SH7722/SH7731 グループ

IIC 送受信制御例 (EEPROM ライト・リード)

RJJ06B1187-0100
Rev.1.00
2010.09.15

要旨

本アプリケーションノートは、SH7722/SH7731 の I^2C バスインタフェース (IIC) を使用して、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

SH7722, SH7731 (※以降表記は SH7722 のみとする)

目次

1. はじめに

1.1 仕様

- マスタデバイスを SH7722、スレーブデバイスを EEPROM として、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 SH7722、スレーブデバイスを EEPROM として、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 64kbit 品 (R1EX24064ASAS0I) を使用します。
- 転送レートは 397.6kHz に設定しています。

1.2 使用機能

- IIC

1.3 適用条件

- 評価ボード: ルネサス エレクトロニクス製 SH7722 リファレンスプラットフォーム
  型番 R0P7722TH001ARK
  外付けメモリ (エリア 0): NOR 型フラッシュメモリ 64M バイト
  Spansion 製 S29GL512N10FF1020
  (エリア 3): SDRAM 64M バイト
  Micron 製 MT48LC8M16A2B475
- マイコン: SH7722 (R8A77220AC266BGV)
- 動作周波数:
  CPU クロック: 266.66MHz
  SH バスクロック: 133.33MHz
  U メモリクロック: 133.33MHz
  バスクロック: 66.66MHz
  SDRAM 用クロック: 106.66MHz
  周辺クロック: 33.33MHz
- エリア 0 バス幅: 16 ビット (MD3 端子 = Low レベル)
- エリア 3 バス幅: 64 ビット
- クロック動作モード: モード 0 (MD0, MD1 端子 = Low レベル)
- エンディアン: リトルエンディアン (MD5 端子 = High 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 での設定
  (-cpu=sh4aldsp -endian=little -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 関連アプリケーションノート

本資料の参考プログラムは, 「SH7722/SH7731 グループ アプリケーションノート SH7722/SH7731 初期設定例 (RJJ06B1090)」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 の概要

本アプリケーションノートでは、I²C バスの規格説明については省略します。詳細は、NXP 社の I²C 関連資料、もしくは、「SH7730 グループ アプリケーションノート SH7730 IIC シングルマスタ送受信制御例 (RJJ06B1057) 2 章 I²C バスの概要」を参照ください。

2.1 SH7722 の I²C バスインタフェース (IIC) 特長

SH7722 I²C モジュールはシングルマスタバスにのみ対応します。本モジュールは常にマスタとなります。スレーブ機能はありません。このため、データ転送中のアービトレーションロスト時はバスを解放して停止します。

ブロック図については「SH7722 ハードウェアマニュアル (RJJ09B0324)」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　SH7722 の IIC 特長

項目	概要
チャネル数	1 チャネル
モード	マスタモードのみをサポート ※SH7722 I²C モジュールはシングルマスタバスにのみ対応します。本モジュールは常にマスタとなります。スレーブ機能はありません。
開始条件/停止条件	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
フォーマット	I²C フォーマットに準拠
ウェイト機能	アクノリッジを除くデータ転送後、SCL をローレベルにしてウェイト状態にすることが可能です。 割り込みフラグをクリアすることでウェイト状態からの解除が可能です。
割り込み要因	• 割り込み要因: 4 種類 ① データ送信イネーブル ② ウェイト状態 ③ 非アクノリッジ検出 ④ アービトレーションロスト (バス競合を検出すると、バスを解放して停止します。)
データ転送速度	標準モード (100kHz) および高速モード (400kHz) に対応します。 クロックコントロールレジスタの設定により SCL クロックを任意に設定可能です。
SCL ラインのクロック同期	SCL ラインのクロック同期処理が可能です。 SCL がハイカウント期間に発生するハザード (スパイクノイズ) は、アービトレーションロストとして検出されます。

2.2 SH7722 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、「SH7722 ハードウェアマニュアル (RJJ09B0324)」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

本アプリケーションノートでは、EEPROM の使用方法については省略します。

EEPROM の使用方法については「R1EX24064ASAS0I/R1EX24064ATAS0I Two-wire serial interface 64k EEPROM (8-kword × 8-bit) (RJJ03C0280)」、もしくは、「SH7730 グループ アプリケーションノート IIC シングルマスタ送受信制御例 (RJJ06B1057) EEPROM 使用方法について」の章を参照ください。

4. 応用例の説明

本応用例では I²C バスインタフェース (IIC) を使用し、マスタデバイスの SH7722 からスレーブデバイスの EEPROM にライト・リードを行います。

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、EEPROM よりデータ{0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a}をリードします。

4.1 本応用例の動作環境

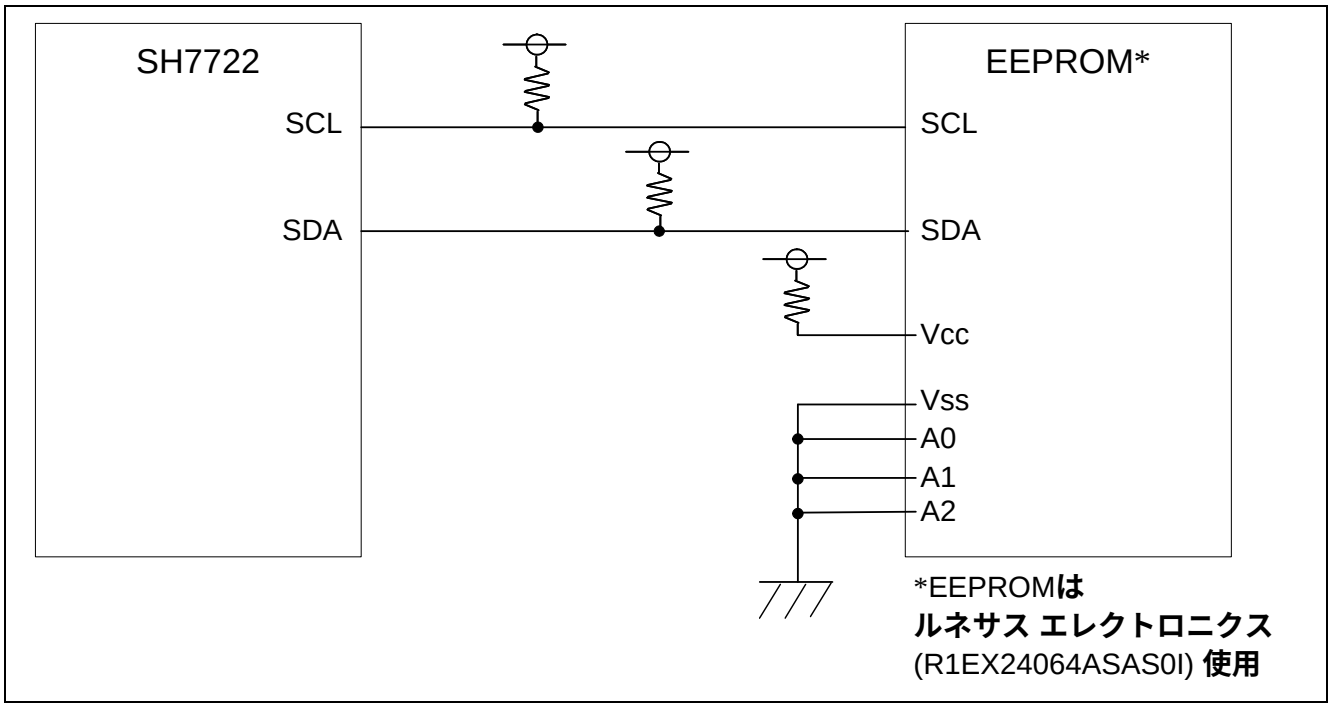

図 1　本応用例の動作環境

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。

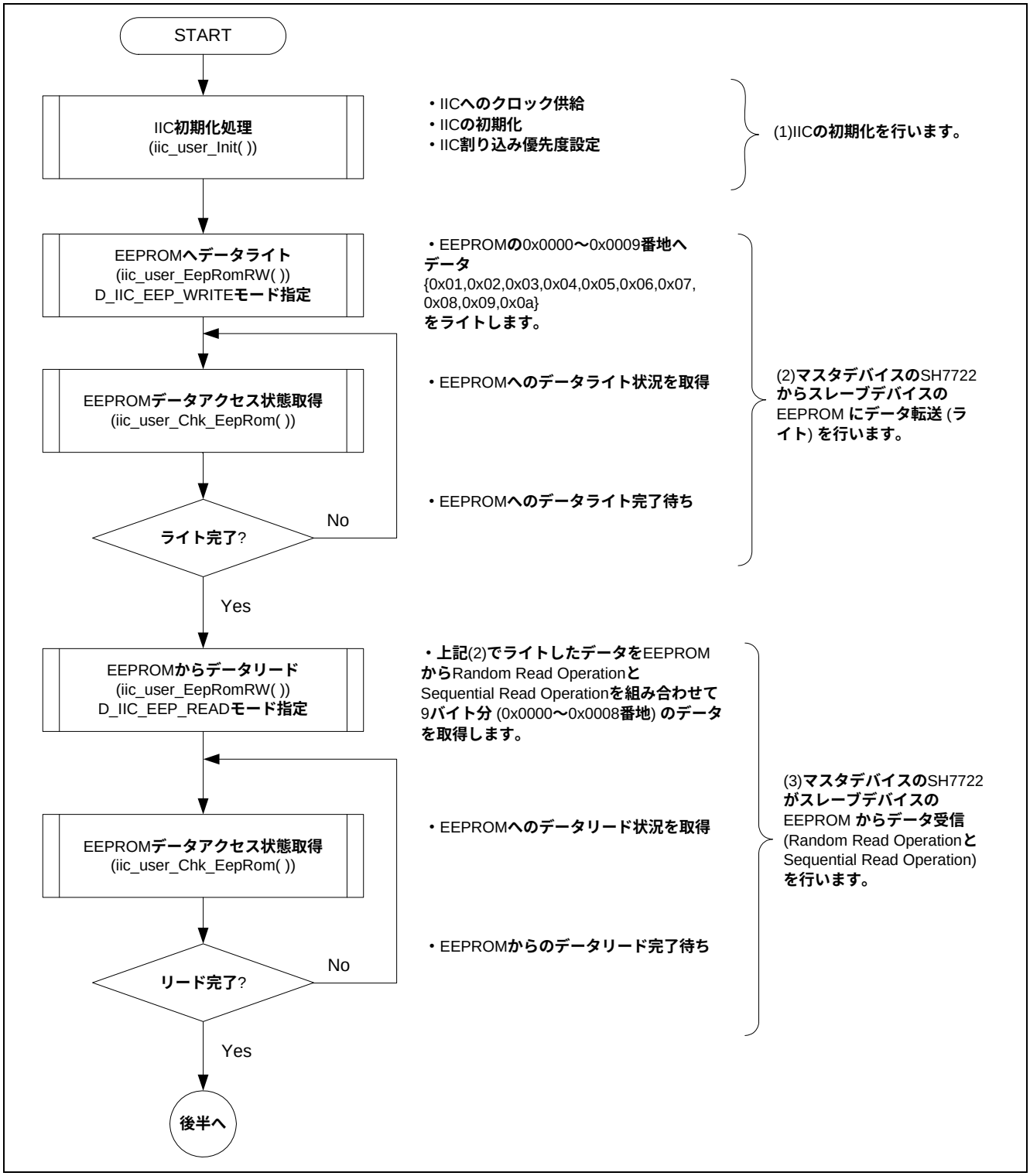

図 2　本応用例の処理フロー (前半)

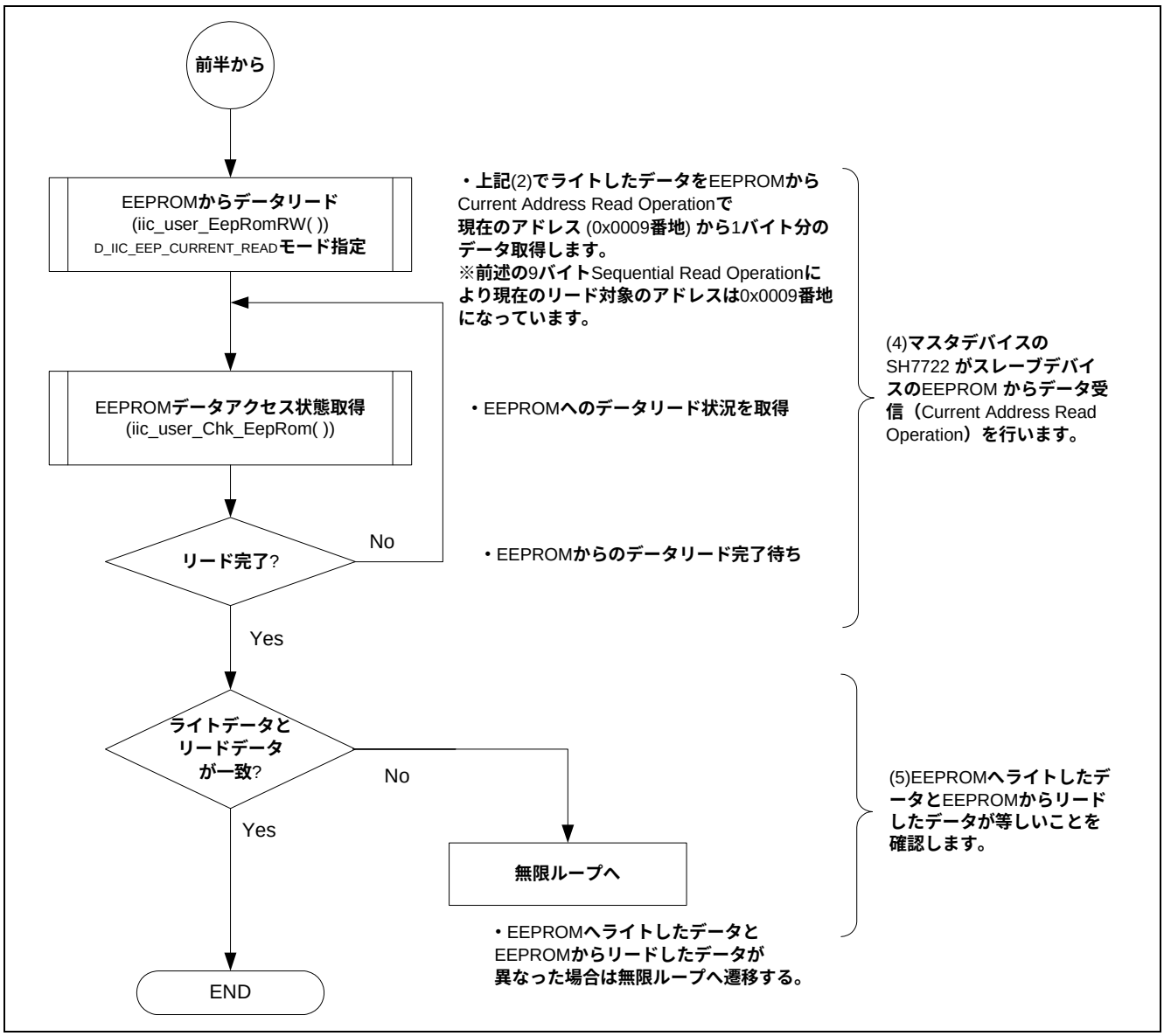

図 3　本応用例の処理フロー (後半)

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、以下のユーザ提供インタフェースがあります。

表 2　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	AL 割り込み処理 (iic_user_int_AL( ))	AL 割り込みハンドラに実装します。 現状態に対応する AL 割り込み発生時の処理を行います。
5	TACK 割り込み処理 (iic_user_int_TACK( ))	TACK 割り込みハンドラに実装します。 現状態に対応する TACK 割り込み発生時の処理を行います。
6	WAIT 割り込み処理 (iic_user_int_WAIT( ))	WAIT 割り込みハンドラに実装します。 現状態に対応する WAIT 割り込み発生時の処理を行います。
7	DTE 割り込み処理 (iic_user_int_DTE( ))	DTE 割り込みハンドラに実装します。 現状態に対応する DTE 割り込み発生時の処理を行います。

【参考】

No1～3 のインタフェース詳細は、「SH7730 グループ アプリケーションノート SH7730 IIC シングルマスタ送受信制御例 (RJJ06B1057) 参考プログラムの提供インタフェースの章」を参照ください。

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 3　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、「R1EX24064ASAS0I/R1EX24064ATAS0I Two-wire serial interface 64k EEPROM (8-kword × 8-bit) (RJJ03C0280)」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 4　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、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	開始条件発行待ちをしている状態となります。 DTE 割り込み発生により、Device Address 送信準備をします。
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。 WAIT 割り込み発生により、Memory Address (上位) 送信準備をします。
C4	【D_IIC_EEP_RESEND_DEVADD_WAIT】 Device Address 再送信待ち状態	D_IIC_EEP_READ モード時のみ使用します。 再送条件 (ReStart) 発行後、リード用の Device Address 送信完了待ちをしている状態となります。
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。 WAIT 割り込み発生により、Memory Address (下位) 送信準備 or 1st データ送信準備をします。
C6	【D_IIC_EEP_SEND_RCV_DATA_WAIT】 データ送受信待ち状態	1 バイト単位のデータ送信 or 受信完了待ちをしている状態となります。 Write 処理の際には、次データ送信準備 or WritePolling 準備をします。 Read 処理の際には、データ受信待ちをしている状態として使用されます。
C7	【D_IIC_EEP_WRITE_POLLING_WAIT】 Write Polling 待ち状態	D_IIC_EEP_WRITE モード時のみ使用します。 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。 Write Polling 完了後、停止条件を発行し、アイドル状態に戻ります。

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 5　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_AL】	AL 割り込み発生
EV4	【D_IIC_EEP_EV_INT_TACK】	TACK 割り込み発生
EV5	【D_IIC_EEP_EV_INT_WAIT】	WAIT 割り込み発生
EV6	【D_IIC_EEP_EV_INT_DTE】	DTE 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 6　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。ソフトウェアの仕様に応じて、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～EV6) については、4.6 章 表 5 参考プログラムのイベント定義を参照ください。
- EV2 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 SH7722 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。

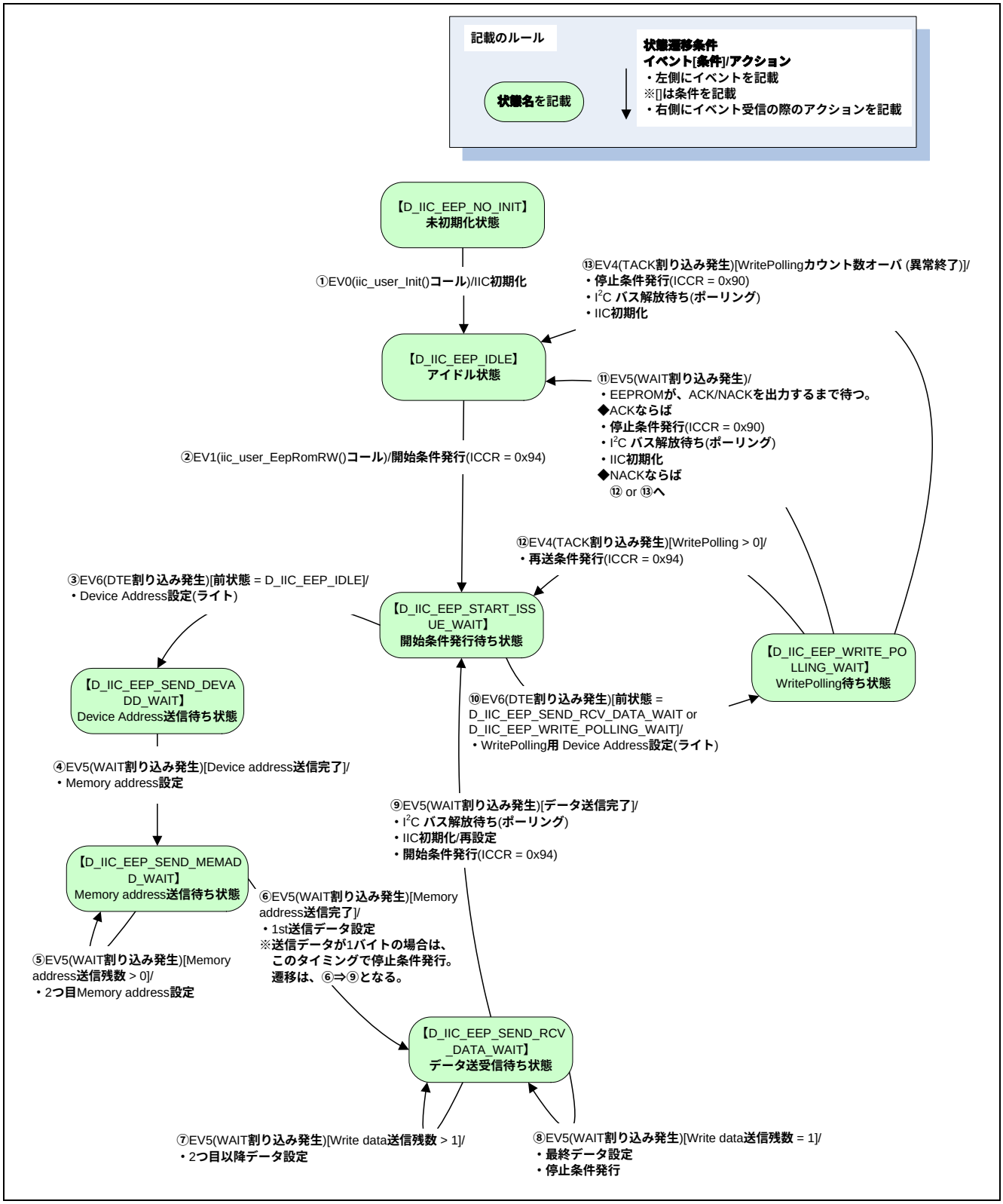

図 4　参考プログラムの状態遷移図 (D_IIC_EEP_WRITE モード)

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 SH7722 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。

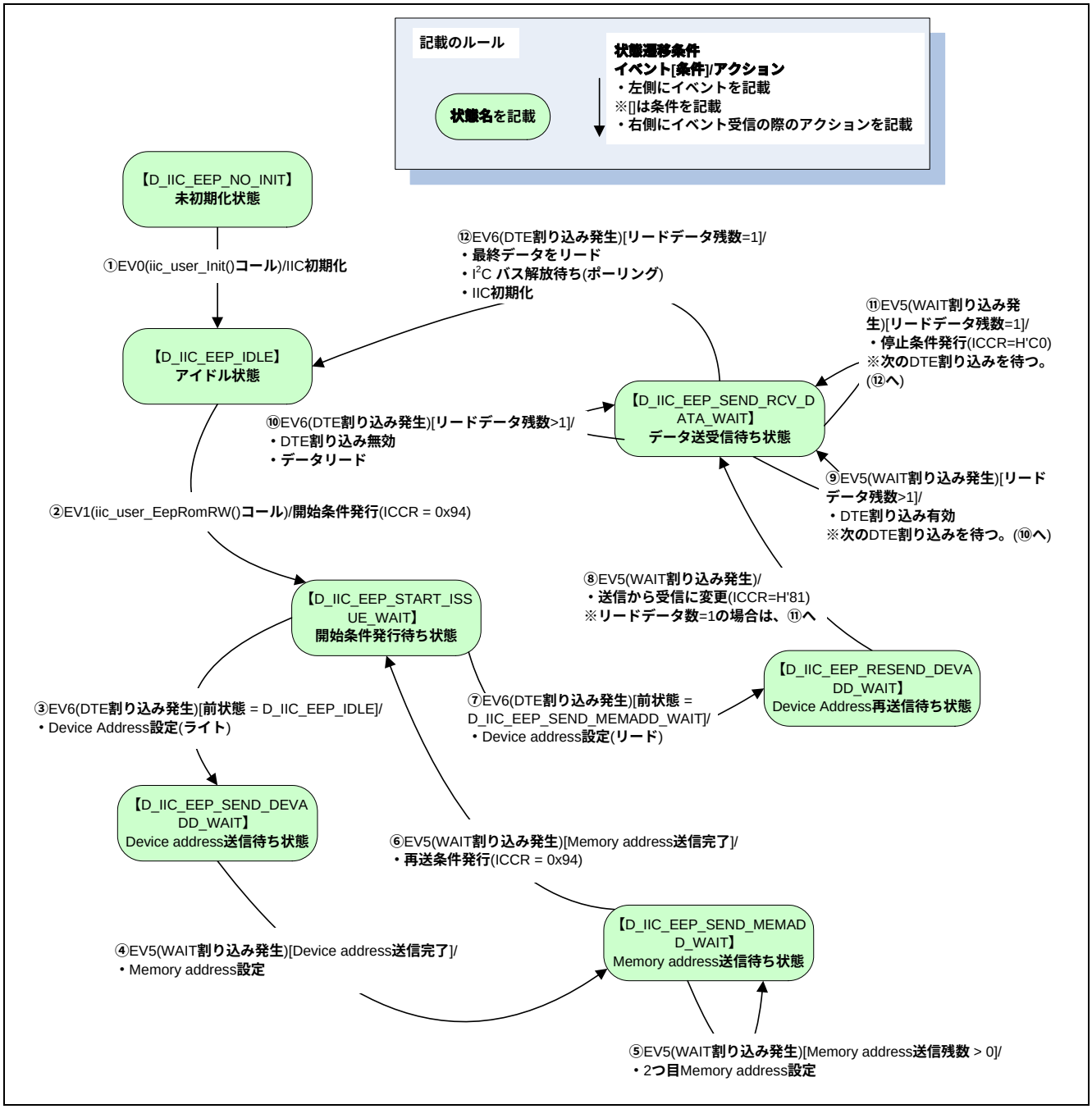

図 5　参考プログラムの状態遷移図 (D_IIC_EEP_READ モード)

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 SH7722 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。

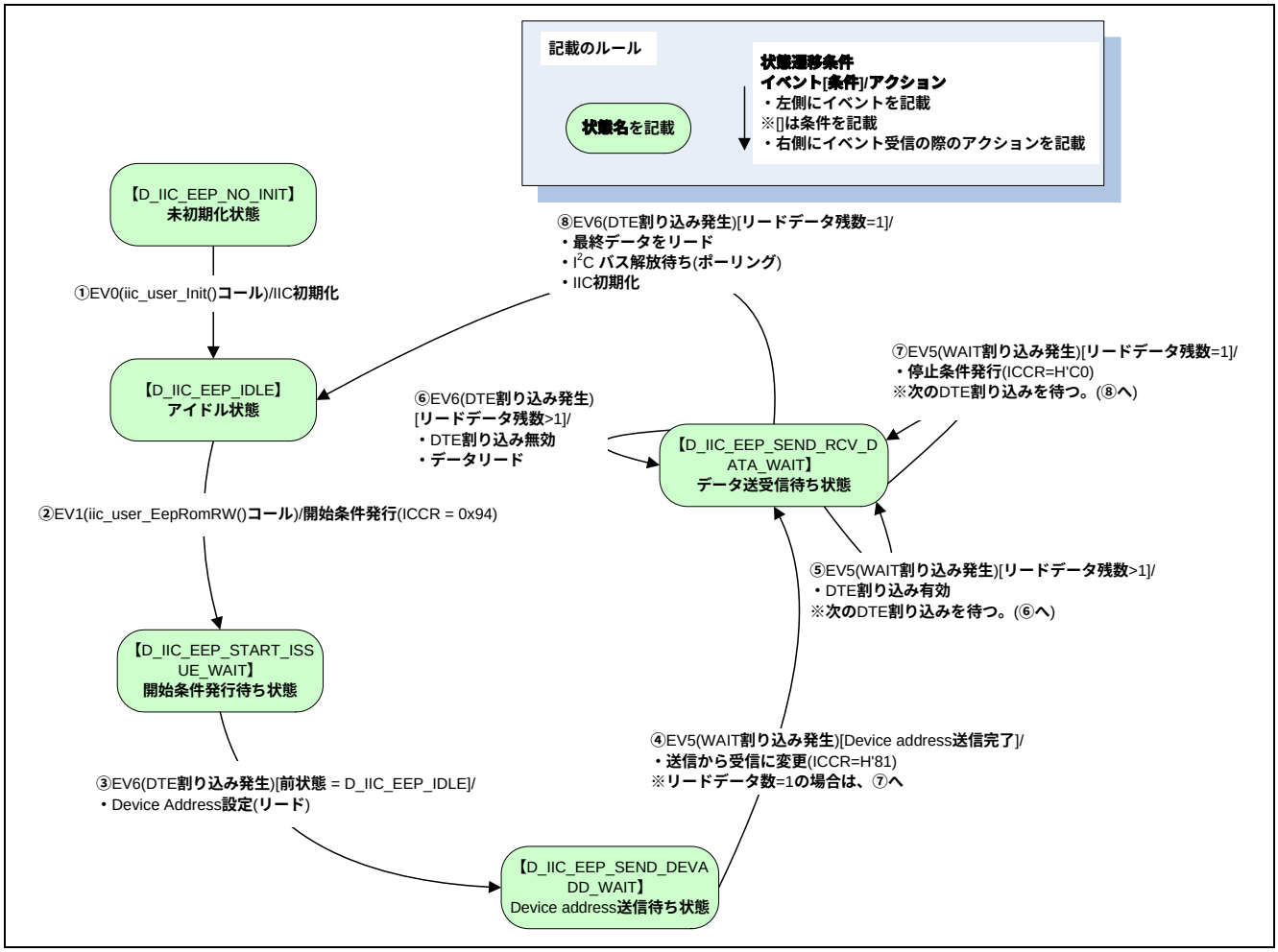

図 6　参考プログラムの状態遷移図 (D_IIC_EEP_CURRENT_READ モード)

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 4 の各状態で、4.6 章 表 5 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 4 の ID を参照ください。EV0～については、4.6 章 表 5 の ID を参照ください。また、Func1～の処理については、表 8 状態遷移表登録関数を参照ください。

表 7　状態遷移表 (g_Iic_Eep_event_tbl[][])

状態＼イベント		EV0	EV1	EV2	EV3	EV4	EV5	EV6
C0	【D_IIC_EEP_NO_INIT】 未初期化状態	Func1	NOP	NOP	NOP	NOP	NOP	NOP
C1	【D_IIC_EEP_IDLE】 アイドル状態	Func1	Func2	Func3	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	Func6
C3	【D_IIC_EEP_SEND_DEVADD_WAIT】 Device Address 送信待ち状態	NOP	NOP	Func3	Func4	Func5	Func7	NOP
C4	【D_IIC_EEP_RESEND_DEVADD_WAIT】 Device Address 再送信待ち状態	NOP	NOP	Func3	Func4	Func5	Func8	NOP
C5	【D_IIC_EEP_SEND_MEMADD_WAIT】 Memory address 送信待ち状態	NOP	NOP	Func3	Func4	Func5	Func9	NOP
C6	【D_IIC_EEP_SEND_RCV_DATA_WAIT】 データ送受信待ち状態	NOP	NOP	Func3	Func4	Func5	Func10	Func11
C7	【D_IIC_EEP_WRITE_POLLING_WAIT】 Write Polling 待ち状態	NOP	NOP	Func3	Func4	Func12	Func13	NOP

【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 7 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 8　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_NACK_rcv ()	ライトモード NACK 受信時処理
Func6	iic_After_start_issue ()	開始条件発行後処理
Func7	Iic_Send_After_Dev_Address ()	初回 Device Address 送信後処理
Func8	iic_After_Re_DevAdd_send ()	Device Address 再送信後処理
Func9	iic_MemAdd_Sending ()	Memory address 送信中処理
Func10	iic_Data_send_rcving ()	データ送受信中処理
Func11	iic_Read_Data_rcv_DTE ()	DTE 割り込み時データ受信処理
Func12	iic_Eep_WritePolling_TACK ()	TACK 割り込み時 WritePolling 処理
Func13	iic_Eep_WritePolling_WAIT ()	WAIT 割り込み時 WritePolling 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 9　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_lic_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 10　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 4bit 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 1bit 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 1bit のみ使用
D_IIC_BUSYCHECK_TMOUT	150000	• I^2C バス解放状態監視タイマ 設定時間の間、I^2C バスが解放されているか監視します。本タイマがタイムアウトした場合は、スレーブデバイスの暴走など (I^2C ラインを Low に引っ張り続けている等) の異常状態と判定します。
D_IIC_EEPROM_tAA_WAIT_CNT	100	• EEPROM が確実に ACK/NACK を応答してくるまでのウェイト時間 ※本応用例では、EEPROM スペック tAA (max900ns) 以上のウェイト時間を設定します。

4.13　参考プログラムの ICCR 設定タイミングについて

4.13.1　マスタ送信時

(1) 停止条件

停止条件を発行、生成する場合は、ICDR に最終データを書き込み後、ICCR に H'90 を書き込みます。停止条件を発行後、停止条件生成までの間、DTE は 1 にセットされません。最終データを送信後に停止条件を生成して停止します。

ただし、ICDR 書き込みと ICCR 書き込みが遅延し、ACK ビットをまたいで処理した場合、停止条件生成前に DTE 割り込みが発生します。そのため、ICCR レジスタに H'90 を書き込み後に DTE 割り込みを禁止設定するか、ACK ビット生成前に ICDR 書き込みと ICCR 書き込みを行う必要があります。

(2) 再送条件

再送条件を発行、生成する場合は、ICDR に最終データを書き込み後、ICCR に H'94 を書き込みます。再送条件を発行後、再送条件生成までの間、DTE は 1 にセットされません。最終データを送信後に再送条件を生成します。

ただし、ICDR 書き込みと ICCR 書き込みが遅延し、ACK ビットをまたいで処理した場合、再送条件生成前に DTE 割り込みが発生し、ソフト側で正常に通信するための制御ができなくなります。そのため、ACK ビット生成前に ICDR 書き込みと ICCR 書き込みを行う必要があります。

(3) マスタ送信からマスタ受信に変更する場合

送信から受信に変更する場合は、最終データ送信完了までに ICCR へ H'81 を書き込んでください。このとき ICCR への書き込みが遅れる場合は、WAIT 割り込みを使用して最終データ送信完了までに ICCR に書き込むようにする必要があります。

4.13.2　マスタ送信時の参考例

例として、全データ送信完了後の停止条件発行について考えます。

(1) 問題の現象発生例

図 7 の①のように、ICDR 書き込みと ICCR 書き込みが遅延し、ACK ビットをまたいで処理した場合、停止条件生成前に DTE 割り込みが発生します。DTE 割り込みで送信データを設定する制御をしていた場合、図 7 の②時点では、最終データを ICDR に設定済みであるのに、不要な DTE 割り込みが発生することになります。

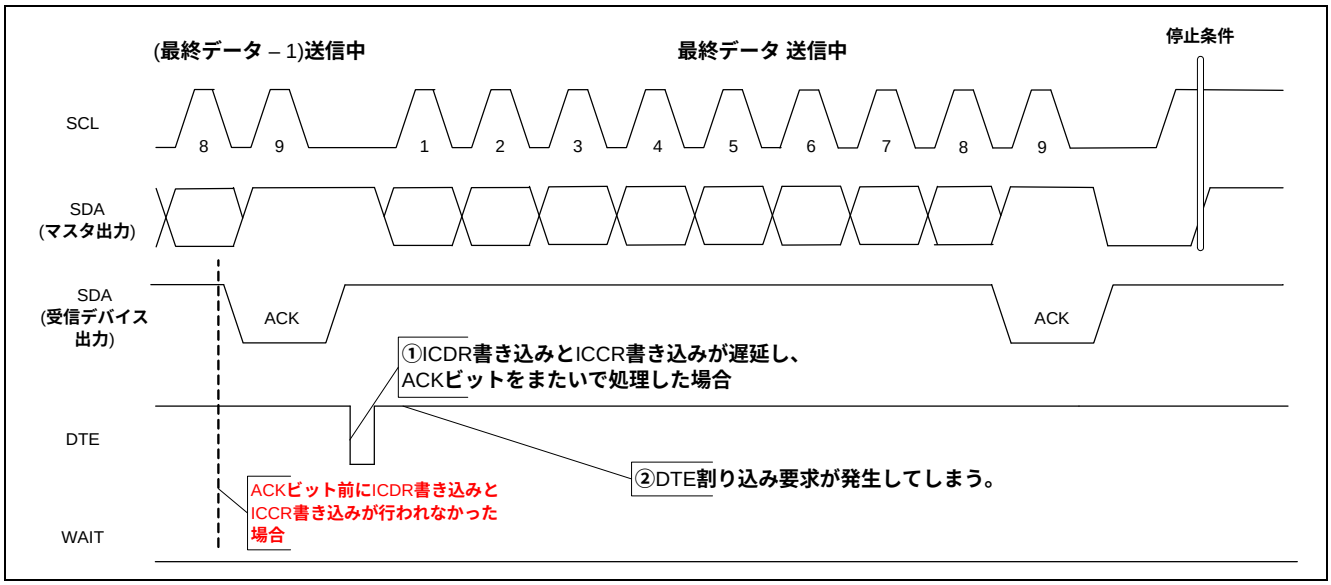

図 7　DTE 割り込みによる送信制御 (WAIT なし)

(2) 本応用例の対応

図 8 の①のように、最終データの 1 つ前のデータ送信時の WAIT 割り込みで、WAIT 割り込みの要因をクリアする前に、ICDR 書き込み (最終データ) と ICCR 書き込み (停止条件) を設定します。このことにより、ACK ビット生成前に ICDR 書き込みと ICCR 書き込みを行うことが保証でき、不要な DTE 割り込みが発生することはありません。

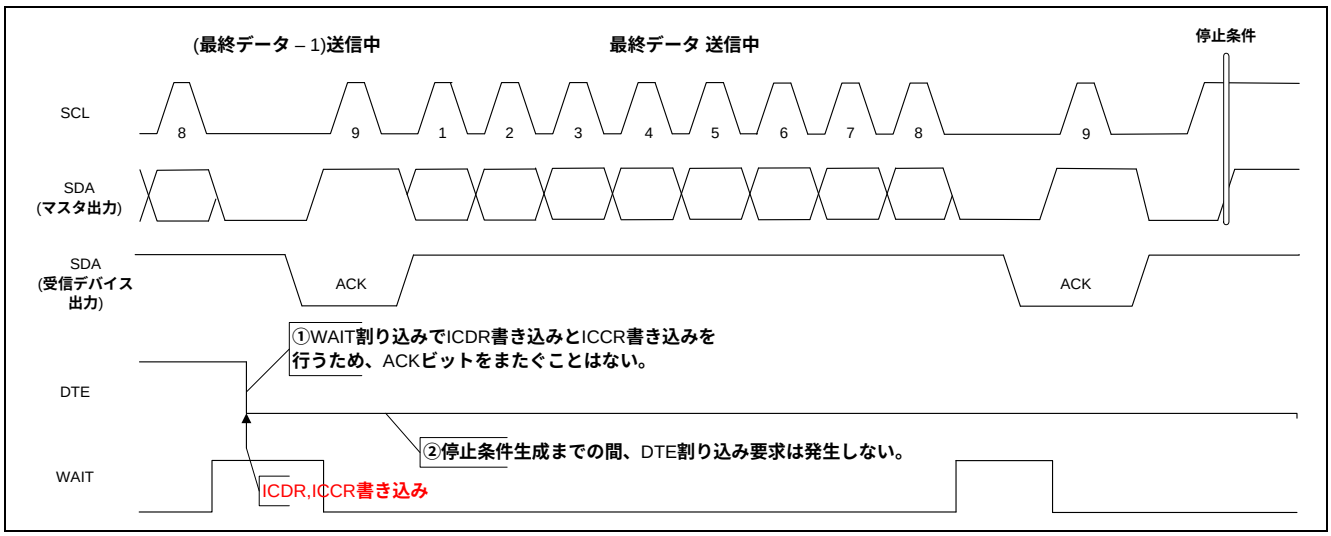

図 8　WAIT 割り込み使用による送信制御

4.13.3 マスタ受信時

(1) 停止条件

停止条件を発行、生成する場合は、ICDR からデータ読み出し後、ICCR に H'C0 を書き込みます。そして停止条件を発行後に最終データを受信完了し、ICDR からデータ読み出し後に停止条件を生成して停止します。

(2) 再送条件

再送条件を発行、生成する場合は、ICDR からデータ読み出し後、ICCR に H'D4 を書き込みます。そして再送条件を発行後に最終データを受信完了し、ICDR からデータ読み出し後に再送条件を生成します。

2 バイト以上の連続データを受信する場合、受信中再送、停止条件生成のために ICDR レジスタから最終データ 1 つ前のデータ読み出しと ICCR レジスタに H'D4、H'C0 の書き込みを行います。

最終データ 1 つ前の ICDR 読み出しと ICCR 書き込みが遅延し、最終データ受信時の NACK ビット生成までに ICDR 読み出しと ICCR 書き込み処理が終わらない場合、最終データ受信時に ACK ビットを生成してしまうため、IIC の通信プロトコルが守れなくなります。そのため、最終データ受信時の NACK ビット生成までに ICDR 読み出しと ICCR 書き込みを行う必要があります。

(3) マスタ受信からマスタ送信に変更する場合

再送条件を生成すると受信から送信に自動変更し送信より開始します。

4.13.4　マスタ受信時の参考例

例として、全データ受信完了後の停止条件発行について考えます。

(1) 問題の現象発生例

図 9 の①のように、最終データ 1 つ前の ICDR 読み出しと ICCR 書き込みが遅延し、最終データ受信時の NACK ビット生成までに ICDR 読み出しと ICCR 書き込み処理が終わらない場合、図 9 の②のように、最終データ受信時に ACK ビットを生成してしまうため、IIC の通信プロトコルが守れなくなります。

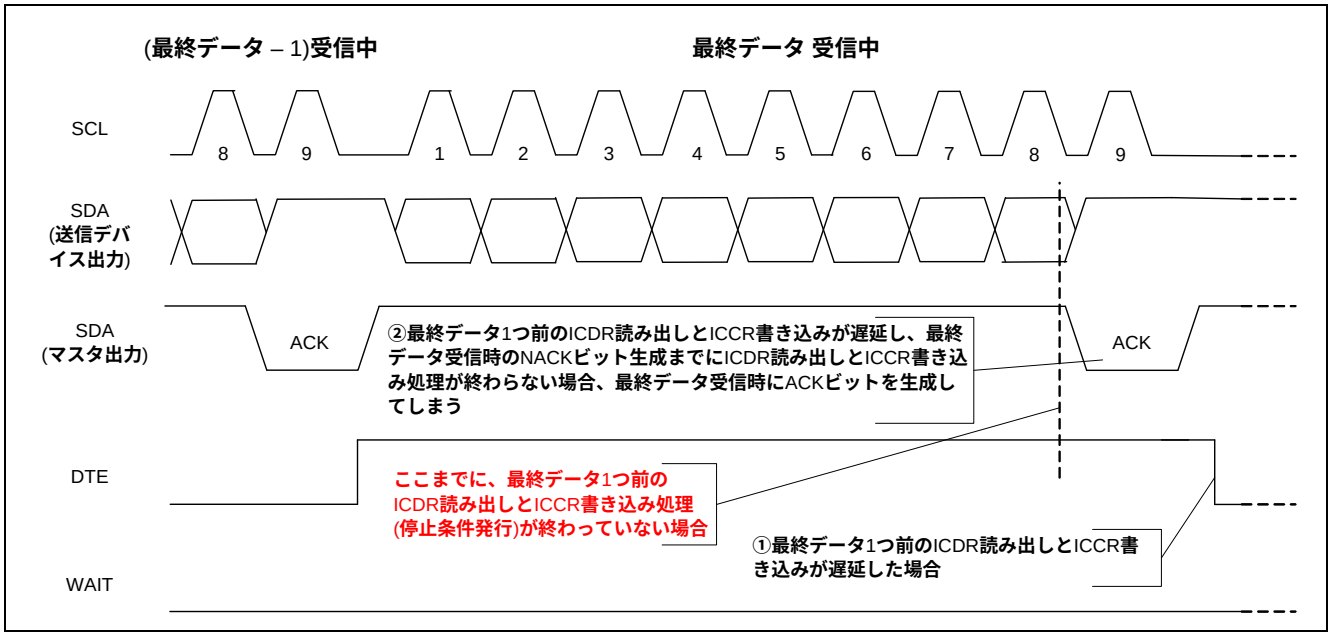

図 9　DTE 割り込みによる受信制御 (WAIT なし)

(2) 本応用例の対応

図 10 の②のように、最終データ受信時の WAIT 割り込みで、WAIT 割り込みの要因をクリアする前に、ICCR 書き込み (停止条件) を設定します。このことにより、9 クロック目前までに ICCR 書き込みを行うことが保証され、NACK ビットが生成できます。

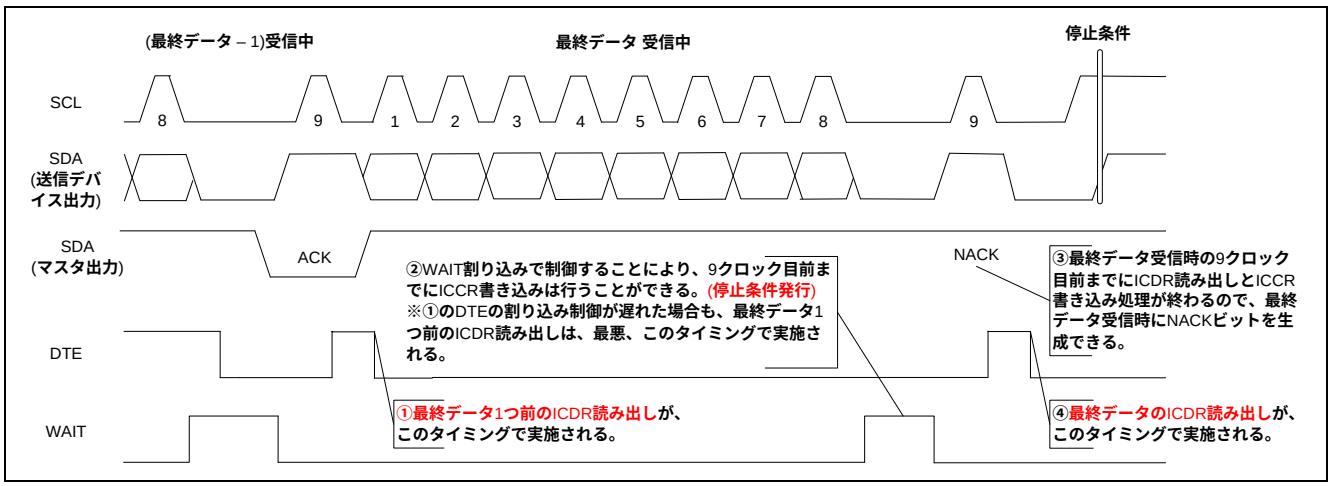

図 10　WAIT 割り込み使用による受信制御

4.14　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。

4.14.1　AL 発生時

本応用例では、アービトレーションロスト割り込み (ALE) を有効にし、IIC 通信中に AL が発生した場合には、AL 割り込みが発生するようにしています。

AL 割り込みは、TACK、WAIT、DTE 割り込みよりも優先度が高くなります。このため、次の状態に遷移し、AL 割り込み以外の割り込み (TACK、WAIT、DTE) が同時に発生していても、AL 割り込みが優先的に行われます。

【本応用例での対応例】

例として、開始条件発行で AL が発生した場合について考えます。

アイドル状態で、開始条件発行後の DTE 割り込みを待つ、開始条件発行待ち状態に遷移しますが、この開始条件発行で AL が発生した場合、DTE 割り込みにより起動される処理 (iic_After_start_issue( )) が実行されず、AL 割り込みで起動される AL 発生時処理 (iic_AL_generate ()) が優先して実行されます。

AL が発生した際に行う処理については、AL 発生時処理 (iic_AL_generate ()) に実装することを想定しています。

【注】 本応用例では、AL 発生時処理 (iic_AL_generate ()) 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。

4.14.2　スレーブデバイス (EEPROM) からの NACK 受信時

本応用例では、非アクノリッジ検出割り込み (TACK) を有効にし、IIC 送信中にスレーブデバイス (EEPROM) から NACK 受信した場合には、TACK 割り込みが発生するようにしています。

TACK 割り込みは、WAIT、DTE 割り込みよりも優先度が高くなります。このため、次の状態に遷移し、TACK 割り込み以外の割り込み (WAIT、DTE) が同時に発生していても、TACK 割り込みが優先的に行われます。

【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 送信でスレーブデバイス (EEPROM) から NACK 受信した場合、WAIT 割り込みにより起動される Memory address 送信中処理 (iic_MemAdd_Sending ( )) が実行されず、TACK 割り込みで起動されるライトモード NACK 受信時処理 (iic_NACK_rcv ()) が優先して実行されます。

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、ライトモード NACK 受信時処理 (iic_NACK_rcv ()) に実装することを想定しています。

【注】 本応用例では、ライトモード NACK 受信時処理 (iic_NACK_rcv ()) 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。

4.15　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。

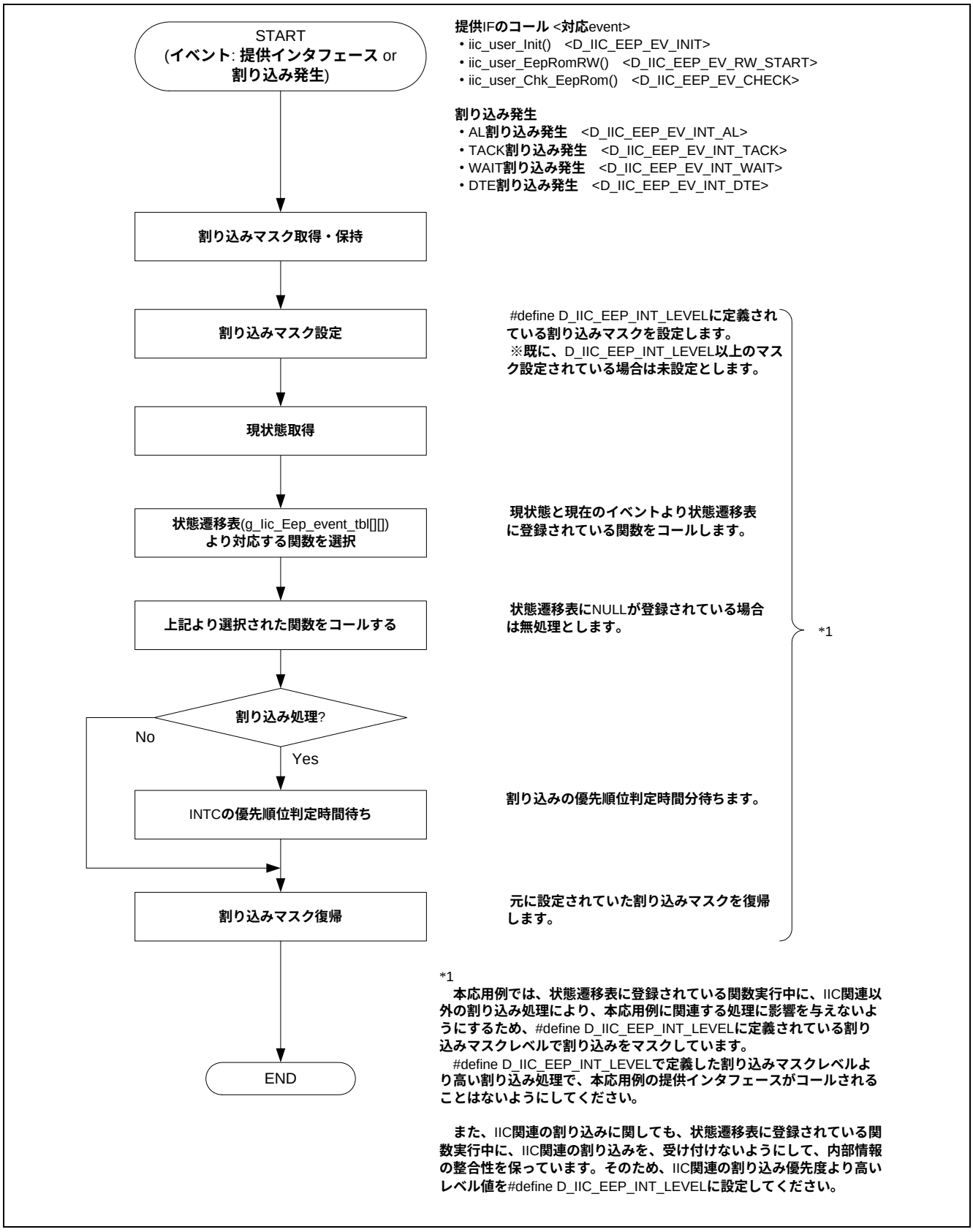

図 11　状態遷移表登録関数がコールされる仕組み

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 7 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

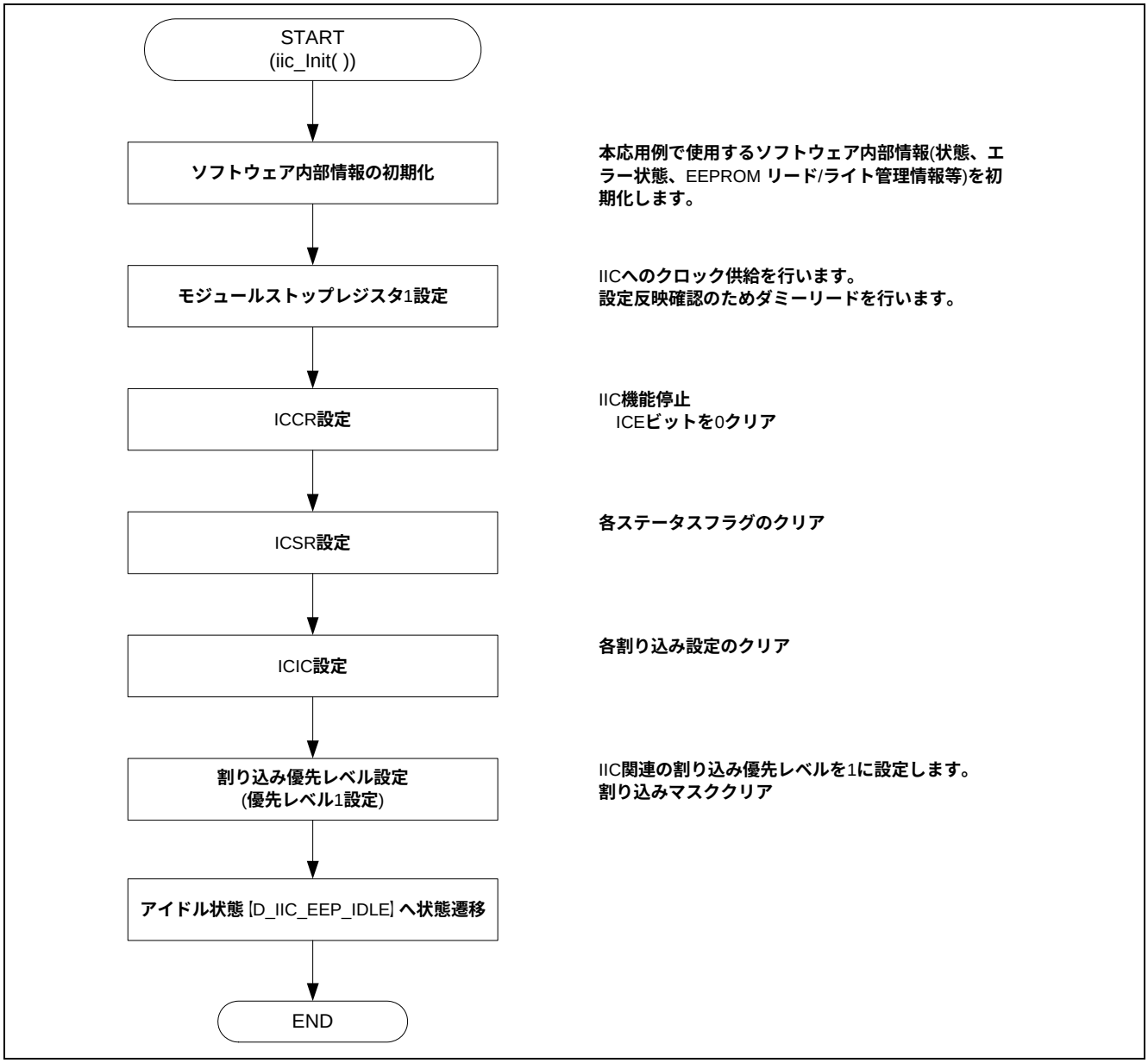

図 12　IIC 初期化処理

4.16.2　EEPROM リード/ライト処理

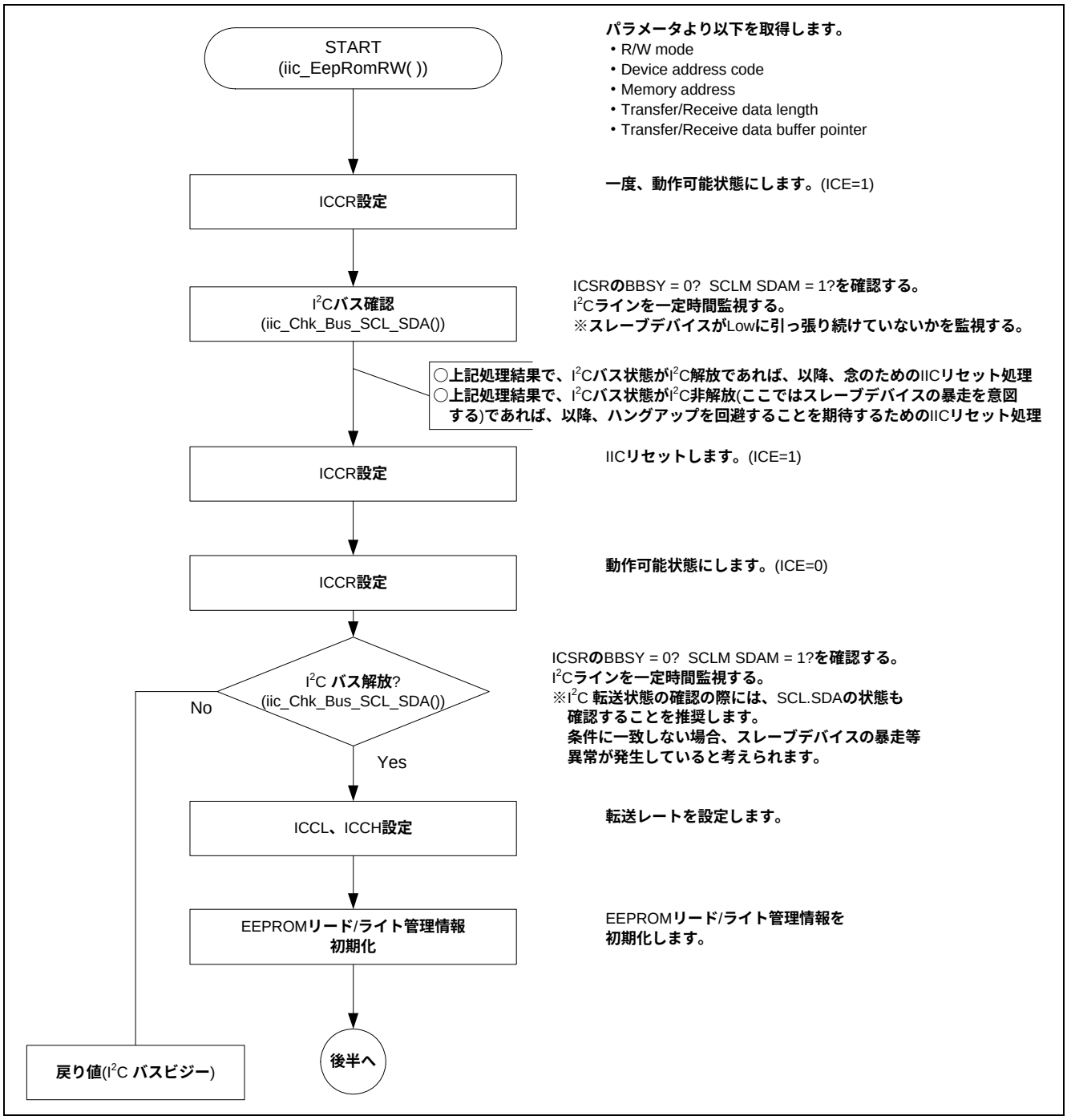

図 13　EEPROM リード/ライト処理 (前半)

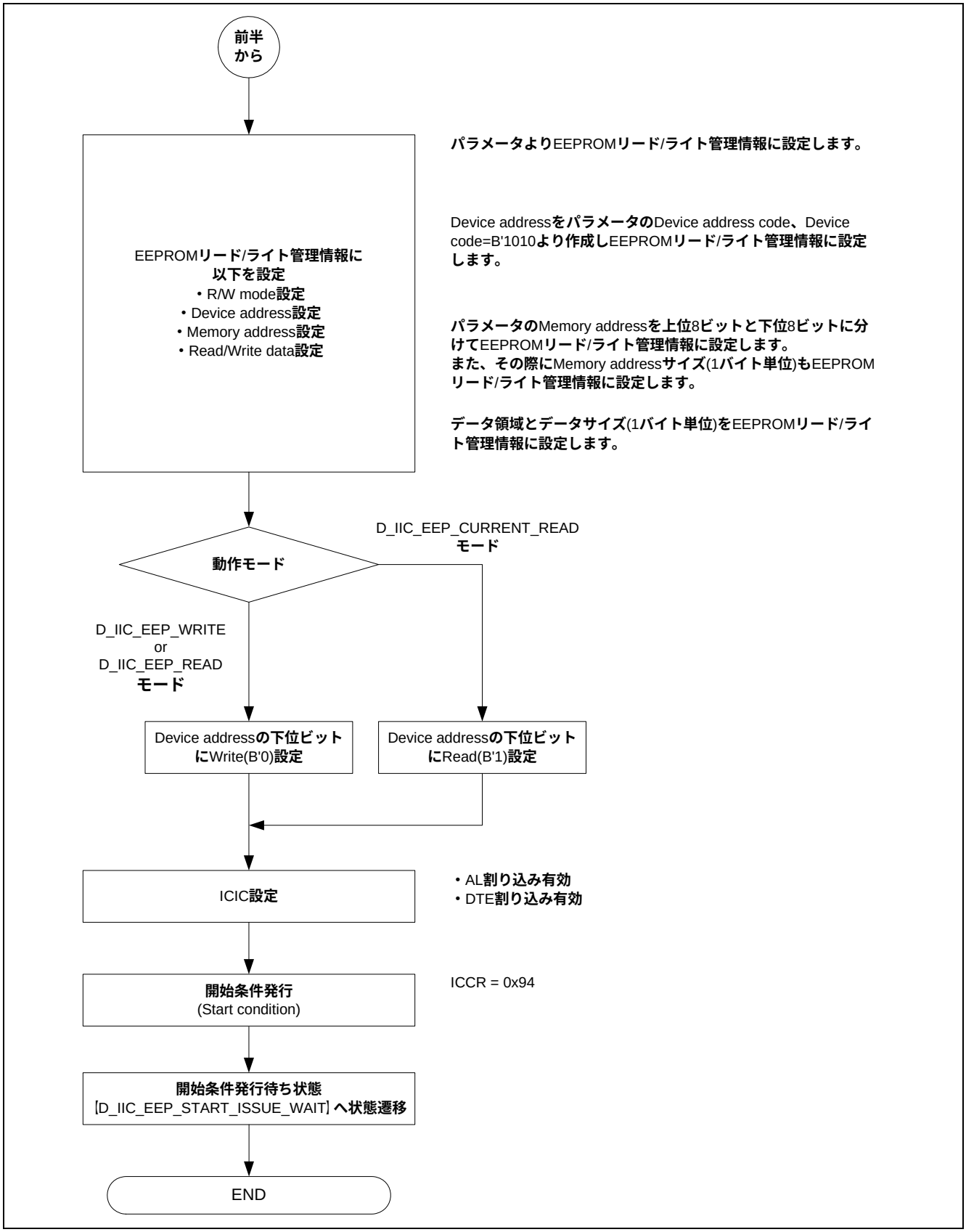

図 14　EEPROM リード/ライト処理 (後半)

4.16.3　EEPROM 状態取得処理

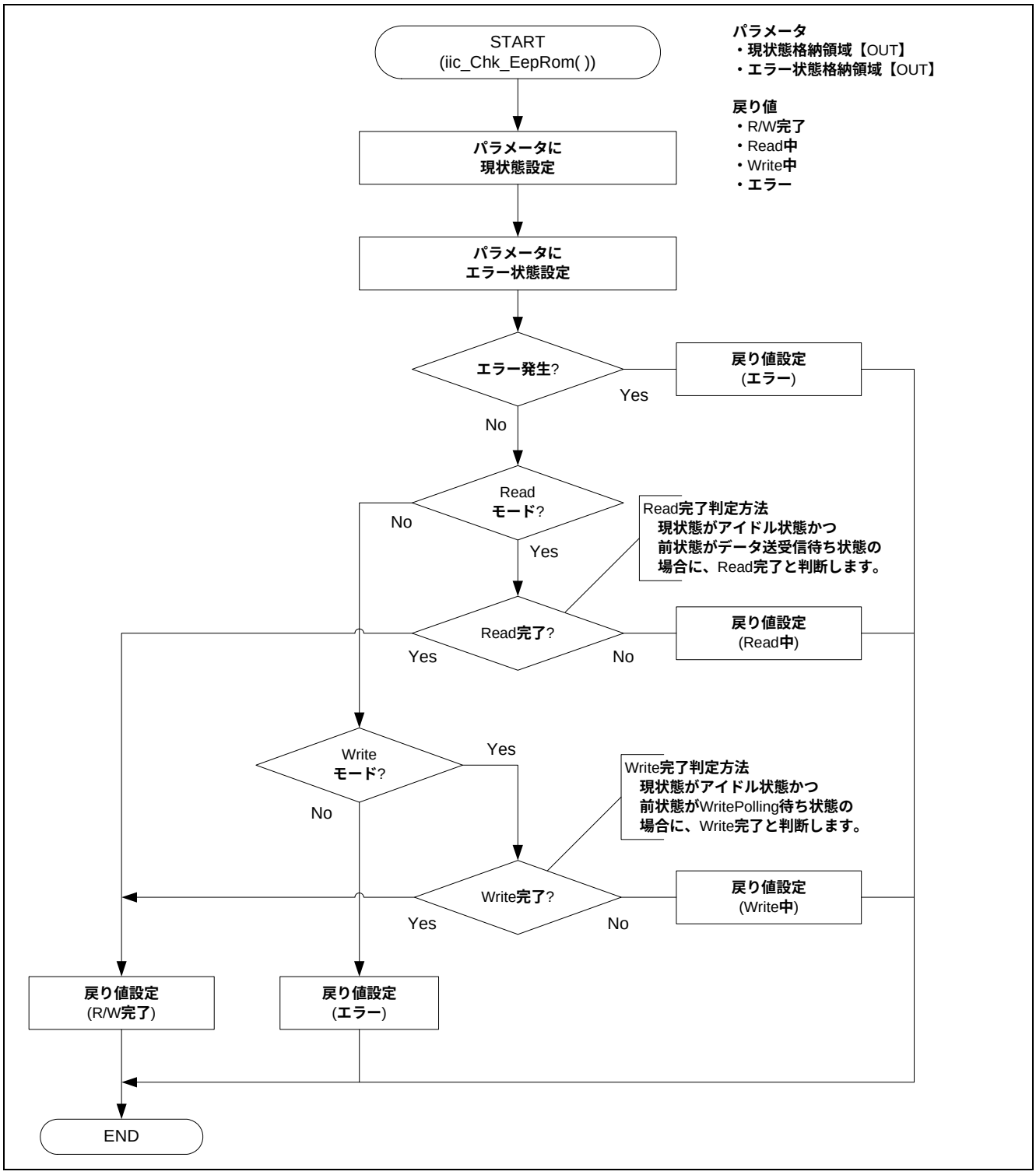

図 15　EEPROM 状態取得処理

4.16.4　AL 発生時処理

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。

4.16.5　送信時 NACK 受信処理

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。

4.16.6　開始条件発行後処理

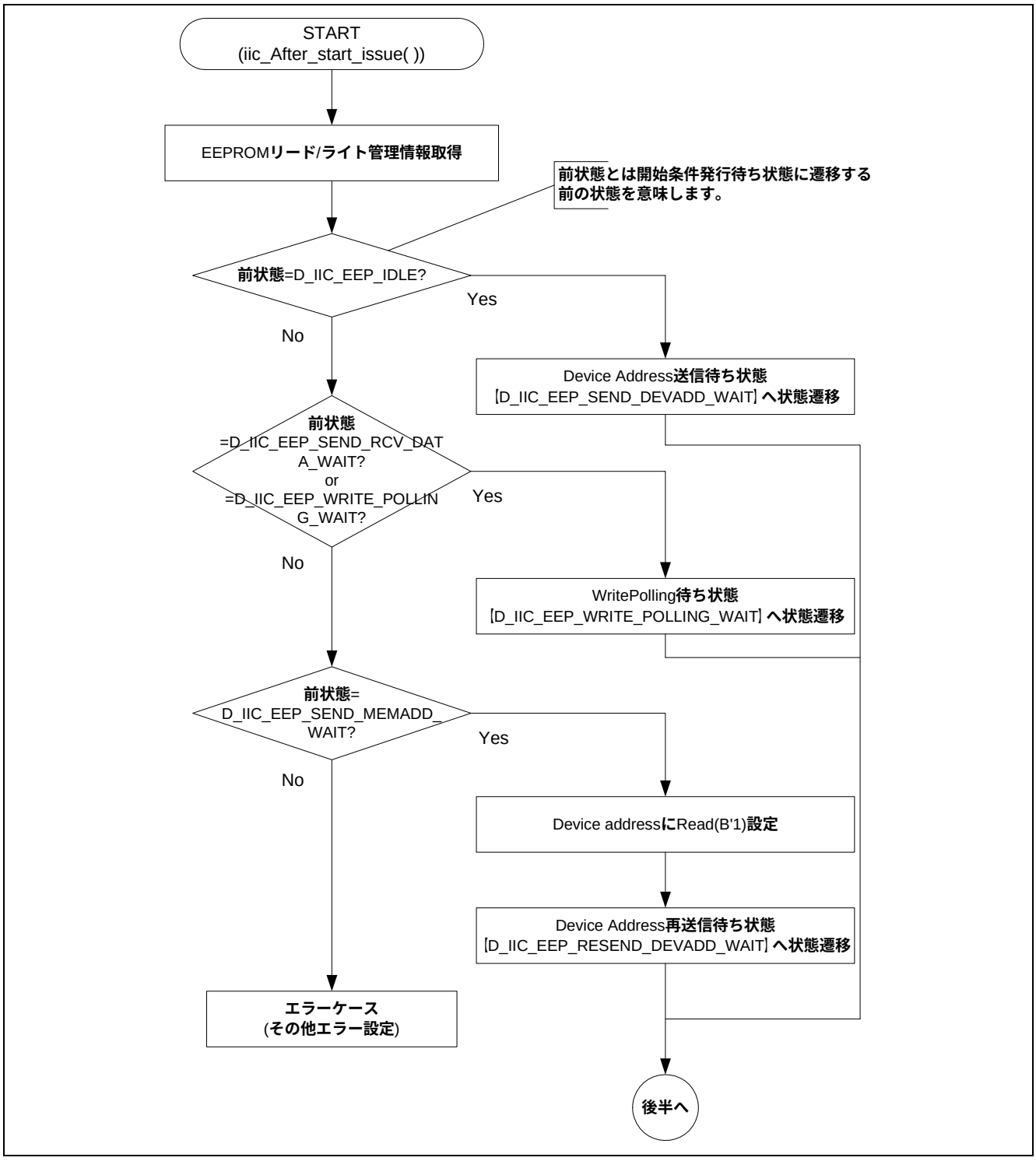

図 16　開始条件発行後処理 (前半)

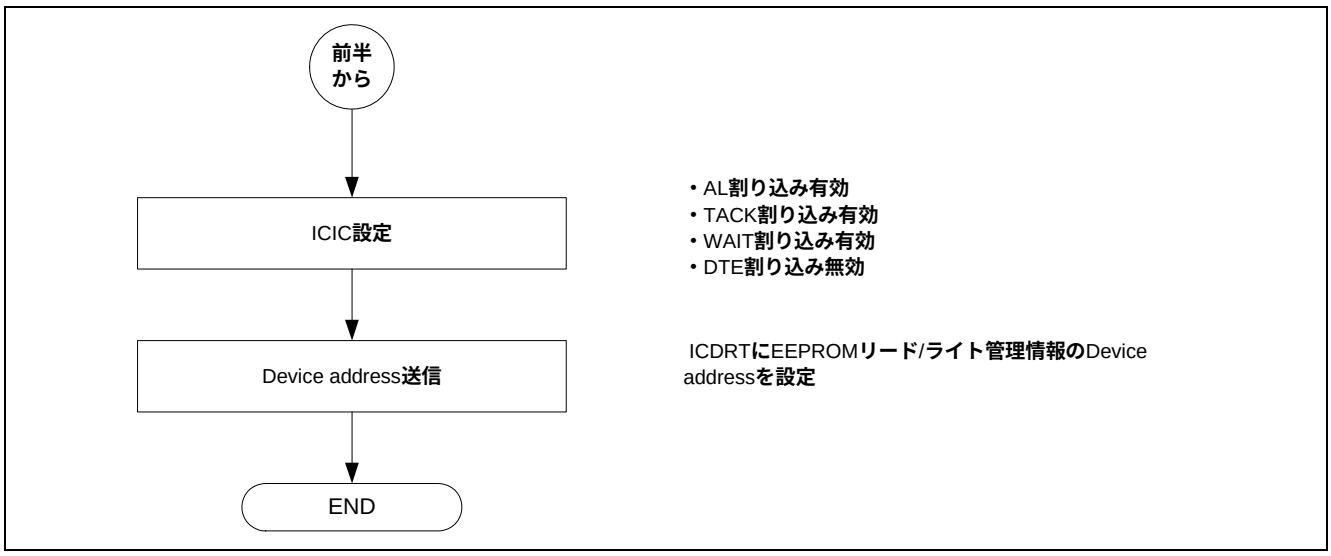

図 17　開始条件発行後処理 (後半)

4.16.7　初回 Device Address 送信後処理

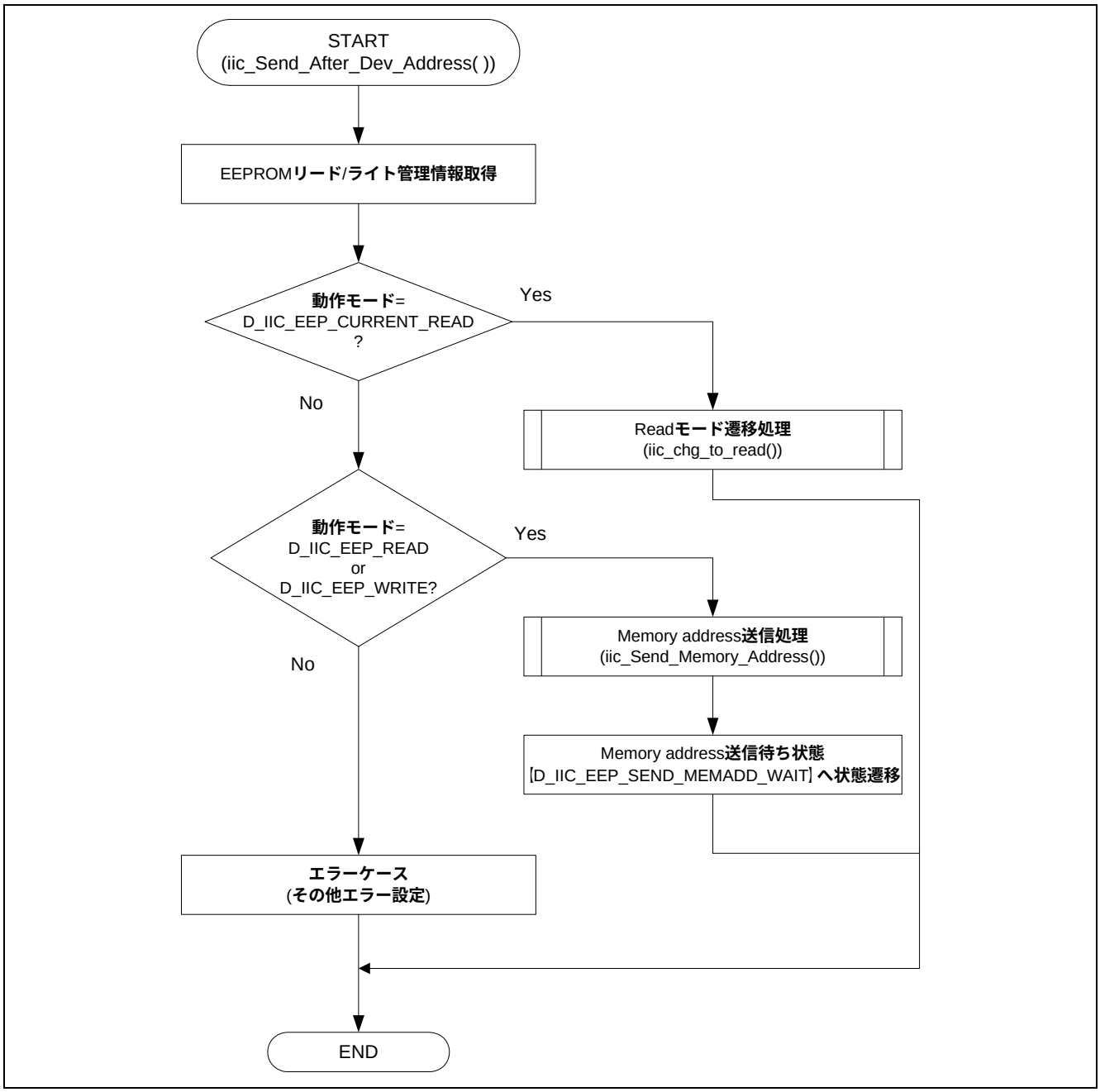

図 18　初回 Device Address 送信後処理

4.16.8　Device Address 再送信後処理

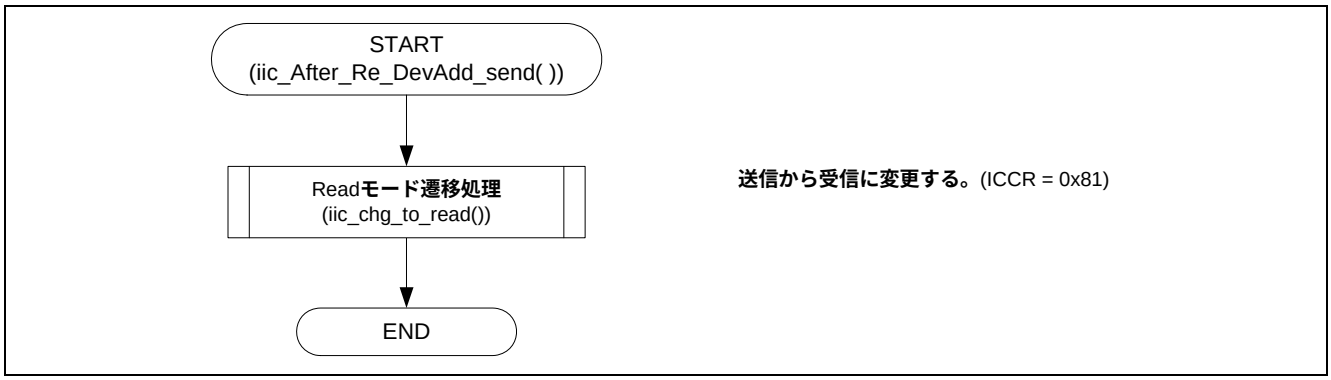

図 19　Device Address 再送信後処理

4.16.9　Memory address 送信中処理

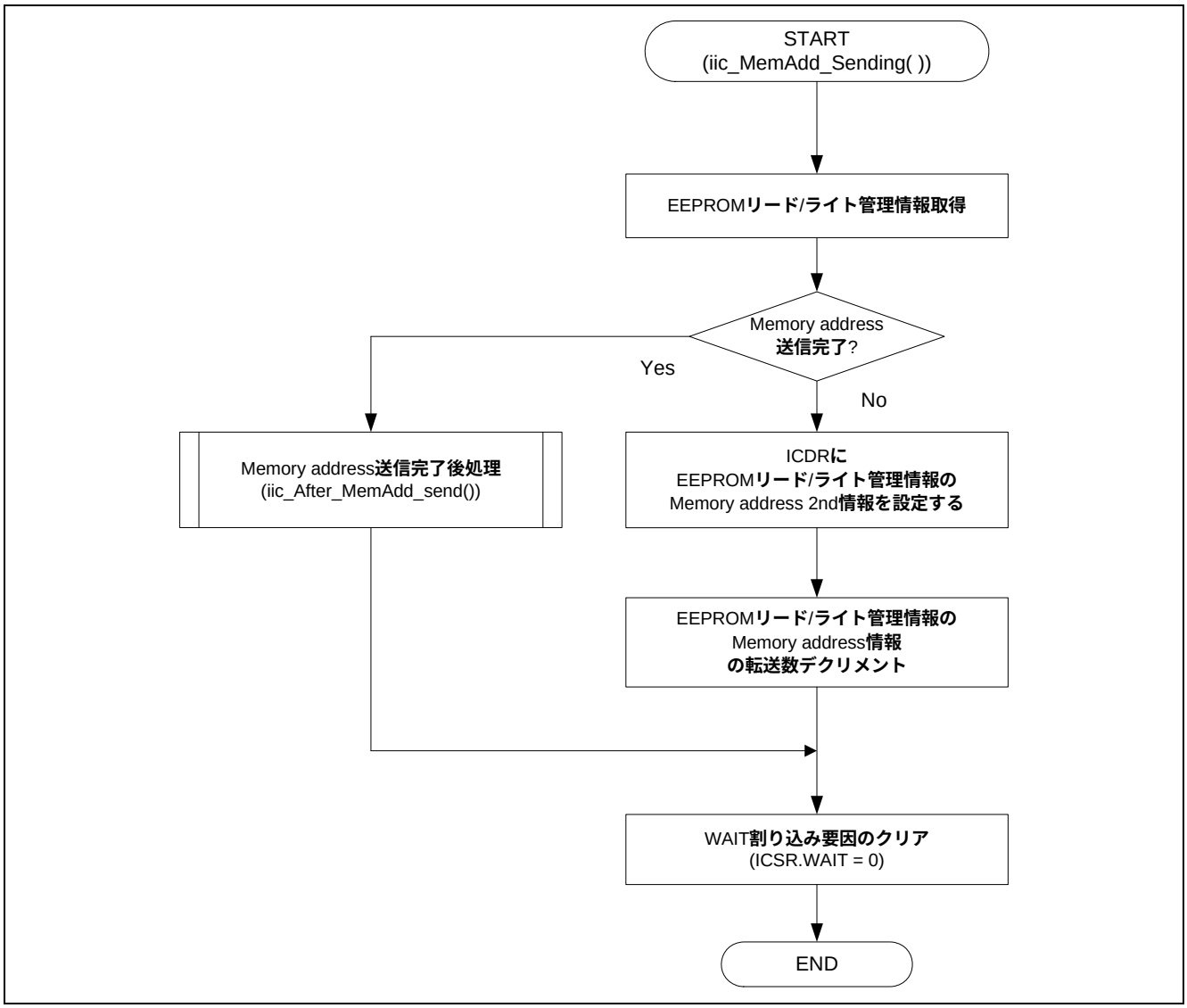

図 20　Memory address 送信中処理

4.16.10　データ送受信中処理

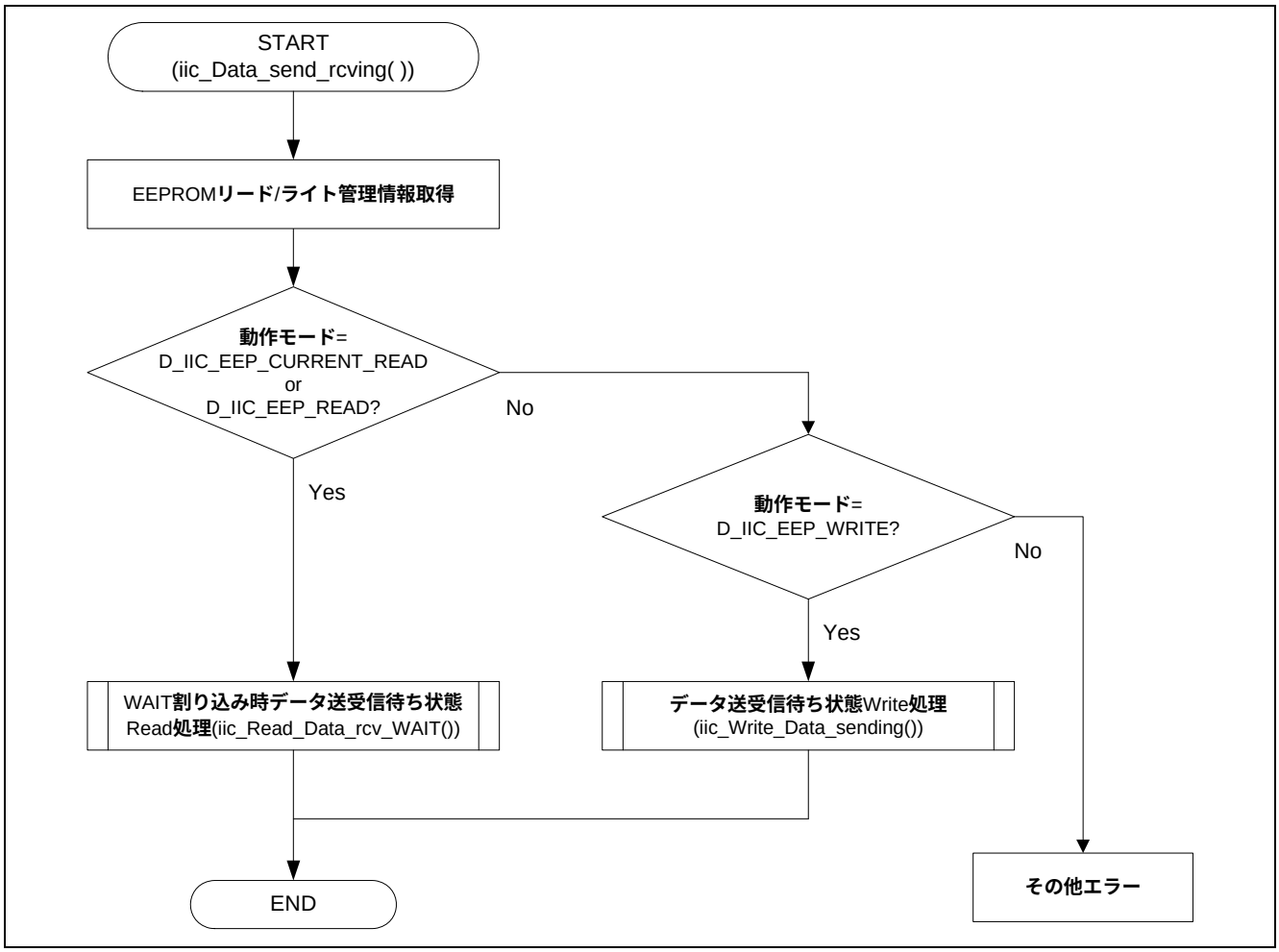

図 21　データ送受信中処理

4.16.11　DTE 割り込み時データ受信処理

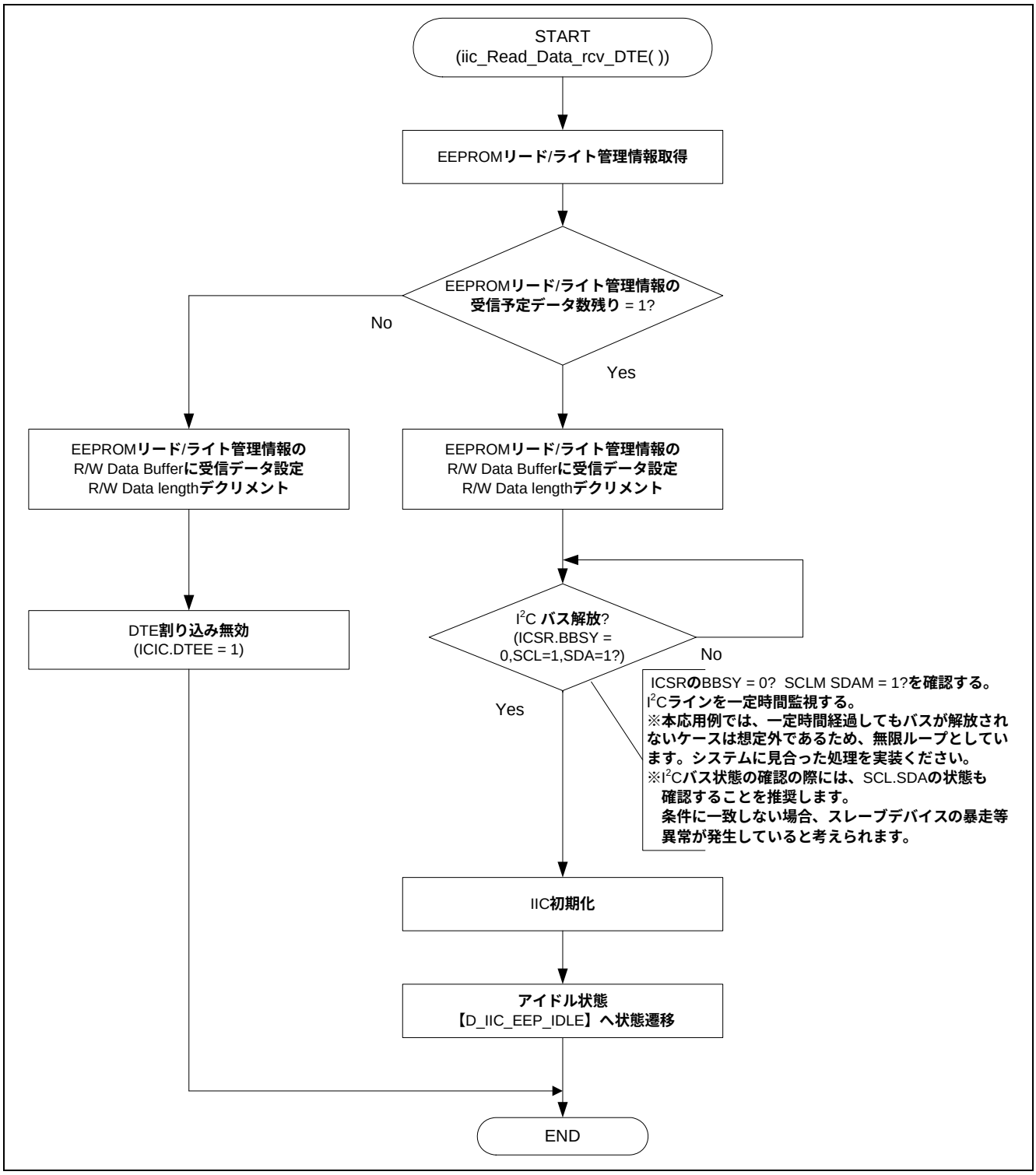

図 22　DTE 割り込み時データ受信処理

4.16.12　TACK 割り込み時 WritePolling 処理

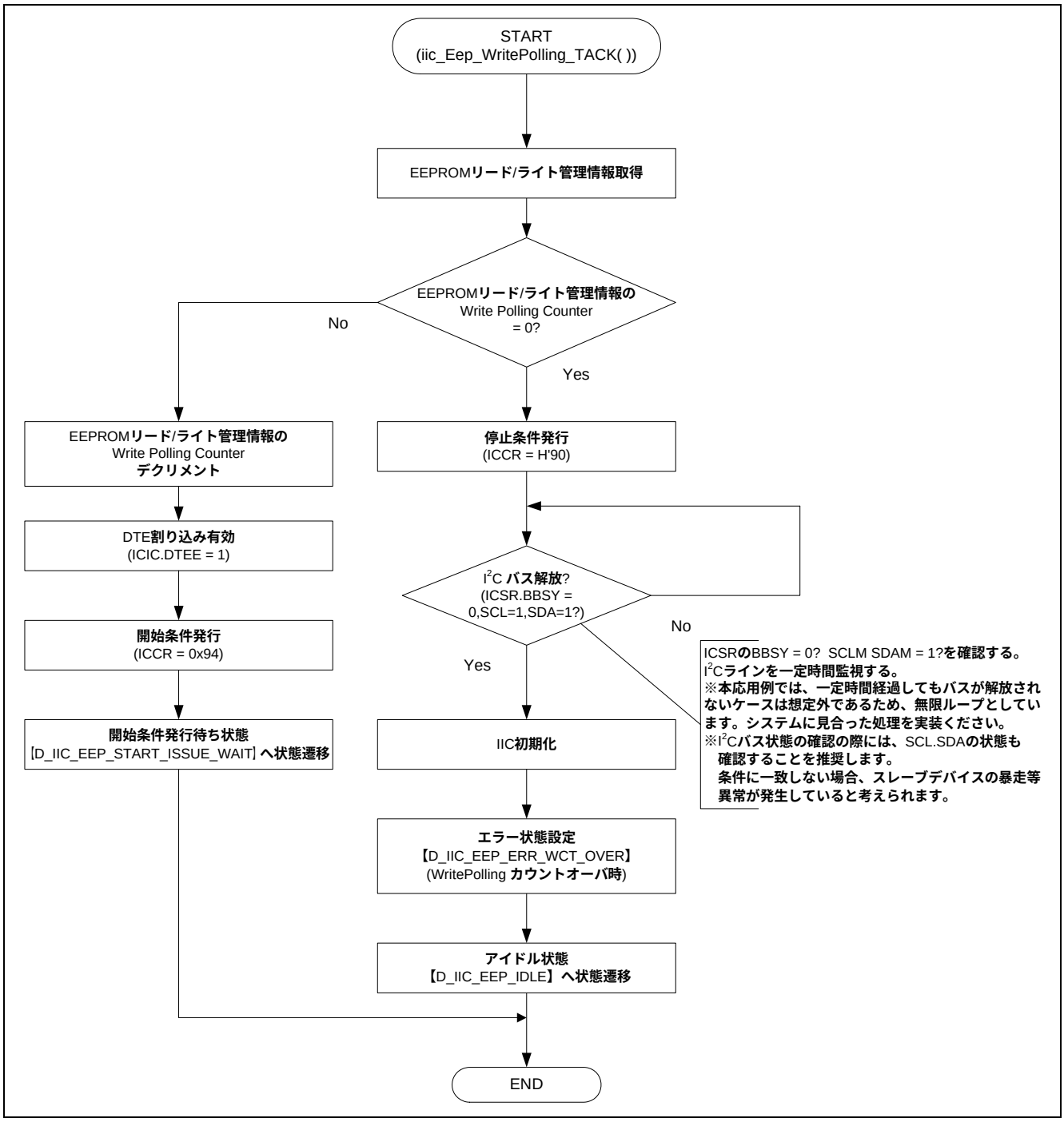

図 23　TACK 割り込み時 WritePolling 処理

4.16.13　WAIT 割り込み時 WritePolling 処理

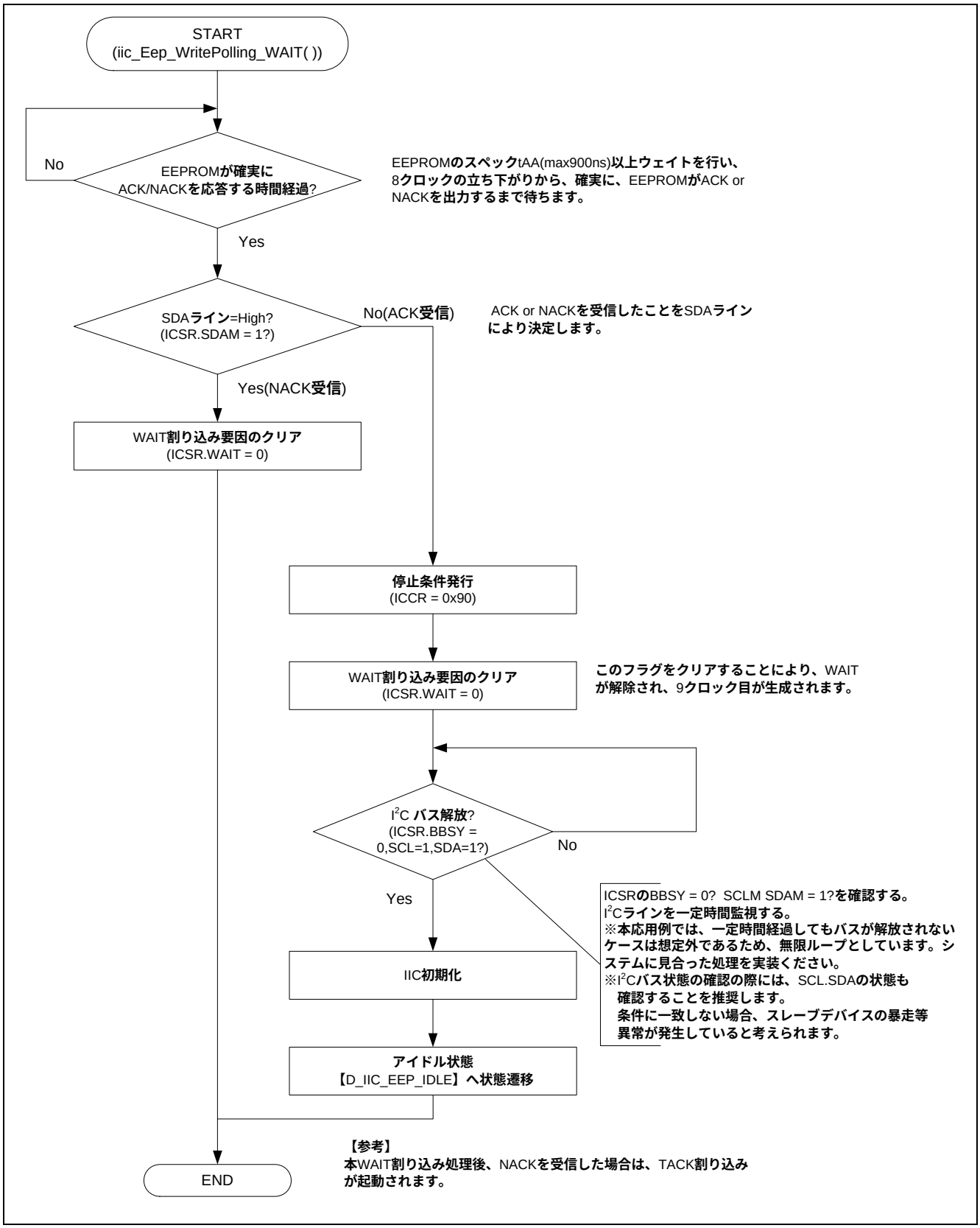

図 24　WAIT 割り込み時 WritePolling 処理

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

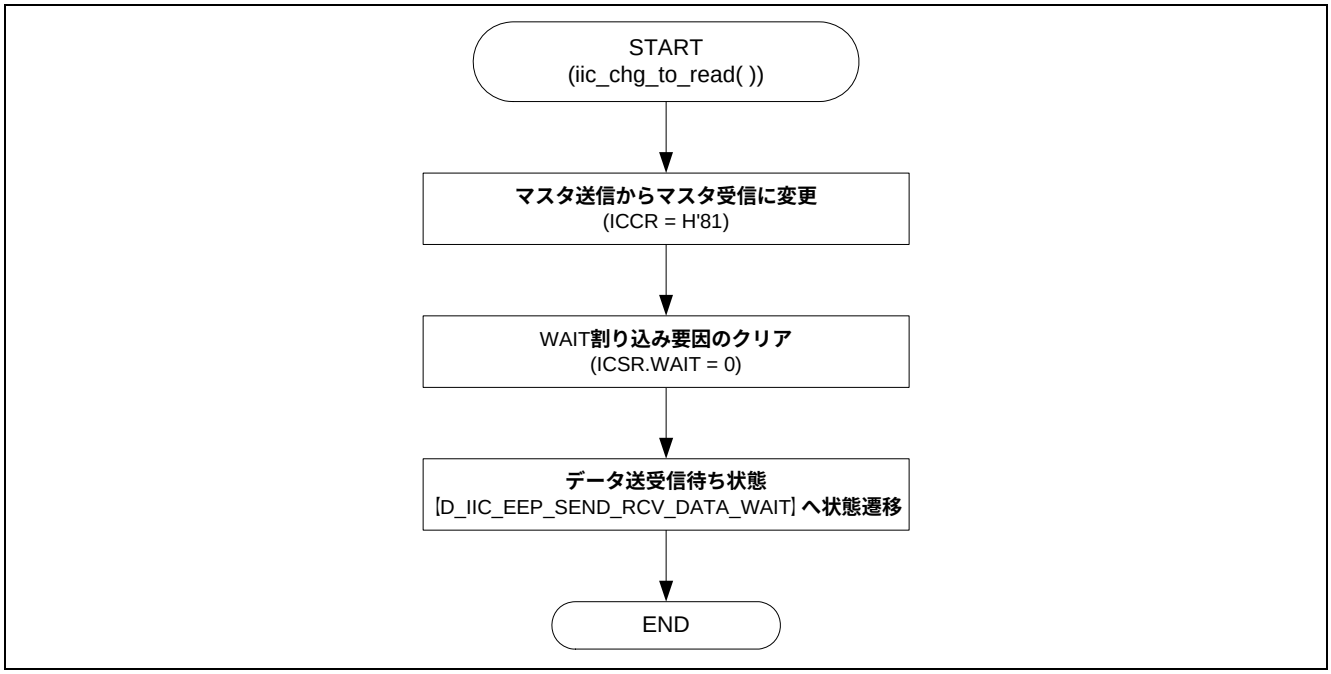

図 25　Read モード遷移処理

【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

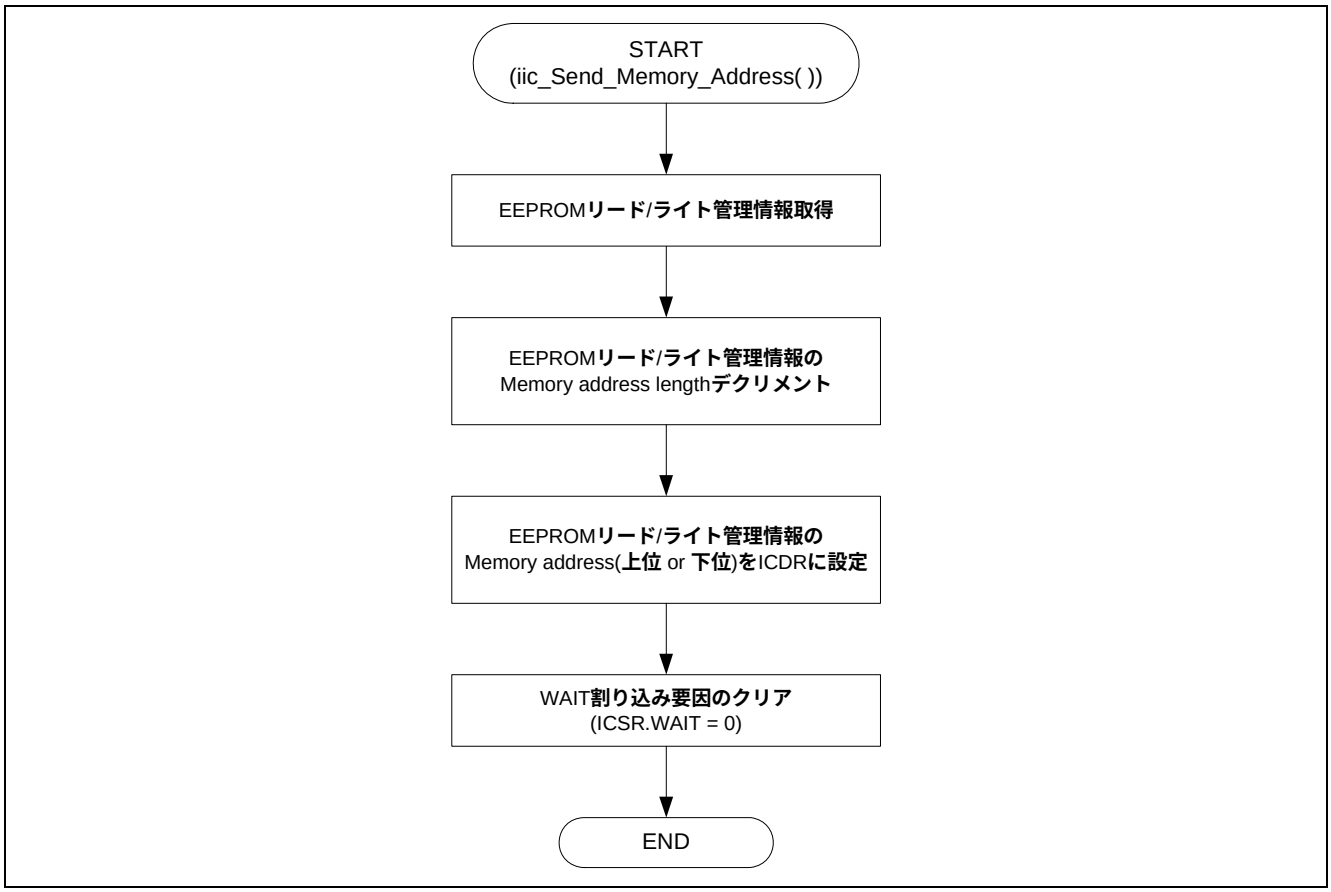

図 26　Memory address 送信処理

【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

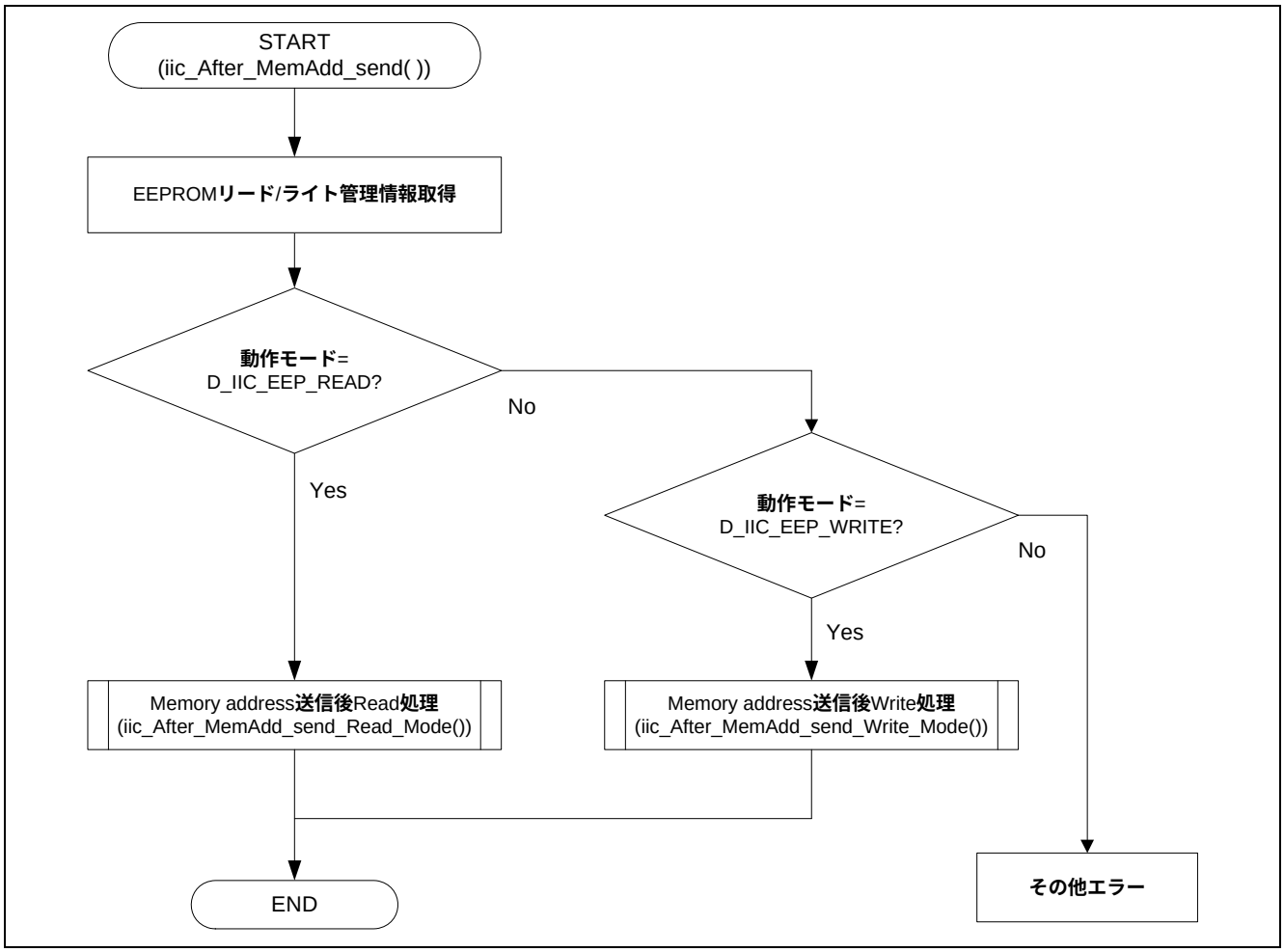

図 27　Memory address 送信完了後処理

【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 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

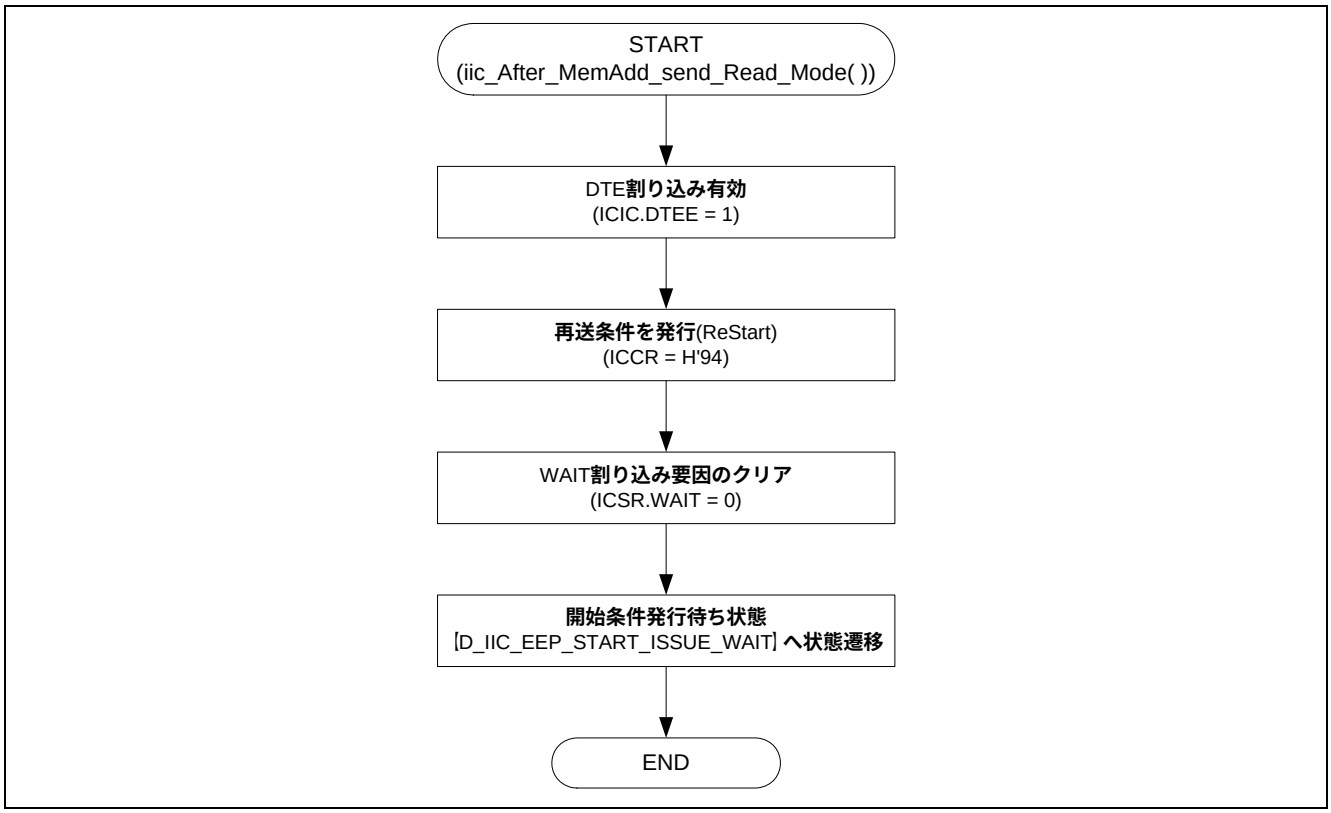

図 28　Memory address **送信後** Read **処理**

【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

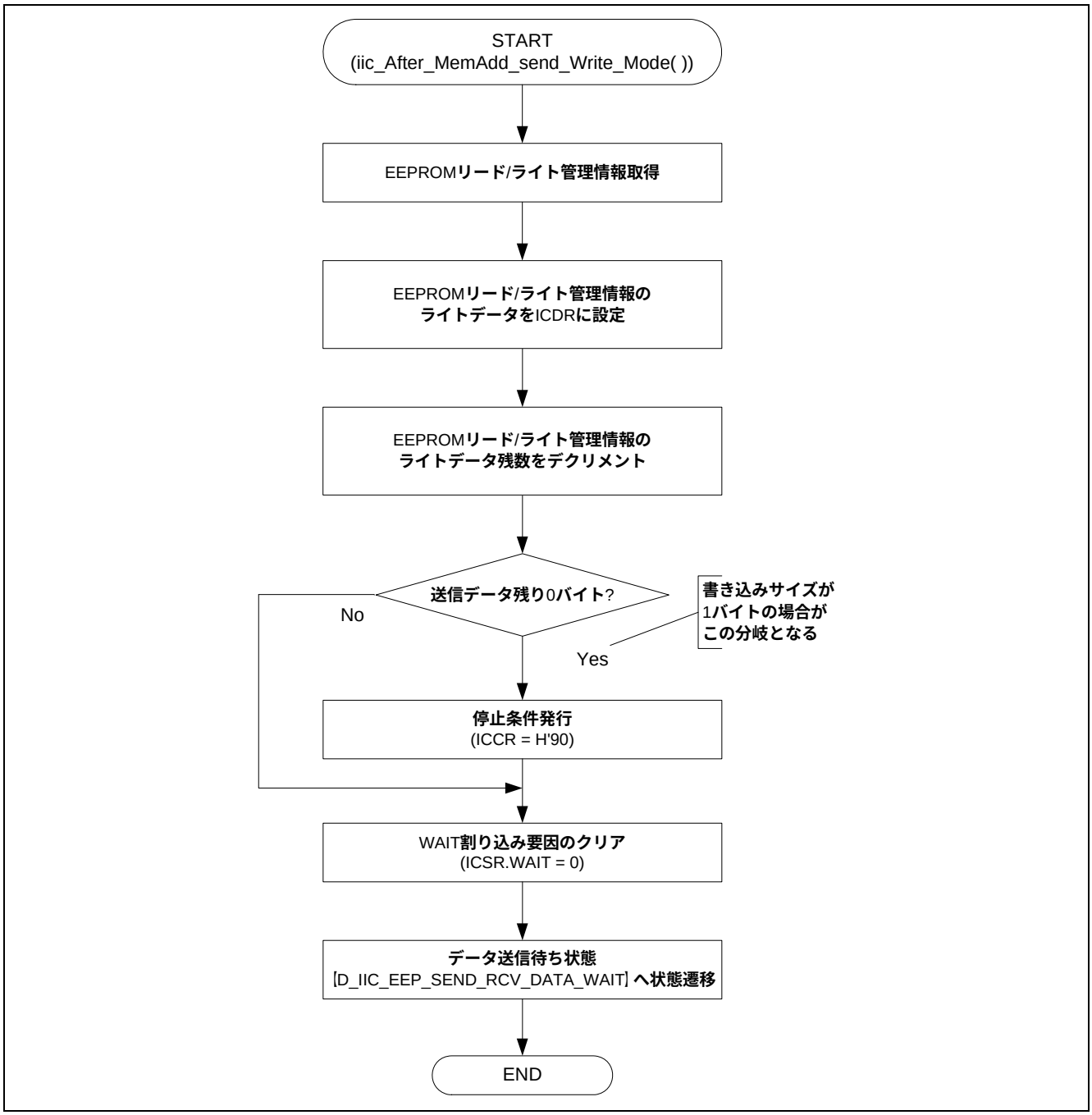

図 29　Memory address 送信後 Write 処理

【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　WAIT 割り込み時データ送受信待ち状態 Read 処理

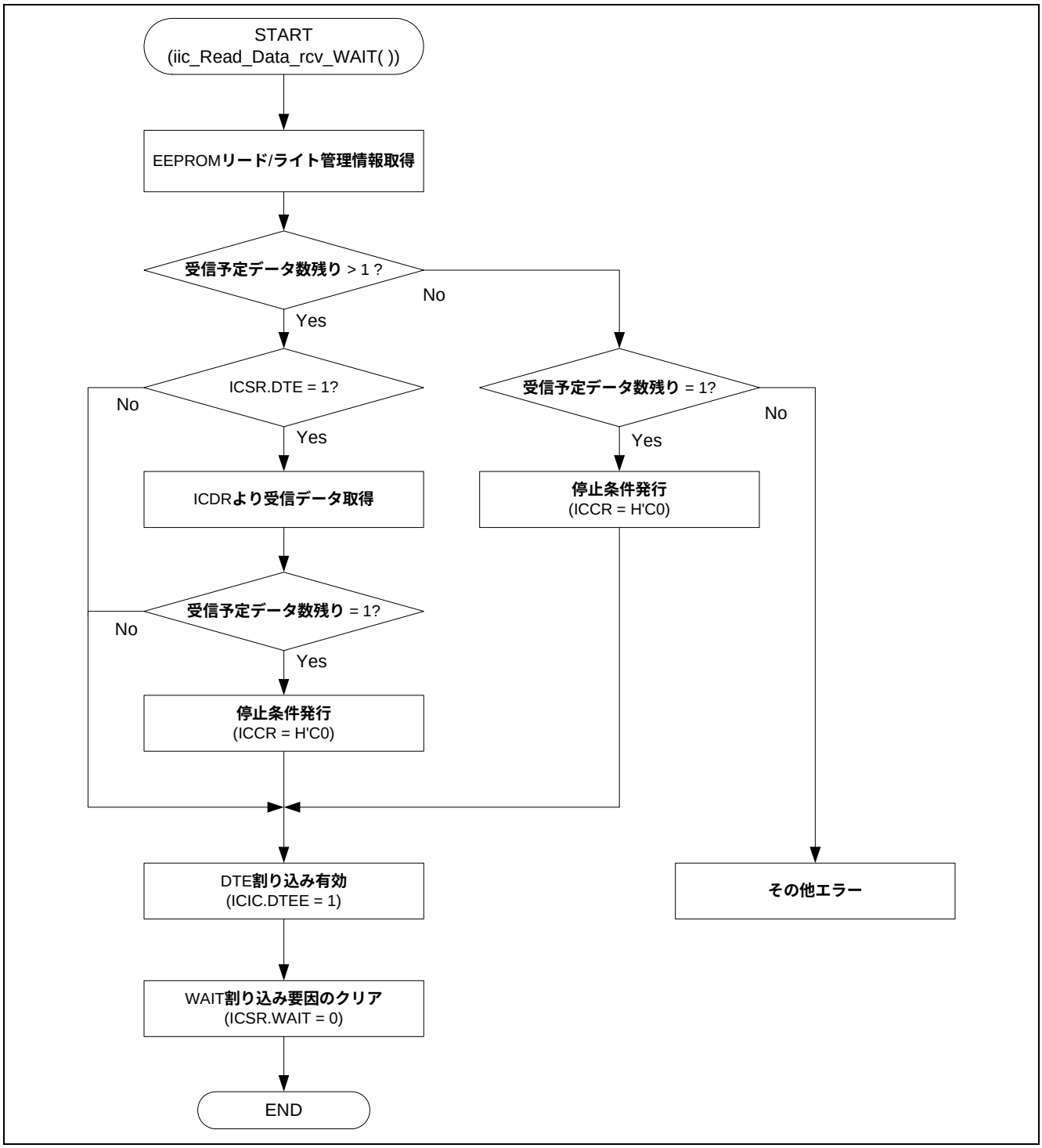

図 30　WAIT 割り込み時データ送受信待ち状態 Read 処理

【呼び出しタイミング】

① D_IIC_EEP_READ or D_IIC_EEP_CURRENT_READ モードで、データ受信中にコールされます。

【参考】

本関数で WAIT 割り込み要因のクリア後、DTE 割り込みが発生し、そのタイミングで受信データを取得します。

4.17.7　データ送受信待ち状態 Write 処理

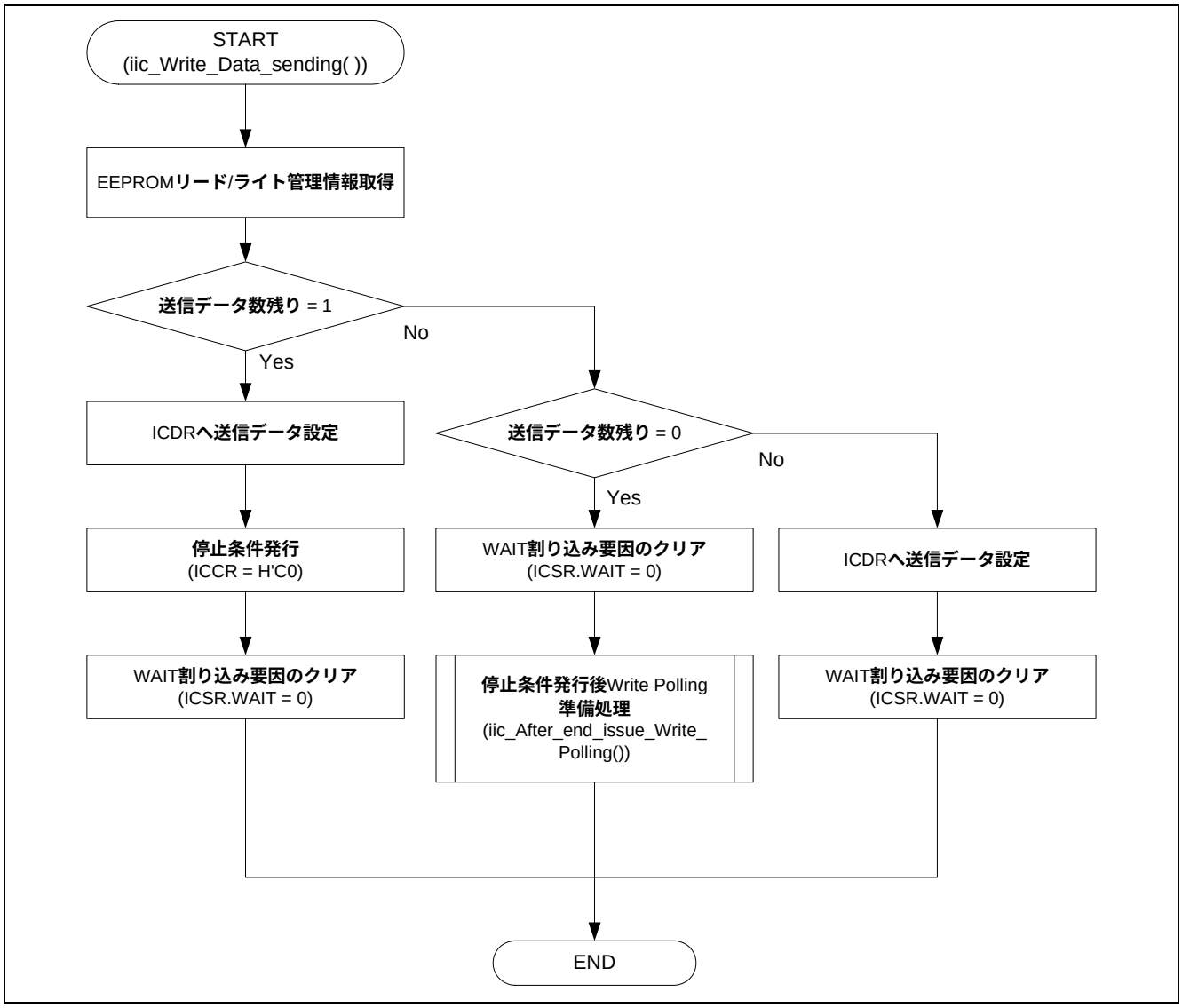

図 31　データ送受信待ち状態 Write 処理

【呼び出しタイミング】

① D_IIC_EEP_WRITE モードで、データ送信中にコールされます。

4.17.8　停止条件発行後 Write Polling 準備処理

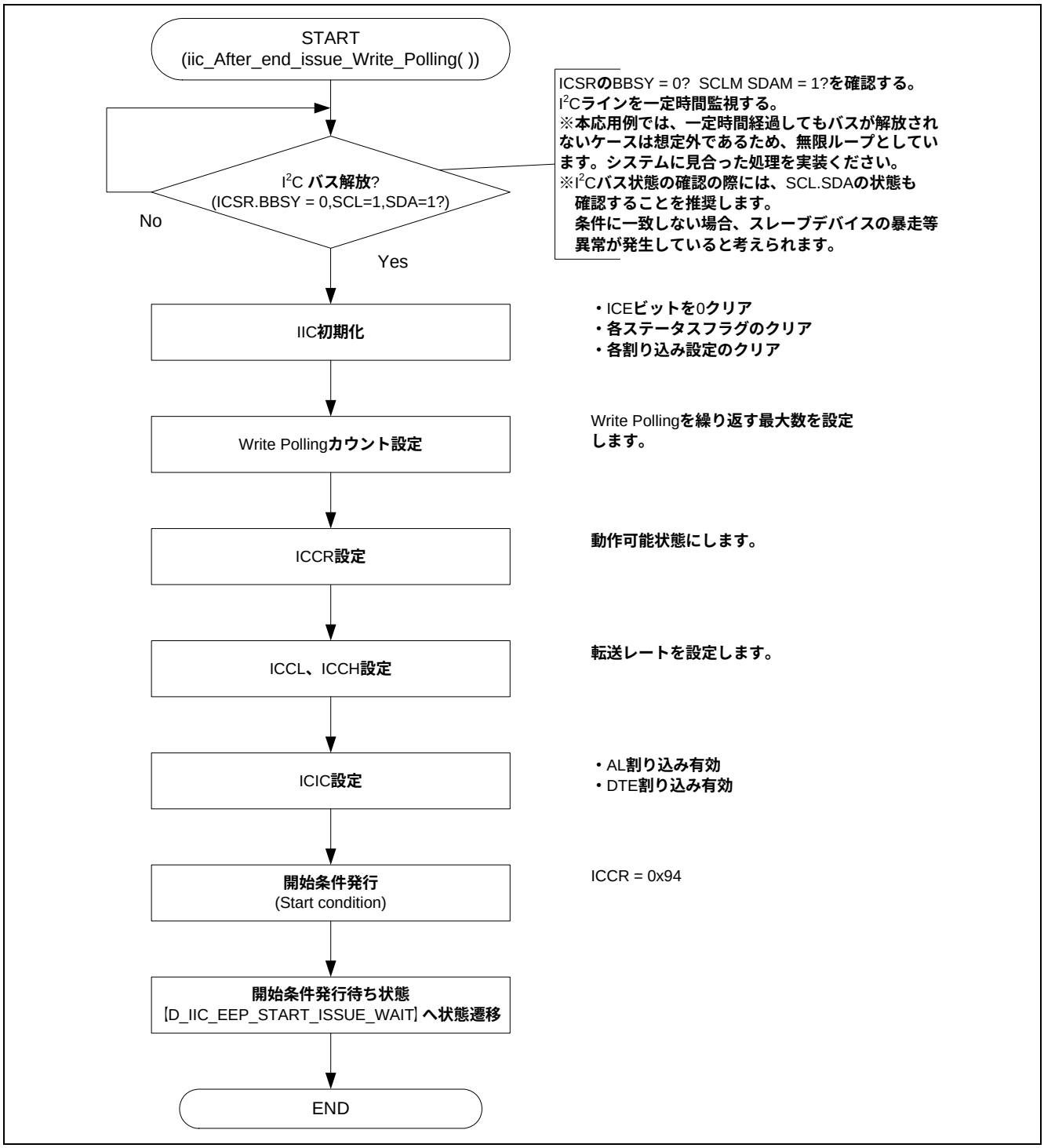

図 32　停止条件発行後 Write Polling 準備処理

【呼び出しタイミング】

① D_IIC_EEP_WRITE モードで、全データ送信して、停止条件発行後に Write Polling を行う準備をする際にコールされます。

5. 参考プログラム例

(1) サンプルプログラムリスト"main.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  : SH7722,SH7731 Sample Program
31    * File Name    : main.c
32    * Abstract     : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
33    * Version      : Ver 1.00
34    * Device       : SH7722,SH7731
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 : R0P7722TH001ARK for SH7722 Reference Platform
39    * Description  : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
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    /* ライトデータ格納領域 */
57    unsigned char gEEPROM_Write_Data[]
58        ={0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a};
59
```

```
60    /* リードデータ格納領域 */
61    unsigned char gEEPROM_Read_Data[D_IIC_EEPROM_READ_DATA_NUM];
62
63    void main(void);
64
65    #pragma section ROMC                /* RAM 上で実行するセクション */
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 SH7722,SH7731、
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  : SH7722,SH7731 Sample Program
31    * File Name    : iic_eeprom_use_if.c
32    * Abstract     : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
33    * Version      : Ver 1.00
34    * Device       : SH7722,SH7731
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 : R0P7722TH001ARK for SH7722 Reference Platform
39    * Description  : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
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    #pragma section ROMC                /* RAM 上で実行するセクション */
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          : D_IIC_EEP_OK
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               : IIC AL 割り込み処理
141   * Include               :
142   * Declaration           : E_Iic_eep_Ret iic_user_int_AL(void)
143   * Description           :
144   *                       :
145   * Argument              : none
146   * Return Value          : E_Iic_eep_Ret
147   * Calling Functions     :
148   *""FUNC COMMENT END""*********************************************/
149   E_Iic_eep_Ret iic_user_int_AL(void)
150   {
151       E_Iic_eep_Ret       ret = D_IIC_EEP_OK;
152
153       ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_AL, NULL);
154
155       return ret;
156
157   }
158
159   /*""FUNC COMMENT""*************************************************
160   * ID                    :
161   * Outline               : IIC TACK 割り込み処理
162   * Include               :
163   * Declaration           : E_Iic_eep_Ret iic_user_int_TACK(void)
164   * Description           :
165   *                       :
166   * Argument              : none
167   * Return Value          : E_Iic_eep_Ret
168   * Calling Functions     :
169   *""FUNC COMMENT END""*********************************************/
170   E_Iic_eep_Ret iic_user_int_TACK(void)
171   {
172       E_Iic_eep_Ret       ret = D_IIC_EEP_OK;
173
174       ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_TACK, NULL);
175
176       return ret;
177
178   }
179
180   /*""FUNC COMMENT""*************************************************
181   * ID                    :
182   * Outline               : IIC WAIT 割り込み処理
183   * Include               :
184   * Declaration           : E_Iic_eep_Ret iic_user_int_WAIT(void)
185   * Description           :
186   *                       :
187   * Argument              : none
188   * Return Value          : E_Iic_eep_Ret
189   * Calling Functions     :
190   *""FUNC COMMENT END""*********************************************/
```

```
191   E_Iic_eep_Ret iic_user_int_WAIT(void)
192   {
193       E_Iic_eep_Ret        ret = D_IIC_EEP_OK;
194
195       ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_WAIT, NULL);
196
197       return ret;
198
199   }
200
201   /*""FUNC COMMENT""************************************************
202   * ID                    :
203   * Outline               : IIC DTE 割り込み処理
204   * Include               :
205   * Declaration           : E_Iic_eep_Ret iic_user_int_DTE(void)
206   * Description           :
207   *                       :
208   * Argument              : none
209   * Return Value          : E_Iic_eep_Ret
210   * Calling Functions      :
211   *""FUNC COMMENT END""*****************************************/
212   E_Iic_eep_Ret iic_user_int_DTE(void)
213   {
214       E_Iic_eep_Ret        ret = D_IIC_EEP_OK;
215
216       ret = iic_mtx_executeFuncTable(D_IIC_EEP_EV_INT_DTE, NULL);
217
218       return ret;
219
220   }
221
222   /* 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  : SH7722,SH7731 Sample Program
31    * File Name    : iic_eeprom.c
32    * Abstract     : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
33    * Version      : Ver 1.00
34    * Device       : SH7722,SH7731
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 : R0P7722TH001ARK for SH7722 Reference Platform
39    * Description  : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
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    #define I_DIV_P        8 /* Iφ:Pφ = 8:1 */
53    #define PCLK_5CYC      ( 5 * I_DIV_P / 2)
54
55    E_Iic_eep_condition      gIic_Eep_Condition;            /* IIC_EEPROM 状態 */
56    E_Iic_eep_condition      gIic_Eep_Before_Condition; /* IIC_EEPROM 前状態 */
57    E_Iic_eep_err_condition  gIic_Eep_Err_Condition;        /* IIC_EEPROM 状態エラー */
58    T_IIC_EEPROM_RW_MANAGE   gIic_Eep_Manage_Info;      /* EEPROM リード/ライト管理情報 */
59
60    /* 状態遷移表定義関数宣言 */
61    static E_Iic_eep_Ret iic_Init(void *info);
62    static E_Iic_eep_Ret iic_EepRomRW(void *info);
```

```
63    static E_Iic_eep_Ret iic_Chk_EepRom(void *info);
64    static E_Iic_eep_Ret iic_AL_generate(void *info);
65    static E_Iic_eep_Ret iic_NACK_rcv(void *info);
66    static E_Iic_eep_Ret iic_After_start_issue(void* info);
67    static E_Iic_eep_Ret iic_Send_After_Dev_Address(void* info);
68    static E_Iic_eep_Ret iic_After_Re_DevAdd_send(void *info);
69    static E_Iic_eep_Ret iic_MemAdd_Sending(void *info);
70    static E_Iic_eep_Ret iic_Data_send_rcving(void *info);
71    static E_Iic_eep_Ret iic_Read_Data_rcv_DTE(void *info);
72    static E_Iic_eep_Ret iic_Eep_WritePolling_TACK(void *info);
73    static E_Iic_eep_Ret iic_Eep_WritePolling_WAIT(void *info);
74
75
76    /* 状態遷移表 */
77    static const T_IIC_EEPROM_MTX_TBL
78    g_Iic_Eep_mtx_tbl[D_IIC_EEP_CONDITION_MAX][D_IIC_EEP_EVENT_MAX] =
79    {
80        /* 未初期化状態 */
81        {
82          { D_IIC_EEP_EV_INIT           ,   iic_Init                  } , /* iic_user_Init()コール時
83    */
84          { D_IIC_EEP_EV_RW_START       ,   NULL                      } , /* iic_user_EepRomRW()
85    コール時    */
86          { D_IIC_EEP_EV_CHECK      ,   NULL                      } , /* iic_user_Chk_EepRom()コー
87    ル時  */
88          { D_IIC_EEP_EV_INT_AL     ,   NULL                      } , /* AL 割り込み時
89    */
90          { D_IIC_EEP_EV_INT_TACK       ,   NULL                      } , /* TACK 割り込み時
91    */
92          { D_IIC_EEP_EV_INT_WAIT       ,   NULL                      } , /* WAIT 割り込み時
93    */
94          { D_IIC_EEP_EV_INT_DTE    ,   NULL                      }   /* DTE 割り込み時
95    */
96
97        },
98
99        /* アイドル状態 */
100       {
101         { D_IIC_EEP_EV_INIT           ,   iic_Init                  } , /* iic_user_Init()コール時
102   */
103         { D_IIC_EEP_EV_RW_START       ,   iic_EepRomRW              } , /* iic_user_EepRomRW()
104   コール時    */
105         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom            } , /* iic_user_Chk_EepRom()コー
106   ル時  */
107         { D_IIC_EEP_EV_INT_AL     ,   NULL                      } , /* AL 割り込み時
108   */
109         { D_IIC_EEP_EV_INT_TACK       ,   NULL                      } , /* TACK 割り込み時
110   */
111         { D_IIC_EEP_EV_INT_WAIT       ,   NULL                      } , /* WAIT 割り込み時
112   */
113         { D_IIC_EEP_EV_INT_DTE    ,   NULL                      }   /* DTE 割り込み時
114   */
115       },
116
117       /* 開始条件発行待ち状態 */
118       {
119         { D_IIC_EEP_EV_INIT           ,   NULL                      } , /* iic_user_Init()コール時
120   */
121         { D_IIC_EEP_EV_RW_START       ,   NULL                      } , /* iic_user_EepRomRW()
122   コール時    */
123         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom            } , /* iic_user_Chk_EepRom()コー
124   ル時  */
125         { D_IIC_EEP_EV_INT_AL     ,   iic_AL_generate           } , /* AL 割り込み時
126   */
```

```
127         { D_IIC_EEP_EV_INT_TACK       ,   iic_NACK_rcv               } , /* TACK 割り込み時
128   */
129         { D_IIC_EEP_EV_INT_WAIT       ,   NULL                       } , /* WAIT 割り込み時
130   */
131         { D_IIC_EEP_EV_INT_DTE    ,   iic_After_start_issue      }   /* DTE 割り込み時
132   */
133       },
134
135       /* Device Address 送信待ち状態  */
136       {
137         { D_IIC_EEP_EV_INIT           ,   NULL                       } , /* iic_user_Init()コール時
138   */
139         { D_IIC_EEP_EV_RW_START       ,   NULL                       } , /* iic_user_EepRomRW()
140   コール時   */
141         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom             } , /* iic_user_Chk_EepRom()コー
142   ル時  */
143         { D_IIC_EEP_EV_INT_AL     ,   iic_AL_generate            } , /* AL 割り込み時
144   */
145         { D_IIC_EEP_EV_INT_TACK       ,   iic_NACK_rcv               } , /* TACK 割り込み時
146   */
147         { D_IIC_EEP_EV_INT_WAIT       ,   iic_Send_After_Dev_Address } , /* WAIT 割り込み時
148   */
149         { D_IIC_EEP_EV_INT_DTE    ,   NULL                       }   /* DTE 割り込み時
150   */
151
152       },
153
154       /* Device Address 再送信待ち状態  */
155       {
156         { D_IIC_EEP_EV_INIT           ,   NULL                       } , /* iic_user_Init()コール時
157   */
158         { D_IIC_EEP_EV_RW_START       ,   NULL                       } , /* iic_user_EepRomRW()
159   コール時   */
160         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom             } , /* iic_user_Chk_EepRom()コー
161   ル時  */
162         { D_IIC_EEP_EV_INT_AL     ,   iic_AL_generate            } , /* AL 割り込み時
163   */
164         { D_IIC_EEP_EV_INT_TACK       ,   iic_NACK_rcv               } , /* TACK 割り込み時
165   */
166         { D_IIC_EEP_EV_INT_WAIT       ,   iic_After_Re_DevAdd_send   } , /* WAIT 割り込み時
167   */
168         { D_IIC_EEP_EV_INT_DTE    ,   NULL                       }   /* DTE 割り込み時
169   */
170
171       },
172
173       /* Memory address 送信待ち状態  */
174       {
175         { D_IIC_EEP_EV_INIT           ,   NULL                       } , /* iic_user_Init()コール時
176   */
177         { D_IIC_EEP_EV_RW_START       ,   NULL                       } , /* iic_user_EepRomRW()
178   コール時   */
179         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom             } , /* iic_user_Chk_EepRom()コー
180   ル時  */
181         { D_IIC_EEP_EV_INT_AL     ,   iic_AL_generate            } , /* AL 割り込み時
182   */
183         { D_IIC_EEP_EV_INT_TACK       ,   iic_NACK_rcv               } , /* TACK 割り込み時
184   */
185         { D_IIC_EEP_EV_INT_WAIT       ,   iic_MemAdd_Sending         } , /* WAIT 割り込み時
186   */
187         { D_IIC_EEP_EV_INT_DTE    ,   NULL                       }   /* DTE 割り込み時
188   */
189
190       },
```

```
191
192       /* データ送受信待ち状態  */
193       {
194         { D_IIC_EEP_EV_INIT           ,   NULL                        } , /* iic_user_Init()コール時
195   */
196         { D_IIC_EEP_EV_RW_START       ,   NULL                        } , /* iic_user_EepRomRW()
197   コール時    */
198         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom            } , /* iic_user_Chk_EepRom()コー
199   ル時  */
200         { D_IIC_EEP_EV_INT_AL     ,   iic_AL_generate           } , /* AL 割り込み時
201   */
202         { D_IIC_EEP_EV_INT_TACK       ,   iic_NACK_rcv              } , /* TACK 割り込み時
203   */
204         { D_IIC_EEP_EV_INT_WAIT       ,   iic_Data_send_rcving      } , /* WAIT 割り込み時
205   */
206         { D_IIC_EEP_EV_INT_DTE    ,   iic_Read_Data_rcv_DTE     }   /* DTE 割り込み時
207   */
208       },
209
210       /* WritePolling 待ち状態  */
211       {
212         { D_IIC_EEP_EV_INIT           ,   NULL                        } , /* iic_user_Init()コール時
213   */
214         { D_IIC_EEP_EV_RW_START       ,   NULL                        } , /* iic_user_EepRomRW()
215   コール時    */
216         { D_IIC_EEP_EV_CHECK      ,   iic_Chk_EepRom            } , /* iic_user_Chk_EepRom()コー
217   ル時  */
218         { D_IIC_EEP_EV_INT_AL     ,   iic_AL_generate           } , /* AL 割り込み時
219   */
220         { D_IIC_EEP_EV_INT_TACK       ,   iic_Eep_WritePolling_TACK } , /* TACK 割り込み時
221   */
222         { D_IIC_EEP_EV_INT_WAIT       ,   iic_Eep_WritePolling_WAIT } , /* WAIT 割り込み時
223   */
224         { D_IIC_EEP_EV_INT_DTE    ,   NULL                        }   /* DTE 割り込み時
225   */
226       }
227
228   };
229
230   /* 内部関数宣言 */
231   static E_Iic_eep_Ret iic_chg_to_read(void);
232   static E_Iic_eep_Ret iic_Send_Memory_Address(void);
233   static void iic_After_MemAdd_send(void);
234   static void iic_After_MemAdd_send_Read_Mode(void);
235   static void iic_After_MemAdd_send_Write_Mode(void);
236   static E_Iic_eep_Ret iic_Read_Data_rcv_WAIT(void);
237   static E_Iic_eep_Ret iic_Write_Data_sending(void);
238   static void iic_After_end_issue_Write_Polling(void);
239   static E_Iic_eep_Ret iic_Chk_Bus_SCL_SDA(E_Iic_eep_mode_bus_chk i_state);
240
241   static void set_Iic_Eep_Condition(E_Iic_eep_condition i_condition);
242   static E_Iic_eep_condition get_Iic_Eep_Condition(void);
243   static void set_Iic_Eep_Before_Condition(E_Iic_eep_condition i_condition);
244   static E_Iic_eep_condition get_Iic_Eep_Before_Condition(void);
245   static E_Iic_eep_mode get_Iic_Eep_Mode(void);
246   static void set_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *i_info);
247   static void get_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *o_info);
248   static void set_Iic_Eep_Err_Condition(E_Iic_eep_err_condition i_err);
249   static E_Iic_eep_err_condition get_Iic_Eep_Err_Condition(void);
250   static void set_internal_info_init(void);
251
252   #pragma section ROMC                /* RAM 上で実行するセクション */
253   /*""FUNC COMMENT""************************************************
254   * ID                    :
```

```
255   * Outline               : 状態遷移処理
256   * Include               :
257   * Declaration            : E_Iic_eep_Ret iic_mtx_executeFuncTable(void)
258   * Description            : 状態遷移処理を行います。
259   *                       :
260   * Argument               : E_Iic_eep_event     event
261   *                       : void              *info
262   * Return Value           : E_Iic_eep_Ret 型参照
263   * Calling Functions       :
264   *""FUNC COMMENT END""**********************************************/
265   static E_Iic_eep_Ret  iic_mtx_executeFuncTable(
266      E_Iic_eep_event     event,                  /* イベント */
267      void              *info                  /* 情報 */
268                                )
269   {
270       E_Iic_eep_Ret        ret = D_IIC_EEP_OK ;       /* 戻り値  */
271       E_Iic_eep_condition     NowCondition;              /* 内部状態 */
272       E_Iic_eep_Ret           (*pFunc)( void *);         /* 実行処理用 */
273       unsigned long           IntMask;                   /* マスク用 */
274       unsigned long           i;
275
276       IntMask = get_imask();              /* 割り込みマスク取得 */
277
278       if(IntMask < D_IIC_EEP_INT_LEVEL)
279       {
280         set_imask(D_IIC_EEP_INT_LEVEL);       /* 割り込みマスク設定 */
281       }
282
283       /* 現状態取得 */
284       NowCondition = get_Iic_Eep_Condition();
285
286       /* テーブル検索処理 */
287       if(( NowCondition < D_IIC_EEP_CONDITION_MAX ) &&
288          ( event < D_IIC_EEP_EVENT_MAX ))
289       {
290         /* テーブルの対応する処理を行う */
291         if( g_Iic_Eep_mtx_tbl[NowCondition][event].proc != NULL )
292         {
293             /* 該当イベント個別処理を実行する */
294             pFunc = g_Iic_Eep_mtx_tbl[NowCondition][event].proc;
295             ret = (*pFunc)(info);
296
297             /* INTC の優先順位判定時間待ち */
298             if(event >= D_IIC_EEP_EV_INT_AL)
299             {
300                 for(i=0;i<PCLK_5CYC;i++);
301             }
302         }
303         /* NULL 定義時無処理 */
304         else
305         {
306             ret = D_IIC_EEP_NOP;
307         }
308       }
309
310       set_imask(IntMask);                 /* 割り込みマスク復帰 */
311
312      return ret;
313   }
314
315   /*""FUNC COMMENT""**************************************************
316   * ID                    :
317   * Outline               : IIC 初期化処理
318   * Include               :
```

```
319   * Declaration            : E_Iic_eep_Ret iic_Init(void *info)
320   * Description            : IIC 初期化処理を行います。
321   *                       :
322   * Argument               : void  *info       : No Use
323   * Return Value           : D_IIC_EEP_OK
324   * Calling Functions       :
325   *""FUNC COMMENT END""*****************************************/
326   static E_Iic_eep_Ret iic_Init(void *info)
327   {
328       unsigned long        dummy;
329
330       set_internal_info_init();           /* 内部情報初期化 */
331
332       /* ==== モジュールストップレジスタ 1 設定 ==== */
333       LOWP.MSTPCR1 &= ~0x00000200;    /* IIC module clock supply */
334       dummy = LOWP.MSTPCR1;               /* 設定反映確認のためダミリード */
335
336       /* ==== PFC の設定 ==== */
337       /* 特に設定不要 */
338
339       /* IIC 初期化 */
340       /* ==== IIC.ICCR の設定 ==== */
341       IIC.ICCR &= ~0x80;                  /* 非動作状態 */
342
343       /* ==== IIC.ICSR の設定 ==== */
344       IIC.ICSR.BYTE = 0x00;               /* 各ステータスフラグのクリア */
345
346       /* ==== IIC.ICIC の設定 ==== */
347       IIC.ICIC.BYTE = 0x00;               /* 各割り込み設定のクリア */
348
349       /* ==== 割り込み優先レベル設定（優先レベル 1）==== */
350       INTC0.IPRH = INTC0.IPRH | 0x0001;
351
352       /* ==== 割り込みマスククリア ==== */
353       INTC0.IMCR7 = 0xf0;
354
355       /* EEPROM アクセス状態設定 */
356       set_Iic_Eep_Condition(D_IIC_EEP_IDLE);  /* アイドル状態 */
357
358       return D_IIC_EEP_OK;
359   }
360
361   /*""FUNC COMMENT""*************************************************
362   * ID                    :
363   * Outline                : EEPROM リード/ライト処理
364   * Include                :
365   * Declaration            : E_Iic_eep_Ret iic_EepRomRW(void *info)
366   * Description            : EEPROM リード/ライト処理を行います。
367   *                       :
368   * Argument               : void *info
369   *                       : パラメータに以下を設定します。
370   *                       : ・R/W mode
371   *                       : ・Device address
372   *                       : ・Memory address
373   *                       : ・Transfer/Receive data length
374   *                       : ・Transfer/Receive data buffer pointer
375   * Return Value           : D_IIC_EEP_OK     : Success
376   *                       : D_IIC_EEP_NG       : Error
377   * Calling Functions       :
378   *""FUNC COMMENT END""*****************************************/
379   static E_Iic_eep_Ret iic_EepRomRW(void *info)
380   {
381       T_IIC_EEPROM_RW_INFO *i_RW_Info;
382       T_IIC_EEPROM_RW_MANAGE   manage_info;
```

```
383
384       if(info == NULL)
385       {
386         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);  /* その他エラー */
387         return D_IIC_EEP_NG;
388       }
389
390       /* ==== IIC.ICCR の設定 ==== */
391       IIC.ICCR |= 0x80;                   /* 動作可能状態 */
392
393       /* I2C バス解放確認処理（スレーブデバイスがラインを解放するまで待つ） */
394       iic_Chk_Bus_SCL_SDA(D_IIC_EEP_BUS_CHK_START);
395
396       /*    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 */
397       /*    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、
398         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 */
399
400       /* ==== IIC.ICCR の設定 ==== */
401       IIC.ICCR &= ~0x80;                  /* 非動作状態 */
402
403       /* ==== IIC.ICCR の設定 ==== */
404       IIC.ICCR |= 0x80;                   /* 動作可能状態 */
405
406       /* I2C バス解放確認処理 */
407       /* IIC リセットを実施しても I2C バス解放されない場合 */
408       if(D_IIC_EEP_NG == iic_Chk_Bus_SCL_SDA(D_IIC_EEP_BUS_CHK_START))
409       {
410         return D_IIC_EEP_BUS_BUSY;
411       }
412
413       /* ==== IIC.ICCL の設定 ==== */
414       IIC.ICCL = 0x33;
415
416       /* ==== IIC.ICCH の設定 ==== */
417       IIC.ICCH = 0x20;
418
419       i_RW_Info = (T_IIC_EEPROM_RW_INFO *)info;
420
421       /* ソフトウェア内部情報初期化 */
422       memset(&manage_info, 0, sizeof(manage_info));
423       set_Iic_Eep_Before_Condition(D_IIC_EEP_NO_INIT);   /* 前状態初期化   */
424       set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_NO);       /* エラーなし状態 */
425       set_Iic_Eep_Manage_Info(&manage_info);
426
427       /* mode set */
428       manage_info.i_mode = i_RW_Info->i_mode;
429
430       /* Device address set */
431       manage_info.i_DevAdr = (i_RW_Info->i_DevAdr & 0x0f);
432       manage_info.i_DevAdr |= D_IIC_DEV_CODE;
433
434       /* モードにより場合わけ */
435       switch(manage_info.i_mode)
436       {
437         case D_IIC_EEP_WRITE:
438         case D_IIC_EEP_READ:
439
440             manage_info.i_DevAdr &= ~D_IIC_R_CODE;        /* Write code set */
441
442             break;
443
444         case D_IIC_EEP_CURRENT_READ:
445
446             manage_info.i_DevAdr |= D_IIC_R_CODE;     /* Read code set */
```

```
447
448             break;
449
450         default:
451             set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);  /* その他エラー */
452             return D_IIC_EEP_NG;
453       }
454
455       /* Memory address info set */
456       manage_info.i_MemAdrLen = D_IIC_EEP_MEMADR_SIZE;
457       manage_info.i_MemAdrBuf[1]  = (unsigned char)(i_RW_Info->i_RomAdr >> 8);
458       manage_info.i_MemAdrBuf[0]  = (unsigned char)i_RW_Info->i_RomAdr;
459
460       /* Read/Write data size set */
461       manage_info.i_RW_Data_info.i_DataLen = i_RW_Info->i_Len;
462       manage_info.i_RW_Data_info.i_DataBuf = i_RW_Info->i_pBuf;
463
464       /* ALE interrupt enable */
465       IIC.ICIC.BIT.ALE = 1;
466
467       /* TACKE interrupt disable */
468       IIC.ICIC.BIT.TACKE = 0;
469
470       /* WAIT interrupt disable */
471       IIC.ICIC.BIT.WAITE = 0;
472
473       /* DTE interrupt enable */
474       IIC.ICIC.BIT.DTEE = 1;
475
476       /* EEPROM リード/ライト管理情報設定 */
477       set_Iic_Eep_Manage_Info(&manage_info);
478
479       /* Start condition generate */
480       IIC.ICCR = 0x94;
481
482       /* EEPROM アクセス状態設定 */
483       set_Iic_Eep_Condition(D_IIC_EEP_START_ISSUE_WAIT); /* 開始条件発行待ち状態 */
484
485       return D_IIC_EEP_OK;
486
487   }
488
489   /*""FUNC COMMENT""************************************************
490   * ID                    :
491   * Outline               : EEPROM 状態取得処理
492   * Include               :
493   * Declaration           : E_Iic_eep_Ret iic_Chk_EepRom(void *info)
494   * Description           : EEPROM リード/ライト状態取得を行います。
495   *                       :
496   * Argument              : void  *info       : T_IIC_EEPROM_CONDITION 型参照
497   * Return Value          : E_Iic_eep_Ret 型参照
498   * Calling Functions     :
499   *""FUNC COMMENT END""*******************************************/
500   static E_Iic_eep_Ret iic_Chk_EepRom(void *info)
501   {
502       E_Iic_eep_Ret           ret = D_IIC_EEP_NG;
503       T_IIC_EEPROM_CONDITION  *condition_info;
504
505       if(info == NULL)
506       {
507         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);
508         return D_IIC_EEP_NG;
509       }
510
```

```
511       condition_info = (T_IIC_EEPROM_CONDITION *)info;
512
513       /* 現状態設定 */
514       condition_info->i_eep_condition = get_Iic_Eep_Condition();
515
516       /* エラー状態設定 */
517       condition_info->i_err_condition = get_Iic_Eep_Err_Condition();
518
519       do
520       {
521         /* エラー判定 */
522         if(D_IIC_EEP_ERR_NO != condition_info->i_err_condition)  /*エラー時 */
523         {
524             break;
525         }
526
527         switch(get_Iic_Eep_Mode())
528         {
529             case D_IIC_EEP_WRITE:          /* Write モード */
530
531                 /* 完了判定 */
532                 /* 今の状態がアイドルかつ前状態が WritePolling 待ち状態の場合 */
533                 if(D_IIC_EEP_IDLE          == get_Iic_Eep_Condition() &&
534                   D_IIC_EEP_WRITE_POLLING_WAIT == get_Iic_Eep_Before_Condition())
535                 {
536                     ret = D_IIC_EEP_OK;               /* 完了 */
537                 }else
538                 {
539                     ret = D_IIC_EEP_WRITING;
540                 }
541                 break;
542
543             case D_IIC_EEP_READ:           /* Read モード */
544             case D_IIC_EEP_CURRENT_READ:
545
546                 /* 完了判定 */
547                 /* 今の状態がアイドルかつ前状態がデータ送受信待ち状態の場合 */
548                 if(D_IIC_EEP_IDLE          == get_Iic_Eep_Condition() &&
549                   D_IIC_EEP_SEND_RCV_DATA_WAIT == get_Iic_Eep_Before_Condition())
550                 {
551                     ret = D_IIC_EEP_OK;               /* 完了 */
552                 }else
553                 {
554                     ret = D_IIC_EEP_READING;
555                 }
556                 break;
557
558             default:
559                 break;
560         }
561
562       }while(0);
563
564       return ret;
565
566   }
567
568   /*""FUNC COMMENT""************************************************
569   * ID                    :
570   * Outline                : AL 発生時処理
571   * Include                :
572   * Declaration             : E_Iic_eep_Ret iic_AL_generate(void* info)
573   *                        :
574   * Description             : AL 発生時時の処理を行います。
```

```
575   *                      :
576   * Argument              : void  *info       : No Use
577   * Return Value          : D_IIC_EEP_OK     : Success
578   * Calling Functions      :
579   *""FUNC COMMENT END""*********************************************/
580   static E_Iic_eep_Ret iic_AL_generate(void* info)
581   {
582
583       /* エラー状態設定 */
584       set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_AL);
585
586       /* システムに見合った処理を実装ください */
587
588       return D_IIC_EEP_OK;
589   }
590
591   /*""FUNC COMMENT""**************************************************
592   * ID                    :
593   * Outline               : ライトモード NACK 受信時処理
594   * Include               :
595   * Declaration           : E_Iic_eep_Ret iic_NACK_rcv(void* info)
596   *                      :
597   * Description           : ライトモード NACK 受信時の処理を行います。
598   *                      :
599   * Argument              : void  *info       : No Use
600   * Return Value          : D_IIC_EEP_OK     : Success
601   * Calling Functions      :
602   *""FUNC COMMENT END""*********************************************/
603   static E_Iic_eep_Ret iic_NACK_rcv(void* info)
604   {
605
606       /* エラー状態設定 */
607       set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_NACK);
608
609       /* システムに見合った処理を実装ください */
610
611       return D_IIC_EEP_OK;
612   }
613
614   /*""FUNC COMMENT""**************************************************
615   * ID                    :
616   * Outline               : 開始条件発行後処理
617   * Include               :
618   * Declaration           : E_Iic_eep_Ret iic_After_start_issue
619   *                      : (void* info)
620   * Description           : 開始条件発行後の処理を行います。
621   *                      :
622   * Argument              : void  *info       : No Use
623   * Return Value          : D_IIC_EEP_OK     : Success
624   *                      : D_IIC_EEP_NG       : Error
625   * Calling Functions      :
626   *""FUNC COMMENT END""*********************************************/
627   static E_Iic_eep_Ret iic_After_start_issue(void* info)
628   {
629       E_Iic_eep_Ret ret = D_IIC_EEP_OK;
630       T_IIC_EEPROM_RW_MANAGE   manage_info;
631       E_Iic_eep_condition     Eep_Condition;
632
633       memset(&manage_info, 0, sizeof(manage_info));
634
635       /* EEPROM リード/ライト管理情報取得 */
636       get_Iic_Eep_Manage_Info(&manage_info);
637
638       /* 前状態により分岐 */
```

```
639       switch(get_Iic_Eep_Before_Condition())
640       {
641       case D_IIC_EEP_IDLE:                              /* アイドル状態 */
642
643         /* EEPROM アクセス状態設定 */
644         Eep_Condition = D_IIC_EEP_SEND_DEVADD_WAIT;      /* Device Address 送信待ち状態 */
645
646         break;
647
648       case D_IIC_EEP_SEND_RCV_DATA_WAIT:                /* データ送受信待ち状態 */
649       case D_IIC_EEP_WRITE_POLLING_WAIT:                /* WritePolling 待ち状態 */
650
651         /* EEPROM アクセス状態設定 */
652         Eep_Condition = D_IIC_EEP_WRITE_POLLING_WAIT;/* WritePolling 待ち状態 */
653
654         break;
655
656       case D_IIC_EEP_SEND_MEMADD_WAIT:                  /* Memory address 送信待ち状態 */
657
658         /* Read code set */
659         manage_info.i_DevAdr |= D_IIC_R_CODE;
660
661         /* EEPROM リード/ライト管理情報設定 */
662         set_Iic_Eep_Manage_Info(&manage_info);
663
664         /* EEPROM アクセス状態設定 */
665         Eep_Condition = D_IIC_EEP_RESEND_DEVADD_WAIT;/* Device Address 再送信待ち状態 */
666
667         break;
668
669       default:
670
671         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);  /* その他エラー */
672         ret = D_IIC_EEP_NG;
673         break;
674
675       }
676
677       if(ret == D_IIC_EEP_OK)
678       {
679
680         /* ALE interrupt enable */
681         IIC.ICIC.BIT.ALE = 1;
682
683         /* TACKE interrupt enable */
684         IIC.ICIC.BIT.TACKE = 1;
685
686         /* WAIT interrupt enable */
687         IIC.ICIC.BIT.WAITE = 1;
688
689         /* DTE interrupt disable */
690         IIC.ICIC.BIT.DTEE = 0;
691
692         /* Device address set to transfer data */
693         IIC.ICDR = manage_info.i_DevAdr;
694
695         /* EEPROM アクセス状態設定 */
696         set_Iic_Eep_Condition(Eep_Condition);
697
698       }
699
700       return ret;
701
702   }
```

```
703
704   /*""FUNC COMMENT""************************************************
705   * ID                    :
706   * Outline               : 初回 Device Address 送信後処理
707   * Include               :
708   * Declaration           : E_Iic_eep_Ret iic_Send_After_Dev_Address
709   *                       : (void* info)
710   * Description           : 初回 Device Address 送信後の処理を行います。
711   *                       :
712   * Argument              : void  *info       : No Use
713   * Return Value          : D_IIC_EEP_OK      : Success
714   *                       : D_IIC_EEP_NG      : Error
715   * Calling Functions     :
716   *""FUNC COMMENT END""*******************************************/
717   static E_Iic_eep_Ret iic_Send_After_Dev_Address(void* info)
718   {
719       T_IIC_EEPROM_RW_MANAGE  manage_info;
720       E_Iic_eep_Ret           ret = D_IIC_EEP_NG;
721
722       memset(&manage_info, 0, sizeof(manage_info));
723
724       /* EEPROM リード/ライト管理情報取得 */
725       get_Iic_Eep_Manage_Info(&manage_info);
726
727       /* モードにより分岐 */
728       switch(manage_info.i_mode)
729       {
730         case D_IIC_EEP_CURRENT_READ:  /* Current Address Read と Sequential Read Operation 組み合わ
731   せモード */
732
733           /* Read モード遷移処理 */
734           ret = iic_chg_to_read();
735
736           break;
737
738         case D_IIC_EEP_READ:          /* Random Read Operation と Sequential Read Operation 組み合わ
739   せモード*/
740         case D_IIC_EEP_WRITE:         /* Write モード */
741
742           /* Memory address 送信処理 */
743           ret = iic_Send_Memory_Address();
744
745           /* EEPROM アクセス状態設定 */
746           set_Iic_Eep_Condition(D_IIC_EEP_SEND_MEMADD_WAIT);   /* Memory address 送信待ち状態 */
747
748           break;
749
750         default:
751           set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);      /* その他エラー */
752           ret = D_IIC_EEP_NG;
753           break;
754       }
755
756       return ret;
757
758   }
759
760   /*""FUNC COMMENT""************************************************
761   * ID                    :
762   * Outline               : Device Address 再送信後処理
763   * Include               :
764   * Declaration           : E_Iic_eep_Ret iic_After_Re_DevAdd_send
765   *                       :  (void* info)
766   * Description           : Device Address 再送信後処理を行います。
```

```
767   *                       :
768   * Argument               : void  *info       : No Use
769   * Return Value           : D_IIC_EEP_OK     : Success
770   *                        : D_IIC_EEP_NG       : Error
771   * Calling Functions       :
772   *""FUNC COMMENT END""*************************************************/
773   static E_Iic_eep_Ret iic_After_Re_DevAdd_send(void* info)
774   {
775       E_Iic_eep_Ret ret;
776
777       /* Read モード遷移処理 */
778       ret = iic_chg_to_read();
779
780       return ret;
781   }
782
783   /*""FUNC COMMENT""*****************************************************
784   * ID                    :
785   * Outline                : Memory address 送信中処理
786   * Include                :
787   * Declaration            : E_Iic_eep_Ret iic_MemAdd_Sending
788   *                        : (void* info)
789   * Description            : Memory address 送信中処理を行います。
790   *                        :
791   * Argument               : void  *info       : No Use
792   * Return Value           : D_IIC_EEP_OK     : Success
793   *                        :
794   * Calling Functions       :
795   *""FUNC COMMENT END""*************************************************/
796   static E_Iic_eep_Ret iic_MemAdd_Sending(void* info)
797   {
798       T_IIC_EEPROM_RW_MANAGE   manage_info;
799
800       memset(&manage_info, 0, sizeof(manage_info));
801
802       /* EEPROM リード/ライト管理情報取得 */
803       get_Iic_Eep_Manage_Info(&manage_info);
804
805       /* 2nd Memory address が無い場合 */
806       if(manage_info.i_MemAdrLen == 0)
807       {
808         /* Memory address 送信完了後処理 */
809         iic_After_MemAdd_send();
810       }
811       else
812       {
813         /* Memory address 送信処理 */
814         iic_Send_Memory_Address();
815       }
816
817       return D_IIC_EEP_OK;
818
819   }
820
821   /*""FUNC COMMENT""*****************************************************
822   * ID                    :
823   * Outline                : データ送受信中処理
824   * Include                :
825   * Declaration            : E_Iic_eep_Ret iic_Data_send_rcving
826   *                        :  (void* info)
827   * Description            :
828   *                        :
829   * Argument               : void  *info       : No Use
830   * Return Value           : D_IIC_EEP_OK     : Success
```

```
831   *                       : D_IIC_EEP_NG       : Error
832   * Calling Functions       :
833   *""FUNC COMMENT END""*********************************************/
834   static E_Iic_eep_Ret iic_Data_send_rcving(void* info)
835   {
836
837       T_IIC_EEPROM_RW_MANAGE      manage_info;
838       E_Iic_eep_Ret               ret = D_IIC_EEP_NG;
839
840       memset(&manage_info, 0, sizeof(manage_info));
841
842       /* EEPROM リード/ライト管理情報取得 */
843       get_Iic_Eep_Manage_Info(&manage_info);
844
845       /* モードにより分岐 */
846       switch(manage_info.i_mode)
847       {
848         case D_IIC_EEP_CURRENT_READ:  /* Current Address Read と Sequential Read Operation 組み合わ
849   せモード */
850         case D_IIC_EEP_READ:          /* Random Read Operation と Sequential Read Operation 組み合わ
851   せモード*/
852
853             /* WAIT 割り込み時データ送受信待ち状態 Read 処理 */
854             ret = iic_Read_Data_rcv_WAIT();
855
856             break;
857
858         case D_IIC_EEP_WRITE:         /* Write モード */
859
860             /* データ送受信待ち状態 Write 処理 */
861             ret = iic_Write_Data_sending();
862
863             break;
864
865         default:
866             set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);      /* その他エラー */
867             ret = D_IIC_EEP_NG;
868             break;
869       }
870
871       return ret;
872
873   }
874
875   /*""FUNC COMMENT""*************************************************
876   * ID                    :
877   * Outline                : DTE 割り込み時データ受信処理
878   * Include                :
879   * Declaration            : E_Iic_eep_Ret iic_Read_Data_rcv_DTE(void *info)
880   * Description            : DTE 割り込み時のデータ受信処理を行います。
881   *                       :
882   * Argument               : void  *info       : No Use
883   * Return Value           : D_IIC_EEP_OK      : Success
884   * Calling Functions       :
885   *""FUNC COMMENT END""*********************************************/
886   static E_Iic_eep_Ret iic_Read_Data_rcv_DTE(void *info)
887   {
888       E_Iic_eep_condition     Eep_Condition;
889       T_IIC_EEPROM_RW_MANAGE  manage_info;
890
891       memset(&manage_info, 0, sizeof(manage_info));
892
893       Eep_Condition = get_Iic_Eep_Condition();           /* 現状態取得 */
894
```

```
895       /* EEPROM リード/ライト管理情報取得 */
896       get_Iic_Eep_Manage_Info(&manage_info);
897
898
899       if(manage_info.i_RW_Data_info.i_DataLen == 1)      /* 受信予定データ数残り == 1 */
900       {
901
902         *manage_info.i_RW_Data_info.i_DataBuf = IIC.ICDR;
903         manage_info.i_RW_Data_info.i_DataBuf++;
904         manage_info.i_RW_Data_info.i_DataLen--;
905
906         /* I2C バス解放確認 */
907         iic_Chk_Bus_SCL_SDA(D_IIC_EEP_BUS_CHK_END);
908
909         /* IIC initialize */
910         IIC.ICIC.BYTE = 0x00;
911         IIC.ICSR.BYTE = 0x00;
912         IIC.ICCR &= ~0x80;
913
914         /* EEPROM アクセス状態設定 */
915         Eep_Condition = D_IIC_EEP_IDLE;
916
917       }
918       else
919       {
920         /* DTE interrupt disable */
921         IIC.ICIC.BIT.DTEE = 0;
922
923         *manage_info.i_RW_Data_info.i_DataBuf = IIC.ICDR;
924         manage_info.i_RW_Data_info.i_DataBuf++;
925         manage_info.i_RW_Data_info.i_DataLen--;
926
927
928       }
929
930       /* EEPROM リード/ライト管理情報更新 */
931       set_Iic_Eep_Manage_Info(&manage_info);
932
933       /* EEPROM アクセス状態設定 */
934       set_Iic_Eep_Condition(Eep_Condition);
935
936       return D_IIC_EEP_OK;
937
938   }
939
940   /*""FUNC COMMENT""************************************************
941   * ID                    :
942   * Outline               : TACK 割り込み時 WritePolling 処理
943   * Include               :
944   * Declaration           : E_Iic_eep_Ret iic_Eep_WritePolling_TACK
945   *                       :  (void* info)
946   * Description           : EEPROM ライト終了後のポーリング処理を
947   *                       : 行います。
948   * Argument              : void  *info       : No Use
949   * Return Value          : D_IIC_EEP_OK      : Success
950   * Calling Functions     :
951   *""FUNC COMMENT END""*******************************************/
952   static E_Iic_eep_Ret iic_Eep_WritePolling_TACK(void* info)
953   {
954       T_IIC_EEPROM_RW_MANAGE  manage_info;
955       E_Iic_eep_condition     Eep_Condition;
956
957       memset(&manage_info, 0, sizeof(manage_info));
958
```

```
959       Eep_Condition = get_Iic_Eep_Condition();
960
961       /* EEPROM リード/ライト管理情報取得 */
962       get_Iic_Eep_Manage_Info(&manage_info);
963
964       if(manage_info.i_WCTCnt == 0)   /* WritePolling == 0 の場合 */
965       {
966
967         set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_WCT_OVER);   /* エラー状態設定 */
968
969         /* Stop condition generate */
970         IIC.ICCR = 0x90;
971
972         /* I2C バス解放確認 */
973         iic_Chk_Bus_SCL_SDA(D_IIC_EEP_BUS_CHK_END);
974
975         /* IIC initialize */
976         IIC.ICCR &= ~0x80;
977         IIC.ICIC.BYTE = 0x00;
978         IIC.ICSR.BYTE = 0x00;
979
980         /* EEPROM アクセス状態設定 */
981         Eep_Condition = D_IIC_EEP_IDLE;   /* アイドル状態 */
982
983       }
984       else                /* WritePolling > 0 の場合 */
985       {
986         /* WritePolling カウントデクリメント */
987         manage_info.i_WCTCnt--;
988
989         /* DTE interrupt enable */
990         IIC.ICIC.BIT.DTEE = 1;
991
992         /* ReStart condition generate */
993         IIC.ICCR = 0x94;
994
995         /* EEPROM アクセス状態設定 */
996         Eep_Condition = D_IIC_EEP_START_ISSUE_WAIT;   /* 開始条件発行待ち状態 */
997
998       }
999
1000      /* TACK 割り込み要因のクリア */
1001      IIC.ICSR.BIT.TACK = 0;
1002
1003      /* EEPROM リード/ライト管理情報設定 */
1004      set_Iic_Eep_Manage_Info(&manage_info);
1005
1006      /* EEPROM アクセス状態設定 */
1007      set_Iic_Eep_Condition(Eep_Condition);
1008
1009      return D_IIC_EEP_OK;
1010
1011  }
1012
1013  /*""FUNC COMMENT""*****************************************************
1014  * ID                    :
1015  * Outline               : WAIT 割り込み時 WritePolling 処理
1016  * Include               :
1017  * Declaration           : E_Iic_eep_Ret iic_Eep_WritePolling_WAIT
1018  *                       :  (void* info)
1019  * Description           : EEPROM ライト終了後のポーリング処理を
1020  *                       : 行います。
1021  * Argument              : void  *info       : No Use
1022  * Return Value          : D_IIC_EEP_OK     : Success
```

```
1023  * Calling Functions       :
1024  *""FUNC COMMENT END""*****************************************/
1025  static E_Iic_eep_Ret iic_Eep_WritePolling_WAIT(void* info)
1026  {
1027
1028      unsigned long    i;
1029
1030      /*
1031        スレーブデバイスが確実に ACK/NACK を応答してくる時間までウェイト
1032        スレーブデバイスのスペック tAA（max900ns）以上ウェイト
1033      */
1034      for(i = 0; i < D_IIC_EEPROM_tAA_WAIT_CNT; i++);
1035
1036      /* ACK,NACK 判定 */
1037      if(IIC.ICSR.BIT.SDAM == 1)      /* NACK 受信 */
1038      {
1039        /* WAIT 割り込み要因のクリア */
1040        IIC.ICSR.BIT.WAIT = 0;
1041
1042        /* NACK 受信時の処理は、TACK 割り込みで制御 */
1043      }
1044      else                          /* ACK 受信 */
1045      {
1046        /* Stop condition generate */
1047        IIC.ICCR = 0x90;
1048
1049        /* WAIT 割り込み要因のクリア */
1050        IIC.ICSR.BIT.WAIT = 0;
1051
1052        /* I2C バス解放確認 */
1053        iic_Chk_Bus_SCL_SDA(D_IIC_EEP_BUS_CHK_END);
1054
1055        /* IIC initialize */
1056        IIC.ICIC.BYTE = 0x00;
1057        IIC.ICSR.BYTE = 0x00;
1058        IIC.ICCR &= ~0x80;
1059
1060        /* EEPROM アクセス状態設定 */
1061        set_Iic_Eep_Condition(D_IIC_EEP_IDLE);    /* アイドル状態 */
1062
1063      }
1064
1065      return D_IIC_EEP_OK;
1066
1067  }
1068
1069  /*****************************************************************************/
1070  /* 内部関数 */
1071  /*""FUNC COMMENT""**************************************************
1072  * ID                    :
1073  * Outline                : Read モード遷移処理
1074  * Include                :
1075  * Declaration            : E_Iic_eep_Ret iic_chg_to_read(void)
1076  *                       :
1077  * Description            : マスタ受信モードに遷移します。
1078  *                       :
1079  * Argument               : none
1080  * Return Value           : D_IIC_EEP_OK     : Success
1081  *                       :
1082  * Calling Functions       :
1083  *""FUNC COMMENT END""*****************************************/
1084  static E_Iic_eep_Ret iic_chg_to_read(void)
1085  {
1086      T_IIC_EEPROM_RW_MANAGE   manage_info;
```

```
1087      unsigned char           dummy;
1088
1089      memset(&manage_info, 0, sizeof(manage_info));
1090
1091      /* 送信から受信に変更 */
1092      IIC.ICCR = 0x81;
1093
1094      /* WAIT 割り込み要因のクリア */
1095      IIC.ICSR.BIT.WAIT = 0;
1096
1097      /* EEPROM アクセス状態設定 */
1098      set_Iic_Eep_Condition(D_IIC_EEP_SEND_RCV_DATA_WAIT);   /* データ送受信待ち状態 */
1099
1100      return D_IIC_EEP_OK;
1101
1102  }
1103
1104  /*""FUNC COMMENT""************************************************
1105  * ID                    :
1106  * Outline                : Memory address 送信処理
1107  * Include                :
1108  * Declaration            : E_Iic_eep_Ret iic_Send_Memory_Address(void)
1109  *                       :
1110  * Description            : Memory address 送信処理を行います。
1111  *                       :
1112  * Argument               : none
1113  * Return Value           : D_IIC_EEP_OK     : Success
1114  *                       :
1115  * Calling Functions       :
1116  *""FUNC COMMENT END""*******************************************/
1117  static E_Iic_eep_Ret iic_Send_Memory_Address(void)
1118  {
1119      T_IIC_EEPROM_RW_MANAGE   manage_info;
1120      E_Iic_eep_Ret            ret = D_IIC_EEP_OK;
1121
1122      memset(&manage_info, 0, sizeof(manage_info));
1123
1124      /* EEPROM リード/ライト管理情報取得 */
1125      get_Iic_Eep_Manage_Info(&manage_info);
1126
1127      /* Memory address set to transfer data */
1128      manage_info.i_MemAdrLen--;
1129      IIC.ICDR = manage_info.i_MemAdrBuf[manage_info.i_MemAdrLen];
1130
1131      /* WAIT 割り込み要因のクリア */
1132      IIC.ICSR.BIT.WAIT = 0;
1133
1134      /* EEPROM リード/ライト管理情報更新 */
1135      set_Iic_Eep_Manage_Info(&manage_info);
1136
1137      return D_IIC_EEP_OK;
1138
1139  }
1140
1141  /*""FUNC COMMENT""************************************************
1142  * ID                    :
1143  * Outline                : Memory address 送信完了後処理
1144  * Include                :
1145  * Declaration            : void iic_After_MemAdd_send(void)
1146  *                       :
1147  * Description            : Memory address 送信完了後処理を行います。
1148  *                       :
1149  * Argument               : none
1150  * Return Value           : none
```

```
1151  * Calling Functions       :
1152  *""FUNC COMMENT END""*********************************************/
1153  static void iic_After_MemAdd_send(void)
1154  {
1155      switch(get_Iic_Eep_Mode())
1156      {
1157      case D_IIC_EEP_READ: /* Read モード */
1158        iic_After_MemAdd_send_Read_Mode();    /* Memory address 送信後 Read 処理 */
1159        break;
1160
1161      case D_IIC_EEP_WRITE:    /* Write モード */
1162        iic_After_MemAdd_send_Write_Mode();       /* Memory address 送信後 Write 処理 */
1163        break;
1164
1165      default:
1166        set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);      /* その他エラー */
1167
1168        break;
1169      }
1170  }
1171
1172  /*""FUNC COMMENT""*************************************************
1173  * ID                    :
1174  * Outline                : Memory address 送信後 Read 処理
1175  * Include                :
1176  * Declaration            : void iic_After_MemAdd_send_Read_Mode(void)
1177  * Description            : Restart condition generate します。
1178  *                       :
1179  *                       :
1180  * Argument               : none
1181  * Return Value           : none
1182  * Calling Functions       :
1183  *""FUNC COMMENT END""*********************************************/
1184  static void iic_After_MemAdd_send_Read_Mode(void)
1185  {
1186      T_IIC_EEPROM_RW_MANAGE  manage_info;
1187
1188      memset(&manage_info, 0, sizeof(manage_info));
1189
1190      /* DTE interrupt enable */
1191      IIC.ICIC.BIT.DTEE = 1;
1192
1193      /* ReStart condition generate */
1194      IIC.ICCR = 0x94;
1195
1196      /* WAIT 割り込み要因のクリア */
1197      IIC.ICSR.BIT.WAIT = 0;
1198
1199      /* EEPROM アクセス状態設定 */
1200      set_Iic_Eep_Condition(D_IIC_EEP_START_ISSUE_WAIT); /* 開始条件発行待ち状態 */
1201
1202      return;
1203
1204  }
1205
1206  /*""FUNC COMMENT""*************************************************
1207  * ID                    :
1208  * Outline                : Memory address 送信後 Write 処理
1209  * Include                :
1210  * Declaration            : void iic_After_MemAdd_send_Write_Mode(void)
1211  * Description            : 初回ライトデータを ICDR に設定します。
1212  *                       :
1213  *                       :
1214  * Argument               : none
```

```
1215  * Return Value           : none
1216  * Calling Functions       :
1217  *""FUNC COMMENT END""*****************************************/
1218  static void iic_After_MemAdd_send_Write_Mode(void)
1219  {
1220      T_IIC_EEPROM_RW_MANAGE   manage_info;
1221
1222      memset(&manage_info, 0, sizeof(manage_info));
1223
1224      /* EEPROM リード/ライト管理情報取得 */
1225      get_Iic_Eep_Manage_Info(&manage_info);
1226
1227      /* Write Data to transfer data */
1228      IIC.ICDR = *manage_info.i_RW_Data_info.i_DataBuf;
1229      manage_info.i_RW_Data_info.i_DataBuf++;
1230      manage_info.i_RW_Data_info.i_DataLen--;
1231
1232      if(manage_info.i_RW_Data_info.i_DataLen == 0)/* 送信データ残り 0 バイト？ */
1233      {
1234        /* Stop condition generate */
1235        IIC.ICCR = 0x90;
1236      }
1237
1238      /* WAIT 割り込み要因のクリア */
1239      IIC.ICSR.BIT.WAIT = 0;
1240
1241      /* EEPROM リード/ライト管理情報更新 */
1242      set_Iic_Eep_Manage_Info(&manage_info);
1243
1244      /* EEPROM アクセス状態設定 */
1245      set_Iic_Eep_Condition(D_IIC_EEP_SEND_RCV_DATA_WAIT);   /* データ送信待ち状態 */
1246
1247      return;
1248
1249  }
1250
1251  /*""FUNC COMMENT""*************************************************
1252  * ID                    :
1253  * Outline                : WAIT 割り込み時データ送受信待ち状態 Read 処理
1254  * Include                :
1255  * Declaration            : E_Iic_eep_Ret iic_Read_Data_rcv_WAIT(void)
1256  * Description            : WAIT 割り込み時のデータ受信処理を行います。
1257  *                       :
1258  * Argument               : void             :
1259  * Return Value           : D_IIC_EEP_OK     : Success
1260  *                       : D_IIC_EEP_NG       : Error
1261  * Calling Functions       :
1262  *""FUNC COMMENT END""*****************************************/
1263  static E_Iic_eep_Ret iic_Read_Data_rcv_WAIT(void)
1264  {
1265      E_Iic_eep_condition     Eep_Condition;
1266      T_IIC_EEPROM_RW_MANAGE  manage_info;
1267      E_Iic_eep_Ret           ret = D_IIC_EEP_OK;
1268
1269      memset(&manage_info, 0, sizeof(manage_info));
1270
1271      /* EEPROM リード/ライト管理情報取得 */
1272      get_Iic_Eep_Manage_Info(&manage_info);
1273
1274      if(manage_info.i_RW_Data_info.i_DataLen > 1)          /* 受信予定データ数残り > 1 */
1275      {
1276        if(IIC.ICSR.BIT.DTE == 1)
1277        {
1278            *manage_info.i_RW_Data_info.i_DataBuf = IIC.ICDR;
```

```
1279            manage_info.i_RW_Data_info.i_DataBuf++;
1280            manage_info.i_RW_Data_info.i_DataLen--;
1281
1282            if(manage_info.i_RW_Data_info.i_DataLen == 1)/* 受信予定データ数残り == 1 */
1283            {
1284                /* Stop condition generate */
1285                IIC.ICCR = 0xC0;
1286            }
1287          }
1288      }
1289      else if(manage_info.i_RW_Data_info.i_DataLen == 1)     /* 受信予定データ数残り == 1 */
1290      {
1291        /* Stop condition generate */
1292        IIC.ICCR = 0xC0;
1293      }
1294      else
1295      {
1296        /* ありえないケース */
1297        set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_OTHER);      /* その他エラー */
1298        ret = D_IIC_EEP_NG;
1299
1300      }
1301
1302      /* EEPROM リード/ライト管理情報更新 */
1303      set_Iic_Eep_Manage_Info(&manage_info);
1304
1305      /* DTE interrupt enable */
1306      IIC.ICIC.BIT.DTEE = 1;
1307
1308      /* WAIT 割り込み要因のクリア */
1309      IIC.ICSR.BIT.WAIT = 0;
1310
1311      return ret;
1312
1313  }
1314
1315  /*""FUNC COMMENT""************************************************
1316  * ID                    :
1317  * Outline               :データ送受信待ち状態 Write 処理
1318  * Include               :
1319  * Declaration            : E_Iic_eep_Ret iic_Write_Data_sending (void)
1320  *                       :
1321  * Description            : WAIT 割り込み時のデータ送信処理を行います。
1322  *                       :
1323  * Argument               : void             :
1324  * Return Value           : D_IIC_EEP_OK     : Success
1325  *                       :
1326  * Calling Functions       :
1327  *""FUNC COMMENT END""*******************************************/
1328  static E_Iic_eep_Ret iic_Write_Data_sending(void)
1329  {
1330      T_IIC_EEPROM_RW_MANAGE  manage_info;
1331
1332      memset(&manage_info, 0, sizeof(manage_info));
1333
1334      /* EEPROM リード/ライト管理情報取得 */
1335      get_Iic_Eep_Manage_Info(&manage_info);
1336
1337      if(manage_info.i_RW_Data_info.i_DataLen == 1)  /* 送信データ残り 1 バイト？ */
1338      {
1339        /* Write Data to transfer data */
1340        IIC.ICDR = *manage_info.i_RW_Data_info.i_DataBuf;
1341        manage_info.i_RW_Data_info.i_DataBuf++;
1342        manage_info.i_RW_Data_info.i_DataLen--;
```

```
1343
1344        /* EEPROM リード/ライト管理情報更新 */
1345        set_Iic_Eep_Manage_Info(&manage_info);
1346
1347        /* Stop condition generate */
1348        IIC.ICCR = 0x90;
1349
1350        /* WAIT 割り込み要因のクリア */
1351        IIC.ICSR.BIT.WAIT = 0;
1352
1353      }
1354      else if(manage_info.i_RW_Data_info.i_DataLen == 0)     /* 送信データ残り 0 バイト？ */
1355      {
1356
1357        /* WAIT 割り込み要因のクリア */
1358        IIC.ICSR.BIT.WAIT = 0;
1359
1360        /* 停止条件発行後 Write Polling 準備処理 */
1361        iic_After_end_issue_Write_Polling();
1362      }
1363      else
1364      {
1365
1366        /* Write Data to transfer data */
1367        IIC.ICDR = *manage_info.i_RW_Data_info.i_DataBuf;
1368        manage_info.i_RW_Data_info.i_DataBuf++;
1369        manage_info.i_RW_Data_info.i_DataLen--;
1370
1371        /* EEPROM リード/ライト管理情報更新 */
1372        set_Iic_Eep_Manage_Info(&manage_info);
1373
1374        /* WAIT 割り込み要因のクリア */
1375        IIC.ICSR.BIT.WAIT = 0;
1376
1377      }
1378
1379      return D_IIC_EEP_OK;
1380
1381  }
1382
1383  /*""FUNC COMMENT""************************************************
1384  * ID                    :
1385  * Outline               : 停止条件発行後 Write Polling 準備処理
1386  * Include               :
1387  * Declaration           : void iic_After_end_issue_Write_Polling(void)
1388  *                       :
1389  * Description           : Write Polling 準備処理を行います。
1390  *                       :
1391  * Argument              : void             :
1392  * Return Value          : void             :
1393  * Calling Functions     :
1394  *""FUNC COMMENT END""*******************************************/
1395  static void iic_After_end_issue_Write_Polling(void)
1396  {
1397      T_IIC_EEPROM_RW_MANAGE  manage_info;
1398
1399      memset(&manage_info, 0, sizeof(manage_info));
1400
1401      /* I2C バス解放確認 */
1402      iic_Chk_Bus_SCL_SDA(D_IIC_EEP_BUS_CHK_END);
1403
1404      /* IIC 設定初期化 */
1405      IIC.ICIC.BYTE = 0x00;
1406      IIC.ICSR.BYTE = 0x00;
```

```
1407      IIC.ICCR &= ~0x80;
1408
1409      /* EEPROM リード/ライト管理情報取得 */
1410      get_Iic_Eep_Manage_Info(&manage_info);
1411
1412      /* Write Polling cycle set */
1413      manage_info.i_WCTCnt = D_IIC_WCT_CNT;
1414
1415      /* IIC 再設定 */
1416      /* ==== IIC.ICCR の設定 ==== */
1417      IIC.ICCR |= 0x80;                   /* 動作可能状態 */
1418
1419      /* ==== IIC.ICCL の設定 ==== */
1420      IIC.ICCL = 0x33;
1421
1422      /* ==== IIC.ICCH の設定 ==== */
1423      IIC.ICCH = 0x20;
1424
1425      /* ALE interrupt enable */
1426      IIC.ICIC.BIT.ALE = 1;
1427
1428      /* TACKE interrupt disable */
1429      IIC.ICIC.BIT.TACKE = 0;
1430
1431      /* WAIT interrupt disable */
1432      IIC.ICIC.BIT.WAITE = 0;
1433
1434      /* DTE interrupt enable */
1435      IIC.ICIC.BIT.DTEE = 1;
1436
1437      /* Start condition generate */
1438      IIC.ICCR = 0x94;
1439
1440      /* EEPROM リード/ライト管理情報更新 */
1441      set_Iic_Eep_Manage_Info(&manage_info);
1442
1443      /* EEPROM アクセス状態設定 */
1444      set_Iic_Eep_Condition(D_IIC_EEP_START_ISSUE_WAIT); /* 開始条件発行待ち状態 */
1445
1446      return;
1447
1448  }
1449
1450  /*""FUNC COMMENT""************************************************
1451  * ID                    :
1452  * Outline               : I2C バス解放確認処理
1453  * Include               :
1454  * Declaration           : E_Iic_eep_Ret iic_Chk_Bus_SCL_SDA
1455  *                       :   (E_Iic_eep_mode_bus_chk i_state)
1456  *                       :
1457  *                       :
1458  * Description           : ・スレーブデバイス状態確認を確認します。
1459  *                       :  ※スレーブデバイス暴走確認
1460  *                       :  (I2C ラインを Low に引っ張り続けていないか？)
1461  *                       :  D_IIC_BUSYCHECK_TMOUT の確認用タイマーに
1462  *                       :  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。
1463  *                       : ・I2C バスの占有／解放状態を確認します。
1464  *                       :
1465  * Argument              : E_Iic_eep_mode_bus_chk i_state
1466  *                       : D_IIC_EEP_BUS_CHK_START : 開始時指定
1467  *                       : D_IIC_EEP_BUS_CHK_END   : 終了時指定
1468  *                       :
1469  * Return Value          : D_IIC_EEP_OK         :I2C バス解放
1470  *                       : D_IIC_EEP_NG         :I2C バス非解放
```

```
1471  * Calling Functions       :
1472  *""FUNC COMMENT END""*********************************************/
1473  static E_Iic_eep_Ret iic_Chk_Bus_SCL_SDA(E_Iic_eep_mode_bus_chk i_state)
1474  {
1475      unsigned long    i;
1476      E_Iic_eep_Ret    ret = D_IIC_EEP_OK;
1477
1478      for (i = D_IIC_BUSYCHECK_TMOUT; i > 0; i--)
1479      {
1480        if ( (IIC.ICSR.BYTE & (D_IIC_BUSY_BIT | D_IIC_SCLSDA_BIT)) == D_IIC_SCLSDA_ON )
1481        {
1482            break;                              /* I2C バス解放 */
1483        }
1484      }
1485
1486      /* I2C バス非解放? */
1487      if(i == 0)
1488      {
1489        ret = D_IIC_EEP_NG;
1490
1491        /* 本応用例では起動完了後、I2C バス解放状態にならない場合、無限ループとしています。
1492           システムに見合った処理を実装ください */
1493        if(i_state == D_IIC_EEP_BUS_CHK_END)
1494        {
1495            while(1);
1496        }
1497      }
1498
1499      return ret;
1500
1501  }
1502
1503  /*""FUNC COMMENT""**************************************************
1504  * ID                    :
1505  * Outline               : IIC_EEPROM 内部情報初期化処理
1506  * Include               :
1507  * Declaration           : void set_internal_info_init(void)
1508  * Description           : IIC_EEPROM 内部情報を初期化します。
1509  *                       :
1510  * Argument              : none
1511  * Return Value          : none
1512  * Calling Functions     :
1513  *""FUNC COMMENT END""*********************************************/
1514  static void set_internal_info_init(void)
1515  {
1516      gIic_Eep_Condition = D_IIC_EEP_NO_INIT;           /* IIC_EEPROM 状態 */
1517      gIic_Eep_Before_Condition = D_IIC_EEP_NO_INIT;    /* IIC_EEPROM 前状態 */
1518
1519      set_Iic_Eep_Err_Condition(D_IIC_EEP_ERR_NO);      /* エラーなし状態 */
1520
1521      /* EEPROM リード/ライト管理情報初期化 */
1522      memset(&gIic_Eep_Manage_Info, 0, sizeof(gIic_Eep_Manage_Info));
1523
1524  }
1525
1526  /*""FUNC COMMENT""**************************************************
1527  * ID                    :
1528  * Outline               : IIC_EEPROM 内部状態設定
1529  * Include               :
1530  * Declaration           : void set_Iic_Eep_Condition(
1531  *                       :     E_Iic_eep_condition i_condition)
1532  * Description           : IIC_EEPROM 内部状態設定を行います。
1533  *                       :
1534  * Argument              : E_Iic_eep_condition i_condition
```

```
1535  *                     : iic_eeprom.h の E_Iic_eep_condition の
1536  *                     : 状態を設定します。
1537  * Return Value          : none
1538  * Calling Functions      :
1539  *""FUNC COMMENT END""*********************************************/
1540  void set_Iic_Eep_Condition(E_Iic_eep_condition i_condition)
1541  {
1542      gIic_Eep_Before_Condition = gIic_Eep_Condition;    /* 前状態設定 */
1543      gIic_Eep_Condition = i_condition;                  /* 現状態設定 */
1544  }
1545
1546  /*""FUNC COMMENT""**************************************************
1547  * ID                   :
1548  * Outline               : IIC_EEPROM 内部現状態取得
1549  * Include               :
1550  * Declaration           : E_Iic_eep_condition
1551  *                      :     get_Iic_Eep_Condition(void)
1552  * Description           : IIC_EEPROM 内部現状態取得を行います。
1553  *                      :
1554  * Argument              : none
1555  * Return Value          : E_Iic_eep_condition
1556  *                      : iic_eeprom.h の E_Iic_eep_condition の
1557  *                      : 状態を取得します。
1558  * Calling Functions      :
1559  *""FUNC COMMENT END""*********************************************/
1560  E_Iic_eep_condition get_Iic_Eep_Condition(void)
1561  {
1562      return gIic_Eep_Condition;
1563  }
1564
1565  /*""FUNC COMMENT""**************************************************
1566  * ID                   :
1567  * Outline               : IIC_EEPROM 内部前状態設定
1568  * Include               :
1569  * Declaration           : void set_Iic_Eep_Before_Condition(
1570  *                      :    E_Iic_eep_condition i_condition)
1571  * Description           : IIC_EEPROM 内部前状態設定を行います。
1572  *                      :
1573  * Argument              : E_Iic_eep_condition i_condition
1574  *                      : iic_eeprom.h の E_Iic_eep_condition の
1575  *                      : 状態を設定します。
1576  * Return Value          : none
1577  * Calling Functions      :
1578  *""FUNC COMMENT END""*********************************************/
1579  void set_Iic_Eep_Before_Condition(E_Iic_eep_condition i_condition)
1580  {
1581      gIic_Eep_Before_Condition = i_condition;           /* 前状態設定 */
1582  }
1583
1584  /*""FUNC COMMENT""**************************************************
1585  * ID                   :
1586  * Outline               : IIC_EEPROM 内部前状態取得
1587  * Include               :
1588  * Declaration           : E_Iic_eep_condition
1589  *                      :     get_Iic_Eep_Before_Condition(void)
1590  * Description           : IIC_EEPROM 内部前状態取得を行います。
1591  *                      :
1592  * Argument              : none
1593  * Return Value          : E_Iic_eep_condition
1594  *                      : iic_eeprom.h の E_Iic_eep_condition の
1595  *                      : 状態を取得します。
1596  * Calling Functions      :
1597  *""FUNC COMMENT END""*********************************************/
1598  E_Iic_eep_condition get_Iic_Eep_Before_Condition(void)
```

```
1599  {
1600      return gIic_Eep_Before_Condition;
1601  }
1602
1603  /*""FUNC COMMENT""************************************************
1604  * ID                    :
1605  * Outline               : IIC_EEPROM 内部モード取得
1606  * Include               :
1607  * Declaration           : E_Iic_eep_mode
1608  *                       :     get_Iic_Eep_Mode(void)
1609  * Description           : IIC 内部モード取得を行います。
1610  *                       :
1611  * Argument              : none
1612  * Return Value          : E_Iic_eep_mode
1613  *                       : iic_eeprom.h の E_Iic_eep_mode の
1614  *                       : モードを取得します。
1615  * Calling Functions     :
1616  *""FUNC COMMENT END""*****************************************/
1617  E_Iic_eep_mode get_Iic_Eep_Mode(void)
1618  {
1619      return gIic_Eep_Manage_Info.i_mode;
1620  }
1621
1622  /*""FUNC COMMENT""************************************************
1623  * ID                    :
1624  * Outline               : IIC_EEPROM 内部管理情報設定
1625  * Include               :
1626  * Declaration           : void set_Iic_Eep_Manage_Info(
1627  *                       :     T_IIC_EEPROM_RW_MANAGE *i_info)
1628  * Description           : IIC_EEPROM 内部管理情報の設定を行います。
1629  *                       :
1630  * Argument              : T_IIC_EEPROM_RW_MANAGE *i_info
1631  * Return Value          : none
1632  * Calling Functions     :
1633  *""FUNC COMMENT END""*****************************************/
1634  void set_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *i_info)
1635  {
1636      memcpy(&gIic_Eep_Manage_Info, i_info, sizeof(T_IIC_EEPROM_RW_MANAGE));
1637  }
1638
1639  /*""FUNC COMMENT""************************************************
1640  * ID                    :
1641  * Outline               : IIC_EEPROM 内部管理情報取得
1642  * Include               :
1643  * Declaration           :  void get_Iic_Eep_Manage_Info(
1644  *                       :     T_IIC_EEPROM_RW_MANAGE *o_info)
1645  * Description           : IIC_EEPROM 内部管理情報の取得を行います。
1646  *                       :
1647  * Argument              : T_IIC_EEPROM_RW_MANAGE *o_info
1648  * Return Value          : none
1649  * Calling Functions     :
1650  *""FUNC COMMENT END""*****************************************/
1651  void get_Iic_Eep_Manage_Info(T_IIC_EEPROM_RW_MANAGE *o_info)
1652  {
1653      memcpy(o_info, &gIic_Eep_Manage_Info, sizeof(T_IIC_EEPROM_RW_MANAGE));
1654  }
1655
1656  /*""FUNC COMMENT""************************************************
1657  * ID                    :
1658  * Outline               : IIC_EEPROM 内部エラー情報設定
1659  * Include               :
1660  * Declaration           :  void set_Iic_Eep_Err_Condition(
1661  *                       :     E_Iic_eep_err_condition i_err)
1662  * Description           : IIC_EEPROM 内部エラー情報の設定を行います。
```

```
1663  *                     :
1664  * Argument              : E_Iic_eep_err_condition i_err
1665  * Return Value          : none
1666  * Calling Functions     :
1667  *""FUNC COMMENT END""*****************************************/
1668  void set_Iic_Eep_Err_Condition(E_Iic_eep_err_condition i_err)
1669  {
1670      gIic_Eep_Err_Condition = i_err;
1671  }
1672
1673  /*""FUNC COMMENT""*************************************************
1674  * ID                   :
1675  * Outline               : IIC_EEPROM 内部エラー情報取得
1676  * Include               :
1677  * Declaration           :  E_Iic_eep_err_condition
1678  *                      :    get_Iic_Eep_Err_Condition()
1679  * Description           : IIC_EEPROM 内部エラー情報の取得を行います。
1680  *                      :
1681  * Argument              : none
1682  * Return Value          : E_Iic_eep_err_condition
1683  * Calling Functions     :
1684  *""FUNC COMMENT END""*****************************************/
1685  E_Iic_eep_err_condition get_Iic_Eep_Err_Condition(void)
1686  {
1687      return gIic_Eep_Err_Condition;
1688  }
1689
1690  /* End of File */
```

(4) サンプルプログラムリスト"iic_eeprom.h"

```
/***********************************************************************
* DISCLAIMER

* This software is supplied by Renesas Electronics Corporation. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Electronics Corporation. and is protected under
* all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
***********************************************************************/
/* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved. */
/*""FILE COMMENT""*********** Technical reference data ***************
* System Name  : SH7722,SH7731 Sample Program
* File Name    : iic_eeprom.h
* Abstract     : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
* Version      : Ver 1.00
* Device       : SH7722,SH7731
* Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
*              : C/C++ Compiler Package for SuperH Family (V.9.03 release00)
* OS           : None
* H/W Platform : R0P7722TH001ARK for SH7722 Reference Platform
* Description  : SH7722,SH7731 IIC 送受信例（EEPROM へのリード・ライト）
*              :
* Operation    :
* Limitation   :
*              :
*********************************************************************
* History      : 18.Jun.2010 Ver. 1.00 First Release
*""FILE COMMENT END""*************************************************/

#ifndef __IIC_EEPROM_DEF_H__
#define __IIC_EEPROM_DEF_H__

/* ==== マクロ定義 ==== */

/* IIC の割り込みマスクレベル設定値 */
#define D_IIC_EEP_INT_LEVEL     2

/* Memory address size */
#define D_IIC_EEP_MEMADR_SIZE   2

/* Write Polling Count 10ms (400Kbps base) */
#define D_IIC_WCT_CNT    480

/* Device Code */
```

```
63    #define D_IIC_DEV_CODE   0xA0
64
65    /* Write code */
66    #define D_IIC_W_CODE     0x00
67
68    /* EPPROM tAA wait time (1μs以上)*/
69    #define D_IIC_EEPROM_tAA_WAIT_CNT   100
70
71    /* Read code */
72    #define D_IIC_R_CODE     0x01
73
74    /* iic busy check */
75    #define D_IIC_BUSY_BIT       0x10
76
77    /* iic scl,sda check */
78    #define D_IIC_SCLSDA_BIT 0xc0
79    #define D_IIC_SCLSDA_ON      D_IIC_SCLSDA_BIT
80
81    /* timeout set */
82    #define D_IIC_BUSYCHECK_TMOUT   150000
83
84    /* 戻り値 */
85    typedef enum
86    {
87        D_IIC_EEP_NG = -1,
88        D_IIC_EEP_OK = 0,
89        D_IIC_EEP_NOP,
90        D_IIC_EEP_READING,
91        D_IIC_EEP_WRITING,
92        D_IIC_EEP_BUS_BUSY,
93
94    } E_Iic_eep_Ret;
95
96    /* NACK判定用 戻り値 */
97    typedef enum
98    {
99        D_IIC_NACK_FOUND = 0,
100       D_IIC_NACK_NOTFOUND
101   } E_Iic_eep_Nack;
102
103   /* EEPROMアクセス状態 */
104   typedef enum
105   {
106      D_IIC_EEP_NO_INIT,                  /* 未初期化状態 */
107      D_IIC_EEP_IDLE,                     /* アイドル状態 */
108      D_IIC_EEP_START_ISSUE_WAIT,         /* 開始条件発行待ち状態 */
109      D_IIC_EEP_SEND_DEVADD_WAIT,         /* Device Address送信待ち状態 */
110      D_IIC_EEP_RESEND_DEVADD_WAIT,       /* Device Address再送信待ち状態 */
111      D_IIC_EEP_SEND_MEMADD_WAIT,         /* Memory address送信待ち状態 */
112      D_IIC_EEP_SEND_RCV_DATA_WAIT,       /* データ送受信待ち状態 */
113      D_IIC_EEP_WRITE_POLLING_WAIT,       /* WritePolling待ち状態 */
114      D_IIC_EEP_CONDITION_MAX             /* ここ以降定義禁止 */
115
116   } E_Iic_eep_condition;
117
118   /* EEPROMアクセスイベント */
119   typedef enum
120   {
121      D_IIC_EEP_EV_INIT,                  /* iic_user_Init()コール時 */
122      D_IIC_EEP_EV_RW_START,              /* iic_user_EepRomRW()コール時 */
123      D_IIC_EEP_EV_CHECK,                 /* iic_user_Chk_EepRom()コール時 */
124      D_IIC_EEP_EV_INT_AL,              /* AL割り込み発生時 */
125      D_IIC_EEP_EV_INT_TACK,              /* TACK割り込み発生時 */
126      D_IIC_EEP_EV_INT_WAIT,              /* WAIT割り込み発生時 */
```

```
127      D_IIC_EEP_EV_INT_DTE,            /* DTE 割り込み発生時 */
128      D_IIC_EEP_EVENT_MAX                  /* ここ以降定義禁止 */
129
130  } E_Iic_eep_event;
131
132  /* EEPROM アクセスエラー状態 */
133  typedef enum
134  {
135      D_IIC_EEP_ERR_NO = 0,            /* エラーなし状態 */
136      D_IIC_EEP_ERR_NACK,                  /* 送信時 NACK 受信 */
137      D_IIC_EEP_ERR_AL,                /* AL 発生 */
138      D_IIC_EEP_ERR_WCT_OVER,              /* Write cycle time over */
139      D_IIC_EEP_ERR_OTHER,             /* その他エラー発生 */
140
141  } E_Iic_eep_err_condition;
142
143  /* EEPROM 動作モード */
144  typedef enum
145  {
146      D_IIC_EEP_NO_MODE = 0,           /* mode なし */
147      D_IIC_EEP_WRITE,             /* Write モード */
148      D_IIC_EEP_READ,                  /* Random Read Operation と Sequential Read Operation 組み合わ
149  せモード */
150      D_IIC_EEP_CURRENT_READ           /* Current Address Read と Sequential Read Operation 組み合わ
151  せモード */
152
153  } E_Iic_eep_mode;
154
155  /* I2C 転送状態チェック */
156  typedef enum
157  {
158      D_IIC_EEP_BUS_CHK_START = 0,     /* 起動前 I2C 転送状態チェック */
159      D_IIC_EEP_BUS_CHK_END,               /* 起動後 I2C 転送状態チェック */
160
161  } E_Iic_eep_mode_bus_chk;
162
163  /** 状態遷移表構造体 */
164  typedef struct t_cbs_mtx_ev_tbl
165  {
166     E_Iic_eep_condition  event_type; /* 受信イベント      */
167     E_Iic_eep_Ret  (*proc)( void *); /* ハンドラ          */
168  } T_IIC_EEPROM_MTX_TBL ;
169
170  /* EEPROM リード/ライト情報 */
171  typedef struct {
172
173      E_Iic_eep_mode          i_mode;        /* mode */
174      unsigned char           i_DevAdr;      /* Device address */
175      unsigned long           i_RomAdr;      /* EEPROM reading address */
176      unsigned long           i_Len;         /* Transfer/Receive data length */
177      unsigned char           *i_pBuf;       /* Transfer/Receive data buffer pointer */
178
179  } T_IIC_EEPROM_RW_INFO;
180
181  /* EEPROM リード/ライトデータ情報 */
182  typedef struct {
183
184      unsigned long           i_DataLen;     /* R/W Data length */
185      unsigned char           *i_DataBuf;    /* R/W Data Buffer pointer */
186
187  } T_IIC_EEPROM_RW_DATA_INFO;
188
189  /* EEPROM リード/ライト管理情報 */
190  typedef struct {
```

```
191
192       E_Iic_eep_mode              i_mode;                         /* mode */
193       unsigned char               i_DevAdr;                       /* Device address */
194       unsigned long               i_MemAdrLen;                    /* Memory address length */
195       unsigned char               i_MemAdrBuf[D_IIC_EEP_MEMADR_SIZE];   /* Memory address buffer */
196       T_IIC_EEPROM_RW_DATA_INFO   i_RW_Data_info;                 /* RW Data Info */
197       unsigned short              i_WCTCnt;                       /* Write polling counter */
198
199   } T_IIC_EEPROM_RW_MANAGE;
200
201   /* EEPROM 状態/エラー状態情報 */
202   typedef struct {
203
204       E_Iic_eep_condition         i_eep_condition;   /* EEPROM 状態 */
205       E_Iic_eep_err_condition     i_err_condition;   /* EEPROM エラー状態 */
206
207   } T_IIC_EEPROM_CONDITION;
208
209
210   /* ==== 関数宣言 ==== */
211   E_Iic_eep_Ret iic_mtx_executeFuncTable(E_Iic_eep_event event, void *info);
212
213   /* ==== User 提供 IF ==== */
214   E_Iic_eep_Ret iic_user_Init(void);
215   E_Iic_eep_Ret iic_user_EepRomRW(T_IIC_EEPROM_RW_INFO *i_RW_Info);
216   E_Iic_eep_Ret iic_user_Chk_EepRom(T_IIC_EEPROM_CONDITION *o_condition_info);
217
218
219   #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'E00 IIC ALI */
5     void INT_IIC_ALI(void)
6     {
7         iic_user_int_AL();
8     }
9
10    /* H'E20 IIC TACKI */
11    void INT_IIC_TACKI(void)
12    {
13        iic_user_int_TACK();
14    }
15
16    /* H'E40 IIC WAITI */
17    void INT_IIC_WAITI(void)
18    {
19        iic_user_int_WAIT();
20    }
21
22    /* H'E60 IIC DTEI */
23    void INT_IIC_DTEI(void)
24    {
25        iic_user_int_DTE();
26    }
27
28    …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     ;IIC
5                 ;H'E00 IIC ALI
6                 .data.b    H'10
7                 ;H'E20 IIC TACKI
8                 .data.b    H'10
9                 ;H'E40 IIC WAITI
10                .data.b    H'10
11                ;H'E60 IIC DTEI
12                .data.b    H'10
13
14    …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 33 のように、提供インタフェースの EEPROM リード/ライト処理 (iic_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。

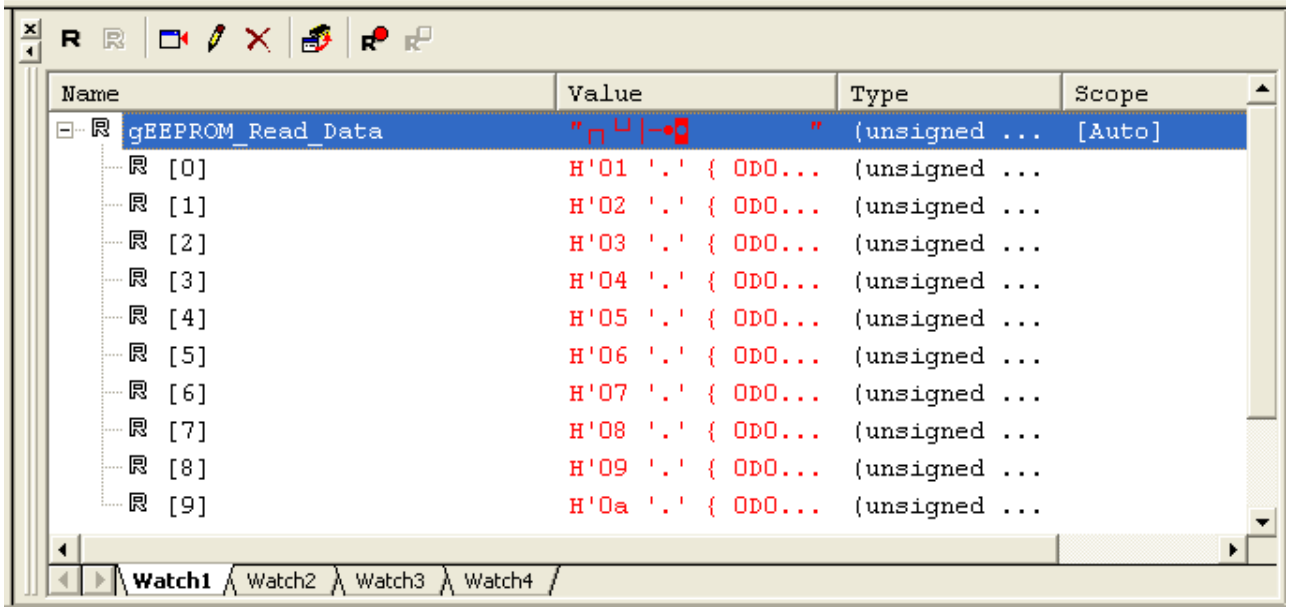

図 33　gEEPROM_Read_Data[D_IIC_EEPROM_READ_DATA_NUM]のメモリダンプ

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
  SH7722 グループ ハードウェアマニュアル (RJJ09B0324)
  (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	2010.09.15	—	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