM 書き換え対象デバイスは、外部制御ツールから ROM に書き込むデータを受信します。また、ROM に書き込むデータの受信を行うため、共通の制御コマンドを設定します。
- 内蔵 ROM 書き換えプログラムはあらかじめ、ユーザブートマット上に格納しておきます。

1.1.2 CAN 通信仕様

- 外部制御ツールから ROM 書き換えの開始、ROM 書き込みデータダウンロード、ROM 書き込み終了を CAN コマンドで制御します。
- MB0、MB2、MB4 を受信用メールボックス、MB1 を送信用メールボックスとして使用します。
- ROM 書き換え対象デバイスと外部制御ツールとの CAN 通信を行う上で、スタートコマンド、書き込みデータ要求コマンド、書き込みデータダウンロードコマンド、書き込み終了コマンドを設定します。

システム構成図を図 1-1 に示します。

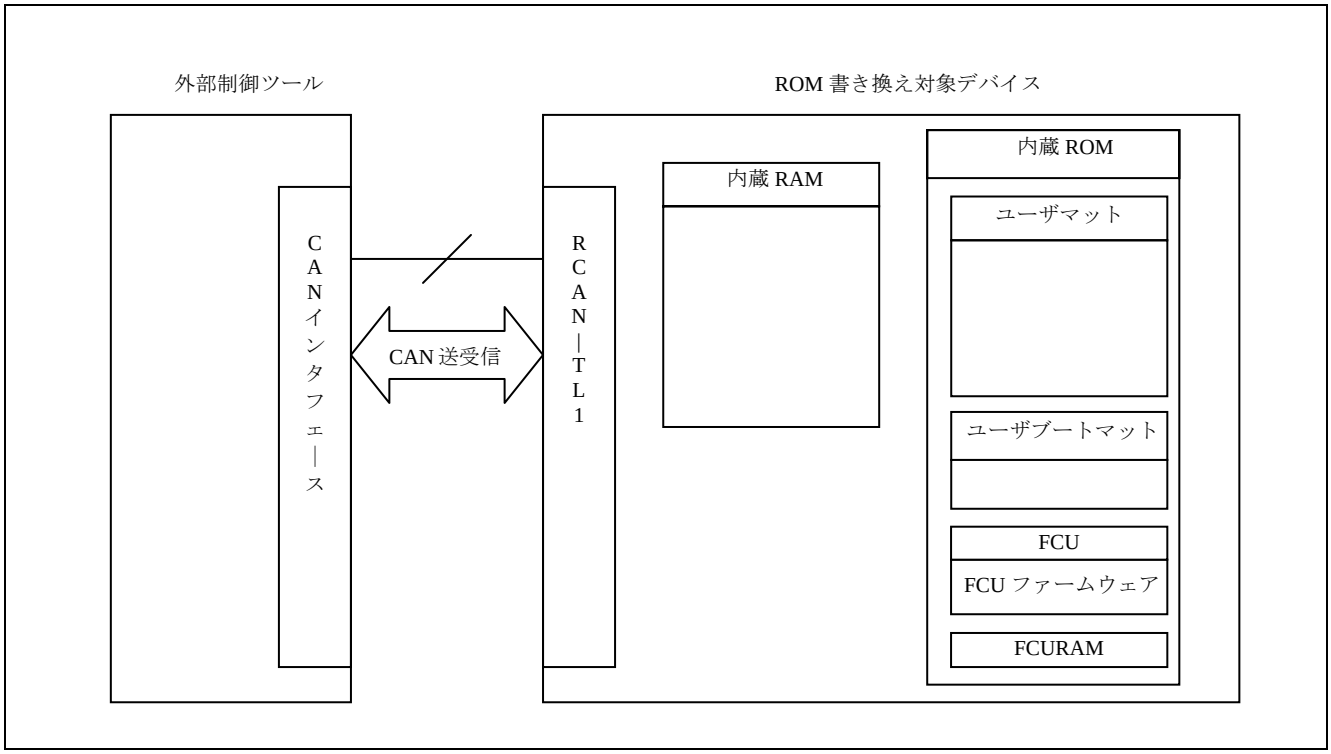


図 1-1　システム構成図

1.2 使用機能

- コントロールエリアネットワーク(RCAN-TL1)

1.3 適用条件

- マイコン :SH72544R
- 動作周波数 :入力クロック 20MHz
  :内部クロック 200MHz
  :周辺クロック 40MHz

1.4 動作モード

本アプリケーションノートでの ROM 書き換えは、内蔵 ROM 有効・外部バス有効のユーザプログラムモード(モード 6)で動作させます。本アプリケーションノートでは、任意のインタフェースとして CAN を使用します。

端子設定を表 1-1 に示します。

表 1-1 端子設定

モード名	端子設定				
	MD4/MD3	MD2	MD1	MD0	FWE
ユーザプログラムモード	0	0	1	0	1

1.5 外部接続端子

ROM 書き換えを行うには、ROM 書き換え対象デバイスの CAN Ach TxD 端子・CAN Ach RxD 端子を外部制御ツールと接続します。接続することで、外部制御ツールから CAN 通信にてコマンドの送受信、ROM 書き込みデータの受信を行います。

2. 使用機能説明

2.1 レジスタ説明

ユーザプログラムモードでの ROM 書き込み/消去で、使用するレジスタの説明を以下に示します。

- FCU ファームウェア転送での使用レジスタ

フラッシュモードレジスタ（FMODR）

FMODR は、FCU の動作モードを指定するレジスタです。内蔵 ROM が無効なモードでは、FMODR の読み出しデータは H'00 になり、書き込みは無効化されます。

フラッシュアクセスエラー割り込み許可レジスタ（FAEINT）

FAEINT は、フラッシュインタフェースエラー割り込み（FIFE）の出力許可／禁止を設定するためのレジスタです。内蔵 ROM が無効なモードでは、FAEINT の読み出しデータは H'00 になり、書き込みは無効化されます。

フラッシュプロテクトレジスタ（FPROTR）

FPROTR は、ロックビットによる書き込み／消去プロテクト機能の有効／無効を設定するためのレジスタです。内蔵 ROM が無効なモードでは、FPROTR の読み出しデータは H'0000 になり、書き込みは無効化されます

FCURAM イネーブルレジスタ(FCURAME）

FCURAME は、FCURAM へのアクセスを許可/禁止に設定するために使用するレジスタです。FCU ファームウェアを FCURAM に転送時、アクセスを許可に設定(H’C401 に設定)します。FCURAME 設定時には、上位ビットを H’C4 に固定してください。それ以外をセットした場合、FCURAME への書き込みは無効になります。

- ROM 書き込み/消去処理での共通使用レジスタ

ROM マット選択レジスタ(ROMMAT)

ROMMAT は、ユーザマット/ユーザブートマットの切り替えを行う時に使用するレジスタです。ユーザマット選択時は H’3B00、ユーザブートマット選択時は H’3B01 に設定します。ROMMAT 設定時は、上位ビットを H’3B に固定してください。それ以外をセットした場合、ROMMAT への書き込みは無効になります。

フラッシュ P/E モードエントリレジスタ(FENTRYR)

FENTRYR は、ROM/EEPROM を P/E モードまたはリードモードに設定するために使用するレジスタです。ROM 書き込み/消去を行う場合は、P/E モードに設定します。また、ROM は FENTRY0（H’80800000～H’808FFFFF）、 FENTRY1（H’80900000～H’809FFFFF）、FENTRY3（H’80A00000～H’80A3FFFF）、FENTRY4（H’80A40000～H’80A7 FFFF）に分かれています。それぞれ書き込み/消去を行う領域ごとに FENTRY0=1 のとき H’AA01、FENTRY1=1 のとき H’AA02、FENTRY3=1 のとき H’AA08、FENTRY4=1 のとき H’AA10 に設定します。全ての領域を一度に P/E モードにすることはできません。また、FENTRYR 設定時は、上位ビットを H’AA に固定してください。それ以外をセットした場合、FENTRYR への書き込みは無効になり、FENTRYR はクリアされます。

【注】* ROM の書き込み/消去用アドレスは H’80800000～H’80A7FFFF となります。

フラッシュプロテクトレジスタ(FPROTR)

FPROTR は、ロックビットによる書き込み/消去プロテクト機能の有効/無効を設定するためのレジスタです。

- エラー処理での使用レジスタ

フラッシュステータスレジスタ(FSTATR0)

FSTATR0 は、FCU の状態を確認するためのレジスタです。書き込み/消去中にエラーが発生した場合、または FCU が不正なコマンドや不正な ROM/EEPROM アクセスを検出した場合に、対応したエラービットがセットされます。

フラッシュステータスレジスタ(FSTATR1)

FSTATR1 は、FCU の状態を確認するためのレジスタです。FCU の CPU 処理においてエラーが発生した場合、または FCURAM の読み出し時に ECC エラーが発生した場合に、対応したエラービットがセットされます。

フラッシュリセットレジスタ(FRESETR)

FRESETR は、FCU の初期化を行うために使用するレジスタです。FCU を初期化する場合は、FRESET ビットを 1 にセットした状態を 20μs 保持します。また、20μs 後に FRESET ビットを 0 クリアすることで初期化が完了となります。また、FRESETR 設定時は、上位ビットを H'CC に固定してください。

2.2 ユーザプログラムモードでの ROM 書き込み/消去

ユーザプログラムモードでの ROM 書き込み/消去は、マイコンに内蔵されている FCU ファームウェアを使用します。FCU ファームウェアには、ROM の 256Byte 書き込み/1 ブロック消去プログラムが内蔵されています。また、FCU ファームウェアに FCU コマンドを発行することで、コマンドに対応した処理を FCU ファームウェアが行います。FCU コマンド発行は、ROM 書き込み/消去用アドレスに対する P バスライトアクセスで実現されます。ROM 書き込み/消去は、内蔵 RAM 上で行います。

ROM 書き込み/消去を行うために必要な手続きプログラムを以下に示します。また、消去用 FCU コマンドを表 2-1、書き込み用 FCU コマンドを表 2-2 に示します。

(1) FCU ファームウェアを制御するプログラム(以下 ROM 書き換え制御プログラムと表記)

(2) (1)を内蔵 RAM に転送するプログラム

(3) FCU ファームウェアを FCURAM に転送するプログラム

表 2-1 消去用 FCU コマンド

コマンド	バス サイクル数	1 サイクル目		2 サイクル目	
		アドレス	データ	アドレス	データ
ブロックイレーズ	2	EB0～31 + H’80800000	H’20	EB0～31 + H’80800000	H’D0

表 2-2 書き込み用 FCU コマンド

コマンド	バス サイクル数	1 サイクル目		2 サイクル目	
		アドレス	データ	アドレス	データ
プログラム	131	H’80800000～ H’80A7FFFF	H’E8	H’80800000～ H’80A7FFFF	H’80

3 サイクル目		4～130 サイクル目		131 サイクル目	
アドレス	データ	アドレス	データ	アドレス	データ
H’80800000～ H’80A7FFFF	ROM 書き込み データ(1 ワード目)	H’80800000～ H’80A7FFFF	ROM 書き込みデータ (2～128 ワード目)	H’80800000～ H’80A7FFFF	H’D0

【注】* ROM アドレスに H’80800000 を加えることで、ROM 書き込み/消去用アドレスになります。

(ROM 読み出し用アドレス：H’00000000～H’0027FFFF、ROM 書き込み/消去用アドレス：H’80800000～H’80A7FFFF)

2.3 ROM 消去

ROM 消去は、メモリマップで指定されているブロック単位で消去を実行します。また、ROM 消去は消去対象ブロックに消去用の FCU コマンドを書き込むことで、CPU が FCU に消去用の FCU コマンドを発行し、FCU が消去対象ブロックを消去します。

消去対象ブロック図を図 2-1 に示し、ROM 消去フローチャートを図 2-2 に示します。

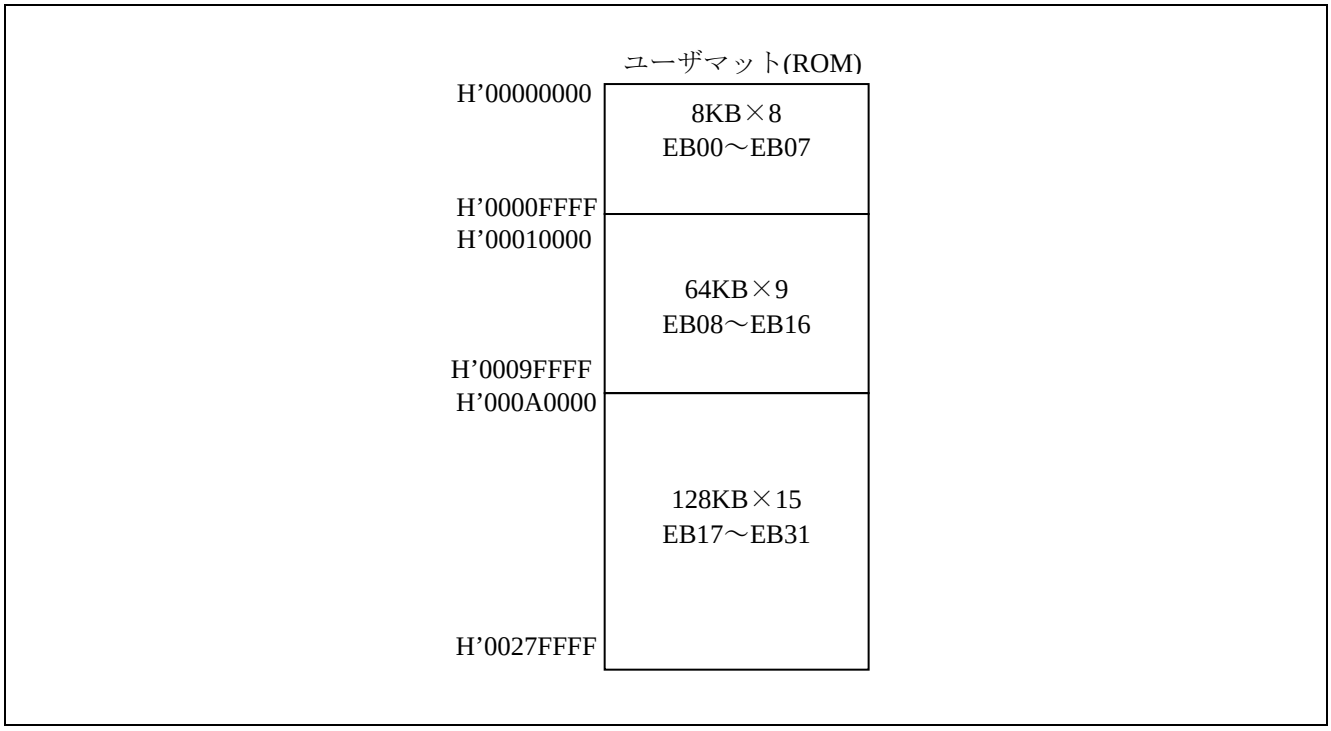


図 2-1　消去対象ブロック図

● ROM 消去手順

(1) ROM 消去を許可するため、FWE を 1 にセットします。(FPMON で確認可能)

(2) FCU ファームウェアを FCURAM に転送します 。

(3) ROM 書き換え制御プログラムを内蔵 RAM に転送します。

(4) 消去ブロックを指定します。また、指定した消去ブロックに乗じて FENTRYR を設定します。

(5) 消去ブロックに FCU コマンド[H’20]を書き込みます。

(6) 消去ブロックに FCU コマンド[H’D0]を書き込みます。(消去処理開始)

(7) 消去完了は、FSTATR0 の FRDY ビットで確認します。(消去時間が長時間かかる場合*1 は、エラーと見なし FCU を初期化します*2)

(8) 消去完了後は、FSTATR0 の ILGLERR / ERSERR ビットを確認します。

【注】 *1　(7)の処理で行うタイムアウト判定には目標値 0.8sec×1.1 の時間を設定します。
*2　(7)の処理でエラー確認後の FCU 初期化には、電気的特性より 20us のウェイトを置きます。
*3　FENTRYR レジスタの FENTRYn ビットを 1 から 0 に変更し ROM リードモードに遷移させる場合は、FENTRYn ビットに 0 を書き込み、FENTRYR レジスタのダミーリード後、NOP 命令を 5 個以上実行してください。

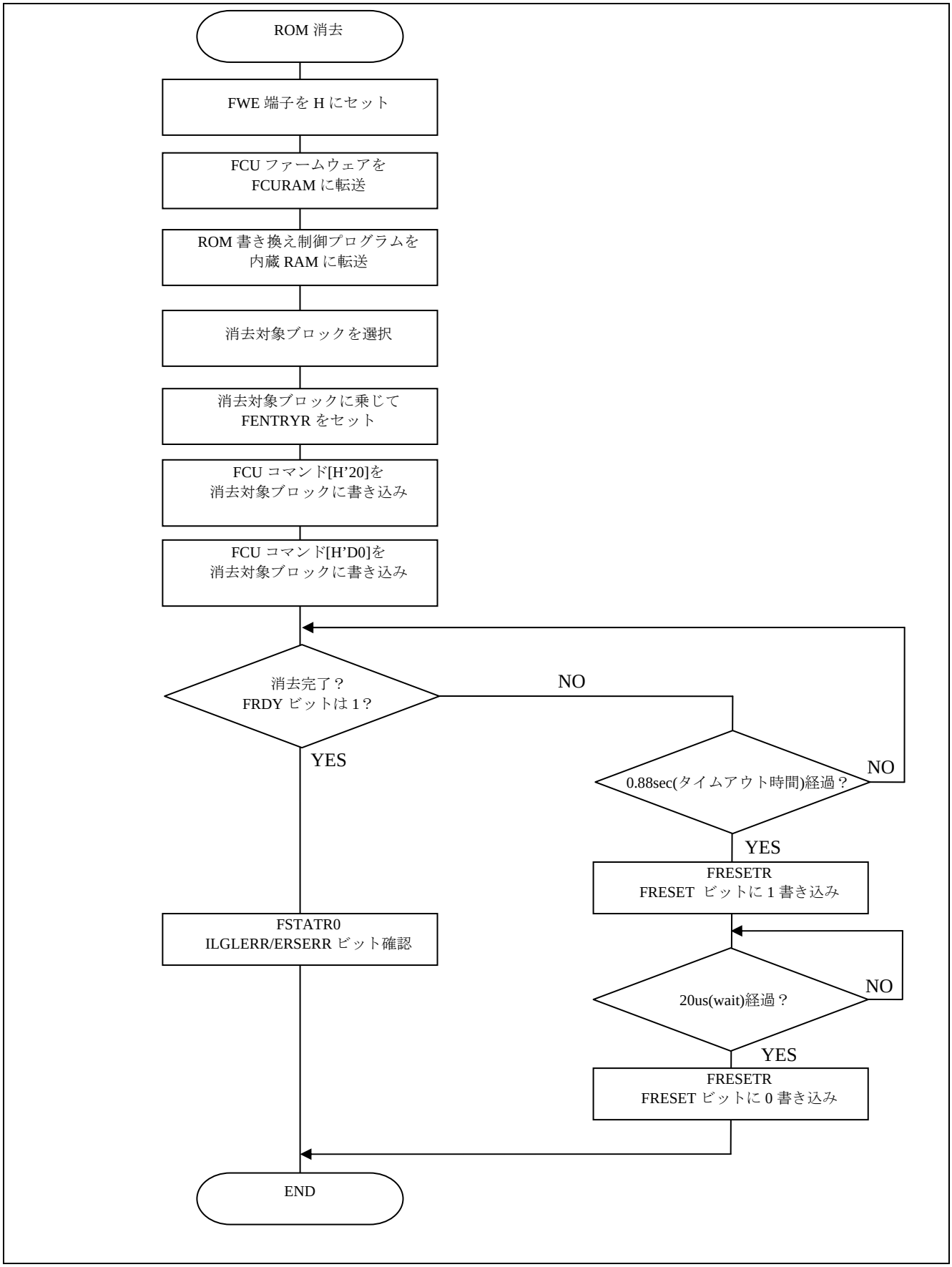


図 2-2　ROM 消去フローチャート

2.4 ROM 書き込み

ROM 書き込みは、書き込み先アドレスに書き込み用の FCU コマンドを書き込むことで、CPU が FCU に書き込み用の FCU コマンドを発行し、FCU は指定された書き込み先アドレスから 256Byte 書き込みを行います。

● ROM 書き込み手順

(1) ROM 書き込みを許可するため、FWE 端子を H にセットします。（FPMON で確認可能）

(2) FCU ファームウェアを FCURAM に転送します 。

(3) ROM 書き換え制御プログラムを内蔵 RAM に転送します。

(4) 書き込み先アドレスを指定します。また、指定した書き込み先アドレスに乗じて FENTRYR を設定します。

(5) 書き込み先アドレスに FCU コマンド[H’E8]を書き込みます。

(6) 書き込み先アドレスに FCU コマンド[H’80]を書き込みます。

(7) 書き込み先アドレスに ROM 書き込みデータをワードサイズで書き込みます。

(8) 256Byte 分の ROM 書き込みデータを書き込み先アドレスに書き込みます。

(9) 書き込み先アドレスに FCU コマンド[H’D0]を書き込みます。(書き込み処理開始)

(10) 書き込み完了は、FSTATR0 の FRDY ビットで確認します。
(書き込み時間が長時間経過した場合*1 は、エラーと見なし FCU を初期化します*2)

(11) 書き込み完了後は、FSTATR0 の ILGLERR / PRGERR ビットを確認します。

【注】 *1 (10)の処理で行うタイムアウト判定には目標値 2ms×1.1 の時間を設定します。
*2 (10)の処理で行うエラー確認後の FCU 初期化には、電気的特性より 20us のウェイトを置きます。
*3 FENTRYR レジスタの FENTRYn ビットを 1 から 0 に変更し ROM リードモードに遷移させる場合は、FENTRYn ビットに 0 を書き込み、FENTRYR レジスタのダミーリード後、NOP 命令を 5 個以上実行してください。

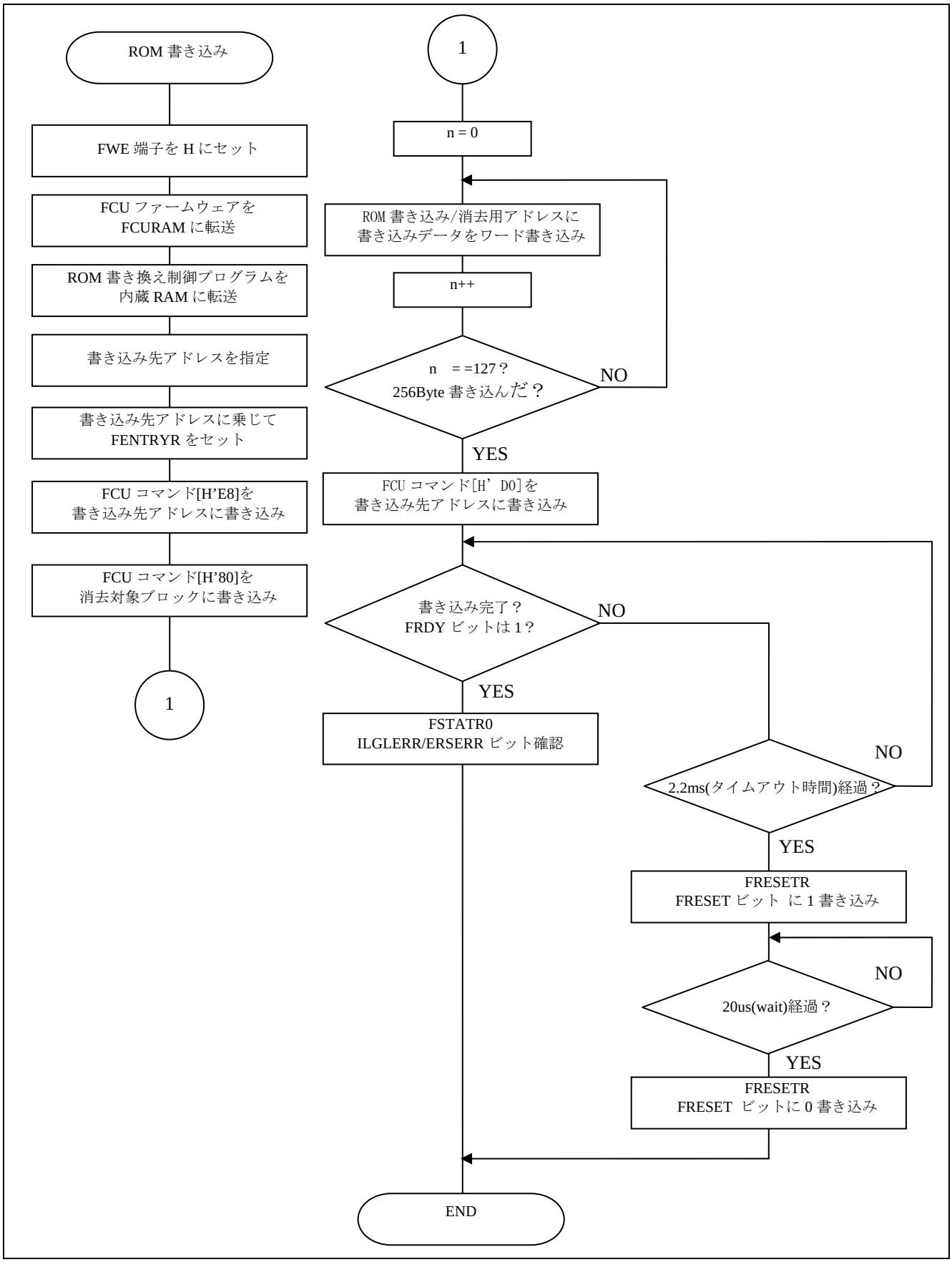


図 2-3　ROM 書き込みフローチャート

3. 外部制御ツール使用した ROM 書き込み/消去

3.1 動作概要

リセットスタート後、外部制御ツールから CAN 通信でスタートコマンドを受信し、ユーザマット上のプログラムを実行。ユーザブートマット上の ROM 書き換え制御プログラムを内蔵 RAM に転送し実行します。

ROM 消去(EB1～31)完了後、書き込み要求コマンドを送信し、書き込み終了コマンドを受信することで書き込みデータの有無を確認。書き込みデータがあるときユーザマット上の ROM 書き換えを行い、無いときはプログラムを終了します。

3.2 動作手順

動作手順①～⑧は、3-3 全体シーケンスに対応します。

3.2.1 ① ユーザマット/RAM 転送(ROM/RAM 転送)

リセットスタート後、スタートコマンドを受信し、「ROM マット切り替えプログラム(ROM)」、「ユーザブートマット/RAM 転送プログラム(ROM)」を RAM に転送します。以降の処理は RAM 上で実行します。

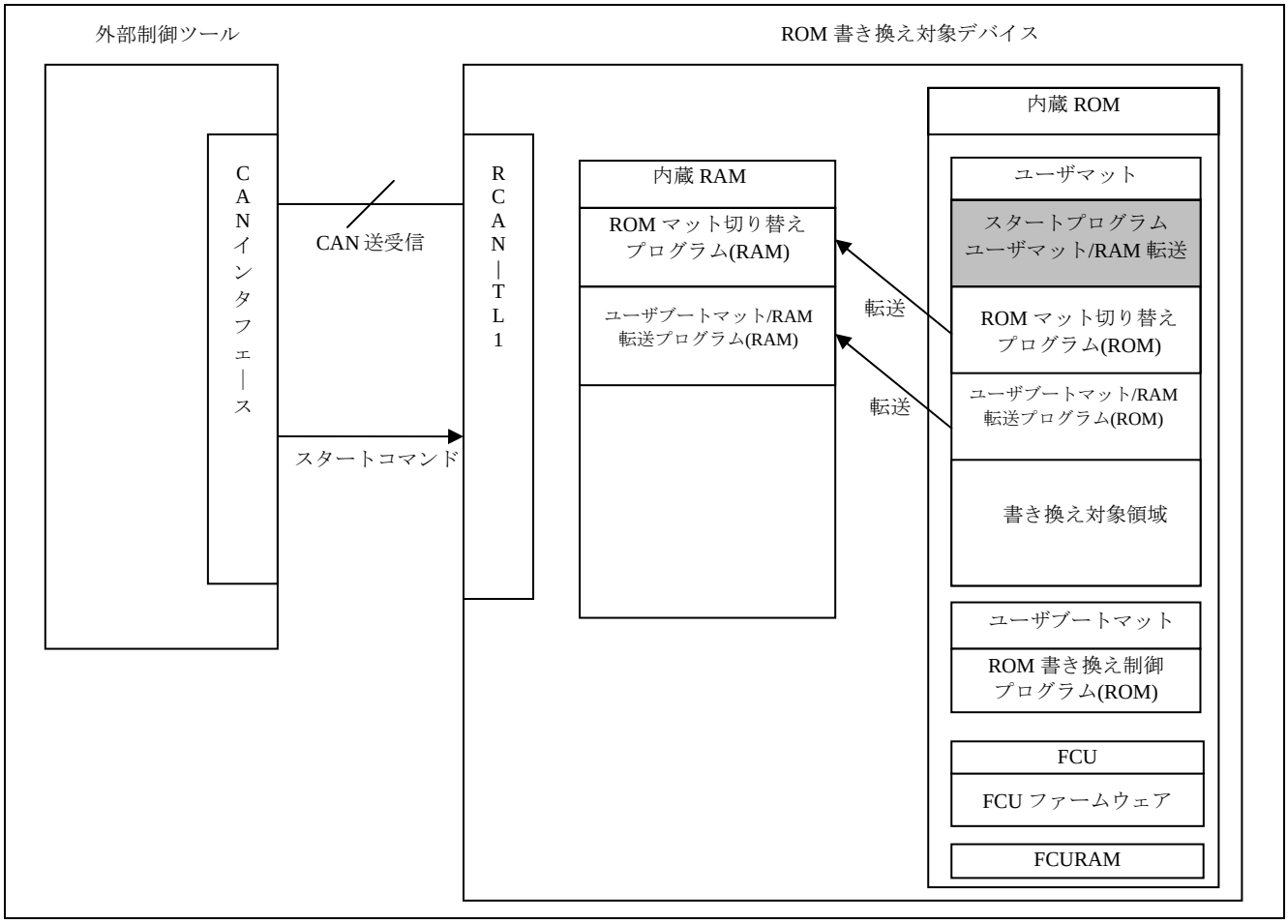


図 3-1 ROM 書き込み/消去開始動作

【注】 *ROM(ユーザマット、ユーザブートマット)上から RAM 上に転送するプログラムは、全て RAM 上で実行します。
(ROM)：ROM 上にあるプログラム、データ、(RAM)：RAM 上にあるプログラム、データと切り分けます。

関数説明

表 3-1 「main()関数」

関数名	概要
main()	プログラム開始。 スタートコマンド受信後、「ROM マット切り替え関数」、 「ユーザブートマット/RAM 転送プログラム」を RAM に転送します。

フローチャート

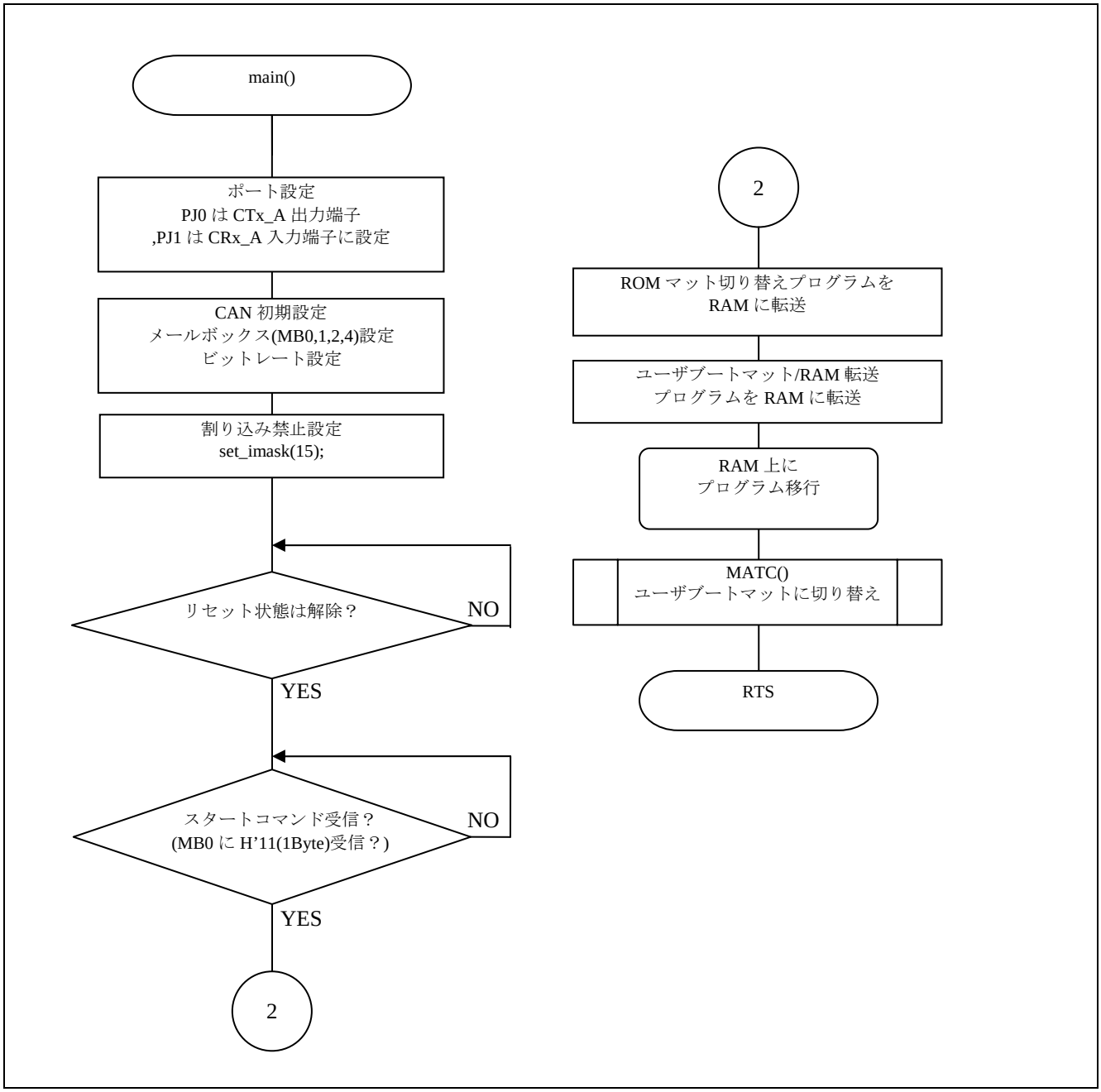


図 3-2 「main 関数()」のフローチャート

「動作手順①」

3.2.2 ② ROM マット切り替え(ユーザマット→ユーザブートマット)

「ROM マット切り替えプログラム(RAM)」を実行し、ROM マットをユーザブートマットに切り替えます。

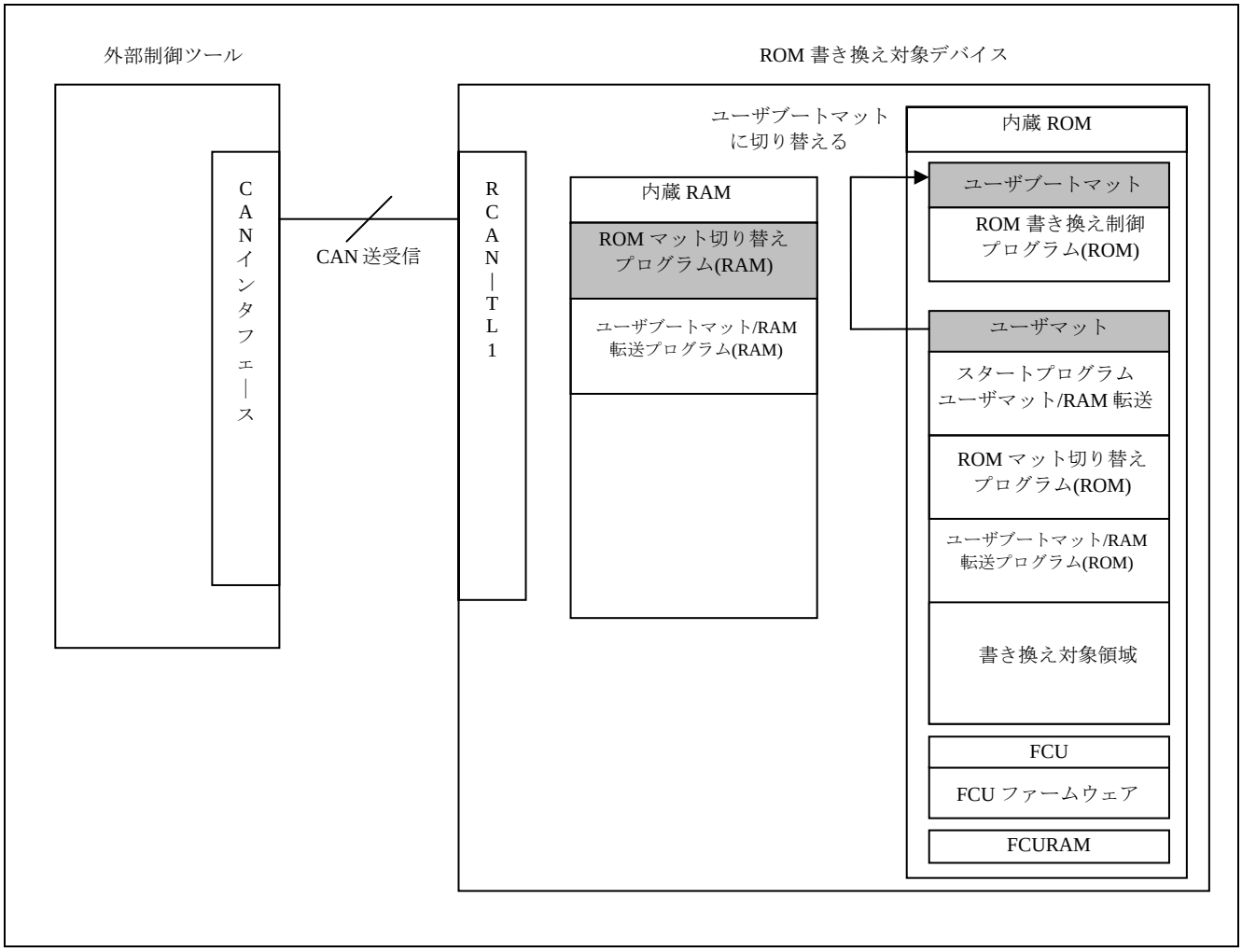


図 3-3 ROM マット切り替え動作(ユーザブートマットへ移行)

「MATC()関数」実行

関数説明

表 3-2 「MATC()関数」

関数名	概要
MATC()	ROM マットを切り替えます。 (ユーザマット→ユーザブートマット) (ユーザブートマット→ユーザマット)

フローチャート

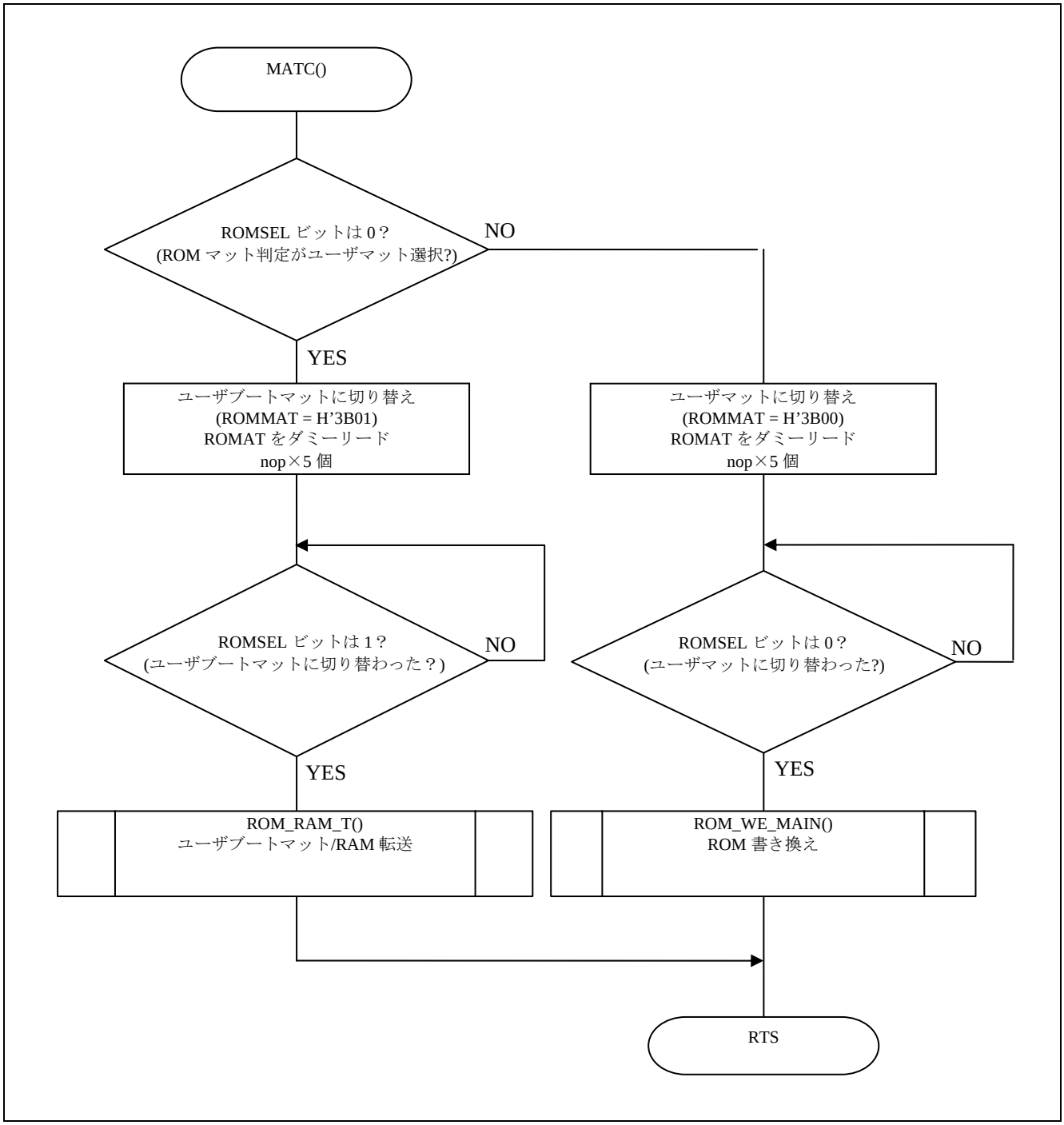


図 3-4　「MATC()関数」のフローチャート

「動作手順②④」

3.2.3 ③ ユーザブートマット/RAM 転送(ROM/RAM 転送)

「ユーザブートマット/RAM 転送プログラム(RAM)」を実行し、「ROM 書き換え制御プログラム(ROM)」を RAM に転送します。

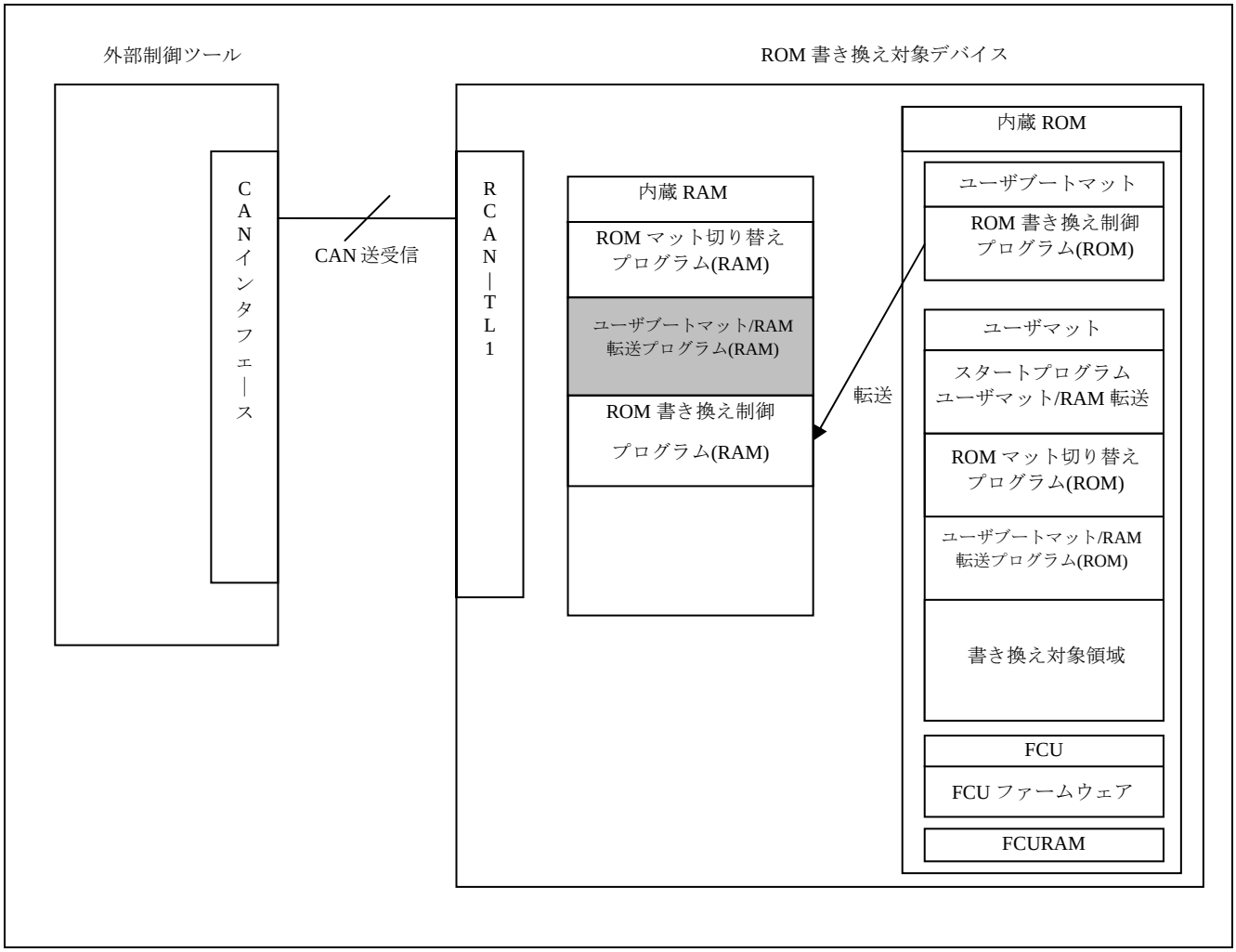


図 3-5 ユーザブートマット/RAM 転送動作

「ROM_RAM_T()関数」実行

関数説明

表 3-3 「ROM_RAM_T()関数」

関数名	概要
ROM_RAM_T()	ROM 書き換え制御関数、FCU ファームウェア転送関数、FCU 初期化関数、ROM 消去関数、ROM 書き込みダウンロード関数、ROM 書き込み関数を RAM に転送します。

フローチャート

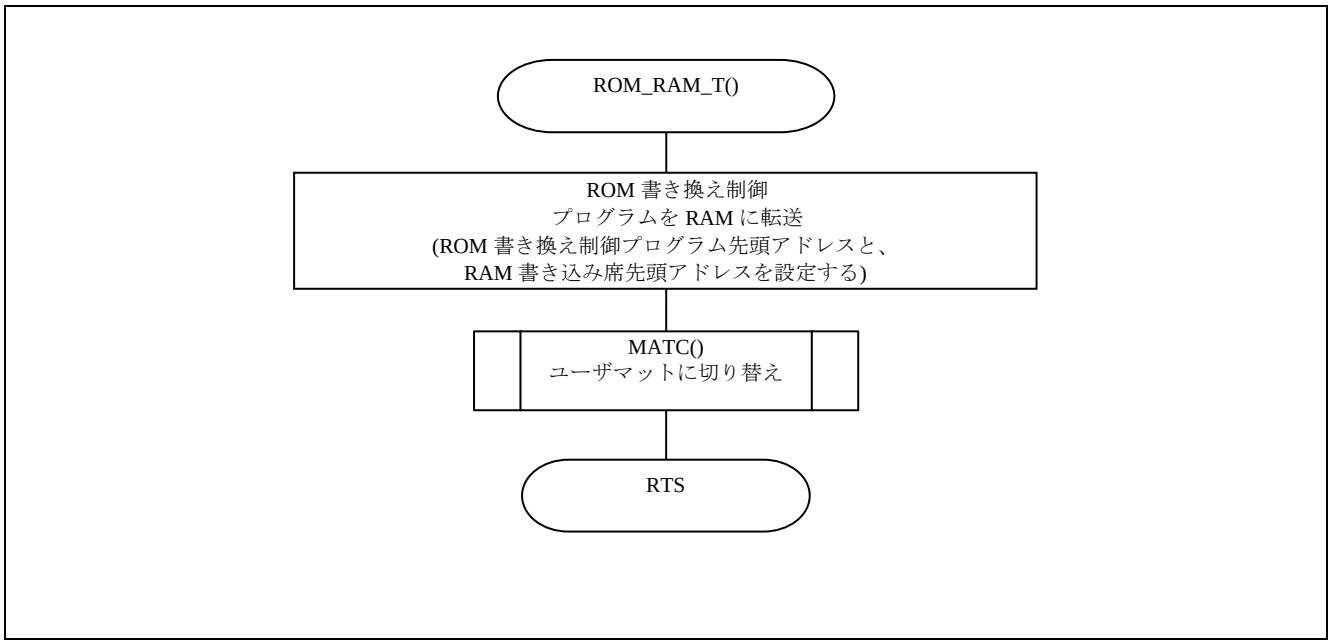


図 3-6 「ROM_RAM_T()関数」のフローチャート

「動作手順③」

3.2.4 ④ ROM マット切り替え(ユーザブートマット→ユーザマット)

「ROM マット切り替えプログラム(RAM)」を実行し、ROM マットをユーザマットに切り替えます。

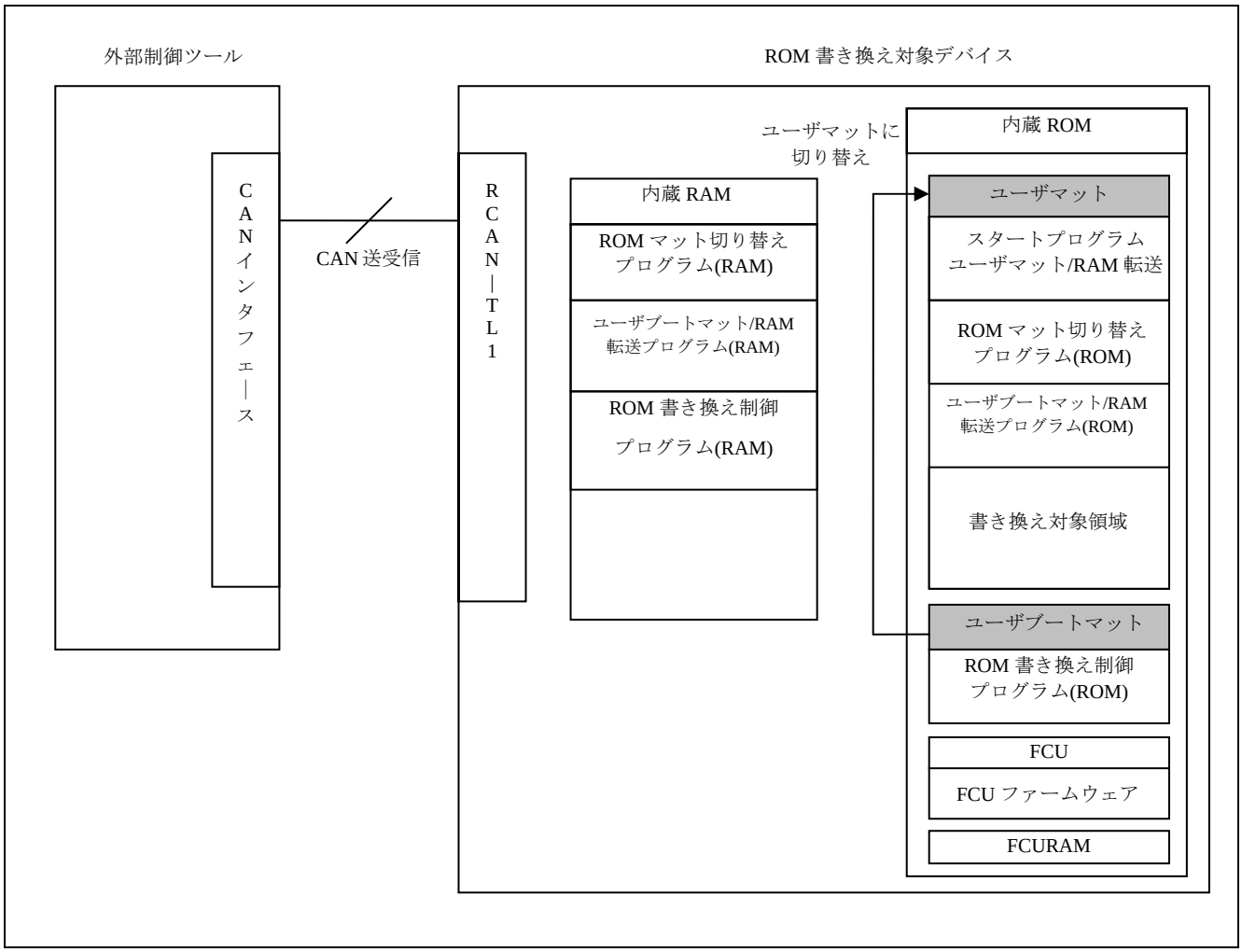


図 3-7 ROM マット切り替え動作(ユーザマットへ移行)

「MATC()関数」実行

関数説明

動作手順②の関数説明を参照。

フローチャート

動作手順②のフローチャート参照。

3.2.5 ⑤ FCU ファームウェア転送

「ROM 書き換え制御プログラム(RAM)」の「ROM 書き換え制御関数」、「FCU ファームウェア転送関数」、「FCU 初期化関数」を実行。FCURAM に FCU ファームウェアを転送し、FCU コマンドの受付を可能にします。

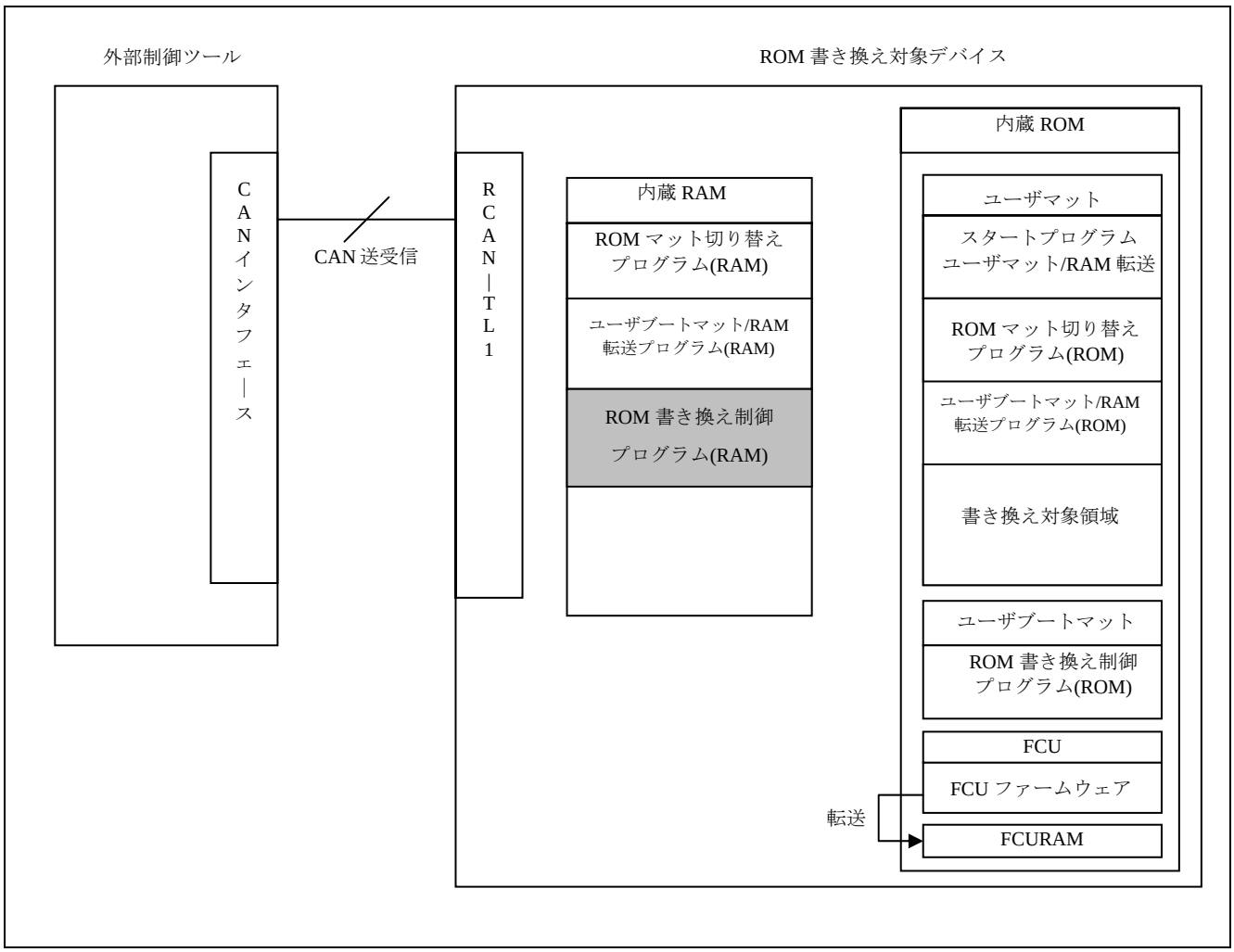


図 3-8 FCU ファームウェア転送動作

「ROM_WE_MAIN()関数」「FCURESET()関数」「FCU_FIRMCOPY()関数」実行

関数説明

表 3-4 「ROM_WE_MAIN()関数」「FCU_RESET()関数」「FCU_FIRMCOPY()関数」

関数名	概要
ROM_WE_MAIN()	ROM 書き換え制御プログラムを開始。
FCU_RESET()	FCU を初期化します。
FCU_FIRMCOPY()	FCU ファームウェアを FCURAM に転送します。

フローチャート

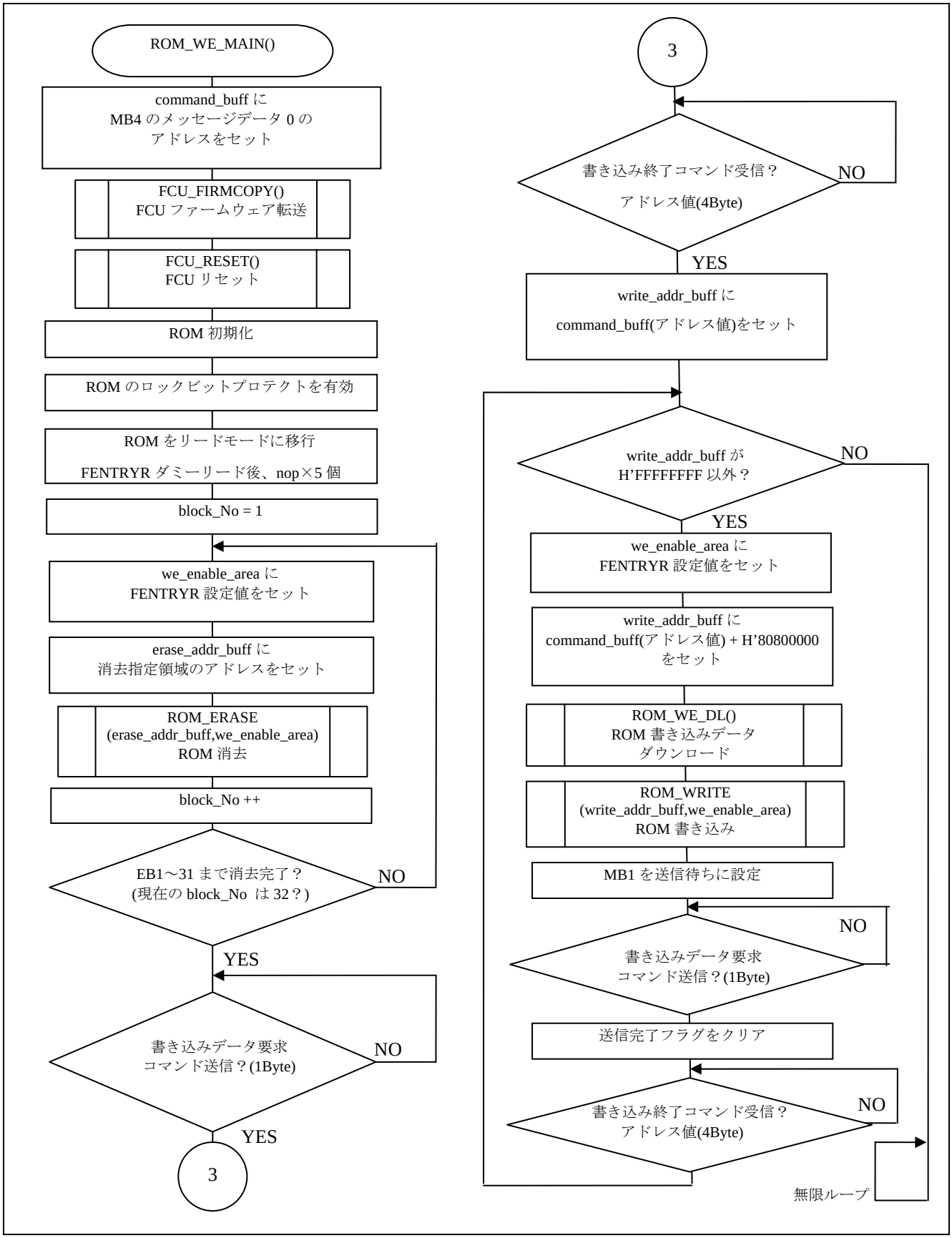


図 3-9　「ROM_WE_MAIN()関数」のフローチャート「動作手順⑤⑥⑦」

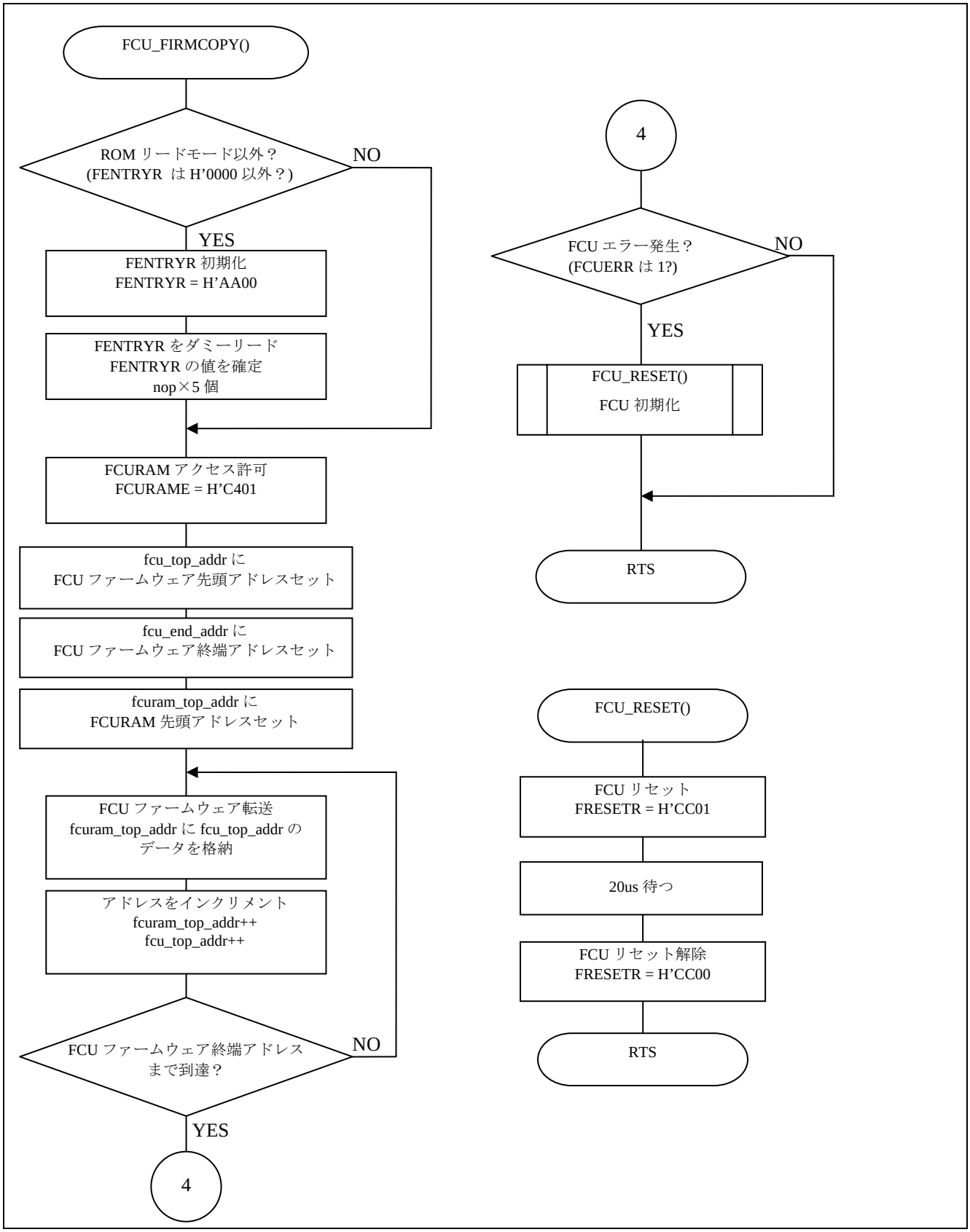


図 3-10　「FCU_FIRMCOPY()関数」「FCUCLEAR()関数」のフローチャート

「動作手順⑤」

3.2.6 ⑥ ROM 消去

ROM 初期化後、「ROM 書き換え制御プログラム(RAM)」の「ROM 消去関数」を実行。

消去対象ブロック(書き込み／消去用アドレス)に消去用 FCU コマンドを発行し、消去します。

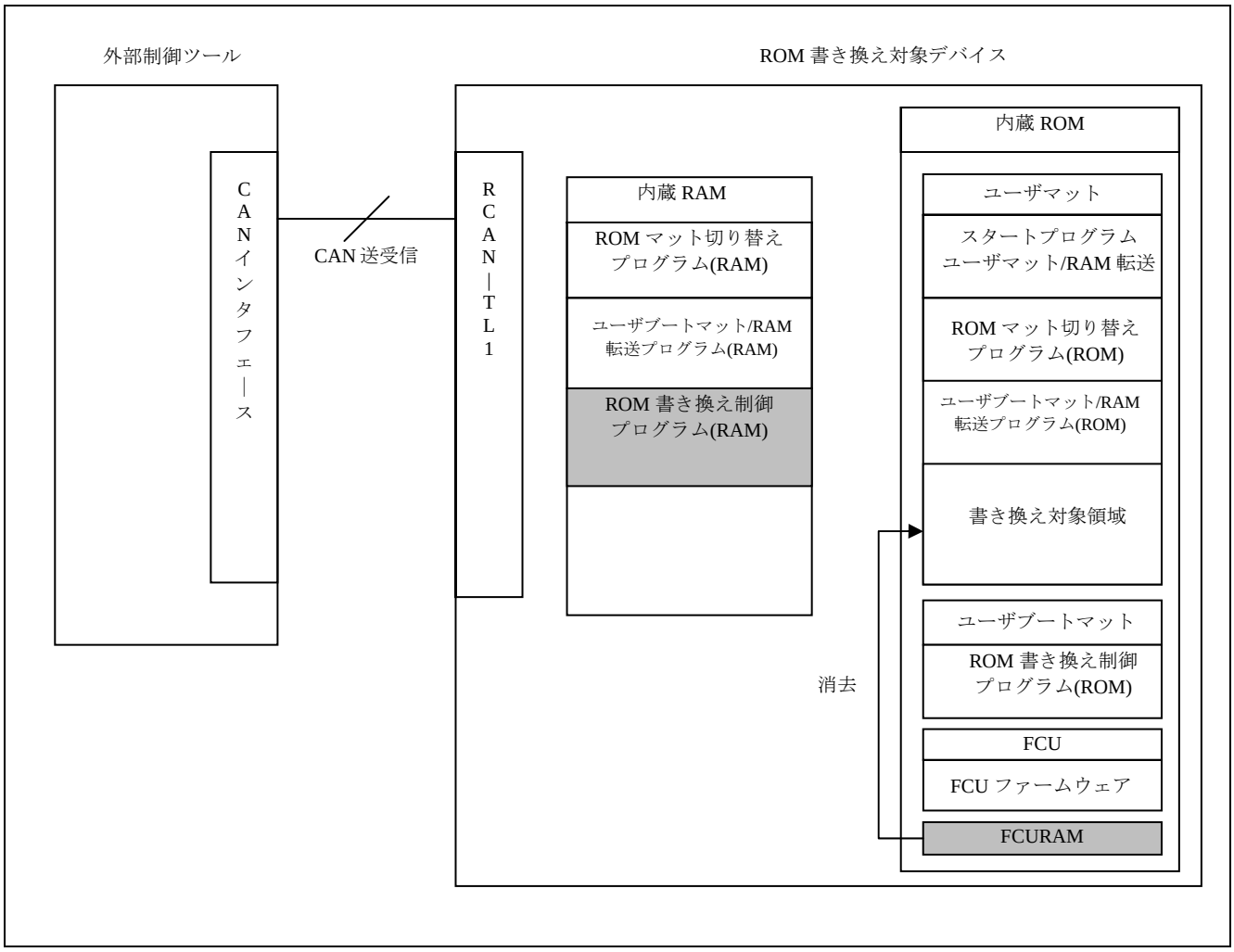


図 3-11　ROM 消去動作

「ROM_ERASE()関数」の実行

関数説明

表 3-5　「ROM_ERASE()関数」

関数名	概要
ROM_ERASE()	ROM 書き換え指定領域を消去します。

フローチャート

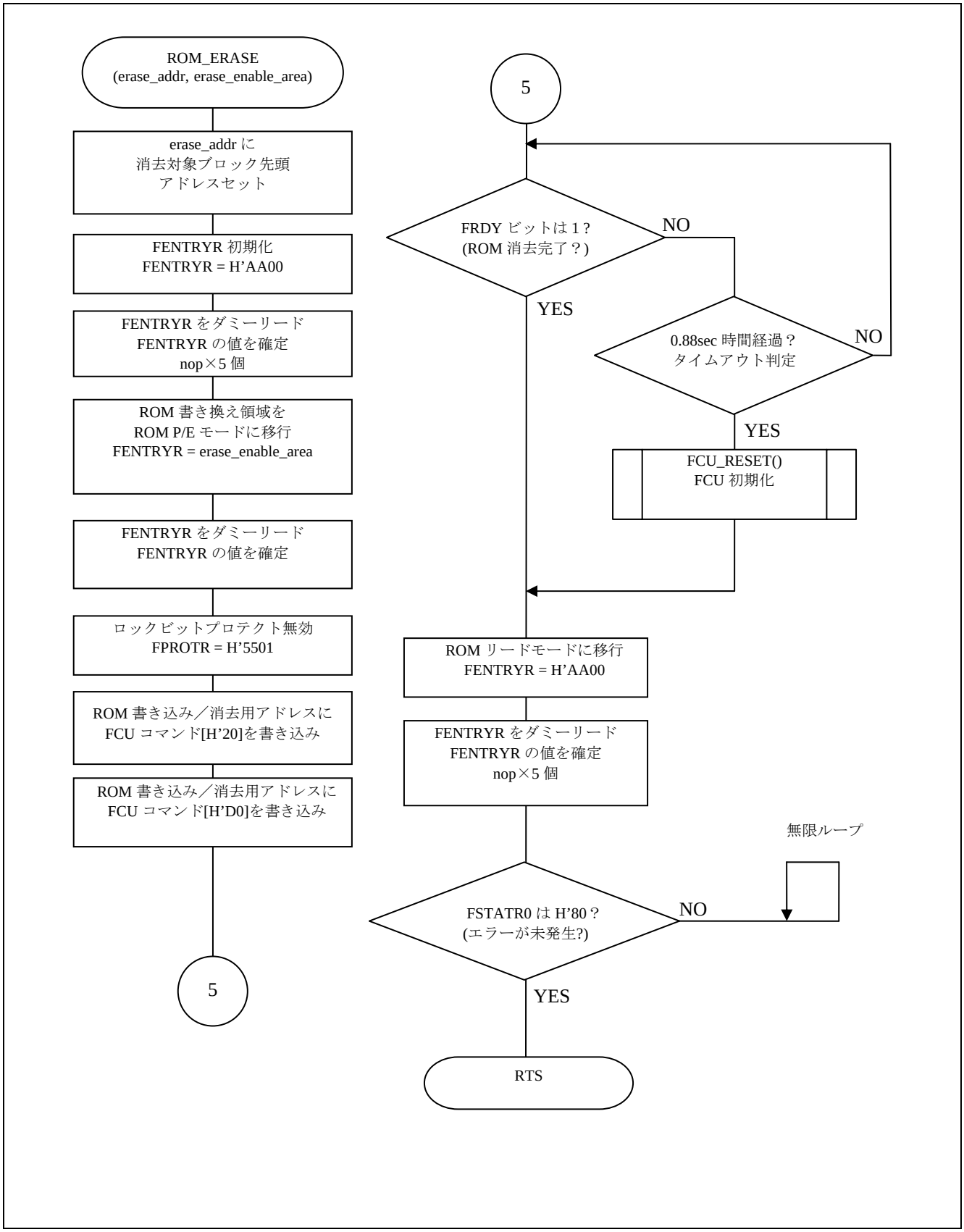


図 3-12　「ROM_ERASE()関数」のフローチャート

「動作手順⑥」

3.2.7 ⑦ ROM 書き込みデータダウンロード

CAN 通信にて①書き込み要求コマンドを送信、②書き込み終了コマンド*を受信し、外部制御ツールに書き込みデータがある場合、「ROM 書き換え制御プログラム(RAM)」の「ROM 書き込みデータダウンロード関数」を実行。

外部制御ツールから③書き込みデータ 256Byte をダウンロードします。

書き込みデータが無い場合、フラッシュ書き換えを終了します。

【注】＊ 書き込み終了コマンド(4Byte)は書き換え対象領域の書き込み先頭アドレスになります。

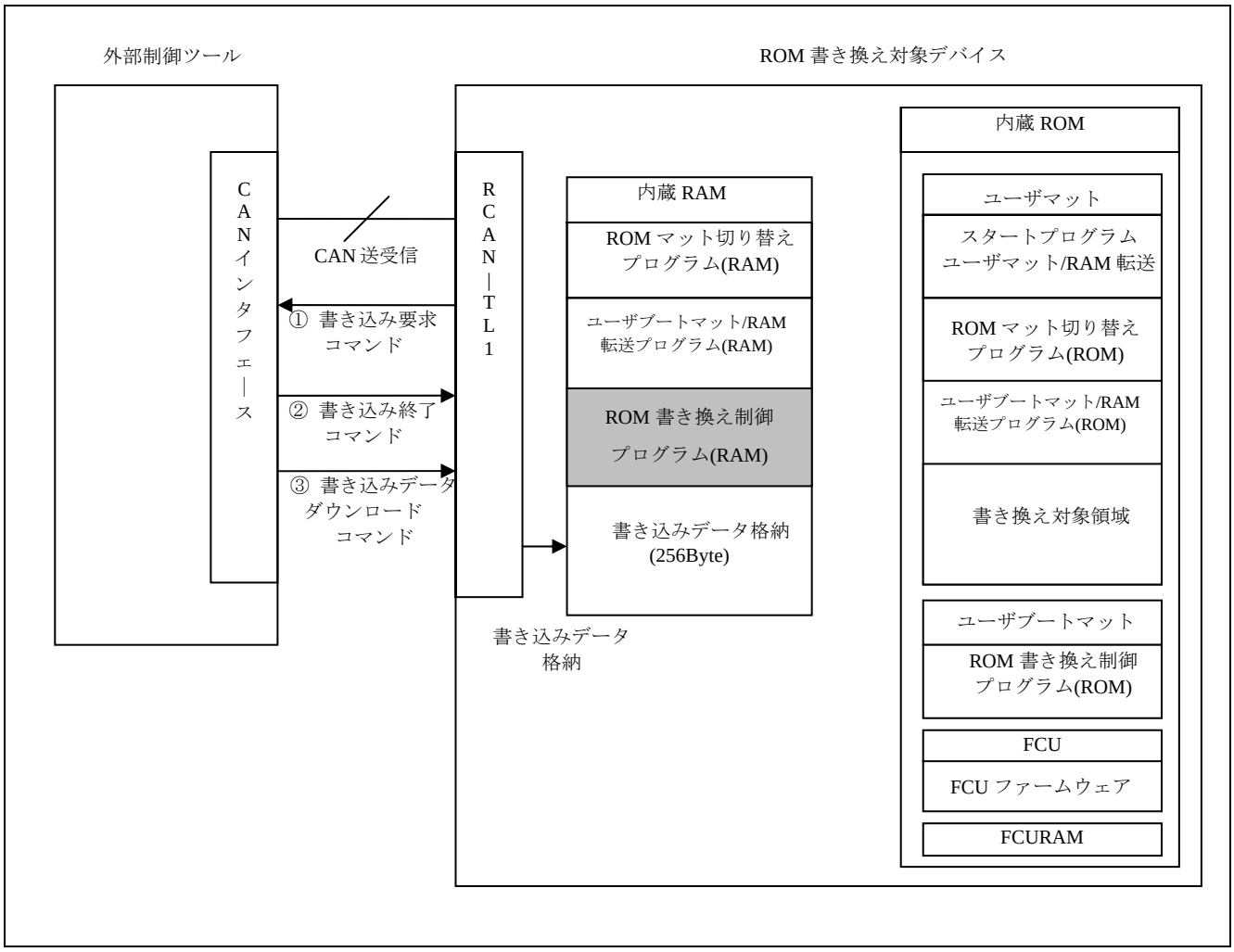


図 3-13 ROM 書き込みデータダウンロード動作

「ROM_WE_DL()関数」実行

関数説明

表 3-6 「ROM_WE_DL()関数」

関数名	概要
ROM_WE_DL()	外部制御ツールから書き込みデータをダウンロードします。

フローチャート

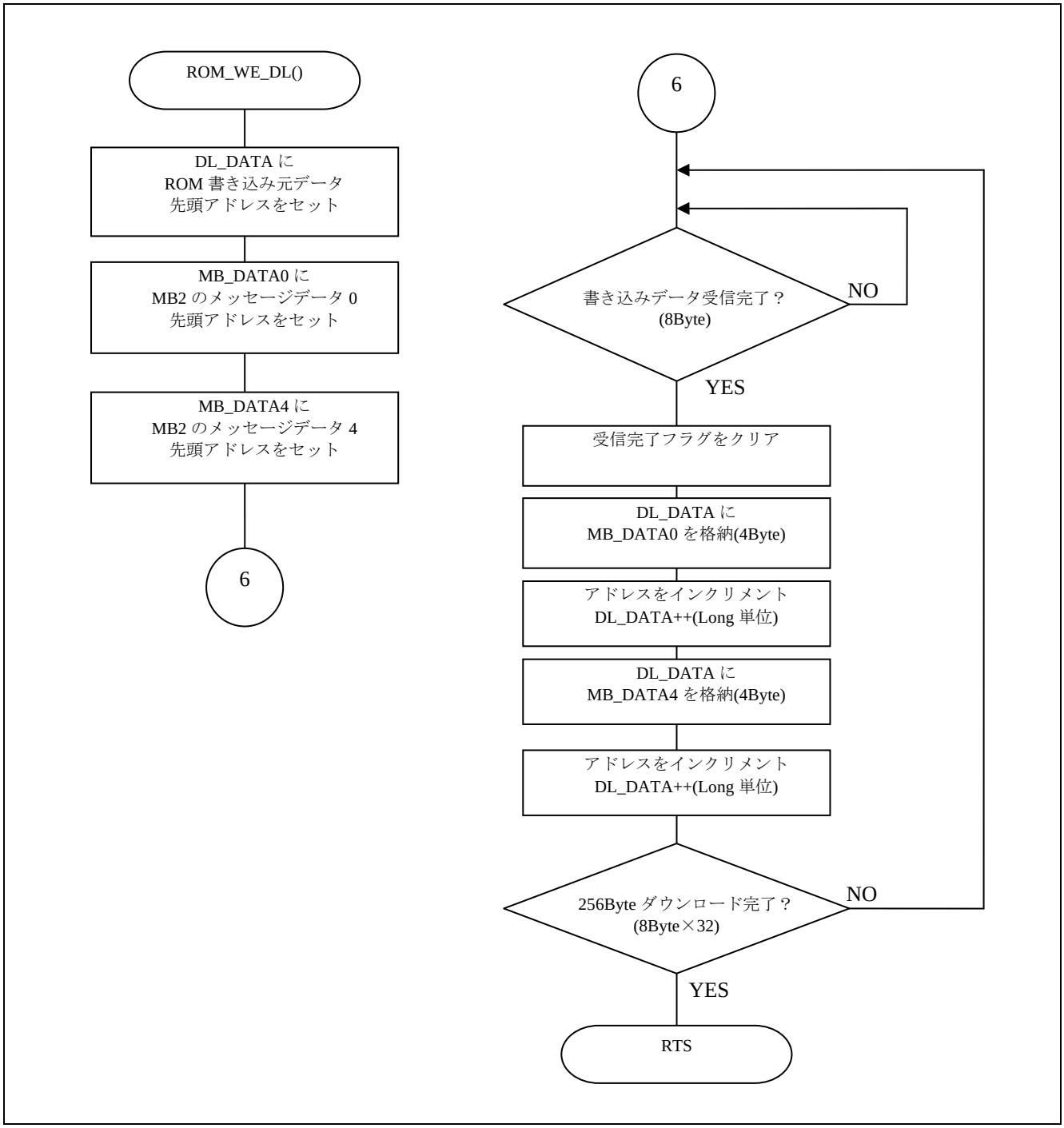


図 3-14　「ROM_WE_DL()関数」のフローチャート

「動作手順⑦」

3.2.8 ⑧ ROM 書き込み

「ROM 書き換え制御プログラム(RAM)」の「ROM 書き込み関数」を実行。

書き換え対象領域の指定領域(書き込み／消去用アドレス)に書き込み用 FCU コマンドを発行し、書き込みます。

書き込み完了後、動作⑦に移行します。

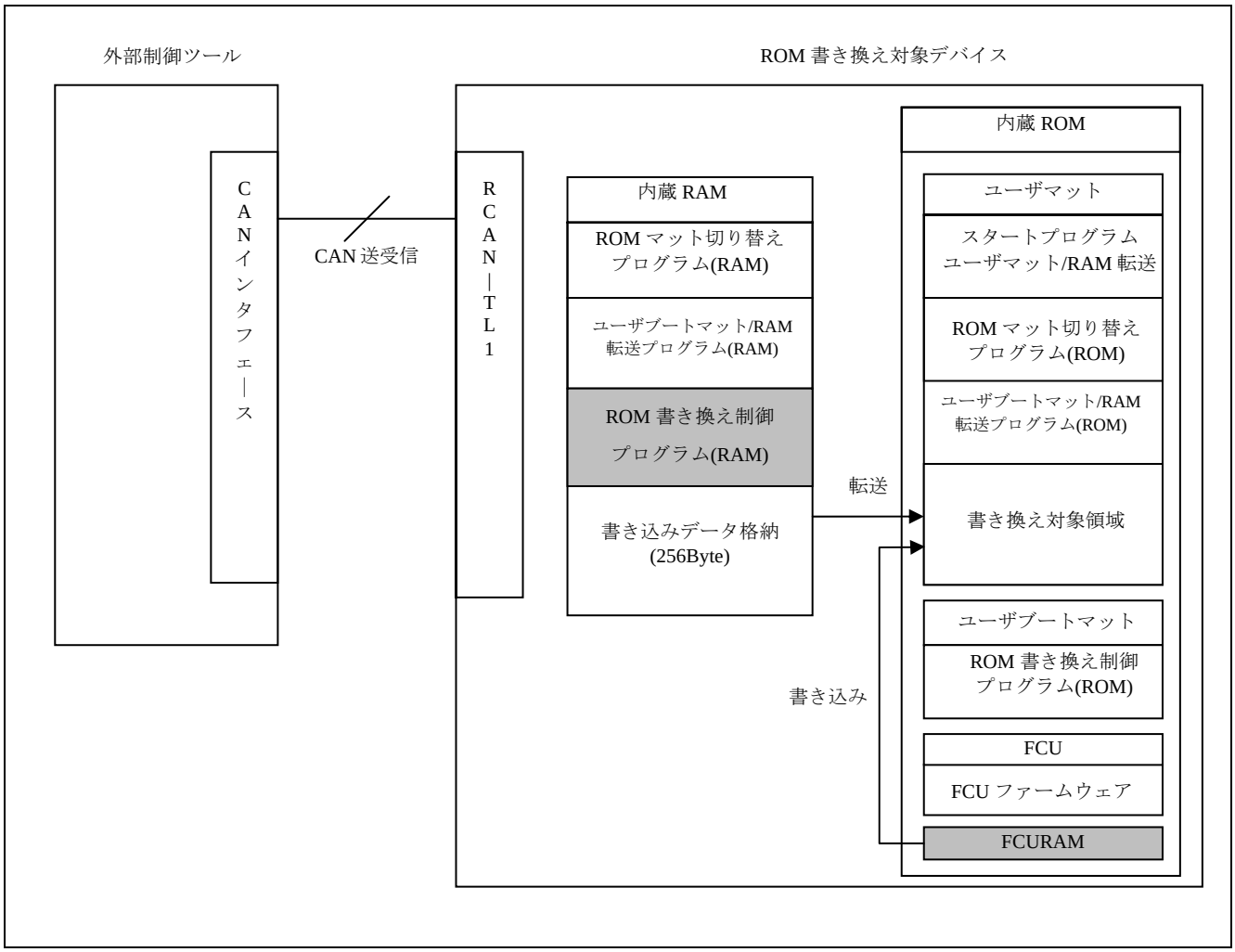


図 3-15　ROM 書き込み動作

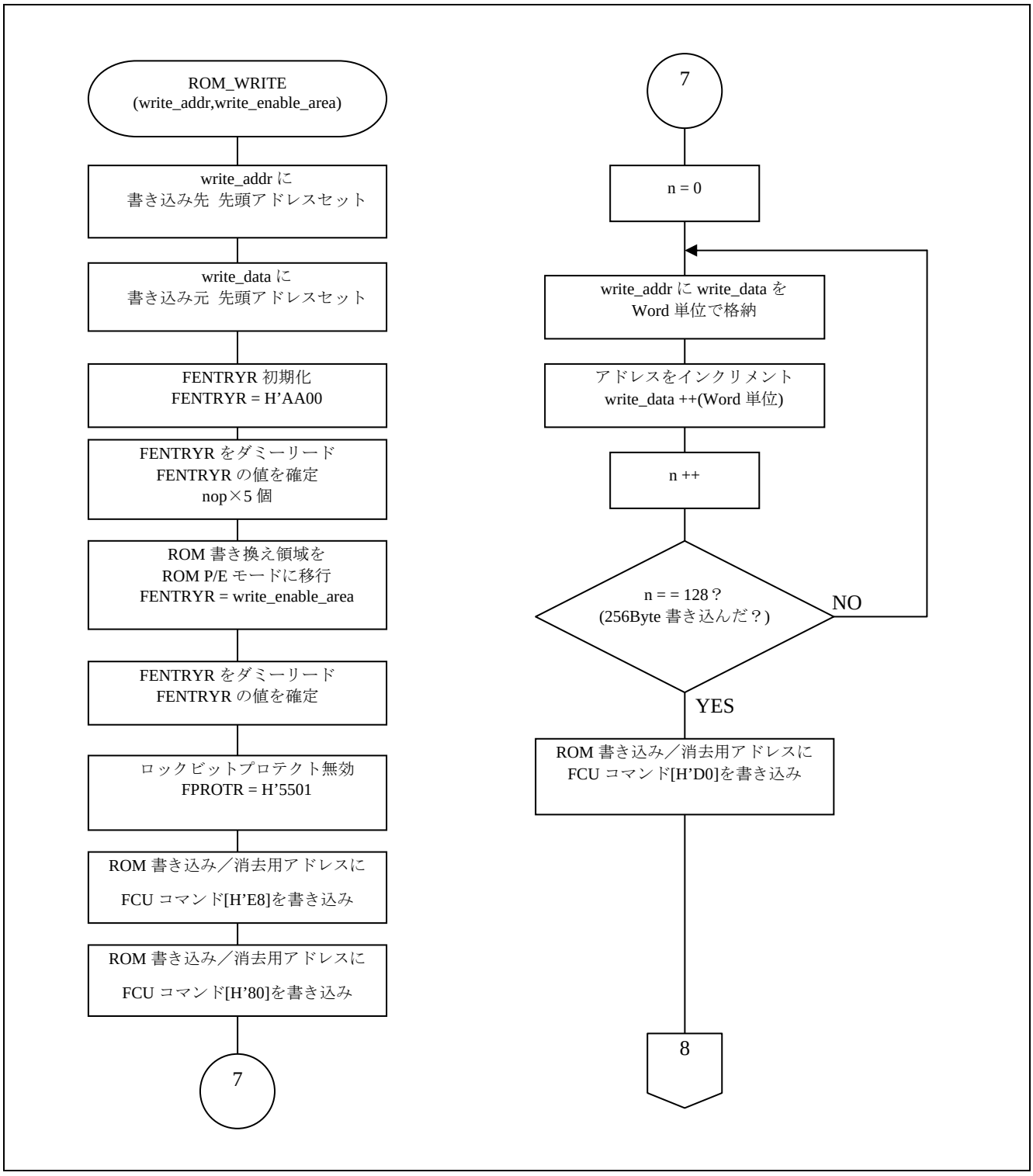
「ROM_WRITE()関数」実行

関数説明

表 3-7　「ROM_WRITE()関数」

関数名	概要
ROM_WRITE()	ROM 書き換え指定領域に書き込みます(256Byte 単位)。

フローチャート


ROM_WRITE
(write_addr,write_enable_area)
write_addr に
書き込み先 先頭アドレスセット
write_data に
書き込み元 先頭アドレスセット
FENTRYR 初期化
FENTRYR = H'AA00
FENTRYR をダミーリード
FENTRYR の値を確定
nop×5 個
ROM 書き換え領域を
ROM P/E モードに移行
FENTRYR = write_enable_area
FENTRYR をダミーリード
FENTRYR の値を確定
ロックビットプロテクト無効
FPROTR = H'5501
ROM 書き込み／消去用アドレスに
FCU コマンド[H'E8]を書き込み
ROM 書き込み／消去用アドレスに
FCU コマンド[H'80]を書き込み
7
7
n = 0
write_addr に write_data を
Word 単位で格納
アドレスをインクリメント
write_data ++(Word 単位)
n ++
n = = 128？
(256Byte 書き込んだ？)
NO
YES
ROM 書き込み／消去用アドレスに
FCU コマンド[H'D0]を書き込み
8

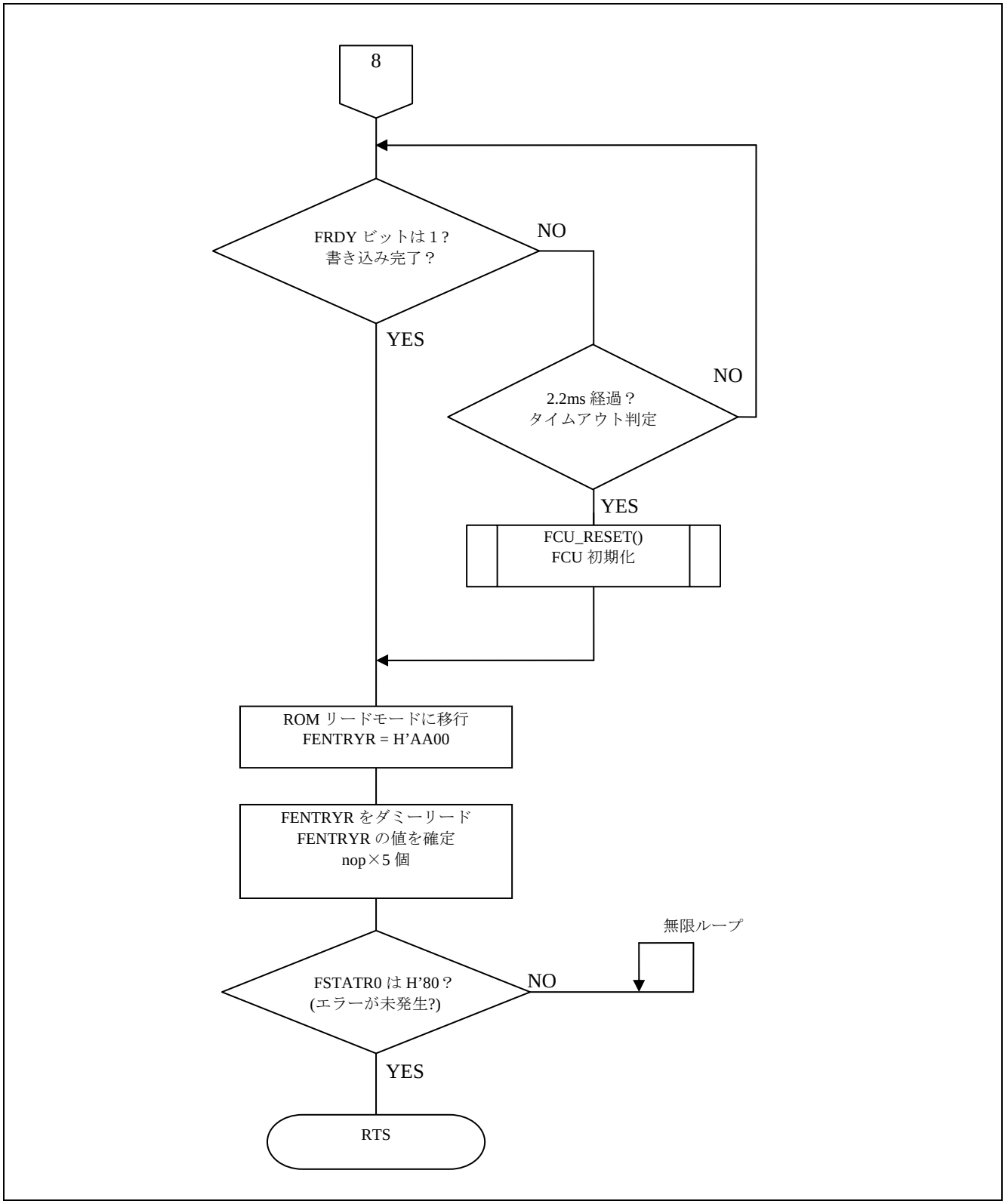
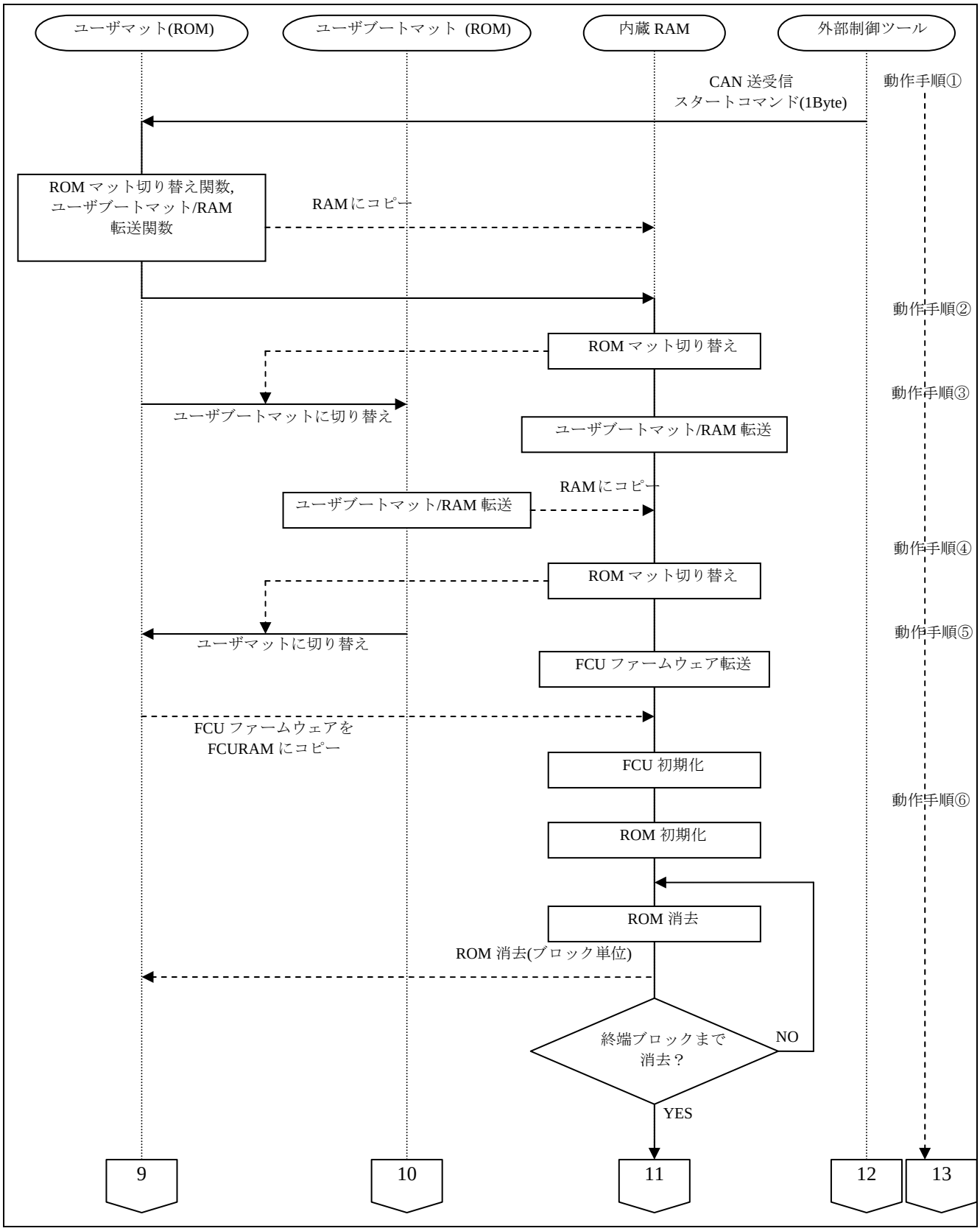


図 3-16　「ROM_WRITE()関数」のフローチャート

「動作手順⑧」

3.3 全体シーケンス

全体シーケンスを図 3-17 に示します。3-2 動作手順に対応します。



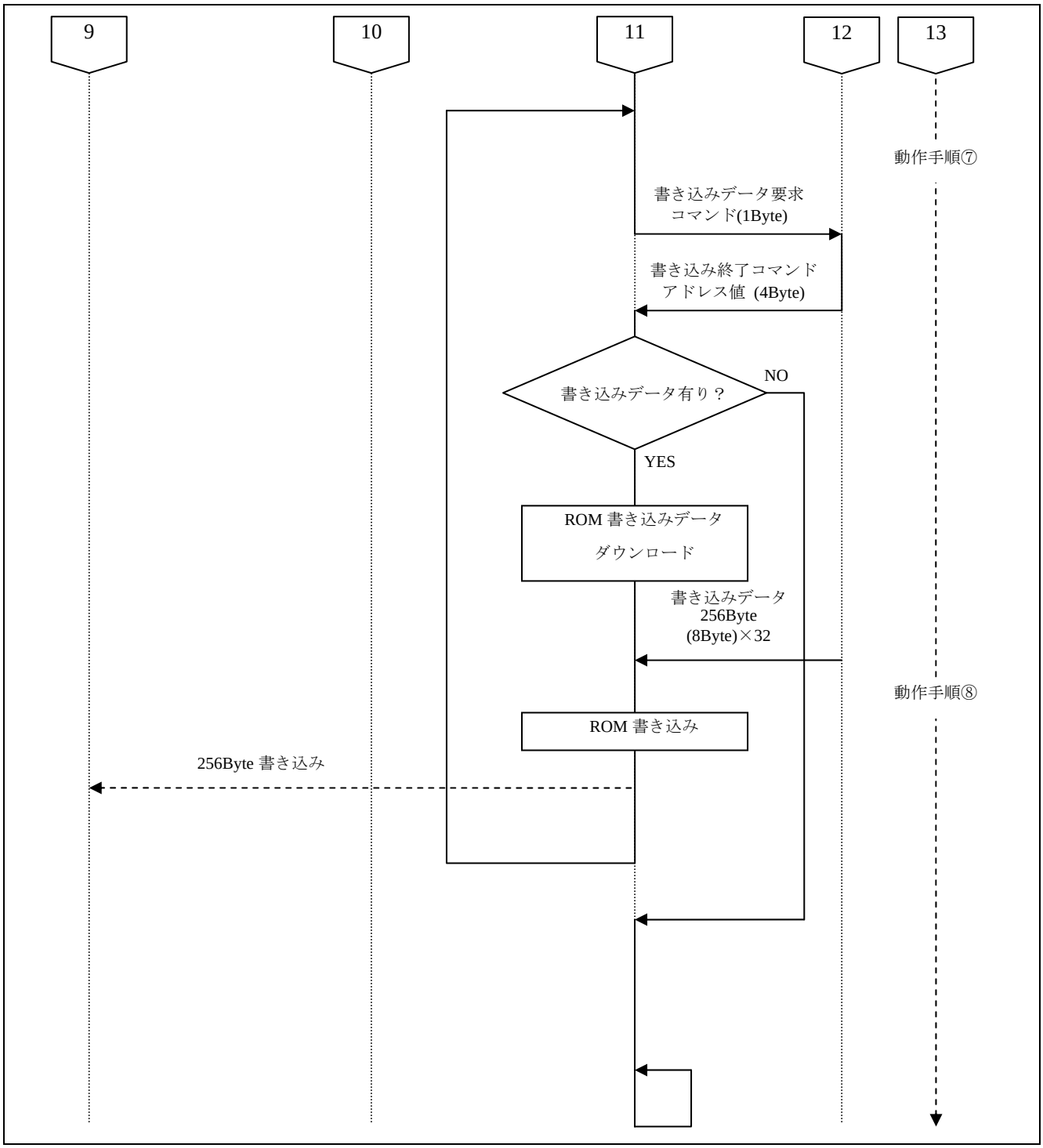


図 3-17　全体シーケンス図

4. 詳細仕様

4.1 セクション設定

4.1.1 セクション配置表

セクション配置表を表 4-1 に示します。

表 4-1 セクション配置表

ユーザマット(セクション)

0x00000000	DVECTTBL
	DINTTBL
0x00000800	PResePRG
	PIntPRG
0x00001000	P
	C
	C$BSEC
	C$DSEC
	D
	PMAT_C
0xFFF80000	B
	R
	RMAT_C
0xFFFBFC00	S

ユーザブートマット(セクション)

0x00000000	PROM_WE
	DROM_WE
0xFFF81000	RROM_WE
	RDROM_WE
	BROM_WE

セクション説明

使用しているセクション内の関数を表 4-2 に示します。

表 4-2 セクション説明

セクション名	セクション説明
PMAT_C	ROM マット切り替え関数。
RMAT_C	RAM 上で行う ROM マット切り替え関数。
PROM_WE	ROM 書き換え制御関数、FCU ファームウェア転送関数、FCU 初期化関数、ROM 消去関数、ROM 書き込みダウンロード関数、ROM 書き込み関数。
RROM_WE	RAM 上で行う、ROM 書き換え制御関数、FCU ファームウェア転送関数、FCU 初期化関数、ROM 消去関数、ROM 書き込みダウンロード関数、ROM 書き込み関数。

4.1.2 ROM/RAM セクションリンク設定

ROM/RAM 転送を行うときに転送元と転送先のセクションをリンクします。(転送元と転送先のアドレスがずれるのを防ぐため)ROM/RAM セクションリンク設定を表 4-3 に示します。

表 4-3　ROM/RAM セクションリンク設定

ユーザマット(セクション)

ROM to RAM mapped sections	
Rom	Ram
D	R
PMAT_C	RMAT_C

ユーザブートマット(セクション)

ROM to RAM mapped sections	
Rom	Ram
PROM_WE	RROM_WE
DROM_WE	RDROM_WE

4.2 CAN 設定

4.2.1 CAN 通信設定

- PJ0(CTxA)を出力端子、PJ1(CRxA)を入力端子に設定します。
- ビットレートを 500kbps に設定します。
- MB0(1Byte)、MB2(8Byte)、MB4(4Byte)を受信メールボックス、MB1(1Byte)を送信メールボックスに設定します。

4.2.2 CAN コマンド設定

フラッシュ書き換えの中で使われている CAN コマンドを表 4-4 に示します。

表 4-4　書き換え対象デバイスの CAN コマンド設定

MB 番号	コマンド名	送/受	スタンダード ID	データ長	データ
MB0	スタートコマンド	受信	H’100	1Byte	H’11
MB1	書き込みデータ要求コマンド	送信	H’101	1Byte	H’22
MB4	書き込み終了コマンド	受信	H’131	4Byte	ROM 書き換え領域のアドレス値
MB2	書き込みデータダウンロードコマンド	受信	H’111	8Byte	ROM 書き込みデータ 256Byte をダウンロード(8Byte×32)

【注】* 書き込み終了コマンドは ROM 書き換え領域の書き込み先頭アドレスをデータとして受信し 256Byte 書き換えます。
H’FFFF FFFF のデータを受信したとき、ROM 書き換えを終了します。

4.2.3 レジスタ説明

CAN 通信で使用するレジスタの説明を以下に示します。

マスタコントロールレジスタ（MCR）

MCR は、16 ビットの読み出し／書き込み可能なレジスタで、RCAN-TL1 を制御します。

ジェネラルステータスレジスタ（GSR）

GSR は、16 ビットの読み出し専用レジスタで、RCAN-TL1 の状態を示します。

インタラプトリクエストレジスタ（IRR）

IRR は、16 ビットの読み出し／書き込み可能なレジスタで、各種割り込み要因のステータスフラグで構成されています。

ビットコンフィギュレーションレジスタ 0、1（BCR0、BCR1）

BCR0、BCR1 は、それぞれ 16 ビットの読み出し／書き込み可能なレジスタで、CAN ビットタイミングパラメータと CAN インタフェースのボーレートプリスケーラを設定します。

データフレーム受信完了レジスタ 1、0（RXPR1、RXPR0）

RXPR1 と RXPR0 は、16 ビットの読み出し／条件付き書き込み可能なレジスタで、受信用に設定されたメールボックスがデータフレームを受信したことを示すフラグで構成されています。CAN データフレームが正常に受信メールボックスに格納されると、RXPR の対応するビットがセットされます。1 を書き込むと対応するビットがクリアされます。0 を書き込むと無効とされます。ただし、メールボックスが MBC（メールボックスコンフィギュレーション）によってデータフレームを受信するように設定されている場合のみビットがセットされます。RXPR のビットがセットされると、対応する MBIMR がセットされていなければ IRR1（データフレーム受信割り込みフラグ）もセットされ、さらに IMR1 がセットされていなければ割り込み信号が生成されます。本レジスタのビットはデータフレームの受信によってのみセットされ、リモートフレーム受信ではセットされません。

送信待ちレジスタ 1、0（TXPR1、TXPR0）

TXPR1 と TXPR0 は連結され、CAN モジュールの送信待ちフラグを格納する 32 ビットのレジスタを構成します。16 ビットバスインタフェースの場合、ロングワードアクセスは、2 つの連続したワードアクセスとして行われます。

送信アクノリッジレジスタ 0（TXACK0）

TXACK1 と TXACK0 は、16 ビットの読み出し／条件付き書き込み可能なレジスタで、メールボックスの送信が正常に行われたことを CPU に通知するために使用します。送信が正常に行われると、RCAN-TL1 は TXACK レジスタの対応するビットをセットします。CPU は、1 を書き込むことによって TXACK のビットをクリアすることができます。0 を書き込むと無視されます。

4.3 ROM/RAM アドレス表

ROM(ユーザマット、ユーザブートマット)、RAM のアドレス表を図 4-1 に示します。
(各プログラムサイズも記入)

EB は消去対象ブロックで、書き換え対象領域には EB1～31(H'00002000～H'0027FFFF)が該当します。

注：下図のアドレス表は本プログラム仕様であって、アドレス、サイズは場合により異なります。

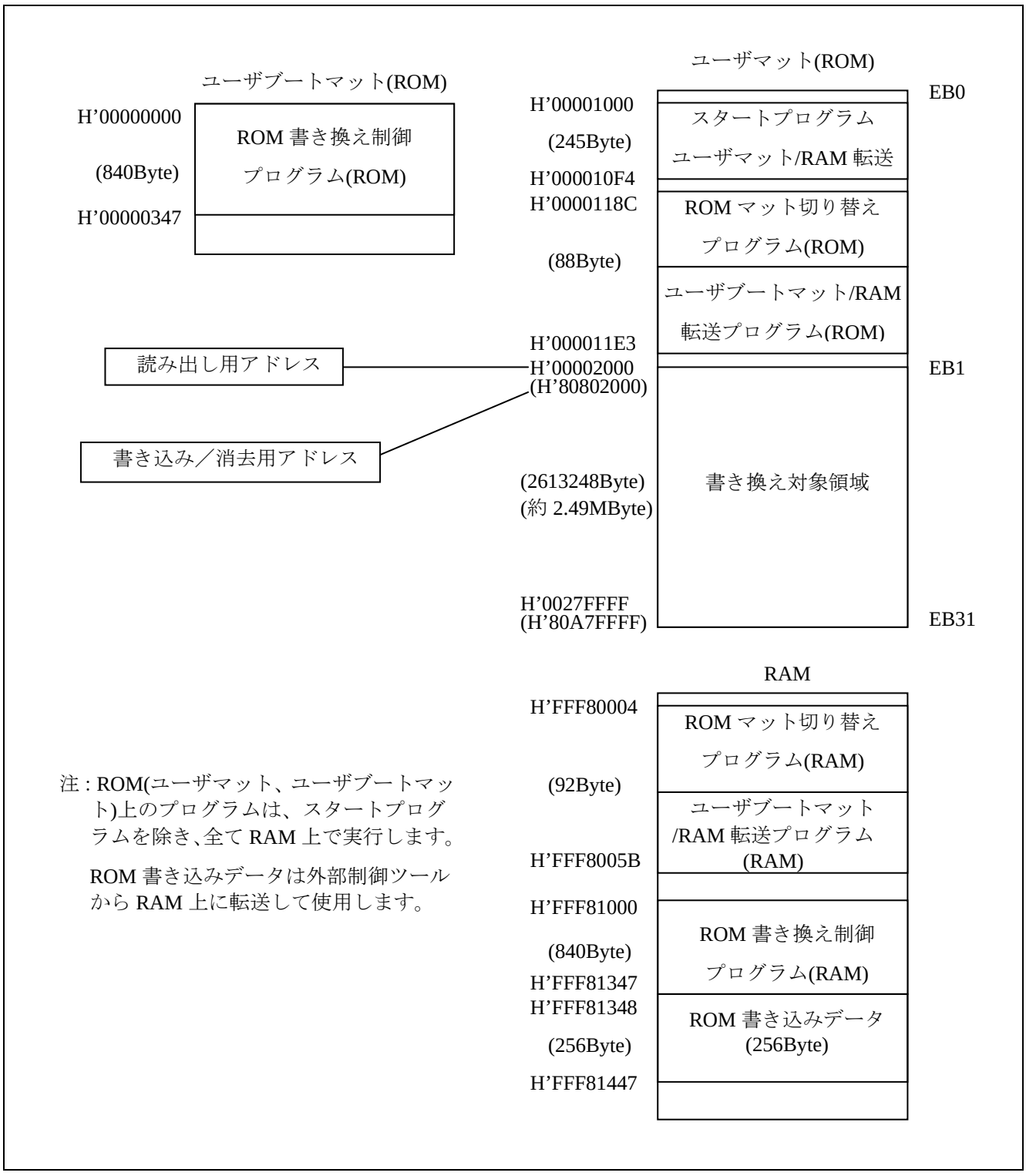


図 4-1　ROM/RAM アドレス

5. サンプルプログラム

5.1 ユーザマットプログラム

```
1   /***********************************************************************
2   * DISCLAIMER
3   * This software is supplied by Renesas Electronics Corporation and is only
4   * intended for use with Renesas products. No other uses are authorized. This
5   * software is owned by Renesas Electronics Corporation and is protected under
6   * all applicable laws, including copyright laws.
7   * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES REGARDING
8   * THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT
9   * LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
10  * AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY DISCLAIMED.
11  * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
12  * ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
13  * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR
14  * ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS AFFILIATES HAVE
15  * BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
16  * Renesas reserves the right, without notice, to make changes to this software
17  * and to discontinue the availability of this software. By using this software,
18  * you agree to the additional terms and conditions found by accessing the
19  * following link:
20  * http://www.renesas.com/disclaimer *
21  * Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
22  ***********************************************************************/
23  /***********************************************************************
24  * File Name    : SH7254R_USERMAT.c
25  * Version      : 1.00
26  * Device(s)    : SH72544R
27  * Tool-Chain   : High-performance Embedded Workshop (Ver.4.08.00).
28  * OS           : None
29  * H/W Platform : SH7254R
30  * Description  : This is the main tutorial code.
31  * Operation    : USERMAT
32  * Limitations  : None
33  ***********************************************************************/
34  /***********************************************************************
35  * History : DD.MM.YYYY Version Description
36  *         : 23.12.2011 1.00    First Release
37  ***********************************************************************/
38  /***********************************************************************
39  Includes   <System Includes> , "Project Includes"
40  ***********************************************************************/
41  #include "iodefine.h" /* 周辺 ﾚｼﾞｽﾀ定義ﾍｯﾀﾞﾌｧｲﾙ */
42  #include <machine.h> /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ */
43  #include "include.h" /*  ｱﾄﾞﾚｽ,ﾃﾞｰﾀ定義ﾍｯﾀﾞﾌｧｲﾙ */
44  /***********************************************************************
45  Private global variables and functions
46  ***********************************************************************/
47  /***********************************************************************/
```

```
48    void main(void); /* ﾕｰｻﾞﾏｯﾄ(ROM)/RAM 転送関数 */
49    void ROM_RAM_T(void); /* ﾕｰｻﾞﾌﾞｰﾄﾏｯﾄ(ROM)/RAM 転送関数 */
50    void MATC(void); /* ROM ﾏｯﾄ切り替え関数 */
51    void (*Addr)(void) = {(void(*)(void))0xFFF81000}; /* RAM 上の ROM 書き換え制御関数先頭
      ｱﾄﾞﾚｽ */
52
53    /***********************************************************************************/
54    /* ﾌﾟﾛｸﾞﾗﾑｽﾀｰﾄ ﾕｰｻﾞﾏｯﾄ(ROM)/RAM 転送関数) */
55    /***********************************************************************************/
56    /***********************************************************************************
57    * Function Name: main
58    * Description  : The main loop
59    * Arguments    : none
60    * Return Value : none
61    ***********************************************************************************/
62    void main(void)
63    {
64
65    unsigned long i;
66    unsigned char *rom_flash;
67    unsigned char *ram_flash;
68    unsigned char *command1;
69    unsigned short *work;
70
71    command1 = MB0_DATA0_ADDR; /* command1 にﾒｰﾙﾎﾞｯｸｽ 0 のﾒｯｾｰｼﾞﾃﾞｰﾀ 0 先頭ｱﾄﾞﾚｽをｾｯﾄ */
72
73    /***********************************************************************************/
74    /* CAN 初期設定 */
75    /***********************************************************************************/
76    /* ﾎﾟｰﾄ設定 */
77    /* Configure PJCR1
78    b15,14 PJ7MD[1:0] = 0 PJ7 入出力（ポート）
79    b13 reserved
80    b12 PJ6MD = 0 PJ6 入出力（ポート）
81    b11 reserved
82    b10 PJ5MD = 0 PJ5 入出力（ポート）
83    b9,8 PJ4MD[1:0] = 0 PJ4 入出力（ポート）
84    b7,6 PJ3MD[1:0] = 0 PJ3 入出力（ポート）
85    b5,4 PJ2MD[1:0] = 0 PJ2 入出力（ポート）
86    b3,2 PJ1MD[1:0] = 2 CRx_A 入力（RCAN-TL1）
87    b1,0 PJ0MD[1:0] = 2 CTx_A 出力（RCAN-TL1）*/
88    PORTJ.CR1.WORD = 0x000A; /* PJ0 は CTx_A 出力端子,PJ1 は CRx_A 入力端子に設定 */
89
90    /* Configure PJIOR
91    b15-10 reserved
92    b9 PJ9IOR = 0 PJ9 入力
93    b8 PJ8IOR = 0 PJ8 入力
94    b7 PJ7IOR = 0 PJ7 入力
95    b6 PJ6IOR = 0 PJ6 入力
96    b5 PJ5IOR = 0 PJ5 入力
```

```
97    b4 PJ4IOR = 0 PJ4 入力
98    b3 PJ3IOR = 0 PJ3 入力
99    b2 PJ2IOR = 0 PJ2 入力
100   b1 PJ1IOR = 0 PJ1 入力
101   b0 PJ0IOR = 1 PJ0 出力*/
102   PORTJ.IOR.WORD = 0x0001; /* PJ0(CTx_A)を出力,PJ1(CRx_A)を入力に設定 */
103
104   /* CAN 初期設定 */
105   /* Configure MCR
106   b15 MCR15 = 0 RCAN-TL1 と HCAN2 は同等の順序)
107   b14 MCR14 = 0 通常の復帰シーケンス（128×11 レセッシブビット）で RCAN-TL1 バスオフ状態を維持
108   b13-11 reserved
109   b10-8 TST[2:0] = 0 ノーマルモード
110   b7 MCR7 = 0 CAN バスのアクティビティによる自動ウェイクモードが無効
111   b6 MCR6 = 0 バスオフ時にホルトモードには入らず、復帰シーケンスが終了するのを待ちます
112   b5 MCR5 = 0 PH2 入出力（ポート）CAN スリープモードが解除されています
113   b4,3 reserved
114   b2 MCR2 = 0 メッセージ ID 優先順に送信
115   b1 MCR1 = 0 ホルトモードリクエストをクリア
116   b0 MCR0 = 1 CAN インタフェースのリセットモード遷移リクエスト*/
117   RCANA.MCR.WORD |= 0x0001; /* ﾘｾｯﾄﾘｸｴｽﾄ(HW ﾘｾｯﾄ時は自動的にｾｯﾄ) */
118   while((RCANA.GSR.WORD & 0x0008) != 0x0008); /* GSR3=1?(RCAN-ET ﾘｾｯﾄ状態) */
119   while((RCANA.IRR.WORD & 0x0001) != 0x0001); /* IRR0=1?(ﾘｾｯﾄ/ﾎﾙﾄ/ｽﾘｰﾌﾟ 割り込み) */
120
121   /* Configure IRR
122   b15 IRR15 = 0 TCMR1 のタイマコンペアマッチが発生していない
123   b14 IRR14 = 0 TCMR0 のタイマコンペアマッチが発生していない
124   b13 IRR13 = 0 イベントトリガモード（テストモードを含む）でタイマ（TCNTR）オーバランが発生していない
125   b12 IRR12 = 0 バスアイドル状態
126   b11 IRR11 = 0 TCMR2 のタイマコンペアマッチが発生していない
127   b10 IRR10 = 0 新しいシステムマトリックスの先頭でない
128   b9 IRR9 = 0 メッセージオーバラン／オーバライト通知がない
129   b8 IRR8 = 0 送信または送信キャンセルするメッセージが処理中でない
130   b7 IRR7 = 0 ［クリア条件］1 を書き込む
131   b6 IRR6 = 0 ［クリア条件］1 を書き込む
132   b5 IRR5 = 0 ［クリア条件］1 を書き込む
133   b4 IRR4 = 0 ［クリア条件］1 を書き込む
134   b3 IRR3 = 0 ［クリア条件］1 を書き込む
135   b2 IRR2 = 0 ［クリア条件］RFPR のすべてのビットがクリア
136   b1 IRR1 = 0 ［クリア条件］RXPR のすべてのビットがクリア
137   b0 IRR0 = 1 ソフトウェアリセットモードまたはホルトモードまたは CAN スリープモードへ遷移*/
138   RCANA.IRR.WORD = 0x0001; /* IRR0 ｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
139
140   /* Configure MCR
141   b15 MCR15 = 1 RCAN-TL1 と HCAN2 は異なる順序
142   b14 MCR14 = 0 通常の復帰シーケンス（128×11 レセッシブビット）で RCAN-TL1 バスオフ状態を維持
143   b13-11 reserved
144   b10-8 TST[2:0] = 0 ノーマルモード
145   b7 MCR7 = 0 CAN バスのアクティビティによる自動ウェイクモードが無効
```

```
146   b6 MCR6 = 0 バスオフ時にホルトモードには入らず、復帰シーケンスが終了するのを待ちます
147   b5 MCR5 = 0 PH2 入出力（ポート）CAN スリープモードが解除されています
148   b4,3 reserved
149   b2 MCR2 = 1 メールボックス番号順（メールボックス 31→メールボックス 1）に送信
150   b1 MCR1 = 0 ホルトモードリクエストをクリア
151   b0 MCR0 = 0 リセットモードリクエストをクリア*/
152   RCANA.MCR.WORD |= 0x8004; /* ID 並び替え：MCR15=1(初期値)に設定,:MCR2=1 に設定 */
153
154   /* ﾒｰﾙﾎﾞｯｸｽ(RAM ｴﾘｱ)初期化 */
155   i=0;
156   work = (unsigned short *)0xFFFFD100;
157   do
158   {
159   if(i>=8)
160   {
161   i=0;
162   work += 8;
163   }
164   *work = 0;
165   work++;
166   i++;
167   }while(work<(unsigned short *)0xFFFFD4F0);
168
169
170   /* MB0(受信)設定 */
171   /* Configure CONTROL0
172   b15 IDE = 0 スタンダードフォーマット
173   b14 RTR = 0 データフレーム
174   b13 reserved
175   b12-2 STDID[10:0] = H'4 スタンダード ID
176   b1,0 EXTID[17:16] = 0 エクステンデッド ID
177   b15-0 EXTID[15:0] = 0 エクステンデッド ID*/
178   RCANA.MSG[0].CONTROL0.LONG = MB0_ID; /* ID:100,ｽﾀﾝﾀﾞｰﾄﾞﾌｫｰﾏｯﾄ,ﾃﾞｰﾀﾌﾚｰﾑ */
179
180   /* Configure LAFM
181   b15 IDE_LAFM = 0 対応する IDE ビットが有効
182   b14,13 reserved
183   b12-2 STDID_LAFM[10:0] = 0 対応する STDID ビットが有効
184   b1,0 EXTID_LAFM[17:16] = 0 対応する EXTID ビットが有効
185   b15-0 EXTID_LAFM[15:0] = 0 対応する EXTID ビットが有効*/
186   RCANA.MSG[0].LAFM.LONG = 0x00000000; /* STD_LAFM, IDE_LAFM の設定 */
187
188   /* Configure CONTROL1
189   b15,14 reserved
190   b13 NMC = 0 オーバランモード
191   b12 ATX = 0 データフレームの自動送信無効
192   b11 DART = 0 再送信有効
193   b10-8 MBC[2:0] = H'2 メールボックスコンフィギュレーション
194   b7-4 reserved
195   b3-0 DLC[3:0] = 0 データ長コード*/
```

```
196   RCANA.MSG[0].CONTROL1.WORD = 0x0200; /* MB0 を受信用に設定 */
197
198   /* MB1(送信)設定 */
199   /* Configure CONTROL0
200   b15 IDE = 0 スタンダードフォーマット
201   b14 RTR = 0 データフレーム
202   b13 reserved
203   b12-2 STDID[10:0] = H'404 スタンダード ID
204   b1,0 EXTID[17:16] = 0 エクステンデッド ID
205   b15-0 EXTID[15:0] = 0 エクステンデッド ID*/
206   RCANA.MSG[1].CONTROL0.LONG = MB1_ID; /* ID:101,ｽﾀﾝﾀﾞｰﾄﾞﾌｫｰﾏｯﾄ,ﾃﾞｰﾀﾌﾚｰﾑ */
207
208   /* Configure CONTROL1
209   b15,14 reserved
210   b13 NMC = 0 オーバランモード
211   b12 ATX = 0 データフレームの自動送信無効
212   b11 DART = 0 再送信有効
213   b10-8 MBC[2:0] = 0 メールボックスコンフィギュレーション
214   b7-4 reserved
215   b3-0 DLC[3:0] = 1 データ長 1 バイト*/
216   RCANA.MSG[1].CONTROL1.WORD = 0x0001; /* MB1 を送信用に設定,データ長：1ﾊﾞｲﾄ設定 */
217   RCANA.MSG[1].DATA.BYTE[0] = WRITE_DATA_CMD1; /* 送信ﾃﾞｰﾀ:0x22 */
218
219   /* MB2(受信)設定 */
220   /* Configure CONTROL0
221   b15 IDE = 0 スタンダードフォーマット
222   b14 RTR = 0 データフレーム
223   b13 reserved
224   b12-2 STDID[10:0] = H'444 スタンダード ID
225   b1,0 EXTID[17:16] = 0 エクステンデッド ID
226   b15-0 EXTID[15:0] = 0 エクステンデッド ID*/
227   RCANA.MSG[2].CONTROL0.LONG = MB2_ID; /* ID:111,ｽﾀﾝﾀﾞｰﾄﾞﾌｫｰﾏｯﾄ,ﾃﾞｰﾀﾌﾚｰﾑ */
228
229   /* Configure LAFM
230   b15 IDE_LAFM = 0 対応する IDE ビットが有効
231   b14,13 reserved
232   b12-2 STDID_LAFM[10:0] = 0 対応する STDID ビットが有効
233   b1,0 EXTID_LAFM[17:16] = 0 対応する EXTID ビットが有効
234   b15-0 EXTID_LAFM[15:0] = 0 対応する EXTID ビットが有効*/
235   RCANA.MSG[2].LAFM.LONG = 0x00000000; /* STD_LAFM, IDE_LAFM の設定 */
236
237   /* Configure CONTROL1
238   b15,14 reserved
239   b13 NMC = 0 オーバランモード
240   b12 ATX = 0 データフレームの自動送信無効
241   b11 DART = 0 再送信有効
242   b10-8 MBC[2:0] = H'2 メールボックスコンフィギュレーション
243   b7-4 reserved
244   b3-0 DLC[3:0] = 1 データ長 1 バイト*/
245   RCANA.MSG[2].CONTROL1.WORD = 0x0200; /* MB2 を受信用に設定 */
```

```
246
247   /* MB4(受信)設定 */
248   /* Configure CONTROL0
249   b15 IDE = 0 スタンダードフォーマット
250   b14 RTR = 0 データフレーム
251   b13 reserved
252   b12-2 STDID[10:0] = H'4C4 スタンダード ID
253   b1,0 EXTID[17:16] = 0 エクステンデッド ID
254   b15-0 EXTID[15:0] = 0 エクステンデッド ID*/
255   RCANA.MSG[4].CONTROL0.LONG = MB4_ID; /* ID:131,ｽﾀﾝﾀﾞｰﾄﾞﾌｫｰﾏｯﾄ,ﾃﾞｰﾀﾌﾚｰﾑ */
256
257   /* Configure LAFM
258   b15 IDE_LAFM = 0 対応する IDE ビットが有効
259   b14,13 reserved
260   b12-2 STDID_LAFM[10:0] = 0 対応する STDID ビットが有効
261   b1,0 EXTID_LAFM[17:16] = 0 対応する EXTID ビットが有効
262   b15-0 EXTID_LAFM[15:0] = 0 対応する EXTID ビットが有効*/
263   RCANA.MSG[4].LAFM.LONG = 0x00000000; /* STD_LAFM, IDE_LAFM の設定 */
264
265   /* Configure CONTROL1
266   b15,14 reserved
267   b13 NMC = 0 オーバランモード
268   b12 ATX = 0 データフレームの自動送信無効
269   b11 DART = 0 再送信有効
270   b10-8 MBC[2:0] = H'2 メールボックスコンフィギュレーション
271   b7-4 reserved
272   b3-0 DLC[3:0] = 0 データ長コード*/
273   RCANA.MSG[4].CONTROL1.WORD = 0x0200; /* MB4 を受信用に設定 */
274
275   /* ﾋﾞｯﾄﾚｰﾄ設定<Pφ=40MHz,500kbps> */
276   /* Configure BCR1
277   b15-12 TSG1[3:0] = H'5 PRSEG + PHSEG1＝6 タイムクォンタ
278   b11 reserved
279   b10-8 TSG2[2:0] = H'2 PHSEG2＝3 タイムクォンタ
280   b7,6 reserved
281   b5,4 SJW[1:0] = 0 同期ジャンプ幅＝1 タイムクォンタ
282   b3-1 reserved
283   b0 BSP = 0 1 か所でビットサンプリングが行われます（タイムセグメント 1 の最後）*/
284   RCANA.BCR1.WORD = 0x5200; /* TSEG1=5(6tq),TSEG2=2(3tq),SJW=0,BSP=0 */
285
286   /* Configure BCR0
287   b15-8 reserved
288   b7-0 BRP[7:0] = 3 8×周辺バスクロック*/
289   RCANA.BCR0.WORD = 0x0003; /* BRP = 3 */
290
291   /* 割り込み MASK ｾｯﾄ */
292   set_imask(15); /* 割り込み MASK ｾｯﾄ */
293
294   /* ﾘｾｯﾄ状態の解除 */
295   /* Configure MCR
```

```
296   b15 MCR15 = 1 RCAN-TL1 と HCAN2 は異なる順序
297   b14 MCR14 = 1 MCR6 がセットされると RCAN-TL1 はバスオフ状態のあと、ただちにホルトモードに入ります
298   b13-11 reserved
299   b10-8 TST[2:0] = 1 リスンオンリモード（受信専用モード）
300   b7 MCR7 = 1 CAN バスのアクティビティによる自動ウェイクモードが有効
301   b6 MCR6 = 1 バスオフ時に MCR1 設定によるホルトモード遷移を有効にします
302   b5 MCR5 = 1 CAN スリープモードへの遷移が有効です
303   b4,3 reserved
304   b2 MCR2 = 1 メールボックス番号順（メールボックス 31→メールボックス 1）に送信
305   b1 MCR1 = 1 ホルトモード遷移リクエスト
306   b0 MCR0 = 0 リセットモードリクエストをクリア*/
307   RCANA.MCR.WORD &= 0xFFFE; /* MCR0 ｸﾘｱ(ﾘｾｯﾄﾘｸｴｽﾄﾋﾞｯﾄのｸﾘｱ) */
308   while ((RCANA.GSR.WORD & 0x0008) != 0x0000); /* GSR3=0?(RCAN-ET ﾘｾｯﾄ状態は解除？) */
309
310   /* ｽﾀｰﾄｺﾏﾝﾄﾞ確認 */
311   while(!(*command1 == START_CMD)) /* ｽﾀｰﾄｺﾏﾝﾄﾞ(H'11)を受信するまでﾙｰﾌﾟ */
312   {
313
314   while((RCANA.RXPR.WORD.RXPR0 & 0x0001) != 0x0001); /* MB0 受信完了待ち */
315
316   /* Configure RXPR0
317   b15-0 RXPR0[15:0] = 0 対応するメールボックスが CAN データフレームを受信した*/
318   RCANA.RXPR.WORD.RXPR0 = 0x0001; /* MB0 受信完了ﾌﾗｸﾞｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
319
320   }
321
322   /* ROM ﾏｯﾄ切替関数、FCU ﾌｧｰﾑｳｪｱ転送関数と ROM 書き換え制御関数を ROM/RAM 転送する関数を RAM に転送 */
323   rom_flash = (unsigned char *)(__sectop("PMAT_C")); /* rom_flash に PMAT_C(ｾｸｼｮﾝ)ROM ﾏｯﾄ切り替えﾌﾟﾛｸﾞﾗﾑ先頭ｱﾄﾞﾚｽをｾｯﾄ */
324   ram_flash = (unsigned char *)(__sectop("RMAT_C")); /* ram_flash に RMAT_C(ｾｸｼｮﾝ)RAM 書き込み先、先頭ｱﾄﾞﾚｽをｾｯﾄ */
325
326   while(rom_flash <= (unsigned char *)(__secend("PMAT_C"))) /* PMAT_C(ｾｸｼｮﾝ)FCU ﾌｧｰﾑｳｪｱ転送関数と ROM 書き換え制御関数を ROM/RAM 転送する関数終端アドレスになるまで繰り返す */
327   {
328   *ram_flash++ = *rom_flash++; /* ROM(ﾕｰｻﾞﾏｯﾄ)から RAM へ転送 */
329   }
330
331   MATC(); /* ROM ﾏｯﾄ切り替え関数呼び出し(ﾕｰｻﾞﾌﾞｰﾄﾏｯﾄに切り替え) */
332   } /* End of function main() */
333
334   /***********************************************************************
335   * Function Name: MATC
336   * Description  : ROM ﾏｯﾄ切り替え関数
337   * Arguments    : none
338   * Return Value : none
339   ***********************************************************************/
340   #pragma section MAT_C
```

```
341   void MATC(void)
342   {
343
344   volatile unsigned dummy_read;
345
346   if((ROM.ROMMAT.WORD & 0x0001) == 0x0000) /* ROM ﾏｯﾄがﾕｰｻﾞﾏｯﾄのとき */
347   {
348   /* ﾕｰｻﾞﾌﾞｰﾄﾏｯﾄに切り替え */
349   /* Configure ROMMAT
350   b15-8 KEY = H'3B キーコード
351   b7-1 reserved
352   b0 ROMSEL = 1 ユーザブートマット選択*/
353   ROM.ROMMAT.WORD = 0x3B01; /* ROMSEL ﾋﾞｯﾄを 1 にｾｯﾄ */
354   dummy_read = ROM.ROMMAT.WORD;
355   nop();
356   nop();
357   nop();
358   nop();
359   nop();
360   while((ROM.ROMMAT.WORD & 0x0001) == 0x0000); /* ﾕｰｻﾞﾌﾞｰﾄﾏｯﾄに切り替わるまで待つ */
361
362   ROM_RAM_T(); /* FCU ﾌｧｰﾑｳｪｱ転送関数呼び出し */
363   }
364   else /* ROM ﾏｯﾄがﾕｰｻﾞﾌﾞｰﾄﾏｯﾄのとき */
365   {
366   /* ﾕｰｻﾞﾏｯﾄに切り替え */
367   /* Configure ROMMAT
368   b15-8 KEY = H'3B キーコード
369   b7-1 reserved
370   b0 ROMSEL = 0 ユーザマット選択*/
371   ROM.ROMMAT.WORD = 0x3B00; /* ROMSEL ﾋﾞｯﾄを 0 にｾｯﾄ */
372   dummy_read = ROM.ROMMAT.WORD;
373   nop();
374   nop();
375   nop();
376   nop();
377   nop();
378   while((ROM.ROMMAT.WORD & 0x0001) == 0x0001); /* ﾕｰｻﾞﾏｯﾄに切り替わるまで待つ */
379
380   Addr(); /* ROM 書き換え制御関数(RAM ｴﾘｱ)呼び出し */
381   }
382
383   } /* End of function MATC() */
384
385   /***********************************************************************
386   * Function Name: ROM_RAM_T
387   * Description  : FCU ﾌｧｰﾑｳｪｱ転送関数、ROM 書き換え制御関数を ROM(ﾕｰｻﾞﾌﾞｰﾄﾏｯﾄ)/RAM 転送する関
      数
388   * Arguments    : none
389   * Return Value : none
```

```
390  ***********************************************************************/
391  void ROM_RAM_T(void)
392  {
393
394  unsigned char *rom_flash;
395  unsigned char *ram_flash;
396
397  /* ROM 書き換え制御ﾌﾟﾛｸﾞﾗﾑを RAM に転送 */
398  rom_flash = UBM_PTOP_ADDR; /* rom_access に PROM_WE(ｾｸｼｮﾝ)ROM 書き換え制御ﾌﾟﾛｸﾞﾗﾑ先頭ｱﾄﾞﾚｽをｾｯﾄ */
399  ram_flash = RAM_WRITE_ADDR; /* ram_access に RROM_WE(ｾｸｼｮﾝ)RAM 書き込み先、先頭ｱﾄﾞﾚｽをｾｯﾄ */
400  while(rom_flash <= UBM_DEND_ADDR) /* PROM_WE(ｾｸｼｮﾝ)ROM 書き換えﾌﾟﾛｸﾞﾗﾑ終端ｱﾄﾞﾚｽになるまで繰り返す */
401  {
402  *ram_flash++ = *rom_flash++; /* ROM(ﾕｰｻﾞﾌﾞｰﾄﾏｯﾄ)から RAM へ転送 */
403  }
404
405  MATC(); /* ROM ﾏｯﾄ切り替え関数呼び出し(ﾕｰｻﾞﾏｯﾄに切り替え) */
406  } /* End of function ROM_RAM_T() */
407  #pragma section
```

5.2 ユーザブートマットプログラム

```
/***********************************************************************
* DISCLAIMER
* This software is supplied by Renesas Electronics Corporation and is only
* intended for use with Renesas products. No other uses are authorized. This
* software is owned by Renesas Electronics Corporation and is protected under
* all applicable laws, including copyright laws.
* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES REGARDING
* THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT
* LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
* AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY DISCLAIMED.
* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR
* ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS AFFILIATES HAVE
* BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
* Renesas reserves the right, without notice, to make changes to this software
* and to discontinue the availability of this software. By using this software,
* you agree to the additional terms and conditions found by accessing the
* following link:
* http://www.renesas.com/disclaimer *
* Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
***********************************************************************/
/***********************************************************************
* File Name    : SH7254R_USERBOOMAT.c
* Version      : 1.00
* Device(s)    : SH72544R
* Tool-Chain   : High-performance Embedded Workshop (Ver.4.08.00).
* OS           : None
* H/W Platform : SH7254R
* Description  : This is the main tutorial code.
* Operation    : USERBOOTMAT
* Limitations  : None
***********************************************************************/
/***********************************************************************
* History : DD.MM.YYYY Version Description
*         : 23.12.2011 1.00    First Release
***********************************************************************/
/***********************************************************************
Includes   <System Includes> , "Project Includes"
***********************************************************************/
#include "iodefine.h" /* 周辺ﾚｼﾞｽﾀ定義ﾍｯﾀﾞﾌｧｲﾙ */
#include <machine.h> /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ */
#include "include.h" /*  ｱﾄﾞﾚｽ,ﾃﾞｰﾀ定義ﾍｯﾀﾞﾌｧｲﾙ */
```

```
44  /***********************************************************************
45  Private global variables and functions
46  ***********************************************************************/
47  void ROM_WE_MAIN(void); /* ROM 書き換え制御関数 */
48  void FCU_FIRMCOPY(void); /* FCU ﾌｧｰﾑｳｪｱ転送関数 */
49  void FCU_RESET(void); /* FCU 初期化関数 */
50  void ROM_WRITE(volatile unsigned short *write_addr,unsigned short
    write_enable_area); /* ROM 書き込み関数 */
51  void ROM_ERASE(volatile unsigned char *erase_addr,unsigned short
    erase_enable_area); /* ROM 消去関数 */
52  void ROM_WE_DL(void); /* ROM 書き込みﾃﾞｰﾀﾀﾞｳﾝﾛｰﾄﾞ関数 */
53
54  #pragma section ROM_WE
55  /* 消去対象ﾌﾞﾛｯｸ宣言 */
56  unsigned long ROM_block[32] = { 0x00000000,0x00002000,0x00004000,0x00006000,
57  0x00008000,0x0000A000,0x0000C000,0x0000E000,
58  0x00010000,0x00020000,0x00030000,0x00040000,
59  0x00050000,0x00060000,0x00070000,0x00080000,
60  0x00090000,
61  0x000A0000,0x000C0000,0x000E0000,0x00100000,
62  0x00120000,0x00140000,0x00160000,0x00180000,
63  0x001A0000,0x001C0000,0x001E0000,0x00200000,
64  0x00220000,0x00240000,0x00260000};
65
66  unsigned long WRITE_DATA_buff[64]; /* ROM 書き込み元ﾃﾞｰﾀ(256Byte)を格納 */
67
68  /***********************************************************************
69  * Function Name: ROM_WE_MAIN
70  * Description  : The main loop
71  * Arguments    : none
72  * Return Value : none
73  ***********************************************************************/
74  void ROM_WE_MAIN(void)
75  {
76
77  unsigned long *command_buff;
78  unsigned short *write_addr_buff;
79  unsigned char *erase_addr_buff;
80  unsigned short we_enable_area;
81  unsigned int block_No;
82  volatile unsigned short dummy_read;
83
84  command_buff = MB4_DATA0_ADDR; /* command_buff にﾒｰﾙﾎﾞｯｸｽ 4 ﾒｯｾｰｼﾞﾃﾞｰﾀ 0 の先頭ｱﾄﾞﾚｽをｾｯ
    ﾄ */
85
86  /* FCU ﾌｧｰﾑｳｪｱ転送 */
87  FCU_FIRMCOPY(); /* FCU ﾌｧｰﾑｳｪｱ関数呼び出し */
88
89  /* FCU 初期化 */
```

```
90    FCU_RESET(); /* FCU 初期化関数呼び出し */
91
92    /* ROM 初期化 */
93    /* Configure FMODR
94    b7-5 reserved
95    b4 FRDMD = 0 メモリ領域リードモード
96    b3-0 reserved*/
97    ROM.FMODR.BYTE = 0x00; /* ﾒﾓﾘ領域ﾓｰﾄﾞに設定 */
98
99    /* Configure FAEINT
100   b7 ROMAEIE = 0 ROMAE＝1 で FIFE 割り込み要求を発生しない
101   b6,5 reserved
102   b4 CMDLKIE = 0 CMDLK＝1 で FIFE 割り込み要求を発生しない
103   b3 EEPAEIE = 0 EEPROM アクセス違反割り込みイネーブル
104   b2 EEPIFEIE = 0 EEPROM 命令フェッチ違反割り込みイネーブル
105   b1 EEPRPEIE = 0 EEPROM リードプロテクト違反割り込みイネーブル
106   b0 EEPWPEIE = 0 EEPROM 書き込み／消去プロテクト違反割り込みイネーブル*/
107   ROM.FAEINT.BYTE = 0x00; /* ﾌﾗｯｼｭｲﾝﾀﾌｪｰｽ割り込みを禁止 */
108
109   /* Configure FPROTR
110   b15-8 FPKEY = H'55 キーコード
111   b7-1 reserved
112   b0 FPROTCN = 0 ロックビットによるプロテクト有効*/
113   ROM.FPROTR.WORD  = 0x5500;   /* ROM のﾛｯｸﾋﾞｯﾄﾌﾟﾛﾃｸﾄを有効に設定 */
114
115   /* Configure FENTRYR
116   b15-8 FPKEY = H'AA キーコード
117   b7 FENTRYD = 0 EEPROM P/E モードエントリビット
118   b6 reserved
119   b5 FENTRY5 = 0 ROM 0.25MB はリードモード
120   b4 FENTRY4 = 0 ROM 0.25MB はリードモード
121   b3 FENTRY3 = 0 ROM 0.25MB はリードモード
122   b2 FENTRY2 = 0 ROM 1MB はリードモード
123   b1 FENTRY1 = 0 ROM 1MB はリードモード
124   b0 FENTRY0 = 0 ROM 1MB はリードモード*/
125   ROM.FENTRYR.WORD  = 0xAA00; /* ROM をﾘｰﾄﾞﾓｰﾄﾞに移行 */
126
127   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
128   nop();
129   nop();
130   nop();
131   nop();
132   nop();
133
134   for(block_No=1; block_No<32; block_No++) /* ﾌﾞﾛｯｸ 0(ﾌﾟﾛｸﾞﾗﾑ格納ｴﾘｱ)以外全ﾌﾞﾛｯｸ消去 */
135   {
136
137   if(block_No >= 30)
138   {
```

```
139     we_enable_area = WE_ENABLE_AREA_4; /* ﾌﾞﾛｯｸ 30 以上のとき,we_enable_area に
        H'AA08(H'00240000～H'0027FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
140     }
141     else if(block_No >= 28)
142     {
143     we_enable_area = WE_ENABLE_AREA_3; /* ﾌﾞﾛｯｸ 28 以上のとき,we_enable_area に
        H'AA04(H'00200000～H'0023FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
144     }
145     else if(block_No >= 20)
146     {
147     we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
        H'AA02(H'00100000～H'001FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
148     }
149     else
150     {
151     we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
        H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
152     }
153
154     erase_addr_buff = ((unsigned char *)(ROM_block[block_No] + WE_OFFSET)); /*
        erase_addr_buff に各ﾌﾞﾛｯｸの先頭ｱﾄﾞﾚｽ+0x80800000 をｾｯﾄ */
155
156     /* ROM 消去関数 */
157     ROM_ERASE(erase_addr_buff,we_enable_area); /* ROM 消去関数呼び出
        し,erase_addr_buff,we_enable_area を引数として渡す */
158
159     }
160
161     /* 書き込みﾃﾞｰﾀ要求ｺﾏﾝﾄﾞ送信 */
162     /* Configure TXPR
163     b15-0 TXPR1[15:0] = 0 対応するメールボックスが送信メッセージアイドル状態
164     b15-1 TXPR0[15:1] = H'2 対応するメールボックスに送信リクエストが発生
165     b0 reserved*/
166     RCANA.TXPR.LONG = 0x00000002; /* ﾒｰﾙﾎﾞｯｸｽ 1 を送信待ちに設定 */
167     while ((RCANA.TXACK.WORD.TXACK0 & 0x0002) != 0x0002); /* 送信完了？ */
168     /* Configure TXACK
169     b15-0 TXACK1[15:0] = 0［クリア条件］1 を書き込む
170     b15-1 TXACK0[15:1] = H'2 対応するメールボックスのメッセージ（データフレームまたはリモートフ
        レーム）が正常に送信された
171     b0 reserved*/
172     RCANA.TXACK.WORD.TXACK0 = 0x0002; /* 送信完了ﾌﾗｸﾞｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
173
174     /* 書き込み終了ｺﾏﾝﾄﾞ(H'FFFFFFFF)受信(以外のとき書き込み続行) */
175     while((RCANA.RXPR.WORD.RXPR0 & 0x0010) != 0x0010); /* ﾒｰﾙﾎﾞｯｸｽ 4 受信完了待ち */
176     /* 書き込みﾃﾞｰﾀ要求ｺﾏﾝﾄﾞ送信 */
177     /* Configure RXPR
178     b15-0 RXPR1[15:0] = 0 ［クリア条件］1 を書き込む
179     b15-1 RXPR0[15:0] = H'10 対応するメールボックスが CAN データフレームを受信した*/
180     RCANA.RXPR.WORD.RXPR0 = 0x0010; /* 受信完了ﾌﾗｸﾞｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
181
```

```
182  write_addr_buff = (unsigned short *)(*command_buff); /* write_addr_buff に受信ﾃﾞｰﾀ:4Byte(ｱﾄﾞﾚｽ)をｾｯﾄ */
183
184
185  while(write_addr_buff != (unsigned short *)END_CMD) /* write_addr_buff が
     END_CMD(H'FFFFFFFF)じゃないとき ROM 書き込み */
186  {
187
188  if(write_addr_buff >= (unsigned short *)ROM_block[30])
189  {
190  we_enable_area = WE_ENABLE_AREA_4; /* ﾌﾞﾛｯｸ 30 以上のとき,we_enable_area に
     H'AA10(H'00240000～H'0027FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
191  }
192  else if(write_addr_buff >= (unsigned short *)ROM_block[28])
193  {
194  we_enable_area = WE_ENABLE_AREA_3; /* ﾌﾞﾛｯｸ 28 以上のとき,we_enable_area に
     H'AA08(H'00200000～H'0023FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
195  }
196  else if(write_addr_buff >= (unsigned short *)ROM_block[20])
197  {
198  we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
     H'AA02(H'00100000～H'001FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
199  }
200  else
201  {
202  we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
     H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
203  }
204
205  write_addr_buff = (unsigned short *)(*command_buff + WE_OFFSET); /* write_addr_buff
     に受信ﾃﾞｰﾀ:8Byte(ｱﾄﾞﾚｽ)+0x80800000 をｾｯﾄ */
206
207  /* ROM 書き込みﾃﾞｰﾀﾀﾞｳﾝﾛｰﾄﾞ関数 */
208  ROM_WE_DL(); /* ROM 書き込みﾃﾞｰﾀﾀﾞｳﾝﾛｰﾄﾞ関数呼び出し */
209
210  /* ROM 書き込み */
211  ROM_WRITE(write_addr_buff,we_enable_area); /* ROM 書き込み関数呼び出
     し,write_addr_buff,we_enable_area を引数として渡す */
212
213
214  /* 書き込みﾃﾞｰﾀ要求ｺﾏﾝﾄﾞ送信 */
215  /* Configure TXPR
216  b15-0 TXPR1[15:0] = 0 対応するメールボックスが送信メッセージアイドル状態
217  b15-1 TXPR0[15:1] = H'2 対応するメールボックスに送信リクエストが発生
218  b0 reserved*/
219  RCANA.TXPR.LONG = 0x00000002; /* ﾒｰﾙﾎﾞｯｸｽ 1 を送信待ちに設定 */
220  while ((RCANA.TXACK.WORD.TXACK0 & 0x0002) != 0x0002); /* 送信完了？ */
221
222  /* Configure TXACK
223  b15-0 TXACK1[15:0] = 0［クリア条件］1 を書き込む
```

```
224   b15-1 TXACK0[15:1] = H'2 対応するメールボックスのメッセージ（データフレームまたはリモートフレーム）が正常に送信された
225   b0 reserved*/
226   RCANA.TXACK.WORD.TXACK0 = 0x0002; /* 送信完了ﾌﾗｸﾞｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
227
228   /* 書き込み終了ｺﾏﾝﾄﾞ(H'FFFFFFFF)受信(以外のとき書き込み続行) */
229   while((RCANA.RXPR.WORD.RXPR0 & 0x0010) != 0x0010); /* ﾒｰﾙﾎﾞｯｸｽ4受信完了待ち */
230   /* 書き込みﾃﾞｰﾀ要求ｺﾏﾝﾄﾞ送信 */
231   /* Configure RXPR
232   b15-0 RXPR1[15:0] = 0 ［クリア条件］1 を書き込む
233   b15-1 RXPR0[15:0] = H'10 対応するメールボックスが CAN データフレームを受信した*/
234   RCANA.RXPR.WORD.RXPR0 = 0x0010; /* 受信完了ﾌﾗｸﾞｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
235
236   write_addr_buff = (unsigned short *)(*command_buff); /* write_addr_buffに受信ﾃﾞｰﾀ:8Byte(ｱﾄﾞﾚｽ)をｾｯﾄ */
237
238   }
239
240   while(1); /* 無限ﾙｰﾌﾟ */
241
242   } /* End of function ROM_WE_MAIN() */
243
244   /***********************************************************************************
245   * Function Name: FCU_FIRMCOPY
246   * Description  : FCUﾌｧｰﾑｳｪｱ転送関数
247   * Arguments    : none
248   * Return Value : none
249   ***********************************************************************************/
250   void FCU_FIRMCOPY(void)
251   {
252
253   unsigned long *fcu_top_addr;
254   unsigned long *fcu_end_addr;
255   unsigned long *fcuram_top_addr;
256
257   volatile unsigned dummy_read;
258
259   if(!(ROM.FENTRYR.WORD == 0x0000))
260   {
261   /* Configure FENTRYR
262   b15-8 FPKEY = H'AA キーコード
263   b7 FENTRYD = 0 EEPROM P/E モードエントリビット
264   b6 reserved
265   b5 FENTRY5 = 0 ROM 0.25MB はリードモード
266   b4 FENTRY4 = 0 ROM 0.25MB はリードモード
267   b3 FENTRY3 = 0 ROM 0.25MB はリードモード
268   b2 FENTRY2 = 0 ROM 1MB はリードモード
269   b1 FENTRY1 = 0 ROM 1MB はリードモード
270   b0 FENTRY0 = 0 ROM 1MB はリードモード*/
271   ROM.FENTRYR.WORD = 0xAA00; /* FENTRYﾚｼﾞｽﾀがH'00以外のとき,ROMﾘｰﾄﾞﾓｰﾄﾞに移行*/
```

```
272
273   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
274   nop();
275   nop();
276   nop();
277   nop();
278   nop();
279   }
280
281   /* Configure FCURAME
282   b15-8 KEY = H'C4 キーコード
283   b7-1 reserved
284   b0 FCRME = 1 FCU RAM へのアクセス許可*/
285   ROM.FCURAME.WORD = 0xC401; /* RAMｱｸｾｽ許可 */
286
287   fcu_top_addr = FCU_TOP_ADDR; /* FCUﾌｧｰﾑｳｪｱ先頭ｱﾄﾞﾚｽをｾｯﾄ */
288   fcu_end_addr = FCU_END_ADDR; /* FCUﾌｧｰﾑｳｪｱ終端ｱﾄﾞﾚｽをｾｯﾄ */
289   fcuram_top_addr = FCURAM_TOP_ADDR; /* FCURAM 先頭ｱﾄﾞﾚｽをｾｯﾄ    */
290
291   while(fcu_top_addr <= fcu_end_addr)
292   { /* FCUﾌｧｰﾑｳｪｱ終端ｱﾄﾞﾚｽになるまで繰り返す */
293   *fcuram_top_addr++ = *fcu_top_addr++; /* FCUﾌｧｰﾑｳｪｱを FCURAM に転送 */
294   }
295
296   if((ROM.FSTATR1.BYTE & 0x80)) /* FCUERRﾋﾞｯﾄが 1(FCU の CPU 処理でｴﾗｰが発生)のとき */
297   {
298   FCU_RESET(); /* FCU 初期化関数呼び出し */
299   }
300
301
302   } /* End of function FCU_FIRMCOPY() */
303
304   /***********************************************************************
305   * Function Name: ROM_WRITE
306   * Description  : ROM 書き込み関数
307   * Arguments    : none
308   * Return Value :  *write_addr, write_enable_area
309   ***********************************************************************/
310   void ROM_WRITE(volatile unsigned short *write_addr,unsigned short
      write_enable_area)
311   {
312
313   unsigned int n;
314   unsigned long Timeout;
315   volatile unsigned short *write_data;
316   volatile unsigned short dummy_read;
317
318   write_data = (volatile unsigned short *)WRITE_DATA_buff; /* write_data に ROM 書き
      込み元データ先頭ｱﾄﾞﾚｽをｾｯﾄ*/
319
320   /* Configure FENTRYR
```

```
321   b15-8 FPKEY = H'AA キーコード
322   b7 FENTRYD = 0 EEPROM P/E モードエントリビット
323   b6 reserved
324   b5 FENTRY5 = 0 ROM 0.25MB はリードモード
325   b4 FENTRY4 = 0 ROM 0.25MB はリードモード
326   b3 FENTRY3 = 0 ROM 0.25MB はリードモード
327   b2 FENTRY2 = 0 ROM 1MB はリードモード
328   b1 FENTRY1 = 0 ROM 1MB はリードモード
329   b0 FENTRY0 = 0 ROM 1MB はリードモード*/
330   ROM.FENTRYR.WORD = 0xAA00; /* FENTRYR ﾚｼﾞｽﾀを初期化 */
331
332   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
333   nop();
334   nop();
335   nop();
336   nop();
337   nop();
338
339   ROM.FENTRYR.WORD = write_enable_area; /* write_enable_area を ROM/PE ﾓｰﾄﾞに設定 */
340
341   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
342
343   /* Configure FPROTR
344   b15-8 FPKEY = H'55 キーコード
345   b7-1 reserved
346   b0 FPROTCN = 1 ロックビットによるプロテクト無効*/
347   ROM.FPROTR.WORD  = 0x5501; /* ROM のﾛｯｸﾋﾞｯﾄﾌﾟﾛﾃｸﾄを無効に設定 */
348
349
350   *(volatile unsigned char *)write_addr = 0xE8; /* 書き込み用 FCU ｺﾏﾝﾄﾞを第 1 ｻｲｸﾙに Byte
      書き込み */
351
352   *(volatile unsigned char *)write_addr = 0x80; /* 書き込み用 FCU ｺﾏﾝﾄﾞを第 2 ｻｲｸﾙに Byte
      書き込み */
353
354   for(n = 0;n < 128; n++) /* 256Byte(1Word 単位)書き込み(128 ｻｲｸﾙ) */
355   {
356   *write_addr = *write_data++; /* 書き込み先に書き込み元ﾃﾞｰﾀを Word 書き込み */
357   }
358
359   *(volatile unsigned char *)write_addr = 0xD0; /* 書き込み用 FCU ｺﾏﾝﾄﾞを第 131 ｻｲｸﾙに Byte
      書き込み */
360
361   /* 書き込みﾀｲﾑｱｳﾄ判定(2.2[ms]) */
362   Timeout = 88000; /* ﾀｲﾑｱｳﾄ時間:2.2[ms],Pφ:40MHz */
363
364   while((ROM.FSTATR0.BYTE & 0x80) == 0x00) /* FRDY ﾋﾞｯﾄが 0(書き込み/消去処理中)のとき */
365   {
366
367   if(Timeout == 0)
```

```
368   {
369   FCU_RESET(); /* FCU 初期化   */
370   break;
371   }
372   Timeout--; /* Timeout をﾃﾞｸﾘﾒﾝﾄ */
373   }
374
375   /* Configure FENTRYR
376   b15-8 FPKEY = H'AA キーコード
377   b7 FENTRYD = 0 EEPROM P/E モードエントリビット
378   b6 reserved
379   b5 FENTRY5 = 0 ROM 0.25MB はリードモード
380   b4 FENTRY4 = 0 ROM 0.25MB はリードモード
381   b3 FENTRY3 = 0 ROM 0.25MB はリードモード
382   b2 FENTRY2 = 0 ROM 1MB はリードモード
383   b1 FENTRY1 = 0 ROM 1MB はリードモード
384   b0 FENTRY0 = 0 ROM 1MB はリードモード*/
385   ROM.FENTRYR.WORD = 0xAA00; /* ROM ﾘｰﾄﾞﾓｰﾄﾞに移行 */
386
387   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
388   nop();
389   nop();
390   nop();
391   nop();
392   nop();
393
394   /* ｴﾗｰ確認 */
395   if(!(ROM.FSTATR0.BYTE == 0x80))
396   { /* FSTATR0 が H'80 以外のとき */
397   while(1); /* 無限ﾙｰﾌﾟ */
398   }
399
400   } /* End of function ROM_WRITE() */
401
402   /***********************************************************************
403   * Function Name: ROM_ERASE
404   * Description  : ROM 消去関数
405   * Arguments    : none
406   * Return Value : *erase_addr, erase_enable_area
407   ***********************************************************************/
408   void ROM_ERASE(volatile unsigned char *erase_addr,unsigned short erase_enable_area)
409   {
410
411   volatile unsigned short dummy_read;
412   unsigned long Timeout;
413
414   /* Configure FENTRYR
415   b15-8 FPKEY = H'AA キーコード
416   b7 FENTRYD = 0 EEPROM P/E モードエントリビット
417   b6 reserved
```

```
418   b5 FENTRY5 = 0 ROM 0.25MB はリードモード
419   b4 FENTRY4 = 0 ROM 0.25MB はリードモード
420   b3 FENTRY3 = 0 ROM 0.25MB はリードモード
421   b2 FENTRY2 = 0 ROM 1MB はリードモード
422   b1 FENTRY1 = 0 ROM 1MB はリードモード
423   b0 FENTRY0 = 0 ROM 1MB はリードモード*/
424   ROM.FENTRYR.WORD = 0xAA00; /* FENTRYR ﾚｼﾞｽﾀを初期化 */
425
426   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
427   nop();
428   nop();
429   nop();
430   nop();
431   nop();
432
433   ROM.FENTRYR.WORD = erase_enable_area; /* erase_enable_area を ROM/PE モードに設定 */
434
435   dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
436
437   /* Configure FPROTR
438   b15-8 FPKEY = H'55 キーコード
439   b7-1 reserved
440   b0 FPROTCN = 1 ロックビットによるプロテクト無効*/
441   ROM.FPROTR.WORD  = 0x5501;   /* ROM のﾛｯｸﾋﾞｯﾄのﾌﾟﾛﾃｸﾄを無効に設定 */
442
443   *(volatile unsigned char *)erase_addr = 0x20; /* 消去用 FCU ｺﾏﾝﾄﾞを第 1 ｻｲｸﾙに Byte 書き込み */
444
445   *(volatile unsigned char *)erase_addr = 0xD0; /* 消去用 FCU ｺﾏﾝﾄﾞを第 2 ｻｲｸﾙに Byte 書き込み */
446
447   /* 消去ﾀｲﾑｱｳﾄ判定(0.88[s]) */
448   Timeout = 35200000; /* ﾀｲﾑｱｳﾄ時間:0.88[s],Pφ=40MHz */
449
450   while((ROM.FSTATR0.BYTE & 0x80) == 0x00) /* FRDY ﾋﾞｯﾄが 0(書き込み/消去処理中)のとき */
451   {
452   if(Timeout == 0)
453   {
454   FCU_RESET(); /* FCU 初期化   */
455   break;
456   }
457   Timeout--; /* Timeout をﾃﾞｸﾘﾒﾝﾄ */
458   }
459   /* Configure FENTRYR
460   b15-8 FPKEY = H'AA キーコード
461   b7 FENTRYD = 0 EEPROM P/E モードエントリビット
462   b6 reserved
463   b5 FENTRY5 = 0 ROM 0.25MB はリードモード
464   b4 FENTRY4 = 0 ROM 0.25MB はリードモード
465   b3 FENTRY3 = 0 ROM 0.25MB はリードモード
```

```
466    b2 FENTRY2 = 0 ROM 1MB はリードモード
467    b1 FENTRY1 = 0 ROM 1MB はリードモード
468    b0 FENTRY0 = 0 ROM 1MB はリードモード*/
469    ROM.FENTRYR.WORD = 0xAA00; /* ROM ﾘｰﾄﾞﾓｰﾄﾞに移行 */
470
471    dummy_read = ROM.FENTRYR.WORD; /* ﾀﾞﾐｰﾘｰﾄﾞして FENTRYR の値を確定 */
472    nop();
473    nop();
474    nop();
475    nop();
476    nop();
477
478    /* ｴﾗｰ確認 */
479    if(!(ROM.FSTATR0.BYTE == 0x80)) /* FSTATR0 が H'80 以外のとき */
480    {
481
482    while(1); /* 無限ﾙｰﾌﾟ */
483    }
484
485    } /* End of function ROM_ERASE() */
486
487    /***********************************************************************
488    * Function Name: ROM_WE_DL
489    * Description  : ROM 書き込みﾃﾞｰﾀﾀﾞｳﾝﾛｰﾄﾞ関数
490    * Arguments    : none
491    * Return Value : none
492    ***********************************************************************/
493    void ROM_WE_DL(void)
494    {
495
496    unsigned long  *DL_DATA,*MB_DATA0,*MB_DATA4;
497    unsigned int i;
498
499    DL_DATA = WRITE_DATA_buff; /* DL_DATA に ROM 書き込み元ﾃﾞｰﾀ先頭ｱﾄﾞﾚｽをｾｯﾄ */
500    MB_DATA0 = MB2_DATA0_ADDR; /* MB_DATA に MB2 のﾒｯｾｰｼﾞﾃﾞｰﾀ 0 の先頭ｱﾄﾞﾚｽをｾｯﾄ */
501    MB_DATA4 = MB2_DATA4_ADDR; /* MB_DATA に MB2 のﾒｯｾｰｼﾞﾃﾞｰﾀ 4 の先頭ｱﾄﾞﾚｽをｾｯﾄ */
502
503
504    for(i=0;i<(256/(sizeof(*DL_DATA) * 2));i++) /* 256Byte ﾃﾞｰﾀをﾀﾞｳﾝﾛｰﾄﾞ */
505    {
506
507    while((RCANA.RXPR.WORD.RXPR0 & 0x0004) != 0x0004); /* MB2 受信完了待ち */
508
509    /* Configure RXPR
510    b15-0 RXPR1[15:0] = 0 ［クリア条件］1 を書き込む
511    b15-1 RXPR0[15:0] = H'4 対応するメールボックスが CAN データフレームを受信した*/
512    RCANA.RXPR.WORD.RXPR0 = 0x0004; /* 受信完了ﾌﾗｸﾞｸﾘｱ(ｸﾘｱ条件：1 ﾗｲﾄ) */
513
514    *DL_DATA++ = *MB_DATA0; /* MB2 に受信した書き換えﾃﾞｰﾀ上位 4byte を RAM に格納*/
515    *DL_DATA++ = *MB_DATA4; /* MB2 に受信した書き換えﾃﾞｰﾀ下位 4byte を RAM に格納*/
```

```
516
517   }
518   } /* End of function ROM_WE_DL() */
519
520   /***********************************************************************************
521   * Function Name: FCU_RESET
522   * Description  : FCU 初期化関数
523   * Arguments    : none
524   * Return Value : none
525   ***********************************************************************************/
526   void FCU_RESET(void)
527   {
528
529   unsigned int i;
530
531   /* Configure FRESETR
532   b15-8 RXPR1[15:0] = H'CC キーコード
533   b7-1 reserved
534   b0 FRESET = H'1 FCU はリセットされる*/
535   ROM.FRESETR.WORD = 0xCC01; /* FCUﾘｾｯﾄ */
536   for(i =0 ; i<0x320 ; i++); /* 20[us]待つ */
537
538   /* Configure FRESETR
539   b15-8 RXPR1[15:0] = H'CC キーコード
540   b7-1 reserved
541   b0 FRESET = 0 FCU はリセットされない*/
542   ROM.FRESETR.WORD = 0xCC00; /* FCUﾘｾｯﾄ解除 */
543   } /* End of function FCU_RESET() */
544
545   #pragma section
```

6. SH7253 SH7256R グループでご使用の場合の変更点

本文 1.～4.及びサンプルソフトは SH7254R グループ用に作成されたものです。SH7253 グループ SH7256R グループでご使用される場合は SH7254R グループのサンプルソフトを下記に従い変更してください。

変更内容

(1)製品ヘッダファイル iodefine.h を SH7253 グループ用 SH7256R のものに差替えてください。また、iodefine.h を差し替えた場合、レジスタ名をファイルに合わせソースを修正してください。

(各説明において、ソースの行番号は、変更前のものを示しています。)

(2)SH7253 グループ SH7256R グループは、ユーザマットの容量及び消去対象ブロック(EB)数が、SH7254 グループとは異なります。消去対象ブロックのアドレスおよびフラッシュ P/E モードエントリレジスタ(FENTRYR)の設定値を対象製品に合わせて変更してください。

FENTRYR の設定値は、include.h 内に WE_ENABLE_AREA_N で定義しています。

製品毎に以下のよう include.h の内容を変更してください。

SH7254R グループ：#define WE_ENABLE_AREA_1　0xAA01

(変更前)　#define WE_ENABLE_AREA_2　0xAA02

#define WE_ENABLE_AREA_3　0xAA08

#define WE_ENABLE_AREA_4　0xAA10

SH7253 グループ　：　#define WE_ENABLE_AREA_1　0xAA01(変更無し)

(変更後)　#define WE_ENABLE_AREA_2　0xAA08(変更)

(AREA3,4 は、削除してください。)

SH7256R グループ：　#define WE_ENABLE_AREA_1　0xAA01(変更無し)

(変更後)　#define WE_ENABLE_AREA_2　0xAA02(変更無し)

#define WE_ENABLE_AREA_3　0xAA04(変更)

#define WE_ENABLE_AREA_4　0xAA08(変更)

#define WE_ENABLE_AREA_5　0xAA10(追加)

#define WE_ENABLE_AREA_6　0xAA20(追加)

#define WE_ENABLE_AREA_7　0xAA40(追加)

具体的には、5.2 ユーザブートマットプルグラムリストの

56 行目の消去ブロック数の記述変更

SH7254R グループ：unsignd long ROM_block[32] (変更前)

SH7253　グループ：unsignd long ROM_block[22] (変更後)

SH7256R グループ：unsignd long ROM_block[44] (変更後)

61 行目から 64 行目に記述されている、消去ブロック先頭アドレスの変更

SH7254R グループ：(変更前)

0x000A0000,0x000C0000,0x000E0000,0x00100000,
0x00120000,0x00140000,0x00160000,0x00180000,
0x001A0000,0x001C0000,0x001E0000,0x00200000,
0x00220000,0x00240000,0x00260000};

SH7253　グループ：(変更後)

0x000A0000,0x000C0000,0x000E0000,0x00100000,
0x00120000,~~0x00140000,0x00160000,0x00180000,~~
~~0x001A0000,0x001C0000,0x001E0000,0x00200000~~,
~~0x00220000,0x00240000,0x00260000~~};

(取り消し線の箇所を削除してください。)

SH7256R　グループ：(変更後)

0x000A0000,0x000C0000,0x000E0000,0x00100000,
0x00120000,0x00140000,0x00160000,0x00180000,
0x001A0000,0x001C0000,0x001E0000,0x00200000,
0x00220000,0x00240000,0x00260000, 0x00280000,
0x002A0000,0x002C0000,0x002E0000,0x03000000,
0x03200000,0x03400000,0x03600000,0x03800000,
0x03A00000,0x03C00000,0x03E00000};

(下線の箇所を追加してください。)

134 行目

SH7254R グループ：for(block_No=1; block_No<32; block_No++) (変更前)

SH7253　グループ：for(block_No=1; block_No<22; block_No++) (変更後)

SH7256R グループ：for(block_No=1; block_No<44; block_No++) (変更後)

137 行目

SH7254R グループ：if(block_No >= 30) (変更前)

SH7253　グループ：if(block_No >= 20) (変更後)

SH7256R グループ：if(block_No >= 42) (変更後)

139 行目から

SH7254R グループ：変更前

```
139     we_enable_area = WE_ENABLE_AREA_4; /* ﾌﾞﾛｯｸ 30 以上のとき,we_enable_area に
        H'AA08(H'00240000～H'0027FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
140     }
141     else if(block_No >= 28)
142     {
143     we_enable_area = WE_ENABLE_AREA_3; /* ﾌﾞﾛｯｸ 28 以上のとき,we_enable_area に
        H'AA04(H'00200000～H'0023FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
144     }
145     else if(block_No >= 20)
146     {
147     we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
        H'AA02(H'00100000～H'001FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
148     }
149     else
150     {
151     we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
        H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
152     }
153
154     erase_addr_buff = ((unsigned char *)(ROM_block[block_No] + WE_OFFSET)); /*
        erase_addr_buff に各ﾌﾞﾛｯｸの先頭ｱﾄﾞﾚｽ+0x80800000 をｾｯﾄ */
155
```

SH7253 グループ：変更後

```
139     we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
        H'AA08(H'00100000～H'0013FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
140     }
149     else
150     {
151     we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
        H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
152     }
153
154     erase_addr_buff = ((unsigned char *)(ROM_block[block_No] + WE_OFFSET)); /*
        erase_addr_buff に各ﾌﾞﾛｯｸの先頭ｱﾄﾞﾚｽ+0x80800000 をｾｯﾄ */
155
```

SH7256R グループ：

```
139     we_enable_area = WE_ENABLE_AREA_7; /* ﾌﾞﾛｯｸ 42 以上のとき,we_enable_area に
        H'AA40(H'003C0000～H'003FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
140     }
141     else if(block_No >= 40)
142     {
143     we_enable_area = WE_ENABLE_AREA_6; /* ﾌﾞﾛｯｸ 40 以上のとき,we_enable_area に
        H'AA20(H'00380000～H'003BFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
144     }
145     else if(block_No >= 38)
146     {
147     we_enable_area = WE_ENABLE_AREA_5; /* ﾌﾞﾛｯｸ 38 以上のとき,we_enable_area に
        H'AA10(H'00340000～H'0037FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
148     }
149     else if(block_No >= 36)
150     {
151     we_enable_area = WE_ENABLE_AREA_4; /* ﾌﾞﾛｯｸ 36 以上のとき,we_enable_area に
        H'AA08(H'00300000～H'0033FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
152     }
        else if(block_No >= 28)
153     {
154     we_enable_area = WE_ENABLE_AREA_3; /* ﾌﾞﾛｯｸ 28 以上のとき,we_enable_area に
        H'AA04(H'00200000～H'002FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
155     }
156     else if(block_No >= 20)
157     {
158     we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
        H'AA02(H'00100000～H'001FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
159     }
160     else
161     {
162     we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
        H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
163     }
164
165     erase_addr_buff = ((unsigned char *)(ROM_block[block_No] + WE_OFFSET)); /*
        erase_addr_buff に各ﾌﾞﾛｯｸの先頭ｱﾄﾞﾚｽ+0x80800000 をｾｯﾄ */
```

188 行目

SH7254R グループ：変更前

```
188   if(write_addr_buff >= (unsigned short *)ROM_block[30])
189   {
190   we_enable_area = WE_ENABLE_AREA_4; /* ﾌﾞﾛｯｸ 30 以上のとき,we_enable_area に
      H'AA10(H'00240000～H'0027FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
191   }
192   else if(write_addr_buff >= (unsigned short *)ROM_block[28])
193   {
194   we_enable_area = WE_ENABLE_AREA_3; /* ﾌﾞﾛｯｸ 28 以上のとき,we_enable_area に
      H'AA08(H'00200000～H'0023FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
195   }
196   else if(write_addr_buff >= (unsigned short *)ROM_block[20])
197   {
198   we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
      H'AA02(H'00100000～H'001FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
199   }
200   else
201   {
202   we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
      H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
203   }
```

SH7253 グループ：変更後

```
188   if(write_addr_buff >= (unsigned short *)ROM_block[20])
189   {
190   we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
      H'AA08(H'00100000～H'0013FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
191   }
200   else
201   {
202   we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
      H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
203   }
```

SH7256R グループ：変更後

```
188     if(write_addr_buff >= (unsigned short *)ROM_block[42])
189     {
190     we_enable_area = WE_ENABLE_AREA_7; /* ﾌﾞﾛｯｸ 42 以上のとき,we_enable_area に
        H'AA40(H'003C0000～H'003FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
191     }
192     else if(block_No >= 40)
193     {
194     we_enable_area = WE_ENABLE_AREA_6; /* ﾌﾞﾛｯｸ 40 以上のとき,we_enable_area に
        H'AA20(H'00380000～H'003BFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
195     }
196     else if(block_No >= 38)
197     {
198     we_enable_area = WE_ENABLE_AREA_5; /* ﾌﾞﾛｯｸ 38 以上のとき,we_enable_area に
        H'AA10(H'00340000～H'0037FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
199     }
200     else if(block_No >= 36)
201     {
202     we_enable_area = WE_ENABLE_AREA_4; /* ﾌﾞﾛｯｸ 36 以上のとき,we_enable_area に
        H'AA08(H'00300000～H'0033FFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
203     }
204     else if(block_No >= 28)
205     {
206     we_enable_area = WE_ENABLE_AREA_3; /* ﾌﾞﾛｯｸ 28 以上のとき,we_enable_area に
        H'AA04(H'00200000～H'002FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
207     }
208     else if(block_No >= 20)
209     {
210     we_enable_area = WE_ENABLE_AREA_2; /* ﾌﾞﾛｯｸ 20 以上のとき,we_enable_area に
        H'AA02(H'00100000～H'001FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
211     }
212     else
213     {
214     we_enable_area = WE_ENABLE_AREA_1; /* ﾌﾞﾛｯｸ 19 以下のとき,we_enable_area に
        H'AA01(H'00000000～H'000FFFFF を ROM P/E ﾓｰﾄﾞ)をｾｯﾄ */
215     }
```

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

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. リザーブアドレスのアクセス禁止

【注意】リザーブアドレス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、事前に問題ないことをご確認下さい。

同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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。

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

Colophon 1.0