

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ (<http://www.renesas.com>)

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

ご注意

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。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。

アプリケーションノート

EEPROM リード・ライト

要旨

H8/3664 の I²C バスインタフェースを使用して、内蔵 EEPROM に 1 バイトのデータの書き込みまたは、読み込みをします。

動作確認デバイス

H8/300H Tiny シリーズ - H8/3664N -

目次

ご注意.....	2
1. 仕様.....	3
2. 使用機能説明.....	4
3. 動作説明.....	5
3.1 EEPROM インタフェース.....	5
3.2 バスフォーマットとタイミング.....	5
3.3 開始条件.....	5
3.4 停止条件.....	5
3.5 アクノリッジ.....	6
3.6 スレーブアドレス (本タスクでは初期状態のまま使用).....	6
3.7 ライト動作.....	7
3.8 アクノリッジポーリング.....	8
3.9 リード動作.....	9
4. ソフトウェア説明.....	11
4.1 モジュール説明.....	11
5. フローチャート.....	15
6. ヘッダーリスト.....	27
7. プログラムリスト.....	43

ご注意

1. 本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合、または国外に持ち出す場合は日本国政府の許可が必要です。
2. 本書に記載された情報の使用に際して、弊社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合、弊社はその責を負いませんので予めご了承ください。
3. 製品及び製品仕様は予告無く変更する場合がありますので、最終的な設計、ご購入、ご使用に際しましては、事前に最新の製品規格または仕様書をお求めになりご確認ください。
4. 弊社は品質・信頼性の向上に努めておりますが、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は、事前に弊社営業担当迄ご相談をお願い致します。
5. 設計に際しては、特に最大定格、動作電源電圧範囲、放熱特性、実装条件及びその他諸条件につきましては、弊社保証範囲内でご使用いただきますようお願い致します。保証値を越えてご使用された場合の故障及び事故につきましては、弊社はその責を負いません。また保証値内のご使用であっても半導体製品について通常予測される故障発生率、故障モードをご考慮の上、弊社製品の動作が原因でご使用機器が人身事故、火災事故、その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。
6. 本製品は耐放射線設計をしておりません。
7. 本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。
8. 本書をはじめ弊社半導体についてのお問い合わせ、ご相談は弊社営業担当迄お願い致します。

Copyright©Hitachi, Ltd., 2003. All rights reserved.

1. 仕様

1. H8/3664 の I²C バスインタフェースを使用して、内蔵 EEPROM に 1 バイトのデータの書き込み、または読み込みをします。

EEPROM 書き込み・読み込みのための

- ・ 1 バイト書き込み (Write_byte_EEPROM)
- ・ 1 バイト読み込み (Read_byte_EEPROM)
- ・ ページ (8 バイト) 書き込み (Write_page_EEPROM)
- ・ ページ (8 バイト) 読み込み (Read_page_EEPROM)
- ・ n (1 ~ 5 1 2) バイト連続読み込み (Read_n_EEPROM)

以上の関数を使用して EEPROM 512 バイト (アドレス H'000 ~ H'1FF) に 3 パターンの書き込みを行う。

- ・ 書き込みパターン : H'00, H'01, H'02 ~ H'FE, H'FF, H'00, H'01, H'02 ~ H'FE, H'FF
 - ・ 書き込みパターン : H'00, H'00, H'01, H'01, H'02, H'02 ~ H'FE, H'FE, H'FF, H'FF
 - ・ 書き込みパターン : H'FF, H'FF ~ H'FF, H'FF (ALL H'FF)
2. 接続する EEPROM のスレーブアドレスは [1010000] とし、EEPROM メモリアドレスの H'00 番地にデータを書き込みます。
 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 (EEPROM) 1 個の構成とします。図 1.1 に H8/3664 と EEPROM の接続例を示します。
 4. 転送クロックの周波数は 400kHz とします。

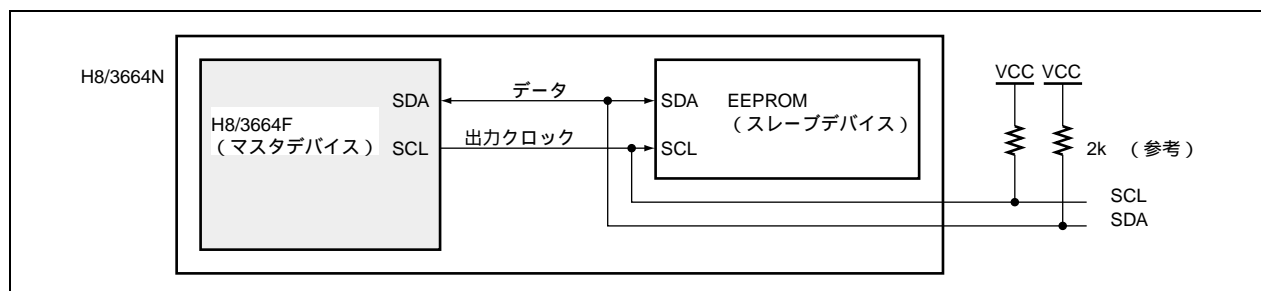


図 1.1 I²C バスインタフェース接続による EEPROM 制御

2. 使用機能説明

1. I²C バスインタフェース基本フォーマット (EEPROM バイトライト) を図 2.1 に示します。

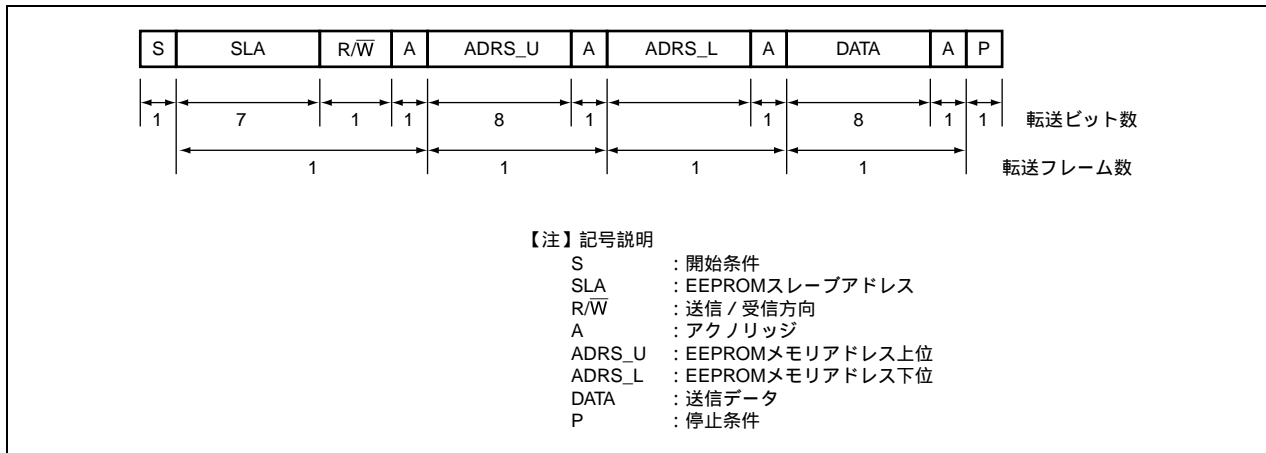


図 2.1 I²C バスインタフェースフォーマット

3. 動作説明

3.1 EEPROM インタフェース

H8/3664N は F-ZTATTTM 版 HD64F3664 と 512 バイト EEPROM の 2 チップを内蔵したマルチチップ構造の LSI です。

EEPROM のインタフェースは I²C バスインターフェースです。この I²C バスは外部にも開放されているため、I²C バスに接続される外部デバイスと通信できます。

3.2 バスフォーマットとタイミング

I²C バスフォーマットと I²C バスタイミングは、「I²C バスフォーマット」に従います。EEPROM の固有バスフォーマットは次の 2 点です。

1. EEPROM アドレスは 2 バイトで構成されており、ライトデータは上位アドレス、下位アドレスの順に各々の MSB 側から転送します。
2. ライトデータは MSB 側から送信します。

EEPROM のバスフォーマットとバスタイミングを図 3.1 に示します。

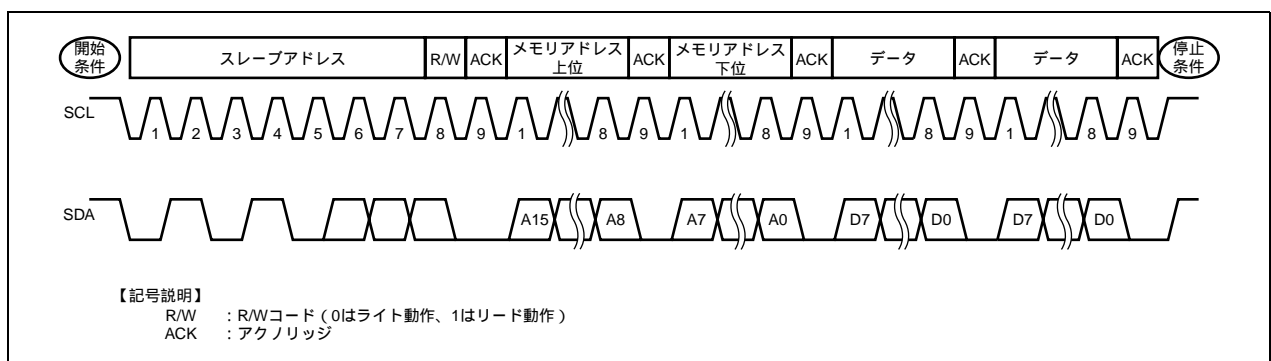


図 3.1 EEPROM バスフォーマットとバスタイミング

3.3 開始条件

リード、ライト動作を開始するには、SCL 入力が High レベルのとき SDA 入力を High レベルから Low レベルに変化させ、開始条件を生成する必要があります。

3.4 停止条件

リード、ライト動作を停止するには、SCL 入力が High レベルのとき SDA 入力を Low レベルから High レベルに変化させ、停止条件を生成する必要があります。

リード動作は停止条件を生成すると、リード動作を終了しアクセス待機状態になります。ライト動作は停止条件を生成すると、書き換えデータの入力を終了し、メモリへの書き込みをライトサイクル時間 (t_{wc}) の期間実施後、アクセス待機状態になります。

3.5 アクノリッジ

アドレス情報やリードデータ、ライトデータなどのシリアルデータは、8ビット単位で送受信が行われます。アクノリッジはこの8ビットデータが正常に送信または受信されたことを示す信号です。

ライト動作の場合、EEPROM はデータ受信完了後の9クロック目にアクノリッジ0を出力します。リード動作の場合、EEPROM はデータ受信完了後のアクノリッジに続いてリードデータを送信します。送信後はバス解放状態となり、アクノリッジ0を検出すると次のリードデータを送信します。アクノリッジ0を検出せずに停止条件を受信すると、リード動作を終了してアクセス待機状態となります。アクノリッジ0を検出せず、かつ停止条件も受信しない場合、データを送信せずにバス開放状態を保持します。

3.6 スレーブアドレス（本タスクでは初期状態のまま使用）

開始条件生成後、7ビットのスレーブアドレスと1ビットのR/Wコードを入力します。この入力ではEEPROMはリード、ライト動作を開始します。

スレーブアドレスは表3.1に示すように、前半のデバイスコード4ビットと後半のスレーブアドレスコード3ビットの7ビット構成です。デバイスコードはデバイスタイプを識別するコードで、本LSIでは汎用EEPROMと同じ1010に固定されています。スレーブアドレスコードはI²Cバスに接続されたデバイスコード1010のデバイス（最大8個）の中からどれを選択するかを決定します。A2,A1,A0の順に入力されたスレーブアドレスコードとスレーブアドレス照会用レジスタ（ESAR）の内容が一致したデバイスを選択します。

スレーブアドレスコードは、EEPROMのH'FF09番地に格納されています。リセット解除後10msの間に、メモリアレイのスレーブアドレスレジスタからESARに転送されます。なお、転送中、EEPROMのアクセスはできません。

EEPROMに書き込まれたスレーブアドレスコードの初期値はH'00です。H'00～H'07の範囲で書き換えることができます。なお、書き換えは必ずバイトライト方式で実行してください。（本タスクでは初期状態のまま使用）

スレーブアドレスの次の1ビットはR/Wコードです。0はライト動作、1はリード動作です。

なお、デバイスコードが1010ではない場合、もしくはスレーブアドレスコードが一致しない場合は、EEPROMはアクセス待機状態になります。

表 3.1 スレーブアドレス

ビット	ビット名	初期値	設定値	備 考
7	デバイスコード D3	—	1	
6	デバイスコード D2	—	0	
5	デバイスコード D1	—	1	
4	デバイスコード D0	—	0	
3	スレーブアドレスコード A2	0	A2	初期値の変更可能
2	スレーブアドレスコード A1	0	A1	初期値の変更可能
1	スレーブアドレスコード A0	0	A0	初期値の変更可能

3.7 ライト動作

ライト動作にはバイトライト、ページライトの2種類があります。ライト動作の起動方法はスレーブアドレスに続く R/W コードに 0 を入力します。

1. バイトライト

開始条件に続いてスレーブアドレス 7 ビットと R/W コード 0 の 8 ビットデータを入力すると、EEPROM は 9 ビット目にアクリッジ “0” を出力し、ライトモードに入ります。その後、メモリアドレス 2 バイトを上位、下位の順に MSB 側から入力します。メモリアドレス 1 バイトを入力するたびにアクリッジ 0 を出力するので、続いてライトデータ 1 バイトを MSB 側から入力します。ライトデータを受け取ると EEPROM はアクリッジ 0 を出力します。ここで停止条件を入力すると、EEPROM 内部で制御される書き換え動作に入り、書き換え動作終了まで、SCL、SDA の入力を受け付けなくなります。書き換え動作が終了すると、EEPROM は自動的にアクセス待機状態に戻ります。バイトライトの動作を図 3.2 に示します。

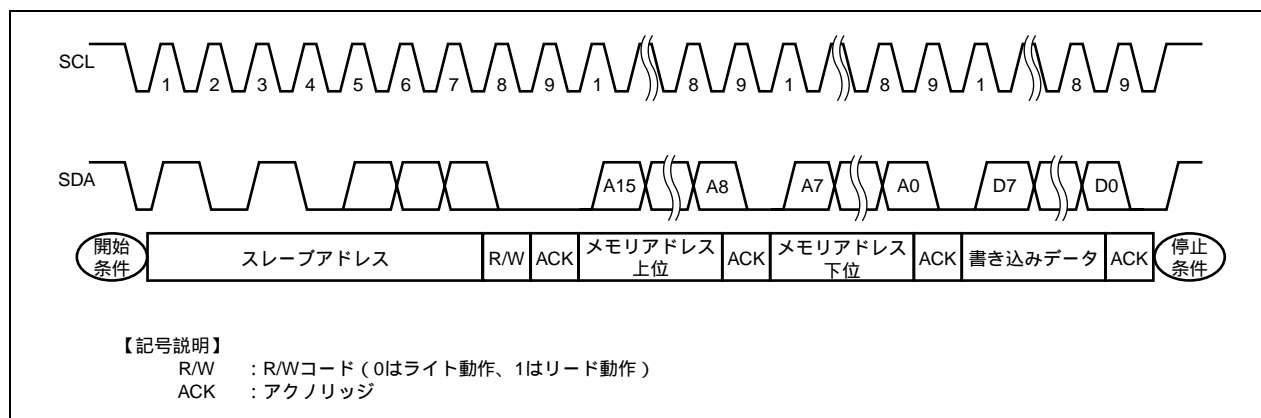


図 3.2 バイトライト動作

2. ページライト

本 LSI は 8 バイトまでの任意のバイト数を一度に書き換えられるページライト機能があります。ページライトではバイトライトと同様に、開始条件→スレーブアドレス+R/W コード→メモリアドレス (n) →ライトデータ (Dn) の順に、9 ビットごとのアクノリッジ “0” 出力を確認しながらライトデータを入力します。ライトデータ (Dn) 入力後に停止条件を入力せずライトデータ (Dn+1) を入力すると、ページライトモードに入ります。ライトデータ (Dn+1) を入力すると、EEPROM アドレスの LSB 側 3 ビット (A2~A0) は自動的にインクリメントされ (n+1) 番地になります。このようにしてライトデータを次々と入力することができます。

ライトデータの入力ごとにページ内アドレスがインクリメントされ、最大 8 バイトのライトデータを入力できます。EEPROM アドレスの LSB 側 3 ビット (A2~A0) がページの最終番地に達したとき、アドレスはロールオーバーしてページの先頭アドレスに戻ります。ロールオーバーした場合、同一アドレスにライトデータが二度もしくは二度以上入力されることとなりますが、最後に入力したライトデータが有効となります。停止条件を入力すると、ライトデータの入力を終了して書き換え動作に入ります。

ページライトの動作を図 3.3 に示します

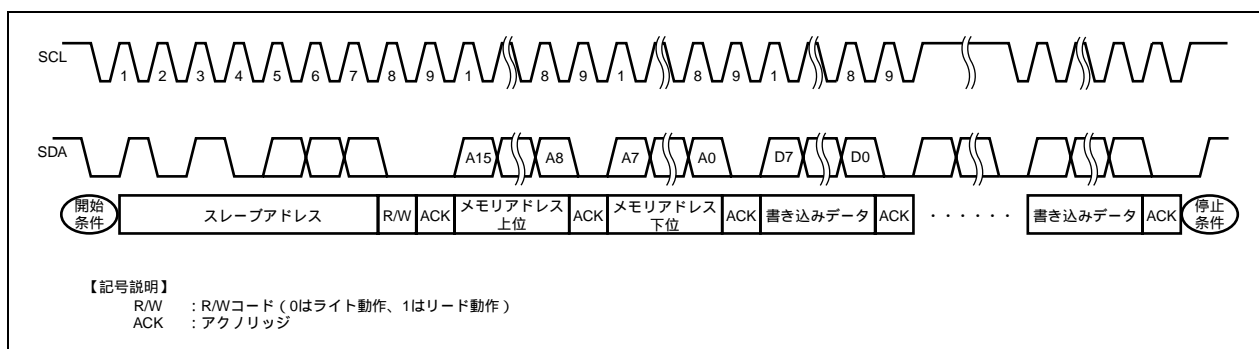


図 3.3 ページライト動作

3.8 アクノリッジポーリング

EEPROM が書き換え中か否かを判定するアクノリッジポーリング機能があります。書き換え期間中、開始条件に続いてスレーブアドレス+R/W コード 8 ビットを入力します。アクノリッジポーリングの場合、リード/ライトコードは 0 にしてください。9 ビット目のアクノリッジで書き換え中か否かを判定します。アクノリッジ “1” は書き換え中、アクノリッジ “0” は書き換え終了を示します。アクノリッジポーリングは、ライトデータの入力後、停止条件が出力された時点から機能します。

3.9 リード動作

リード動作にはカレントアドレスリード、ランダムアドレスリード、シーケンシャルリードの3種類があります。リード動作の起動方法はライトと同様ですが、スレーブアドレスに続く R/W コードには 1 を入力してください。

1. カレントアドレスリード

EEPROM 内部のアドレスカウンタは、前回のリード動作もしくはライト動作で最後にアクセスしたアドレス(n)を 1 番地インクリメントした (n+1) 番地を保持しています。カレントアドレスリードは、この内部アドレスカウンタが保持している (n+1) 番地をリードするモードです。

ライト動作と同様に、開始条件→スレーブアドレス+R/W コード (R/W=1) の順に入力すると、アクノリッジ “0” を出力した後に (n+1) 番地のデータ 1 バイトが MSB 側からシリアルに出力されます。この後、アクノリッジ “1” (アクノリッジの入力をせずにバスを解放しても可) →停止条件の順に入力するとリード動作を終了し、アクセス待機状態に戻ります。

なお、前회가リードモードで最終アドレス H'01FF 番地をアクセスして終了した場合、カレントアドレスはロールオーバーして 0 番地になります。また前회가ライトモードでページの最終アドレスをアクセスした場合、カレントアドレスはページ内でロールオーバーしてページの先頭アドレスになります。カレントアドレスは電源を OFF にしない限り有効です。電源 ON 後のカレントアドレスは不定になります。電源 ON 後のリードはランダムアドレスリードでアドレスを指定してください。

カレントアドレスリードの動作を図 3.4 に示します。

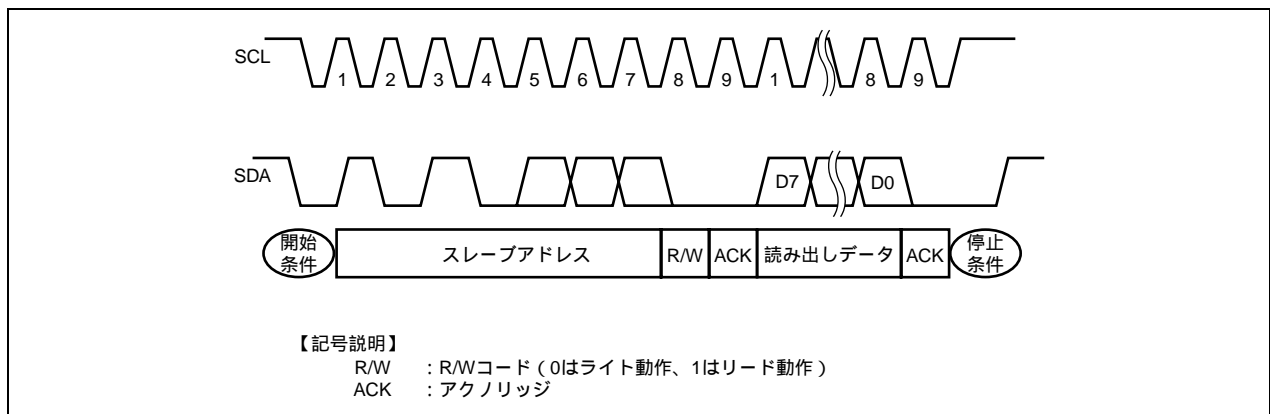


図 3.4 カレントアドレスリード動作

2. ランダムアドレスリード

アドレスを指定してリードするモードです。ダミーのライトモードでリードすべきアドレスを入力します。開始条件→スレーブアドレス+R/W コード (R/W=0) →メモリアドレス (上位) →メモリアドレス (下位) の順に入力します。メモリアドレス (下位) 入力後のアクリッジ “0” 出力確認後、再度開始条件を入力してカレントアドレスを実行すると、ダミーのライトモードで指定されたアドレスのデータを出力します。データ出力後に、アクリッジ “1” (アクリッジの入力をせずにバスを解放しても可) →停止条件の順で入力するとリードを終了し、アクセス待機状態に戻ります。ランダムアドレスリードの動作を図 3.5 に示します。

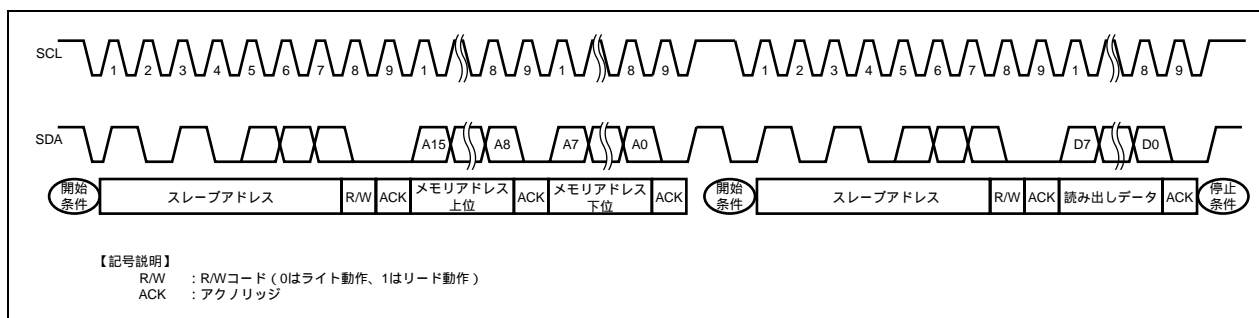


図 3.5 ランダムアドレスリード動作

3. シーケンシャルリード

データを連続してリードするモードです。カレントアドレスリードでもランダムアドレスリードでも、データを連続してリードできます。1 バイトのデータを出力した後、アクリッジ “0” を入力すると、アドレスがインクリメントされて次の 1 バイトのデータを出力します。データ出力後にアクリッジ “0” の入力を続けると、アドレスをインクリメントしながら次々とデータを出力します。アドレスが最終アドレス H'01FF 番地になった場合、0 番地にロールオーバーします。ロールオーバー後もシーケンシャルリードは可能です。動作を終了するには、カレントアドレスリード、ランダムアドレスリードと同様に、アクリッジ “1” (アクリッジの入力をせずにバスを解放しても可) →停止条件の順で入力します。

カレントアドレスリードを行ったときにシーケンシャルリードを使用した様子を図 3.6 に示します。

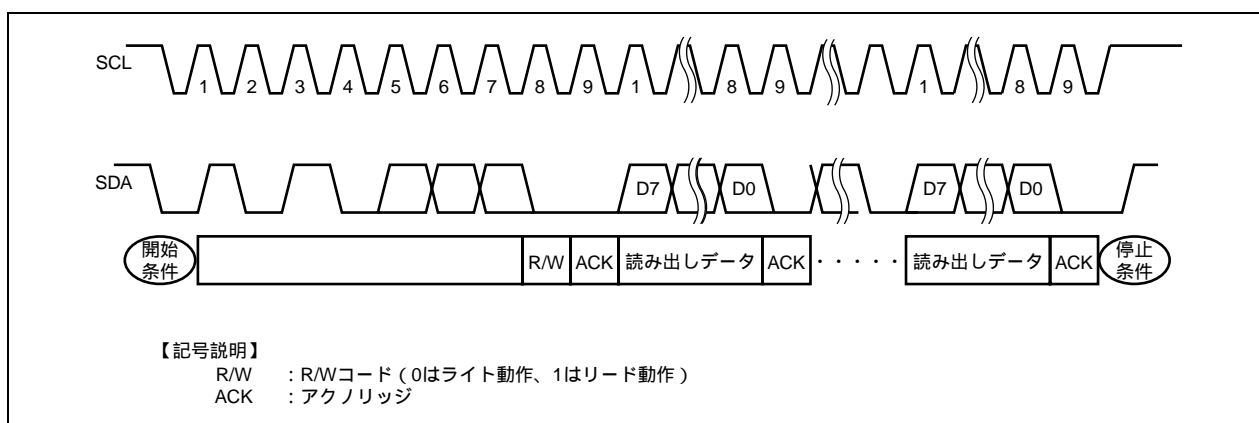


図 3.6 シーケンシャルリード動作 (カレントアドレスリードを使用した場合)

4. ソフトウェア説明

4.1 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 4.1 に示します。

表 4.1 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（割り込み禁止） mainへジャンプする
main	なし	なし	メインモジュール
Test_EEPROM	test_code (テスト項目：0～5)	なし	EEPROM動作テスト項目 0：処理なし 1：1バイト書き込み・1バイト読み込み 書き込みデータ：H'00～H'FF 繰り返し 2：8バイト書き込み 書き込みデータ：H'00、H'00～H'FF、H'FF 3：8バイト読み込み 4：8バイト書き込み 書き込みデータ：ALL H'FF 5：512バイト読み込み
wait	ウェイトループ数	なし	ウェイト (100：約100μsec)
Write_byte_EEPROM	adrs（書き込みアドレス） data（書き込みデータ）	ack（0： noack/1:ack）	EEPROM1バイト書き込み アクノリッジをチェックして書き込み不可能状態 (ACK=1)のときは1msecウェイトし、 10回リトライする
Read_byte_EEPROM	adrs（読み込みアドレス）	data（読み込み データ）	EEPROM1バイト読み込み
Write_page_EEPROM	adrs（書き込みアドレス）wr_ptr（書 き込みデータ格納アドレス）	ack（0： noack/1:ack）	EEPROM8バイト書き込み アクノリッジをチェックして書き込み不可能状態 (ACK=1)のときは1msecウェイトし、 10回リトライする
Read_page_EEPROM	adrs（読み込みアドレス）rd_ptr（読 み込みデータ格納アドレス）	ack（0： noack/1:ack）	EEPROM8バイト読み込み
Read_n_EEPROM	adrs（読み込みアドレス）rd_ptr（読 み込みデータ格納アドレス）、 リード数	ack（0： noack/1:ack）	EEPROMnバイト読み込み
Set_adrs_EEPROM	adrs（設定アドレス）	ack（0： noack/1:ack）	書き込み時のEEPROMアドレス指定、 読み込み時のEEPROMアドレス指定 (ダミーライト)
Write_data_EEPROM	wr_data（書き込みデータ）	ack（0： noack/1:ack）	書き込みデータのI ² C送信
Recv_data1_EEPROM	なし	data（読み込み データ）	I ² C1バイト受信
Recv_datan_EEPROM	adrs（読み込みアドレス）data（読み込 みデータ格納アドレス）	ack（0： noack/1:ack）	I ² Cnバイト受信
wait_e	ウェイトループ数	なし	ウェイト (100：約100μsec)

本タスク例におけるグローバル変数の説明を表 4.2 に示します。

表 4.2 使用グローバル変数説明

変数名	型	サイズ	用途
eeprom_buf	unsigned char	512	EEPROM 書き込み・読み込み用バッファ (内蔵 EEPROM と同じサイズ)
test_code	unsigned char	1	EEPROM 動作テスト項目指示
dummy	unsigned char	1	内蔵レジスタダミーリード用エリア

本タスク例における定数 (define 定義) の説明を表 4.3 に示します。

表 4.3 使用グローバル変数説明

変数名	型	サイズ	用途
DEVICE_CODE	unsigned char	0xA0	EEPROM (スレーブ) に送信するするための デバイスコード (固定[bit7-0]:1010 ----)
SLAVE_ADRS	unsigned char	0x00	EEPROM (スレーブ) に送信するするための アドレスコード (デフォルト[bit7-0] : ---- 000-)
IIC_DATA_W	unsigned char	0x00	EEPROM に書き込み指示コード (W[bit7-0] : ---- ---0)
IIC_DATA_R	unsigned char	0x01	EEPROM に読み込み指示コード (R[bit7-0] : ---- ---1)
WR_RETRY_CNT	unsigned char	10	EEPROM に書き込みサイクルリミット 書き込み後 1 m s 毎に ACK が受信するまで 10 回繰返す
WR_OK	unsigned char	11	書き込み後 ACK が受信した時点でループ処理を 抜けるための値

本タスク例で使用する内部レジスタ説明を表 4.4 に示します。

表 4.4 使用内部レジスタ説明

レジスタ名		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	—
SAR	FS	SARX の FSX ビット、DDCSWR の SW ビットとともに、転送フォーマットを設定	未操作	0
SARX	FSX	SAR の FS ビット、DDCSWR の SW ビットとともに、転送フォーマットを設定	未操作	1
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 CSK1=0 CSK0=1
	BC2 to BC0	I ² C 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、I ² C バスインタフェースの動作 (SCL/SDA 端子はポート機能) /非動作 (SCL/SDA 端子はバス駆動状態) の選択	設定	0/1
	IEIC	I ² C バスインタフェース割り込み要求を禁止	設定	0
	MST	I ² C バスインタフェースをマスタモードで使用	設定	1
	TRS	I ² C バスインタフェースを送信モードで使用	設定	0/1
	ACKE	アクリッジビットが"1"の場合、連続的な転送を中断	設定	0/1
	BBSY	I ² C 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リッジデータを格納	参照	—
TSCR	IICRST	I ² C コントロール部リセット	未操作	—
	IICX	転送レート選択	設定	—

モジュール階層図を図 4.1 に示します。

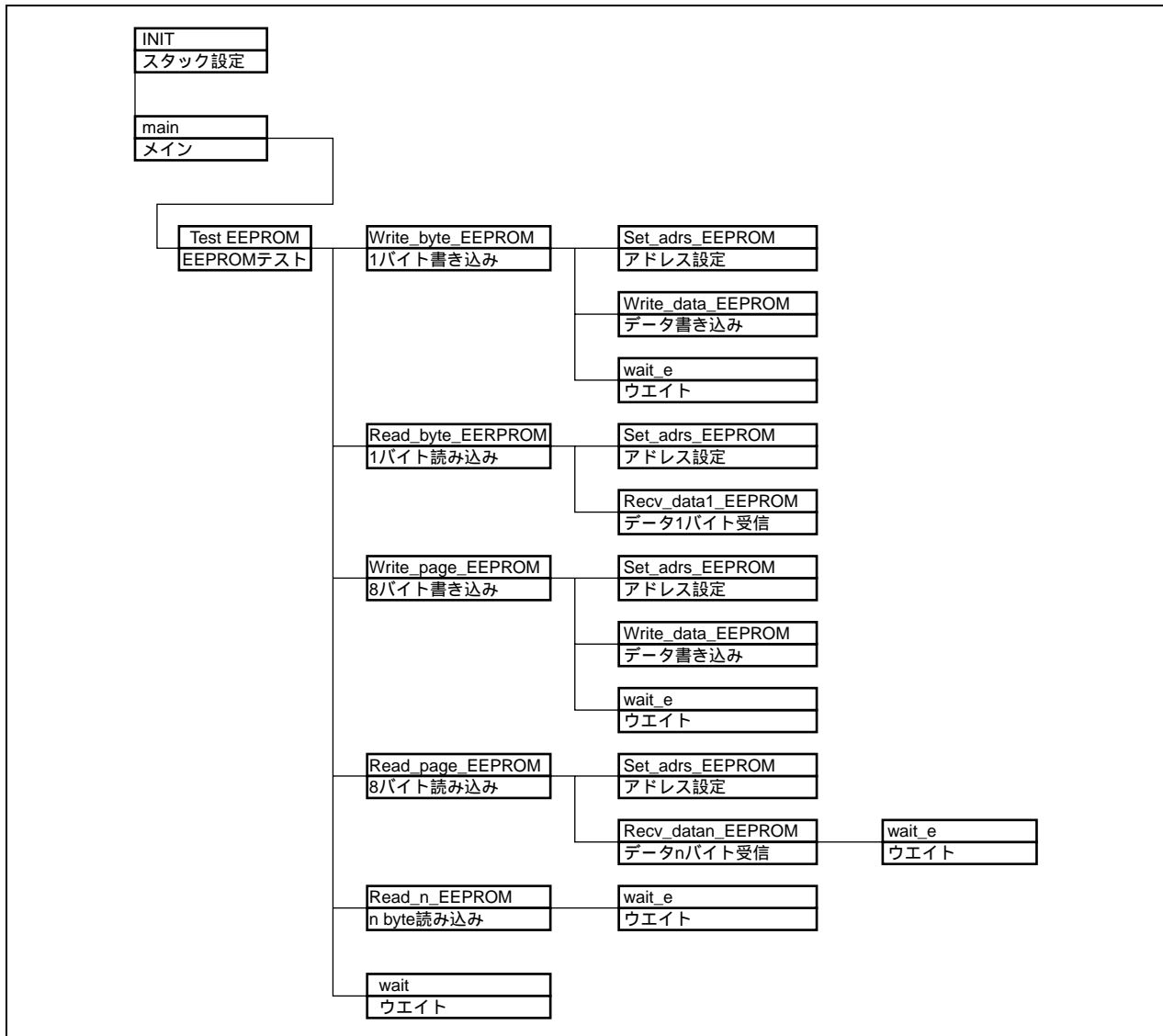
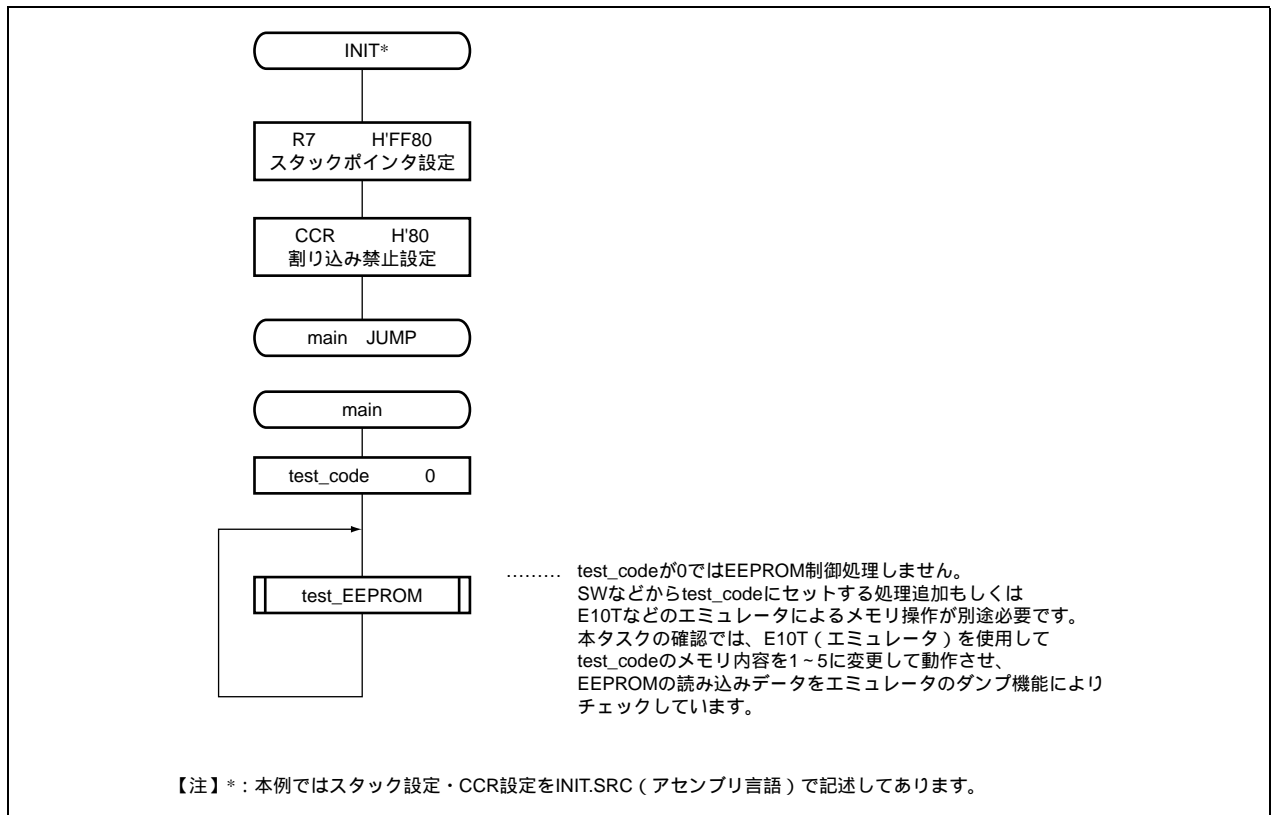
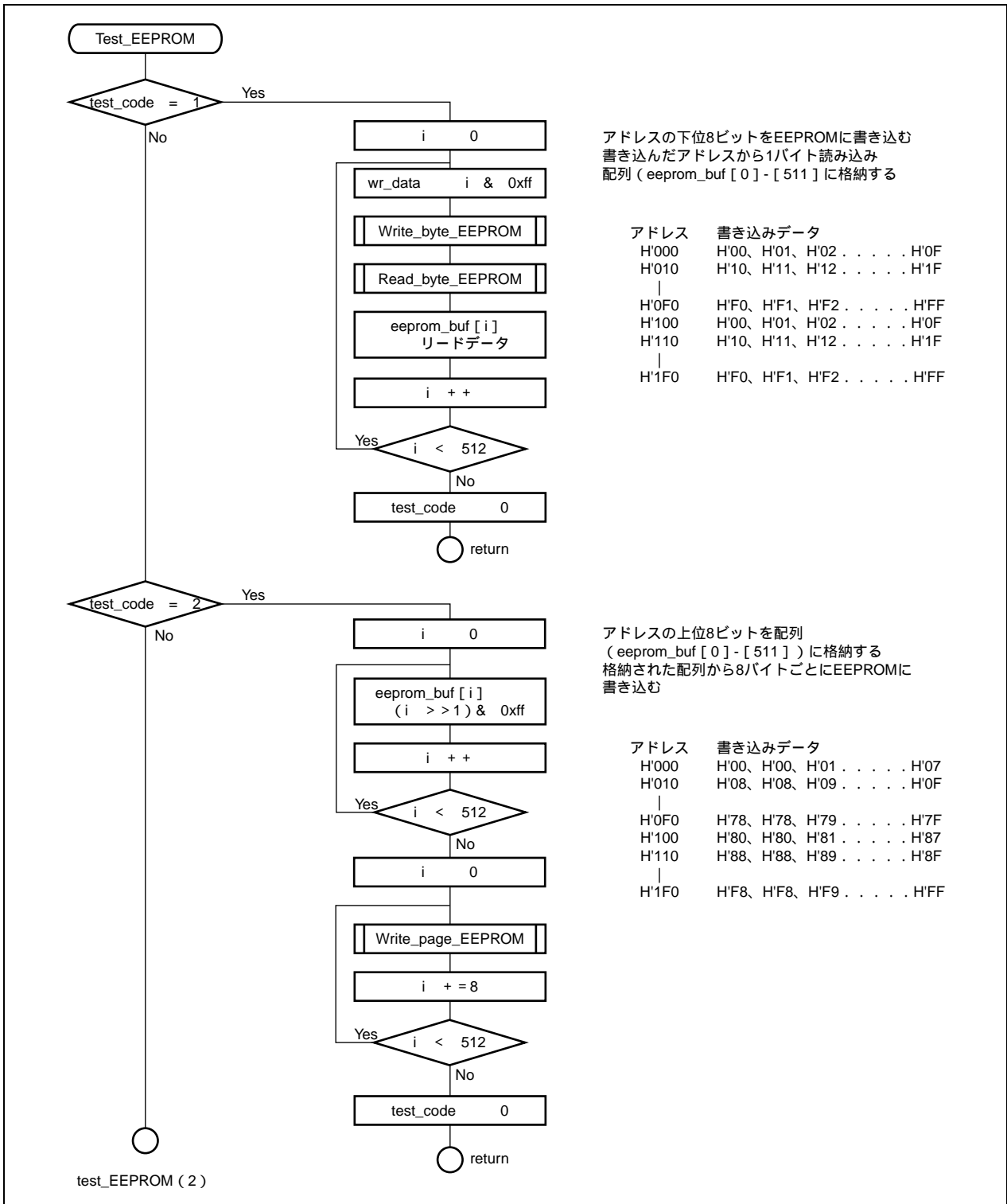
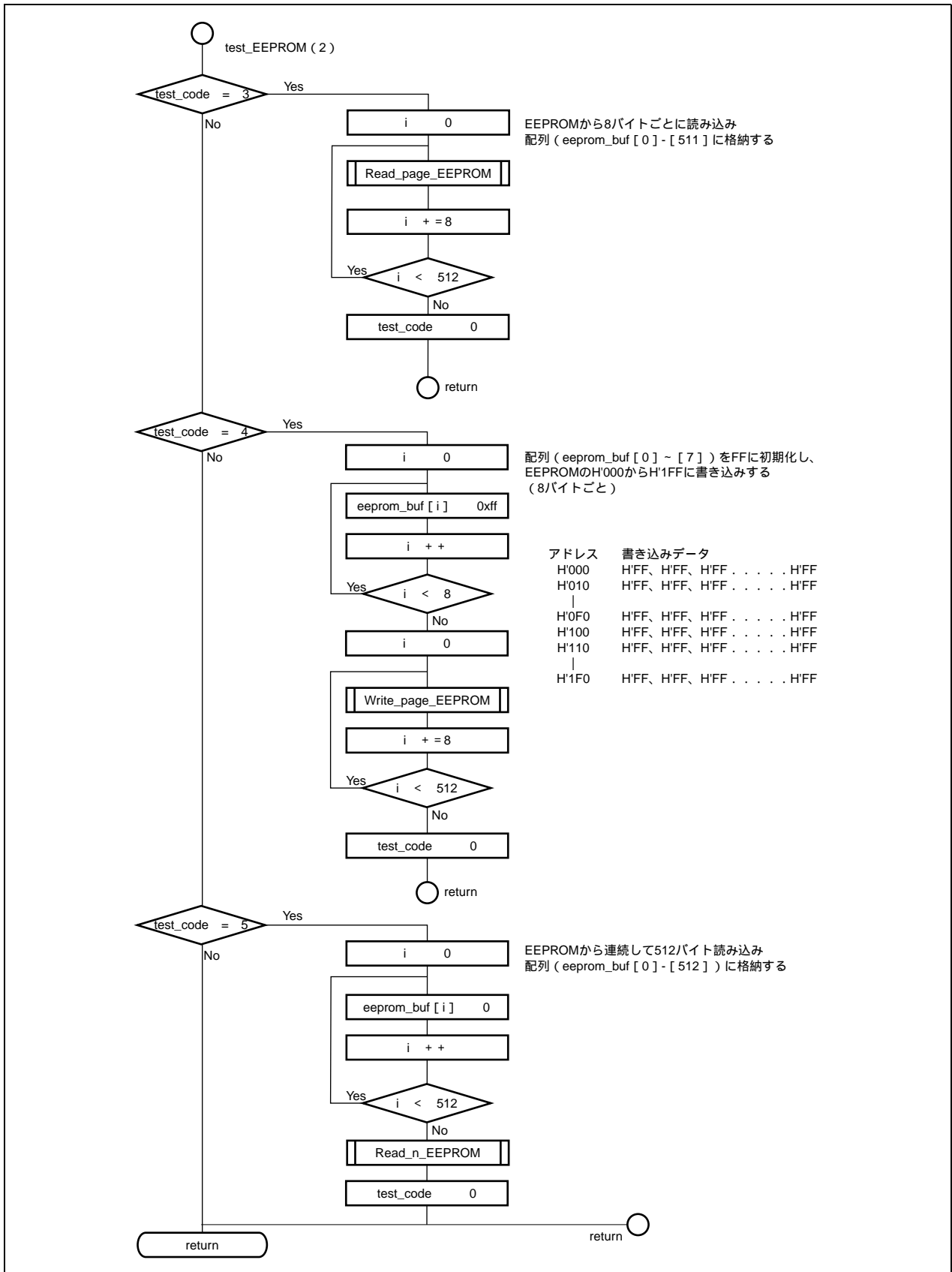


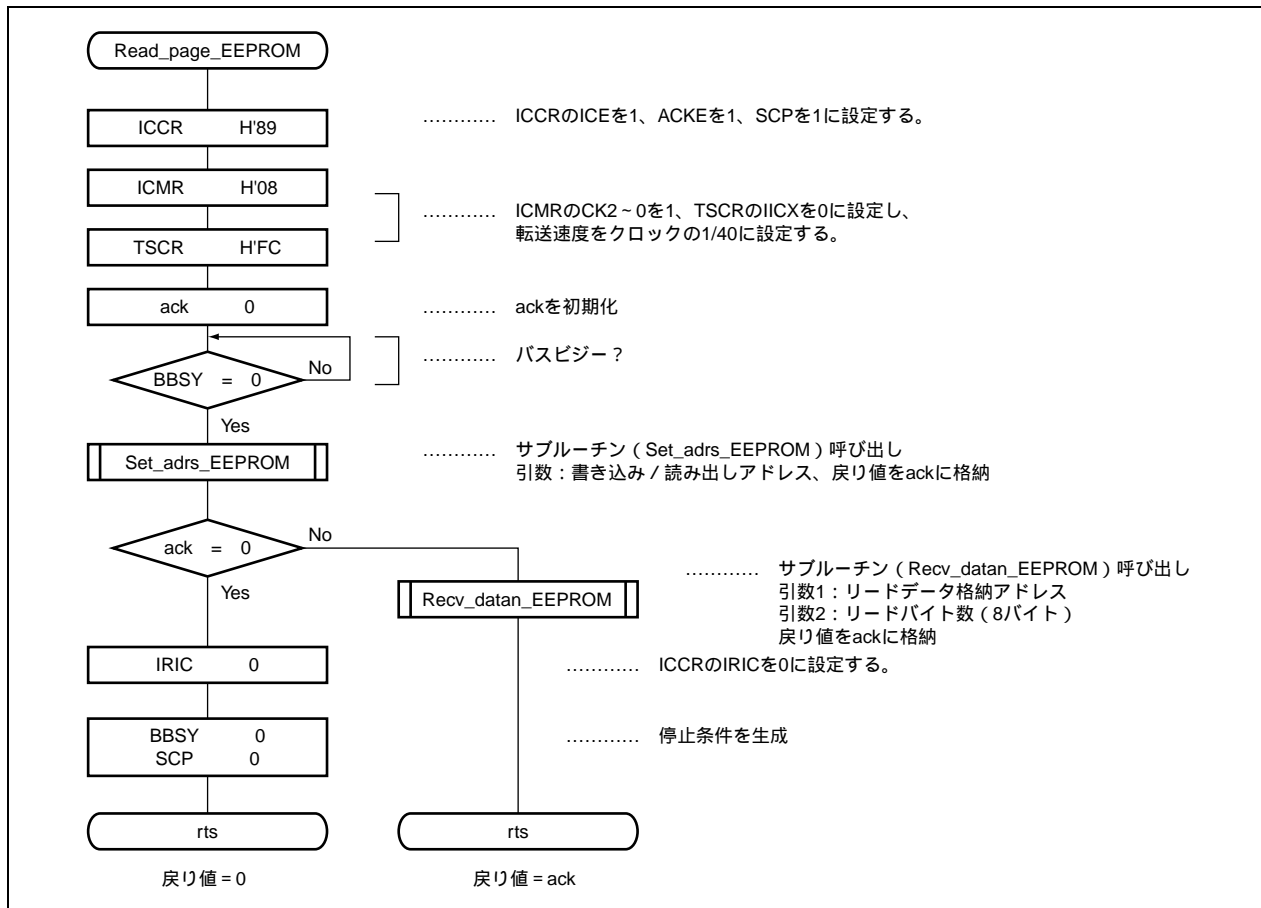
図 4.1 モジュール階層図

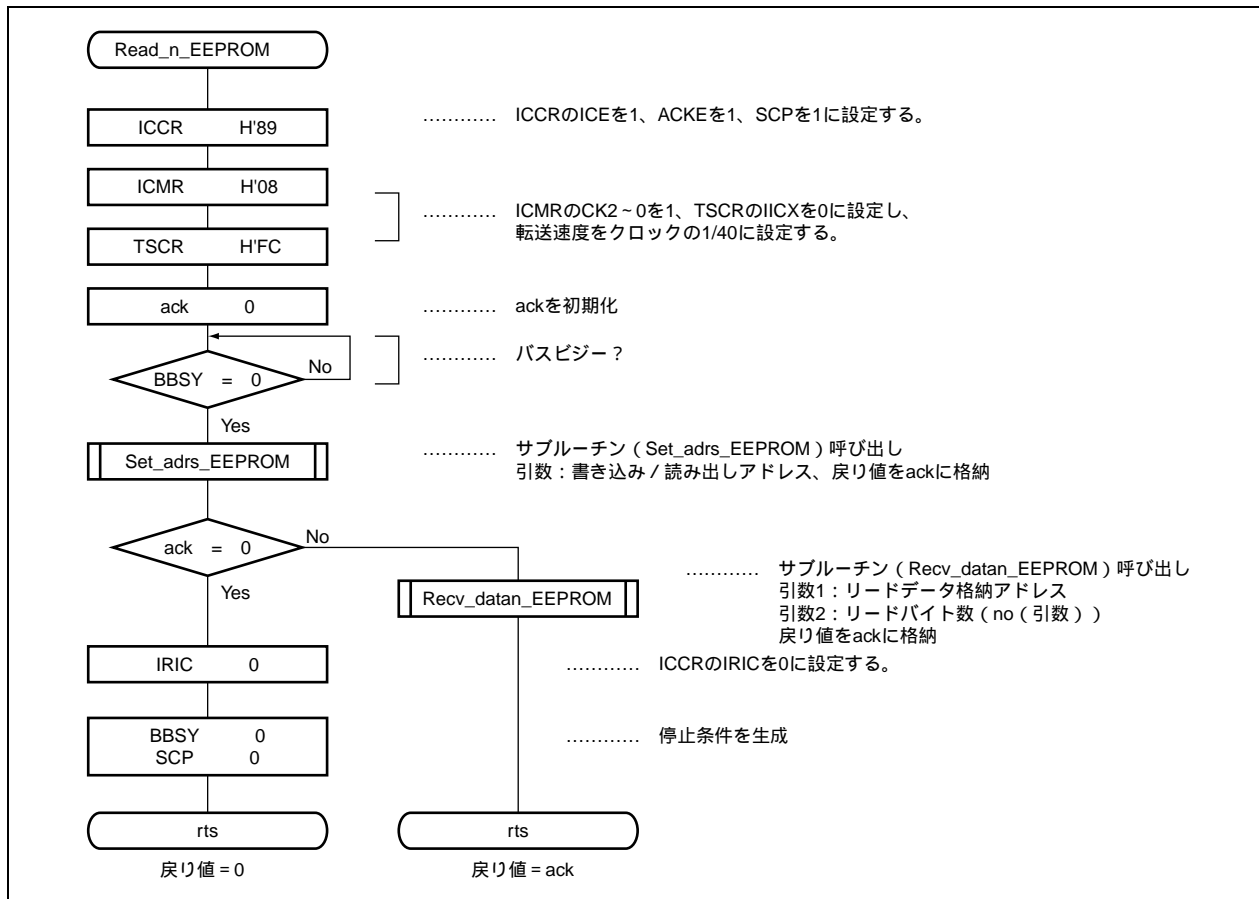
5. フローチャート

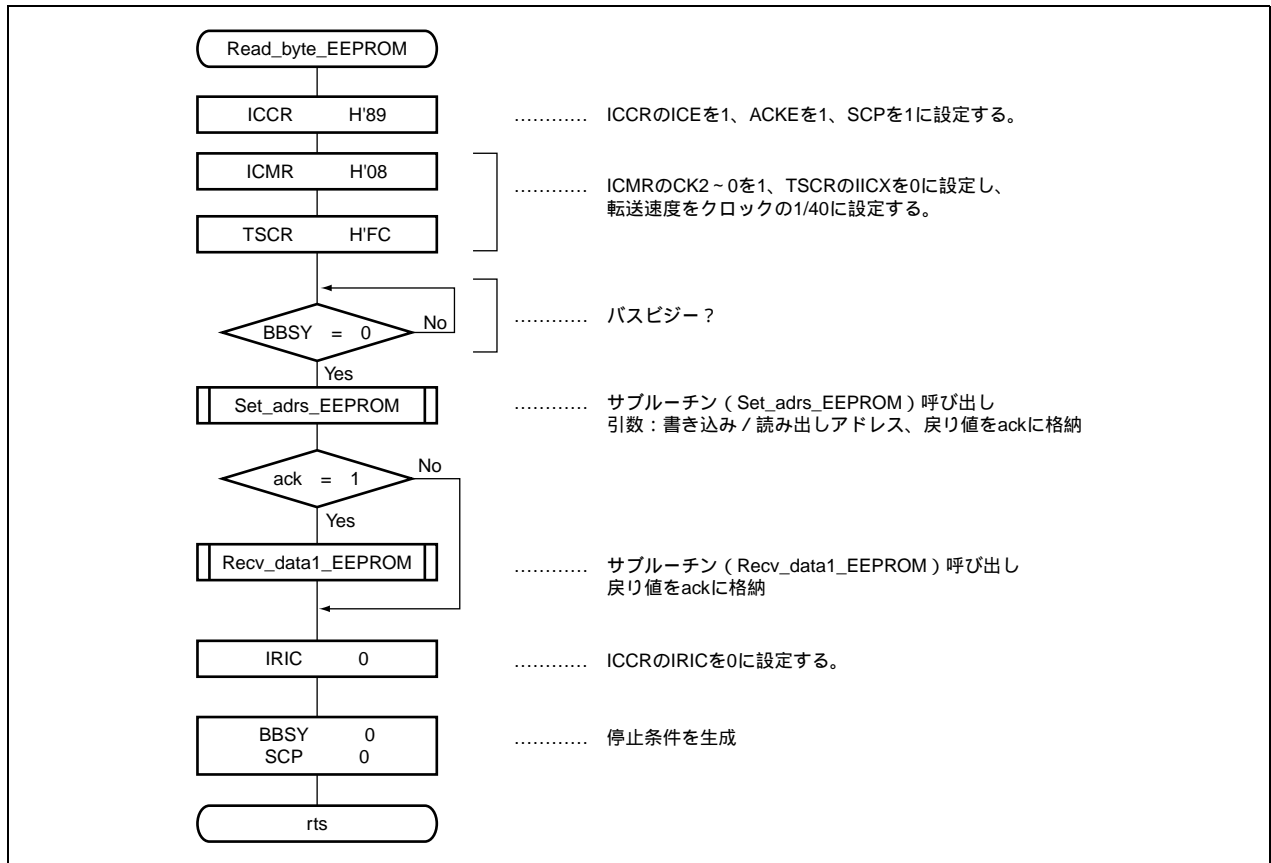


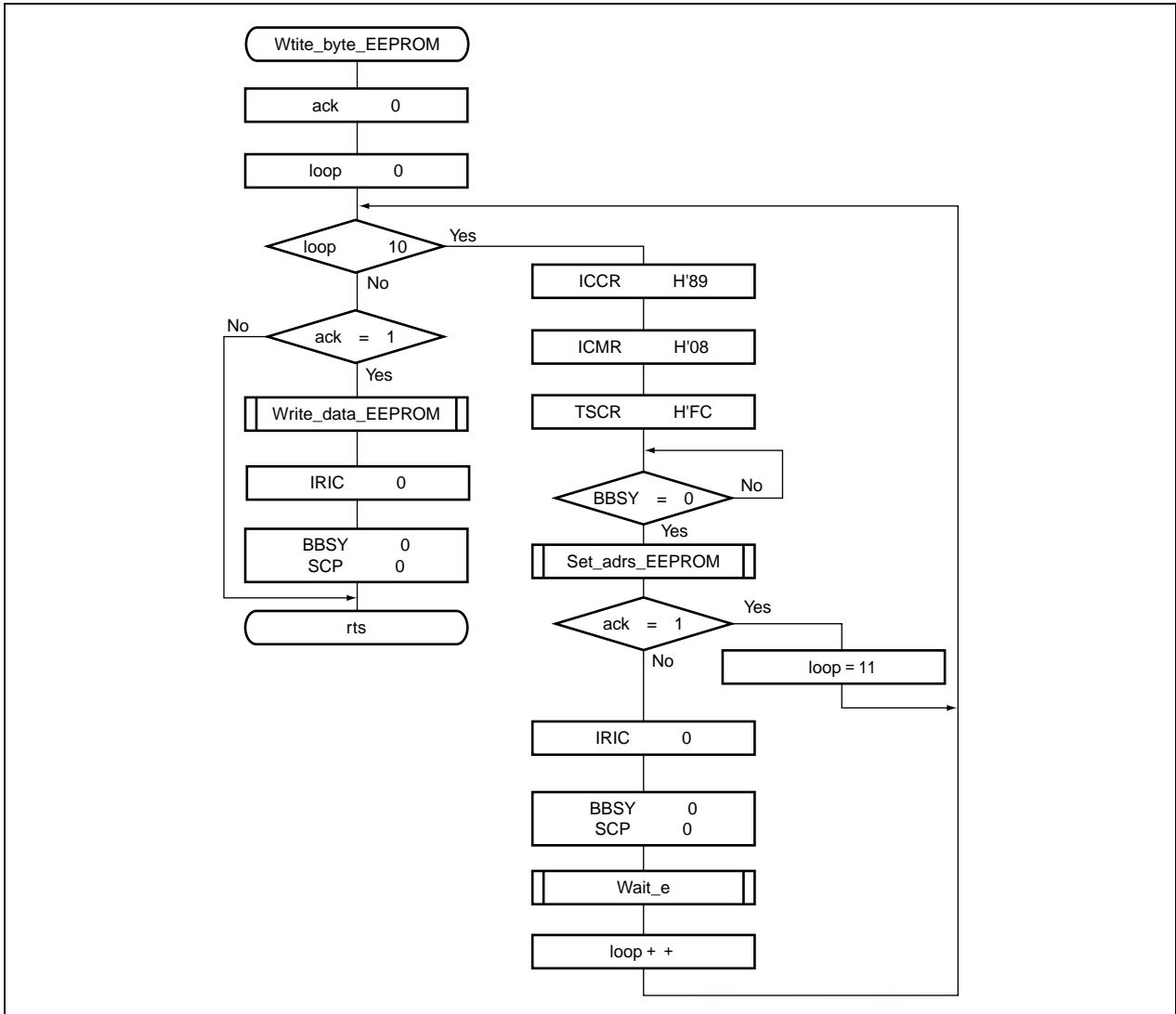


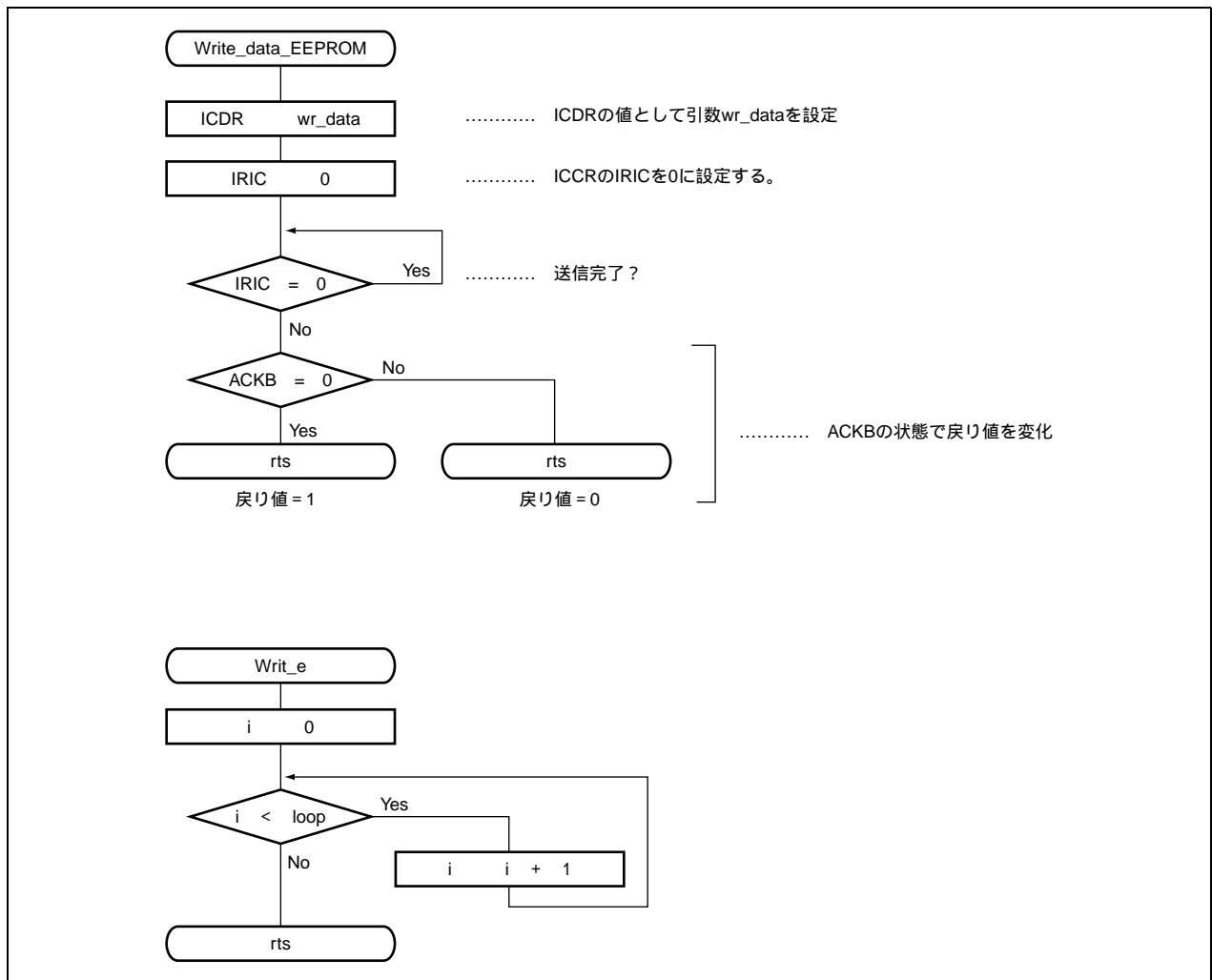


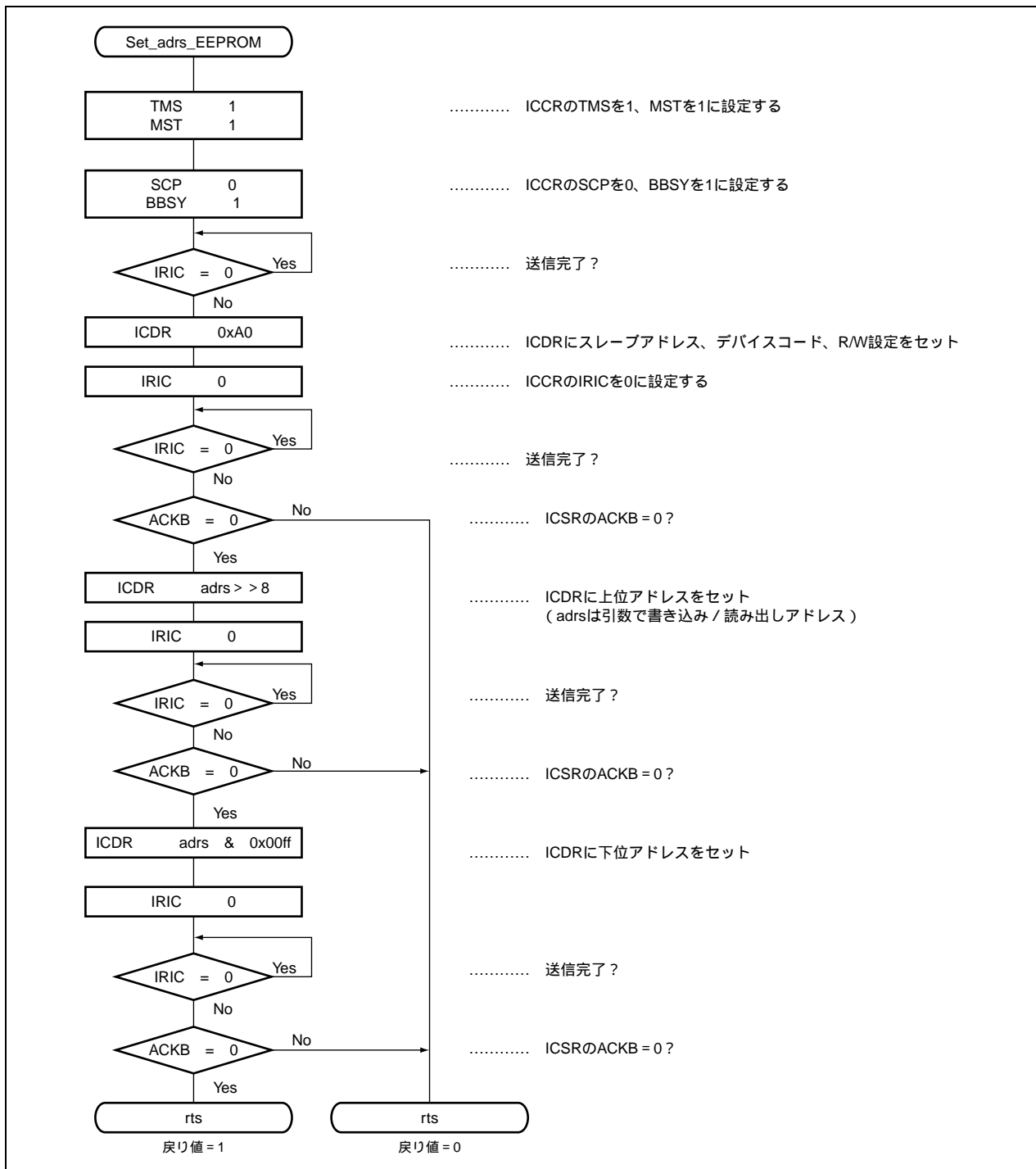


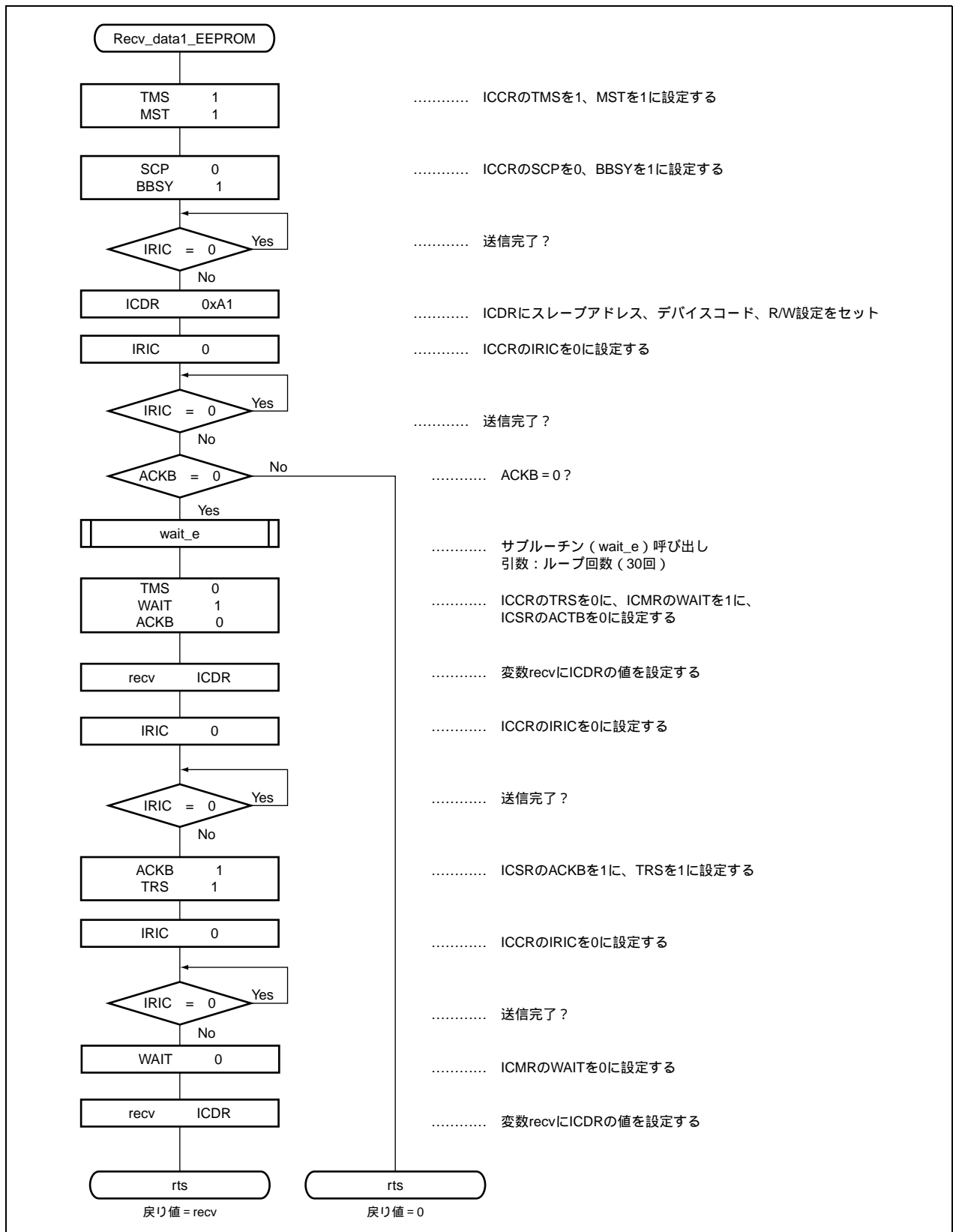


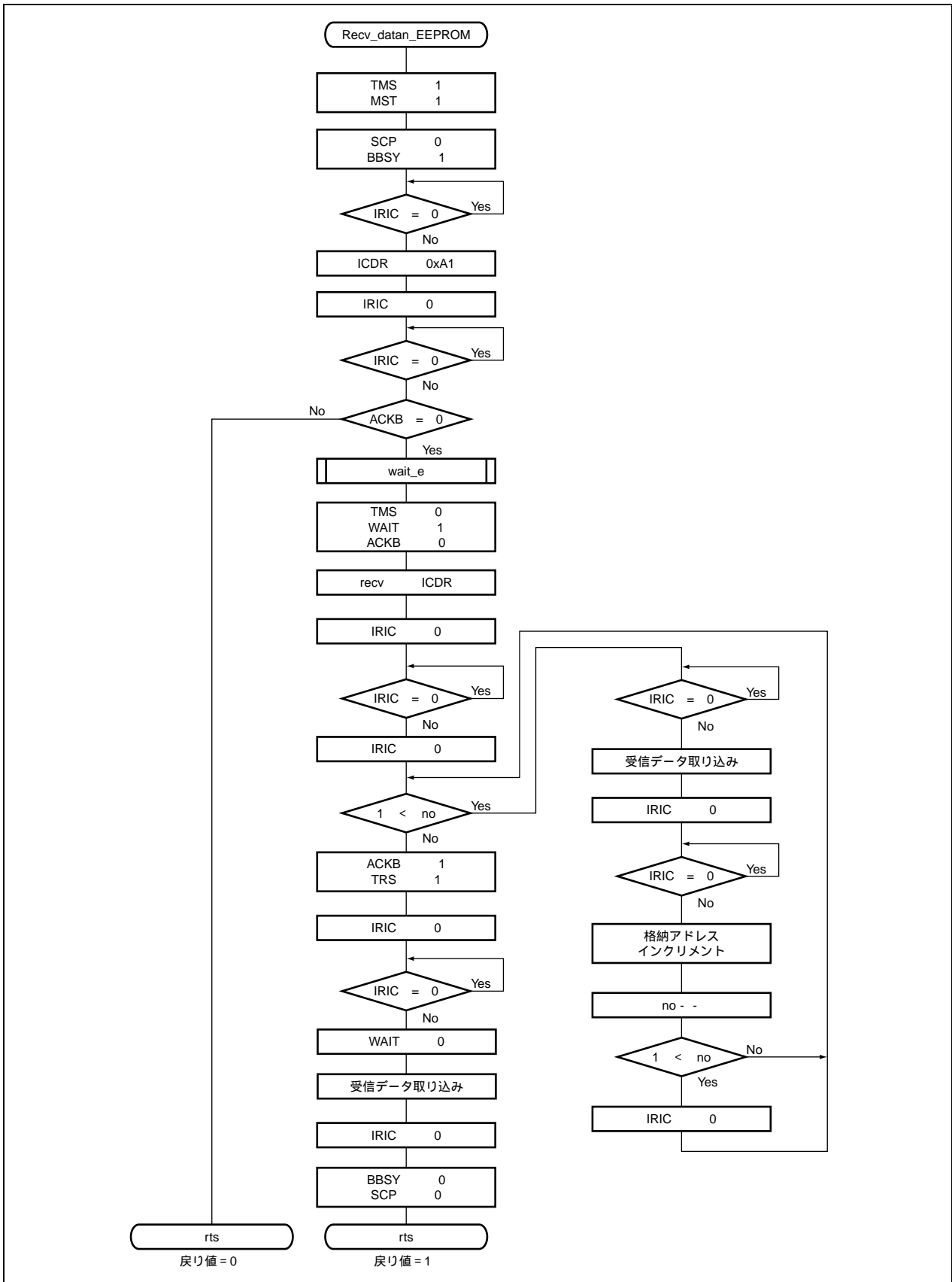












6. ヘッダーリスト

```

File : H8_3664_C.h
/*****/
/*      H8/3664F Include File                Ver 0.1                */
/*****/

struct st_flash {                                /* struct FLASH                */
union {                                          /* FLMCR1                      */
unsigned char BYTE;                            /* Byte Access                 */
struct {                                        /* Bit Access                  */
unsigned char wk :1;                          /*                             */
unsigned char SWE:1;                          /* SWE                         */
unsigned char ESU:1;                          /* ESU                         */
unsigned char PSU:1;                          /* PSU                         */
unsigned char EV :1;                          /* EV                          */
unsigned char PV :1;                          /* PV                          */
unsigned char E  :1;                          /* E                           */
unsigned char P  :1;                          /* P                           */
} BIT;                                         /*                             */
} FLMCR1;                                       /*                             */
union {                                          /* FLMCR2                      */
unsigned char BYTE;                            /* Byte Access                 */
struct {                                        /* Bit Access                  */
unsigned char FLER:1;                          /* FLER                        */
} BIT;                                         /*                             */
} FLMCR2;                                       /*                             */
union {                                          /* FLPWCR                      */
unsigned char BYTE;                            /* Byte Access                 */
struct {                                        /* Bit Access                  */
unsigned char PDWND:1;                        /* PDWND                       */
} BIT;                                         /*                             */
} FLPWCR;                                       /*                             */
union {                                          /* EBR1                        */
unsigned char BYTE;                            /* Byte Access                 */
struct {                                        /* Bit Access                  */
unsigned char wk :3;                          /*                             */
unsigned char EB4:1;                          /* EB4                         */
unsigned char EB3:1;                          /* EB3                         */
unsigned char EB2:1;                          /* EB2                         */
unsigned char EB1:1;                          /* EB1                         */
unsigned char EB0:1;                          /* EB0                         */
} BIT;                                         /*                             */
} EBR1;                                         /*                             */
char wk[7];                                    /*                             */
}

```

```

union {
    unsigned char BYTE;
    struct {
        unsigned char FLSHE:1;
    } BIT;
    } FENR;
};

struct st_ta {
    union {
        unsigned char BYTE;
        struct {
            unsigned char CKSO:3;
            unsigned char    :1;
            unsigned char CKSI:4;
        } BIT;
    } TMA;
    unsigned char TCA;
};

struct st_tv {
    union {
        unsigned char BYTE;
        struct {
            unsigned char CMIEB:1;
            unsigned char CMIEA:1;
            unsigned char OVIE :1;
            unsigned char CCLR :2;
            unsigned char CKS  :3;
        } BIT;
    } TCRV0;
    union {
        unsigned char BYTE;
        struct {
            unsigned char CMFB:1;
            unsigned char CMFA:1;
            unsigned char OVF :1;
            unsigned char    :1;
            unsigned char OS  :4;
        } BIT;
    } TCSR;
    unsigned char TCORA;
    unsigned char TCORB;
    unsigned char TCNTV;
};

```

```

union {
    unsigned char BYTE;
    struct {
        unsigned char wk :3;
        unsigned char TVEG:2;
        unsigned char TRGE:1;
        unsigned char :1;
        unsigned char ICKS:1;
    } BIT;
    } TCRV1;
};

struct st_tw {
    union {
        unsigned char BYTE;
        struct {
            unsigned char CTS :1;
            unsigned char :1;
            unsigned char BUFEA:1;
            unsigned char BUFEB:1;
            unsigned char :1;
            unsigned char PWMD :1;
            unsigned char PWMC :1;
            unsigned char PWMB :1;
        } BIT;
        } TMRW;
    union {
        unsigned char BYTE;
        struct {
            unsigned char CCLR:1;
            unsigned char CKS :3;
            unsigned char TOD :1;
            unsigned char TOC :1;
            unsigned char TOB :1;
            unsigned char TOA :1;
        } BIT;
        } TCRW;
    union {
        unsigned char BYTE;
        struct {
            unsigned char OVIE :1;
            unsigned char :3;
            unsigned char IMIED:1;
        } BIT;
        } TIERW;
};

```

```

unsigned char IMIEC:1;          /* IMIEC          */
unsigned char IMIEB:1;          /* IMIEB          */
unsigned char IMIEA:1;          /* IMIEA          */
} BIT;                          /*                */
} TIERW;                          /*                */
union {                          /* TSRW          */
unsigned char BYTE;            /* Byte Access    */
struct {                       /* Bit Access     */
unsigned char OVF :1;          /* OVF            */
unsigned char   :3;          /*                */
unsigned char IMFD:1;          /* IMFD           */
unsigned char IMFC:1;          /* IMFC           */
unsigned char IMFB:1;          /* IMFB           */
unsigned char IMFA:1;          /* IMFA           */
} BIT;                          /*                */
} TSRW;                          /*                */

union {                          /* TIOR0         */
unsigned char BYTE;            /* Byte Access    */
struct {                       /* Bit Access     */
unsigned char wk :1;          /*                */
unsigned char IOB:3;          /* IOB            */
unsigned char   :1;          /*                */
unsigned char IOA:3;          /* IOA            */
} BIT;                          /*                */
} TIOR0;                          /*                */
union {                          /* TIOR1         */
unsigned char BYTE;            /* Byte Access    */
struct {                       /* Bit Access     */
unsigned char wk :1;          /*                */
unsigned char IOD:3;          /* IOD            */
unsigned char   :1;          /*                */
unsigned char IOC:3;          /* IOC            */
} BIT;                          /*                */
} TIOR1;                          /*                */
unsigned int TCNT;              /* TCNT          */
unsigned int GRA;               /* GRA           */
unsigned int GRB;               /* GRB           */
unsigned int GRC;               /* GRC           */
unsigned int GRD;               /* GRD           */
};                                /*                */
struct st_sci3 {                /* struct SCI3   */
union {                          /* SMR          */

```

```

unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char COM :1; /* COM */
unsigned char CHR :1; /* CHR */
unsigned char PE :1; /* PE */
unsigned char PM :1; /* PM */
unsigned char STOP:1; /* STOP */
unsigned char MP :1; /* MP */
unsigned char CKS :2; /* CKS */
} BIT; /* */
} SMR; /* */
unsigned char BRR; /* BRR */
union { /* SCR3 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char TIE :1; /* TIE */
unsigned char RIE :1; /* RIE */
unsigned char TE :1; /* TE */
unsigned char RE :1; /* RE */
unsigned char MPIO:1; /* MPIO */
unsigned char TEIE:1; /* TEIE */
unsigned char CKE :2; /* CKE */
} BIT; /* */
} SCR3; /* */
unsigned char TDR; /* TDR */

union { /* SSR */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char TDRE:1; /* TDRE */
unsigned char RDRF:1; /* RDRF */
unsigned char OER :1; /* OER */
unsigned char FER :1; /* FER */
unsigned char PER :1; /* PER */
unsigned char TEND:1; /* TEND */
unsigned char MPBR:1; /* MPBR */
unsigned char MPBT:1; /* MPBT */
} BIT; /* */
} SSR; /* */
unsigned char RDR; /* RDR */
}; /* */
struct st_ad { /* struct A/D */

```

```

unsigned int DRA; /* ADDRA */
unsigned int DRB; /* ADDRb */
unsigned int DRC; /* ADDRc */
unsigned int DRD; /* ADDRd */
union {
unsigned char BYTE; /* Byte Access */
struct {
unsigned char ADF :1; /* ADF */
unsigned char ADIE:1; /* ADIE */
unsigned char ADST:1; /* ADST */
unsigned char SCAN:1; /* SCAN */
unsigned char CKS :1; /* CKS */
unsigned char CH :3; /* CH */
} BIT; /*
} CSR; /*
union {
unsigned char BYTE; /* Byte Access */
struct {
unsigned char TRGE:1; /* TRGE */
} BIT; /*
} CR; /*
}; /*

struct st_wdt { /* struct WDT */
union {
unsigned char BYTE; /* Byte Access */
struct {
unsigned char B6WI :1; /* B6WI */
unsigned char TCWE :1; /* TCWE */
unsigned char B4WI :1; /* B4WI */
unsigned char TCSRWE:1; /* TCSRWE */
unsigned char B2WI :1; /* B2WI */
unsigned char WDON :1; /* WDON */
unsigned char B0WI :1; /* B0WI */
unsigned char WRST :1; /* WRST */
} BIT; /*
} TCSRWD; /*

unsigned char TCWD; /* TCWD */
union {
unsigned char BYTE; /* Byte Access */
struct {
unsigned char wk :4; /*
unsigned char CKS:4; /* CKS */

```

```

} BIT; /* */
} TMWD; /* */
}; /* */
struct st_iic { /* struct IIC */
union { /* ICCR */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char ICE :1; /* ICE */
unsigned char IEIC:1; /* IEIC */
unsigned char MST :1; /* MST */
unsigned char TRS :1; /* TRS */
unsigned char ACKE:1; /* ACKE */
unsigned char BBSY:1; /* BBSY */
unsigned char IRIC:1; /* IRIC */
unsigned char SCP :1; /* SCP */
} BIT; /* */
} ICCR; /* */
union { /* ICSR */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char ESTP:1; /* ESTP */
unsigned char STOP:1; /* STOP */
unsigned char IRTR:1; /* IRTR */
unsigned char AASX:1; /* AASX */
unsigned char AL :1; /* AL */
unsigned char AAS :1; /* AAS */
unsigned char ADZ :1; /* ADZ */
unsigned char ACKB:1; /* ACKB */
} BIT; /* */
} ICSR; /* */
union { /* */
struct { /* */
union { /* SARX */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char SVAX:7; /* SVAX */
unsigned char FSX :1; /* FSX */
} BIT; /* */
} UN_SARX; /* */
} SAR; /* SAR */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char SVA:7; /* SVA */

```

```

unsigned char FS :1;          /* FS          */
} BIT;                       /*             */
} UN_SAR;                    /*             */
} ICE0;                      /*             */
struct {                     /*             */
unsigned char UN_ICDR;      /* ICDR        */
union {                    /* ICMR        */
unsigned char BYTE;       /* Byte Access */
struct {                 /* Bit Access  */
unsigned char MLS :1;    /* MLS         */
unsigned char WAIT:1;   /* WAIT        */
unsigned char CKS :3;   /* CKS         */
unsigned char BC :3;   /* BC          */
} BIT;                /*             */
} UN_ICMR;             /*             */
} ICE1;                /*             */
} EQU;                 /*             */
};                    /*             */
struct st_abrk {        /* struct ABRK */
union {                /* ABRKCR      */
unsigned char BYTE;   /* Byte Access */
struct {             /* Bit Access  */
unsigned char RTINTE:1; /* RTINTE     */
unsigned char CSEL :2; /* CSEL       */
unsigned char ACMP :3; /* ACMP       */
unsigned char DCMP :2; /* DCMP       */
} BIT;            /*             */
} CR;              /*             */
union {            /* ABRKSR      */
unsigned char BYTE; /* Byte Access */
struct {         /* Bit Access  */
unsigned char ABIF:1; /* ABIF       */
unsigned char ABIE:1; /* ABIE       */
} BIT;          /*             */
} SR;          /*             */
void *BAR;     /* BAR         */
unsigned int BDR; /* BDR        */
};            /*             */
struct st_io { /* struct IO   */
union {      /* PUCR1      */
unsigned char BYTE; /* Byte Access */
struct {         /* Bit Access  */
unsigned char B7:1; /* Bit 7      */

```



```

} BIT; /* */
} PDR2; /* */
char wk2[2]; /* */
union { /* PDR5 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char B7:1; /* Bit 7 */
unsigned char B6:1; /* Bit 6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */
unsigned char B3:1; /* Bit 3 */
unsigned char B2:1; /* Bit 2 */
unsigned char B1:1; /* Bit 1 */
unsigned char B0:1; /* Bit 0 */
} BIT; /* */
} PDR5; /* */
char wk3; /* */

union { /* PDR7 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk:1; /* Bit 7 */
unsigned char B6:1; /* Bit 6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */
} BIT; /* */
} PDR7; /* */
union { /* PDR8 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char B7:1; /* Bit 7 */
unsigned char B6:1; /* Bit 6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */
unsigned char B3:1; /* Bit 3 */
} BIT; /* */
} PUCR1; /* */
union { /* PUCR5 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk:2; /* Bit 7,6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */

```

```

unsigned char B3:1;          /* Bit 3          */
unsigned char B2:1;          /* Bit 2          */
unsigned char B1:1;          /* Bit 1          */
unsigned char B0:1;          /* Bit 0          */
} BIT;                       /*                */
} PUCR5;                     /*                */
char wk1[2];                 /*                */
union {                       /* PDR1          */
unsigned char BYTE;         /* Byte Access   */
struct {                    /* Bit Access    */
unsigned char B7:1;         /* Bit 7         */
unsigned char B6:1;         /* Bit 6         */
unsigned char B5:1;         /* Bit 5         */
unsigned char B4:1;         /* Bit 4         */
unsigned char B3:1;         /* Bit 3         */
unsigned char B2:1;         /* Bit 2         */
unsigned char B1:1;         /* Bit 1         */
unsigned char B0:1;         /* Bit 0         */
} BIT;                       /*                */
} PDR1;                       /*                */
union {                       /* PDR2          */
unsigned char BYTE;         /* Byte Access   */
struct {                    /* Bit Access    */
unsigned char wk:5;         /* Bit 7-3      */
unsigned char B2:1;         /* Bit 2         */
unsigned char B1:1;         /* Bit 1         */
unsigned char B0:1;         /* Bit 0         */
} BIT;                       /*                */
} PDR2;                       /*                */

char wk2[2];                 /*                */
union {                       /* PDR5          */
unsigned char BYTE;         /* Byte Access   */
struct {                    /* Bit Access    */
unsigned char B7:1;         /* Bit 7         */
unsigned char B6:1;         /* Bit 6         */
unsigned char B5:1;         /* Bit 5         */
unsigned char B4:1;         /* Bit 4         */
unsigned char B3:1;         /* Bit 3         */
unsigned char B2:1;         /* Bit 2         */
unsigned char B1:1;         /* Bit 1         */
unsigned char B0:1;         /* Bit 0         */
} BIT;                       /*                */
}

```

```

} PDR5; /* */
char wk3; /* */
union { /* PDR7 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk:1; /* Bit 7 */
unsigned char B6:1; /* Bit 6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */
} BIT; /* */
} PDR7; /* */
union { /* PDR8 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char B7:1; /* Bit 7 */
unsigned char B6:1; /* Bit 6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */
unsigned char B3:1; /* Bit 3 */
unsigned char B2:1; /* Bit 2 */
unsigned char B1:1; /* Bit 1 */
unsigned char B0:1; /* Bit 0 */
} BIT; /* */
} PDR8; /* */
char wk4; /* */
union { /* PDRB */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char B7:1; /* Bit 7 */
unsigned char B6:1; /* Bit 6 */
unsigned char B5:1; /* Bit 5 */
unsigned char B4:1; /* Bit 4 */
unsigned char B3:1; /* Bit 3 */
unsigned char B2:1; /* Bit 2 */
unsigned char B1:1; /* Bit 1 */
unsigned char B0:1; /* Bit 0 */
} BIT; /* */
} PDRB; /* */
char wk5[2]; /* */
union { /* PMR1 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char IRQ3:1; /* IRQ3 */

```

```

unsigned char IRQ2:1;          /* IRQ2          */
unsigned char IRQ1:1;          /* IRQ1          */
unsigned char IRQ0:1;          /* IRQ0          */
unsigned char   :2;            /*                */
unsigned char TXD :1;          /* TXD           */
unsigned char TMOW:1;          /* TMOW          */
} BIT;                          /*                */
} PMR1;                          /*                */
union {                          /* PMR5          */
unsigned char BYTE;            /* Byte Access   */
struct {                          /* Bit Access    */
unsigned char wk :2;            /*                */
unsigned char WKP5:1;          /* WKP5          */
unsigned char WKP4:1;          /* WKP4          */
unsigned char WKP3:1;          /* WKP3          */
unsigned char WKP2:1;          /* WKP2          */
unsigned char WKP1:1;          /* WKP1          */
unsigned char WKP0:1;          /* WKP0          */
} BIT;                          /*                */
} PMR5;                          /*                */
char wk6[2];                    /*                */
unsigned char PCR1;             /* PCR1          */
unsigned char PCR2;             /* PCR2          */
char wk7[2];                    /*                */
unsigned char PCR5;             /* PCR5          */
char wk8;                       /*                */
unsigned char PCR7;             /* PCR7          */
unsigned char PCR8;             /* PCR8          */
};                                /*                */
union un_syscr1 {                /* union SYSCR1  */
unsigned char BYTE;            /* Byte Access   */
struct {                          /* Bit Access    */
unsigned char SSBY :1;          /* SSBY          */
unsigned char STS :3;          /* STS           */
unsigned char NESEL:1;         /* NESEL         */
} BIT;                          /*                */
};                                /*                */
union un_syscr2 {                /* union SYSCR2  */
unsigned char BYTE;            /* Byte Access   */
struct {                          /* Bit Access    */
unsigned char SMSEL:1;         /* SMSEL         */
unsigned char LSON :1;         /* LSON          */
unsigned char DTON :1;         /* DTON          */

```

```

unsigned char MA :3; /* MA */
unsigned char SA :2; /* SA */
} BIT; /* */
}; /* */
*/

union un_iegr1 { /* union IEGR1 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char NMIEG:1; /* NMIEG */
unsigned char :3; /* */
unsigned char IEG3 :1; /* IEG3 */
unsigned char IEG2 :1; /* IEG2 */
unsigned char IEG1 :1; /* IEG1 */
unsigned char IEG0 :1; /* IEG0 */
} BIT; /* */
}; /* */

union un_iegr2 { /* union IEGR2 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk :2; /* */
unsigned char WPEG5:1; /* WPEG5 */
unsigned char WPEG4:1; /* WPEG4 */
unsigned char WPEG3:1; /* WPEG3 */
unsigned char WPEG2:1; /* WPEG2 */
unsigned char WPEG1:1; /* WPEG1 */
unsigned char WPEG0:1; /* WPEG0 */
} BIT; /* */
}; /* */
*/

union un_ienr1 { /* union IENR1 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char IENDT:1; /* IENDT */
unsigned char IENTA:1; /* IENTA */
unsigned char IENWP:1; /* IENWP */
unsigned char :1; /* */
unsigned char IEN3 :1; /* IEN3 */
unsigned char IEN2 :1; /* IEN2 */
unsigned char IEN1 :1; /* IEN1 */
unsigned char IEN0 :1; /* IEN0 */
} BIT; /* */
}; /* */

union un_irr1 { /* union IRR1 */

```

```

unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char IRRDT:1; /* IRRDT */
unsigned char IRRTA:1; /* IRRTA */
unsigned char :2; /* */
unsigned char IRRI3:1; /* IRRI3 */
unsigned char IRRI2:1; /* IRRI2 */
unsigned char IRRI1:1; /* IRRI1 */
unsigned char IRRIO:1; /* IRRIO */
} BIT; /* */
}; /* */

union un_iwpr { /* union IWPR */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk :2; /* */
unsigned char IWPF5:1; /* IWPF5 */
unsigned char IWPF4:1; /* IWPF4 */
unsigned char IWPF3:1; /* IWPF3 */
unsigned char IWPF2:1; /* IWPF2 */
unsigned char IWPF1:1; /* IWPF1 */
unsigned char IWPF0:1; /* IWPF0 */
} BIT; /* */
}; /* */

union un_mstcr1 { /* union MSTCR1 */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk :1; /* */
unsigned char MSTIIC:1; /* MSTIIC */
unsigned char MSTS3 :1; /* MSTS3 */
unsigned char MSTAD :1; /* MSTAD */
unsigned char MSTWD :1; /* MSTWD */
unsigned char MSTTW :1; /* MSTTW */
unsigned char MSTTV :1; /* MSTTV */
unsigned char MSTTA :1; /* MSTTA */
} BIT; /* */
}; /* */

union un_tscr { /* union TSCR */
unsigned char BYTE; /* Byte Access */
struct { /* Bit Access */
unsigned char wk :6; /* */
unsigned char IICRST:1; /* IICRST */
unsigned char IICX :1; /* IICX */

```

```

} BIT; /* */
}; /* */
#define FLASH (*(volatile struct st_flash *)0xFF90) /* FLASH Address */
#define TA (*(volatile struct st_ta *)0xFFA6) /* TA Address */
#define TV (*(volatile struct st_tv *)0xFFA0) /* TV Address */
#define TW (*(volatile struct st_tw *)0xFF80) /* TW Address */
#define SCI3 (*(volatile struct st_sci3 *)0xFFA8) /* SCI3 Address */
#define AD (*(volatile struct st_ad *)0xFFB0) /* A/D Address */
#define WDT (*(volatile struct st_wdt *)0xFFC0) /* WDT Address */
#define IIC (*(volatile struct st_iic *)0xFFC4) /* IIC Address */
#define ICDR EQU.ICE1.UN_ICDR /* ICDR Change */
#define ICMR EQU.ICE1.UN_ICMR /* ICDR Change */
#define SAR EQU.ICE0.UN_SAR /* SAR Change */
#define SARX EQU.ICE0.UN_SARX /* SARX Change */
#define ABRK (*(volatile struct st_abrk *)0xFFC8) /* ABRK Address */
#define IO (*(volatile struct st_io *)0xFFD0) /* IO Address */
#define SYSCR1 (*(volatile union un_syscr1*)0xFFF0) /* SYSCR1Address */
#define SYSCR2 (*(volatile union un_syscr2*)0xFFF1) /* SYSCR2Address */

#define IEGR1 (*(volatile union un_iegr1 *)0xFFF2) /* IEGR1 Address */
#define IEGR2 (*(volatile union un_iegr2 *)0xFFF3) /* IEGR2 Address */
#define IENR1 (*(volatile union un_ienr1 *)0xFFF4) /* IENR1 Address */
#define IRR1 (*(volatile union un_irr1 *)0xFFF6) /* IRR1 Address */
#define IWPR (*(volatile union un_iwpr *)0xFFF8) /* IWPR Address */
#define MSTCR1 (*(volatile union un_mstcr1*)0xFFF9) /* MSTCR1Address */
#define TSCR (*(volatile union un_tscr *)0xFFFC) /* TSCR Address */

#define EKR (*(volatile unsigned char *)0xFF10) /* EKR Address */

```

7. プログラムリスト

```
File : INIT.SRC
;
; /*      Assembler routine                                     */
;
.EXPORT      _INIT
.IMPORT      _main
.SECTION     P, CODE
_INIT:
MOV.W        #H'FF80, R7
LDC.B        #B'10000000, CCR
JMP          @_main
;
.END

File : EPR.c
/*****
/*
/*      FILE              :EPR.c                               */
/*      DATE              :Jun 29, 2001                       */
/*      DESCRIPTION       :Main Program                       */
/*      CPU TYPE         :H8/3664F                            */
/*
/*      This file is generated by Hitachi Project Generator (Ver.1.0)
/*
*****/
#include <machine.h>
#include "H8_3664_C.H"

extern      void INIT( void );
extern      unsigned char Read_page_EEPROM( unsigned short adrs , unsigned char *rd_ptr );
extern      unsigned char Write_page_EEPROM( unsigned short adrs , unsigned char *wr_ptr );
extern      unsigned char Write_byte_EEPROM( unsigned short adrs , unsigned char wr_data );
extern      unsigned char Read_byte_EEPROM( unsigned short adrs );
extern      unsigned char Read_n_EEPROM( unsigned short adrs, unsigned char *rd_ptr, unsigned short no );

void        main ( void );
void        wait_ack( unsigned short limit ) ;
void        wait ( unsigned short limit );
void        Test_EEPROM( void );
```

```
/*;*****  
/*; Vector Address */  
/*;*****  
#pragma section          V1                      /* VECTOR SECTOIN SET */  
void (*const VEC_TBL1[])(void) = {  
/* 0x00 - 0x0f */  
INIT,                      /* 00 RESET */  
};  
  
#ifdef __cplusplus  
extern "C" {  
#endif  
void abort(void);  
#ifdef __cplusplus  
}  
#endif  
  
#pragma section                      /* section:B */  
unsigned char eeprom_buf[512];  
unsigned char test_code;  
unsigned char dummy;  
  
#pragma section                      /* section:P */  
/*;*****  
/*; Main Program */  
/*;*****  
void main(void)  
{  
test_code = 1;  
  
while (1) {  
  
Test_EEPROM( );  
}  
}
```

```

/*;*****/
/*; IIC EEPROM                                     */
/*          test_code          test job          */
/*          0                   nop              */
/*          1                   byte write/read  */
/*          2                   page write 0-ff   */
/*          3                   page read        */
/*          4                   page write all"ff"*/
/*          5                   all read         */
/*;*****/

void Test_EEPROM( void ) {
unsigned char ng_flag,wr_data;
unsigned short          i;

if (test_code == 1) {
i = 0;
while(i < 512)          {
wr_data = (unsigned char)(i & 0xff);
ng_flag = Write_byte_EEPROM(i , wr_data );
eeprom_buf[i] = Read_byte_EEPROM(i);
i++;
}
test_code = 0;
}

else if (test_code == 2) {
for (i = 0;i < 512;i++) {
eeprom_buf[i] = (unsigned char)((i>>1) & 0x00ff);          /* buf set          */
}
for (i = 0;i < 512;i+=8) {
ng_flag = Write_page_EEPROM( i , &eeprom_buf[i] );          /* write:0-511          */
}
test_code = 0;
}

else if (test_code == 3) {
for (i = 0;i < 512;i+=8) {
ng_flag = Read_page_EEPROM( i , &eeprom_buf[i] );          /* read:0-511          */
wait(100);
}
test_code = 0;
}

else if (test_code == 4) {
for (i = 0;i < 8;i++) {

```

```
    eeprom_buf[i] = 0xff;                /* buf:FF set          */
    }
    for (i = 0;i < 512;i+=8) {
    ng_flag = Write_page_EEPROM( i , &eeprom_buf[0] ); /* write:0-511      */
    }
    test_code = 0;
    }
    else if (test_code == 5)            {
    for (i = 0;i < 512;i++) {          /* buf clear        */
    eeprom_buf[i] = 0x00;
    }
    ng_flag = Read_n_EEPROM( 0x0000 , &eeprom_buf[0] ,512); /* read:0-511      */
    test_code = 0;
    }
    }

    void wait( unsigned short limit ) {
    unsigned int cnt;

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

    void abort(void)
    {

    }
```

```

File : IIC_EEPROM.c

/*-----*/
/*          IIC EEPROM Read/Write          */
/*          Sub routine for H8/3664N built in EEPROM          */
/*-----*/

#include <machine.h>
#include "H8_3664_C.H"

#define DEVICE_CODE          0xa0          /* EEPROM DEVICE CODE:101          */
#define SLAVE_ADRS          0x00          /* SLAVE ADRS:0          */
#define IIC_DATA_W          0x00          /* WRITE_DATA          */
#define IIC_DATA_R          0x01          /* READ_DAT          */

#define WR_RETRY_CNT          10          /* tWC(1ms)*10=10ms          */
#define WR_OK 11          /* loop break          */

unsigned char Write_data_EEPROM( unsigned char wr_data );
unsigned char Set_adrs_EEPROM( unsigned short adrs );
unsigned char Recv_data1_EEPROM( void );
unsigned char Recv_data_n_EEPROM( unsigned char *rd_ptr , unsigned short no );
void          wait_e( unsigned int loop);

extern          unsigned char dummy;

/*-----*/
/*          Read_page_EEPROM          (8byte)          */
/*          Argument1: Read address(unsigned short)          */
/*          Argument2: Storing read data address (unsigned char *)          */
/*          Return value: 1: OK / 0: NG EEPROM NOACK(unsigned char)          */
/*-----*/
unsigned char Read_page_EEPROM( unsigned short adrs , unsigned char *rd_ptr ) {
    unsigned char i,ack;

    IIC.ICCR.BYTE = 0x89;          /* ICE=1 P57,P56->SCL,SDA          */
    IIC.ICMR.BYTE = 0x08;
    TSCR.BYTE = 0xfc;

    ack = 0;
    while (IIC.ICCR.BIT.BBSY != 0)          /* Bus busy?          */
        ;

    ack = Set_adrs_EEPROM(adrs);          /* Addressing(dummy write)          */
}

```

```

if (ack == 0) {
IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa;          /* Generate stop condition */
return(0);
}

ack = Recv_datan_EEPROM(rd_ptr,8);             /* Data reading of 8 bytes */

return(ack);
}

/*-----*/
/*      Read_page_EEPROM      (8byte)          */
/*      Argument1: Read address (unsigned short) */
/*      Argument2: Storing read data address (unsigned char *) */
/*      Argument3: Number of read data (unsigned short) */
/*      Return value: 1: OK / 0: NG EEPROM NOACK(unsigned char) */
/*-----*/
unsigned char Read_n_EEPROM( unsigned short adrs , unsigned char *rd_ptr , unsigned short no ) {
unsigned char i,ack;

IIC.ICCR.BYTE = 0x89;          /* ICE=1 P57,P56->SCL,SDA */
IIC.ICMR.BYTE = 0x08;
TSCR.BYTE = 0xfc;

ack = 0;
while (IIC.ICCR.BIT.BBSY != 0) /* Bus busy? */
;

ack = Set_adrs_EEPROM(adrs);   /* Addressing (dummy write) */
if (ack == 0) {
IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa;          /* Generate stop condition */
return(0);
}

ack = Recv_datan_EEPROM(rd_ptr,no);           /* Data reading of n bytes */

return(ack);
}

```

```

/*-----*/
/*      Write_page_EEPROM    (8byte)                                */
/*      Argument1: Write address (unsigned short)                  */
/*      Argument2: Storing write data address (unsigned char *)    */
/*      Return value: 1: OK / 0: NG EEPROM NOACK(unsigned char)    */
/*-----*/
unsigned char Write_page_EEPROM( unsigned short adrs , unsigned char *wr_ptr ) {
unsigned char loop,i,ack;

ack = 0;
loop = 0;
while (loop <= WR_RETRY_CNT) {
IIC.ICCR.BYTE = 0x89; /* ICE=1(P57,P56->SCL,SDA) */
IIC.ICMR.BYTE = 0x08;
TSCR.BYTE = 0xfc;

while (IIC.ICCR.BIT.BBSY != 0) /* Bus busy? */
;

ack = Set_adrs_EEPROM(adrs); /* Addressing */
if ( ack == 1 ) {
loop = WR_OK;
}
else {
IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa; /* Generation of stop condition
*/
wait_e(1000); /* tWC=1ms */
loop++;
}
}

if (ack == 1) {
IIC.ICCR.BIT.IRIC = 0;
i = 0;
while ((i < 8) && (ack == 1)) {
ack = Write_data_EEPROM(*wr_ptr); /* Writing of data of 8 bytes */
wr_ptr++;
i++;
}
IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa; /* Generation of stop condition
*/
}
}

```

```

return(ack);
}
/*-----*/
/*      wait_e (Wait sub routine for EEPROM)      */
/*      Argument1: Number of loops (unsigned short) */
/*      Return value: None                        */
/*-----*/
void wait_e( unsigned int loop) {
unsigned int i;

for (i=0;i < loop;i++) {
dummy++;
}
}

/*-----*/
/*      Read_byte_EEPROM                          */
/*      Argument1: Read address (unsigned short)   */
/*      Return value: Read data (unsigned char)   */
/*-----*/
unsigned char Read_byte_EEPROM( unsigned short adrs ) {
unsigned char data,ack;

IIC.ICCR.BYTE = 0x89;          /* ICE=1 P57,P56->SCL,SDA */
IIC.ICMR.BYTE = 0x08;
TSCR.BYTE = 0xfc;

while (IIC.ICCR.BIT.BBSY != 0) /* Bus busy? */
;

ack = Set_adrs_EEPROM(adrs);  /* Addressing (dummy write) */

if (ack == 1) {
data = Recv_data1_EEPROM();
}

IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa; /* Generate stop condition */

return(data);
}

```

```

/*-----*/
/*      Write_byte_EEPROM                                     */
/*      Argument1: Write address (unsigned short)           */
/*      Argument2: Write data (unsigned char)               */
/*      Return value: 1: OK / 0: NG EEPROM NOACK(unsigned char) */
/*-----*/
unsigned char Write_byte_EEPROM( unsigned short adrs , unsigned char wr_data ) {
unsigned char loop,ack;

ack = 0;
loop = 0;
while (loop <= WR_RETRY_CNT) {

IIC.ICCR.BYTE = 0x89;                                     /* ICE=1(P57,P56->SCL,SDA) */
IIC.ICMR.BYTE = 0x08;
TSCR.BYTE = 0xfc;

while (IIC.ICCR.BIT.BBSY != 0)                           /* Bus busy? */
;

ack = Set_adrs_EEPROM(adrs);
if ( ack == 1 ) {
loop = WR_OK;
}

else {
IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa;                   /* Generate stop condition */
wait_e(1000);                                           /* tWC=1ms */
loop++;
}
}

if (ack == 1) {
ack = Write_data_EEPROM(wr_data);
IIC.ICCR.BIT.IRIC = 0;
IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa;                   /* Generate stop condition */
}

return(ack);
}

```

```

/*-----*/
/*      Write_data_EEPROM      */
/*      Argument1: Write address (unsigned short)      */
/*      Return value: 1: OK / 0: NG EEPROM NOACK(unsigned char)      */
/*-----*/
/*-----*/
/* (BYTE WRITE ACTION)      */
/*      <4>      */
/*      123456789      */
/*      000000000      */
/*      Write data A      */
/*-----*/
unsigned char Write_data_EEPROM( unsigned char wr_data ) {

                                                    /* <4>      */
IIC.ICDR = wr_data;                               /* <4> Set writing data      */
IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0)                    /* <4> Transmission complete? */
;
if ( IIC.ICSR.BIT.ACKB != 0 ) {                  /* ACK?      */
return(0);
}

return(1);
}

```

```

/*-----*/
/*      Set_adrs_EEPROM                                     */
/*      Argument1: Write / read address (unsigned short)  */
/*      Return value: 1: OK / 0: NG EEPROM NOACK(unsigned char) */
/*-----*/
/*-----*/
/* ADDRESS SET ACTION / DUMMY WRITE ACTION)                */
/* <1>                <2>                <3>                */
/* 123456789          123456789          123456789          */
/* 101000000          000000000          000000000          */
/* Slave WA  Address HI A      Address LO A                */
/*-----*/
unsigned char Set_adrs_EEPROM( unsigned short adrs ) {
unsigned char ret;

IIC.ICCR.BYTE |= 0x30;                                     /* Master transmission setting (MST=1,TRS=1) */
IIC.ICCR.BYTE = ((IIC.ICCR.BYTE & 0xfe) | 0x04);

/* Generating of start condition */
while (IIC.ICCR.BIT.IRIC == 0)                            /* Transmission OK? */
;

/* <1> */
IIC.ICDR = (unsigned char)(DEVICE_CODE | SLAVE_ADRS | IIC_DATA_W);
/* <1>Slave address set */

IIC.ICCR.BIT.IRIC = 0;

while (IIC.ICCR.BIT.IRIC == 0)                            /* <1> Transmission complete? */
;

if ( IIC.ICSR.BIT.ACKB != 0 ) {                          /* ACK? */
return(0);
}

/* <2> */
IIC.ICDR = (unsigned char)(adrs >> 8);                   /* <2> Upper address set */
IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0)                            /* <2> Transmission complete? */
;
if ( IIC.ICSR.BIT.ACKB != 0 ) {                          /* ACK? */
return(0);
}

/* <3> */
IIC.ICDR = (unsigned char)(adrs & 0x00ff);               /* <3> Lower address set */
IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0)                            /* <3> Transmission complete? */
;

```

```

;

if ( IIC.ICSR.BIT.ACKB != 0 ) {                                /* ACK?                */
return(0);
}

return(1);
}

/*-----*/
/*          Recv_data1_EEPROM                                */
/*          Return value: Read data (unsigned char)         */
/*-----*/
/*-----*/
/* (CURRENT ADDRESS READ ACTION)                            */
/* <1>                <2>                                   */
/* 123456789          123456789                             */
/* 101000010          000000000                             */
/* Slave RA  Read data A                                    */
/*-----*/

unsigned char Recv_data1_EEPROM( void ) {
unsigned char recv;

IIC.ICCR.BYTE |= 0x30;                                        /* Master transmission setting (MST=1,TRS=1) */
IIC.ICCR.BYTE = ((IIC.ICCR.BYTE & 0xfe) | 0x04);          /* Generating of start condition */
while (IIC.ICCR.BIT.IRIC == 0)                             /* Transmission OK? */
;

IIC.ICDR = (unsigned char)(DEVICE_CODE | SLAVE_ADRES | IIC_DATA_R);
/* <1> Slave address set */

IIC.ICCR.BIT.IRIC = 0;

while (IIC.ICCR.BIT.IRIC == 0)                             /* <1> Transmission complete? */
;
if ( IIC.ICSR.BIT.ACKB != 0 ) {                            /* ACK?                */
return(0);
}

wait_e(30);                                                /* dummy wait */
/* Master reception setting */

IIC.ICCR.BIT.TRS = 0;
IIC.ICMR.BIT.WAIT = 1;

```

```

IIC.ICSR.BIT.ACKB = 0;

recv = IIC.ICDR; /* dummy read */
IIC.ICCR.BIT.IRIC = 0;

while (IIC.ICCR.BIT.IRIC == 0) /* Reception complete? */
;

/* < The last reception > */

IIC.ICSR.BIT.ACKB = 1;
IIC.ICCR.BIT.TRIS = 1;
IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0) /* Reception complete? */
;

IIC.ICMR.BIT.WAIT = 0;

recv = IIC.ICDR; /* Received data read */

return(recv);
}
/*-----*/
/*      Recv_datan_EEPROM */
/*      Argument1: Storing read data address (unsigned char *) */
/*      Argument2: Read byte number1-512 (unsigned char) */
/*      Return value: 1: OK / 0: NG EEPROM NOACK (unsigned char) */
/*-----*/
/*-----*/
/* (CURRENT ADDRESS READ ACTION) */
/* <1>          <2>          <3>          <n> */
/* 123456789      123456789      123456789      123456789 */
/* 101000010      000000000      000000000      000000000 */
/* Slave RA  Read data A      Read data A  Read data A */
/*-----*/
unsigned char Recv_datan_EEPROM( unsigned char *rd_ptr , unsigned short no ) {
unsigned char recv;

IIC.ICCR.BYTE |= 0x30; /* Master transmission setting (MST=1,TRS=1) */
IIC.ICCR.BYTE = ((IIC.ICCR.BYTE & 0xfe) | 0x04); /* Generating of start condition */

while (IIC.ICCR.BIT.IRIC == 0) /* Transmission OK? */
;

```

```

IIC.ICDR = (unsigned char)(DEVICE_CODE | SLAVE_ADRS | IIC_DATA_R);
                                                    /* <1> Slave address set          */
IIC.ICCR.BIT.IRIC = 0;

while (IIC.ICCR.BIT.IRIC == 0)                    /* <1> Transmission complete?      */
;

if ( IIC.ICSR.BIT.ACKB != 0 ) {                  /* ACK?                             */
return(0);
}

wait_e(30);                                       /* dummy wait                        */
                                                    /* Master reception setting         */

IIC.ICCR.BIT.TRIS = 0;
IIC.ICMR.BIT.WAIT = 1;
IIC.ICSR.BIT.ACKB = 0;

recv = IIC.ICDR;                                  /* dummy read                         */
IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0)                  /* Reception complete?              */
;

IIC.ICCR.BIT.IRIC = 0;

while (1 < no) {
while (IIC.ICCR.BIT.IRIC == 0)                  /* Reception complete?              */
;

*rd_ptr = IIC.ICDR;                              /* Received data read                */

IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0)                  /* Reception complete?              */
;

rd_ptr++;                                         /* Storage address increment         */
no--;
if (1 < no) {
IIC.ICCR.BIT.IRIC = 0;
}
}

                                                    /* < The last reception >          */

IIC.ICSR.BIT.ACKB = 1;

```

```

IIC.ICCR.BIT.TRIS = 1;

IIC.ICCR.BIT.IRIC = 0;
while (IIC.ICCR.BIT.IRIC == 0)          /* Reception complete?          */
;

IIC.ICMR.BIT.WAIT = 0;

*rd_ptr = IIC.ICDR;                    /* Received data read          */

IIC.ICCR.BIT.IRIC = 0;

IIC.ICCR.BYTE = IIC.ICCR.BYTE & 0xfa;  /* Generation of stop condition */

return(1);
}

/*-----*/
/* IIC Register function          */
/*-----*/
/*-----*/
/* ICSR      7      6      5      4      3      2      1      0          */
/*          ESTP   STOP   IRTR   AASX   AL     AAS    ADZ    ACKB          */
/*-----*/
/*-----*/
/* ICCR      7      6      5      4      3      2      1      0          */
/*          ICE    IEIC   MST    TRS    ACKE   BBSY   IRIC   SCP          */
/* 0x89     1          0      0      0      0      1      0      0          1          */
/*          SCLSDA                                ACK Enable          */
/*-----*/
/*-----*/
/* ICMR      7      6      5      4      3      2      1      0          */
/*          MLS    WAIT   CKS2   CKS1   CKS0   BC2    BC1    BC0          */
/* 0x08     0      0      0      0      1      0      0      0          */
/*          CKS = 400kHz                          format = 9bit          */
/*-----*/
/*-----*/
/* TCSR      7      6      5      4      3      2      1      0          */
/*          1      1      1      1      1      1      IICRST  IICX          */
/* 0xfc     1      1      1      1      1      1      0      0          */
/*-----*/

```

```
/*-----*/  
/*      EEPROM Read/Write End      */  
/*-----*/
```

Link address specification

Section Name	Address
CV1	H'0000
P	H'0100
B	H'FF80