SH7730 グループ
IIC シングルマスタ送受信制御例 (EEPROM ライト・リード)

RJJ06B1057-0100
Rev.1.00
2010.09.06

要旨

本アプリケーションノートは、SH7730 の I²C バスインタフェース (IIC) をシングルマスタで使用し、EEPROM にライト・リードアクセスする例について説明します。

動作確認デバイス

SH7730

目次

1. はじめに 2
2. I²C バスの概要 4
3. EEPROM 使用方法について 10
4. 応用例の説明 16
5. 参考プログラム例 59
6. 実行結果 97
7. 参考ドキュメント 98

1. はじめに

1.1 仕様

- マスタデバイスを SH7730、スレーブデバイスを EEPROM として、シングルマスタで、EEPROM の Memory address 0x0000 番地へ 10 バイト分のデータをライトします。
- マスタデバイスを SH7730、スレーブデバイスを EEPROM として、シングルマスタで、EEPROM の Memory address 0x0000 番地から 10 バイト分のデータをリードします。
- 本応用例では、EEPROM は、ルネサス エレクトロニクス製メモリサイズ 64k ビット品 (R1EX24064ASAS0I) を使用します。
- 転送レートは 333kHz に設定しています。

1.2 使用機能

- I^2C バスインタフェース (IIC) のチャネル 0

1.3 適用条件

- 評価ボード: アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
  外付けメモリ (エリア 0): NOR 型フラッシュメモリ 4M バイト
  Spansion 製 S29AL032D70TFI04
  (エリア 3): SDR-SDRAM 32M バイト (16M バイト ×2 個)
  Samsung 製 K4S281632F-UC75
- マイコン: SH7730 (R8A77301)
- 動作周波数
  CPU クロック: 266.66 MHz
  SuperHyway バスクロック: 133.33 MHz
  バスクロック: 66.66 MHz
  周辺クロック: 33.33 MHz
- エリア 0 バス幅: 16 ビット固定 (MD3 端子 = Low レベル)
- クロック動作モード: モード 2 (MD0 端子 = Low レベル、MD1 端子 = High レベル)
- エンディアン: ビッグエンディアン (MD5 端子 = Low レベル)
- ツールチェーン: ルネサス エレクトロニクス製 SuperH RISC engine Standard Toolchain Ver.9.3.0.0
- コンパイルオプション: High-performance Embedded Workshop でのデフォルト設定
  (-cpu=sh4a -include="$(PROJDIR)¥inc" -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -optimize=0 -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo)

1.4 関連するアプリケーションノート

本資料の参考プログラムは, 「SH7730 グループ アプリケーションノート SH7730 初期設定例 (RJJ06B0864)」の設定条件で動作確認しています。

1.5　本アプリケーションノートで用いる用語の説明

開始条件の発行:
　ICCR2 レジスタの BBSY に 1、SCP に 0 を設定したタイミングです。
開始条件の生成:
　バス上に開始条件が出力されたタイミングです。
停止条件の発行:
　ICCR2 レジスタの BBSY に 0、SCP に 0 を設定したタイミングです。
停止条件の生成:
　バス上に停止条件が出力されたタイミングです。
ACK:
　Acknowledge (アクノリッジ) が"0"の状態を表します。
NACK:
　Acknowledge (アクノリッジ)　が"1"の状態を表します。

2. I²C バスの概要

2.1 I²C バスの特長

2.1.1 I²C バスの特長

I²C バスの特長を以下に示します。詳細な説明については、NXP 社の I²C 関連資料を参照ください。

I²C バスは、シリアルデータライン (SDA)、シリアルクロックライン (SCL) の 2 本のバスラインで構成されます。I²C バス装置の拡張が容易です。装置間にはマスタとスレーブという関係が常に成り立ち、各装置は固有のアドレスを持っています。マスタとなる装置が、最初に通信相手の有する固有アドレスを指定することにより通信のパスが形成され、データ通信が可能となります。I²C バスインタフェースは、データ破壊を防ぐためのバス権競合回避のシステムが定義されています。I²C バスシステムにおける装置の総数は、システムのバス負荷容量の上限値 400pF で決定されます。

2.1.2 I²C バスの接続形式

図 1 に I²C バスインタフェースの接続形式を示します。この図のように I²C バスは、クロックライン SCL とデータライン SDA から構成され、それぞれプルアップ抵抗でバス電源 VBB に接続されます。

デバイス 1 とデバイス 2 の各 SCL 端子/SDA 端子はそれぞれ SCL ラインと SDA ラインにワイヤード AND 接続されます。デバイス 1 が SCL ラインを"Low"にドライブしているとき、デバイス 2 は SCL ラインの状態をモニタすることにより他のデバイスがバスを使用していることを確認します。またワイヤード AND 接続により、デバイス 1 がバスを使用中で SCL ラインをドライブしていても、デバイス 2 が SCL を"Low"にドライブし、デバイス 1 に対して通信動作を"待ち"状態にすることができます。

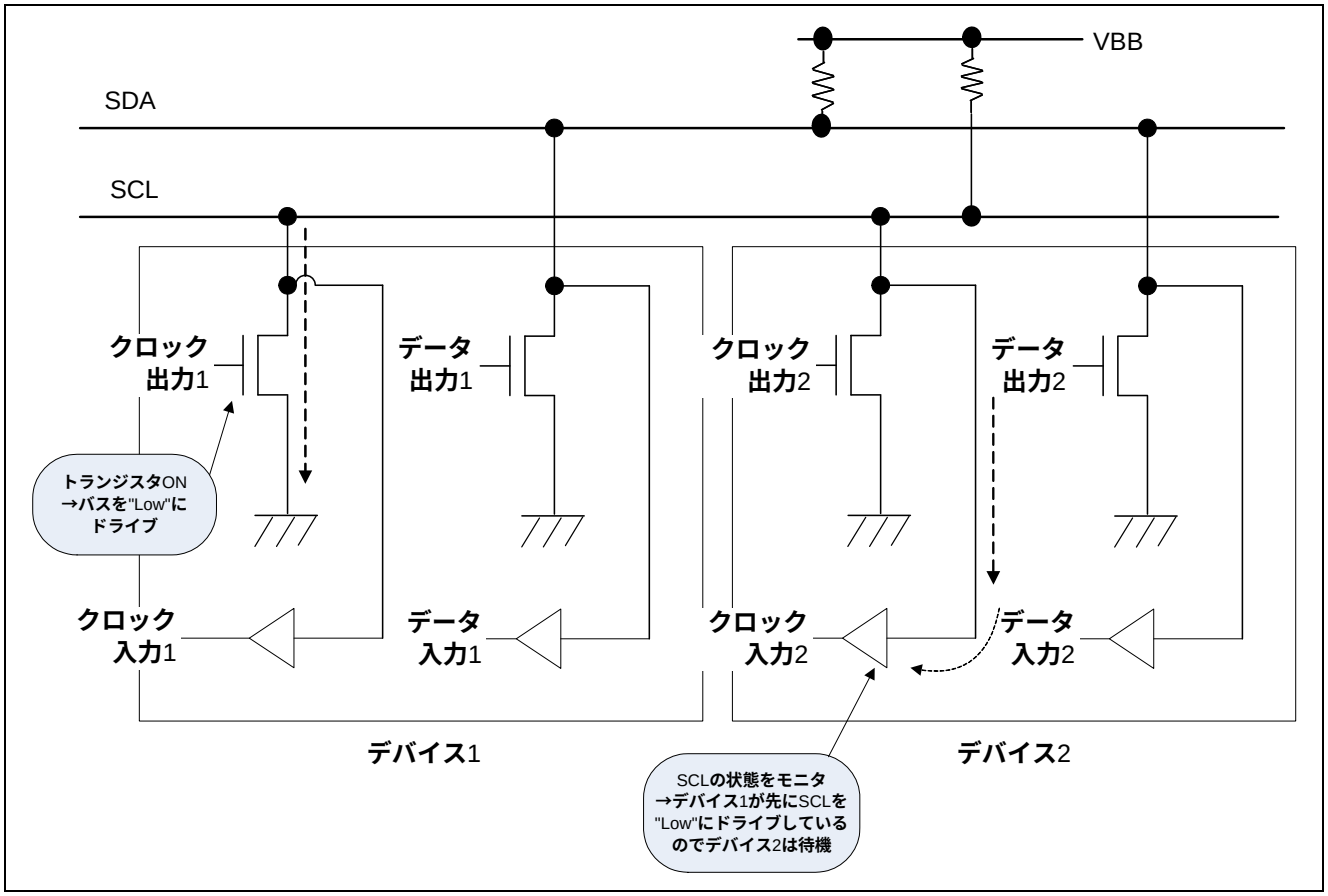


図 1　I²C バスの接続形式 (デバイス 1 が最初に SCL を"Low"にドライブした場合)

2.2 I^2C バスを使用したデータ転送方法

2.2.1 I^2C バスを使用したデータ転送の基本事項

はじめに I^2C バスを使用したデータ転送の基本事項を説明します。

(1) マスタデバイス
マスタデバイスは、データ通信を行うための同期化クロックを生成し、データ通信の開始/停止を示す開始条件/停止条件を発行します。

(2) スレーブデバイス
スレーブデバイスは、マスタデバイス以外の I^2C バスデバイスです。マスタデバイスからアドレス指定されます。

(3) 送信デバイス
送信デバイスとは、データをバスに送信するデバイスです。マスタデバイスとスレーブデバイスの場合があります。

(4) 受信デバイス
受信デバイスとは、データをバスから受信するデバイスです。マスタデバイスとスレーブデバイスの場合があります。

(5) 開始条件と停止条件
開始条件とは、図 2 のように SCL ラインが"High"のときに、SDA ラインが"High"から"Low"に変化する動作です。これによりデータ通信動作が開始されます。
停止条件とは、図 2 のように SCL ラインが"High"のときに、SDA ラインが"Low"から"High"に変化する動作です。これによりデータ通信動作が停止されます。
開始条件と停止条件は、必ずマスタによって生成されます。開始条件が発生した後には、バスがビジー状態になります。停止条件が生成されると、その後しばらく、バスは再びフリー状態になります。

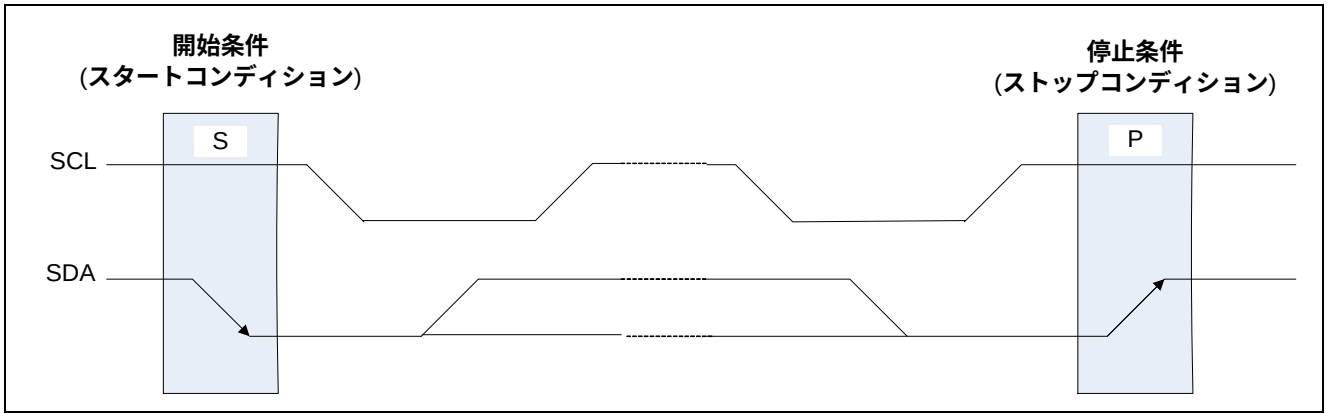


図 2　開始条件と停止条件

(6) データ出力タイミング
図 3 のようにデータ出力タイミングは、SCL ラインが"Low"のとき、SDA ライン上のデータが更新され、SCL ラインが"High"のとき SDA ライン上のデータが確定します。SCL ラインが"High"のとき SDA ラインが変化するのは前記「開始条件」および「停止条件」のときのみです。

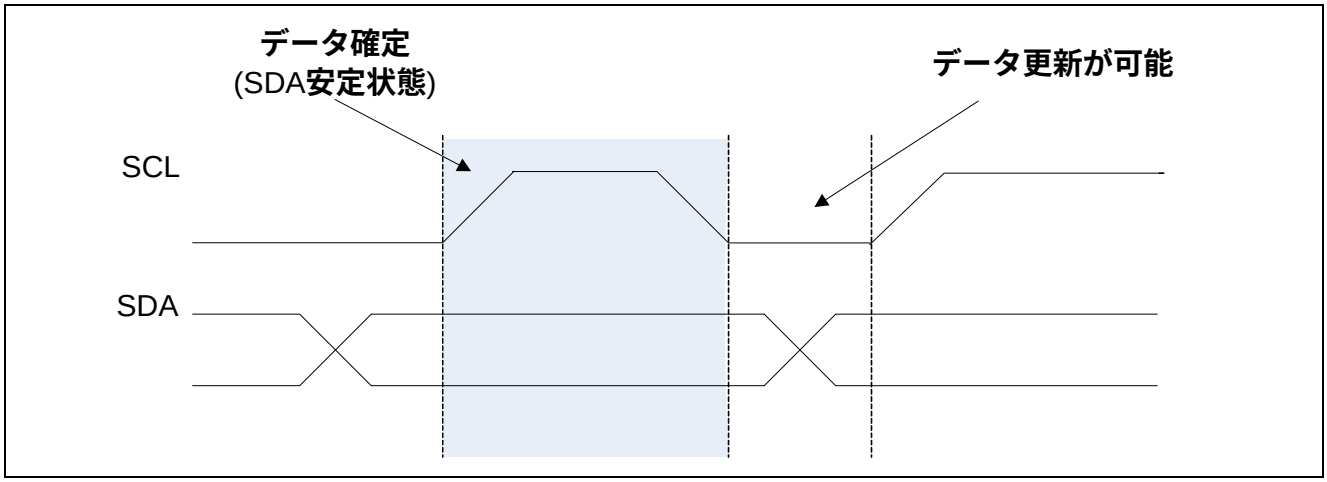


図 3　データ出力タイミング

(7) マスタ送信動作
マスタ送信動作とは、マスタデバイスが送信デバイスの場合の動作です。開始条件発行後のスレーブアドレスの送信やスレーブデバイスへのコマンドなどの送信をする場合があります。

(8) マスタ受信動作
マスタ受信動作とは、マスタデバイスが受信デバイスの場合の動作です。

(9) スレーブ送信動作
スレーブ送信動作とは、スレーブデバイスが送信デバイスの場合の動作です。

(10) スレーブ受信動作
スレーブ受信動作とは、スレーブデバイスが受信デバイスの場合の動作です。開始条件後のマスタデバイスによるスレーブアドレス送信フレームでは、スレーブデバイスは受信動作となります。

(11) バス解放状態
すべての I^2C バスデバイスが通信していない状態です。SCL, SDA ラインとも定常的に"High"状態です。

(12) バス占有状態
バス占有状態とは、I^2C バスデバイスがデータ通信を行っている状態です。マスタデバイスが停止条件を発行した時点でバス開放状態に戻ります。

(13) I^2C バスフォーマット
I^2C バスフォーマットについては「SH7730 グループ ハードウェアマニュアル (RJJ09B0339)」の「I^2C バスインタフェース (IIC)」の章を参照してください。

2.2.2　データ転送手順

【参考】
ここでの説明では、マスタデバイス＝送信デバイス、スレーブデバイス＝受信デバイスとします。

図 4 にマスタデバイスがスレーブデバイスに 1 バイトのデータを送信する場合の例を示します。まず、マスタデバイスは開始条件を発行し、SCL ラインが"High"のときに、SDA ラインを"High"から"Low"に変化させます。次にマスタは、SCL ライン上にクロックを出力するとともに、SDA ライン上に通信対象となるスレーブのアドレスを出力します。スレーブのアドレスは 7 ビットで定義され、8 ビット目に通信方向を表すビットを付加されます。

マスタデバイスは 9 クロック目に SDA ラインを開放し、スレーブデバイスからのアクノリッジに備えます。スレーブデバイスは、9 クロック目に SDA ラインを"Low"にドライブしアクノリッジを返します。マスタデバイスはスレーブアドレスからのアクノリッジを受信し、次の送信データが準備できるまで、SCL ラインを"Low"に保持します。送信データの準備ができたところでマスタデバイスは、SCL ラインにクロックを出力しながら、データを SDA ラインに出力します。前回と同様に 9 クロック目にスレーブデバイスはマスタデバイスにアクノリッジを返し、正常にデータが受信できたことを通知します。マスタデバイスは、スレーブデバイスからのアクノリッジを受け取ると、SCL ラインを"Low"に保持します。そして停止条件を発行し、SCL ラインが"High"のとき、SDA ラインが"Low"から"High"に変化させます。

データ通信中、もし、スレーブデバイスが他の処理を行っているため、すぐにデータを受信できない場合は、スレーブデバイス側で SCL ラインを"Low"に保持し、マスタデバイスを待ち状態にすることができます。スレーブデバイスが SCL を"Low"にドライブできるタイミングは、マスタデバイスが SCL を"Low"にドライブしているときです。

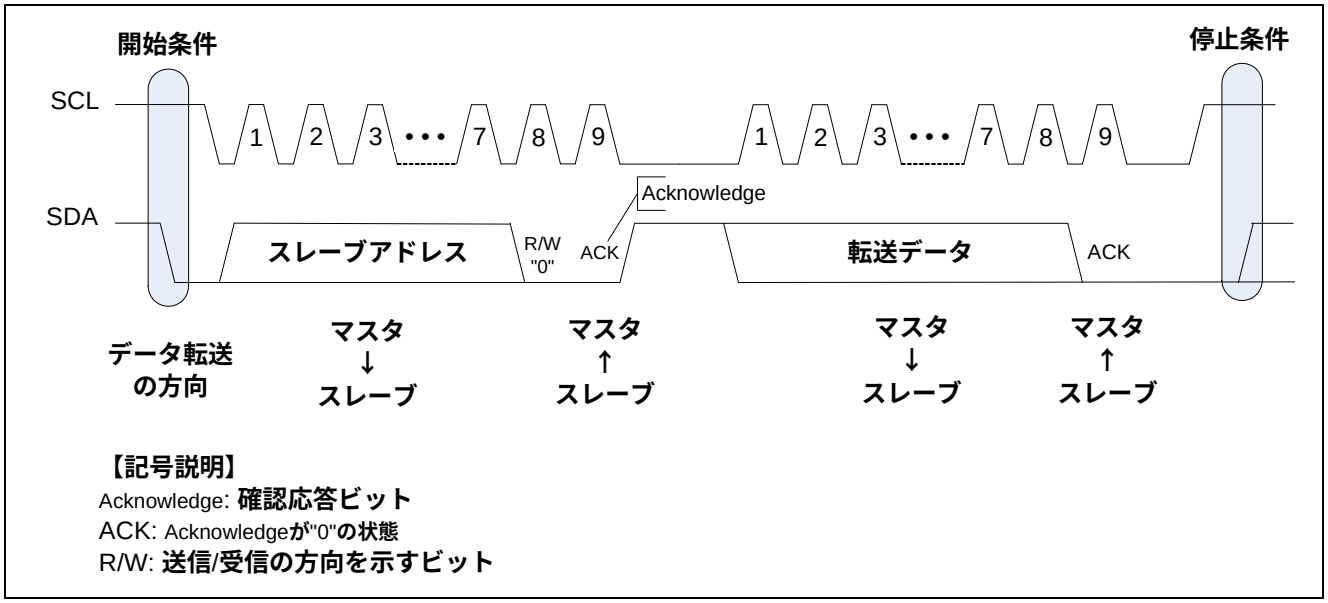


図 4　データ転送フォーマット (マスタ = 送信デバイス、スレーブ = 受信デバイスの場合)

2.3　シングルマスタとマルチマスタの構成

2.3.1　シングルマスタ

マスタデバイスは「開始条件」および「停止条件」を発行し、データ通信を管理します。また SCL ライン上にデータを送受信するための同期化クロックやスレーブアドレスを出力します。マスタデバイスが常に固定されている図 5 のようなシステム構成をシングルマスタ構成といいます。

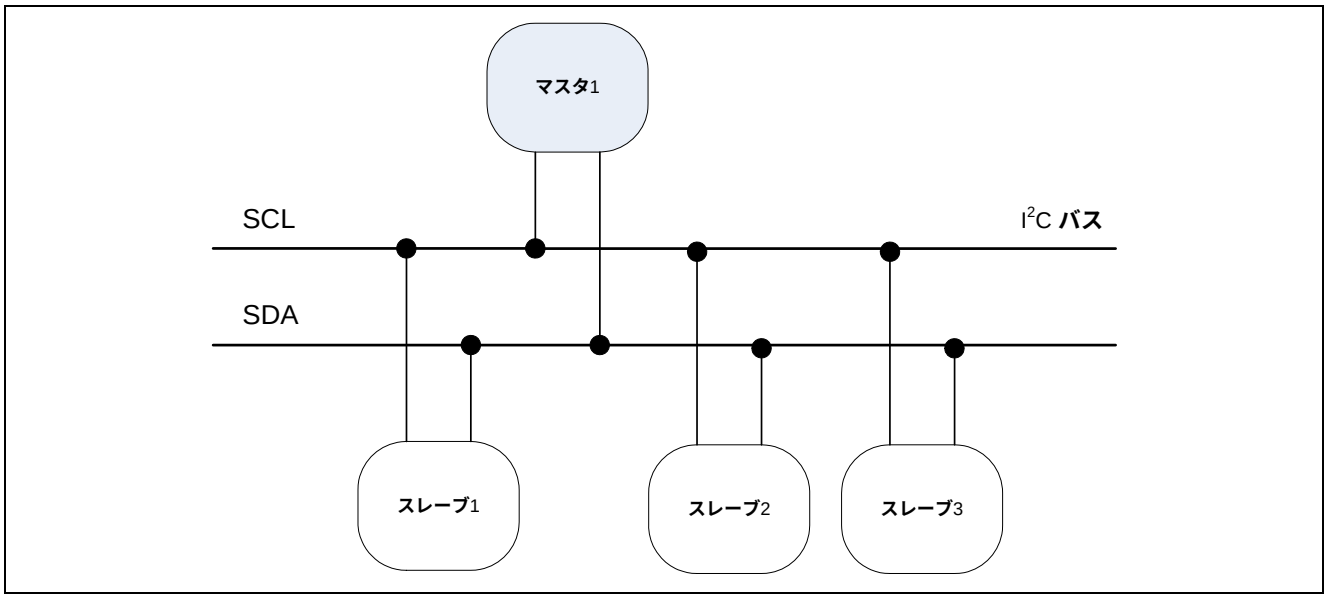


図 5　シングルマスタ構成

2.3.2　マルチマスタ

図 6 のように、1 つのシステム内にマスタと成り得るデバイスが 2 個以上存在する構成をマルチマスタ構成といいます。

マスタデバイスはバスが解放状態のときのみデータ転送を開始することができますがマルチマスタ構成の場合、複数のマスタデバイスが同時にデータ転送を開始しようとする可能性があります。つまり、バス権の衝突が生じます。このため、I2C バスの仕様にはバス権の衝突が生じた場合の通信調整手順が規定されています。

【注】　本応用例は、シングルマスタでの通信例であるため、マルチマスタの詳細な制御方法については省略します。

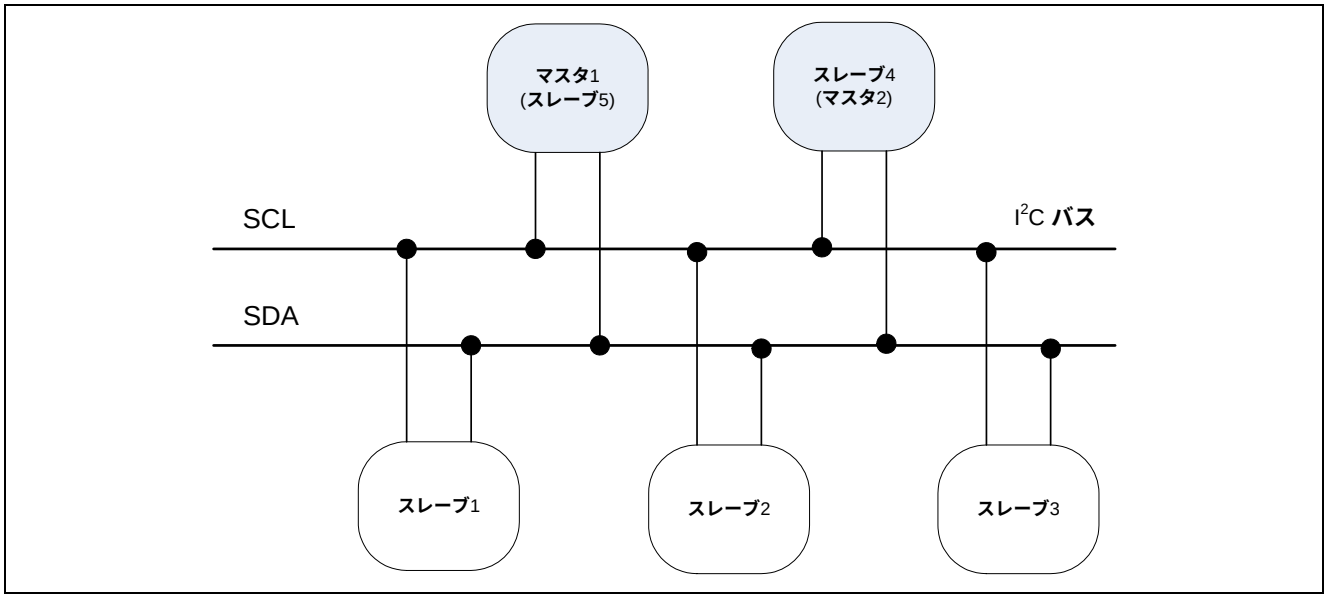


図 6　マルチマスタ構成

2.4　SH7730 の I²C バスインタフェース (IIC) 概要

SH7730 の I²C バスインタフェース (IIC) の概要を以下に記載します。ブロック図については「SH7730 グループ ハードウェアマニュアル (RJJ09B0339)」の「I²C バスインタフェースのブロック図」を参照してください。

表 1　SH7730 の I²C バスインタフェース (IIC) 概要

項目	概要
チャネル数	2 チャネル
モード	マスタモードおよびスレーブモードをサポート
連続送信/受信	シフトレジスタ、送信データレジスタ、受信データレジスタがそれぞれ独立しているため、連続送信/受信が可能
開始条件/停止条件	マスタモードでは開始条件、停止条件の自動生成
アクノリッジの出力レベル	受信時、アクノリッジの出力レベルを選択可能
アクノリッジビット	送信時、アクノリッジビットを自動ロード
ビット同期機能内蔵	マスタモードではビットごとに SCL の状態をモニタして自動的に同期を取ります。転送準備ができていない場合には、SCL を Low レベルにして待機させます。
割り込み要因	• 割り込み要因: 6 種類 ① 送信データエンプティ (スレーブアドレス一致時を含む) ② 送信終了 ③ 受信データフル (スレーブアドレス一致時を含む) ④ アービトレーションロスト ⑤ NACK 検出 ⑥ 停止条件検出
バスを直接駆動可能	SCL、SDA の 2 端子は、バス駆動機能選択時 NMOS オープンドレイン出力

2.5　SH7730 IIC 使用方法について

IIC レジスタの基本的な設定手順、各ステータスフラグのセットタイミング、各割り込み発生タイミングについては、「SH7730 グループ ハードウェアマニュアル (RJJ09B0339)」の「I²C バスインタフェース (IIC)」の章を参照ください。

3. EEPROM 使用方法について

本章では、本応用例で使用する EEPROM のライト・リードに関する設定や動作について簡単に説明します。

3.1 本応用例で使用する EEPROM について

Type No.:　ルネサス エレクトロニクス製 R1EX24064ASAS0I

メモリサイズ:　64k ビット

動作周波数:　400kHz (Max)　※本応用例では 333kHz で使用します。

※詳細は、使用する EEPROM のデータシートを参照ください。

3.2 EEPROM ライトについて

本応用例では、IIC をマスタ送信モードに設定し、EEPROM の Memory address 0x0000 番地から 10 バイトの Page Write を行います。

3.2.1 Page Write Operation

Page Write では、スタート・コンディション→デバイス・アドレス・ワード→メモリ・アドレス(n)→Write データ(Dn)の順に、9 ビットごとの Acknowledge"0"出力を確認しながら入力します。Write データ(Dn)入力後にストップ・コンディションを入力しないで、Write データ(Dn+1)を入力すると、Page Write モードに入ります。Write データ(Dn+1)を入力した時点で、ページ内アドレス(a0～a4)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、Write データを次々と入力することができ、Write データ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 32 バイト連続して Write データを入力できます。ページ内アドレス(a0～a4)がページの最終番地に達した場合は、アドレスは"Roll Over"して、ページの先頭アドレスに戻ります。ストップ・コンディションを入力すると、Write データの入力を終了し、書き換え動作に入ります。

【注】 本応用例では、10 バイトの Write データを想定しています。
32 バイト以上のデータの書き込み、または、Roll Over するアドレス境界での書き込みを Page Write で行う場合、再度、スタート・コンディションから実施してください。

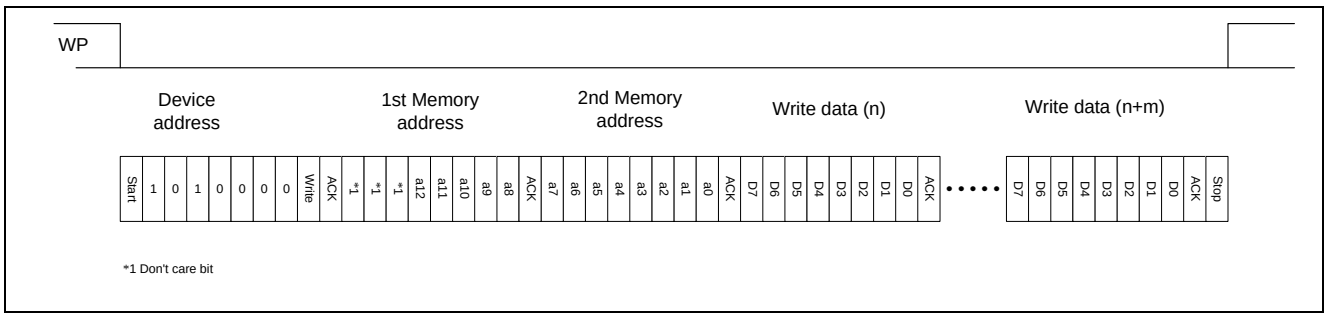


図 7　Page Write Operation フォーマット

3.2.2 Write Operation の Acknowledge 信号について

アドレス情報、Read 情報等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。Acknowledge は、この 8 ビットの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、SCL の 9 クロック目に受信側が"0"を出力します。送信側は、この 9 クロック目で Acknowledge 信号を受信するために、バスを解放します。

EEPROM から見ると、Write の場合はすべて受信となるため、8 ビットの受信が完了したら、9 クロック目に EEPROM から Acknowledge "0"を出力します。

3.2.3 Device Address Word

スタート・コンディションに続いて 8 ビットの Device Address Word を入力します。この入力で EEPROM は Read or Write の動作を開始します。Device address Word は Device code 4 ビット、Device address code 3 ビット、Read/Write code 1 ビットの 3 つのコードで構成されています。Device Address Word の上位 4 ビットはデバイスタイプを識別するデバイス・コードで、今回のケースでは、B ' 1010 となります。Device code に続けて Device address code を入力します。Device address code はバスに最大 8 つ接続されたデバイスのうち、どれを選択するかを決定します。デバイス・アドレス端子 A2～A0 の High、Low の接続が、入力された Device address code と一致したデバイスが選択されます。Device address word の 8 ビット目は R/W(Read/Write)コードです。B'0 入力の場合は Write 動作、B'1 入力の場合は Read 動作になります。

本応用例では、A2～A0 は B'0 とします。

表 2　Device address Word

Size	Device address Word (8-ビット)							
	Device code (fixed)				Device address code			R/W
64k ビット	1	0	1	0	A2	A1	A0	W

3.2.4 Acknowledge Polling

EEPROM が書き換え中か否かを判定する機能として、Acknowledge Polling があります。書き換え期間中にスタート・コンディションに続いて Device address word 8 ビットを入力します。Acknowledge Polling の場合、Read/Write コードは"0"にしてください。9 ビット目の Acknowledge で書き換え中か否かを判定します。Acknowledge の"1"は書き換え中、Acknowledge の"0"は書き換え終了を示します。Acknowledge Polling は、Write データ入力後、ストップ・コンディションが入力された時点から機能します。

3.2.5 本応用例での Page Write Operation シーケンス

本応用例では、EEPROM に対して以下のようなシーケンス処理を行います。

① 開始条件発行
② Device address 送信 (W モード)
③ 1st Memory address 送信
④ 2nd Memory address 送信
⑤ Write 処理 (1 バイト目)
⑥ Write 処理 (2 バイト目)
　　:
⑦ Write 処理 (10 バイト目)
⑧ 停止条件発行
⑨ 開始条件発行
⑩ Device address 送信 (W モード)
:　※ACK 受信時は、⑫へ。
:　※NACK 受信時は、⑪⑩を繰り返す。
⑪ 開始条件 (再送) 発行
⑫ ACK 受信後、停止条件発行

⑩～⑪⑩ : Acknowledge Polling

3.3 EEPROM リードについて

本応用例では、IIC をマスタ受信モードに設定し、EEPROM の 0x0000 番地～0x0008 番地の 9 バイト分のデータを Random Read Operation と Sequential Read Operation を組み合わせてリードします。残り 0x0009 番地の 1 バイト分のデータを Current Address Read Operation でリードします。

3.3.1 Current Address Read Operation

EEPROM 内部のアドレス・カウンタは、前回の Read もしくは Write 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) 番地をキープしています。Current Address Read は、この内部のアドレス・カウンタがキープしている (n+1) 番地を Read するモードです。

- スタート・コンディション→デバイス・アドレス・ワード (ただし R/W="1") の順に入力すると、Acknowledge "0"を出力したのち、(n+1) 番地のデータ 8 ビット が上位からシリアルに出力されます。
- この後、Acknowledge "1" (Acknowledge の入力をせずに、バスを解放しても可) →ストップ・コンディションの順で入力すると Read を終了し、スタンバイ状態に戻ります。

3.3.2 Random Read Operation

アドレスを指定して Read するモードです。

- はじめに、Write モードで Read すべきアドレスを入力します。
- スタート・コンディション→デバイス・アドレス・ワード (R/W="0") →メモリ・アドレス 8 ビット ×2 の順に入力します。
- メモリ・アドレス入力後の Acknowledge "0"出力を確認したら、再度スタート・コンディションを入力し、Current Address Read を行います。
- 上記、Write モードで指定したアドレスのデータが出力されます。
- データ出力後に、Acknowledge "1" (Acknowledge の入力をせずに、バスを解放しても可) → ストップ・コンディションの順で入力すると Read を終了し、スタンバイ状態に戻ります。

3.3.3 Sequential Read Operation

データを連続して Read するモードで、Current Address Read、Random Read ともに使用できます。

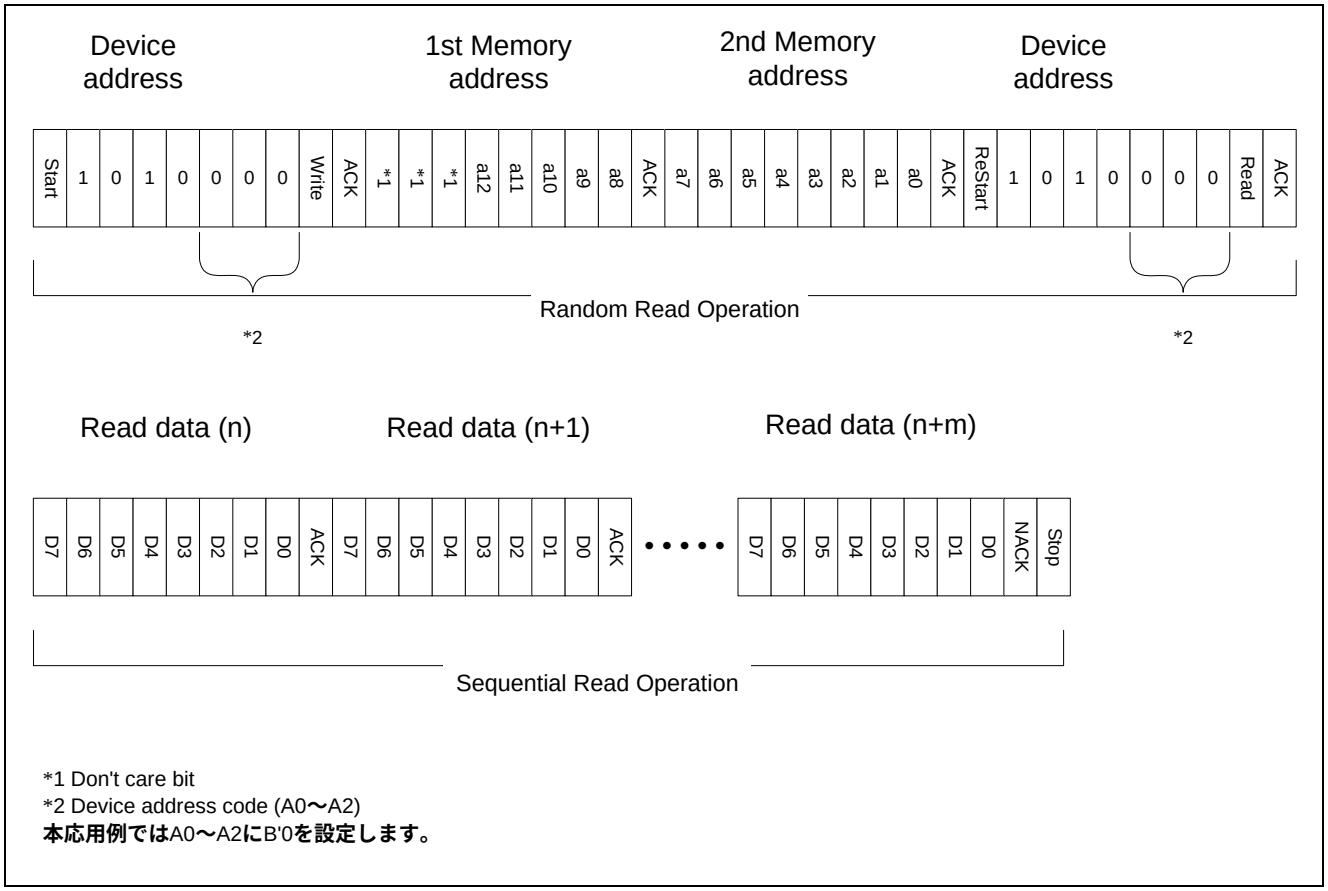
- 8 ビットのデータを出力した後、Acknowledge "0"を入力すると、アドレスがインクリメントされ、次の 8 ビットのデータが出力されます。
- データ出力後に Acknowledge "0"の入力を続けると、アドレスをインクリメントしながら次々とデータを出力します。
- アドレスが最終アドレスになった場合は、0 番地に"Roll Over"します。
- "Roll Over"後も Sequential Read が可能です。
- 動作を終了するには、Current Address Read、Random Read と同様に、Acknowledge "1" (Acknowledge の入力をせずに、バスを解放しても可) →ストップ・コンディションの順で入力します。

3.3.4　本応用例での Read Operation

(1) Random Read Operation と Sequential Read Operation の組み合わせ処理

Random Read で、EEPROM のリードするアドレス番地を指定し、その後、データを連続してリードするために、Sequential Read Operation を行います。

本応用例では、EEPROM リード/ライト処理 (iic_user_EepRomRW ( )) のパラメータのモードに D_IIC_EEP_READ を指定した場合にこの処理が行われます。



*1 Don't care bit
*2 Device address code (A0〜A2)
本応用例ではA0〜A2**に**B'0**を設定します。**

図 8　Random Read **と** Sequential Read **の組み合わせ処理**

(2) Current Address Read Operation

内部アドレス・カウンタがキープしている、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) 番地をリードします。

本応用例では、EEPROM リード/ライト処理 (iic_user_EepRomRW ( )) のパラメータのモードに D_IIC_EEP_CURRENT_READ を指定し、リードデータサイズに 1 を設定した場合にこの処理が行われます。

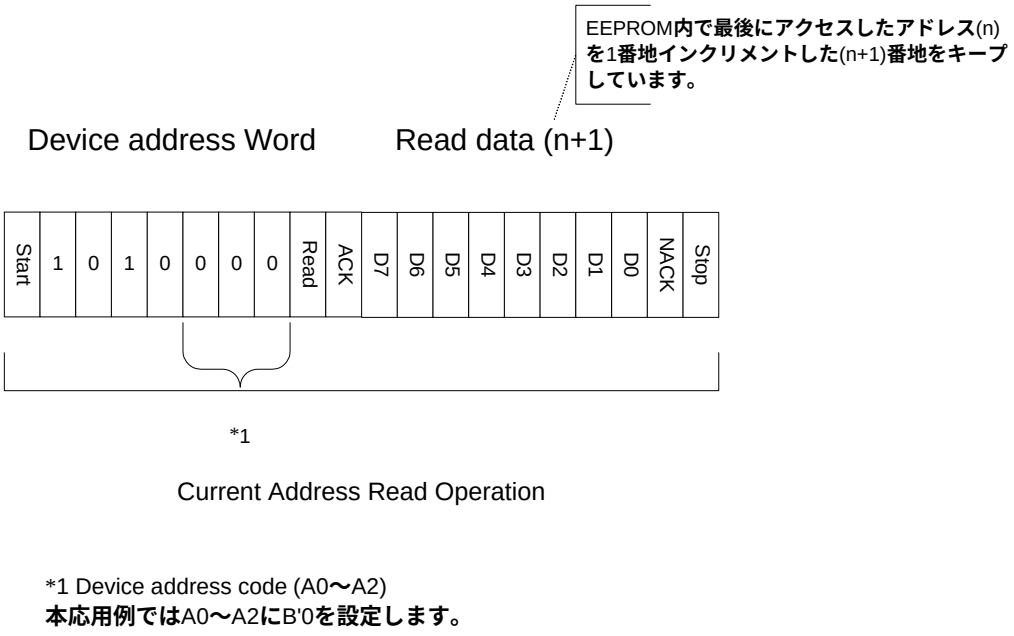


図 9　Current Address Read Operation

3.3.5 Read Operation の Acknowledge 信号について

EEPROM は、スタート・コンディションの後の Device address 受信後、Memory address 受信後、または、リスタート・コンディションの後の Device address 受信後に Acknowledge "0"を出力します。

これに続いて、EEPROM は Read データを 8 ビット単位で出力しますが、出力後はバスを解放し、マスタ側から Acknowledge "0"が送られるのを待ちます。

Acknowledge "0"検出すると、EEPROM は次のアドレスの Read データを出力します。

Acknowledge "0"が検出されずにストップ・コンディションを受信すると、Read 動作を終了しスタンバイ状態になります。

3.3.6 本応用例での Read Operation シーケンス

本応用例では、EEPROM に対して以下のようなシーケンス処理を行います。

① 開始条件発行
② Device address 送信 (W モード)
③ 1st Memory address 送信
④ 2nd Memory address 送信
⑤ 開始条件発行 (ReStart)
⑥ Device address 送信 (R モード)
⑦ Read 処理 1 (Read 用のクロック送出)
⑧ Read 処理 2 (Read 用のクロック送出)
⑨ Read 処理 3 (Read 用のクロック送出)
　　:
⑩ Read 処理 9 (Read 用のクロック送出)
⑪ NACK 送信後、停止条件発行
⑫ 開始条件発行
⑬ Device address 送信 (R モード)
⑭ Read 処理 10 (Read 用のクロック送出)
⑮ NACK 送信後、停止条件発行

Random Read と Sequential Read の組み合わせ処理

Acknowledge Polling

4. 応用例の説明

本応用例ではライトデータを 10 バイト{0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a}とします。

EEPROM にデータライト後、先頭 9 バイト分{0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09}については、Random Read Operation と Sequential Read Operation を組み合わせ、Memory address を指定してリードします。最終 1 バイトの{0x0a}については、Current Address Read Operation で、現在のアドレス位置のデータをリードします。

【参考】

Random Read Operation、Sequential Read Operation、Current Address Read Operation **については 3.3 章を参照ください。**

4.1 本応用例の動作環境

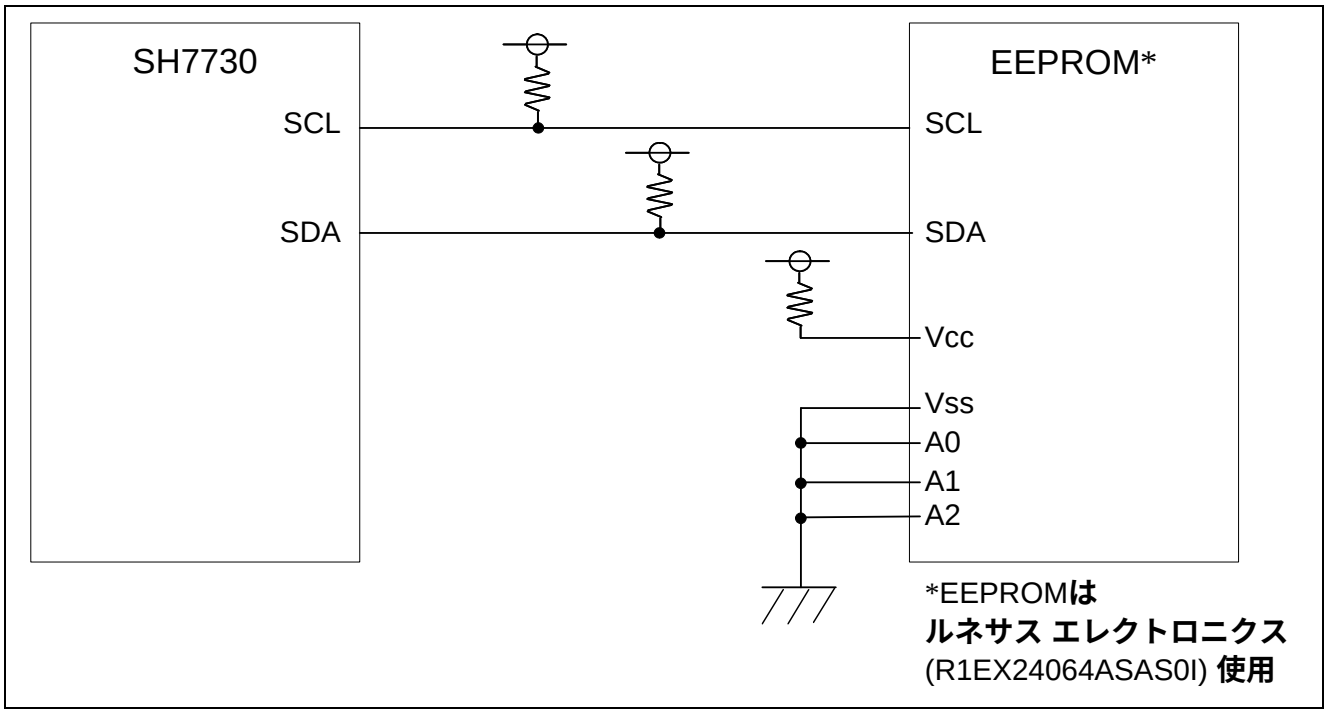


図 10　本応用例の動作環境

4.2　本応用例の処理概要

本応用例のサンプルコードでは、以下のことを行います。

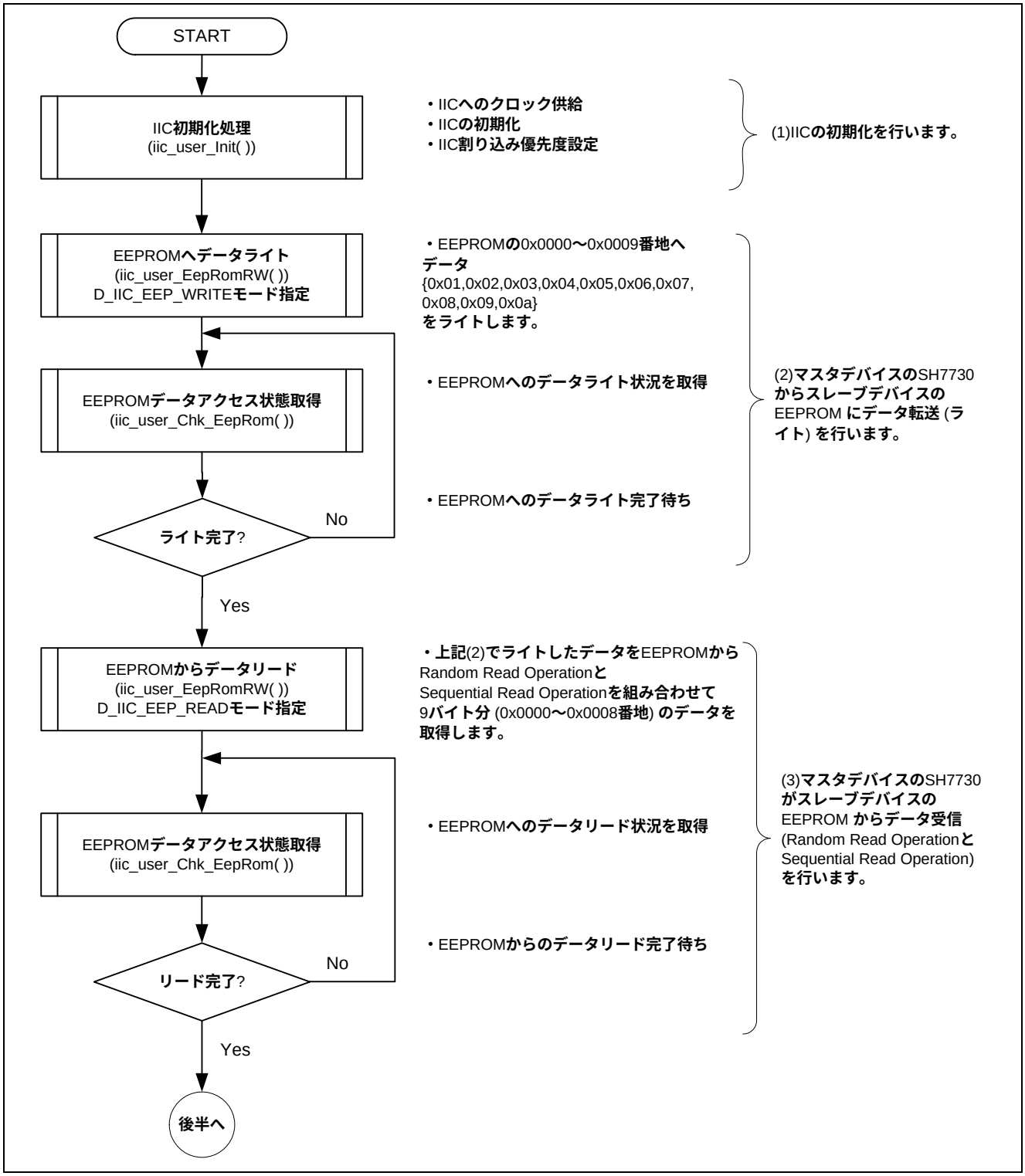


図 11　本応用例の処理フロー (前半)

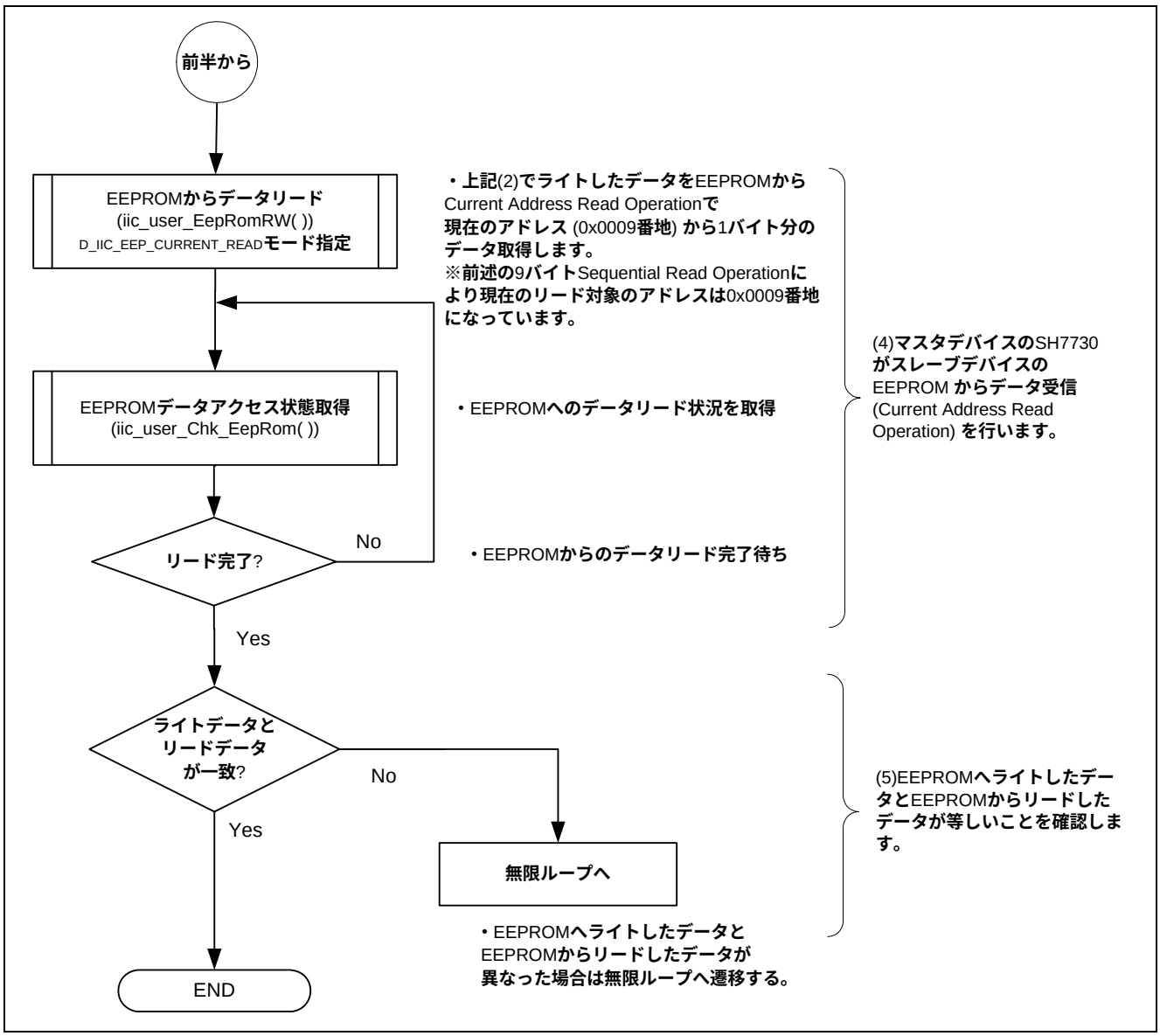


図 12　本応用例の処理フロー (後半)

4.3　参考プログラムの提供インタフェース

本応用例では、以下のユーザー提供インタフェースがあります。

表 3　提供インタフェース

No.	インタフェース	使用例
1	IIC 初期化処理 (iic_user_Init ( ))	IIC を初期化する際にコールします。 最初に必ずコールします。
2	EEPROM リード/ライト処理 (iic_user_EepRomRW ( ))	パラメータに動作モード、ライト or リード情報を設定し、ライト or リードを開始します。 ※動作モードは、4.4 章を参照ください。
3	EEPROM 状態取得処理 (iic_user_Chk_EepRom ( ))	EEPROM リード/ライト状態取得を行います。 EEPROM リード/ライトの完了が判断できます。
4	IIC 割り込み選択処理 (iic_user_int_select ( ))	割り込みハンドラに実装します。 現状態に対応する割り込み発生時の処理を行います。

4.3.1　提供インタフェース詳細

(1) IIC 初期化処理 (iic_user_Init ( ))

【概要】

IIC の初期化処理を行います。内部でクロック供給、PFC 設定、割り込み優先度設定、内部情報のクリア等を行います。

【パラメータ】

なし。

(2) EEPROM リード/ライト処理 (iic_user_EepRomRW ( ))

【概要】

指定されたパラメータに従って、EEPROM へのリード、ライトを行います。
IIC 初期化処理 (iic_user_Init ( )) 処理後、コールしてください。

【パラメータ】

構造体	型	変数	内容
T_IIC_EEPROM_RW_INFO	E_Iic_eep_mode	i_mode	**リード時** D_IIC_EEP_READ **指定** **【内容】** Random Read Operation **と** Sequential Read Operation **の組み合わせでリード処理を行います。** **現在のアドレスリード時** D_IIC_EEP_CURRENT_READ **指定** **【内容】** Current Address Read Operation **と** Sequential Read Operation **の組み合わせでリード処理を行います。** **現在の** EEPROM **内部のアドレスに対応するデータを取得します。** **【参考】** Random Read Operation、Sequential Read Operation、Current Address Read Operation **については「**3.3 **章** EEPROM **リードについて」を参照ください。** **ライト時** D_IIC_EEP_WRITE **指定**
	unsigned char	i_DevAdr	Device address code **のみを指定** ※Device code **と最終ビットの** R/W **は本インタフェース内部で設定します。** Device address code **については、**3.2.3 **章を参照ください。**
	unsigned long	i_RomAdr	Memory address **指定** ※EEPROM **へのアクセス開始アドレスを指定してください。**
	unsigned long	i_Len	**リード** or **ライトするデータサイズ** (1 **バイト単位**) **を指定してください。**
	unsigned char	*i_pBuf	**リード時** **現在のアドレスリード時** **リードデータを格納する領域を指定** **ライト時** **ライトするデータ領域を指定**

(3) EEPROM 状態取得処理 (iic_user_Chk_EepRom ( ))

【概要】

EEPROM にアクセスしている状態を取得します。

【パラメータ】

構造体	型	変数	内容
T_IIC_EEPROM_CONDITION	E_Iic_eep_condition	i_eep_condition	EEPROM アクセス状態 ※詳細は、4.5 章 表 5 参考プログラムの状態定義を参照ください。
	E_Iic_eep_err_condition	i_err_condition	EEPROM アクセスエラー状態 ※詳細は、4.7 章 表 7 EEPROM アクセスエラー状態を参照ください。

【戻り値】

EEPROM リード/ライト成功完了時	D_IIC_EEP_OK
EEPROM リード動作中	D_IIC_EEP_READING
EEPROM ライト動作中	D_IIC_EEP_WRITING
エラー発生時	D_IIC_EEP_NG

4.4　参考プログラムの動作モード

本応用例では、以下の動作モードがあります。

表 4　動作モード

No.	動作モード	概要
1	アドレス指定ライトモード (D_IIC_EEP_WRITE)	Page Write Operation と Write Cycle Polling Using ACK の組み合わせで EEPROM へライトします。
2	アドレス指定リードモード (D_IIC_EEP_READ)	Random Read Operation と Sequential Read Operation の組み合わせで EEPROM からリードします。
3	現アドレスリードモード (D_IIC_EEP_CURRENT_READ)	Current Address Read と Sequential Read Operation の組み合わせで EEPROM からリードします。

【参考】

Page Write Operation、Write Cycle Polling Using ACK、Random Read Operation、Sequential Read Operation、Current Address Read については、「3 章 EEPROM 使用方法について」を参照ください。

4.5　参考プログラムの状態

本応用例では、IIC 制御の状態を以下のように定義します。

表 5　参考プログラムの状態定義

ID	状態	状態の定義
C0	【D_IIC_EEP_NO_INIT】 未初期化状態	未初期化状態となります。 (iic_user_Init()コール前)
C1	【D_IIC_EEP_IDLE】 アイドル状態	クロック供給、PFC 設定、割り込み優先度設定・・等が完了している状態となります。(iic_user_Init()コール後) iic_user_EepRomRW()をコールすることが可能な状態となります。
C2	【D_IIC_EEP_START_ISSUE_WAIT】 開始条件発行待ち状態	開始条件（再送）発行完了待ちをしている状態となります。 TXI 割り込み発生により、開始条件（再送）発行を確認します。
C3	【D_IIC_EEP_SEND_DEVADD_WAIT】 Device Address 送信待ち状態	Device Address 送信完了待ちをしている状態となります。 TEI 割り込み発生により、Device Address 送信完了を確認します。
C4	【D_IIC_EEP_RESEND_DEVADD_WAIT】 Device Address 再送信待ち状態	Device Address 再送信完了待ちをしている状態となります。 Memory address 指定後のリード用 Device Address 送信ケースで使用される状態となります。 TEI 割り込み発生により、Device Address 再送信完了を確認します。
C5	【D_IIC_EEP_SEND_MEMADD_WAIT】 Memory address 送信待ち状態	Memory address 送信完了待ちをしている状態となります。 TEI 割り込み発生により、Memory address 送信完了を確認します。
C6	【D_IIC_EEP_SEND_RCV_DATA_WAIT】 データ送受信待ち状態	データ送受信完了待ちをしている状態となります。 TEI 割り込み発生により、データ送信完了を確認します。 RXI 割り込み発生により、データ受信完了を確認します。
C7	【D_IIC_EEP_WRITE_END_ISSUE_WAIT】 Write 完了後 停止条件発行待ち状態	Write データ送信完了後、停止条件発行完了待ちをしている状態となります。 STPI 割り込み発生により、停止条件発行完了を確認します。
C8	【D_IIC_EEP_WRITE_POLLING_WAIT】 Write Polling 待ち状態	EEPROM が書き換え完了しているかどうか確認するため、Write Polling 完了待ちをしている状態となります。 TEI 割り込み発生により、Write Polling 完了を確認します。
C9	【D_IIC_EEP_END_ISSUE_WAIT】 停止条件発行待ち状態	Write Polling 完了後、停止条件発行完了待ちをしている状態となります。 STPI 割り込み発生により、停止条件発行完了を確認します。

4.6　参考プログラムのイベント

本参考プログラムでのイベントを、以下のように定義します。

割り込み関連のみでなく、iic_user_Init()や iic_user_EepRomRW( )等の提供インタフェースがコールされた際も、本応用例では、イベントとして定義します。

表 6　参考プログラムのイベント定義

ID	イベント	イベントの定義
EV0	【D_IIC_EEP_EV_INIT】	iic_user_Init()コール
EV1	【D_IIC_EEP_EV_RW_START】	iic_user_EepRomRW()コール
EV2	【D_IIC_EEP_EV_CHECK】	iic_user_Chk_EepRom ()コール
EV3	【D_IIC_EEP_EV_INT_NAKI】	NAKI 割り込み発生
EV4	【D_IIC_EEP_EV_INT_TXI】	TXI 割り込み発生
EV5	【D_IIC_EEP_EV_INT_TEI】	TEI 割り込み発生
EV6	【D_IIC_EEP_EV_INT_RXI】	RXI 割り込み発生
EV7	【D_IIC_EEP_EV_INT_STPI】	STPI 割り込み発生

4.7　参考プログラムのエラー状態

本参考プログラムでのエラー状態を、以下のように定義します。

表 7　EEPROM アクセスエラー状態

ID	状態	状態の定義
EE0	【D_IIC_EEP_ERR_NO】	エラーなし
EE1	【D_IIC_EEP_ERR_NACK】	送信時 NACK 受信した場合
EE2	【D_IIC_EEP_ERR_AL】	AL (アービトレーションロスト) 発生した場合
EE3	【D_IIC_EEP_ERR_WCT_OVER】	Write Polling count over 時
EE4	【D_IIC_EEP_ERR_OTHER】	その他エラー

【参考】

本応用例では、エラーケースにおいて、空関数を実装するのみで、特に何も実装しておりません。ユースケースに応じて、個別に対応する処理を追加ください。

4.8　参考プログラムの状態遷移

本応用例では、提供インタフェースのコール、または、IIC 関連割り込み発生をトリガに状態遷移します。

以下 (4.8.1～4.8.3) に、各動作モードの状態遷移を記載します。

- 主に①～の順番で状態遷移が行われます。(※エラーケースは省略します。)
- 図中のイベント (EV0～) については、4.6 章 表 6 参考プログラムのイベント定義を参照ください。
- EV2 (iic_user_Chk_EepRom ()コール) については、状態をチェックするのみで状態遷移を行いませんので記載を省略します。

4.8.1　D_IIC_EEP_WRITE モードの状態遷移

本応用例での、マスタデバイスの SH7730 からスレーブデバイスの EEPROM にデータ転送 (D_IIC_EEP_WRITE モード) する際の状態遷移を記載します。

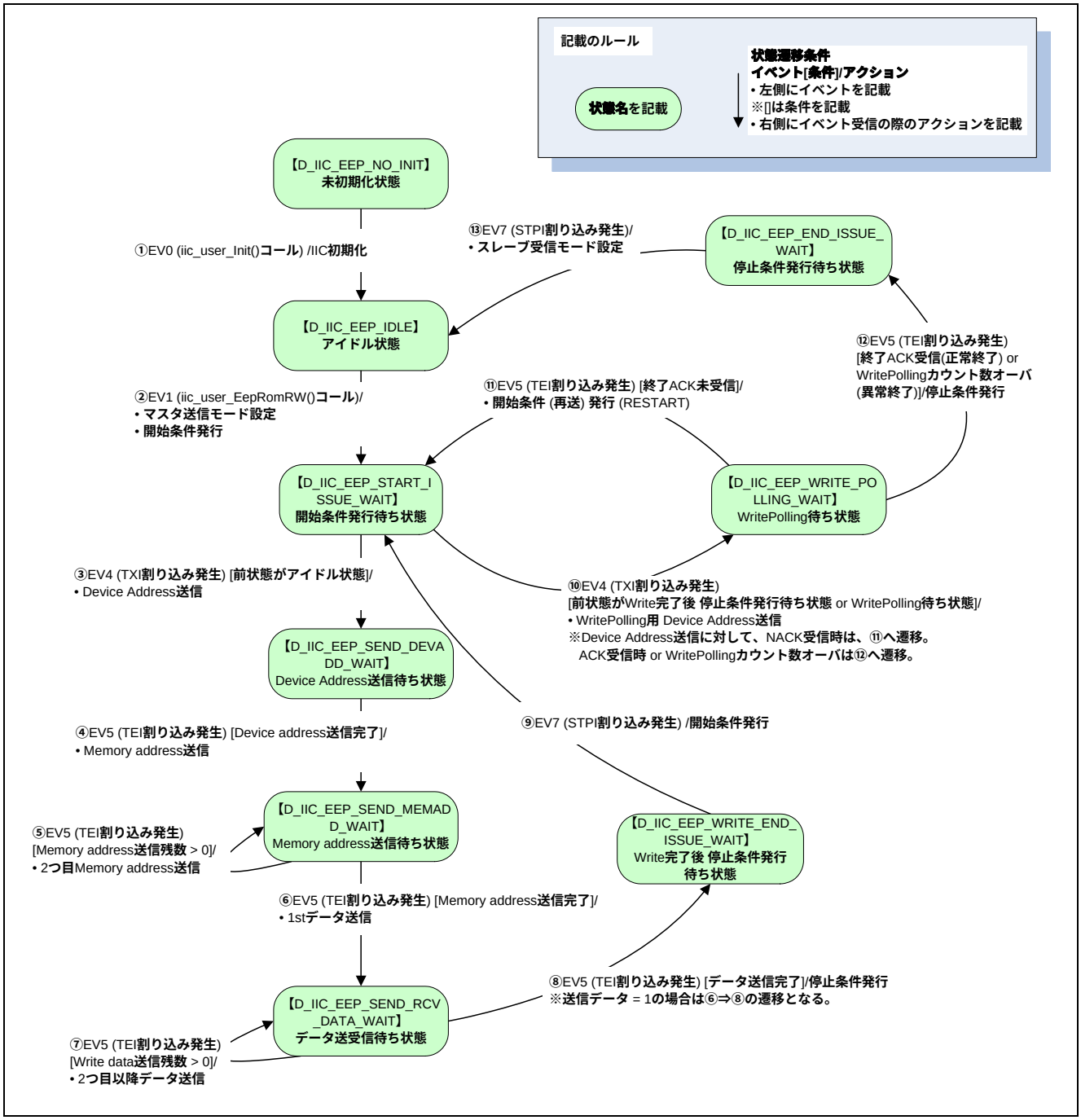


図 13　参考プログラムの状態遷移図 (D_IIC_EEP_WRITE モード)

4.8.2　D_IIC_EEP_READ モードの状態遷移

本応用例での、マスタデバイスの SH7730 からスレーブデバイス EEPROM のデータを受信 (D_IIC_EEP_READ モード) する際の状態遷移を記載します。

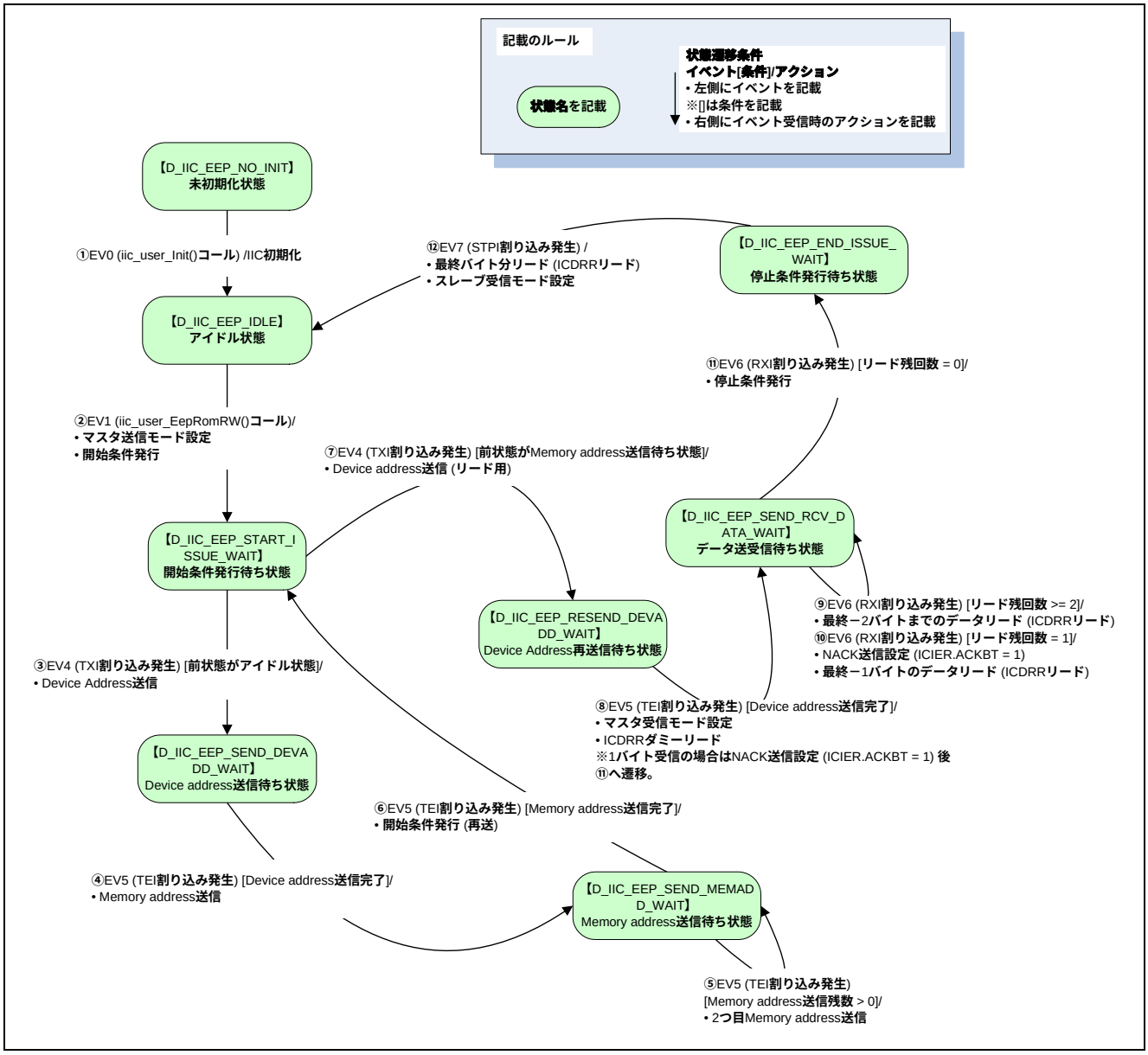


図 14　参考プログラムの状態遷移図 (D_IIC_EEP_READ **モード**)

4.8.3　D_IIC_EEP_CURRENT_READ モードの状態遷移

本応用例での、マスタデバイスの SH7730 からスレーブデバイス EEPROM のデータを受信 (D_IIC_EEP_CURRENT_READ モード) する際の状態遷移を記載します。

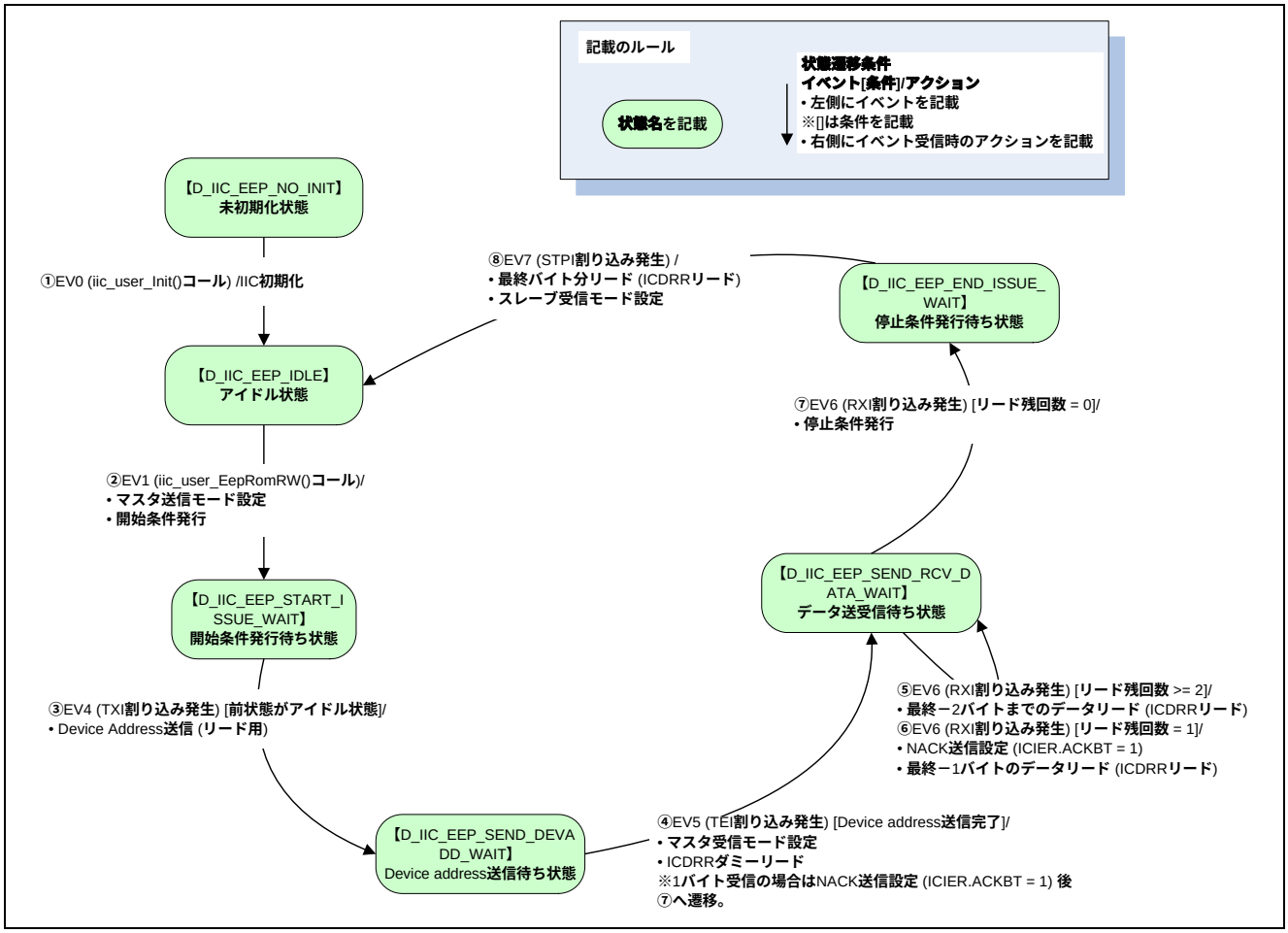


図 15　参考プログラムの状態遷移図 (D_IIC_EEP_CURRENT_READ モード)

4.9　参考プログラムの状態遷移表

本参考プログラムでは、4.5 章 表 5 の各状態で、4.6 章 表 6 のイベントを受信した際に動作する処理を、以下の状態遷移表 (g_Iic_Eep_event_tbl[][]) に定義しています。C0～については、4.5 章 表 5 の ID を参照ください。EV0～については、4.6 章 表 6 の ID を参照ください。また、Func1～の処理については、表 9 状態遷移表登録関数を参照ください。

表 8　状態遷移表 (g_Iic_Eep_event_tbl[][])

状態＼イベント		EV0	EV1	EV2	EV3	EV4	EV5	EV6	EV7
C0	【D_IIC_EEP_NO_INIT】 **未初期化状態**	Func1	NOP	NOP	NOP	NOP	NOP	NOP	NOP
C1	【D_IIC_EEP_IDLE】 **アイドル状態**	Func1	Func2	Func3	Func4	NOP	NOP	NOP	NOP
C2	【D_IIC_EEP_START_ISSUE_WAIT】 **開始条件発行待ち状態**	NOP	NOP	Func3	Func4	Func5	NOP	NOP	NOP
C3	【D_IIC_EEP_SEND_DEVADD_WAIT】 Device Address **送信待ち状態**	NOP	NOP	Func3	Func4	NOP	Func6	NOP	NOP
C4	【D_IIC_EEP_RESEND_DEVADD_WAIT】 Device Address **再送信待ち状態**	NOP	NOP	Func3	Func4	NOP	Func7	NOP	NOP
C5	【D_IIC_EEP_SEND_MEMADD_WAIT】 Memory address **送信待ち状態**	NOP	NOP	Func3	Func4	NOP	Func8	NOP	NOP
C6	【D_IIC_EEP_SEND_RCV_DATA_WAIT】 **データ送受信待ち状態**	NOP	NOP	Func3	Func4	NOP	Func9	Func10	NOP
C7	【D_IIC_EEP_WRITE_END_ISSUE_WAIT】 Write **完了後 停止条件発行待ち状態**	NOP	NOP	Func3	Func4	NOP	NOP	NOP	Func11
C8	【D_IIC_EEP_WRITE_POLLING_WAIT】 Write Polling **待ち状態**	NOP	NOP	Func3	Func4	NOP	Func12	NOP	NOP
C9	【D_IIC_EEP_END_ISSUE_WAIT】 **停止条件発行待ち状態**	NOP	NOP	Func3	Func4	NOP	NOP	NOP	Func13

【参考】

NOP **は無処理を表します。本応用例では、ある状態で意図しないイベントが通知された場合には、全て** NOP **としています。**

4.10　状態遷移表登録関数

状態遷移表に登録する関数 (表 8 の Func1～) を、以下のように定義します。

各関数の詳細は、4.16 参考プログラムの処理フローを参照ください。

表 9　状態遷移表登録関数

処理	関数名	概要
Func1	iic_Init ()	IIC 初期化処理
Func2	iic_EepRomRW ()	EEPROM リード/ライト処理
Func3	iic_Chk_EepRom ()	EEPROM 状態取得処理
Func4	iic_AL_generate ()	AL 発生時処理
Func5	iic_After_start_issue ()	開始条件発行後処理
Func6	iic_Send_After_Dev_Address ()	初回 Device Address 送信後処理
Func7	iic_After_Re_DevAdd_send ()	Device Address 再送信後処理
Func8	iic_MemAdd_Sending ()	Memory address 送信中処理
Func9	iic_Write_Data_sending ()	WriteData 送信中処理
Func10	iic_Read_Data_rcv ()	ReadData 受信処理
Func11	ic_After_end_issue_Write_Polling ()	停止条件発行後 Write Polling 準備処理
Func12	iic_Eep_WritePolling ()	WritePolling 処理
Func13	iic_After_end_issue ()	停止条件発行後処理

4.11　参考プログラムの内部情報

本参考プログラムで使用する管理情報は以下となります。

4.11.1　EEPROM リード/ライト管理情報 (T_IIC_EEPROM_RW_MANAGE)

表 10　EEPROM リード/ライト管理情報 (T_IIC_EEPROM_RW_MANAGE)

名称	型	内容
R/W モード	E_iic_eep_mode	現在の動作モードを管理するための情報です。
Device address	unsigned char	EEPROM へ送信する Device address 情報を格納します。
Memory address length	unsigned long	EEPROM へ送信する Memory address の送信回数 (1 バイト単位) のカウンタ情報を格納します。
Memory address buffer	unsigned char	EEPROM へ送信する Memory address 情報を格納します。 ※本応用例で使用する EEPROM では、Memory address として 2 バイト分の容量を確保します。
R/W Data length	unsigned long	EEPROM へライト or リードするデータの送信回数 (1 バイト単位) のカウンタ情報を格納します。
R/W Data Buffer pointer • リード時 リードデータ格納領域 • ライト時 ライトデータ格納領域	unsigned char *	• リード時 EEPROM からリードする情報を格納するリードデータ格納領域のポインタを指定します。 • ライト時 EEPROM へライトする情報が格納されている領域のポインタを指定します。
Write polling counter	unsigned short	Write cycle polling の NACK 受信時の Device address 再送処理の実行回数カウンタを格納します。

4.12　参考プログラムのマクロ定義

本参考プログラムで事前に設定が必要なマクロは以下となります。

表 11　参考プログラムのマクロ定義

マクロ定義	設定値	機能
D_IIC_EEP_INT_LEVEL	2	状態遷移表 (g_Iic_Eep_event_tbl[][]) に登録されている関数が動作中の割り込みマスクレベルを設定します。 本応用例では、この割り込みマスク設定により、IIC 関連の割り込みよりも、状態遷移表 (g_Iic_Eep_event_tbl[][]) に登録されている関数が優先的に行われるようにして、内部情報の整合性を保っています。 【注】 ・ IIC 関連の割り込み優先度設定以上の値を設定して使用ください。 ・ 本応用例では、この設定値より優先度が高い割り込み処理で、内部情報に影響を与える、提供インタフェースのコールは想定しておりません。
D_IIC_EEP_MEMADR_SIZE	2	送信する Memory Address のバイト数を設定します。 本応用例で使用する EEPROM は、Memory Address として 2 バイト分必要とするため 2 を設定します。
D_IIC_DEV_CODE	H'A0	Device Code の設定値 ※下位 4 ビットは使用しません。 本応用例では、Device Code は B'1010 となります。
D_IIC_W_CODE	H'00	Device Address Word の R/W が Write の場合の設定値 ※下位 1 ビットのみ使用
D_IIC_R_CODE	H'01	Device Address Word の R/W が Read の場合の設定値 ※下位 1 ビットのみ使用

4.13　参考プログラムのエラー対応

本応用例での、IIC 通信中の AL (アービトレーションロスト) 発生時 、または、送信時のスレーブデバイス (EEPROM) からの NACK 受信時の対応について説明します。

4.13.1　AL 発生時

本応用例では、NACK 受信インタラプトイネーブル (NAKIE) を有効にし、かつ、ICIER レジスタのアクノリッジビット判定選択 (ACKE) ビットを 0 に設定することにより、AL 発生時に、NACK 受信割り込み (NAKI) を発生させています。

AL が発生した際に、呼び出される関数は、AL 発生時処理 (iic_AL_generate ()) となります。

【注】 本応用例では、シングルマスタのみを想定し、かつ、以下の状況は発生しないことを想定しているため、AL 発生時処理 (iic_AL_generate ()) の実装はしておりません。
- **マスタ送信モードの場合、SCL の立ち上がりで内部 SDA と SDA 端子のレベルが不一致のとき**
- **マスタモードの場合、開始条件検出時、SDA 端子が High レベルのとき**

4.13.2　スレーブデバイス (EEPROM) からの NACK 受信時

本応用例では、ICIER レジスタのアクノリッジビット判定選択 (ACKE) ビットを 0 に設定することで、NACK 受信時に NACK 受信割り込み (NAKI) は発生しません。

このため、スレーブデバイス (EEPROM) へデータ送信後 (TEI 割り込み発生) のタイミングで、ICIER レジスタの ACKBR ビットを参照し、ACK or NACK の判定をしております。

【本応用例での対応例】

例として、Device address 送信時にスレーブデバイス (EEPROM) から NACK 受信した場合について考えます。

Device address 送信待ち状態で、Device address 送信 ＋ (ACK or NACK) 受信後に、TEI 割り込みが発生し、状態遷移表に登録されている、初回 Device Address 送信後処理 (iic_Send_After_Dev_Address()) が実行されます。

初回 Device Address 送信後処理 (iic_Send_After_Dev_Address()) の先頭で、NACK 検出判定処理 (iic_judge_NACK()) をコールし、その中で、ICIER レジスタの ACKBR ビットを参照し、ACKBR = 1 であれば、NACK 受信と判定しています。

【注】 本応用例では、NACK 検出判定処理 (iic_judge_NACK()) の実装はしておりませんので、システムに見合った処理を実装ください。

4.14 参考プログラムの使用上の注意事項

4.14.1 マスタ受信モード時の注意

8 クロック目の立ち下がり付近で I²C バス受信データレジスタ (ICDRR) をリードすると、受信データが取得できない場合があります。

また、受信バッファフルかつ 8 クロック目の立ち下がり付近で I²C バスコントローラレジスタの受信ディスエーブルビット (RCVD) を 1 に設定すると、停止条件の発行ができなくなる場合があります。

そのため、以下の 1.もしくは 2.どちらかの方法で対応する必要があります。

1. マスタ受信モードで ICDRR をリードする処理は 8 クロックの立ち下がりまでに行ってください。
2. マスタ受信モードは RCVD を 1 にセットし、1 バイトごとの通信で処理してください。

本参考プログラムでは、この対応として、RCVD を 1 にセットして 1 バイトごとの通信を行っています。

4.14.2 マスタ受信モード、ACKBT 設定時の注意

マスタ受信モード動作時、連続転送している最終データの 8 つ目の SCL が立ち下がる前に ACKBT を設定してください。スレーブ送信側デバイスがオーバランする可能性があります。

本参考プログラムでは、RCVD を 1 にセットして 1 バイトごとの通信を行っていますので本注意事項には該当しません。

4.14.3 停止/開始条件発行時の注意

停止条件の発行および開始条件 (再送) の発行は 9 クロック目の立ち下がりを認識してから行ってください。9 クロック目の立ち下がりは I²C バスコントロールレジスタ 2 (ICCR2) の SCLO ビットをチェックすることにより認識することができます。下記 1.または 2.の条件下で、かつ特定のタイミングで停止条件の発行および開始条件 (再送) の発行を行ったとき、停止条件および開始条件 (再送) が正常に出力されない場合があります。この条件以外での使用は問題ありません。

1. SCL バスの負荷 (負荷容量、プルアップ抵抗) により SCL の立ち上がりが「ビット同期回路」の項に規定されている時間以上なまっている場合
2. スレーブデバイスが 8 クロック目と 9 クロック目の Low レベル期間を引っ張ってビット同期回路が働いた場合

本参考プログラムでは、停止条件の発行および開始条件 (再送) の発行の際に、SCLO ビットをチェックしています。

4.14.4 I²C バス動作中における ICE および IICRST のアクセスの注意

I²C バス動作中に、ICCR1 の ICE に 0 をライトもしくは ICCR2 の IICRST に 1 をライトすると、ICCR2 の BBSY と ICSR の STOP は不定となります。

本参考プログラムでは、I²C バス動作中に、ICCR1 の ICE に 0 をライトもしくは ICCR2 の IICRST に 1 をライトする処理を行なわないようにしています。

4.15　状態遷移時の処理

提供インタフェースをコールした際、割り込み発生した際に、状態遷移表 (g_Iic_Eep_event_tbl[][]) に登録されている関数がコールされる仕組みについて記載します。

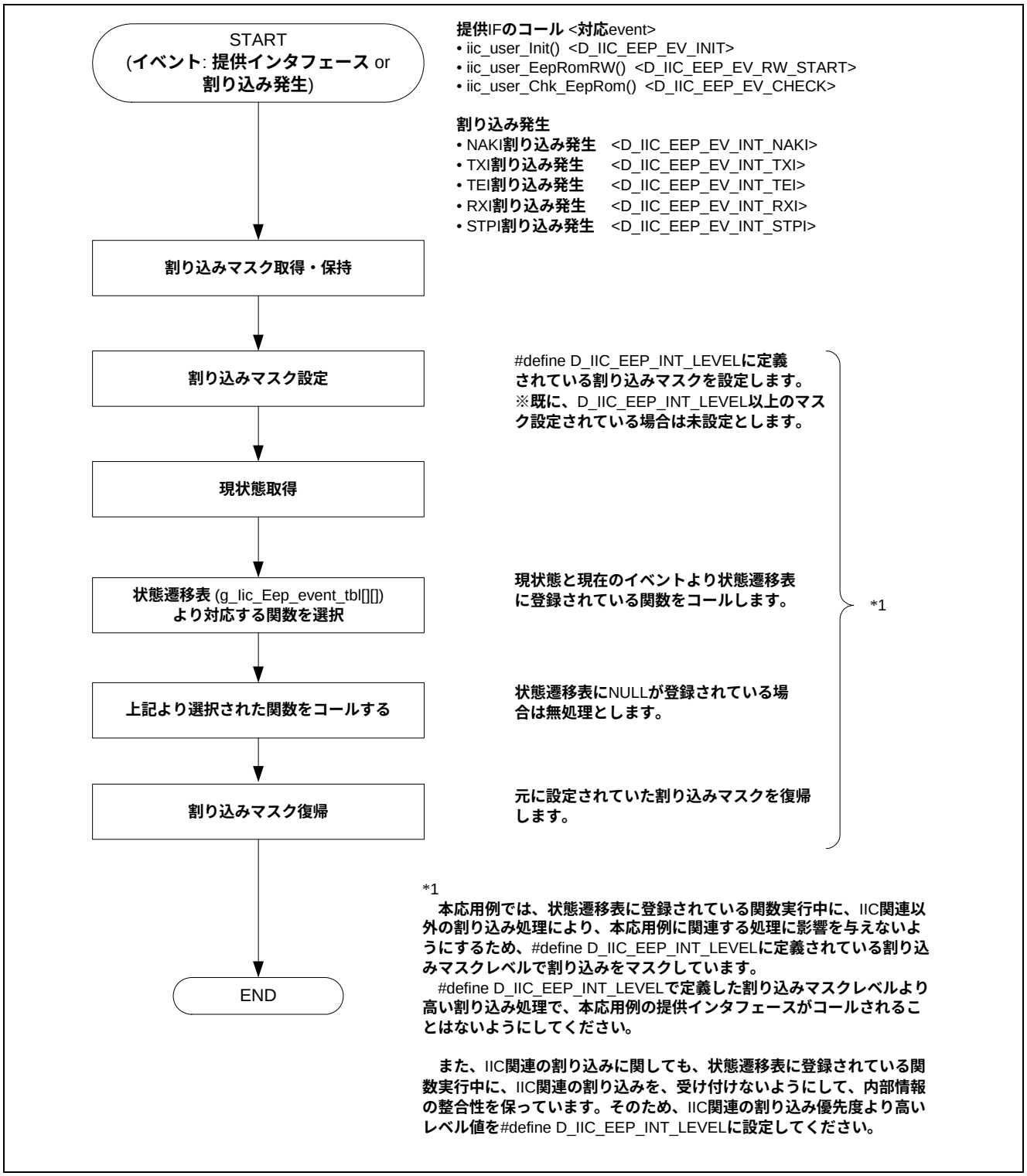


図 16　状態遷移表登録関数がコールされる仕組み

4.16　参考プログラムの処理フロー (状態遷移表登録関数)

本章では、表 8 状態遷移表 (g_Iic_Eep_event_tbl[][]) に定義されている各関数の処理フローを記載します。

4.16.1　IIC 初期化処理

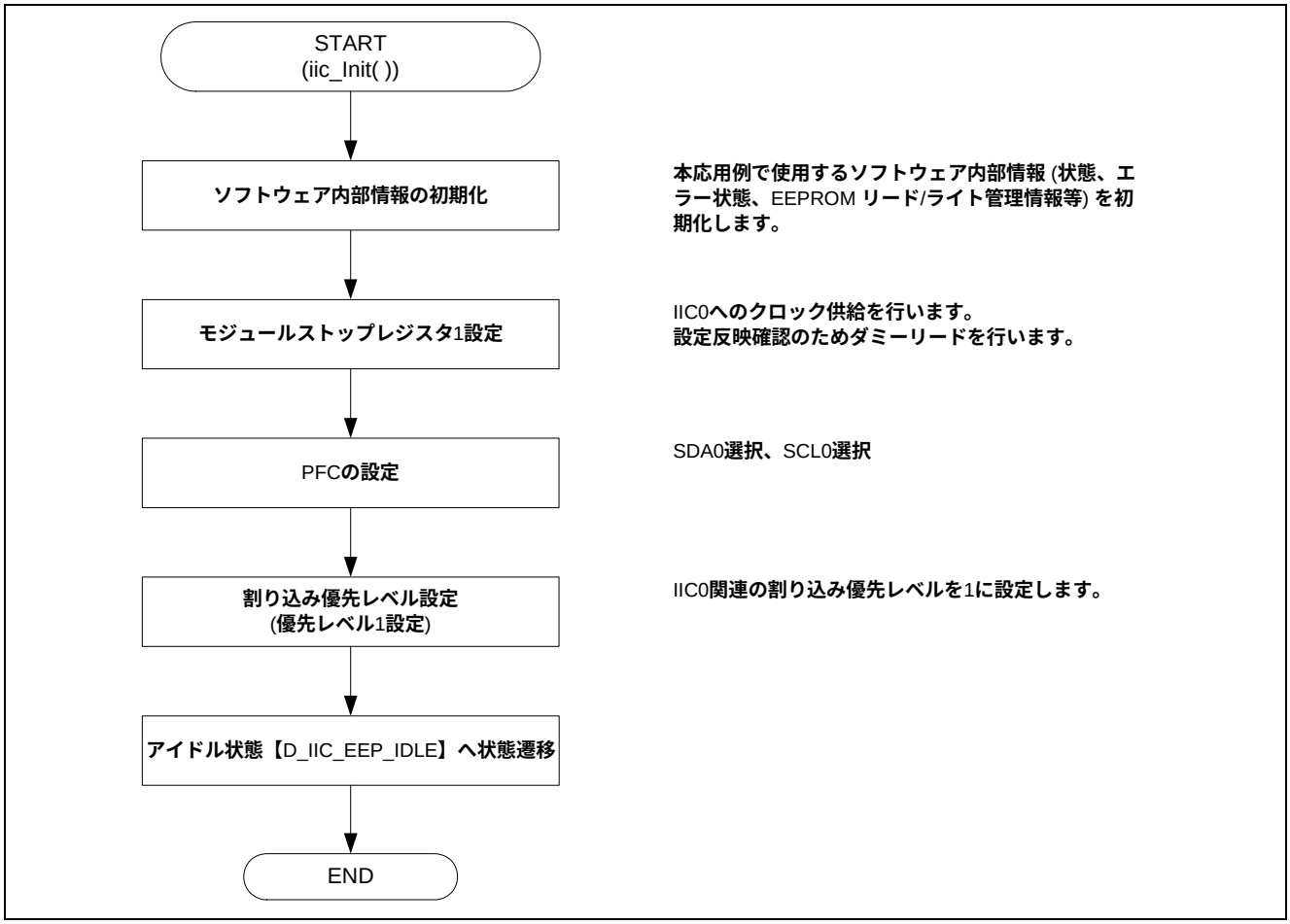


図 17　IIC 初期化処理

4.16.2 EEPROM リード/ライト処理

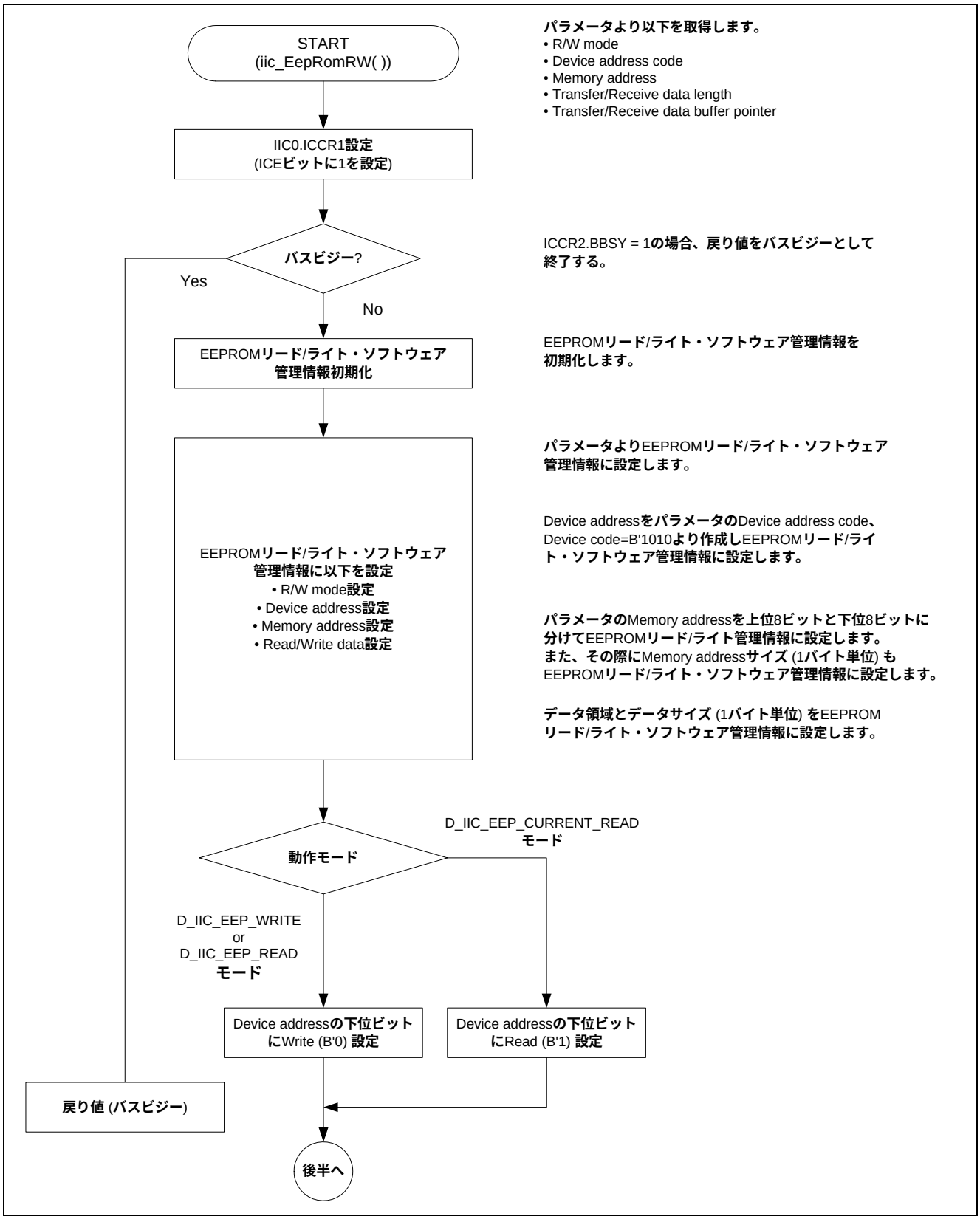


図 18　EEPROM リード/ライト処理 (前半)

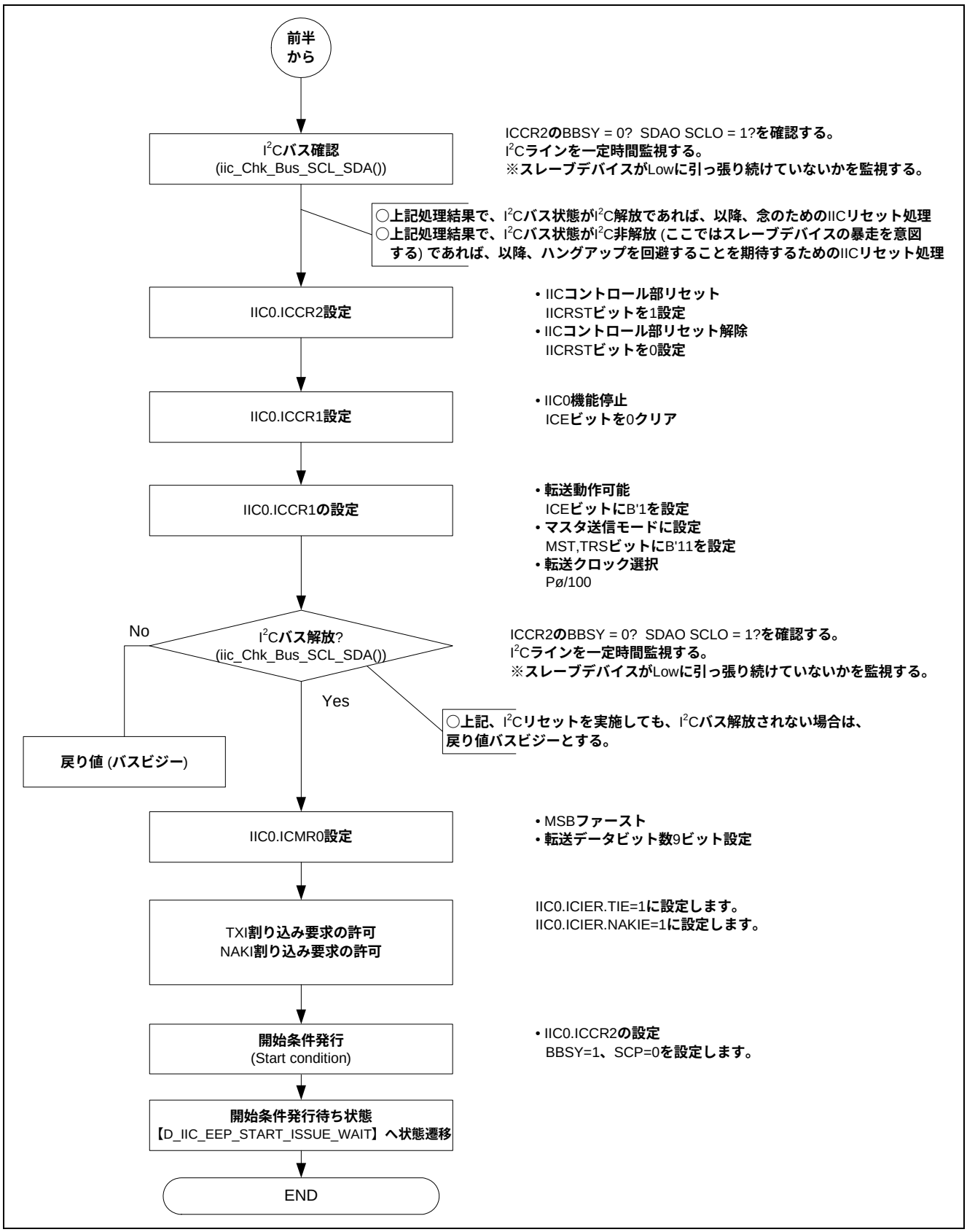


図 19　EEPROM リード/ライト処理 (後半)

4.16.3　EEPROM 状態取得処理

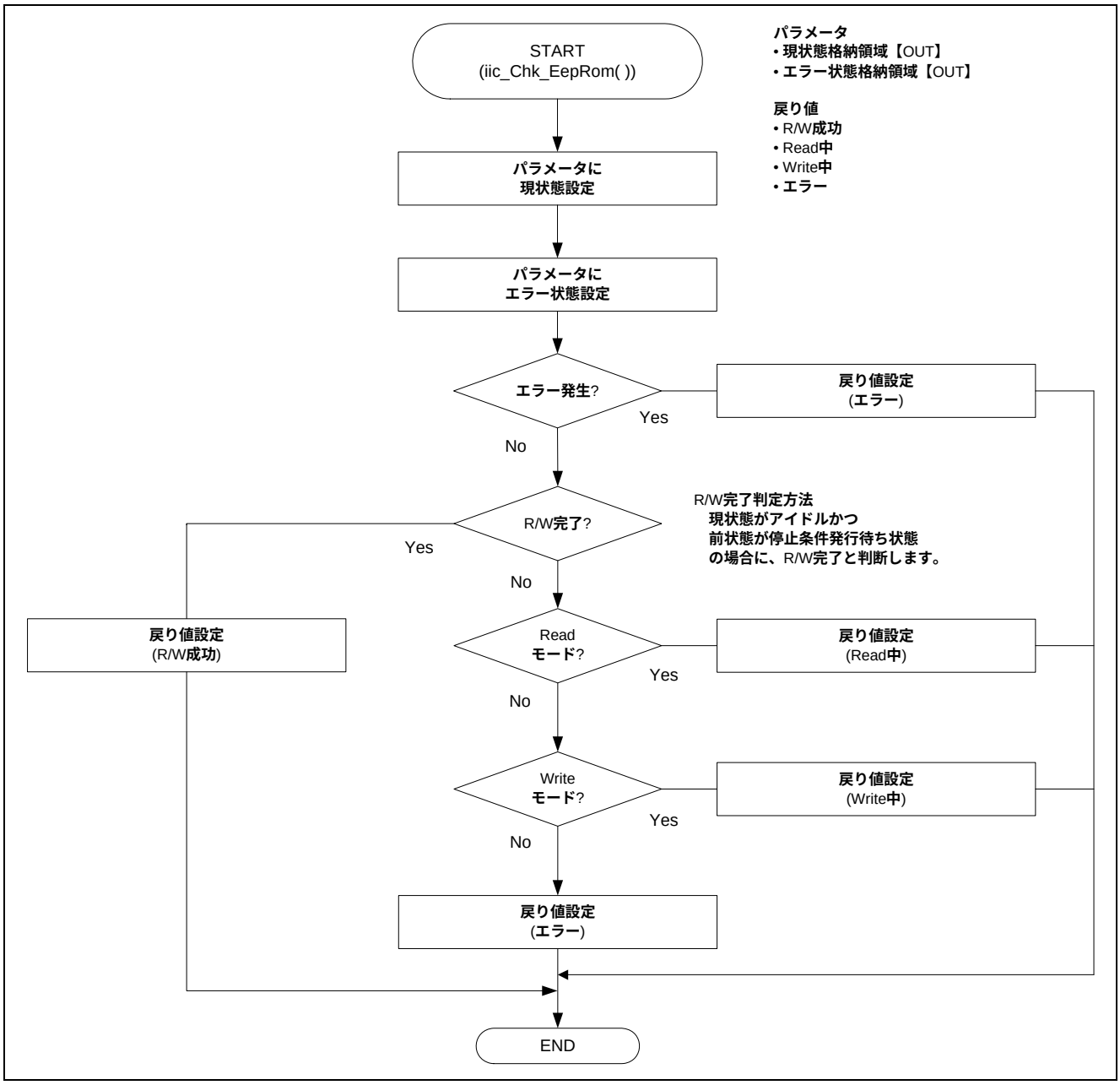


図 20　EEPROM 状態取得処理

4.16.4　AL 発生時処理

本応用例では、AL 発生は想定外とします。

処理実装はしておりませんので、システムに見合った処理を実装ください。

4.16.5　IIC 開始条件発行後処理

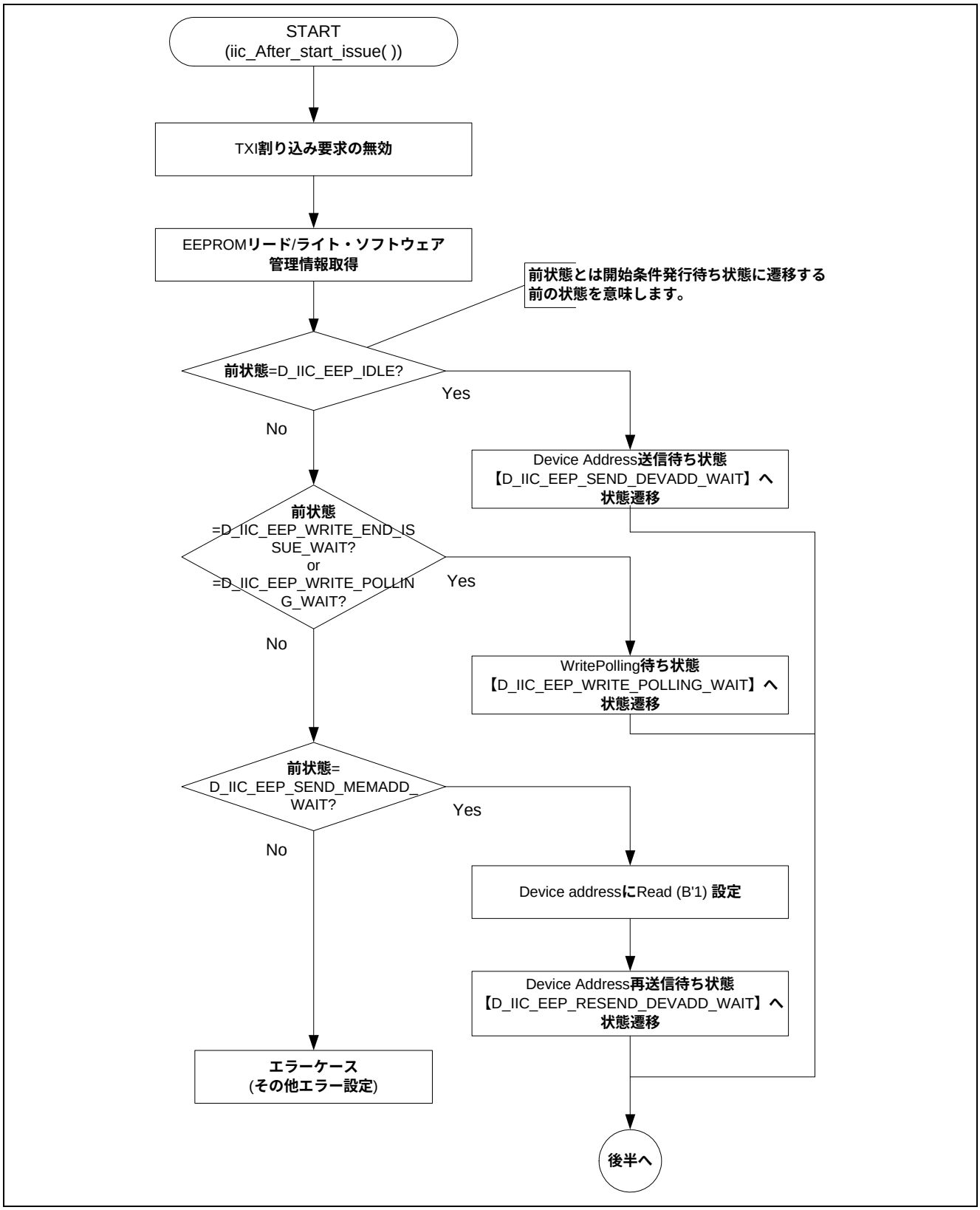


図 21　IIC 開始条件発行後処理 (前半)

図 22　開始条件発行後処理 (後半)

4.16.6　初回 Device Address 送信後処理

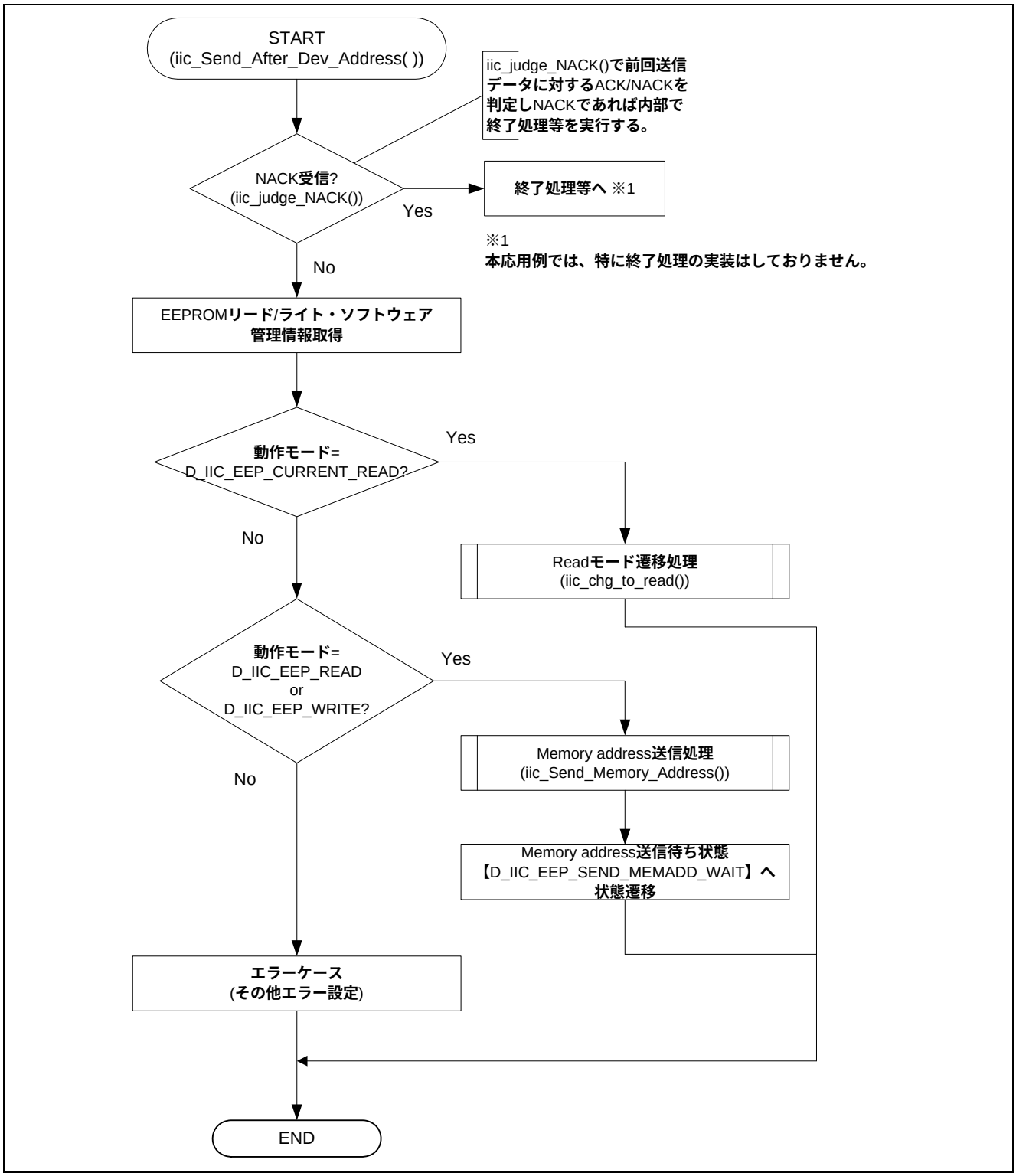


図 23　初回 Device Address 送信後処理

4.16.7　Device Address 再送信後処理

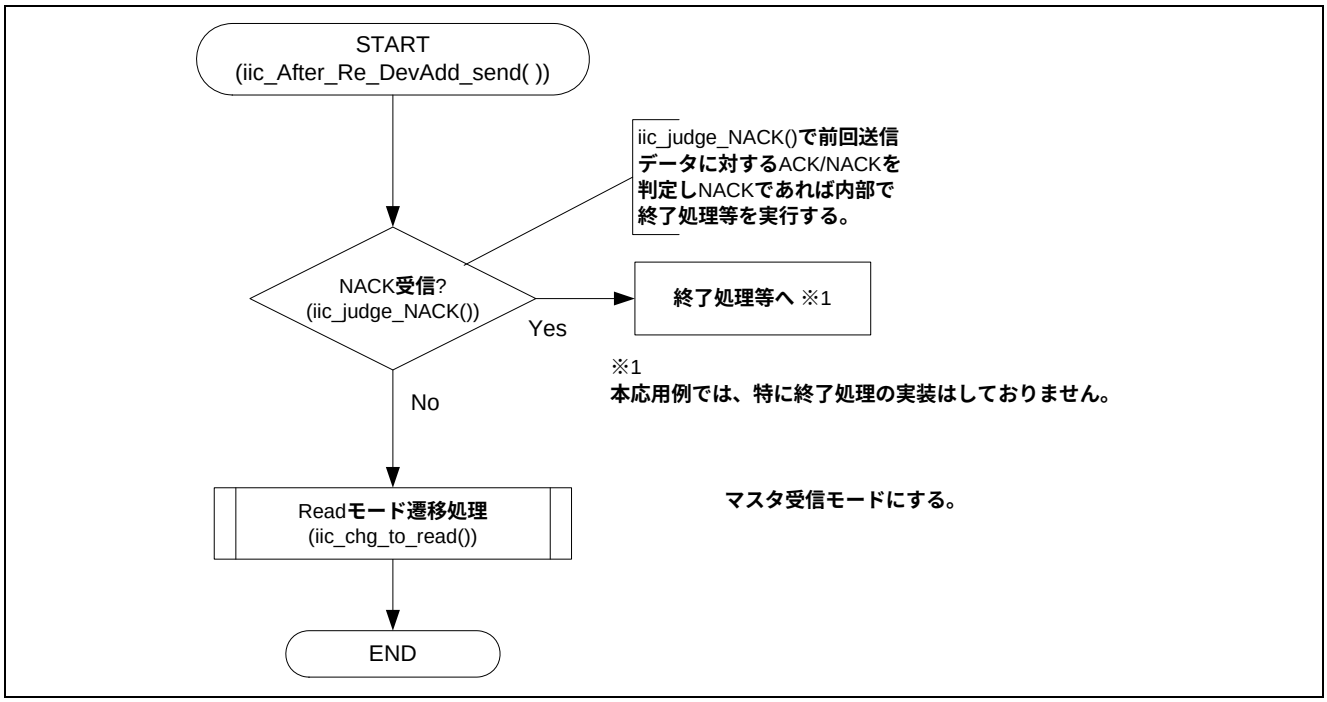


図 24　Device Address 再送信後処理

4.16.8　Memory address 送信中処理

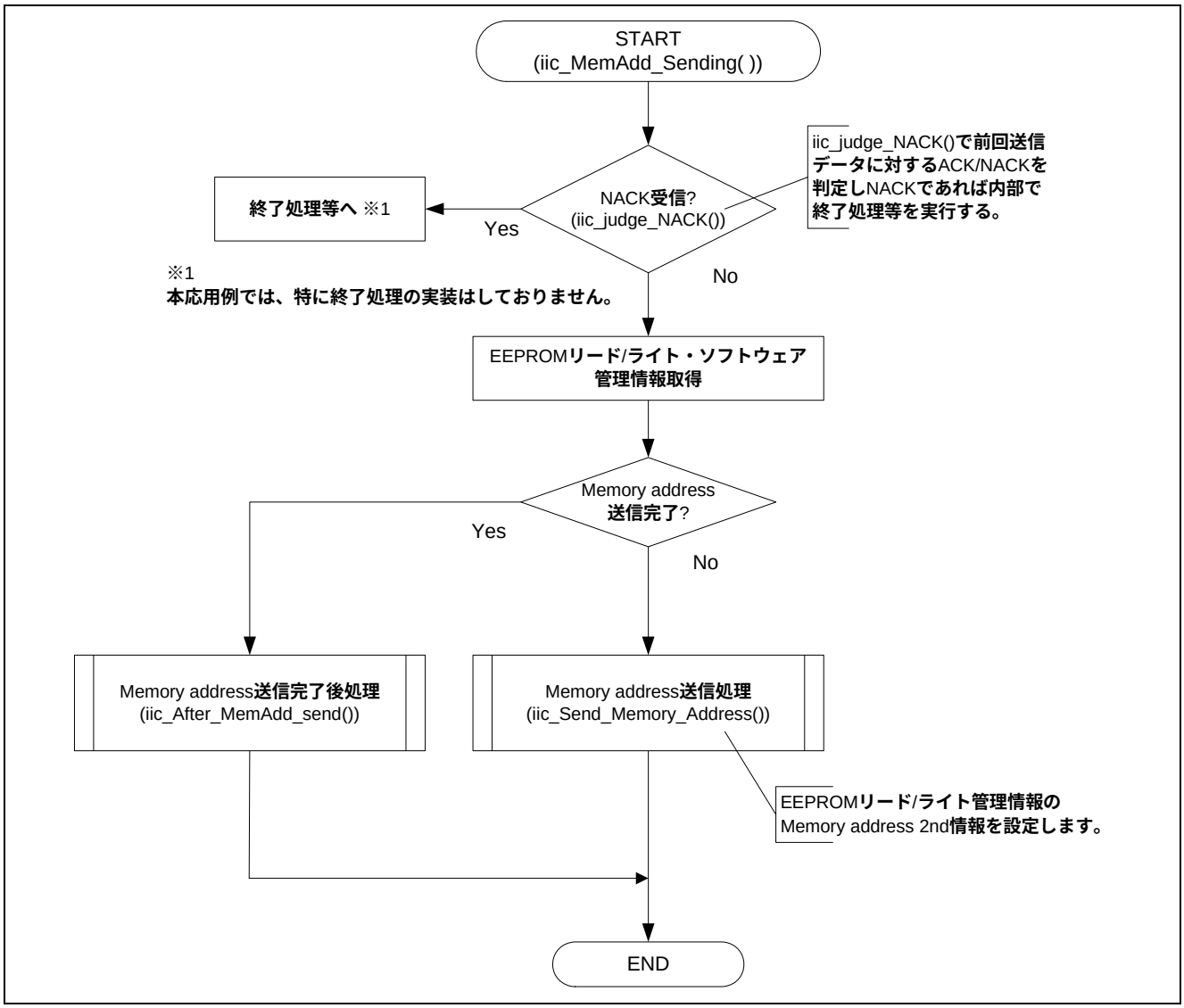


図 25　Memory address 送信中処理

4.16.9　WriteData 送信中処理

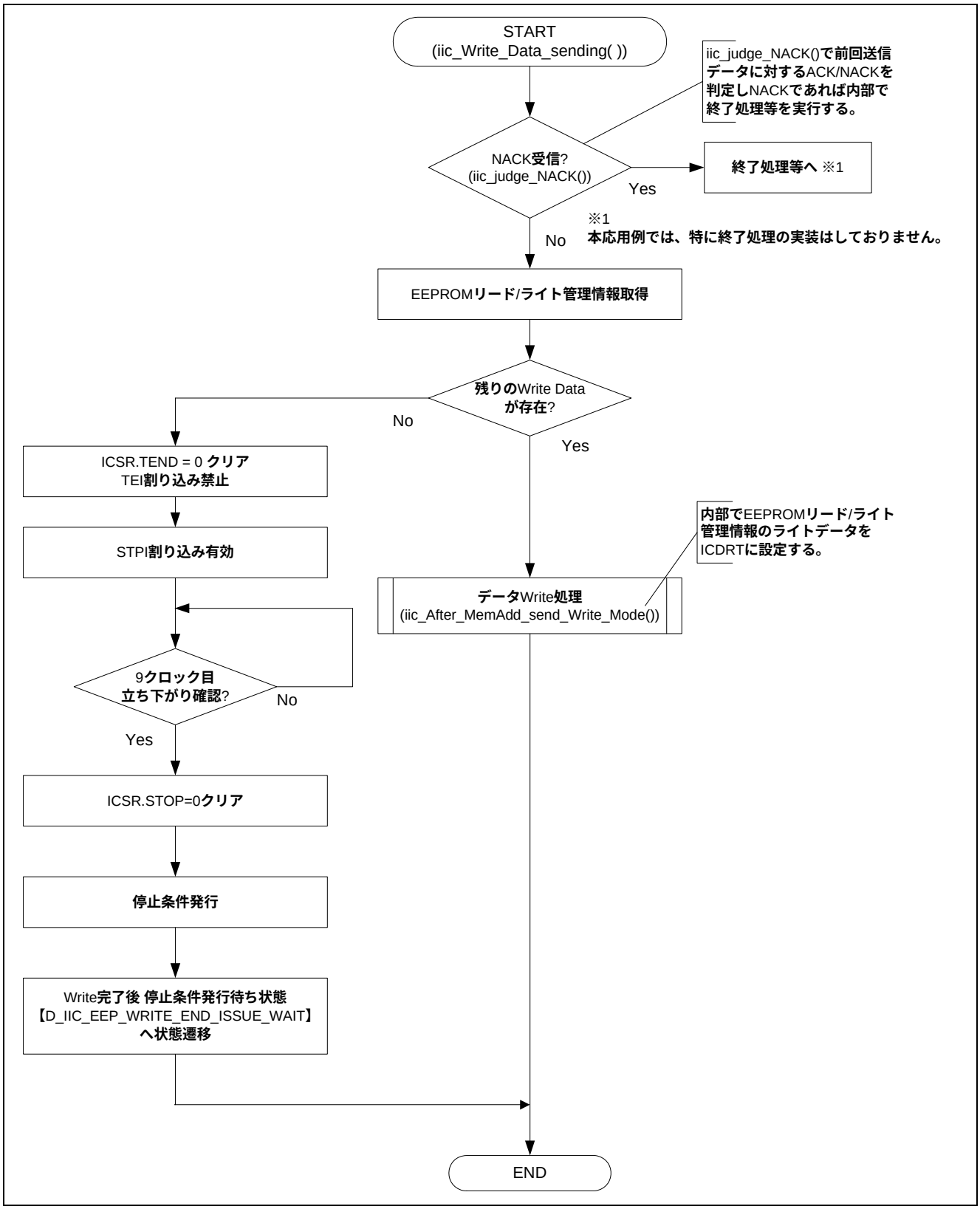


図 26　WriteData 送信中処理

4.16.10　ReadData 受信処理

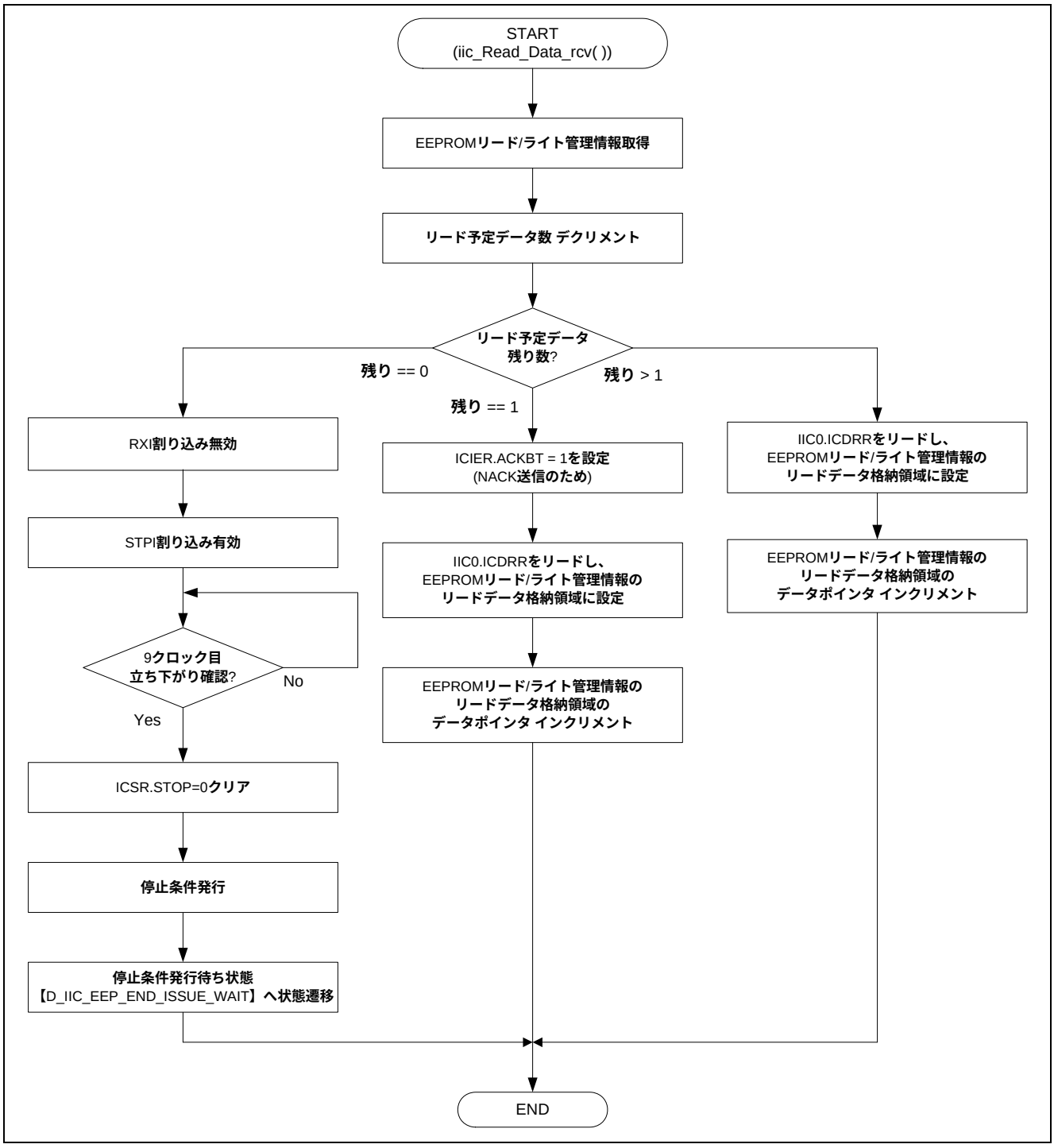


図 27　ReadData 受信処理

4.16.11　停止条件発行後 Write Polling 準備処理

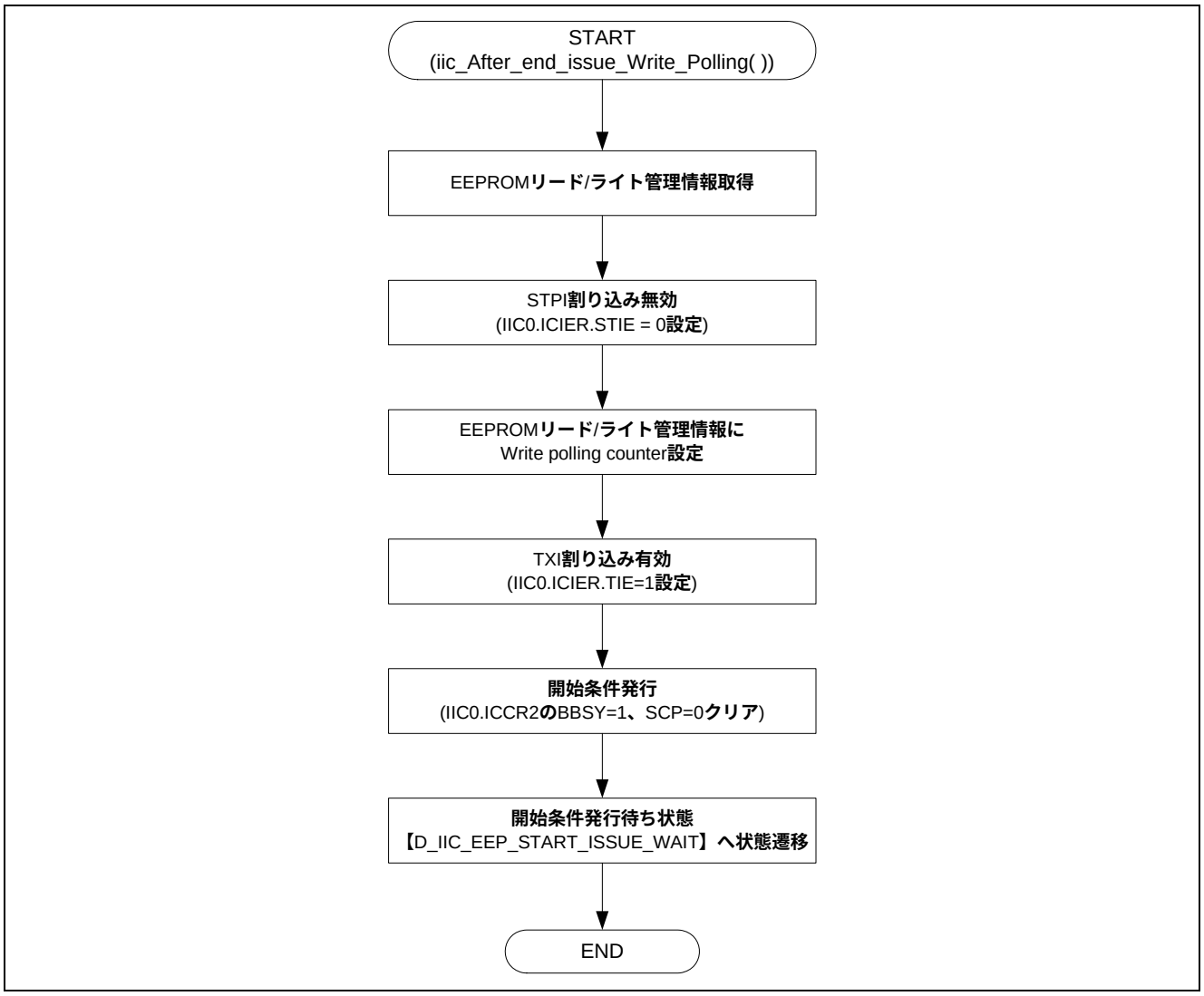


図 28　停止条件発行後 Write Polling 準備処理

4.16.12　WritePolling 処理

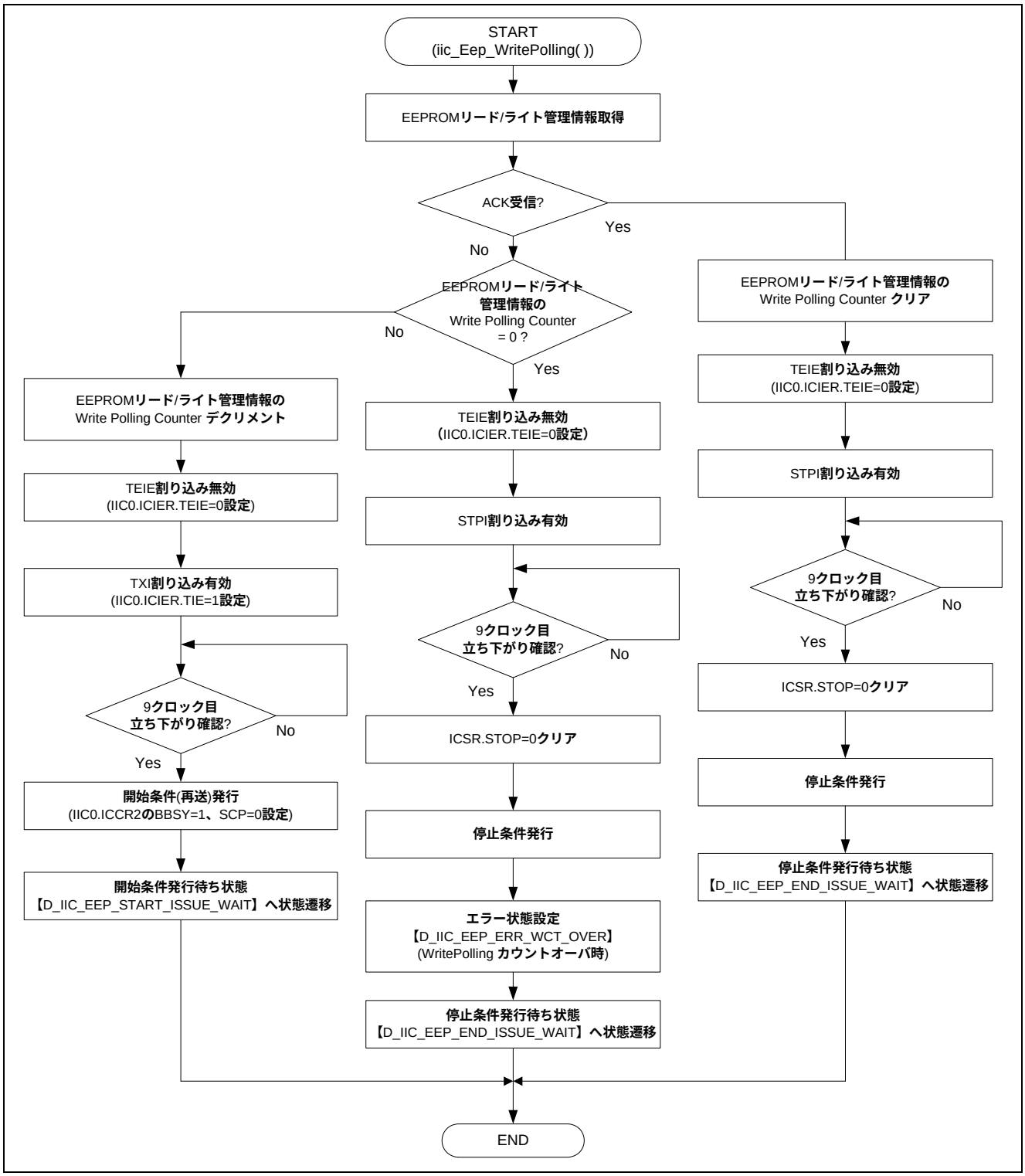


図 29　WritePolling 処理

4.16.13　停止条件発行後処理

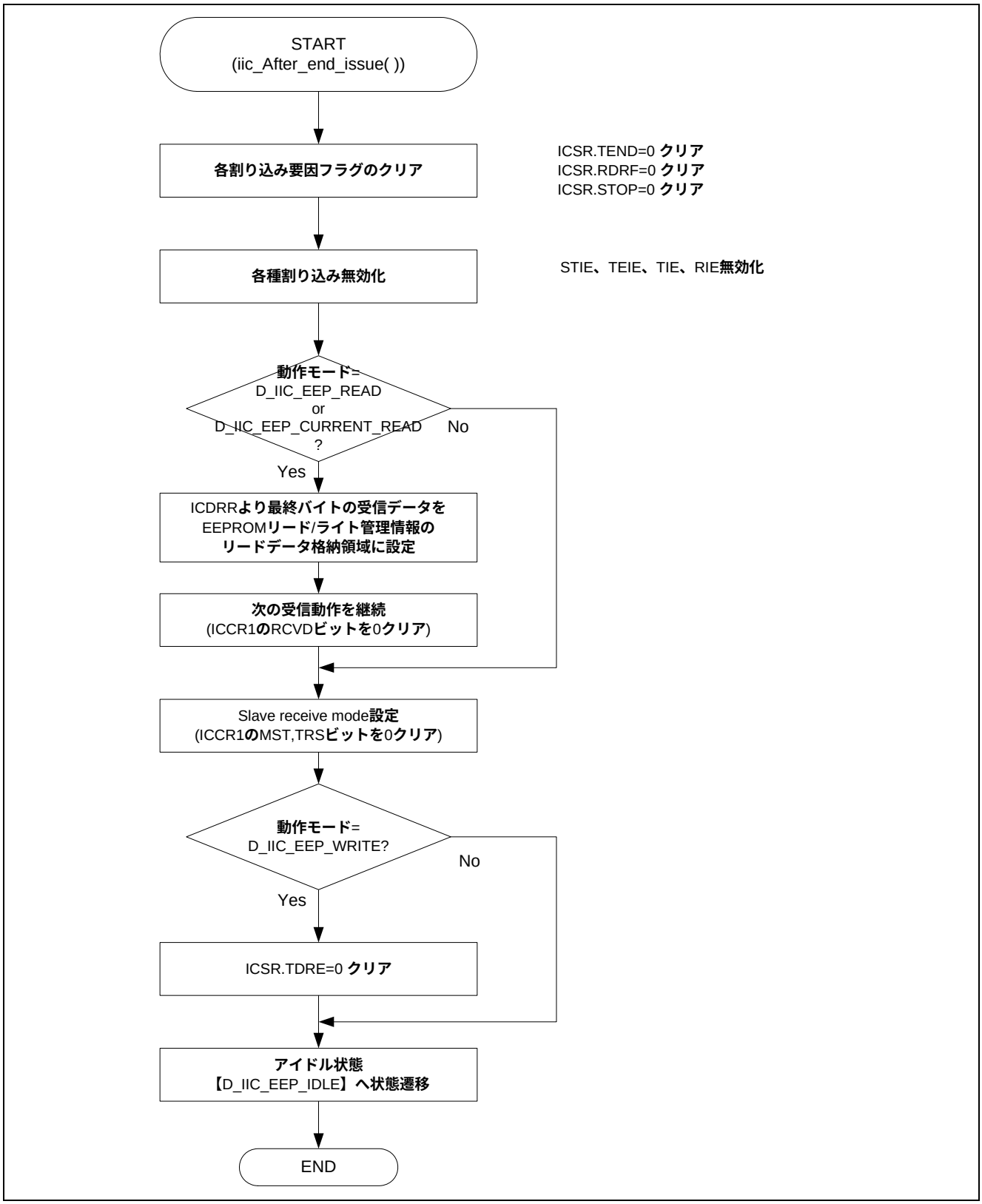


図 30　停止条件発行後処理

4.17　参考プログラムの処理フロー (内部関数)

本章では、状態遷移表に登録されている関数からコールされるに内部関数の中で、ポイントとなる関数のみ記載します。

4.17.1　Read モード遷移処理

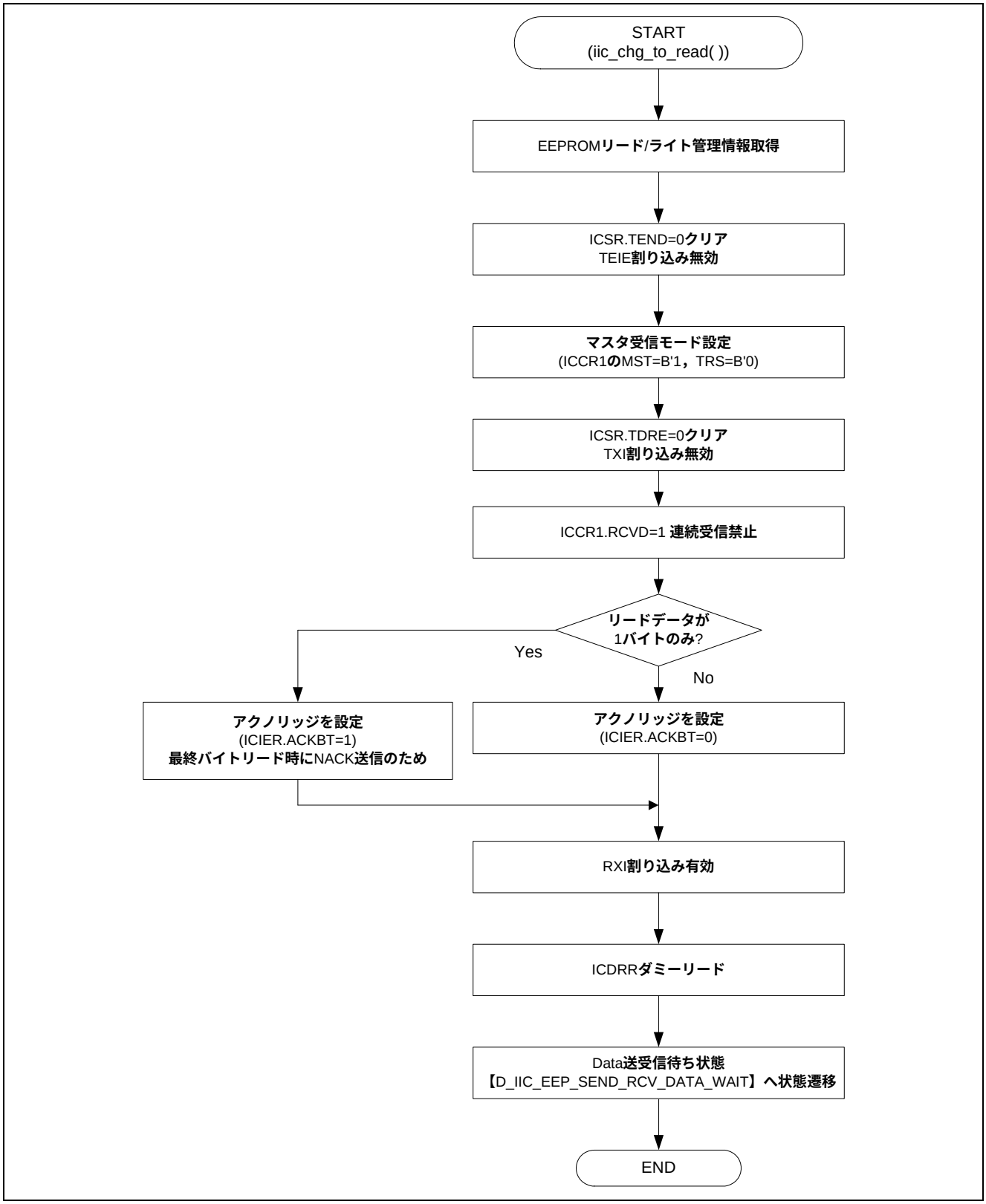


図 31　Read モード遷移処理

【呼び出しタイミング】

① D_IIC_EEP_CURRENT_READ モードで、初回 Device Address 送信後のマスタ送信からマスタ受信変更の際にコールされます。

② D_IIC_EEP_READ モードで、リード用 Device Address 送信後のマスタ送信からマスタ受信変更の際にコールされます。

4.17.2　Memory address 送信処理

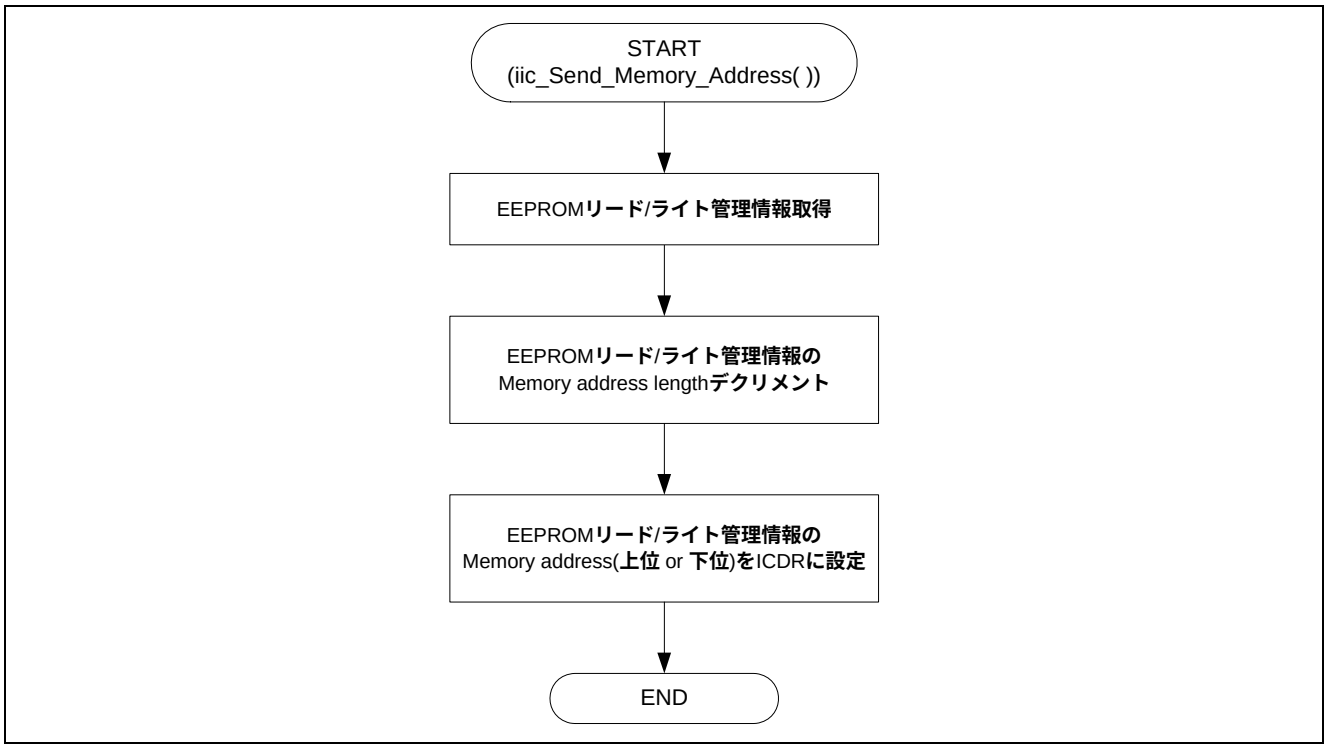


図 32　Memory address 送信処理

【呼び出しタイミング】

① D_IIC_EEP_READ or D_IIC_EEP_WRITE モードで、初回 Device Address 送信後の 1stMemory address 設定の際にコールされます。

② D_IIC_EEP_READ or D_IIC_EEP_WRITE モードで、1st Memory address 送信後の 2nd Memory address 設定の際にコールされます。

4.17.3　Memory address 送信完了後処理

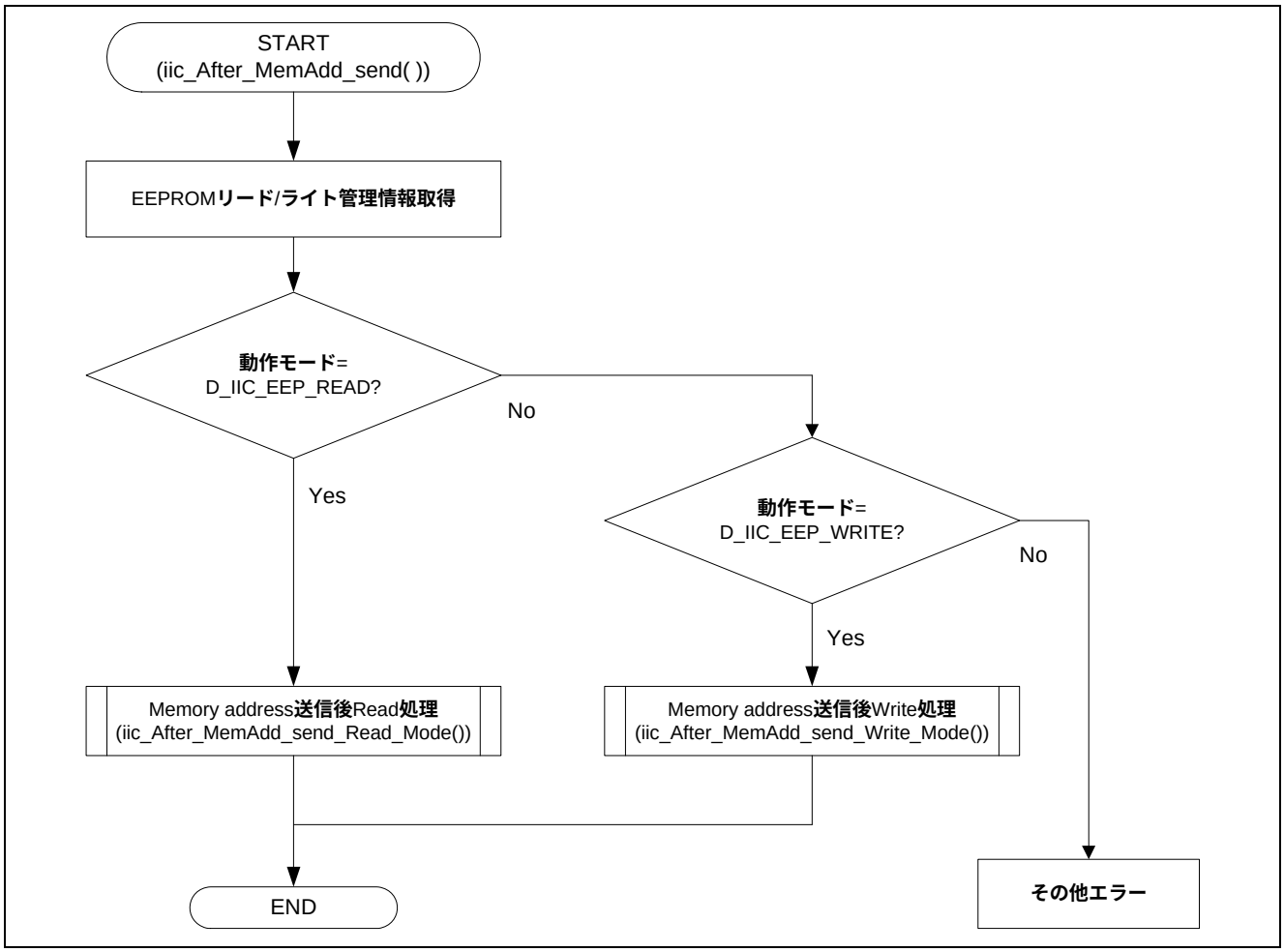


図 33　Memory address 送信完了後処理

【呼び出しタイミング】

① D_IIC_EEP_READ or D_IIC_EEP_WRITE モードで、Memory address 送信後の 1st データ設定 (D_IIC_EEP_ WRITE) or 1st データ取得 (D_IIC_EEP_READ) の際にコールされます。

4.17.4　Memory address 送信後 Read 処理

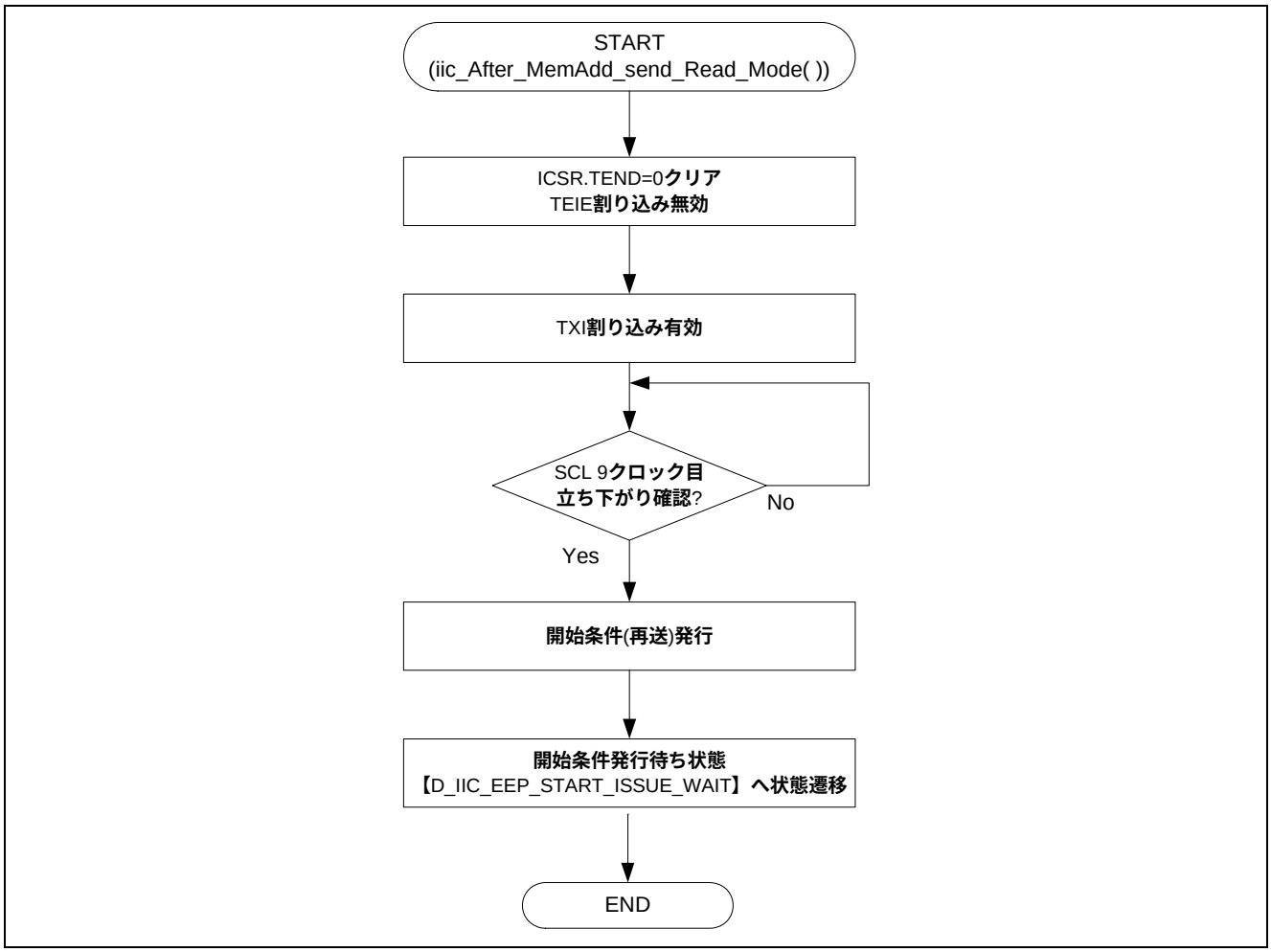


図 34　Memory address 送信後 Read 処理

【呼び出しタイミング】

① D_IIC_EEP_READ モードで、Memory address 送信後の Memory address 送信完了後処理 (iic_After_MemAdd_send()) からコールされます。

4.17.5　Memory address 送信後 Write 処理

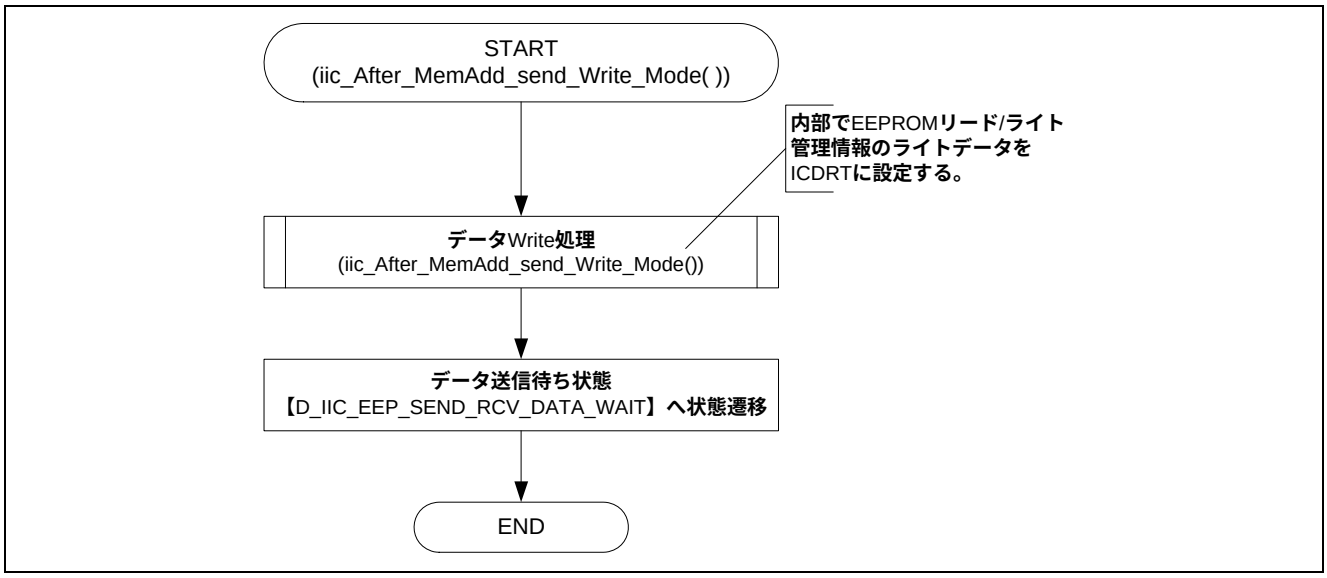


図 35　Memory address 送信後 Write 処理

【呼び出しタイミング】

① D_IIC_EEP_WRITE モードで、Memory address 送信後の Memory address 送信完了後処理 (iic_After_MemAdd_send()) からコールされます。

4.17.6　NACK 検出判定処理

スレーブデバイス (EEPROM) から NACK 受信した場合に行う処理については、本関数に実装することを想定しています。

本応用例では、処理実装はしておりませんので、システムに見合った処理を実装ください。

4.18　IIC 割り込み処理フロー

intprg.c に実装する割り込み関数について記載します。

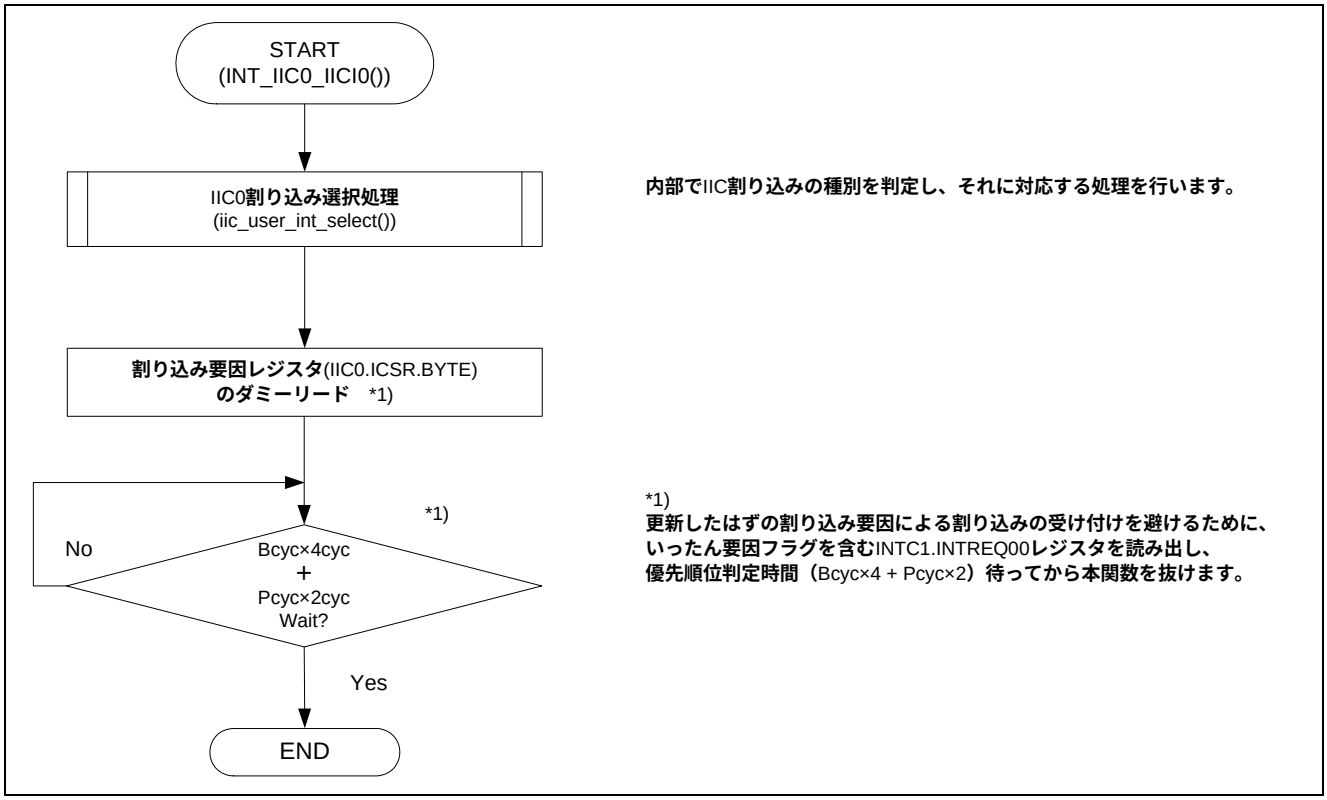


図 36　IIC 割り込み処理フロー

5. 参考プログラム例

(1) サンプルプログラムリスト "sh7730.c"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation. and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation. and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************/
28    /* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
29    /*""FILE COMMENT""*********** Technical reference data ***************
30    * System Name  : SH7730 Sample Program
31    * File Name    : sh7730.c
32    * Abstract     : SH7730 IIC 設定例 Sample Program（EEPROM 接続）
33    * Version      : Ver 1.00
34    * Device       : SH7730
35    * Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
36    *              : C/C++ Compiler Package for SuperH Family (V.9.03 release00)
37    * OS           : None
38    * H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
39    * Description  : SH7730 IIC 設定例（EEPROM 接続）のサンプルプログラムです。
40    *              :
41    * Operation    :
42    * Limitation   :
43    *              :
44    ********************************************************************
45    * History      : 18.Jun.2010 Ver. 1.00 First Release
46    *""FILE COMMENT END""*************************************************/
47
48    #include <machine.h>
49    #include <string.h>
50    #include "iodefine.h"
51    #include "iic_eeprom.h"
52
53    #define D_IIC_EEPROM_SEND_DATA_NUM  10      /* EEPROM へのデータ書き込みサイズ   */
54    #define D_IIC_EEPROM_READ_DATA_NUM  10      /* EEPROM からのデータ読み込みサイズ */
55
56
57    /* ライトデータ格納領域 */
58    unsigned char gEEPROM_Write_Data[]
59        ={0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a};
```

```
60
61    /* リードデータ格納領域 */
62    unsigned char gEEPROM_Read_Data[D_IIC_EEPROM_READ_DATA_NUM];
63
64    void main(void);
65
66    /*""FUNC COMMENT""************************************************
67    * ID                    :
68    * Outline               : メイン関数
69    * Include               :
70    * Declaration           : void main(void)
71    * Description           : サンプルプログラムのメイン関数です。
72    *                       : マスタデバイスを SH7730、
73    *                       : スレーブデバイスを EEPROM として、
74    *                       : シングルマスタで、EEPROM へ 10 バイト分の
75    *                       : データをライトします。
76    *                       : その後、先頭 9 バイト分については、
77    *                       : Random Read Operation と
78    *                       : Sequential Read Operation を組み合わせて
79    *                       : Memory address を指定してリードします。
80    *                       : 最終 1 バイトについては、
81    *                       : Current Address Read Operation で現在の
82    *                       : アドレス位置のデータをリードします。
83    *                       : 最後に EEPROM にライトしたデータと
84    *                       : EEPROM からリードしたデータが等しいことを
85    *                       : 確認します。
86    *                       :
87    * Argument              : none
88    * Return Value          : none
89    * Calling Functions     :
90    *""FUNC COMMENT END""*******************************************/
91    void main(void)
92    {
93        E_Iic_eep_mode          ret;
94        int                     i;
95        T_IIC_EEPROM_RW_INFO gWrite_info;       /* Write 用情報 */
96        T_IIC_EEPROM_RW_INFO gRead_info;        /* Random Read Operation と Sequential Read Operation
97    用情報 */
98        T_IIC_EEPROM_RW_INFO gCurrent_Read_info;    /* Current Address Read Operation 用情報 */
99        T_IIC_EEPROM_CONDITION  condition_info;    /* EEPROM アクセス状態用情報 */
100
101       memset(&gWrite_info, 0x00, sizeof(gWrite_info));
102       memset(&gRead_info, 0x00, sizeof(gRead_info));
103       memset(&gCurrent_Read_info, 0x00, sizeof(gCurrent_Read_info));
104       memset(gEEPROM_Read_Data, 0x00, sizeof(gEEPROM_Read_Data));
105       memset(&condition_info, 0x00, sizeof(condition_info));
106
107       /* IIC 初期化処理 */
108       iic_user_Init();
109
110       /* EEPROM ライト処理用パラメータ設定 */
111       gWrite_info.i_mode = D_IIC_EEP_WRITE;
112       gWrite_info.i_DevAdr = 0x00;
113       gWrite_info.i_RomAdr = 0x00000000;
114       gWrite_info.i_Len = D_IIC_EEPROM_SEND_DATA_NUM;
115       gWrite_info.i_pBuf = gEEPROM_Write_Data;
116
117       /* EEPROM Random Read Operation と Sequential Read Operation を
118          組み合わせた処理用パラメータ設定 */
119       gRead_info.i_mode = D_IIC_EEP_READ;
120       gRead_info.i_DevAdr = 0x00;
121       gRead_info.i_RomAdr = 0x00000000;
122       gRead_info.i_Len = D_IIC_EEPROM_READ_DATA_NUM - 1;
123       gRead_info.i_pBuf = gEEPROM_Read_Data;
```

```
124
125       /* EEPROM Current Address Read Operation 用パラメータ設定 */
126       gCurrent_Read_info.i_mode = D_IIC_EEP_CURRENT_READ;
127       gCurrent_Read_info.i_DevAdr = 0x00;
128       gCurrent_Read_info.i_Len = 1;
129       gCurrent_Read_info.i_pBuf = &gEEPROM_Read_Data[9];
130
131       do
132       {
133         /* EEPROM ライト処理 */
134         /* アドレス 0x0000 から 10 バイト分ライトします。 */
135         ret = iic_user_EepRomRW(&gWrite_info);
136
137         if(D_IIC_EEP_OK != ret)
138         {
139             break;
140         }
141
142         /* EEPROM ライト完了まで待ち */
143         do
144         {
145             ret = iic_user_Chk_EepRom(&condition_info);
146
147         }while(ret != D_IIC_EEP_OK);
148
149         /* EEPROM Random Read Operation と Sequential Read Operation 組み合わせた処理 */
150         /* アドレス 0x0000 から 9 バイト分リードします。 */
151         ret = iic_user_EepRomRW(&gRead_info);
152
153         if(D_IIC_EEP_OK != ret)
154         {
155             break;
156         }
157
158         /* EEPROM Random Read Operation と Sequential Read Operation
159            組み合わせた処理完了まで待ち */
160         do
161         {
162             ret = iic_user_Chk_EepRom(&condition_info);
163
164         }while(ret != D_IIC_EEP_OK);
165
166         /* EEPROM Current Address Read Operation */
167         /* 現在のアドレス(0x0009)を 1 バイト分リードします。 */
168         ret = iic_user_EepRomRW(&gCurrent_Read_info);
169
170         if(D_IIC_EEP_OK != ret)
171         {
172             break;
173         }
174
175         /* EEPROM Current Address Read Operation 完了まで待ち */
176         do
177         {
178             ret = iic_user_Chk_EepRom(&condition_info);
179
180         }while(ret != D_IIC_EEP_OK);
181
182       }while(0);
183
184       /* EEPROM へのライトデータと EEPROM からのリードデータが等しいことを確認 */
185       /* 一致しない場合は無限ループに遷移します。 */
186       for(i = 0 ; i < D_IIC_EEPROM_SEND_DATA_NUM; i++)
187       {
```

```
188         if(gEEPROM_Write_Data[i] != gEEPROM_Read_Data[i])
189         {
190             while(1)
191             {
192             }
193         }
194       }
195
196       /* 上記無限ループに入らないことにより、データが一致していることを確認します。 */
197       while(1)
198       {
199       }
200
201   }
202
203   /* End of File */
```

(2) サンプルプログラムリスト "iic_eeprom_use_if.c"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation. and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation. and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************/
28    /* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
29    /*""FILE COMMENT""************ Technical reference data ****************
30    * System Name  : SH7730 Sample Program
31    * File Name    : iic_eeprom_use_if.c
32    * Abstract     : SH7730 IIC 設定例 Sample Program（EEPROM 接続）
33    * Version      : Ver 1.00
34    * Device       : SH7730
35    * Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
36    *              : C/C++ Compiler Package for SuperH Family (V.9.03 release00)
37    * OS           : None
38    * H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
39    * Description  : SH7730 IIC 設定例（EEPROM 接続）のサンプルプログラムです。
40    *              :
41    * Operation    :
42    * Limitation   :
43    *              :
44    ********************************************************************
45    * History      : 18.Jun.2010 Ver. 1.00 First Release
46    *""FILE COMMENT END""*********************************************/
47    #include <machine.h>
48    #include <stdio.h>
49    #include "iodefine.h"
50    #include "iic_eeprom.h"
51
52
53    /*""FUNC COMMENT""*************************************************
54    * ID                    :
55    * Outline               : IIC0 初期化処理
56    * Include               :
57    * Declaration           : E_Iic_eep_Ret iic_user_Init(void)
58    * Description           : IIC 初期化処理を行います。
59    *                       :
60    * Argument              : none
61    * Return Value          : E_Iic_eep_Ret 型参照
62    * Calling Functions     :
63    *""FUNC COMMENT END""*******************************************/
64    E_Iic_eep_Ret iic_user_Init(void)
65    {
```

```
66        E_Iic_eep_Ret        ret;
67
68        ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INIT, NULL);
69
70        return ret;
71    }
72
73    /*""FUNC COMMENT""*************************************************
74    * ID                   :
75    * Outline               : EEPROM リード/ライト処理
76    * Include               :
77    * Declaration            : E_Iic_eep_Ret iic_user_EepRomRW
78    *                       :   (T_IIC_EEPROM_RW_INFO *i_RW_Info)
79    * Description            : EEPROM リード/ライト処理を行います。
80    *                       :
81    * Argument              : T_IIC_EEPROM_RW_INFO *i_RW_Info
82    *                       : パラメータに以下を設定します。
83    *                       : ・R/W mode
84    *                       : ・Device address
85    *                       : ・Memory address
86    *                       : ・Transfer/Receive data length
87    *                       : ・Transfer/Receive data buffer pointer
88    * Return Value           : D_IIC_EEP_OK     : Success
89    *                       : D_IIC_EEP_BUS_BUSY : Bus Busy
90    * Calling Functions       :
91    *""FUNC COMMENT END""*********************************************/
92    E_Iic_eep_Ret iic_user_EepRomRW(T_IIC_EEPROM_RW_INFO *i_RW_Info)
93    {
94        E_Iic_eep_Ret        ret;
95        void             *pdata;
96
97        pdata = (void *)i_RW_Info;
98
99        ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_RW_START, pdata);
100
101       return ret;
102
103   }
104
105   /*""FUNC COMMENT""*************************************************
106   * ID                   :
107   * Outline               : EEPROM 状態取得処理
108   * Include               :
109   * Declaration            : E_Iic_eep_Ret iic_user_Chk_EepRom(
110   *                       :  T_IIC_EEPROM_CONDITION *o_condition_info)
111   * Description            : EEPROM リード/ライト状態取得を行います。
112   *                       :
113   * Argument              : T_IIC_EEPROM_CONDITION *o_condition_info
114   *                       : EEPROM 状態
115   * Return Value           : E_Iic_eep_Ret
116   *                       : D_IIC_EEP_OK      : Success
117   *                       : D_IIC_EEP_NG       : Error
118   *                       : D_IIC_EEP_READING  : READING
119   *                       : D_IIC_EEP_WRITING  : WRITING
120   * Calling Functions       :
121   *""FUNC COMMENT END""*********************************************/
122   E_Iic_eep_Ret iic_user_Chk_EepRom
123   (
124       T_IIC_EEPROM_CONDITION *o_condition_info
125   )
126   {
127       E_Iic_eep_Ret ret = D_IIC_EEP_OK;
128       void             *pdata;
129
130       pdata = (void *)o_condition_info;
131
132       ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_CHECK, pdata);
```

```
133
134       return ret;
135
136   }
137
138   /*""FUNC COMMENT""************************************************
139   * ID                    :
140   * Outline               : IIC0 割り込み選択処理
141   * Include               :
142   * Declaration           : E_Iic_eep_Ret iic_user_int_select(void)
143   * Description           : IIC0 の、どの割り込みが発生したか判断し
144   *                       : 対応する割り込み処理をコールします。
145   * Argument              : none
146   * Return Value          : E_Iic_eep_Ret 型参照
147   * Calling Functions      :
148   *""FUNC COMMENT END""*******************************************/
149   E_Iic_eep_Ret iic_user_int_select(void)
150   {
151       E_Iic_eep_Ret        ret = D_IIC_EEP_OK;
152
153       /* 割り込みにより判定 */
154       if(IIC0.ICSR.BIT.ALOVE == 1 &&
155          IIC0.ICIER.BIT.NAKIE == 1)           /* NAKI 割り込み */
156       {
157         ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_NAKI, NULL);
158       }
159
160       else if(IIC0.ICSR.BIT.TEND == 1 &&
161                IIC0.ICIER.BIT.TEIE == 1)     /* TEI 割り込み */
162       {
163         ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_TEI, NULL);
164       }
165
166       else if(IIC0.ICSR.BIT.TDRE == 1 &&
167            IIC0.ICIER.BIT.TIE == 1)      /* TXI 割り込み */
168       {
169         ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_TXI, NULL);
170       }
171
172       else if(IIC0.ICSR.BIT.RDRF == 1 &&
173            IIC0.ICIER.BIT.RIE == 1)      /* RXI 割り込み */
174       {
175         ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_RXI, NULL);
176       }
177
178       else if(IIC0.ICSR.BIT.STOP == 1 &&
179            IIC0.ICIER.BIT.STIE == 1)     /* STPI 割り込み */
180       {
181         ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_STPI, NULL);
182       }
183
184       else
185       {
186       }
187
188       return ret;
189
190   }
191
192   /* End of File */
```

(3) サンプルプログラムリスト "iic_eeprom.c"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation. and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation. and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************/
28    /* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
29    /*""FILE COMMENT""*********** Technical reference data ***************
30    * System Name  : SH7730 Sample Program
31    * File Name    : iic_eeprom.c
32    * Abstract     : SH7730 IIC 設定例 Sample Program（EEPROM 接続）
33    * Version      : Ver 1.00
34    * Device       : SH7730
35    * Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
36    *              : C/C++ Compiler Package for SuperH Family (V.9.03 release00)
37    * OS           : None
38    * H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
39    * Description  : SH7730 IIC 設定例（EEPROM 接続）のサンプルプログラムです。
40    *              :
41    * Operation    :
42    * Limitation   :
43    *              :
44    ***********************************************************************
45    * History      : 18.Jun.2010 Ver. 1.00 First Release
46    *""FILE COMMENT END""*************************************************/
47    #include <machine.h>
48    #include <stdio.h>
49    #include "iodefine.h"
50    #include "iic_eeprom.h"
51
52    E_Iic_eep_condition      gIic_Eep_Condition;           /* IIC_EEPROM 状態 */
53    E_Iic_eep_condition      gIic_Eep_Before_Condition; /* IIC_EEPROM 前状態 */
54    E_Iic_eep_err_condition  gIic_Eep_Err_Condition;       /* IIC_EEPROM 状態エラー */
55    T_IIC_EEPROM_RW_MANAGE   gIic_Eep_Manage_Info;     /* EEPROM リード/ライト管理情報 */
56
57    /* 状態遷移表定義関数宣言 */
58    static E_Iic_eep_Ret iic_Init(void *info);
59    static E_Iic_eep_Ret iic_EepRomRW(void *info);
60    static E_Iic_eep_Ret iic_Chk_EepRom(void *info);
61    static E_Iic_eep_Ret iic_AL_generate(void *info);
62    static E_Iic_eep_Ret iic_After_start_issue(void *info);
63    static E_Iic_eep_Ret iic_Send_After_Dev_Address(void* info);
64    static E_Iic_eep_Ret iic_After_Re_DevAdd_send(void *info);
```

```
65    static E_Iic_eep_Ret iic_MemAdd_Sending(void *info);
66    static E_Iic_eep_Ret iic_Write_Data_sending(void *info);
67    static E_Iic_eep_Ret iic_Read_Data_rcv(void *info);
68    static E_Iic_eep_Ret iic_After_end_issue_Write_Polling(void *info);
69    static E_Iic_eep_Ret iic_Eep_WritePolling(void *info);
70    static E_Iic_eep_Ret iic_After_end_issue(void *info);
71
72
73    /* 状態遷移表 */
74    static const T_IIC_EEPROM_MTX_TBL
75    g_Iic_Eep_mtx_tbl[D_IIC_EEP_CONDITION_MAX][D_IIC_EEP_EVENT_MAX] =
76    {
77        /* 未初期化状態 */
78        {
79          { D_IIC_EEP_EV_INIT           ,   iic_Init           } , /* iic_user_Init()コール時
80    */
81          { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
82    */
83          { D_IIC_EEP_EV_CHECK      ,   NULL               } , /* iic_user_Chk_EepRom()コール時  */
84          { D_IIC_EEP_EV_INT_NAKI       ,   NULL               } , /* NAKI 割り込み時               */
85          { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
86          { D_IIC_EEP_EV_INT_TEI    ,   NULL               } , /* TEI 割り込み時                */
87          { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
88          { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
89        },
90
91        /* アイドル状態 */
92        {
93          { D_IIC_EEP_EV_INIT           ,   iic_Init           } , /* iic_user_Init()コール時
94    */
95          { D_IIC_EEP_EV_RW_START       ,   iic_EepRomRW       } , /* iic_user_EepRomRW()コール時
96    */
97          { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
98          { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
99          { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
100         { D_IIC_EEP_EV_INT_TEI    ,   NULL               } , /* TEI 割り込み時                */
101         { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
102         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
103       },
104
105       /* 開始条件発行待ち状態  */
106       {
107         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
108   */
109         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
110   */
111         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
112         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
113         { D_IIC_EEP_EV_INT_TXI    ,   iic_After_start_issue} ,   /* TXI 割り込み時             */
114         { D_IIC_EEP_EV_INT_TEI    ,   NULL               } , /* TEI 割り込み時                */
115         { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
116         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
117       },
118
119       /* Device Address 送信待ち状態  */
120       {
121         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
122   */
123         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
124   */
125         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
126         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
127         { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
128         { D_IIC_EEP_EV_INT_TEI    ,   iic_Send_After_Dev_Address} ,  /* TEI 割り込み時          */
129         { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
130         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
```

```
131       },
132
133       /* Device Address 再送信待ち状態  */
134       {
135         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
136   */
137         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
138   */
139         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
140         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
141         { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
142         { D_IIC_EEP_EV_INT_TEI    ,   iic_After_Re_DevAdd_send} ,    /* TEI 割り込み時            */
143         { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
144         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
145       },
146
147       /* Memory address 送信待ち状態  */
148       {
149         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
150   */
151         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
152   */
153         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
154         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
155         { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
156         { D_IIC_EEP_EV_INT_TEI    ,   iic_MemAdd_Sending } , /* TEI 割り込み時                */
157         { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
158         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
159       },
160
161       /* データ送受信待ち状態  */
162       {
163         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
164   */
165         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
166   */
167         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
168         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
169         { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
170         { D_IIC_EEP_EV_INT_TEI    ,   iic_Write_Data_sending } , /* TEI 割り込み時             */
171         { D_IIC_EEP_EV_INT_RXI    ,   iic_Read_Data_rcv  } , /* RXI 割り込み時                */
172         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
173       },
174
175       /* Write 完了後 停止条件発行待ち状態  */
176       {
177         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
178   */
179         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
180   */
181         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
182         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時               */
183         { D_IIC_EEP_EV_INT_TXI    ,   NULL               } , /* TXI 割り込み時                */
184         { D_IIC_EEP_EV_INT_TEI    ,   NULL               } , /* TEI 割り込み時                */
185         { D_IIC_EEP_EV_INT_RXI    ,   NULL               } , /* RXI 割り込み時                */
186         { D_IIC_EEP_EV_INT_STPI       ,   iic_After_end_issue_Write_Polling }   /* STPI 割り込み時
187   */
188       },
189
190       /* WritePolling 待ち状態  */
191       {
192         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
193   */
194         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
195   */
196         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
```

```
197         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時                */
198         { D_IIC_EEP_EV_INT_TXI     ,   NULL               } , /* TXI 割り込み時               */
199         { D_IIC_EEP_EV_INT_TEI     ,   iic_Eep_WritePolling} ,    /* TEI 割り込み時               */
200         { D_IIC_EEP_EV_INT_RXI     ,   NULL               } , /* RXI 割り込み時               */
201         { D_IIC_EEP_EV_INT_STPI       ,   NULL               }   /* STPI 割り込み時               */
202       },
203
204       /* 停止条件発行待ち状態  */
205       {
206         { D_IIC_EEP_EV_INIT           ,   NULL               } , /* iic_user_Init()コール時
207   */
208         { D_IIC_EEP_EV_RW_START       ,   NULL               } , /* iic_user_EepRomRW()コール時
209   */
210         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom     } , /* iic_user_Chk_EepRom()コール時  */
211         { D_IIC_EEP_EV_INT_NAKI       ,   iic_AL_generate    } , /* NAKI 割り込み時                */
212         { D_IIC_EEP_EV_INT_TXI     ,   NULL               } , /* TXI 割り込み時               */
213         { D_IIC_EEP_EV_INT_TEI     ,   NULL               } , /* TEI 割り込み時               */
214         { D_IIC_EEP_EV_INT_RXI     ,   NULL               } , /* RXI 割り込み時               */
215         { D_IIC_EEP_EV_INT_STPI       ,   iic_After_end_issue}   /* STPI 割り込み時               */
216       }
217   };
218
219   /* 内部関数宣言 */
220   static E_Iic_eep_Ret iic_Chk_Bus_SCL_SDA(void);
221   static E_Iic_eep_Ret iic_chg_to_read(void);
222   static E_Iic_eep_Ret iic_Send_Memory_Address(void);
223   static void iic_After_MemAdd_send(void);
224   static void iic_After_MemAdd_send_Read_Mode(void);
225   static void iic_After_MemAdd_send_Write_Mode(void);
226   static void iic_send_Write_Data(void);
227   static E_Iic_eep_Nack iic_judge_NACK(void);
228   static void iic_chk_scl_down(void);
229   static void set_internal_info_init(void);
230
231   static void set_Iic_Eep_Condition(E_Iic_eep_condition i_condition);
232   static E_Iic_eep_condition get_Iic_Eep_Condition(void);
233   static void set_Iic_Eep_Before_Condition(E_Iic_eep_condition i_condition);
234   static E_Iic_eep_condition get_Iic_Eep_Before_Condition(void);
235   static void set_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *i_info);
236   static void get_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *o_info);
237   static void set_Iic_Eep_Err_Condition(E_Iic_eep_err_condition i_err);
238   static E_Iic_eep_err_condition get_Iic_Eep_Err_Condition(void);
239   static E_Iic_eep_mode get_Iic_Eep_Mode(void);
240
241   /*""FUNC COMMENT""************************************************
242   * ID                    :
243   * Outline               : 状態遷移処理
244   * Include               :
245   * Declaration           : E_Iic_eep_Ret iic_mtx_executeFuncTable(void)
246   * Description           : 状態遷移処理を行います。
247   *                       :
248   * Argument              : E_Iic_eep_event     event
249   *                       : void              *info
250   * Return Value          : E_Iic_eep_Ret 型参照
251   * Calling Functions     :
252   *""FUNC COMMENT END""*******************************************/
253   static E_Iic_eep_Ret  iic_mtx_executeFuncTable(
254      E_Iic_eep_event     event,                  /* イベント */
255      void             *info                 /* 情報 */
256                                )
257   {
258       E_Iic_eep_Ret       ret = D_IIC_EEP_OK ;       /* 戻り値  */
259       E_Iic_eep_condition     NowCondition;             /* 内部状態 */
260       E_Iic_eep_Ret           (*pFunc)( void *);        /* 実行処理用 */
261       unsigned long           IntMask;                  /* マスク用 */
262
```

```
263       IntMask = get_imask();              /* 割り込みマスク取得 */
264
265       if(IntMask < D_IIC_EEP_INT_LEVEL)
266       {
267         set_imask(D_IIC_EEP_INT_LEVEL);       /* 割り込みマスク設定 */
268       }
269
270       /* 現状態取得 */
271       NowCondition = get_Iic_Eep_Condition();
272
273       /* テーブル検索処理 */
274       if(( NowCondition < D_IIC_EEP_CONDITION_MAX ) &&
275          ( event < D_IIC_EEP_EVENT_MAX ))
276       {
277         /* テーブルの対応する処理を行う */
278         if( g_Iic_Eep_mtx_tbl[NowCondition][event].proc != NULL )
279         {
280             /* 該当イベント個別処理を実行する */
281             pFunc = g_Iic_Eep_mtx_tbl[NowCondition][event].proc;
282             ret = (*pFunc)(info);
283
284         }
285         /* NULL 定義時無処理 */
286         else
287         {
288             ret = D_IIC_EEP_NOP;
289         }
290       }
291
292       set_imask(IntMask);                 /* 割り込みマスク復帰 */
293
294      return ret;
295   }
296
297   /*""FUNC COMMENT""*****************************************************
298   * ID                    :
299   * Outline               : IIC 初期化処理
300   * Include               :
301   * Declaration           : E_Iic_eep_Ret iic_Init(void *info)
302   * Description           : iic 初期化処理を行います。
303   *                       :
304   * Argument              : void              *info
305   * Return Value          : D_IIC_EEP_OK
306   * Calling Functions     :
307   *""FUNC COMMENT END""*************************************************/
308   static E_Iic_eep_Ret iic_Init(void *info)
309   {
310       unsigned long       dummy;
311
312       set_internal_info_init();           /* 内部情報初期化 */
313
314       /* ==== モジュールストップレジスタ 1 設定 ==== */
315       LOWP.MSTPCR1 &= ~0x00000200;    /* IIC module clock supply */
316       dummy = LOWP.MSTPCR1;               /* 設定反映確認のためダミリード */
317
318       /* ==== PFC の設定 ==== */
319       PFC.PLCR.BIT.PL0MD = 0;             /* SDA0 選択 */
320       PFC.PLCR.BIT.PL1MD = 0;             /* SCL0 選択 */
321
322       /* ==== 割り込み優先レベル設定（優先レベル 1）==== */
323       INTC0.IPRH = INTC0.IPRH | 0x0001;
324
325       /* EEPROM アクセス状態設定 */
326       set_Iic_Eep_Condition(D_IIC_EEP_IDLE);  /* アイドル状態 */
327
328       return D_IIC_EEP_OK;
```

```
329   }
330
331   /*""FUNC COMMENT""************************************************
332   * ID                    :
333   * Outline               : EEPROM リード/ライト処理
334   * Include               :
335   * Declaration           : E_Iic_eep_Ret iic_EepRomRW(void *info)
336   * Description           : EEPROM リード/ライト処理を行います。
337   *                       :
338   * Argument              : void *info
339   *                       : パラメータに以下を設定します。
340   *                       : ・R/W mode
341   *                       : ・Device address
342   *                       : ・Memory address
343   *                       : ・Transfer/Receive data length
344   *                       : ・Transfer/Receive data buffer pointer
345   * Return Value          : D_IIC_EEP_OK     : Success
346   *                       : D_IIC_EEP_NG     : Error
347   *                       : D_IIC_EEP_BUS_BUSY : Bus Busy
348   * Calling Functions     :
349   *""FUNC COMMENT END""*******************************************/
350   static E_Iic_eep_Ret iic_EepRomRW(void *info)
351   {
352       unsigned char           work;
353       E_Iic_eep_Ret           ret;
354       T_IIC_EEPROM_RW_INFO *i_RW_Info;
355       T_IIC_EEPROM_RW_MANAGE  manage_info;
356
357       if(info == NULL)
358       {
359         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);  /* その他エラー */
360         return D_IIC_EEP_NG;
361       }
362
363       /* Master transfer mode setting */
364       IIC0.ICCR1 = 0xf6;                  /* ICE  : 転送動作可能
365                                              RCVD : 次の受信動作を禁止
366                                              MST  : B'1 マスタ送信モード選択
367                                              TRS  : B'1
368                                              CKS  : Pφ/100
369                                             */
370
371       /* BBSY フラグ取得 */
372       work = IIC0.ICCR2;
373       work = work >> 7;
374
375       /* Bus busy check */
376       if(work == 1)
377       {
378         return D_IIC_EEP_BUS_BUSY;
379       }
380
381       i_RW_Info = (T_IIC_EEPROM_RW_INFO *)info;
382
383       /* 内部情報初期化 */
384       memset(&manage_info, 0, sizeof(manage_info));
385       set_Iic_Eep_Before_Condition(D_IIC_EEP_NO_INIT);   /* 前状態初期化   */
386       set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_NO);       /* エラーなし状態 */
387       set_Iic_Eep_Manage_Info(&manage_info);
388
389       /* mode set */
390       manage_info.i_mode = i_RW_Info->i_mode;
391
392       /* Device address set */
393       manage_info.i_DevAdr = (i_RW_Info->i_DevAdr & 0x0f);
394       manage_info.i_DevAdr |= D_IIC_DEV_CODE;
```

```
395
396       /* モードにより場合わけ */
397       switch(manage_info.i_mode)
398       {
399         case D_IIC_EEP_WRITE:
400         case D_IIC_EEP_READ:
401
402             manage_info.i_DevAdr &= ~D_IIC_R_CODE;        /* Write code set */
403
404             break;
405
406         case D_IIC_EEP_CURRENT_READ:
407
408             manage_info.i_DevAdr |= D_IIC_R_CODE;     /* Read code set */
409
410             break;
411
412         default:
413             set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);  /* その他エラー */
414             return D_IIC_EEP_NG;
415       }
416
417       /* Memory address info set */
418       manage_info.i_MemAdrLen = D_IIC_EEP_MEMADR_SIZE;
419       manage_info.i_MemAdrBuf[1]  = (unsigned char)((i_RW_Info->i_RomAdr >> 8) & 0x000000FF);
420       manage_info.i_MemAdrBuf[0]  = (unsigned char)(i_RW_Info->i_RomAdr & 0x000000FF);
421
422       /* Read/Write data size set */
423       manage_info.i_RW_Data_info.i_DataLen = i_RW_Info->i_Len;
424       manage_info.i_RW_Data_info.i_DataBuf = i_RW_Info->i_pBuf;
425
426       /* I2C バス状態を確認（スレーブデバイスがラインを解放するまで待つ）*/
427       iic_Chk_Bus_SCL_SDA();
428
429       /*    上記処理結果で、I2C バス状態が I2C 解放であれば、以降、念のための IIC リセット処理 */
430       /*    上記処理結果で、I2C バス状態が I2C 非解放（ここではスレーブデバイスの暴走を意図する）であれば、
431         以降、ハングアップを回避することを期待するための IIC リセット処理 */
432
433       /* ==== IIC.ICCR2 の設定 ==== */
434       IIC0.ICCR2 |= 0x02;                 /* IIC コントロール部リセット */
435       work = IIC0.ICCR2;
436       IIC0.ICCR2 &= ~0x02;            /* IIC コントロール部リセット解除 */
437
438       /* ==== IIC.ICCR1 の設定 ==== */
439       IIC0.ICCR1 &= ~0x80;            /* 機能停止状態 */
440
441       /* Master transfer mode setting */
442       IIC0.ICCR1 = 0xf6;                  /* ICE  : 転送動作可能
443                                             RCVD : 次の受信動作を禁止
444                                             MST  : B'1 マスタ送信モード選択
445                                             TRS  : B'1
446                                             CKS  : Pφ/100
447                                           */
448
449       /* I2C バス状態を確認 */
450       ret = iic_Chk_Bus_SCL_SDA();
451
452       /* 上記、IIC リセットを実施しても I2C バス解放されない場合 */
453       if(ret == D_IIC_EEP_NG)
454       {
455         return D_IIC_EEP_BUS_BUSY;
456       }
457
458       /* ==== IIC0.ICMR0 の設定 ==== */
459       IIC0.ICMR = 0x30;                   /* MSB ファースト
460                                             転送データビット数=9
```

```
461                                     */
462
463       /* TXI interrupt enable */
464       IIC0.ICIER.BIT.TIE = 1;
465
466       /* NAKI interrupt enable */
467       IIC0.ICIER.BIT.NAKIE = 1;
468
469       /* Start condition generate */
470       IIC0.ICCR2 = ((IIC0.ICCR2 & 0xbf) | 0x80);
471
472       /* EEPROM リード/ライト管理情報設定 */
473       set_Iic_Eep_Manage_Info(&manage_info);
474
475       /* EEPROM アクセス状態設定 */
476       set_Iic_Eep_Condition(D_IIC_EEP_START_ISSUE_WAIT); /* 開始条件発行待ち状態 */
477
478       return D_IIC_EEP_OK;
479
480   }
481
482   /*""FUNC COMMENT""************************************************
483   * ID                    :
484   * Outline               : EEPROM 状態取得処理
485   * Include               :
486   * Declaration           : E_Iic_eep_Ret iic_Chk_EepRom(void *info)
487   * Description           : EEPROM リード/ライト状態取得を行います。
488   *                       :
489   * Argument              : T_IIC_EEPROM_CONDITION 型参照
490   * Return Value          : E_Iic_eep_Ret 型参照
491   * Calling Functions     :
492   *""FUNC COMMENT END""*******************************************/
493   static E_Iic_eep_Ret iic_Chk_EepRom(void *info)
494   {
495       E_Iic_eep_Ret           ret = D_IIC_EEP_NG;
496       T_IIC_EEPROM_CONDITION  *condition_info;
497
498       if(info == NULL)
499       {
500         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);
501         return D_IIC_EEP_NG;
502       }
503
504       condition_info = (T_IIC_EEPROM_CONDITION *)info;
505
506       /* 現状態設定 */
507       condition_info->i_eep_condition = get_Iic_Eep_Condition();
508
509       /* エラー状態設定 */
510       condition_info->i_err_condition = get_Iic_Eep_Err_Condition();
511
512       do
513       {
514         /* エラー判定 */
515         if(D_IIC_EEP_ERR_NO != condition_info->i_err_condition)  /*エラー時 */
516         {
517             break;
518         }
519
520         /* 完了判定 */
521         /* 今の状態がアイドルかつ前状態が停止条件発行待ち状態の場合 */
522         if(D_IIC_EEP_IDLE         == get_Iic_Eep_Condition() &&
523            D_IIC_EEP_END_ISSUE_WAIT == get_Iic_Eep_Before_Condition())
524         {
525             ret = D_IIC_EEP_OK;               /* 完了 */
526             break;
```

```
527         }
528
529         switch(get_Iic_Eep_Mode())
530         {
531             case D_IIC_EEP_WRITE:          /* Write モード */
532                 ret = D_IIC_EEP_WRITING;
533                 break;
534
535             case D_IIC_EEP_READ:           /* Read モード */
536             case D_IIC_EEP_CURRENT_READ:
537                 ret = D_IIC_EEP_READING;
538                 break;
539
540             default:
541                 break;
542         }
543
544       }while(0);
545
546       return ret;
547
548   }
549
550   /*""FUNC COMMENT""************************************************
551   * ID                    :
552   * Outline                : AL 発生時処理
553   * Include                :
554   * Declaration            : E_Iic_eep_Ret iic_AL_generate(void* info)
555   *                       :
556   * Description            : AL 発生時時の処理を行います。
557   *                       :
558   * Argument               : void  *info       : No Use
559   * Return Value           : D_IIC_EEP_OK     : Success
560   * Calling Functions       :
561   *""FUNC COMMENT END""*******************************************/
562   static E_Iic_eep_Ret iic_AL_generate(void* info)
563   {
564
565       /* 本応用例では、AL 発生は想定外としております */
566       /* システムに見合った処理を実装ください */
567
568       if(IIC0.ICSR.BIT.ALOVE == 1)
569       {
570
571         /* エラー状態設定 */
572         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_AL);
573
574         IIC0.ICSR.BIT.ALOVE = 0;
575
576       }
577
578       return D_IIC_EEP_OK;
579   }
580
581   /*""FUNC COMMENT""************************************************
582   * ID                    :
583   * Outline                : IIC 開始条件発行後処理
584   * Include                :
585   * Declaration            : E_Iic_eep_Ret iic_After_start_issue(void *info)
586   * Description            : 開始条件発行後、Device address を
587   *                       : 設定します。
588   *                       : 本関数は、TXI 割り込みでコールされます。
589   * Argument               : void  *info       : No Use
590   * Return Value           : D_IIC_EEP_OK     : Success
591   *                       : D_IIC_EEP_NG       : Error
592   * Calling Functions       :
```

```
593   *""FUNC COMMENT END""*********************************************/
594   static E_Iic_eep_Ret iic_After_start_issue(void *info)
595   {
596       E_Iic_eep_Ret ret = D_IIC_EEP_OK;
597       T_IIC_EEPROM_RW_MANAGE  manage_info;
598       E_Iic_eep_condition     Eep_Condition;
599
600       memset(&manage_info, 0, sizeof(manage_info));
601
602       /* TXI interrupt disable */
603       IIC0.ICIER.BIT.TIE = 0;
604
605       /* EEPROM リード/ライト管理情報取得 */
606       get_Iic_Eep_Manage_Info(&manage_info);
607
608       /* 前状態により分岐 */
609       switch(get_Iic_Eep_Before_Condition())
610       {
611       case D_IIC_EEP_IDLE:                /* アイドル状態 */
612
613         /* EEPROM アクセス状態設定 */
614         Eep_Condition = D_IIC_EEP_SEND_DEVADD_WAIT;      /* Device Address 送信待ち状態 */
615
616         break;
617
618       case D_IIC_EEP_WRITE_END_ISSUE_WAIT:    /* Write 完了後 停止条件発行待ち状態 */
619       case D_IIC_EEP_WRITE_POLLING_WAIT:      /* WritePolling 待ち状態 */
620
621         /* EEPROM アクセス状態設定 */
622         Eep_Condition = D_IIC_EEP_WRITE_POLLING_WAIT; /* WritePolling 待ち状態 */
623
624         break;
625
626       case D_IIC_EEP_SEND_MEMADD_WAIT:        /* Memory address 送信待ち状態 */
627
628         /* Read code set */
629         manage_info.i_DevAdr |= D_IIC_R_CODE;
630
631         /* EEPROM リード/ライト管理情報設定 */
632         set_Iic_Eep_Manage_Info(&manage_info);
633
634         /* EEPROM アクセス状態設定 */
635         Eep_Condition = D_IIC_EEP_RESEND_DEVADD_WAIT; /* デバイスアドレス再送信待ち状態 */
636
637         break;
638
639       default:
640
641         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);          /* その他エラー */
642         ret = D_IIC_EEP_NG;
643         break;
644
645       }
646
647       if(ret == D_IIC_EEP_OK)
648       {
649         /* TEI interrupt enable */
650         IIC0.ICIER.BIT.TEIE = 1;
651
652         /* Device address set to transfer data */
653         IIC0.ICDRT = manage_info.i_DevAdr;
654
655         /* EEPROM アクセス状態設定 */
656         set_Iic_Eep_Condition(Eep_Condition);
657
658       }
```

```
659
660       return ret;
661
662   }
663
664   /*""FUNC COMMENT""************************************************
665   * ID                    :
666   * Outline               : 初回 Device Address 送信後処理
667   * Include               :
668   * Declaration           : E_Iic_eep_Ret iic_Send_After_Dev_Address
669   *                       : (void* info)
670   * Description           : 初回 Device Address 送信後の処理を行います。
671   *                       :
672   * Argument              : void  *info       : No Use
673   * Return Value          : D_IIC_EEP_OK      : Success
674   *                       : D_IIC_EEP_NG      : Error
675   * Calling Functions     :
676   *""FUNC COMMENT END""*****************************************/
677   static E_Iic_eep_Ret iic_Send_After_Dev_Address(void* info)
678   {
679       T_IIC_EEPROM_RW_MANAGE  manage_info;
680       E_Iic_eep_Ret           ret = D_IIC_EEP_NG;
681
682       memset(&manage_info, 0, sizeof(manage_info));
683
684       /* NACK found ? */
685       if(D_IIC_NACK_FOUND == iic_judge_NACK())
686       {                                                   /* 停止条件発行 */
687         return D_IIC_EEP_NG;
688       }
689
690       /* EEPROM リード/ライト管理情報取得 */
691       get_Iic_Eep_Manage_Info(&manage_info);
692
693       /* モードにより分岐 */
694       switch(manage_info.i_mode)
695       {
696         case D_IIC_EEP_CURRENT_READ:  /* Current Address Read と Sequential Read Operation 組み合わ
697   せモード */
698
699           /* Read モード遷移処理 */
700           ret = iic_chg_to_read();
701
702           break;
703
704       case D_IIC_EEP_READ:          /* Random Read Operation と Sequential Read Operation 組み合わ
705   せモード*/
706       case D_IIC_EEP_WRITE:         /* Write モード */
707
708           /* Memory address 送信処理 */
709           ret = iic_Send_Memory_Address();
710
711           /* EEPROM アクセス状態設定 */
712           set_Iic_Eep_Condition(D_IIC_EEP_SEND_MEMADD_WAIT);   /* Memory address 送信待ち状態 */
713
714           break;
715
716       default:
717           set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);      /* その他エラー */
718           ret = D_IIC_EEP_NG;
719           break;
720       }
721
722       return ret;
723
724   }
```

```
725
726   /*""FUNC COMMENT""************************************************
727   * ID                   :
728   * Outline               : Device Address 再送信後処理
729   * Include               :
730   * Declaration            : E_Iic_eep_Ret iic_After_Re_DevAdd_send
731   *                       :  (void* info)
732   * Description            : デバイスアドレス再送信後処理を
733   *                       : 行います。
734   * Argument               : void  *info       : No Use
735   * Return Value           : D_IIC_EEP_OK     : Success
736   *                       : D_IIC_EEP_NG       : Error
737   * Calling Functions       :
738   *""FUNC COMMENT END""*******************************************/
739   static E_Iic_eep_Ret iic_After_Re_DevAdd_send(void* info)
740   {
741       E_Iic_eep_Ret ret;
742
743       /* NACK found ? */
744       if(D_IIC_NACK_FOUND == iic_judge_NACK())
745       {                                                  /* 停止条件発行 */
746         return D_IIC_EEP_NG;
747       }
748
749       /* Read モード遷移処理 */
750       ret = iic_chg_to_read();
751
752       return ret;
753   }
754
755   /*""FUNC COMMENT""************************************************
756   * ID                   :
757   * Outline               : Memory address 送信中処理
758   * Include               :
759   * Declaration            : E_Iic_eep_Ret iic_MemAdd_Sending
760   *                       : (void* info)
761   * Description            : Memory address 送信処理を行います。
762   *                       : 本関数は、TEI 割り込みでコールされます。
763   *                       :
764   * Argument               : void  *info       : No Use
765   * Return Value           : D_IIC_EEP_OK     : Success
766   *                       : D_IIC_EEP_NG       : Error
767   * Calling Functions       :
768   *""FUNC COMMENT END""*******************************************/
769   static E_Iic_eep_Ret iic_MemAdd_Sending(void* info)
770   {
771       T_IIC_EEPROM_RW_MANAGE  manage_info;
772
773       memset(&manage_info, 0, sizeof(manage_info));
774
775       /* NACK found ? */
776       if(D_IIC_NACK_FOUND == iic_judge_NACK())
777       {                                                  /* 停止条件発行 */
778         return D_IIC_EEP_NG;
779       }
780
781       /* EEPROM リード/ライト管理情報取得 */
782       get_Iic_Eep_Manage_Info(&manage_info);
783
784       /* Memory address 送信完了? */
785       if(manage_info.i_MemAdrLen == 0)
786       {
787         /* Memory address 送信完了後処理 */
788         iic_After_MemAdd_send();
789       }
790       else
```

```
791       {
792         /* 2nd Memory address to transfer data */
793         iic_Send_Memory_Address();
794       }
795
796       return D_IIC_EEP_OK;
797
798   }
799
800   /*""FUNC COMMENT""************************************************
801   * ID                    :
802   * Outline                : WriteData 送信中処理
803   * Include                :
804   * Declaration            : E_Iic_eep_Ret iic_Write_Data_sending
805   *                       :  (void* info)
806   * Description            : WriteData 送信中処理を行います。
807   *                       : 未送信 Write データが存在する場合は、
808   *                       : 送信処理を行います。
809   *                       : 未送信 Write データが存在しない場合は、
810   *                       : 停止条件を発行します。
811   *                       : 本関数は、TEI 割り込みでコールされます。
812   *                       :
813   * Argument               : void  *info       : No Use
814   * Return Value           : D_IIC_EEP_OK     : Success
815   *                       : D_IIC_EEP_NG       : Error
816   * Calling Functions       :
817   *""FUNC COMMENT END""*******************************************/
818   static E_Iic_eep_Ret iic_Write_Data_sending(void* info)
819   {
820       T_IIC_EEPROM_RW_MANAGE   manage_info;
821
822       memset(&manage_info, 0, sizeof(manage_info));
823
824       /* NACK found ? */
825       if(D_IIC_NACK_FOUND == iic_judge_NACK())
826       {                                                  /* 停止条件発行 */
827         return D_IIC_EEP_NG;
828       }
829
830       /* EEPROM リード/ライト管理情報取得 */
831       get_Iic_Eep_Manage_Info(&manage_info);
832
833       if(manage_info.i_RW_Data_info.i_DataLen == 0)  /* 送信データがない場合 */
834       {
835         /* TEND クリア */
836         IIC0.ICSR.BIT.TEND = 0;
837
838         /* TEI interrupt disable */
839         IIC0.ICIER.BIT.TEIE = 0;
840
841         /* STPI interrupt enable */
842         IIC0.ICIER.BIT.STIE = 1;
843
844         /* SCL ライン監視処理 */
845         /* 9 クロック目の立ち下がりを認識してから停止条件発行 */
846         iic_chk_scl_down();
847
848         /* Stop condition generate */
849         IIC0.ICSR.BIT.STOP  = 0;       /* STOP flag clear */
850         IIC0.ICCR2 = IIC0.ICCR2 & (0x7F & 0xBF);
851
852         /* EEPROM アクセス状態設定 */
853         set_Iic_Eep_Condition(D_IIC_EEP_WRITE_END_ISSUE_WAIT);   /* Write 完了後 停止条件発行待ち状態
854   */
855
856       }
```

```
857       else                                          /* 送信データがある場合 */
858       {
859         /* データ Write 処理  */
860         iic_send_Write_Data();
861       }
862
863       return D_IIC_EEP_OK;
864
865   }
866
867   /*""FUNC COMMENT""*************************************************
868   * ID                    :
869   * Outline               : ReadData 受信処理
870   * Include               :
871   * Declaration           : E_Iic_eep_Ret iic_Read_Data_rcv(void *info)
872   * Description           : RXI 割り込み時のデータ受信処理を行います。
873   *                       :
874   * Argument              : void  *info       : No Use
875   * Return Value          : D_IIC_EEP_OK      : Success
876   * Calling Functions     :
877   *""FUNC COMMENT END""*********************************************/
878   static E_Iic_eep_Ret iic_Read_Data_rcv(void *info)
879   {
880       unsigned char           dummy;
881       E_Iic_eep_condition     Eep_Condition;
882       T_IIC_EEPROM_RW_MANAGE  manage_info;
883
884       memset(&manage_info, 0, sizeof(manage_info));
885
886       Eep_Condition = get_Iic_Eep_Condition();           /* 現状態取得 */
887
888       /* EEPROM リード/ライト管理情報取得 */
889       get_Iic_Eep_Manage_Info(&manage_info);
890
891       manage_info.i_RW_Data_info.i_DataLen--;           /* 受信予定データ数デクリメント */
892
893       if(manage_info.i_RW_Data_info.i_DataLen == 0)     /* リード残回数 = 0 */
894       {
895         IIC0.ICIER.BIT.RIE = 0;                         /* RXI interrupt disable */
896         IIC0.ICIER.BIT.STIE = 1;                      /* STPI interrupt enable */
897
898         /* SCL ライン監視処理 */
899         /* 9 クロック目の立ち下がりを認識してから停止条件発行 */
900         iic_chk_scl_down();
901
902         /* Stop condition generate */
903         IIC0.ICSR.BIT.STOP  = 0;                      /* STOP flag clear */
904         IIC0.ICCR2 = IIC0.ICCR2 & (0x7F & 0xBF);
905
906         /* EEPROM アクセス状態設定 */
907         Eep_Condition = D_IIC_EEP_END_ISSUE_WAIT;     /* 停止条件発行待ち状態 */
908
909       }
910       else if(manage_info.i_RW_Data_info.i_DataLen == 1) /* リード残回数 = 1 */
911       {
912         IIC0.ICIER.BIT.ACKBT = 1;                     /* Send NACK */
913
914         *manage_info.i_RW_Data_info.i_DataBuf = IIC0.ICDRR;
915         manage_info.i_RW_Data_info.i_DataBuf++;
916
917       }
918       else
919       {
920         *manage_info.i_RW_Data_info.i_DataBuf = IIC0.ICDRR;
921         manage_info.i_RW_Data_info.i_DataBuf++;
922
```

```
923       }
924
925       /* EEPROM リード/ライト管理情報更新 */
926       set_Iic_Eep_Manage_Info(&manage_info);
927
928       /* EEPROM アクセス状態設定 */
929       set_Iic_Eep_Condition(Eep_Condition);
930
931       return D_IIC_EEP_OK;
932
933   }
934
935   /*""FUNC COMMENT""************************************************
936   * ID                    :
937   * Outline               : 停止条件発行後 Write Polling 準備処理
938   * Include               :
939   * Declaration           : E_Iic_eep_Ret iic_After_end_issue_Write_Polling
940   *                       : (void *info)
941   * Description           : 停止条件発行後の処理を行います。
942   *                       :
943   * Argument              : void  *info       : No Use
944   * Return Value          : D_IIC_EEP_OK      : Success
945   * Calling Functions     :
946   *""FUNC COMMENT END""*******************************************/
947   static E_Iic_eep_Ret iic_After_end_issue_Write_Polling(void *info)
948   {
949       T_IIC_EEPROM_RW_MANAGE  manage_info;
950
951       memset(&manage_info, 0, sizeof(manage_info));
952
953       /* EEPROM リード/ライト管理情報取得 */
954       get_Iic_Eep_Manage_Info(&manage_info);
955
956       IIC0.ICSR.BIT.STOP = 0;         /* STOP clear */
957
958       IIC0.ICIER.BIT.STIE = 0;    /* STPI interrupt disable */
959
960       /* Write Polling cycle set */
961       manage_info.i_WCTCnt = D_IIC_WCT_CNT;
962
963       /* TXI interrupt enable */
964       IIC0.ICIER.BIT.TIE = 1;
965
966       /* Start condition generate */
967       IIC0.ICCR2 = ((IIC0.ICCR2 & 0xbf) | 0x80);
968
969       /* EEPROM リード/ライト管理情報更新 */
970       set_Iic_Eep_Manage_Info(&manage_info);
971
972       /* EEPROM アクセス状態設定 */
973       set_Iic_Eep_Condition(D_IIC_EEP_START_ISSUE_WAIT); /* 開始条件発行待ち状態 */
974
975       return D_IIC_EEP_OK;
976
977   }
978
979   /*""FUNC COMMENT""************************************************
980   * ID                    :
981   * Outline               : WritePolling 処理
982   * Include               :
983   * Declaration           : E_Iic_eep_Ret iic_Eep_WritePolling
984   *                       : (void* info)
985   * Description           : EEPROM ライト終了後のポーリング処理を
986   *                       : 行います。
987   * Argument              : void  *info       : No Use
988   * Return Value          : D_IIC_EEP_OK      : Success
```

```
989   * Calling Functions       :
990   *""FUNC COMMENT END""*********************************************/
991   static E_Iic_eep_Ret iic_Eep_WritePolling(void* info)
992   {
993       T_IIC_EEPROM_RW_MANAGE  manage_info;
994       E_Iic_eep_condition     Eep_Condition;
995
996       memset(&manage_info, 0, sizeof(manage_info));
997
998       Eep_Condition = get_Iic_Eep_Condition();           /* 現状態取得 */
999
1000      /* EEPROM リード/ライト管理情報取得 */
1001      get_Iic_Eep_Manage_Info(&manage_info);
1002
1003      /* NACK found ? */
1004      if(IIC0.ICIER.BIT.ACKBR == 1)   /* NACK 検出? */
1005      {
1006        if(manage_info.i_WCTCnt == 0) /* WritePolling == 0 の場合 */
1007        {
1008            /* TEND クリア */
1009            IIC0.ICSR.BIT.TEND = 0;
1010
1011            /* TEI interrupt disable */
1012            IIC0.ICIER.BIT.TEIE = 0;
1013
1014            /* STPI interrupt enable */
1015            IIC0.ICIER.BIT.STIE = 1;
1016
1017            /* SCL ライン監視処理 */
1018            /* 9 クロック目の立ち下がりを認識してから停止条件発行 */
1019            iic_chk_scl_down();
1020
1021            /* Stop condition generate */
1022            IIC0.ICSR.BIT.STOP  = 0;       /* STOP flag clear */
1023            IIC0.ICCR2 = IIC0.ICCR2 & (0x7F & 0xBF);
1024
1025            set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_WCT_OVER);   /* エラー状態設定 */
1026
1027            /* EEPROM アクセス状態設定 */
1028            Eep_Condition = D_IIC_EEP_END_ISSUE_WAIT;     /* 停止条件発行待ち状態 */
1029
1030        }
1031        else                          /* WritePolling > 0 の場合 */
1032        {
1033            /* WritePolling カウントデクリメント */
1034            manage_info.i_WCTCnt--;
1035
1036            /* TEND クリア */
1037            IIC0.ICSR.BIT.TEND = 0;
1038
1039            /* TEI interrupt disable */
1040            IIC0.ICIER.BIT.TEIE = 0;
1041
1042            /* TXI interrupt enable */
1043            IIC0.ICIER.BIT.TIE = 1;
1044
1045            /* SCL ライン監視処理 */
1046            /* 9 クロック目の立ち下がりを認識してから再送条件発行 */
1047            iic_chk_scl_down();
1048
1049            /* ReStart condition generate */
1050            IIC0.ICCR2 = ((IIC0.ICCR2 & 0xbf) | 0x80);
1051
1052            /* EEPROM アクセス状態設定 */
1053            Eep_Condition = D_IIC_EEP_START_ISSUE_WAIT;      /* 開始条件発行待ち状態 */
1054
```

```
1055        }
1056      }
1057      else                          /* ACK 検出? */
1058      {
1059        manage_info.i_WCTCnt = 0;
1060
1061        /* TEND クリア */
1062        IIC0.ICSR.BIT.TEND = 0;
1063
1064        /* TEI interrupt disable */
1065        IIC0.ICIER.BIT.TEIE = 0;
1066
1067        /* STPI interrupt enable */
1068        IIC0.ICIER.BIT.STIE = 1;
1069
1070        /* SCL ライン監視処理 */
1071        /* 9 クロック目の立ち下がりを認識してから停止条件発行 */
1072        iic_chk_scl_down();
1073
1074        /* Stop condition generate */
1075        IIC0.ICSR.BIT.STOP  = 0;       /* STOP flag clear */
1076        IIC0.ICCR2 = IIC0.ICCR2 & (0x7F & 0xBF);
1077
1078        /* EEPROM アクセス状態設定 */
1079        Eep_Condition = D_IIC_EEP_END_ISSUE_WAIT;     /* 停止条件発行待ち状態 */
1080
1081      }
1082
1083      /* EEPROM リード/ライト管理情報設定 */
1084      set_Iic_Eep_Manage_Info(&manage_info);
1085
1086      /* EEPROM アクセス状態設定 */
1087      set_Iic_Eep_Condition(Eep_Condition);
1088
1089      return D_IIC_EEP_OK;
1090
1091  }
1092
1093  /*""FUNC COMMENT""************************************************
1094  * ID                    :
1095  * Outline               : 停止条件発行後処理
1096  * Include               :
1097  * Declaration           : E_Iic_eep_Ret iic_After_end_issue
1098  *                       :  (void *info)
1099  * Description           : STPI 割り込み時の処理を行います。
1100  *                       :
1101  * Argument              : void  *info       : No Use
1102  * Return Value          : D_IIC_EEP_OK      : Success
1103  * Calling Functions     :
1104  *""FUNC COMMENT END""*******************************************/
1105  static E_Iic_eep_Ret iic_After_end_issue(void *info)
1106  {
1107      E_Iic_eep_Ret           ret;
1108      unsigned char           work;
1109      T_IIC_EEPROM_RW_MANAGE  manage_info;
1110
1111      memset(&manage_info, 0, sizeof(manage_info));
1112
1113      /* flg clear */
1114      IIC0.ICSR.BIT.TEND = 0;         /* TEND clear */
1115      IIC0.ICSR.BIT.RDRF = 0;         /* RDRF clear */
1116      IIC0.ICSR.BIT.STOP = 0;         /* STOP clear */
1117
1118      /* IIC interrupt disable */
1119      IIC0.ICIER.BIT.STIE = 0;    /* STPI interrupt disable */
1120      IIC0.ICIER.BIT.TEIE = 0;    /* TEI interrupt disable */
```

```
1121      IIC0.ICIER.BIT.TIE = 0;         /* TXI interrupt disable */
1122      IIC0.ICIER.BIT.RIE = 0;         /* RXI interrupt disable */
1123
1124      work = IIC0.ICCR1;
1125
1126      if( D_IIC_EEP_READ == get_Iic_Eep_Mode() ||
1127        D_IIC_EEP_CURRENT_READ == get_Iic_Eep_Mode())
1128      {
1129        /* EEPROM リード/ライト管理情報取得 */
1130        get_Iic_Eep_Manage_Info(&manage_info);
1131
1132        *manage_info.i_RW_Data_info.i_DataBuf = IIC0.ICDRR;  /* 最終データ取得 */
1133        work = work & ~0x40;                              /* RCVD = 0 設定 */
1134
1135        /* EEPROM リード/ライト管理情報更新 */
1136        set_Iic_Eep_Manage_Info(&manage_info);
1137
1138      }
1139
1140      work &= 0xcf;                   /* Slave receive mode set */
1141      IIC0.ICCR1 = work;
1142
1143      if( D_IIC_EEP_WRITE == get_Iic_Eep_Mode() )
1144      {
1145        IIC0.ICSR.BIT.TDRE = 0;           /* TDRE clear */
1146      }
1147
1148      /* I2C バス状態を確認 */
1149      ret = iic_Chk_Bus_SCL_SDA();
1150
1151      /* I2C バスが解放されない場合、本応用例では想定外であるため無限ループとしている */
1152      if(ret != D_IIC_EEP_OK)
1153      {
1154        while(1);
1155      }
1156
1157      /* EEPROM アクセス状態設定 */
1158      set_Iic_Eep_Condition(D_IIC_EEP_IDLE);  /* アイドル状態 */
1159
1160      return D_IIC_EEP_OK;
1161
1162  }
1163
1164  /*****************************************************************************************/
1165  /* 内部関数 */
1166
1167  /*""FUNC COMMENT""*************************************************
1168  * ID                    :
1169  * Outline               : I2C バス解放確認処理
1170  * Include               :
1171  * Declaration           : E_Iic_eep_Ret iic_Chk_Bus_SCL_SDA(void)
1172  *                       :
1173  *                       :
1174  * Description           : ・スレーブデバイス状態確認を確認します。
1175  *                       :  ※スレーブデバイス暴走確認
1176  *                       :  (I2C ラインを Low に引っ張り続けていないか？)
1177  *                       :  D_IIC_BUSYCHECK_TMOUT に確認用タイマーに
1178  *                       :  スレーブデバイス状態を確認し続ける時間をセットください。
1179  *                       : ・I2C バスの占有／解放状態を確認します。
1180  *                       :
1181  * Argument              : none
1182  * Return Value          : D_IIC_EEP_OK        :I2C バス解放
1183  *                       : D_IIC_EEP_NG        :I2C バス非解放
1184  * Calling Functions     :
1185  *""FUNC COMMENT END""*******************************************/
1186  static E_Iic_eep_Ret iic_Chk_Bus_SCL_SDA(void)
```

```
1187  {
1188      unsigned long    i;
1189      E_Iic_eep_Ret    ret = D_IIC_EEP_OK;
1190
1191      for (i = D_IIC_BUSYCHECK_TMOUT; i > 0; i--)
1192      {
1193        /* BBSY,SDAO,SCLO のフラグ確認 */
1194        if ( (IIC0.ICCR2 & 0xA8) == 0x28 )/* BBSY = 0?  SDAO SCLO = 1? */
1195        {
1196            break;                              /* I2C バス解放 */
1197        }
1198      }
1199
1200      /* I2C バス非解放? */
1201      if(i == 0)
1202      {
1203        ret = D_IIC_EEP_NG;
1204      }
1205
1206      return ret;
1207  }
1208
1209  /*""FUNC COMMENT""************************************************
1210  * ID                    :
1211  * Outline               : Read モード遷移処理
1212  * Include               :
1213  * Declaration           : E_Iic_eep_Ret iic_chg_to_read(void)
1214  *                       :
1215  * Description           : マスタ受信モードに遷移します。
1216  *                       :
1217  * Argument              : none
1218  * Return Value          : D_IIC_EEP_OK     : Success
1219  * Calling Functions     :
1220  *""FUNC COMMENT END""*******************************************/
1221  static E_Iic_eep_Ret iic_chg_to_read(void)
1222  {
1223      T_IIC_EEPROM_RW_MANAGE  manage_info;
1224      unsigned char           dummy;
1225
1226      memset(&manage_info, 0, sizeof(manage_info));
1227
1228      /* EEPROM リード/ライト管理情報取得 */
1229      get_Iic_Eep_Manage_Info(&manage_info);
1230
1231      /* TEND clear */
1232      IIC0.ICSR.BIT.TEND = 0;
1233
1234      /* TEI interrupt disable */
1235      IIC0.ICIER.BIT.TEIE = 0;
1236
1237      /* Master receive mode */
1238      IIC0.ICCR1 &= ~0x10;
1239
1240      /* TDRE clear */
1241      IIC0.ICSR.BIT.TDRE = 0;
1242
1243      /* TXI interrupt disable */
1244      IIC0.ICIER.BIT.TIE = 0;
1245
1246      /* ICCR1.RCVD 連続受信禁止 */
1247      IIC0.ICCR1 |= 0x40;
1248
1249      /* Read データが 1 バイトのみ? */
1250      if(manage_info.i_RW_Data_info.i_DataLen == 1)
1251      {
1252        IIC0.ICIER.BIT.ACKBT = 1;
```

```
1253      }
1254      else
1255      {
1256        IIC0.ICIER.BIT.ACKBT = 0;
1257      }
1258
1259      /* RXI interrupt enable */
1260      IIC0.ICIER.BIT.RIE = 1;
1261
1262      /* dummy Read */
1263      dummy = IIC0.ICDRR;
1264
1265      /* EEPROM アクセス状態設定 */
1266      set_Iic_Eep_Condition(D_IIC_EEP_SEND_RCV_DATA_WAIT);   /* データ送受信待ち状態 */
1267
1268      return D_IIC_EEP_OK;
1269
1270  }
1271
1272  /*""FUNC COMMENT""************************************************
1273  * ID                    :
1274  * Outline               : Memory address 送信処理
1275  * Include               :
1276  * Declaration           : E_Iic_eep_Ret iic_Send_Memory_Address
1277  *                       : (void)
1278  * Description           : Memory address 送信処理を行います。
1279  *                       :
1280  * Argument              : none
1281  * Return Value          : D_IIC_EEP_OK     : Success
1282  * Calling Functions     :
1283  *""FUNC COMMENT END""*******************************************/
1284  static E_Iic_eep_Ret iic_Send_Memory_Address(void)
1285  {
1286      T_IIC_EEPROM_RW_MANAGE  manage_info;
1287
1288      memset(&manage_info, 0, sizeof(manage_info));
1289
1290      /* EEPROM リード/ライト管理情報取得 */
1291      get_Iic_Eep_Manage_Info(&manage_info);
1292
1293      /* Memory address set to transfer data */
1294      manage_info.i_MemAdrLen--;
1295      IIC0.ICDRT = manage_info.i_MemAdrBuf[manage_info.i_MemAdrLen];
1296
1297      /* EEPROM リード/ライト管理情報更新 */
1298      set_Iic_Eep_Manage_Info(&manage_info);
1299
1300      return D_IIC_EEP_OK;
1301
1302  }
1303
1304  /*""FUNC COMMENT""************************************************
1305  * ID                    :
1306  * Outline               : Memory address 送信完了後処理
1307  * Include               :
1308  * Declaration           : void iic_After_MemAdd_send(void)
1309  *                       :
1310  * Description           : Memory address 送信完了後処理を行います。
1311  *                       :
1312  * Argument              : none
1313  * Return Value          : none
1314  * Calling Functions     :
1315  *""FUNC COMMENT END""*******************************************/
1316  static void iic_After_MemAdd_send(void)
1317  {
1318      switch(get_Iic_Eep_Mode())
```

```
1319      {
1320      case D_IIC_EEP_READ: /* Read モード */
1321        iic_After_MemAdd_send_Read_Mode();    /* Memory address 送信後 Read 処理 */
1322        break;
1323
1324      case D_IIC_EEP_WRITE:    /* Write モード */
1325        iic_After_MemAdd_send_Write_Mode();   /* Memory address 送信後 Write 処理 */
1326        break;
1327
1328      default:
1329        break;
1330      }
1331  }
1332
1333  /*""FUNC COMMENT""*************************************************
1334  * ID                    :
1335  * Outline                : Memory address 送信後 Read 処理
1336  * Include                :
1337  * Declaration            : void iic_After_MemAdd_send_Read_Mode(void)
1338  * Description            : Restart condition generate します。
1339  *                       :
1340  *                       :
1341  * Argument               : none
1342  * Return Value           : none
1343  * Calling Functions       :
1344  *""FUNC COMMENT END""*********************************************/
1345  static void iic_After_MemAdd_send_Read_Mode(void)
1346  {
1347      /* TEND クリア */
1348      IIC0.ICSR.BIT.TEND = 0;
1349
1350      /* TEI interrupt disable */
1351      IIC0.ICIER.BIT.TEIE = 0;
1352
1353      /* TXI interrupt enable */
1354      IIC0.ICIER.BIT.TIE = 1;
1355
1356      /* SCL ライン監視処理 */
1357      /* 9 クロック目の立ち下がりを認識してから再送条件発行 */
1358      iic_chk_scl_down();
1359
1360      /* ReStart condition generate */
1361      IIC0.ICCR2 = ((IIC0.ICCR2 & 0xbf) | 0x80);
1362
1363      /* EEPROM アクセス状態設定 */
1364      set_Iic_Eep_Condition(D_IIC_EEP_START_ISSUE_WAIT); /* 開始条件発行待ち状態 */
1365
1366      return;
1367
1368  }
1369
1370  /*""FUNC COMMENT""*************************************************
1371  * ID                    :
1372  * Outline                : データ Write 処理
1373  * Include                :
1374  * Declaration            : void iic_After_MemAdd_send_Write_Mode(void)
1375  * Description            : 初回ライトデータを ICDRT に設定します。
1376  *                       :
1377  *                       :
1378  * Argument               : none
1379  * Return Value           : none
1380  * Calling Functions       :
1381  *""FUNC COMMENT END""*********************************************/
1382  static void iic_After_MemAdd_send_Write_Mode(void)
1383  {
1384      /* データ Write 処理 */
```

```
1385      iic_send_Write_Data();
1386
1387      /* EEPROM アクセス状態設定 */
1388      set_Iic_Eep_Condition(D_IIC_EEP_SEND_RCV_DATA_WAIT);   /* データ送信待ち状態 */
1389
1390      return;
1391
1392  }
1393
1394  /*""FUNC COMMENT""************************************************
1395  * ID                    :
1396  * Outline               : データ Write 処理
1397  * Include               :
1398  * Declaration           : void iic_send_Write_Data(void)
1399  * Description           : ライトデータを ICDRT に設定します。
1400  *                       :
1401  *                       :
1402  * Argument              : none
1403  * Return Value          : none
1404  * Calling Functions     :
1405  *""FUNC COMMENT END""*******************************************/
1406  static void iic_send_Write_Data(void)
1407  {
1408      T_IIC_EEPROM_RW_MANAGE  manage_info;
1409
1410      memset(&manage_info, 0, sizeof(manage_info));
1411
1412      /* EEPROM リード/ライト管理情報取得 */
1413      get_Iic_Eep_Manage_Info(&manage_info);
1414
1415      /* Write Data to transfer data */
1416      IIC0.ICDRT = *manage_info.i_RW_Data_info.i_DataBuf;
1417      manage_info.i_RW_Data_info.i_DataBuf++;
1418      manage_info.i_RW_Data_info.i_DataLen--;
1419
1420      /* EEPROM リード/ライト管理情報更新 */
1421      set_Iic_Eep_Manage_Info(&manage_info);
1422
1423      return;
1424
1425  }
1426
1427  /*""FUNC COMMENT""************************************************
1428  * ID                    :
1429  * Outline               : NACK 検出判定処理
1430  * Include               :
1431  * Declaration           : E_Iic_eep_Nack iic_judge_NACK(void)
1432  * Description           : 前回送信処理に対して NACK を検出したか
1433  *                       : 判定します。
1434  *                       :
1435  *                       :
1436  * Argument              : void
1437  * Return Value          : E_Iic_eep_Nack
1438  *                       : D_IIC_NACK_FOUND    : NACK あり
1439  *                       : D_IIC_NACK_NOTFOUND : NACK なし
1440  * Calling Functions     :
1441  *""FUNC COMMENT END""*******************************************/
1442  static E_Iic_eep_Nack iic_judge_NACK(void)
1443  {
1444      E_Iic_eep_Nack      ret = D_IIC_NACK_NOTFOUND;
1445
1446      /* システムに見合った処理を実装ください */
1447
1448      if(IIC0.ICIER.BIT.ACKBR == 1)   /* NACK 検出 */
1449      {
1450        /* エラー状態の設定 */
```

```
1451        set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_NACK);
1452
1453        ret = D_IIC_NACK_FOUND;
1454
1455      }
1456
1457      return ret;
1458  }
1459
1460  /*""FUNC COMMENT""************************************************
1461  * ID                    :
1462  * Outline               : SCL ライン監視処理
1463  * Include               :
1464  * Declaration           : void iic_chk_scl_down(void)
1465  * Description           : SCL ラインが立ち下がるまで待ちます。
1466  *                       :
1467  * Argument              : none
1468  * Return Value          : none
1469  * Calling Functions     :
1470  *""FUNC COMMENT END""*******************************************/
1471  static void iic_chk_scl_down(void)
1472  {
1473      /* SCL ライン監視処理 */
1474      do
1475      {
1476        if(((IIC0.ICCR2 & 0x08) >> 3) == 0)
1477        {
1478            break;
1479        }
1480
1481      }while(1);
1482
1483  }
1484
1485  /*""FUNC COMMENT""************************************************
1486  * ID                    :
1487  * Outline               : IIC_EEPROM 内部情報初期化処理
1488  * Include               :
1489  * Declaration           : void set_internal_info_init(void)
1490  * Description           : IIC_EEPROM 内部情報を初期化します。
1491  *                       :
1492  * Argument              : none
1493  * Return Value          : none
1494  * Calling Functions     :
1495  *""FUNC COMMENT END""*******************************************/
1496  static void set_internal_info_init(void)
1497  {
1498      gIic_Eep_Condition = D_IIC_EEP_NO_INIT;           /* IIC_EEPROM 状態 */
1499      gIic_Eep_Before_Condition = D_IIC_EEP_NO_INIT;    /* IIC_EEPROM 前状態 */
1500
1501      set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_NO);      /* エラーなし状態 */
1502
1503      /* EEPROM リード/ライト管理情報初期化 */
1504      memset(&gIic_Eep_Manage_Info, 0, sizeof(gIic_Eep_Manage_Info));
1505
1506  }
1507
1508  /*""FUNC COMMENT""************************************************
1509  * ID                    :
1510  * Outline               : IIC_EEPROM 内部状態設定
1511  * Include               :
1512  * Declaration           : void set_Iic_Eep_Condition(
1513  *                       :     E_Iic_eep_condition i_condition)
1514  * Description           : IIC_EEPROM 内部状態設定を行います。
1515  *                       :
1516  * Argument              : E_Iic_eep_condition i_condition
```

```
1517  *                     : iic_eeprom.h の E_Iic_eep_condition の
1518  *                     : 状態を設定します。
1519  * Return Value          : none
1520  * Calling Functions      :
1521  *""FUNC COMMENT END""**********************************************/
1522  void set_Iic_Eep_Condition(E_Iic_eep_condition i_condition)
1523  {
1524      gIic_Eep_Before_Condition = gIic_Eep_Condition;    /* 前状態設定 */
1525      gIic_Eep_Condition = i_condition;                  /* 現状態設定 */
1526  }
1527
1528  /*""FUNC COMMENT""****************************************************
1529  * ID                   :
1530  * Outline               : IIC_EEPROM 内部現状態取得
1531  * Include               :
1532  * Declaration           : E_Iic_eep_condition
1533  *                      :     get_Iic_Eep_Condition(void)
1534  * Description           : IIC_EEPROM 内部現状態取得を行います。
1535  *                      :
1536  * Argument              : none
1537  * Return Value          : E_Iic_eep_condition
1538  *                      : iic_eeprom.h の E_Iic_eep_condition の
1539  *                      : 状態を取得します。
1540  * Calling Functions      :
1541  *""FUNC COMMENT END""**********************************************/
1542  E_Iic_eep_condition get_Iic_Eep_Condition(void)
1543  {
1544      return gIic_Eep_Condition;
1545  }
1546
1547  /*""FUNC COMMENT""****************************************************
1548  * ID                   :
1549  * Outline               : IIC_EEPROM 内部前状態設定
1550  * Include               :
1551  * Declaration           : void set_Iic_Eep_Before_Condition(
1552  *                      :     E_Iic_eep_condition i_condition)
1553  * Description           : IIC_EEPROM 内部前状態設定を行います。
1554  *                      :
1555  * Argument              : E_Iic_eep_condition i_condition
1556  *                      : iic_eeprom.h の E_Iic_eep_condition の
1557  *                      : 状態を設定します。
1558  * Return Value          : none
1559  * Calling Functions      :
1560  *""FUNC COMMENT END""**********************************************/
1561  void set_Iic_Eep_Before_Condition(E_Iic_eep_condition i_condition)
1562  {
1563      gIic_Eep_Before_Condition = i_condition;           /* 前状態設定 */
1564  }
1565
1566  /*""FUNC COMMENT""****************************************************
1567  * ID                   :
1568  * Outline               : IIC_EEPROM 内部前状態取得
1569  * Include               :
1570  * Declaration           : E_Iic_eep_condition
1571  *                      :     get_Iic_Eep_Before_Condition(void)
1572  * Description           : IIC_EEPROM 内部前状態取得を行います。
1573  *                      :
1574  * Argument              : none
1575  * Return Value          : E_Iic_eep_condition
1576  *                      : iic_eeprom.h の E_Iic_eep_condition の
1577  *                      : 状態を取得します。
1578  * Calling Functions      :
1579  *""FUNC COMMENT END""**********************************************/
1580  E_Iic_eep_condition get_Iic_Eep_Before_Condition(void)
1581  {
1582      return gIic_Eep_Before_Condition;
```

```
1583  }
1584
1585  /*""FUNC COMMENT""************************************************
1586  * ID                   :
1587  * Outline               : IIC_EEPROM 内部モード取得
1588  * Include               :
1589  * Declaration            : E_Iic_eep_mode
1590  *                      :     get_Iic_Eep_Mode(void)
1591  * Description            : IIC 内部モード取得を行います。
1592  *                      :
1593  * Argument              : none
1594  * Return Value           : E_Iic_eep_mode
1595  *                      : iic_eeprom.h の E_Iic_eep_mode の
1596  *                      : モードを取得します。
1597  * Calling Functions       :
1598  *""FUNC COMMENT END""*******************************************/
1599  E_Iic_eep_mode get_Iic_Eep_Mode(void)
1600  {
1601      return gIic_Eep_Manage_Info.i_mode;
1602  }
1603
1604  /*""FUNC COMMENT""************************************************
1605  * ID                   :
1606  * Outline               : IIC_EEPROM 内部管理情報設定
1607  * Include               :
1608  * Declaration            : void set_Iic_Eep_Manage_Info(
1609  *                      :     T_IIC_EEPROM_RW_MANAGE *i_info)
1610  * Description            : IIC_EEPROM 内部管理情報の設定を行います。
1611  *                      :
1612  * Argument              : T_IIC_EEPROM_RW_MANAGE *i_info
1613  * Return Value           : none
1614  * Calling Functions       :
1615  *""FUNC COMMENT END""*******************************************/
1616  void set_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *i_info)
1617  {
1618      memcpy(&gIic_Eep_Manage_Info, i_info, sizeof(T_IIC_EEPROM_RW_MANAGE));
1619  }
1620
1621  /*""FUNC COMMENT""************************************************
1622  * ID                   :
1623  * Outline               : IIC_EEPROM 内部管理情報取得
1624  * Include               :
1625  * Declaration            :  void get_Iic_Eep_Manage_Info(
1626  *                      :     T_IIC_EEPROM_RW_MANAGE *o_info)
1627  * Description            : IIC_EEPROM 内部管理情報の取得を行います。
1628  *                      :
1629  * Argument              : T_IIC_EEPROM_RW_MANAGE *o_info
1630  * Return Value           : none
1631  * Calling Functions       :
1632  *""FUNC COMMENT END""*******************************************/
1633  void get_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *o_info)
1634  {
1635      memcpy(o_info, &gIic_Eep_Manage_Info, sizeof(T_IIC_EEPROM_RW_MANAGE));
1636  }
1637
1638  /*""FUNC COMMENT""************************************************
1639  * ID                   :
1640  * Outline               : IIC_EEPROM 内部エラー情報設定
1641  * Include               :
1642  * Declaration            :  void set_Iic_Eep_Err_Condition(
1643  *                      :     E_Iic_eep_err_condition i_err)
1644  * Description            : IIC_EEPROM 内部エラー情報の設定を行います。
1645  *                      :
1646  * Argument              : E_Iic_eep_err_condition i_err
1647  * Return Value           : none
1648  * Calling Functions       :
```

```
1649  *""FUNC COMMENT END""*********************************************/
1650  void set_Iic_Eep_Err_Condition(E_Iic_eep_err_condition i_err)
1651  {
1652      gIic_Eep_Err_Condition = i_err;
1653  }
1654
1655  /*""FUNC COMMENT""**************************************************
1656  * ID                    :
1657  * Outline               : IIC_EEPROM 内部エラー情報取得
1658  * Include               :
1659  * Declaration           :  E_Iic_eep_err_condition
1660  *                       :     get_Iic_Eep_Err_Condition()
1661  * Description           : IIC_EEPROM 内部エラー情報の取得を行います。
1662  *                       :
1663  * Argument              : none
1664  * Return Value          : E_Iic_eep_err_condition
1665  * Calling Functions     :
1666  *""FUNC COMMENT END""*********************************************/
1667  E_Iic_eep_err_condition get_Iic_Eep_Err_Condition(void)
1668  {
1669      return gIic_Eep_Err_Condition;
1670  }
1671
1672  /* End of File */
```

(4) サンプルプログラムリスト "iic_eeprom.h"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation. and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation. and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************/
28    /* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
29    /*""FILE COMMENT""*********** Technical reference data ***************
30    * System Name  : SH7730 Sample Program
31    * File Name    : iic_eeprom.h
32    * Abstract     : SH7730 IIC 設定例 Sample Program（EEPROM 接続）
33    * Version      : Ver 1.00
34    * Device       : SH7730
35    * Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
36    *              : C/C++ Compiler Package for SuperH Family (V.9.03 release00)
37    * OS           : None
38    * H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
39    * Description  : SH7730 IIC 設定例（EEPROM 接続）のサンプルプログラムです。
40    *              :
41    * Operation    :
42    * Limitation   :
43    *              :
44    ***********************************************************************
45    * History      : 18.Jun.2010 Ver. 1.00 First Release
46    *""FILE COMMENT END""*************************************************/
47
48    #ifndef __IIC_EEPROM_DEF_H__
49    #define __IIC_EEPROM_DEF_H__
50
51    /* ==== マクロ定義 ==== */
52
53    /* IIC の割り込みマスクレベル設定値 */
54    /* システムにより変更してください */
55    #define D_IIC_EEP_INT_LEVEL     2
56
57    /* Memory address size */
58    #define D_IIC_EEP_MEMADR_SIZE   2
59
60    /* Write Polling Count 10ms (400Kbps base) */
61    #define D_IIC_WCT_CNT    480
62
```

```
63    /* I2C バス確認用のタイムアウト時間 ※スレーブデバイス状態を確認し続ける時間をセットください。*/
64    #define D_IIC_BUSYCHECK_TMOUT   150000
65
66    /* Device Code */
67    #define D_IIC_DEV_CODE   0xA0
68
69    /* Write code */
70    #define D_IIC_W_CODE     0x00
71
72    /* Read code */
73    #define D_IIC_R_CODE     0x01
74
75    /* 戻り値 */
76    typedef enum
77    {
78        D_IIC_EEP_NG = -1,
79        D_IIC_EEP_OK = 0,
80        D_IIC_EEP_NOP,
81        D_IIC_EEP_READING,
82        D_IIC_EEP_WRITING,
83        D_IIC_EEP_BUS_BUSY,
84
85    } E_Iic_eep_Ret;
86
87    /* NACK 判定用 戻り値 */
88    typedef enum
89    {
90        D_IIC_NACK_FOUND = 0,
91        D_IIC_NACK_NOTFOUND
92    } E_Iic_eep_Nack;
93
94    /* EEPROM アクセス状態 */
95    typedef enum
96    {
97       D_IIC_EEP_NO_INIT,               /* 未初期化状態 */
98       D_IIC_EEP_IDLE,                  /* アイドル状態 */
99       D_IIC_EEP_START_ISSUE_WAIT,      /* 開始条件発行待ち状態 */
100      D_IIC_EEP_SEND_DEVADD_WAIT,      /* Device Address 送信待ち状態 */
101      D_IIC_EEP_RESEND_DEVADD_WAIT,    /* Device Address 再送信待ち状態 */
102      D_IIC_EEP_SEND_MEMADD_WAIT,      /* Memory address 送信待ち状態 */
103      D_IIC_EEP_SEND_RCV_DATA_WAIT,    /* データ送受信待ち状態 */
104      D_IIC_EEP_WRITE_END_ISSUE_WAIT,  /* Write 完了後 停止条件発行待ち状態 */
105      D_IIC_EEP_WRITE_POLLING_WAIT,    /* WritePolling 待ち状態 */
106      D_IIC_EEP_END_ISSUE_WAIT,        /* 停止条件発行待ち状態 */
107      D_IIC_EEP_CONDITION_MAX          /* ここ以降定義禁止 */
108
109   } E_Iic_eep_condition;
110
111   /* EEPROM アクセスイベント */
112   typedef enum
113   {
114      D_IIC_EEP_EV_INIT,               /* iic_user_Init()コール時 */
115      D_IIC_EEP_EV_RW_START,           /* iic_user_EepRomRW()コール時 */
116      D_IIC_EEP_EV_CHECK,              /* iic_user_Chk_EepRom()コール時 */
117      D_IIC_EEP_EV_INT_NAKI,           /* NAKI 割り込み発生時 */
118      D_IIC_EEP_EV_INT_TXI,        /* TXI 割り込み発生時 */
119      D_IIC_EEP_EV_INT_TEI,        /* TEI 割り込み発生時 */
120      D_IIC_EEP_EV_INT_RXI,        /* RXI 割り込み発生時 */
121      D_IIC_EEP_EV_INT_STPI,           /* STPI 割り込み発生時 */
122      D_IIC_EEP_EVENT_MAX              /* ここ以降定義禁止 */
123
124   } E_Iic_eep_event;
125
126   /* EEPROM アクセスエラー状態 */
```

```
127   typedef enum
128   {
129       D_IIC_EEP_ERR_NO = 0,            /* エラーなし状態 */
130       D_IIC_EEP_ERR_NACK,                  /* 送信時 NACK 受信 */
131       D_IIC_EEP_ERR_AL,                /* AL 発生時 */
132       D_IIC_EEP_ERR_WCT_OVER,              /* Write cycle time over */
133       D_IIC_EEP_ERR_OTHER,             /* その他エラー発生 */
134
135   } E_Iic_eep_err_condition;
136
137   /* EEPROM 動作モード */
138   typedef enum
139   {
140       D_IIC_EEP_NO_MODE = 0,           /* mode なし */
141       D_IIC_EEP_WRITE,             /* Write モード */
142       D_IIC_EEP_READ,                  /* Random Read Operation と Sequential Read Operation 組み合わ
143   せモード */
144       D_IIC_EEP_CURRENT_READ           /* Current Address Read と Sequential Read Operation 組み合わ
145   せモード */
146
147   } E_Iic_eep_mode;
148
149   /** 状態遷移表構造体 */
150   typedef struct t_cbs_mtx_ev_tbl
151   {
152      E_Iic_eep_condition  event_type;         /* 受信イベント      */
153      E_Iic_eep_Ret  (*proc)( void *);         /* ハンドラ          */
154   } T_IIC_EEPROM_MTX_TBL ;
155
156   /* EEPROM リード/ライト情報 */
157   typedef struct {
158
159       E_Iic_eep_mode           i_mode;         /* mode */
160       unsigned char            i_DevAdr;       /* Device address */
161       unsigned long            i_RomAdr;       /* EEPROM reading address */
162       unsigned long            i_Len;          /* Transfer/Receive data length */
163       unsigned char            *i_pBuf;        /* Transfer/Receive data buffer pointer */
164
165   } T_IIC_EEPROM_RW_INFO;
166
167   /* EEPROM リード/ライトデータ情報 */
168   typedef struct {
169
170       unsigned long            i_DataLen;      /* R/W Data length */
171       unsigned char            *i_DataBuf;     /* R/W Data Buffer pointer */
172
173   } T_IIC_EEPROM_RW_DATA_INFO;
174
175   /* EEPROM リード/ライト管理情報 */
176   typedef struct {
177
178       E_Iic_eep_mode               i_mode;                             /* mode */
179       unsigned char                i_DevAdr;                           /* Device address */
180       unsigned long                i_MemAdrLen;                        /* Memory address length */
181       unsigned char                i_MemAdrBuf[D_IIC_EEP_MEMADR_SIZE];   /* Memory address buffer */
182       T_IIC_EEPROM_RW_DATA_INFO    i_RW_Data_info;                     /* RW Data Info */
183       unsigned short               i_WCTCnt;                           /* Write cycle time waiting
184   counter */
185
186   } T_IIC_EEPROM_RW_MANAGE;
187
188   /* EEPROM 状態/エラー状態情報 */
189   typedef struct {
190
```

```
191       E_Iic_eep_condition         i_eep_condition;   /* EEPROM 状態 */
192       E_Iic_eep_err_condition     i_err_condition;   /* EEPROM エラー状態 */
193
194   } T_IIC_EEPROM_CONDITION;
195
196
197   /* ==== 関数宣言 ==== */
198   E_Iic_eep_Ret iic_mtx_executeFuncTable(E_Iic_eep_event event, void *info);
199
200   /* ==== User 提供 IF ==== */
201   E_Iic_eep_Ret iic_user_Init(void);
202   E_Iic_eep_Ret iic_user_EepRomRW(T_IIC_EEPROM_RW_INFO *i_RW_Info);
203   E_Iic_eep_Ret iic_user_Chk_EepRom(T_IIC_EEPROM_CONDITION *o_condition_info);
204   E_Iic_eep_Ret iic_user_int_select(void);
205
206
207   #endif /* __IIC_EEPROM_DEF_H__ */
```

(5) サンプルプログラムリスト "intprg.c"
IIC 割り込み処理関数を定義しています。

```
1
2       …途中省略…
3
4       /* H'E60 IIC0 IICI0 */
5       void INT_IIC0_IICI0(void)
6       {
7           unsigned short   dummy;
8
9           /* IIC0 割り込み選択処理 */
10          iic_user_int_select();
11
12          /* 更新したはずの割り込み要因による割り込みの受け付けを避ける対応 */
13          dummy = IIC0.ICSR.BYTE;
14          int_responstime_wait(INTC_RESPONSEWAIT);  /* 優先順位判定時間分待ち */
15
16      }
17
18      …途中省略…
```

(6) サンプルプログラムリスト "vecttbl.src"
IIC 割り込み処理実行時の割り込み優先度を設定しています。
IIC 割り込みの優先度を 1 に設定しているため、IIC 割り込み処理中に新たな IIC 割り込み処理が発生しないように優先度に 1 を設定しています。

```
1
2       …途中省略…
3
4       ;IIC0
5                       ;H'E60 IIC0 IICI0
6                       .data.b     H'10
7
8       …途中省略…
```

6. 実行結果

上記サンプルプログラムの EEPROM へのライト/リードの実行結果については、ライトデータとリードデータを比較処理するループを抜けているため、EEPROM へライトしたデータが EEPROM からリードできていることが確認できます。

また、図 37 のように、提供インタフェースの EEPROM リード/ライト処理 (iic0_user_EepRomRW( )) でパラメータ指定したリードデータを格納する領域 (gEEPROM_Read_Data[D_IIC_EEPROM_READ_DATA_NUM]) を High-performance Embedded Workshop で出力しても EEPROM へライトしたデータが EEPROM からリードできていることが確認できます。

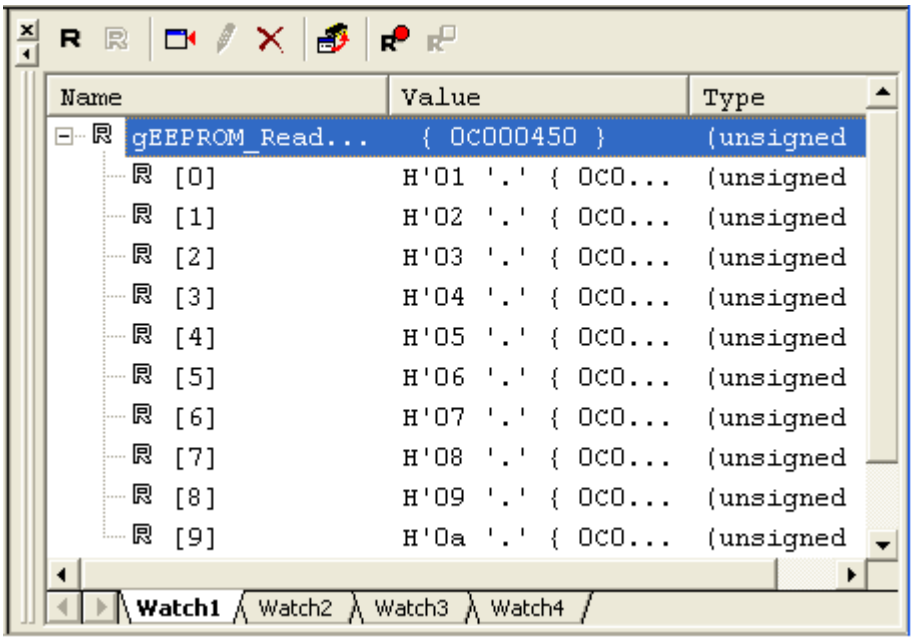


図 37　gEEPROM_Read_Data[D_IIC_EEPROM_READ_DATA_NUM]のメモリダンプ

7. 参考ドキュメント

- ソフトウェアマニュアル
  SH-4A ソフトウェアマニュアル (RJJ09B0090)
  (最新版をルネサス エレクトロニクスホームページから入手してください。)
- ハードウェアマニュアル
  SH7730 グループ ハードウェアマニュアル (RJJ09B0339)
  (最新版をルネサス エレクトロニクスホームページから入手してください。)

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.09.06	—	初版発行

ご注意書き

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

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

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

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

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

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

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0