

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ひ申し上げます。

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のですが、誤りが無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、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と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。

H8/300L SLP シリーズ

H8/38024F ユーザモードフラッシュメモリ書き込み 8ビットパラレル通信編

要旨

マスタ側フラッシュメモリ上にある書き込みデータを、スレーブ側のフラッシュメモリに書き込みます。書き込みデータの転送は、8ビットパラレル通信で行います。

動作確認デバイス

H8/38024

目次

1. 仕様	2
2. 詳細仕様説明	3
3. 動作概要	8
4. シーケンス図	12
5. スレーブ側通常プログラム	18
6. スレーブ側書き込み / 消去制御プログラム	24
7. マスタ側プログラム	51
8. プログラムリスト	64

注意事項：入出力端子前のチルダ(~)は、負論理を表します。

本資料に掲載してるオンボード書き換えのアルゴリズムは、保証アルゴリズムではありません。
また、書き換え時間、その他データも保証値ではありません。
あくまでもお客様がシステム設計の際、ご参考として役立てて頂けるよう纏めたものです。

1. 仕様

- (1) ユーザモードによるフラッシュメモリ書き込みを行います。
- (2) マスタ側フラッシュメモリ上にある書き込みデータを，スレーブ側のフラッシュメモリに書き込みます。
- (3) 書き込みデータの転送は，8ビットパラレル通信で行います。
- (4) マスタ側のスイッチ 0(SW0)が ON になると，マスタ側からスレーブ側にフラッシュメモリ書き込み開始コマンドを送信し，スレーブ側フラッシュメモリへの書き込みを開始します。
- (5) マスタ側，スレーブ側共に，フラッシュメモリ書き込み動作中は LED1 が消灯，LED2 が点灯し，フラッシュメモリ書き込み動作終了後は LED1 が点灯，LED2 が消灯します。
- (6) マスタ側の \sim IRQ0 端子にスイッチ 0(SW0)が接続されています。
- (7) マスタ側，スレーブ側共に，P92 出力端子に LED1 が接続，P93 出力端子に LED2 が接続されています。
- (8) P92 は，大電流ポートです。
- (9) オンボード書き込みの構成例を図 1 に示します。

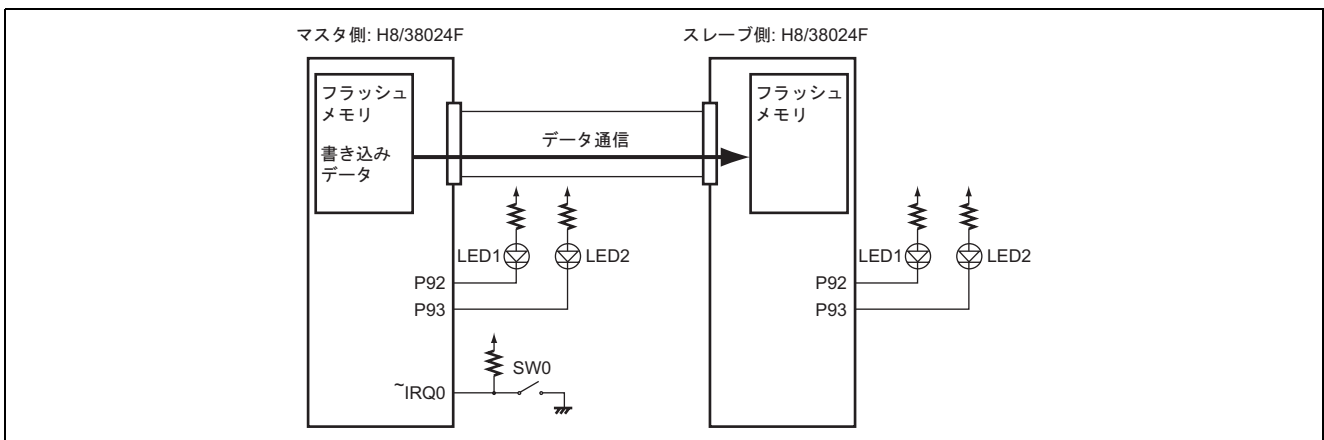


図 1 オンボード書き込み構成例

2. 詳細仕様説明

2.1 オンボードプログラミング動作条件

- デバイス：HD64F38024(H8/38024F)
- CPU 動作：ユーザモード
- 動作電圧：3.3V
- 動作周波数：5MHz

2.2 オンボードプログラミングモード

- ユーザモード
書き込み / 消去制御プログラム, RAM 転送プログラムをブートモードもしくはライターモードで予め書き込んでいることを前提条件とします。

2.3 書き込み方式

- 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, 8ビットパラレル通信を使用します。マスタを転送元とし, スレーブを受信側とします。

2.4 書き込み手順のフローチャート

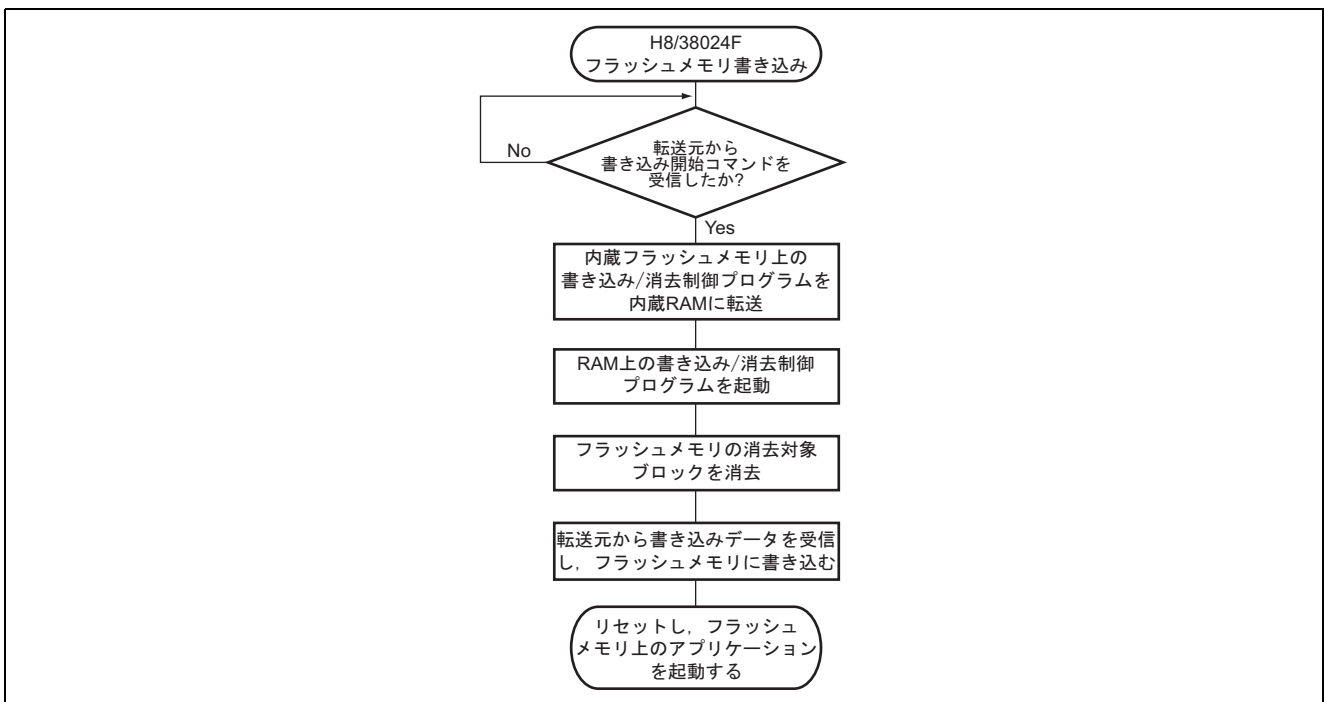


図2 ユーザモード書き込み手順

2.5 マスタ-スレーブ間の接続図

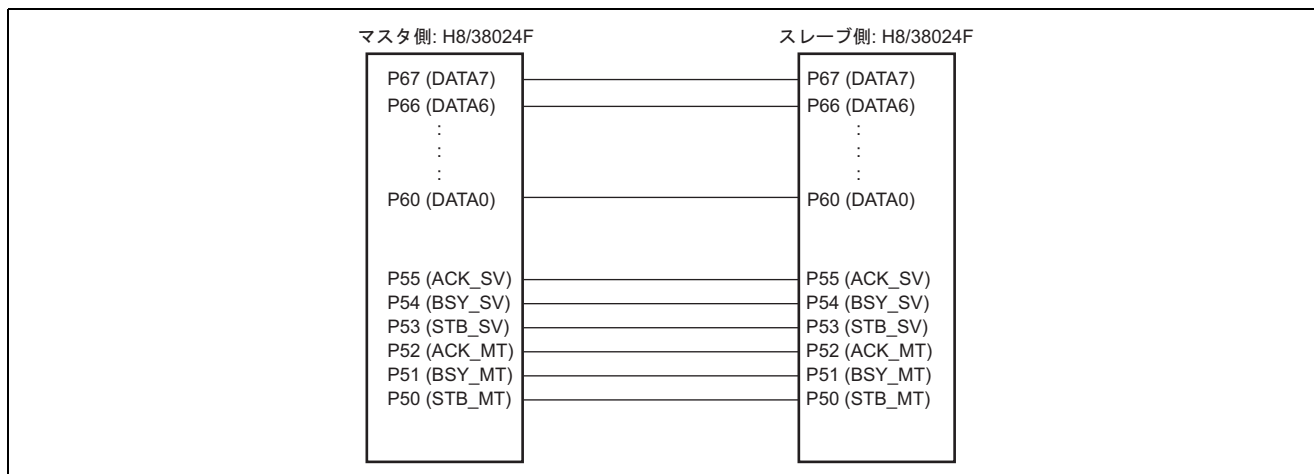


図 3 マスタ-スレーブ間の接続図

2.6 通信仕様

表 1 通信仕様

通信方式	8ビットパラレル通信
データビット	8
制御線数	6

2.7 通信コマンド

表 2 通信コマンド

通信コマンド	説明
0x00	通信正常(コマンド名: OK コマンド)
0x01	通信異常(コマンド名: NG コマンド)
0x11	送信開始要求
0x55	書き込み開始コマンド
0x77	消去コマンド
0x88	書き込みコマンド

2.8 信号線一覧

- 8 ビットパラレル通信で使用する信号線の名称とピン配置は、以下のようになります。

表 3 信号線一覧

信号名	対応ポート	信号の方向	機能
DATA7 ~ DATA0	P67 ~ P60	マスタ側 スレーブ側	データの信号線
ACK_SV	P55	スレーブ側	スレーブ側の受信許可 ACK_SV=0 : スレーブ側は受信可能である。 ACK_SV=1 : スレーブ側は受信不可である。
BSY_SV	P54	スレーブ側	スレーブ側の処理状態 BSY_SV=0 : スレーブ側の内部処理が終了した。 BSY_SV=1 : スレーブ側が内部処理をしている。
STB_SV	P53	スレーブ側	スレーブ側の信号状態 STB_SV=0 : スレーブ側からデータ送信していない。 STB_SV=1 : スレーブ側からデータ送信している。
ACK_MT	P52	マスタ側	マスタ側の受信許可 ACK_MT=0 : マスタ側は受信可能である。 ACK_MT=1 : マスタ側は受信不可である。
BSY_MT	P51	マスタ側	マスタ側の処理状態 BSY_MT=0 : マスタ側の内部処理が終了した。 BSY_MT=1 : マスタ側が内部処理をしている。
STB_MT	P50	マスタ側	マスタ側の信号状態 STB_MT=0 : マスタ側からデータ送信していない。 STB_MT=1 : マスタ側からデータ送信している。

2.9 メモリ配置

- H8/38024F のフラッシュメモリ消去ブロックを表 4 に示します。

表 4 フラッシュメモリ消去ブロック

ブロック(サイズ)	アドレス
EB0 (1k バイト)	0x0000 ~ 0x03FF
EB1 (1k バイト)	0x0400 ~ 0x07FF
EB2 (1k バイト)	0x0800 ~ 0x0BFF
EB3 (1k バイト)	0x0C00 ~ 0x0FFF
EB4 (28k バイト)	0x1000 ~ 0x7FFF

- H8/38024F の通常動作時，書き込み動作時のメモリマップを図4に示します。

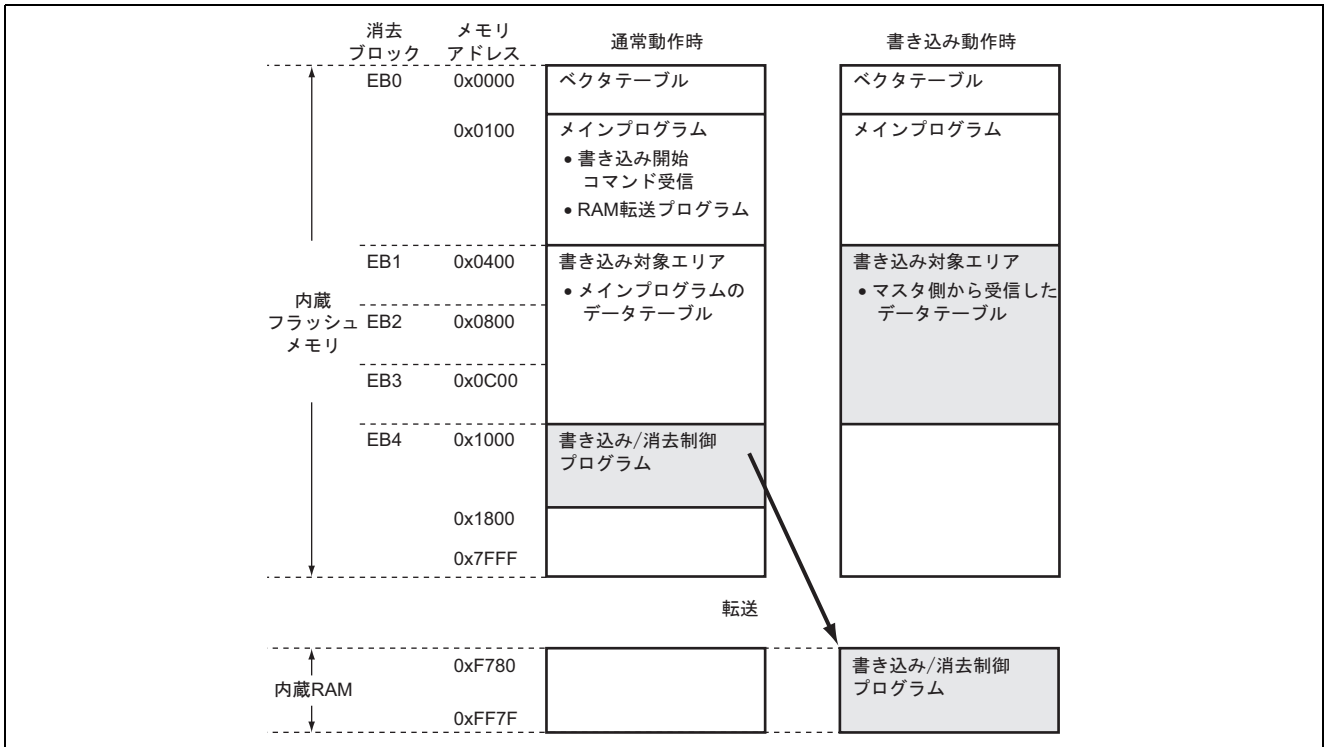


図4 メモリマップ

2.10 コンパイル，リンクオプション

(1) コンパイルオプション

今回のプログラム例で使用するコンパイルオプションを表5に示します。

表5 コンパイルオプション

項目	コマンドライン形式	オプション	指定内容
CPU / 動作モード	-cpu=	300	H8/300 オブジェクトを作成する
オブジェクト形式	-code=	machinecode	マシンコードを出力
最適化内容	-speed=	register	関数の入口 / 出口でレジスタを退避 / 回復する コードを実行時ルーチンを使用せずに PUSH , POP 命令で展開する。

(2) バッチファイル例

今回のプログラム例で使用したバッチファイルを下記に示します。

(2) スレーブ側バッチファイル

```
CH38 -cpu=300 -code=machinecode -speed=register main.c (main.c : スレーブ側通常プログラム)
CH38 -cpu=300 -code=machinecode -speed=register slvf.c (slvf.c : スレーブ側書き込み / 消去制御プログラム)
ASM38 init.src -cpu=300 (init.src : スタックポインタの設定プログラム)
LNK -sub=link.sub
CNVS slvf.abs
```

link.sub の内容

```
output slvf.abs
print slvf.map
input ..\init.obj main.obj slvf.obj lcdt.obj
lib C:\Hewlett-Packard\Tools\Hitachi\H8\3_0a_0\lib\c38reg.lib
start CV1(00000),P(00100),DLCDDT1(00400),DLCDDT2(00800),DLCDDT3(00FFA)
start FZTAT,PFZTAT,DFZTAT,FZEND(01000),RAM,PRAM,DRAM,B(0F780)
exit
```

(b) マスタ側バッチファイル

```
CH38 -cpu=300 -code=machinecode mst.c (mst.c : マスタ側プログラム)
ASM38 init.src -cpu=300 (init.src : スタックポインタの設定プログラム)
LNK -sub=link.sub
CNVS mst.abs
```

link.sub の内容

```
output mst.abs
print mst.map
input ..\init.obj mst.obj lcdt.obj
lib C:\Hewlett-Packard\Tools\Hitachi\H8\3_0a_0\lib\c38reg.lib
start CV1(00000),CV2(00008),P(01000)
start D(00100),DLCDDT1(00400),DLCDDT2(00800),DLCDDT3(00FFA)
start B(0FB80)
exit
```

(3) 使用コンパイラバージョン

今回のプログラム例で使用した開発環境バージョンを表 6 に示します。

表 6 開発環境バージョン

ソフト名	使用バージョン
C コンパイラ	H8S,H8/300 SERIES C/C++ Compiler Ver3.0A
クロスアセンブラ	H8S,H8/300 SERIES CROSS ASSEMBLER Ver3.0B
リンケージエディタ	H SERIES LINKAGE EDITOR Ver.6.0D
オブジェクトコンバータ	H SERIES SYSROF STYPE OBJECT CONVERTER Ver.2.0A

3. 動作概要

3.1 通常動作

- (1) スレーブ側のフラッシュメモリ上に書き込み / 消去制御プログラムを書き込んでおきます。
- (2) 通常アプリケーションは、通常フラッシュメモリ上のデータテーブルをアクセスするものとし、データテーブルは、マスタ側から受信し、書き込まれます。
- (3) 書き込み開始コマンド受信プログラム、RAM 転送プログラムは、あらかじめスレーブ側のフラッシュメモリ上に書き込んでおきます。
- (4) マスタ、スレーブ間のデータ通信は、8ビットパラレル通信を使用します。

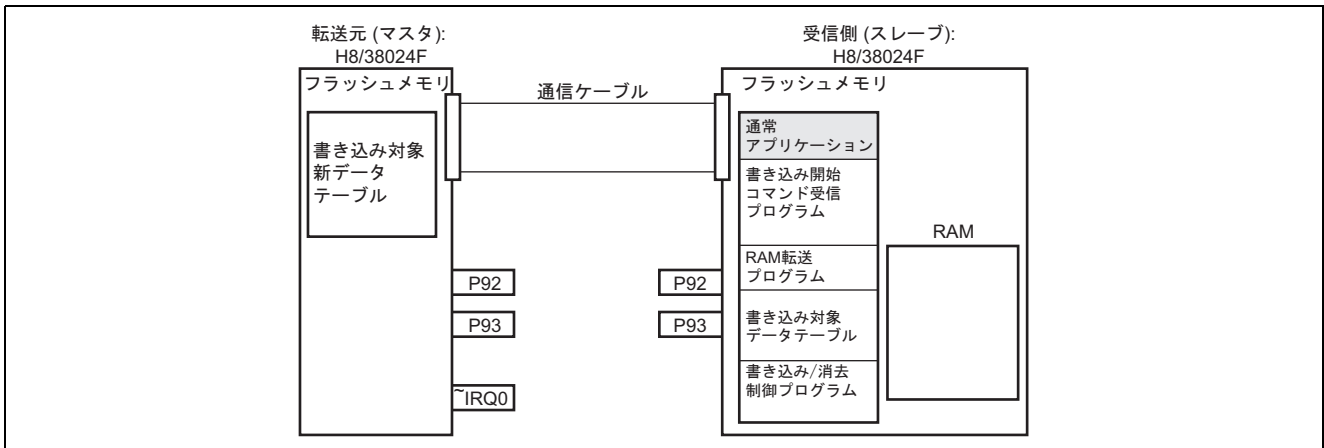


図5 通常動作

3.2 オンボード書き込み準備

- (1) マスタ側の~IRQ0 端子に Low トリガが入力されると、マスタ側から書き込み開始コマンド 0x55 を送信します。
- (2) このとき、マスタ側の P92 は High レベル、P93 は Low レベルを出力します。

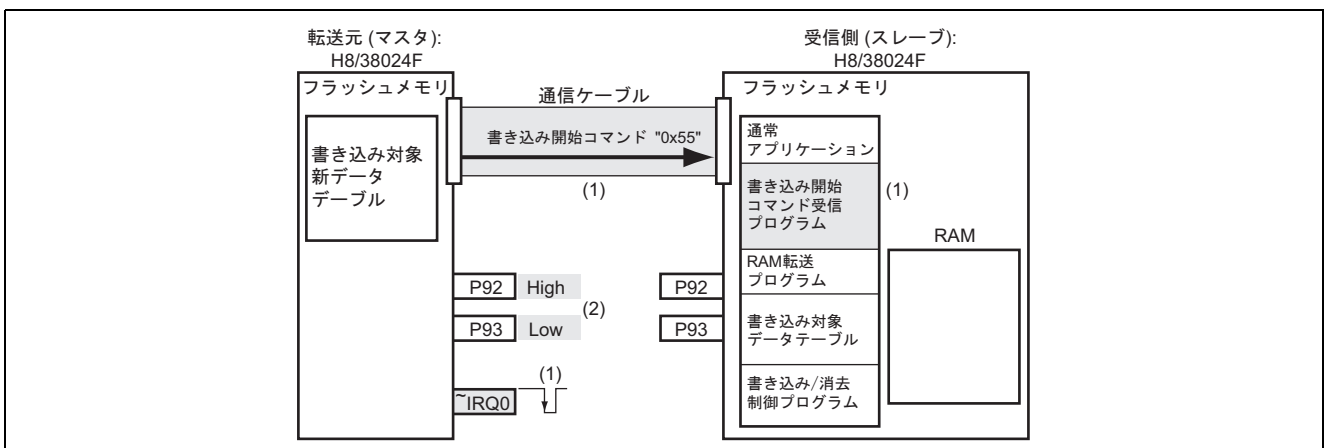


図6 オンボード書き込み準備

3.3 オンボード書き込み開始

- (1) スレーブ側は 0x55 を受信すると、RAM 転送プログラムを起動し、書き込み / 消去制御プログラムを内蔵 RAM に転送します。
- (2) このとき、スレーブ側の P92 は High レベル、P93 は Low レベルを出力します。

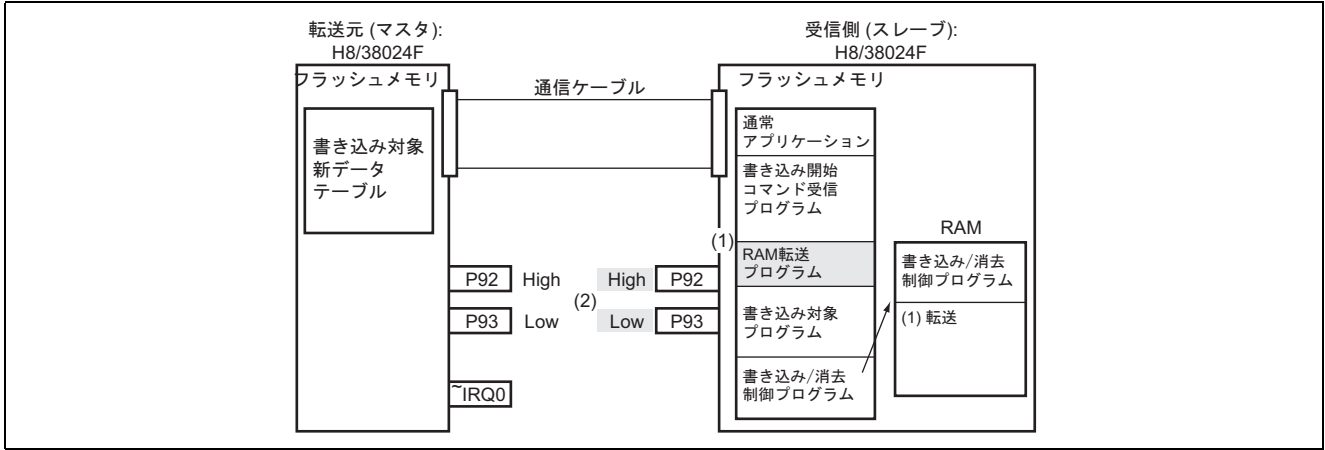


図7 オンボード書き込み開始

3.4 書き込み / 消去制御プログラム起動

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

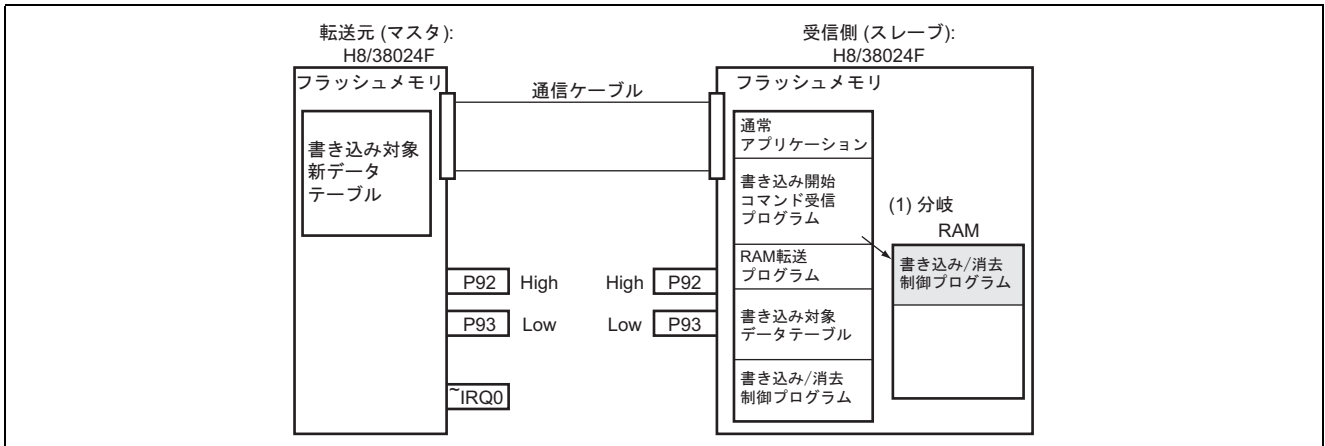


図8 書き込み / 消去制御プログラム起動

3.5 フラッシュメモリ消去

- (1) マスタ側から消去コマンド 0x77 を受信します。
- (2) 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。

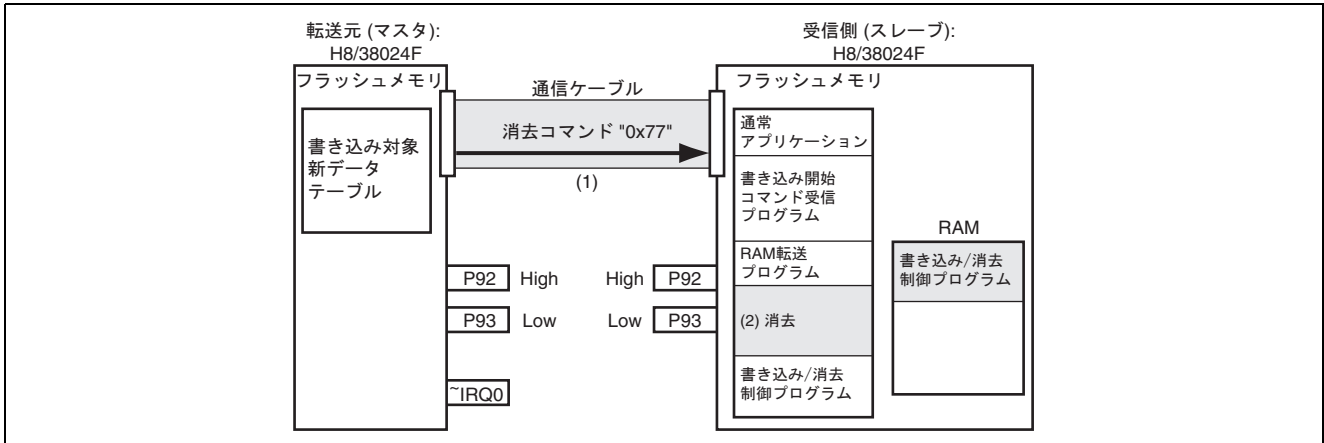


図9 フラッシュメモリ消去

3.6 フラッシュメモリ書き込み

- (1) 転送元から書き込みコマンド 0x88 を受信します。
- (2) 書き込み / 消去制御プログラムは、転送元から新データテーブルを受信し、フラッシュメモリに書き込みます。
- (3) 書き込み終了後、マスタ、スレーブ共に P92 は Low、P93 は High を出力します。

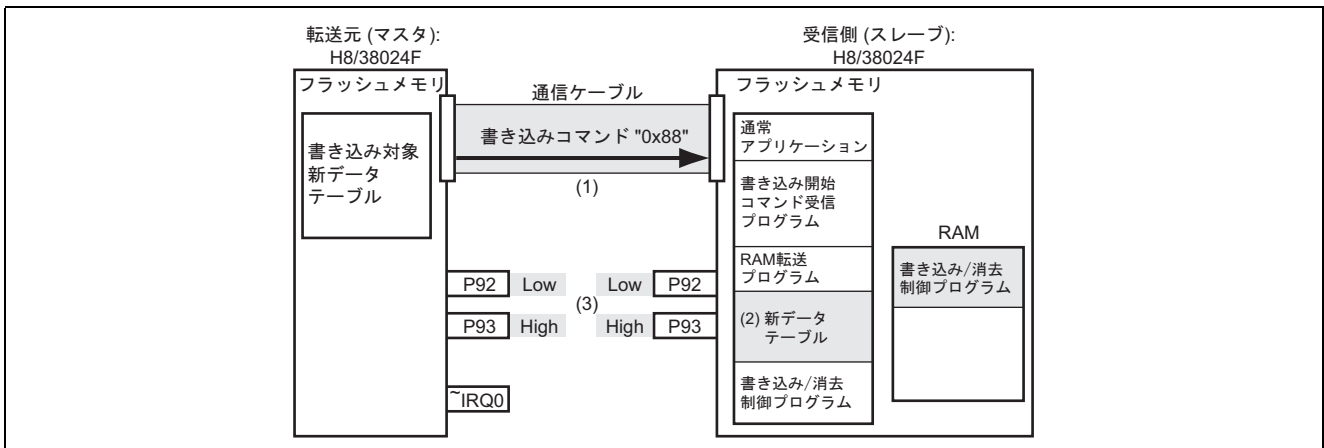


図10 フラッシュメモリ書き込み

3.7 プログラム起動

(1) リセットし、新データテーブルをアクセスする通常アプリケーションを起動します。

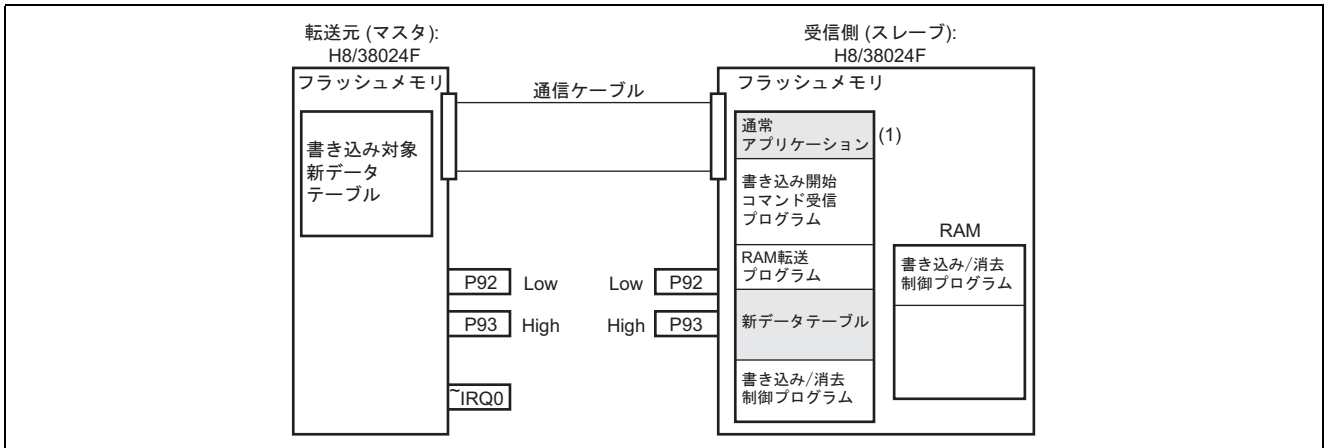


図 11 プログラム起動

4. シーケンス図

(1) 通常時

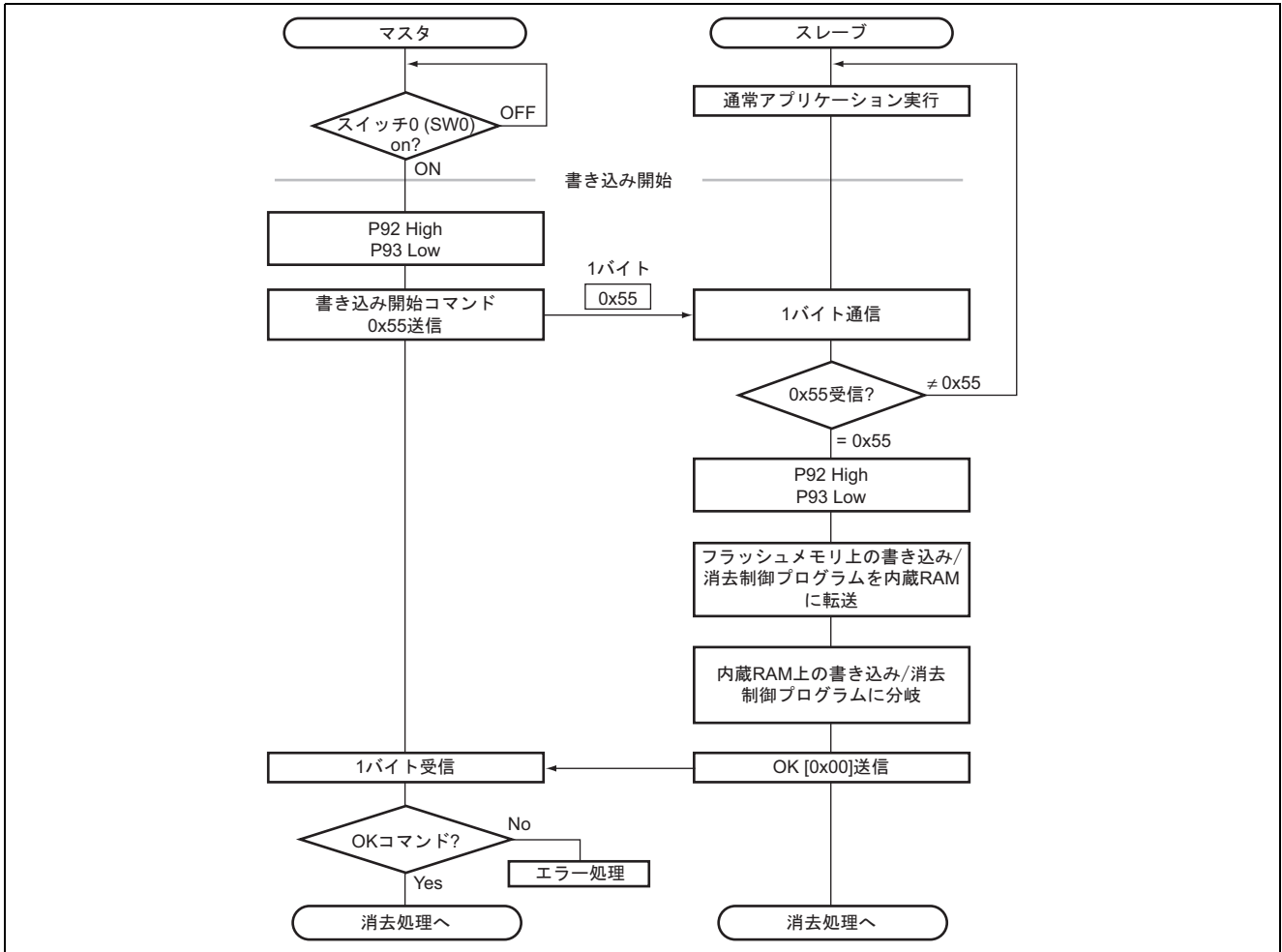


図 12 通常動作

(2) 消去処理

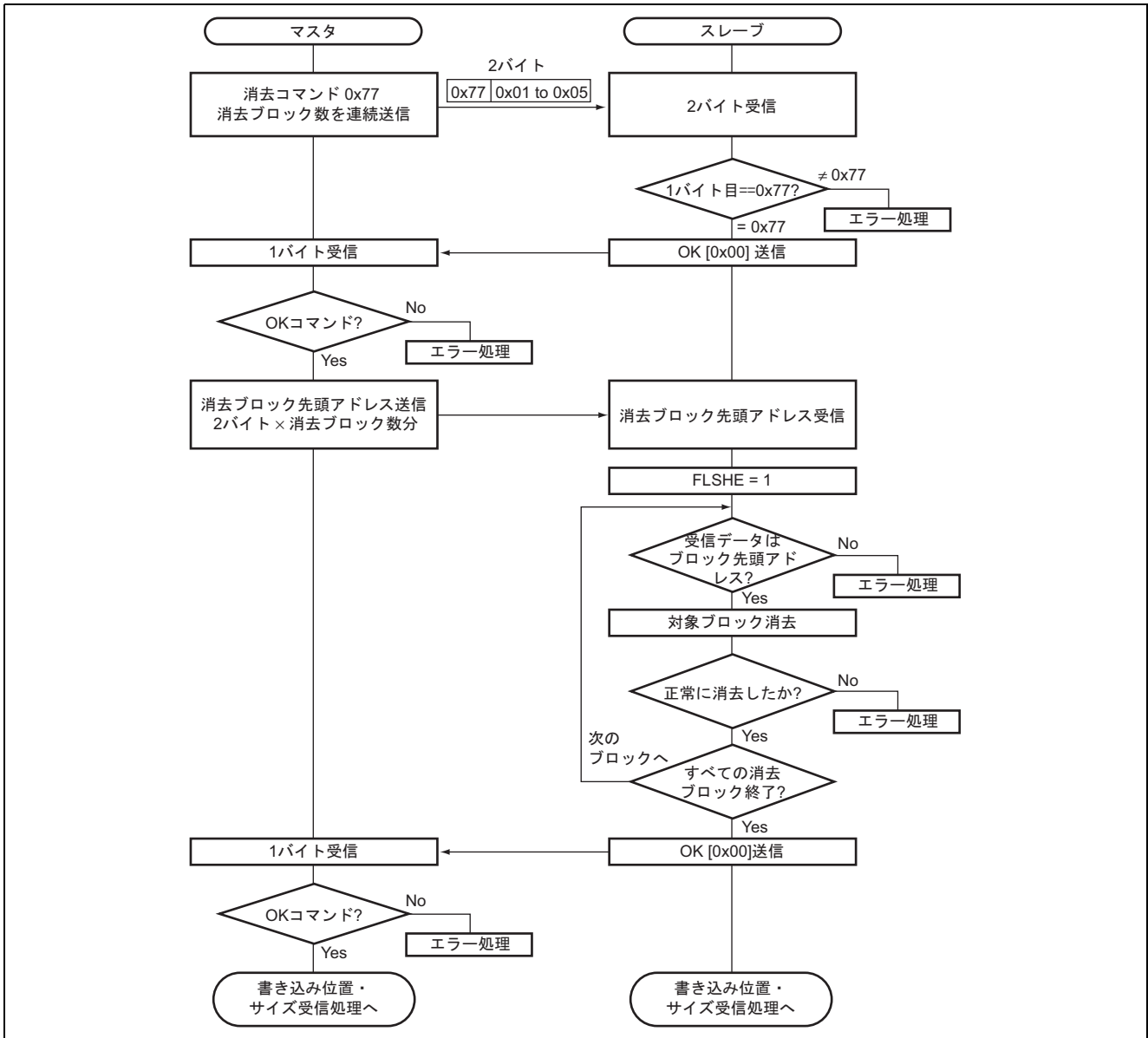


図 13 消去処理

(3) 書き込み位置・サイズ受信処理

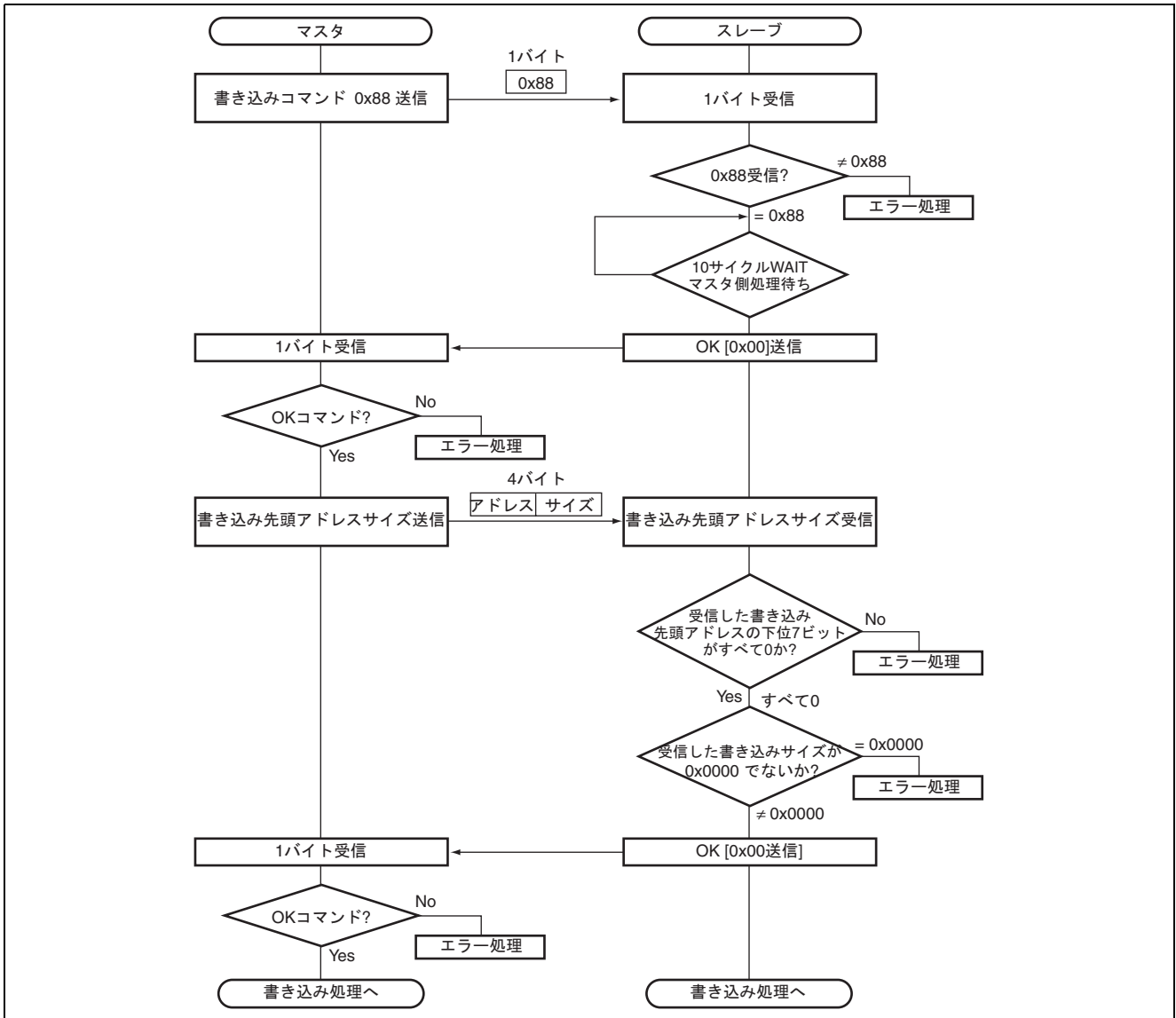


図 14 書き込み位置・サイズ受信処理

(4) 書き込み処理

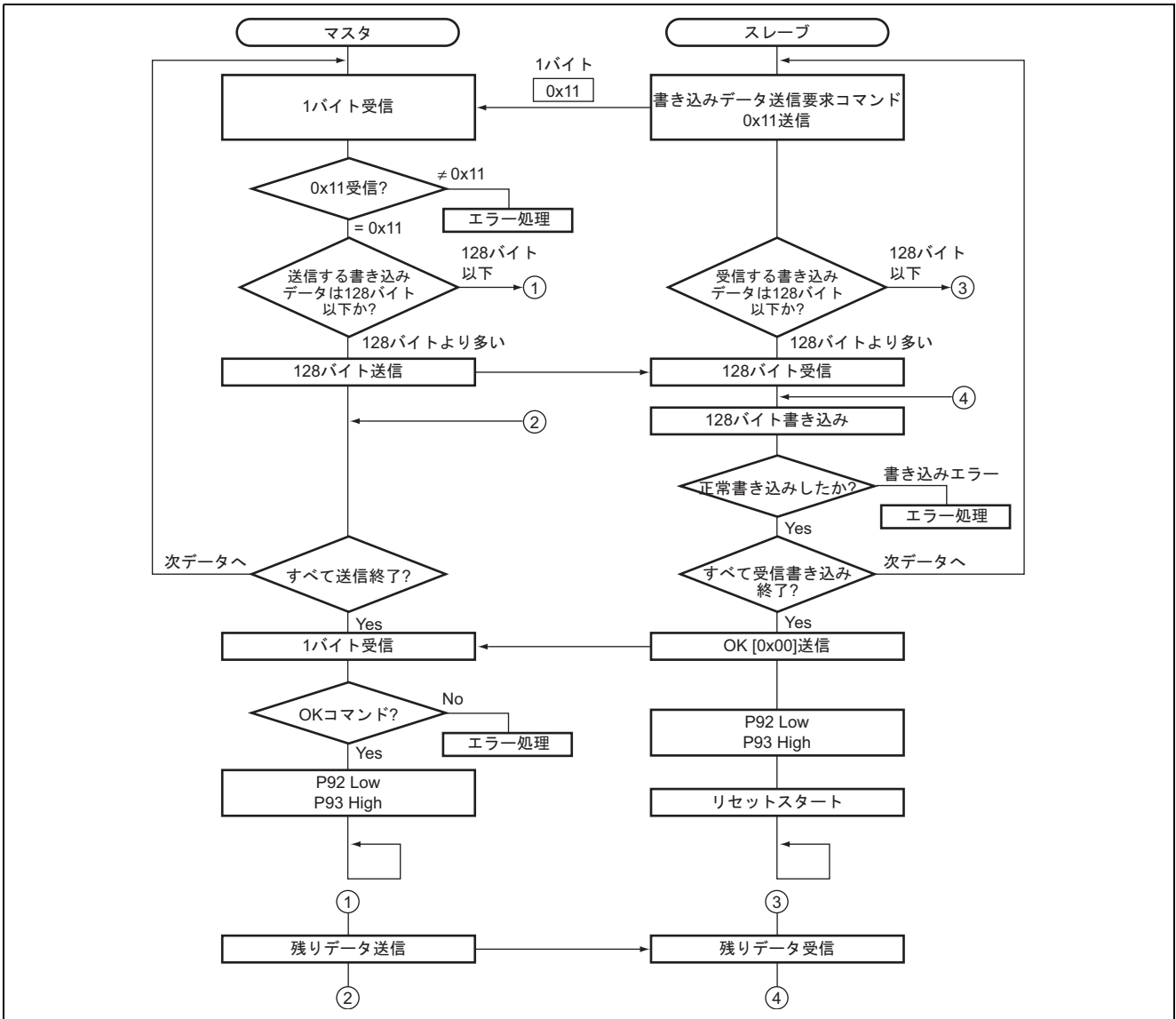


図 15 書き込み処理

(5) エラー処理

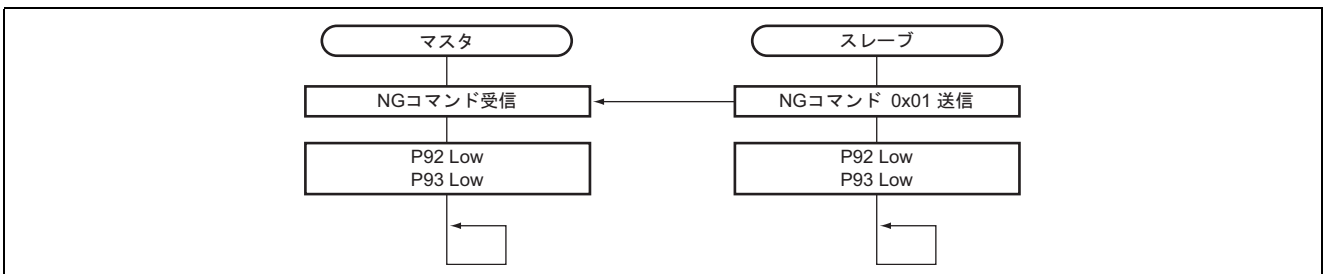


図 16 エラー処理

(6) マスタ側からスレーブ側への1バイト通信処理

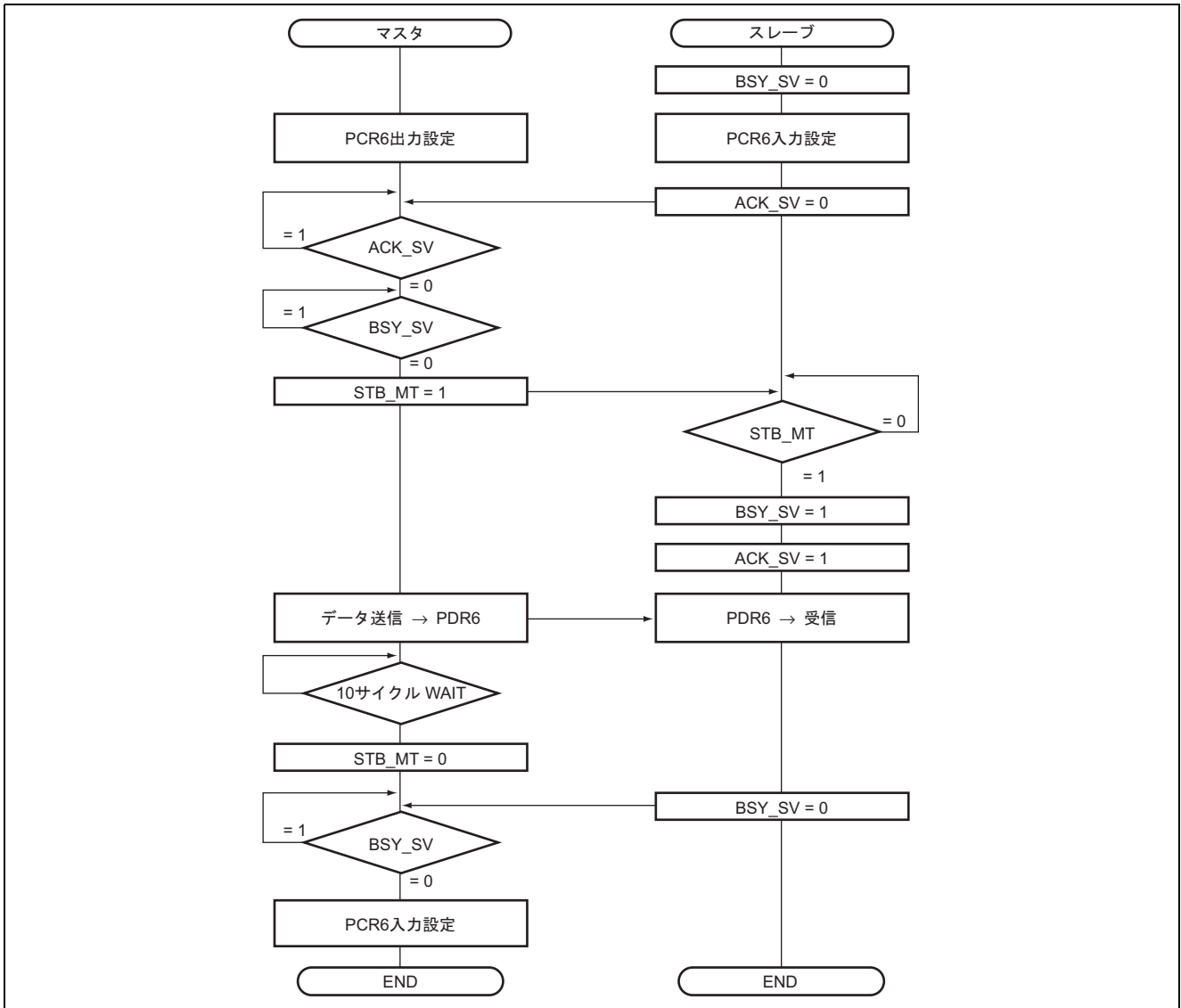


図 17 マスタ側からスレーブ側への1バイト通信処理

(7) スレーブ側からマスタ側への1バイト通信処理

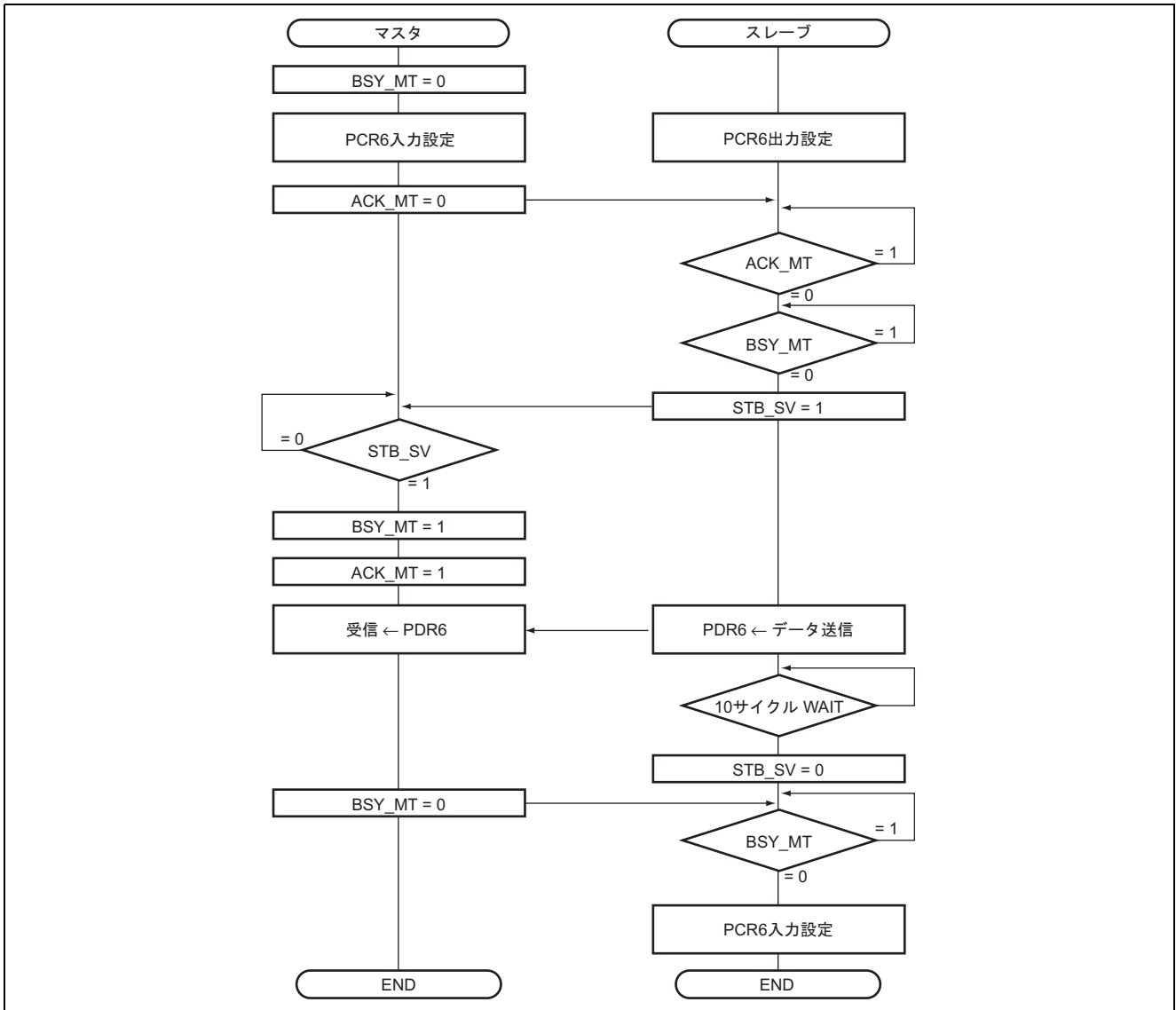


図 18 スレーブ側からマスタ側への1バイト通信処理

5. スレーブ側通常プログラム

5.1 階層構造

フラッシュメモリ上で実行するスレーブ側通常プログラム(スレーブ側通常プログラム)は、ユーザアプリケーションプログラム(通常アプリケーション)の実施、書き込み開始コマンド受信、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 に転送する処理を行います。スレーブ側通常プログラムで使用するルーチンの階層構造を図 19 に示します。

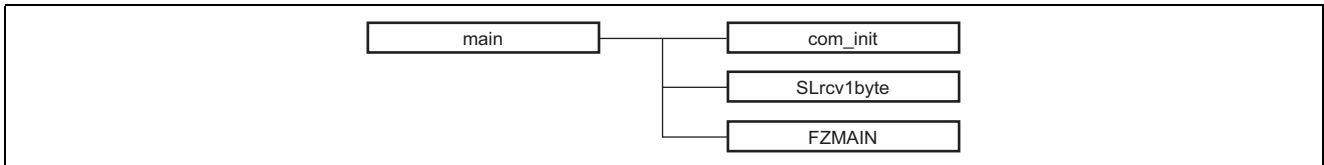


図 19 スレーブ側通常プログラム

5.2 関数一覧

表 7 スレーブ側通常プログラム関数一覧

関数名	概要
main	通常アプリケーションの実施，書き込み開始コマンド受信，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
com_init	通信設定の初期化
SLrcv1byte	データを 1 バイト受信する
FZMAIN	フラッシュメモリの書き込み/消去制御プログラム

5.3 関数説明

(1) main()関数

(a) 仕様

void main(void)

(b) 動作説明

- ユーザアプリケーションプログラム(通常アプリケーション)の実施
- 書き込み開始コマンド受信処理
- 書き込み/消去制御プログラムの RAM 転送処理
- 書き込み/消去制御プログラムへの分岐

(c) 引数の説明

- 入力値：なし
- 出力値：なし

(d) グローバル変数

なし

(e) 使用サブルーチン

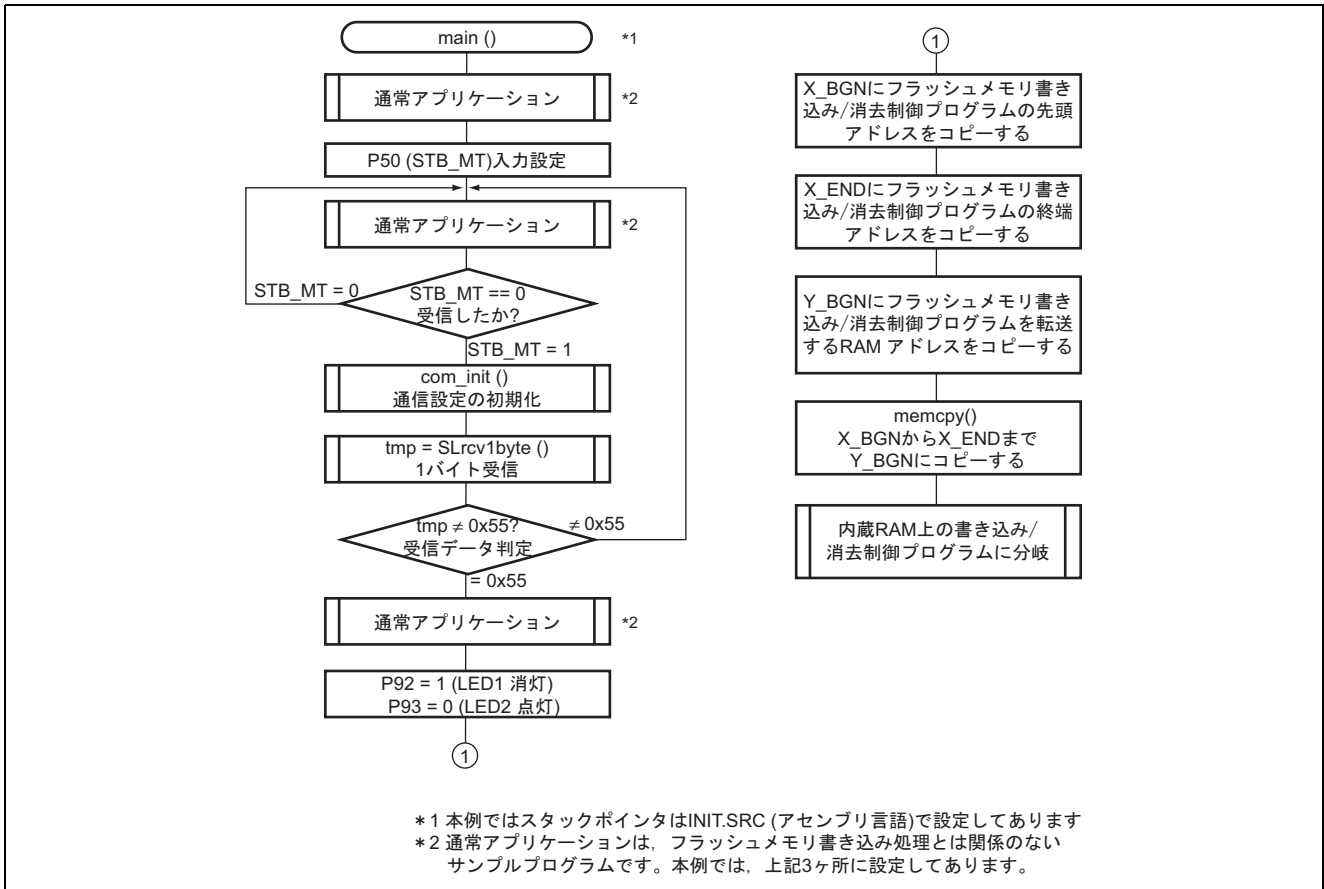
- com_init()：通信設定の初期化
- SLrcv1byte()：データを 1 バイト受信する
- FZMAIN()：書き込み/消去制御プログラムへの分岐

(f) 使用内部レジスタ

表 8 main()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
PDR9	P93	ポートデータレジスタ 9(ポートデータレジスタ 93) : P93=0 のとき, P93 端子の出力レベルは"Low" : P93=1 のとき, P93 端子の出力レベルは"High"	0xFFDC ビット 3	0
	P92	ポートデータレジスタ 9(ポートデータレジスタ 92) : P92=0 のとき, P92 端子の出力レベルは"Low" : P92=1 のとき, P92 端子の出力レベルは"High"	0xFFDC ビット 2	1
LPCR		LCD ポートコントロールレジスタ サンプルの通常アプリケーションで使用する	0xFFC0	—
LCR		LCD コントロールレジスタ サンプルの通常アプリケーションで使用する	0xFFC1	—
LCR2		LCD コントロールレジスタ 2 サンプルの通常アプリケーションで使用する	0xFFC2	—
LCDRAM		LCD RAM サンプルの通常アプリケーションで使用する	0xF740 ~ 0xF74F	—
PDR3	P37	ポートデータレジスタ 3(ポートデータレジスタ 37) サンプルの通常アプリケーションで使用する	0xFFDC ビット 7	—
PDR5	P50	ポートデータレジスタ 5(ポートデータレジスタ 50) STB_MT として用いるポート : STB_MT(P50)=0 マスタ側からデータ送信していない : STB_MT(P50)=1 マスタ側からデータ送信している	0xFFD8 ビット 0	—

(g) フローチャート



(2) com_init()関数

(a) 仕様

void com_init(void)

(b) 動作説明

- 8ビットパラレル通信の初期化

(c) 引数の説明

- 入力値：なし
- 出力値：なし

(d) グローバル変数

なし

(e) 使用サブルーチン

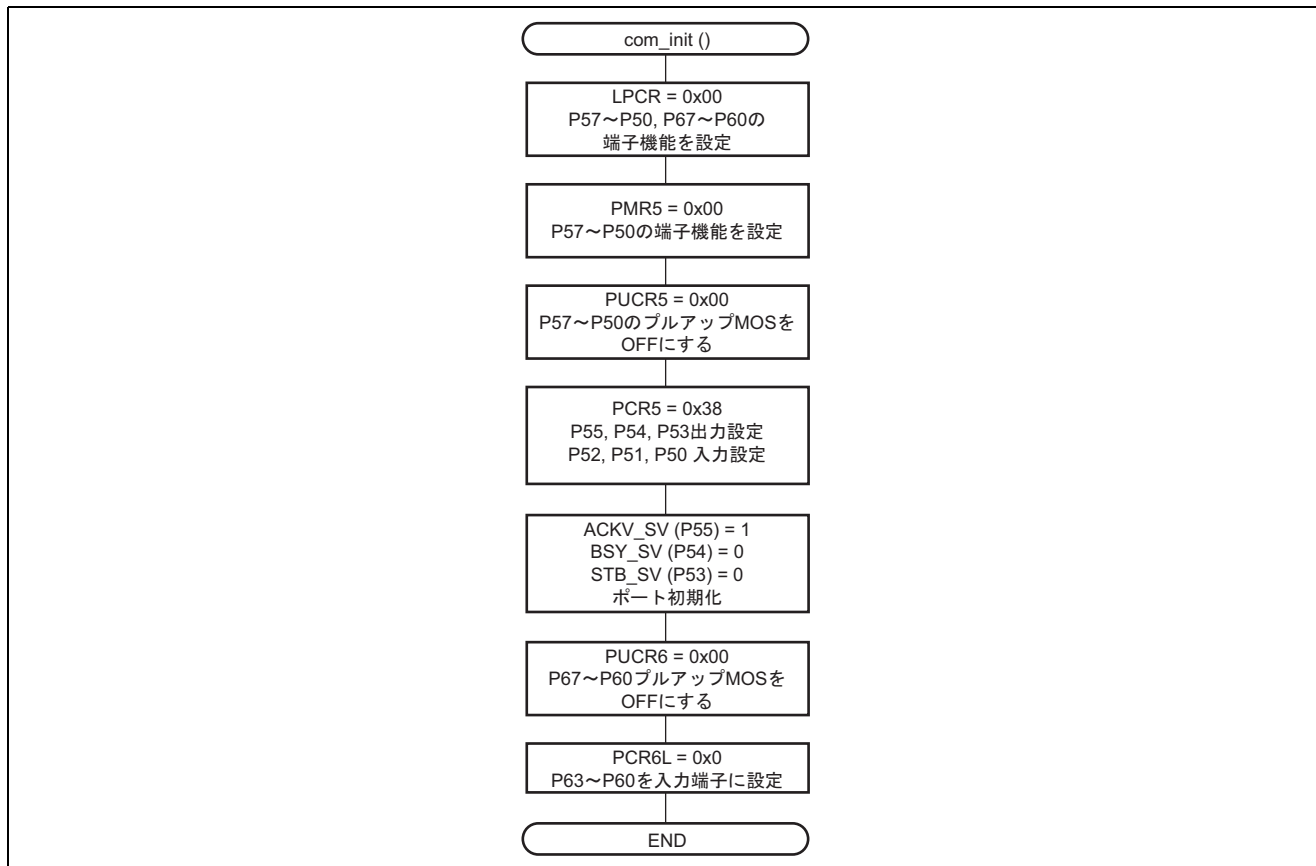
なし

(f) 使用内部レジスタ

表 9 com_init()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
LPCR	SGS3	LCDポートコントロールレジスタ(セグメントドライバ選択) : SGS3=0, SGS2=0, SGS1=0, SGS0=0 のとき, SEG32 ~ SEG1 端子は, ポートとして機能	0xFFC0 ビット 3 ビット 2 ビット 1 ビット 0	SGS3=0 SGS2=0 SGS1=0 SGS0=0
	SGS2			
	SGS1			
	SGS0			
PMR5		ポートモードレジスタ 5 : PMR5=0x00 のとき, ~WKP7 ~ ~WKP0 入力端子は, P57 ~ P50 端子として機能	0xFFCC	0x00
PDR5	P55	ポートデータレジスタ 5(ポートデータレジスタ 55) ACK_SV として用いるポート : ACK_SV(P55)=0 スレーブ側は受信可能である : ACK_SV(P55)=1 スレーブ側は受信不可である	0xFFD8 ビット 5	1
	P54	ポートデータレジスタ 5(ポートデータレジスタ 54) BSY_SV として用いるポート : BSY_SV(P54)=0 スレーブ側の内部処理が終了した : BSY_SV(P54)=1 スレーブ側が内部処理をしている	0xFFD8 ビット 4	0
	P53	ポートデータレジスタ 5(ポートデータレジスタ 53) STB_SV として用いるポート : STB_SV(P53)=0 スレーブ側からデータ送信していない : STB_SV(P53)=1 スレーブ側からデータ送信している	0xFFD8 ビット 3	0
PUCR5		ポートプルアップコントロールレジスタ 5 : PUCR5=0x00 のとき, P57 ~ P50 のプルアップ MOS を OFF に 設定	0xFFE2	0x00
PUCR6		ポートプルアップコントロールレジスタ 6 : PUCR6=0x00 のとき, P67 ~ P60 のプルアップ MOS を OFF に 設定	0xFFE3	0x00
PCR5		ポートコントロールレジスタ 5 : PCR5=0x38 のとき, P55, P54, P53 を出力端子に, P52, P51, P50 を入力端子に設定	0xFFE8	0x38
PCR6		ポートコントロールレジスタ 6 : PCR6=0x00 のとき, P67 ~ P60 を入力端子に設定	0xFFE9	0x00

(g) フローチャート



(3) SLrcv1byte()関数

(a) 仕様

unsigned char SLrcv1byte(void)

(b) 動作説明

- 8ビットパラレルデータを1バイト受信

(c) 引数の説明

- 入力値：なし
- 出力値：1バイト受信データ

(d) グローバル変数

なし

(e) 使用サブルーチン

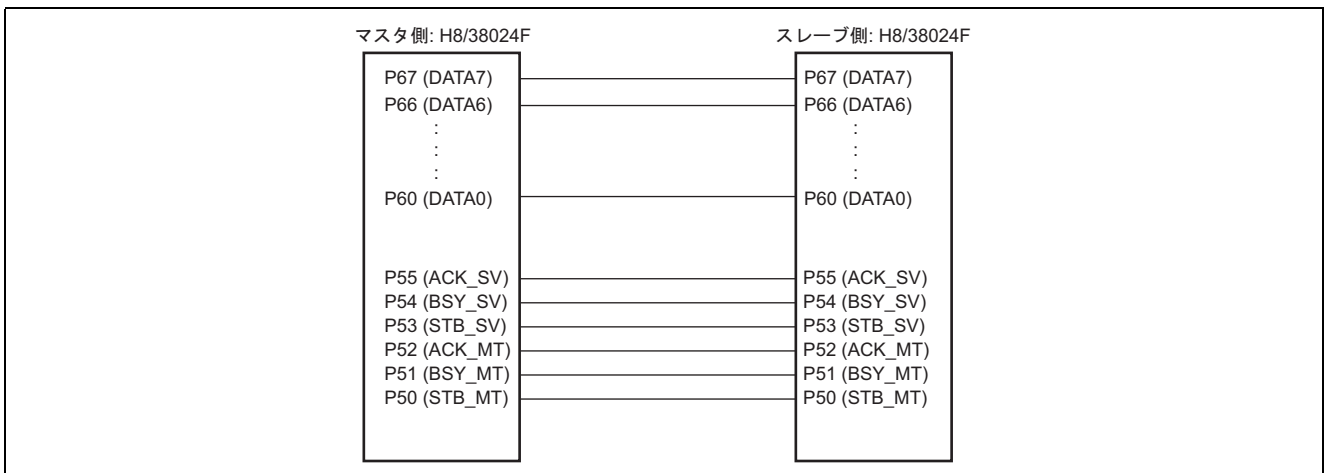
なし

(f) 使用内部レジスタ

表 10 SLrcv1byte()関数の使用レジスタ

レジスタ名	機能	アドレス	設定値
PDR5	P55 ポートデータレジスタ 5(ポートデータレジスタ 55) ACK_SV として用いるポート : ACK_SV(P55)=0 スレーブ側は受信可能である : ACK_SV(P55)=1 スレーブ側は受信不可である	0xFFD8 ビット 5	0
	P54 ポートデータレジスタ 5(ポートデータレジスタ 54) BSY_SV として用いるポート : BSY_SV(P54)=0 スレーブ側の内部処理が終了した : BSY_SV(P54)=1 スレーブ側が内部処理をしている	0xFFD8 ビット 4	1
	P50 ポートデータレジスタ 5(ポートデータレジスタ 50) STB_MT として用いるポート : STB_MT(P50)=0 マスタ側からデータ送信していない : STB_MT(P50)=1 マスタ側からデータ送信している	0xFFD8 ビット 0	-
PDR6	ポートデータレジスタ 6 : 受信データを格納する 8 ビットのレジスタ	0xFFD9	-

(g) フローチャート



(4) FZMAIN()関数

書き込み / 消去制御プログラムのメインルーチンを呼び出す

6. スレーブ側書き込み / 消去制御プログラム

6.1 階層構造

書き込み / 消去制御プログラムは、消去ブロック単位の消去、フラッシュメモリ書き込みデータの受信、フラッシュメモリへの書き込みを行います。書き込み / 消去制御プログラムで使用するルーチンの階層構造を図 20 に示します。FZMAIN()関数を除くサブルーチンは、通信処理とフラッシュメモリ書き込み / 消去処理に分けられます。

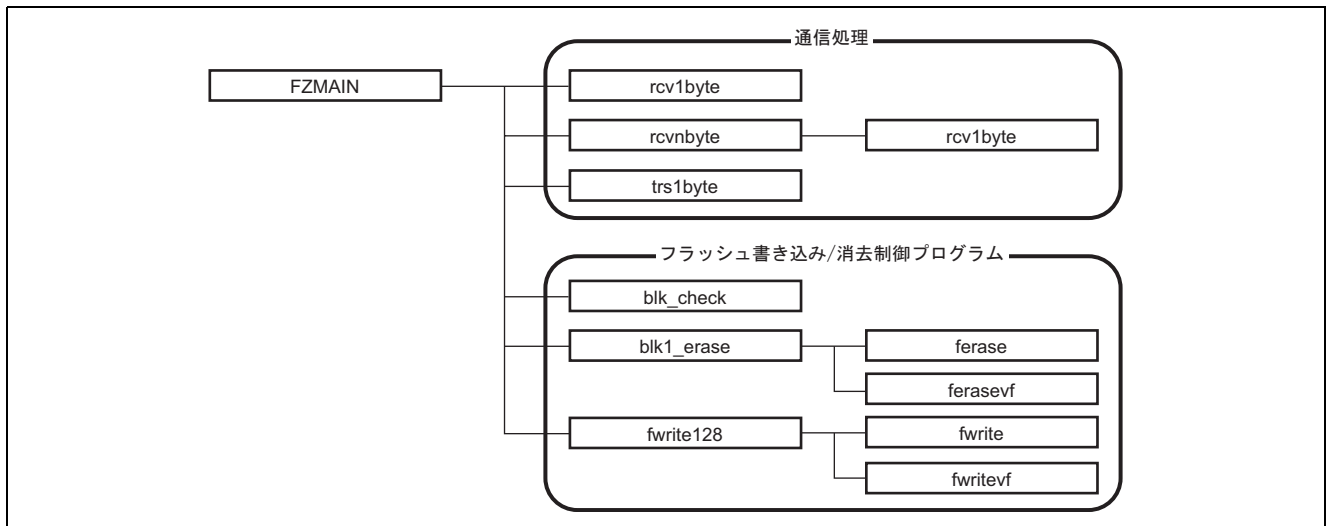


図 20 書き込み / 消去制御プログラム

6.2 関数一覧

表 11 書き込み / 消去制御プログラム関数一覧

関数名	概要
FZMAIN	書き込み / 消去制御プログラムのメインルーチン
rcv1byte	データを 1 バイト受信する
rcvnbyte	データを n バイト受信する
trs1byte	データを 1 バイト送信する
blk_check	消去先頭アドレスから消去対象ブロック番号を判定する
blk1_erase	フラッシュメモリの指定ブロックを消去する
ferase	指定ブロックの消去
ferasevf	指定ブロックの消去ベリファイ
fwrite128	128 バイトの書き込み, ベリファイ
fwrite	対象アドレスの書き込み
fwritevf	対象アドレスのベリファイ, 再書き込みデータの作成

6.3 定数一覧

表 12 定数一覧

定数名	値	内容
OK	0x00	正常時の戻り値
NG	0x01	異常時の戻り値
WNG	0x02	書き込みエラー
MAXBLK1	0x0A	フラッシュメモリの全ブロック数(5)×2
WDT_ERASE	0x00	フラッシュメモリ消去時の WDT カウント
WDT_WRITE	0xFB	フラッシュメモリ書き込み時の WDT カウント
OW_COUNT	0x06	再書き込み回数
WLOOP1	$1 \times \text{MHZ} / \text{KEISU} + 1 = 0x01$	WAIT 文実行回数 1 μsWAIT
WLOOP2	$2 \times \text{MHZ} / \text{KEISU} + 1 = 0x02$	WAIT 文実行回数 2 μsWAIT
WLOOP4	$4 \times \text{MHZ} / \text{KEISU} + 1 = 0x03$	WAIT 文実行回数 4 μsWAIT
WLOOP5	$5 \times \text{MHZ} / \text{KEISU} + 1 = 0x04$	WAIT 文実行回数 5 μsWAIT
WLOOP10	$10 \times \text{MHZ} / \text{KEISU} + 1 = 0x07$	WAIT 文実行回数 10 μsWAIT
WLOOP20	$20 \times \text{MHZ} / \text{KEISU} + 1 = 0x0D$	WAIT 文実行回数 20 μsWAIT
WLOOP50	$50 \times \text{MHZ} / \text{KEISU} + 1 = 0x20$	WAIT 文実行回数 50 μsWAIT
WLOOP100	$100 \times \text{MHZ} / \text{KEISU} + 1 = 0x3F$	WAIT 文実行回数 100 μsWAIT
TIME10	$10 \times \text{MHZ} / \text{KEISU} = 0x06$	WAIT 文実行回数 10 μsWAIT
TIME30	$30 \times \text{MHZ} / \text{KEISU} = 0x12$	WAIT 文実行回数 30 μsWAIT
TIME200	$200 \times \text{MHZ} / \text{KEISU} = 0x7D$	WAIT 文実行回数 200 μsWAIT
TIME10000	$10000 \times \text{MHZ} / \text{KEISU} = 0x186A$	WAIT 文実行回数 10msWAIT

MHZ:5 ……動作周波数 5MHz

KEISU:8 ……for 文で繰り返しを行ったときの 1 ループステップ数

6.4 通信処理関数の説明

(1) FZMAIN()関数

(a) 仕様

void FZMAIN(void)

(b) 動作説明

- フラッシュメモリの消去
- フラッシュメモリ書き込みデータの受信
- フラッシュメモリへのデータ書き込み
- 書き込み終了後にリセットスタートを実施

(c) 引数の説明

- 入力値：なし
- 出力値：なし

(d) グローバル変数

なし

(e) 使用サブルーチン

rcv1byte()：データを 1 バイト受信する

rcvnbyte()：データを n バイト受信する

trs1byte()：データを 1 バイト送信する

fwrite128()：128 バイトの書き込み，ベリファイを行う

blk_check()：消去先頭アドレスから消去対象ブロック番号を判定する

blk1_erase()：フラッシュメモリの指定ブロックを消去する

(f) 使用内部レジスタ

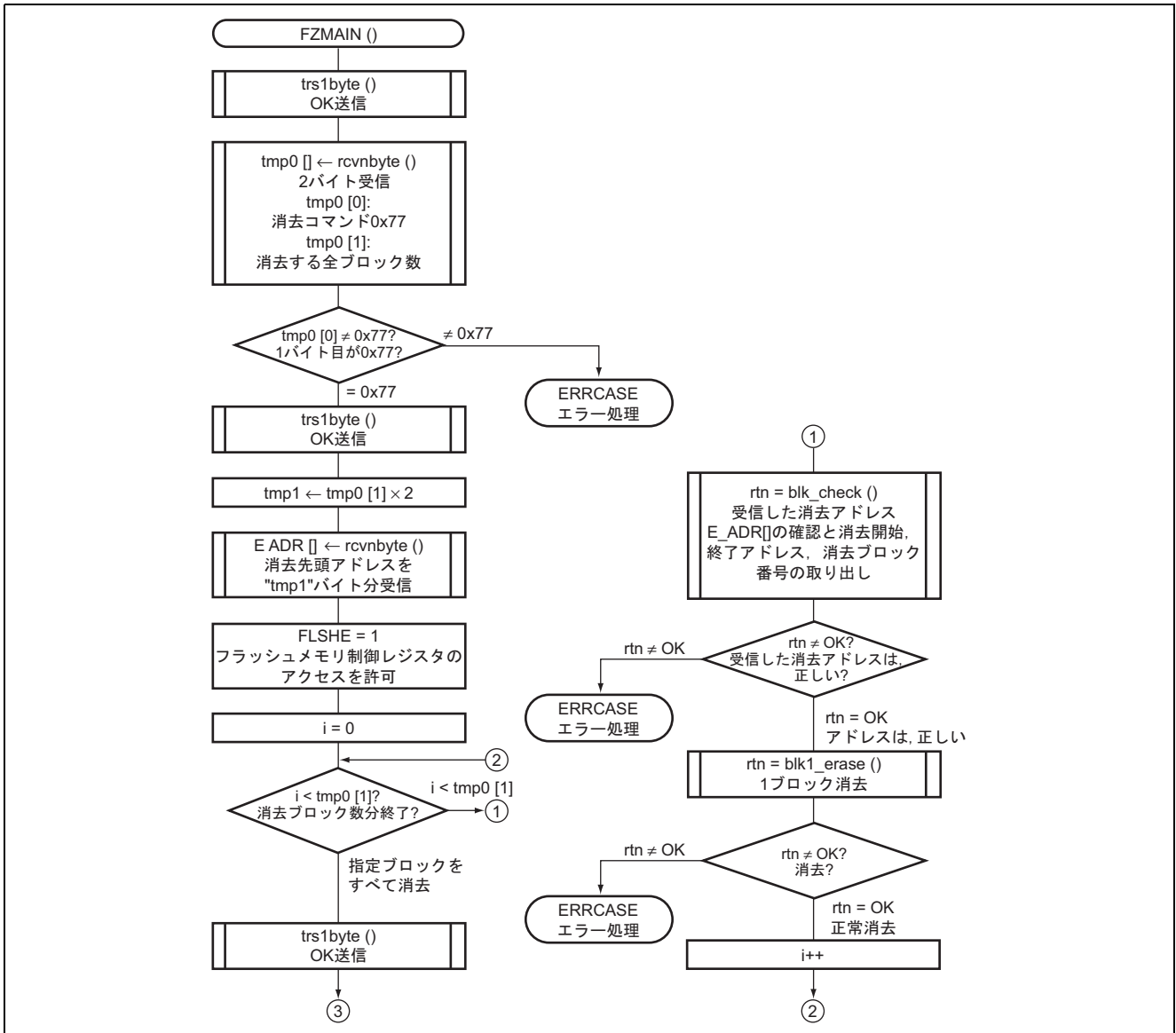
表 13 FZMAIN()関数の使用レジスタ

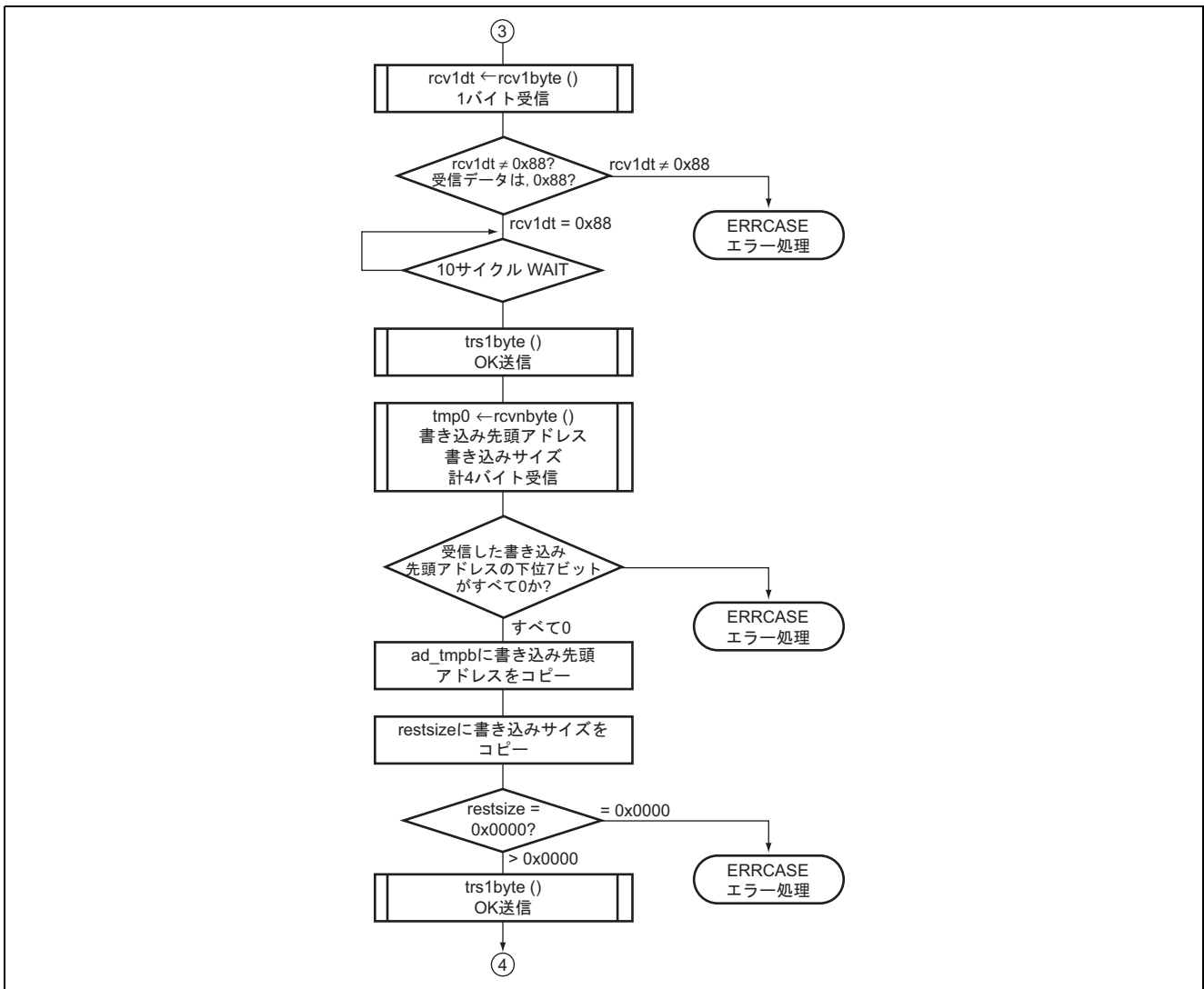
レジスタ名		機能	アドレス	設定値
FENR	FLSHE	フラッシュメモリエnableレジスタ (フラッシュメモリコントロールレジスタenable) ：FLSHE=0 のとき，フラッシュメモリ制御レジスタをアクセスできる ：FLSHE=1 のとき，フラッシュメモリ制御レジスタをアクセスできない	0xF02B ビット 7	1
TCSRW	B6WI	タイマコントロール/ステータスレジスタ W (ビット 6 書き込み禁止) ：B6WI=0 のとき，TCSRW ビット 6 への書き込みを許可 ：B6WI=1 のとき，TCSRW ビット 6 への書き込みを禁止	0xFFC0 ビット 7	0
	TCWE	タイマコントロール/ステータスレジスタ W (タイマカウンタ W 書き込み許可) ：TCWE=1 のとき，TCW への 8 ビットデータの書き込みを許可	0xFFC0 ビット 6	1
	B4WI	タイマコントロール/ステータスレジスタ W (ビット 4 書き込み禁止) ：B4WI=0 のとき，TCSRW ビット 4 への書き込みを許可 ：B4WI=1 のとき，TCSRW ビット 4 への書き込みを禁止	0xFFC0 ビット 5	0
	TCSRWE	タイマコントロール/ステータスレジスタ W (タイマコントロール/ステータスレジスタ W 書き込み許可) ：TCSRWE=1 のとき，TCSRW ビット 2 およびビット 0 への書き込みを許可	0xFFC0 ビット 4	1

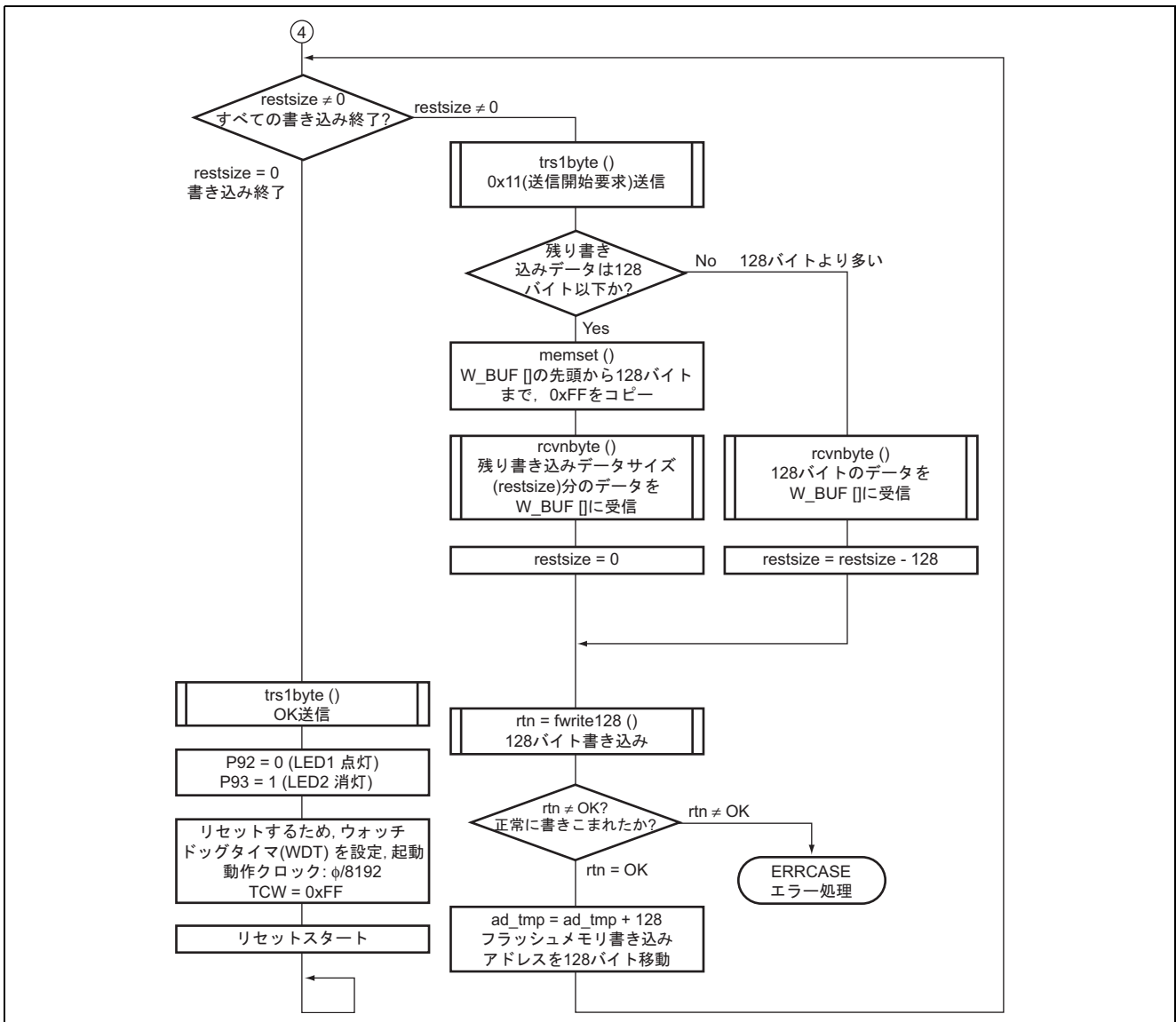
表 13 FZMAIN()関数の使用レジスタ(つづき)

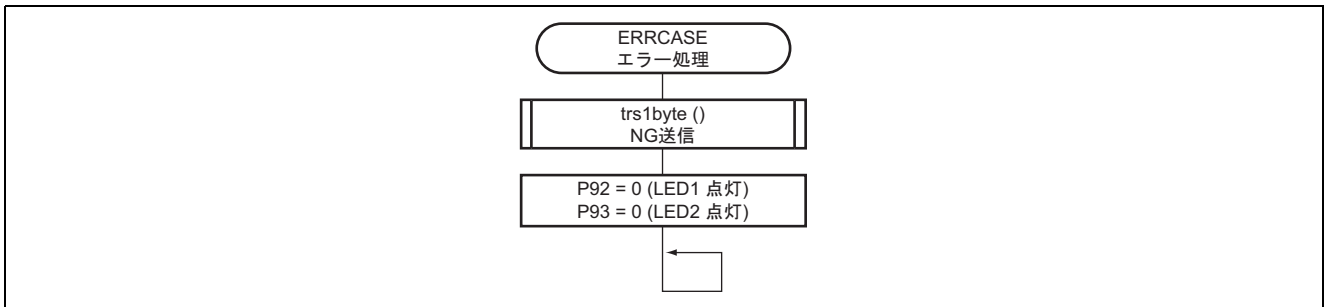
レジスタ名		機能	アドレス	設定値
TCSRW	B2WI	タイマコントロール/ステータスレジスタ W (ビット 2 書き込み禁止) : B2WI=0 のとき, TCSRW ビット 2 への書き込みを許可 : B2WI=1 のとき, TCSRW ビット 2 への書き込みを禁止	0xFFC0 ビット 3	0
	WDON	タイマコントロール/ステータスレジスタ W (ウォッチドッグタイマオン) : WDON=0 のとき, ウォッチドッグタイマの動作を禁止 : WDON=1 のとき, ウォッチドッグタイマの動作を許可	0xFFC0 ビット 2	0
	B0WI	タイマコントロール/ステータスレジスタ W (ビット 0 書き込み禁止) : B0WI=0 のとき, TCSRW ビット 0 への書き込みを許可 : B0WI=1 のとき, TCSRW ビット 0 への書き込みを禁止	0xFFC0 ビット 1	0
	WRST	タイマコントロール/ステータスレジスタ W (ウォッチドッグタイマリセット) : WRST=0 のとき, TCW はオーバフローしておらず, 内部リセット信号が発生していないことを示す : WRST=1 のとき, TCW がオーバフローし内部リセット信号が発生したことを示す	0xFFC0 ビット 0	0
TCW		タイマカウンタ W : システムクロックの 8192 分周のクロックを入力とする 8 ビットのカウンタ	0xFFC1	0xFF
PDR9	P93	ポートデータレジスタ 9(ポートデータレジスタ 93) : P93=0 のとき, P93 端子の出力レベルは"Low" : P93=1 のとき, P93 端子の出力レベルは"High"	0xFFDC ビット 3	1
	P92	ポートデータレジスタ 9(ポートデータレジスタ 92) : P92=0 のとき, P92 端子の出力レベルは"Low" : P92=1 のとき, P92 端子の出力レベルは"High"	0xFFDC ビット 2	0
PMR2	WDCKS	ポートモードレジスタ 2 (ウォッチドッグタイマソースクロック) : WDCKS=0 のとき, ウォッチドッグタイマのソースクロックに (システムクロック)/8192 を選択 : WDCKS=1 のとき, ウォッチドッグタイマのソースクロックに w(サブクロック)/32 を選択	0xFFE0 ビット 2	0

(g) フローチャート









(2) rcv1byte()関数

(a) 仕様

unsigned char rcv1byte(void)

(b) 動作説明

- 8ビットパラレルデータを1バイト受信

(c) 引数の説明

- 入力値：なし
- 出力値：1バイト受信データ

(d) グローバル変数

なし

(e) 使用サブルーチン

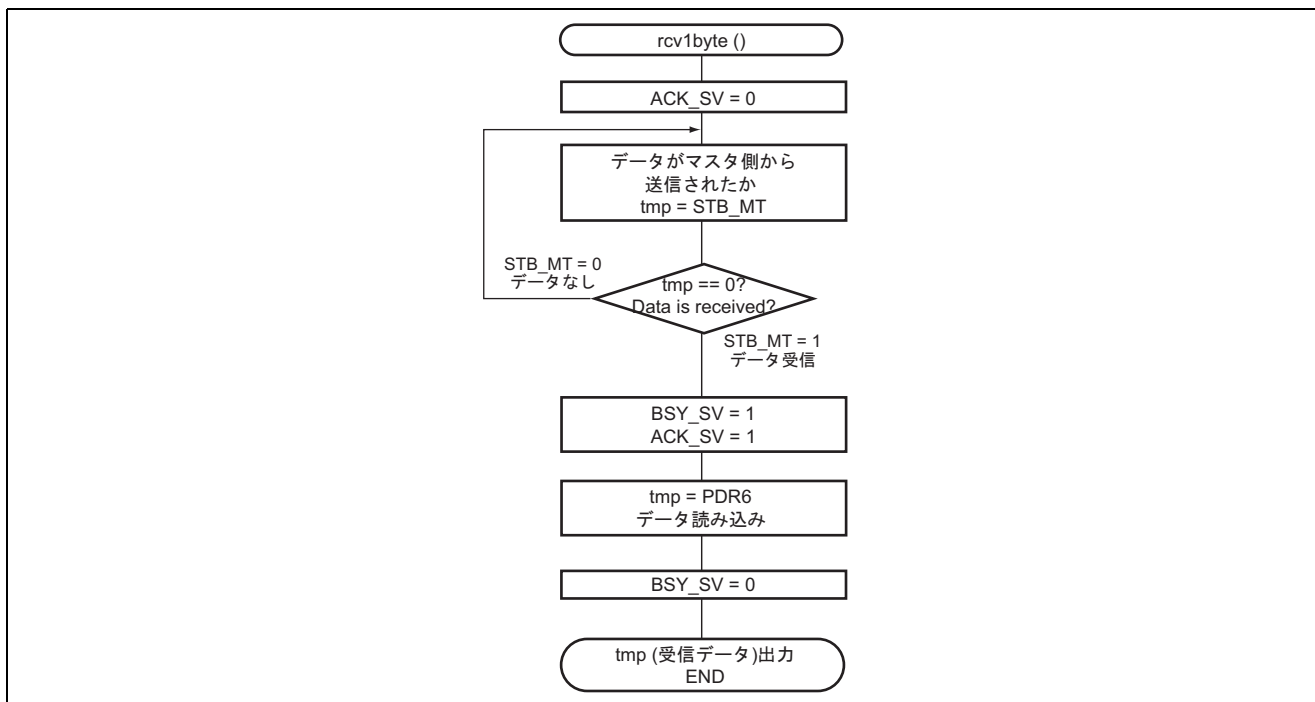
なし

(f) 使用内部レジスタ

表 14 rcv1byte()関数の使用レジスタ

レジスタ名	機能	アドレス	設定値
PDR5	P55 ポートデータレジスタ 5(ポートデータレジスタ 55) ACK_SV として用いるポート : ACK_SV(P55)=0 スレーブ側は受信可能である : ACK_SV(P55)=1 スレーブ側は受信不可である	0xFFD8 ビット 5	0
	P54 ポートデータレジスタ 5(ポートデータレジスタ 54) BSY_SV として用いるポート : BSY_SV(P54)=0 スレーブ側の内部処理が終了した : BSY_SV(P54)=1 スレーブ側が内部処理をしている	0xFFD8 ビット 4	1
	P50 ポートデータレジスタ 5(ポートデータレジスタ 50) STB_MT として用いるポート : STB_MT(P50)=0 マスタ側からデータ送信していない : STB_MT(P50)=1 マスタ側からデータ送信している	0xFFD8 ビット 0	—
PDR6	ポートデータレジスタ 6 : 受信データを格納する 8 ビットのレジスタ	0xFFD9	—

(g) フローチャート



(3) rcvnbyte()関数

(a) 仕様

```
void rcvnbyte(
    unsigned char dtno,
    unsigned char *ram
)
```

(b) 動作説明

- 8 ビットパラレルデータを n バイト受信

(c) 引数の説明

- 入力値：
 - dtno：受信バイト数
 - *ram：受信データを格納する RAM 先頭アドレス
- 出力値：1 バイト受信データ
 - *ram：受信データ

(d) グローバル変数

なし

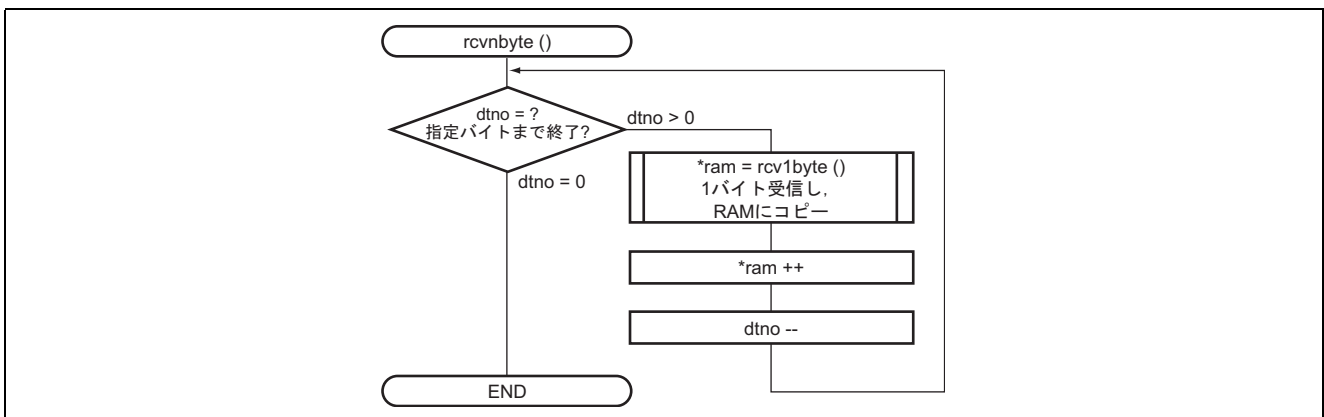
(e) 使用サブルーチン

rcv1byte：データを 1 バイト受信する

(f) 使用内部レジスタ

なし

(g) フローチャート



(4) trs1byte()関数

(a) 仕様

```
void trs1byte(unsigned char tdt)
```

(b) 動作説明

- 8 ビットパラレルデータを 1 バイト送信

(c) 引数の説明

- 入力値：
 - tdt:1 バイト送信データ
- 出力値：なし

(d) グローバル変数

なし

(e) 使用サブルーチン

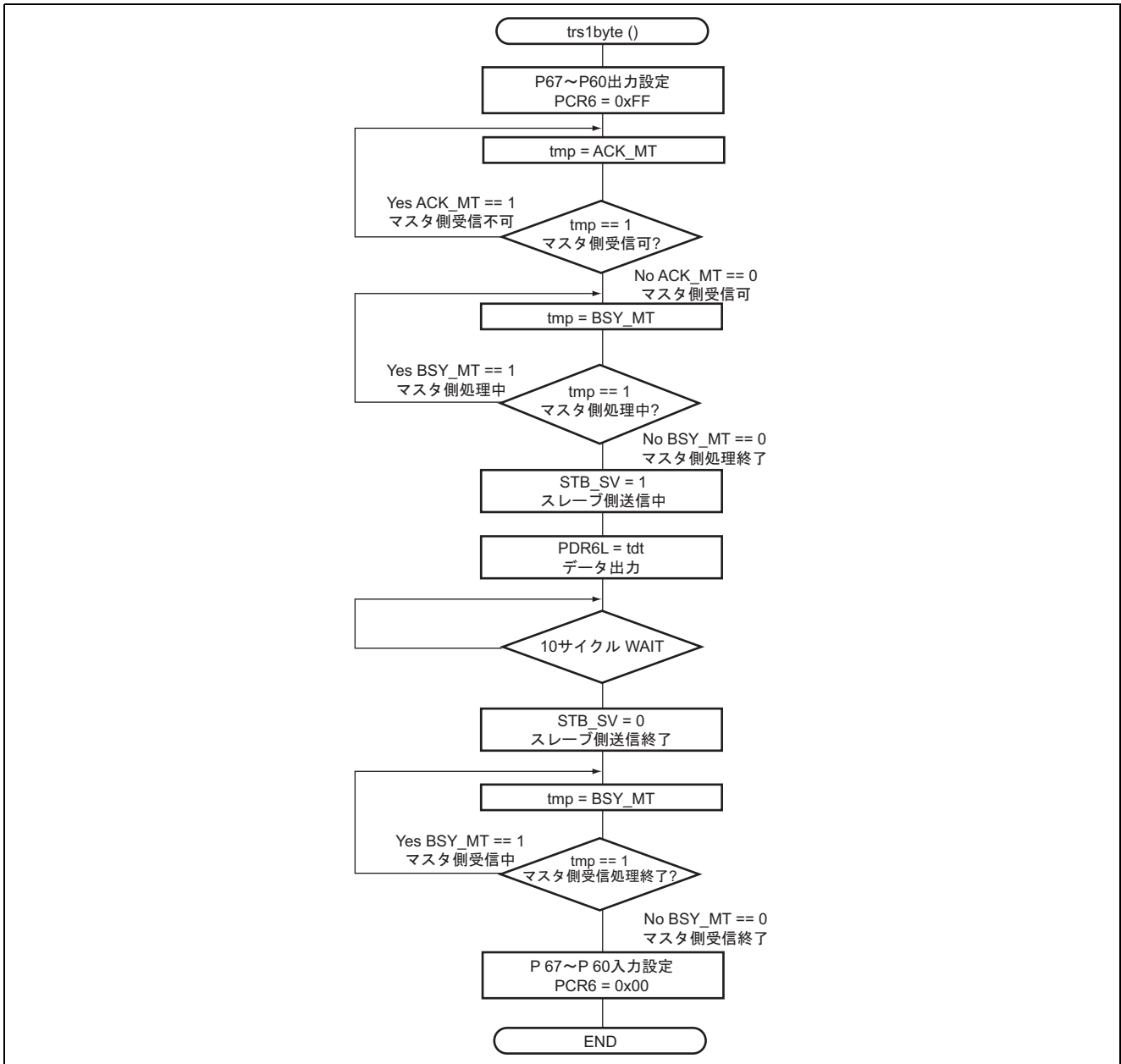
なし

(f) 使用内部レジスタ

表 15 trs1byte()関数の使用レジスタ

レジスタ名	機能	アドレス	設定値
PDR5	P53 ポートデータレジスタ 5(ポートデータレジスタ 53) STB_SV として用いるポート : STB_SV(P53)=0 スレーブ側からデータ送信していない : STB_SV(P53)=1 スレーブ側からデータ送信している	0xFFD8 ビット 3	1
	P52 ポートデータレジスタ 5(ポートデータレジスタ 52) ACK_MT として用いるポート : ACK_MT(P52)=0 マスタ側は受信可能である : ACK_MT(P52)=1 マスタ側は受信不可である	0xFFD8 ビット 2	—
	P51 ポートデータレジスタ 5(ポートデータレジスタ 51) BSY_MT として用いるポート : BSY_MT(P51)=0 マスタ側の内部処理が終了した : BSY_MT(P51)=1 マスタ側が内部処理をしている	0xFFD8 ビット 1	—
PDR6	ポートデータレジスタ 6 : 送信データを格納する 8 ビットのレジスタ	0xFFD9	—
PCR6	ポートコントロールレジスタ 6 : PCR6=0xFF のとき, P67 ~ P60 を出力端子に設定	0xFFE9	0xFF

(g) フローチャート



6.5 フラッシュメモリ書き込み / 消去処理関数の説明

(1) blk_check()関数

(a) 仕様

```
char blk_check(
    unsigned short ersad,
    unsigned short *evf_st,
    unsigned short *evf_ed,
    unsigned char *blk_no
)
```

(b) 動作説明

- 消去先頭アドレスから消去対象ブロック番号を判定する。
- 受信した消去先頭アドレスが正しい値かどうか，BLOCKADR1[]と比較判定し，結果フラグ，消去先頭アドレス，消去終了アドレス，消去対象ブロック番号を返す

(c) 引数の説明

- 入力値：
 - ersad : 消去先頭アドレス
 - *evf_st : ベリファイ後の消去先頭アドレス
 - *evf_ed : ベリファイ後の消去終了アドレス
 - *blk_no : 消去対象ブロック番号
- 出力値：
 - 戻り値 : 結果フラグ(OK=0x00,NG=0x01)
 - *evf_st : ベリファイ後の消去先頭アドレス
 - *evf_ed : ベリファイ後の消去終了アドレス
 - *blk_no : 消去対象ブロック番号

(d) グローバル変数

BLOCKADR1[] : フラッシュメモリ各ブロックの先頭，終端アドレスを格納する

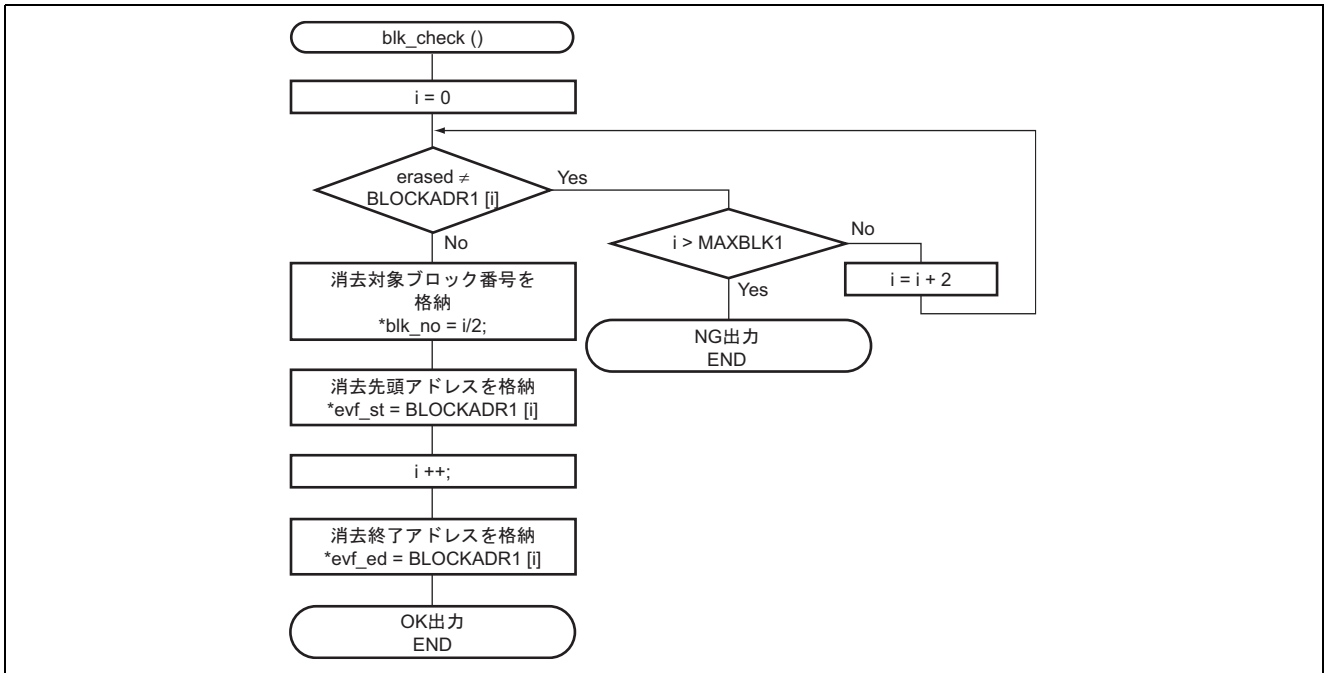
(e) 使用サブルーチン

なし

(f) 使用内部レジスタ

なし

(g) フローチャート



(2) blk1_erase()関数

(a) 仕様

```

char blk1_erase(
    unsigned short evf_st,
    unsigned short evf_ed,
    unsigned char blk_no,
    unsigned char ET_COUNT
)
  
```

(b) 動作説明

- フラッシュメモリの指定ブロックを消去する

(c) 引数の説明

- 入力値：
 - evf_st：消去先頭アドレス
 - evf_ed：消去終了アドレス
 - blk_no：消去対象ブロックのビット番号
 - ET_COUNT：消去最大回数
- 出力値：
 - 戻り値：結果フラグ(OK=0x00,NG=0x01)

(d) 外部 RAM

なし

(e) 使用サブルーチン

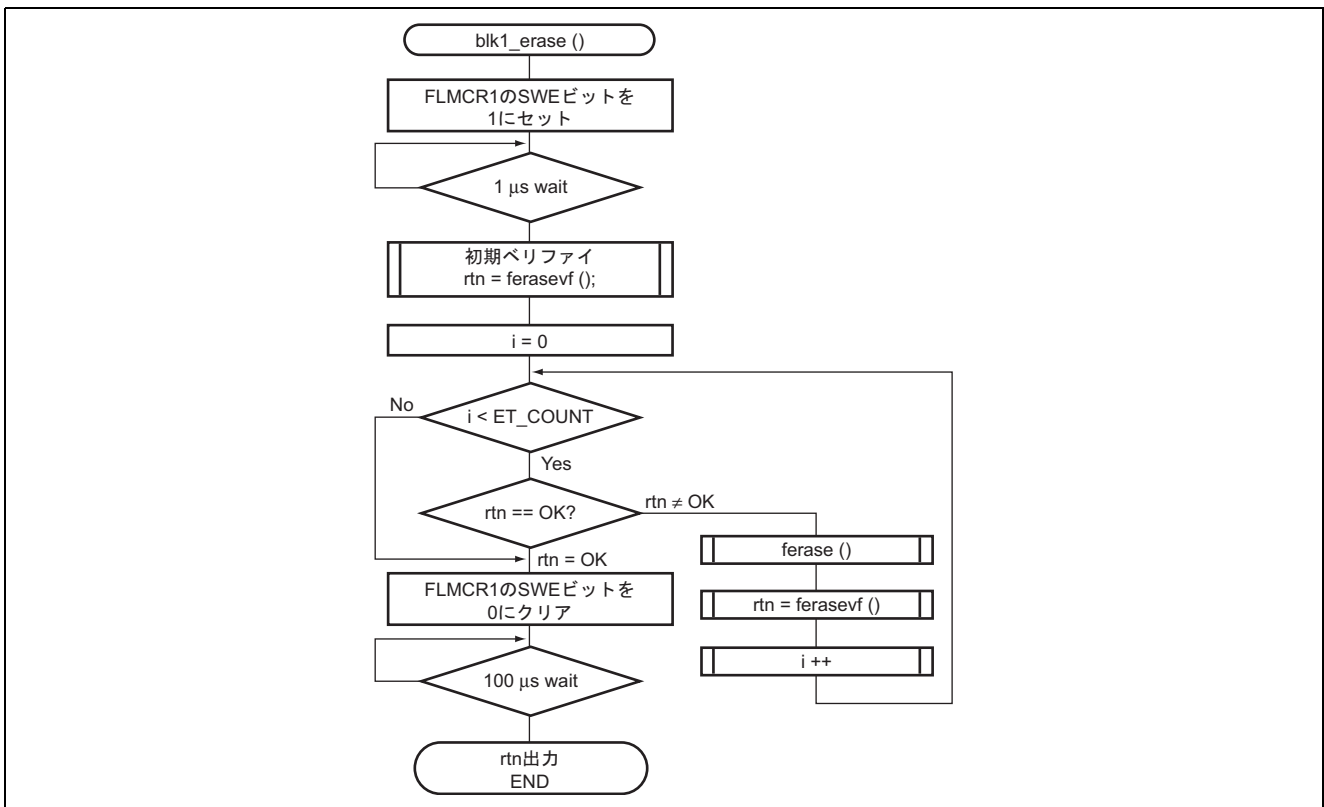
- ferase()：指定ブロックの消去
- ferasevf()：指定ブロックの消去ベリファイ

(f) 使用内部レジスタ

表 16 blk1_erase()関数の使用レジスタ

レジスタ名	機能	アドレス	設定値
FLMCR1	フラッシュメモリコントロールレジスタ 1 (ソフトウェアライトイネーブル) : SWE=0 のとき, フラッシュメモリの書き込み / 消去が無効 : SWE=1 のとき, フラッシュメモリの書き込み / 消去が可能	0xF020 ビット 6	1

(g) フローチャート



(3) ferase()関数

(a) 仕様

void ferase(unsigned char blk_no)

(b) 動作説明

- フラッシュメモリの指定ブロックを消去する

(c) 引数の説明

- 入力値
blk_no : 消去対象ブロック番号
- 出力値
なし

(d) グローバル変数

なし

(e) 使用サブルーチン

なし

(f) 使用内部レジスタ

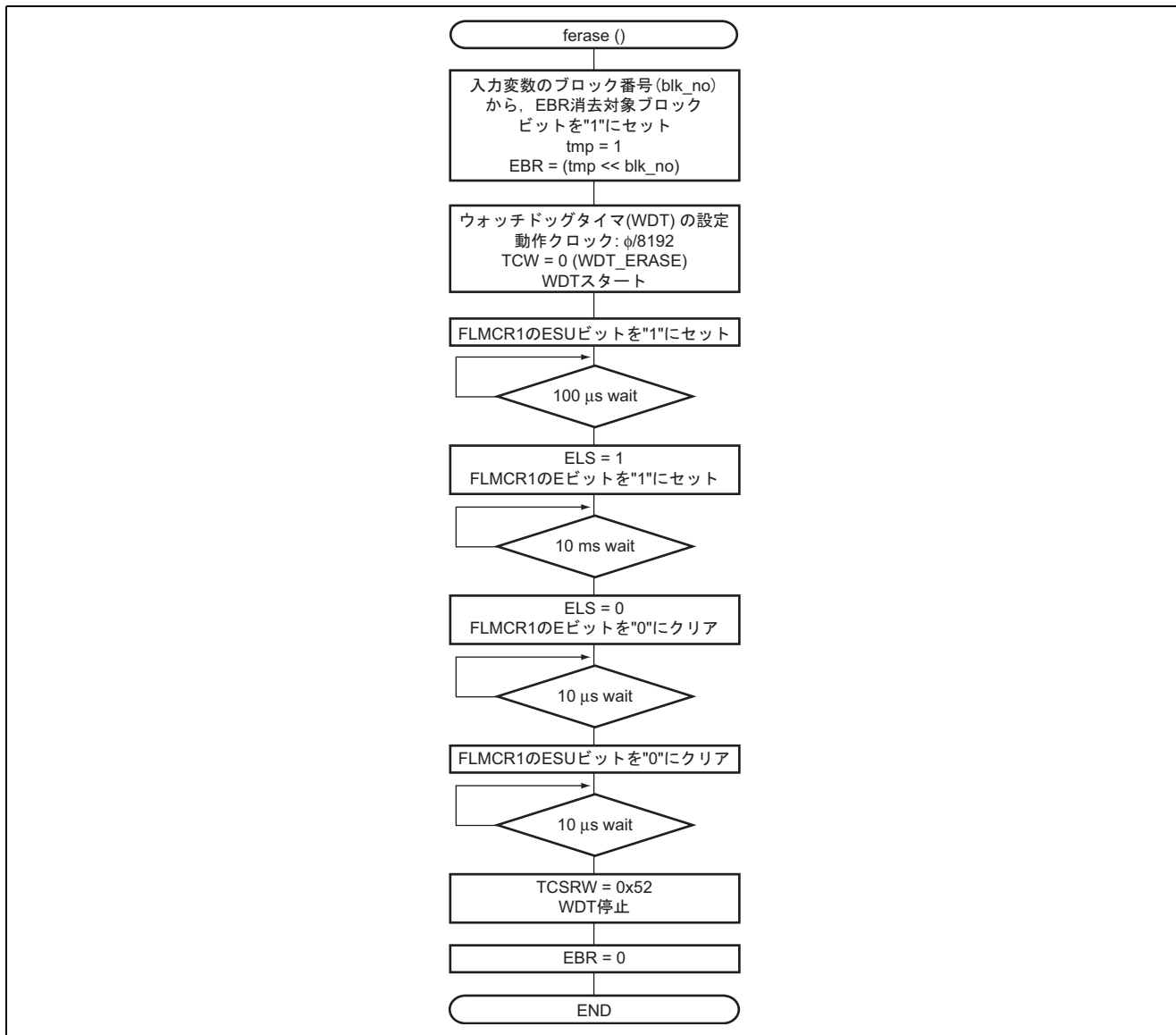
表 17 ferase()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
FLMCR1	ESU	フラッシュメモリコントロールレジスタ 1 (イレースセットアップ) : ESU=0 のとき, イレースセットアップ状態を解除 : ESU=1 のとき, イレースセットアップ状態に遷移	0xF020 ビット 5	1
	E	フラッシュメモリコントロールレジスタ 1(イレース) : E=0 のとき, イレースモードを解除 : SWE=1, ESU=1 の状態で E=1 のとき, イレースモードに 遷移	0xF020 ビット 1	1
EBR	EB4 EB3 EB2 EB1 EB0	ブロック指定レジスタ : EB4 ~ EB0 の各ビットを 1 に設定すると対応するフラッシュ メモリのブロックが可能	0xF023	—
TCSRW	B6WI	タイマコントロール/ステータスレジスタ W (ビット 6 書き込み禁止) : B6WI=0 のとき, TCSRW ビット 6 への書き込みを許可 : B6WI=1 のとき, TCSRW ビット 6 への書き込みを禁止	0xFFC0 ビット 7	0
	TCWE	タイマコントロール/ステータスレジスタ W (タイマカウンタ W 書き込み許可) : TCWE=1 のとき, TCW への 8 ビットデータの書き込みを 許可	0xFFC0 ビット 6	1
	B4WI	タイマコントロール/ステータスレジスタ W (ビット 4 書き込み禁止) : B4WI=0 のとき, TCSRW ビット 4 への書き込みを許可 : B4WI=1 のとき, TCSRW ビット 4 への書き込みを禁止	0xFFC0 ビット 5	0
	TCSRWE	タイマコントロール/ステータスレジスタ W (タイマコントロール/ステータスレジスタ W 書き込み許可) : TCSRWE=1 のとき, TCSRW ビット 2 およびビット 0 への 書き込みを許可	0xFFC0 ビット 4	1
	B2WI	タイマコントロール/ステータスレジスタ W (ビット 2 書き込み禁止) : B2WI=0 のとき, TCSRW ビット 2 への書き込みを許可 : B2WI=1 のとき, TCSRW ビット 2 への書き込みを禁止	0xFFC0 ビット 3	0
	WDON	タイマコントロール/ステータスレジスタ W (ウォッチドッグタイマオン) : WDON=0 のとき, ウォッチドッグタイマの動作を禁止 : WDON=1 のとき, ウォッチドッグタイマの動作を許可	0xFFC0 ビット 2	0
	B0WI	タイマコントロール/ステータスレジスタ W (ビット 0 書き込み禁止) : B0WI=0 のとき, TCSRW ビット 0 への書き込みを許可 : B0WI=1 のとき, TCSRW ビット 0 への書き込みを禁止	0xFFC0 ビット 1	0

表 17 ferase()関数の使用レジスタ(つづき)

レジスタ名		機能	アドレス	設定値
	WRST	タイマコントロール/ステータスレジスタ W (ウォッチドッグタイマリセット) : WRST=0 のとき, TCW はオーバフローしておらず, 内部リセット信号が発生していないことを示す : WRST=1 のとき, TCW がオーバフローし内部リセット信号が発生したことを示す	0xFFC0 ビット 0	0
	TCW	タイマカウンタ W : システムクロックの 8192 分周のクロックを入力とする 8 ビットのカウンタ	0xFFC1	0x00
PMR2	WDCKS	ポートモードレジスタ 2 (ウォッチドッグタイマソースクロック) : WDCKS=0 のとき, ウォッチドッグタイマのソースクロックに (システムクロック)/8192 を選択 : WDCKS=1 のとき, ウォッチドッグタイマのソースクロックに w(サブクロック)/32 を選択	0xFFE0 ビット 2	0

(g) フローチャート



(4) ferasevf()関数

(a) 仕様

```
char ferasevf(
    unsigned short evf_st,
    unsigned short evf_ed
)
```

(b) 動作説明

- フラッシュメモリの指定ブロックが消去されたかベリファイする

(c) 引数の説明

- 入力値
 - evf_st : 消去先頭アドレス
 - evf_ed : 消去終了アドレス
- 出力値
 - 戻り値 : 結果フラグ(OK=0x00,NG=0x01)

(d) グローバル変数

なし

(e) 使用サブルーチン

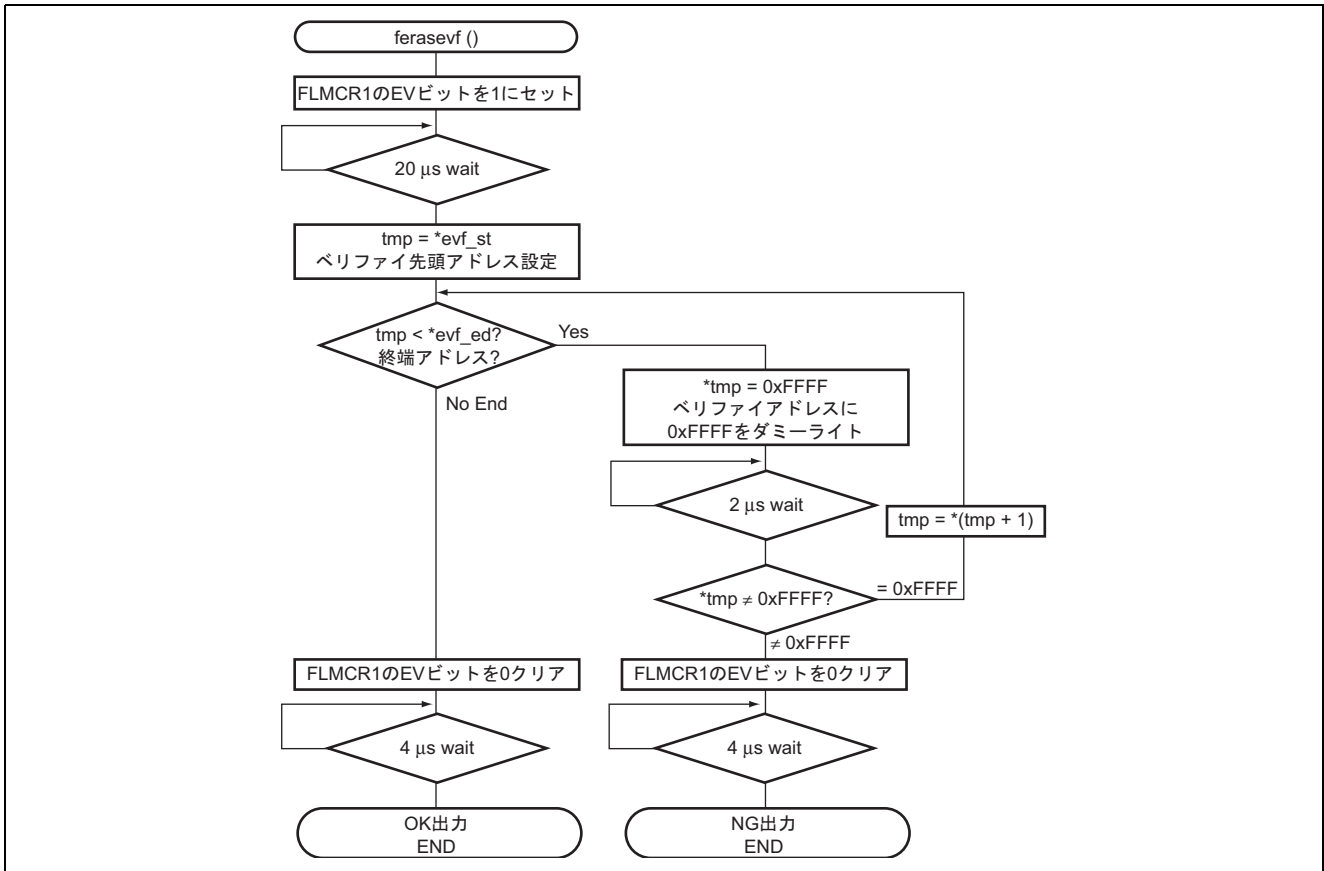
なし

(f) 使用内部レジスタ

表 18 ferasevf()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
FLMCR1	EV	フラッシュメモリコントロールレジスタ 1 (イレースベリファイ) : EV=0 のとき, イレースベリファイモードを解除 : EV=1 のとき, イレースベリファイモードに遷移	0xF020 ビット 3	1

(g) フローチャート



(5) fwrite128()関数

(a) 仕様

```
char fwrite128(
    unsigned char *BUFF,
    unsigned char *OWBUFF,
    unsigned char *w_adr,
    unsigned char *w_buf,
    unsigned short WT_COUNT
)
```

(b) 動作説明

- 128 バイトの書き込み，ベリファイ

(c) 引数の説明

- 入力値
 - *BUFF：書き込みデータバッファ
 - *OWBUFF：追加書き込みデータバッファ
 - *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
- 出力値
 - 戻り値：結果フラグ(OK=0x00,NG=0x01,WNG=0x02)
 - *BUFF：書き込みデータバッファ
 - *OWBUFF：追加書き込みデータバッファ
 - *w_adr：書き込みアドレス
 - *w_buf：書き込みデータ 128 バイト

(d) グローバル変数

なし

(e) 使用サブルーチン

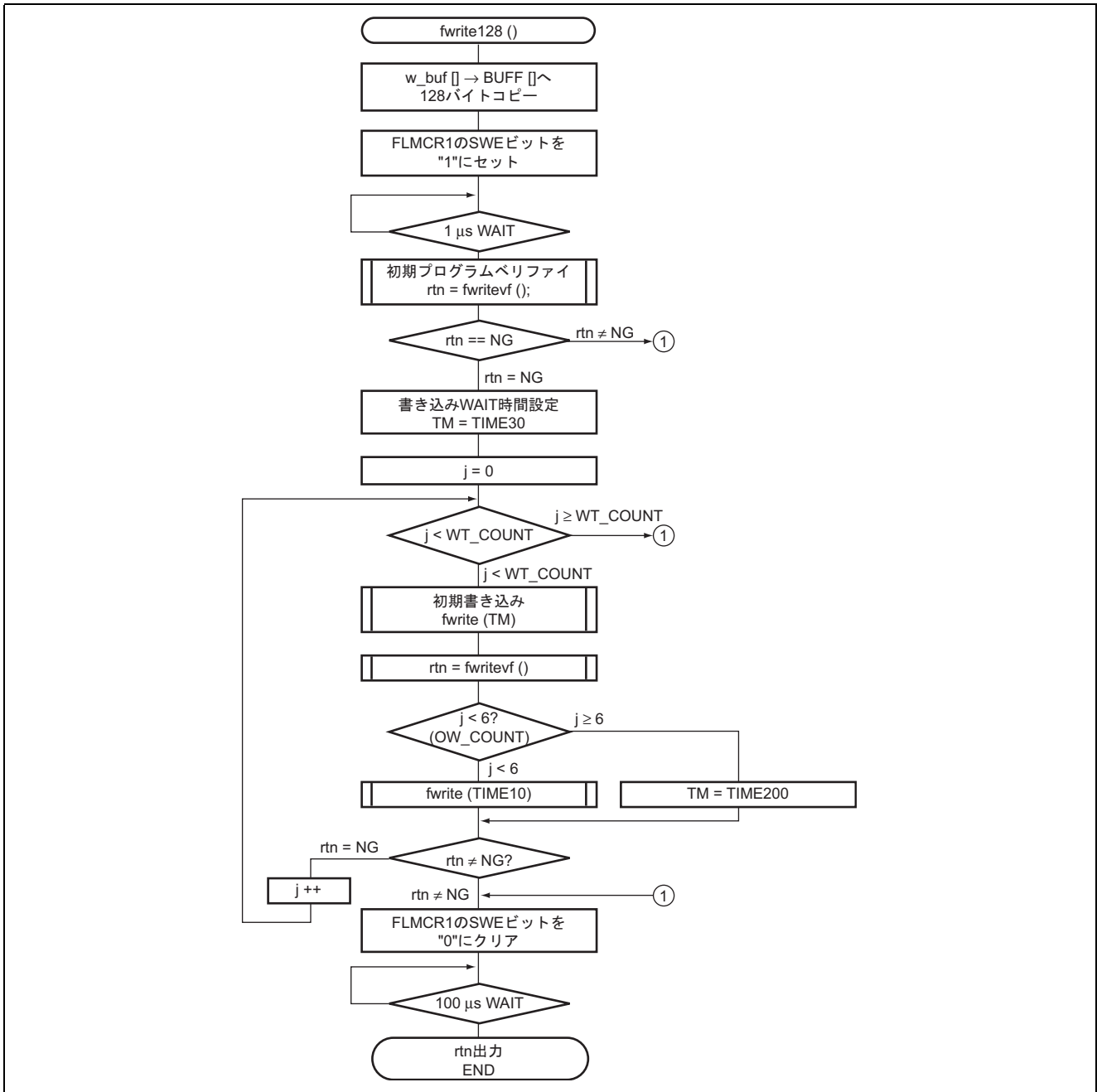
fwrite：対象アドレスの書き込み
fwritevf：対象アドレスのベリファイ，再書き込みデータの作成

(f) 使用内部レジスタ

表 19 fwrite128()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
FLMCR1	SWE	フラッシュメモリコントロールレジスタ 1 (ソフトウェアライトイネーブル) : SWE=0 のとき，フラッシュメモリの書き込み / 消去が無効 : SWE=1 のとき，フラッシュメモリの書き込み / 消去が可能	0xF020 ビット 6	1

(g) フローチャート



(6) fwrite()関数

(a) 仕様

```
void fwrite(
    unsigned char *buf,
    unsigned char *w_adr,
    unsigned char ptime
)
```

(b) 動作説明

- 対象アドレスの書き込み

(c) 引数の説明

- 入力値
 - *buf : 書き込みデータの先頭アドレス(再書き込みデータ or 追加書き込みデータ)
 - *w_adr : 書き込みアドレス
 - ptime : P ビットセット時間(10 μs or 30 μs or 2000 μs)
- 出力値
 - なし

(d) グローバル変数

なし

(e) 使用サブルーチン

なし

(f) 使用内部レジスタ

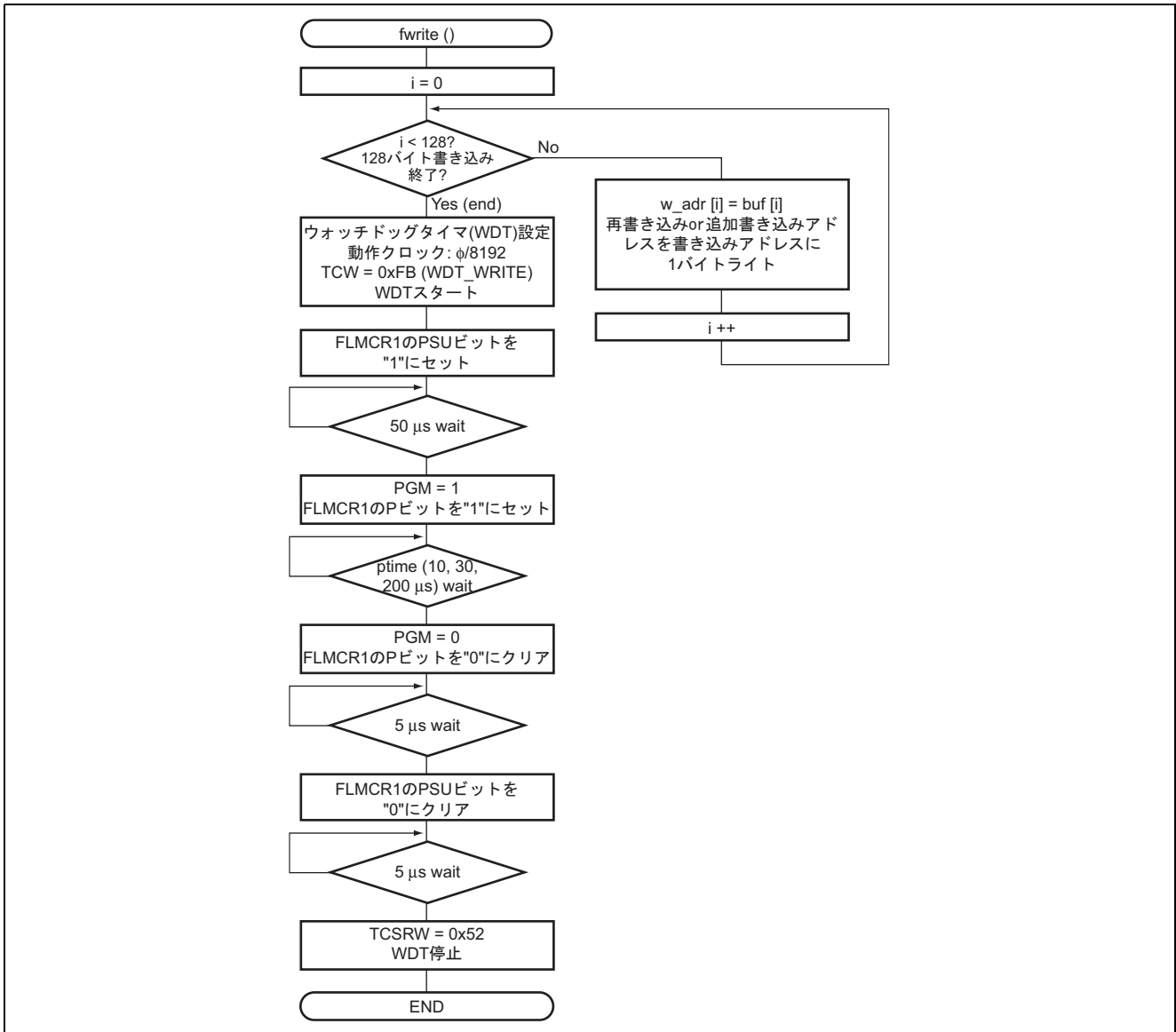
表 20 fwrite()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
FLMCR1	PSU	フラッシュメモリコントロールレジスタ 1 (プログラムセットアップ) : PSU=0 のとき, プログラムセットアップ状態を解除 : PSU=1 のとき, プログラムセットアップ状態に遷移	0xF020 ビット 4	1
	P	フラッシュメモリコントロールレジスタ 1 (プログラム) : P=0 のとき, プログラムモードを解除 : SWE=1, PSU=1 の状態で P=1 のとき, プログラムモードに遷移	0xF020 ビット 0	1
TCSRW	B6WI	タイマコントロール/ステータスレジスタ W (ビット 6 書き込み禁止) : B6WI=0 のとき, TCSRW ビット 6 への書き込みを許可 : B6WI=1 のとき, TCSRW ビット 6 への書き込みを禁止	0xFFC0 ビット 7	0
	TCWE	タイマコントロール/ステータスレジスタ W (タイマカウンタ W 書き込み許可) : TCWE=1 のとき, TCW への 8 ビットデータの書き込みを許可	0xFFC0 ビット 6	1
	B4WI	タイマコントロール/ステータスレジスタ W (ビット 4 書き込み禁止) : B4WI=0 のとき, TCSRW ビット 4 への書き込みを許可 : B4WI=1 のとき, TCSRW ビット 4 への書き込みを禁止	0xFFC0 ビット 5	0

表 20 fwrite()関数の使用レジスタ(つづき)

レジスタ名		機能	アドレス	設定値
TCSRW	TCSRWE	タイマコントロール/ステータスレジスタ W (タイマコントロール/ステータスレジスタ W 書き込み許可) : TCSRWE=1 のとき, TCSRW ビット 2 およびビット 0 への書き込みを許可	0xFFC0 ビット 4	1
	B2WI	タイマコントロール/ステータスレジスタ W (ビット 2 書き込み禁止) : B2WI=0 のとき, TCSRW ビット 2 への書き込みを許可 : B2WI=1 のとき, TCSRW ビット 2 への書き込みを禁止	0xFFC0 ビット 3	0
	WDON	タイマコントロール/ステータスレジスタ W (ウォッチドッグタイマオン) : WDON=0 のとき, ウォッチドッグタイマの動作を禁止 : WDON=1 のとき, ウォッチドッグタイマの動作を許可	0xFFC0 ビット 2	0
	B0WI	タイマコントロール/ステータスレジスタ W (ビット 0 書き込み禁止) : B0WI=0 のとき, TCSRW ビット 0 への書き込みを許可 : B0WI=1 のとき, TCSRW ビット 0 への書き込みを禁止	0xFFC0 ビット 1	0
	WRST	タイマコントロール/ステータスレジスタ W (ウォッチドッグタイマリセット) : WRST=0 のとき, TCW はオーバフローしておらず, 内部リセット信号が発生していないことを示す : WRST=1 のとき, TCW がオーバフローし内部リセット信号が発生したことを示す	0xFFC0 ビット 0	0
TCW		タイマカウンタ W : システムクロックの 8192 分周のクロックを入力とする 8 ビットのカウンタ	0xFFC1	0xFB
PMR2	WDCKS	ポートモードレジスタ 2 (ウォッチドッグタイマソースクロック) : WDCKS=0 のとき, ウォッチドッグタイマのソースクロックに (システムクロック)/8192 を選択 : WDCKS=1 のとき, ウォッチドッグタイマのソースクロックに w(サブクロック)/32 を選択	0xFFE0 ビット 2	0

(g) フローチャート



(7) fwritevf()関数

(a) 仕様

```
char fwritevf(
    unsigned char *owbuff,
    unsigned char *buff,
    unsigned char *w_adr,
    unsigned char *w_buf
```

)

(b) 動作説明

- 対象アドレスのベリファイ、再書き込みデータの作成

(c) 引数の説明

- 入力値
 - *owbuff: 追加書き込みデータ 128 バイト
 - *buff: 再書き込みデータ 128 バイト
 - *w_adr: 書き込みアドレス
 - *w_buf: 書き込みデータ 128 バイト
- 出力値
 - 戻り値: 結果フラグ(OK=0x00,NG=0x01,WNG=0x02)
 - *owbuff: 追加書き込みデータ 128 バイト
 - *buff: 再書き込みデータ 128 バイト
 - *w_adr: 書き込みアドレス
 - *w_buf: 書き込みデータ 128 バイト

(d) グローバル変数

なし

(e) 使用サブルーチン

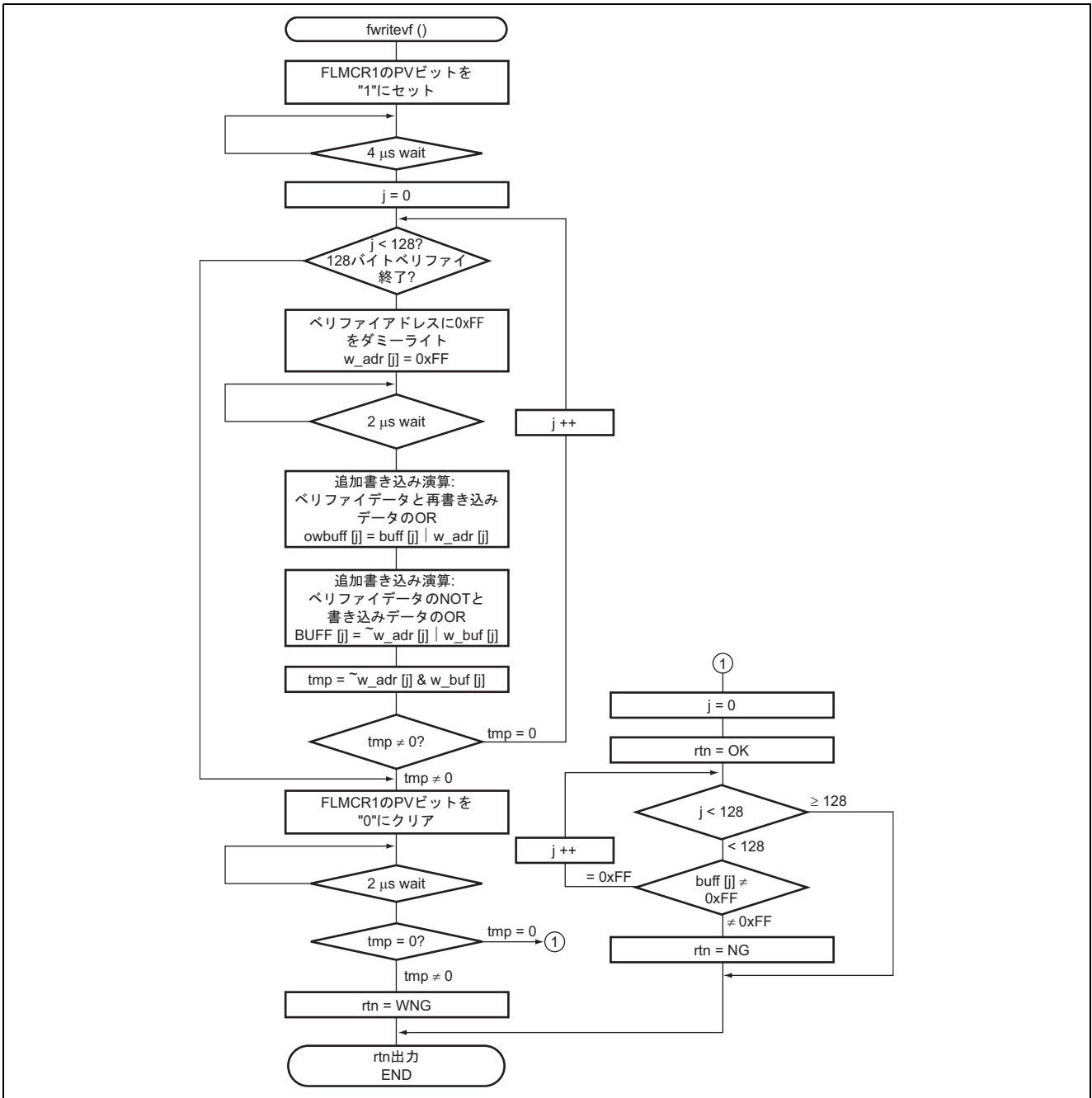
なし

(f) 使用内部レジスタ

表 21 fwritevf()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
FLMCR1	PV	フラッシュメモリコントロールレジスタ 1 (プログラムベリファイ) : PV=0 のとき, プログラムベリファイモードを解除 : PV=1 のとき, プログラムベリファイモードに遷移	0xF020 ビット 2	1

(g) フローチャート



7. マスタ側プログラム

7.1 階層構造

マスタ側プログラムは、スレーブ側と通信し、スレーブ側フラッシュメモリの消去、書き込みの指示、及び書き込みデータの送信を行います。マスタ側プログラムで使用するルーチンの階層構造を図 21 に示します。

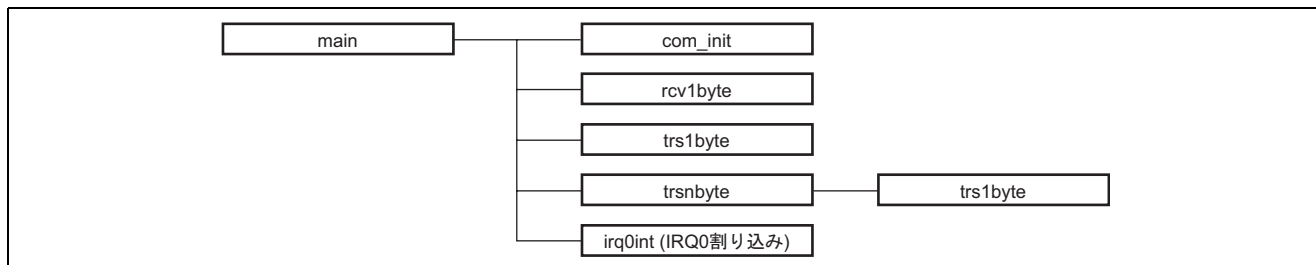


図 21 マスタ側書き込み / 消去制御プログラム

7.2 関数一覧

表 22 書き込み / 消去制御プログラム関数一覧

関数名	概要
main	書き込み / 消去制御プログラムのメインルーチン
com_init	通信設定の初期化
rcv1byte	データを 1 バイト受信する
trs1byte	データを 1 バイト送信する
trsnbyte	データを n バイト送信する
irq0int	IRQ0 割り込み処理で、書き込み開始コマンド送信処理に移行するためのフラグ設定を行う

7.3 定数一覧

表 23 定数一覧

定数名	値	内容
OK	0x00	正常時の戻り値
NG	0x01	異常時の戻り値

7.4 関数説明

(1) main()関数

(a) 仕様

void main(void)

(b) 動作説明

- コマンド発行を開始し，スレーブ側とデータの送受信を行う

(c) 引数の説明

- 入力値：なし
- 出力値：なし

(d) グローバル変数

- ramf：書き込み開始コマンド送信処理への分岐を判定する
ramf=0 のとき，書き込み開始コマンド送信処理へ分岐
ramf=1 のとき，サンプルの通常アプリケーションを実行

(e) 使用サブルーチン

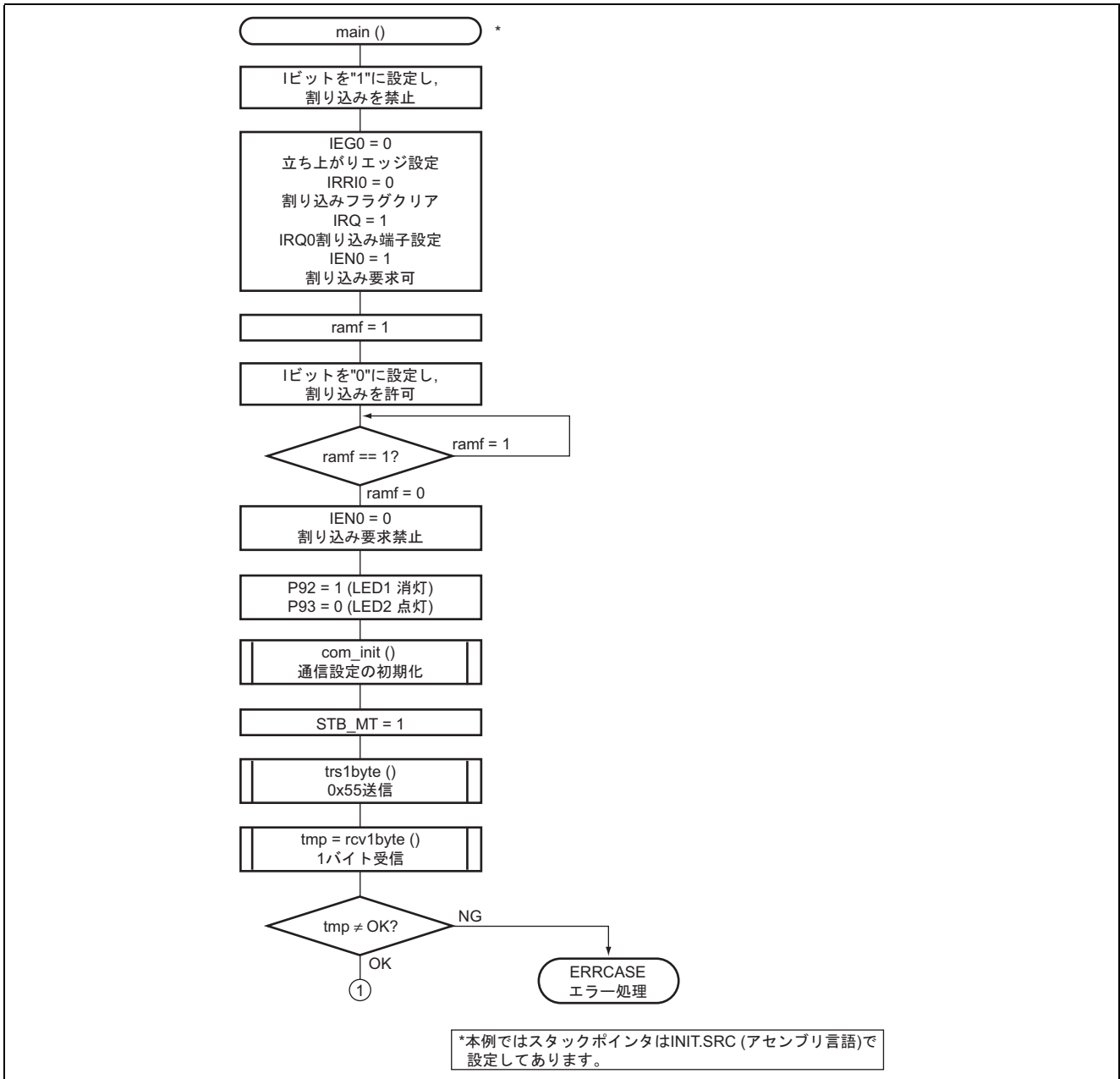
com_init()：通信設定の初期化
 rev1byte()：データを 1 バイト受信する
 trs1byte()：データを 1 バイト送信する
 trsnbyte()：データを n バイト送信する

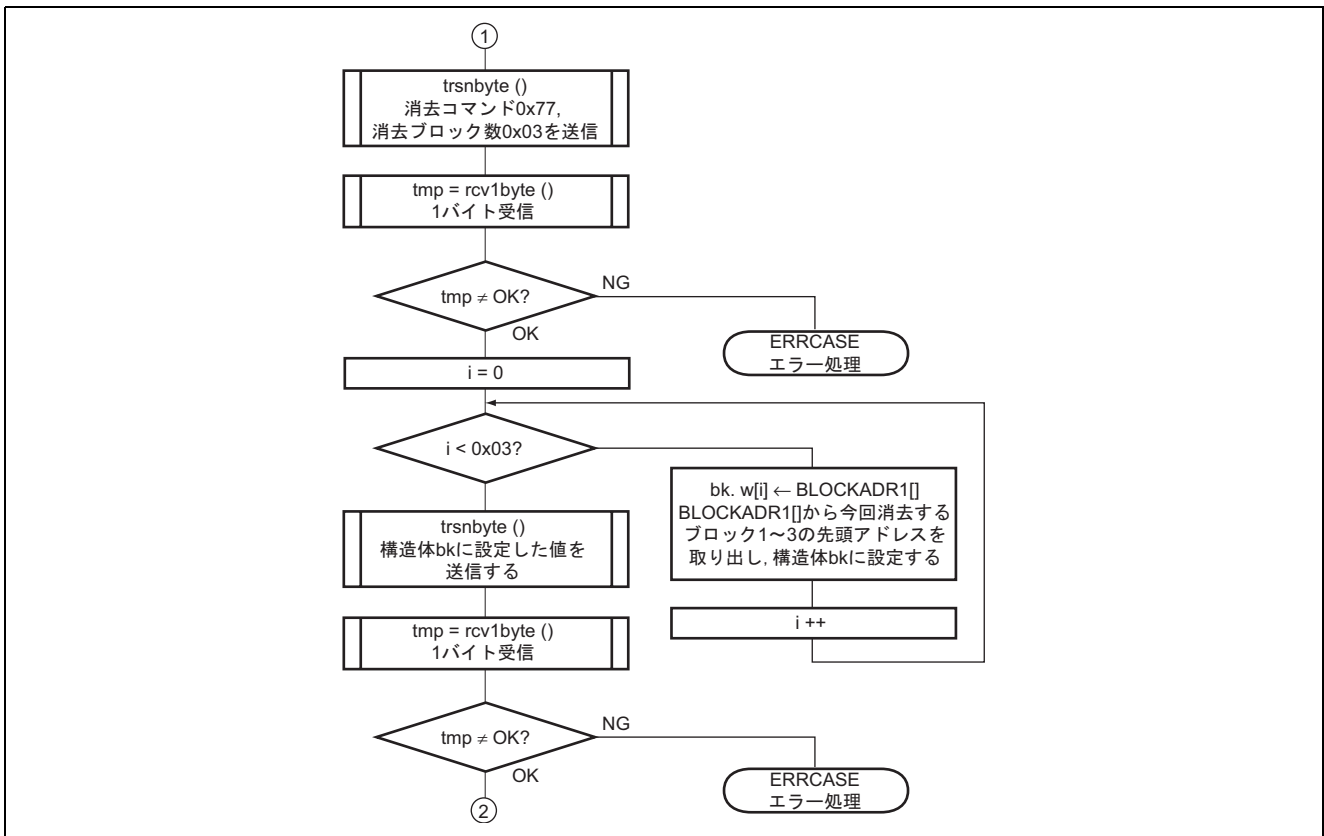
(f) 使用内部レジスタ

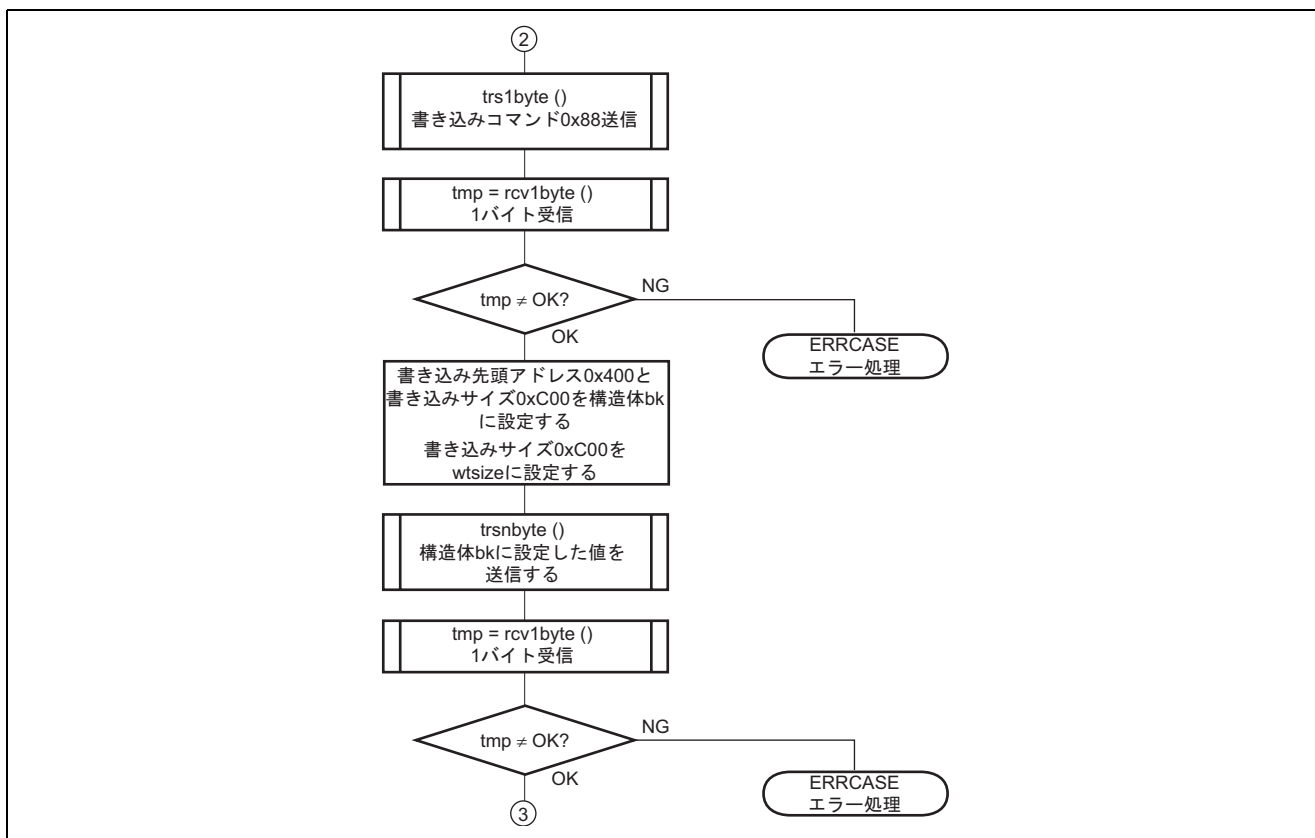
表 24 main()関数の使用レジスタ

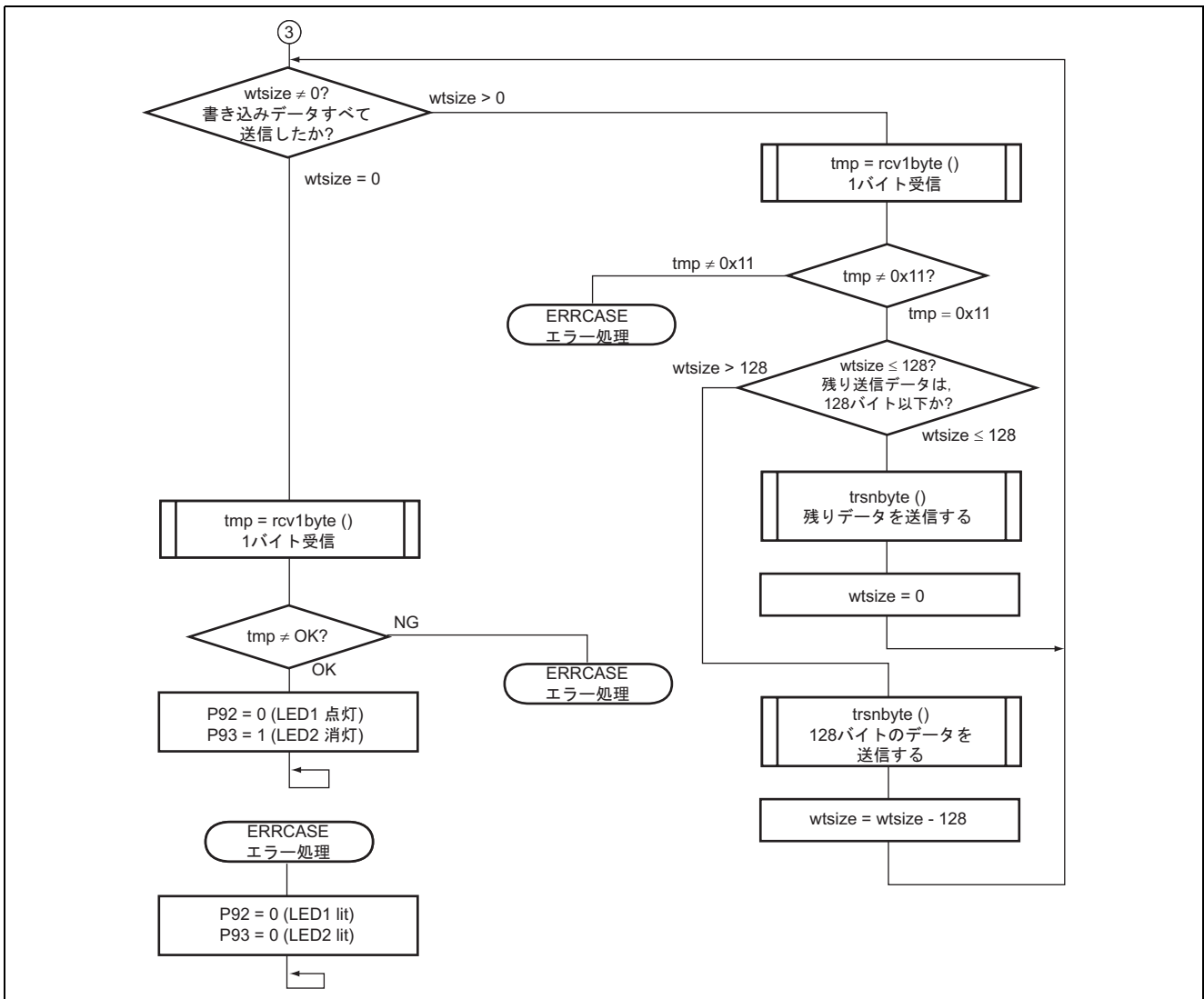
レジスタ名		機能	アドレス	設定値
PDR9	P93	ポートデータレジスタ 9(ポートデータレジスタ 93) ： P93=0 のとき，P93 端子の出力レベルは"Low" ： P93=1 のとき，P93 端子の出力レベルは"High"	0xFFDC ビット 3	0
	P92	ポートデータレジスタ 9(ポートデータレジスタ 92) ： P92=0 のとき，P92 端子の出力レベルは"Low" ： P92=1 のとき，P92 端子の出力レベルは"High"	0xFFDC ビット 2	1
PMR2	IRQ0	ポートモードレジスタ 2(P43/IRQ0 端子機能切り替え) ： IRQ0=0 のとき，P43 入出力端子として機能 ： IRQ0=1 のとき，IRQ0 入力端子として機能	0xFFE0 ビット 0	1
IEGR	IEG0	IRQ エッジセレクトレジスタ(IRQ0 エッジセレクト) ： IEG0=0 のとき，IRQ0 端子入力の検出エッジに立ち下がりエッジを選択 ： IEG0=1 のとき，IRQ0 端子入力の検出エッジに立ち上がりエッジを選択	0xFFFF2 ビット 0	0
IENR1	IEN0	割り込み許可レジスタ 1(IRQ0 割り込みイネーブル) ： IEN0=0 のとき，IRQ0 端子の割り込み要求を禁止 ： IEN0=1 のとき，IRQ0 端子の割り込み要求を許可	0xFFFF3 ビット 0	1
IRR1	IRRI0	割り込み要求レジスタ 1(IRQ0 割り込み要求フラグ) ： IRRI0=0 のとき，IRQ0 割り込みが要求されていない ： IRRI0=1 のとき，IRQ0 割り込みが要求されている	0xFFFF6 ビット 0	0

(g) フローチャート









(2) com_init()関数

(a) 仕様

void com_init(void)

(b) 動作説明

- 8ビットパラレル通信の初期化

(c) 引数の説明

- 入力値：なし
- 出力値：なし

(d) グローバル変数

なし

(e) 使用サブルーチン

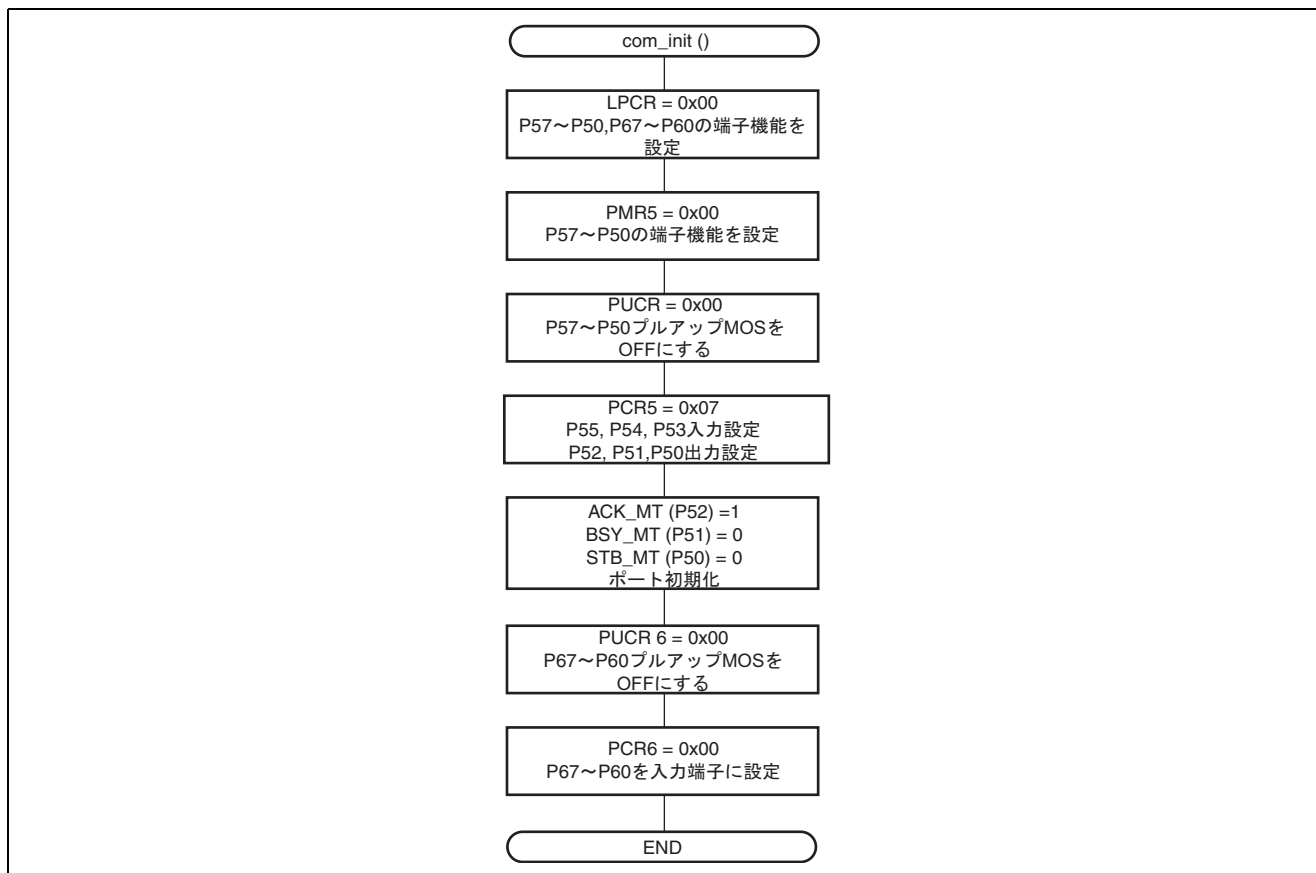
なし

(f) 使用内部レジスタ

表 25 com_init()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
LPCR	SGS3	LCD ポートコントロールレジスタ(セグメントドライバ選択) : SGS3=0, SGS2=0, SGS1=0, SGS0=0 のとき, SEG32 ~ SEG1 端子はポートとして機能	0xFFC0 ビット 3 ビット 2 ビット 1 ビット 0	SGS3=0 SGS2=0 SGS1=0 SGS0=0
	SGS2			
	SGS1			
	SGS0			
PMR5		ポートモードレジスタ 5 : PMR5=0x00 のとき, ~WKP7 ~ ~WKP0 入力端子を P57 ~ P50 端 子に設定	0xFFCC	0x00
PDR5	P52	ポートデータレジスタ 5(ポートデータレジスタ 52) ACK_MT として用いるポート : ACK_MT(P52)=0 マスタ側は受信可能である : ACK_MT(P52)=1 マスタ側は受信不可である	0xFFD8 ビット 2	1
	P51	ポートデータレジスタ 5(ポートデータレジスタ 51) BSY_MT として用いるポート : BSY_MT(P51)=0 マスタ側の内部処理が終了した : BSY_MT(P51)=1 マスタ側が内部処理をしている	0xFFD8 ビット 1	0
	P50	ポートデータレジスタ 5(ポートデータレジスタ 50) STB_MT として用いるポート : STB_MT(P50)=0 マスタ側からデータ送信していない : STB_MT(P50)=1 マスタ側からデータ送信している	0xFFD8 ビット 0	0
PUCR5		ポートプルアップコントロールレジスタ 5 : PUCR5=0x00 のとき, P57 ~ P50 のプルアップ MOS を OFF に 設定	0xFFE2	0x00
PUCR6		ポートプルアップコントロールレジスタ 6 : PUCR6=0x00 のとき, P67 ~ P60 のプルアップ MOS を OFF に 設定	0xFFE3	0x00
PCR5		ポートコントロールレジスタ 5 : PCR5=0x07 のとき, P55, P54, P53 を入力端子に, P52, P51, P50 を出力端子に設定	0xFFE8	0x07
PCR6		ポートコントロールレジスタ 6 : PCR6=0x00 のとき, P67 ~ P60 を入力端子に設定	0xFFE9	0x00

(g) フローチャート



(3) rcv1byte()関数

(a) 仕様

unsigned char rcv1byte(void)

(b) 動作説明

- 8ビットパラレルデータを1バイト受信

(c) 引数の説明

- 入力値：なし
- 出力値：1バイト受信データ

(d) グローバル変数

なし

(e) 使用サブルーチン

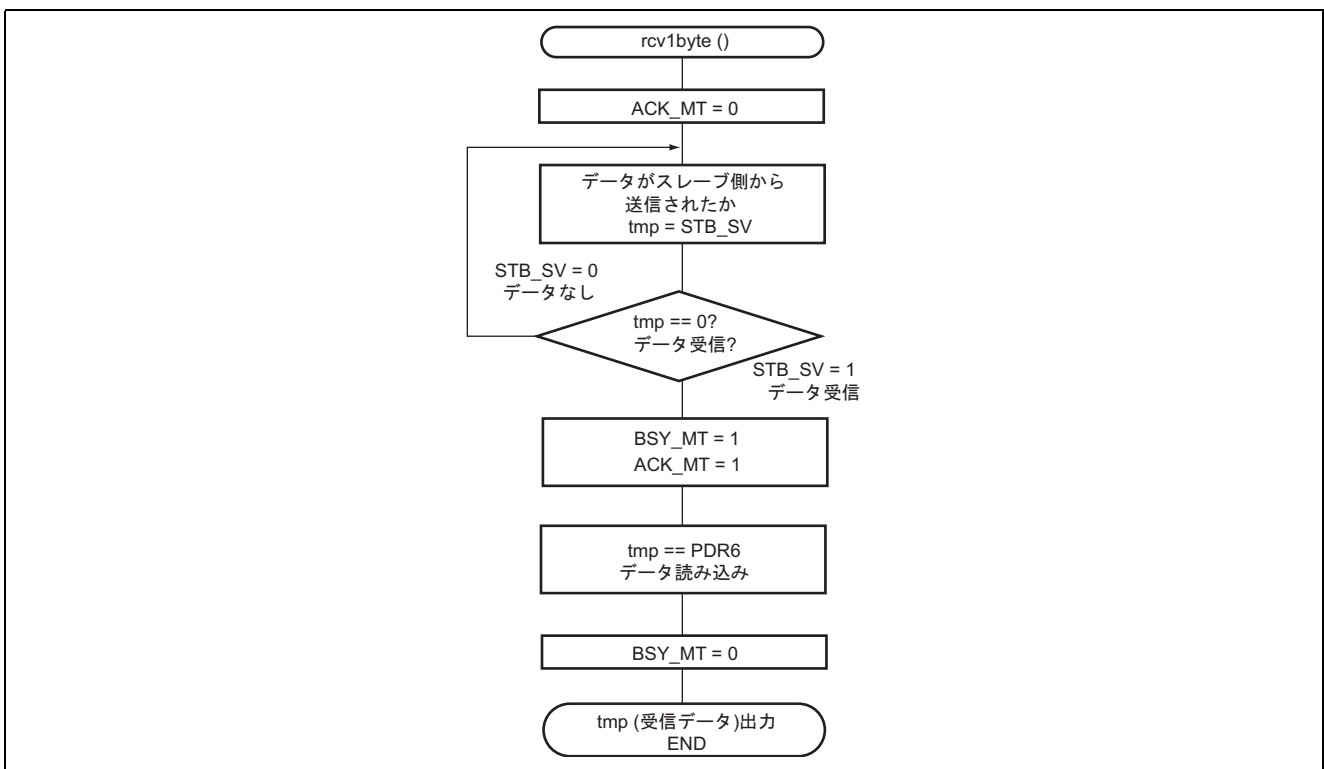
なし

(f) 使用内部レジスタ

表 26 rcv1byte()関数の使用レジスタ

レジスタ名	機能	アドレス	設定値
PDR5	P53 ポートデータレジスタ 5(ポートデータレジスタ 53) STB_SV として用いるポート : STB_SV(P53)=0 スレーブ側からデータ送信していない : STB_SV(P53)=1 スレーブ側からデータ送信している	0xFFD8 ビット 3	—
	P52 ポートデータレジスタ 5(ポートデータレジスタ 52) ACK_MT として用いるポート : ACK_MT(P52)=0 マスタ側は受信可能である : ACK_MT(P52)=1 マスタ側は受信不可である	0xFFD8 ビット 2	0
	P51 ポートデータレジスタ 5(ポートデータレジスタ 51) BSY_MT として用いるポート : BSY_MT(P51)=0 マスタ側の内部処理が終了した : BSY_MT(P51)=1 マスタ側が内部処理をしている	0xFFD8 ビット 1	1
PDR6	ポートデータレジスタ 6 : 受信データを格納する 8 ビットのレジスタ	0xFFD9	—

(g) フローチャート



(4) trs1byte()関数

(a) 仕様

void trs1byte(unsigned char tdt)

(b) 動作説明

- 8 ビットパラレルデータを 1 バイト送信

(c) 引数の説明

- 入力値 :
tdt:1 バイト送信データ
- 出力値 : なし

(d) グローバル変数

なし

(e) 使用サブルーチン

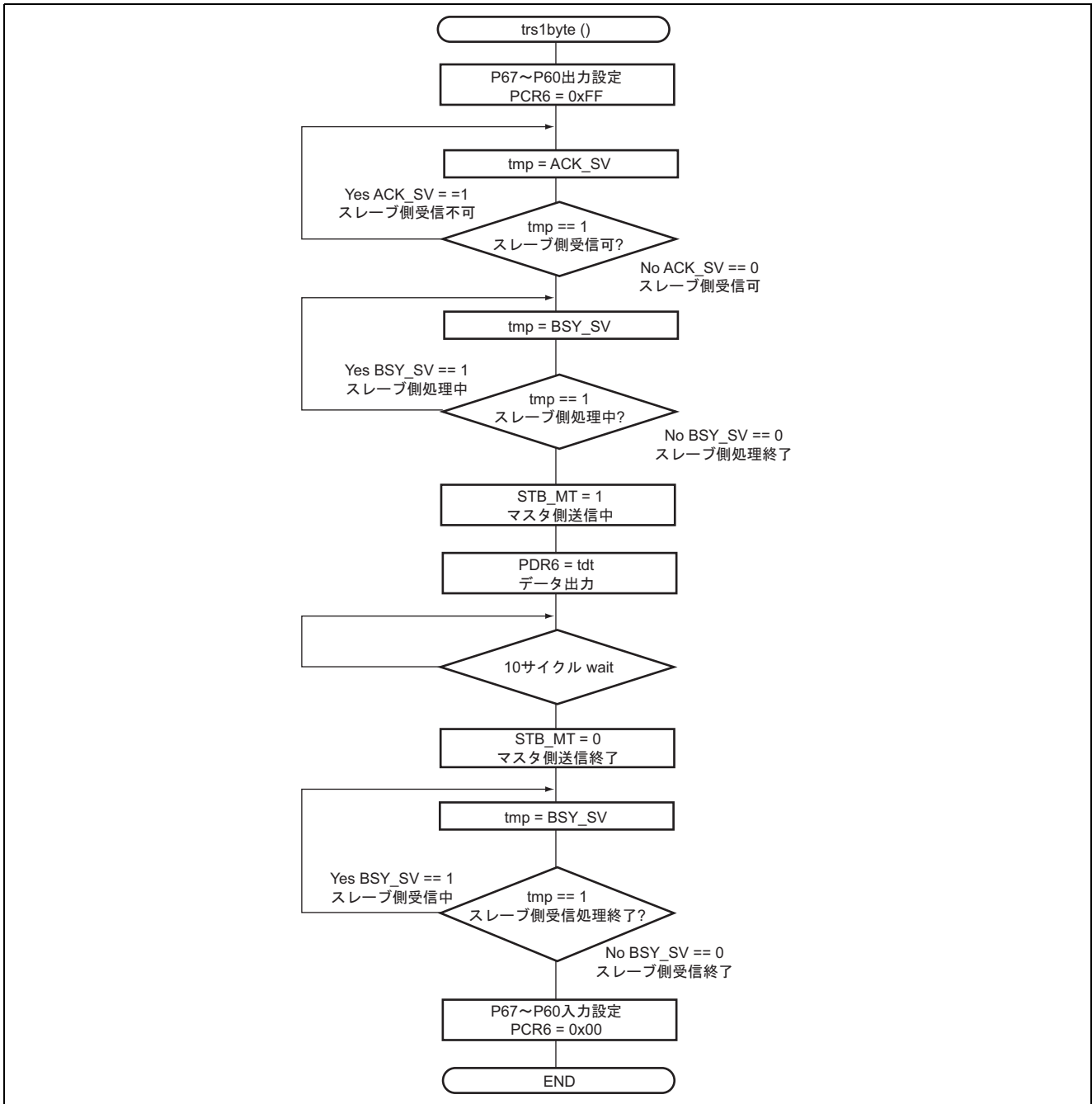
なし

(f) 使用内部レジスタ

表 27 trs1byte()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
PDR5	P55	ポートデータレジスタ 5(ポートデータレジスタ 55) ACK_SV として用いるポート : ACK_SV(P55)=0 スレーブ側は受信可能である : ACK_SV(P55)=1 スレーブ側は受信不可である	0xFFD8 ビット 5	—
	P54	ポートデータレジスタ 5(ポートデータレジスタ 54) BSY_SV として用いるポート : BSY_SV(P54)=0 スレーブ側の内部処理が終了した : BSY_SV(P54)=1 スレーブ側が内部処理をしている	0xFFD8 ビット 4	—
	P50	ポートデータレジスタ 5(ポートデータレジスタ 50) STB_MT として用いるポート : STB_MT(P50)=0 マスタデータ送信していない : STB_MT(P50)=1 マスタデータ送信している	0xFFD8 ビット 0	1
PDR6	ポートデータレジスタ 6 : 送信データを格納する 8 ビットのレジスタ	0xFFD9	—	
PCR6	ポートコントロールレジスタ 6 : PCR6=0xFF のとき, P67 ~ P60 を出力端子に設定	0xFFE9	0xFF	

(g) フローチャート



(5) trsnbyte()関数

(a) 仕様

```
void trsnbyte(
    short dtno,
    unsigned char *tdt
)
```

(b) 動作説明

- 8 ビットパラレルデータを n バイト送信

(c) 引数の説明

- 入力値：
 - dtno：送信バイト数
 - *tdt：送信データを格納する RAM 先頭アドレス
- 出力値：なし

(d) グローバル変数

なし

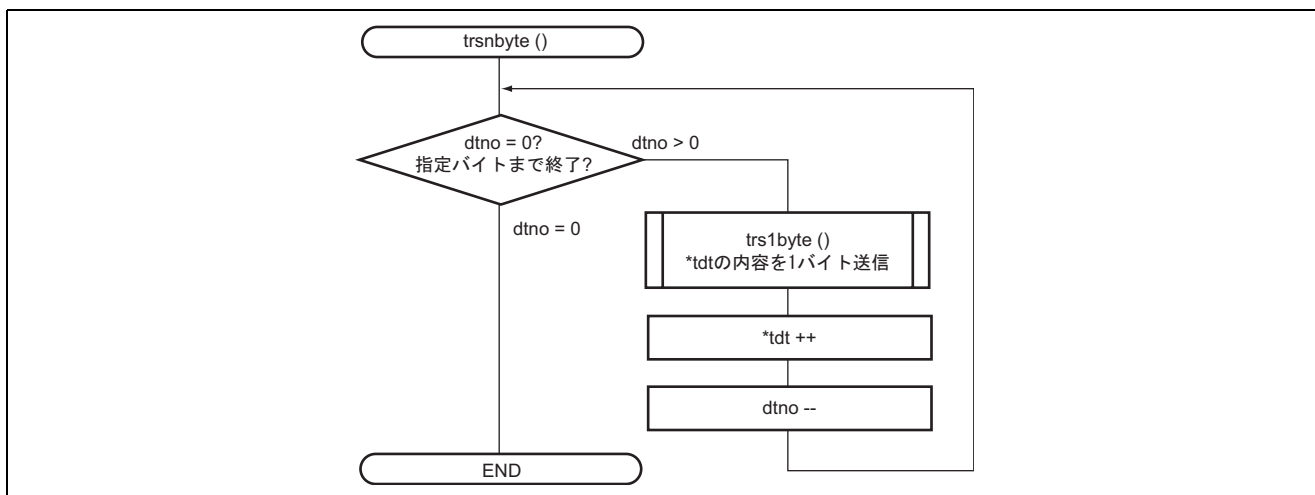
(e) 使用サブルーチン

trs1byte()：データを 1 バイト送信する

(f) 使用内部レジスタ

なし

(g) フローチャート



(6) irq0int()関数

(a) 仕様

void irq0int(void)

(b) 動作説明

- IRQ0 割り込み処理で、書き込み開始コマンド送信処理へ分岐するためのフラグ設定を行う

(c) 引数の説明

- 入力値：なし
- 出力値：なし

(d) グローバル変数

- ramf :
ramf=0 に設定し、割り込み終了後、書き込み開始コマンド送信処理へ分岐する

(e) 使用サブルーチン

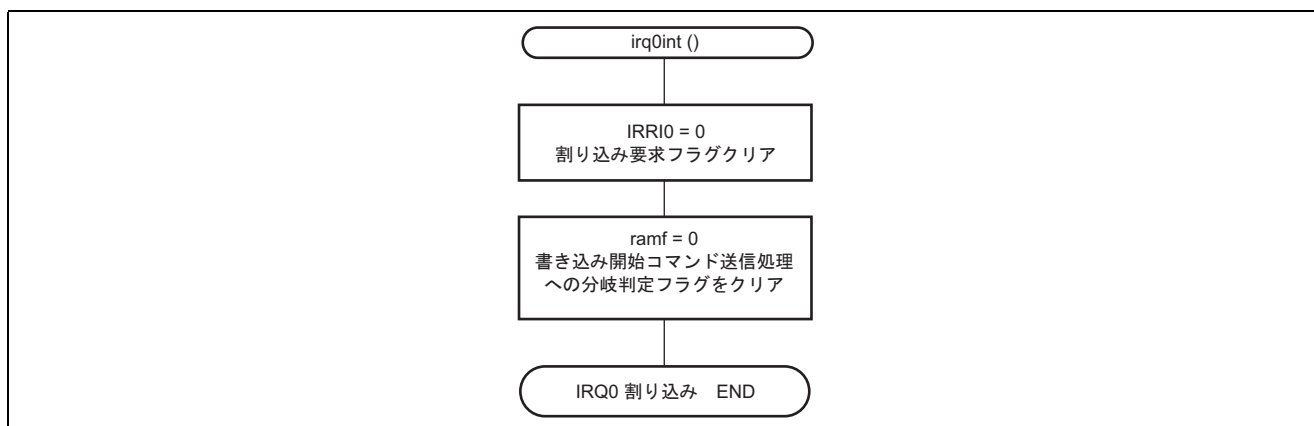
なし

(f) 使用内部レジスタ

表 28 irq0int()関数の使用レジスタ

レジスタ名		機能	アドレス	設定値
IRR1	IRRI0	割り込み要求レジスタ 1(IRQ0 割り込み要求フラグ) : IRRI0=0 のとき、IRQ0 割り込みが要求されていない : IRRI0=1 のとき、IRQ0 割り込みが要求されている	0xFFFF6 ビット 0	0

(g) フローチャート



8. プログラムリスト

8.1 スタックポインタの設定プログラム

INIT.SRC(マスタ側, スレーブ側共通)

```

.EXPORT  _INIT
.IMPORT  _main
;
.SECTION P, CODE
_INIT:
MOV.W   #H'FF80,R7
LDC.B   #B'10000000,CCR
JMP     @_main
;
.END

```

8.2 スレーブ側通常プログラム

```

/*****/
/*                                     */
/* H8/300L Super Low Power Series      */
/*   -H8/38024 Series-                 */
/* Flash Memory Write/Erase Application Note */
/*                                     */
/* Communication Interface             */
/* : 8bit Parallel Interface           */
/* Function                             */
/* : Slave Main Program                */
/*                                     */
/* External Clock : 10MHz              */
/* Internal Clock : 5MHz                */
/* Sub Clock      : 32.768kHz           */
/*                                     */
/*****/
#include <machine.h>
#include "string.h"

/*****/
/* Symbol Definition                    */
/*****/
struct BIT {
    unsigned char  b7:1;    /* bit7 */
    unsigned char  b6:1;    /* bit6 */
    unsigned char  b5:1;    /* bit5 */
    unsigned char  b4:1;    /* bit4 */
    unsigned char  b3:1;    /* bit3 */
    unsigned char  b2:1;    /* bit2 */
    unsigned char  b1:1;    /* bit1 */
    unsigned char  b0:1;    /* bit0 */
};

#define  LPCR      *(volatile unsigned char *)0xFFC0 /* LCD Port Control Register */
#define  LCR       *(volatile unsigned char *)0xFFC1 /* LCD Control Register      */
#define  LCR2      *(volatile unsigned char *)0xFFC2 /* LCD Control Register 2    */
#define  LCDRAM    (volatile unsigned char *)0xF740 /* LCD RAM                    */
#define  PDR3_BIT  (*(struct BIT *)0xFFD6)          /* Port Data Register 3      */
#define  P37       PDR3_BIT.b7                      /* Port 37                    */

```

```

#define PMR5      *(volatile unsigned char *)0xFFCC /* Port Mode Register 5      */
#define PDR5      *(volatile unsigned char *)0xFFD8 /* Port Data Register 5      */
#define PDR5_BIT  (*(struct BIT *)0xFFD8)          /* Port Data Register 5      */
#define P55      PDR5_BIT.b5                       /* Port 55                    */
#define P54      PDR5_BIT.b4                       /* Port 54                    */
#define P53      PDR5_BIT.b3                       /* Port 53                    */
#define P52      PDR5_BIT.b2                       /* Port 52                    */
#define P51      PDR5_BIT.b1                       /* Port 51                    */
#define P50      PDR5_BIT.b0                       /* Port 50                    */
#define PDR6      *(volatile unsigned char *)0xFFD9 /* Port Data Register 6      */
#define PDR9_BIT  (*(struct BIT *)0xFFDC)          /* Port Data Register 9      */
#define P93      PDR9_BIT.b3                       /* Port 93                    */
#define P92      PDR9_BIT.b2                       /* Port 92                    */
#define PUCR5     *(volatile unsigned char *)0xFFE2 /* Port Pull-Up Control Register 5 */
#define PUCR6     *(volatile unsigned char *)0xFFE3 /* Port Pull-Up Control Register 6 */
#define PCR5      *(volatile unsigned char *)0xFFE8 /* Port Control Register 5   */
#define PCR6      *(volatile unsigned char *)0xFFE9 /* Port Control Register 6   */

#define ACK_SV    P55
#define BSY_SV    P54
#define STB_SV    P53
#define ACK_MT    P52
#define BSY_MT    P51
#define STB_MT    P50

/*****
/* Function define
/*****/
extern void INIT(void); /* SP Set */
extern void FZMAIN(void);
void main(void);
unsigned char SLrcvbyte(void);
void com_init(void);
extern unsigned char LCDDT1[6]; /* 0x0400 - 0x0405 Sample Data */
extern unsigned char LCDDT2[6]; /* 0x0800 - 0x0805 Sample Data */
extern unsigned char LCDDT3[6]; /* 0x0FAA - 0x0FFF Sample Data */

/*****
/* Vector Address
/*****/
#pragma section V1 /* VECTOR SECTOIN SET */
void (*const VEC_TBL1[])(void) = {
/* 0x00 - 0x0f */
INIT /* 00 Reset */
};

#pragma section /* P */
/*****
/* Main Program
/*****/
void main(void)
{
unsigned char i,tmp,tmp2;
unsigned char *lcdram,sw16cnt;
char *X_BGN;
char *X_END;

```

```

char *Y_BGN;

LPCR = 0xC9; /* 1/4 Duty / SEG32-SEG5 ON */
LCR = 0xFE; /* LCD ON */
LCR2 = 0x60;

lcdram = LCDRAM;
for(i = 0; i < 0x0F; i++){
    lcdram[i] = 0;
}

sw16cnt = 1; /* User Application Program Sample */
do{
    PMR5 = 0x00; /* Port5 set */
    PUCR5 = 0x00; /* Port PullUp Control Register5 OFF*/
    PCR5 = 0x38; /* P55-P53 Output/ P52-P50 Input */
    LPCR = 0xC9; /* 1/4 Duty / SEG32-SEG5 ON */

    do{
        if(sw16cnt == 1){
            for(i = 2; i < 6; i++){
                lcdram[i] = LCDDT1[i];
            }
        }
        else if(sw16cnt == 2){
            for(i = 2; i < 6; i++){
                lcdram[i] = LCDDT2[i];
            }
        }
        else if(sw16cnt == 3){
            for(i = 2; i < 6; i++){
                lcdram[i] = LCDDT3[i];
            }
        }
        else{
            for(i = 2; i < 6; i++){
                lcdram[i] = 0;
            }
        }

        sw16cnt++;
        if(sw16cnt > 3){
            sw16cnt = 1;
        }

        do{
            tmp = P37;
            tmp2 = STB_MT;
            tmp = tmp & (~tmp2);
        }while(tmp);
    }while(tmp2 == 0); /* Data Receive? */

    com_init(); /* Communication Initialize */
    tmp = SLrcv1byte();
}while(tmp != 0x55); /* Flash Memory Erase/Write Start? */

```

```

for(i = 0; i < 6; i++){
    lcdram[i] = 0;
}

/*----- Flash Memory Write Mode -----*/
P92 = 1; /* LED1 OFF */
P93 = 0; /* LED2 ON */

X_BGN = (char *)__sectop("FZTAT"); /* Flash , Ram Address Copy */
X_END = (char *)__secend("FZEND");
Y_BGN = (char *)__sectop("RAM");

memcpy(Y_BGN,X_BGN,X_END-X_BGN); /* Flash -> RAM Copy */

FZMAIN(); /* Flash Memory Write Main Program */
}

/*****
/* Receive 1 byte
/*****/
unsigned char SLrcv1byte(void)
{
    unsigned char tmp;

    ACK_SV = 0; /* Slave Acknowledge OFF */

    do{
        tmp = STB_MT;
    }while(tmp == 0); /* Master Strobe ON? */

    BSY_SV = 1; /* Slave Busy ON */
    ACK_SV = 1; /* Slave Acknowledge ON */
    tmp = PDR6; /* Receive Data (1byte) */
    BSY_SV = 0; /* Slave Busy OFF */

    return(tmp);
}

/*****
/* Communication Initialize
/*****/
void com_init(void)
{
    LPCR = 0x00; /* Port6,Port5 set */

    PMR5 = 0x00; /* Port5 set */
    PUCR5 = 0x00; /* Port PullUp Control Register5 OFF*/
    PCR5 = 0x38; /* P55-P53 Output/ P52-P50 Input */
    STB_SV = 0; /* Slave Strobe OFF */
    BSY_SV = 0; /* Slave Busy OFF */
    ACK_SV = 1; /* Slave Acknowledge ON */

    PUCR6 = 0x00; /* Port PullUp Control Register6 OFF*/
    PCR6 = 0x00; /* P67-P60 Input Set */
}

```

リンクアドレス指定

セクション名	アドレス
CV1	0x0000
P	0x0100
DLCDDT1	0x0400
DLCDDT2	0x0800
DLCDDT3	0x0FFA
FZTAT,PFZTAT,DFZTAT,FZEND	0x1000
RAM,PRAM,DRAM,B	0xF780

8.3 スレーブ側書き込み / 消去制御プログラム

```

/*****/
/*                               */
/* H8/300L Super Low Power Series */
/*   -H8/38024 Series-           */
/* Flash Memory Write/Erase Application Note */
/*                               */
/* Communication Interface       */
/* : 8bit Parallel Interface     */
/* Function                       */
/* : Slave Flash Memory Write/Erase Control Program */
/*                               */
/* External Clock : 10MHz        */
/* Internal Clock : 5MHz         */
/* Sub Clock      : 32.768kHz    */
/*                               */
/*****/
#pragma section FZTAT

#include <machine.h>
#include "string.h"

/***** WAIT TIME *****/
#define MHZ      5                /* 5MHz (10MHz/2)*/
#define KEISU    8                /* 1Loop 8Step <-- DEC.B(2)+MOV.B(2)+BNE(4) */

#define WLOOP1   1*MHZ/KEISU+1    /* LOOP WAIT TIME*/
#define WLOOP2   2*MHZ/KEISU+1
#define WLOOP4   4*MHZ/KEISU+1
#define WLOOP5   5*MHZ/KEISU+1
#define WLOOP10  10*MHZ/KEISU+1
#define WLOOP20  20*MHZ/KEISU+1
#define WLOOP50  50*MHZ/KEISU+1
#define WLOOP100 100*MHZ/KEISU+1
#define TIME10   10*MHZ/KEISU     /* WRITE WAIT TIME */
#define TIME30   30*MHZ/KEISU     /* WRITE WAIT TIME */
#define TIME200  200*MHZ/KEISU    /* WRITE WAIT TIME */
#define TIME1000 1000*MHZ/KEISU   /* ERASE WAIT TIME */

#define MAXBLK1  10
#define OK        0
#define NG        1
#define WNG       2

/*****/
/* Symbol Definition */
/*****/
struct BIT {
    unsigned char b7:1; /* bit7 */
    unsigned char b6:1; /* bit6 */
    unsigned char b5:1; /* bit5 */
    unsigned char b4:1; /* bit4 */
    unsigned char b3:1; /* bit3 */
    unsigned char b2:1; /* bit2 */
    unsigned char b1:1; /* bit1 */
    unsigned char b0:1; /* bit0 */
}

```

```

};

#define FLMCR1      *(volatile unsigned char *)0xF020 /* Flash Memory Control Register 1 */
#define FLMCR1_BIT  (*(struct BIT *)0xF020)          /* Flash Memory Control Register 1 */
#define SWE        FLMCR1_BIT.b6                    /* Software Write Enable */
#define ESU        FLMCR1_BIT.b5                    /* Erase Setup */
#define PSU        FLMCR1_BIT.b4                    /* Program Setup */
#define EV         FLMCR1_BIT.b3                    /* Erase Verify */
#define PV         FLMCR1_BIT.b2                    /* Program Verify */
#define ELS        FLMCR1_BIT.b1                    /* Erase */
#define PGM        FLMCR1_BIT.b0                    /* Program */
#define EBR        *(volatile unsigned char *)0xF023 /* Erase Block Register */
#define FENR       *(volatile unsigned char *)0xF02B /* Flash Memory Enable Register */
#define FENR_BIT   (*(struct BIT *)0xF02B)          /* Flash Memory Enable Register */
#define FLSHE      FENR_BIT.b7                      /* Flash Memory Control Register Enable */
#define TCSRW      *(volatile unsigned char *)0xFFB2 /* Timer Control/Status Register W */
#define TCSRW_BIT  (*(struct BIT *)0xFFB2)          /* Timer Control/Status Register W */
#define B6WI       TCSRW_BIT.b7                     /* Bit-6 Write Disable */
#define TCWE       TCSRW_BIT.b6                     /* Timer Counter W Write Enable */
#define B4WI       TCSRW_BIT.b5                     /* Bit-4 Write Disable */
#define TCSRWE     TCSRW_BIT.b4                     /* Timer Control/Status Register W Write Enable */
#define B2WI       TCSRW_BIT.b3                     /* Bit-2 Write Disable */
#define WDON       TCSRW_BIT.b2                     /* Watchdog Timer ON */
#define B0WI       TCSRW_BIT.b1                     /* Bit-0 Write Disable */
#define WRST       TCSRW_BIT.b0                     /* Watchdog Timer Reset */
#define TCW        *(volatile unsigned char *)0xFFB3 /* Timer Counter W */
#define PMR2       *(volatile unsigned char *)0xFFC9 /* Port Mode Register 2 */
#define PMR2_BIT   (*(struct BIT *)0xFFC9)          /* Prot Mode Register 2 */
#define WDCKS      PMR2_BIT.b2                      /* Watchdog Timer Source Clock */
#define PMR5       *(volatile unsigned char *)0xFFCC /* Port Mode Register 5 */
#define PDR5       *(volatile unsigned char *)0xFFD8 /* Port Data Register 5 */
#define PDR5_BIT   (*(struct BIT *)0xFFD8)          /* Port Data Register 5 */
#define P55        PDR5_BIT.b5                      /* Port 55 */
#define P54        PDR5_BIT.b4                      /* Port 54 */
#define P53        PDR5_BIT.b3                      /* Port 53 */
#define P52        PDR5_BIT.b2                      /* Port 52 */
#define P51        PDR5_BIT.b1                      /* Port 51 */
#define P50        PDR5_BIT.b0                      /* Port 50 */
#define PDR6       *(volatile unsigned char *)0xFFD9 /* Port Data Register 6 */
#define PDR9_BIT   (*(struct BIT *)0xFFDC)          /* Port Data Register 9 */
#define P93        PDR9_BIT.b3                      /* Port 93 */
#define P92        PDR9_BIT.b2                      /* Port 92 */
#define PUCR5      *(volatile unsigned char *)0xFFE2 /* Port Pull-Up Control Register 5 */
#define PUCR6      *(volatile unsigned char *)0xFFE3 /* Port Pull-Up Control Register 6 */
#define PCR5       *(volatile unsigned char *)0xFFE8 /* Port Control Register 5 */
#define PCR6       *(volatile unsigned char *)0xFFE9 /* Port Control Register 6 */

#define ACK_SV     P55
#define BSY_SV     P54
#define STB_SV     P53
#define ACK_MT     P52
#define BSY_MT     P51
#define STB_MT     P50

/*****
/* Function define

```



```

/*****/
void  FZMAIN(void);
unsigned char  rcv1byte(void);
void  rcvnbyte(unsigned char dtno,unsigned char *ram);
void  trslbyte(unsigned char tdt);
void  fwrite(unsigned char *buf,unsigned char *w_adr,unsigned char ptime);
char  fwritevf(unsigned char *owbuff,unsigned char *buff,unsigned char *w_adr,unsigned char *w_buf);
char  fwrite128(unsigned char *BUFF,unsigned char *OWBUFF,unsigned char *w_adr, unsigned char
*w_buf,unsigned short WT_COUNT);
char  blk_check(unsigned short ersad,unsigned short *evf_st, unsigned short *evf_ed, unsigned char
*blk_no);
void  ferase(unsigned char blk_no);
char  ferasevf(unsigned short evf_st, unsigned short evf_ed);
char  blk1_erase(unsigned short evf_st, unsigned short evf_ed, unsigned char blk_no, unsigned char
ET_COUNT);

unsigned short BLOCKADR1[10] ={
                                /* Erase Block Address          */
    0x0000,0x03ff,              /* EB0  1KBYTE          */
    0x0400,0x07ff,              /* EB1  1KBYTE          */
    0x0800,0x0bff,              /* EB2  1KBYTE          */
    0x0c00,0x0fff,              /* EB3  1KBYTE          */
    0x1000,0x7fff,              /* EB4  28KBYTE         */
};

#define  WDT_ERASE  0x00          /* Watch Dog Timer      */
#define  WDT_WRITE  0xFB         /* Watch Dog Timer      */
#define  OW_COUNT   6            /* Over Write Count     */

/*****
/*  Flash Memory Write Main Program
/*****/
void  FZMAIN(void)
{
    char  tmp1,rtn;
    unsigned char  rcvldt;
    unsigned short  ad_tmp;
    unsigned short  E_ADR[10];
    unsigned short  restsize;          /* Write Size          */
    unsigned char  tmp0[10];
    unsigned char  blk_no,i;
    unsigned short  evf_st,evf_ed;
    unsigned char  BUFF[128];          /* Retry Write Data Area  */
    unsigned char  W_BUF[128];        /* Write Data Area       */
    unsigned char  OWBUFF[128];       /* Over Write Data Area  */

    trslbyte(OK);                      /* SEND OF OK Code      */
/*----- Erase -----*/
    rcvnbyte(2,tmp0);                  /* RECEIVE ERASE BLOCK NUMBER  */
    if(tmp0[0] != 0x77)                /* Recive Code = 0x77?      */
        goto  ERRCASE;

    trslbyte(OK);                      /* SEND OF OK Code      */

    tmp1 = tmp0[1] << 1;
    rcvnbyte(tmp1,(unsigned char*)E_ADR); /* Recive ERASE BLOCK Address  */

```

```

FLSHE = 1;
for(i = 0; i < tmp0[1]; i++){
    rtn = blk_check(E_ADR[i],&evf_st,&evf_ed,&blk_no); /* CHECK BLOCK START ADDRESS */
    if(rtn != OK)
        goto ERRCASE;

    rtn = blk1_erase(evf_st,evf_ed,blk_no, 3); /* 1 block Erase */
    if(rtn != OK)
        goto ERRCASE;
}

trslbyte(OK); /* SEND OF OK Code */

/*----- Write Address / Size Recive -----*/

rcvldt = rcv1byte(); /* Recive 1byte Data -> RAM Area */

if(rcvldt != 0x88)
    goto ERRCASE;

for(i = 0; i < 10; i++){
    trslbyte(OK); /* SEND OF OK Code */

    rcvnbyte(4,tmp0); /* Recive Write Top Address & Size */

    if(tmp0[1] & 0x7F)
        goto ERRCASE;

    ad_tmp = tmp0[0]; /* Address Copy */
    ad_tmp <<= 8;
    ad_tmp = ad_tmp | tmp0[1];

    restsize = tmp0[2]; /* Size Copy */
    restsize <<= 8;
    restsize = restsize | tmp0[3];
    if(restsize == 0x0000){
        goto ERRCASE;
    }

    trslbyte(OK); /* SEND OF OK Code */

/*----- 128 byte Flash Memory Write -----*/

while(restsize != 0){
    trslbyte(0x11); /* SEND OF Request */

    if(restsize <= 128){ /* Receive WriteData from HOST */
        memset(W_BUF,0xFF,128); /* INITIALIZE RECEIVE BUFFER WITH 0xFF */
        rcvnbyte((unsigned char)restsize,W_BUF); /* "rtsize" byte Receive */
        restsize = 0;
    }
    else{
        rcvnbyte(128,W_BUF); /* 128byte Receive */
        restsize -= 128;
    }
}

```

```

rtn = fwrite128(BUFF,OWBUFF,(unsigned char*)ad_tmp,W_BUF,1000);
if(rtn != OK)
    goto ERRCASE;

    ad_tmp = ad_tmp+128;
}

trslbyte(OK);                                /* SEND OF OK Code          */
P92 = 0;                                     /* LED1 ON                  */
P93 = 1;                                     /* LED2 OFF                 */

PMR2 &= 0xFB;                               /* PMR2 WDCKS=0 phi/8192   */
TCSRW = 0x50;                               /* WDT STOP,TCW ENABLE     */
TCW = 0xFF;                                 /* INITIALIZED WDT COUNT   */
TCSRW = 0x56;                               /* WDT START               */

while(1);                                   /* OK End                   */

/*----- Error Case -----*/
ERRCASE:                                     /* Error Case              */
    trslbyte(NG);
    P92 = 0;                                 /* LED1 ON                  */
    P93 = 0;                                 /* LED2 ON                  */
    while(1);
}

/*****
/* Receive 1 byte
/*****/
unsigned char  rcvlbyte(void)
{
    unsigned char  tmp;

    ACK_SV = 0;                             /* Slave Acknowledge OFF   */

    do{
        tmp = STB_MT;
    }while(tmp == 0);                       /* Master Strobe ON?      */

    BSY_SV = 1;                             /* Slave Busy ON           */
    ACK_SV = 1;                             /* Slave Acknowledge ON   */
    tmp = PDR6;                             /* Receive Data (1byte)   */
    BSY_SV = 0;                             /* Slave Busy OFF         */

    return(tmp);
}

/*****
/* Receive N byte
/*****/
void  rcvnbyte(unsigned char dtno,unsigned char *ram)
{
    while(dtno--){                          /* dtno = 0 ?             */
        *ram = rcvlbyte();                  /* 1byte Receive Data -> RAM */
    }
}

```

```

        *ram++;
    }
}

/*****
/* Transmit 1 byte
/*****/
void trslbyte(unsigned char tdt)
{
    unsigned char tmp,i;

    PCR6 = 0xFF; /* P67-P60 Output Set */
    do{
        tmp = ACK_MT;
    }while(tmp == 1); /* Master Acknowledge ON ? */

    do{
        tmp = BSY_MT;
    }while(tmp == 1); /* Master Busy ON ? */

    STB_SV = 1; /* Slave Strobe ON */
    PDR6 = tdt; /* Transmit Data (1byte) */
    for(i = 0; i < 10; i++);
    STB_SV = 0; /* Slave Strobe OFF */

    do{
        tmp = BSY_MT;
    }while(tmp == 1); /* Master Busy ON ? */

    PCR6 = 0x00; /* P67-P60 Input Set */
}

/*****
/* Erase Block Check Routin
/*****/
char blk_check(unsigned short ersad,unsigned short *evf_st, unsigned short *evf_ed, unsigned char
*blk_no)
{
    unsigned char i;

    for(i = 0; ersad != BLOCKADR1[i]; i += 2){ /* COMPARE BLOCK_START_ADDRESS */
        if(MAXBLK1 < i) /* BLOCK NUMBER MAX? */
            return(NG); /* ERASE BLOCK ADDRESS ERROR */
    }

    *blk_no = i>>1; /* ERASE BLOCK NUMBER */
    *evf_st = BLOCKADR1[i]; /* ERASE START ADDRESS */
    i++;
    *evf_ed = BLOCKADR1[i]; /* ERASE END ADDRESS */

    return(OK);
}

/*****
/* Flash Memory 1 block Erase
/*****/

```

```

char   blk1_erase(unsigned short evf_st, unsigned short evf_ed, unsigned char blk_no, unsigned char
ET_COUNT)
{
    unsigned char   i;
    char   rtn;

    SWE = 1;                               /* Set the SWE bit           */
    for(i = 0; i < WLOOP1; i++);           /* Need to wait lusec       */

    rtn = ferasevf(evf_st, evf_ed);        /* Erase Verify             */

    for(i = 0; i < ET_COUNT; i++){         /* Count Check (Max Erase count) */
        if(!rtn)
            break;
        ferase(blk_no);                   /* Erase                     */
        rtn = ferasevf(evf_st, evf_ed);    /* Erase Verify             */
    }

    SWE = 0;                               /* Clear the SWE bit        */
    for(i = 0; i < WLOOP100; i++);        /* Need to wait 100usec     */
    return(rtn);
}

/*****
/* Erase
*****/
void   ferase(unsigned char blk_no)
{
    unsigned char   tmp;
    unsigned char   i;
    unsigned short  j;

    tmp = 1;
    tmp <<= blk_no;
    EBR = tmp;                               /* Set the EBR Erase Block bit */

    PMR2 &= 0xFB;                            /* PMR2 WDCKS=0 phi/8192     */
    TCSRW = 0x50;                             /* WDT STOP, TCW ENABLE      */
    TCW   = WDT_ERASE;                         /* INITIALIZED WDT COUNT     */
    TCSRW = 0x56;                             /* WDT START                 */

    ESU = 1;                                  /* Set the ESU bit           */
    for(i = 0; i < WLOOP100; i++);           /* Need to wait 100 usec     */

    ELS = 1;                                  /* Set the E bit (ERASE)     */
    for(j = 0; j < TIME10000; j++);         /* Need to wait 10 msec      */

    ELS = 0;                                  /* Clear the E bit           */
    for(i = 0; i < WLOOP10; i++);           /* Need to wait 10 usec      */

    ESU = 0;                                  /* Clear the ESU bit         */
    for(i = 0; i < WLOOP10; i++);           /* Need to wait 10 usec      */

    TCSRW = 0x52;                             /* WDT STOP                 */

    EBR = 0;

```

```

}

/*****
/* Erase Verify
*****/
char  ferasevf(unsigned short evf_st, unsigned short evf_ed)
{
    unsigned short  *tmp;
    unsigned char   i;

    EV = 1;                               /* Set the EV bit                */
    for(i = 0; i < WLOOP20; i++);         /* Need to wait 20 usec        */

    for(tmp = (unsigned short*)evf_st; tmp < (unsigned short*)evf_ed; tmp = (unsigned short*)(tmp+1)){
        *tmp = 0xFFFF;                   /* Perform dummy write         */
        for(i = 0; i < WLOOP2; i++);      /* Need to wait 2 usec         */

        if(*tmp != 0xFFFF){              /* Verify                       */
            EV = 0;                       /* Clear the EV bit            */
            for(i = 0; i < WLOOP4; i++);  /* Need to wait 4 usec        */
            return(NG);                   /* NG flag set                 */
        }
    }

    EV = 0;                               /* Clear the EV bit            */
    for(i = 0; i < WLOOP4; i++);         /* Need to wait 4 usec        */

    return(OK);                           /* OK flag set                 */
}

/*****
/* Flash Memory 128 byte Write
*****/
char  fwritel28(unsigned char *BUFF, unsigned char *OWBUFF, unsigned char *w_adr, unsigned char
*w_buf, unsigned short WT_COUNT)
{
    char  rtn;
    unsigned char  TM, i;
    unsigned short j;

    memcpy(BUFF, w_buf, 128);             /* W_BUF -> BUFF BLOCK COPY    */

    SWE = 1;                             /* Set the SWE bit             */
    for(i = 0; i < WLOOP1; i++);         /* Need to wait 1 usec        */

    rtn = fwritevf(OWBUFF, BUFF, w_adr, w_buf); /* 1st Program Verify          */
    if(rtn == NG){                       /* 1st Verify END             */
        TM = TIME30;
        for(j = 0; j < WT_COUNT; j++){
            fwrite(BUFF, w_adr, TM);      /* Input P Pulse(30 usec)     */
            rtn = fwritevf(OWBUFF, BUFF, w_adr, w_buf);

            if(j < OW_COUNT){            /* Count Check(additive Write Count)*/
                fwrite(OWBUFF, w_adr, TIME10);
            }
        }
    }
}

```

```

        else{
            TM = TIME200;                /* Input P Pulse(200 usec)      */
        }

        if(rtn != NG){
            break;                        /* NG Write Over Error      */
        }
    }
}

SWE = 0;                               /* Clear the SWE bit        */
for(i = 0; i < WLOOP100; i++);        /* Need to wait 100 usec    */
return(rtn);
}

/*****
/* Flash Memory Write
*****/
void fwrite(unsigned char *buf,unsigned char *w_adr,unsigned char ptime)
{
    unsigned char i;

    for(i = 0; i < 128; i++){          /* 128 byte repeat          */
        w_adr[i] = buf[i];            /* Rewrite data dummy write */
    }

    PMR2 &= 0xFB;                      /* PMR2 WDCKS=0 phi/8192    */
    TCSRW = 0x50;                       /* WDT STOP,TCW ENABLE      */
    TCW = WDT_WRITE;                    /* INITIALIZED WDT COUNT    */
    TCSRW = 0x56;                       /* WDT START                 */

    PSU = 1;                            /* Set the PSU bit          */
    for(i = 0; i < WLOOP50; i++);      /* Need to wait 50 usec     */

    PGM = 1;                            /* Set the P bit            */
    for(i = 0; i < ptime; i++);        /* Writing Time 10/30/200 usec */

    PGM = 0;                            /* Clear the P bit          */
    for(i = 0; i < WLOOP5; i++);      /* Need to wait 5 usec      */

    PSU = 0;                            /* Clear the PSU bit        */
    for(i = 0; i < WLOOP5; i++);      /* Need to wait 5 usec      */

    TCSRW = 0x52;                       /* WDT STOP                 */
}

/*****
/* Flash Memory Verify
*****/
char fwritevf(unsigned char *owbuff,unsigned char *buff,unsigned char *w_adr,unsigned char *w_buf)
{
    unsigned char i,j;
    unsigned char tmp;
    char rtn;

    PV = 1;                             /* Set the PV bit           */

```

```

for(i = 0; i < WLOOP4; i++); /* Need to wait 4 usec */

for(j = 0; j < 128; j++){
    w_adr[j] = 0xFF; /* Dummy Write */
    for(i = 0; i < WLOOP2; i++); /* Need to wait 2 usec */

    owbuff[j] = buff[j] | w_adr[j];

    tmp = ~w_adr[j];
    buff[j] = tmp | w_buf[j];

    tmp = tmp & w_buf[j]; /* Error Check */
    if(tmp != 0)
        break;
}

PV = 0; /* PV bit Clear */
for(i = 0; i < WLOOP2; i++); /* Need to wait 2 usec */

if(tmp == 0){
    j=0;
    rtn = OK;
    while(j < 128){ /* 128 byte OK? */
        if(buff[j++] != 0xFF){ /* Error Check */
            rtn = NG;
            break;
        }
    }
}
else{
    rtn = WNG; /* Write Error */
}

return(rtn);
}
#pragma section FZEND

```

リンクアドレス指定

セクション名	アドレス
CV1	0x0000
P	0x0100
DLCDDT1	0x0400
DLCDDT2	0x0800
DLCDDT3	0x0FFA
FZTAT,PFZTAT,DFZTAT,FZEND	0x1000
RAM,PRAM,DRAM,B	0xF780

8.4 マスタ側プログラム

```

/*****/
/*                               */
/* H8/300L Super Low Power Series */
/*   -H8/38024 Series-           */
/* Flash Memory Write/Erase Application Note */
/*                               */
/* Communication Interface       */
/* : 8bit Parallel Interface     */
/* Function                       */
/* : Master Main Program        */
/*                               */
/* External Clock : 10MHz       */
/* Internal Clock : 5MHz        */
/* Sub Clock      : 32.768kHz    */
/*                               */
/*****/
#include <machine.h>
#include "string.h"

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

/*****/
/* Symbol Definition             */
/*****/
struct BIT {
    unsigned char  b7:1;    /* bit7 */
    unsigned char  b6:1;    /* bit6 */
    unsigned char  b5:1;    /* bit5 */
    unsigned char  b4:1;    /* bit4 */
    unsigned char  b3:1;    /* bit3 */
    unsigned char  b2:1;    /* bit2 */
    unsigned char  b1:1;    /* bit1 */
    unsigned char  b0:1;    /* bit0 */
};

#define LPCR      *(volatile unsigned char *)0xFFC0 /* LCD Port Control Register */
#define PMR2      *(volatile unsigned char *)0xFFC9 /* Port Mode Register 2 */
#define PMR2_BIT  (*(struct BIT *)0xFFC9)          /* Prot Mode Register 2 */
#define IRQ0      PMR2_BIT.b0                      /* P43/IRQ0 Select */
#define PMR5      *(volatile unsigned char *)0xFFCC /* Port Mode Register 5 */
#define PDR5      *(volatile unsigned char *)0xFFD8 /* Port Data Register 5 */
#define PDR5_BIT  (*(struct BIT *)0xFFD8)          /* Port Data Register 5 */
#define P55       PDR5_BIT.b5                      /* Port 55 */
#define P54       PDR5_BIT.b4                      /* Port 54 */
#define P53       PDR5_BIT.b3                      /* Port 53 */
#define P52       PDR5_BIT.b2                      /* Port 52 */
#define P51       PDR5_BIT.b1                      /* Port 51 */
#define P50       PDR5_BIT.b0                      /* Port 50 */
#define PDR6      *(volatile unsigned char *)0xFFD9 /* Port Data Register 6 */
#define PDR9_BIT  (*(struct BIT *)0xFFDC)          /* Port Data Register 9 */
#define P93       PDR9_BIT.b3                      /* Port 93 */
#define P92       PDR9_BIT.b2                      /* Port 92 */
#define PUCR5     *(volatile unsigned char *)0xFFE2 /* Port Pull-Up Control Register 5 */

```

```

#define PUCR6      *(volatile unsigned char *)0xFFE3 /* Port Pull-Up Control Register 6 */
#define PCR5      *(volatile unsigned char *)0xFFE8 /* Port Control Register 5 */
#define PCR6      *(volatile unsigned char *)0xFFE9 /* Port Control Register 6 */
#define IEGR_BIT  (*(struct BIT *)0xFFF2)          /* Interrupt Edge Select Register 1 */
#define IEGR0     IEGR_BIT.b0                      /* IEGR0 Edge Select */
#define IENR1_BIT (*(struct BIT *)0xFFF3)          /* Interrupt Enable Register 1 */
#define IENR0     IENR1_BIT.b0                    /* IENR0 Interrupt Enable */
#define IRR1_BIT  (*(struct BIT *)0xFFF6)          /* Interrupt Request Register 1 */
#define IRR10     IRR1_BIT.b0                      /* IRR10 Interrupt Request Register */

#define ACK_SV    P55
#define BSY_SV    P54
#define STB_SV    P53
#define ACK_MT    P52
#define BSY_MT    P51
#define STB_MT    P50

#pragma interrupt (irq0int)
/*****
/* Function define
*****/
extern void INIT(void); /* SP Set */
void main(void);
void irq0int(void);
unsigned char rcv1byte(void);
void trslbyte(unsigned char tdt);
void trsnbyte(short dtno,unsigned char *tdt);
void com_init(void);

volatile char ramf;
extern unsigned char LCDDT1[0x0C00];
unsigned short BLOCKADR1[10] = { /* Erase Block Address */
    0x0000,0x03ff, /* EB0 1KBYTE */
    0x0400,0x07ff, /* EB1 1KBYTE */
    0x0800,0x0bff, /* EB2 1KBYTE */
    0x0c00,0x0fff, /* EB3 1KBYTE */
    0x1000,0x7fff, /* EB4 28KBYTE */
};

/*****
/* Vector Address
*****/
#pragma section V1 /* VECTOR SECTOIN SET */
void (*const VEC_TBL1[])(void) = {
/* 0x00 - 0x0f */
    INIT /* 00 Reset */
};
#pragma section V2 /* VECTOR SECTOIN SET */
void (*const VEC_TBL2[])(void) = {
    irq0int /* 08 IRQ0 Interrupt */
};

#pragma section /* P */
/*****
/* Main Program
*****/

```

```

void main(void)
{
    unsigned char  senddt[10],tmp;
    unsigned short wtsize;
    unsigned char  i;
    unsigned short j;
    union trsbk{
        unsigned char  b[10];
        unsigned short w[5];
    }bk;

    set_imask_ccr(1);                /* Interrupt Disable          */

    IEG0 = 0;                        /* Initialize IRQ0 Terminal Input Edge */
    IRRIO = 0;                       /* Initialize IRQ0 Interrupt Request Flag */
    IRQ0 = 1;
    IEN0 = 1;                        /* IRQ0 Interrupt Enable      */

    ramf = 1;
    set_imask_ccr(0);                /* Interrupt Enable          */
    while(ramf);                     /* Flash Memory Write Mode Check */
    IEN0 = 0;                        /* IRQ0 Interrupt Enable      */

    /*-----*/
    P92 = 1;                         /* LED1 OFF                  */
    P93 = 0;                         /* LED2 ON                   */

    com_init();
    STB_MT = 1;                      /* Master Strobe ON          */

    /*----- Flash Erase/Write Program Start -----*/

    trslbyte(0x55);                  /* Start Command            */

    tmp = rcvlbyte();
    if(tmp != OK)
        goto ERRCASE;

    /*----- Erase -----*/

    senddt[0] = 0x77;                /* Erase Command(0x77)      */
    senddt[1] = 0x03;                /* Erase Area Block Count   */
    trsnbyte(2,senddt);

    tmp = rcvlbyte();
    if(tmp != OK)                    /* Error Check              */
        goto ERRCASE;

    for(i = 0; i < senddt[1]; i++){  /* Erase Block No 1 & 2 & 3 */
        tmp = i+1;
        tmp <<= 1;
        bk.w[i] = BLOCKADR1[tmp];
    }
    trsnbyte(2*senddt[1],bk.b);     /* Send of Erase Block      */

    tmp = rcvlbyte();

```

```

if(tmp != OK)                                /* Error Check                */
    goto ERRCASE;

/*----- Send of Write Address & Size -----*/

trslbyte(0x88);                               /* SEND OF Write Command(0x88) */

tmp = rcvlbyte();
if(tmp != OK)                                /* Error Check                */
    goto ERRCASE;

bk.w[0] = 0x400;                              /* Write Address = 0x400       */
wtsize = 0x0c00;                              /* Write Size = 0x0c00 byte    */
bk.w[1] = wtsize;
trsnbyte(4,bk.b);                             /* Send of Address & Size     */

tmp = rcvlbyte();
if(tmp != OK)                                /* Error Check                */
    goto ERRCASE;

/*----- Send of Write Data -----*/

j = 0;
while(wtsize != 0){
    tmp = rcvlbyte();                          /* Recive of Request          */
    if(tmp != 0x11)                            /* Error Check                */
        goto ERRCASE;

    if(wtsize <= 128){
        tmp = wtsize;
        trsnbyte(tmp,&(LCDDT1[j]));
        wtsize = 0;
    }
    else{
        trsnbyte(128,&(LCDDT1[j]));
        j += 128;
        wtsize -= 128;
    }
}

tmp = rcvlbyte();
if(tmp != OK)                                /* Error Check                */
    goto ERRCASE;

P92 = 0;                                     /* LED1 ON                    */
P93 = 1;                                     /* LED2 OFF                    */
while(1);                                    /* OK End                      */

/*----- Error Case -----*/
ERRCASE:                                     /* Error Case                  */
    P92 = 0;
    P93 = 0;
    while(1);
}

/*****

```

```

/* IRQ0 Interrupt
/*****
void  irq0int(void)
{
    IRRIO = 0;                /* Initialize IRQ0 Interrupt Request Flag */

    ramf = 0;
}

/*****
/* Receive 1 byte
/*****
unsigned char  rcvlbyte(void)
{
    unsigned char  tmp;

    ACK_MT = 0;                /* Master Acknowledge OFF      */

    do{
        tmp = STB_SV;
    }while(tmp == 0);          /* Slave Strobe ON ?          */

    BSY_MT = 1;                /* Master Busy ON              */
    ACK_MT = 1;                /* Master Acknowledge ON      */
    tmp = PDR6;                /* Receive Data (1byte)       */
    BSY_MT = 0;                /* Master Busy OFF            */

    return(tmp);
}

/*****
/* Transmit 1 byte
/*****
void  trslbyte(unsigned char tdt)
{
    unsigned char  tmp,i;

    PCR6 = 0xFF;              /* P67-P60 Output Set        */

    do{
        tmp = ACK_SV;
    }while(tmp == 1);          /* Slave Acknowledge ON ?    */

    do{
        tmp = BSY_SV;
    }while(tmp == 1);          /* Slave Busy ON ?          */

    STB_MT = 1;                /* Master Strobe ON          */
    PDR6 = tdt;                /* Transmit Data (1byte)     */
    for(i = 0; i < 10; i++);
    STB_MT = 0;                /* Master Strobe OFF        */

    do{
        tmp = BSY_SV;
    }while(tmp == 1);          /* Slave Busy ON ?          */

    PCR6 = 0x00;              /* P67-P60 Input Set        */

```

```

}

/*****
/* Transmit N byte
/*****/
void trsnbyte(short dtno,unsigned char *tdt)
{
    while(dtno--){
        trslbyte((unsigned char)*tdt);
        *tdt++;
    }
}

/*****
/* Communication Initialize
/*****/
void com_init(void)
{
    LPCR = 0x00;
    PMR5 = 0x00;
    PUCR5 = 0x00;
    PCR5 = 0x07;
    STB_MT = 0;
    BSY_MT = 0;
    ACK_MT = 1;

    PUCR6 = 0x00;
    PCR6 = 0x00;
}

```

リンクアドレス指定

セクション名	アドレス
CV1	0x0000
CV2	0x0008
D	0x0100
DLCDDT1	0x0400
DLCDDT2	0x0800
DLCDDT3	0x0FFA
P	0x1000
B	0xF780

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.12.19	—	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注意ください。

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すとは、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を負いません。
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、その他お気づ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。