お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

H8S/2623F グループ

CAN バスを使用したオンボード書き換え

目次

1. 仕　様

1.1 動作条件

- 使用デバイス：HD64F2623FA20
- CPU動作モード：モード 7
- 動作電圧：Vcc=3.3V,PVcc=5.0V
- 動作周波数：20MHz

1.2 オンボードプログラミングモード

- ユーザプログラムモード
  ただし、書き込み／消去プログラム、RAM転送プログラム、FWE判定プログラムなどをブートモードもしくはライタモードであらかじめ書き込んでいることを前提条件とします。

1.3 書き換え方式

- 転送元から書き込みデータを受信しフラッシュメモリに書き込みます。
- 転送元との通信手段には、CAN(Controller Area Network)を使用します。以下に転送フォーマットを示します。マスタを転送元とし、スレーブを受信側とします。

1.4 FWE 端子の制御

- FWE端子の回路は、ルネサステクノロジ製オンボード書き込みツールを使用せずユーザが作製する例です。FWE端子は、受信側のH8S/2623FがI/Oポート(PB0)で制御します。

1.5 オンボード書き換え構成

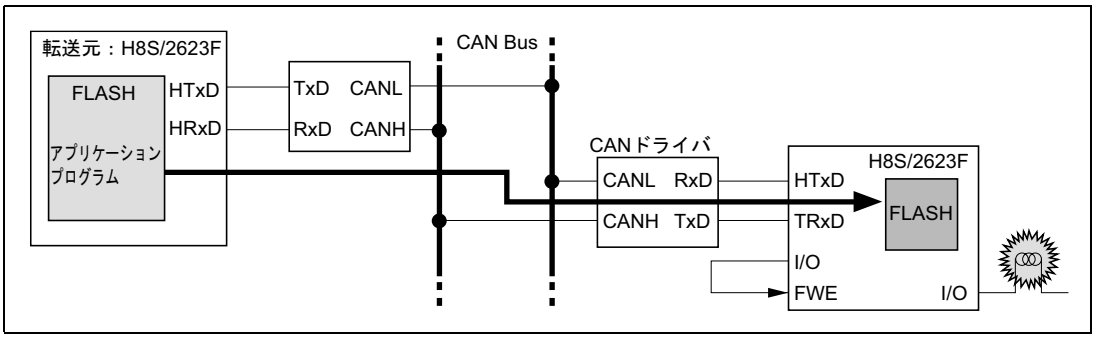


図 1.1　CAN を使用したオンボード書き換え構成図

1.6 H8S/2623F のオンボード書き換え実行手順

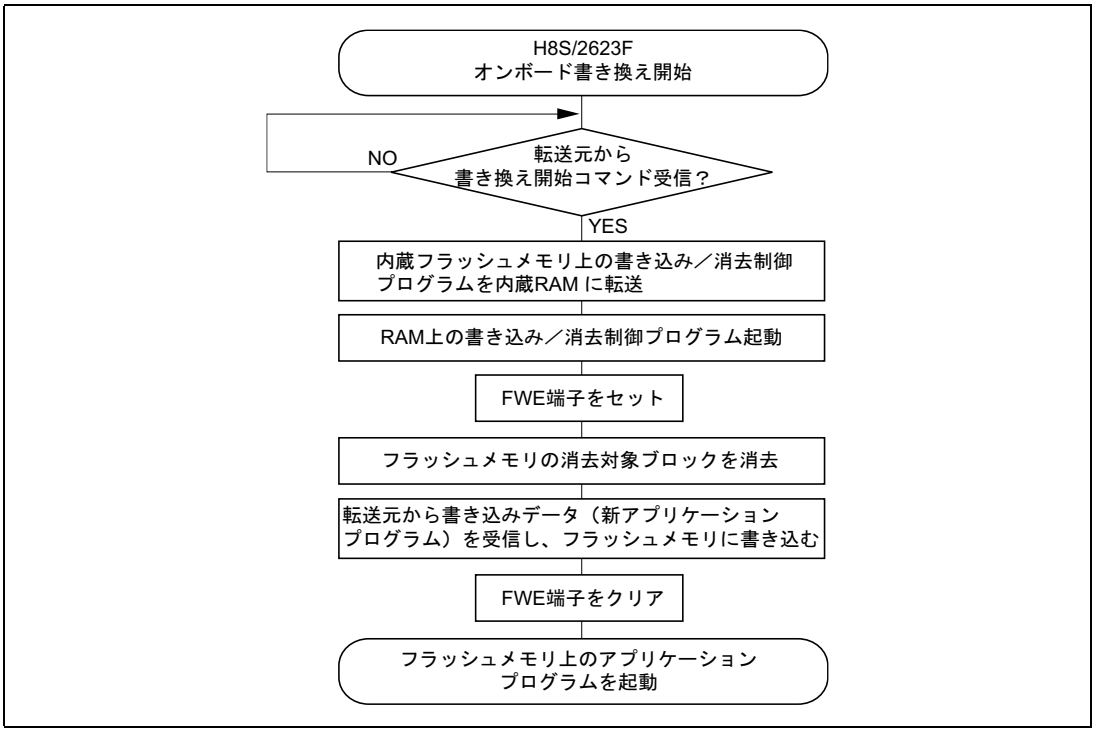
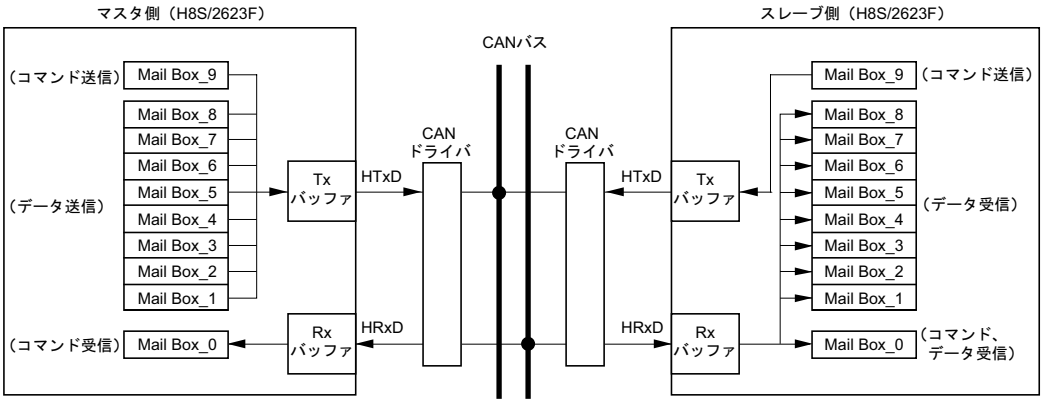


図 1.2 CAN を使用したオンボード書き換え手順

1.7 CAN 仕様

- ビットレート：1Mbps
- 使用メールボックス数：10個
- フォーマット：スタンダードフォーマット（Identifier：11ビット)
- メッセージ優先順位：メールボックス番号順(若い方から)



	マスタ側（Identifier）	スレーブ側（Identifier）
コマンド	送信：Mail Box_9 (H'00):1～4バイト	受信：Mail Box_0 (H'00):1～8バイト
	受信：Mail Box_0 (H'09):1バイト	送信：Mail Box_9 (H'09):1 バイト
データ転送	送信：Mail Box_1 (H'01):8バイト	受信：Mail Box_1 (H'01):8バイト
	送信：Mail Box_2 (H'02):8バイト	受信：Mail Box_2 (H'02):8バイト
	送信：Mail Box_3 (H'03):8バイト	受信：Mail Box_3 (H'03):8バイト
	送信：Mail Box_4 (H'04):8バイト	受信：Mail Box_4 (H'04):8バイト
	送信：Mail Box_5 (H'05):8バイト	受信：Mail Box_5 (H'05):8バイト
	送信：Mail Box_6 (H'06):8バイト	受信：Mail Box_6 (H'06):8バイト
	送信：Mail Box_7 (H'07):8バイト	受信：Mail Box_7 (H'07):8バイト
	送信：Mail Box_8 (H'08):8バイト	受信：Mail Box_8 (H'08):8バイト

図 1.3　メールボックスの割り付け

2. 動作概要

以下にCANを使用したオンボード書き換えの動作概要を示します。

2.1 通常動作

(1) フラッシュメモリ上のアプリケーションプログラムの一部に書き込み／消去制御プログラムを書き込んでおきます。
また、FWE端子設定手段、アプリケーション転送手段をあらかじめ設けておきます。

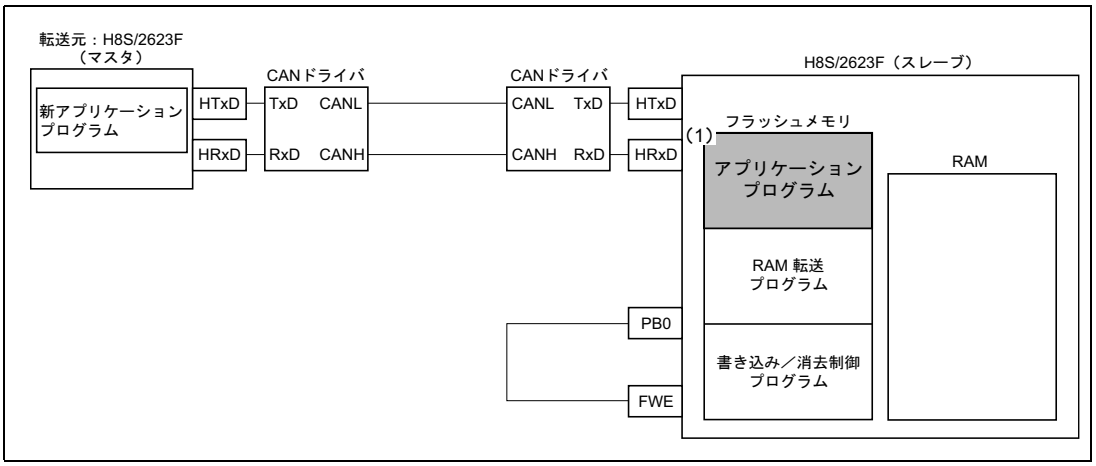


図 2.1 通常動作

2.2 オンボード書き換え開始

(2) 転送元から書き換え開始コマンドを受信します。

(3) アプリケーションプログラムは、RAM転送プログラムを起動しフラッシュメモリ上の書き込み／消去制御プログラムを内蔵RAMに転送します。

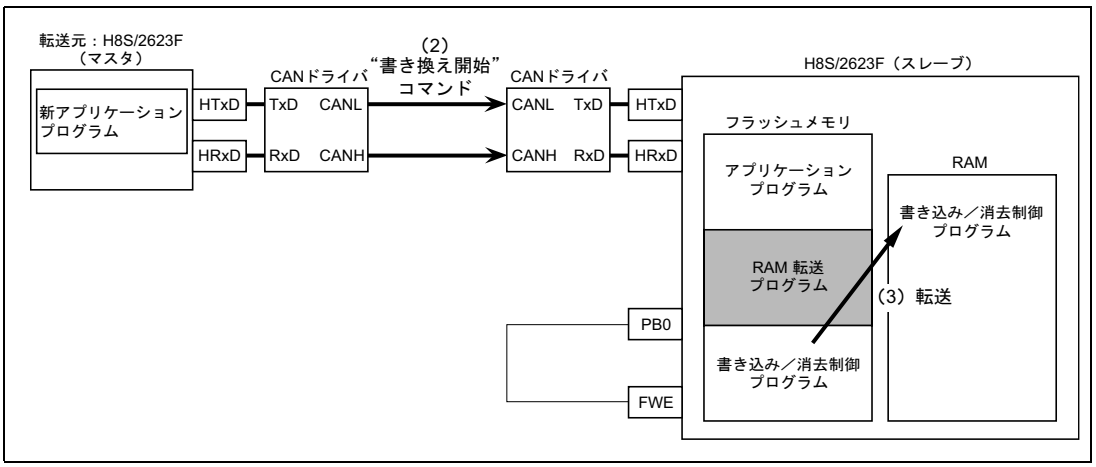


図 2.2 オンボード書き換え開始

2.3 書き込み／消去制御プログラム起動

(4) RAM転送プログラムは転送終了後、書き込み／消去制御プログラムへ分岐します。

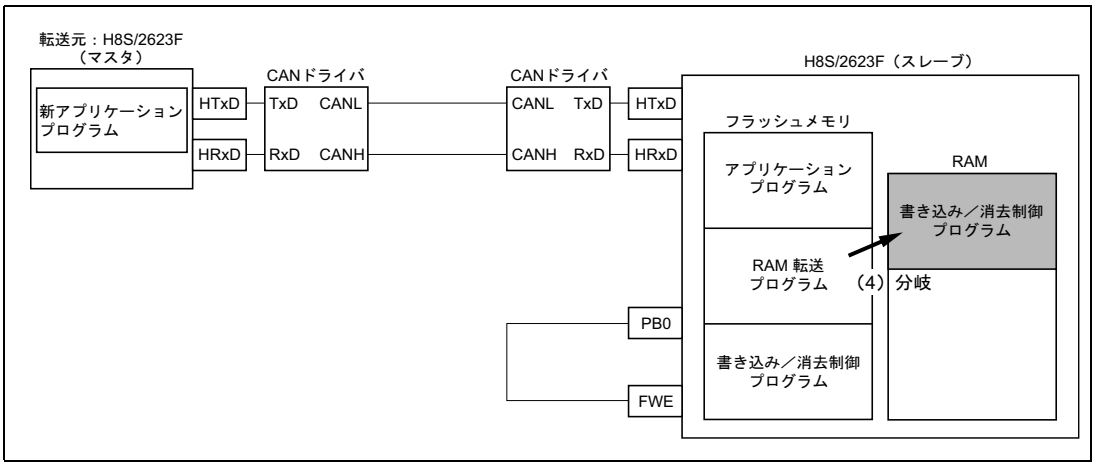


図 2.3 書き込み／消去制御プログラム起動

2.4 FWE 端子設定

(5) 転送元からFWE端子設定コマンドを受信します。

(6) 書き込み／消去制御プログラムはPB0を制御し、FWE端子を“1”に設定します。

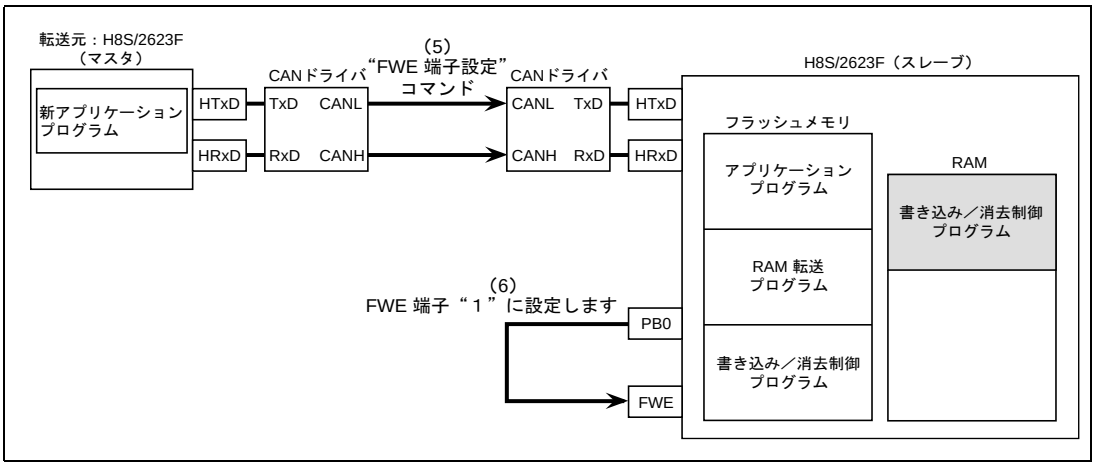


図 2.4 FWE 端子設定

2.5 フラッシュメモリ消去

(7) 転送元から消去コマンドを受信します。

(8) 書き込み／消去制御プログラムはフラッシュメモリの消去対象ブロックを消去します。

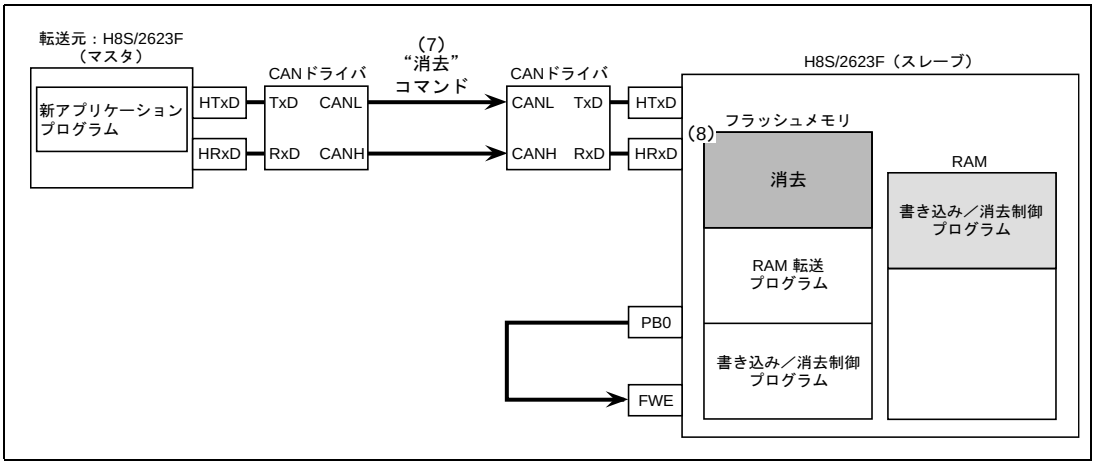


図 2.5 フラッシュメモリ消去

2.6 フラッシュメモリ書き込み

(9) 転送元から書き込みコマンド受信します。

(10) 書き込み／消去制御プログラムは転送元から新アプリケーションプログラムを受信し、フラッシュメモリに書き込みます。

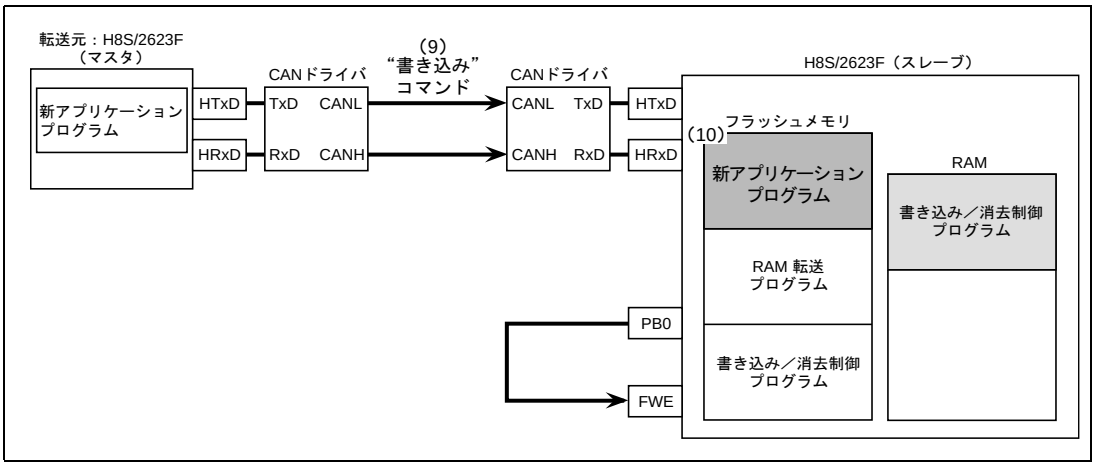


図 2.6 フラッシュメモリ書き込み

2.7 FWE 端子解除

(11) 転送元からFWE端子解除コマンドを受信します。

(12) 書き込み／消去制御プログラムPB0を制御し、FWE端子を“0”に設定します。

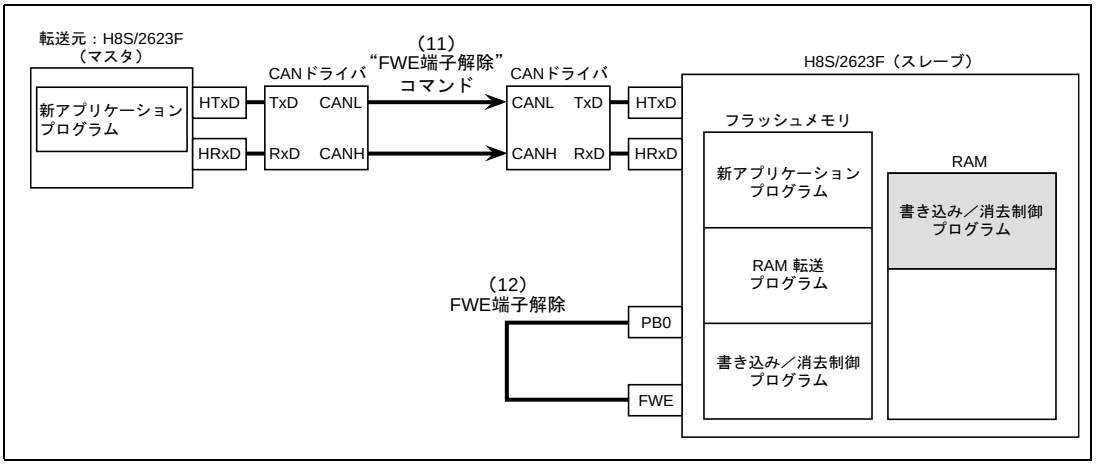


図 2.7 FWE 端子解除

2.8 新アプリケーションプログラム起動

(13) 書き込み／消去制御プログラムはフラッシュメモリ上の新アプリケーションプログラムに分岐します。

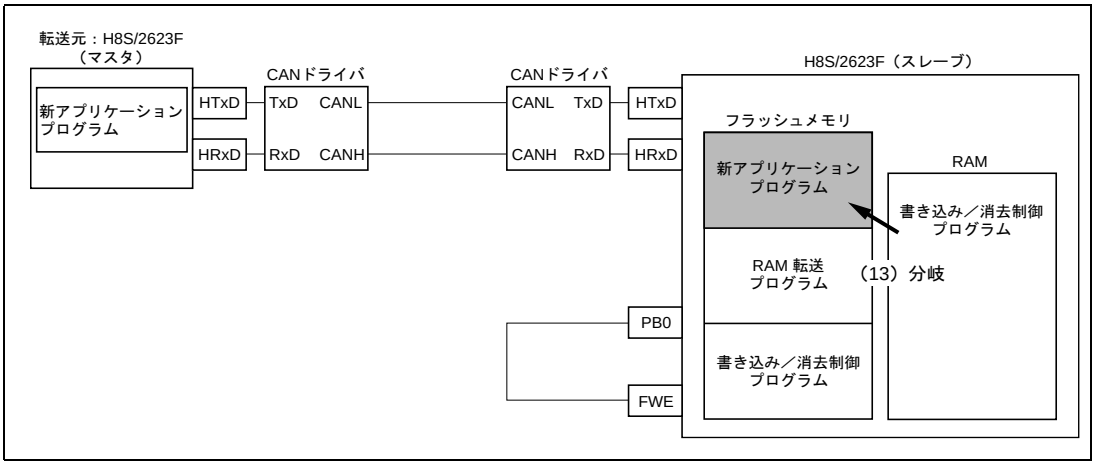


図 2.8 新アプリケーションプログラム起動

3. 動作説明

オンボード書き換え通信内容と、CANバスインタフェースを図3.1に示します。

なお、転送元が送信するパリティおよび、受信側のアクノリッジ送信については省略します。

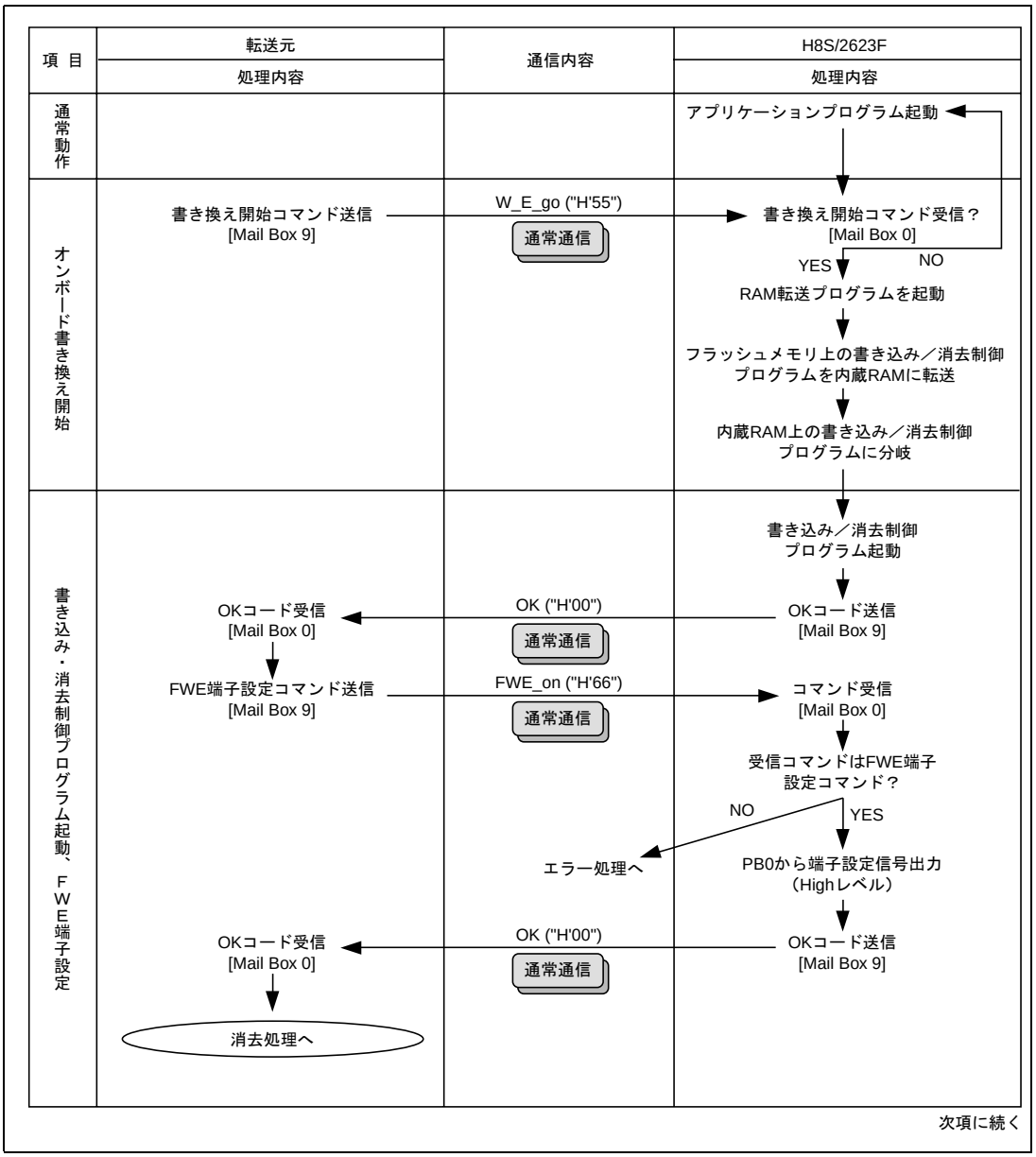
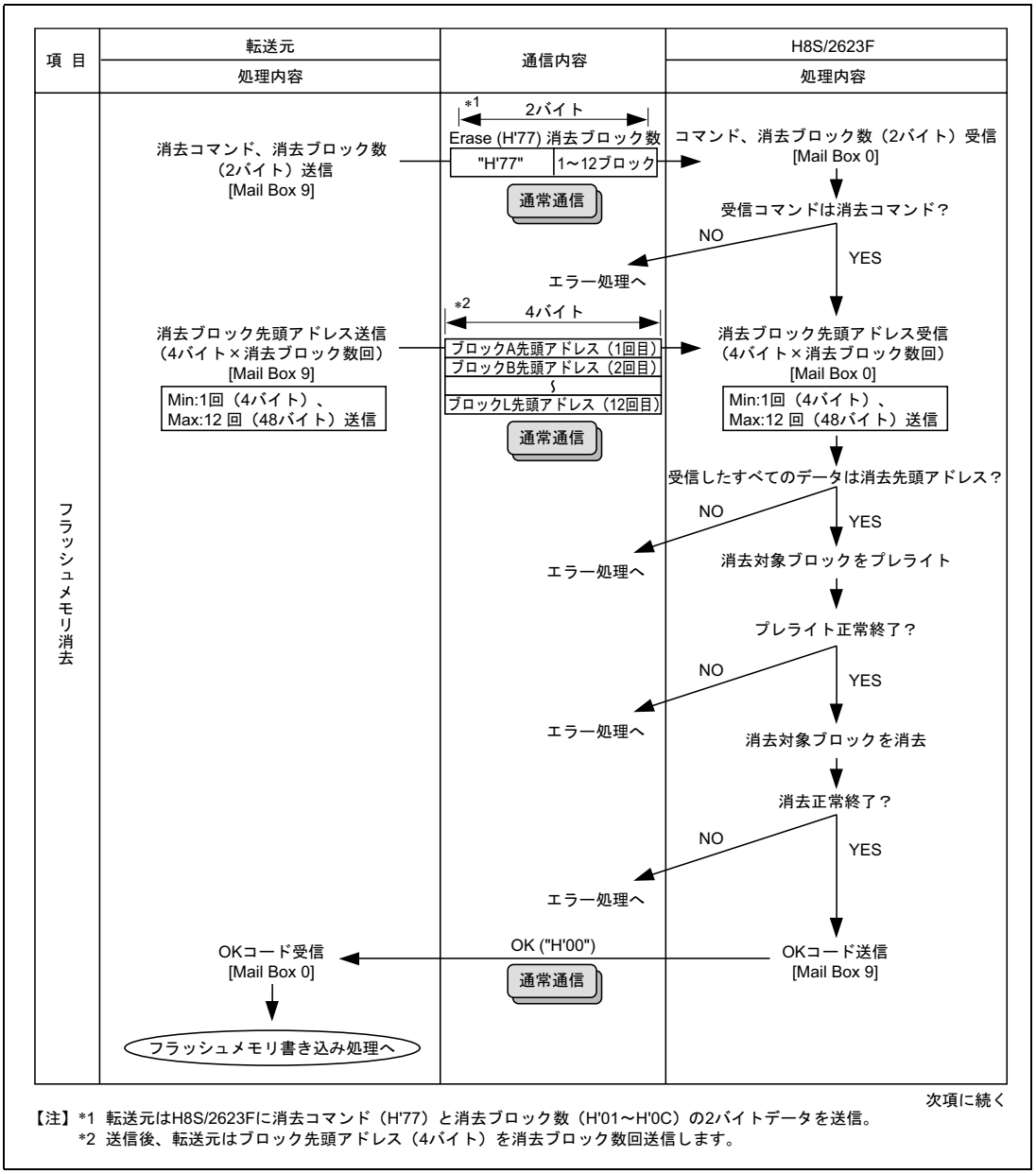
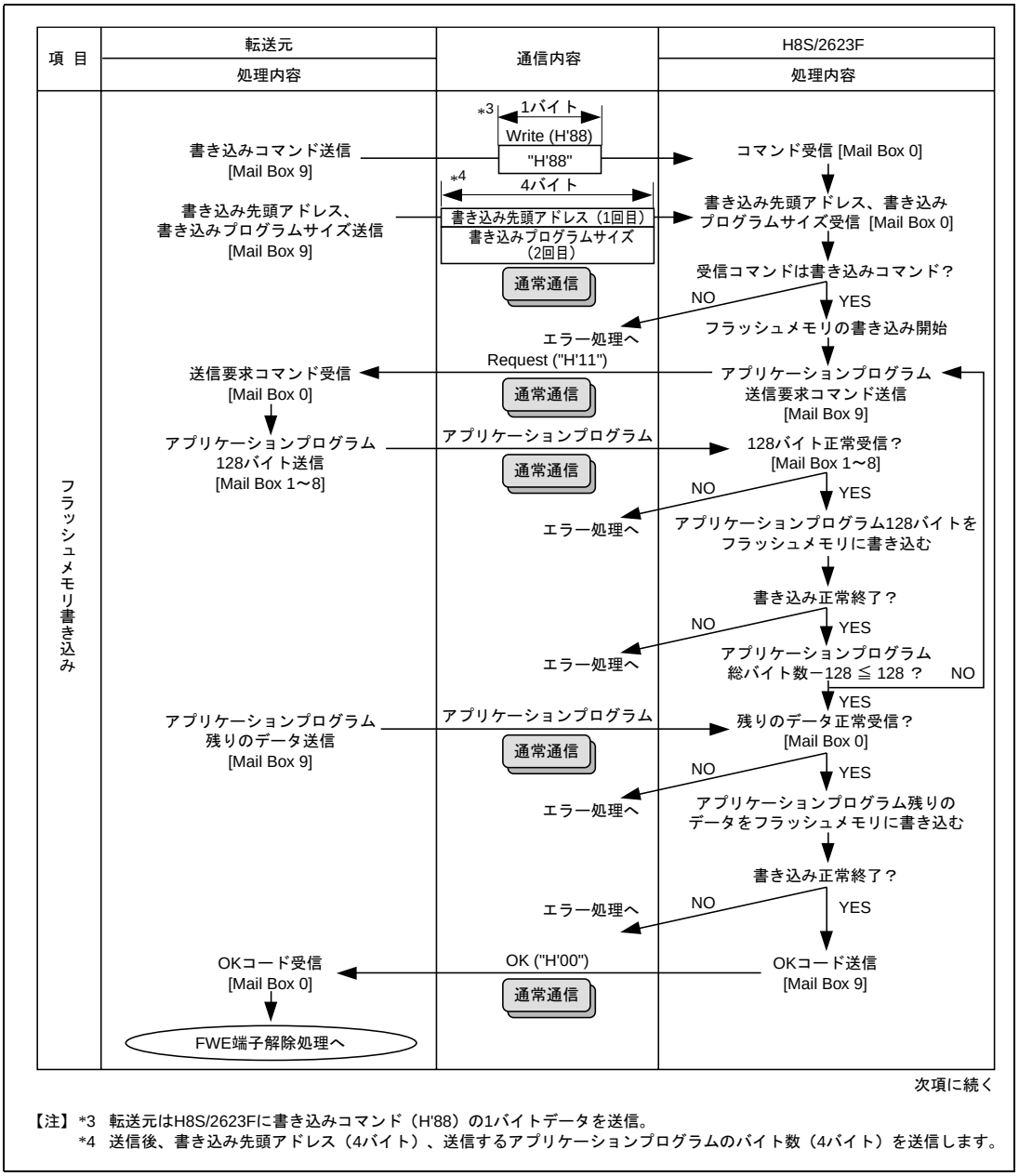


図 3.1　オンボード書き換え時の通信内容（1）



【注】*1 転送元はH8S/2623Fに消去コマンド（H'77）と消去ブロック数（H'01～H'0C）の2バイトデータを送信。
*2 送信後、転送元はブロック先頭アドレス（4バイト）を消去ブロック数回送信します。

図 3.2 オンボード書き換え時の通信内容（2）



【注】*3　転送元はH8S/2623Fに書き込みコマンド（H'88）の1バイトデータを送信。
*4　送信後、書き込み先頭アドレス（4バイト）、送信するアプリケーションプログラムのバイト数（4バイト）を送信します。

図 3.3　オンボード書き換え時の通信内容（3）

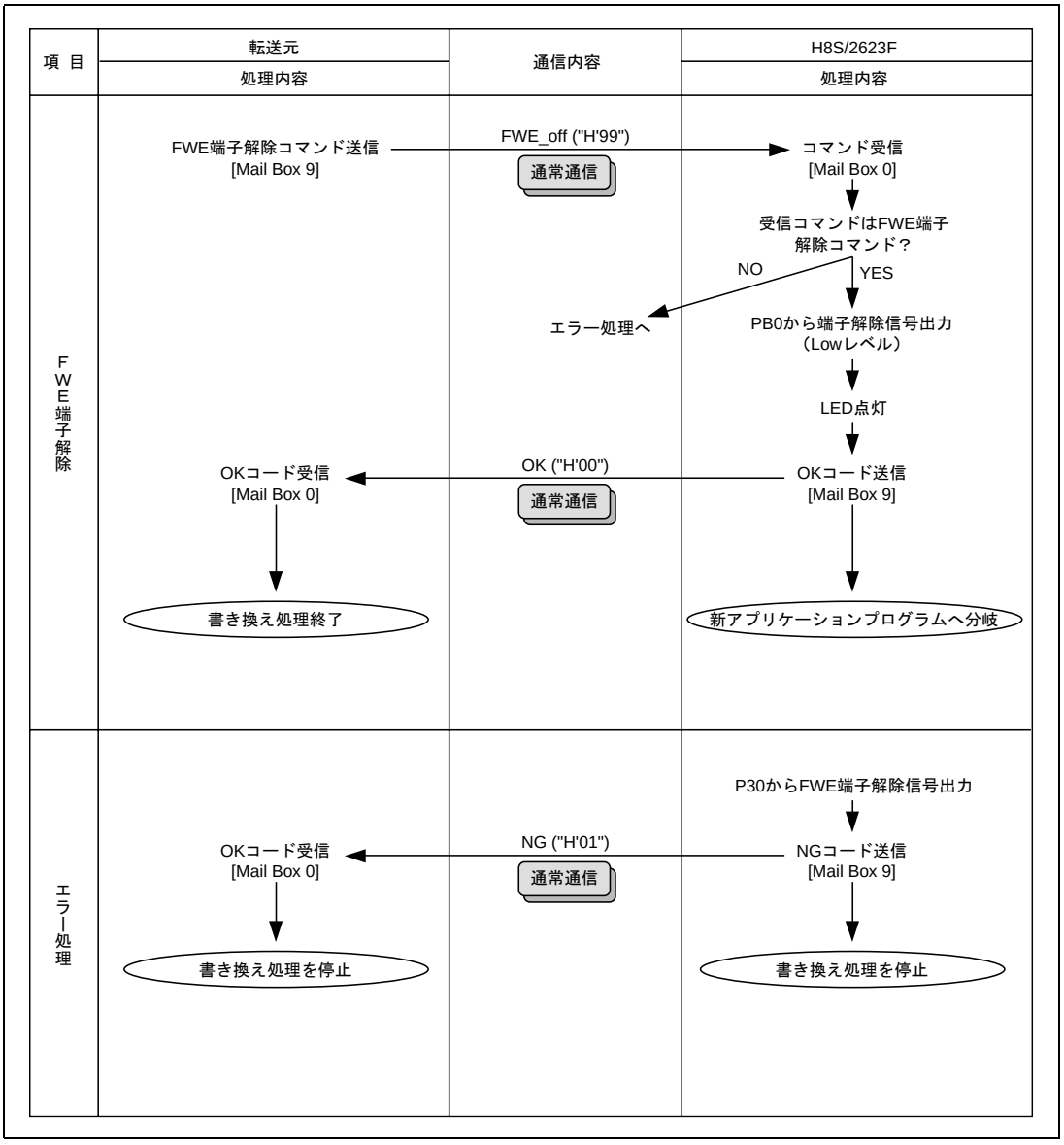


図 3.4　オンボード書き換え時の通信内容（4）

4. ソフトウェア説明

4.1 ユーザプログラムモード起動用プログラムの階層構造

フラッシュメモリ上で実行するユーザプログラムモード起動用プログラム（書き換え開始コマンド受信、フラッシュメモリ上の書き込み／消去制御プログラムを内蔵RAMに転送する処理）の階層構造を図4.1に示します。

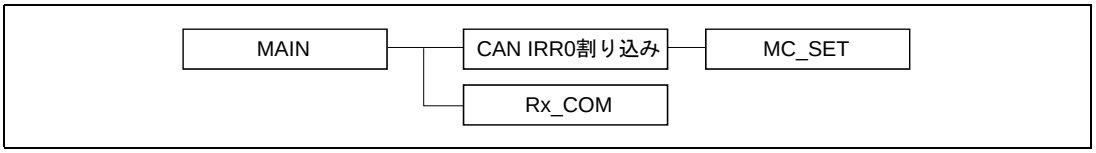


図 4.1 ユーザプログラムモード起動用プログラム階層構造

4.2 ユーザプログラムモード起動用プログラムのモジュール説明

表 4.1 ユーザプログラム起動用プログラムのモジュール説明

モジュール名	引 数		戻り値		機能割り付け
	内 容	レジスタ（データ長）	内 容	レジスタ（データ長）	
MAIN	—	—	—	—	メインルーチンです。
RX_COM	—	—	受信データ (1 バイト)	R2L (1 バイト)	転送元から CAN バスを介しデータを受信します。
IRR0	—	—	—	—	CAN IRR0 割り込みルーチンです。
MC_SET	MCn レジスタのアドレス	ER5 (4 バイト)	MCn レジスタ	@ER5 (3 バイト)	MCn(メッセージコントロール)を設定します。
	データ長	E6(2 バイト)			
	Identifier	R6L(1 バイト)			

4.3 書き込み／消去制御プログラムの階層構造

内蔵RAM上で実行する書き込み／消去制御プログラムの階層構造を図4.2に示します。

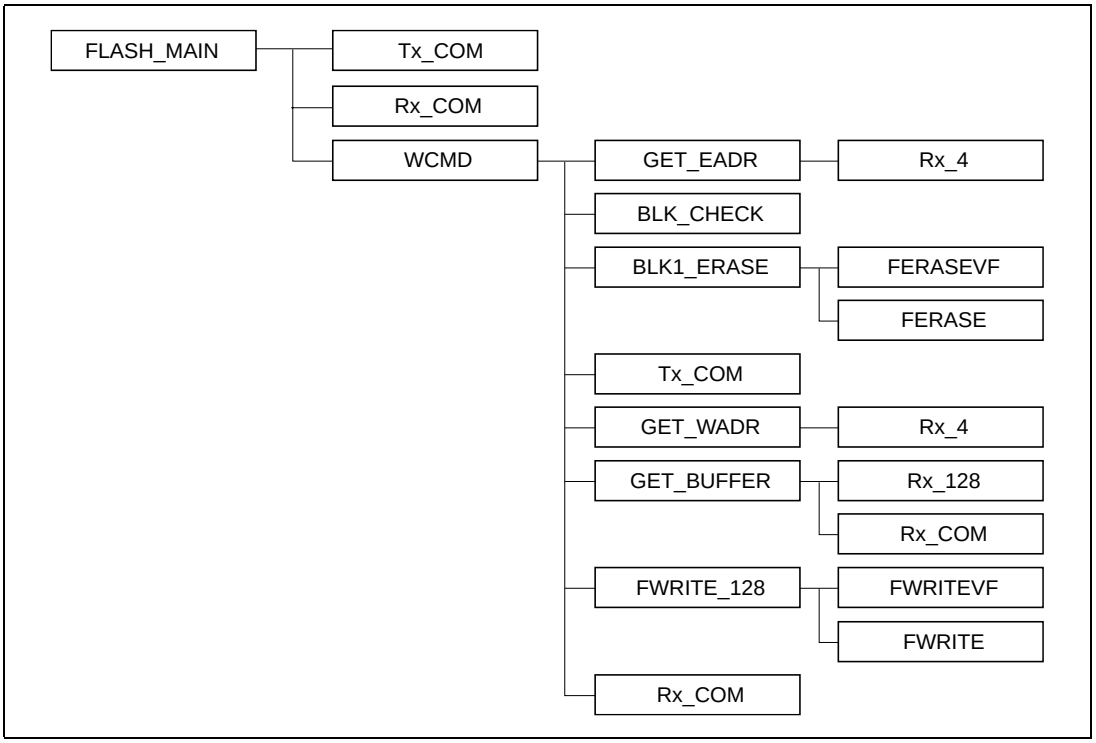


図 4.2　書き込み／消去制御プログラムの階層構造

4.4 書き込み／消去制御プログラムのモジュール説明

表 4.2 書き込み／消去制御プログラムのモジュール説明

モジュール名	引 数		戻り値		機能割り付け
	内 容	レジスタ（データ長）	内 容	レジスタ（データ長）	
FLASH_MAIN	—	—	—	—	書き込み／消去制御プログラムのメインルーチンです。
RX_COM	—	—	受信データ (1 バイト)	R2L (1 バイト)	転送元から CAN バスを介しデータを受信します。
TX_COM	送信データ	R2L (1 バイト)	—	—	転送元へ CAN バスを介しデータを送信します。
WCMD	—	—	—	—	書き込み／消去制御ルーチンです。 転送先とのコマンドの送受信も行います。
GET_EADR	—	—	正常時“0”, エラー時“1”	R0L (1 バイト)	転送元から CAN バスを介し消去ブロック数、消去先頭アドレスを受信します。
			消去ブロック数	ERASEBLOCK (1 バイト)	
			消去先頭アドレス	E_ADR (4～48 バイト)	
RX_4	受信データ格納アドレス	ER3 (4 バイト)	格納アドレス	@ER3 (4 バイト)	転送元から CAN バスを介し受信したデータを @ER3 に格納します。
BLK_CHECK	消去先頭アドレス	ER3 (4 バイト)	正常時“0”, エラー時“1”	R0L (1 バイト)	消去先頭アドレスがどのブロックに相当するかをチェックし、対応する FLMCR,ERB レジスタのアドレスを汎用レジスタにロードします。また、消去対象ブロック No.、消去ベリファイ先頭と最終アドレスをそれぞれワーク RAM に格納します。
			EBR レジスタのアドレス	ER5 (4 バイト)	
			FLMCR レジスタのアドレス	ER6 (4 バイト)	
			消去対象ブロック No.	BLK_NO (1 バイト)	
			消去ベリファイ先頭アドレス	EVF_ST (4 バイト)	
			消去ベリファイ最終アドレス	EVF_ED (4 バイト)	
BLK 1 _ERASE	—	—	正常時“0”, エラー時“1”	R0L (1 バイト)	フラッシュ消去処理を行います。
FERASEVF	FLMCR レジスタのアドレス	ER6 (4 バイト)	正常時“0”, エラー時“1”	R0L (1 バイト)	消去ベリファイ先頭アドレスから、最終アドレスのデータを読み出し、正常に消去されているか確認します。
	消去ベリファイ先頭アドレス	EVF_ST (4 バイト)			
	消去ベリファイ最終アドレス	EVF_ED (4 バイト)			
FERASE	EBR レジスタのアドレス	ER5 (4 バイト)	正常時“0”, エラー時“1”	R0L (1 バイト)	消去対象ブロック No.のエリアを消去します。
	FLMCR レジスタのアドレス	ER5 (4 バイト)			
	消去対象ブロック No.	BLK_NO (1 バイト)			

モジュール名	引　数		戻り値		機能割り付け
	内　容	レジスタ（データ長）	内　容	レジスタ（データ長）	
GET_WADR	—	—	正常時“0”, エラー時“1”	R0L （1 バイト）	転送元から CAN バスを介し、書き込み先頭アドレス、書き込みデータサイズを受信します。
			書き込み先頭アドレス	W_ADR （4 バイト）	
			書き込みデータサイズ	RESTSIZE （4 バイト）	
GET_BUFFER	書き込みデータサイズ	RESTSIZE （4 バイト）	書き込みデータサイズ	RESTSIZE （4 バイト）	書き込みデータエリアに 128 バイトのデータを格納するルーチンです。
FWRITE_128	書き込み先頭アドレス	W_ADR （4 バイト）	正常時“0”, エラー時“1”	R0L （1 バイト）	128 バイト単位でのフラッシュ書き込み処理を行います。
	書き込みデータ	W_BUFF （128 バイト）	再書き込みデータ	BUFF （128 バイト）	
FWRITEVF	FLMCR レジスタのアドレス	ER6 （4 バイト）	正常時“0”, エラー時“1”	R0L （1 バイト）	再書き込み演算、追加書き込み演算を行い、128 バイトの書き込みデータが正常に書き込まれているか確認します。
	再書き込みデータ	BUFF （128 バイト）	再書き込みデータ	BUFF （128 バイト）	
	書き込みデータ	W_BUFF （128 バイト）	追加書き込みデータ	OW_BUFF （128 バイト）	
	追加書き込みデータ	OW_BUFF （128 バイト）			
FWRITE	P ビットセット時間	ER3 （4 バイト）	—	—	128 バイト単位の書き込み(Flash 電圧印加)を行います。
	FLMCR レジスタのアドレス	ER6 （4 バイト）			
	書き込み先頭アドレス	W_ADR （4 バイト）			

4.5 使用 RAM 説明

表4.3に書き込み制御プログラムが引数、戻り値の格納エリアおよびワークエリアとして使用する内蔵RAMを示します。

表 4.3 使用 RAM 説明

ラベル名	機 能	データ長	使用モジュール名
ERASEBLOCK	消去ブロック数を格納（戻り値）	1 バイト	GET_EADR
E_ADR	消去先頭アドレス（戻り値）	4～48 バイト	GET_EADR
BLK_NO	消去対象ブロック No.を格納（引数,戻り値）	1 バイト	BLK_CHECK. FERASE
EVF_ST	消去ベリファイ先頭アドレスを格納（引数,戻り値）	4 バイト	BLK_CHECK, FERASEVF
EVF_ED	消去ベリファイ最終アドレスを格納（引数,戻り値）	4 バイト	BLK_CHECK, FERASEVF
W_ADR	書き込み先頭アドレスを格納（引数,戻り値）	4 バイト	GET_WADR, FWRITE_128, FWRITE
RESTSIZE	書き込みデータサイズを格納（引数,戻り値）	4 バイト	GET_WADR, GET_BUFFER
W_BUFF	128 バイトの書き込むプログラムデータを格納（引数,戻り値）	128 バイト	RX_128, FWRITE_128, FWRITEVF
BUFF	再書き込み演算結果を格納（引数,戻り値）	128 バイト	FWRITE_128, FWRITEVF
OW_BUFF	追加書き込み演算結果を格納（引数,戻り値）	128 バイト	FWRITEVF

5. フローチャート

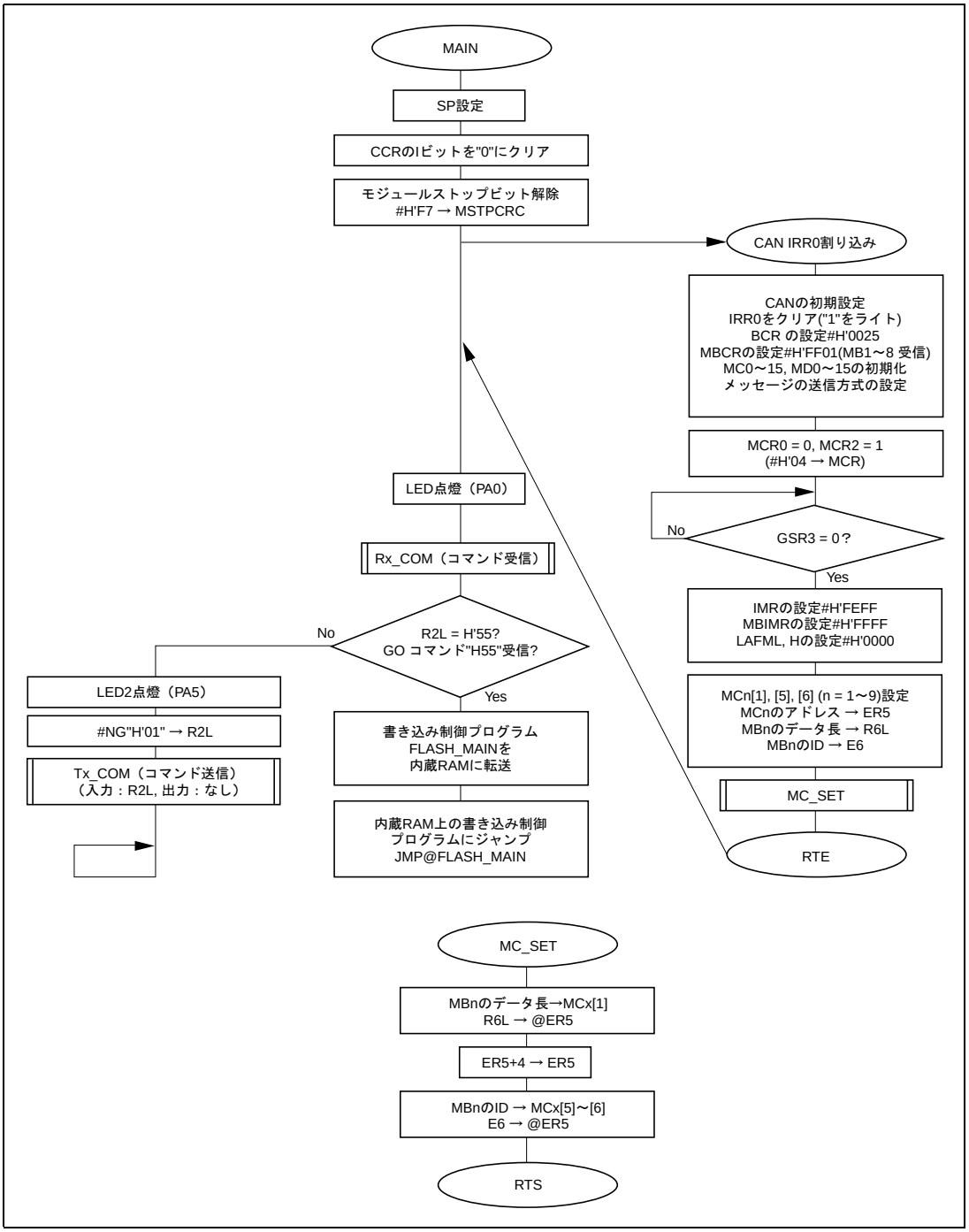


図 5.1　フローチャート(1)

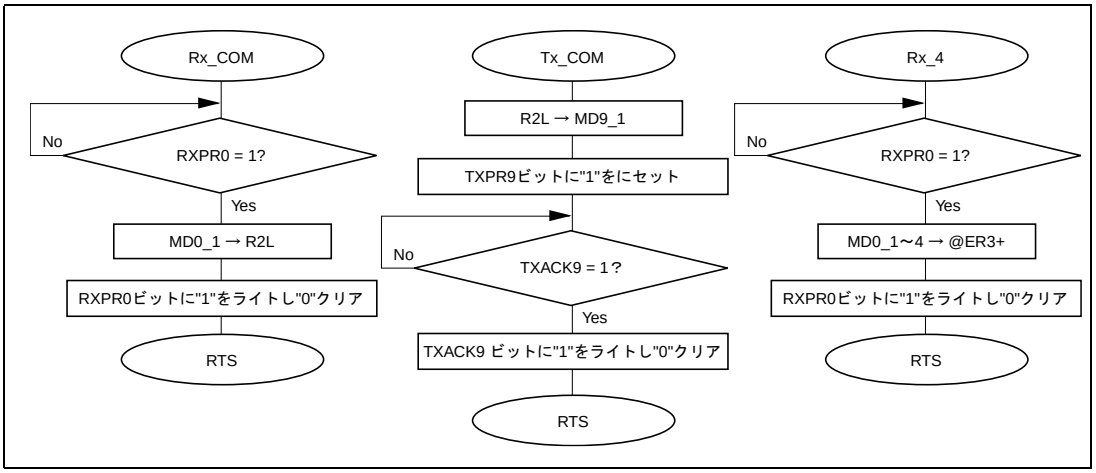


図 5.2　フローチャート(2)

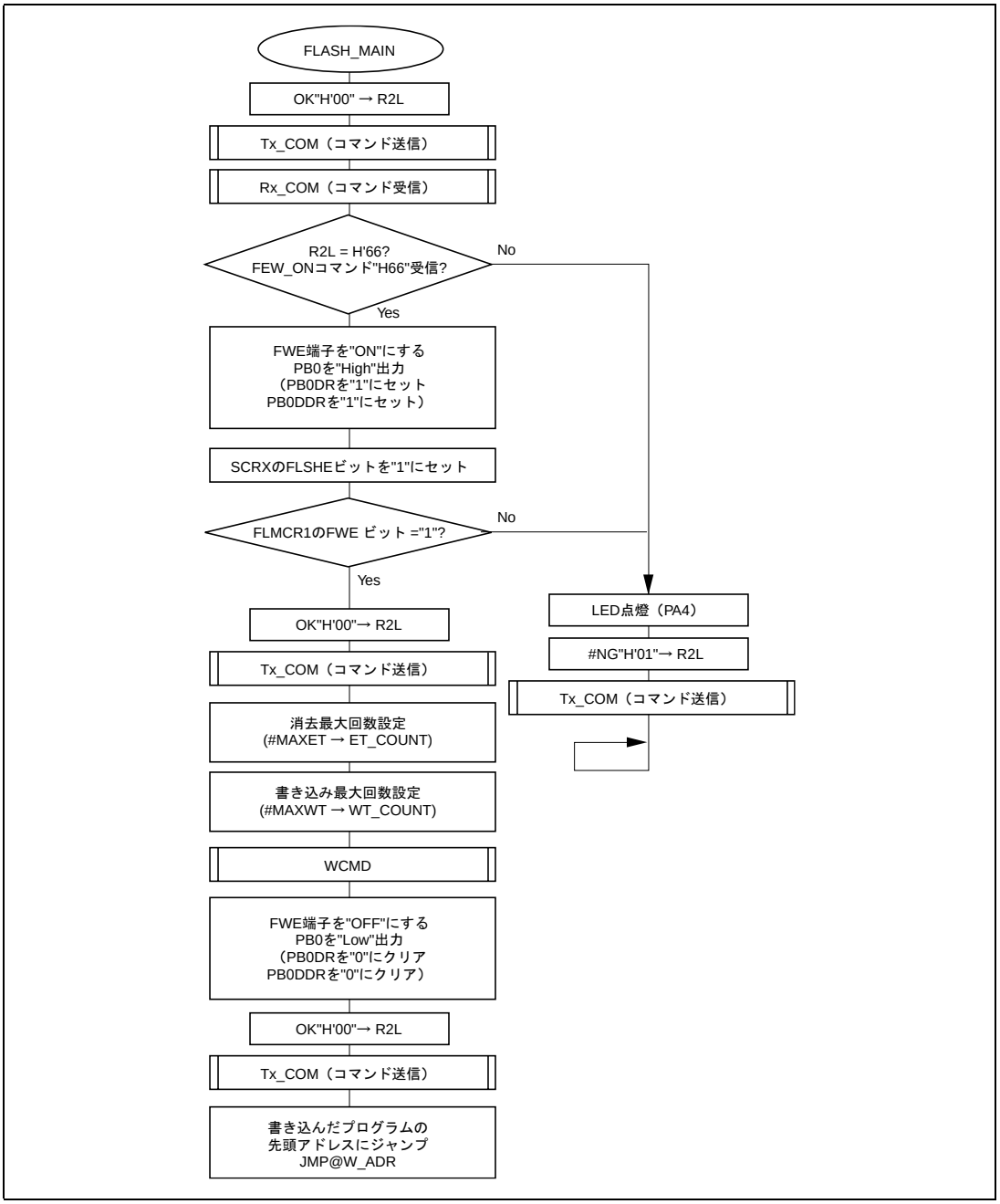


図 5.3　フローチャート(3)

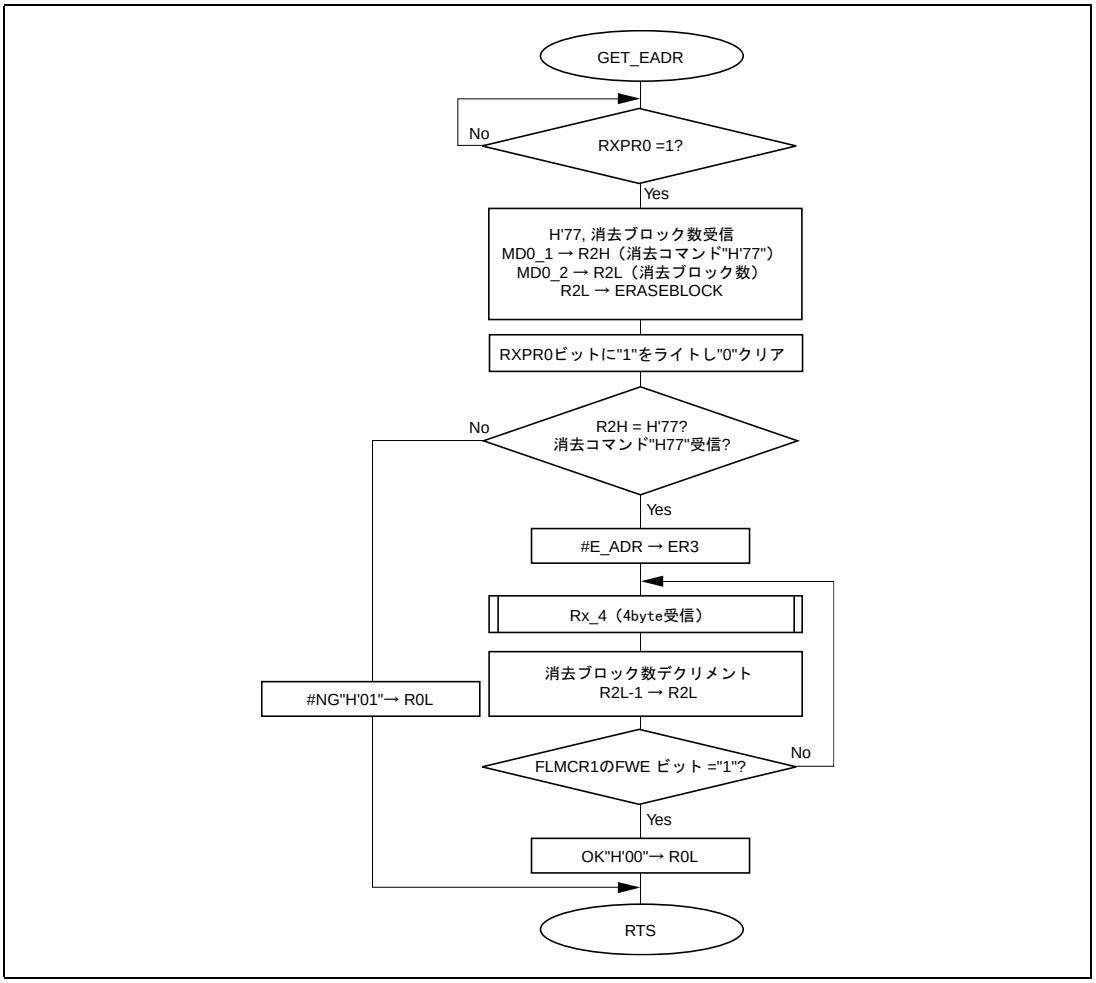


図 5.4　フローチャート(4)

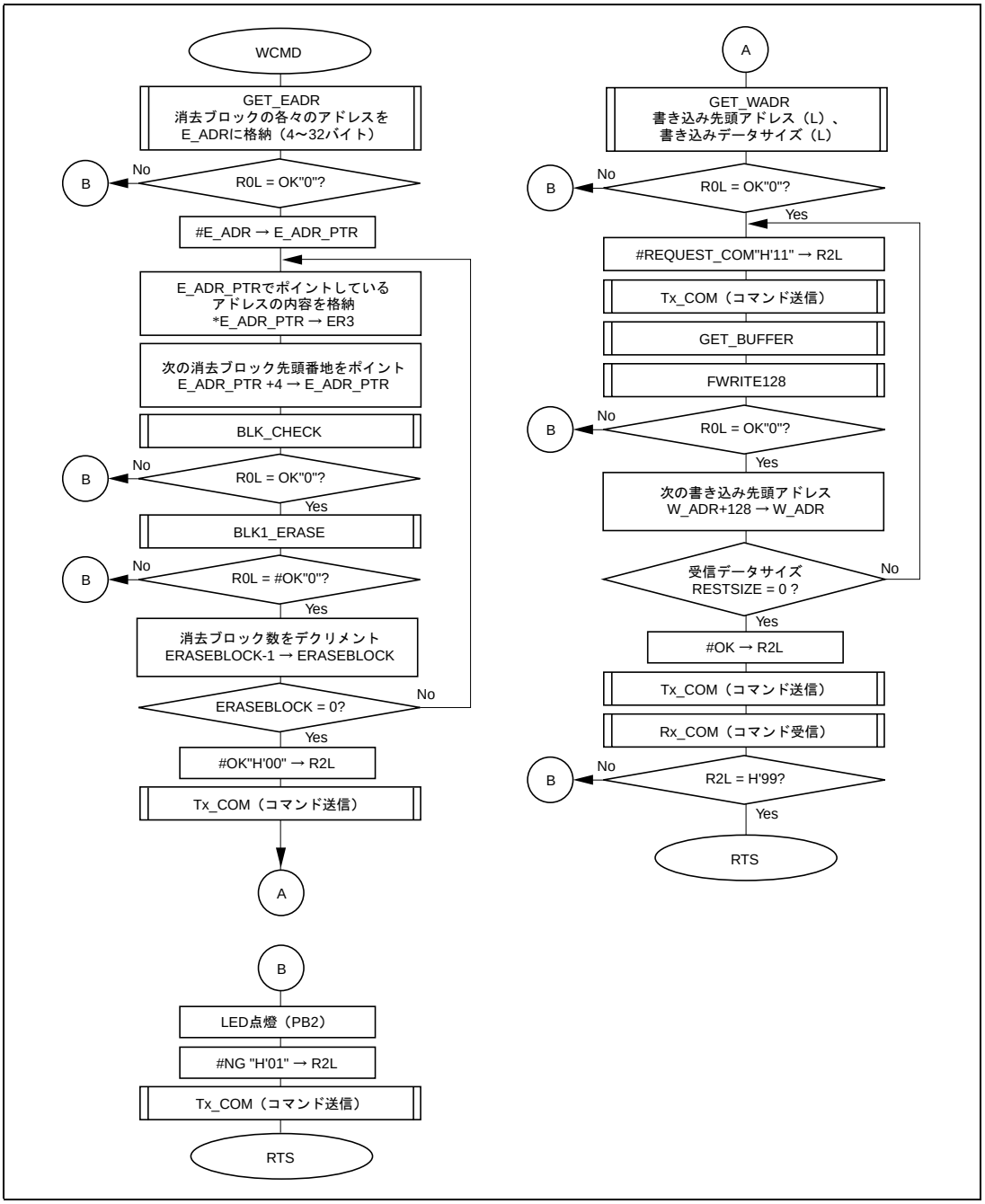


図 5.5　フローチャート(5)

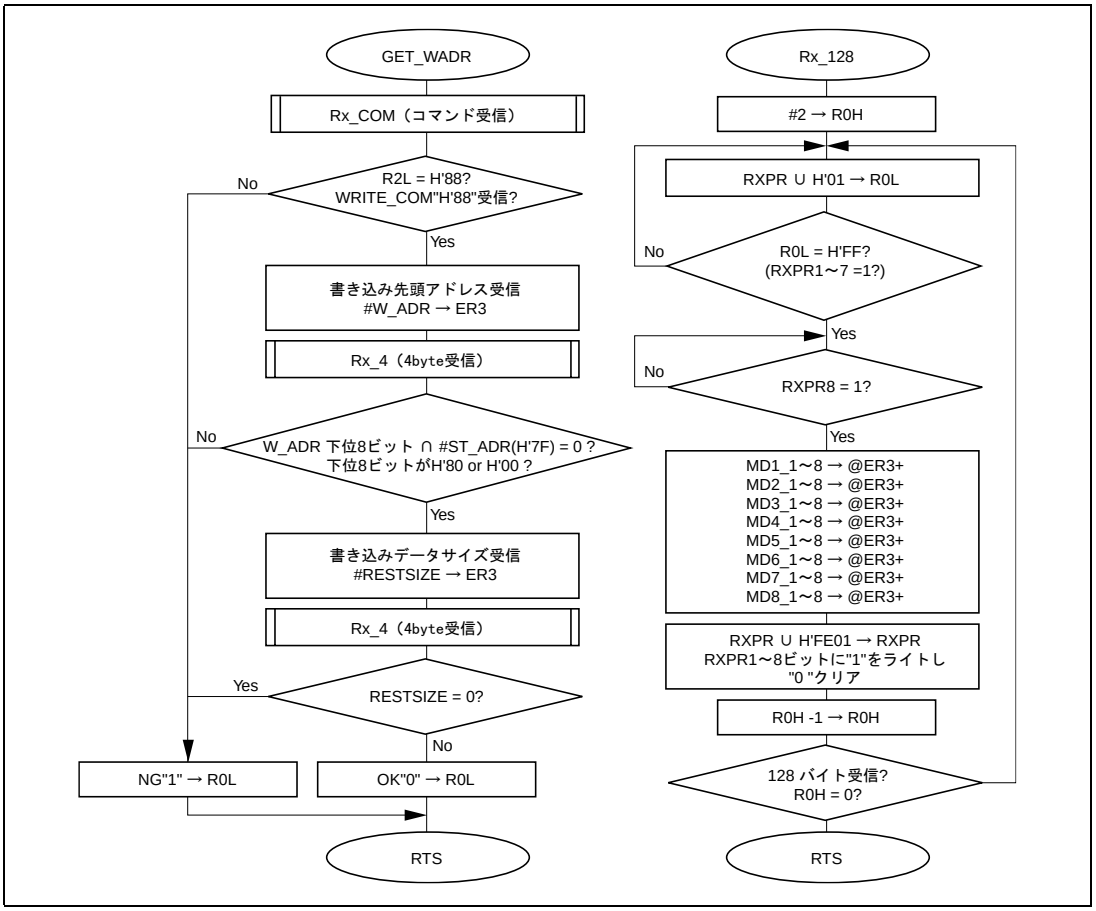


図 5.6　フローチャート(6)

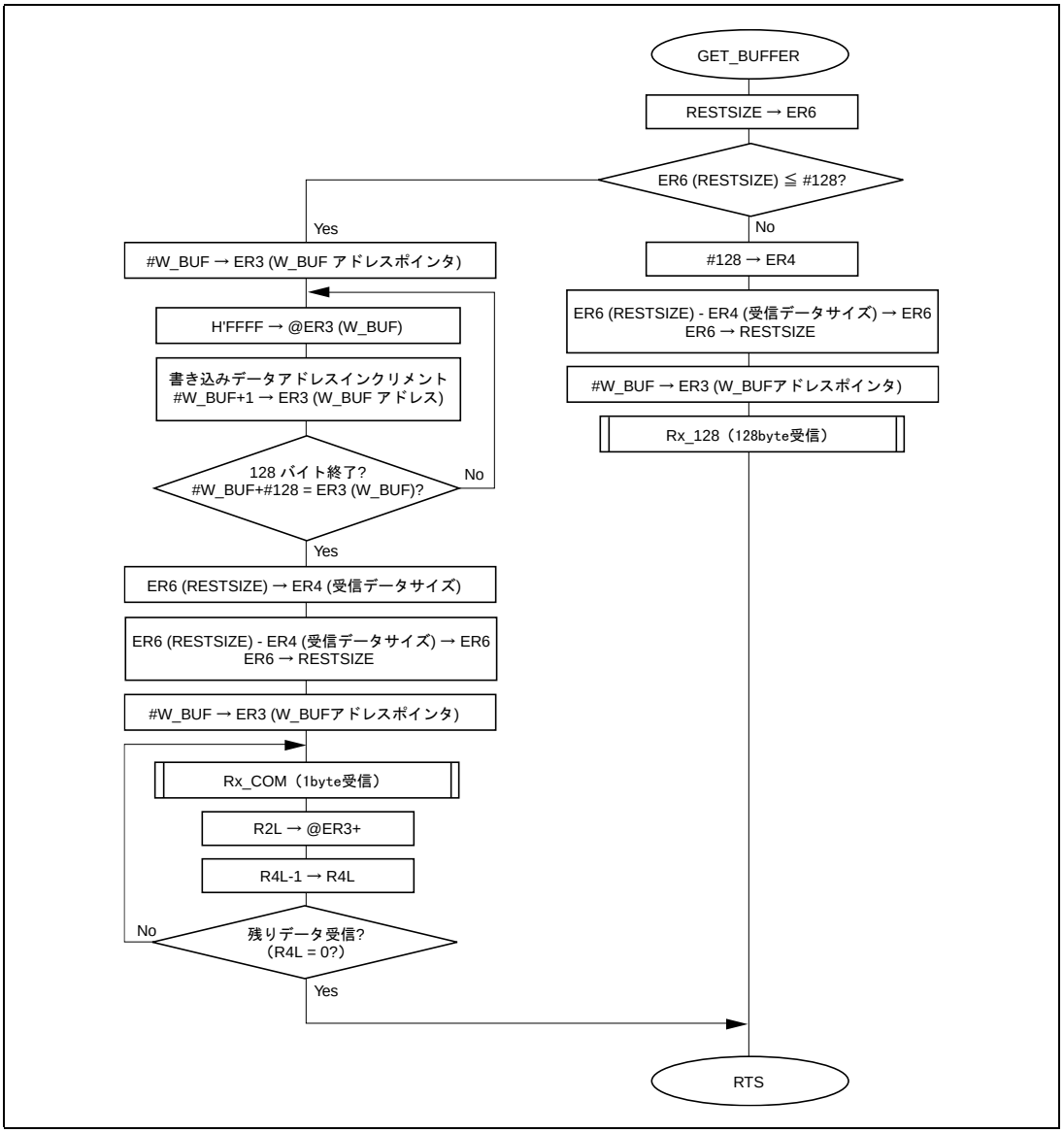


図 5.7　フローチャート(7)

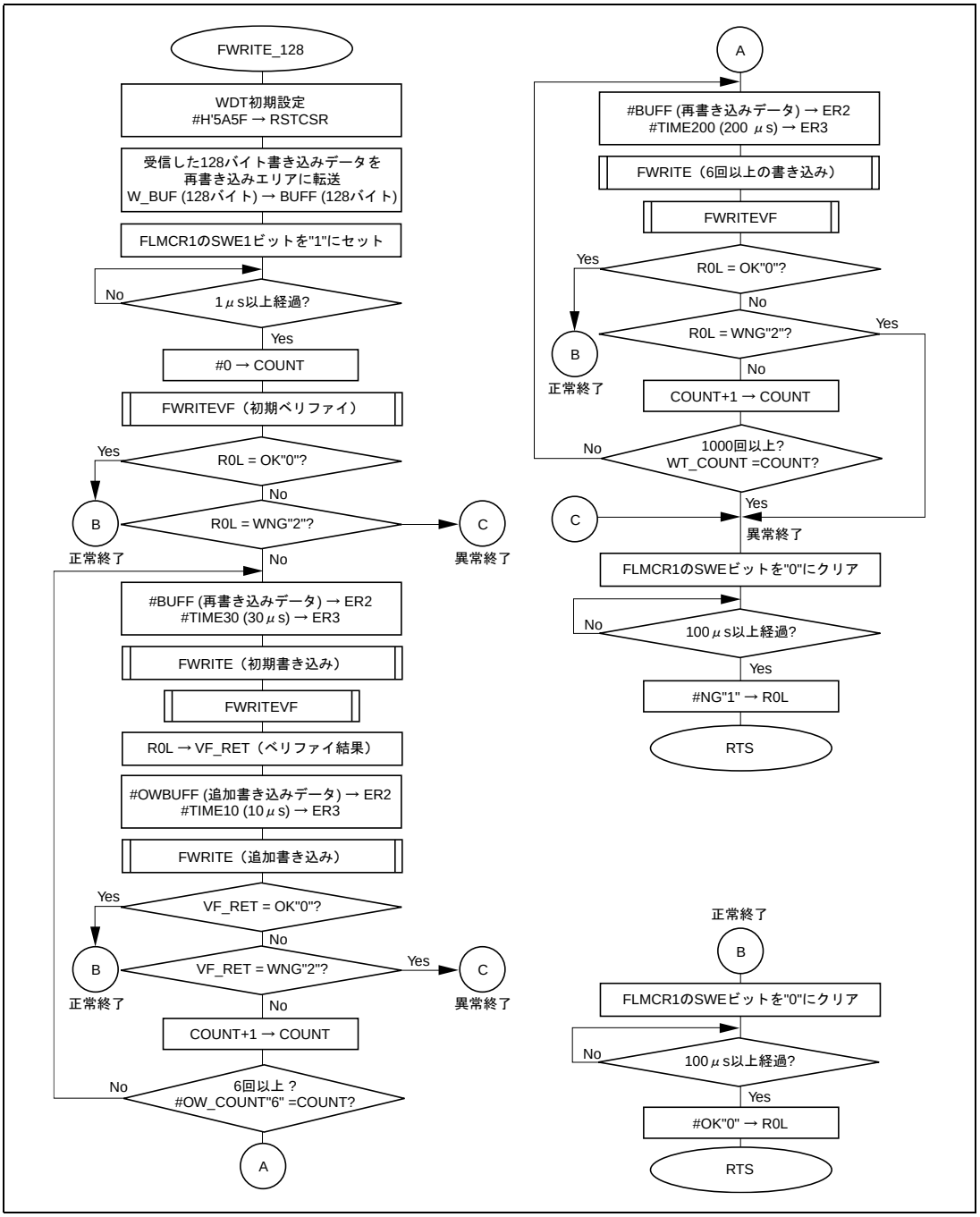


図 5.8　フローチャート(8)

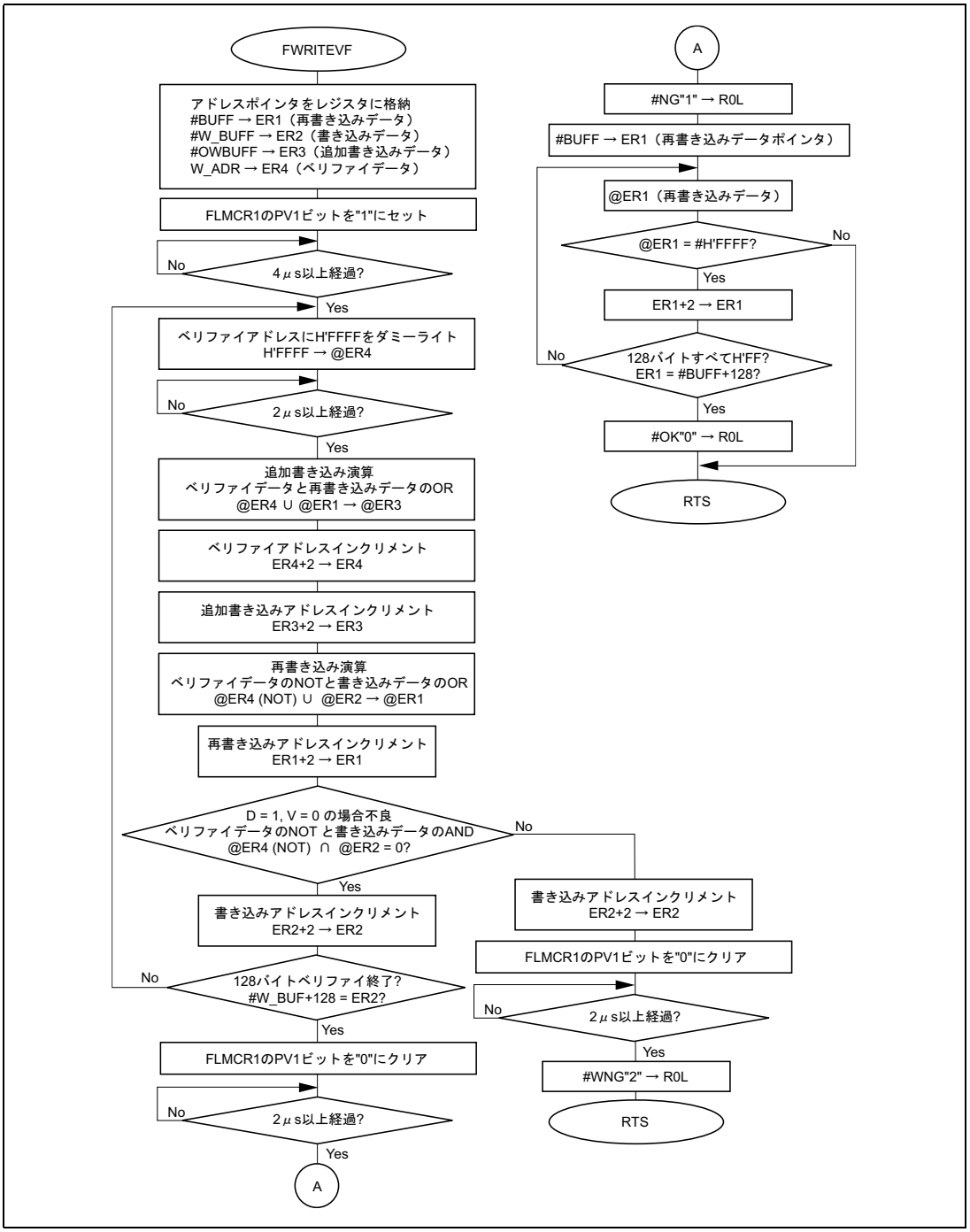


図 5.9　フローチャート(9)

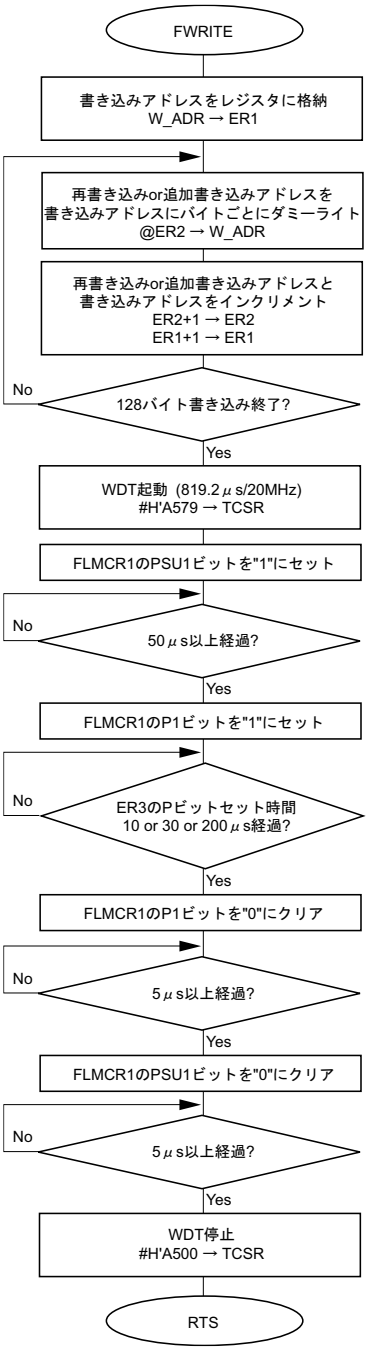


図 5.10　フローチャート(10)

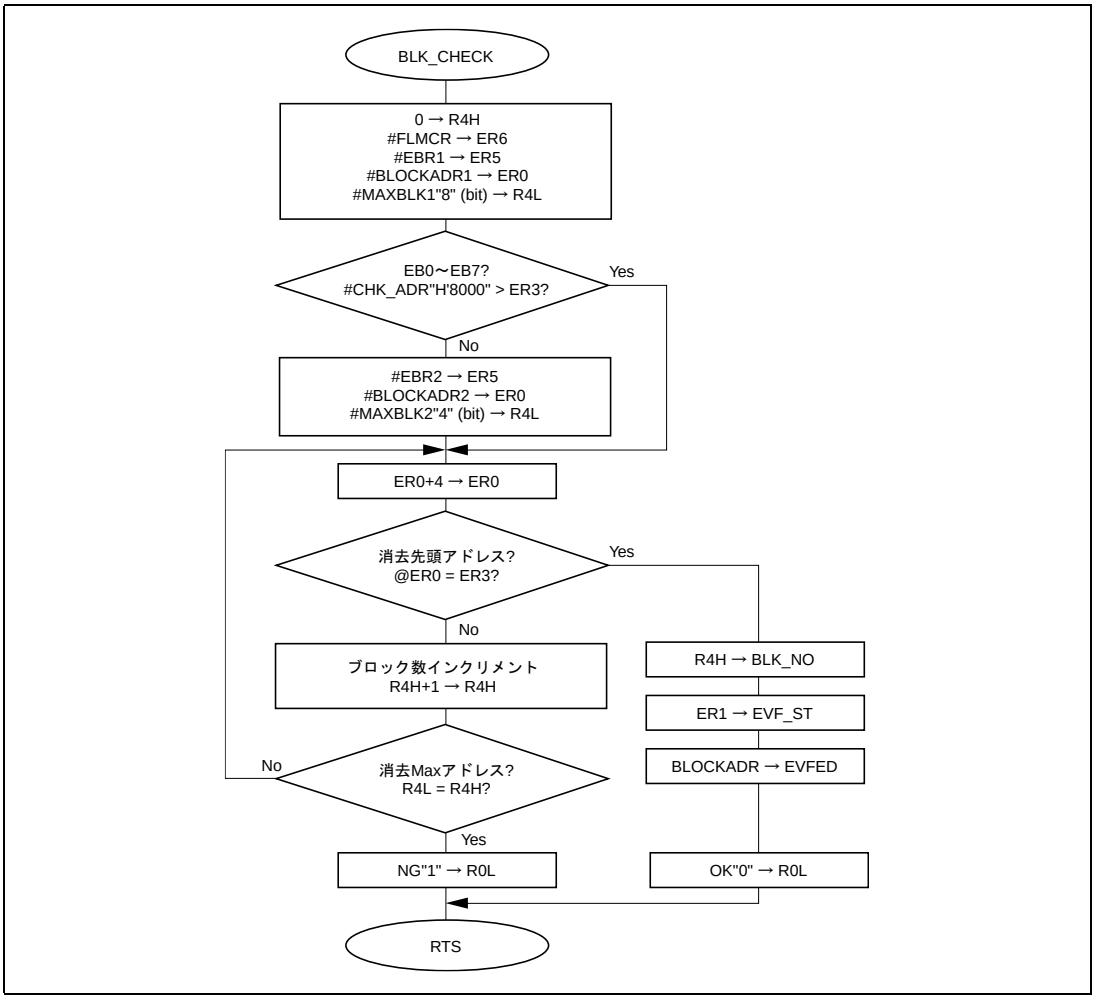


図 5.11　フローチャート(11)

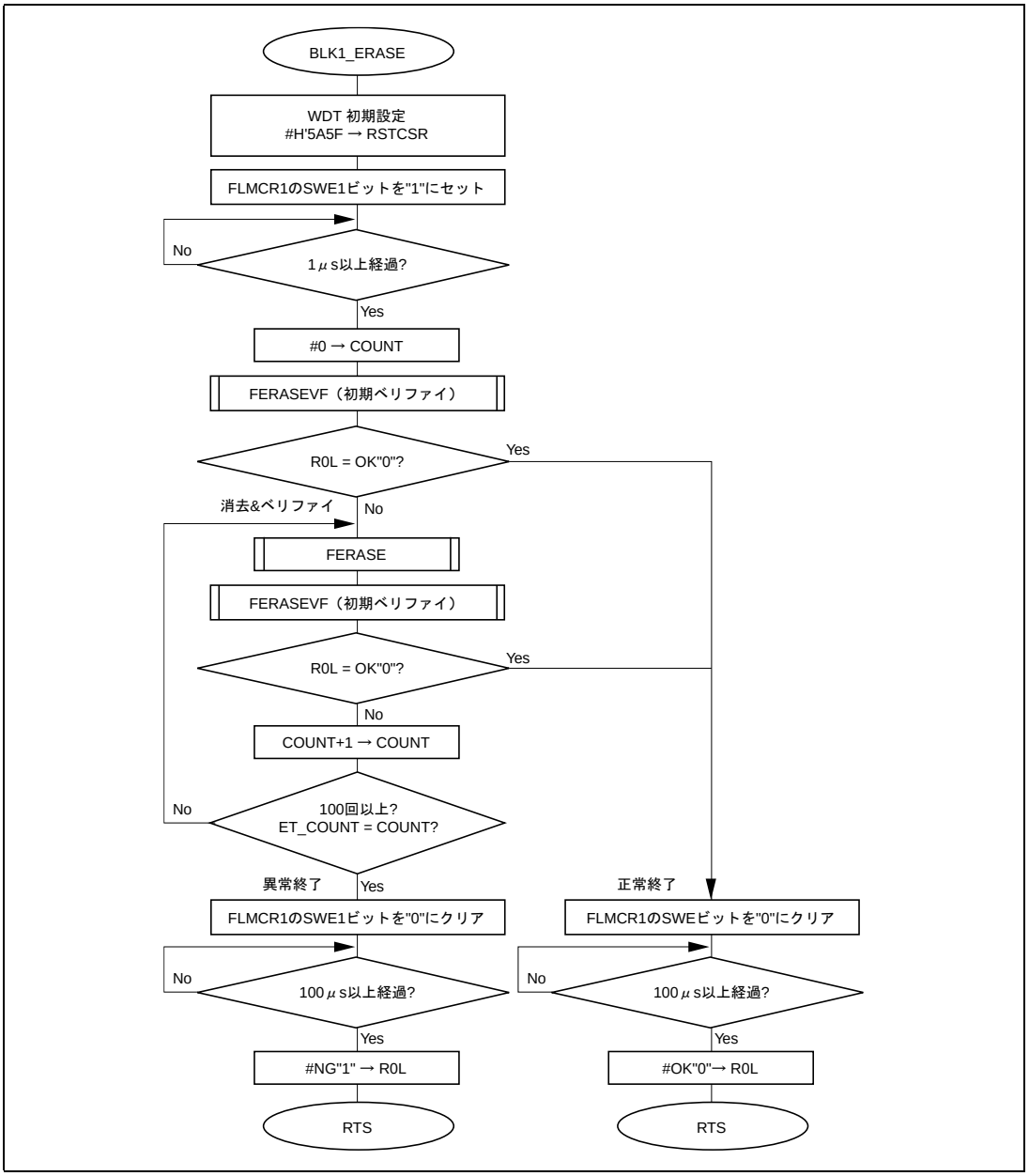


図 5.12　フローチャート(12)

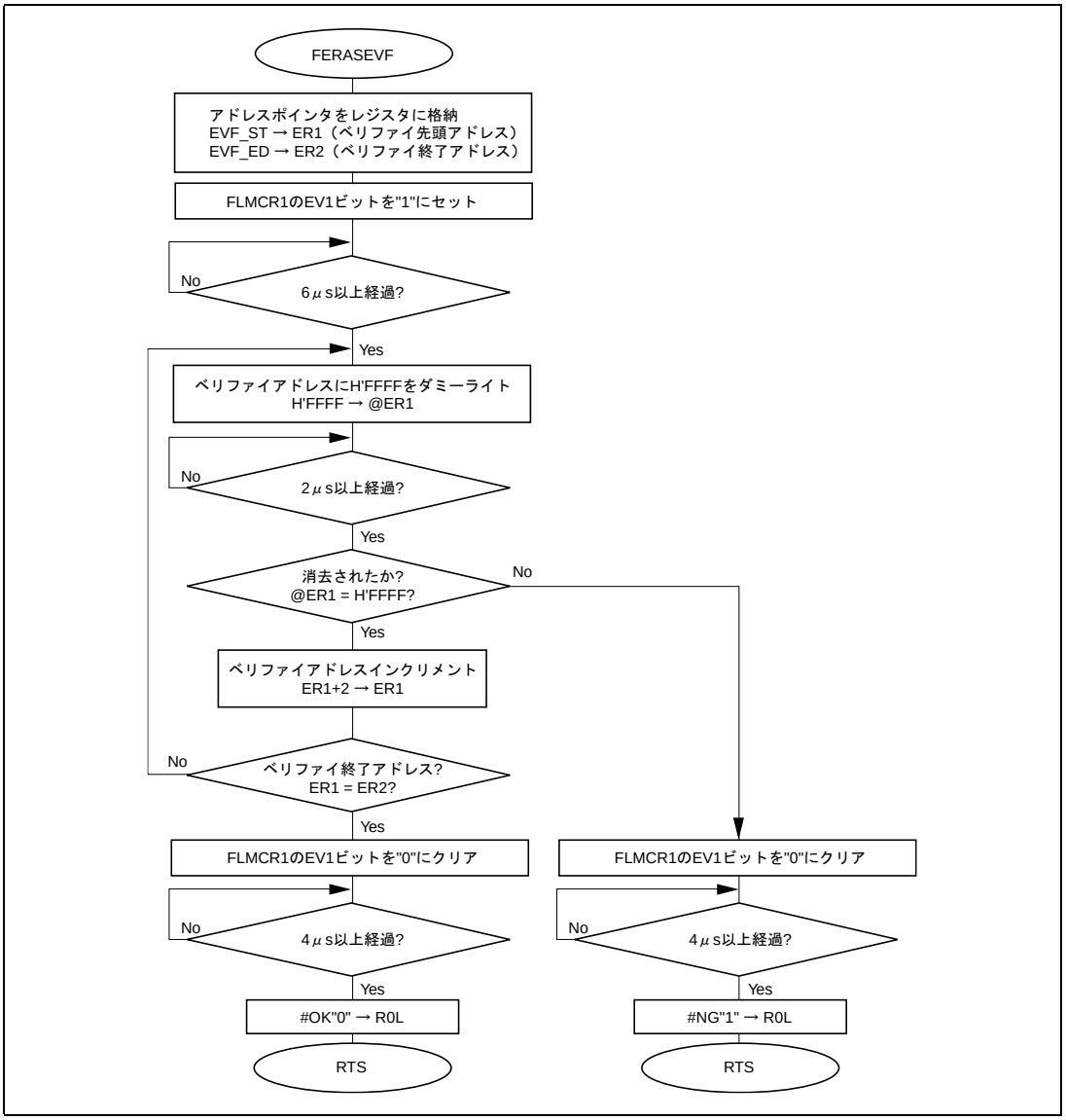


図 5.13　フローチャート(13)

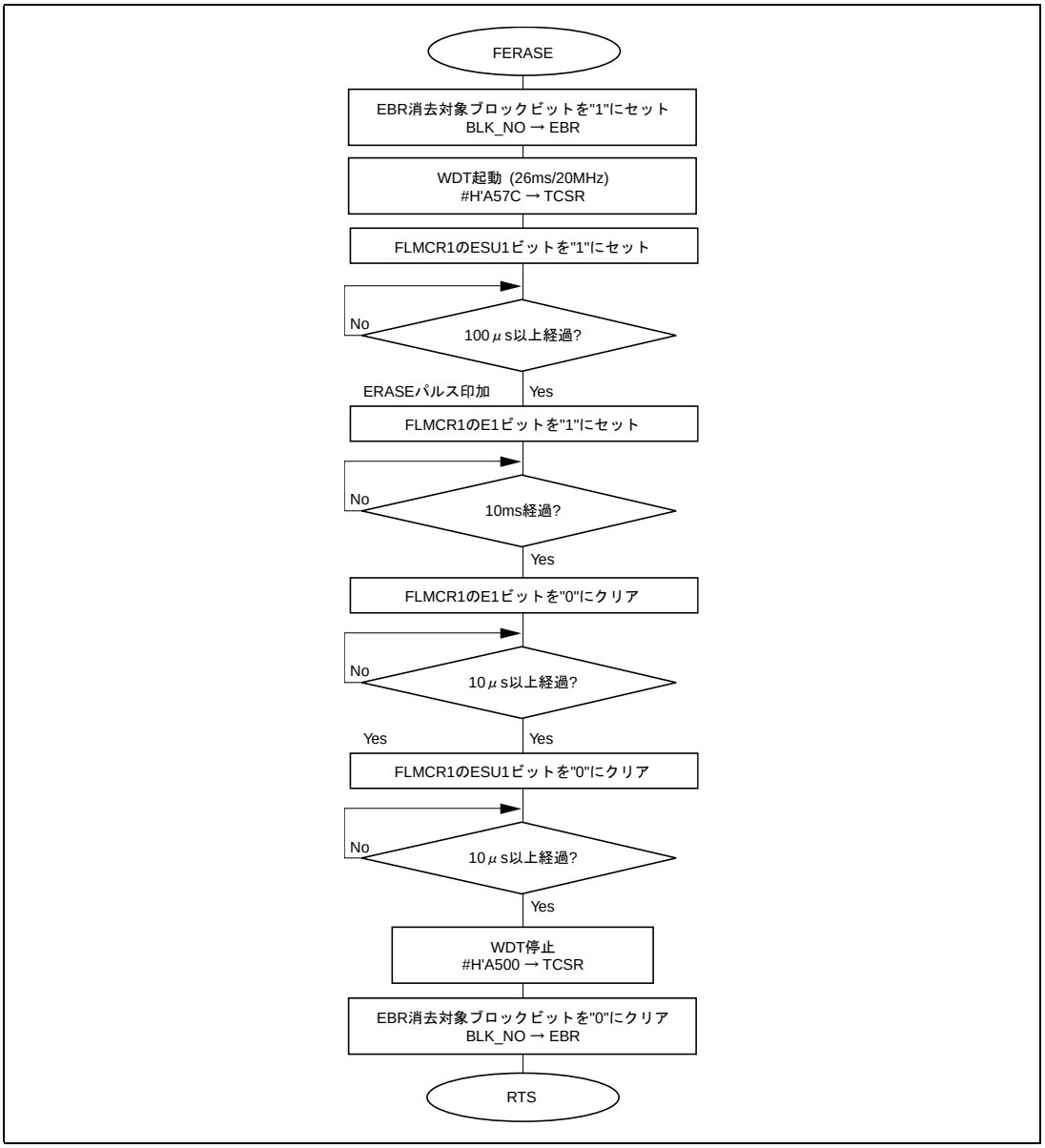


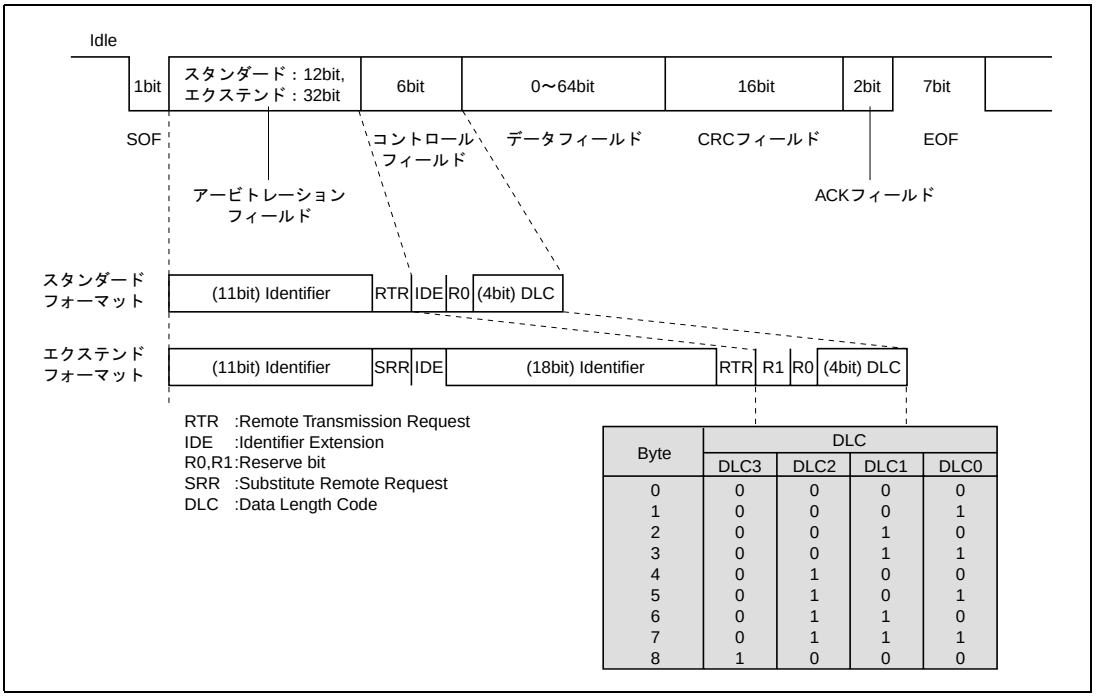
図 5.14　フローチャート(14)

6. ハードウェア説明

6.1 CAN 転送フォーマット

データフォーマット仕様を図6.1に示します。

本書き換え例では、スタンダードフォーマット (Identifier:11bit)を使用します。



Byte	DLC			
	DLC3	DLC2	DLC1	DLC0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0

図 6.1 データフォーマット

6.2 メモリマップ

H8S/2623Fの通常動作時、書き換え動作時のメモリマップを図6.2に示します。

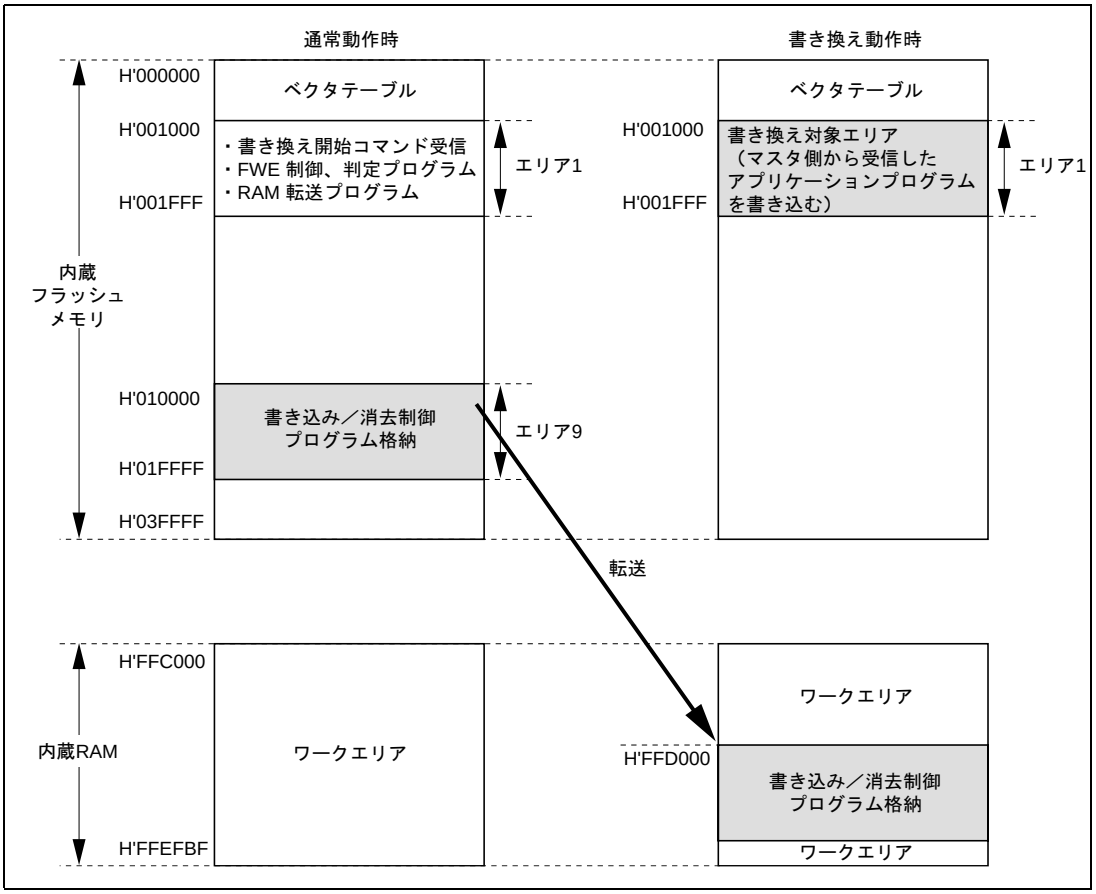


図 6.2 メモリマップ

【注】 図 6.2 の書き込み／消去制御プログラムの格納エリア、実行エリアなどは一例です。

6.3 FWE 出力／停止

書き込み／消去制御プログラムは図6.3に示すタイミングで、FWE端子の出力／停止を制御します。

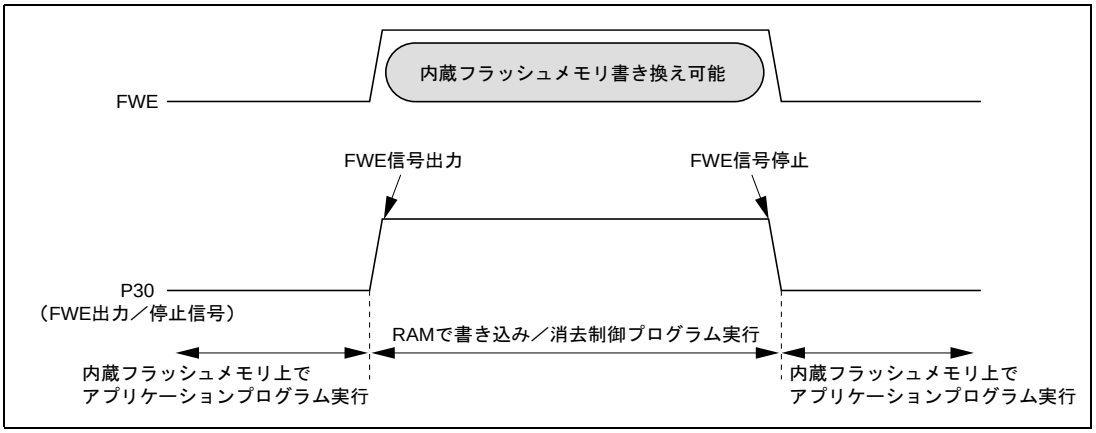
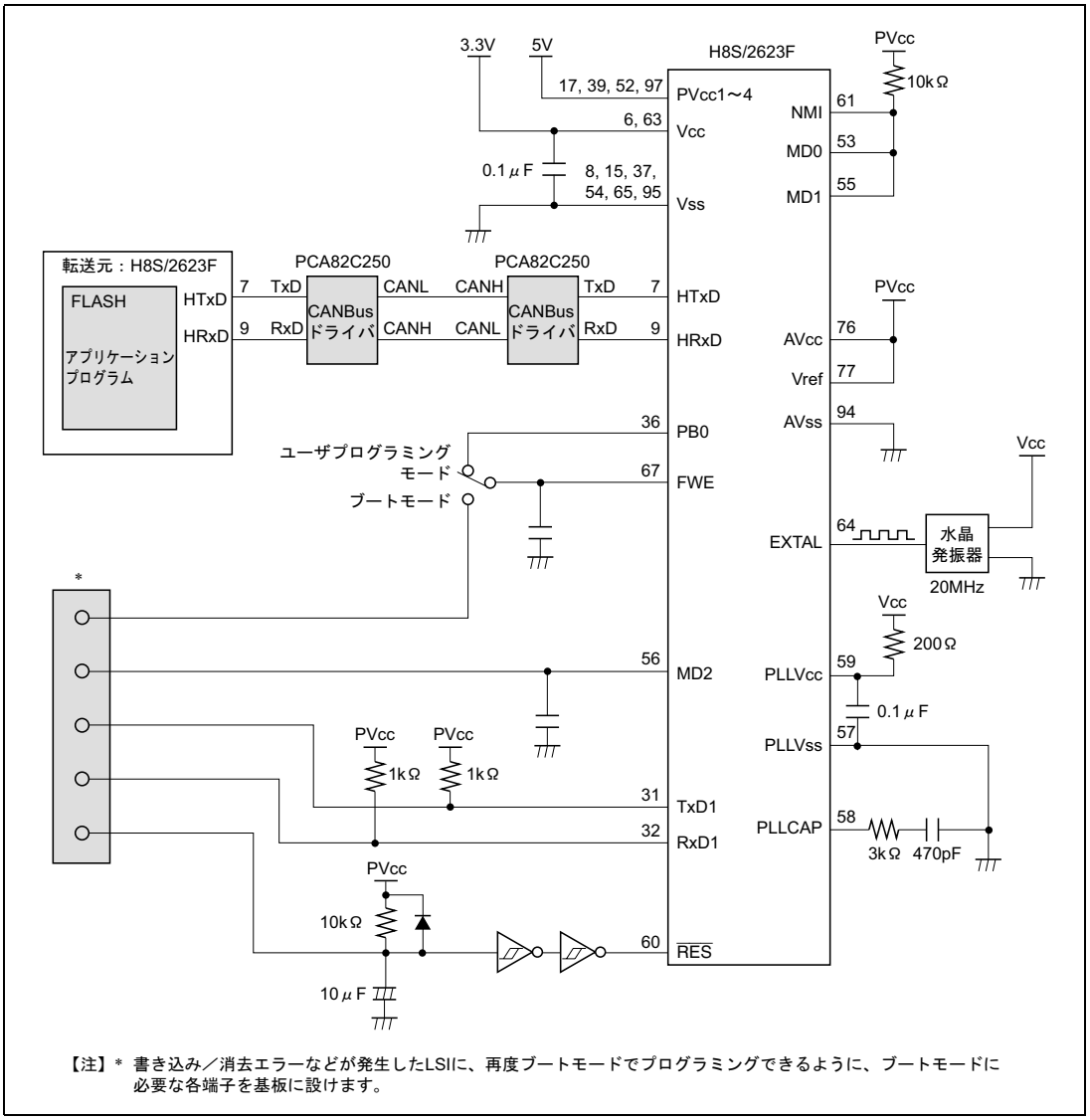


図 6.3 FWE 出力／停止タイミング

7. 回路図



【注】* 書き込み／消去エラーなどが発生したLSIに、再度ブートモードでプログラミングできるように、ブートモードに必要な各端子を基板に設けます。

図 7.1　CAN を使用したオンボード書き換え回路図

8. プログラムリスト

```
;*********************************************************************
;* H8S/2623F                                                         *
;* EXAMPLE PROGRAM FOR "CAN F-ZTAT WRITING CONTROL PROGRAM (VER1.0)"    *
;* MICROSOFT WINDOWS OPERATING SYSTEM                                *
;*********************************************************************
    .CPU           2600A:24
MHZ         .EQU    20*100                      ; 20MHZ
;
;*****  WAIT TIME  ************************************************************
WLOOP1              .EQU      1*MHZ/400+1       ; ROOP WAIT TIME
WLOOP2              .EQU      2*MHZ/400+1
WLOOP4              .EQU      4*MHZ/400+1
WLOOP5              .EQU      5*MHZ/400+1
WLOOP6              .EQU      6*MHZ/400+1
WLOOP10             .EQU     10*MHZ/400+1
WLOOP30             .EQU     30*MHZ/400+1
WLOOP50             .EQU     50*MHZ/400+1
WLOOP100            .EQU    100*MHZ/400+1
TIME10              .EQU     10*MHZ/400         ; WRITE WAIT TIME
TIME30              .EQU     30*MHZ/400         ; WRITE WAIT TIME
TIME200             .EQU    200*MHZ/400         ; WRITE WAIT TIME
TIME10000           .EQU  10000*MHZ/400         ; ERASE WAIT TIME
;*****  H8S/2623F REGISTER  ***************************************************
FLMCR1              .EQU    H'FFFFA8            ; FLASH MEMORY CONTROL REGISTER 1
 SWE:               .EQU    6
 ESU:               .EQU    5
 PSU:               .EQU    4
 EV:                .EQU    3
 PV:                .EQU    2
 E:                 .EQU    1
 P:                 .EQU    0
FLMCR2              .EQU    H'FFFFA9            ; FLASH MEMORY CONTROL REGISTER 2
EBR1                .EQU    H'FFFFAA            ; OBJECT BLOCK DESIGNATED REGISTER 1
EBR2                .EQU    H'FFFFAB            ; OBJECT BLOCK DESIGNATED REGISTER 2
SCRX                .EQU    H'FFFDB4            ; SERIAL CONTOL REGISTER X
FLSHE:              .EQU    3
;*****  MODULE STOP RESISUTOR  ************************************************
MSTPCRA             .EQU    H'FFFDE8            ; MODULE STOP CONTROL REGISTER A
MSTPCRB             .EQU    H'FFFDE9            ; MODULE STOP CONTROL REGISTER B
MSTPCRC             .EQU    H'FFFDEA            ; MODULE STOP CONTROL REGISTER C
;*****  WDT CN.0  *************************************************************
TCSR                .EQU    H'FFFF74            ; WATCH DOG TIMER REGISTOR
RSTCSR              .EQU    H'FFFF76            ; RESET CONTROL REGESTER
```

```
;*****  CAN     ********************************************************************
MCR                 .EQU    H'FFF800            ; MASTER CONTROL REGISTER
GSR                 .EQU    H'FFF801            ; GENERAL STATUS REGISTER
BCR                 .EQU    H'FFF802            ; BIT CONFIGURATION REGISTER
BCR_L               .EQU    H'FFF803            ; BIT CONFIGURATION REGISTER_LOW
MBCR                .EQU    H'FFF804            ; MAILBOX CONFIGURATION REGISTER
MBCR_L              .EQU    H'FFF805            ; MAILBOX CONFIGURATION REGISTER_LOW
TXPR                .EQU    H'FFF806            ; TRANSMIT WAIT REGISTER
TXPR_L              .EQU    H'FFF807            ; TRANSMIT WAIT REGISTER_LOW
TXCR                .EQU    H'FFF808            ; TRANSMIT WAIT CANCEL REGISTER
TXCR_L              .EQU    H'FFF809            ; TRANSMIT WAIT CANCEL REGISTER_LOW
TXACK               .EQU    H'FFF80A            ; TRANSMIT ACKNOWLEDGE REGISTER
TXACK_L             .EQU    H'FFF80B            ; TRANSMIT ACKNOWLEDGE REGISTER_LOW
ABACK               .EQU    H'FFF80C            ; ABORT ACKNOWLEDGE REGISTER
RXPR                .EQU    H'FFF80E            ; RECEIVE COMPLETE REGISTER
RXPR_L              .EQU    H'FFF80F            ; RECEIVE COMPLETE REGISTER_LOW
RFPR                .EQU    H'FFF810            ; REMOTE REQUEST WAIT REGISTER
RFPR_L              .EQU    H'FFF811            ; REMOTE REQUEST WAIT REGISTER_LOW
IRR                 .EQU    H'FFF812            ; INTERRUPT REGISTER
IRR_L               .EQU    H'FFF813            ; INTERRUPT REGISTER_LOW
MBIMR               .EQU    H'FFF814            ; MAILBOX INTERRUPT MASK REGISTER
MBIMR_L             .EQU    H'FFF815            ; MAILBOX INTERRUPT MASK REGISTER_LOW
IMR                 .EQU    H'FFF816            ; INTERRUPT MASK REGISTER
IMR_L               .EQU    H'FFF817            ; INTERRUPT MASK REGISTER
REC                 .EQU    H'FFF818            ; RECEIVE ERROR COUNTER
TEC                 .EQU    H'FFF819            ; TRANSMIT ERROR COUNTER
UMSR                .EQU    H'FFF81A            ; UNREAD MESSAGE STATUS REGISTER
UMSR_L              .EQU    H'FFF81B            ; UNREAD MESSAGE STATUS REGISTER_LOW
LAFML               .EQU    H'FFF81C            ; LOCAL ACCEPTANCE FILTER MASK L
LAFMH               .EQU    H'FFF81E            ; LOCAL ACCEPTANCE FILTER MASK H
;
MC0_1               .EQU    H'FFF820            ; MESSAGE CONTROL0
MC1_1               .EQU    H'FFF828            ; MESSAGE CONTROL1
MC2_1               .EQU    H'FFF830            ; MESSAGE CONTROL2
MC3_1               .EQU    H'FFF838            ; MESSAGE CONTROL3
MC4_1               .EQU    H'FFF840            ; MESSAGE CONTROL4
MC5_1               .EQU    H'FFF848            ; MESSAGE CONTROL5
MC6_1               .EQU    H'FFF850            ; MESSAGE CONTROL6
MC7_1               .EQU    H'FFF858            ; MESSAGE CONTROL7
MC8_1               .EQU    H'FFF860            ; MESSAGE CONTROL8
MC9_1               .EQU    H'FFF868            ; MESSAGE CONTROL9
MC10_1              .EQU    H'FFF870            ; MESSAGE CONTROL10
MC11_1              .EQU    H'FFF878            ; MESSAGE CONTROL11
MC12_1              .EQU    H'FFF880            ; MESSAGE CONTROL12
MC13_1              .EQU    H'FFF888            ; MESSAGE CONTROL13
MC14_1              .EQU    H'FFF890            ; MESSAGE CONTROL14
MC15_1              .EQU    H'FFF898            ; MESSAGE CONTROL15
MC_END              .EQU    H'FFF8A0            ; END OF MESSAGE CONTROL
;
MD0_1               .EQU    H'FFF8B0            ; MESSAGE DATA0
MD1_1               .EQU    H'FFF8B8            ; MESSAGE DATA1
MD2_1               .EQU    H'FFF8C0            ; MESSAGE DATA2
MD3_1               .EQU    H'FFF8C8            ; MESSAGE DATA3
MD4_1               .EQU    H'FFF8D0            ; MESSAGE DATA4
```

```
MD5_1               .EQU    H'FFF8D8            ; MESSAGE DATA5
MD6_1               .EQU    H'FFF8E0            ; MESSAGE DATA6
MD7_1               .EQU    H'FFF8E8            ; MESSAGE DATA7
MD8_1               .EQU    H'FFF8F0            ; MESSAGE DATA8
MD9_1               .EQU    H'FFF8F8            ; MESSAGE DATA9
MD10_1              .EQU    H'FFF900            ; MESSAGE DATA10
MD11_1              .EQU    H'FFF908            ; MESSAGE DATA11
MD12_1              .EQU    H'FFF910            ; MESSAGE DATA12
MD13_1              .EQU    H'FFF918            ; MESSAGE DATA13
MD14_1              .EQU    H'FFF920            ; MESSAGE DATA14
MD15_1              .EQU    H'FFF928            ; MESSAGE DATA15
MD_END              .EQU    H'FFF930            ; END OF MESSAGE DATA
;*****  PORT  ************************************************************
PADDR               .EQU    H'FFFE39            ; PORTA DATA DIRECTION REGISTER
PBDDR               .EQU    H'FFFE3A            ; PORTB DATA DIRECTION REGISTER
PADR                .EQU    H'FFFF09            ; PORTA DATA REGISTER
PBDR                .EQU    H'FFFF0A            ; PORTB DATA REGISTER
PORTA               .EQU    H'FFFFB9            ; PORTA REGISTER
PORTB               .EQU    H'FFFFBA            ; PORTB REGISTER
PAPCR               .EQU    H'FFFE40            ; PORTA MOS PULL-UP CONTROL REGISTER
PBPCR               .EQU    H'FFFE41            ; PORTB MOS PULL-UP CONTROL REGISTER
PAODR               .EQU    H'FFFE47            ; PORTA OPEN-DRAIN CONTROL REGISTER
PBODR               .EQU    H'FFFE48            ; PORTB OPEN-DRAIN CONTROL REGISTER
;*****  RAM ADDRES POINT  ****************************************************
USER_TOP            .EQU    H'FFC000            ; USER RAM TOP
USER_RAM            .EQU    H'FFD000            ; WORK RAM TOP
USER_SP             .EQU    H'FFE000            ; USER SP
;****  フラグ定義  ************************************************************
CHK_ADR             .EQU    H'08000         ; EBR2 対象の先頭アドレス
ST_ADR              .EQU    H'7F            ; 書き込み先頭アドレス ( 128 バイト書き込み )
OK                  .EQU    H'0             ; OK FLAG
NG                  .EQU    H'1             ; ERR FLAG
WNG                 .EQU    H'2             ; WRITE ERR
;****  コマンド定義  ************************************************************
GO_COM              .EQU    H'55                ; 書き換え開始コマンド
OK_COM              .EQU    H'00                ; OK コマンド
FWE_ON_COM          .EQU    H'66                ; FWE 端子設定コマンド
ERASE_COM           .EQU    H'77                ; 消去コマンド
WRITE_COM           .EQU    H'88                ; 書き込みコマンド
REQUEST_COM         .EQU    H'11                ; 送信要求コマンド
FWE_OFF_COM         .EQU    H'99                ; FWE 端子解除コマンド
NG_COM              .EQU    H'01                ; NG コマンド
```

```
;*****  FLASH 定数  ************************************************************
MAXWT               .EQU    1000                ; MAX WRITE COUNT
MAXET               .EQU    100                 ; MAX ERASE COUNT
OW_COUNT            .EQU    6                   ; OVER WRITE COUNT
MAXBLK1             .EQU    8                   ; MAX BLOCK 1
MAXBLK2             .EQU    4                   ; MAX BLOCK 2
;
;//H8S/2623F 書き込み/消去プログラム RAM エリア//
    .SECTION   RAM,DATA,LOCATE = USER_TOP
    .ALIGN  2
W_ADR               .RES.B  4                   ; WRITE ADDRESS AREA
LAST_JMP            .RES.B  4                   ; LAST JMP ADDRESS
W_BUF               .RES.B  128                 ; WRITE DATA AREA
BUFF                .RES.B  128                 ; RETRY WRITE DATA AREA
OWBUFF              .RES.B  128                 ; OVER WRITE DATA ARIA
COUNT               .RES.B  2                   ; W_COUNT,E_COUNT
ET_COUNT            .RES.B  2                   ; MAX E_COUNT
WT_COUNT            .RES.B  2                   ; MAX W_COUNT
EVF_ST              .RES.B  4                   ; ERASE VERIFY START ADDRESS
EVF_ED              .RES.B  4                   ; ERASE VERIFY END ADDRESS
BLK_NO              .RES.B  1                   ; ERASE BIT NUMBER
VF_RET              .RES.B  1                   ; VERIFY CHECK
    .ALIGN  2
RESTSIZE            .RES.B  4                   ; WRITE DATA SIZE
E_ADR               .RES.B  48                  ; BLOCK ERASE ADDRESS
E_ADR_PTR           .RES.B  4                   ; BLOCK ERASE ADDRESS POINTER
ERASEBLOCK          .RES.B  1                   ; ERASE BLOCK NUMBER
;
; ***************************************************************************
; * 名称 / ベクタテーブル                                                    *
; * 機能 / -                                                                *
; * 入力 / -                                                                *
; * 出力 / -                                                                *
; ***************************************************************************
    .SECTION   VECT,DATA,LOCATE=H'000000
    .DATA.L         MAIN
    .ORG            H'0001A0
    .DATA.L         MAIN
    .DATA.L         IRR0
    .DATA.L         MAIN
    .DATA.L         MAIN
    .DATA.L         MAIN
```

```
; ********************************************************************************
; * 名称 / メインルーチン                                                        *
; * 機能 / メイン側の GO コマンド受信後、RAM 転送プログラム実行                  *
; * 入力 / -                                                                     *
; * 出力 / -                                                                     *
; ********************************************************************************
    .SECTION   PROG1,CODE,LOCATE = H'001000
MAIN        .EQU    $
            MOV.L   #USER_SP,ER7            ; INITIALIZED STACK
            MOV.B   #H'FF,R0L               ;
            MOV.B   R0L,@PADR               ; ポート A HIGH レベル
            MOV.B   R0L,@PADDR              ; ポート A 出力端子
            LDC.B   #H'00,CCR               ; I ビットクリアにより割り込み許可
            MOV.B   #H'F7,R0L               ;
            MOV.B   R0L,@MSTPCRC            ; CAN モジュールストップ解除
            BCLR    #0,@PADR                ; ポート A0"LOW"出力
            JSR     @RX_COM                 ; マスタ側からコマンド受信
            CMP.B   #H'55,R2L               ; マスタ側から書き換え開始コマンド受信?
            BEQ     TRANS_RAM               ;
            BCLR    #5,@PADR                ; ポート A5"LOW"出力
            MOV.B   #NG_COM,R2L             ;
            JSR     @TX_COM                 ; マスタ側へ NG コマンド送信
ERROR_1     NOP                             ;
            BRA     ERROR_1                 ; 無限ループ
;
TRANS_RAM:
    MOV.L   #FLASH_MAIN,ER5                 ; 転送プログラムの先頭アドレス
    MOV.L   #USER_RAM,ER6                   ; 転送する RAM の先頭アドレス
    MOV.W   #ENDFILE-FLASH_MAIN,R4          ; 転送プログラムの最終アドレス
    EEPMOV.W                                ; 制御プログラムを RAM へ転送
    JMP     @USER_RAM                       ; RAM 上の制御プログラムへ JMP
;
```

```
; ********************************************************************************
; * 名称 / CAN IRR0割り込み                                                      *
; * 機能 / CAN 初期設定                                                          *
; ********************************************************************************
IRR0        .EQU    $
            BSET    #0,@IRR             ; IRR0 を"1"ライトし"0"クリア
            MOV.W   #H'0025,R5          ;
            MOV.W   R5,@BCR             ; 1Mbps (at 20MHz)にボーレート設定
            MOV.W   #H'FF01,R5          ;
            MOV.W   R5,@MBCR            ; メールボックス送信、受信設定
;
            MOV.L   #MC0_1,ER5          ; MC0～15 初期化(ALL H'00 ライト)
            SUB.L   ER6,ER6             ;
CLEAR_1     MOV.L   ER6,@ER5            ;
            ADDS    #4,ER5              ; MC アドレスインクリメント
            CMP.L   #MC_END,ER5         ; MC15_8 まで終了?
            BNE     CLEAR_1             ;
;
            MOV.L   #MD0_1,ER5          ; MD0～15 初期化(ALL H'00 ライト)
CLEAR_2     MOV.L   ER6,@ER5            ;
            ADDS    #4,ER5              ; MD アドレスインクリメント
            CMP.L   #MD_END,ER5         ; MD15_8 まで終了?
            BNE     CLEAR_2             ;
;
            MOV.B   #H'04,R6L           ;
            MOV.B   R6L,@MCR            ; 送信方式:MB 順,CAN 通常モードへ遷移
LOOP_1      BTST    #3,@GSR             ; CAN 通常モード? GSR3=0?
            BNE     LOOP_1              ;
;
            SUB.L   ER6,ER6             ;
            MOV.L   ER6,@LAFML          ; MB0 の ID は全一致
            MOV.W   #H'FFFF,R6          ;
            MOV.W   R6,@MBIMR           ; CPU への割り込み要求禁止
            MOV.W   #H'FEFF,R6          ;
            MOV.W   R6,@IMR             ; CPU への割り込み要求禁止
;
            MOV.L   #MC0_1,ER5
            MOV.B   #H'08,R6L           ; MB0 のデータ長 8 バイト
            MOV.W   #H'0000,E6          ; MD0 の ID"H'00"
            BSR     MC_SET
            MOV.L   #MC1_1,ER5
            MOV.B   #H'08,R6L           ; MB1 のデータ長 8 バイト
            MOV.W   #H'2000,E6          ; MB1 の ID"H'01"
            BSR     MC_SET
            MOV.L   #MC2_1,ER5
            MOV.B   #H'08,R6L           ; MB2 のデータ長 8 バイト
            MOV.W   #H'4000,E6          ; MB2 の ID"H'02"
            BSR     MC_SET
            MOV.L   #MC3_1,ER5
            MOV.B   #H'08,R6L           ; MB3 のデータ長 8 バイト
            MOV.W   #H'6000,E6          ; MB3 の ID"H'03"
            BSR     MC_SET
            MOV.L   #MC4_1,ER5
            MOV.B   #H'08,R6L           ; MB4 のデータ長 8 バイト
```

```
            MOV.W   #H'8000,E6              ; MB4 の ID"H'04"
            BSR     MC_SET
            MOV.L   #MC5_1,ER5
            MOV.B   #H'08,R6L               ; MB5 のデータ長 8 バイト
            MOV.W   #H'A000,E6              ; MB5 の ID"H'05"
            BSR     MC_SET
            MOV.L   #MC6_1,ER5
            MOV.B   #H'08,R6L               ; MB6 のデータ長 8 バイト
            MOV.W   #H'C000,E6              ; MB6 の ID"H'06"
            BSR     MC_SET
            MOV.L   #MC7_1,ER5
            MOV.B   #H'08,R6L               ; MB7 のデータ長 8 バイト
            MOV.W   #H'E000,E6              ; MB7 の ID"H'07"
            BSR     MC_SET
            MOV.L   #MC8_1,ER5
            MOV.B   #H'08,R6L               ; MB8 のデータ長 8 バイト
            MOV.W   #H'0001,E6              ; MB8 の ID"H'08"
            BSR     MC_SET
            MOV.L   #MC9_1,ER5
            MOV.B   #H'01,R6L               ; MB9 のデータ長 1 バイト
            MOV.W   #H'2001,E6              ; MB9 の ID"H'09"
            BSR     MC_SET
            RTE
;
; ********************************************************************************
; * 名称 / メッセージコントロールレジスタ設定                                      *
; * 機能 /使用する メールボックス 0～9 の ID,データサイズを設定                    *
; * 入力 / ER5     = メッセージコントロールレジスタ n の先頭アドレス               *
; *    R6L     = データ長設定値                                                    *
; *    E6      = ID 設定値                                                         *
; * 出力 / -                                                                      *
; ********************************************************************************
MC_SET      .EQU    $
            MOV.B   R6L,@ER5                ; MBn のデータ長設定
            ADDS    #4,ER5                  ; MDn_5
            MOV.W   E6,@ER5                 ; MBn の ID 設定
            RTS
;
```

```
; ********************************************************************************
; * 名称 / FLASH 書き込みメインルーチン                                          *
; * 機能 / マスタ側からのコマンドチェック,消去回数/書き込み回数の設定            *
; * 入力 / -                                                                    *
; * 出力 / -                                                                    *
; ********************************************************************************
    .SECTION   PROG2,CODE,LOCATE = H'10000
;
FLASH_MAIN  .EQU    $
            MOV.B   #OK_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ OK コマンド送信
            BSR     RX_COM                  ; マスタ側からコマンド受信
            CMP.B   #FWE_ON_COM,R2L         ; マスタ側から FWE 端子設定コマンド受信?
            BEQ     SET_FWE_PIN             ;
            BRA     ERROR_2                 ;
;
SET_FWE_PIN:
            MOV.B   #H'01,R0L               ;
            MOV.B   R0L,@PBDDR              ; ポート B0 出力
            BSET    #0,@PBDR                ; ポート B0"HIGH"出力にし FWE 端子"ON"
            BSET.B  #FLSHE,@SCRX            ; FLASH ENABLE
            BTST    #7,@FLMCR1              ; FWE=1?
            BNE     FWE_OK                  ;
ERROR_2:
            BCLR    #4,@PADR                ; ポート A4"LOW"出力
            MOV.B   #NG_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ NG コマンド送信
ERROR_3     NOP                             ;
            BRA     ERROR_3                 ; 無限ループ
;
FWE_OK:     MOV.B   #OK_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ OK コマンド送信
            MOV.W   #MAXET,R0               ; 消去最大回数設定
            MOV.W   R0,@ET_COUNT
            MOV.W   #MAXWT,R0               ; 書き込み最大回数設定
            MOV.W   R0,@WT_COUNT
            BSR     WCMD                    ;
;
CLEAR_FWE_PIN:
            BCLR    #0,@PBDR                ; ポート B0"LOW"出力にし FWE 端子"OFF"
            MOV.B   #OK_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ OK コマンド送信
            SUB.L   ER0,ER0                 ;
            JMP     @ER0                    ; リセットに JMP
;
```

```
; ********************************************************************************
; * 名称 / 書き込み、ブロック消去メインルーチン                                  *
; * 機能 / １２８バイトの書き込み、ブロック消去サポート                          *
; * 入力 / -                                                                    *
; * 出力 / -                                                                    *
; ********************************************************************************
WCMD        .EQU    $
;------- 消去ブロック数、消去先頭アドレス受信 --------
            BSR     GET_EADR                ;
            CMP.B   #OK,R0L                 ;
            BNE     WCMD_ERR                ; エラーケース
            MOV.L   #E_ADR,ER1              ;
            MOV.L   ER1,@E_ADR_PTR          ;
;
NEXTBLOCK   MOV.L   @E_ADR_PTR,ER1          ;
            MOV.L   @ER1+,ER3               ;
            MOV.L   ER1,@E_ADR_PTR          ;
;------- 先頭ブロックアドレスチェック --------
            BSR     BLK_CHECK               ; 消去先頭アドレスをチェック
            CMP.B   #OK,R0L                 ;
            BNE     WCMD_ERR                ; 消去先頭アドレスエラー
;------- F-ZTAT 消去処理 --------
            BSR     BLK1_ERASE              ; 消去処理
            CMP.B   #OK,R0L                 ;
            BNE     WCMD_ERR                ; 消去エラー
;
            MOV.B   @ERASEBLOCK,R3L         ;
            DEC.B   R3L                     ; 消去ブロック数をデクリメント
            MOV.B   R3L,@ERASEBLOCK         ;
            BNE     NEXTBLOCK               ; 次のブロックの消去
;
            MOV.B   #OK_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ OK コマンド送信
;
;------- 書き込み先頭アドレス、書き込みデータサイズ受信 --------
            BSR     GET_WADR                ; 書き込み先頭アドレス、サイズ受信
            CMP.B   #OK,R0L                 ;
            BNE     WCMD_ERR                ; エラーケース
;
WCMD_W10    MOV.B   #REQUEST_COM,R2L        ; マスタ側に 128 バイトデータのリクエスト
            BSR     TX_COM                  ;
;
;------- １２８バイトデータを受信し、書き込みエリアに転送  --------
            BSR     GET_BUFFER              ;
;
;------- １２８バイト単位の書き込み処理 --------
            BSR     FWRITE128               ; フラッシュ書き込み処理(128 バイト単位)
            CMP.B   #OK,R0L                 ;
            BNE     WCMD_ERR                ; 書き込みエラー
;
            MOV.L   @W_ADR,ER2              ; 書き込みエリアのアドレス
            ADD.L   #128,ER2                ; アドレスを 128 加算
            MOV.L   ER2,@W_ADR              ;
;
```

```
            MOV.L   @RESTSIZE,ER0           ; 送信プログラムバイト数
            CMP.L   #0,ER0                  ; すべてのプログラムを受信?
            BNE     WCMD_W10                ; 次の 128 バイトデータを受信し書き込む
;
            MOV.B   #OK_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ OK コマンド送信
            BSR     RX_COM                  ; マスタ側からコマンド受信
            CMP.B   #FWE_OFF_COM,R2L        ; マスタ側から FWE 端子解除コマンド受信?
            BNE     WCMD_ERR                ;
            RTS
;
;------- ERR END ------------------------------
WCMD_ERR:
            MOV.B   #H'04,R0L               ;
            MOV.B   R0L,@PBDDR              ; ポート B2 出力
            BCLR    #2,@PBDR                ; ポート B2"LOW"出力
            MOV.B   #NG_COM,R2L             ;
            BSR     TX_COM                  ; マスタ側へ NG コマンド送信
ERROR_5     NOP                             ;
            BRA     ERROR_5                 ; 無限ループ
            RTS
;
; ********************************************************************************
; * 名称 / 消去アドレス受信ルーチン                                              *
; * 機能 / 書き込み先頭アドレス、サイズを受信しエラーの場合はエラーコードを返す  *
; * 入力 / R2L   = 受信アドレス数                                                *
; * 出力 / E_ADR = 各消去先頭アドレス                                            *
; ********************************************************************************
GET_EADR    .EQU    $
            BTST    #0,@RXPR                ; RXPR0=1?(MB0 受信完了?)
            BEQ     GET_EADR                ;
            MOV.W   @MD0_1,R2               ;
            MOV.B   R2L,@ERASEBLOCK         ; 受信した消去ブロック数を格納
            BSET    #0,@RXPR                ; PXPR0 に"1"ライトし、"0"クリア
            CMP.B   #ERASE_COM,R2H          ; マスタ側から消去コマンド受信?
            BNE     EADR_ERR                ;
            MOV.L   #E_ADR,ER3              ;
GET_LONG_EADR       BSR     RX_4            ;
            DEC.B   R2L                     ; 消去ブロック数インクリメント
            BNE     GET_LONG_EADR           ;
            MOV.B   #OK,R0L                 ; OK フラグセット
            RTS
;------- ERR ----------------------------------
EADR_ERR    MOV.B   #NG,R0L                 ; NG フラグセット
            RTS
;
```

```
; ********************************************************************************
; * 名称 / 消去ブロックチェックルーチン                                          *
; * 機能 / 消去先頭アドレスよりブロックビットを判定し消去関連レジスタの設定      *
; * 入力 / ER3 = 消去先頭アドレス                                                *
; * 出力 / R0L = 結果フラグ(OK=H'00,NG=H'01)                                     *
; ********************************************************************************
;
BLK_CHECK   .EQU    $
            MOV.B   #0,R4H                  ; ブロック数カウンタ初期化
            MOV.L   #FLMCR1,ER6             ; FLMCR アドレスセット
            MOV.L   #EBR1,ER5               ; EBR1 アドレスセット
            MOV.L   #((BLOCKADR1-FLASH_MAIN)+USER_RAM),ER0 ; EBR1 テーブルセット
            MOV.B   #MAXBLK1,R4L            ; 8 ブロックセット(EBR1)
            CMP.L   #CHK_ADR,ER3            ; H'8000 番地
            BCS     BLK_CHK10               ; ブロック先頭アドレス < H'8000 番地?
;
            MOV.L   #EBR2,ER5               ; EBR2 アドレスセット
            MOV.L   #((BLOCKADR2-FLASH_MAIN)+USER_RAM),ER0 ; EBR2 テーブルセット
            MOV.B   #MAXBLK2,R4L            ; 4 ブロックセット(EBR2)
BLK_CHK10   MOV.L   @ER0+,ER1               ;
            CMP.L   ER1,ER3                 ; テーブルのアドレスと一致?
            BEQ     BLK_CHK20               ;
;
            ADD.B   #1,R4H                  ; ブロック数カウンタインクリメント
            CMP.B   R4L,R4H                 ; ブロック数 MAX?
            BEQ     BLK_ERR                 ; エラー
            BRA     BLK_CHK10               ;
;
BLK_CHK20
    MOV.B   R4H,@BLK_NO                     ; 消去対象ブロック
    MOV.L   ER1,@EVF_ST                     ; 消去先頭アドレスセット
            MOV.L   @ER0,ER2                ; ERASE END ADDRESS
    MOV.L   ER2,@EVF_ED                     ; 消去最終アドレスセット
            MOV.B   #OK,R0L                 ; 正常終了
            RTS
;
;-------- BLK ERR ------------------------------
BLK_ERR     MOV.B   #NG,R0L                 ; 消去ブロックアドレスエラー
            RTS
;
; ********************************************************************************
; * 名称 / 書き込み先頭アドレス、サイズ受信ルーチン                              *
; * 機能 / 書き込み先頭アドレス、サイズを受信しエラーの場合はエラーコードを返す  *
; * 入力 / -                                                                     *
; * 出力 /  W_ADR     = 書き込み先頭アドレス                                     *
; *         RESTSIZE  = 書き込みサイズ                                           *
; *         R0L       = 結果フラグ(OK=H'00,NG=H'01)                              *
; ********************************************************************************
;
GET_WADR    .EQU    $
            BSR     RX_COM                  ; マスタ側からコマンド受信
            CMP.B   #WRITE_COM,R2L          ; マスタ側から書き込みコマンド受信?
            BNE     ADR_ERR                 ;
            MOV.L   #W_ADR,ER3              ; 書き込み先頭アドレス受信
            BSR     RX_4                    ;
            SUBS    #4,ER3                  ; W_ADR アドレス
```

```
            MOV.L   @ER3,ER6                ;
            MOV.L   ER6,@LAST_JMP           ; 最後の JMP 先(W_ADR 先頭番地)
            AND.B   #ST_ADR,R1L             ; W_ADR 下位 8BIT が H'80 or H'00?
            BNE     ADR_ERR         ; @W_ADR = H'XXXXXX80,XXXXXX00 以外はエラーへ
;
            MOV.L   #RESTSIZE,ER3           ; 書き込みプログラムサイズ受信
            BSR     RX_4                    ;
            CMP.L   #0,ER1                  ; 書き込みプログラムサイズ=0?
            BEQ     ADR_ERR                 ; @RESTSIZE = H'0000 の場合はエラーへ
;
            MOV.B   #OK,R0L                 ; OK フラグセット
            RTS
;
;------- ERR ----------------------------------
ADR_ERR     MOV.B   #NG,R0L         ; NG フラグセット
            RTS
;
; ********************************************************************************
; * 名称 / 書き込みデータ受信                                                    *
; * 機能 / ホストより書き込みデータを受信する                                    *
; * 入力 / RESTSIZE = 書き込みサイズ                                             *
; * 出力 / W_BUF    = 書き込みデータ１２８バイト                                 *
; ********************************************************************************
GET_BUFFER .EQU     $
            MOV.L   @RESTSIZE,ER6           ; 書き込みプログラムサイズ
            CMP.L   #128,ER6                ; 書き込みプログラムサイズ < 128 バイト
            BCC     BUFFER10                ; ER6 <= 128
;
            MOV.L   #W_BUF,ER3              ; 書き込みデータエリア
BUFFER00    MOV.B   #H'FF,R5L               ;
            MOV.B   R5L,@ER3                ; 先ず、H'FF ライト
            INC.L   #1,ER3                  ;
            CMP.L   #W_BUF+128,ER3          ; 書き込みデータエリアすべて H'FF ライト完了？
            BNE     BUFFER00                ;
;
            MOV.L   ER6,ER4                 ; ER4 = 残りのデータサイズ
            SUB.L   ER4,ER6                 ;
            MOV.L   ER6,@RESTSIZE           ; 書き込みプログラムサイズ = 0
            MOV.L   #W_BUF,ER3              ; 書き込みデータエリアの先頭アドレスセット
BUFFER01    BSR     RX_COM                  ; マスタ側 MB9 から MB0 へ 1 バイトずつ受信
            MOV.B   R2L,@ER3                ; 受信データを書き込みデータエリアに格納
            INC.L   #1,ER3                  ; W_BUF アドレスインクリメント
            DEC.B   R4L                     ; 残りのデータ受信完了?
            BNE     BUFFER01                ;
            RTS                             ;
;
BUFFER10    MOV.L   #128,ER4                ;
            SUB.L   ER4,ER6                 ; 書き込みプログラムサイズ-128
            MOV.L   ER6,@RESTSIZE           ;
            MOV.L   #W_BUF,ER3              ; 書き込みデータエリアの先頭アドレスセット
            BSR     RX_128                  ;
            RTS
;
```

```
; ********************************************************************************
; * 名称 / 1バイト受信ルーチン                                                   *
; * 機能 / データを1バイト受信する、受信エラー時の場合は無限ループ               *
; * 入力 / -                                                                     *
; * 出力 / R2L = 受信データ                                                      *
; ********************************************************************************
;
RX_COM      .EQU    $
            BTST    #0,@RXPR                ; RXPR0=1?(MB0 受信完了?)
            BEQ     RX_COM
            MOV.B   @MD0_1,R2L
            BSET    #0,@RXPR                ; PXPR0 に"1"ライトし、"0"クリア
            RTS                             ;
;
; ********************************************************************************
; * 名称 / 4バイト受信ルーチン                                                   *
; * 機能 / データを4バイト受信                                                   *
; * 入力 / -                                                                     *
; * 出力 / ER2                                                                   *
; ********************************************************************************
;
RX_4        .EQU    $
            BTST    #0,@RXPR                ; RXPR0=1?(MB0 受信完了?)
            BEQ     RX_4
            MOV.L   @MD0_1,ER1
            MOV.L   ER1,@ER3                ; 受信した先頭アドレスを@ER3 に格納
            ADDS    #4,ER3                  ; ER3 アドレスインクリメント
            BSET    #0,@RXPR                ; PXPR0 に"1"ライトし、"0"クリア
            RTS
;
; ********************************************************************************
; * 名称 / 1バイト送信ルーチン                                                   *
; * 機能 / 引数より得たデータを1バイト送信する                                   *
; * 入力 / R2L = 送信データ                                                      *
; * 出力 / -                                                                     *
; ********************************************************************************
;
TX_COM      .EQU    $
            MOV.B   R2L,@MD9_1              ;
            BSET    #1,@TXPR_L              ; TXPR9 に"1"セット
TX_LOOP1    BTST    #1,@TXACK_L             ; TXACK9=1?
            BEQ     TX_LOOP1                ;
            BSET    #1,@TXACK_L             ; TXACK9 に"1"ライトし、"0"クリア
            RTS
;
```

```
; ********************************************************************************
; * 名称 / １２８バイト受信ルーチン                                              *
; * 機能 / マスタ側 MB１～８の１２８バイトデータを受信(６４バイト２回受信)       *
; * 入力 / ER3(#W_BUF)                                                           *
; * 出力 / @ER3(W_BUF)                                                           *
; ********************************************************************************
RX_128          .EQU $
                MOV.B       #2,R0H          ;カウンタ=2
RX_128_LOOP1    MOV.B       @RXPR,R0L       ;
                OR.B        #H'01,R0L       ; RXPR1～7=ALL 1?
                CMP.B       #H'FF,R0L       ;
                BNE         RX_128_LOOP1    ;
RX_128_LOOP2    BTST        #0,@RXPR_L      ;
                BEQ         RX_128_LOOP2    ;
;
                MOV.W       #64,R4          ;
                MOV.L       #MD1_1,ER5      ;転送元：MD1_1
                MOV.L       ER3,ER6         ;転送先：W_BUF
                EEPMOV.W                    ; MD1_1～MD8_8 -> W_BUF ブロック転送
;
                MOV.W       @RXPR,E0        ;
                OR.W        #H'FE01,E0      ;
                MOV.W       E0,@RXPR        ;RXPR1～8 に"1"ライトし、"0"クリア
                MOV.L       ER6,ER3         ;W_BUF+128 を ER3 に格納
                DEC.B       R0H             ;カウンタデクリメント
                BNE         RX_128_LOOP1    ;128 バイト(2 回)受信?
                RTS
;
; ********************************************************************************
; * 名称 / フラッシュ任意の１２８バイト書き込みルーチン                          *
; * 機能 / １２８バイトの書き込み、ベリファイ                                    *
; * 入力 / @W_ADR    = 書き込みアドレス                                          *
; *        @W_BUF    = 書き込みデータ１２８バイト                                *
; *        @WT_COUNT = 書き込み最大回数                                          *
; * 出力 / R0L       = 結果フラグ(OK=H'00,NG=H'01)                               *
; ********************************************************************************
FWRITE128     .EQU  $
              MOV.W     #H'5A5F,R0          ; WDT 初期設定
              MOV.W     R0,@RSTCSR
;
              MOV.W     #128,R4
              MOV.L     #W_BUF,ER5
              MOV.L     #BUFF,ER6
              EEPMOV.W                      ; W_BUF -> BUFF ブロック転送
;
              MOV.L     #FLMCR1,ER6         ; フラッシュ制御レジスタポインタ
;
              MOV.W     #WLOOP1,R0
              BSET.B    #SWE,@ER6           ; SWE ビットセット
FWRTE10                 DEC.W     #1,R0     ; SWE 設定後ウェイト(1μS 以上)
              BNE       FWRTE10:16
;
              XOR.W     R0,R0               ; 書き込み回数カウンタクリア
              MOV.W     R0,@COUNT
```

```
;
;======= 初期ベリファイ ========================
            BSR       FWRITEVF          ; 初期プログラムベリファイ
            CMP.B     #OK,R0L
            BEQ       FWRTE40           ; 初期ベリファイ完了
;
            CMP.B     #WNG,R0L
            BEQ       FWRTE30           ; 書き込み済みエラー
;
;======= 初期書き込み（追加書き込みあり） ======
FWRTE15     MOV.L     #BUFF,ER2         ; 再書き込みデータ
            MOV.L     #TIME30,ER3       ; P パルス印可（30μS）
            BSR       FWRITE            ; プログラム
            BSR       FWRITEVF          ; プログラムベリファイ
            MOV.B     R0L,@VF_RET
            MOV.L     #OWBUFF,ER2       ; 追加書き込みデータ
            MOV.L     #TIME10,ER3       ; P パルス印可（10μS）
            BSR       FWRITE            ; 追加プログラム
            MOV.B     @VF_RET,R0L
            CMP.B     #OK,R0L
            BEQ       FWRTE40           ; プログラム完了
;
            CMP.B     #WNG,R0L
            BEQ       FWRTE30           ; 書き込み済みエラー
;
            MOV.W     @COUNT,R0         ; 書き込み回数カウンタ @COUNT＋1
            INC.W     #1,R0
            MOV.W     R0,@COUNT
            CMP.W     #OW_COUNT,R0
            BNE       FWRTE15           ; 回数判定（追加書き込み回数）
;
;======= 通常書き込み（追加書き込みなし）=======
FWRTE20     MOV.L     #BUFF,ER2         ; 再書き込みデータ
            MOV.L     #TIME200,ER3      ; P パルス印可（200μS）
            BSR       FWRITE            ; プログラム
            BSR       FWRITEVF          ; プログラムベリファイ
            CMP.B     #OK,R0L
            BEQ       FWRTE40           ; プログラム完了
;
            CMP.B     #WNG,R0L
            BEQ       FWRTE30           ; 書き込み済みエラー
;
            MOV.W     @COUNT,R0         ; 書き込み回数カウンタ @COUNT＋1
            INC.W     #1,R0
            MOV.W     R0,@COUNT
            MOV.W     @WT_COUNT,E0
            CMP.W     E0,R0
            BNE       FWRTE20           ; 回数判定（最大書き込み回数）
;
```

```
;------- 異常終了 ------------------------------
FWRTE30     MOV.W     #WLOOP100,R0
            BCLR.B    #SWE,@ER6          ; SWE ビットクリア
FWRTE35     DEC.W     #1,R0              ; SWE 解除後ウェイト(100μS 以上)
            BNE       FWRTE35:16
;
            MOV.B     #NG,R0L            ; NG セット
            RTS
;
;------- 正常終了 ------------------------------
FWRTE40     MOV.W     #WLOOP100,R0
            BCLR.B    #SWE,@ER6          ; SWE ビットクリア
FWRTE45     DEC.W     #1,R0              ; SWE 解除後ウェイト(100μS 以上)
            BNE       FWRTE45:16
;
            MOV.B     #OK,R0L            ; OK セット
            RTS
;
; ********************************************************************************
; * 名称 / 書き込みベリファイ                                                    *
; * 機能 / 対象アドレスのベリファイ、再書き込みデータの作成                      *
; * 入力 /  ER6      = FLMCR レジスタのアドレス                                  *
; *         @W_ADR   = 書き込みアドレス                                          *
; *         @W_BUF   = 書き込みデータ１２８バイト                                *
; *         @BUFF    = 再書き込みデータ１２８バイト                              *
; * 出力 /  @BUFF    = 再書き込みデータ１２８バイト                              *
; *         @OWBUFF  = 追加書き込みデータ１２８バイト                            *
; *         R0L      = ベリファイ結果(OK=H'00,NG=H'01,WNG=H'02)                  *
; ********************************************************************************
FWRITEVF    .EQU    $
            MOV.W   #H'FFFF,R5         ; アドレスラッチ用ダミーライトデータ
            MOV.L   #BUFF,ER1          ; 再書き込みデータバッファ
            MOV.L   #W_BUF,ER2         ; 書き込みデータバッファ
            MOV.L   @W_ADR,ER4         ; フラッシュ ROM 書き込みアドレス
;======== 追加書き込みデータバッファ ===========
            MOV.L   #OWBUFF,ER3        ; 追加書き込みデータバッファ
;============================================
;
            MOV.W   #WLOOP4,E0
            BSET.B  #PV,@ER6           ; PV ビットセット
FWVF10      DEC.W   #1,E0              ; PV 設定後ウェイト(4μS 以上)
            BNE     FWVF10:16
;
FWVF20      MOV.W   #WLOOP2,E0
            MOV.W   R5,@ER4            ; ダミーライト(ラッチ)
FWVF30      DEC.W   #1,E0              ; ラッチウェイト(2μS 以上)
            BNE     FWVF30:16
;
            MOV.W   @ER4+,R0           ; フラッシュＲＯＭデータ
```

```
;======== 追加書き込みデータの作成 =============
            MOV.W   @ER1,E0                 ; 初期書き込み（６回のみ使用）
            OR.W    R0,E0                   ; @COUNT = 0,1,2,3,4,5 有効データ
            MOV.W   E0,@ER3                 ; @COUNT = 6～999      無効データ
            ADDS    #2,ER3
;================================================
            NOT.W   R0
            MOV.W   @ER2,E0
            OR.W    R0,E0                   ; 反転データ OR 書き込みデータ
            MOV.W   E0,@ER1                 ; 再書き込みデータ設定
            ADDS    #2,ER1
            MOV.W   @ER2+,E0
            AND.W   E0,R0                   ; 読み出しデータ０ AND 書き込みデータ１のとき
            BNE     FWVF70                  ; 書けないエラーへ
;
            CMP.L   #W_BUF+128,ER2
            BNE     FWVF20                  ; 128 バイトベリファイ
;
            MOV.W   #WLOOP2,E0
            BCLR.B  #PV,@ER6                ; PV ビットクリア
FWVF40      DEC.W   #1,E0                   ; PV 解除後ウェイト(2μS 以上)
            BNE     FWVF40:16
;
            MOV.B   #NG,R0L                 ; NG セット
            MOV.L   #BUFF,ER1               ; 再書き込みデータ先頭アドレス
FWVF50      MOV.W   @ER1+,E0
            CMP.W   R5,E0                   ; 再書き込みデータ 128 バイトすべて FF か?
            BNE     FWVF60                  ; 判定 H'FF でないならベリファイエラーへ
;
            CMP.L   #BUFF+128,ER1
            BNE     FWVF50                  ; 128 バイト確認
;
            MOV.B   #OK,R0L                 ; OK セット
FWVF60      RTS
;
;-------  FLASH ROM ERR  -----------------------
FWVF70      MOV.W   #WLOOP2,E0
            BCLR.B  #PV,@ER6                ; PV ビットクリア
FWVF80      DEC.W   #1,E0                   ; PV 解除後ウェイト(2μS 以上)
            BNE     FWVF80:16
;
            MOV.B   #WNG,R0L                ; WNG セット
            RTS
;
```

```
; ********************************************************************************
; * 名称 / 書き込みルーチン                                                      *
; * 機能 / 対象アドレスの書き込み                                                *
; * 入力 /   ER6     = FLMCR レジスタのアドレス                                  *
; *          @W_ADR  = 書き込みアドレス                                          *
; *          ER2     = 書き込みの先頭アドレス( 再書き込みデータ or 追加書き込みデータ )  *
; *          ER3     = Ｐビットセット時間( 10μS or 30μS or 2000μS )             *
; * 出力 / -                                                                     *
; ********************************************************************************
FWRITE      .EQU    $
            MOV.L   @W_ADR,ER1              ; 書き込みアドレス
;
            MOV.W   #128,E0
FWRT10      MOV.B   @ER2+,R0L               ; 再書き込みデータ（バイト単位）
            MOV.B   R0L,@ER1                ; ダミーライト（バイト単位）
            ADDS    #1,ER1
            DEC.W   #1,E0
            BNE     FWRT10                  ; 128 バイト分 繰り返し
;
            MOV.W   #H'A579,R0              ; 25MHZ ( 655.36μS )
            MOV.W   R0,@TCSR                ; ウォッチドッグタイマ設定
            MOV.W   #WLOOP50,E0
            BSET.B  #PSU,@ER6               ; PSU ビットセット
FWRT20      DEC.W   #1,E0                   ; PSU 設定後ウェイト(50μS以上)
            BNE     FWRT20:16
;
;======= WRITE パルス印可 ========================
            BSET.B  #P,@ER6                 ; P ビットセット（書き込み）
FWRT40      DEC.L   #1,ER3                  ; 書き込み時間：10μS or 30μS or 200μS
            BNE     FWRT40:16
;================================================
            MOV.W   #WLOOP5,E0
            BCLR.B  #P,@ER6                 ; P ビットをクリア
FWRT50      DEC.W   #1,E0                   ; P 解除後ウェイト(5μS)
            BNE     FWRT50:16
;
            MOV.W   #WLOOP5,E0
            BCLR.B  #PSU,@ER6               ; PSU ビットをクリア
FWRT60      DEC.W   #1,E0                   ; PSU 解除後ウェイト(5μS以上)
            BNE     FWRT60:16
;
            MOV.W   #H'A500,R0
            MOV.W   R0,@TCSR                ; ウォッチドッグタイマ停止
            RTS
;
```

```
; ********************************************************************************
; * 名称 / フラッシュ 1 ブロック消去ルーチン                                      *
; * 機能 / フラッシュの指定ブロックを消去する                                     *
; * 入力 /  ER6          = FLMCR レジスタのアドレス                              *
; *         ER5          = EBR レジスタのアドレス                                *
; *         @EVF_ST      = イレース先頭アドレス                                  *
; *         @EVF_ED      = イレース終了アドレス                                  *
; *         @BLK_NO      = 消去対象ブロックのビット番号                          *
; *         @ET_COUNT    = 消去最大回数                                          *
; * 出力 /  R0L          = 結果フラグ(OK=H'00,NG=H'01)                           *
; ********************************************************************************
BLK1_ERASE .EQU    $
           MOV.W   #H'5A5F,R0              ; WDT 初期設定
           MOV.W   R0,@RSTCSR
;
           MOV.W   #WLOOP1,R0
           BSET.B  #SWE,@ER6               ; SWE ビットセット
BLK1_10    DEC.W   #1,R0                   ; SWE 設定後ウェイト(1μS 以上)
           BNE     BLK1_10:16
;
           XOR.W   R0,R0                   ; 消去回数カウンタクリア
           MOV.W   R0,@COUNT
;
;======= 初期ベリファイ ========================
           BSR     FERASEVF                ; 初期イレースベリファイ
           CMP.B   #OK,R0L
           BEQ     BLK1_40                 ; 初期ベリファイ完了
;
;======= 消去&ベリファイ ======================
BLK1_20    BSR     FERASE                  ; イレース
           BSR     FERASEVF                ; イレースベリファイ
           CMP.B   #OK,R0L
           BEQ     BLK1_40                 ; イレース完了
;
           MOV.W   @COUNT,R0               ; 消去回数カウンタ @COUNT+ 1
           INC.W   #1,R0
           MOV.W   R0,@COUNT
           MOV.W   @ET_COUNT,E0
           CMP.W   E0,R0
           BNE     BLK1_20                 ; 回数判定（最大消去回数）

;------- 異常終了 ------------------------------
BLK1_30    MOV.W   #WLOOP100,R0
           BCLR.B  #SWE,@ER6               ; SWE ビットクリア
BLK1_35    DEC.W   #1,R0                   ; SWE 解除後ウェイト(100μS 以上)
           BNE     BLK1_35:16
;
           MOV.B   #NG,R0L                 ; NG セット
           RTS
;
```

```
;------- 正常終了 ----------------------------
BLK1_40     MOV.W   #WLOOP100,R0
            BCLR.B  #SWE,@ER6               ; SWE ビットクリア
BLK1_45     DEC.W   #1,R0                   ; SWE 解除後ウェイト(100μS 以上)
            BNE     BLK1_45:16
;
            MOV.B   #OK,R0L                 ; OK セット
            RTS
;
; ********************************************************************************
; * 名称 / 消去ベリファイルーチン                                                *
; * 機能 / 指定ブロックの消去ベリファイ                                          *
; * 入力 /   ER6       = FLMCR レジスタのアドレス                                *
; *          @EVF_ST   = ベリファイ先頭アドレス                                  *
; *          @EVF_ED   = ベリファイ終了アドレス                                  *
; * 出力 /   ROL       = ベリファイ結果(OK=H'00,NG=H'01)                         *
; ********************************************************************************
FERASEVF    .EQU    $
            MOV.L   @EVF_ST,ER1
            MOV.L   @EVF_ED,ER2
            MOV.W   #H'FFFF,E0              ; ダミーライト、消去確認データ
            MOV.W   #WLOOP6,R0
            BSET.B  #EV,@ER6                ; EV ビットセット
VRF10       DEC.W   #1,R0                   ; EV 設定後ウェイト(6μS 以上)
            BNE     VRF10:16
;
VRF30       MOV.W   #WLOOP2,R0
            MOV.W   E0,@ER1                 ; ダミーライト（アドレス・ラッチ）
VRF40       DEC.W   #1,R0                   ; ラッチ後ウェイト（2μS 以上）
            BNE     VRF40:16
;
            MOV.W   @ER1+,R0
            CMP.W   E0,R0                   ; ベリファイ
            BNE     VRF60                   ; 対象アドレスが未消去のとき、終了
            CMP.L   ER1,ER2
            BNE     VRF30
;
            MOV.W   #WLOOP4,R0
            BCLR.B  #EV,@ER6                ; EV ビットをクリア
VRF50       DEC.W   #1,R0                   ; EV 解除後ウェイト（4μS 以上）
            BNE     VRF50:16
            MOV.B   #OK,R0L                 ; OK フラグセット
            RTS
;
;-------  FERASEVF ERR  ------------------------
VRF60       MOV.W   #WLOOP4,R0
            BCLR.B  #EV,@ER6                ; EV ビットをクリア
VRF70       DEC.W   #1,R0                   ; EV 解除後ウェイト（4μS 以上）
            BNE     VRF70:16
;
            MOV.B   #NG,R0L                 ; NG フラグセット
            RTS
;
```

```
; ********************************************************************************
; * 名称 / 消去ルーチン                                                          *
; * 機能 / 指定ブロックの消去                                                    *
; * 入力 /  ER6      = FLMCR レジスタのアドレス                                  *
; *         ER5      = EBR レジスタのアドレス                                    *
; *         @BLK_NO  = 消去対象ブロックのビット番号                              *
; * 出力 / -                                                                     *
; ********************************************************************************
FERASE      .EQU    $
            MOV.B   @BLK_NO,R0H
            BSET.B  R0H,@ER5                ; 消去対象ブロックのビット EBR にセット
            MOV.W   #H'A57C,R0              ; 25MHZ ( 20.8mS )
            MOV.W   R0,@TCSR                ; ウォッチドッグタイマ設定
            MOV.W   #WLOOP100,R0
            BSET.B  #ESU,@ER6               ; ESU ビットセット
FERS10      DEC.W   #1,R0                   ; ESU 設定後ウェイト(100μS 以上)
            BNE     FERS10:16
;
            MOV.L   #TIME10000,ER0          ; 10mS
;======= ERASE パルス印可 ======================
            BSET.B  #E,@ER6                 ; E ビットセット（消去）
FERS20      DEC.L   #1,ER0                  ; 消去時間：10mS
            BNE     FERS20:16
;================================================
            MOV.W   #WLOOP10,R0
            BCLR.B  #E,@ER6                 ; E ビットをクリア
FERS30      DEC.W   #1,R0                   ; E 解除後ウェイト(10μS 以上)
            BNE     FERS30:16
;
            MOV.W   #WLOOP10,R0
            BCLR.B  #ESU,@ER6               ; ESU ビットをクリア
FERS40      DEC.W   #1,R0                   ; ESU 解除ウェイト(10μS 以上)
            BNE     FERS40:16
;
            MOV.W   #H'A500,R0
            MOV.W   R0,@TCSR                ; ウォッチドッグタイマ停止
            MOV.B   @BLK_NO,R0H
            BCLR.B  R0H,@ER5                ; EBR 中の消去対象ブロックをクリア
            RTS
;
;
```

```
; 消去ブロック先頭アドレステーブル
    .ALIGN  2
BLOCKADR1   .EQU    $
            .DATA.L         H'00000000      ; EB0   4KBYTE
            .DATA.L         H'00001000      ; EB1   4KBYTE
            .DATA.L         H'00002000      ; EB2   4KBYTE
            .DATA.L         H'00003000      ; EB3   4KBYTE
            .DATA.L         H'00004000      ; EB4   4KBYTE
            .DATA.L         H'00005000      ; EB5   4KBYTE
            .DATA.L         H'00006000      ; EB6   4KBYTE
            .DATA.L         H'00007000      ; EB7   4KBYTE
            .DATA.L         H'00008000      ; END_ADRESS1
BLOCKADR2   .EQU    $
            .DATA.L         H'00008000      ; EB8  32KBYTE
            .DATA.L         H'00010000      ; EB9  64KBYTE
            .DATA.L         H'00020000      ; EB10 64KBYTE
            .DATA.L         H'00030000      ; EB11 64KBYTE
            .DATA.L         H'00040000      ; END_ADRESS2
            .DATA.L         H'00000000      ; DUMMY
            .DATA.L         H'00000000      ; DUMMY
            .DATA.L         H'00000000      ; DUMMY
            .DATA.L         H'00000000      ; DUMMY
ENDFILE     .EQU    $
    .END
```

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。