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。

H8/300H Tiny シリーズ

I²C バスを使用した FLASH 書き換え

要旨

H8/3664 の I²C バスインタフェースを使用して、内蔵 FLASH メモリの内容を書き換えます。

動作確認デバイス

H8/3664

目次

1. 仕様

（1）H8/3664 の I2C バスインタフェースを使用して、内蔵 FLASH メモリの内容を書き換えます。
転送元（H8/3664）は、①送信 SW が ON することにより、転送元の内蔵 FLASH メモリのユーザプログラム(H'1000～H'7FFF 番地)の内容を 128+2(CRC)バイトごとに I²C を経由しデータ送信をします。
転送先（H8/3664）は、②受信 SW が ON することにより、内蔵 FLASH メモリの H'1000～H'7FFF を消去します。次に転送元から送られてくる I²C の受信データを内蔵 FLASH メモリの H'1000 から順次書き込みます。
（2）送信 SW・受信 SW が規定時間(約 5 秒)以内に ON しない場合は、ユーザプログラムを実行します。
本タスクのユーザプログラムは参考例として LED を点滅するプログラムとします。
（3）本システムの I²C バスに接続されているデバイスは、マスタデバイス（H8/3664）1 個、スレーブデバイス（H8/3664）1 個の構成とします。図. 1 に H8/3664 の接続例を示します。
（4）接続する H8/3664 のスレーブアドレスは[1000000]とし、転送クロックの周波数は 400kHz とします。
（5）送信データには CRC を付加し、受信側で CRC のチェックをします。

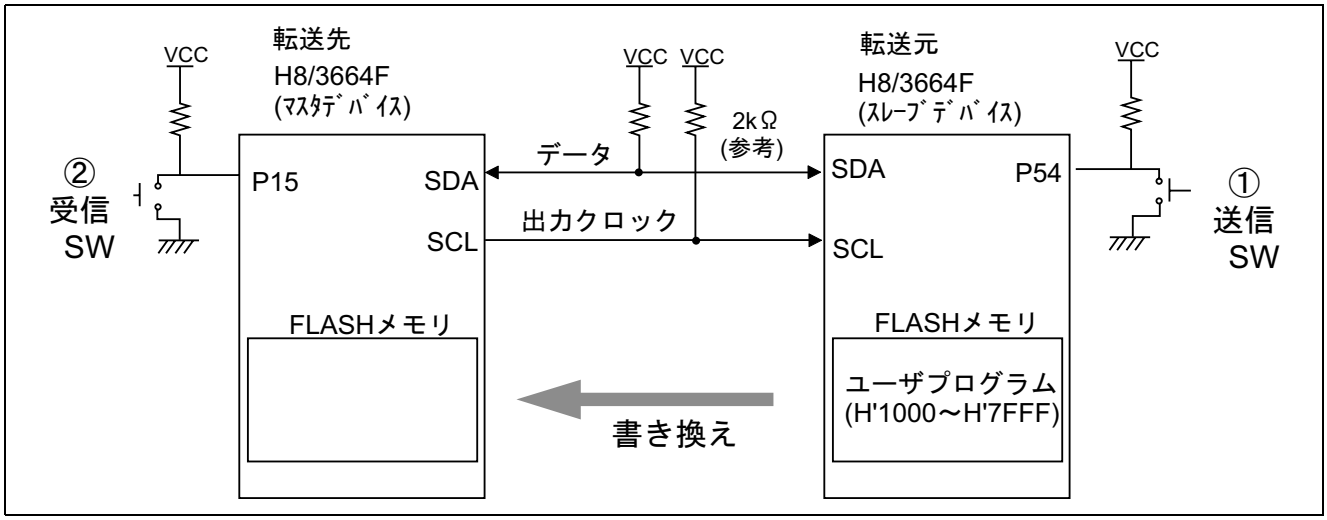


図. 1　IIC **バスインタフェース接続による** FLASH **メモリ書き換え**

2. 詳細仕様説明

（1）I²C バスインタフェース基本フォーマット(送信要求・データ)を図.2 に示します。

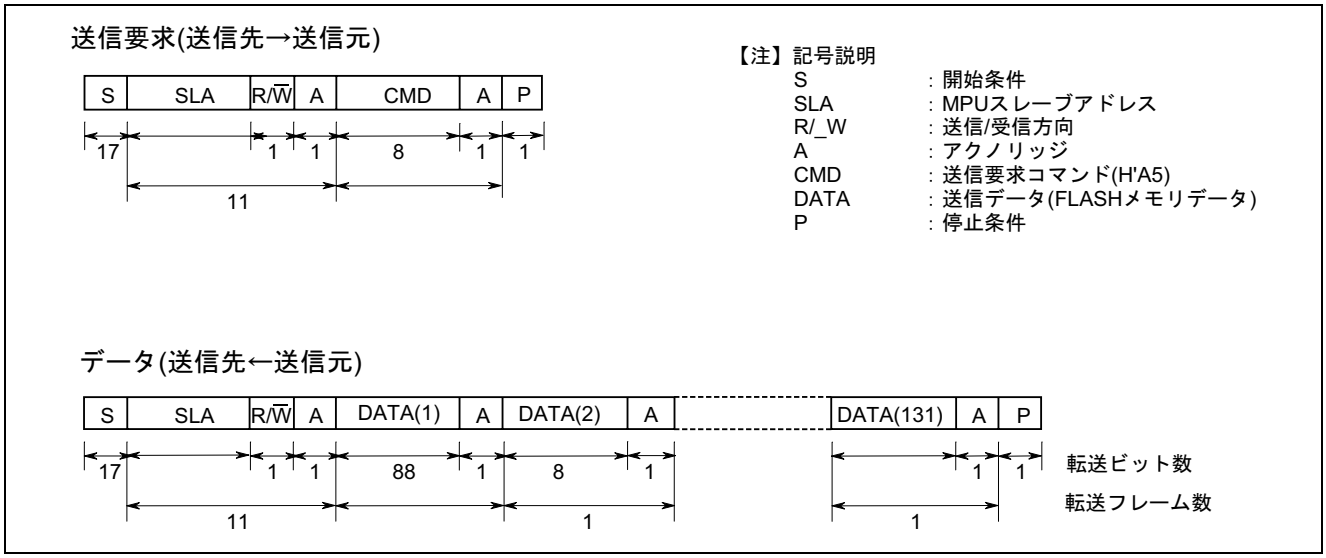


図.2 I²C **バスインタフェースフォーマット**

2.1 レジスタの説明

フラッシュメモリには以下のレジスタがあります。

・フラッシュメモリコントロールレジスタ 1（FLMCR1）
・フラッシュメモリコントロールレジスタ 2（FLMCR2）
・ブロック指定レジスタ 1（EBR1）
・フラッシュメモリパワーコントロールレジスタ（FLPWCR）
・フラッシュメモリイネーブルレジスタ（FENR）

2.1.1 フラッシュメモリコントロールレジスタ 1（FLMCR1）

FLMCR1 はフラッシュメモリをプログラムモード、プログラムベリファイモード、イレースモード、イレースベリファイモードに遷移させます。

ビット	ビット名	初期値	R/W	機能
7	—	0	—	リザーブビットです。読み出すと常に 0 が読み出されます。
6	SWE	0	R/W	ソフトウェアライトイネーブル このビットが 1 のときフラッシュメモリの書き込み／消去が可能となります。0 のときこのレジスタの他のビットと EBR1 の各ビットはセットできません。
5	ESU	0	R/W	イレースセットアップ 1 にセットするとイレースセットアップ状態となり、クリアするとセットアップ状態を解除します。FLMCR1 の E ビットを 1 にセットする前にセットしてください。
4	PSU	0	R/W	プログラムセットアップ 1 にセットするとプログラムセットセットアップ状態となり、クリアするとセットアップ状態を解除します。FLMCR1 の P ビットを 1 にセットする前にセットしてください。
3	EV	0	R/W	イレースベリファイ 1 にセットするとイレースベリファイモードへ遷移し、クリアするとイレースベリファイモードを解除します。
2	PV	0	R/W	プログラムベリファイ 1 にセットするとプログラムベリファイモードへ遷移し、クリアするとプログラムベリファイモードを解除します。
1	E	0	R/W	イレース SWE＝1、ESU＝1 の状態でこのビットを 1 にセットするとイレースモードへ遷移し、クリアするとイレースモードを解除します。
0	P	0	R/W	プログラム SWE＝1、PSU＝1 の状態でこのビットを 1 にセットするとプログラムモードへ遷移し、クリアするとプログラムモードを解除します。

2.1.2 フラッシュメモリコントロールレジスタ 2（FLMCR2）

FLMCR2 はフラッシュメモリの書き込み／消去の状態を表示します。FLMCR2 は読み出し専用レジスタです。書き込みはしないでください。

ビット	ビット名	初期値	R/W	機能
7	FLER	0	R	このビットはフラッシュメモリの書き込み／消去中にエラーを検出し、エラープロテクト状態となったときセットされます。
6〜0	—	0	—	リザーブビットです。読み出すと常に 0 が読み出されます。

2.1.3 ブロック指定レジスタ 1（EBR1）

フラッシュメモリの消去ブロックを指定するレジスタです。FLMCR1 の SWE ビットが 0 のときは EBR1 は H'00 に初期化されます。このレジスタは 2 ビット以上同時に 1 に設定しないでください。設定すると EBR1 は 0 にオートクリアされます。

ビット	ビット名	初期値	R/W	機能
7～5	—	0	—	リザーブビットです。読み出すと常に 0 が読み出されます。
4	EB4	0	R/W	このビットが 1 のとき H'1000～H'7FFF の 28k バイトが消去対象となります。
3	EB3	0	R/W	このビットが 1 のとき H'0C00～H'0FFF の 1k バイトが消去対象となります。
2	EB2	0	R/W	このビットが 1 のとき H'0800～H'0BFF の 1k バイトが消去対象となります。
1	EB1	0	R/W	このビットが 1 のとき H'0400～H'07FF の 1k バイトが消去対象となります。
0	EB0	0	R/W	このビットが 1 のとき H'0000～H'03FF の 1k バイトが消去対象となります。

2.1.4 フラッシュメモリパワーコントロールレジスタ（FLPWCR）

マイコンがサブアクティブモードに遷移するときフラッシュメモリを低消費電力モードにするかどうかを選択します。低消費電力モードでは電源回路の一部が停止しますが、サブアクティブモードでは読み出し可能です。

ビット	ビット名	初期値	R/W	機能
7	PDWND	0	R/W	パワーダウンイネーブル／ディスエーブル このビットが 0 のときサブアクティブモードに遷移するとフラッシュメモリは低消費電力モードとなります。 このビットが 1 のときはサブアクティブモードに遷移してもフラッシュメモリは通常モードで動作します。
6～0	—	0	—	リザーブビットです。読み出すと常に 0 が読み出されます。

2.1.5 フラッシュメモリイネーブルレジスタ（FENR）

FENR はフラッシュメモリ制御レジスタ FLMCR1、FLMCR2、EBR1、FLPWCR の CPU からのアクセスを制御します。

ビット	ビット名	初期値	R/W	機能
7	FLSHE	0	R/W	フラッシュメモリコントロールレジスタイネーブル／ディスエーブル このビットを 1 にセットすると、フラッシュメモリ制御レジスタがアクセス可能となります。0 のときは制御レジスタはアクセスできません。
6～0	—	0	—	リザーブビットです。読み出すと常に 0 が読み出されます。

2.2 ユーザモードでの書き込み／消去

ユーザモードでもユーザが用意した書き込み／消去プログラムに分岐することで任意のブロックをオンボードで消去し書き換えることができます。分岐のための条件設定やオンボードでの書き換えデータ供給手段をユーザ側で用意する必要があります。
また、必要に応じてフラッシュメモリの一部に書き込み／消去プログラムを書き込んでおくか、書き込み／消去プログラムを外部から供給するためのプログラムを書き込んでおく必要があります。書き込み／消去中はフラッシュメモリを読み出せないため、ブートモードと同様書き込み／消去プログラムは内蔵 RAM に転送して実行してください。用意する書き込み／消去プログラムは 2.3 の仕様に書き込み／消去プログラムに沿ったものを用意してください。

2.3 書き込み／消去プログラム

オンボードでのフラッシュメモリの書き込み／消去は CPU を用いてソフトウェアで行う方式を採用しています。フラッシュメモリは FLMCR1 の設定によってプログラムモード、プログラムベリファイモード、イレースモード、イレースベリファイモードに遷移します。ブートモードでの書き込み制御プログラム、ユーザモードでの書き込み／消去プログラムではこれらのモードを組み合わせて書き込み／消去を行います。フラッシュメモリへの書き込みは「2.3.1 章　プログラム／プログラムベリファイ」に沿って、また、フラッシュメモリの消去は「2.3.2 章　イレース／イレースベリファイ」に沿って行ってください。

2.3.1 プログラム／プログラムベリファイ

1. 書き込みは消去状態で行い、既に書き込まれたアドレスへの再書き込みは行わないでください。
2. 1 回の書き込みは 128 バイト単位です。128 バイトに満たないデータを書き込む場合もフラッシュメモリに 128 バイトのデータを転送する必要があります。書き込む必要のないアドレスのデータは H'FF にして書き込んで下さい。
3. RAM 上に書き込みデータエリア 128 バイト、再書き込みデータエリア 128 バイト、追加書き込みデータエリア 128 バイトの領域を確保して下さい。再書き込みデータの演算は表. 1 に、追加書き込みデータの演算は表. 2 にしたがってください。
4. 再書き込みデータエリアあるいは追加書き込みデータエリアからフラッシュメモリへはバイト単位で 128 バイト連続転送してください。プログラムアドレスと 128 バイトのデータがフラッシュメモリ内にラッチされます。転送先のフラッシュメモリの先頭アドレスは下位 8 ビットを H'00 または H'80 としてください。
5. P ビットがセットされている時間が書き込み時間となります。書き込み時間は表. 3 にしたがってください。
6. ウォッチドックタイマの設定はプログラムの暴走等による過剰書き込みを避けるためのものです。オーバフロー周期は 6.6ms 程度としてください。
7. ベリファイアドレスへのダミーライトは、下位 2 ビットが b'00 のアドレスに H'FF を 1 バイト書き込んでください。ベリファイデータはダミーライトを行った番地からロングワードで読み出せます。
8. 同一ビットに対するプログラム／プログラムベリファイシーケンスの繰り返しは、1,000 回を超えないようにしてください。

表.1 再書き込みデータ演算表

書き込みデータ	ベリファイデータ	再書き込みデータ	備考
0	0	1	**書き込み完了ビット**
0	1	0	**再書き込みビット**
1	0	1	—
1	1	1	**消去状態のまま**

表.2 追加書き込みデータ演算表

書き込みデータ	ベリファイデータ	追加書き込みデータ	備考
0	0	1	追加書き込みビット
0	1	0	追加書き込みは実施しない
1	0	1	追加書き込みは実施しない
1	1	1	追加書き込みは実施しない

表.3 書き込み時間

n（書き込み回数）	書き込み時[μs]	追加書き込み時[μs]	備考
1〜6	30	10	
7〜1,000	200	—	

2.3.2 イレース／イレースベリファイ

1. 消去の前にプレライト（消去するメモリの全データをすべて 0 にする）を行う必要はありません。
2. 消去はブロック単位で行います。ブロック指定レジスタ 1（EBR1）により消去するブロックを 1 ブロックだけ選択してください。複数のブロックを消去する場合も 1 ブロックずつ順次消去してください。
3. E ビットが設定されている時間が消去時間となります。
4. ウォッチドックタイマの設定はプログラムの暴走等による過剰書き込みを避けるためのものです。オーバフロー周期は 19.8ms 程度としてください。
5. ベリファイアドレスへのダミーライトは、下位 2 ビットが b'00 のアドレスに H'FF を 1 バイト書き込んでください。ベリファイデータはダミーライトを行った番地からロングワードで読み出せます。
6. 読み出したデータが未消去の場合は再度イレースモードに設定し、同様にイレース／イレースベリファイシーケンスを繰り返します。ただし、この繰り返し回数が 100 回を超えないようにしてください。

2.3.3 フラッシュメモリの書き込み／消去時の割り込み

フラッシュメモリへの書き込み／消去中またはブートプログラム実行中は以下の理由から NMI を含むすべての割り込み要求を禁止してください。

1. 書き込み／消去中に割り込みが発生すると、正常な書き込み／消去アルゴリズムに沿った動作が保証できなくなる。
2. ベクタアドレスが書き込まれる前、または書き込み／消去中に割り込み例外処理を開始すると、正常なベクタフェッチができず CPU が暴走する。
3. ブートプログラム実行中に割り込みが発生すると、正常なブートモードのシーケンスを実行できなくなる。

2.4 通信プロトコル（通信手順）

FLASH 書き換えの通信プロトコルを説明します。図 XX.3 に通信プロトコルを示します。

マスタ側（転送先）からデータ送信要求を出力し、スレーブ側（転送元）はこのデータ送信要求の受信により 128 バイトのデータ送信を出力します。同様に手順で H'1000～H'107F(1 番目)、H'1080～H'10FF(2 番目)、・・・・H'7F80～H'7FFF(224 番目)まで繰り返します。

通信エラー（CRC の不一致など）が発生した場合、通信および書込み処理を中止します。

また、正常終了状態・通信エラー状態いずれも RESET により復帰します。

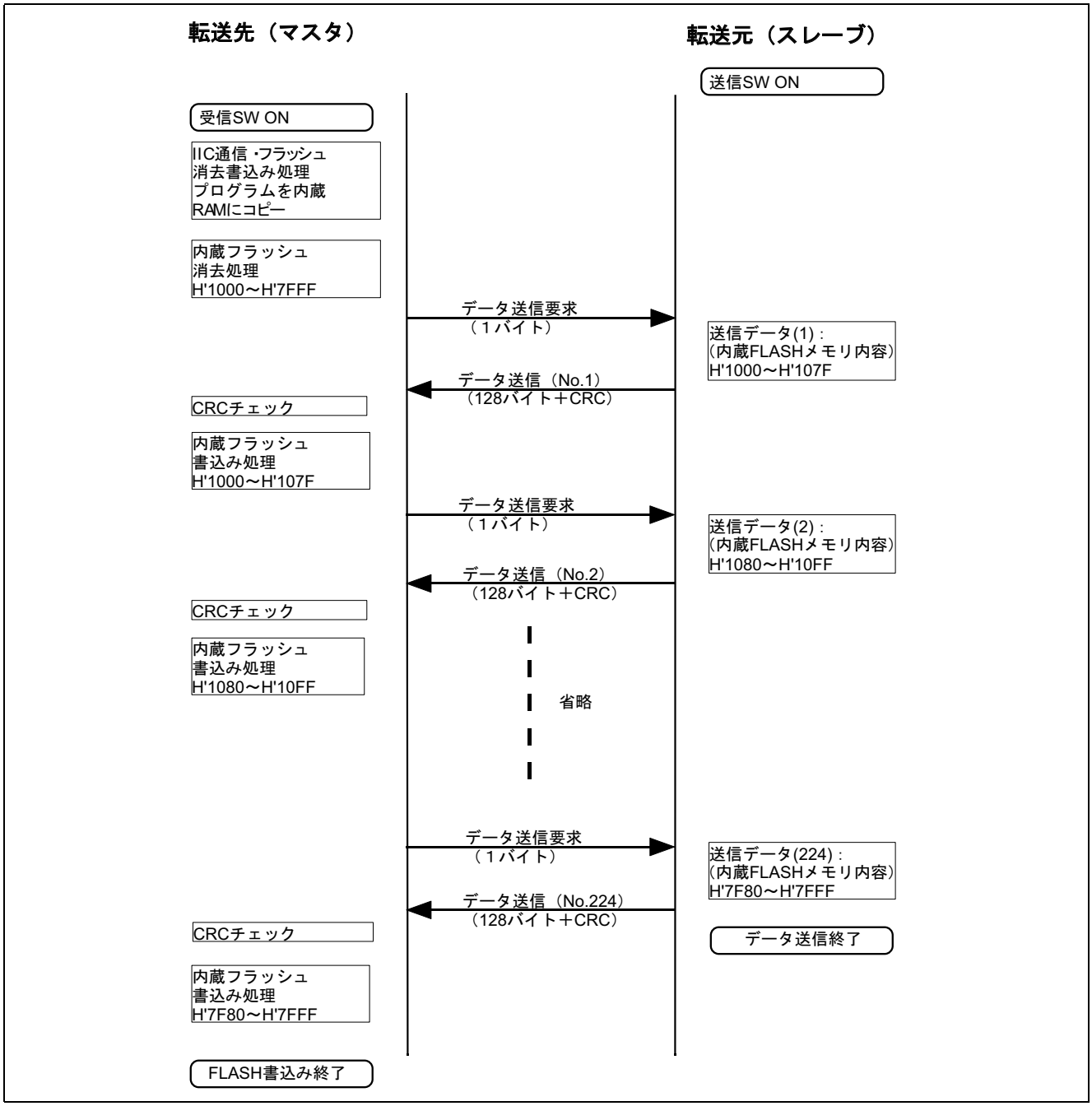


図.3 通信プロトコル（通信手順）

2.5 プログラム構成とメモリマップ

FLASH 書き換えのプログラム構成を説明します。

FLASH メモリの H'0400～H'0BFF に I²C 通信プログラムと FLASH 消去・書込みプログラムが格納されています。

転送元の場合、上記のエリアからプログラム実行されますが、転送先では、I²C 通信プログラムと FLASH 消去・書込みプログラムは内蔵 RAM 空間(H'F780～H'FC7F)に書き込まれ、内蔵 RAM 空間からの実行になります。

＜ユーザベクタ＞ ユーザ割込み処理の変更に対応するためベクタテーブルを H'1000 番地に実現しています。

＜内蔵 RAM 占有＞FLASH 書き換えが起動されると内蔵 RAM のほとんどを占有します。しかし書き換えが起動されなければ内蔵 RAM を占有しませんのでユーザプログラムでは内蔵 RAM 空間を自由に使用できます。

＜E10T エミュレータ使用＞ E10T を使用してエミュレータ動作と共にプログラムを書き込んだ場合下図の H'7000 番地にはエミュレータのワーク領域として使用されます。本タスクではこのエミュレータのワーク領域も含めて書き換えを行います。

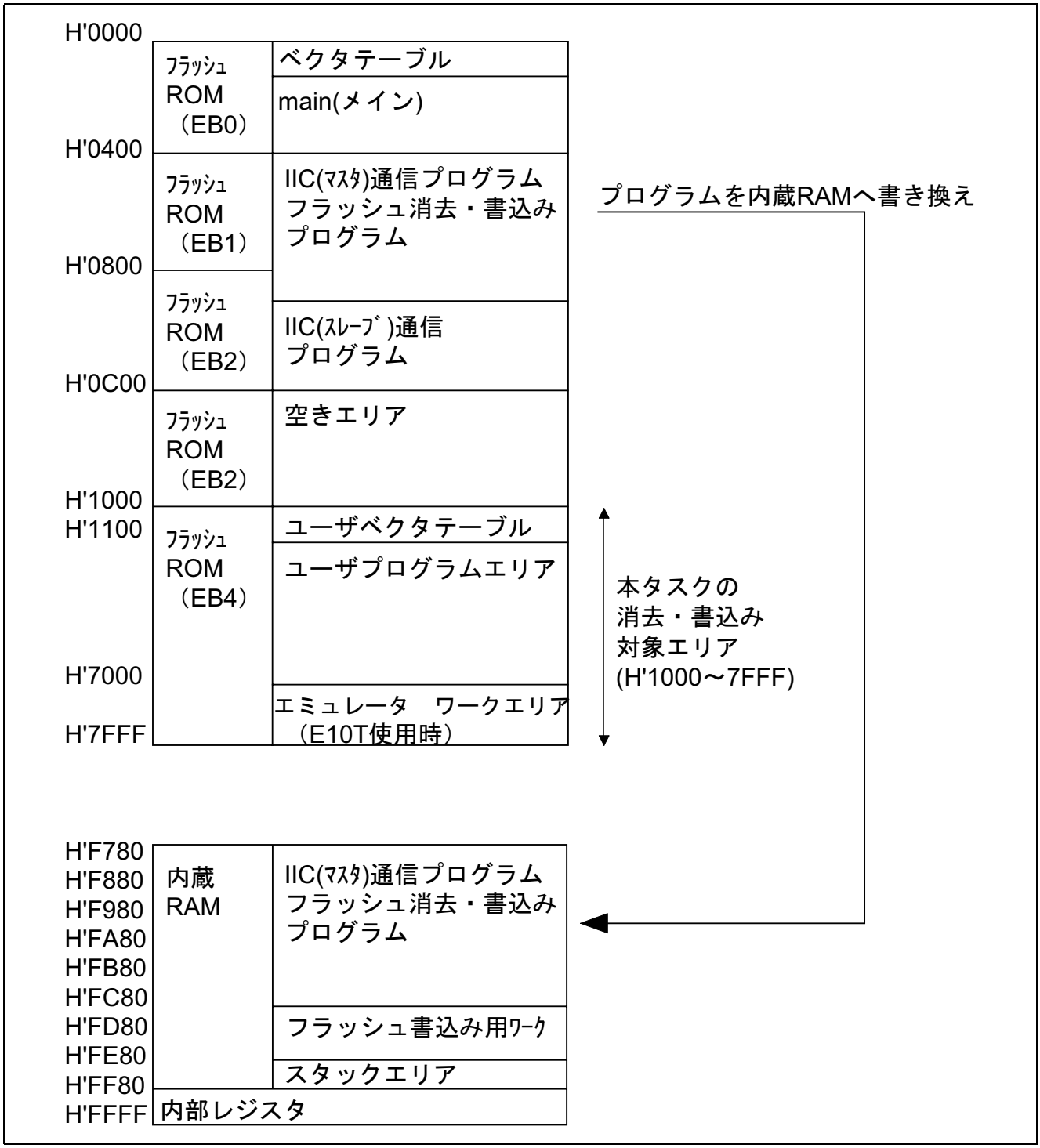


図.4 プログラム構成（メモリマップ）

3. ソフトウェア説明

モジュール説明

本タスク例におけるモジュール説明（引数・リターン値）を表.4 に示します。

表.4 モジュール説明表

モジュール（関数）名	引数	リターン値	機能
INIT(アセンブリ言語)	なし	なし	スタックポインタの設定(R7 に H'FF80 をセット) CCR の設定（割込み禁止） main へジャンプする
main	なし	なし	メインモジュール
flprg_cpy	なし	なし	0x0400～0x08FF のデータを 0xF780～0xFC7F にコピーする
jump_prog (アセンブリ言語)	R0 (Jump 先の番地)	なし	R0 番地へ Jump する
wait	limit（ウェイト）	なし	ウエイト
_SL_TRANS (アセンブリ言語)	なし	なし	スレーブモード送受信
SL_RECV_DATA (アセンブリ言語)	R4（受信データ格納アドレス）、R5（受信バイト数）	R0L（受信結果）	スレーブモード受信処理
SL_SEND_DATA (アセンブリ言語)	R4（送信データ格納アドレス）、R5（送信バイト数）	R0L（送信結果）	スレーブモード送信処理
CAL_CRC16 (アセンブリ言語)	R4（受信データ格納アドレス）	R0（CRC 演算結果）	CRC 演算処理
_IIC_TEST (アセンブリ言語)	なし	なし	フラッシュ消去／書き込み処理
FL_ER_BLK (アセンブリ言語)	R0H（イレースブロック指定）	R0L（消去結果）	Flash データ消去処理
BLK1_ERASE (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）、ER5（EBR レジスタのアドレス）	R0L（消去結果）	フラッシュ指定ブロック消去処理
FERASEVF (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）	R0L（ベリファイ結果）	フラッシュ消去ベリファイ処理
FERASE (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）、ER5（EBR レジスタのアドレス）	なし	フラッシュ指定ブロック消去処理
FL_WAIT (アセンブリ言語)	R0（ウエイト）	なし	ウエイト
MA_SEND_DATA (アセンブリ言語)	R4（送信データ格納アドレス）、R5（送信バイト数）	R0L（送信結果）	マスターモード送信処理
MA_RECV_DATA (アセンブリ言語)	R4（受信データ格納アドレス）、R5（受信バイト数）	R0L（受信結果）	マスターモード受信処理
FWRITE128 (アセンブリ言語)	なし	R0L（書き込み結果）	フラッシュ任意の 128Byte 書き込み処理
FWRITEVF (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）	R0L（ベリファイ結果）	フラッシュ書き込みベリファイ処理
FWRITE (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）、ER2（書き込みの先頭アドレス）、ER3（P ビットセット時間）	なし	フラッシュ書き込み処理

アセンブリ言語で書かれたモジュールを C プログラムで参照する時は、先頭のアンダーバー（_）を除いた名前で参照します。例えばアセンブリ言語で書かれたモジュール_SL_TRNS を C プログラムで参照する場合は SL_TRNS となります。

ファイル構成

本ソフトウェアで使用しているファイルと各ファイルの機能を表.5 に示します。

表.5 ファイル構成

ファイル名	機能
dbdct.c	未初期値エリアの初期化処理
u_vect.src	割り込み用レジスタ定義
fl_equ.h	各種レジスタ及びビット定義、各種定数の設定
iic_ram.h	消去／書き込み処理用の RAM エリアの設定
init.src	立ち上げ処理、メインモジュールへのジャンプ
FLWR.c	メインモジュール、データコピー、アドレスジャンプ、ウェイト処理
IIC_SL.src	スレーブモード送受信処理
IIC_MA.src	マスターモード送受信処理、CRC 演算処理
fl_erwr.src	フラッシュの消去／書き込み処理
u_vect.src	割り込み処理
LED.c	ユーザープログラム

本タスク例における定数の説明を表.6 に示します。

表.6 使用定数説明

定義名	値	用途
WLOOP1	1*MHZ/400(=2)	ウエイト文実行回数（ウエイト時間：1μs）
WLOOP2	2*MHZ/400(=5)	ウエイト文実行回数（ウエイト時間：2μs）
WLOOP4	4*MHZ/400(=11)	ウエイト文実行回数（ウエイト時間：4μs）
WLOOP5	5*MHZ/400(=13)	ウエイト文実行回数（ウエイト時間：5μs）
WLOOP10	10*MHZ/400(=27)	ウエイト文実行回数（ウエイト時間：10μs）
WLOOP20	20*MHZ/400(=55)	ウエイト文実行回数（ウエイト時間：20μs）
WLOOP50	50*MHZ/400(=137)	ウエイト文実行回数（ウエイト時間：50μs）
WLOOP100	100*MHZ/400(=275)	ウエイト文実行回数（ウエイト時間：100μs）
TIME	10*MHZ/400(=27)	ウエイト文実行回数（ウエイト時間：10μs）
TIME	30*MHZ/400(=82)	ウエイト文実行回数（ウエイト時間：20μs）
TIME	200*MHZ/400(=550)	ウエイト文実行回数（ウエイト時間：200μs）
TIME	10000*MHZ/400(=27500)	ウエイト文実行回数（ウエイト時間：10ms）
MAXWT	1000	フラッシュ書き込み最大回数
MAXET	100	消去最大回数
OW_COUNT	6	再書き込み回数

本タスク例における使用 RAM の説明を表.7 に示します。

表.7　使用 RAM 説明

ラベル名	機能	アドレス	使用モジュール名
W_BUF	書き込み用データバッファ（128Byte）	H'FC80	_IIC_TEST、_SL_TRNS、FWRITE128、FWRITEVF
BUFF	再書き込み用データバッファ（128Byte）	H'FD00	FWRITE128、FWRITEVF
OWBUFF	追加書き込み用データバッファ（128Byte）	H'FD80	_SL_TRNS、FWRITE128、FWRITEVF
COUNT	書き込み回数／消去回数	H'FE00	FWRITE128、BLK_ERASE
W_ADR	書き込み先頭アドレス	H'FE02	_IIC_TEST、FWRITEVF、FWRITE
W_ADR_ED	書き込み終了アドレス	H'FE04	_IIC_TEST
ET_COUNT	消去最大回数	H'FE08	FL_ER_BLK、BLK1_ERASE
WT_COUNT	書き込み最大回数	H'FE0A	FWRITE128、_IIC_TEST
EVF_ST	消去先頭アドレス	H'FE0C	FL_ER_BLK、FERASEVF
EVF_ED	消去終了アドレス	H'FE0E	FL_ER_BLK、FERASEVF
BLK_NO	消去対象ブロック	H'FE10	FL_ER_BLK、FERASE
VF_RET	書き込みベリファイ結果	H'FE11	FWRITE128
IIC_SBUF	送信データ格納アドレス	H'FE14	_IIC_TEST

本タスク例で使用する内部レジスタ説明を表.8 に示します。

表.8　使用内部レジスタ説明

レジスタ		機能	操作	設定値
ICDR		送信データ・受信データを格納	格納・参照	—
ICMR	MLS	MSB ファーストによるデータ転送の設定	設定	0
	WAIT	データとアクノリッジの連続的な転送を設定	設定	0
	CKS2 to CKS0	STCR の IICX ビットと組み合わせて、転送クロックの周波数を 400kHz に設定	設定	CKS2＝0 CKS1＝0 CKS0＝1
	BC2 to BC0	I²C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	設定	BC2＝0 BC1＝0 BC0＝0
ICCR	ICE	ICMR、ICDR/SAR、SARX レジスタのアクセス制御、I²C バスインターフェースの動作（SCL/SDA 端子はポート機能）/非動作（SCL/SDA 端子はバス駆動機能）の選択	設定	0/1
	IEIC	I²C バスインターフェース割り込み要求を禁止	設定	0/1
	MST	I²C バスインターフェースをマスタモードで使用	設定	0/1
	TRS	I²C バスインターフェースを送信モードで使用	設定	0/1
	ACKE	アクノリッジビットが “ 1 ” の場合、連続的な転送を中断	設定	0/1
	BBSY	I²C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	設定・参照	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝ “ 1 ” の検出	設定	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	設定	0/1
ICSR	ESTP	エラー停止条件検出フラグ（スレーブモード有効）	未操作	—
	STOP	正常停止条件検出フラグ（スレーブモード有効）	未操作	—
	IRTR	連続送受信割り込み要求フラグ	未操作	—
	AASX	第 2 スレーブアドレス認識フラグ	未操作	—
	AL	アービトレーションロストフラグ	未操作	—
	AAS	スレーブアドレス認識フラグ	未操作	—
	ADZ	ゼネラルコールアドレス認識フラグ	未操作	—
	ACKB	EEPROM より送信されたアクノリッジデータを格納	未操作	—
TSCR	IICRST	IIC コントロール部リセット	参照	—
	IICX	転送レート選択	設定	0
FLMCR1	SWE	SWE=1 の時フラッシュの書き込み／消去可能	設定	0
	ESU	ESU=1 の時イレースセットアップモード、ESU=0 で解除	設定	0/1
	PSU	PSU=1 の時プログラムセットアップモード、PSU=0 で解除	設定	0/1
	EV	EV=1 の時イレースベリファイモード、EV=0 で解除	設定	0/1
	PV	PV=1 の時プログラムベリファイモード、PV=0 で解除	設定	0/1
	E	SWE=1、ESU=1 で E=1 の時イレースモード、E=0 で解除	設定	0/1
	P	SWE=1、PSU=1 で P=1 の時プログラムモード、P=0 で解除	設定	0/1
EBR1	EB4〜0	フラッシュメモリの対象ブロックを H’1000〜H’7FFF までの 28K Byte に設定	設定	EB4〜0 ＝H’10
FENR	FLSHE	FLMCR1、EBR1 レジスタを有効にする	設定	0/1

レジスタ		機能	操作	設定値
TCSRWD	B6WI	B6WI＝0 で書き込みを行った時のみ、TCWE の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCWE	TCWE＝1 の時、TCWD レジスタへの書き込み有効	設定	1
	B4WI	B4WI＝0 で書き込みを行った時のみ、TCSRWE の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCSRWE	TCSRWE＝1 の時、WDON、WRST ビットへの書き込み有効	設定	1
	B2WI	B2WI＝0 で書き込みを行った時のみ、WDON の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCWE	WDON＝1 の時 TCWD をカウントアップ、WDON＝0 の時停止	設定	0/1
	B0WI	B0WI＝0 で書き込みを行った時のみ、WRST の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCSRWE	ウォッチドックタイマリセット	設定	1
TMWD	CKS3～0	TCWD に入力するクロックを選択する。 CKS3～0＝H'8：内部クロック（Φ）/64 CKS3～0＝H'D：内部クロック（Φ）/2048	設定	CKS3～0＝H'8 or H'D
TCWD		リード／ライト可能の 8 ビット幅タイマカウンタ	設定	166 or 100

セクション定義

本タスクのセクション定義を表.9 に示します。

表.9 セクション定義表

アドレス	セクション	説明
H'0000	V0	RESET などのベクタアドレス
H'0010	V1	TRAP などのベクタアドレス
H'002E	V2	SCI などのベクタアドレス
H'0040	PM	プログラム領域
H'0400	PF_1	プログラム領域
H'0900	PF_2	プログラム領域
H'1000	UV	ユーザベクタテーブル領域
H'1100	P	ユーザプログラム領域
	C$DSEC	初期化データ領域（DBSCT.C にて定義）
	C$BSEC	未初期化データ領域（DBSCT.C にて定義）
	D	初期化データ領域
H'DE80	B	未初期化データ領域
	R	初期化データ領域

4. モジュール階層図

モジュール階層図を図.5 に示します。

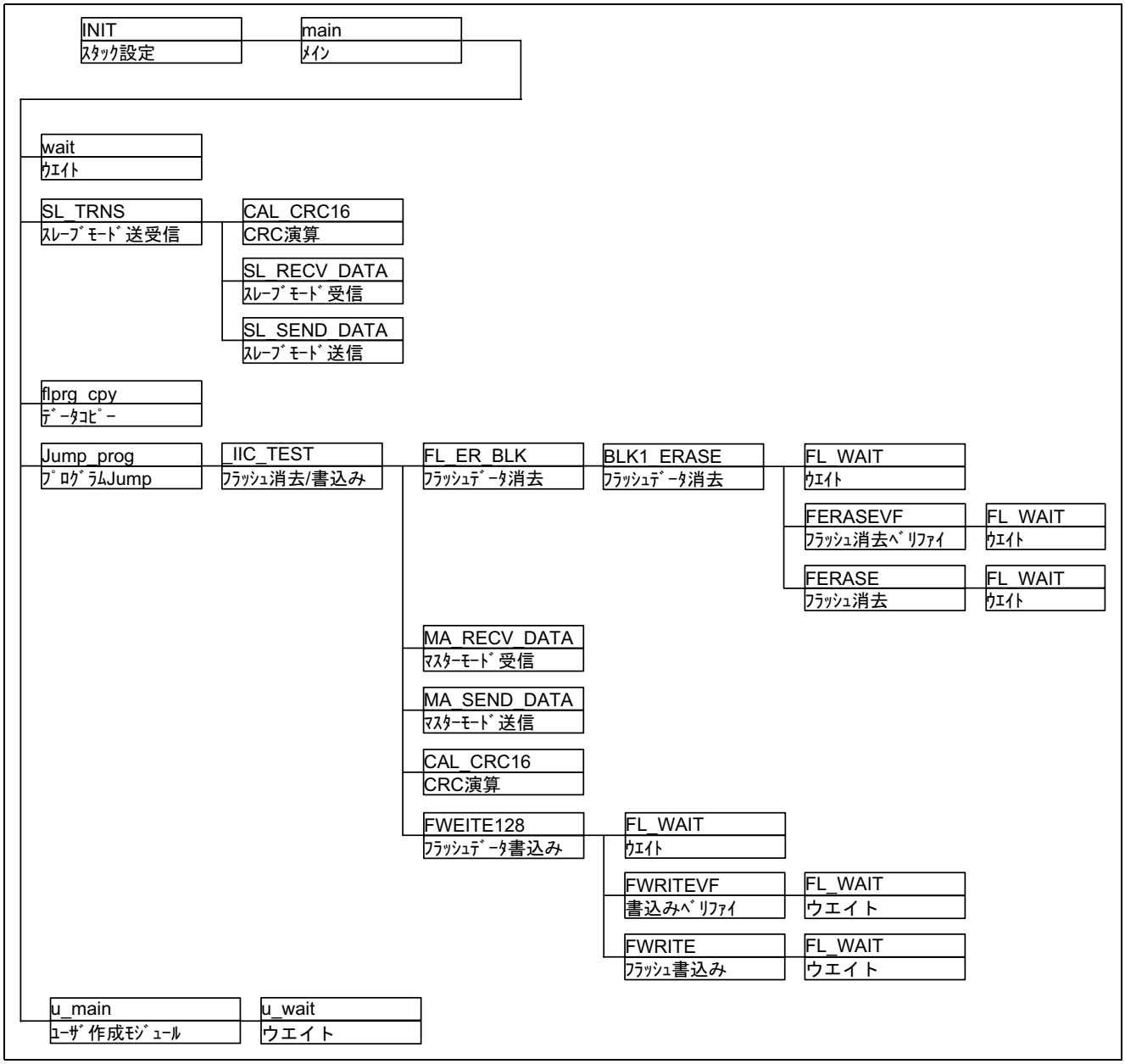
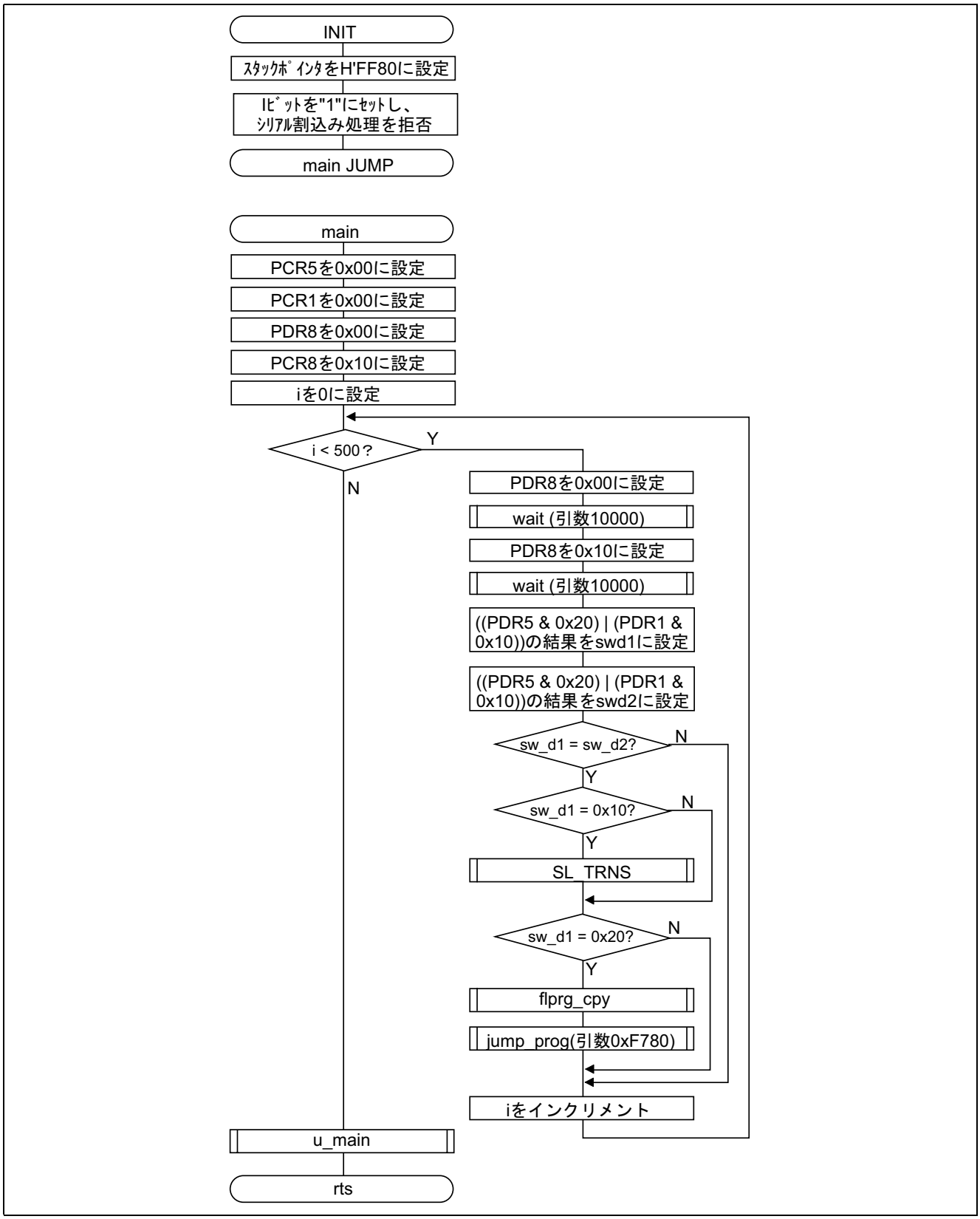
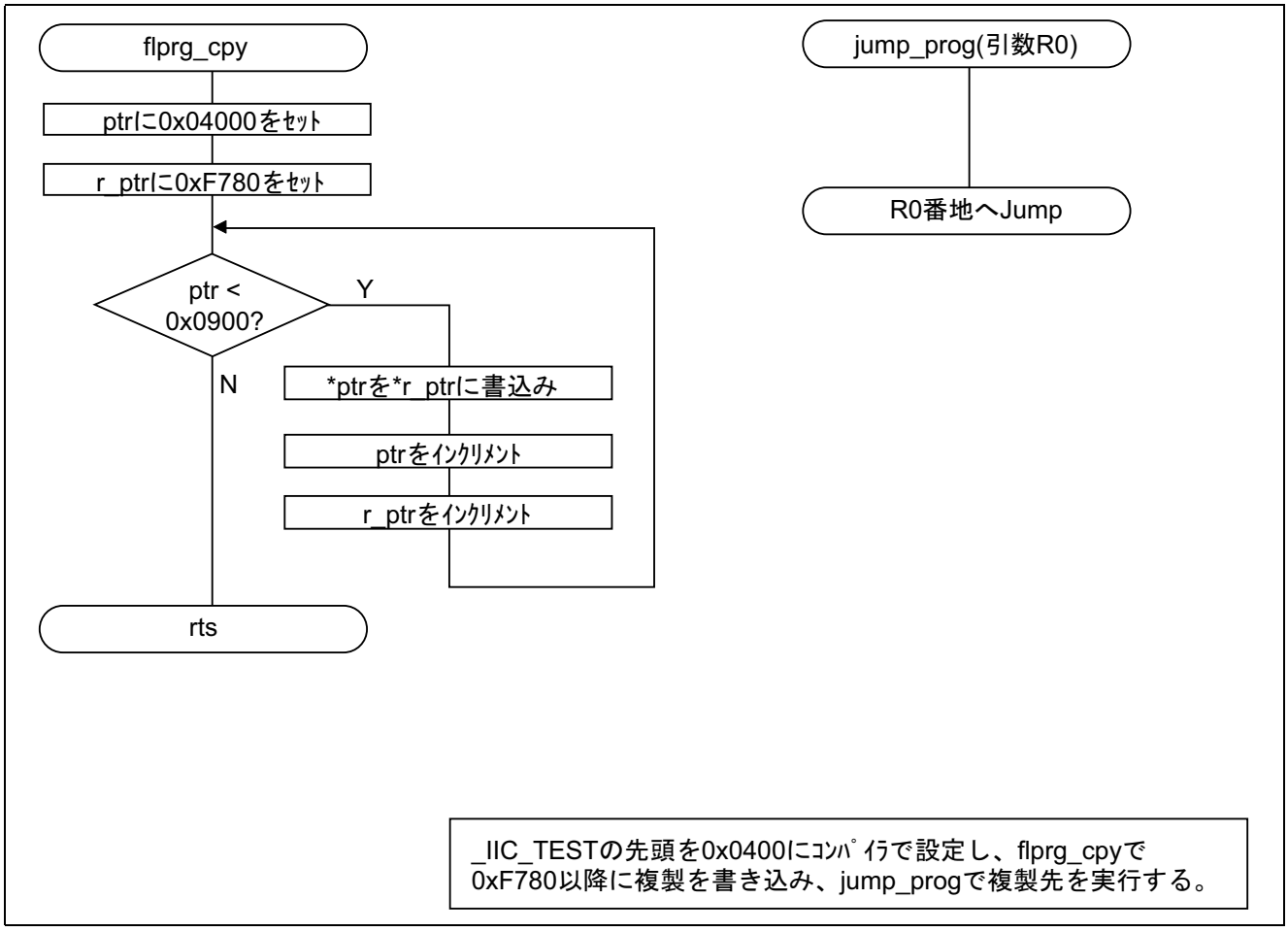
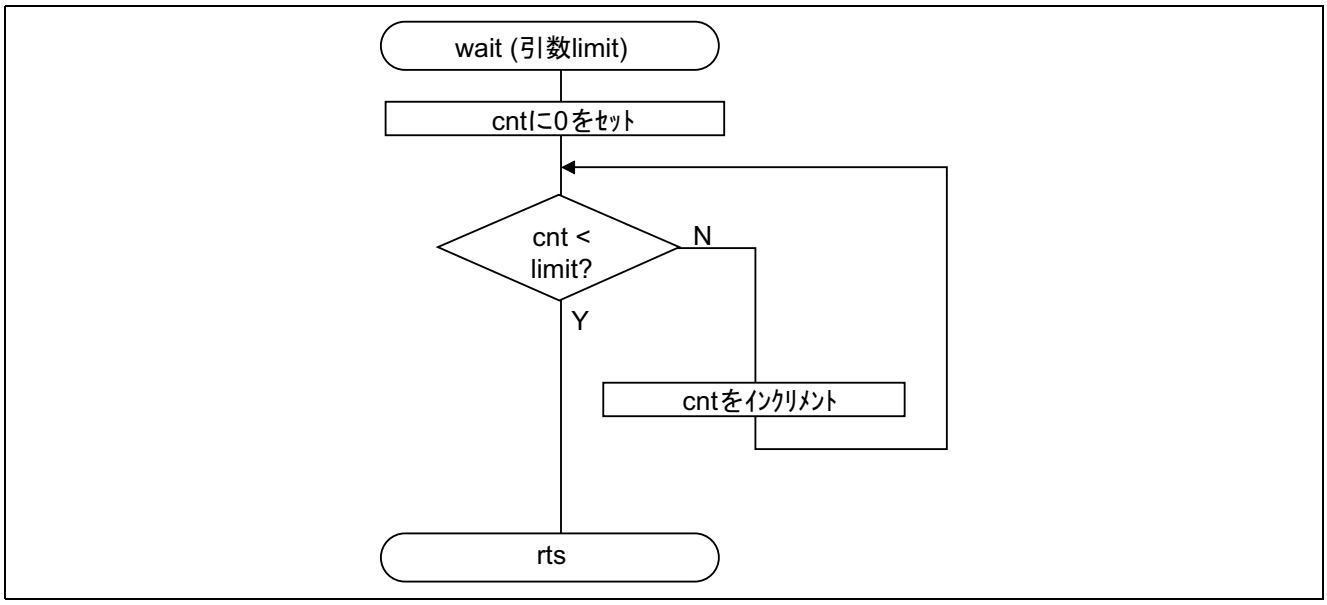


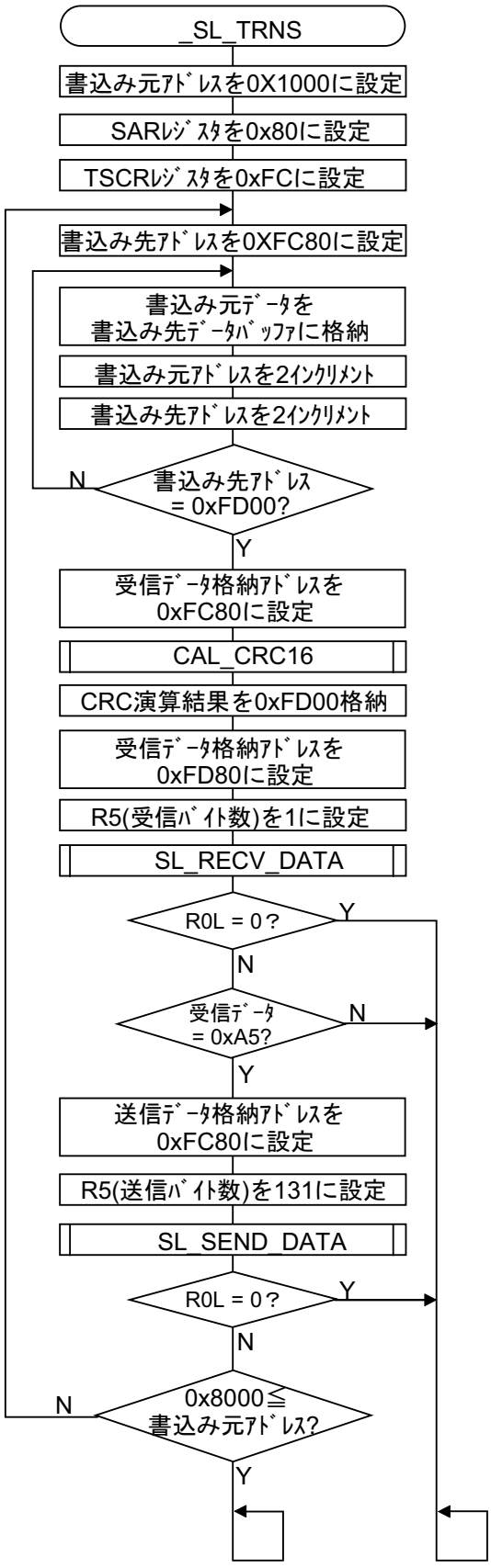
図.5　モジュール階層図

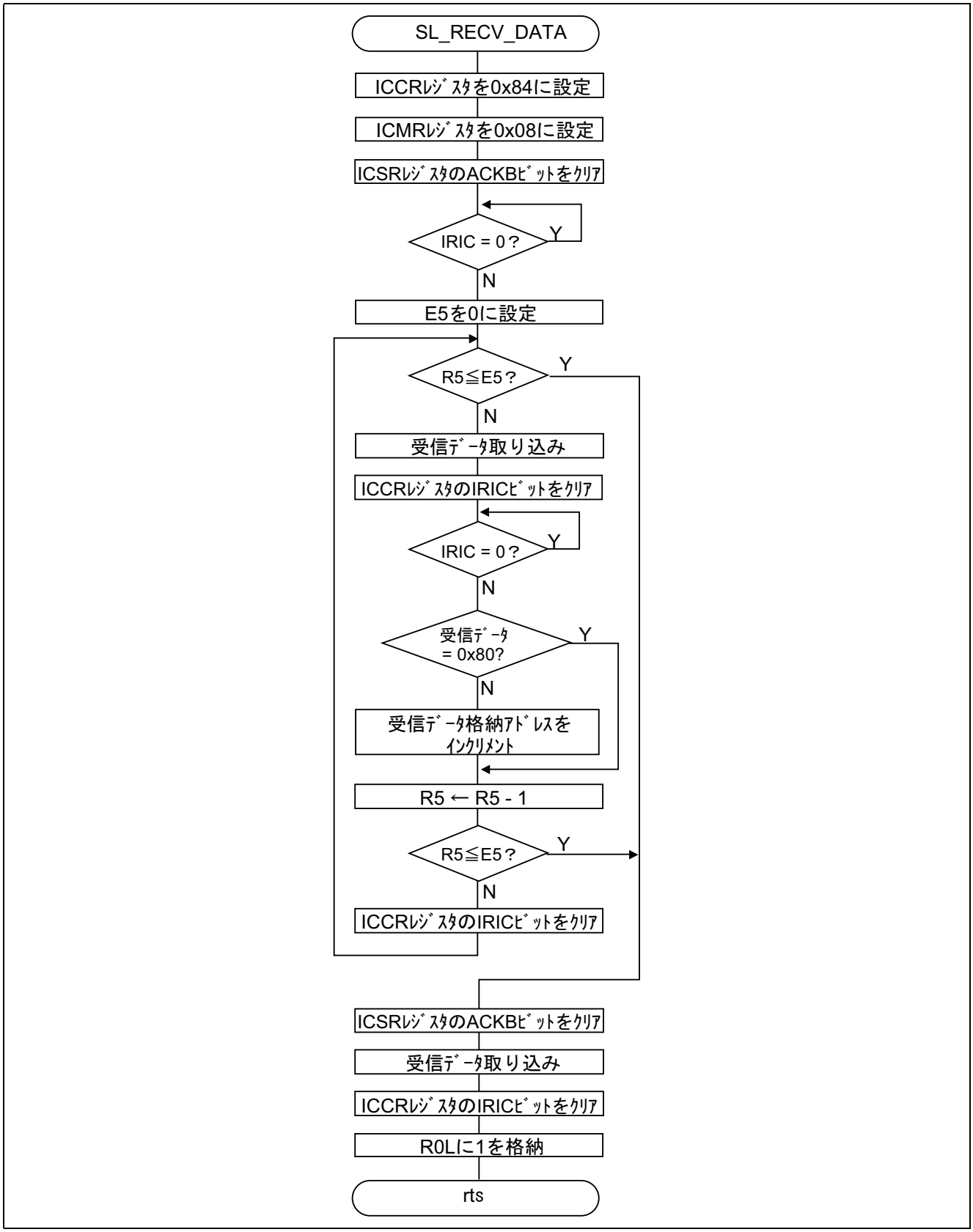
5. フローチャート

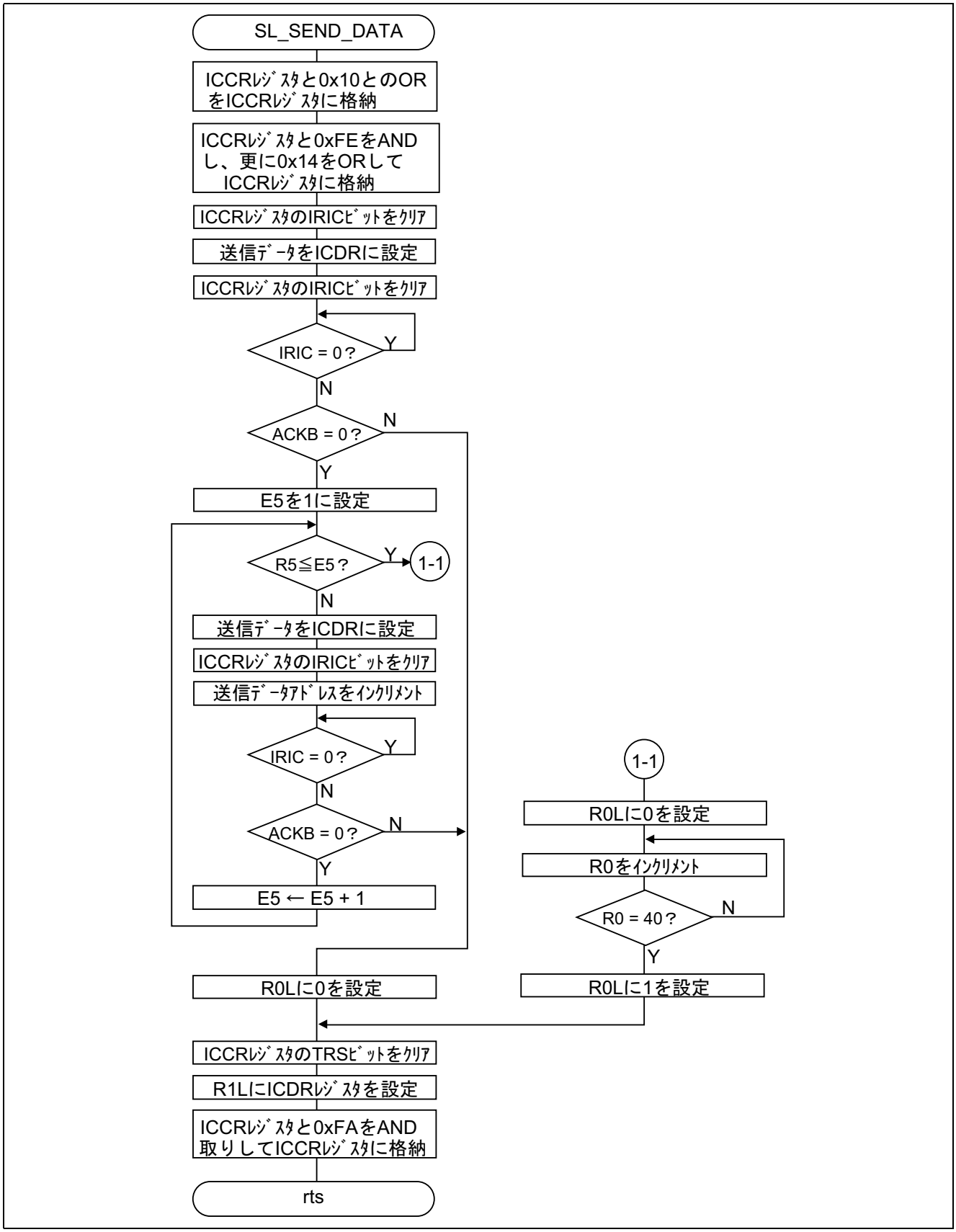

INIT
ｽﾀｯｸﾎﾟｲﾝﾀをH'FF80に設定
Iﾋﾞｯﾄを"1"にｾｯﾄし、
ｼﾘｱﾙ割込み処理を拒否
main JUMP
main
PCR5を0x00に設定
PCR1を0x00に設定
PDR8を0x00に設定
PCR8を0x10に設定
iを0に設定
i < 500？
Y
N
PDR8を0x00に設定
wait (引数10000)
PDR8を0x10に設定
wait (引数10000)
((PDR5 & 0x20) | (PDR1 & 0x10))の結果をswd1に設定
((PDR5 & 0x20) | (PDR1 & 0x10))の結果をswd2に設定
sw_d1 = sw_d2?
N
Y
sw_d1 = 0x10?
N
Y
SL_TRNS
sw_d1 = 0x20?
N
Y
flprg_cpy
jump_prog(引数0xF780)
iをインクリメント
u_main
rts

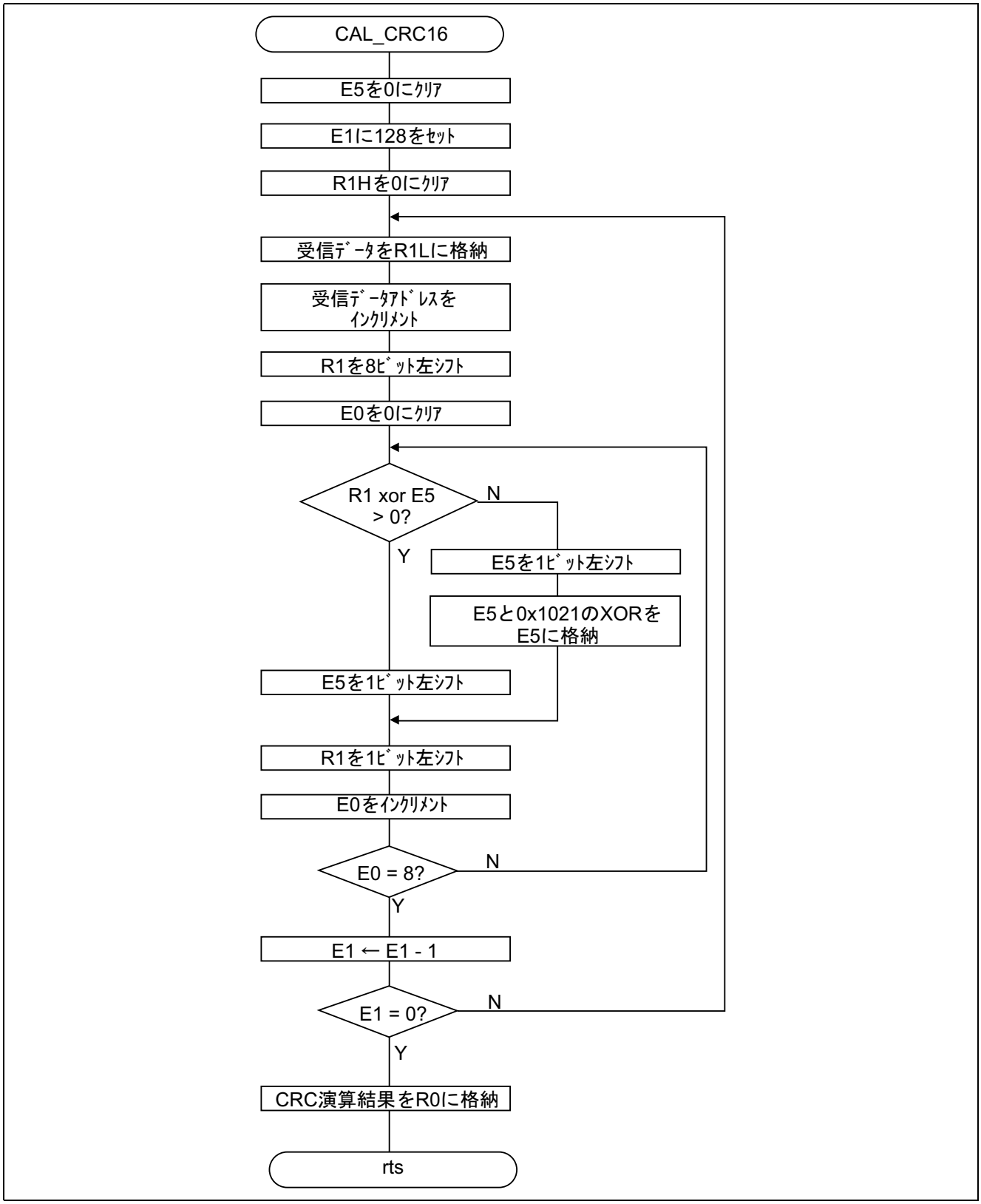

flprg_cpy
ptrに0x04000をｾｯﾄ
r_ptrに0xF780をｾｯﾄ
ptr < 0x0900?
Y
N
*ptrを*r_ptrに書込み
ptrをｲﾝｸﾘﾒﾝﾄ
r_ptrをｲﾝｸﾘﾒﾝﾄ
rts
jump_prog(引数R0)
R0番地へJump
_IIC_TESTの先頭を0x0400にｺﾝﾊﾟｲﾗで設定し、flprg_cpyで
0xF780以降に複製を書き込み、jump_progで複製先を実行する。

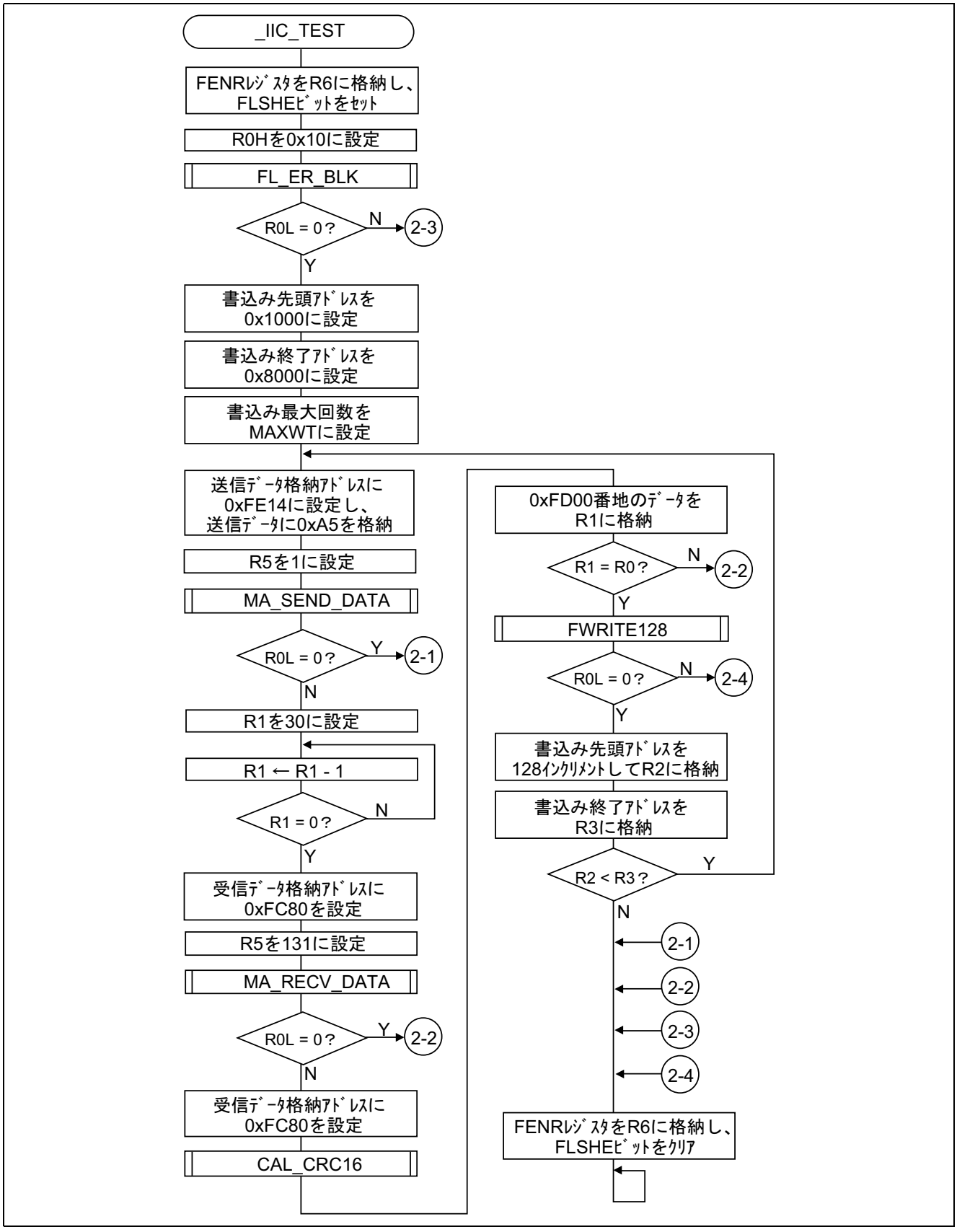

wait (引数limit)
cntに0をｾｯﾄ
cnt < limit?
N
Y
cntをｲﾝｸﾘﾒﾝﾄ
rts

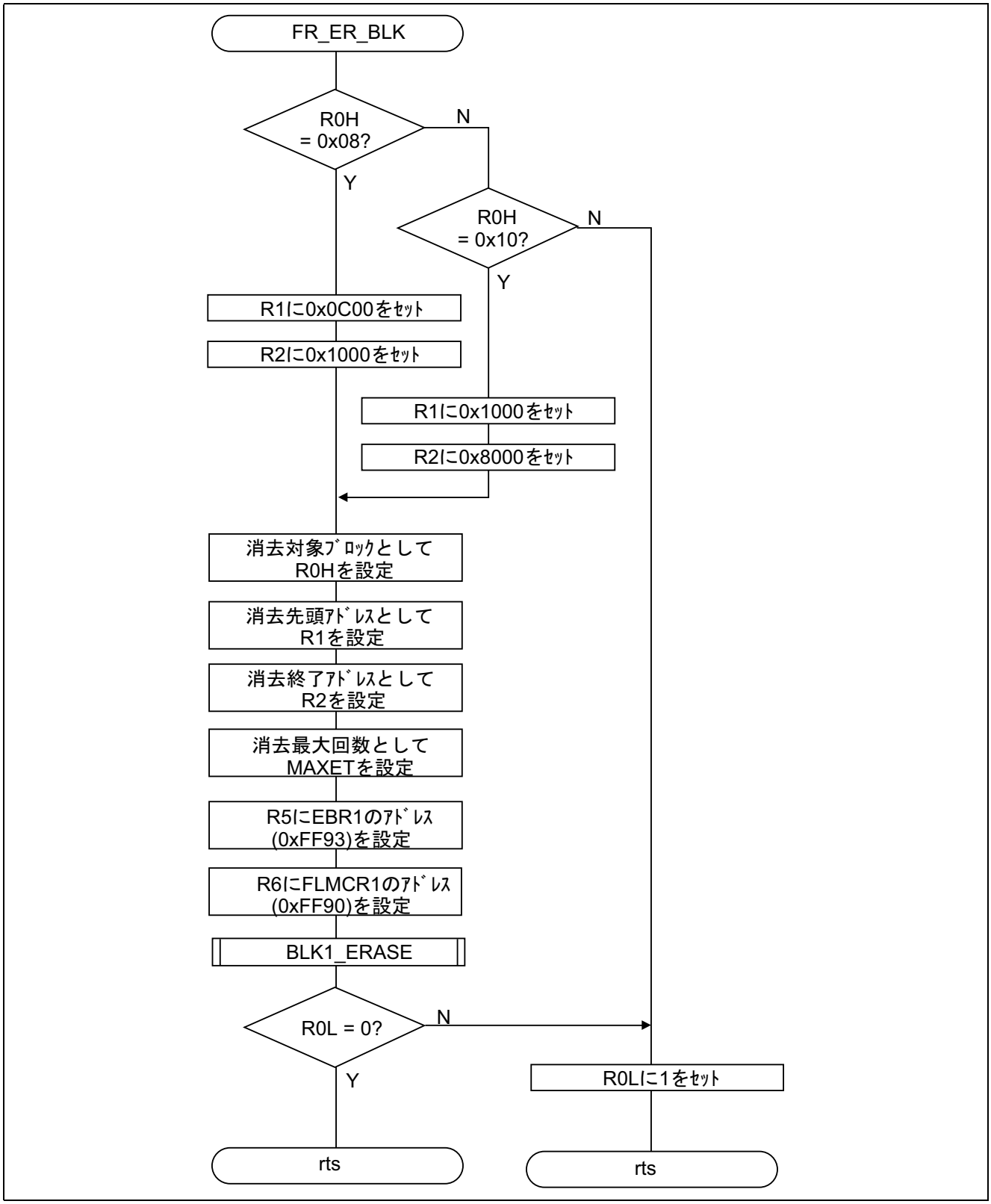

_SL_TRNS
書込み元ｱﾄﾞﾚｽを0X1000に設定
SARﾚｼﾞｽﾀを0x80に設定
TSCRﾚｼﾞｽﾀを0xFCに設定
書込み先ｱﾄﾞﾚｽを0XFC80に設定
書込み元ﾃﾞｰﾀを
書込み先ﾃﾞｰﾀﾊﾞｯﾌｧに格納
書込み元ｱﾄﾞﾚｽを2ｲﾝｸﾘﾒﾝﾄ
書込み先ｱﾄﾞﾚｽを2ｲﾝｸﾘﾒﾝﾄ
書込み先ｱﾄﾞﾚｽ
= 0xFD00?
N
Y
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
0xFC80に設定
CAL_CRC16
CRC演算結果を0xFD00格納
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
0xFD80に設定
R5(受信ﾊﾞｲﾄ数)を1に設定
SL_RECV_DATA
R0L = 0？
Y
N
受信ﾃﾞｰﾀ
= 0xA5?
N
Y
送信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
0xFC80に設定
R5(送信ﾊﾞｲﾄ数)を131に設定
SL_SEND_DATA
R0L = 0？
Y
N
0x8000≦
書込み元ｱﾄﾞﾚｽ?
N
Y

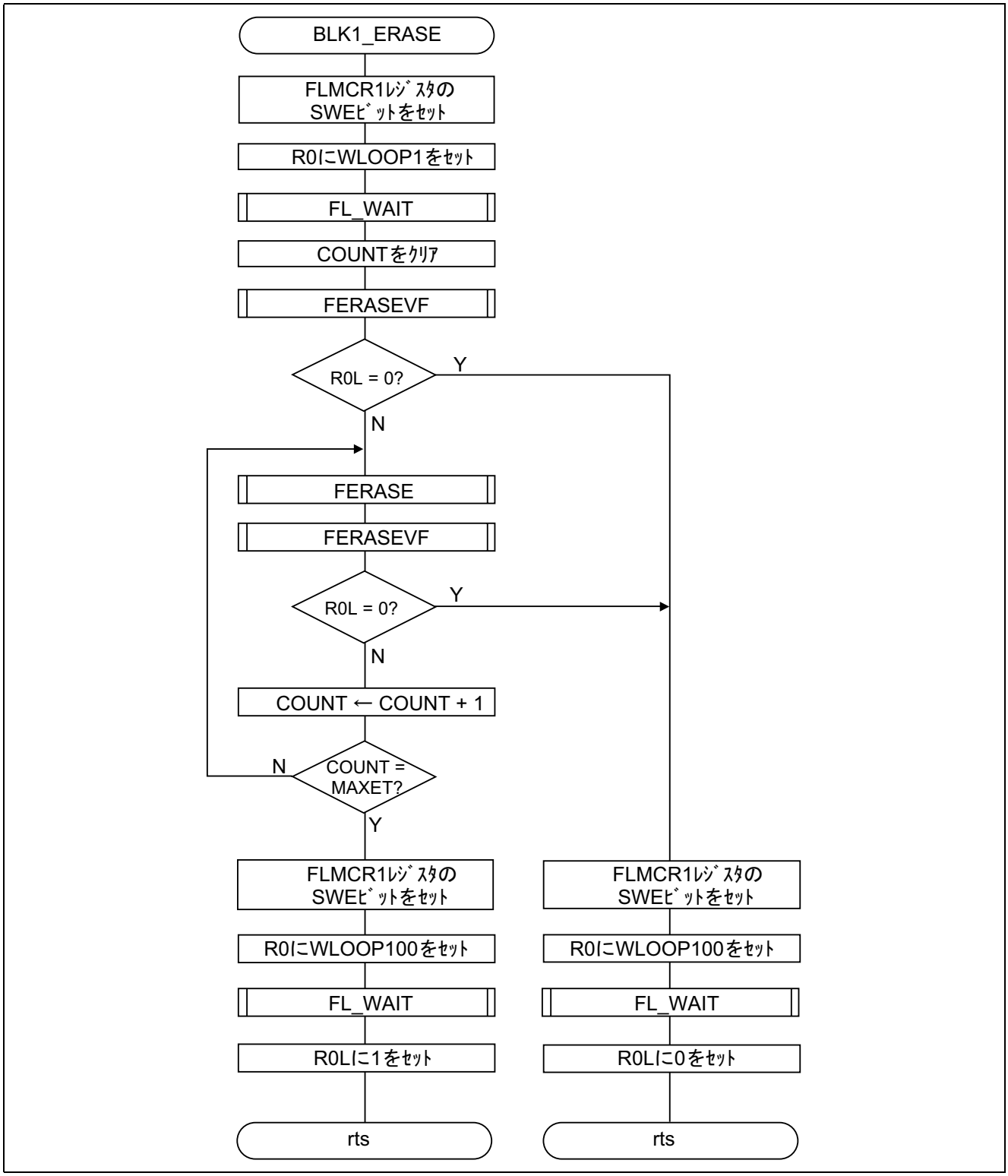

SL_RECV_DATA
ICCRﾚｼﾞｽﾀを0x84に設定
ICMRﾚｼﾞｽﾀを0x08に設定
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
E5を0に設定
R5≦E5？
Y
N
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
受信ﾃﾞｰﾀ
= 0x80?
Y
N
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
R5 ← R5 - 1
R5≦E5？
Y
N
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｸﾘｱ
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
R0Lに1を格納
rts

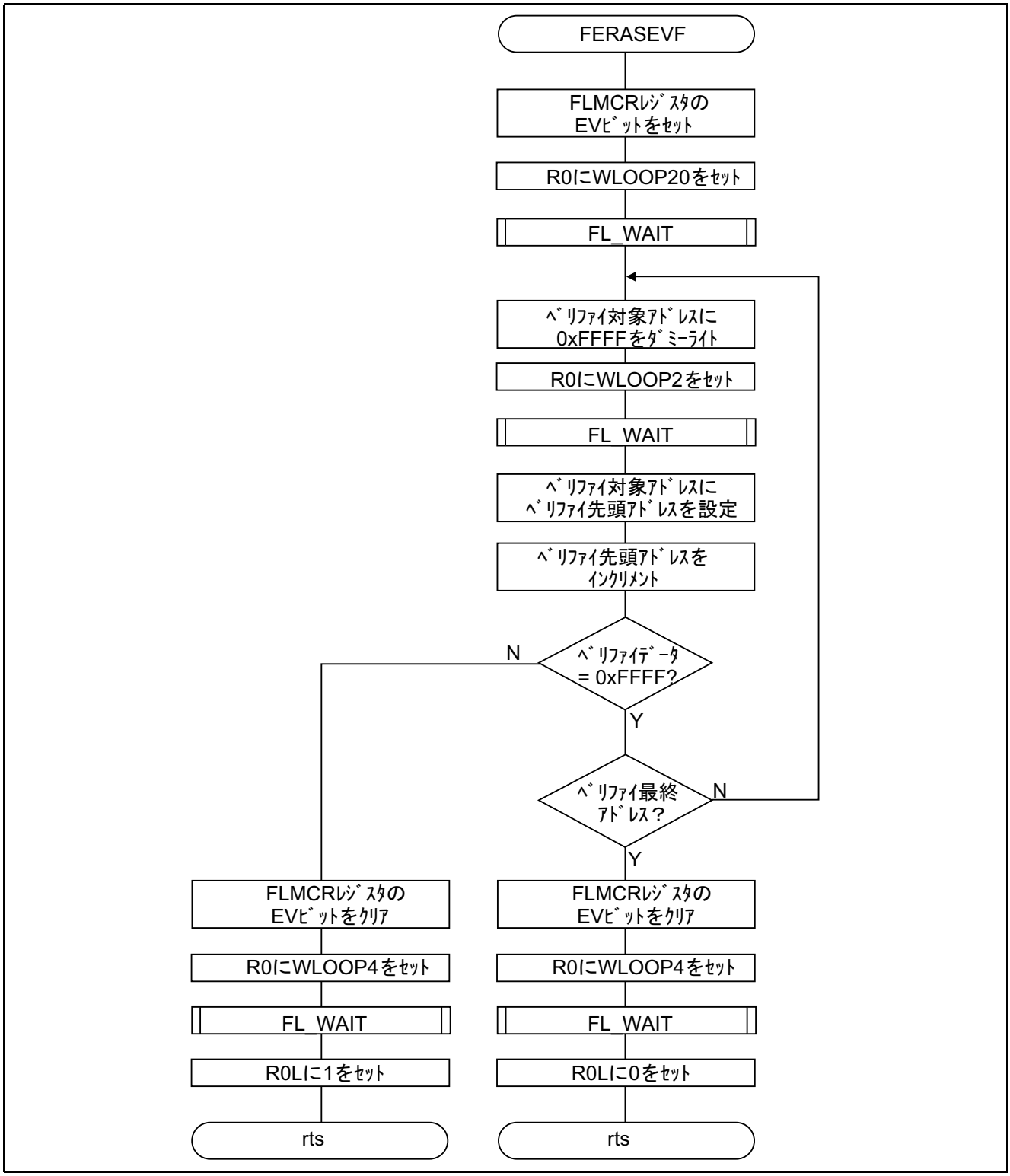

SL_SEND_DATA
ICCRﾚｼﾞｽﾀと0x10とのOR をICCRﾚｼﾞｽﾀに格納
ICCRﾚｼﾞｽﾀと0xFEをAND し、更に0x14をORして ICCRﾚｼﾞｽﾀに格納
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
送信ﾃﾞｰﾀをICDRに設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5を1に設定
R5≦E5？
Y
1-1
N
送信ﾃﾞｰﾀをICDRに設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
送信ﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5 ← E5 + 1
1-1
R0Lに0を設定
R0をｲﾝｸﾘﾒﾝﾄ
R0 = 40？
N
Y
R0Lに1を設定
R0Lに0を設定
ICCRﾚｼﾞｽﾀのTRSﾋﾞｯﾄをｸﾘｱ
R1LにICDRﾚｼﾞｽﾀを設定
ICCRﾚｼﾞｽﾀと0xFAをAND 取りしてICCRﾚｼﾞｽﾀに格納
rts

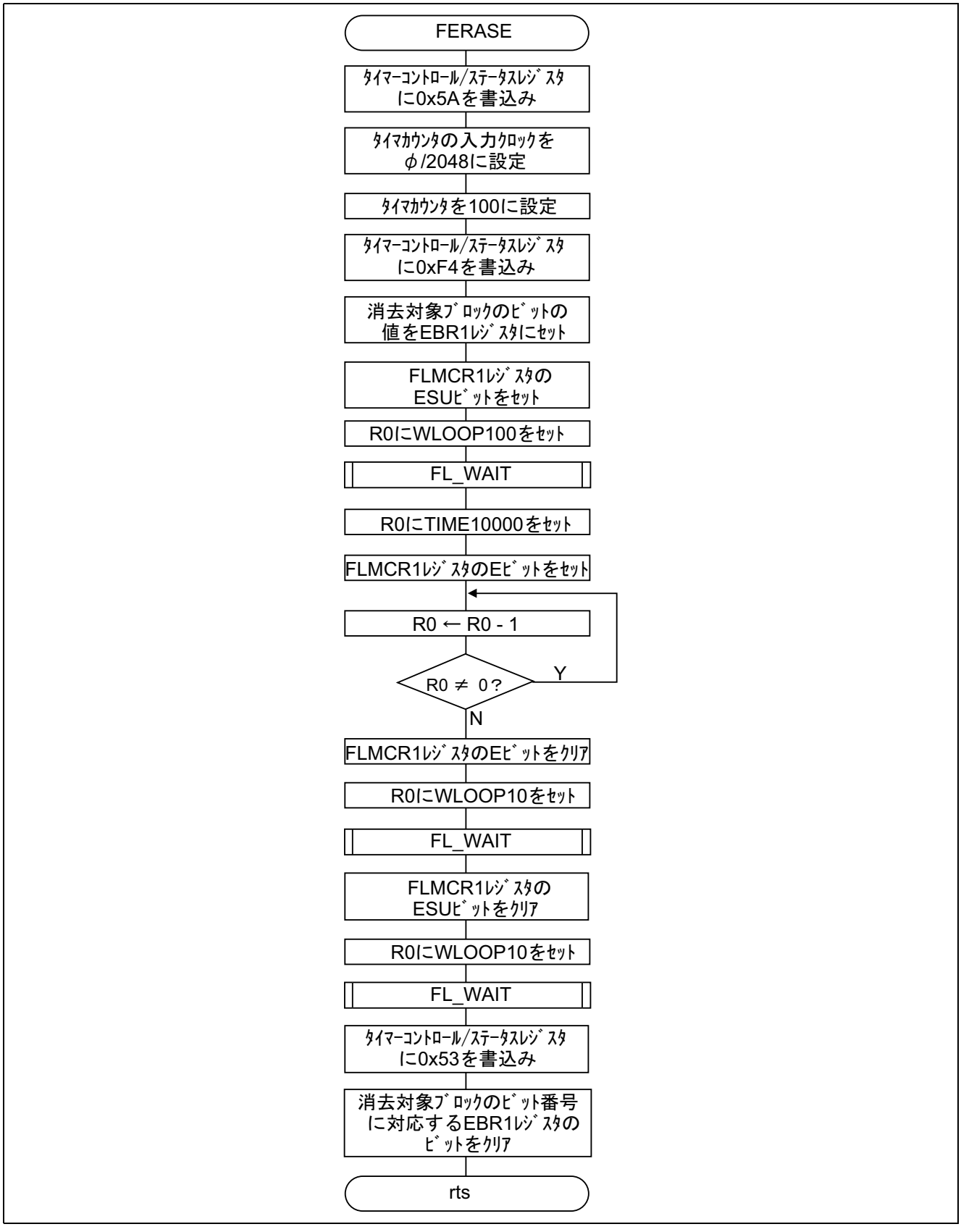

CAL_CRC16
E5を0にｸﾘｱ
E1に128をｾｯﾄ
R1Hを0にｸﾘｱ
受信ﾃﾞｰﾀをR1Lに格納
受信ﾃﾞｰﾀｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
R1を8ﾋﾞｯﾄ左ｼﾌﾄ
E0を0にｸﾘｱ
R1 xor E5
> 0?
N
Y
E5を1ﾋﾞｯﾄ左ｼﾌﾄ
E5と0x1021のXORを
E5に格納
E5を1ﾋﾞｯﾄ左ｼﾌﾄ
R1を1ﾋﾞｯﾄ左ｼﾌﾄ
E0をｲﾝｸﾘﾒﾝﾄ
E0 = 8?
N
Y
E1 ← E1 - 1
E1 = 0?
N
Y
CRC演算結果をR0に格納
rts

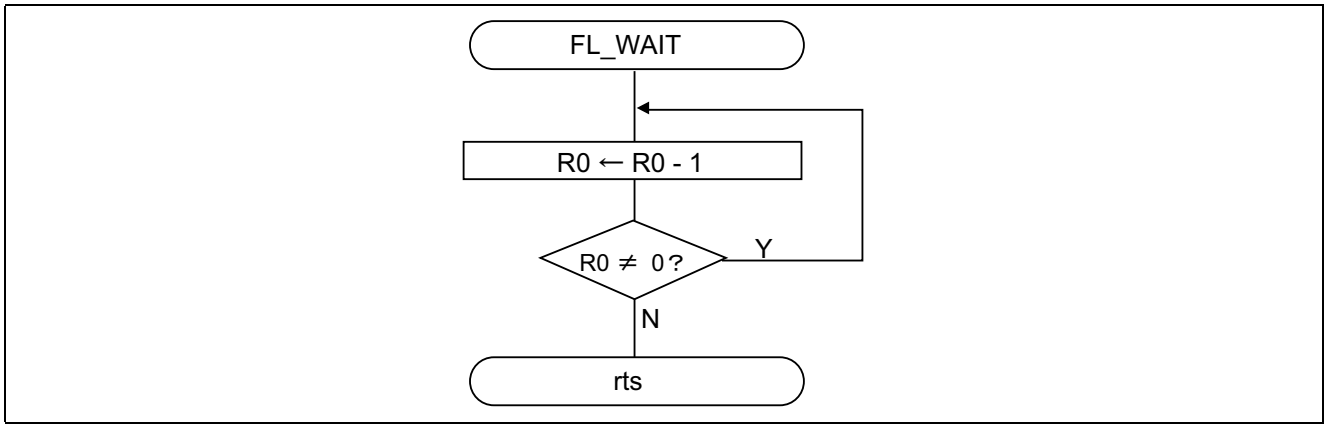

_IIC_TEST
FENRﾚｼﾞｽﾀをR6に格納し、FLSHEﾋﾞｯﾄをｾｯﾄ
R0Hを0x10に設定
FL_ER_BLK
R0L = 0？
N
2-3
Y
書込み先頭ｱﾄﾞﾚｽを0x1000に設定
書込み終了ｱﾄﾞﾚｽを0x8000に設定
書込み最大回数をMAXWTに設定
送信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽに0xFE14に設定し、送信ﾃﾞｰﾀに0xA5を格納
R5を1に設定
MA_SEND_DATA
R0L = 0？
Y
2-1
N
R1を30に設定
R1 ← R1 - 1
R1 = 0？
N
Y
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽに0xFC80を設定
R5を131に設定
MA_RECV_DATA
R0L = 0？
Y
2-2
N
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽに0xFC80を設定
CAL_CRC16
0xFD00番地のﾃﾞｰﾀをR1に格納
R1 = R0？
N
2-2
Y
FWRITE128
R0L = 0？
N
2-4
Y
書込み先頭ｱﾄﾞﾚｽを128ｲﾝｸﾘﾒﾝﾄしてR2に格納
書込み終了ｱﾄﾞﾚｽをR3に格納
R2 < R3？
Y
N
2-1
2-2
2-3
2-4
FENRﾚｼﾞｽﾀをR6に格納し、FLSHEﾋﾞｯﾄをｸﾘｱ

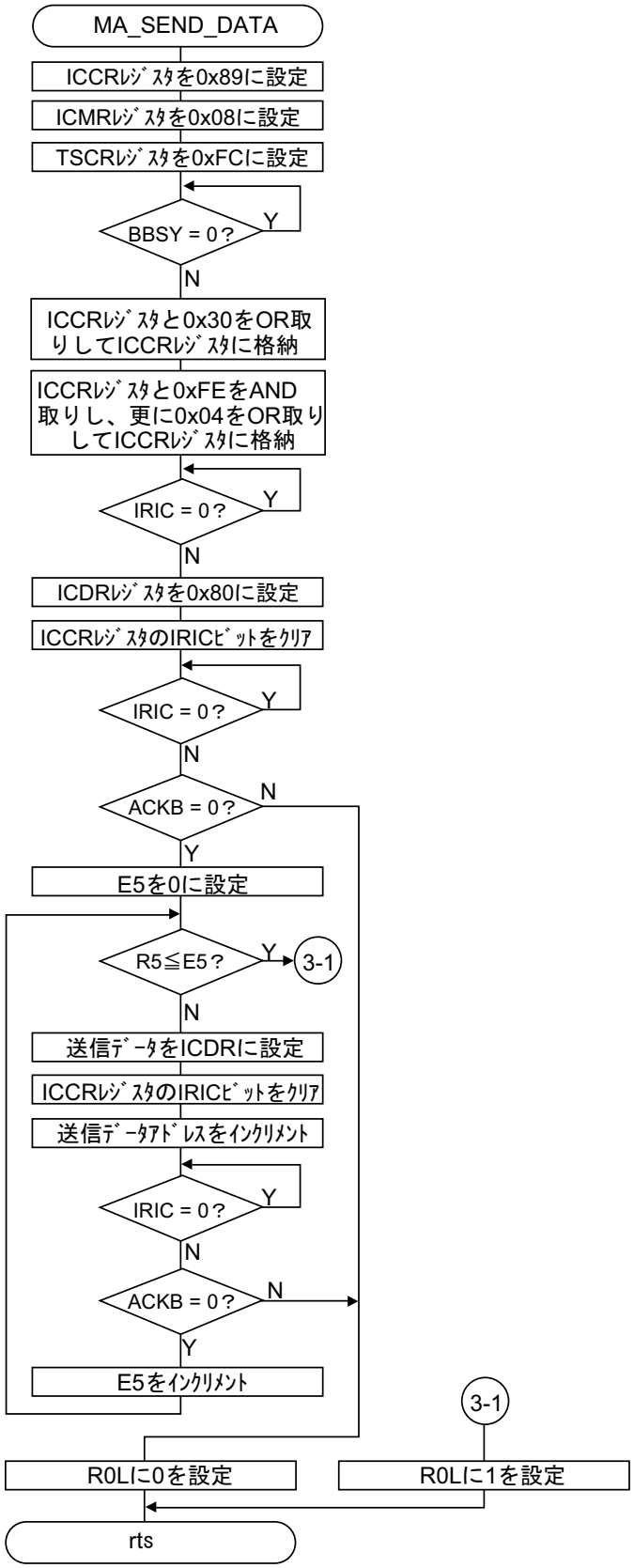

FR_ER_BLK
R0H
= 0x08?
N
Y
R0H
= 0x10?
N
Y
R1に0x0C00をｾｯﾄ
R2に0x1000をｾｯﾄ
R1に0x1000をｾｯﾄ
R2に0x8000をｾｯﾄ
消去対象ﾌﾞﾛｯｸとして
R0Hを設定
消去先頭ｱﾄﾞﾚｽとして
R1を設定
消去終了ｱﾄﾞﾚｽとして
R2を設定
消去最大回数として
MAXETを設定
R5にEBR1のｱﾄﾞﾚｽ
(0xFF93)を設定
R6にFLMCR1のｱﾄﾞﾚｽ
(0xFF90)を設定
BLK1_ERASE
R0L = 0?
N
Y
R0Lに1をｾｯﾄ
rts
rts

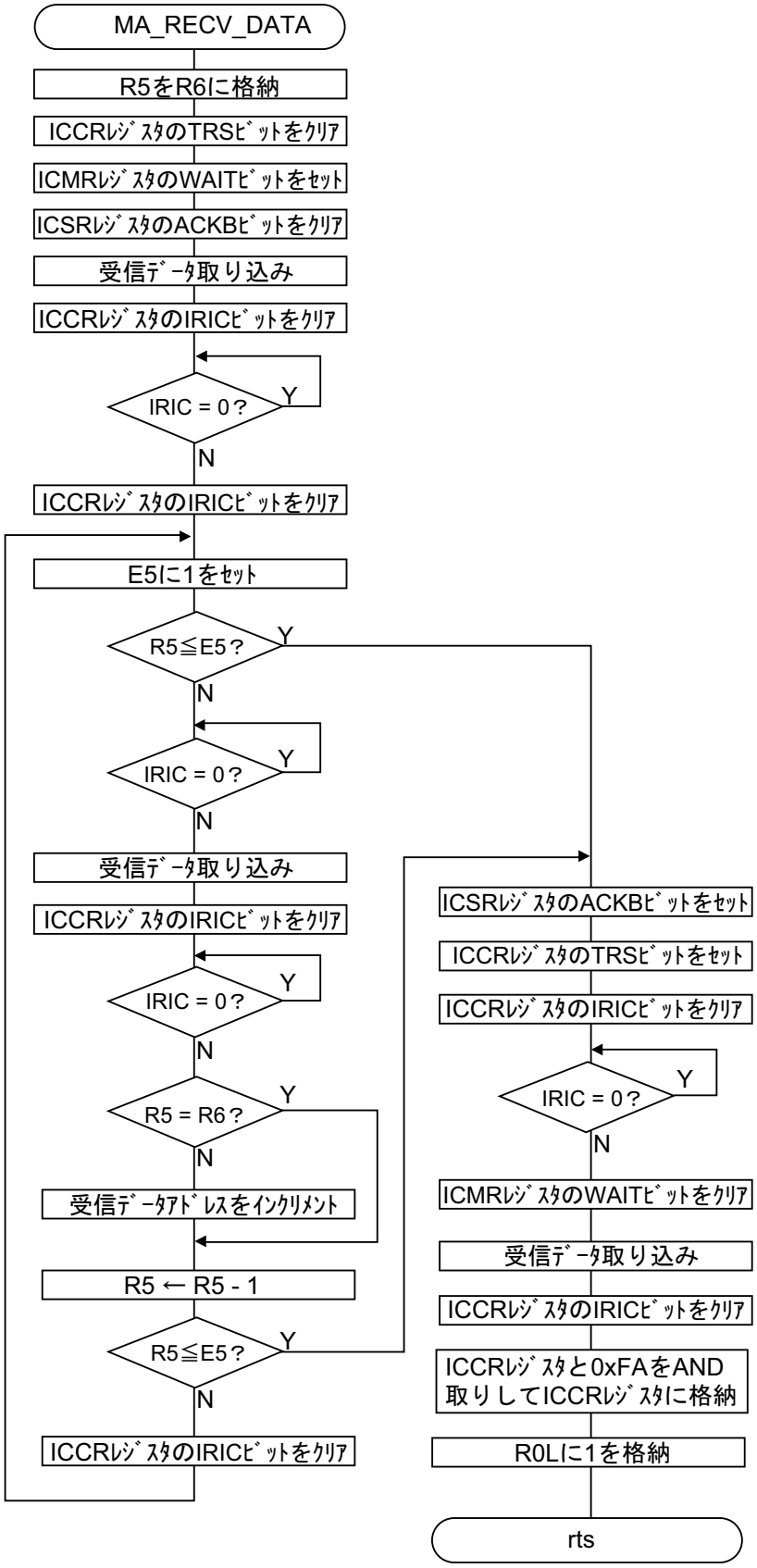

BLK1_ERASE
FLMCR1ﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP1をｾｯﾄ
FL_WAIT
COUNTをｸﾘｱ
FERASEVF
R0L = 0?
Y
N
FERASE
FERASEVF
R0L = 0?
Y
N
COUNT ← COUNT + 1
N
COUNT = MAXET?
Y
FLMCR1ﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
rts
FLMCR1ﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに0をｾｯﾄ
rts

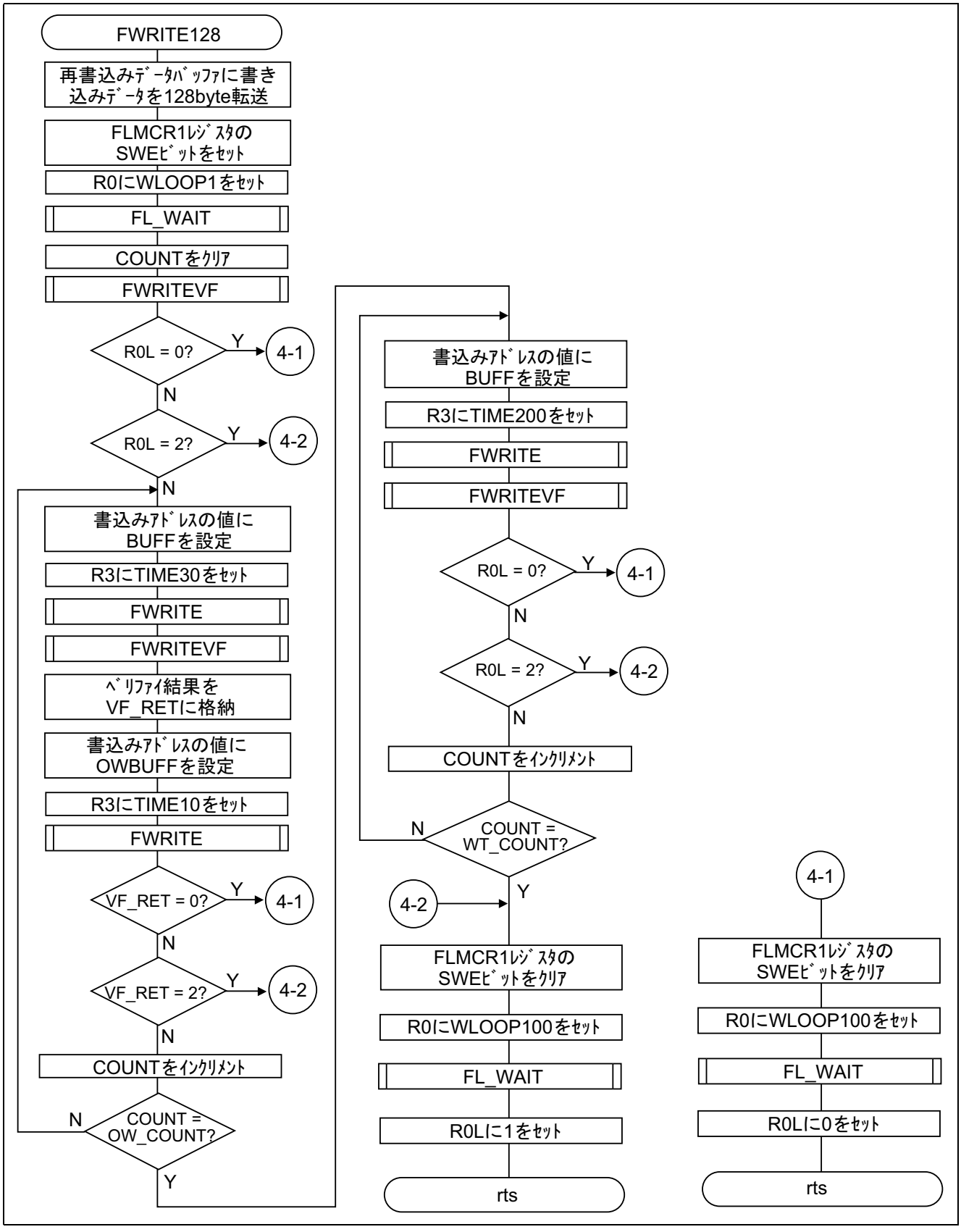

FERASEVF
FLMCRﾚｼﾞｽﾀの
EVﾋﾞｯﾄをｾｯﾄ
R0にWLOOP20をｾｯﾄ
FL_WAIT
ﾍﾞﾘﾌｧｲ対象ｱﾄﾞﾚｽに
0xFFFFをﾀﾞﾐｰﾗｲﾄ
R0にWLOOP2をｾｯﾄ
FL_WAIT
ﾍﾞﾘﾌｧｲ対象ｱﾄﾞﾚｽに
ﾍﾞﾘﾌｧｲ先頭ｱﾄﾞﾚｽを設定
ﾍﾞﾘﾌｧｲ先頭ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
ﾍﾞﾘﾌｧｲﾃﾞｰﾀ
= 0xFFFF?
N
Y
ﾍﾞﾘﾌｧｲ最終
ｱﾄﾞﾚｽ?
N
Y
FLMCRﾚｼﾞｽﾀの
EVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP4をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
rts
FLMCRﾚｼﾞｽﾀの
EVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP4をｾｯﾄ
FL_WAIT
R0Lに0をｾｯﾄ
rts

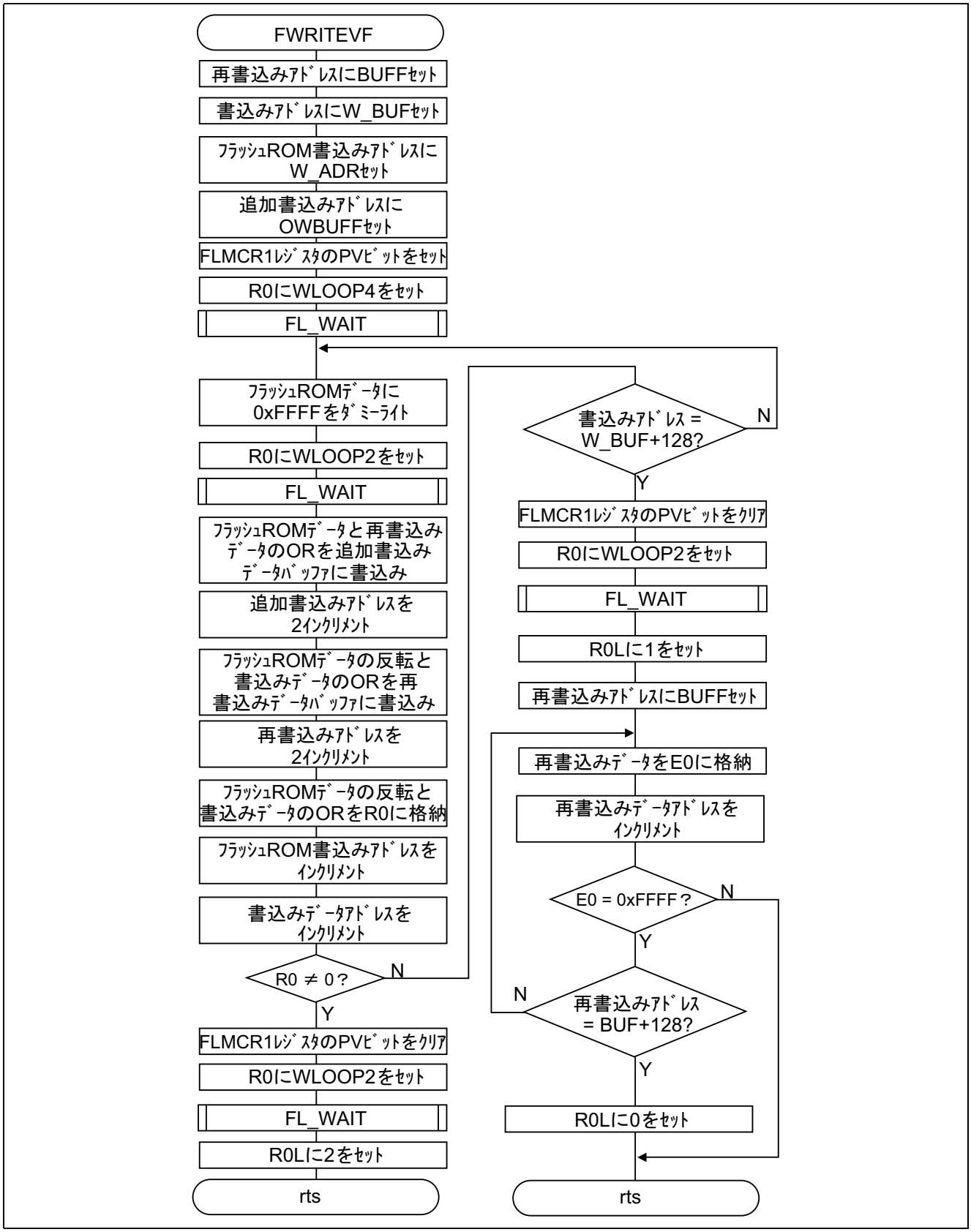

FERASE
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x5Aを書込み
ﾀｲﾏｶｳﾝﾀの入力ｸﾛｯｸを
φ/2048に設定
ﾀｲﾏｶｳﾝﾀを100に設定
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0xF4を書込み
消去対象ﾌﾞﾛｯｸのﾋﾞｯﾄの
値をEBR1ﾚｼﾞｽﾀにｾｯﾄ
FLMCR1ﾚｼﾞｽﾀの
ESUﾋﾞｯﾄをｾｯﾄ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0にTIME10000をｾｯﾄ
FLMCR1ﾚｼﾞｽﾀのEﾋﾞｯﾄをｾｯﾄ
R0 ← R0 - 1
R0 ≠ 0?
Y
N
FLMCR1ﾚｼﾞｽﾀのEﾋﾞｯﾄをｸﾘｱ
R0にWLOOP10をｾｯﾄ
FL_WAIT
FLMCR1ﾚｼﾞｽﾀの
ESUﾋﾞｯﾄをｸﾘｱ
R0にWLOOP10をｾｯﾄ
FL_WAIT
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x53を書込み
消去対象ﾌﾞﾛｯｸのﾋﾞｯﾄ番号
に対応するEBR1ﾚｼﾞｽﾀの
ﾋﾞｯﾄをｸﾘｱ
rts


FL_WAIT
R0 ← R0 - 1
R0 ≠ 0 ?
Y
N
rts

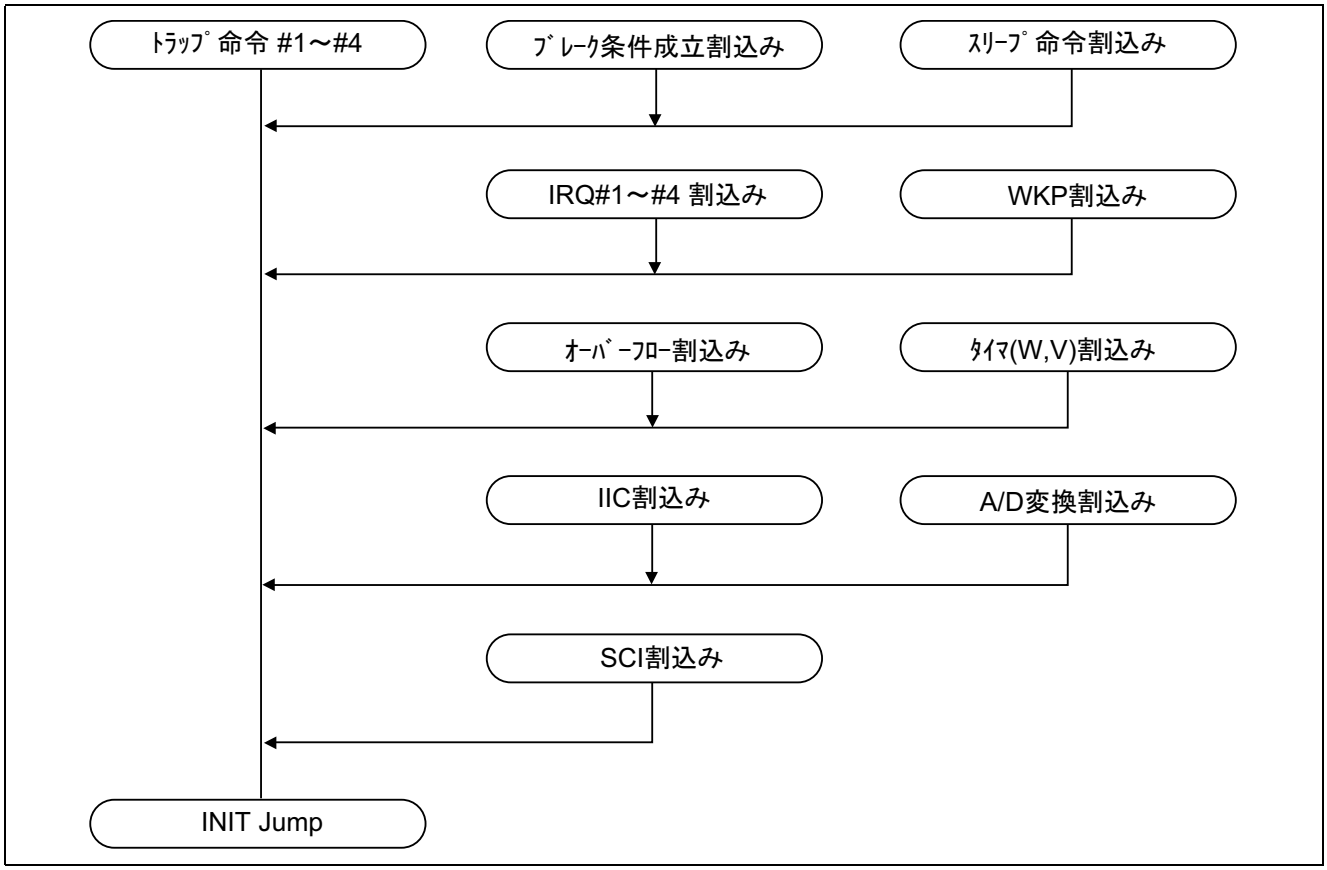

MA_SEND_DATA
ICCRﾚｼﾞｽﾀを0x89に設定
ICMRﾚｼﾞｽﾀを0x08に設定
TSCRﾚｼﾞｽﾀを0xFCに設定
BBSY = 0？
Y
N
ICCRﾚｼﾞｽﾀと0x30をOR取りしてICCRﾚｼﾞｽﾀに格納
ICCRﾚｼﾞｽﾀと0xFEをAND取りし、更に0x04をOR取りしてICCRﾚｼﾞｽﾀに格納
IRIC = 0？
Y
N
ICDRﾚｼﾞｽﾀを0x80に設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5を0に設定
R5≦E5？
Y
3-1
N
送信ﾃﾞｰﾀをICDRに設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
送信ﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5をｲﾝｸﾘﾒﾝﾄ
3-1
R0Lに0を設定
R0Lに1を設定
rts


MA_RECV_DATA
R5をR6に格納
ICCRﾚｼﾞｽﾀのTRSﾋﾞｯﾄをｸﾘｱ
ICMRﾚｼﾞｽﾀのWAITﾋﾞｯﾄをｾｯﾄ
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｸﾘｱ
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
E5に1をｾｯﾄ
R5≦E5？
Y
N
IRIC = 0？
Y
N
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
R5 = R6？
Y
N
受信ﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
R5 ← R5 - 1
R5≦E5？
Y
N
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｾｯﾄ
ICCRﾚｼﾞｽﾀのTRSﾋﾞｯﾄをｾｯﾄ
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ICMRﾚｼﾞｽﾀのWAITﾋﾞｯﾄをｸﾘｱ
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
ICCRﾚｼﾞｽﾀと0xFAをAND
取りしてICCRﾚｼﾞｽﾀに格納
R0Lに1を格納
rts


FWRITE128
再書込みﾃﾞｰﾀﾊﾞｯﾌｧに書き込みﾃﾞｰﾀを128byte転送
FLMCR1ﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP1をｾｯﾄ
FL_WAIT
COUNTをｸﾘｱ
FWRITEVF
R0L = 0?
Y
4-1
N
R0L = 2?
Y
4-2
N
書込みｱﾄﾞﾚｽの値にBUFFを設定
R3にTIME30をｾｯﾄ
FWRITE
FWRITEVF
ﾍﾞﾘﾌｧｲ結果をVF_RETに格納
書込みｱﾄﾞﾚｽの値にOWBUFFを設定
R3にTIME10をｾｯﾄ
FWRITE
VF_RET = 0?
Y
4-1
N
VF_RET = 2?
Y
4-2
N
COUNTをｲﾝｸﾘﾒﾝﾄ
N
COUNT = OW_COUNT?
Y
書込みｱﾄﾞﾚｽの値にBUFFを設定
R3にTIME200をｾｯﾄ
FWRITE
FWRITEVF
R0L = 0?
Y
4-1
N
R0L = 2?
Y
4-2
N
COUNTをｲﾝｸﾘﾒﾝﾄ
N
COUNT = WT_COUNT?
Y
4-2
FLMCR1ﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｸﾘｱ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
rts
4-1
FLMCR1ﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｸﾘｱ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに0をｾｯﾄ
rts


FWRITEVF
再書込みｱﾄﾞﾚｽにBUFFｾｯﾄ
書込みｱﾄﾞﾚｽにW_BUFｾｯﾄ
ﾌﾗｯｼｭROM書込みｱﾄﾞﾚｽにW_ADRｾｯﾄ
追加書込みｱﾄﾞﾚｽにOWBUFFｾｯﾄ
FLMCR1ﾚｼﾞｽﾀのPVﾋﾞｯﾄをｾｯﾄ
R0にWLOOP4をｾｯﾄ
FL_WAIT
ﾌﾗｯｼｭROMﾃﾞｰﾀに0xFFFFをﾀﾞﾐｰﾗｲﾄ
R0にWLOOP2をｾｯﾄ
FL_WAIT
ﾌﾗｯｼｭROMﾃﾞｰﾀと再書込みﾃﾞｰﾀのORを追加書込みﾃﾞｰﾀﾊﾞｯﾌｧに書込み
追加書込みｱﾄﾞﾚｽを2ｲﾝｸﾘﾒﾝﾄ
ﾌﾗｯｼｭROMﾃﾞｰﾀの反転と書込みﾃﾞｰﾀのORを再書込みﾃﾞｰﾀﾊﾞｯﾌｧに書込み
再書込みｱﾄﾞﾚｽを2ｲﾝｸﾘﾒﾝﾄ
ﾌﾗｯｼｭROMﾃﾞｰﾀの反転と書込みﾃﾞｰﾀのORをR0に格納
ﾌﾗｯｼｭROM書込みｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
書込みﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
R0 ≠ 0？
N
Y
FLMCR1ﾚｼﾞｽﾀのPVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP2をｾｯﾄ
FL_WAIT
R0Lに2をｾｯﾄ
rts
書込みｱﾄﾞﾚｽ = W_BUF+128?
N
Y
FLMCR1ﾚｼﾞｽﾀのPVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP2をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
再書込みｱﾄﾞﾚｽにBUFFｾｯﾄ
再書込みﾃﾞｰﾀをE0に格納
再書込みﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
E0 = 0xFFFF？
N
Y
再書込みｱﾄﾞﾚｽ = BUF+128?
N
Y
R0Lに0をｾｯﾄ
rts


FWRITE
E0に128をｾｯﾄ
書込み元ｱﾄﾞﾚｽのﾃﾞｰﾀを
書込み先ｱﾄﾞﾚｽに書込み
書込み元ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
書込み先ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
E0← E0 - 1
E0 ≠ 0？
Y
N
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x5Aを書込み
ﾀｲﾏｶｳﾝﾀの入力ｸﾛｯｸを
φ/64に設定
ﾀｲﾏｰｶｳﾝﾀを166に設定
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0xF4を書込み
FLMCR1ﾚｼﾞｽﾀの
PSUﾋﾞｯﾄをｸﾘｱ
R0にWLOOP50をｾｯﾄ
FL_WAIT
FLMCR1ﾚｼﾞｽﾀのPﾋﾞｯﾄをｸﾘｱ
R3← R3 - 1
R0 ≠ 0？
Y
N
FLMCR1ﾚｼﾞｽﾀのPﾋﾞｯﾄをｸﾘｱ
R0にWLOOP5をｾｯﾄ
FL_WAIT
FLMCR1ﾚｼﾞｽﾀの
PSUﾋﾞｯﾄをｸﾘｱ
R0にWLOOP5をｾｯﾄ
FL_WAIT
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x53を書込み
rts


ﾄﾗｯﾌﾟ命令 #1～#4
ﾌﾞﾚｰｸ条件成立割込み
ｽﾘｰﾌﾟ命令割込み
IRQ#1～#4 割込み
WKP割込み
ｵｰﾊﾞｰﾌﾛｰ割込み
ﾀｲﾏ(W,V)割込み
IIC割込み
A/D変換割込み
SCI割込み
INIT Jump

6. モジュール説明

本タスク例におけるモジュール説明（引数・リターン値）を表 10 に示します。

表 10 モジュール説明表

モジュール（関数）名	引数	リターン値	機能
INIT(アセンブリ言語)	なし	なし	スタックポインタの設定(R7 に H'FF80 をセット) CCR の設定（割込み禁止） main へジャンプする
main	なし	なし	メインモジュール
flprg_cpy	なし	なし	0x0400～0x08FF のデータを 0xF780～0xFC7F にコピーする
jump_prog (アセンブリ言語)	R0 (Jump 先の番地)	なし	R0 番地へ Jump する
wait	limit（ウェイト）	なし	ウエイト
_SL_TRANS (アセンブリ言語)	なし	なし	スレーブモード送受信
SL_RECV_DATA (アセンブリ言語)	R4（受信データ格納アドレス）、R5（受信バイト数）	R0L（受信結果）	スレーブモード受信処理
SL_SEND_DATA (アセンブリ言語)	R4（送信データ格納アドレス）、R5（送信バイト数）	R0L（送信結果）	スレーブモード送信処理
CAL_CRC16 (アセンブリ言語)	R4（受信データ格納アドレス）	R0（CRC 演算結果）	CRC 演算処理
_IIC_TEST (アセンブリ言語)	なし	なし	フラッシュ消去／書き込み処理
FL_ER_BLK (アセンブリ言語)	R0H（イレースブロック指定）	R0L（消去結果）	Flash データ消去処理
BLK1_ERASE (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）、ER5（EBR レジスタのアドレス）	R0L（消去結果）	フラッシュ指定ブロック消去処理
FERASEVF (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）	R0L（ベリファイ結果）	フラッシュ消去ベリファイ処理
FERASE (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）、ER5（EBR レジスタのアドレス）	なし	フラッシュ指定ブロック消去処理
FL_WAIT (アセンブリ言語)	R0（ウエイト）	なし	ウエイト
MA_SEND_DATA (アセンブリ言語)	R4（送信データ格納アドレス）、R5（送信バイト数）	R0L（送信結果）	マスターモード送信処理
MA_RECV_DATA (アセンブリ言語)	R4（受信データ格納アドレス）、R5（受信バイト数）	R0L（受信結果）	マスターモード受信処理
FWRITE128 (アセンブリ言語)	なし	R0L（書き込み結果）	フラッシュ任意の 128Byte 書き込み処理
FWRITEVF (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）	R0L（ベリファイ結果）	フラッシュ書き込みベリファイ処理
FWRITE (アセンブリ言語)	ER6（FLMCR レジスタのアドレス）、ER2（書き込みの先頭アドレス）、ER3（P ビットセット時間）	なし	フラッシュ書き込み処理

アセンブリ言語で書かれたモジュールを C プログラムで参照する時は、先頭のアンダーバー（_）を除いた名前で参照します。例えばアセンブリ言語で書かれたモジュール_SL_TRNS を C プログラムで参照する場合は SL_TRNS となります。

本タスク例における定数の説明を表 11 に示します。

表 11　使用定数説明

定義名	値	用途
WLOOP1	1*MHZ/400(=2)	**ウエイト文実行回数（ウエイト時間：1μs）**
WLOOP2	2*MHZ/400(=5)	**ウエイト文実行回数（ウエイト時間：2μs）**
WLOOP4	4*MHZ/400(=11)	**ウエイト文実行回数（ウエイト時間：4μs）**
WLOOP5	5*MHZ/400(=13)	**ウエイト文実行回数（ウエイト時間：5μs）**
WLOOP10	10*MHZ/400(=27)	**ウエイト文実行回数（ウエイト時間：10μs）**
WLOOP20	20*MHZ/400(=55)	**ウエイト文実行回数（ウエイト時間：20μs）**
WLOOP50	50*MHZ/400(=137)	**ウエイト文実行回数（ウエイト時間：50μs）**
WLOOP100	100*MHZ/400(=275)	**ウエイト文実行回数（ウエイト時間：100μs）**
TIME	10*MHZ/400(=27)	**ウエイト文実行回数（ウエイト時間：10μs）**
TIME	30*MHZ/400(=82)	**ウエイト文実行回数（ウエイト時間：20μs）**
TIME	200*MHZ/400(=550)	**ウエイト文実行回数（ウエイト時間：200μs）**
TIME	10000*MHZ/400(=27500)	**ウエイト文実行回数（ウエイト時間：10ms）**
MAXWT	1000	**フラッシュ書き込み最大回数**
MAXET	100	**消去最大回数**
OW_COUNT	6	**再書き込み回数**

本タスク例における使用 RAM の説明を表 12 に示します。

表 12　使用 RAM 説明

ラベル名	機能	アドレス	使用モジュール名
W_BUF	**書き込み用データバッファ**（128Byte）	H'FC80	_IIC_TEST、_SL_TRNS、FWRITE128、FWRITEVF
BUFF	**再書き込み用データバッファ**（128Byte）	H'FD00	FWRITE128、FWRITEVF
OWBUFF	**追加書き込み用データバッファ**（128Byte）	H'FD80	_SL_TRNS、FWRITE128、FWRITEVF
COUNT	**書き込み回数／消去回数**	H'FE00	FWRITE128、BLK_ERASE
W_ADR	**書き込み先頭アドレス**	H'FE02	_IIC_TEST、FWRITEVF、FWRITE
W_ADR_ED	**書き込み終了アドレス**	H'FE04	_IIC_TEST
ET_COUNT	**消去最大回数**	H'FE08	FL_ER_BLK、BLK1_ERASE
WT_COUNT	**書き込み最大回数**	H'FE0A	FWRITE128、_IIC_TEST
EVF_ST	**消去先頭アドレス**	H'FE0C	FL_ER_BLK、FERASEVF
EVF_ED	**消去終了アドレス**	H'FE0E	FL_ER_BLK、FERASEVF
BLK_NO	**消去対象ブロック**	H'FE10	FL_ER_BLK、FERASE
VF_RET	**書き込みベリファイ結果**	H'FE11	FWRITE128
IIC_SBUF	**送信データ格納アドレス**	H'FE14	_IIC_TEST

本タスク例で使用する内部レジスタ説明を表.13 に示します。

表.13　使用内部レジスタ説明

レジスタ		機能	操作	設定値
ICDR		送信データ・受信データを格納	格納・参照	—
ICMR	MLS	MSB ファーストによるデータ転送の設定	設定	0
	WAIT	データとアクノリッジの連続的な転送を設定	設定	0
	CKS2 to CKS0	STCR の IICX ビットと組み合わせて、転送クロックの周波数を 400kHz に設定	設定	CKS2＝0 CKS1＝0 CKS0＝1
	BC2 to BC0	I²C バスフォーマットで次に転送するデータのビット数を 9 ビット／フレームに設定	設定	BC2＝0 BC1＝0 BC0＝0
ICCR	ICE	ICMR、ICDR/SAR、SARX レジスタのアクセス制御、I²C バスインターフェースの動作（SCL/SDA 端子はポート機能）/非動作（SCL/SDA 端子はバス駆動機能）の選択	設定	0/1
	IEIC	I²C バスインターフェース割り込み要求を禁止	設定	0/1
	MST	I²C バスインターフェースをマスタモードで使用	設定	0/1
	TRS	I²C バスインターフェースを送信モードで使用	設定	0/1
	ACKE	アクノリッジビットが “ 1 ” の場合、連続的な転送を中断	設定	0/1
	BBSY	I²C バスが占有されているか解放されているかの確認、および SCP ビットと組み合わせて開始条件、停止条件を発行	設定・参照	0/1
	IRIC	開始条件の検出、データ送信の終了判定、アクノリッジ＝ “ 1 ” の検出	設定	0/1
	SCP	BBSY ビットと組み合わせて開始条件、停止条件を発行	設定	0/1
ICSR	ESTP	エラー停止条件検出フラグ（スレーブモード有効）	未操作	—
	STOP	正常停止条件検出フラグ（スレーブモード有効）	未操作	—
	IRTR	連続送受信割り込み要求フラグ	未操作	—
	AASX	第 2 スレーブアドレス認識フラグ	未操作	—
	AL	アービトレーションロストフラグ	未操作	—
	AAS	スレーブアドレス認識フラグ	未操作	—
	ADZ	ゼネラルコールアドレス認識フラグ	未操作	—
	ACKB	EEPROM より送信されたアクノリッジデータを格納	未操作	—
TSCR	IICRST	IIC コントロール部リセット	参照	—
	IICX	転送レート選択	設定	0
FLMCR1	SWE	SWE=1 の時フラッシュの書き込み／消去可能	設定	0
	ESU	ESU=1 の時イレースセットアップモード、ESU=0 で解除	設定	0/1
	PSU	PSU=1 の時プログラムセットアップモード、PSU=0 で解除	設定	0/1
	EV	EV=1 の時イレースベリファイモード、EV=0 で解除	設定	0/1
	PV	PV=1 の時プログラムベリファイモード、PV=0 で解除	設定	0/1
	E	SWE=1、ESU=1 で E=1 の時イレースモード、E=0 で解除	設定	0/1
	P	SWE=1、PSU=1 で P=1 の時プログラムモード、P=0 で解除	設定	0/1
EBR1	EB4〜0	フラッシュメモリの対象ブロックを H’1000〜H’7FFF までの 28K Byte に設定	設定	EB4〜0 ＝H’10
FENR	FLSHE	FLMCR1、EBR1 レジスタを有効にする	設定	0/1

レジスタ		機能	操作	設定値
TCSRWD	B6WI	B6WI＝0 で書き込みを行った時のみ、TCWE の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCWE	TCWE＝1 の時、TCWD レジスタへの書き込み有効	設定	1
	B4WI	B4WI＝0 で書き込みを行った時のみ、TCSRWE の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCSRWE	TCSRWE＝1 の時、WDON、WRST ビットへの書き込み有効	設定	1
	B2WI	B2WI＝0 で書き込みを行った時のみ、WDON の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCWE	WDON＝1 の時 TCWD をカウントアップ、WDON＝0 の時停止	設定	0/1
	B0WI	B0WI＝0 で書き込みを行った時のみ、WRST の書き込み有効。リード時は 1 固定出力	設定	0/1
	TCSRWE	ウォッチドックタイマリセット	設定	1
TMWD	CKS3～0	TCWD に入力するクロックを選択する。 CKS3～0＝H'8：内部クロック（Φ）/64 CKS3～0＝H'D：内部クロック（Φ）/2048	設定	CKS3～0＝H'8 or H'D
TCWD		リード／ライト可能の 8bit 幅タイマカウンタ	設定	166 or 100

7. モジュール階層図

モジュール階層図を図 6 に示します。

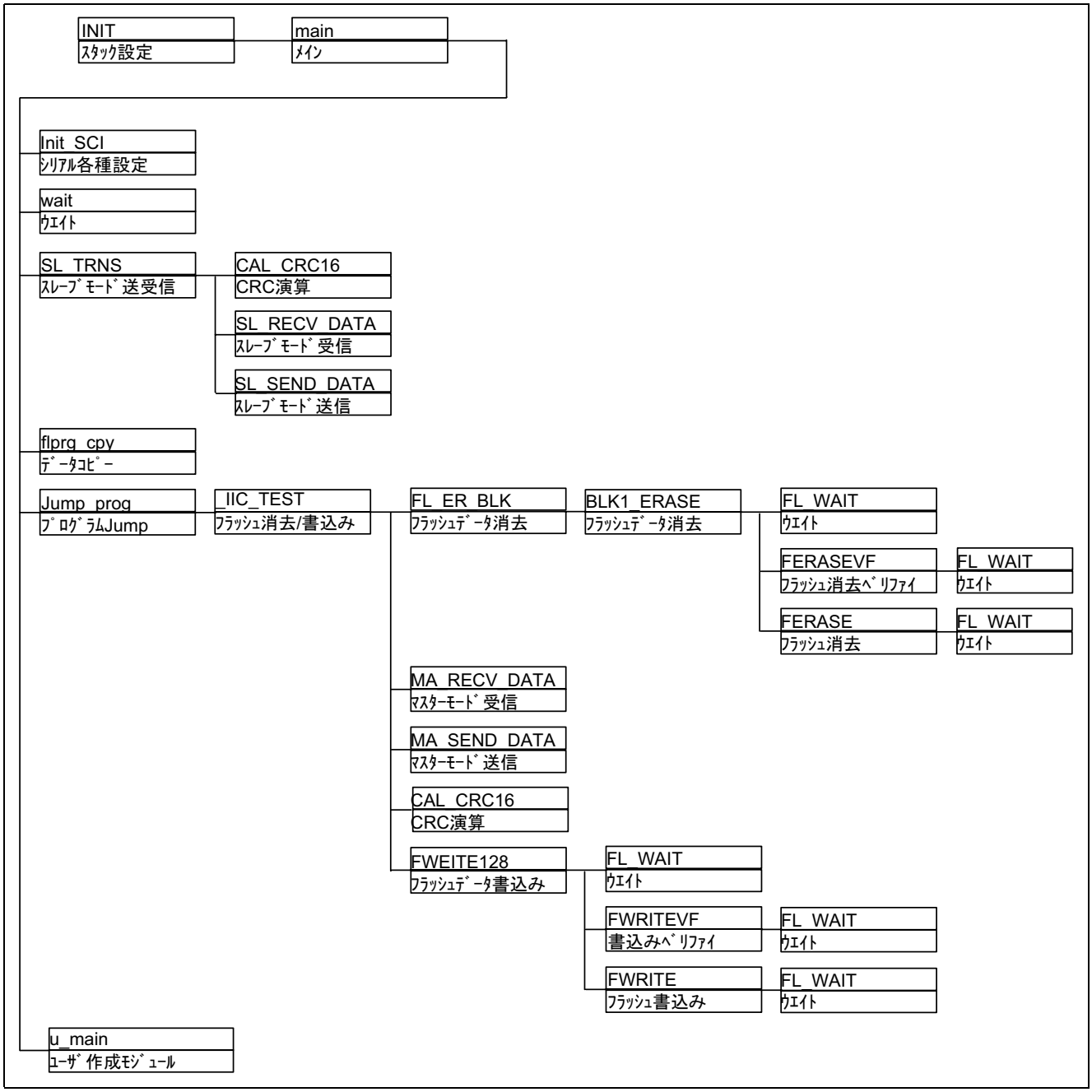
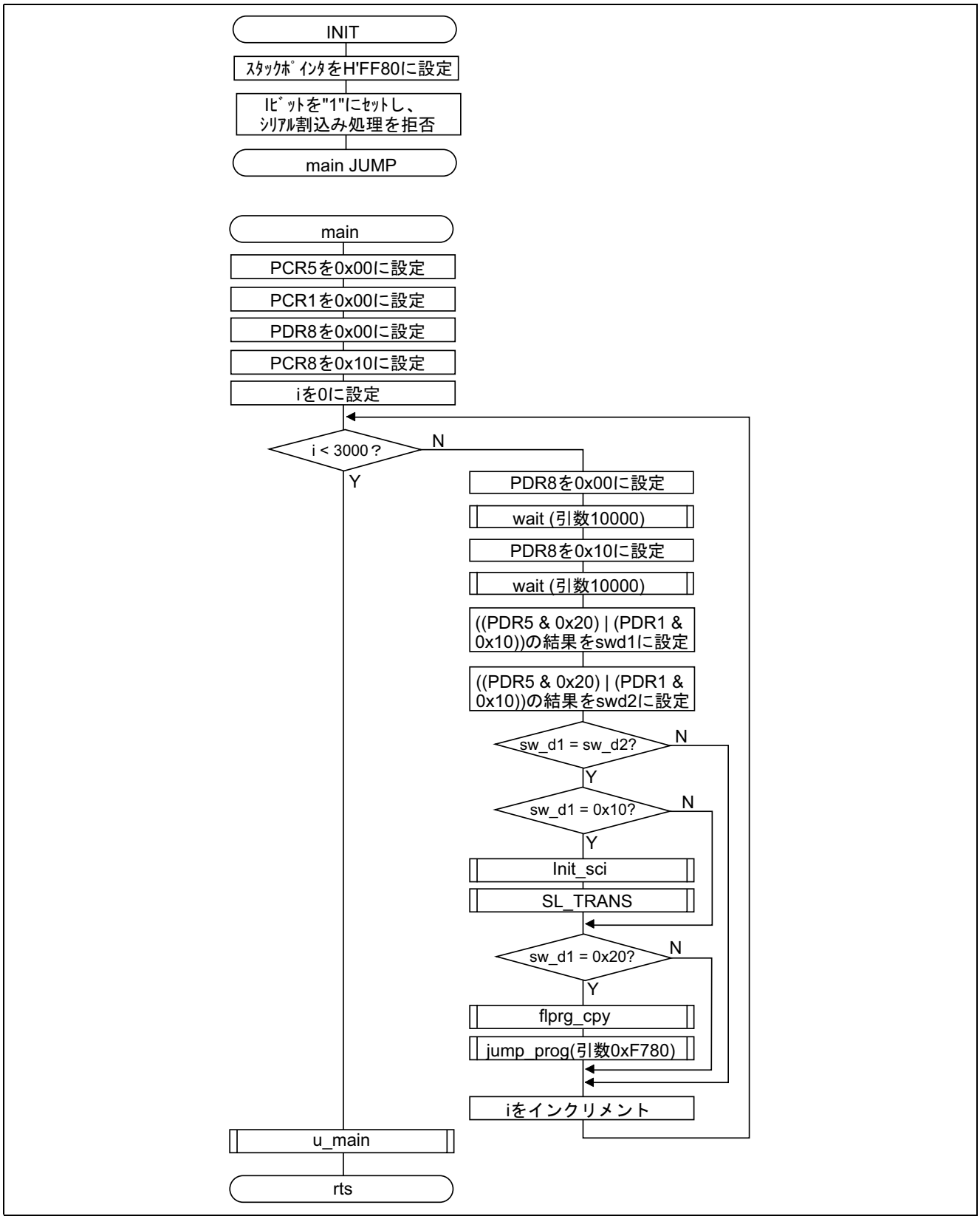
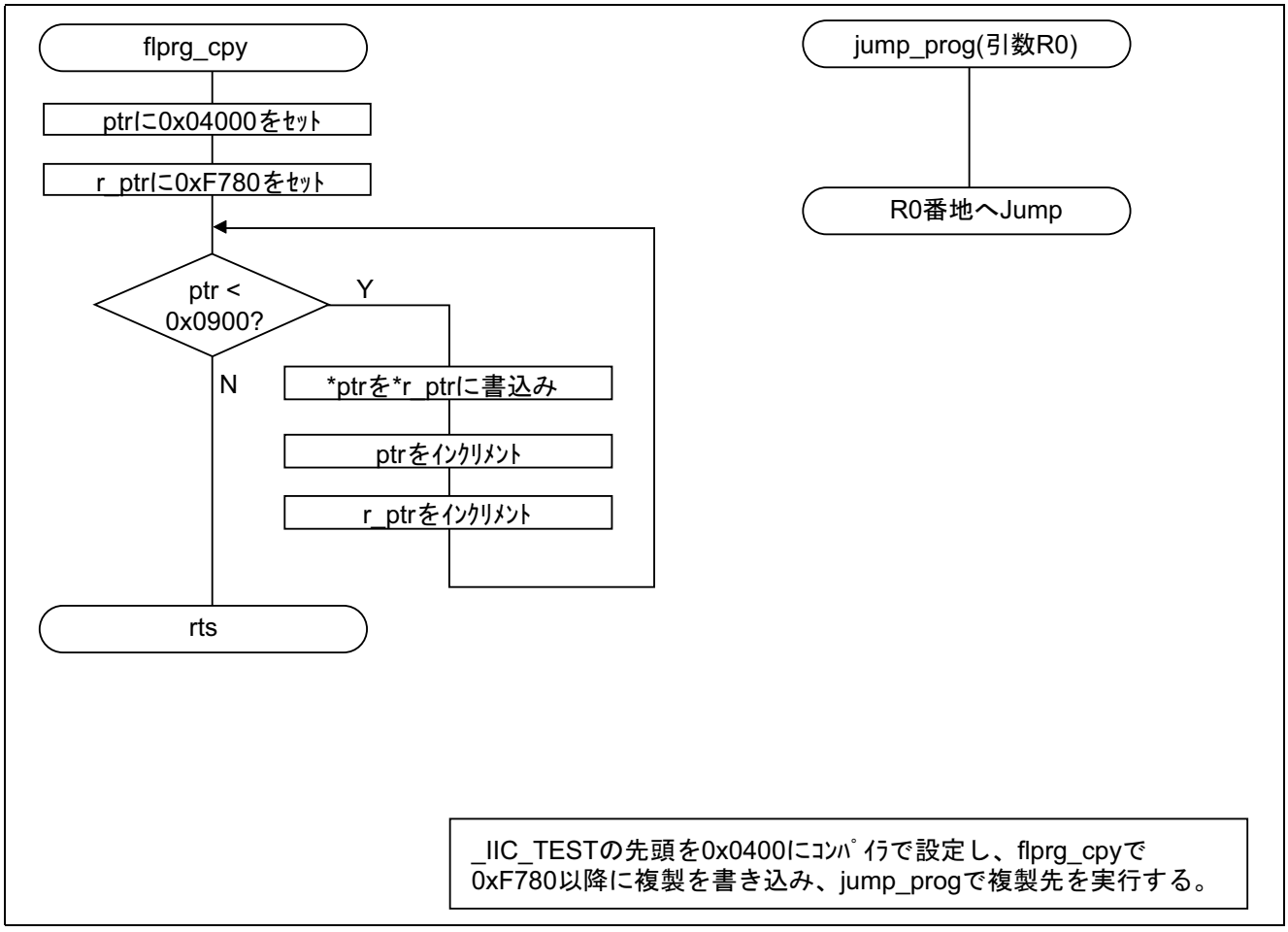
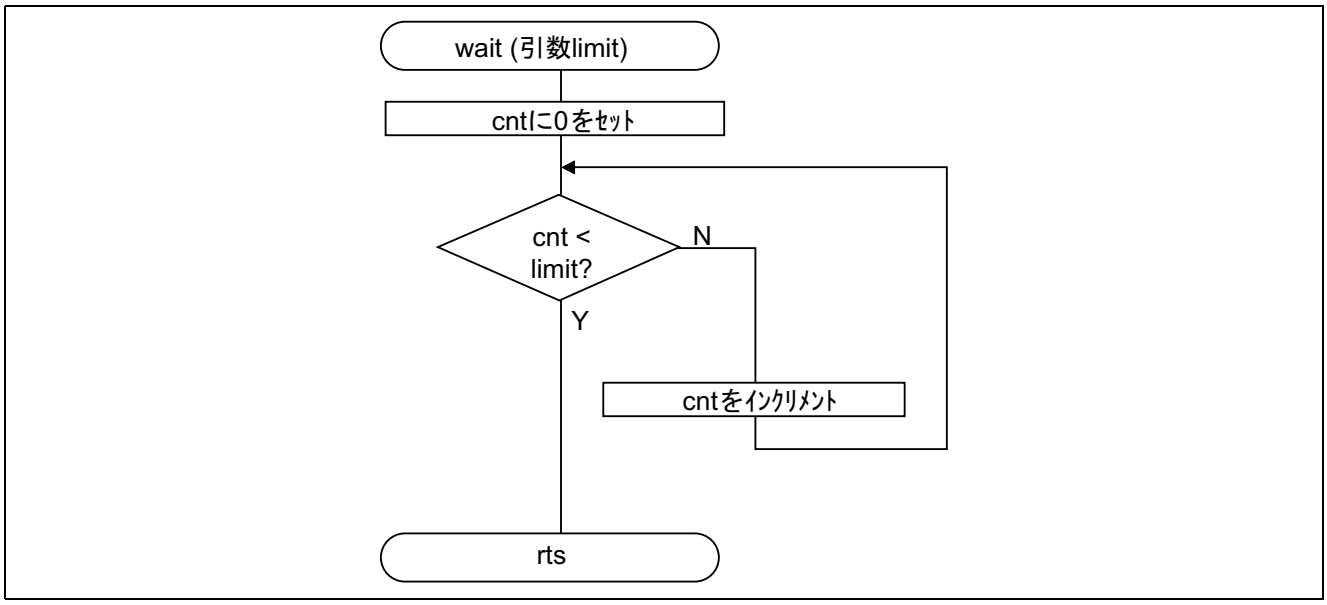


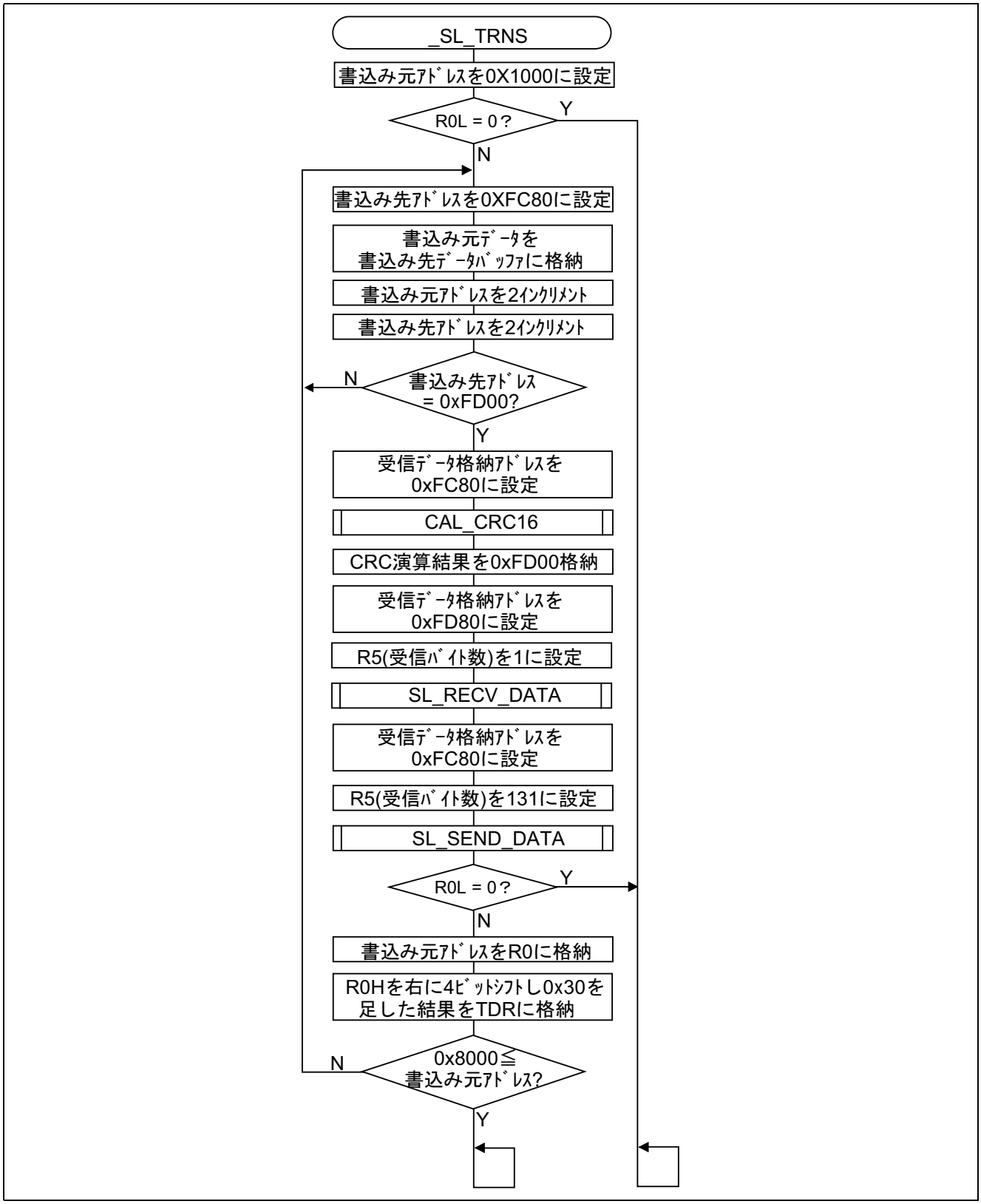
図 6　モジュール階層図

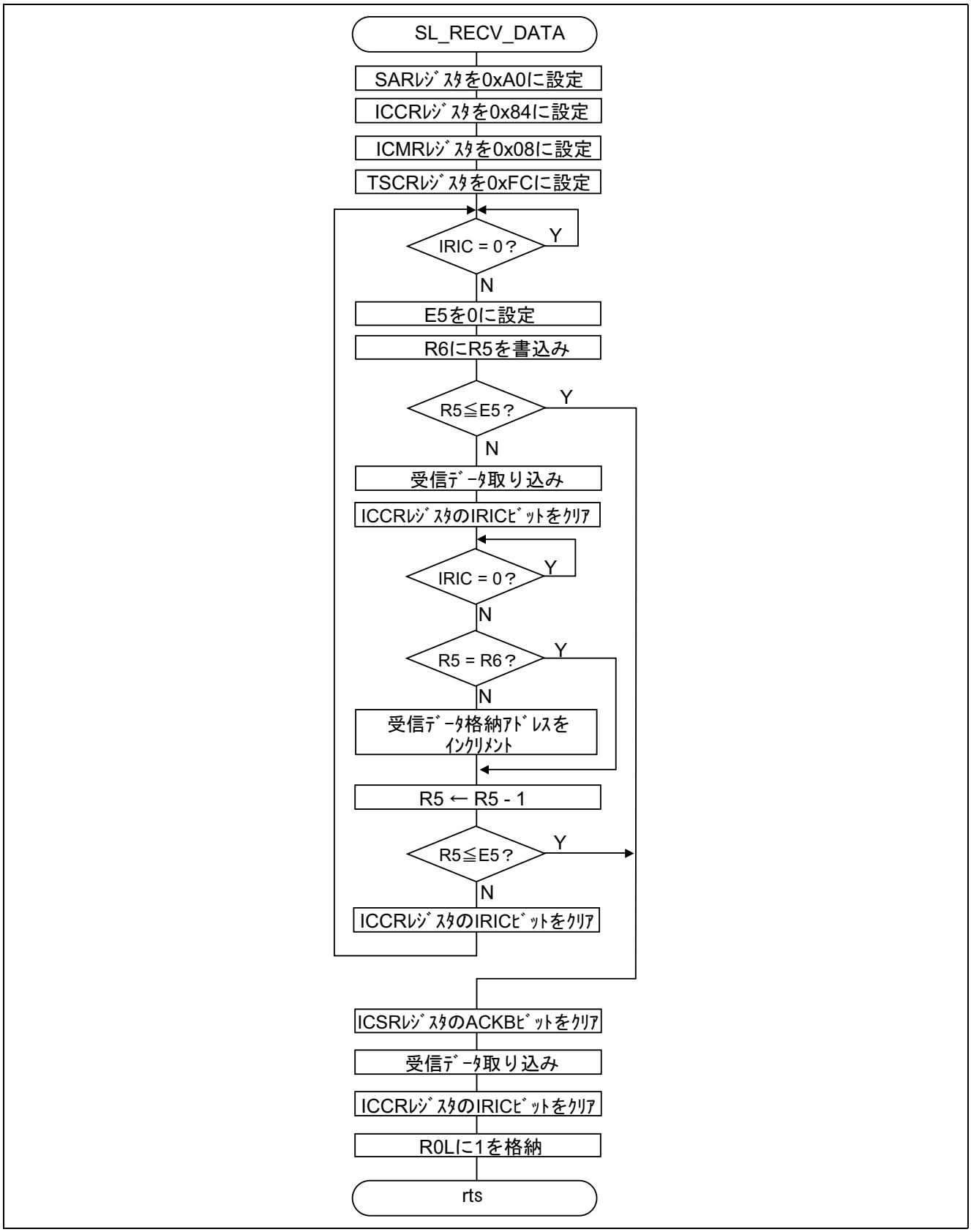
8. フローチャート

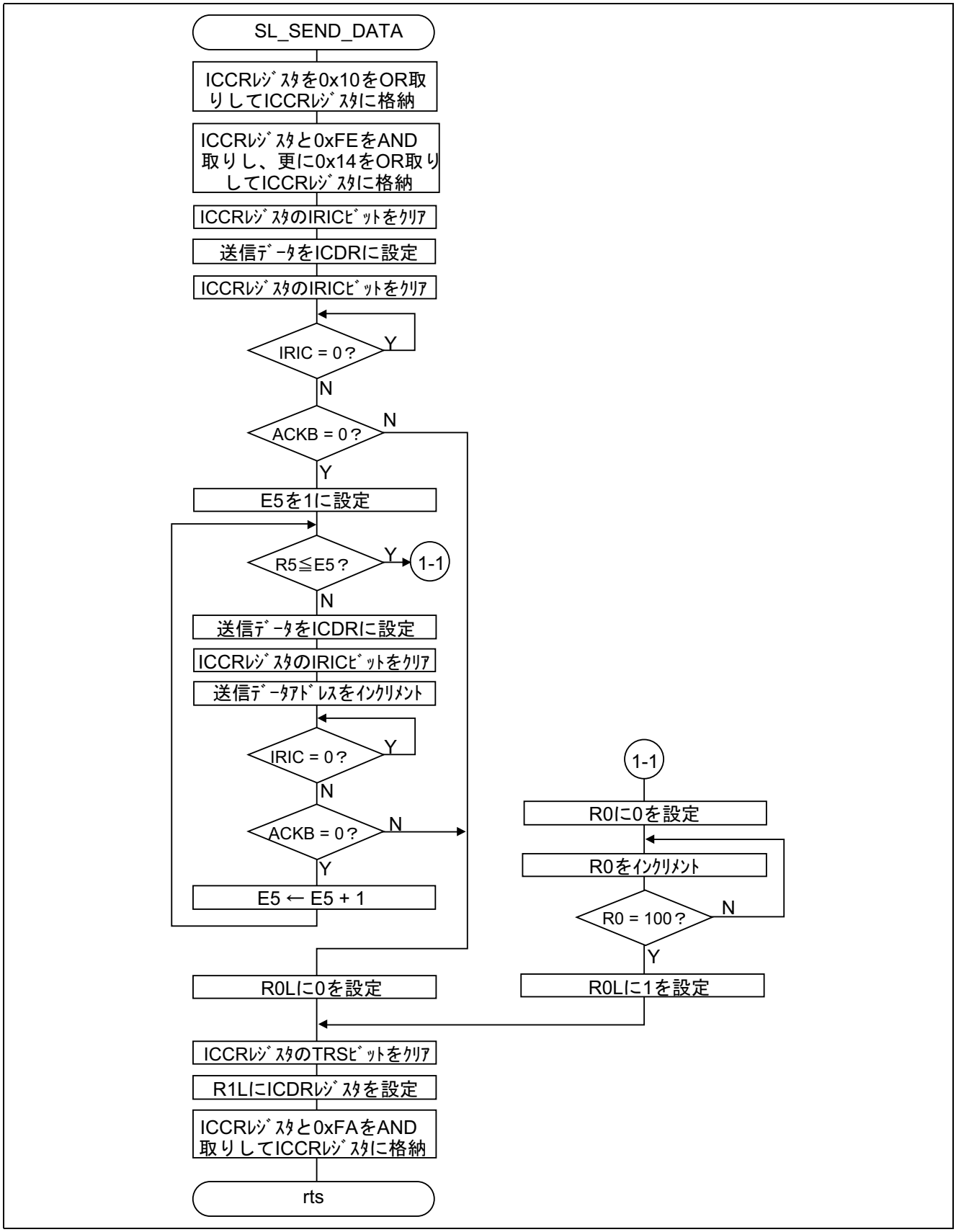

INIT
ｽﾀｯｸﾎﾟｲﾝﾀをH'FF80に設定
Iﾋﾞｯﾄを"1"にｾｯﾄし、
ｼﾘｱﾙ割込み処理を拒否
main JUMP
main
PCR5を0x00に設定
PCR1を0x00に設定
PDR8を0x00に設定
PCR8を0x10に設定
iを0に設定
i < 3000？
N
Y
PDR8を0x00に設定
wait (引数10000)
PDR8を0x10に設定
wait (引数10000)
((PDR5 & 0x20) | (PDR1 & 0x10))の結果をswd1に設定
((PDR5 & 0x20) | (PDR1 & 0x10))の結果をswd2に設定
sw_d1 = sw_d2?
N
Y
sw_d1 = 0x10?
N
Y
Init_sci
SL_TRANS
sw_d1 = 0x20?
N
Y
flprg_cpy
jump_prog(引数0xF780)
iをインクリメント
u_main
rts

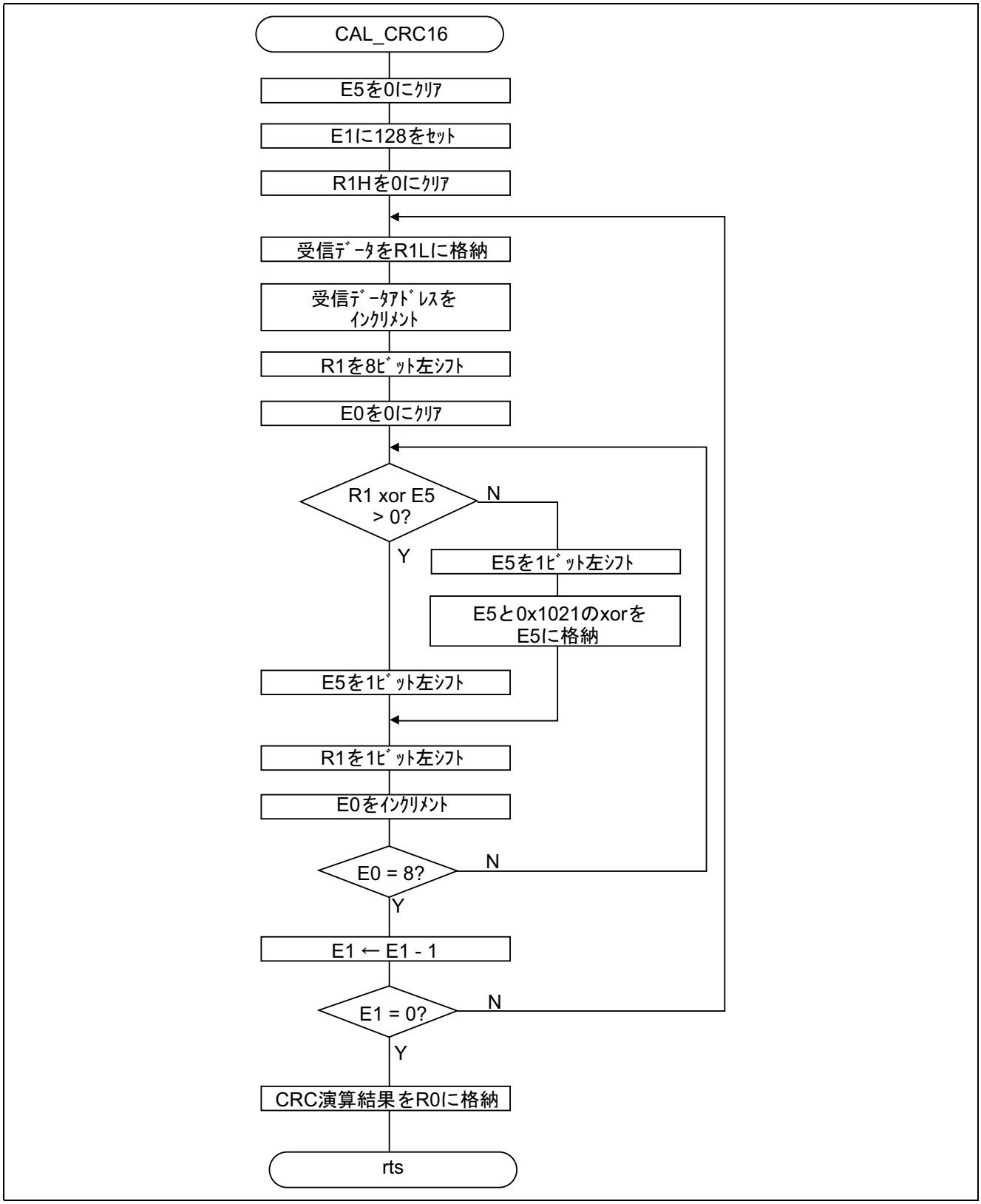

flprg_cpy
ptrに0x04000をｾｯﾄ
r_ptrに0xF780をｾｯﾄ
ptr < 0x0900?
Y
N
*ptrを*r_ptrに書込み
ptrをｲﾝｸﾘﾒﾝﾄ
r_ptrをｲﾝｸﾘﾒﾝﾄ
rts
jump_prog(引数R0)
R0番地へJump
_IIC_TESTの先頭を0x0400にｺﾝﾊﾟｲﾗで設定し、flprg_cpyで
0xF780以降に複製を書き込み、jump_progで複製先を実行する。

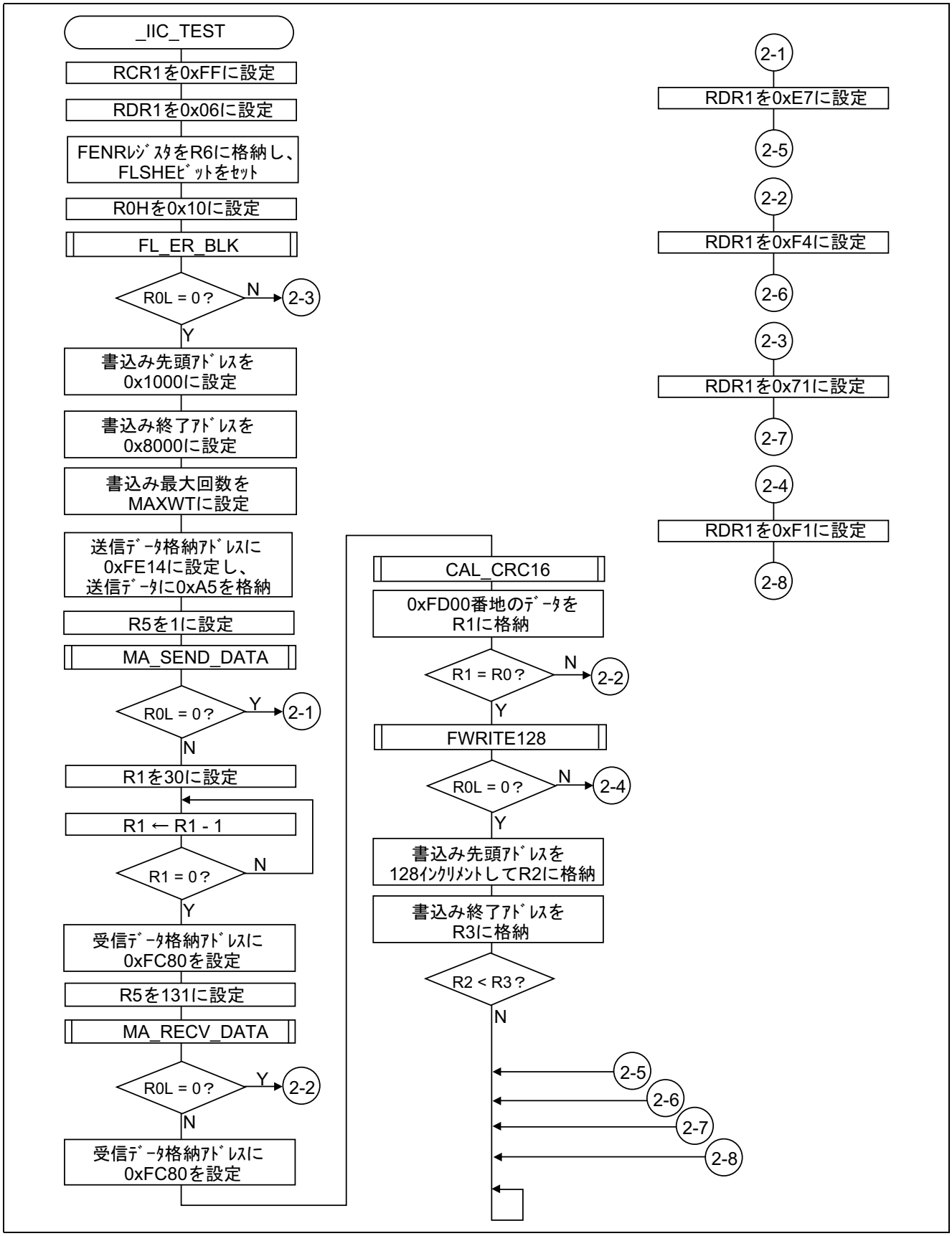

wait (引数limit)
cntに0をｾｯﾄ
cnt < limit?
N
Y
cntをｲﾝｸﾘﾒﾝﾄ
rts

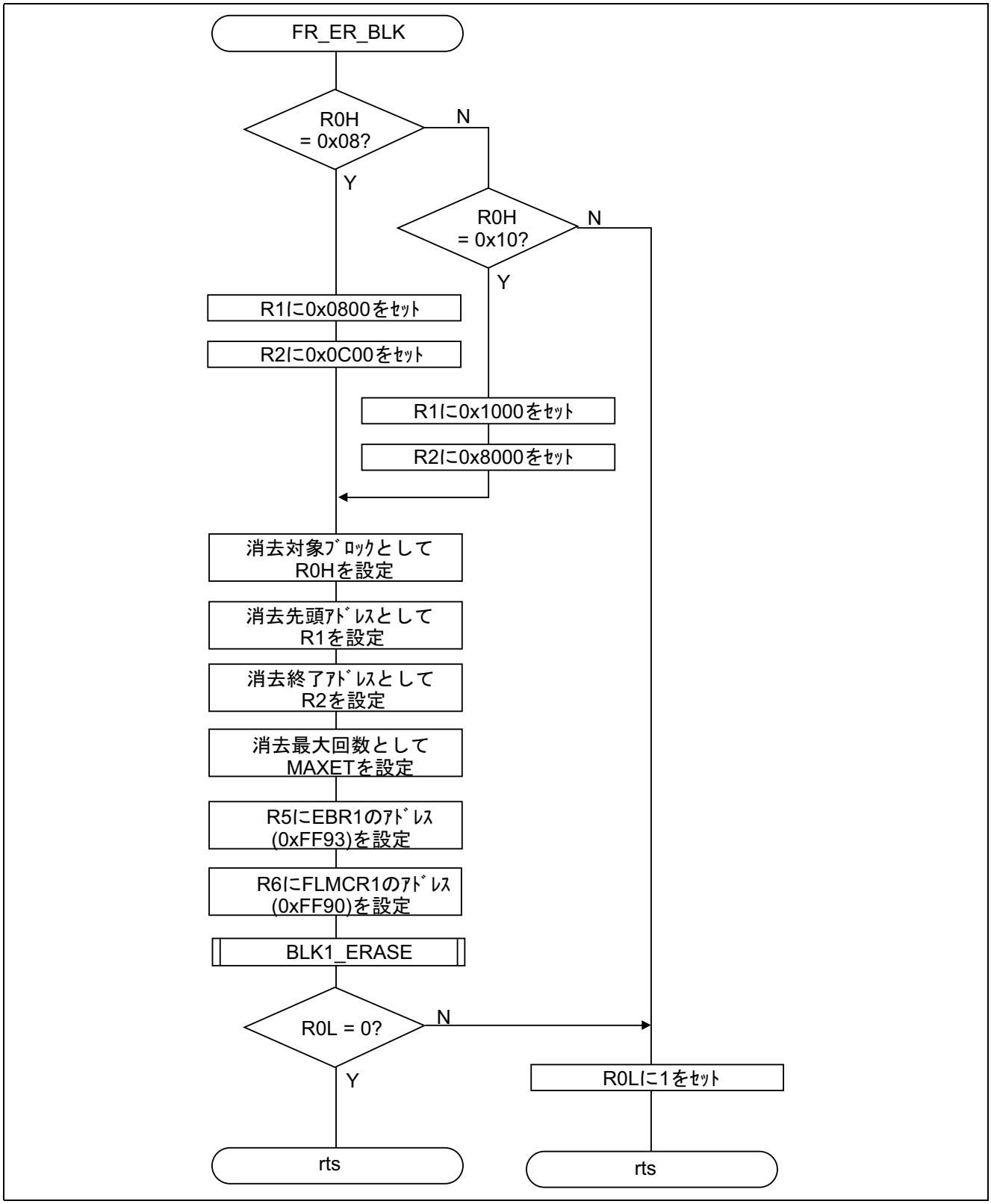

_SL_TRNS
書込み元ｱﾄﾞﾚｽを0X1000に設定
R0L = 0？
Y
N
書込み先ｱﾄﾞﾚｽを0XFC80に設定
書込み元ﾃﾞｰﾀを
書込み先ﾃﾞｰﾀﾊﾞｯﾌｧに格納
書込み元ｱﾄﾞﾚｽを2ｲﾝｸﾘﾒﾝﾄ
書込み先ｱﾄﾞﾚｽを2ｲﾝｸﾘﾒﾝﾄ
N
書込み先ｱﾄﾞﾚｽ
= 0xFD00?
Y
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
0xFC80に設定
CAL_CRC16
CRC演算結果を0xFD00格納
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
0xFD80に設定
R5(受信ﾊﾞｲﾄ数)を1に設定
SL_RECV_DATA
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
0xFC80に設定
R5(受信ﾊﾞｲﾄ数)を131に設定
SL_SEND_DATA
R0L = 0？
Y
N
書込み元ｱﾄﾞﾚｽをR0に格納
R0Hを右に4ﾋﾞｯﾄｼﾌﾄし0x30を
足した結果をTDRに格納
N
0x8000≦
書込み元ｱﾄﾞﾚｽ?
Y

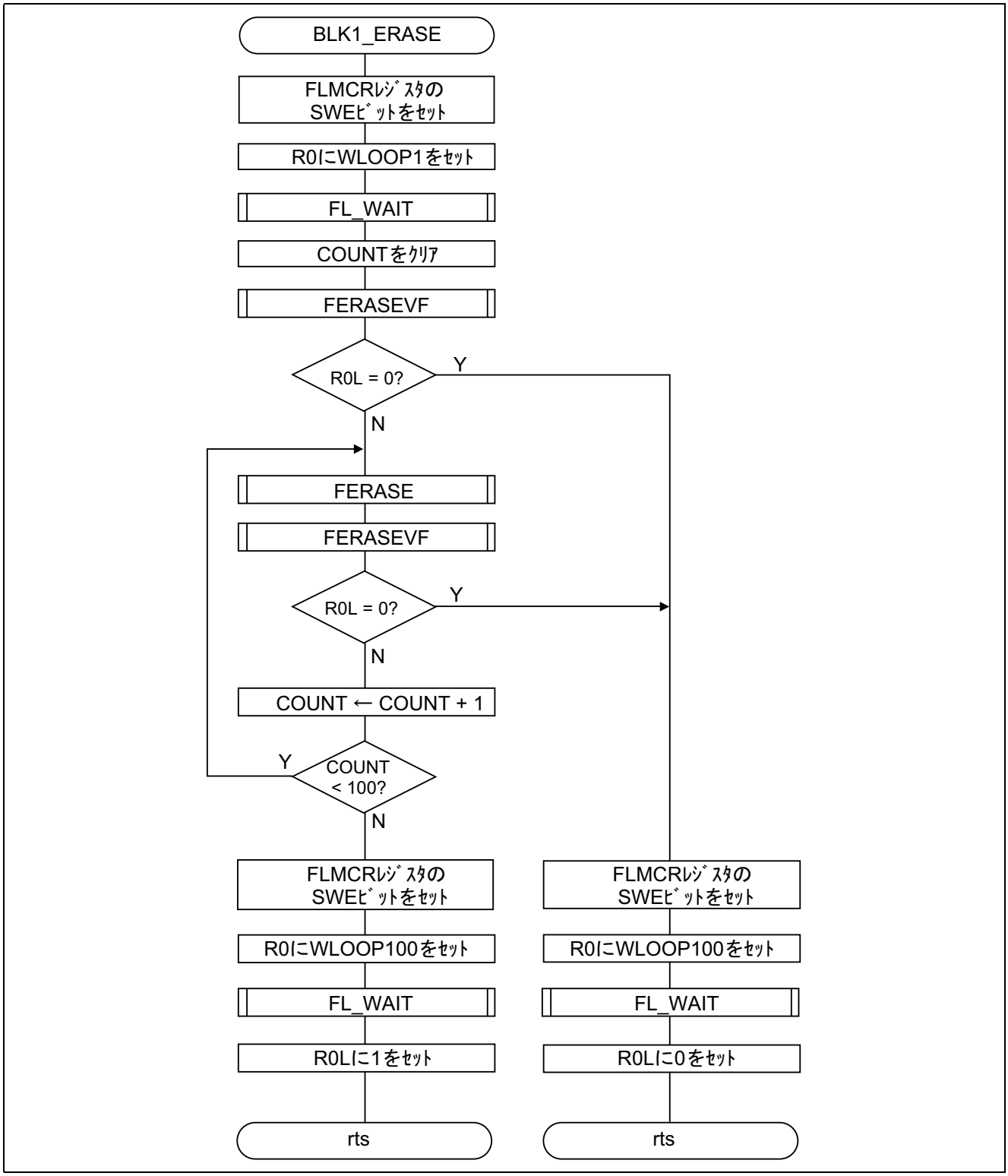

SL_RECV_DATA
SARﾚｼﾞｽﾀを0xA0に設定
ICCRﾚｼﾞｽﾀを0x84に設定
ICMRﾚｼﾞｽﾀを0x08に設定
TSCRﾚｼﾞｽﾀを0xFCに設定
IRIC = 0？
Y
N
E5を0に設定
R6にR5を書込み
R5≦E5？
Y
N
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
R5 = R6？
Y
N
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
R5 ← R5 - 1
R5≦E5？
Y
N
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｸﾘｱ
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
R0Lに1を格納
rts

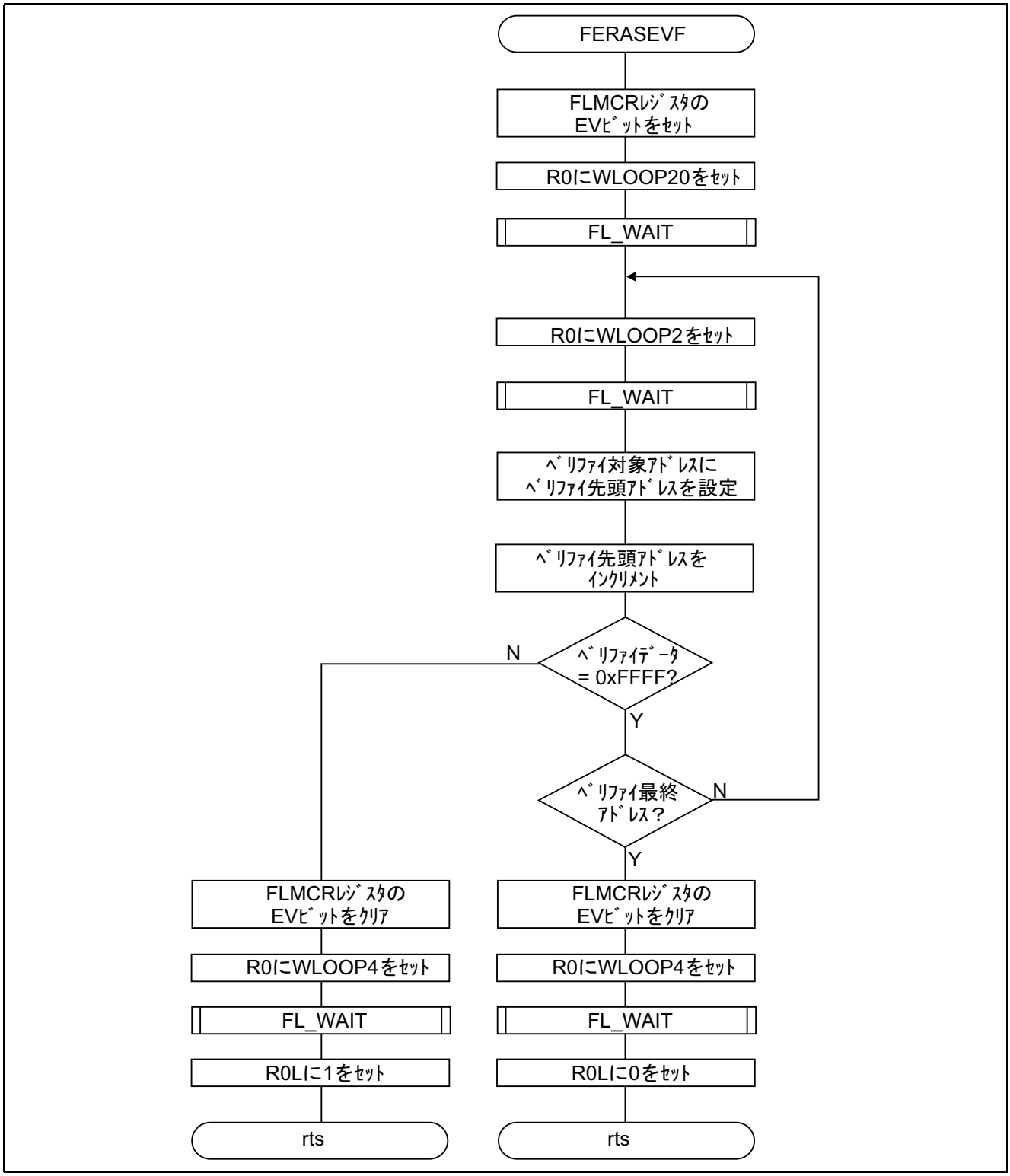

SL_SEND_DATA
ICCRﾚｼﾞｽﾀを0x10をOR取りしてICCRﾚｼﾞｽﾀに格納
ICCRﾚｼﾞｽﾀと0xFEをAND取りし、更に0x14をOR取りしてICCRﾚｼﾞｽﾀに格納
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
送信ﾃﾞｰﾀをICDRに設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5を1に設定
R5≦E5？
Y
1-1
N
送信ﾃﾞｰﾀをICDRに設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
送信ﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5 ← E5 + 1
1-1
R0に0を設定
R0をｲﾝｸﾘﾒﾝﾄ
R0 = 100？
N
Y
R0Lに1を設定
R0Lに0を設定
ICCRﾚｼﾞｽﾀのTRSﾋﾞｯﾄをｸﾘｱ
R1LにICDRﾚｼﾞｽﾀを設定
ICCRﾚｼﾞｽﾀと0xFAをAND取りしてICCRﾚｼﾞｽﾀに格納
rts

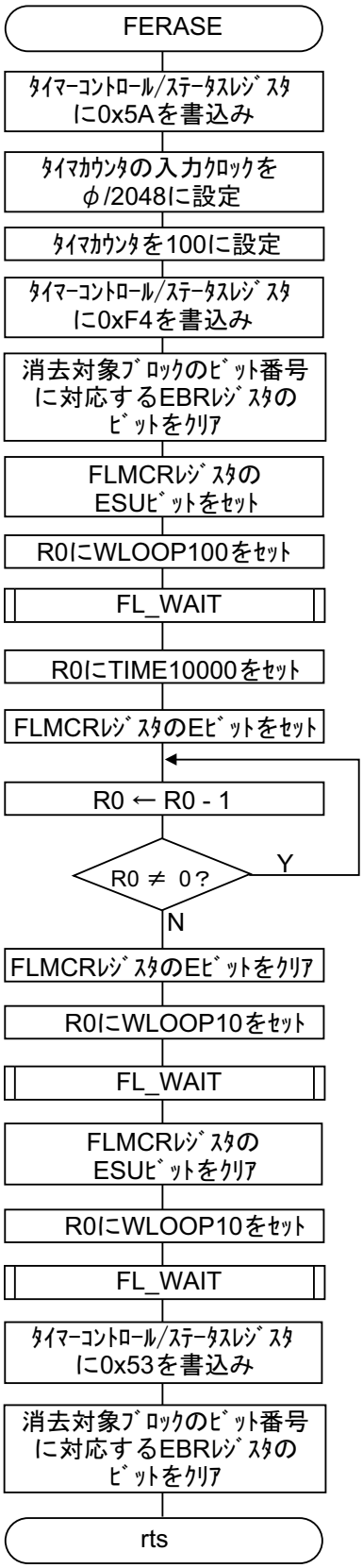

CAL_CRC16
E5を0にｸﾘｱ
E1に128をｾｯﾄ
R1Hを0にｸﾘｱ
受信ﾃﾞｰﾀをR1Lに格納
受信ﾃﾞｰﾀｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
R1を8ﾋﾞｯﾄ左ｼﾌﾄ
E0を0にｸﾘｱ
R1 xor E5
> 0?
N
Y
E5を1ﾋﾞｯﾄ左ｼﾌﾄ
E5と0x1021のxorを
E5に格納
E5を1ﾋﾞｯﾄ左ｼﾌﾄ
R1を1ﾋﾞｯﾄ左ｼﾌﾄ
E0をｲﾝｸﾘﾒﾝﾄ
E0 = 8?
N
Y
E1 ← E1 - 1
E1 = 0?
N
Y
CRC演算結果をR0に格納
rts

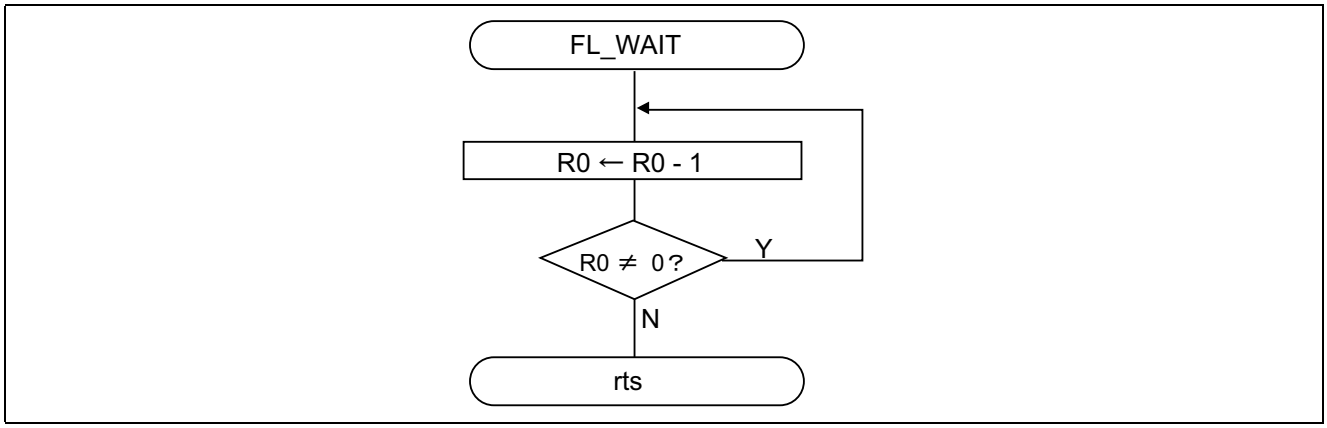

_IIC_TEST
RCR1を0xFFに設定
RDR1を0x06に設定
FENRﾚｼﾞｽﾀをR6に格納し、FLSHEﾋﾞｯﾄをｾｯﾄ
R0Hを0x10に設定
FL_ER_BLK
R0L = 0？
N
2-3
Y
書込み先頭ｱﾄﾞﾚｽを0x1000に設定
書込み終了ｱﾄﾞﾚｽを0x8000に設定
書込み最大回数をMAXWTに設定
送信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽに0xFE14に設定し、送信ﾃﾞｰﾀに0xA5を格納
R5を1に設定
MA_SEND_DATA
R0L = 0？
Y
2-1
N
R1を30に設定
R1 ← R1 - 1
R1 = 0？
N
Y
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽに0xFC80を設定
R5を131に設定
MA_RECV_DATA
R0L = 0？
Y
2-2
N
受信ﾃﾞｰﾀ格納ｱﾄﾞﾚｽに0xFC80を設定
CAL_CRC16
0xFD00番地のﾃﾞｰﾀをR1に格納
R1 = R0？
N
2-2
Y
FWRITE128
R0L = 0？
N
2-4
Y
書込み先頭ｱﾄﾞﾚｽを128ｲﾝｸﾘﾒﾝﾄしてR2に格納
書込み終了ｱﾄﾞﾚｽをR3に格納
R2 < R3？
N
2-5
2-6
2-7
2-8
2-1
RDR1を0xE7に設定
2-5
2-2
RDR1を0xF4に設定
2-6
2-3
RDR1を0x71に設定
2-7
2-4
RDR1を0xF1に設定
2-8

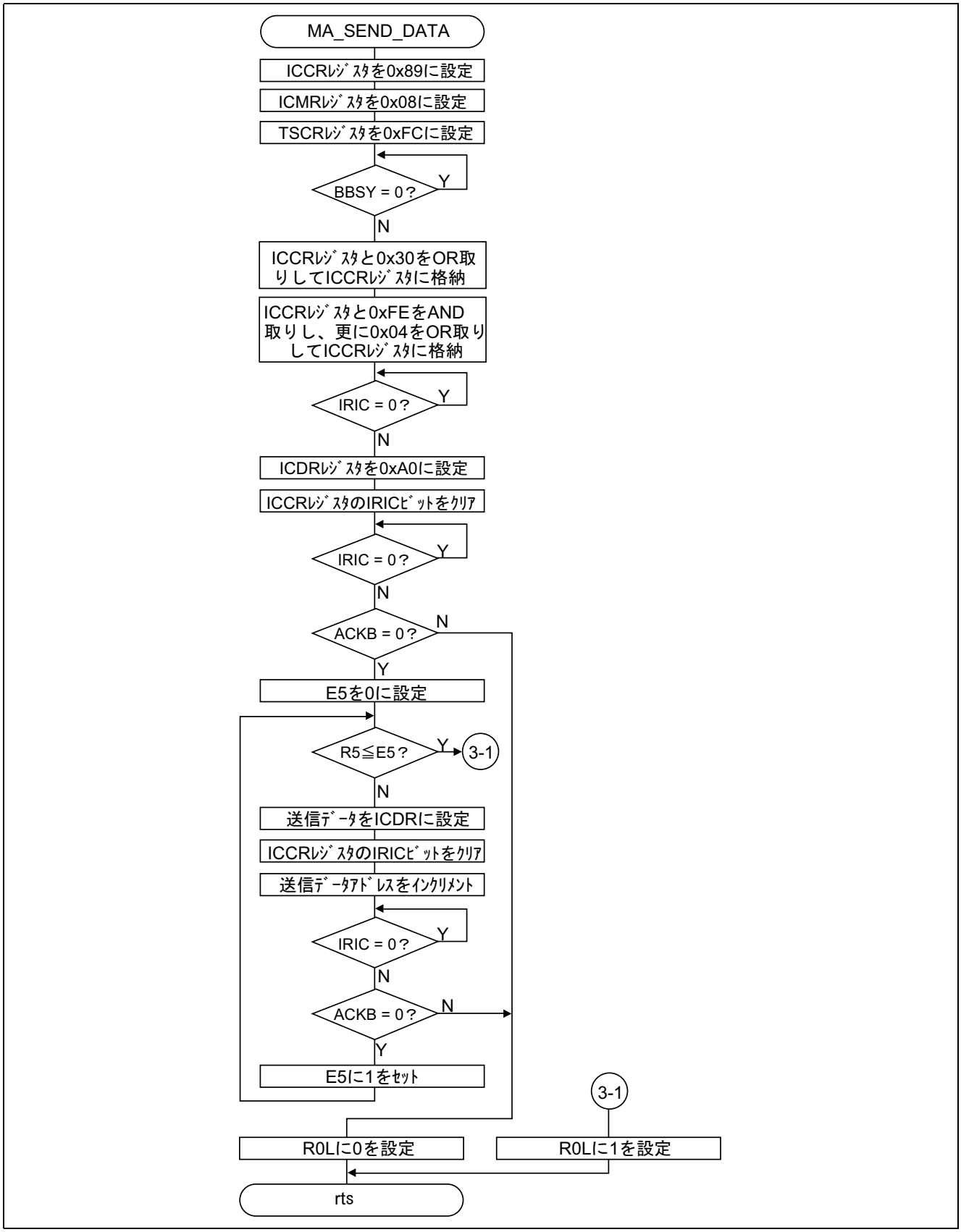

FR_ER_BLK
R0H
= 0x08?
N
Y
R0H
= 0x10?
N
Y
R1に0x0800をｾｯﾄ
R2に0x0C00をｾｯﾄ
R1に0x1000をｾｯﾄ
R2に0x8000をｾｯﾄ
消去対象ﾌﾞﾛｯｸとして
R0Hを設定
消去先頭ｱﾄﾞﾚｽとして
R1を設定
消去終了ｱﾄﾞﾚｽとして
R2を設定
消去最大回数として
MAXETを設定
R5にEBR1のｱﾄﾞﾚｽ
(0xFF93)を設定
R6にFLMCR1のｱﾄﾞﾚｽ
(0xFF90)を設定
BLK1_ERASE
R0L = 0?
N
Y
R0Lに1をｾｯﾄ
rts
rts

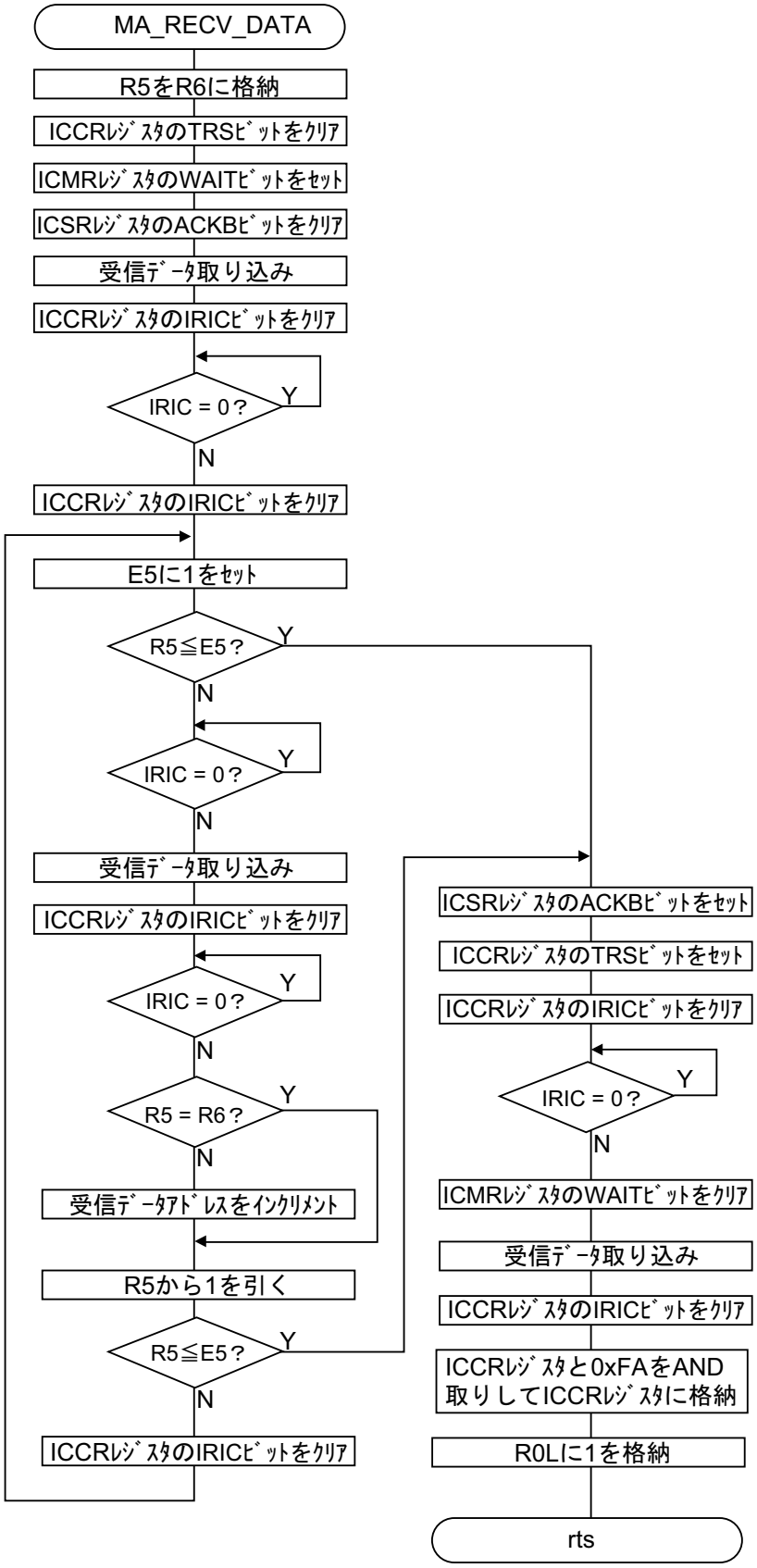

BLK1_ERASE
FLMCRﾚｼﾞｽﾀの SWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP1をｾｯﾄ
FL_WAIT
COUNTをｸﾘｱ
FERASEVF
R0L = 0?
Y
N
FERASE
FERASEVF
R0L = 0?
Y
N
COUNT ← COUNT + 1
COUNT < 100?
Y
N
FLMCRﾚｼﾞｽﾀの SWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
rts
FLMCRﾚｼﾞｽﾀの SWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに0をｾｯﾄ
rts

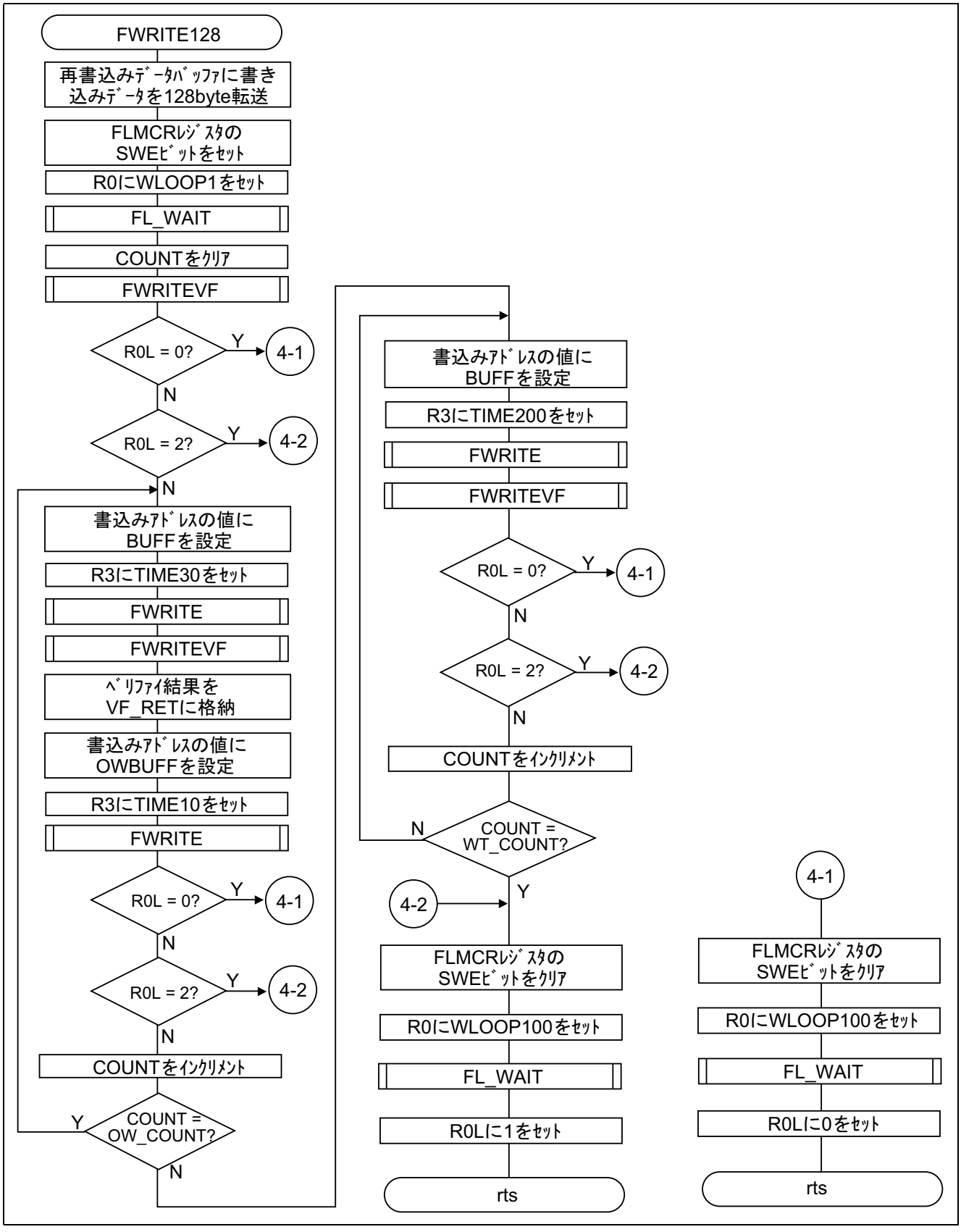

FERASEVF
FLMCRﾚｼﾞｽﾀの
EVﾋﾞｯﾄをｾｯﾄ
R0にWLOOP20をｾｯﾄ
FL_WAIT
R0にWLOOP2をｾｯﾄ
FL_WAIT
ﾍﾞﾘﾌｧｲ対象ｱﾄﾞﾚｽに
ﾍﾞﾘﾌｧｲ先頭ｱﾄﾞﾚｽを設定
ﾍﾞﾘﾌｧｲ先頭ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
ﾍﾞﾘﾌｧｲﾃﾞｰﾀ
= 0xFFFF?
N
Y
ﾍﾞﾘﾌｧｲ最終
ｱﾄﾞﾚｽ?
N
Y
FLMCRﾚｼﾞｽﾀの
EVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP4をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
rts
FLMCRﾚｼﾞｽﾀの
EVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP4をｾｯﾄ
FL_WAIT
R0Lに0をｾｯﾄ
rts

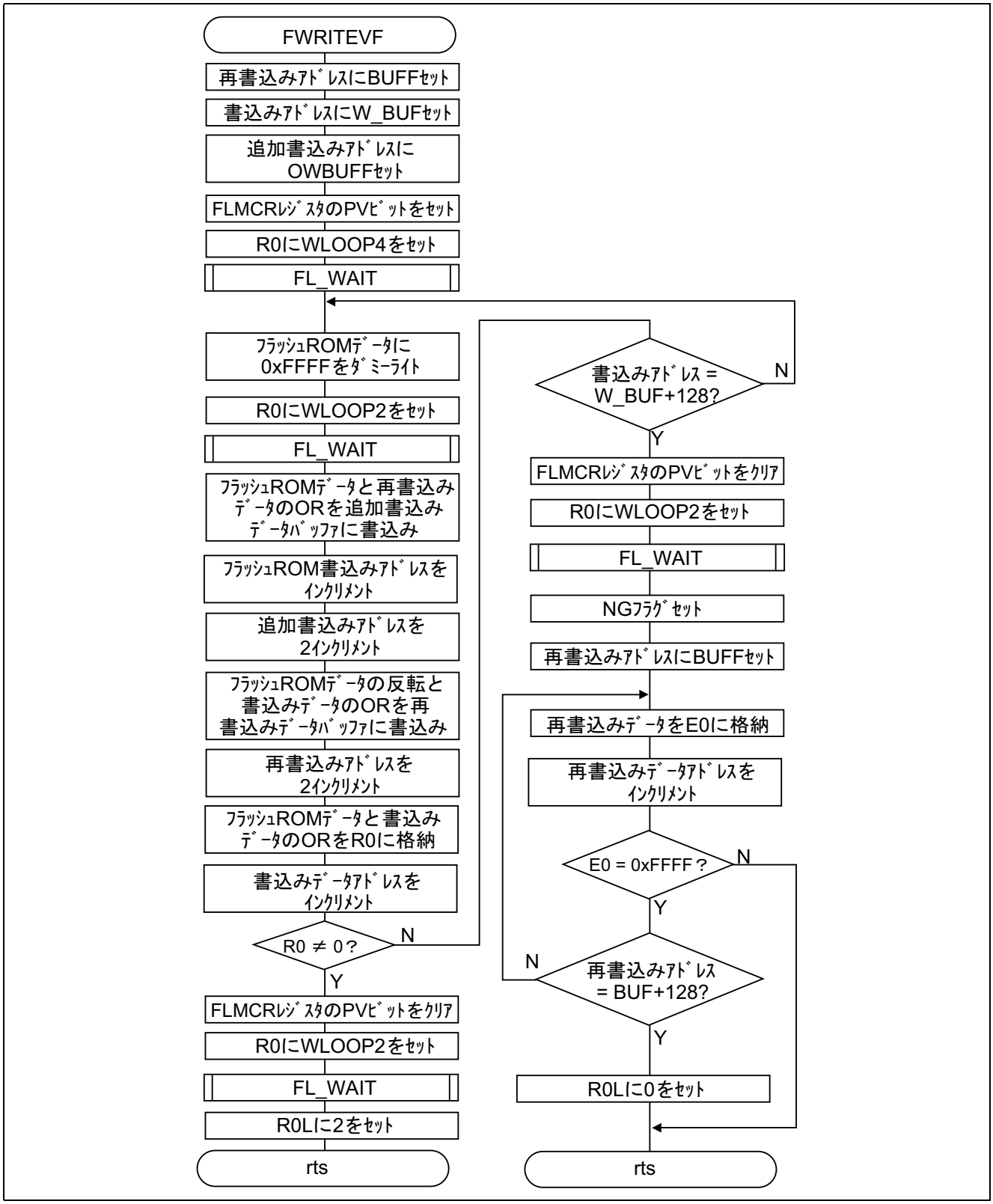

FERASE
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x5Aを書込み
ﾀｲﾏｶｳﾝﾀの入力ｸﾛｯｸを
φ/2048に設定
ﾀｲﾏｶｳﾝﾀを100に設定
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0xF4を書込み
消去対象ﾌﾞﾛｯｸのﾋﾞｯﾄ番号
に対応するEBRﾚｼﾞｽﾀの
ﾋﾞｯﾄをｸﾘｱ
FLMCRﾚｼﾞｽﾀの
ESUﾋﾞｯﾄをｾｯﾄ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0にTIME10000をｾｯﾄ
FLMCRﾚｼﾞｽﾀのEﾋﾞｯﾄをｾｯﾄ
R0 ← R0 - 1
R0 ≠ 0?
Y
N
FLMCRﾚｼﾞｽﾀのEﾋﾞｯﾄをｸﾘｱ
R0にWLOOP10をｾｯﾄ
FL_WAIT
FLMCRﾚｼﾞｽﾀの
ESUﾋﾞｯﾄをｸﾘｱ
R0にWLOOP10をｾｯﾄ
FL_WAIT
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x53を書込み
消去対象ﾌﾞﾛｯｸのﾋﾞｯﾄ番号
に対応するEBRﾚｼﾞｽﾀの
ﾋﾞｯﾄをｸﾘｱ
rts


FL_WAIT
R0 ← R0 - 1
R0 ≠ 0 ?
Y
N
rts

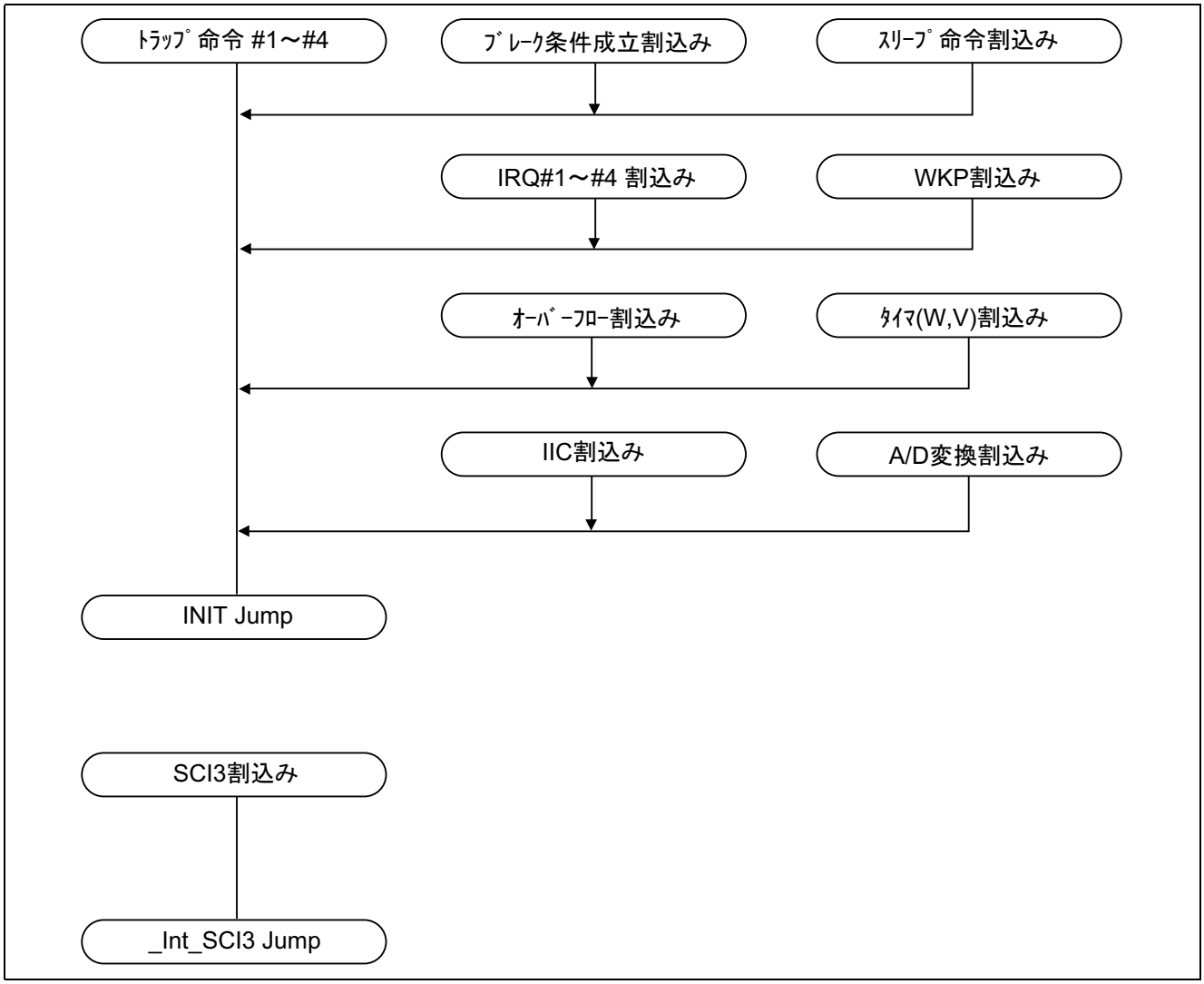

MA_SEND_DATA
ICCRﾚｼﾞｽﾀを0x89に設定
ICMRﾚｼﾞｽﾀを0x08に設定
TSCRﾚｼﾞｽﾀを0xFCに設定
BBSY = 0？
Y
N
ICCRﾚｼﾞｽﾀと0x30をOR取りしてICCRﾚｼﾞｽﾀに格納
ICCRﾚｼﾞｽﾀと0xFEをAND取りし、更に0x04をOR取りしてICCRﾚｼﾞｽﾀに格納
IRIC = 0？
Y
N
ICDRﾚｼﾞｽﾀを0xA0に設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5を0に設定
R5≦E5？
Y
3-1
N
送信ﾃﾞｰﾀをICDRに設定
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
送信ﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
IRIC = 0？
Y
N
ACKB = 0？
N
Y
E5に1をｾｯﾄ
3-1
R0Lに0を設定
R0Lに1を設定
rts


MA_RECV_DATA
R5をR6に格納
ICCRﾚｼﾞｽﾀのTRSﾋﾞｯﾄをｸﾘｱ
ICMRﾚｼﾞｽﾀのWAITﾋﾞｯﾄをｾｯﾄ
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｸﾘｱ
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
E5に1をｾｯﾄ
R5≦E5？
Y
N
IRIC = 0？
Y
N
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
R5 = R6？
Y
N
受信ﾃﾞｰﾀｱﾄﾞﾚｽをｲﾝｸﾘﾒﾝﾄ
R5から1を引く
R5≦E5？
Y
N
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
ICSRﾚｼﾞｽﾀのACKBﾋﾞｯﾄをｾｯﾄ
ICCRﾚｼﾞｽﾀのTRSﾋﾞｯﾄをｾｯﾄ
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
IRIC = 0？
Y
N
ICMRﾚｼﾞｽﾀのWAITﾋﾞｯﾄをｸﾘｱ
受信ﾃﾞｰﾀ取り込み
ICCRﾚｼﾞｽﾀのIRICﾋﾞｯﾄをｸﾘｱ
ICCRﾚｼﾞｽﾀと0xFAをAND
取りしてICCRﾚｼﾞｽﾀに格納
R0Lに1を格納
rts


FWRITE128
再書込みﾃﾞｰﾀﾊﾞｯﾌｧに書き込みﾃﾞｰﾀを128byte転送
FLMCRﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｾｯﾄ
R0にWLOOP1をｾｯﾄ
FL_WAIT
COUNTをｸﾘｱ
FWRITEVF
R0L = 0?
Y
4-1
N
R0L = 2?
Y
4-2
N
書込みｱﾄﾞﾚｽの値にBUFFを設定
R3にTIME30をｾｯﾄ
FWRITE
FWRITEVF
ﾍﾞﾘﾌｧｲ結果をVF_RETに格納
書込みｱﾄﾞﾚｽの値にOWBUFFを設定
R3にTIME10をｾｯﾄ
FWRITE
R0L = 0?
Y
4-1
N
R0L = 2?
Y
4-2
N
COUNTをｲﾝｸﾘﾒﾝﾄ
Y
COUNT = OW_COUNT?
N
書込みｱﾄﾞﾚｽの値にBUFFを設定
R3にTIME200をｾｯﾄ
FWRITE
FWRITEVF
R0L = 0?
Y
4-1
N
R0L = 2?
Y
4-2
N
COUNTをｲﾝｸﾘﾒﾝﾄ
N
COUNT = WT_COUNT?
Y
4-2
FLMCRﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｸﾘｱ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに1をｾｯﾄ
rts
4-1
FLMCRﾚｼﾞｽﾀのSWEﾋﾞｯﾄをｸﾘｱ
R0にWLOOP100をｾｯﾄ
FL_WAIT
R0Lに0をｾｯﾄ
rts


FWRITEVF
再書込みｱﾄﾞﾚｽにBUFFｾｯﾄ
書込みｱﾄﾞﾚｽにW_BUFｾｯﾄ
追加書込みｱﾄﾞﾚｽに
OWBUFFｾｯﾄ
FLMCRﾚｼﾞｽﾀのPVﾋﾞｯﾄをｾｯﾄ
R0にWLOOP4をｾｯﾄ
FL_WAIT
ﾌﾗｯｼｭROMﾃﾞｰﾀに
0xFFFFをﾀﾞﾐｰﾗｲﾄ
R0にWLOOP2をｾｯﾄ
FL_WAIT
ﾌﾗｯｼｭROMﾃﾞｰﾀと再書込み
ﾃﾞｰﾀのORを追加書込み
ﾃﾞｰﾀﾊﾞｯﾌｧに書込み
ﾌﾗｯｼｭROM書込みｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
追加書込みｱﾄﾞﾚｽを
2ｲﾝｸﾘﾒﾝﾄ
ﾌﾗｯｼｭROMﾃﾞｰﾀの反転と
書込みﾃﾞｰﾀのORを再
書込みﾃﾞｰﾀﾊﾞｯﾌｧに書込み
再書込みｱﾄﾞﾚｽを
2ｲﾝｸﾘﾒﾝﾄ
ﾌﾗｯｼｭROMﾃﾞｰﾀと書込み
ﾃﾞｰﾀのORをR0に格納
書込みﾃﾞｰﾀｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
R0 ≠ 0?
N
Y
FLMCRﾚｼﾞｽﾀのPVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP2をｾｯﾄ
FL_WAIT
R0Lに2をｾｯﾄ
rts
書込みｱﾄﾞﾚｽ =
W_BUF+128?
N
Y
FLMCRﾚｼﾞｽﾀのPVﾋﾞｯﾄをｸﾘｱ
R0にWLOOP2をｾｯﾄ
FL_WAIT
NGﾌﾗｸﾞｾｯﾄ
再書込みｱﾄﾞﾚｽにBUFFｾｯﾄ
再書込みﾃﾞｰﾀをE0に格納
再書込みﾃﾞｰﾀｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
E0 = 0xFFFF ?
N
Y
再書込みｱﾄﾞﾚｽ
= BUF+128?
N
Y
R0Lに0をｾｯﾄ
rts


FWRITE
E0に128をｾｯﾄ
書込み元ｱﾄﾞﾚｽのﾃﾞｰﾀを
書込み先ｱﾄﾞﾚｽに書込み
書込み元ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
書込み先ｱﾄﾞﾚｽを
ｲﾝｸﾘﾒﾝﾄ
E0← E0 - 1
E0 ≠ 0？
Y
N
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x5Aを書込み
ﾀｲﾏｶｳﾝﾀの入力ｸﾛｯｸを
φ/64に設定
ﾀｲﾏｰｶｳﾝﾀを166に設定
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0xF4を書込み
FLMCRﾚｼﾞｽﾀの
PSUﾋﾞｯﾄをｸﾘｱ
R0にWLOOP50をｾｯﾄ
FL_WAIT
FLMCRﾚｼﾞｽﾀのPﾋﾞｯﾄをｸﾘｱ
R3← R3 - 1
R0 ≠ 0？
Y
N
FLMCRﾚｼﾞｽﾀのPﾋﾞｯﾄをｸﾘｱ
R0にWLOOP5をｾｯﾄ
FL_WAIT
FLMCRﾚｼﾞｽﾀの
PSUﾋﾞｯﾄをｸﾘｱ
R0にWLOOP5をｾｯﾄ
FL_WAIT
ﾀｲﾏｰｺﾝﾄﾛｰﾙ/ｽﾃｰﾀｽﾚｼﾞｽﾀ
に0x53を書込み
rts


ﾄﾗｯﾌﾟ命令 #1～#4
ﾌﾞﾚｰｸ条件成立割込み
ｽﾘｰﾌﾟ命令割込み
IRQ#1～#4 割込み
WKP割込み
ｵｰﾊﾞｰﾌﾛｰ割込み
ﾀｲﾏ(W,V)割込み
IIC割込み
A/D変換割込み
INIT Jump
SCI3割込み
_Int_SCI3 Jump

9. プログラムリスト

ファイル：Fl_equ.h

```
;*********************************************************************
;              FLASH ER/WR EQU                    2001.07.09
;
;*********************************************************************
PDR1             .EQU    H'FFD4         ; 7segLED Data
PCR1             .EQU    H'FFE4         ; port direction
TDR              .EQU    H'FFAB         ; SCI OUT
;*********************************************************************
;              H8/3664F           IIC REG
;*********************************************************************
ICCR             .EQU    H'FFC4         ;
ICSR             .EQU    H'FFC5         ;
ICDR             .EQU    H'FFC6         ;
ICMR             .EQU    H'FFC7         ;
SAR              .EQU    H'FFC7         ;
TSCR             .EQU    H'FFFC         ;
;
TRS             .BEQU    4,ICCR         ;
ACKE            .BEQU    3,ICCR         ;
BBSY            .BEQU    2,ICCR         ;
IRIC            .BEQU    1,ICCR         ;
ACKB            .BEQU    0,ICSR         ;
WAIT            .BEQU    6,ICMR         ;
;
;*********************************************************************
;DEVICE_CODE    .EQU    H'A0            ;/* EEPROM DEVICE CODE:1010 */
DEVICE_CODE     .EQU    H'80            ;/* MPU DEVICE CODE:1000    */
SLAVE_ADRS      .EQU    H'00            ;/* SLAVE ADRS:0             */
IIC_DATA_W      .EQU    H'00            ;/* WRITE_DATA               */
IIC_DATA_R      .EQU    H'01            ;/* READ_DATA                */
;
;*********************************************************************
; H8/3664F           FLASH REG
;*********************************************************************
FLMCR1          .EQU    H'FF90         ; 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'FF91         ; FLASH MEMORY CONTROL REGISTER 2
FLER:           .EQU    7
EBR1            .EQU    H'FF93         ; OBJECT BLOCK DESIGNATED REGISTER 1
FENR            .EQU    H'FF9B         ; FLASH MEMORY ENABLE REGISTER
FLSHE:          .EQU    7
TCSRWD          .EQU    H'FFC0
TCWD            .EQU    H'FFC1
TMWD            .EQU    H'FFC2
;*********************************************************************
;    WAIT TIME
```

```
;*********************************************************************
;
MHZ             .EQU    11*100          ; 16MHZ
;
WLOOP1          .EQU    1*MHZ/400       ; ROOP WAIT TIME
WLOOP2          .EQU    2*MHZ/400
WLOOP4          .EQU    4*MHZ/400
WLOOP5          .EQU    5*MHZ/400
WLOOP6          .EQU    6*MHZ/400
WLOOP10         .EQU    10*MHZ/400
WLOOP20         .EQU    20*MHZ/400
WLOOP30         .EQU    30*MHZ/400
WLOOP50         .EQU    50*MHZ/400
WLOOP100        .EQU    100*MHZ/400
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
;*********************************************************************
;       定数
;*********************************************************************
MAXWT           .EQU    1000            ; MAX WRITE COUNT
MAXET           .EQU    100             ; MAX ERASE COUNT
OW_COUNT        .EQU    6               ; OVER WRITE COUNT
OK              .EQU    H'0             ; OK FLAG
NG              .EQU    H'1             ; ERR FLAG
WNG             .EQU    H'2             ; WRITE ERR
```

ファイル：Iic_ram.h

```
;                                       2001.07.05
;//H8S/3664F 書き込み/消去プログラム RAM エリア/ /
;
W_BUF           .EQU    H'FC80          ;.B128 WRITE DATA AREA
BUFF            .EQU    H'FD00          ;.B128 RETRY WRITE DATA AREA
OWBUFF          .EQU    H'FD80          ;.B128 OVER WRITE DATA ARIA
COUNT           .EQU    H'FE00          ;.W1 W_COUNT,E_COUNT
W_ADR           .EQU    H'FE02          ;.W1 WRITE ADDRESS AREA
W_ADR_ED        .EQU    H'FE04          ;.W1 WRITE ADDRESS AREA
ET_COUNT        .EQU    H'FE08          ;.W1 MAX E_COUNT
WT_COUNT        .EQU    H'FE0A          ;.W1 MAX W_COUNT
EVF_ST          .EQU    H'FE0C          ;.W1 ERASE VERIFY START ADDRESS
EVF_ED          .EQU    H'FE0E          ;.W1 ERASE VERIFY END ADDRESS
BLK_NO          .EQU    H'FE10          ;.B1 ERASE BIT NUMBER
VF_RET          .EQU    H'FE11          ;.B1 VERIFY CHECK
IIC_SBUF        .EQU    H'FE14          ;.B12 IIC SEND BUF
```

ファイル：INIT.SRC

```
        .EXPORT _INIT
        .EXPORT _jump_prog
        .IMPORT _main
;
        .SECTION    PM,CODE
```

```
_INIT:
    MOV.W    #H'FF80,R7
    LDC.B    #B'10000000,CCR
    JMP      @_main
;
;/*     アセンブラルーチン*     /
;
_jump_prog:
    JSR      @R0
;
   .END
```

ファイル：FLWR.c

```
/************************************************************************/
/*                                                                      */
/*   FILE           :FLWR.c                                             */
/*   DATE           :Thr, Aug 09, 2001                                  */
/*   DESCRIPTION    :Main Program                                       */
/*   CPU TYPE       :H8/3664F                                           */
/*                                                                       */
/*   This file is generated by Renesas Project Generator (Ver.1.2).     */
/*                                                                      */
/************************************************************************/
#include     "machine.h"
#define           PDR5     *(volatile unsigned char *)0xFFD8
#define           PMR5     *(volatile unsigned char *)0xFFE1
#define           PCR5     *(volatile unsigned char *)0xFFE8
#define           PDR8     *(volatile unsigned char *)0xFFDB
#define           PCR8     *(volatile unsigned char *)0xFFEB
#define           PCR1     *(volatile unsigned char *)0xFFE4  /* 7segLED Data */
#define           PDR1     *(volatile unsigned char *)0xFFD4  /* 7segLED Data */
/*;*********************************************************/
/*;      関数定義                                          */
/*;*********************************************************/
extern void   INIT(void);
extern void   jump_prog( unsigned short );
extern void   SL_TRNS ( void );
extern void   u_main ( void );

void          flprg_cpy ( void );
void          main ( void );
void          wait ( unsigned int limit );

#ifdef __cplusplus
extern "C" {
#endif
void abort(void);
#ifdef __cplusplus
}
#endif

#pragma       section                        M /*       P    */
/*;*********************************************************/
```

```
/*;      Main Program                                          */
/*;*****************************************************/
void main(void)
{
int i;
unsigned char sw_d1,sw_d2;
   PCR5 = 0x00;                     /* Port5       入力    */
   PCR1 = 0x00;                     /* Port1       入力    */
   PDR8 = 0x00;                     /* Port84      LOW     */
   PCR8 = 0x10;                     /* Port84      出力    */
   i = 0;
   while (i < 500)   {              /* 5 seconds wait     */
      PDR8 = 0x00;
      wait(10000);
      PDR8 = 0x10;
      wait(10000);

      sw_d1 = ((PDR5 & 0x20) | (PDR1 & 0x10));
      sw_d2 = ((PDR5 & 0x20) | (PDR1 & 0x10));
      if (sw_d1 == sw_d2) {
         if (sw_d1 == 0x10) {
            SL_TRNS();               /* IIC trns */
         }
         if (sw_d1 == 0x20) {
            flprg_cpy();
            jump_prog( 0xf780 );     /* IIC recv & FLASH_WR */
         }
      }
      i++;
   }
   u_main();
}
/*;*******************************************************/
/*; Program Copy     ( ROM:0400-08FF -> RAM:F780-FB7F )     */
/*;*******************************************************/
void flprg_cpy( void ) {
unsigned short *ptr,*r_ptr;

   ptr=(unsigned short*)0x0400;
   r_ptr=(unsigned short*)0xf780;
   while(ptr < (unsigned short*)0x900) {
      *r_ptr++ = *ptr++;
   }
}
void wait( unsigned int limit) {
 unsigned int cnt;

   cnt = 0;
   while (cnt < limit) {
      cnt++;
   }
}

void abort(void)
```

```
{
}
```

ファイル：IIC_SL.src

```
;************************************************************************
;     IIC_SL_SUB             Ver1.0                      2001.07.16
;
;************************************************************************
      .INCLUDE      "IIC_RAM.H"
      .INCLUDE      "FL_EQU.H"         ; FLASH ER/WR EQU
;
      .IMPORT       CAL_CRC16
      .EXPORT       SL_TRNS
;
      .SECTION      PF_2
      .DISPSIZE     FBR=16
;
;************************************************************************
;     SL_TRNS
;************************************************************************
_SL_TRNS:
      MOV.W   #H'1000,R3              ; ptr = 0x1000
;
      MOV.B   #DEVICE_CODE|SLAVE_ADRS,R1L
      MOV.B   R1L,@SAR
;
      MOV.B   #H'FC,R1L
      MOV.B   R1L,@TSCR
;
SL_TRNS10
      MOV.W  #W_BUF,R1
SL_TRNS20
      MOV.W   @R3,R0                  ; *ptr
      MOV.W   R0,@R1                  ; -> W_BUF[n]
      ADD.W   #2,R3                   ; ptr++
      ADD.W   #2,R1                   ; W_BUF[n++]
      CMP.W   #W_BUF+128,R1           ;
      BNE     SL_TRNS20               ; 128 < R1
;
      MOV.W   #W_BUF,R4
      JSR     @CAL_CRC16              ; cal crc16
      MOV.W   #W_BUF+128,R4
      MOV.W   R0,@R4                  ; crc set
;
      MOV.W   #OWBUFF,R4
      MOV.W   #1,R5
      BSR     SL_RECV_DATA            ; recv ok ?
      BEQ     SL_TRNS_ERR
;
      MOV.B   @OWBUFF,R0L
      CMP.B   #H'A5,R0L               ; recv_data = send_req ?
      BNE     SL_TRNS_ERR
;
      MOV.W   #W_BUF,R4
```

```
      MOV.W   #131,R5
      BSR     SL_SEND_DATA
      CMP.B   #0,R0L                  ; return = 0 NG
      BEQ     SL_TRNS_ERR
;
      CMP.W   #H'8000,R3              ;
      BGT     SL_TRNS10               ; H'8000 < R3
SL_TRNS_OK
      BRA     SL_TRNS_OK
;
SL_TRNS_ERR
      BRA     SL_TRNS_ERR
;*********************************************************************
;   IIC SL_SEND_DATA
;       INPUT :R4 送信データ格納アドレス(unsigned char * )
;       INPUT :R5 送信バイト数(unsigned short)
;       OUTPUT :R0L 1=OK/0=NG
;       WORK :R1,E5
;*********************************************************************
SL_SEND_DATA:
      MOV.B   @ICCR,R1L
      OR.B    #H'10,R1L
      MOV.B   R1L,@ICCR               ;/* スレーブ送信設定(MST=0,TRS=1) */
;
      MOV.B   @ICCR,R1L
      AND.B   #H'FE,R1L
      OR.B    #H'04,R1L
      MOV.B   R1L,@ICCR               ;/* 開始条件生成*/
;
      BCLR.B  IRIC
;
      MOV.B   @R4,R1L
      MOV.B   R1L,@ICDR               ;/* 送信データセット*/
;
      BCLR.B  IRIC
SL_SEND_D30
      BTST.B  IRIC                    ;/* 送信完了? */
      BEQ     SL_SEND_D30
;
      BTST.B  ACKB
      BNE     SL_SEND_NG              ;/* ACK? */
;
      MOV.W   #1,E5                   ; cnt = 1;
SL_SEND_D40
      CMP.W   R5,E5                   ; no <= cnt
      BCC     SL_SEND_D60             ; R5 <= E5 (C=1)
;                                     ; no > cnt
      MOV.B   @R4,R1L                 ;ICDR = *ptr / * < n>データセット*/
      MOV.B   R1L,@ICDR               ;
      BCLR.B  IRIC
      ADD.W   #1,R4                   ;ptr++
SL_SEND_D50
      BTST.B  IRIC                    ;/* <n>送信完了? */
```

```
      BEQ     SL_SEND_D50
;
      BTST.B  ACKB
      BNE     SL_SEND_NG            ;/* ACK? */
;
      ADD.W  #1,E5                  ; cnt++
      BRA    SL_SEND_D40
;
SL_SEND_D60
      MOV.W  #0,R0                  ; dummy wait
SL_SEND_D70                         ;
      ADD.W  #1,R0                  ;
      CMP.W  #40,R0                 ;
      BNE    SL_SEND_D70            ;
;
      MOV.B  #1,R0L                 ; return(1)
SL_SEND_D90
      BCLR   TRS                    ; ICCR.TRS = 0;
      MOV.B  @ICDR,R1L              ; dummy = ICDR;
;
      MOV.B  @ICCR,R1L              ;
      AND.B  #H'FA,R1L              ;
      MOV.B  R1L,@ICCR              ; ICCR &= 0xfa;
;
      RTS
;
SL_SEND_NG
      MOV.B  #0,R0L                 ; return(0)
      BRA    SL_SEND_D90
;*********************************************************************
;   IIC SL_RECV_DATA
;      INPUT :R4 受信データ格納アドレス(unsigned char * )
;      INPUT :R5 受信バイト数(unsigned short)
;      OUTPUT :R0L 1=OK
;      WORK :R1,E5,R6
;*********************************************************************
SL_RECV_DATA:
      MOV.B  #H'84,R1L              ;/* ICE=1(P57,P56->SCL,SDA), */
      MOV.B  R1L,@ICCR
      MOV.B  #H'08,R1L
      MOV.B  R1L,@ICMR
;
      BCLR.B ACKB
;
SL_RECV_D10
      BTST.B IRIC                   ;/* 受信完了? */
      BEQ     SL_RECV_D10
;
      MOV.W  #0,E5
SL_RECV_D20
      CMP.W  R5,E5                  ; no < 0
      BCC    SL_RECV_D60            ; R5 <= E5 (C=1)
;                                   ; no > 1
      MOV.B  @ICDR,R1L
```

```
      MOV.B  R1L,@R4                ;/ * 受信データ取込み*/
      BCLR.B IRIC
SL_RECV_D40
      BTST.B IRIC                   ;/* 受信完了? */
      BEQ    SL_RECV_D40
;
      MOV.B   @R4,R1L
      CMP.B   #DEVICE_CODE|SLAVE_ADRS,R1L
      BEQ    SL_RECV_D50
;
      ADD.W   #1,R4                 ; ptr++
SL_RECV_D50
      SUB.W   #1,R5
      CMP.W   R5,E5                 ; no <= 1
      BCC    SL_RECV_D60            ; R5 <= E5 (C=1)
;
      BCLR.B IRIC
;
      BRA    SL_RECV_D20
;
SL_RECV_D60
      BCLR.B ACKB
      MOV.B  @ICDR,R1L
      MOV.B  R1L,@R4                ;/ * 受信データ取込み*/
      BCLR.B IRIC
;
      MOV.B  #1,R0L
      RTS
;*********************************************************************
      .END
```

ファイル：IIC_MA.src

```
;*********************************************************************
;     IIC_MA_SUB             Ver1.0                       2001.07.05
;
;*********************************************************************
      .INCLUDE       "IIC_RAM.H"
      .INCLUDE       "FL_EQU.H"      ; FLASH ER/WR EQU
;
      .EXPORT              _IIC_TEST
      .EXPORT              CAL_CRC16
;*********************************************************************
;       IIC TEST                                          2001.07.05
;
;*********************************************************************
      .SECTION       PF_1
      .DISPSIZE      FBR=16
;*********************************************************************
_IIC_TEST:
;==============フラッシュメモリイネーブルレジスタを ON====================
      MOV.W   #FENR,R6
      BSET.B  #FLSHE,@R6            ; FLSHE ビットセット
```

```
;=====================================================================
;============フラッシュメモリコントロールレジスタ１を ON==================
;      MOV.W  #FLMCR1,R0
;      BSET.B #SWE,@R0           ; SWE ビットセット
;=====================================================================
;
     MOV.B  #H'10,R0H            ; EB4 set
     BSR    FL_ER_BLK            ; BLOCK ERASE
     CMP.B  #OK,R0L              ;
     BNE    FL_ER_ERR
;
     MOV.W  #H'1000,R0
     MOV.W  R0,@W_ADR           ; 書き込み先頭アドレス
     MOV.W  #H'8000,R0
     MOV.W  R0,@W_ADR_ED         ; 書き込み終了アドレス
     MOV.W  #MAXWT,R0            ; 書込み最大回数設定
     MOV.W  R0,@WT_COUNT
;
;    MOV.W  #10,R3               ; loop cnt
IIC_TEST00
;
FL_TEST20
     MOV.W  #IIC_SBUF,R4         ; input:R4(buf_adrs)
     MOV.B  #H'A5,R1L            ;
     MOV.B  R1L,@R4              ; IIC_BUF[0]=H'A5
     MOV.W  #1,R5               ; input:R5(cnt)
     BSR    MA_SEND_DATA         ; /* 送信*/
     BEQ    IIC_SEND_ERR
;
     MOV.W  #30,R1
IIC_TEST10
     SUB.W  #1,R1
     BNE    IIC_TEST10           ; wait
;
     MOV.W  #W_BUF,R4            ; input:R4(buf_adrs)
     MOV.W  #131,R5             ; input:R5(cnt)
     BSR    MA_RECV_DATA         ; /* 受信*/
     BEQ    IIC_RECV_ERR
;
     MOV.W #W_BUF,R4            ; input:R4(buf_adrs)
     BSR    CAL_CRC16           ;
     MOV.W #W_BUF+128,R4        ; input:R4(crc_adrs)
     MOV.W @R4,R1
     CMP.W R1,R0
     BNE    IIC_RECV_ERR
;
     BSR    FWRITE128            ; フラッシュ書き込み処理(128 バイト単位)
     CMP.B  #OK,R0L
     BNE    FL_WR_ERR
;
     MOV.W  @W_ADR_ED,R3         ; 書き込み終了アドレス
     MOV.W  @W_ADR,R2           ; 書き込み先頭アドレス
```

```
      ADD.W  #128,R2              ; アドレスを 128 加算
      MOV.W  R2,@W_ADR            ;
      CMP.W  R2,R3                ;
      BHI    FL_TEST20            ; R3(END) > R2 (unsigned)
;
;*********************************************************************
;     IIC_TEST END
;*********************************************************************
;
IIC_TEST_OK
IIC_SEND_ERR
IIC_RECV_ERR
FL_ER_ERR
FL_WR_ERR
;==============フラッシュメモリイネーブルレジスタを OFF================
      MOV.W   #FENR,R6
      BCLR.B  #FLSHE,@R6          ; F LSHE ビットセット
;=====================================================================
ERR_LOOP
      BRA     ERR_LOOP
;*********************************************************************
; IIC MA_SEND_DATA
;       INPUT : R4 送信データ格納アドレス(unsigned char *)
;       INPUT : R5 送信バイト数(unsigned short)
;       OUTPUT : R0L 1=OK/0=NG
;       WORK : R1,E5
;*********************************************************************
MA_SEND_DATA:
      MOV.B   #H'89,R1L               ;/* ICE=1(P57,P56->SCL,SDA), */
      MOV.B   R1L,@ICCR
      MOV.B   #H'08,R1L
      MOV.B   R1L,@ICMR
      MOV.B   #H'FC,R1L
      MOV.B   R1L,@TSCR
MA_SEND_D10
      BTST.B  BBSY                    ;/* バスビジー？ */
      BNE     MA_SEND_D10
;
MOV.B         @ICCR,R1L
OR.B          #H'30,R1L
MOV.B         R1L,@ICCR               ;/* マスタ送信設定(MST=1,TRS=1) */
;
      MOV.B   @ICCR,R1L
      AND.B   #H'FE,R1L
      OR.B    #H'04,R1L
      MOV.B   R1L,@ICCR               ;/* 開始条件生成*/
MA_SEND_D20
      BTST.B  IRIC                    ;/* 送信 OK? */
      BEQ     MA_SEND_D20
;
      MOV.B   #DEVICE_CODE|SLAVE_ADRS|IIC_DATA_W,R1L
      MOV.B   R1L,@ICDR               ;/* < start>スレーブアドレスセット*/
;
```

```
     BCLR.B  IRIC
MA_SEND_D30
     BTST.B  IRIC                    ;/* 送信完了? */
     BEQ     MA_SEND_D30
;
     BTST.B  ACKB
     BNE     MA_SEND_NG              ;/* ACK? */
;
     MOV.W   #0,E5                   ; cnt = 0;
MA_SEND_D40
     CMP.W   R5,E5                   ; no <= cnt
     BCC     MA_SEND_D60             ; R5 <= E5 (C=1)
; ; no > cnt
     MOV.B   @R4,R1L                 ;ICDR = *ptr / * < n>データセット*/
     MOV.B   R1L,@ICDR               ;
     BCLR.B  IRIC
     ADD.W   #1,R4                   ;ptr++
MA_SEND_D50
     BTST.B  IRIC                    ;/* <n>送信完了? */
     BEQ     MA_SEND_D50
;
     BTST.B  ACKB
     BNE     MA_SEND_NG              ;/* ACK? */
;
     ADD.W   #1,E5                   ; cnt++
     BRA     MA_SEND_D40
;
MA_SEND_D60
     MOV.B   #1,R0L                  ; return(1)
MA_SEND_D90
     RTS
MA_SEND_NG
     MOV.B   #0,R0L                  ; return(0)
     BRA     MA_SEND_D90
;*********************************************************************
;    IIC MA_RECV_DATA
;      INPUT : R4 受信データ格納アドレス(unsigned char *)
;      INPUT : R5 受信バイト数(unsigned short)
;      OUTPUT : R0L 1=OK
;      WORK : R1,E5,R6
;*********************************************************************
MA_RECV_DATA:
     MOV.W   R5,R6
;
     BCLR.B  TRS                      ;/* マスタ受信設定*/
     BSET.B  WAIT
     BCLR.B  ACKB
;
     MOV.B   @ICDR,R1L               ;/* ダミーリード*/
     MOV.B   R1L,@R4                 ;/ * 受信データ取込み*/
;
     BCLR.B  IRIC
MA_RECV_D10
```

```
      BTST.B  IRIC                    ;/* 受信完了? */
      BEQ     MA_RECV_D10
;
      BCLR.B  IRIC
MA_RECV_D20
      MOV.W   #1,E5
      CMP.W   R5,E5                   ; no <= 1
      BCC     MA_RECV_D60             ; R5 <= E5 (C=1)
;                                     ; no > 1
MA_RECV_D30
      BTST.B  IRIC                    ;/* 受信完了? */
      BEQ     MA_RECV_D30
;
      MOV.B   @ICDR,R1L
      MOV.B   R1L,@R4                 ;/ * 受信データ取込み*/
      BCLR.B  IRIC
MA_RECV_D40
      BTST.B  IRIC                    ;/* 受信完了? */
      BEQ     MA_RECV_D40
;
      CMP.W   R5,R6                   ;
      BEQ    MA_RECV_D50
;
      ADD.W   #1,R4                   ; ptr++
MA_RECV_D50
      SUB.W   #1,R5
      CMP.W   R5,E5                   ; no <= 1
      BCC    MA_RECV_D60              ; R5 <= E5 (C=1)
;
      BCLR.B  IRIC
;
      BRA     MA_RECV_D20
;
MA_RECV_D60
      BSET.B  ACKB
      BSET.B  TRS
      BCLR.B  IRIC
MA_RECV_D70
      BTST.B  IRIC                    ;/* 受信完了? */
      BEQ     MA_RECV_D70
;
      BCLR.B  WAIT
      MOV.B   @ICDR,R1L
      MOV.B   R1L,@R4                 ;/ * 受信データ取込み*/
      BCLR.B  IRIC
;
      MOV.B   @ICCR,R1L               ;
      AND.B   #H'FA,R1L               ;
      MOV.B   R1L,@ICCR               ; ICCR & 0xfa
;
      MOV.B   #1,R0L
      RTS
;*********************************************************************
```

```
;     CAL_CRC16
;       INPUT   : R4 受信データ格納アドレス(unsigned char *)
;       WORK    : E0(k),R0(0x1021),E1(cnt),R1(data),R2(work),E5(crc)
;*********************************************************************
CAL_CRC16:
      MOV.W   #H'1021,R0
      MOV.W   #0,E5                    ; crc = 0;
      MOV.W   #128,E1                  ; cnt = 128
      MOV.B   #0,R1H
CAL_CRC10
      MOV.B   @R4,R1L                  ; data = *ptr
      ADD.W   #1,R4                    ; ptr++
      SHLL.W  R1                       ; data << 1 (1)
      SHLL.W  R1                       ; data << 1 (2)
      SHLL.W  R1                       ; data << 1 (3)
      SHLL.W  R1                       ; data << 1 (4)
      SHLL.W  R1                       ; data << 1 (5)
      SHLL.W  R1                       ; data << 1 (6)
      SHLL.W  R1                       ; data << 1 (7)
      SHLL.W  R1                       ; data << 1 (8)
;
      MOV.W   #0,E0                    ; k= 0;
CAL_CRC20
      MOV.W   R1,R2
      XOR.W   E5,R2                    ; crc ^ data
      BPL     CAL_CRC30
      SHLL.W  E5                       ; crc << 1
      XOR.W   R0,E5                    ; 0x1021 ^ crc
      BRA     CAL_CRC40
CAL_CRC30
      SHLL.W  E5                       ; crc << 1
CAL_CRC40
      SHLL.W  R1                       ; data >> 1
      ADD.W   #1,E0                    ; k++
      CMP.W   #8,E0
      BNE      CAL_CRC20
;
      SUB.W   #1,E1                    ; cnt--;
      BNE     CAL_CRC10
;
      MOV.W  E5,R0                     ; return(crc)
      RTS
;*********************************************************************
;*********************************************************************
      .INCLUDE       "FL_ERWR.SRC"     ; FLASH ER/WR SUB
;*********************************************************************
;*********************************************************************
      .END
```

ファイル：fl_erwr.src

```
;*********************************************************************
;     FL_ERWR                     Ver1.0                2001.07.05
;
```

```
;*********************************************************************
;*********************************************************************
;
;   FL_ER_BLK
;      INPUT:(R0H) イレースブロック指定
;              H'01,H'02,H'04,H'08,H'10
;              (EB0, EB1, EB2, EB3, EB4)
;      OUTPUT:(ROL)OK/NG
;*********************************************************************
;
FL_ER_BLK:
;              CMP.B   #H'01,R0H
;              BNE     FL_ER_B10
;              MOV.W   #H'0000,R1      ; EB0(01) 0000:03FF+1
;              MOV.W   #H'0400,R2
;              BRA     FL_ER_B50
FL_ER_B10
;              CMP.B   #H'02,R0H
;              BNE     FL_ER_B20
;              MOV.W   #H'0400,R1      ; EB1(02) 0400:07FF+1
;              MOV.W   #H'0800,R2
;              BRA     FL_ER_B50
FL_ER_B20
;              CMP.B  #H'04,R0H
;              BNE    FL_ER_B30
;              MOV.W  #H'0800,R1       ; EB2(04) 0800:0BFF+1
;              MOV.W  #H'0C00,R2
;              BRA    FL_ER_B50
FL_ER_B30
               CMP.B  #H'08,R0H
               BNE    FL_ER_B40
               MOV.W  #H'0C00,R1        ; EB3(08) 0C00:0FFF+1
               MOV.W  #H'1000,R2
               BRA    FL_ER_B50
FL_ER_B40
               CMP.B  #H'10,R0H
               BNE    FL_ER_BERR        ; EB4(10) 1000:7FFF+1
               MOV.W  #H'1000,R1
               MOV.W  #H'8000,R2
;
FL_ER_B50
               MOV.B  R0H,@BLK_NO        ; 消去対象ブロック
               MOV.W  R1,@EVF_ST         ; 消去先頭アドレスセット
               MOV.W  R2,@EVF_ED         ; 消去最終アドレスセット
               MOV.W  #MAXET,R5          ; 消去最大回数設定
               MOV.W  R5,@ET_COUNT
               MOV.W  #EBR1,R5           ; EBR1 アドレスセット
               MOV.W  #FLMCR1,R6         ; FLMCR アドレスセット
;
               BSR     BLK1_ERASE        ; 消去処理
               CMP.B  #OK,R0L            ;
               BNE     FL_ER_BERR        ; 消去エラー
               RTS
;
FL_ER_BERR
```

```
                MOV.B   #NG,R0L          ; 消去ブロックエラー
                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 = 書込み最大回数                                             *
; * 出力/ R 0L = 結果フラグ(OK=H'00,NG=H'01)                                      *
; ********************************************************************************
FWRITE128       .EQU    $
                MOV.W  #128,R4
                MOV.W  #W_BUF,R5
                MOV.W  #BUFF,R6
                EEPMOV.W                ; W_BUF -> BUFF ブロック転送
;
                MOV.W  #FLMCR1,R6       ; フラッシュ制御レジスタポインタ
;
                BSET.B #SWE,@R6         ; SWE ビットセット
                MOV.W  #WLOOP1,R0       ; 1μS 以上
                BSR     FL_WAIT
;
                XOR.W  R0,R0            ; 書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.W  #BUFF,R2         ; 再書き込みデータ
                MOV.W  #TIME30,R3       ; 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.W  #OWBUFF,R2       ; 追加書込みデータ
                MOV.W  #TIME10,R3       ; 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（追加書込回数）
;
;======= 通常書き込み（追加書き込みなし）=======
FWRTE20         MOV.W  #BUFF,R2           ; 再書き込みデータ
                MOV.W  #TIME200,R3        ; 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 F  WRTE40             ; 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（最大書込回数）
;
;------- 異常終了------------------------------
FWRTE30
                BCLR.B #SWE,@R6           ; SWEビットクリア
                MOV.W  #WLOOP100,R0       ; 100μS以上
                BSR    FL_WAIT
;
                MOV.B  #NG,R0L            ; NG セット
                RTS
;
;------- 正常終了------------------------------
FWRTE40
                BCLR.B #SWE,@R6           ; SWEビットクリア
                MOV.W  #WLOOP100,R0       ; 100μS以上
                BSR    FL_WAIT
;
                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/ E R6 = 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 = 追加書込みデータ１２８バイト                                *
; *       R 0L = 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.W  #BUFF,R1        ; 再書込みデータバッファ
              MOV.W  #W_BUF,R2       ; 書き込みデータバッファ
              MOV.W  @W_ADR,R4       ; フラッシュ ROM 書込みアドレス
;======== 追加書き込みデータバッファ===========
              MOV.W  #OWBUFF,R3      ; 追加書き込みデータバッファ
;=============================================
;
              BSET.B  #PV,@R6        ; PV ビットセット
              MOV.W   #WLOOP4,R0     ; 4μS 以上
              BSR     FL_WAIT
;
FWVF20
              MOV.W   R5,@R4         ; ダミーライト(ラッチ)
              MOV.W   #WLOOP2,R0     ; 2μS 以上
              BSR     FL_WAIT
;
              MOV.W   @R4+,R0        ; フラッシュＲＯＭデータ
;======== 追加書き込みデータの作成=============
              MOV.W   @R1,E0         ; 初期書込み（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,@R3         ; @COUNT = 6～999 無効データ
              ADD.W   #2,R3
;=============================================
              NOT.W   R0
              MOV.W   @R2,E0
              OR.W    R0,E0          ; 反転データ OR 書込みデータ
              MOV.W   E0,@R1         ; 再書込みデータ設定
              ADD.W   #2,R1
              MOV.W   @R2+,E0
              AND.W   E0,R0          ; 読み出しデータ０ AND 書込みデータ１の時
              BNE     FWVF70         ; 書けないエラーへ
;
              CMP.W   #W_BUF+128,R2
              BNE     FWVF20         ; 128 バイトベリファイ
;
              BCLR.B  #PV,@R6        ; PV ビットクリア
              MOV.W   #WLOOP2,R0     ; 2μS 以上
              BSR     FL_WAIT
;
              MOV.B   #NG,R0L        ; NG セット
              MOV.W   #BUFF,R1       ; 再書込みデータ先頭アドレス
```

```
FWVF50          MOV.W   @ER1+,E0
                CMP.W   R5,E0           ; 再書込みデータ 128 バイト全て FF か？
                BNE     FWVF60          ; 判定 H'FF でないならベリファイエラーへ
;
                CMP.W   #BUFF+128,R1
                BNE     FWVF50          ; 128 バイト確認
;
                MOV.B  #OK,R0L          ; OK セット
FWVF60
                RTS
;
;------- FLASH ROM ERR ----------------------
FWVF70
                BCLR.B  #PV,@R6         ; PV ビットクリア
                MOV.W   #WLOOP2,R0      ; 2μS 以上
                BSR     FL_WAIT
;
                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/ E R6 = F LMCR レジスタのアドレス                                       *
; *       @ W_ADR = 書き込みアドレス                                             *
; *       E R2 = 書込みの先頭アドレス( 再書き込みデータ or 追加書込みデータ)     *
; *       E R3 = Pビットセット時間( 10μS o r 3 0μS or 2000μS )                  *
; * 出力/ -                                                                      *
; ********************************************************************************
FWRITE          .EQU  $
                MOV.W @W_ADR,R1         ; 書き込みアドレス
;
                MOV.W  #128,E0
FWRT10          MOV.B  @R2+,R0L         ; 再書込みデータ（バイト単位）
                MOV.B  R0L,@R1          ; ダミーライト（バイト単位）
                ADD.W  #1,R1
                DEC.W  #1,E0
                BNE     FWRT10          ; 128 バイト分繰り返し
;                                         <WDT 初期化>
                MOV.B  #H'5A,R0H        ;
                MOV.B  R0H,@TCSRWD      ; TCSRWD = H'5A
                MOV.B  #H'F8,R0H        ;
                MOV.B  R0H,@TMWD        ; TMWD = H'F8(φ/64)
                MOV.B  #166,R0H         ;
                MOV.B  R0H,@TCWD        ; TCWD = 166:(256-99)*4us=360us
                MOV.B  #H'F4,R0H        ;
                MOV.B  R0H,@TCSRWD      ; TCSRWD = H'F4 WDT On
;
                BSET.B #PSU,@R6         ; PSU ビットセット
                MOV.W  #WLOOP50,R0      ; 50μS 以上
```

```
                BSR     FL_WAIT
;
;======= WRITE パルス印可=======================
                BSET.B  #P,@R6          ; P ビットセット（書込み）
FWRT40          DEC.W   #1,R3           ; 書込み時間：10μS or 30μS or 200μS
                BNE     FWRT40:16
;================================================
                BCLR.B  #P,@R6          ; P ビットをクリア
                MOV.W   #WLOOP5,R0      ; 5μS
                BSR      FL_WAIT
;
                BCLR.B  #PSU,@R6        ; PSU ビットをクリア
                MOV.W   #WLOOP5,R0      ; 5μS 以上
                BSR      FL_WAIT
;
                MOV.B   #H'53,R0H       ;
                MOV.B   R0H,@TCSRWD     ; TCSRWD = H'53 WDT Off
                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/ E R6 = F LMCR レジスタのアドレス                                       *
; *       E R5 = 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/ R 0L = 結果フラグ(OK=H'00,NG=H'01)                                     *
; ********************************************************************************
BLK1_ERASE      .EQU    $
                BSET.B #SWE,@R6         ; SWE ビットセット
                MOV.W  #WLOOP1,R0       ; 1μS 以上
                BSR     FL_WAIT
;
                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
                BCLR.B #SWE,@R6        ; SWE ビットクリア
                MOV.W  #WLOOP100,R0 ; 100μS 以上
                BSR    FL_WAIT
;
                MOV.B  #NG,R0L         ; NG セット
                RTS
;
;------- 正常終了-----------------------------
BLK1_40
                BCLR.B #SWE,@R6        ; SWE ビットクリア
                MOV.W  #WLOOP100,R0 ; 100μS 以上
                BSR    FL_WAIT
;
                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/ E R6 = 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/ R OL = ベリファイ結果(OK=H'00,NG=H'01)                                 *
; ********************************************************************************
FERASEVF        .EQU $
                MOV.W  @EVF_ST,R1
                MOV.W  @EVF_ED,R2
                MOV.W  #H'FFFF,E0      ; ダミーライト、消去確認データ
                BSET.B #EV,@R6         ; EV ビットセット
                MOV.W  #WLOOP20,R0  ; 20μS 以上
                BSR     FL_WAIT
;
VRF30
                MOV.W  E0,@R1          ; ダミーライト（アドレス・ラッチ）
                MOV.W  #WLOOP2,R0   ; 2μS 以上
                BSR     FL_WAIT
;
                MOV.W  @R1+,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.W  R1,R2
```

```
                BNE     VRF30
;
                BCLR.B  #EV,@R6         ; EV ビットをクリア
                MOV.W   #WLOOP4,R0      ; 4μS 以上
                BSR     FL_WAIT
;
                MOV.B   #OK,R0L         ; OK フラグセット
                RTS
;
;------- FERASEVF ERR -----------------------
VRF60
                BCLR.B  #EV,@R6         ; EV ビットをクリア
                MOV.W   #WLOOP4,R0      ; 4μS 以上
                BSR     FL_WAIT
;
                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/ E R6 = FLMCR レジスタのアドレス*
; *       E R5 = 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    $
;                 <WDT 初期化>
                MOV.B   #H'5A,R0H     ;
                MOV.B   R0H,@TCSRWD ; TCSRWD = H'5A
                MOV.B   #H'Fd,R0H     ;
                MOV.B   R0H,@TMWD     ; TMWD = H'Fd(φ/2048)
                MOV.B   #100,R0H      ;
                MOV.B   R0H,@TCWD     ; TCWD = 100:(256-166)*0.128ms=20ms
                MOV.B   #H'F4,R0H     ;
                MOV.B   R0H,@TCSRWD   ; TCSRWD = H'F4 WDT On
;
                MOV.B   @BLK_NO,R0H
                MOV.B   R0H,@R5         ; 消去対象ブロックのビット EBR にセット
                BSET.B  #ESU,@R6        ; ESU ビットセット
                MOV.W   #WLOOP100,R0    ; 100μS 以上
                BSR     FL_WAIT
;
                MOV.L   #TIME10000,ER0 ; 10mS
;======= ERASE パルス印可======================
                BSET.B  #E,@R6          ; E ビットセット（消去）
FERS20          DEC.W   #1,R0           ; 消去時間：10mS
                BNE      FERS20:16
;==============================================
                BCLR.B  #E,@R6          ; E ビットをクリア
```

```
              MOV.W   #WLOOP10,R0   ; 10μS 以上
              BSR     FL_WAIT
;
              BCLR.B  #ESU,@R6      ; ESU ビットをクリア
              MOV.W   #WLOOP10,R0   ; 10μS 以上
              BSR     FL_WAIT
;
              MOV.B   #H'53,R0H     ;
              MOV.B   R0H,@TCSRWD   ; TCSRWD = H'53 WDT Off
              MOV.B   #0,R0H
              MOV.B   R0H,@R5       ; EBR 中の消去対象ブロックをクリア
              RTS
;
;==========WAIT サブルーチン====================
FL_WAIT DEC.W #1,R0
              BNE FL_WAIT
              RTS
;===============================================
```

ファイル：vect.src

```
;*********************************************************/
;    Vector Table                                         */
;*********************************************************/
     .IMPORT            _INIT
     .IMPORT            VTRAP0,VTRAP1,VTRAP2,VTRAP3
     .IMPORT            VBRAK,VSLEP
     .IMPORT            VIRQ0,VIRQ1,VIRQ2,VIRQ3
     .IMPORT            VWKP,VOVRF
     .IMPORT            VTMRW,VTMRV
     .IMPORT           VSCI3,VIIC,VADC
;
     .SECTION    V0,CODE,LOCATE=H'0000
;*********************************************************/
;                   adrs
;*********************************************************/
     .DATA.W            _INIT ; 00 RESET VECTER ADDRESS
;
     .SECTION    V1,CODE,LOCATE=H'0010
     .DATA.W            VTRAP0    ; 10 TRAP#0
     .DATA.W            VTRAP1    ; 12 TRAP#1
     .DATA.W            VTRAP2    ; 14 TRAP#2
     .DATA.W            VTRAP3    ; 16 TRAP#3
     .DATA.W            VBRAK     ; 18 BREAK
     .DATA.W            VSLEP     ; 1A SLEEP
     .DATA.W            VIRQ0     ; 1C IRQ0
     .DATA.W            VIRQ1     ; 1E IRQ1
     .DATA.W            VIRQ2     ; 20 IRQ2
     .DATA.W            VIRQ3     ; 22 IRQ3
     .DATA.W            VWKP      ; 24 WKP
     .DATA.W            VOVRF     ; 26 OVER FLOW
;
     .SECTION    V2,CODE,LOCATE=H'002A
```

```
      .DATA.W           VTMRW     ; 2A TIMER W
      .DATA.W           VTMRV     ; 2C TIMER V
      .DATA.W           VSCI3     ; 2E SCI3_RX/TX/ERR
      .DATA.W           VIIC      ; 30 IIC
      .DATA.W           VADC      ; 32 ADC
;*********************************************************/
      .END
```

ファイル：u_vect.src

```
;*********************************************************/
;     User Vector Table                                   */
;*********************************************************/
      .IMPORT    _INIT
      .EXPORT    VTRAP0,VTRAP1,VTRAP2,VTRAP3
      .EXPORT    VBRAK,VSLEP
      .EXPORT    VIRQ0,VIRQ1,VIRQ2,VIRQ3
      .EXPORT    VWKP,VOVRF
      .EXPORT    VTMRW,VTMRV
      .EXPORT    VSCI3,VIIC,VADC
;
      .SECTION   UV,CODE,LOCATE=H'1000
;*********************************************************/
; Jump user_program
;*********************************************************/
VTRAP0:
         JMP     @_INIT
VTRAP1:
         JMP     @_INIT
VTRAP2:
         JMP    @_INIT
VTRAP3:
         JMP     @_INIT
VBRAK:
         JMP     @_INIT
VSLEP:
         JMP     @_INIT
VIRQ0:
JMP @_INIT
VIRQ1:
         JMP     @_INIT
VIRQ2:
         JMP     @_INIT
VIRQ3:
         JMP     @_INIT
VWKP:
         JMP    @_INIT
VOVRF:
         JMP     @_INIT
VTMRW:
         JMP     @_INIT
VTMRV:
         JMP     @_INIT
VSCI3:
         JMP     @_INIT
```

```
VIIC:
        JMP     @_INIT
VADC:
        JMP     @_INIT
;************************************************************/
     .END
```

ファイル：LED.c(ユーザープログラム(参考))

```
/************************************************************/
/*                                                          */
/*   FILE         :LED.c                                    */
/*   DATE         :Tue, Jul 31, 2001                         */
/*   DESCRIPTION  :Main Program                              */
/*   CPU TYPE     :H8/3664N                                  */
/*                                                          */
/*      LED    Test                                         */
/*                                                          */
/************************************************************/
#include      "machine.h"

#define       PDR5    *(volatile unsigned char *)0xFFD8
#define       PMR5    *(volatile unsigned char *)0xFFE1
#define       PCR5    *(volatile unsigned char *)0xFFE8
#define       PDR8    *(volatile unsigned char *)0xFFDB
#define       PCR8    *(volatile unsigned char *)0xFFEB

unsigned  int cnt1;
unsigned  int cnt2;

/*;*********************************************************/
/*;     関数定義                                            */
/*;*********************************************************/
void   u_main ( void );
void   u_wait ( void );

#pragma       section                          /* P         */
/*;*********************************************************/
/*;    Main Program                                        */
/*;*********************************************************/
void u_main ( void ) {

    PDR5 = 0x00;
    PDR8 = 0x00;
    PMR5 = 0x00;
    PCR5 = 0x10;
    PCR8 = 0x10;
    while (1) {
        u_wait();
        PDR5 = 0x10;
        u_wait();
        PDR5 = 0x00;
        u_wait();
        PDR8 = 0x10;
```

```
        u_wait();
        PDR8 = 0x00;
    }
}
/*;*****************************************************/
/*;    Sub Program                                      */
/*;*****************************************************/
void u_wait(void) {

    cnt1 = 0;
    cnt2 = 0;
    while (cnt2 < 1000) {
        while (cnt1 < 300) {
            cnt1++;
        }
    cnt2++;
    cnt1 = 0;
    }
}
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.12.22	—	初版発行

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

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