

RL78/G14、RL78/G1C、RL78/L12、RL78/L13、RL78/L1C グループ IICA を使った I²C シングルマスタ制御ソフトウェア

R01AN1074JJ0104
Rev.1.04
2016.03.31

要旨

本アプリケーションノートでは、RL78/G14、RL78/G1C、RL78/L12、RL78/L13、RL78/L1C グループ シリアル・インタフェース IICA（以下、IICA）を使用したシングルマスタ制御方法とサンプルコードの使用方法を説明します。

本サンプルコードでは、スレーブデバイスを制御するためのソフトウェアを上位層、I²C シングルマスタ基本プロトコル制御を実現するソフトウェアを下位層としています。上位層は下位層で用意されているプロトコルを組み合わせることでスレーブデバイスを制御します。

本サンプルコードは、I²C シングルマスタ制御するための下位層に位置するソフトウェアです。本サンプルコードを使用して、上位層にあたるスレーブデバイスを制御するためのソフトウェアを作成してください。

なお、スレーブデバイス制御のためのソフトウェア例を別途用意していますので、必要に応じて以下から入手してください。なお、スレーブデバイス制御ソフトウェアが追加になった場合、本アプリケーションノートの更新が間に合わないことがあります。最新のスレーブデバイス制御ソフトウェアとの組み合わせ情報は、以下の URL を参照してください。

- I²C シリアル EEPROM ドライバ
http://japan.renesas.com/driver/i2c_serial_eeprom

対象デバイス

対応 MCU RL78/G14、RL78/G1C グループ
 RL78/L12、RL78/L13、RL78/L1C グループ

動作確認に使用したデバイス ルネサス エレクトロニクス製 R1EX24xxx シリーズ I²C Serial EEPROM

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様に合わせて変更し、十分評価してください。

なお、以降では、対象デバイスが複数グループ MCU であるため、説明の都合上 “RL78 ファミリ MCU” として記述しています。

目次

1. 仕様.....	4
2. 動作確認条件.....	6
3. 関連アプリケーションノート.....	12
4. 周辺機能説明.....	12
5. ハードウェア説明.....	13
5.1 使用端子一覧.....	13
5.2 参考回路.....	13
5.3 複数スレーブデバイス制御.....	14
5.4 最大転送速度.....	14
6. ソフトウェア説明.....	15
6.1 ソフトウェア構成.....	15
6.2 動作概要.....	16
6.2.1 マスタ送信.....	16
6.2.2 マスタ受信.....	18
6.2.3 マスタ複合.....	19
6.3 ソフトウェア動作.....	20
6.4 ソフトウェア動作シーケンス.....	21
6.5 スレーブデバイス制御の実現方法.....	22
6.6 通信の実現方法.....	23
6.6.1 制御時の状態.....	23
6.6.2 制御時のイベント.....	23
6.6.3 プロトコル状態遷移.....	24
6.6.4 プロトコル状態遷移表.....	28
6.6.5 プロトコル状態遷移登録関数.....	28
6.6.6 プロトコル状態遷移時の処理.....	29
6.7 割り込み発生タイミング.....	30
6.7.1 マスタ送信.....	30
6.7.2 マスタ受信.....	31
6.7.3 マスタ複合.....	31
6.8 コールバック関数.....	31
6.9 データバッファと送信／受信データの関係.....	32
6.10 必要メモリサイズ.....	33
6.11 ファイル構成.....	36
6.12 定数一覧.....	37
6.12.1 各種定義.....	37
6.13 構造体／共用体一覧.....	39
6.13.1 I ² C 通信情報構造体.....	39
6.13.2 内部情報管理構造体.....	41
6.14 列挙型.....	42
6.15 変数一覧.....	43
6.16 関数一覧.....	44

6.17	状態遷移図	45
6.17.1	エラー状態の定義	46
6.17.2	状態遷移時の各フラグの状態	47
6.18	関数仕様	48
6.18.1	関数共通処理	48
6.18.2	I ² C ドライバ初期化関数	53
6.18.3	マスタ送信開始関数	57
6.18.4	マスタ受信開始関数	61
6.18.5	マスタ複合開始関数	65
6.18.6	アドバンス関数	69
6.18.7	SCL 疑似クロック生成関数	74
6.18.8	I ² C ドライバリセット関数	77
7.	応用例	79
7.1	r_iic_drv_api.h	79
7.2	r_iic_drv_sfr.h	81
7.3	r_iic_drv_int.c	83
7.3.1	統合開発環境 CS+ for CA,CX (旧 CubeSuite+) の場合	83
7.3.2	統合開発環境 CS+ for CC の場合	83
7.3.3	統合開発環境 IAR Embedded Workbench の場合	83
7.4	r_iic_drv_sfr.c	85
7.5	r_iic_drv_os.c	85
7.6	復帰処理例	86
8.	使用上の注意事項	88
8.1	組み込み時の注意事項	88
8.2	初期化時の注意事項	88
8.3	チャンネル状態フラグとデバイス状態フラグについて	88
8.4	同一チャンネル上の複数スレーブデバイス制御時のアドバンス関数のコール方法	88
8.5	アドバンス関数を割り込み内で処理する場合と OS 制御について	88
8.6	転送速度設定	89
8.7	#define 定義「IICAx_ENABLE」と「MAX_IIC_CH_NUM」の設定について	89
8.8	割り込み関数 #pragma interrupt の定義について	89
8.9	ユーザ API コール時の注意事項	89
8.10	型宣言の重複によるワーニングについて	89
8.11	コンパイル時の注意事項	90

1. 仕様

RL78 ファミリー MCU のシリアル・インタフェース IICA を使用し、I²C シングルマスタ制御を行います。スレーブデバイス制御のための上位層に位置するソフトウェアを入手、もしくは作成してください。

表 1-2 に使用する周辺機器と用途を、図 1-1 に使用例を示します。

以下に、機能概略を示します。

- マスタデバイスを RL78 ファミリー MCU とし、シリアル・インタフェース IICA を使った I²C シングルマスタ用デバイスドライバです。
- I²C バス規格のプロトコルを実現します。本サンプルコードでは、マスタ送信、マスタ受信、マスタ複合（マスタ送信→マスタ受信）をサポートします。
- マスタ送信は 4 種類の送信パターンを設定可能です。動作パターンを表 1-1 に示します。
- 複数のチャンネルをサポートします。複数チャンネルを使用した同時通信が可能です。
- 1 つのチャンネル・バス上の複数かつ型名が異なるスレーブデバイスを制御できます。ただし、通信中（スタートコンディション生成から、ストップコンディション生成完了までの期間）は、そのデバイス以外の通信はできません。
- 各プロトコル制御を開始する関数（開始関数）と、通信を監視し処理を進める関数（アドバンス関数）により通信を実現します。アドバンス関数のリターン値により、通信状態を判断できます。
- 開始関数はスタートコンディション生成からスレーブアドレス送信までを行います。それ以降は、ストップコンディションが生成されるまでは、アドバンス関数をコールし処理を進めます。
- スレーブアドレス送信、データ送信、データ受信、ストップコンディション生成のいずれかの処理が完了すると割り込みが発生します。
- 通信速度レートは、ユーザ設定が可能です。[対応レート：～400kHz(max)]。ただし、同一チャンネルに複数のデバイスが接続されている場合には、通信速度が最小のスレーブデバイスにあわせて設定してください。
- 通信中にノイズ等の影響により、通信が停止した場合（割り込みが発生しない等）、アドバンス関数によりエラーを返すことができます。アドバンス関数のコール回数が上限を超えた場合、異常による通信停止と判断し、“未応答エラー”を返します。上限値はユーザで設定できます。
- NACK エラーが発生した場合、ストップコンディションを生成します。
- SCL クロック生成処理があります。ノイズ等の影響によりマスタとスレーブ間で同期ズレが発生し、SDA=Low ホールドになった場合、SCL 疑似クロック生成関数をコールし、スレーブデバイス内部の状態を正常終了させることができます。
- 7bit アドレスデバイス間の通信のみをサポートします。特別なアドレス（ジェネラルコール等のアドレス）をサポートしていません。

表 1-1 マスタ送信の動作パターン

	ST 生成	スレーブアドレス送信	1st データ送信	2nd データ送信	SP 生成
パターン 1	○	○	○	○	○
パターン 2	○	○	—	○	○
パターン 3	○	○	—	—	○
パターン 4	○	—	—	—	○

備考

ST：スタートコンディション

SP：ストップコンディション

表 1-2 使用する周辺機器と用途

周辺機器	用途
IICA	シリアル・インタフェース IICA 1ch (必須)

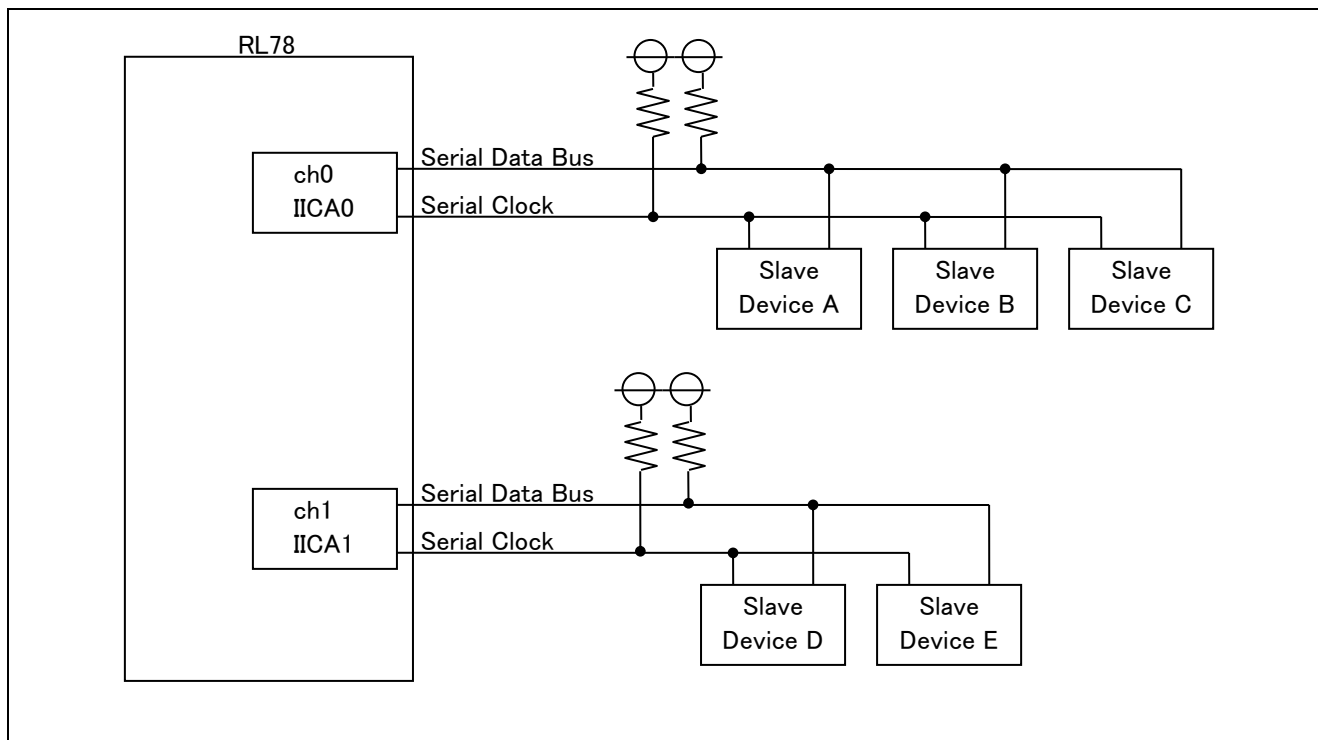


図 1-1 使用例

2. 動作確認条件

本アプリケーションノートのサンプルコードは、以下の動作条件で動作を確認しています。

(1) RL78/G14 IICA 統合開発環境 CS+ for CA,CX の場合 (コンパイラ : CA78K0R)

表 2-1 動作確認条件

項目	内容
評価に使用したマイコン	RL78/G14 (プログラム ROM 256KB / RAM 24KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 32MHz 周辺ハードウェア・クロック : 32MHz 転送クロック : 400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CS+ for CA,CX V3.01.00
Cコンパイラ	ルネサス エレクトロニクス製 RL78,78K0R コンパイラ CA78K0R V1.71 コンパイルオプション 統合開発環境のデフォルト設定 ("-qx2") を使用しています。
サンプルコードのバージョン	Ver.1.03
評価に使用したソフトウェア	RL78/G14、RL78/G1C、RL78/L12、RL78/L13、RL78/L1C グループ IICA を使った I ² C シングルマスタ制御ソフトウェア (R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/G14

(2) RL78/G14 IICA 統合開発環境 CS+ for CC の場合 (コンパイラ : CC-RL)

表 2-2 動作確認条件

項目	内容
評価に使用したマイコン	RL78/G14 (プログラム ROM 256KB / RAM 24KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 32MHz 周辺ハードウェア・クロック : 32MHz 転送クロック : 400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CS+ for CC V3.03.00
Cコンパイラ	ルネサス エレクトロニクス製 RL78 コンパイラ CC-RL V1.02.00 コンパイルオプション 統合開発環境のデフォルト設定 (既定の最適化を行う(なし)) を使用しています。
サンプルコードのバージョン	Ver.1.03
評価に使用したソフトウェア	RL78/G14、RL78/G1C、RL78/L12、RL78/L13、RL78/L1C グループ IICA を使った I ² C シングルマスタ制御ソフトウェア (R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/G14

(3) RL78/G14 IICA 統合開発環境 IAR Embedded Workbench の場合

表 2-3 動作確認条件

項目	内容
評価に使用したマイコン	RL78/G14 (プログラム ROM 256KB / RAM 24KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 32MHz 周辺ハードウェア・クロック : 32MHz 転送クロック : 400kHz
動作電圧	3.3V
統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 (Ver.1.30.2)
Cコンパイラ	IAR Systems 製 IAR Assembler for Renesas RL78 (Ver.1.30.2.50666) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.2.50666) コンパイルオプション 統合開発環境のデフォルト設定 ("レベル 低") を使用しています。
サンプルコードのバージョン	Ver.1.01
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.00
評価に使用したボード	Renesas Starter Kit for RL78/G14

(4) RL78/G1C IICA 統合開発環境 CubeSuite+の場合

表 2-4 動作確認条件

項目	内容
評価に使用したマイコン	RL78/G1C (プログラム ROM 32KB / RAM 5.5KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CubeSuite+ V2.01.00
Cコンパイラ、アセンブラ	ルネサス エレクトロニクス製 RL78,78K0R コンパイラ CA78K0R V1.70 コンパイルオプション 統合開発環境のデフォルト設定 ("-qx2") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas RL78/G1C ターゲット・ボード QB-R5F10JGC-TB

(5) RL78/G1C IICA 統合開発環境 IAR Embedded Workbench の場合

表 2-5 動作確認条件

項目	内容
評価に使用したマイコン	RL78/G1C (プログラム ROM 32KB / RAM 5.5KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
Cコンパイラ	IAR Systems 製 IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) コンパイルオプション 統合開発環境のデフォルト設定 ("レベル 低") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas RL78/G1C ターゲット・ボード QB-R5F10JGC-TB

(6) RL78/L12 IICA 統合開発環境 CubeSuite+の場合

表 2-6 動作確認条件

項目	内容
評価に使用したマイコン	RL78/L12 (プログラム ROM 32KB / RAM 1.5KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CubeSuite+ V2.01.00
Cコンパイラ、アセンブラ	ルネサス エレクトロニクス製 RL78,78K0R コンパイラ CA78K0R V1.70 コンパイルオプション 統合開発環境のデフォルト設定 ("-qx2") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/L12

(7) RL78/L12 IICA 統合開発環境 IAR Embedded Workbench の場合

表 2-7 動作確認条件

項目	内容
評価に使用したマイコン	RL78/L12 (プログラム ROM 32KB / RAM 1.5KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C コンパイラ	IAR Systems 製 IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) コンパイルオプション 統合開発環境のデフォルト設定 ("レベル 低") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/L12

(8) RL78/L13 IICA 統合開発環境 CubeSuite+の場合

表 2-8 動作確認条件

項目	内容
評価に使用したマイコン	RL78/L13 (プログラム ROM 128KB / RAM 8KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CubeSuite+ V2.01.00
C コンパイラ、アセンブラ	ルネサス エレクトロニクス製 RL78,78K0R コンパイラ CA78K0R V1.70 コンパイルオプション 統合開発環境のデフォルト設定 ("-qx2") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/L13

(9) RL78/L13 IICA 統合開発環境 IAR Embedded Workbench の場合

表 2-9 動作確認条件

項目	内容
評価に使用したマイコン	RL78/L13 (プログラム ROM 128KB / RAM 8KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
Cコンパイラ	IAR Systems 製 IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) コンパイルオプション 統合開発環境のデフォルト設定 ("レベル 低") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/L13

(10) RL78/L1C IICA 統合開発環境 CubeSuite+の場合

表 2-10 動作確認条件

項目	内容
評価に使用したマイコン	RL78/L1C (プログラム ROM 256KB / RAM 16KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CubeSuite+ V2.01.00
Cコンパイラ、アセンブラ	ルネサス エレクトロニクス製 RL78,78K0R コンパイラ CA78K0R V1.70 コンパイルオプション 統合開発環境のデフォルト設定 ("-qx2") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/L1C

(11) RL78/L1C IICA 統合開発環境 IAR Embedded Workbench の場合

表 2-11 動作確認条件

項目	内容
評価に使用したマイコン	RL78/L1C (プログラム ROM 256KB / RAM 16KB)
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX24xxx/HN58X24xxx シリーズ I ² C Serial EEPROM
動作周波数	メイン・システム・クロック : 24MHz 周辺ハードウェア・クロック : 24MHz シリアル・クロック : 400kHz
動作電圧	3.3V
統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
Cコンパイラ	IAR Systems 製 IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) コンパイルオプション 統合開発環境のデフォルト設定 ("レベル 低") を使用しています。
サンプルコードのバージョン	Ver.1.02
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX24xxx シリーズ I ² C Serial EEPROM 制御ソフトウェア(R01AN1075JJ) Ver.1.01
評価に使用したボード	Renesas Starter Kit for RL78/L1C

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。合わせて参照してください。

- ルネサス エレクトロニクス製 R1EX24xxx シリーズ I²C Serial EEPROM 制御ソフトウェア(R01AN1075JJ)

4. 周辺機能説明

RL78 ファミリー MCU の I²C 制御周辺機能として、シリアル・インタフェース IICA とシリアル・アレイ・ユニットの簡易 I²C があります。

本アプリケーションノートでは、シリアル・インタフェース IICA を使用します。

5. ハードウェア説明

5.1 使用端子一覧

表 5-1 に、使用端子と機能を示します。

表 5-1 使用端子と機能

端子名	入出力	内容
SCLA (図 5-1 の SCL)	出力	シリアル・クロック出力
SDAA (図 5-1 の SDA)	入出力	シリアル・データ入出力

5.2 参考回路

図 5-1 に接続図を示します。シリアル・クロック・ラインおよびシリアル・データ・バス・ラインは、出力が N-ch オープン・ドレインのため、外部にプルアップ抵抗が必要です。システムに応じて適正な抵抗を検討してください。また、各信号ラインの回路的マッチングを取るためのダンピング抵抗やコンデンサの付加を検討してください。

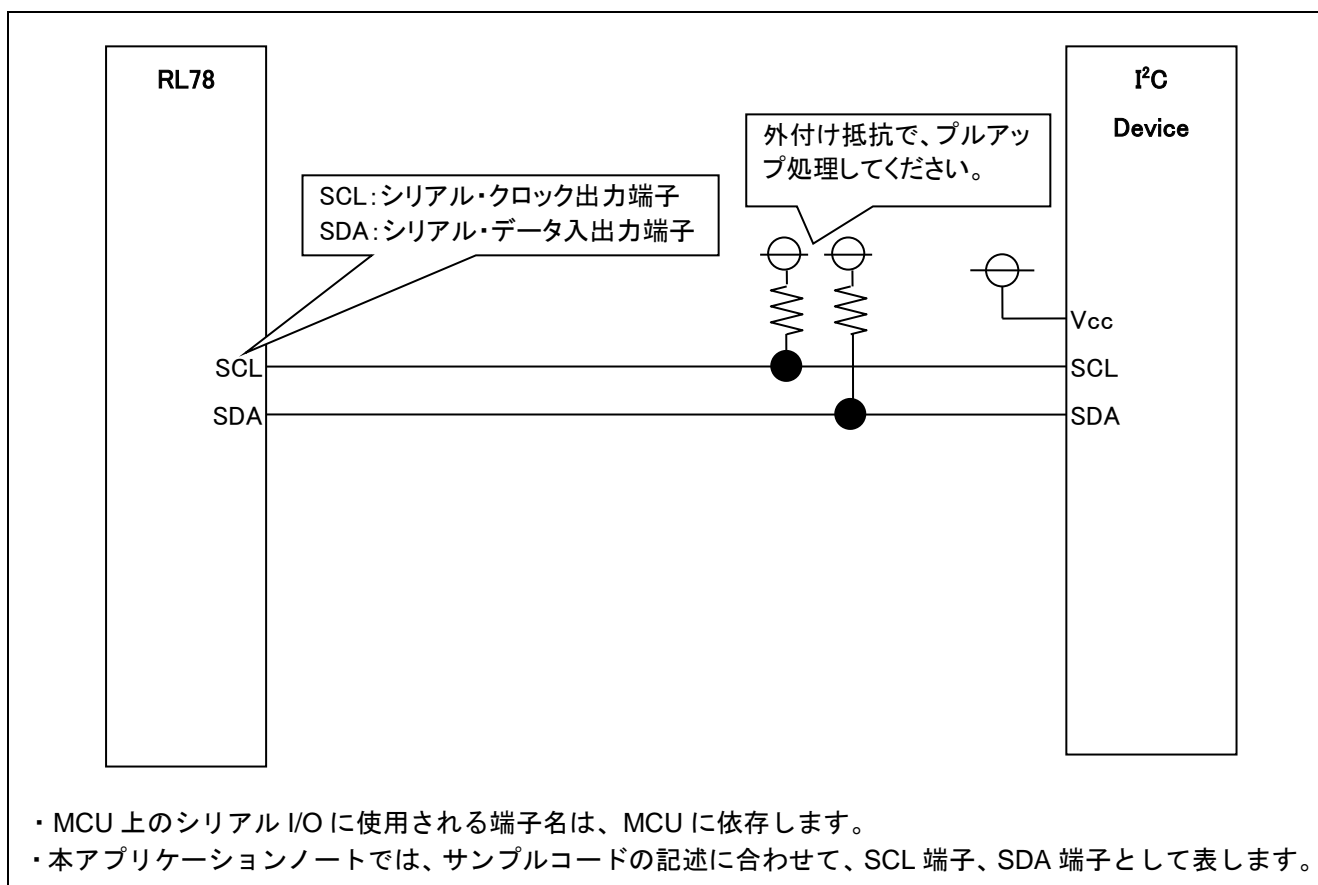


図 5-1 RL78 ファミリ MCU シリアル・インタフェース IICA と I²C スレーブデバイスの接続例

5.3 複数スレーブデバイス制御

本サンプルコードでは複数のチャンネルをサポートします。また、1つのチャンネル・バス上に、複数かつ型名が異なるスレーブデバイスを搭載し、制御できます。ただし、スタートコンディション生成から、ストップコンディション生成完了までの期間は、そのデバイス以外の通信はできません。

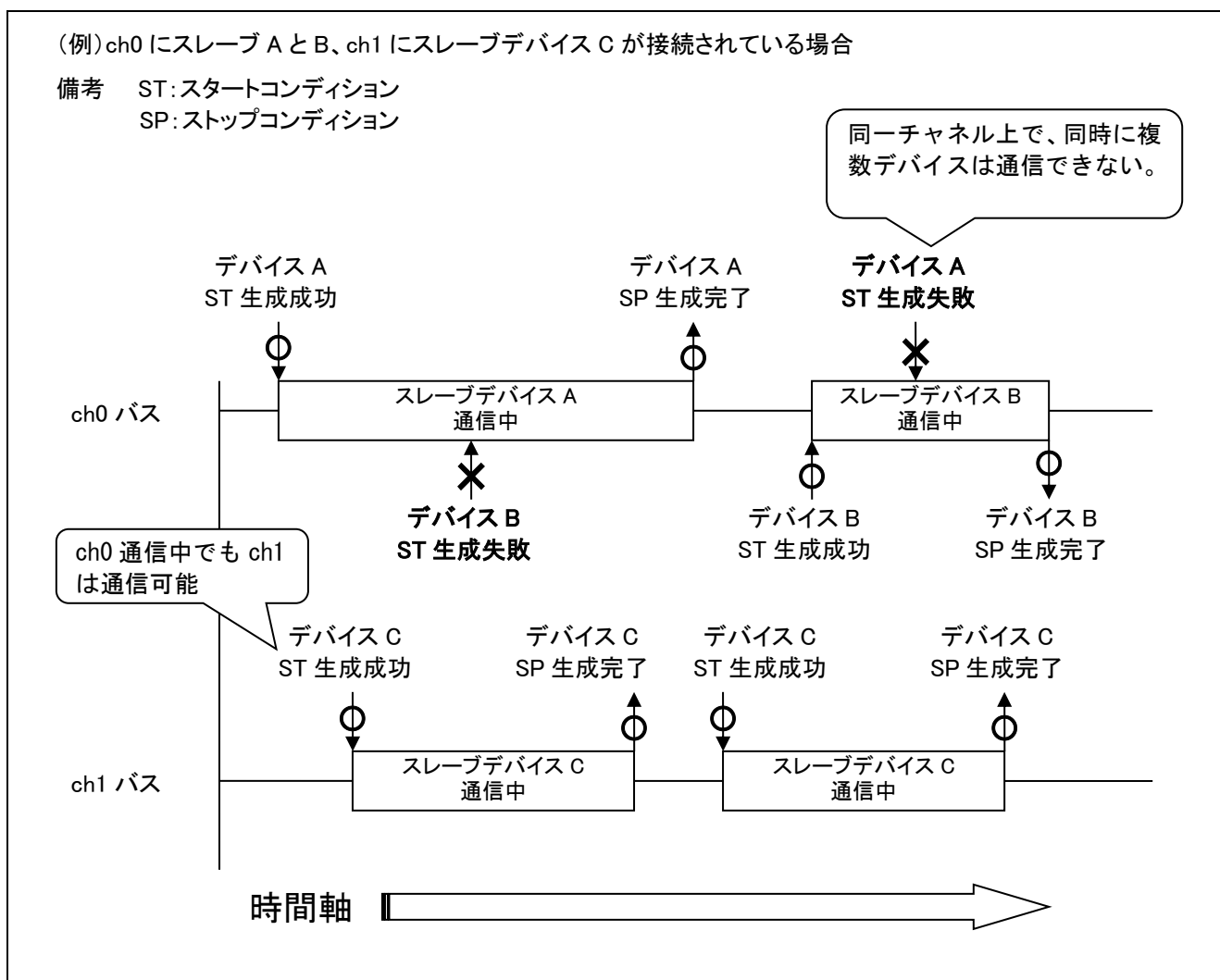


図 5-2 複数スレーブデバイス制御例

5.4 最大転送速度

最大 400kHz までの設定可能です。

ただし、標準モード・デバイスとファーストモード・デバイスが混載されるチャンネルでは、標準モードの最大 100kHz に設定する必要があります。

以下に、混載バス・システムでの最大転送速度を示します。

表 5-2 混載バス・システムでの最大転送速度

通信デバイス	混載デバイス	
	ファーストモード	標準モード
ファーストモード	0 - 400kHz	0 - 100kHz
標準モード	0 - 100kHz	0 - 100kHz

6. ソフトウェア説明

6.1 ソフトウェア構成

スレーブデバイスを制御するためのソフトウェアを上位層、I²C シングルマスタ基本プロトコル制御を実現するソフトウェアを下位層とします。上位層は下位層で用意されているプロトコルを組み合わせることでスレーブデバイスを制御します。

本サンプルコードは、I²C シングルマスタ制御するための下位層に位置するソフトウェアです。

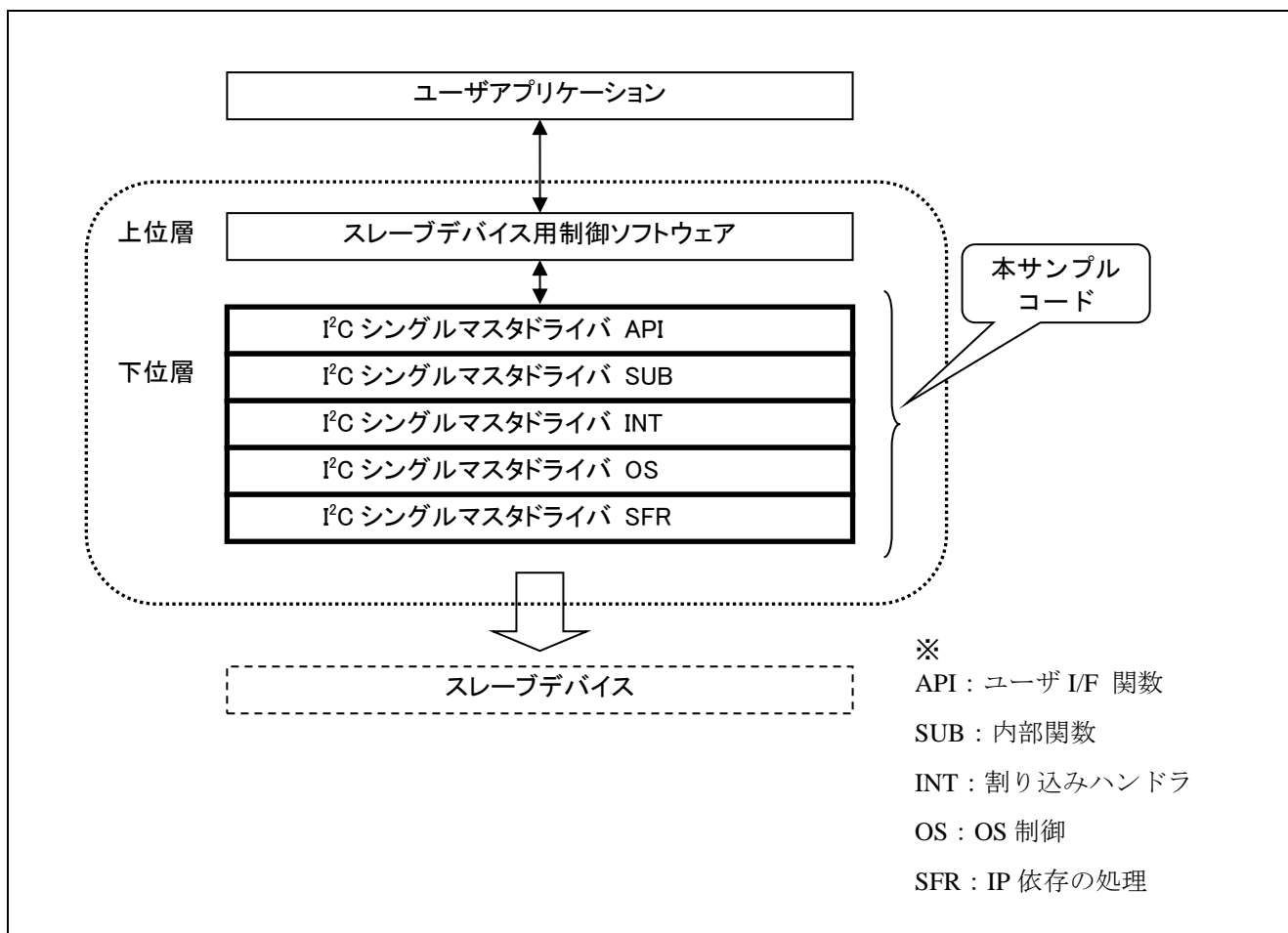


図 6-1 ソフトウェア構成

6.2 動作概要

RL78 シリアル・インタフェース IICA を使用して、I²C シングルマスタ制御を実現します。

本サンプルコードは、シングルマスタとして以下のプロトコルを実現します。

表 6-1 制御プロトコル一覧

No	制御プロトコル	概要
1	マスタ送信	マスタ (MCU) からスレーブデバイスヘータを送信します。 4 種類の送信パターンを設定可能です。
2	マスタ受信	マスタ (MCU) はスレーブデバイスからデータを受信します。
3	マスタ複合	マスタ送信後、マスタ受信を行います。

6.2.1 マスタ送信

マスタ送信には、4 種類の送信パターンがあります。通信情報を管理する I²C 通信情報構造体の設定方法により機能を選択することができます。設定方法については、6.13.1 I²C 通信情報構造体を参照ください。

(1) パターン 1

マスタ (MUC) からスレーブデバイスヘータを送信します。

初めにスタートコンディション (ST) を生成し、次にスレーブデバイスのアドレスを送信します。このとき、8 ビット目は転送方向指定ビットになりますので、データ送信時には“0” (Write) を送信します。次に 1st データを送信します。1st データとは、データ送信を行う前に、事前に送信したいデータがある場合に使用します。例えばスレーブデバイスが EEPROM の場合、EEPROM 内部のアドレスを送信することができます。次に 2nd データを送信します。2nd データがスレーブデバイスへ書き込むデータになります。データ送信を開始し、全データの送信が完了すると、ストップコンディション (SP) を生成してバスを解放します。

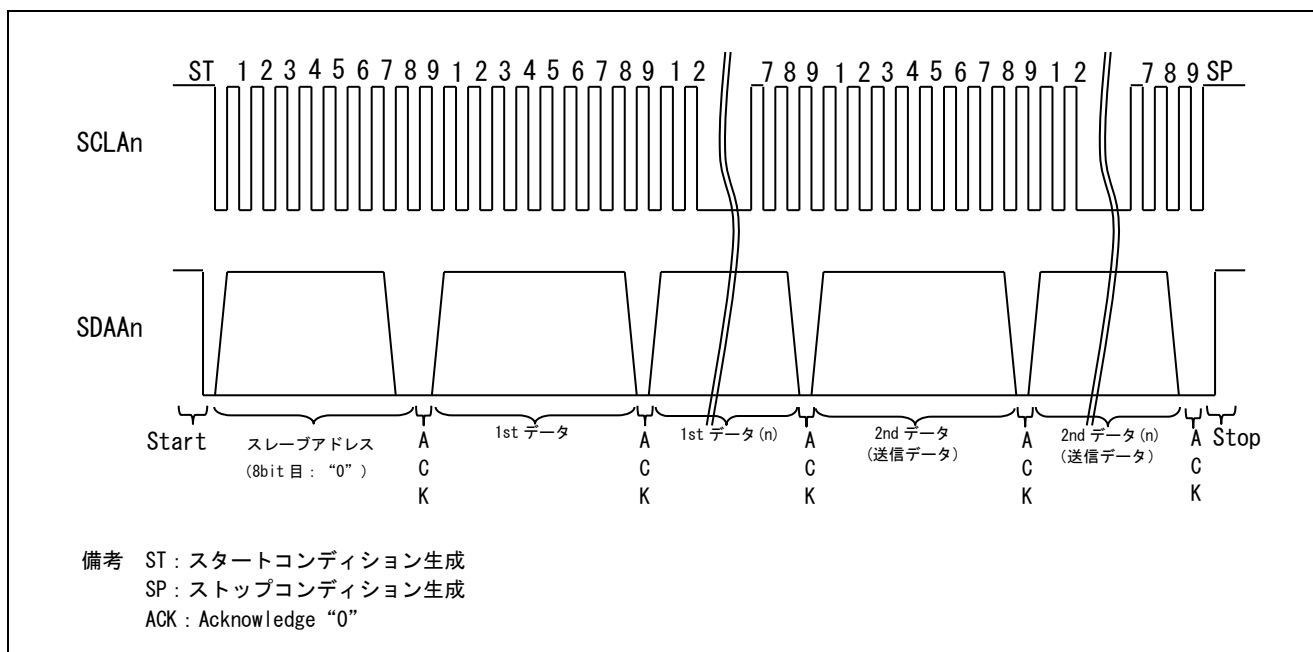


図 6-2 マスタ送信 (パターン 1) 信号図

(2) パターン 2

マスタ (MUC) からスレーブデバイスへデータを送信します。但し、1st データは送信しません。

スタートコンディション (ST) を生成からスレーブデバイスのアドレスを送信まではパターン 1 と同様に動作します。次に 1st データを送信せず、2nd データを送信します。全データの送信が完了すると、ストップコンディション (SP) を生成してバスを解放します。

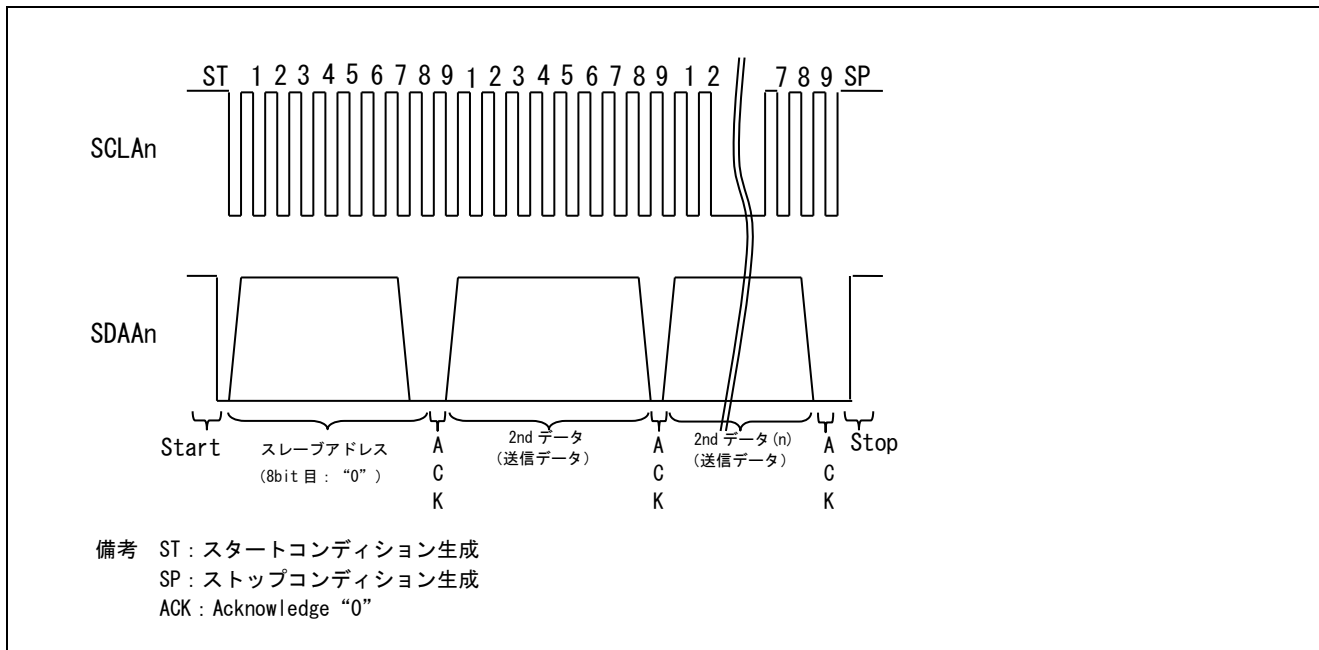


図 6-3 マスタ送信 (パターン 2) 信号図

(3) パターン 3

スタートコンディション (ST) を生成から、スレーブアドレス送信までは通常時と同様に動作します。スレーブアドレス送信後、1st データ/2nd データを設定していない場合、データ送信は行わず、ストップコンディション (SP) を生成してバスを解放します。

接続されているデバイスを検索する場合や、EEPROM 書き換え状態を確認する Acknowledge Polling を行う際に有効な処理です。

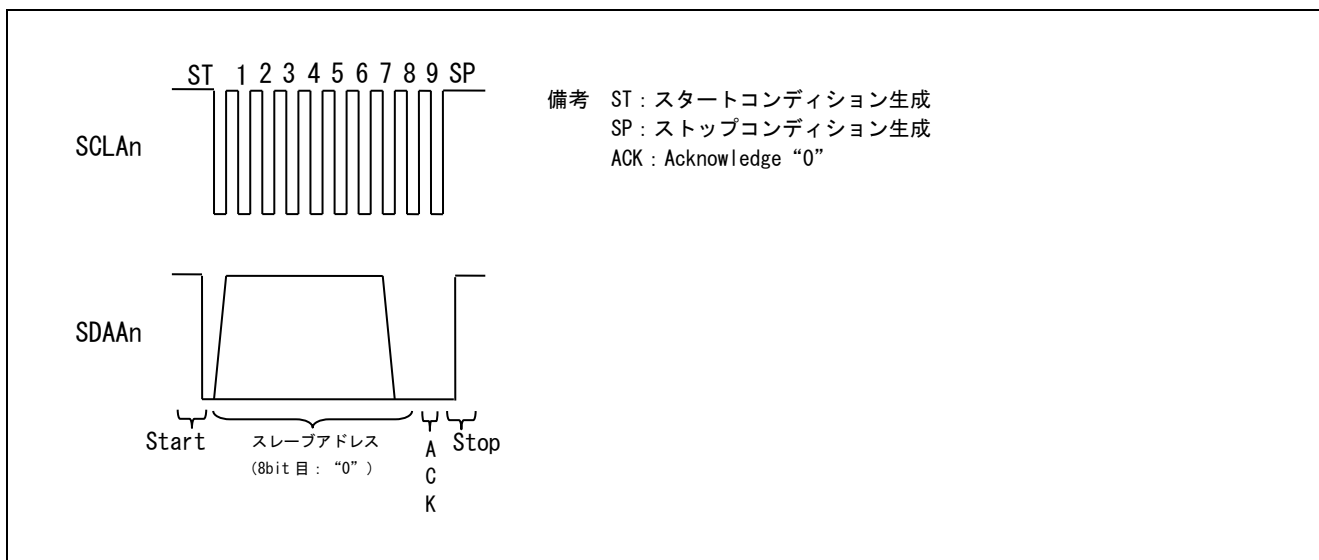


図 6-4 マスタ送信 (パターン 3 動作) 信号図

(4) パターン 4

スタートコンディション (ST) を生成後、スレーブアドレスと 1st データ/2nd データを設定していない場合、スレーブアドレス送信とデータの送信は行わず、ストップコンディション (SP) を生成してバスを解放します。

バス解放のみを行いたい場合に有効な処理です。

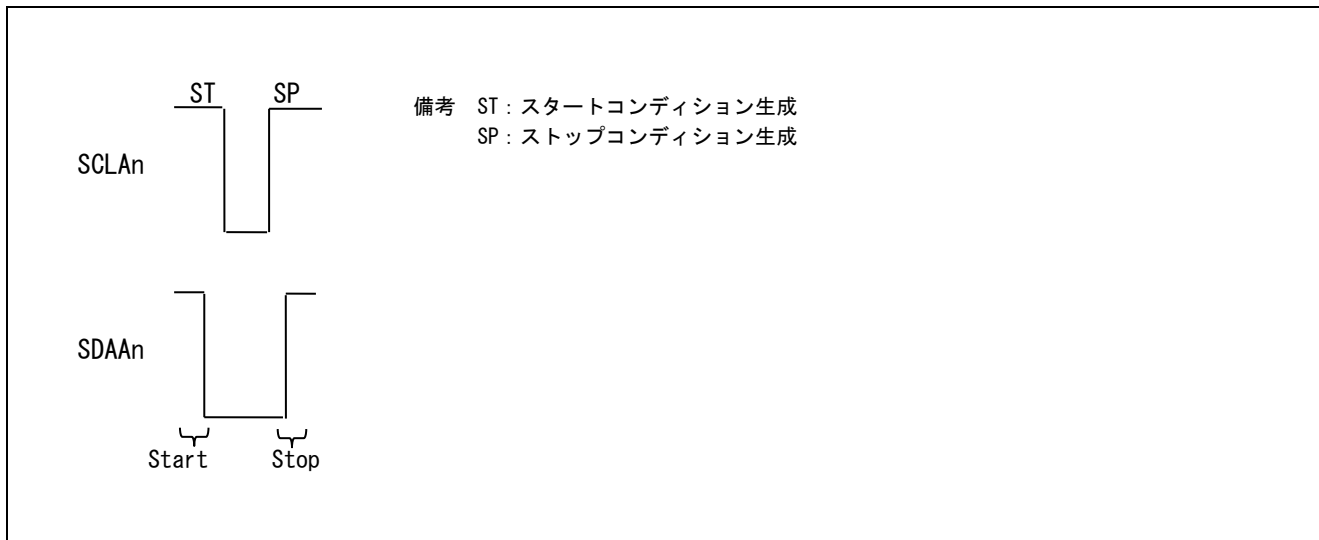


図 6-5 マスタ送信 (パターン 4 動作) 信号図

6.2.2 マスタ受信

マスタ (MUC) はスレーブデバイスからデータを受信します。

初めにスタートコンディション (ST) を生成し、次にスレーブデバイスのアドレスを送信します。このとき、8 ビット目は転送方向指定ビットになりますので、データ送信時には“1” (Read) を送信します。次にデータ受信を開始します。受信中は、1 バイト受信するごとに ACK を送信しますが、最終データ時のみ NACK を送信し、スレーブデバイスへ受信処理が完了したことを通知します。全データの受信が完了すると、ストップコンディション (SP) を生成してバスを解放します。

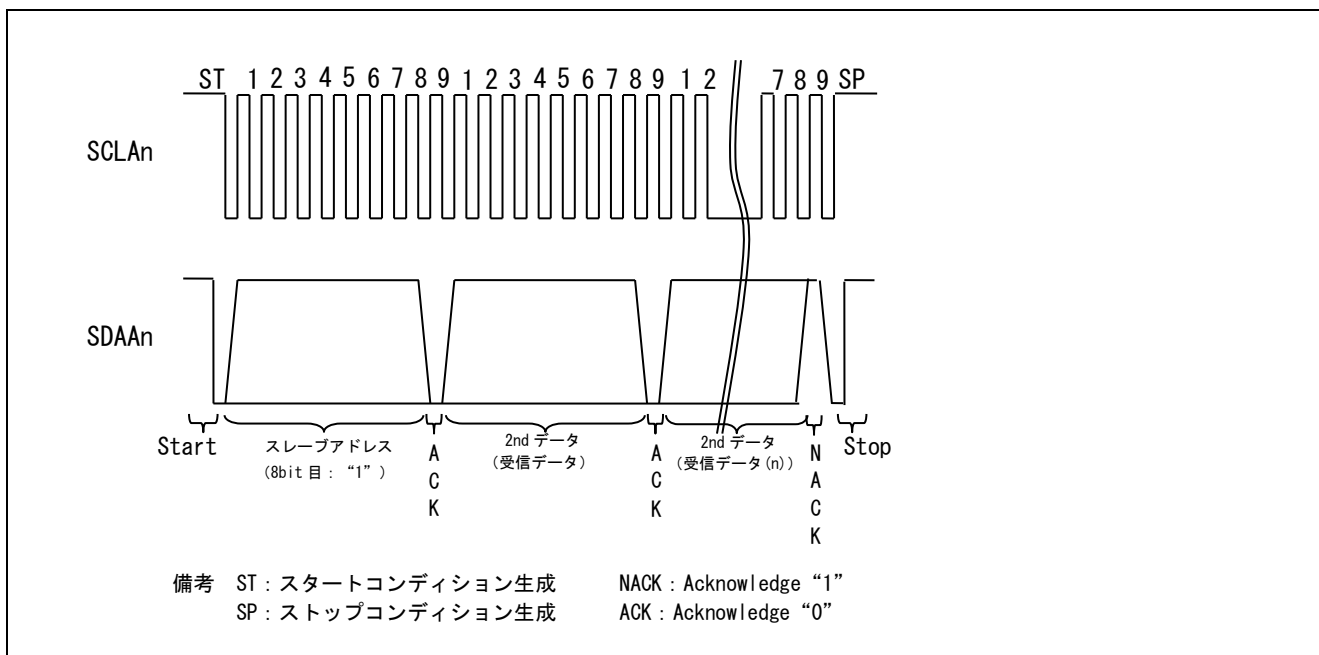


図 6-6 マスタ受信 信号図

6.2.3 マスタ複合

マスタ (MCU) からスレーブデバイスへデータを送信します (マスタ送信)。送信完了後、リスタートコンディションを生成し、転送方向を “1” (Read) に変更して、マスタはスレーブデバイスからデータを受信します (マスタ受信)。

初めにスタートコンディション (ST) を生成し、次にスレーブデバイスのアドレスを送信します。このとき、8 ビット目は転送方向指定ビットになりますので、データ送信時には “0” (Write) を送信します。次にデータを送信します。データの送信が完了すると、リスタートコンディション (RST) を生成し、スレーブアドレスを送信します。このとき、転送方向指定ビットは “1” (Read) を送信します。次にデータ受信を開始します。受信中は、1 バイト受信するごとに ACK を送信しますが、最終データ時のみ NACK を送信し、スレーブデバイスへ受信処理が完了したことを通知します。全データの受信が完了すると、ストップコンディション (SP) を生成してバスを解放します。

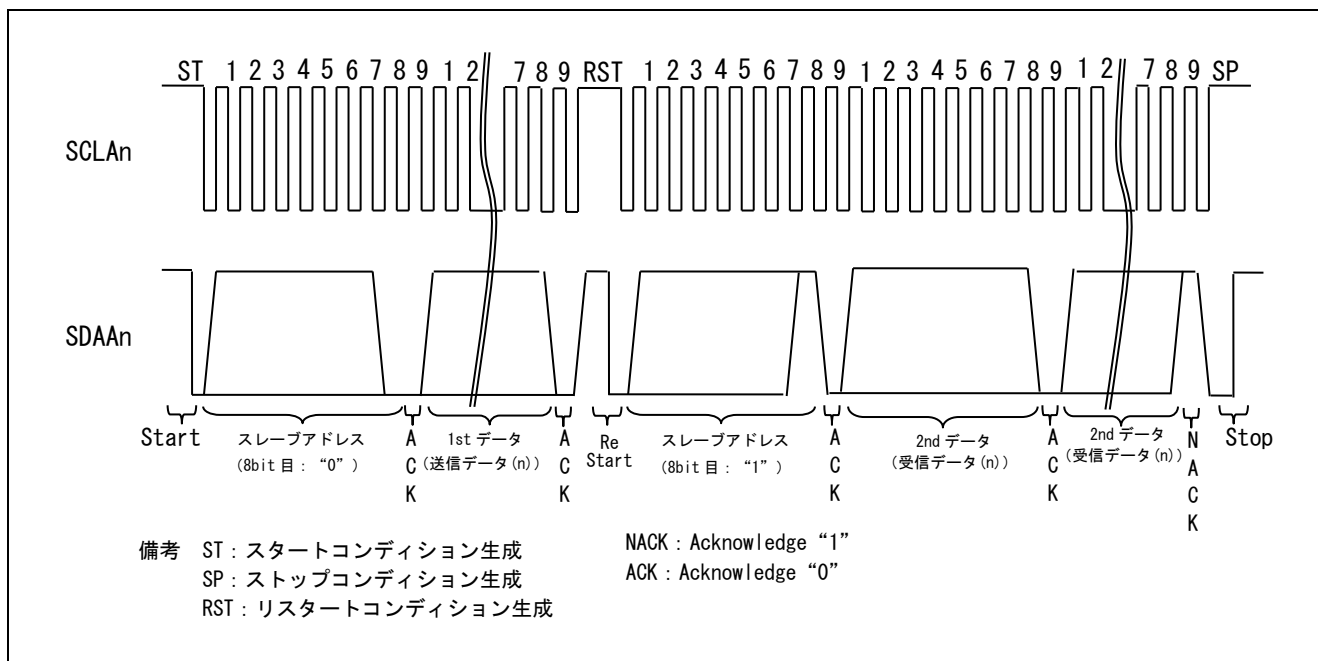


図 6-7 マスタ複合 信号図

6.3 ソフトウェア動作

本サンプルコードは、OS 制御 (※ 1) を考慮した仕様としています。それぞれの処理方法を以下に示します。

(1) 通常制御 (OS 無し)

開始関数をコールすることで通信を開始します。その後、ユーザがコールするアドバンス関数により、I²C 通信を進める処理を行います。I²C 通信を進める処理を行うかは、I²C 割り込みの発生有無により、アドバンス関数が判断します。アドバンス関数を割り込みではなく、メイン処理で行うことで I²C チャンネルの多重コールにも対応可能な仕様としています。

OS 無し制御では、割り込みが発生するとイベントフラグ (g_iic_Event[]) がセットされます。そのフラグを確認したアドバンス関数が通信を実行します。フラグの詳細については、表 6-21 を参照ください。

また、通信中の状態は、アドバンス関数のリターン値により確認することができます。

(2) 通常制御 (OS 有り)

本制御の処理は、組み込まれていません。使用される場合、必要に応じて修正してください。

OS 有りの場合、イベントフラグに代わり、OS のシステムコールがイベントになります。

開始関数コール後、アドバンス関数をコールすると、イベントが発生するまではシステムコール待ち状態になります。割り込みが発生すると OS のシステムコールが発生し、アドバンス関数でタスク (I²C 通信を進める処理) を実行します。

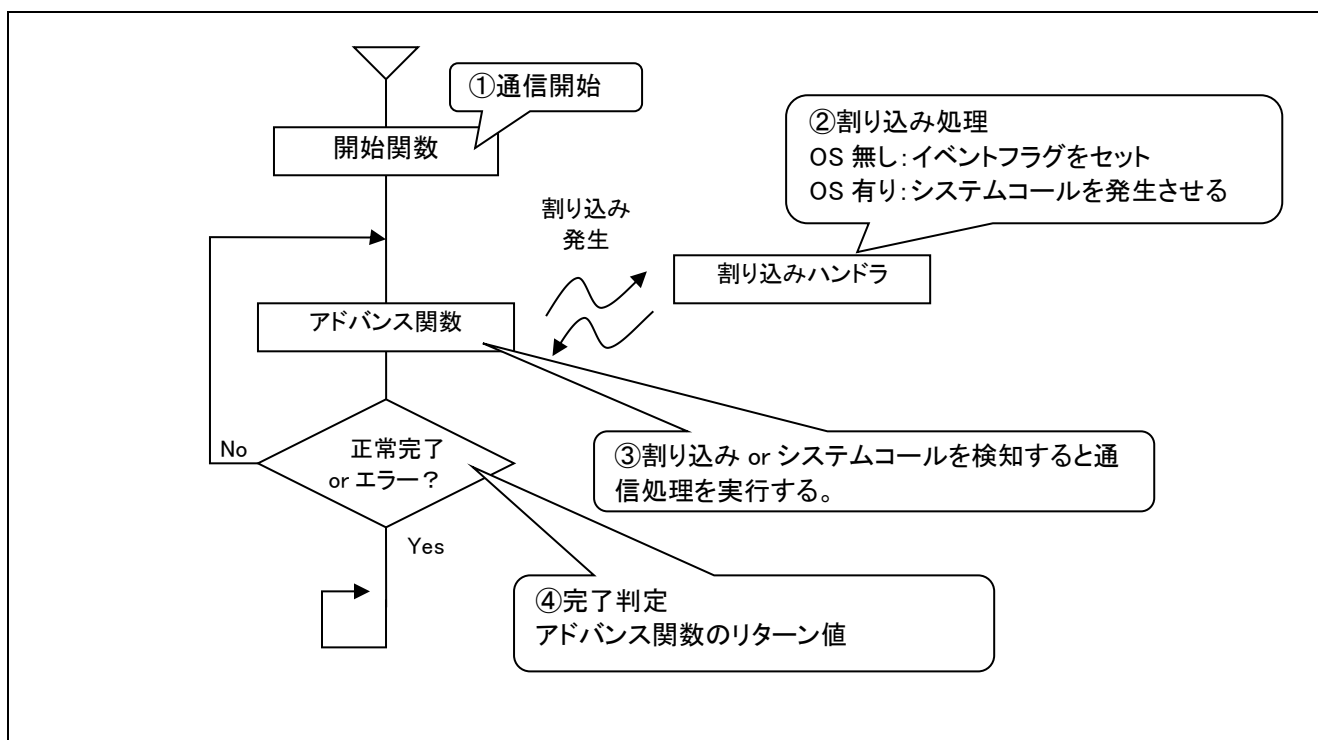


図 6-8 ソフトウェア動作概略 (OS 無し/有り制御)

※ 1 : 本サンプルコードの OS 制御は、μITRON4.0 を想定しています。

6.4 ソフトウェア動作シーケンス

(1) 通常動作 (OS 無し/OS 有り)

以下に、通常動作 (OS 無し/OS 有り) 時の動作シーケンスを示します。

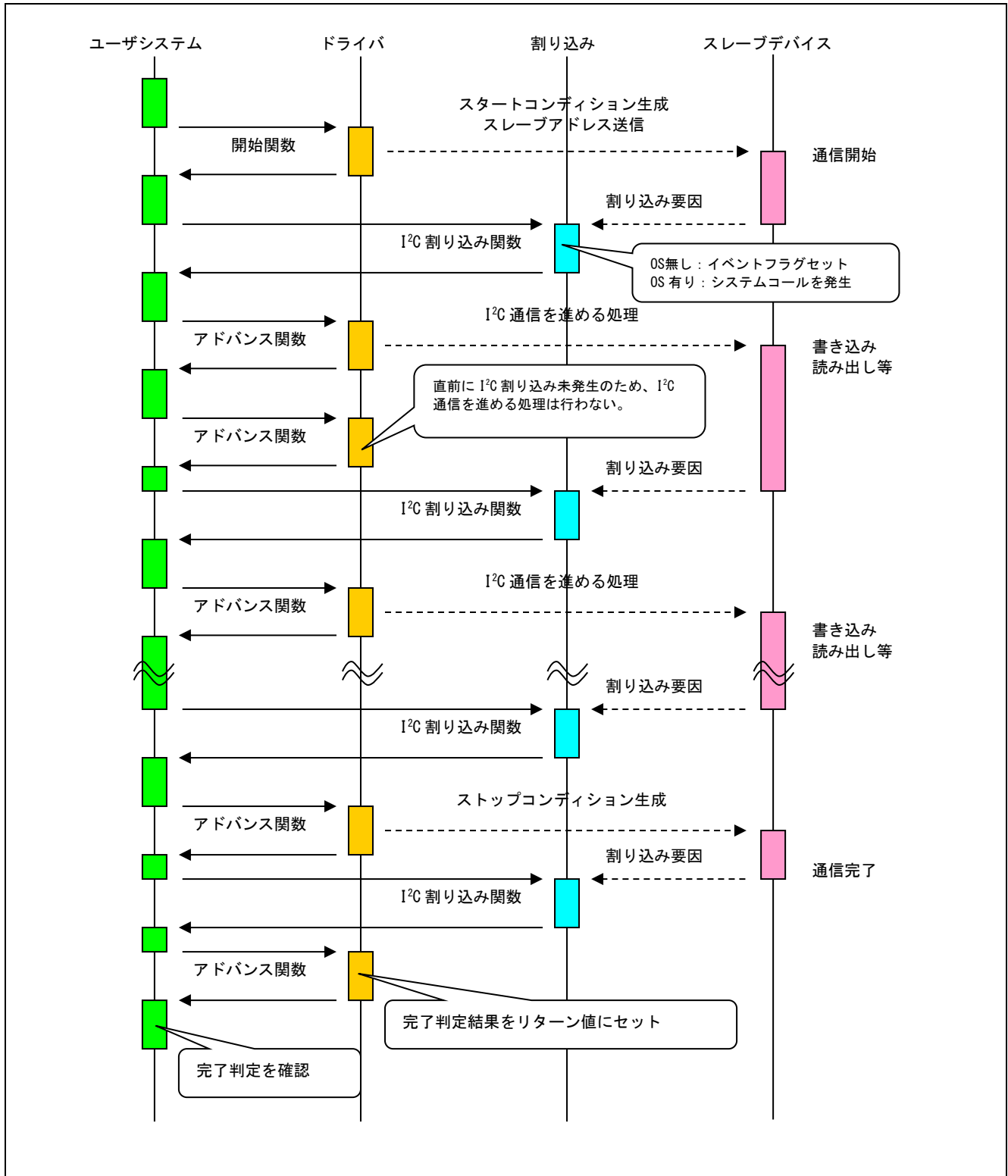


図 6-9 通常動作 (OS 有り/OS 無し) シーケンス図

6.5 スレーブデバイス制御の実現方法

(1) 各スレーブデバイス管理

使用するチャンネルや通信するためのデータ等の情報は、構造体で管理します。制御するスレーブデバイス毎に構造体を設定することで、同一チャンネル上の複数デバイスの通信を実現します。

構造体の詳細は、6.13.1 I²C 通信情報構造体を参照ください。

(2) 各チャンネル状態管理

チャンネル状態フラグ `g_iic_ChStatus[]`により、1つのバス上に接続された複数スレーブデバイスの排他制御を行います。チャンネル状態フラグの詳細については、6.15 変数一覧の `g_iic_ChStatus[]`を参照ください。

本フラグは、各チャンネルに対して1つ存在し、グローバル変数で管理します。I²C ドライバ初期化処理を完了し、対象バスで通信が行われていない場合、本フラグは“R_IIC_IDLE/R_IIC_FINISH/R_IIC_NACK”（アイドル状態（通信可能））となり、通信が可能です。通信中の本フラグの状態は、“R_IIC_COMMUNICATION”（通信中）になります。通信開始時、必ず本フラグの確認を行うため、通信中に同一チャンネル上の他デバイスの通信を開始しません。本フラグをチャンネル毎に管理することで、複数チャンネルの同時通信を実現します。

(3) 各デバイス状態管理

I²C 通信情報構造体メンバのデバイス状態フラグ `*pDevStatus` により、同一チャンネル上の複数のスレーブデバイスの制御を行うことができます。デバイス状態フラグには、そのデバイスの通信状態が格納されます。使用方法については、8.4 同一チャンネル上の複数スレーブデバイス制御時のアドバンス関数のコール方法を参照ください。

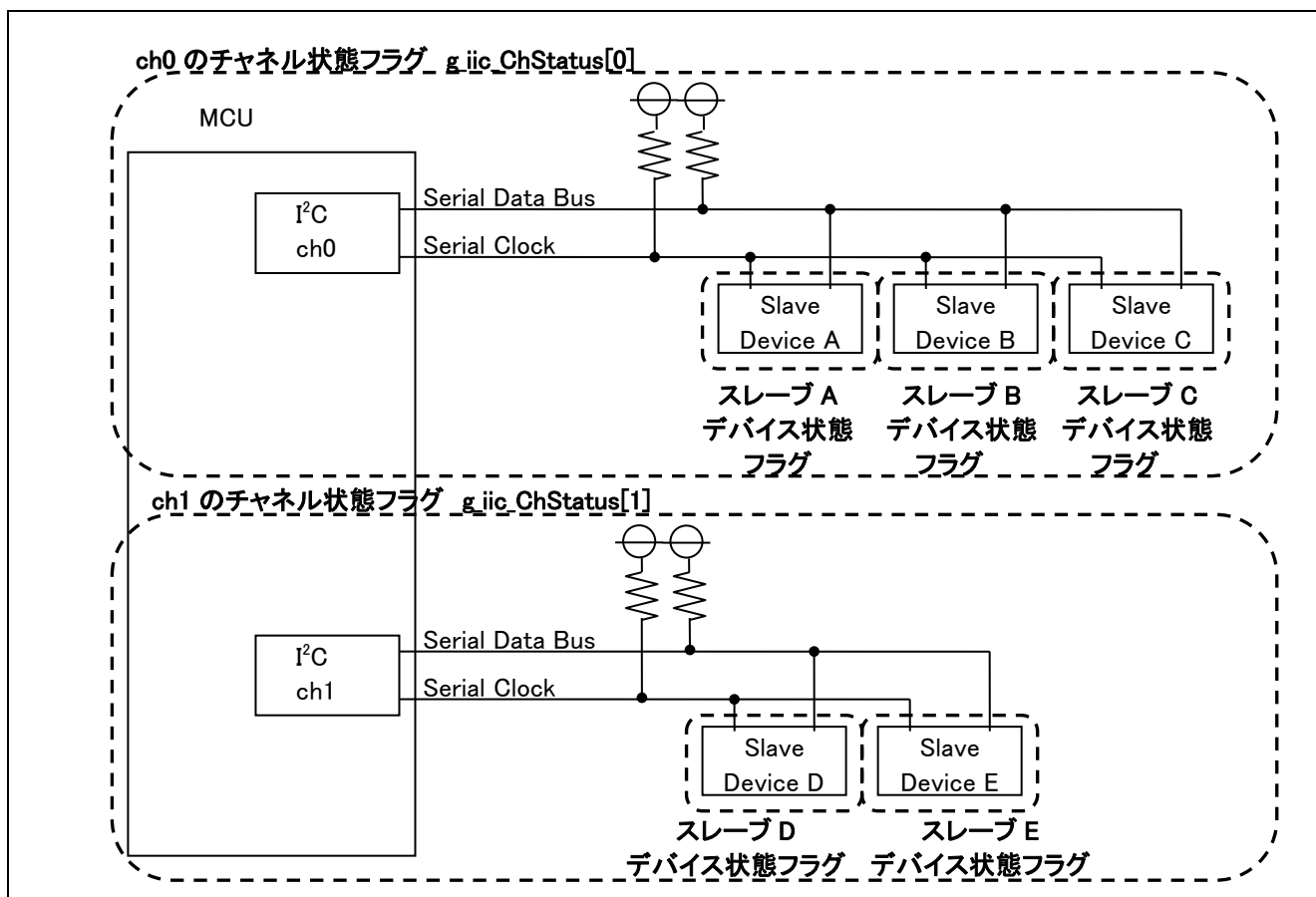


図 6-10 スレーブデバイス制御

6.6 通信の実現方法

本サンプルコードでは、スタートコンディションやスレーブアドレス送信等の処理を1つのプロトコルとして管理しており、このプロトコルを組み合わせることで通信を実現します。

6.6.1 制御時の状態

プロトコル制御を実現するための状態を以下のように定義します。

表 6-2 プロトコル制御のための状態一覧 (enum r_iic_drv_internal_status_t)

No	定数名	内容
STS0	R_IIC_STS_NO_INIT	未初期化状態
STS1	R_IIC_STS_IDLE	アイドル状態
STS2	R_IIC_STS_ST_COND_WAIT	スタートコンディション生成完了待ち状態
STS3	R_IIC_STS_SEND_SLVADR_W_WAIT	スレーブアドレス[Write]送信完了待ち状態
STS4	R_IIC_STS_SEND_SLVADR_R_WAIT	スレーブアドレス[Read]送信完了待ち状態
STS5	R_IIC_STS_SEND_DATA_WAIT	データ送信完了待ち状態
STS6	R_IIC_STS_RECEIVE_DATA_WAIT	データ受信完了待ち状態
STS7	R_IIC_STS_SP_COND_WAIT	ストップコンディション生成完了待ち状態

6.6.2 制御時のイベント

プロトコル制御時に発生するイベントを以下のように定義します。

割り込みだけでなく、本サンプルコードが提供するインタフェースがコールされた際も、イベントとして定義します。

表 6-3 プロトコル制御のためのイベント一覧 (enum r_iic_drv_internal_event_t)

No	イベント	イベントの定義
EV0	R_IIC_EV_INIT	r_iic_drv_init_driver()コール
EV1	R_IIC_EV_GEN_START_COND	r_iic_drv_generate_start_cond()コール
EV2	R_IIC_EV_SEND_SLVADR	r_iic_drv_send_slvadr()コール
EV3	R_IIC_EV_RE_SEND_SLVADR	r_iic_drv_re_send_slvadr()コール
EV4	R_IIC_EV_INT_ADD	アドレス送信完了割り込み発生
EV5	R_IIC_EV_INT_SEND	データ送信完了割り込み発生
EV6	R_IIC_EV_INT_RECEIVE	データ受信完了割り込み発生
EV7	R_IIC_EV_INT_STOP	ストップコンディション検出割り込み発生
EV8	R_IIC_EV_INT_AL	アービットレーションロスト検出割り込み発生
EV9	R_IIC_EV_INT_NACK	NACK 検出割り込み発生

6.6.3 プロトコル状態遷移

本サンプルコードでは、提供インタフェースのコール、または、I²C 割り込み発生をトリガに状態が遷移します。各プロトコルの状態遷移を以下に示します。

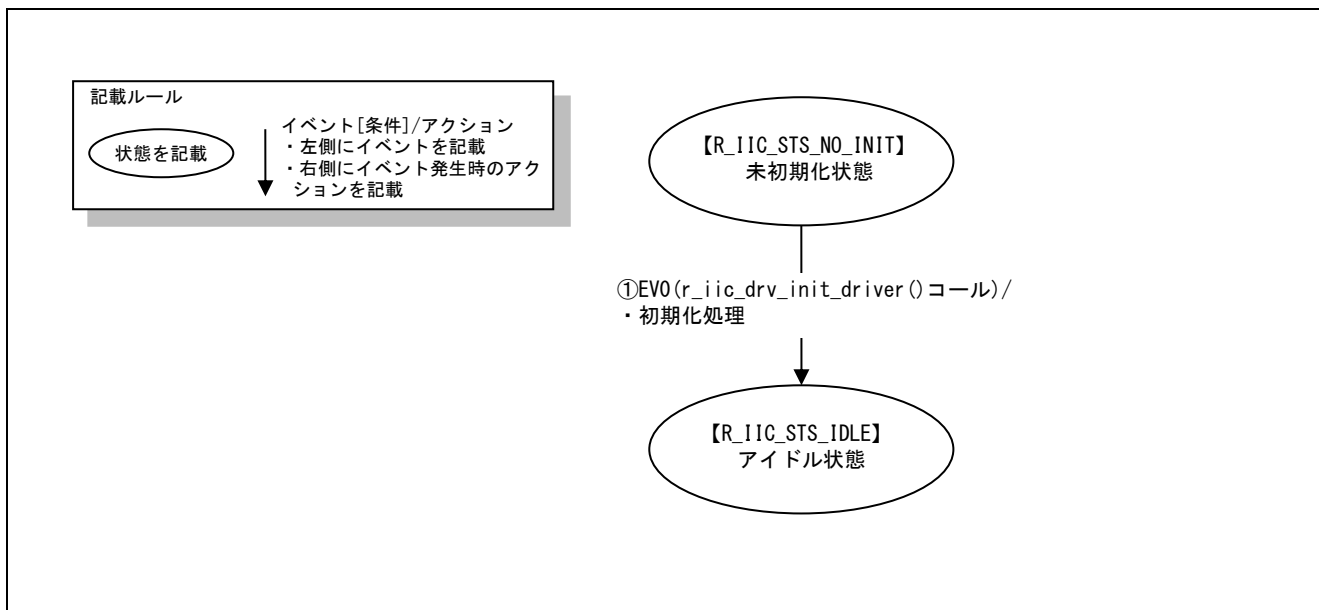


図 6-11 初期化 状態遷移図

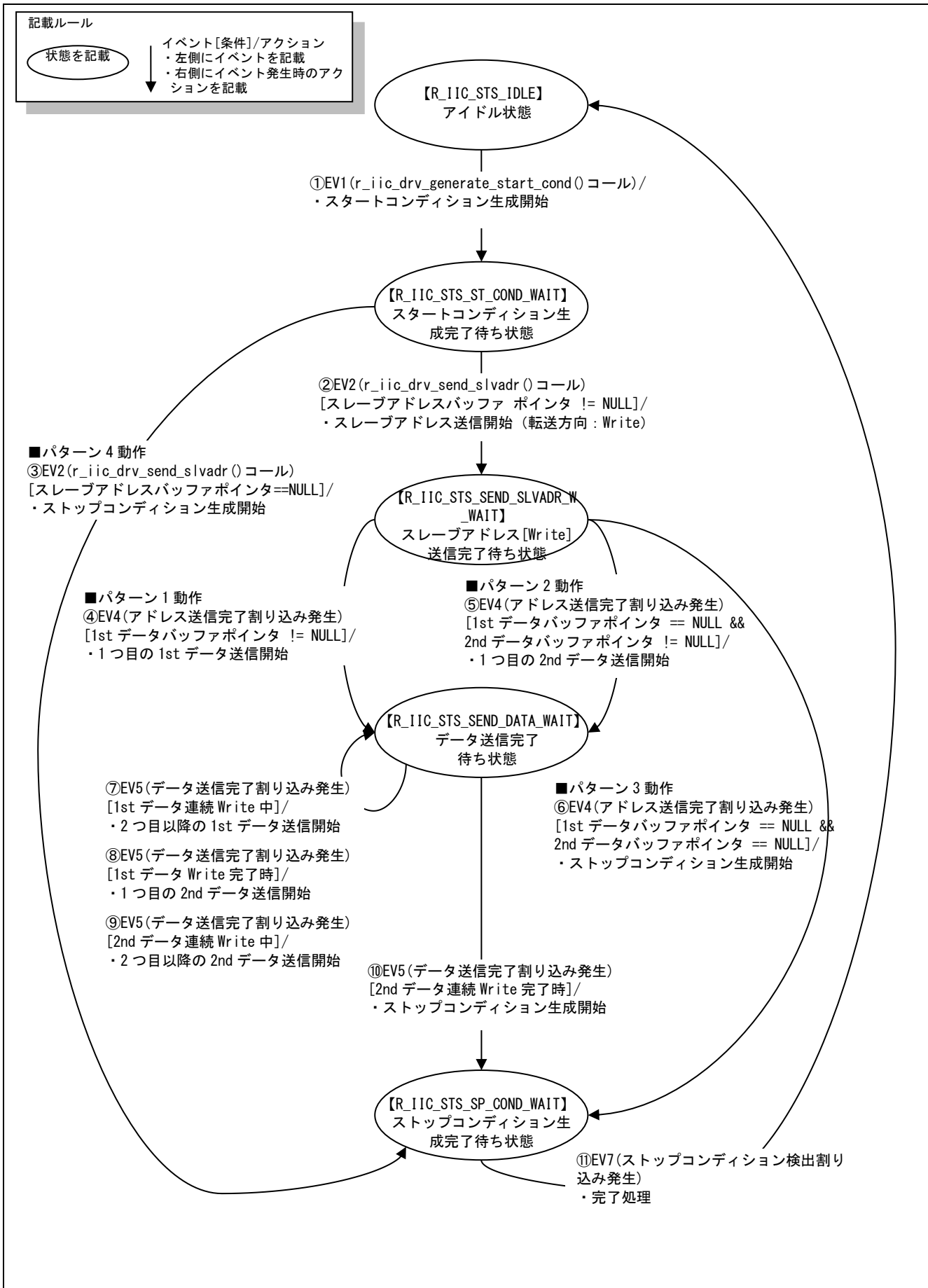


図 6-12 マスタ送信 状態遷移図

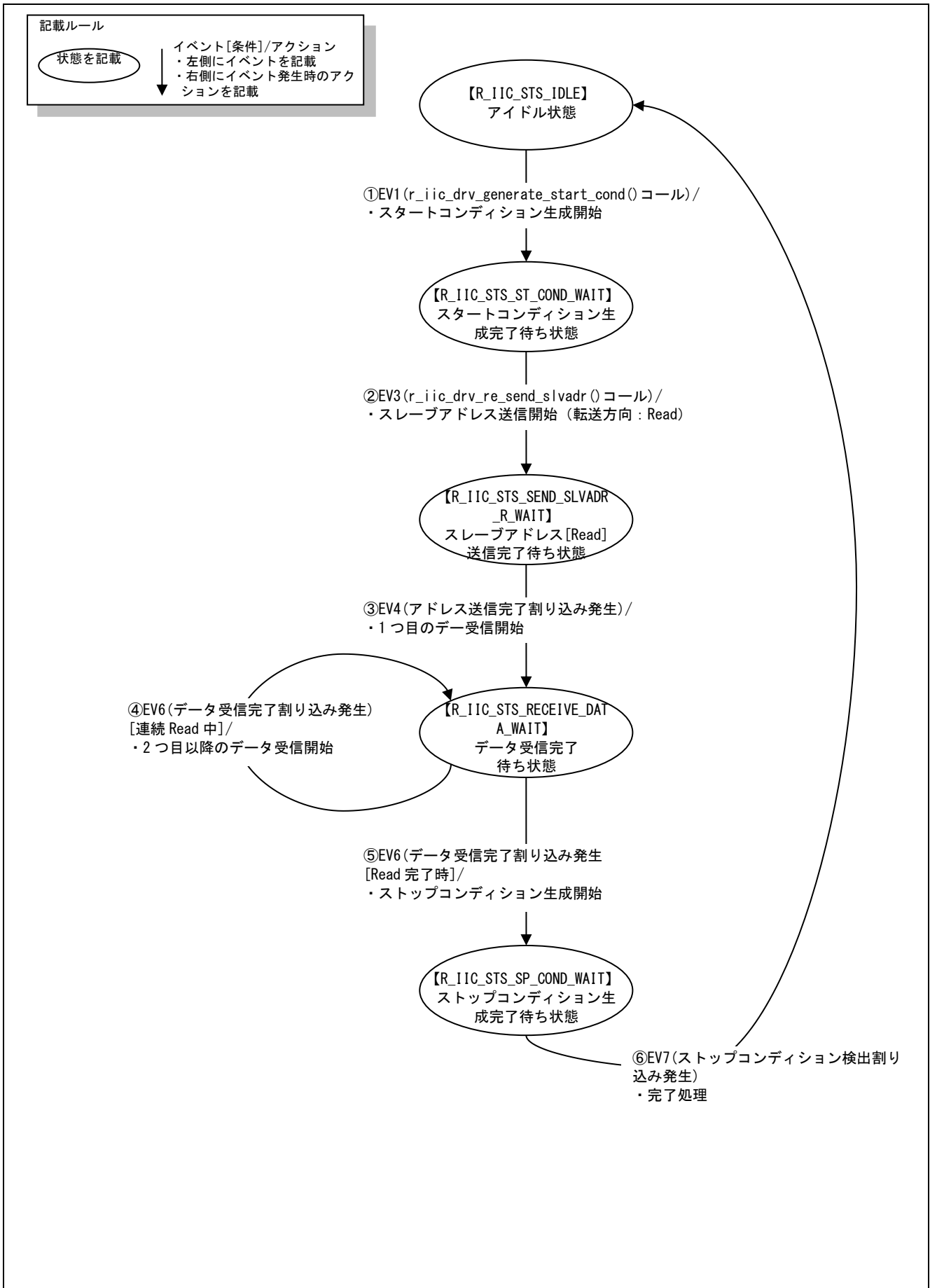


図 6-13 マスタ受信 状態遷移図

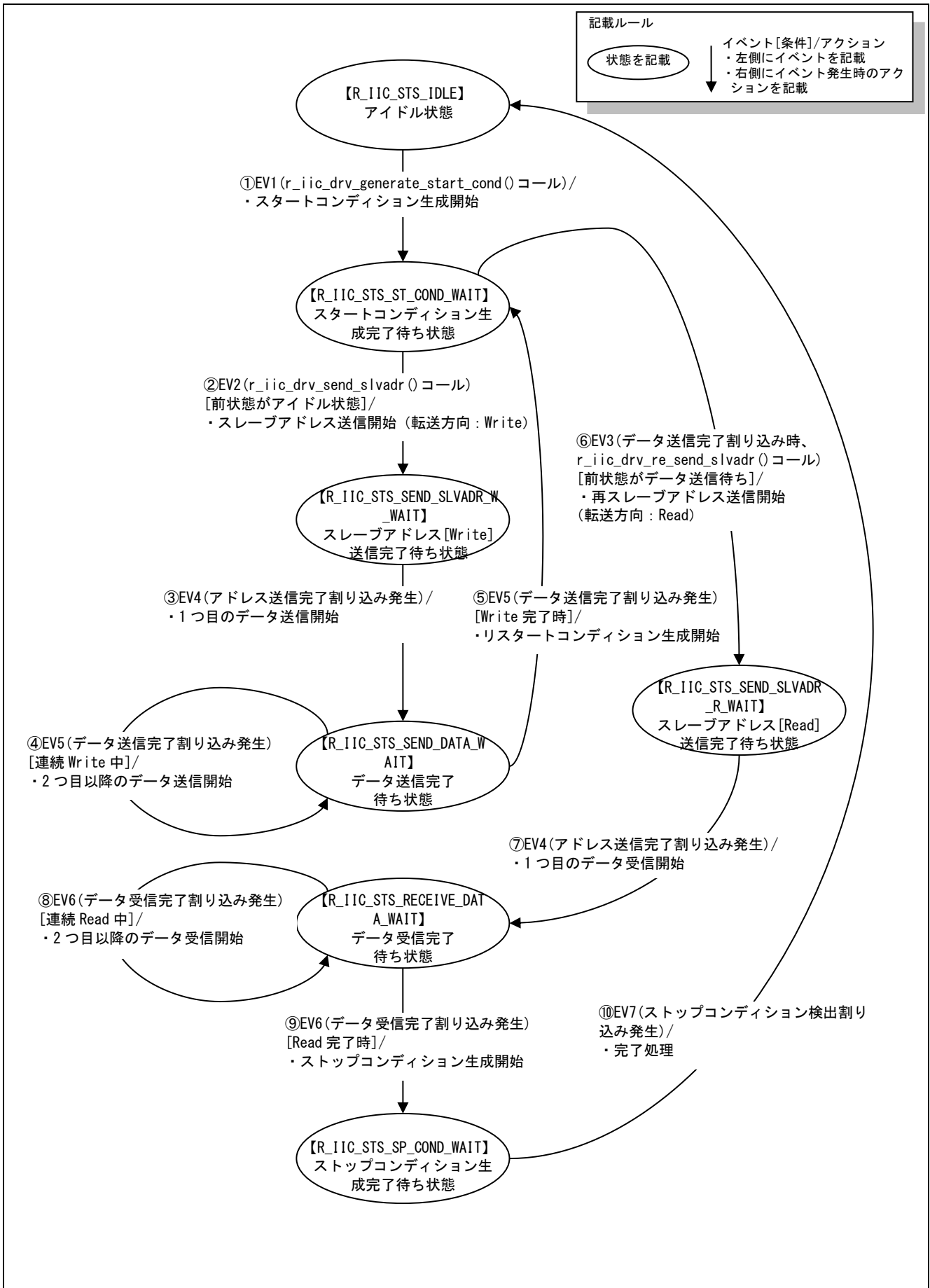


図 6-14 マスタ複合 状態遷移図

6.6.4 プロトコル状態遷移表

表 6-2 の各状態で、表 6-3 のイベントが発生した際に動作する処理を、以下の状態遷移表に定義します。

STS0 以降は表 6-2 の No を参照ください。EV0 以降は表 6-3 の No を参照ください。Func0 以降は表 6-5 を参照してください。

表 6-4 プロトコル状態遷移表 (g_iic_mtx_tbl[])

状態	イベント	EV0	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	EV9
STS0	未初期化状態 【R_IIC_STS_NO_INIT】	Func0	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
STS1	アイドル状態 【R_IIC_STS_IDLE】	ERR	Func1	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
STS2	スタートコンディション生成完了待ち状態 【R_IIC_STS_ST_COND_WAIT】	ERR	ERR	Func2	Func3	ERR	ERR	ERR	ERR	ERR	ERR
STS3	スレーブアドレス[Write]送信完了待ち状態 【R_IIC_STS_SEND_SLVADR_W_WAIT】	ERR	ERR	ERR	ERR	Func4	ERR	ERR	ERR	Func8	Func9
STS4	スレーブアドレス[Read]送信完了待ち状態 【R_IIC_STS_SEND_SLVADR_R_WAIT】	ERR	ERR	ERR	ERR	Func4	ERR	ERR	ERR	Func8	Func9
STS5	データ送信完了待ち状態 【R_IIC_STS_SEND_DATA_WAIT】	ERR	ERR	ERR	ERR	ERR	Func5	ERR	ERR	Func8	Func9
STS6	データ受信完了待ち状態 【R_IIC_STS_RECEIVE_DATA_WAIT】	ERR	ERR	ERR	ERR	ERR	ERR	Func6	ERR	Func8	Func9
STS7	ストップコンディション生成完了待ち状態 【R_IIC_STS_SP_COND_WAIT】	ERR	ERR	ERR	ERR	ERR	ERR	ERR	Func7	Func8	Func9

備考：ERR は R_IIC_ERR_OTHER を表します。ある状態で意図しないイベントが通知された場合には、すべてエラー処理を行います。

6.6.5 プロトコル状態遷移登録関数

状態遷移表に登録されている関数を以下のように定義します。

表 6-5 プロトコル状態遷移登録関数一覧

処理	関数名	概要
Func0	r_iic_drv_init_driver()	初期設定処理
Func1	r_iic_drv_generate_start_cond()	スタートコンディション生成処理
Func2	r_iic_drv_send_slvadr()	スレーブアドレス送信処理
Func3	r_iic_drv_re_send_slvadr()	再スレーブアドレス送信処理
Func4	r_iic_drv_after_send_slvadr()	スレーブアドレス送信完了後処理
Func5	r_iic_drv_write_data_sending()	データ送信処理
Func6	r_iic_drv_read_data_receiving()	データ受信処理
Func7	r_iic_drv_release()	通信完了処理
Func8	r_iic_drv_arbitration_lost()	アービトレーションロストエラー処理
Func9	r_iic_drv_nack()	NACK エラー処理

6.6.6 プロトコル状態遷移時の処理

プロトコル状態遷移時の処理 `r_iic_drv_func_table()` (以降、通信を進める処理とする) について以下に説明します。

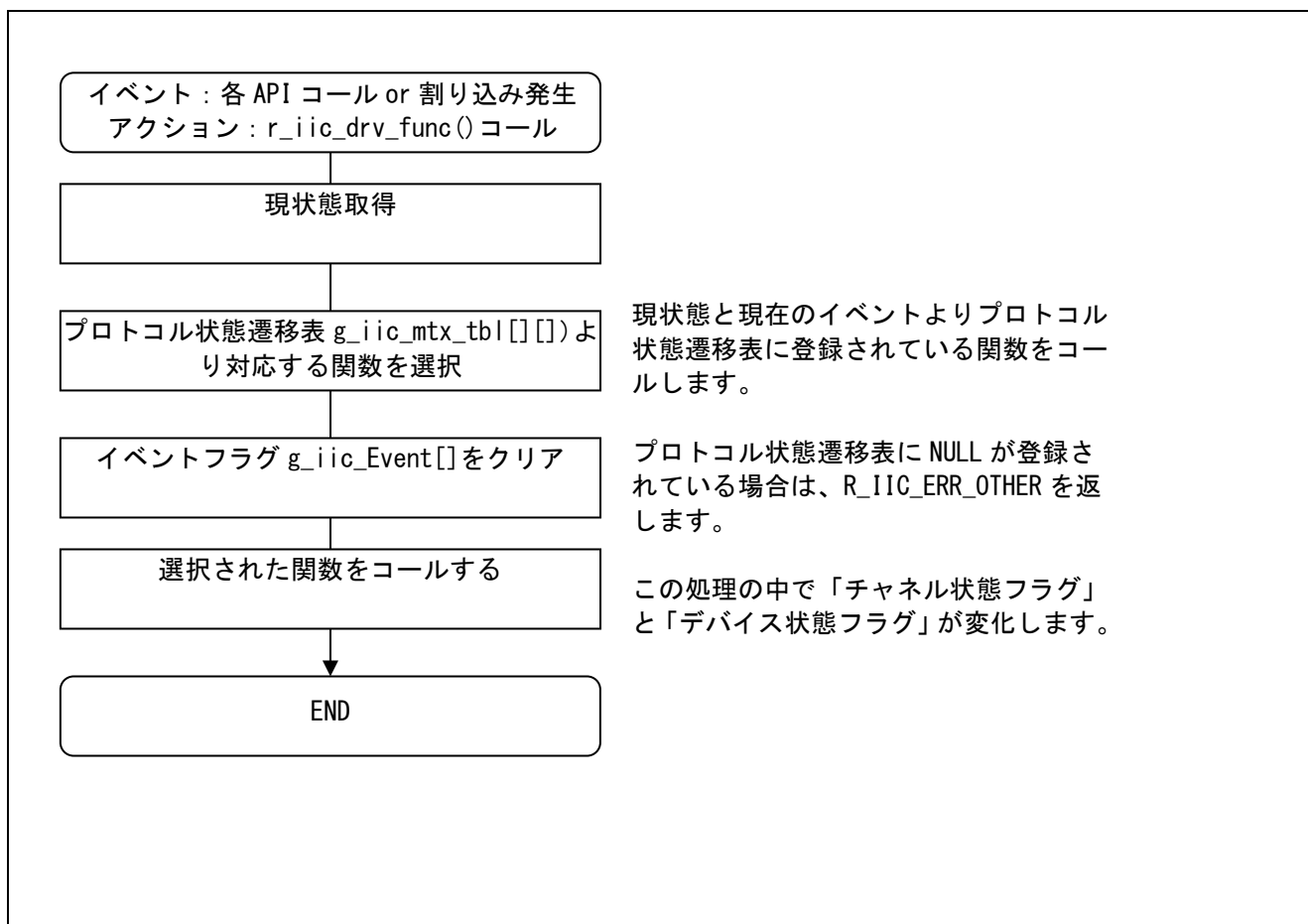
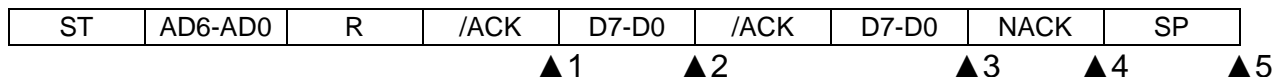


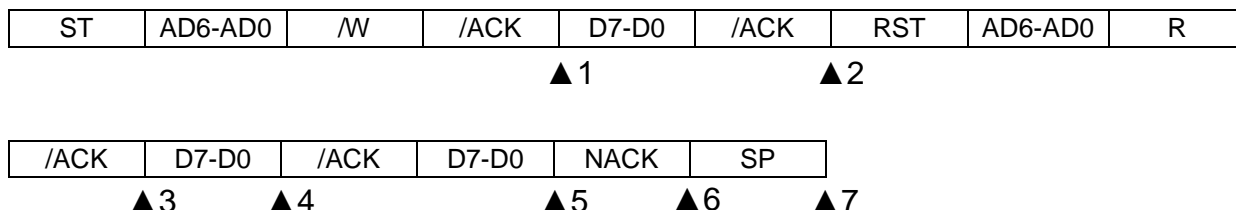
図 6-15 通信を進める処理がコールされる仕組み

6.7.2 マスタ受信



- ▲ 1 : アドレス送信完了割り込み ※ 1
- ▲ 2、▲ 3 : データ受信完了割り込み (2nd データ) ※ 3
- ▲ 4 : データ受信完了割り込み (2nd データ) ※ 4
- ▲ 5 : ストップコンディション検出割り込み

6.7.3 マスタ複合



- ▲ 1 : アドレス送信完了割り込み (転送方向ビット : Write) ※ 1
- ▲ 2 : データ送信完了割り込み (1st データ) ※ 2
- ▲ 3 : アドレス送信完了割り込み (転送方向ビット : Read) ※ 1
- ▲ 4、▲ 5 : データ受信完了割り込み (2nd データ) ※ 3
- ▲ 6 : データ受信完了割り込み (2nd データ) ※ 4
- ▲ 7 : ストップコンディション検出割り込み

- ※ 1 : アドレス送信時の 9 クロック目の立下りで発生
- ※ 2 : データ送信時の 9 クロック目の立下りで発生
- ※ 3 : データ受信時の 8 クロック目の立下りで発生
- ※ 4 : データ受信時の 9 クロック目の立下りで発生

6.8 コールバック関数

通信が正常に終了した場合、もしくはエラー終了した場合にコールされます。使用する場合は、I²C 通信情報構造体メンバの `CallBackFunc` に関数名を指定してください。構造体については、6.13.1 I²C 通信情報構造体を参照ください。

6.9 データバッファと送信／受信データの関係

本サンプルコードは、ブロック型デバイスドライバであり、送信／受信データポインタを引数として設定します。RAM 上のデータバッファのデータ並びと送信／受信順番の関係は、以下のとおりで、エンディアンや使用するシリアル通信機能に関係なく、送信データバッファの並びの順に送信し、また、受信の順に受信データバッファに書き込みます。

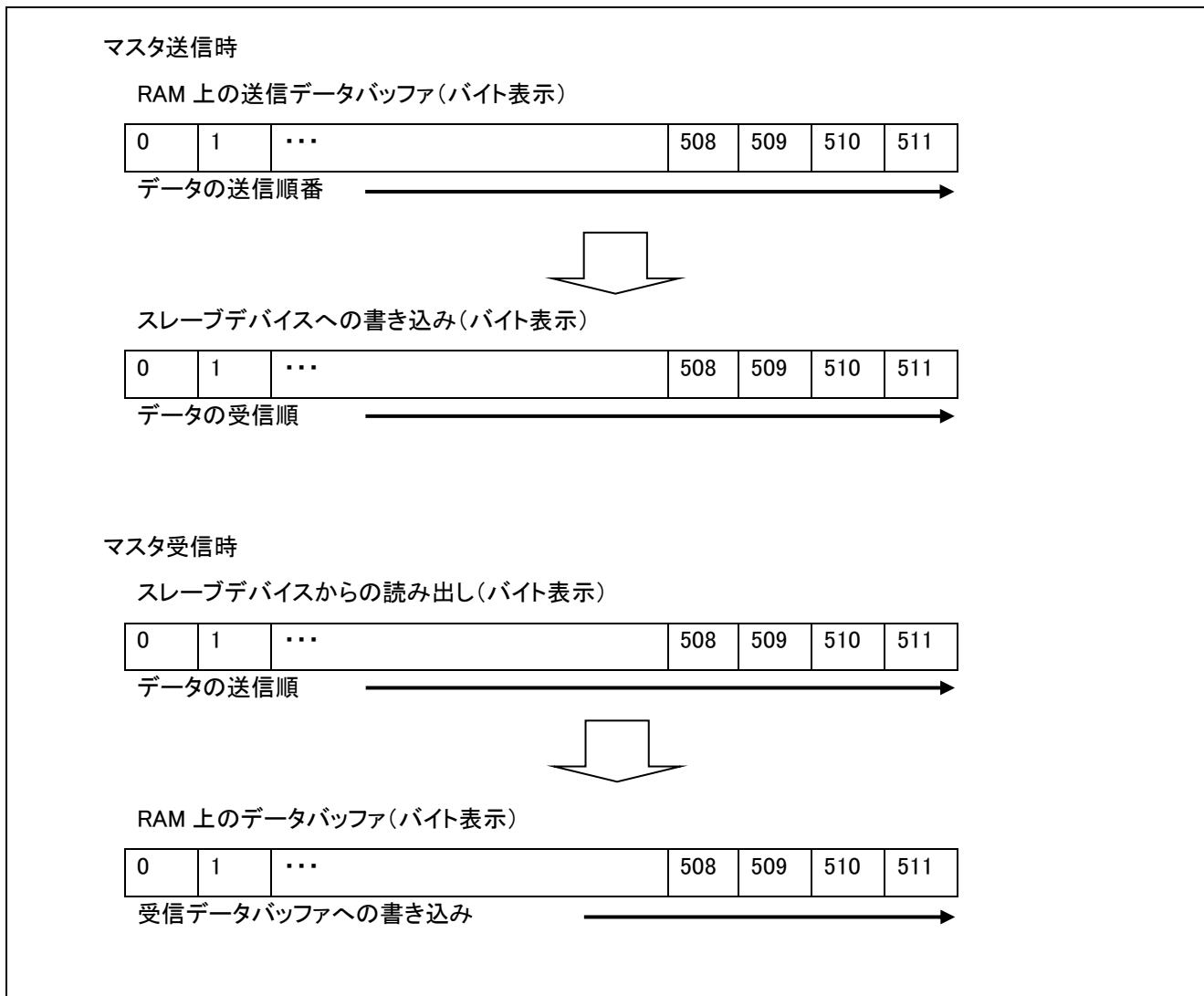


図 6-16 転送データの格納

6.10 必要メモリサイズ

(1) RL78/G14 IICA 統合開発環境 CS+ for CA,CX の場合（コンパイラ：CA78K0R）

表 6-6 に必要とするメモリサイズを示します。

表 6-6 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	5,856 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
RAM	32 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
最大使用ユーザスタック	42 バイト	
最大使用割り込みスタック	6 バイト	

必要メモリサイズは、C コンパイラのバージョンやコンパイルオプションにより異なります。

(2) RL78/G14 IICA 統合開発環境 CS+ for CC の場合（コンパイラ：CC-RL）

表 6-7 に必要とするメモリサイズを示します。

表 6-7 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	4,035 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
RAM	30 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
最大使用ユーザスタック	30 バイト	
最大使用割り込みスタック	6 バイト	

必要メモリサイズは、C コンパイラのバージョンやコンパイルオプションにより異なります。

(3) RL78/G14 IICA 統合開発環境 IAR Embedded Workbench の場合

表 6-8 に必要とするメモリサイズを示します。

表 6-8 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	9,230 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
RAM	48 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
最大使用ユーザスタック	272 バイト	
最大使用割り込みスタック	6 バイト	

必要メモリサイズは、C コンパイラのバージョンやコンパイルオプションにより異なります。

(4) RL78/L13 IICA 統合開発環境 CubeSuite+ の場合

表 6-9 に必要とするメモリサイズを示します。

表 6-9 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	5,835 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
RAM	50 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
最大使用ユーザスタック	30 バイト	
最大使用割り込みスタック	6 バイト	

必要メモリサイズは、C コンパイラのバージョンやコンパイルオプションにより異なります。

(5) RL78/L13 IICA 統合開発環境 IAR Embedded Workbench の場合

表 6-10 に必要とするメモリサイズを示します。

表 6-10 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	9,633 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
RAM	48 バイト	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c
最大使用ユーザスタック	146 バイト	
最大使用割り込みスタック	6 バイト	

必要メモリサイズは、C コンパイラのバージョンやコンパイルオプションにより異なります。
最大使用ユーザスタックサイズは、プロジェクト全体のスタックサイズです。

6.11 ファイル構成

表 6-11 に、サンプルコードで使用するファイルを示します。なお、統合開発環境で自動生成するファイルを除きます。

表 6-11 ファイル構成

¥an_r01an1074jj0104_rl78_iic	<DIR>	サンプルコードのフォルダ
r01an1074jj0104_rl78.pdf		アプリケーションノート
¥ source	<DIR>	プログラム格納用フォルダ
¥ r_iic_drv_rl78	<DIR>	I ² C シングルマスタ制御ソフトウェアのフォルダ
r_iic_drv_api.c		API ソースファイル
r_iic_drv_api.h		API ヘッダファイル
r_iic_drv_int.c		割り込みハンドラソースファイル
r_iic_drv_int.h		割り込みハンドラヘッダファイル
r_iic_drv_os.c		OS 処理ソースファイル
r_iic_drv_os.h		OS 処理ヘッダファイル
r_iic_drv_sfr.h.rl78g1c		レジスタ用共通定義ヘッダファイル (RL78/G1C 用)
r_iic_drv_sfr.h.rl78g14		レジスタ用共通定義ヘッダファイル (RL78/G14 用)
r_iic_drv_sfr.h.rl78l1c		レジスタ用共通定義ヘッダファイル (RL78/L1C 用)
r_iic_drv_sfr.h.rl78l12		レジスタ用共通定義ヘッダファイル (RL78/L12 用)
r_iic_drv_sfr.h.rl78l13		レジスタ用共通定義ヘッダファイル (RL78/L13 用)
r_iic_drv_sfr_rl78g1c.c		レジスタ用共通定義ソースファイル (RL78/G1C 用)
r_iic_drv_sfr_rl78g14.c		レジスタ用共通定義ソースファイル (RL78/G14 用)
r_iic_drv_sfr_rl78l1c.c		レジスタ用共通定義ソースファイル (RL78/L1C 用)
r_iic_drv_sfr_rl78l12.c		レジスタ用共通定義ソースファイル (RL78/L12 用)
r_iic_drv_sfr_rl78l13.c		レジスタ用共通定義ソースファイル (RL78/L13 用)
r_iic_drv_sub.c		内部関数ソースファイル
r_iic_drv_sub.h		内部関数ヘッダファイル

注) r_iic_drv_sfr.h.XXX は、MCU 毎に作成したものです。どれか一つを r_iic_drv_sfr.h にリネームして使用してください。対象 MCU のものが無い場合には、参照して、r_iic_drv_sfr.h を作成してください。

6.12 定数一覧

6.12.1 各種定義

以下に、サンプルコードで使用する各種定義を示します。

表 6-12 マクロ定義（リターン値、チャンネル状態フラグ、デバイス状態フラグの管理値）

定数名	設定値	内容
R_IIC_NO_INIT	(error_t)(0)	未初期化状態
R_IIC_IDLE	(error_t)(1)	アイドル状態：通信可能
R_IIC_FINISH	(error_t)(2)	アイドル状態：前通信終了、通信可能
R_IIC_NACK	(error_t)(3)	アイドル状態：前通信 NACK 終了、通信可能
R_IIC_COMMUNICATION	(error_t)(4)	通信中
R_IIC_LOCK_FUNC	(error_t)(5)	API 処理中 以下の場合に発生します。 ・API 処理中に他の API をコールした場合
R_IIC_BUS_BUSY	(error_t)(6)	バスビジー 以下の場合に発生します。 ・通信中に初期化関数または各開始関数をコールした場合 ・同一チャンネル上の他のデバイスが通信中に、各開始関数またはアドバンス関数をコールした場合
R_IIC_ERR_PARAM	(error_t)(-1)	パラメータエラー
R_IIC_ERR_AL	(error_t)(-2)	アービトレーションロストエラー
R_IIC_ERR_NON_REPLY	(error_t)(-3)	未応答エラー
R_IIC_ERR_SDA_LOW_HOLD	(error_t)(-4)	SCL 疑似クロック生成関数コール時、SDA Low ホールドエラー
R_IIC_ERR_OTHER	(error_t)(-5)	その他エラー

表 6-13 マクロ定義（変更禁止）

定数名	設定値	内容
W_CODE	(uint8_t)(0x00)	スレーブアドレスの転送方向が Write の場合の設定値
R_CODE	(uint8_t)(0x01)	スレーブアドレスの転送方向が Read の場合の設定値
R_IIC_TRUE	(uint8_t)(0x01)	Flag "ON"
R_IIC_FALSE	(uint8_t)(0x00)	Flag "OFF"
R_IIC_HI	(uint8_t)(0x01)	Port "H"
R_IIC_LOW	(uint8_t)(0x00)	Port "L"
R_IIC_OUT	(uint8_t)(0x00)	Port Output
R_IIC_IN	(uint8_t)(0x01)	Port Input

表 6-14 マクロ定義 (ユーザ変更可能)

定数名	設定値	内容
MAX_IIC_CH_NUM	(uint8_t)(2)	同時に使用する最大チャンネル番号+1 本サンプルコードでは 2 を設定しています。
R_IIC_CH0_LCLK	(uint8_t)(20)	IICA ロー・レベル幅設定レジスタ 0 (IICWL0) 設定値 (※1)
R_IIC_CH0_HCLK	(uint8_t)(18)	IICA ハイ・レベル幅設定レジスタ 0 (IICWH0) 設定値 (※1)
R_IIC_CH1_LCLK	(uint8_t)(20)	IICA ロー・レベル幅設定レジスタ 1 (IICWL1) 設定値 (※1)
R_IIC_CH1_HCLK	(uint8_t)(18)	IICA ハイ・レベル幅設定レジスタ 1 (IICWH1) 設定値 (※1)
REPLY_CNT	(uint32_t)(10000)	アドバンス関数カウント値 (※2)
START_COND_WAIT	(uint16_t)(100)	スタートコンディション生成待ちカウント値 (※2)
STOP_COND_WAIT	(uint16_t)(100)	ストップコンディション生成待ちカウント値 (※2)
BUSCHK_CNT	(uint16_t)(100)	バスビジーチェックカウント値 (※2)
SDACHK_CNT	(uint16_t)(100)	SDA レベルチェックカウント値 (※2)
SCLCHK_CNT	(uint16_t)(100)	SCL レベルチェックカウント値 (※2)
SCL_L_WAIT	(uint16_t)(100)	SCL 疑似クロック Low 幅待ち時間 (※3)
SCL_H_WAIT	(uint16_t)(100)	SCL 疑似クロック High 幅待ち時間 (※3)

※1 : 転送速度設定

使用するチャンネル毎に設定する必要があります。詳しい設定方法については、RL78 ファミリー MCU ユーザーズマニュアル ハードウェア編をご参照ください。

最大 400kHz までの設定可能です。ただし、標準モード・デバイスとファーストモード・デバイスが混載されるバスでは、標準モードの最大 100kHz に設定する必要があります。

※2 : カウンタ値の設定について

ソフトウェアループ用のカウンタです。したがって使用するシステム・クロックによりループ時間が変化します。使用するシステム・クロックに合わせて設定してください。

※3 : 疑似クロック出力の転送レートについて

本サンプルコードでは、スレーブデバイスが SDA を Low ホールドした際に、SCL 疑似クロックを生成して SDA を解放させるための処理の SCL 疑似クロック生成関数があります。

本関数ではポート機能を用いてクロックの High/Low を実現します。疑似クロック出力の High/Low 幅は I²C バス仕様書で規定されている min 値以上確保する必要があります。(下表参照)

High/Low 幅の設定待ちは、ソフトウェアループにより実現しています。したがって、使用するシステム・クロックにより、待ち時間が変化します。

High/Low 幅は、マクロ定義の "SCL_L_WAIT" と "SCL_H_WAIT" で設定します。使用するシステム・クロックに合わせて、I²C バス仕様書の規格を満たす値をユーザ側で管理してください。

表 6-15 I²C バス仕様書で規定されている High/Low 幅の min 値

	ファーストモード	標準モード
Low 幅 (tLow)	1.3 μs	4.7 μs
High 幅 (tHigh)	0.6 μs	4.0 μs

6.13 構造体／共用体一覧

6.13.1 I²C 通信情報構造体

サンプルコードで使用する I²C 通信情報構造体を以下に示します。本構造体は、使用するスレーブデバイス毎に設定する必要があります。

```
typedef struct
{
    uint8_t      *pSlvAdr;          /* Pointer for Slave address buffer */
    uint8_t      *pData1st;        /* Pointer for 1st Data buffer */
    uint8_t      *pData2nd;       /* Pointer for 2nd Data buffer */
    error_t      *pDevStatus;     /* Device status flag */
    uint32_t     Cnt1st;          /* 1st Data counter */
    uint32_t     Cnt2nd;          /* 2nd Data counter */
    r_iic_callback CallbackFunc; /* Callback function */
    uint8_t      ChNo;           /* Channel No. */
    uint8_t      rsv1;
    uint8_t      rsv2;
    uint8_t      rsv3;
} r_iic_drv_info_t;
```

図 6-17 I²C 通信情報構造体

(1) 構造体メンバ説明

構造体の説明を以下に示します。“r_iic_drv_info_t”メンバの設定方法は、表 6-17 および表 6-18 を参照してください。

表 6-16 構造体 “r_iic_drv_info_t” のメンバー一覧

構造体メンバ	内容
*pSlvAdr	スレーブアドレス格納バッファポインタ 1 バイト確保してください。
*pData1st	1st データ格納バッファポインタ
*pData2nd	2nd データ格納バッファポインタ
*pDevStatus	デバイス状態フラグポインタ 同一チャンネル上に複数デバイスを接続している時に、通信中のデバイスを確認することができます。1 バイト確保してください。 使用例については、8.4 項を参照ください。
Cnt1st	1st データカウンタ (バイト数)
Cnt2nd	2nd データカウンタ (バイト数)
CallbackFunc	コールバック関数
ChNo	使用デバイスのチャンネル番号 使用するバスのチャンネル番号を設定してください。
rsv1 rsv2 rsv3	アライメント調整用

(2) 設定方法

表 6-17 にマスタ送信時、表 6-18 にマスタ受信及びマスタ複合時の構造体 “r_iic_drv_info_t” メンバのユーザ設定可能範囲を示します。

表 6-17 マスタ送信時、構造体 “r_iic_drv_info_t” メンバのユーザ設定可能範囲

構造体メンバ	ユーザ設定可能範囲			
	マスタ送信 パターン 1	マスタ送信 パターン 2	マスタ送信 パターン 3	マスタ送信 パターン 4
*pSlvAdr	スレーブアドレス 格納元アドレス	スレーブアドレス 格納元アドレス	スレーブアドレス 格納元アドレス	NULL
*pData1st	1st データ格納元ア ドレス	NULL	NULL	NULL
*pData2nd	2nd データ（送信 データ）格納元アド レス	2nd データ（送信 データ）格納元アド レス	NULL	NULL
*pDevStatus	デバイス状態格納 元アドレス	デバイス状態格納 元アドレス	デバイス状態格納 元アドレス	デバイス状態格納 元アドレス
Cnt1st	0000 0001h（※） ～FFFF FFFFh	（設定無効）	（設定無効）	（設定無効）
Cnt2nd	0000 0001h（※） ～FFFF FFFFh	0000 0001h（※） ～FFFF FFFFh	（設定無効）	（設定無効）
CallBackFunc	使用する場合は関 数名を指定、使用し ない場合は NULL。	使用する場合は関 数名を指定、使用し ない場合は NULL。	使用する場合は関 数名を指定、使用し ない場合は NULL。	使用する場合は関 数名を指定、使用し ない場合は NULL。
ChNo	00h～FFh	00h～FFh	00h～FFh	00h～FFh
rsv1,rsv2,rsv3	（設定無効）	（設定無効）	（設定無効）	（設定無効）

表 6-18 マスタ受信及びマスタ複合時、構造体 “r_iic_drv_info_t” メンバのユーザ設定可能範囲

構造体メンバ	ユーザ設定可能範囲	
	マスタ受信	マスタ複合
*pSlvAdr	スレーブアドレス格納元アドレス	スレーブアドレス格納元アドレス
*pData1st	（設定無効）	1st データ格納元アドレス
*pData2nd	2nd（受信データ）格納先アドレス	2nd データ（受信データ）格納先アドレス
*pDevStatus	デバイス状態格納元アドレス	デバイス状態格納元アドレス
Cnt1st	（設定無効）	0000 0001h（※）～FFFF FFFFh
Cnt2nd	0000 0001h（※）～FFFF FFFFh	0000 0001h（※）～FFFF FFFFh
CallBackFunc	使用する場合は関数名を指定、使用 しない場合は NULL。	使用する場合は関数名を指定、使用し ない場合は NULL。
ChNo	00h～FFh	00h～FFh
rsv1,rsv2,rsv3	（設定無効）	（設定無効）

※：表 6-17、表 6-18 共に “0” は設定禁止です。

(3) コールバック関数

通信が正常に終了した場合、もしくはエラー終了した場合にコールされます。使用する場合は、`CallBackFunc` に関数名を指定してください。

(4) 設定に関する注意事項

マスタ送信では本構造体で設定されたデータを参照し、どの動作を行うか決定します。表 6-17 以外の設定を行うと、正常に動作できなくなりますので、設定の際には十分に注意してください。

6.13.2 内部情報管理構造体

サンプルコードで使用する内部情報管理構造体を以下に示します。本構造体はサンプルコードで制御するため、設定は不要です。

```
typedef struct
{
    r_iic_drv_internal_mode_t      Mode;          /* Mode of Control Protocol          */
    r_iic_drv_internal_status_t   N_status;      /* Internal Status of NOW            */
    r_iic_drv_internal_status_t   B_status;      /* Internal Status of BEFORE         */
} r_iic_drv_internal_info_t;
```

図 6-18 内部情報管理構造体

(1) 構造体メンバ説明

構造体の説明を以下に示します。

表 6-19 構造体 “r_iic_drv_internal_info_t” のメンバー一覧

構造体メンバ	内容
Mode	I ² C プロトコル動作モード 格納されるデータの定義については、表 6-20 を参照ください。
N_status	プロトコル制御のための現状態です。表 6-2 に定義されている値が格納されます。
B_status	プロトコル制御のための前状態です。表 6-2 に定義されている値が格納されます。

6.14 列挙型

本サンプルコードで使用する列挙型の定義を以下に示します。

表 6-20 I²C プロトコル動作モード一覧 (enum r_iic_drv_internal_mode_t)

定義名	内容
R_IIC_MODE_NONE	未通信状態 未初期化状態またはアイドル状態のとき、このモードに遷移します。
R_IIC_MODE_WRITE	マスタ送信動作中 マスタ送信開始関数 R_IIC_Drv_MasterTx()により通信を開始すると、このモードに遷移します。
R_IIC_MODE_READ	マスタ受信動作中 マスタ受信開始関数 R_IIC_Drv_MasterRx()により通信を開始すると、このモードに遷移します。
R_IIC_MODE_COMBINED	マスタ複合動作中 マスタ複合開始関数 R_IIC_Drv_MasterTRx()により通信を開始すると、このモードに遷移します。

6.15 変数一覧

表 6-21 にグローバル変数を示します。

表 6-21 グローバル変数

型	変数名	内容	使用関数
uint8_t	g_iic_ChStatus[MAX_IIC_CH_NUM]	<p>チャンネル状態フラグ</p> <p>表 6-12 に定義する通信状態を確認することができます。</p> <p>初期化する場合、“R_IIC_NO_INIT” を設定してください。</p> <p>以降、強制的に初期化を行う場合を除いて、フラグの変更は行わないでください。</p>	<p>R_IIC_Drv_Init</p> <p>R_IIC_Drv_MasterTx</p> <p>R_IIC_Drv_MasterRx</p> <p>R_IIC_Drv_MasterTRx</p> <p>R_IIC_Drv_Advance</p> <p>R_IIC_Drv_GenClk</p> <p>R_IIC_Drv_Reset</p>
r_iic_drv_internal_event_t	g_iic_Event[MAX_IIC_CH_NUM]	<p>イベントフラグ</p> <p>割り込みが発生した際にセットされ、アドバンス関数でクリアされます。表 6-3 を参照ください。</p>	<p>R_IIC_Drv_Init</p> <p>R_IIC_Drv_MasterTx</p> <p>R_IIC_Drv_MasterRx</p> <p>R_IIC_Drv_MasterTRx</p> <p>R_IIC_Drv_Advance</p>
r_iic_drv_internal_info_t	g_iic_InternalInfo[MAX_IIC_CH_NUM]	<p>内部情報管理</p> <p>本サンプルコードで管理するため、設定は禁止です。</p>	<p>R_IIC_Drv_Init</p> <p>R_IIC_Drv_MasterTx</p> <p>R_IIC_Drv_MasterRx</p> <p>R_IIC_Drv_MasterTRx</p> <p>R_IIC_Drv_Advance</p>
uint32_t	g_iic_ReplyCnt[MAX_IIC_CH_NUM]	<p>アドバンス関数カウンタ</p> <p>アドバンス関数のコール回数の上限值です。ユーザがコールするアドバンス関数で減算されます。イベントが発生すると初期化されます。“0” になるとチャンネル状態フラグとデバイス状態フラグに“R_IIC_ERR_NON_REPLY”がセットされます。</p> <p>カウンタ値はマクロ定義“REPLY_CNT”で変更可能です。システムに合わせて設定してください。</p>	<p>R_IIC_Drv_Init</p> <p>R_IIC_Drv_MasterTx</p> <p>R_IIC_Drv_MasterRx</p> <p>R_IIC_Drv_MasterTRx</p> <p>R_IIC_Drv_Advance</p>
bool	g_iic_Api[MAX_IIC_CH_NUM]	<p>API フラグ</p> <p>本サンプルコードの API の多重コールを防止するために使用します。</p> <p>API 処理開始時にセットされ、終了後にクリアされます。</p>	<p>R_IIC_Drv_Init</p> <p>R_IIC_Drv_MasterTx</p> <p>R_IIC_Drv_MasterRx</p> <p>R_IIC_Drv_MasterTRx</p> <p>R_IIC_Drv_Advance</p> <p>R_IIC_Drv_GenClk</p> <p>R_IIC_Drv_Reset</p>

6.16 関数一覧

表 6-22 に関数一覧を示します。

表 6-22 関数一覧

関数名	説明
R_IIC_Drv_Init()	I ² C ドライバ初期化関数
R_IIC_Drv_MasterTx()	マスタ送信開始関数
R_IIC_Drv_MasterRx()	マスタ受信開始関数
R_IIC_Drv_MasterTRx()	マスタ複合開始関数
R_IIC_Drv_Advance()	アドバンス関数
R_IIC_Drv_GenClk()	SCL 疑似クロック生成関数
R_IIC_Drv_Reset()	I ² C ドライバリセット関数

6.17 状態遷移図

図 6-15 に、各チャネルの状態遷移図を示します。

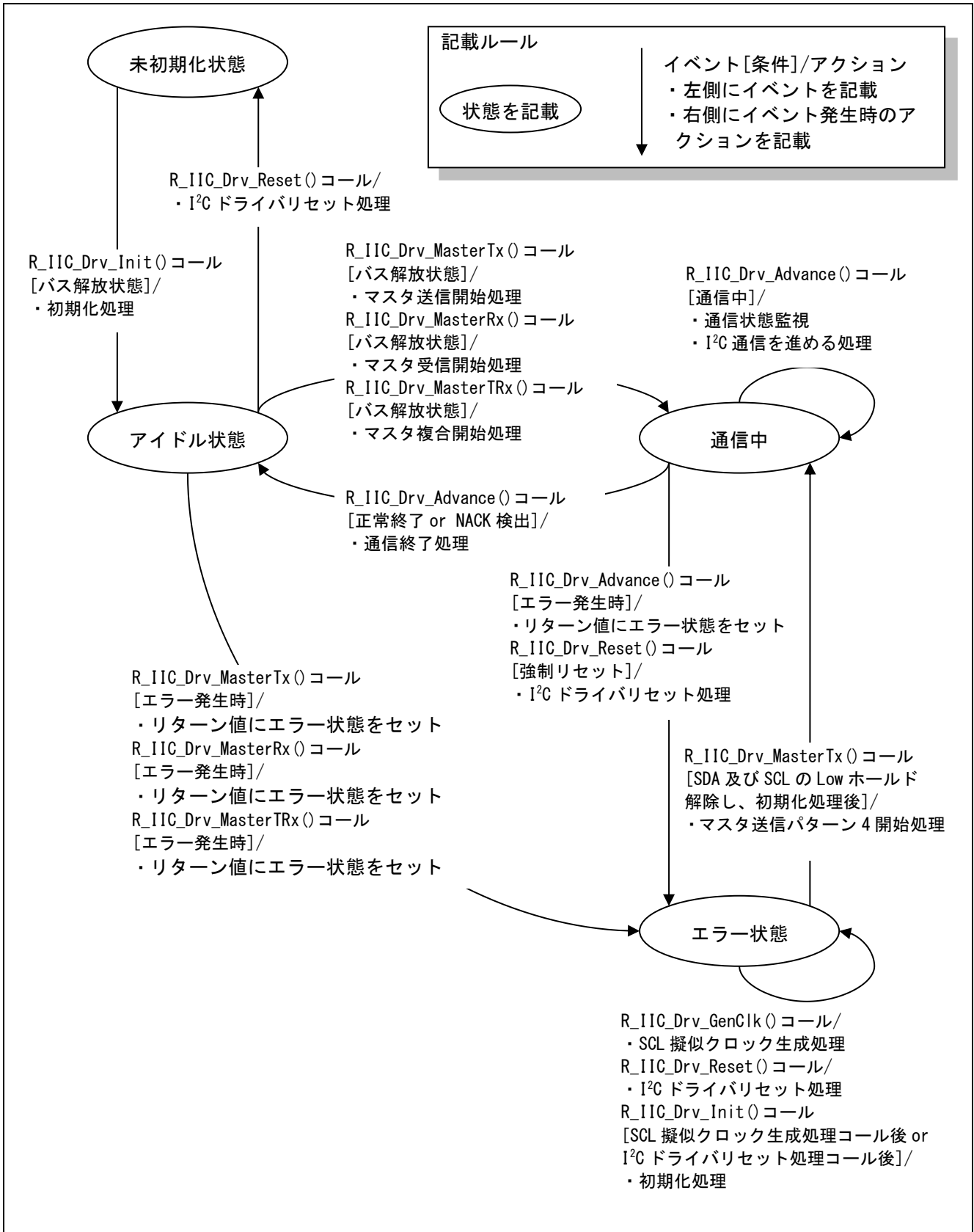


図 6-15 状態遷移図

6.17.1 エラー状態の定義

本サンプルコードでは、以降に示す現象が発生した場合をエラー状態と定義します。エラーの発生は、各 API コール後のリターン値により確認することができます。エラー時の対応方法については、6.18 関数仕様の各 API 項のリターン値を参照ください。

(1) パラメータエラー

リターン値：R_IIC_ERR_PARAM

各 API をコールする際、引数を設定していない場合。

(2) アービトレーションロスト

リターン値：R_IIC_ERR_AL

アービトレーションロストが発生した場合。発生条件は RL78 ファミリ MCU ユーザーズマニュアル ハードウェア編を参照ください。

(3) 未応答エラー

リターン値：R_IIC_ERR_NON_REPLY

以下の場合を未応答エラーとします。

- アドバンス関数のコール回数が上限を超えた場合。
- 各開始関数コール時、一定時間バスを監視したが、バスが解放されなかった場合。
- スタートコンディション生成処理を行ったが、一定時間が経過しても検出できなかった場合。
- アドバンス関数コール時、ストップコンディション生成処理を行ったが、一定時間が経過しても検出できなかった場合。

尚、バス監視とスタートコンディション生成の待ち時間はソフトウェアループで行います。カウンタ値はユーザで設定可能です。使用するシステム・クロックに合わせて設定してください。カウンタの定義については、表 6-14 を参照ください。

(4) SDA Low ホールド（復帰不可）

リターン値：R_IIC_ERR_SDA_LOW_HOLD

SCL 疑似クロック生成を行ったが、SDA が Low ホールドのままの場合。

(5) その他エラー

リターン値：R_IIC_ERR_OTHER

上記（1）～（4）以外エラーとします。

6.17.2 状態遷移時の各フラグの状態

状態遷移時の各フラグの状態を表 6-23 に示します。

表 6-23 状態遷移時の各フラグの状態一覧

状態	チャンネル状態フラグ	デバイス状態フラグ (通信のデバイス)	I ² C プロトコルの動作 モード	プロトコル制御の現状態
	g_iic_ChStatus[]	I ² C 通信情報構造体 *pDevStatus	内部通信情報構造体 Mode	内部通信情報構造体 N_status
未初期化状態	R_IIC_NO_INIT	R_IIC_NO_INIT	R_IIC_MODE_NONE	R_IIC_STS_NO_INIT
アイドル状態	R_IIC_IDLE R_IIC_FINISH R_IIC_NACK	R_IIC_IDLE R_IIC_FINISH R_IIC_NACK	R_IIC_MODE_NONE	R_IIC_STS_IDLE
通信中 (マスタ 送信)	R_IIC_COMMUNICATION	R_IIC_COMMUNICATION	R_IIC_MODE_WRITE	R_IIC_STS_ST_COND_WAIT
				R_IIC_STS_SEND_SLVADR_W_WAIT
				R_IIC_STS_SEND_SLVADR_R_WAIT
				R_IIC_STS_SEND_DATA_WAIT
				R_IIC_STS_RECEIVE_DATA_WAIT
通信中 (マスタ 受信)	R_IIC_COMMUNICATION	R_IIC_COMMUNICATION	R_IIC_MODE_READ	R_IIC_STS_ST_COND_WAIT
				R_IIC_STS_SEND_SLVADR_W_WAIT
				R_IIC_STS_SEND_SLVADR_R_WAIT
				R_IIC_STS_SEND_DATA_WAIT
				R_IIC_STS_RECEIVE_DATA_WAIT
通信中 (マスタ 複合)	R_IIC_COMMUNICATION	R_IIC_COMMUNICATION	R_IIC_MODE_COMBINED	R_IIC_STS_ST_COND_WAIT
				R_IIC_STS_SEND_SLVADR_W_WAIT
				R_IIC_STS_SEND_SLVADR_R_WAIT
				R_IIC_STS_SEND_DATA_WAIT
				R_IIC_STS_RECEIVE_DATA_WAIT
エラー状態	R_IIC_ERR_PARAM	R_IIC_ERR_PARAM	-	-
	R_IIC_ERR_AL	R_IIC_ERR_AL	-	-
	R_IIC_ERR_NON_REPLY	R_IIC_ERR_NON_REPLY	-	-
	R_IIC_ERR_SCL_GENCLK	R_IIC_ERR_SCL_GENCLK	-	-
	R_IIC_ERR_OTHER	R_IIC_ERR_OTHER	-	-

6.18 関数仕様

6.18.1 関数共通処理

本サンプルコードでは、1 度に動作可能な API は 1 つであり、API 処理実行中に本サンプルコードの API がコールされた場合、処理を行わずに終了します。“R_IIC_LOCK_FUNC” がリターン値として返ります。

API の同時コールを防止するために API フラグを用意しています。このフラグは、API 処理が行われている間セットされます。各 API 処理の最初に必ずチェックされ、フラグがセットされていない場合は、その処理を実行する仕組みです。図 6-20～図 6-23 に概略フローを示します。

この処理は、6.16 項で定義している関数に対して実施します。以降、6.18.2 項以降では図 6-20 以降の「API 関数処理」の処理内容について記載します。

また、μITRON の OS を使用した場合、I²C ドライバ初期化時と通信開始時にセマフォを使用したタスク管理を行います。図 6-19 にセマフォの動作例を示します。

I²C ドライバ初期化関数では、初期化処理前にセマフォを取得し、初期化処理後にセマフォを解放します。通信時では、各開始関数の処理前にセマフォを取得します。各開始関数で通信を開始した場合、そのままセマフォを保持し、アドバンス関数で通信が終了するか、エラー終了した際にセマフォを解放します。

なお、本サンプルコードには μITRON の OS 制御が組み込まれていません。OS 制御例を以下の各関数の概略フローに記載しています。ご参照いただき、ユーザにて処理を追加してください。

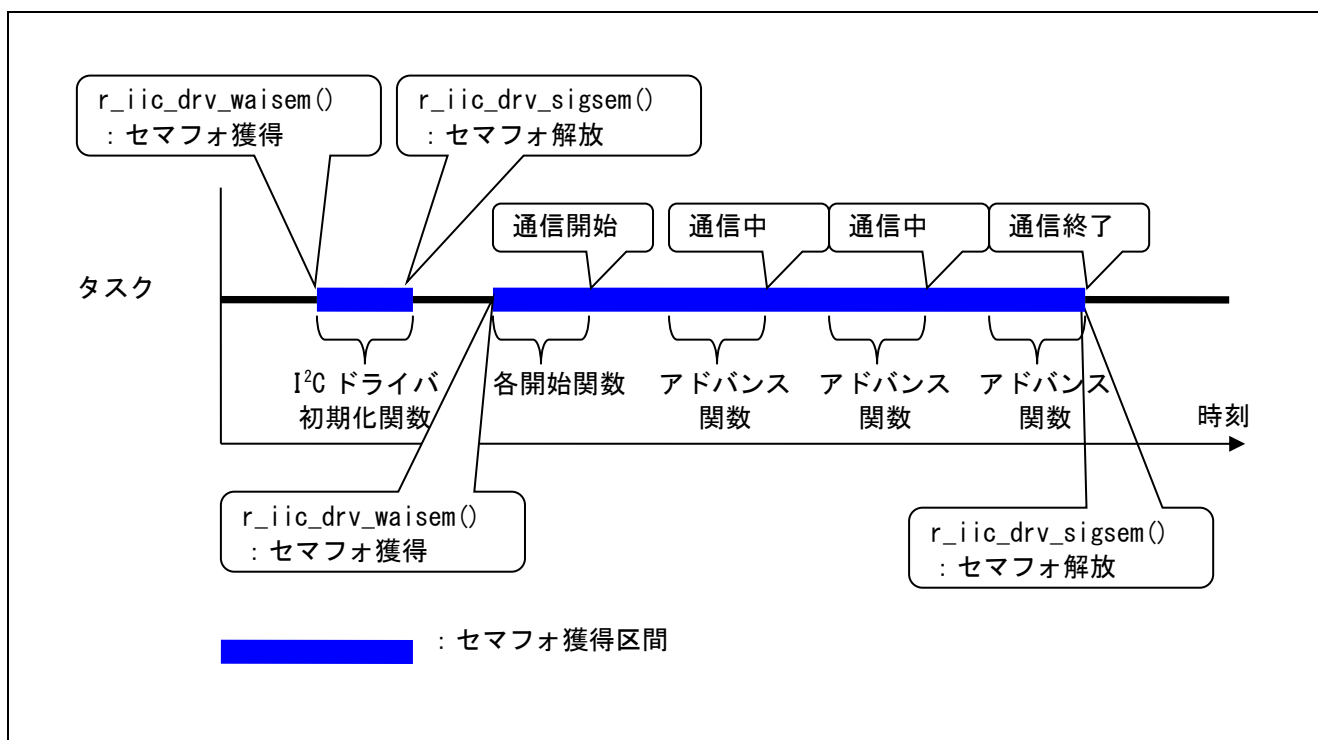


図 6-19 セマフォの動作例

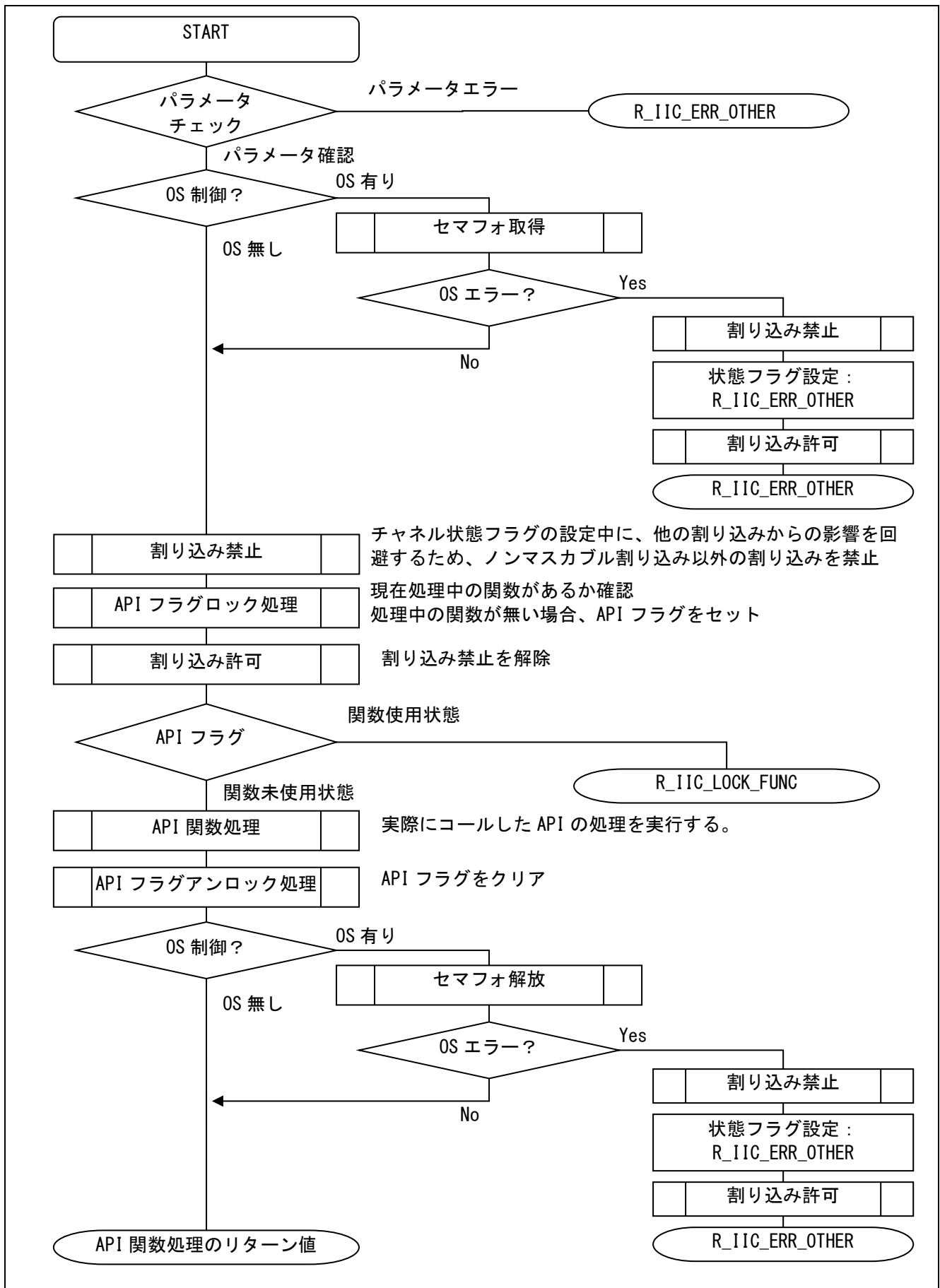


図 6-20 I²C ドライバ初期化関数多重コール防止処理概略フロー

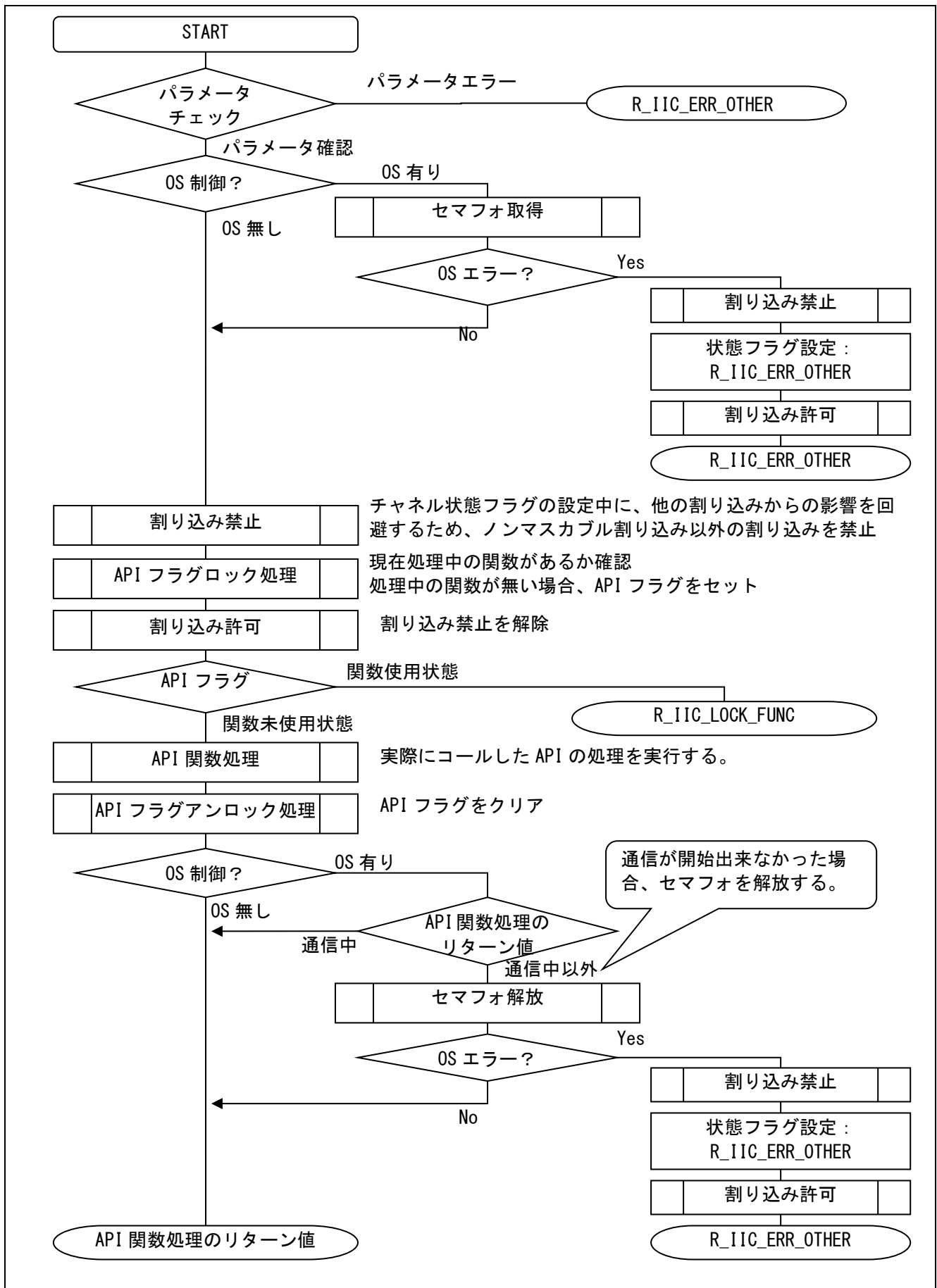


図 6-21 各開始関数多重コール防止処理概略フロー

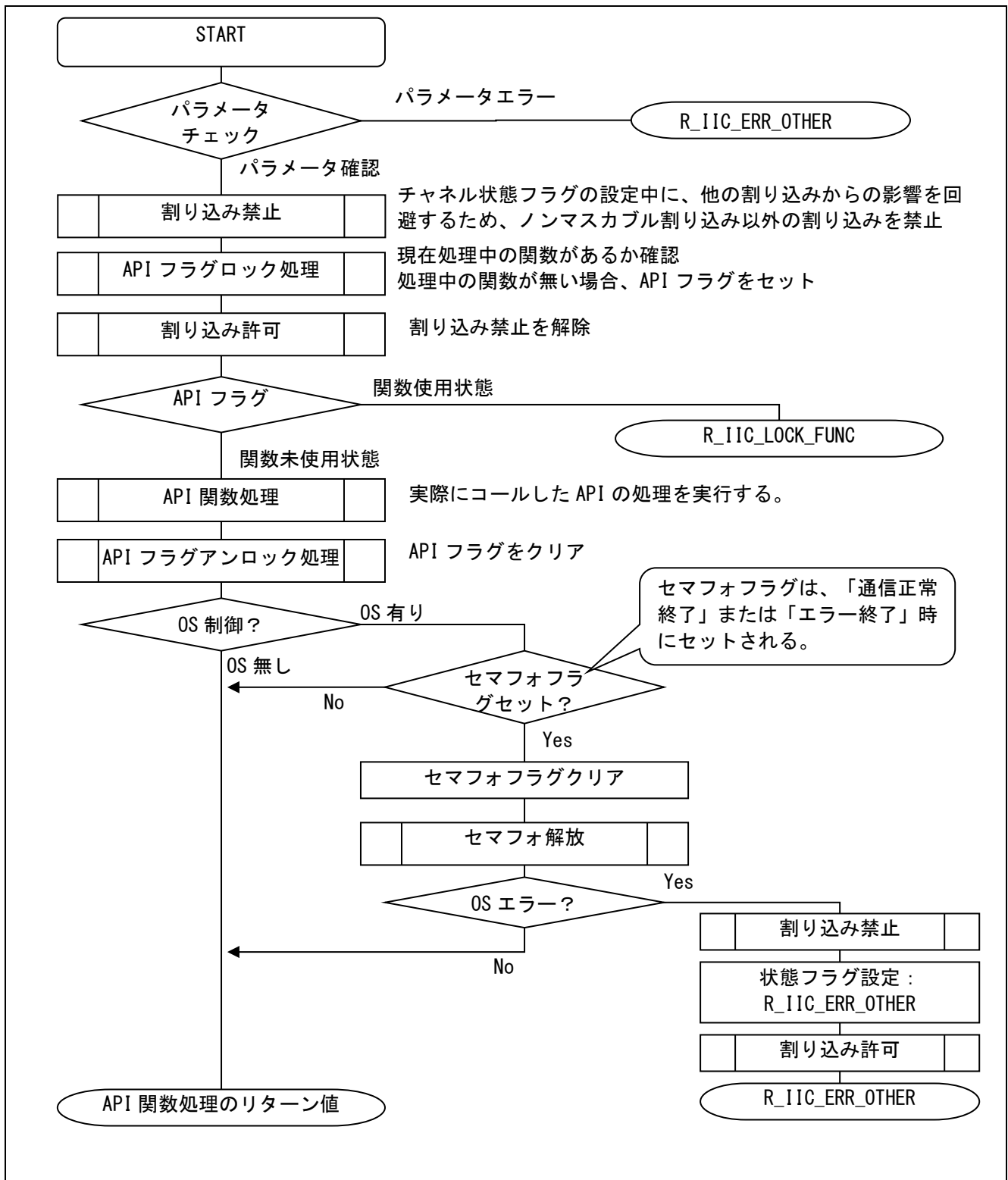


図 6-22 アドバンス関数多重コール防止処理概略フロー

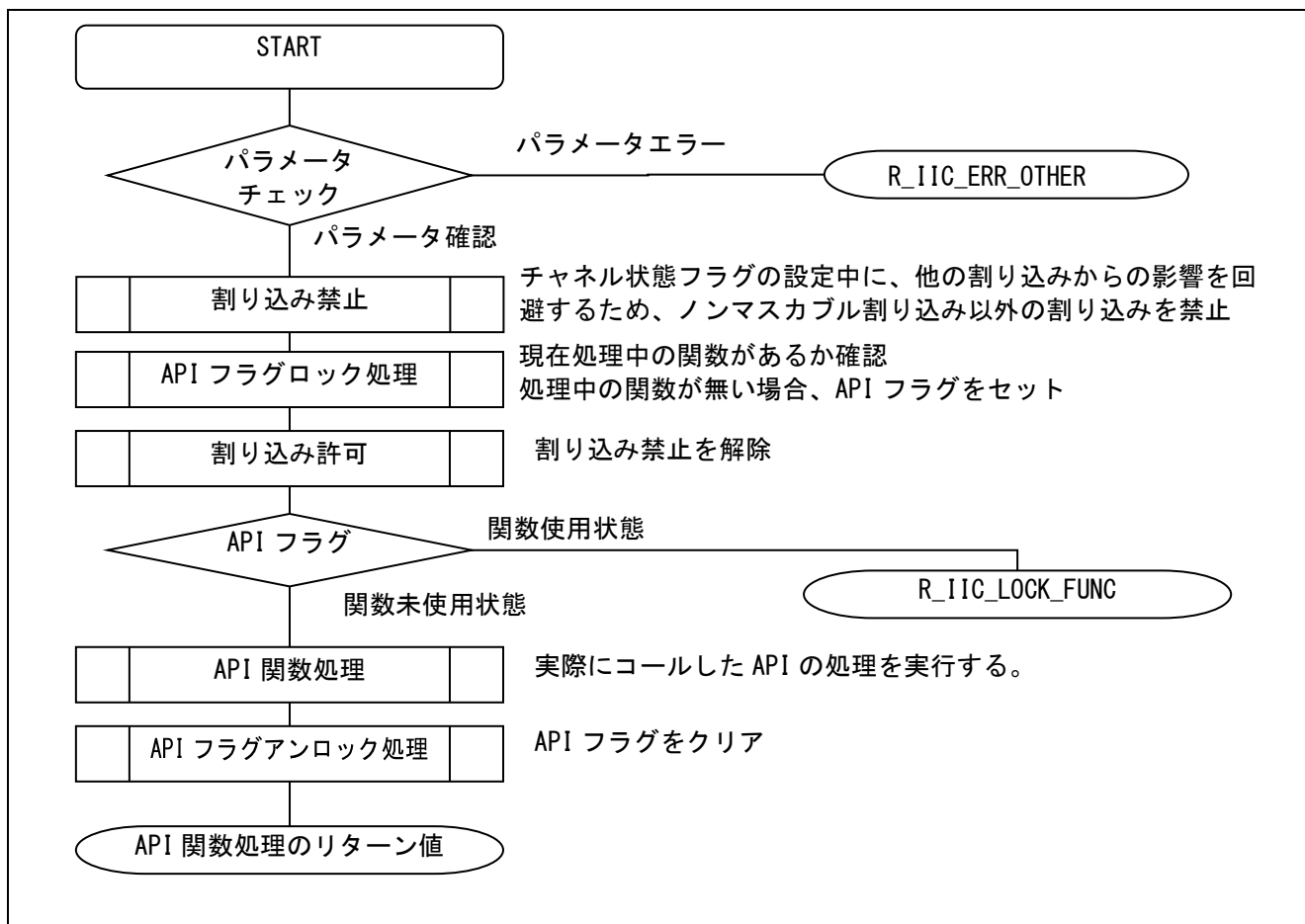


図 6-23 SCL 疑似クロック生成関数、及び I²C ドライバリセット関数多重コール防止処理概略フロー

6.18.2 I²C ドライバ初期化関数

R_IIC_Drv_Init	
概要	I ² C ドライバ初期化関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_Init(r_iic_drv_info_t *pRlic_Info)
説明	<ul style="list-style-type: none"> 対象チャンネルの初期設定を行います。 本処理を行うためには、以下の設定が必要です。 構造体 “r_iic_drv_info_t” メンバの ChNo ;使用チャンネル番号 チャンネル状態フラグ (g_iic_ChStatus[]) ;“R_IIC_NO_INIT” をセット (※1) デバイス状態フラグ (*(pRlic_Info.pDevStatus)) ;“R_IIC_NO_INIT” をセット (※1)
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
リターン値	<p>R_IIC_IDLE 【チャンネルが未初期化状態】の場合、初期化を行いアイドル状態に遷移しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_IDLE” を設定しました。</p> <p>【すでに初期化済み】の場合、初期化は行わず、デバイス状態フラグに “R_IIC_IDLE” を設定しました。 →開始関数をコールすることで通信が可能です。</p> <p>R_IIC_FINISH / R_IIC_NACK 前処理の実行結果です。開始関数が実行可能であるため、初期化は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →開始関数をコールすることで通信が可能です。</p> <p>R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。</p> <p>R_IIC_BUS_BUSY 通信中です。初期化できませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →アドバンス関数をコールして通信を終了させてください。</p> <p>R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。</p> <p>R_IIC_ERR_AL アービトレーションロストが発生しています。チャンネル状態フラグとデバイス状態フラグは変更されません。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_NON_REPLY 未応答エラーが発生しています。チャンネル状態フラグとデバイス状態フラグは変更されません。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_SDA_LOW_HOLD SDA は Low ホールドから復帰できていない状態です。チャンネル状態フラグとデバイス状態フラグは変更されません。 →スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。</p> <p>R_IIC_ERR_OTHER その他エラーが発生しています。チャンネル状態フラグとデバイス状態フラグに</p>

備考

- “R_IIC_ERR_OTHER” を設定しました。
- すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。
- 以下を確認してください。
- ・ I²C 通信情報構造体の設定が正しく行われているか確認してください。
 - ・ OS 制御でエラーが発生していないか確認してください。
 - ・ パラメータを確認します。
 - ・ チャンネル状態フラグの状態を確認します。
 - ・ 初期設定処理を行います。以下の処理が行われます。
 - I²C 周辺ハードウェアへのクロック供給を許可 (I²C レジスタ設定可能)
 - ポート設定を入力モード、出力ラッチは 0 に設定
 - 転送クロックの設定、自局アドレスの設定、開始条件の設定
 - I²C 割り込み禁止設定
 - RAM 初期化
 - ・ チャンネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスカブル割り込み以外の割り込みを禁止しています。
- ※ 1 : 初期化関数をコールする場合、その前に “R_IIC_NO_INIT” をセットしてください。設定しないまま初期化関数をコールすると、初期化処理を行わない場合があります。

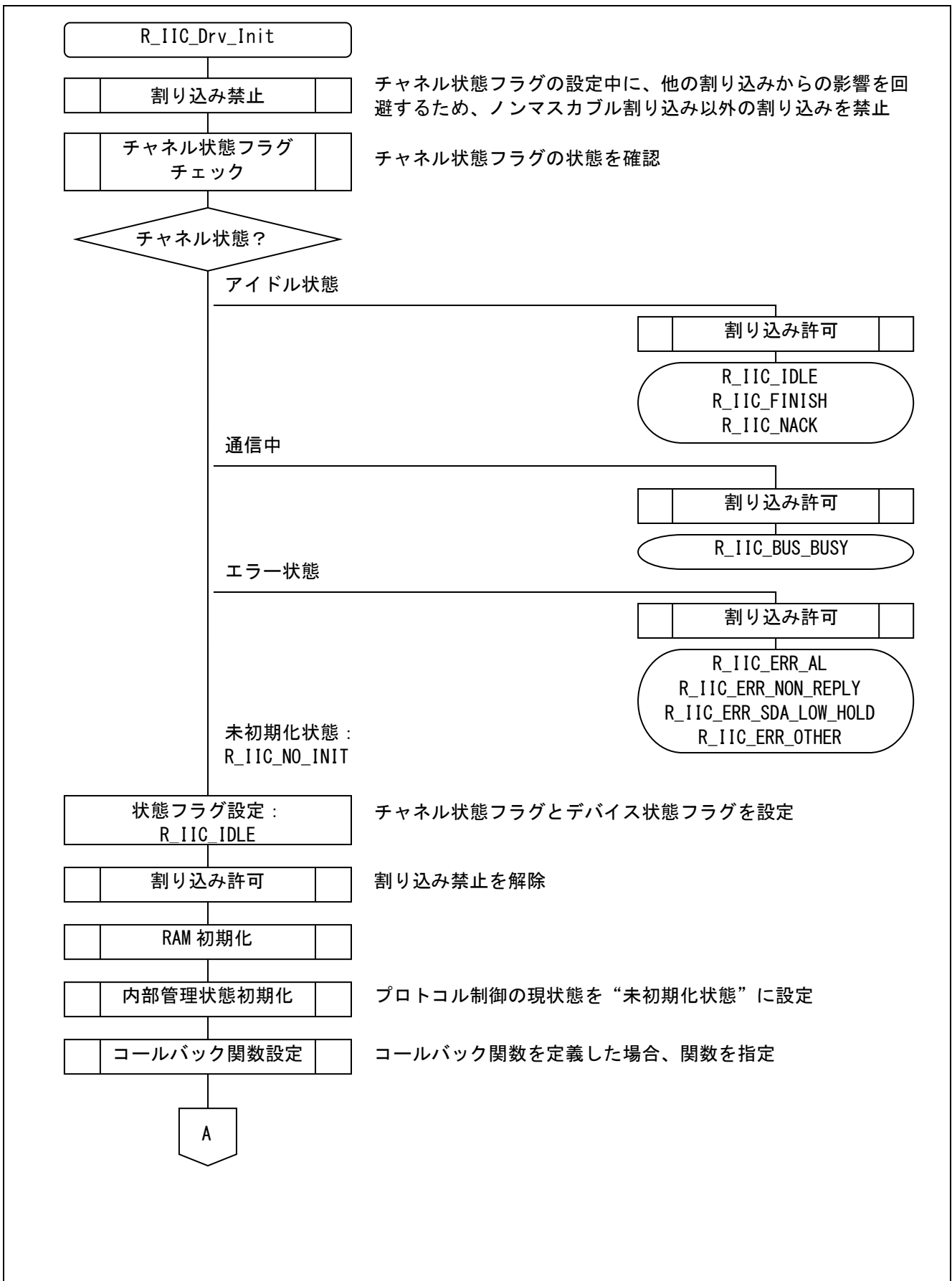


図 6-24 I²C ドライバ初期化関数概略フロー (1/2)

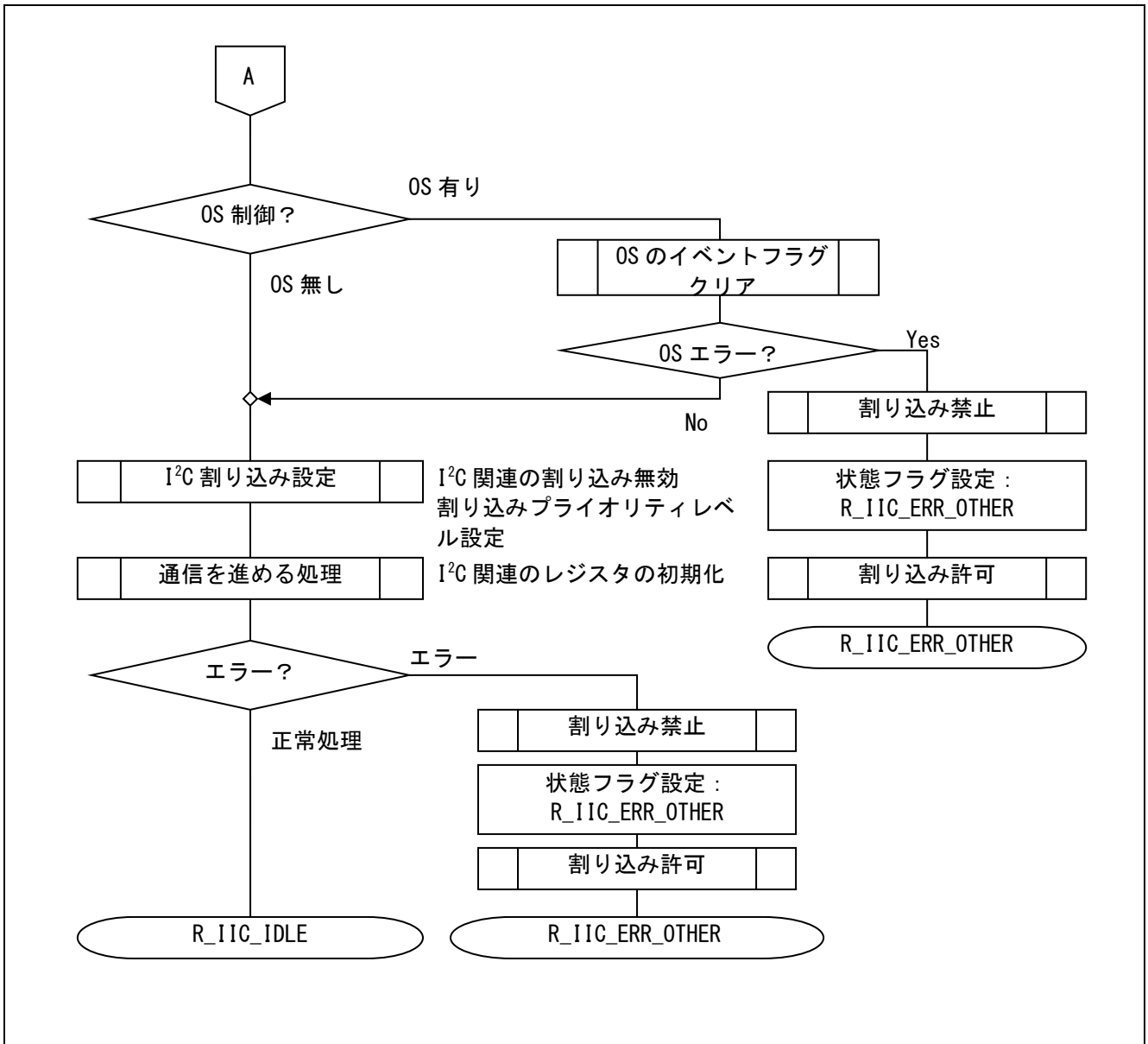


図 6-25 I²C ドライバ初期化関数概略フロー (2/2)

6.18.3 マスタ送信開始関数

R_IIC_Drv_MasterTx	
概要	マスタ送信開始関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_MasterTx (r_iic_drv_info_t *pRlic_Info)
説明	<ul style="list-style-type: none"> ・ マスタ送信を開始します。 ・ 本処理を行うためには、I²C 通信情報構造体 “r_iic_drv_info_t” の設定が必要です。設定方法は表 6-17 を参照してください。
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
リターン値	<p>R_IIC_COMMUNICATION マスタ送信を開始しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_COMMUNICATION” を設定しました。 →アドバンス関数をコールして通信を完了させてください。</p> <p>R_IIC_NO_INIT 初期化されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →初期化関数をコールして初期化を終了させてください。</p> <p>R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。</p> <p>R_IIC_BUS_BUSY 通信中です。マスタ送信を開始できませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →アドバンス関数をコールして通信を終了させてください。</p> <p>R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。</p> <p>R_IIC_ERR_AL アービトラクションロストが発生しています。チャンネル状態フラグとデバイス状態フラグは変更されません。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_NON_REPLY バスが開放されていないか、スタートコンディションを検出できませんでした。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_NON_REPLY” を設定しました。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_SDA_LOW_HOLD SDA は Low ホールドから復帰できていない状態です。チャンネル状態フラグとデバイス状態フラグは変更されません。 →スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。</p> <p>R_IIC_ERR_OTHER その他エラーが発生しています。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_OTHER” を設定しました。 すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →以下を確認してください。 ・ I²C 通信情報構造体の設定が正しく行われているか確認してください。</p>

備考

- ・ OS 制御でエラーが発生していないか確認してください。
- ・ パラメータを確認します。
- ・ チャネルの通信状態を確認します。(チャンネル状態フラグ g_iic_ChStatus[])
- ・ バス状態 (通信/解放) を確認します。(IICA レジスタの I²C バス状態フラグ)
- ・ I²C プロトコル動作モード (g_iic_InternalInfo[].Mode) に “R_IIC_MODE_WRITE (マスタ送信)” を設定します。
- ・ I²C 割り込みを許可します。
- ・ ポート設定を出力モードにします。
- ・ スタートコンディションを生成します。
- ・ スレーブアドレス [転送方向:Write] を送信します。
- ・ アドバンス関数カウンタを初期化します。
- ・ 本関数から戻った時点では、I²C 通信は終了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。
- ・ 開始関数コール後の通信状態は、アドバンス関数のリターン値で確認できます。
- ・ チャネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスカブル割り込み以外の割り込みを禁止しています。

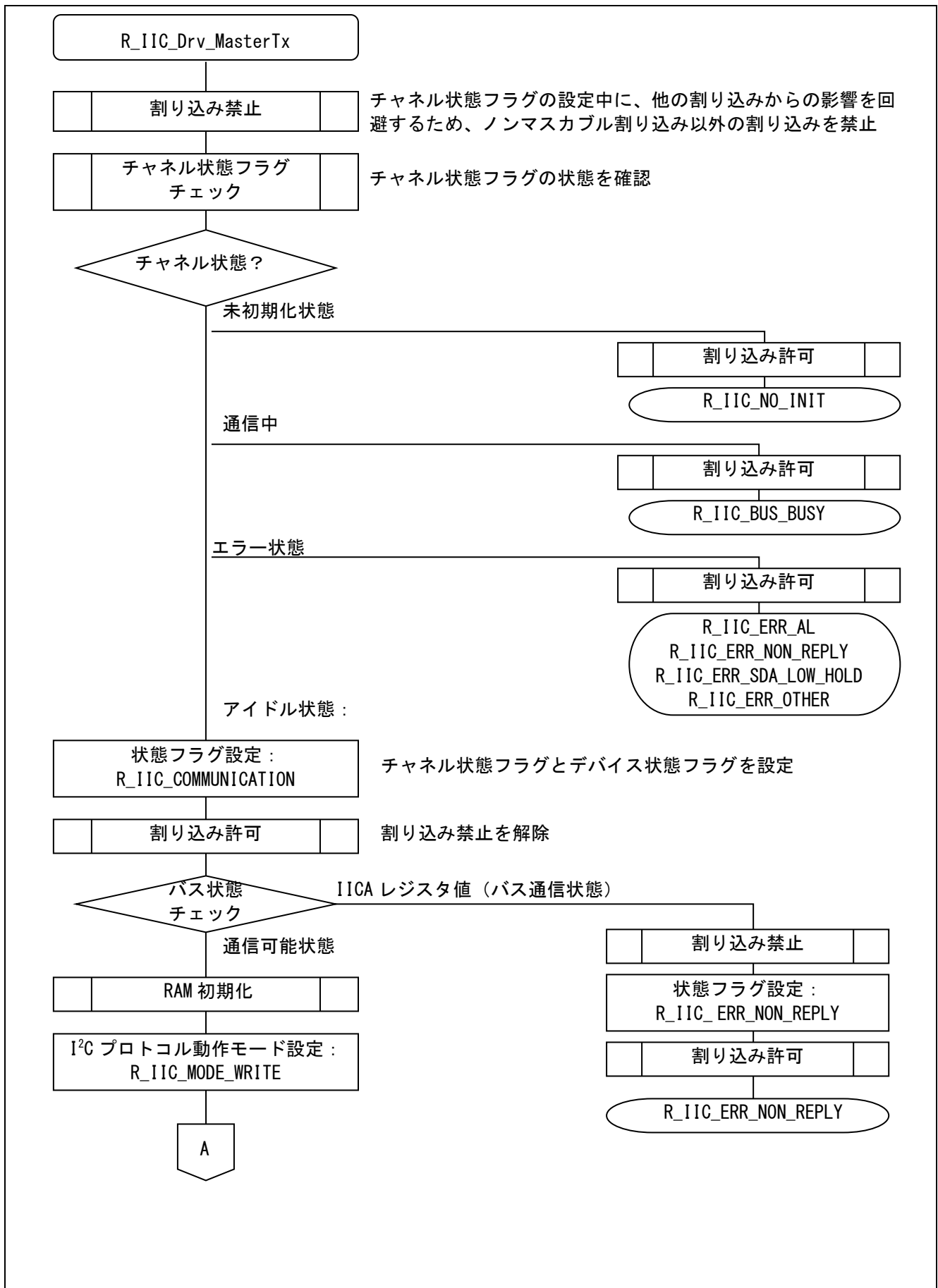


図 6-26 マスタ送信開始関数概略フロー (1/2)

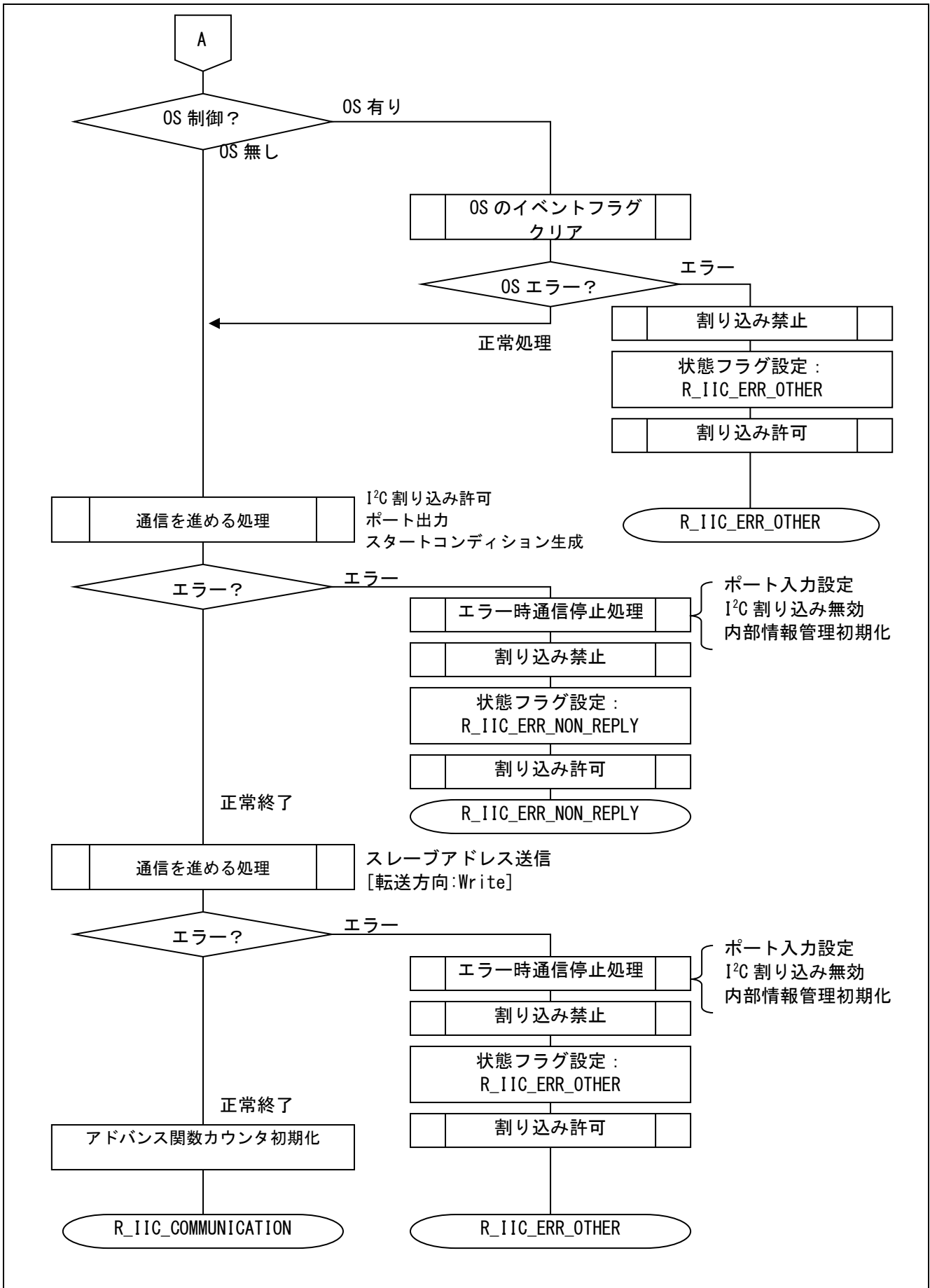


図 6-27 マスタ送信開始関数概略フロー (2/2)

6.18.4 マスタ受信開始関数

R_IIC_Drv_MasterRx	
概要	マスタ受信開始関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_MasterRx (r_iic_drv_info_t *pRlic_Info)
説明	<ul style="list-style-type: none"> ・ マスタ受信を開始します。 ・ 本処理を行うためには、I²C 通信情報構造体 “r_iic_drv_info_t” の設定が必要です。設定方法は表 6-18 を参照してください。
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
リターン値	<p>R_IIC_COMMUNICATION マスタ受信を開始しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_COMMUNICATION” を設定しました。 →アドバンス関数をコールして通信を終了させてください。</p> <p>R_IIC_NO_INIT 初期化されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →初期化関数をコールして初期化を終了させてください。</p> <p>R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。</p> <p>R_IIC_BUS_BUSY 通信中です。マスタ受信を開始できませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →アドバンス関数をコールして通信を終了させてください。</p> <p>R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。</p> <p>R_IIC_ERR_AL アービトラクションロストが発生しています。チャンネル状態フラグとデバイス状態フラグは変更されません。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_NON_REPLY バスが開放されていないか、スタートコンディションを検出できませんでした。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_NON_REPLY” を設定しました。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_SDA_LOW_HOLD SDA は Low ホールドから復帰できていない状態です。チャンネル状態フラグとデバイス状態フラグは変更されません。 →スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。</p> <p>R_IIC_ERR_OTHER その他エラーが発生しています。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_OTHER” を設定しました。 すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →以下を確認してください。 ・ I²C 通信情報構造体の設定が正しく行われているか確認してください。</p>

備考

- ・ OS 制御でエラーが発生していないか確認してください。
- ・ パラメータを確認します。
- ・ チャネルの通信状態を確認します。(チャンネル状態フラグ g_iic_ChStatus[])
- ・ バス状態 (通信/解放) を確認します。(IICA レジスタのバス状態フラグ)
- ・ I²C プロトコル動作モード (g_iic_InternalInfo[].Mode) に “R_IIC_MODE_READ (マスタ受信)” を設定します。
- ・ I²C 割り込みを許可します。
- ・ ポート設定を出力モードにします。
- ・ スタートコンディションを生成します。
- ・ スレーブアドレス [転送方向:Read] を送信します。
- ・ アドバンス関数カウンタを初期化します。
- ・ 本関数から戻った時点では、I²C 通信は終了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。
- ・ 開始関数コール後の通信状態は、アドバンス関数のリターン値で確認できます。
- ・ チャネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスカブル割り込み以外の割り込みを禁止しています。

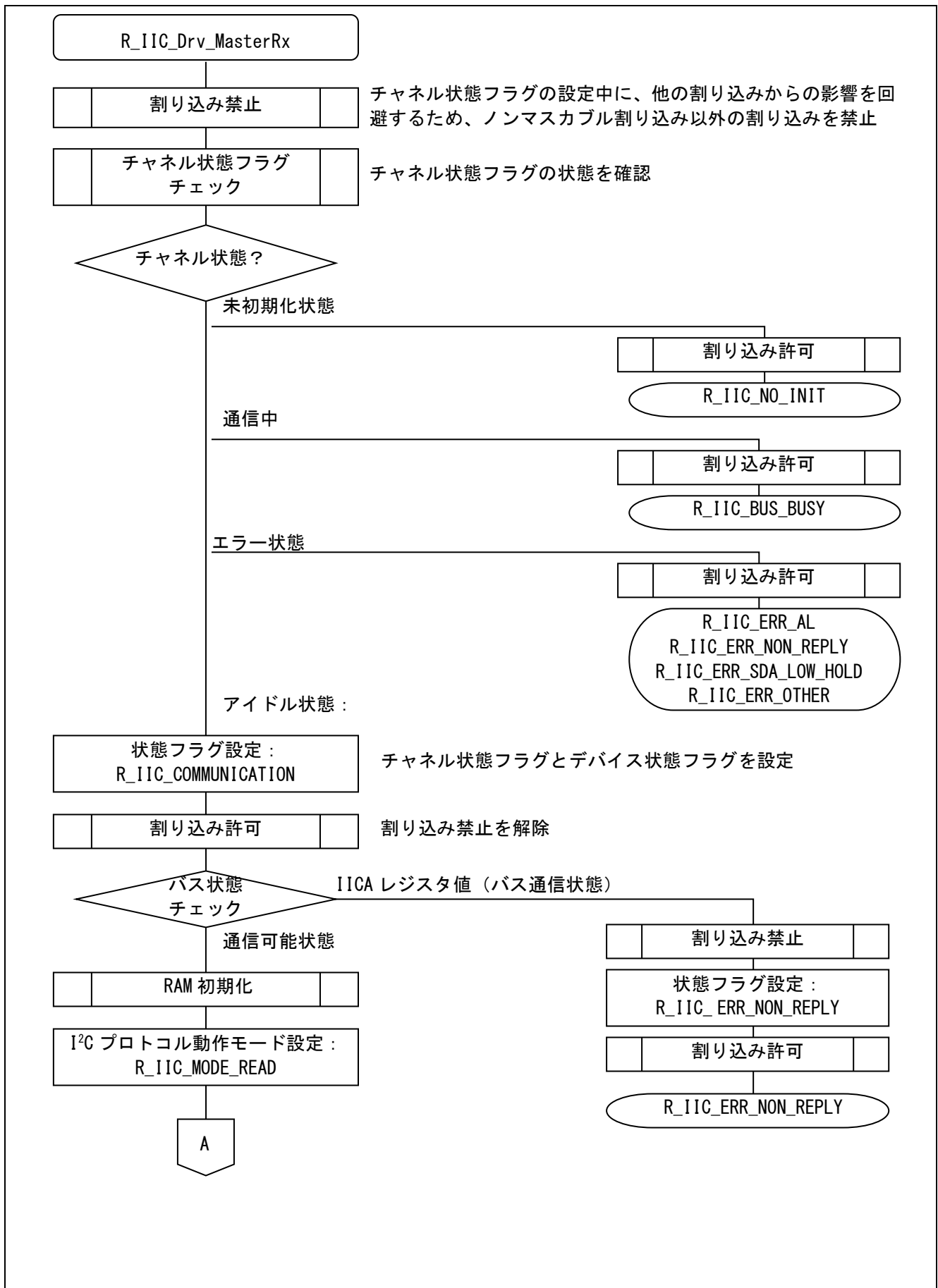


図 6-28 マスタ受信開始関数概略フロー (1/2)

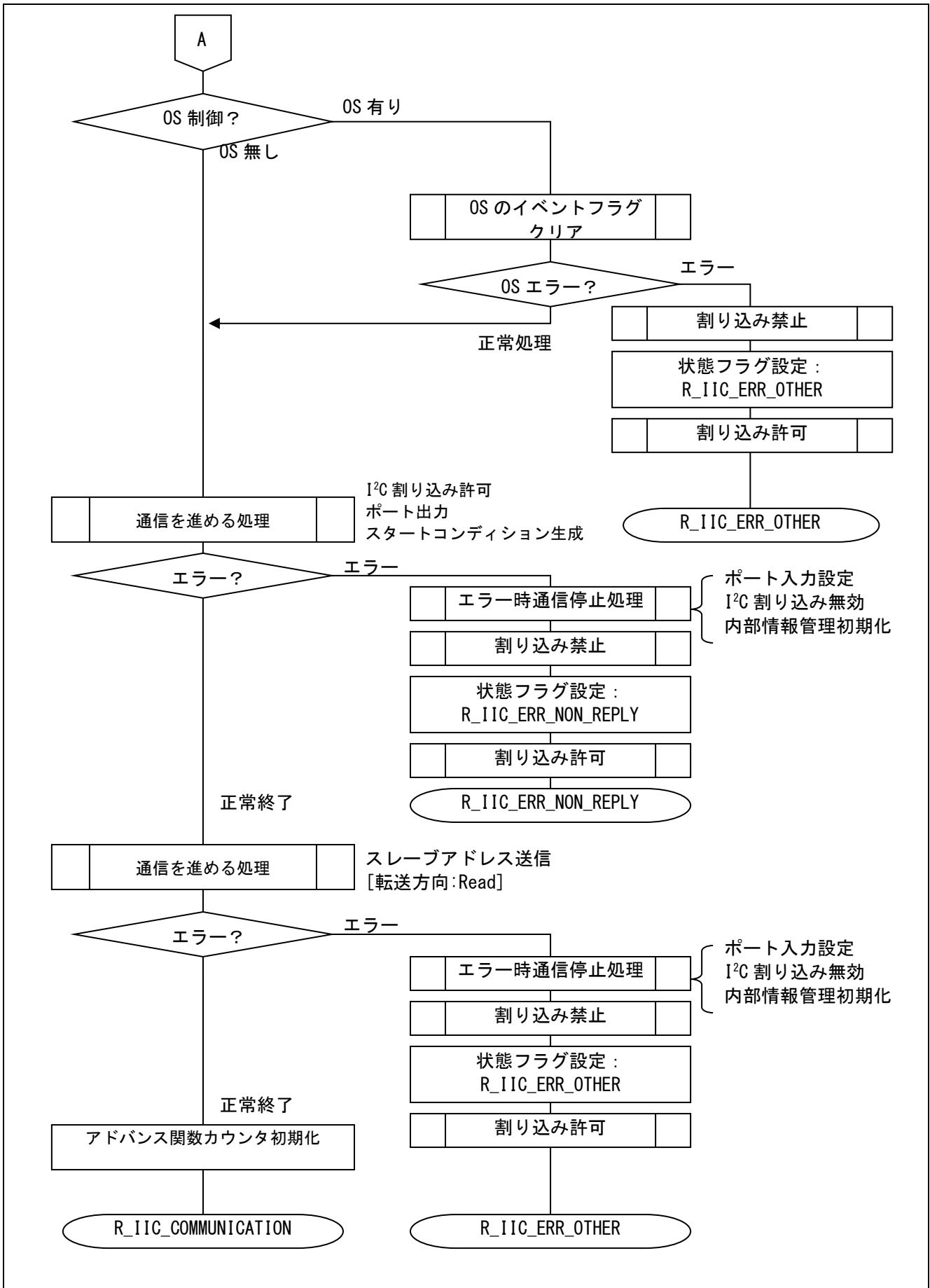


図 6-29 マスタ受信開始関数概略フロー (2/2)

6.18.5 マスタ複合開始関数

R_IIC_Drv_MasterTRx	
概要	マスタ複合開始関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_MasterTRx (r_iic_drv_info_t *pRlic_Info)
説明	<ul style="list-style-type: none"> ・ マスタ複合を開始します。 ・ 本処理を行うためには、I²C 通信情報構造体 “r_iic_drv_info_t” の設定が必要です。設定方法は表 6-18 を参照してください。
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
リターン値	<p>R_IIC_COMMUNICATION マスタ複合を開始しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_COMMUNICATION” を設定しました。 →アドバンス関数をコールして通信を終了させてください。</p> <p>R_IIC_NO_INIT 初期化されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →初期化関数をコールして初期化を終了させてください。</p> <p>R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。</p> <p>R_IIC_BUS_BUSY 通信中です。マスタ複合を開始できませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →アドバンス関数をコールして通信を終了させてください。</p> <p>R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。</p> <p>R_IIC_ERR_AL アービトラクションロストが発生しています。チャンネル状態フラグとデバイス状態フラグは変更されません。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_NON_REPLY バスが開放されていないか、スタートコンディションを検出できませんでした。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_NON_REPLY” を設定しました。 →7.6 復帰処理例を参照し、復帰処理を行ってください。</p> <p>R_IIC_ERR_SDA_LOW_HOLD SDA は Low ホールドから復帰できていない状態です。チャンネル状態フラグとデバイス状態フラグは変更されません。 →スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。</p> <p>R_IIC_ERR_OTHER その他エラーが発生しています。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_OTHER” を設定しました。 すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →以下を確認してください。 ・ I²C 通信情報構造体の設定が正しく行われているか確認してください。</p>

備考

- ・ OS 制御でエラーが発生していないか確認してください。
- ・ パラメータを確認します。
- ・ チャネルの通信状態を確認します。(チャネル状態フラグ g_iic_ChStatus[])
- ・ バス状態 (通信/解放) を確認します。(IICA レジスタのバス状態フラグ)
- ・ I²C プロトコル動作モード (g_iic_InternallInfo[].Mode) に “R_IIC_MODE_COMBINED (マスタ複合)” を設定します。
- ・ I²C 割り込みを許可します。
- ・ ポート設定を出力モードにします。
- ・ スタートコンディションを生成します。
- ・ スレーブアドレス[転送方向:Write]を送信します。
- ・ アドバンス関数カウンタを初期化します。
- ・ 本関数から戻った時点では、I²C 通信は終了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。
- ・ 開始関数コール後の通信状態は、アドバンス関数のリターン値で確認できます。
- ・ チャネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスカブル割り込み以外の割り込みを禁止しています。

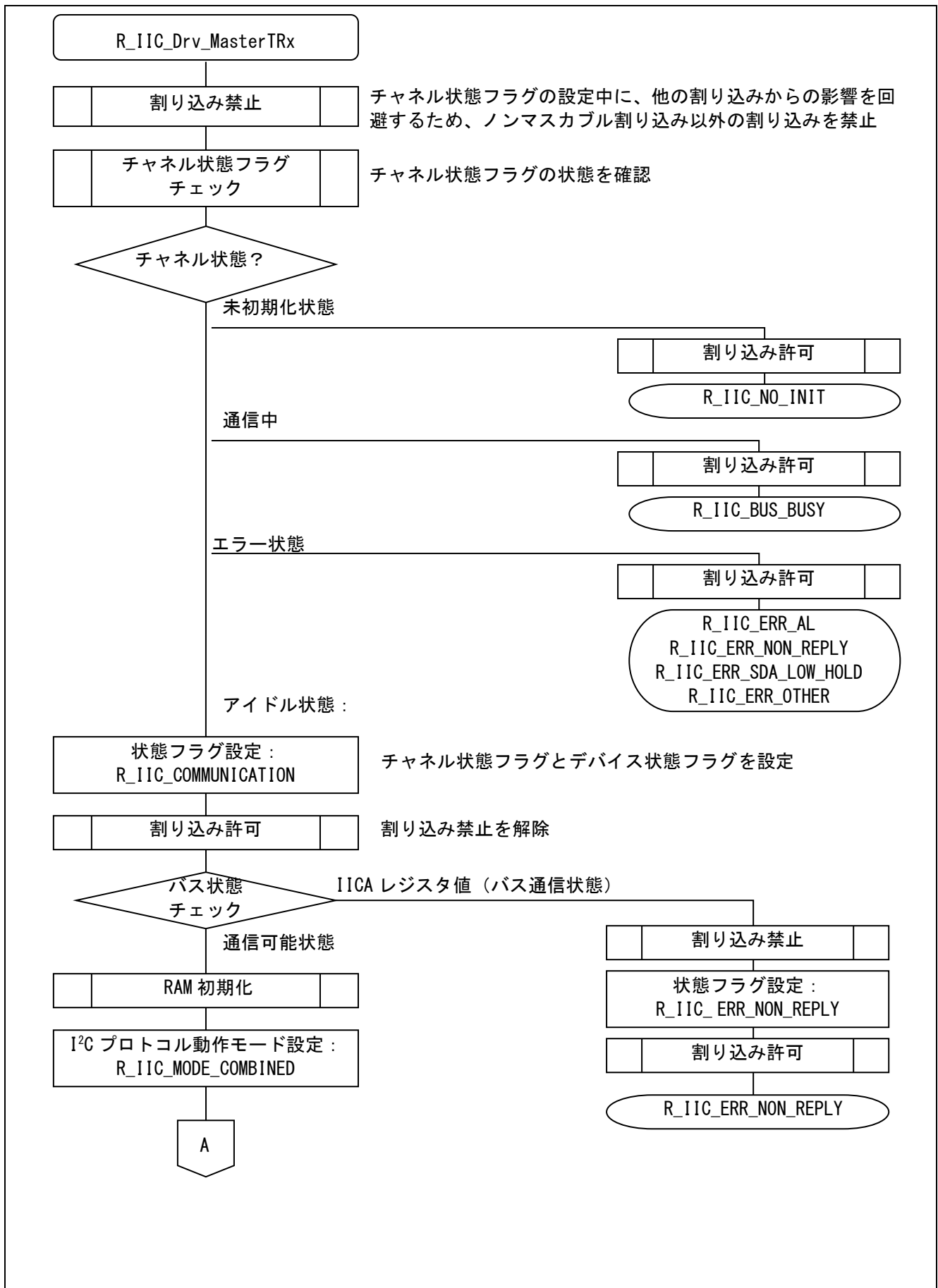


図 6-30 マスタ複合開始関数概略フロー (1/2)

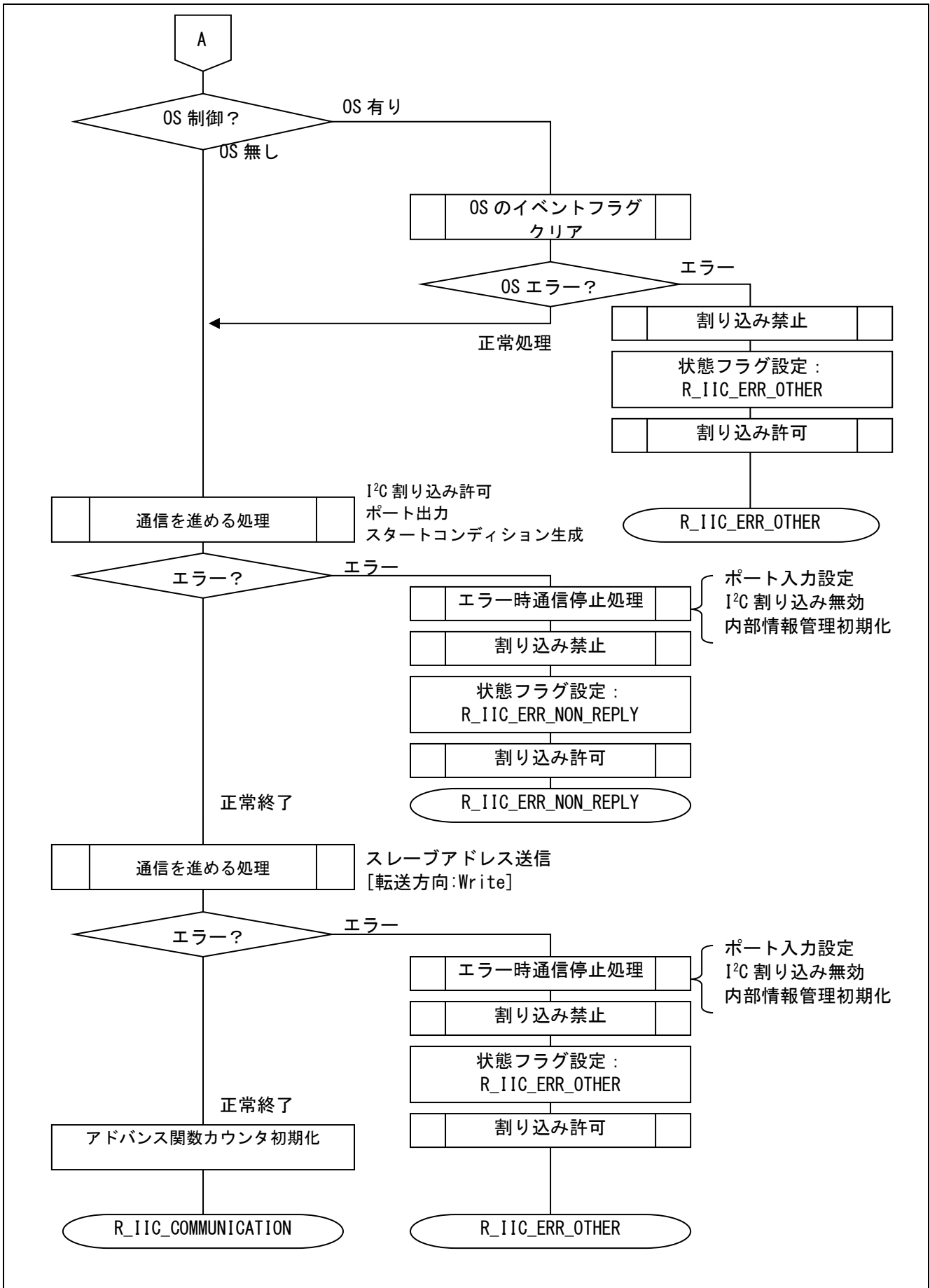


図 6-31 マスタ複合開始関数概略フロー (2/2)

6.18.6 アドバンス関数

R_IIC_Drv_Advance	
概要	アドバンス関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_Advance (r_iic_drv_info_t *pRlic_Info)
説明	<ul style="list-style-type: none"> ・通信を監視し、通信を進める処理を実行します。リターン値に通信の状態を返します。 ・次の通信を開始するには、アドバンス関数で通信を終了させる必要があります。
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
リターン値	<p>R_IIC_COMMUNICATION 通信中です。チャンネル状態フラグとデバイス状態フラグは変更されません。 →アドバンス関数をコールして、通信を終了させてください。</p> <p>R_IIC_FINISH 全通信は正常終了しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_FINISH” を設定しました。 すでに通信が終了している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →開始関数をコールすることで通信が可能です。</p> <p>R_IIC_NACK NACK を検出しました。ストップコンディションを生成し、通信を終了しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_NACK” を設定しました。 すでに通信が終了している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →開始関数をコールすることで通信が可能です。</p> <p>R_IIC_NO_INIT 初期化されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →初期化関数をコールして初期化を終了させてください。</p> <p>R_IIC_IDLE アイドル状態です。チャンネル状態フラグとデバイス状態フラグは変更されません。 →開始関数をコールすることで通信が可能です。</p> <p>R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。</p> <p>R_IIC_BUS_BUSY 同一チャンネル上の他のデバイスが通信中のため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他のデバイスの通信を終了させてください。</p> <p>R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。</p> <p>R_IIC_ERR_AL アービトレーションロストが発生しました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_ERR_AL” を設定しました。</p>

すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。

→7.6 復帰処理例を参照し、復帰処理を行ってください。

R_IIC_ERR_NON_REPLY

下記が発生しました。チャンネル状態フラグとデバイス状態フラグに“R_IIC_ERR_NON_REPLY”を設定しました。

- ・アドバンス関数のコール回数が上限を超えた
- ・ストップコンディション生成処理を行ったが、一定時間が経過しても検出できなかった

すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。

→ノイズ等の影響により SDA または SCL が Low ホールドした可能性があります。7.6 復帰処理例を参照し、復帰処理を行ってください。

R_IIC_ERR_SDA_LOW_HOLD

SDA は Low ホールドから復帰できていない状態です。チャンネル状態フラグとデバイス状態フラグは変更されません。

→スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。

R_IIC_ERR_OTHER

その他エラーが発生しています。チャンネル状態フラグとデバイス状態フラグに“R_IIC_ERR_OTHER”を設定しました。

すでにエラーが発生している場合、何も処理は行いません。チャンネル状態フラグとデバイス状態フラグは変更されません。

→以下を確認してください。

- ・I²C 通信情報構造体の設定が正しく行われているか確認してください。
- ・OS 制御でエラーが発生していないか確認してください。

備考

- ・パラメータを確認します。
- ・イベントフラグ (g_iic_Event[]) がセットされている場合、以下の処理を行います。

アドバンス関数カウンタ (g_iic_ReplyCnt[]) を初期化します。

通信を進める処理を行います。

正常に処理された場合、全通信が終了したか確認します。全通信終了時、チャンネル状態フラグに“R_IIC_FINISH”を設定します。

- ・イベントフラグ (g_iic_Event[]) がセットされていない場合、以下の処理を行います。

アドバンス関数カウンタ (g_iic_ReplyCnt[]) を減算します。

アドバンス関数カウンタが 0 の場合、リターン値

に、“R_IIC_ERR_NON_REPLY”を設定します。

- ・チャンネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスカブル割り込み以外の割り込みを禁止しています。

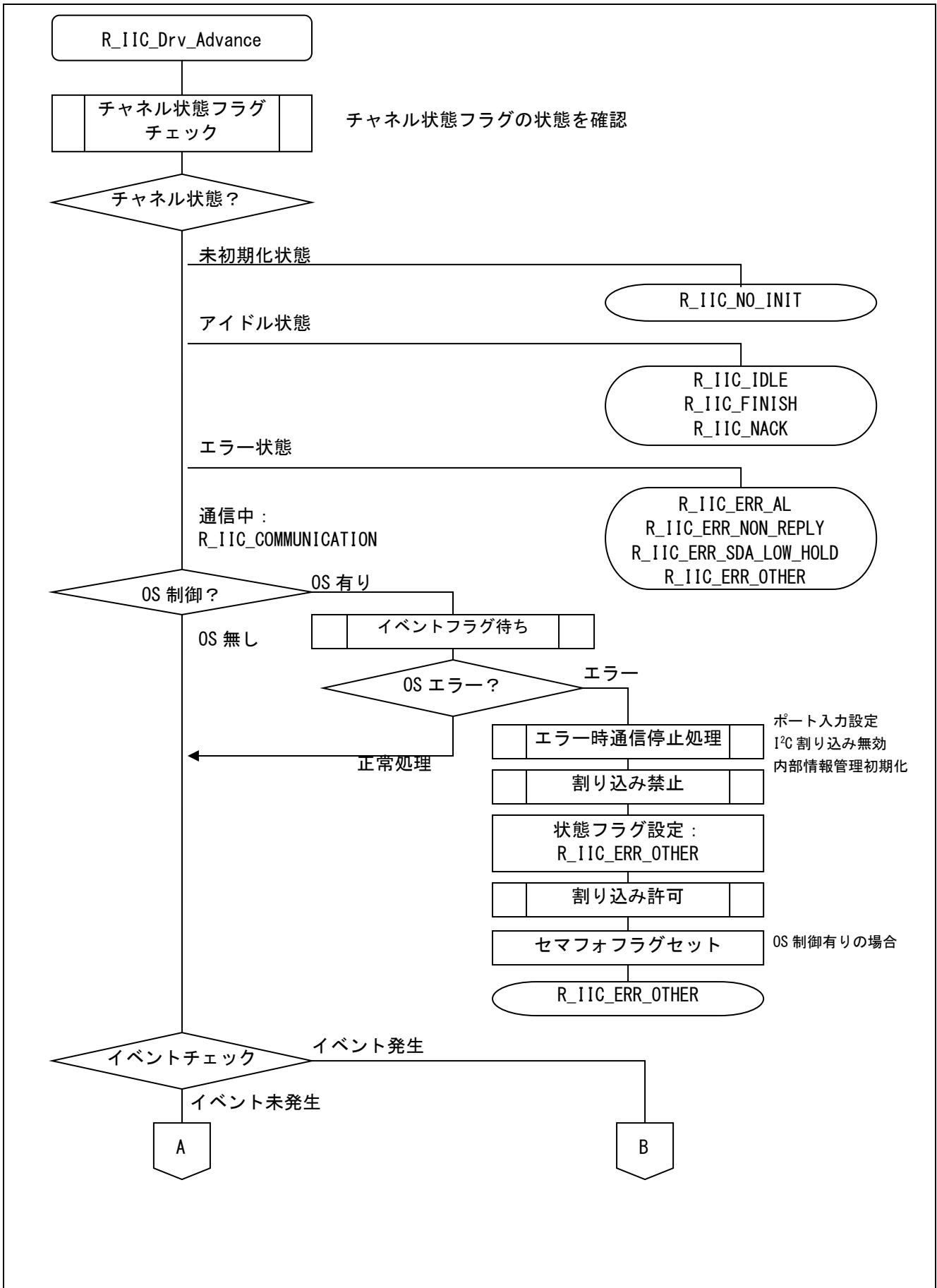


図 6-32 アドバンス関数概略フロー (1/3)

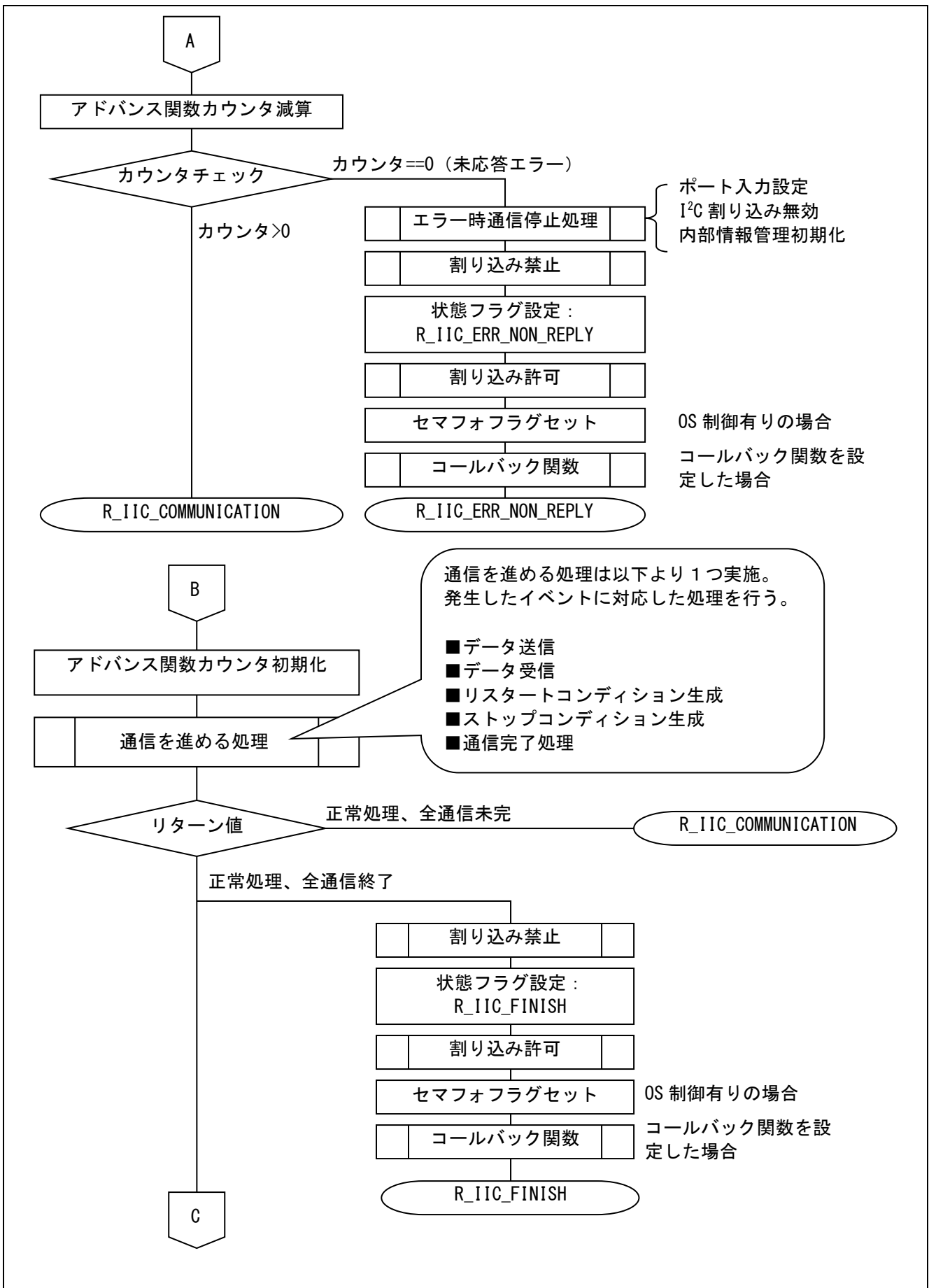


図 6-33 アドバンス関数概略フロー (2/3)

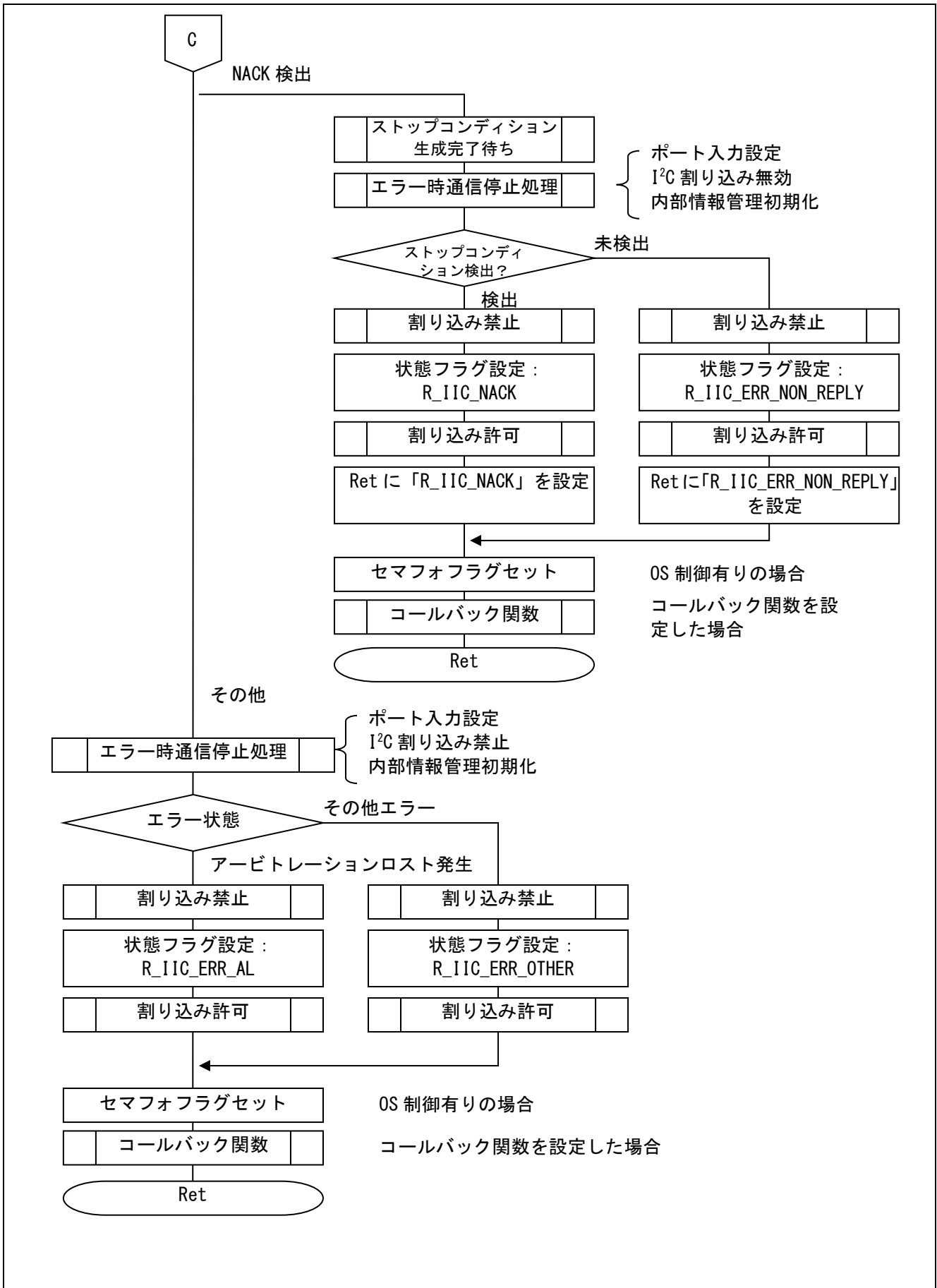


図 6-34 アドバンス関数概略フロー (3/3)

6.18.7 SCL 疑似クロック生成関数

R_IIC_Drv_GenClk	
概要	SCL 疑似クロック生成関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_GenClk (r_iic_drv_info_t *pRlic_Info, uint8_t ClkCnt)
説明	<ul style="list-style-type: none"> ・ SCL 疑似クロック生成を行います。ノイズ等の影響によりマスタとスレーブ間で同期ズレが発生し、SDA が Low ホールドした場合に、スレーブデバイス内部の状態を正常終了させることができます。 ・ 通常時は使用しないでください。正常な通信動作中に使用すると通信異常の原因になります。 ・ 本処理を行うためには、以下の設定が必要です。 構造体 “r_iic_drv_info_t” メンバの ChNo ;使用チャネル番号 クロック回数 “ClkCnt” ;01h~FFh クロック High/Low 幅 ;※ 1 参照
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ uint8_t ClkCnt ;SCL クロック回数
リターン値	R_IIC_NO_INIT SDA=High になり、スレーブデバイス内部の動作は正常終了し、未初期化状態になりました。チャネル状態フラグとデバイス状態フラグに “R_IIC_NO_INIT” を設定しました。 →通信を再開するには、以下の手順で処理を行ってください。 ①初期化関数コール ②マスタ送信のパターン 4 コール ③アドバンス関数をコールして通信を終了させる。
	R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。
	R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。
	R_IIC_ERR_SDA_LOW_HOLD SCL 疑似クロック生成を行いましたが、SDA は Low ホールドから復帰できていない状態です。チャネル状態フラグとデバイス状態フラグに “R_IIC_ERR_SDA_LOW_HOLD” を設定しました。 →スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。
備考	<ul style="list-style-type: none"> ・ SDA を Hi-z 状態にした時、SDA=Low であれば、バスが解放されていないと判断します。 ・ SDA=Low の場合、SCL 端子をポート出力に切り替え、SDA=High になるまで Low->High クロックを入力します。 ・ 設定回数クロックを生成しても SDA=Low の場合、エラーを返します。 ・ 通信単位が 9 クロックの場合が多いため、クロック回数は 9 回以上設定することを推奨します。 ・ チャネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスクブル割り込み以外の割り込みを禁止しています。

※ 1 : 疑似クロック出力の転送レートについて

疑似クロック出力の High/Low 幅は I²C バス仕様書で規定されている min 値以上確保する必要があります。(下表参照)

High/Low 幅の設定待ちは、ソフトウェアループにより実現しています。したがって、使用するシステム・クロックにより、待ち時間が変化します。

High/Low 幅の設定待ちは、マクロ定義の "SCL_L_WAIT" と "SCL_H_WAIT" で設定します。使用するシステム・クロックに合わせて、I²C バス仕様書の規格を満たす値をユーザ側で管理してください。(マクロ定義は 6.12.1 の表 6-14 を参照)

表 I²C バス仕様書で規定されている High/Low 幅の min 値

	ファーストモード	標準モード
Low 幅 (tLow)	1.3 μ s	4.7 μ s
High 幅 (tHigh)	0.6 μ s	4.0 μ s

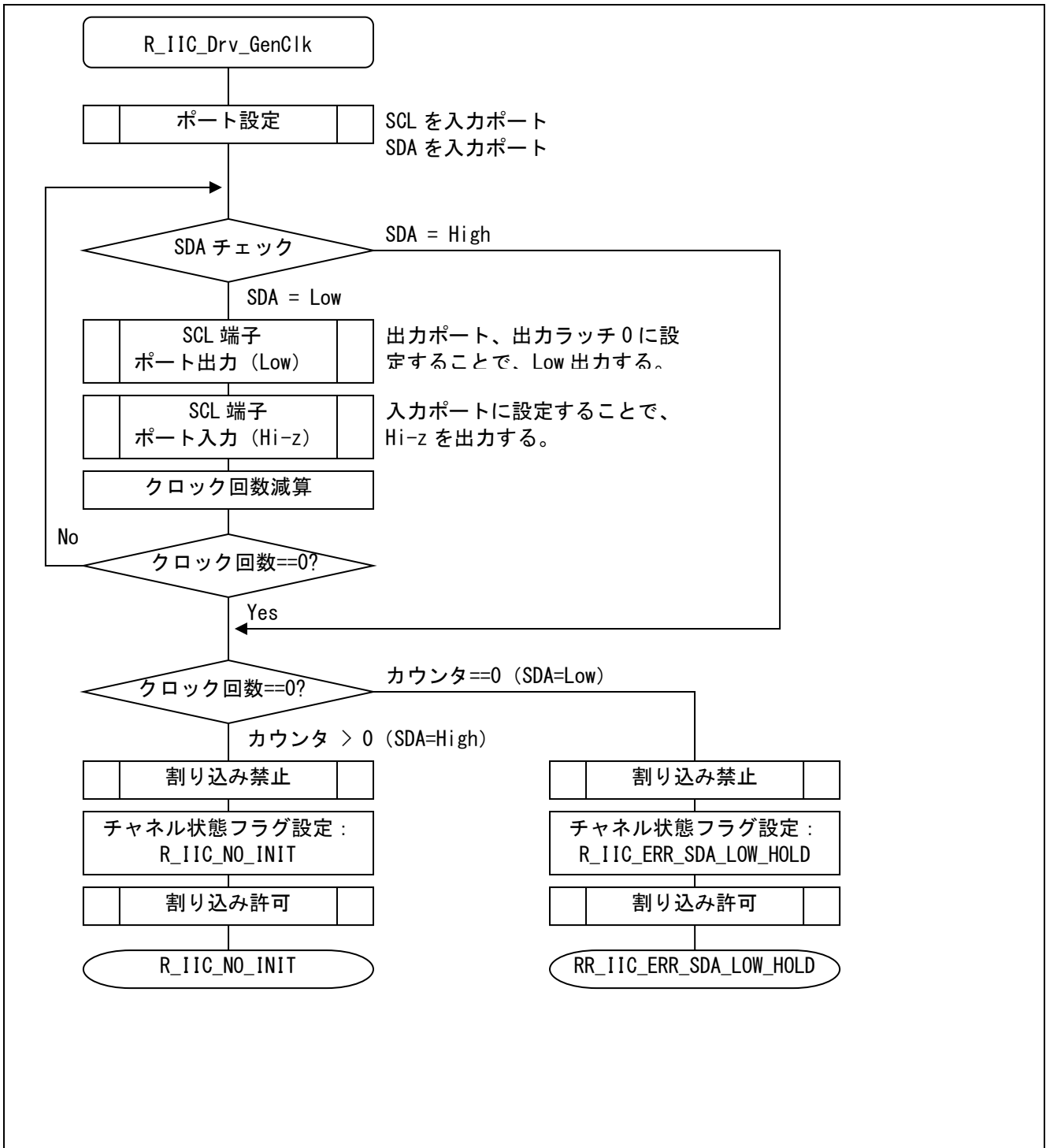


図 6-35 SCL 疑似クロック生成関数概略フロー

6.18.8 I²C ドライバリセット関数

R_IIC_Drv_Reset	
概要	I ² C ドライバリセット関数
ヘッダ	r_iic_drv_api.h, r_iic_drv_sub.h, r_iic_drv_int.h, r_iic_drv_sfr.h, r_iic_drv_os.h
宣言	error_t R_IIC_Drv_Reset(r_iic_drv_info_t *pRlic_Info)
説明	<ul style="list-style-type: none"> 対象チャンネルの I²C ドライバをリセットします。 IICE=0 により IICA を停止し、IICA 関連のレジスタをクリアします (※1)。 通信中に本関数をコールした場合でも、強制的に通信を停止します。 本処理を行うためには、以下の設定が必要です。 構造体 “r_iic_drv_info_t” メンバの ChNo ;使用チャンネル番号 r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
引数	r_iic_drv_info_t *pRlic_Info ;I ² C 通信情報構造体ポインタ
リターン値	R_IIC_NO_INIT 内部リセットを行い、未初期化状態になりました。チャンネル状態フラグとデバイス状態フラグに “R_IIC_NO_INIT” を設定しました。 →通信を再開するには、以下の手順で処理を行ってください。 ①初期化関数コール ②マスタ送信のパターン 4 コール ③アドバンス関数をコールして通信を終了させる。 R_IIC_LOCK_FUNC 他の API 処理が行われているため、処理は行われませんでした。チャンネル状態フラグとデバイス状態フラグは変更されません。 →他の API 処理終了後、本関数をコールしてください。
備考	R_IIC_ERR_PARAM パラメータエラーです。引数が設定されていません。チャンネル状態フラグとデバイス状態フラグは変更されません。 →引数を設定してください。 <ul style="list-style-type: none"> 再度通信を開始するには、I²C ドライバ初期化関数をコールする必要があります。 通信中に強制的に IICA を停止した場合、その通信は保証しません。 チャンネル状態フラグの設定中は、他の割り込みからの影響を回避するため、ノンマスカブル割り込み以外の割り込みを禁止しています。 ※1 : リセットされるのは、IICA ステータス・レジスタ (IICA) , IICA フラグ・レジスタ (IICF) の STCF, IICBSY ビット, IICA コントロール・レジスタ 1 (IICCTL1) レジスタの CLD, DAD ビットです。

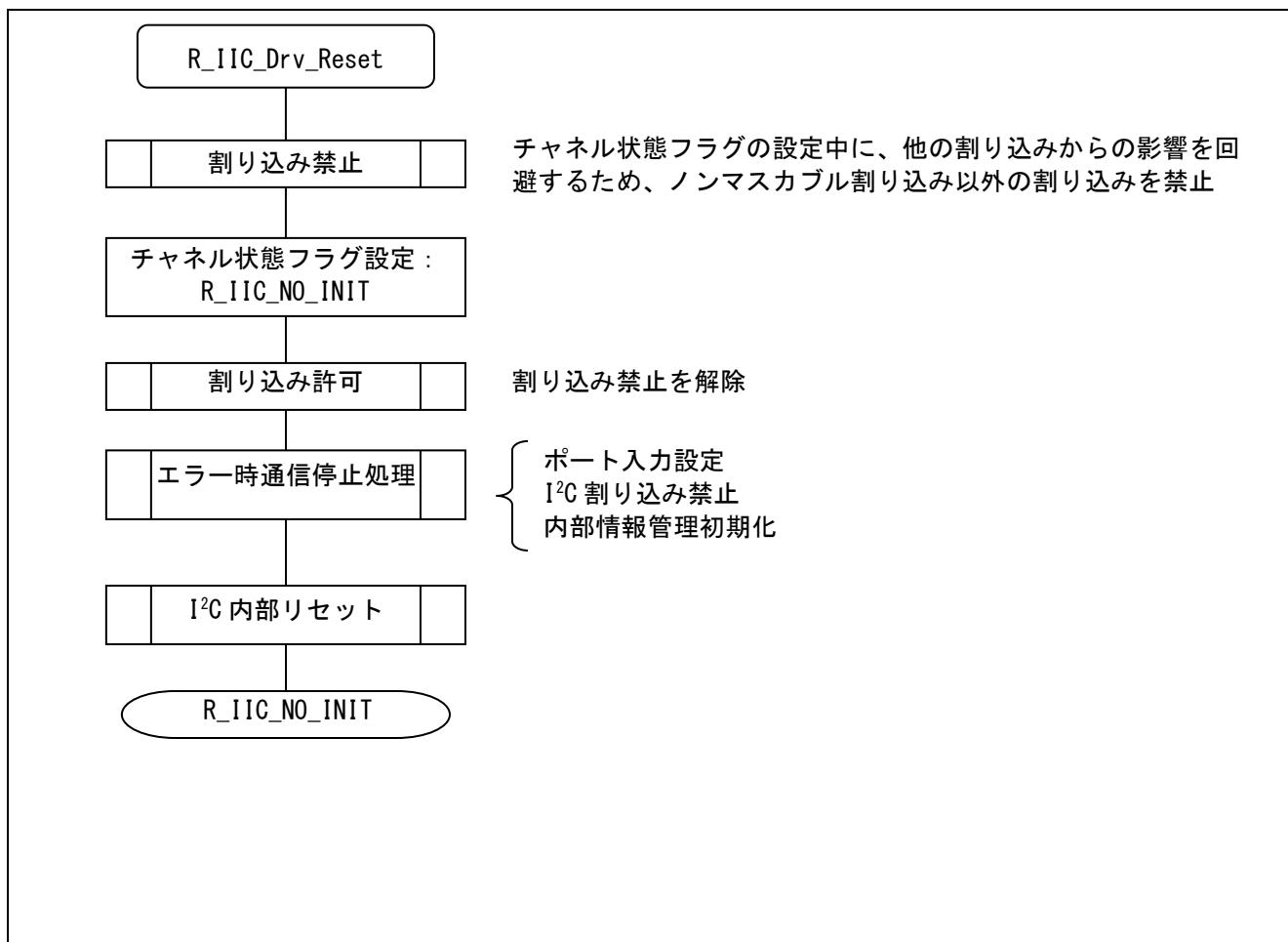


図 6-36 I²C ドライバリセット関数概略フロー

7. 応用例

7.1 r_iic_drv_api.h

使用する上での設定例を以下に示します。

設定箇所は、各ファイル中の「**** SET ****」というコメントの部分です。

(1) 使用する IICA チャンネルの選択

使用する I²C チャンネルを指定してください。使用しないチャンネルをコメントアウトすることで、使用 ROM 容量を縮小することができます。

下記の例は、チャンネル 0 と 1 を使用する場合があります。

```
/*-----*/
/*  Select channels to enable.                               */
/*-----*/
#define IICA0_ENABLE
#define IICA1_ENABLE
```

(2) 使用する最大チャンネル番号の定義

「使用する最大チャンネル番号 + 1」の値を設定してください。

下記の例は、チャンネル 0 とチャンネル 1 を使用する場合があります。この場合、使用する最大チャンネル番号は”1”であり、設定値は”2”になります。

```
/*-----*/
/*  Define channel No.(max) + 1.                             */
/*-----*/
#define MAX_IIC_CH_NUM          (uint8_t) (2)
```


(3) 転送クロックの定義

IICA ロー・レベル幅設定レジスタ x (IICWLx) と IICA ハイ・レベル幅設定レジスタ x (IICWHx) の設定値を定義してください (x: チャンネル番号)。使用するチャンネル毎に設定する必要があります。詳しい設定方法については、RL78 ファミリー MCU ユーザーズマニュアル ハードウェア編をご参照ください。

最大 400kHz までの設定可能です。ただし、標準モード・デバイスは、ファーストモード・デバイスが混載されるバスでは、標準モードの最大 100kHz に設定する必要があります。また、SDA, SCL 信号の立ち上がり時間 (tR) と立ち下がり時間 (tF) は、プルアップ抵抗と配線容量によって異なるため、必要に応じて設定値を見直してください。

下記の例は、以下の条件での設定値です。

- ・システム・クロック : 32MHz
- ・転送クロック : 400kHz
- ・SDA, SCL 信号の立ち上がり時間 (tR) 、立ち下がり時間 (tF) : 300ns

```

/*-----*/
/* Define frequency as iic channel. (Please add a channel as needed.) */
/*-----*/
/* Freq = 400KHz at main system clock = 32MHz */
#define R_IIC_CH0_LCLK (uint8_t)(20) /* Channel 0 IICWL0 register setting */
#define R_IIC_CH0_HCLK (uint8_t)(18) /* Channel 0 IICWH0 register setting */
#define R_IIC_CH1_LCLK (uint8_t)(20) /* Channel 1 IICWL1 register setting */
#define R_IIC_CH1_HCLK (uint8_t)(18) /* Channel 1 IICWL1 register setting */

```

(4) 各カウンタの定義

ソフトウェアループ用のカウンタです。したがって使用するシステム・クロックによりループ時間が変化します。必要に応じて設定値を見直してください。

```

/*-----*/
/* Define counter. */
/*-----*/
#define REPLY_CNT (uint32_t)(100000) /* Counter of non-reply errors */
#define START_COND_WAIT (uint16_t)(100)
/* Counter of waiting start condition generation */
#define STOP_COND_WAIT (uint16_t)(100)
/* Counter of waiting stop condition generation */
#define BUSCHK_CNT (uint16_t)(100) /* Counter of checking bus busy */
#define SDACHK_CNT (uint16_t)(100) /* Counter of checking SDA level */
#define SCLCHK_CNT (uint16_t)(100) /* Counter of checking SCL level */
#define SCL_L_WAIT (uint16_t)(100)
/* Counter of waiting SCL Low clock setting */
#define SCL_H_WAIT (uint16_t)(100)
/* Counter of waiting SCL High clock setting */

```

7.2 r_iic_drv_sfr.h

r_iic_drv_sfr.h.XXX は、MCU 毎に作成したものです。どれか一つを r_iic_drv_sfr.h にリネームして使用してください。対象 MCU のものが無い場合には、参照して、r_iic_drv_sfr.h を作成してください。

使用する上での設定例を以下に示します。

設定箇所は、各ファイル中の「**** SET ****」というコメントの部分です。

(1) 使用する端子の定義

使用する端子のポート番号を定義してください。使用するチャンネル毎に設定する必要があります。

下記の例は、チャンネル 0 を使用する場合があります。

```

/*-----*/
/* Define channel register. */
/*-----*/
#ifdef IICA0_ENABLE

/* Define port. */
#define R_IIC_PM_SCL0    PM6.0    /* SCL0 Port mode registers */
#define R_IIC_PM_SDA0    PM6.1    /* SDA0 Port mode registers */
#define R_IIC_P_SCL0     P6.0     /* SCL0 Port registers */
#define R_IIC_P_SDA0     P6.1     /* SDA0 Port registers */

```

(2) 入力クロック供給の制御ビット (IICAxEN) の定義

シリアル・インタフェース IICA のクロック供給の制御ビット (IICAxEN) を定義してください。

下記の例は、チャンネル 0 を使用する場合があります。

```

/* Define peripheral enable register */
#define R_IIC_IICA0EN    IICA0EN /* IICA0 peripheral enable register */

```

(3) IICA コントロール・レジスタ x1 (IICCTLx1) の定義

使用する最大転送レートにより設定を変更してください。最大転送レートが 100kHz と 400kHz では設定値が異なります。

下記の例は、使用する最大転送レートが 400kHz の場合です。

```
/*-----*/
/* Define register setting. */
/*-----*/
#define R_IIC_IICCTL1_INIT (uint8_t) (0x0D) /* SMC=1 (400KHz), DFC=1, PRS=1 */
```

下記の例は、使用する最大転送レートが 100kHz の場合です。

```
/*-----*/
/* Define register setting. */
/*-----*/
#define R_IIC_IICCTL1_INIT (uint8_t) (0x01) /* SMC=0 (100KHz), DFC=0, PRS=1 */
```

(4) IICA 割り込み優先順位の定義

使用するチャンネルの割り込み優先順位を定義してください。

下記の例は、チャンネル 0 を使用し、優先順位レベル 2 を設定する場合です。

```
/* Interrupt register setting */
#ifdef IICA0_ENABLE
#define R_IIC_CH0_PR0_INIT (uint8_t) (0x00) /* Channel 0 */
#define R_IIC_CH0_PR1_INIT (uint8_t) (0x01) /* Priority level 2 */
/* Priority level 2 */
```

7.3 r_iic_drv_int.c

使用する上での設定例を以下に示します。

設定箇所は、各ファイル中の「/** SET **/」というコメントの部分です。

7.3.1 統合開発環境 CS+ for CA,CX (旧 CubeSuite+) の場合

(1) 割り込み関数 #pragma interrupt の定義

使用する I²C チャンネルの割り込み関数 #pragma interrupt を定義してください。使用しないチャンネルはコメントアウトしてください。チャンネルが存在しない MCU に対して定義を行うと、コンパイルエラーが発生します。

下記の例は、チャンネル 0 を使用し、チャンネル 1 を使用しない場合です。

```
#pragma interrupt INTIICA0 r_iic_drv_intiica0_isr
/* Uses RL78 channel 0 interrupt IICA. */
/* #pragma interrupt INTIICA1 r_iic_drv_intiica1_isr */
/* Uses RL78 channel 1 interrupt IICA. */
```

7.3.2 統合開発環境 CS+ for CC の場合

(1) 割り込み関数 #pragma interrupt の定義

使用する I²C チャンネルの割り込み関数 #pragma interrupt を定義してください。使用しないチャンネルはコメントアウトしてください。チャンネルが存在しない MCU に対して定義を行うと、コンパイルエラーが発生します。

下記の例は、チャンネル 0 を使用し、チャンネル 1 を使用しない場合です。

```
#pragma interrupt r_iic_drv_intiica0_isr(vect=INTIICA0)
/* Uses RL78 channel 0 interrupt IICA. */ /** SET **/
/* #pragma interrupt r_iic_drv_intiica1_isr(vect=INTIICA1) */
/* Uses RL78 channel 1 interrupt IICA. */ /** SET **/
```

7.3.3 統合開発環境 IAR Embedded Workbench の場合

(1) SFR 領域用デファインの設定

使用する MCU が RL78 であり、IAR Systems 製の統合開発環境を使用する場合には、使用する MCU の SFR が定義されているヘッダファイルを設定する必要があります。

MCU 個別のクロック同期式シングルマスタ制御ソフトウェアも参照してください。

本設定は、SPI スレーブデバイスセレクト制御信号用に使用します。

表 7-1 MCU と SFR 領域用デファインの設定

統合開発環境	MCU	SFR 設定の要・不要	設定方法
CubeSuite+ CS+	RL78	不要	不要
IAR Embedded Workbench	RL78	要	#ifdef __ICCRL78__ #include <ior5f104pj.h> ←MCU に合わせて変更 #include <ior5f104pj_ext.h> ←MCU に合わせて変更 #endif

以下は、RL78/G14 100pin を使用する場合の例です。

```
#ifndef __ICCRL78__ /* IAR RL78 Compiler */
#include <ior5f104pj.h> /* for RL78/G14 100pin (R5F104PJ) */
#include <ior5f104pj_ext.h> /* for RL78/G14 100pin (R5F104PJ) */
#endif /* __ICCRL78__ */
```

(2) 割り込みハンドラ処理の追加／削除

使用する RL78 ファミリ MCU の IICA チャンネル数に応じて、割り込みハンドラ処理を追加／削除してください。

本サンプルコードでは割り込みハンドラ処理を 2 チャンネル定義しています（関数名：r_iic_drv_intiica0_isr(), r_iic_drv_intiica1_isr()）。使用する RL78 ファミリ MCU の IICA のチャンネル数が 1 チャンネルの場合には、r_iic_drv_intiica1_isr() をコメントアウトしてください。IICA のチャンネル数が 3 チャンネル以上の場合には、r_iic_drv_intiica0_isr() を参照し、割り込みハンドラ処理を追加してください。

下記の例は、チャンネル 0 を使用し、チャンネル 1 を使用しない場合です。

```
/* #pragma vector=INTIICA1_vect */ /* SET */
__interrupt __root void r_iic_drv_intiica1_isr(void)
```

表 7-2 に RL78 ファミリ MCU の IICA チャンネル数を示します。表 7-2 以外の MCU を使用する場合は、ユーザズマニュアル ハードウェア編を参照し、IICA のチャンネル数を確認し、ご対応ください。

表 7-2 RL78 ファミリ MCU IICA チャンネル数一覧

RL78 ファミリ MCU	ピン数	IICA ch 数	設定方法
RL78/G14	30, 32, 36, 40, 44, 48, 52, 56, 64 ピン	1ch	r_iic_drv_intiica1_isr() の削除が必要
	80, 100 ピン	2ch	追加／削除不要 サンプルコードをそのまま使用可能
RL78/G1C	全ピン	1ch	r_iic_drv_intiica1_isr() の削除が必要
RL78/L12	全ピン	1ch	r_iic_drv_intiica1_isr() の削除が必要
RL78/L13	全ピン	1ch	r_iic_drv_intiica1_isr() の削除が必要
RL78/L1C	全ピン	1ch	r_iic_drv_intiica1_isr() の削除が必要

7.4 r_iic_drv_sfr.c

使用する上での設定例を以下に示します。

設定箇所は、各ファイル中の「/** SET **/」というコメントの部分です。

(1) SFR 領域用デファインの設定

使用する MCU が RL78 であり、IAR 製の統合開発環境を使用する場合には、使用する MCU の SFR が定義されているヘッダファイルを設定する必要があります。

MCU 個別のクロック同期式シングルマスタ制御ソフトウェアも参照してください。

本設定は、SPI スレーブデバイスセレクト制御信号用に使用します。

表 7-3 MCU と SFR 領域用デファインの設定

統合開発環境	MCU	SFR 設定の要・不要	設定方法
CubeSuite+ CS+	RL78	不要	不要
IAR Embedded Workbench	RL78	要	<pre> #ifdef __ICCRL78__ #include <ior5f104pj.h> ←MCU に合わせて変更 #include <ior5f104pj_ext.h> ←MCU に合わせて変更 #endif </pre>

以下は、RL78/G14 100pin を使用する場合の例です。

```

#ifdef __ICCRL78__
#include <ior5f104pj.h>
#include <ior5f104pj_ext.h>
#endif
/* __ICCRL78__ */
                    
```

7.5 r_iic_drv_os.c

OS 制御させる場合、修正してください。

7.6 復帰処理例

SDA または SCL が Low ホールドした場合の通信への復帰処理を説明します。以下の手順で処理を行ってください。図 7-1 に SCL 疑似クロック生成による復帰処理例を示します。

尚、本復帰処理後は、アイドル状態になります。開始関数をコールすることで通信が可能です。

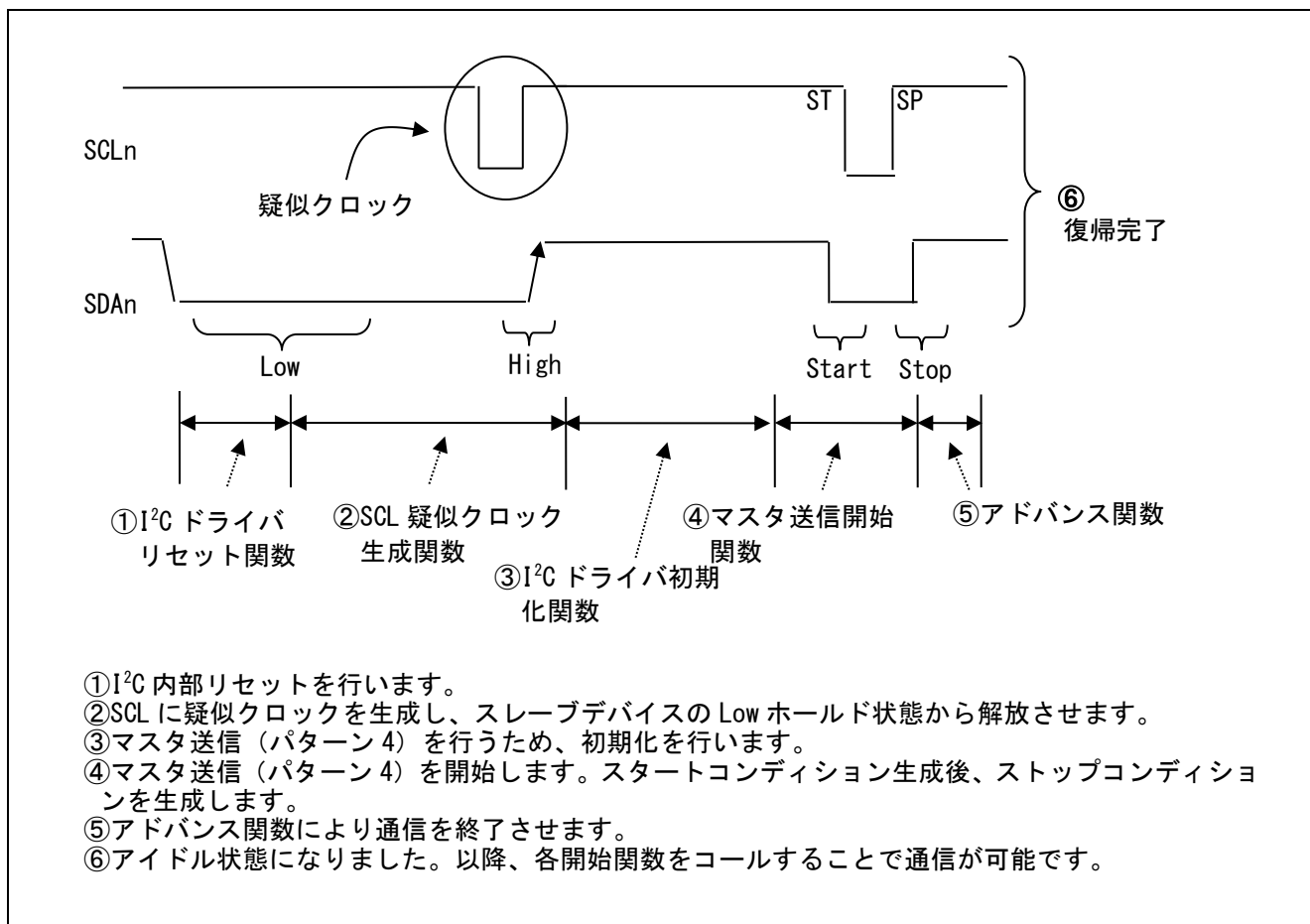


図 7-1 SCL 疑似クロック生成による復帰処理例

① I²C ドライバリセット関数 R_IIC_Drv_Reset()

本処理は、内部リセットを行うことで、SDA 及び SCL のホールドを解除します。

本処理後に、SDA 及び SCL が High であることを確認してください。リセット後も SDA または SCL が Low ホールドしている場合、スレーブデバイスが Low ホールドしているか、マスタデバイスから Low 信号が出力されているなどの可能性があります。

② SCL 疑似クロック生成関数 R_IIC_Drv_GenClk()

本処理は、スレーブデバイスにより SDA を Low ホールドされた場合に、SCL に疑似クロックを生成することで、スレーブデバイスの内部処理を終了させて SDA を High にします。

リターン値が”R_IIC_SDA_HIGH”であれば SDA=High となり解除されました。”R_IIC_ERR_SDA_LOW_HOLD”なら SDA Low ホールドから復帰できていません。システムの状態を見直してください。

③ I²C 初期化 R_IIC_Drv_Init()

上記処理で、SDA 及び SCL の Low ホールドを解除した場合、I²C ドライバ初期化関数をコールしてください。

④バス解放処理 **R_IIC_Drv_MasterTx()** パターン 4

ストップコンディション生成により、バスを解放します。本サンプルコードでは、マスタ送信パターン 4 により『スタートコンディション生成→ストップコンディション生成』を行うことで実現します。

パターン 4 の設定を行い、マスタ送信開始関数をコールしてください。

⑤アドバンス関数 **R_IIC_Drv_Advance()**

④項で開始した処理をアドバンス関数コールにより終了させてください。

正常に終了するとアイドル状態になります。開始関数をコールすることで通信が可能です。

8. 使用上の注意事項

8.1 組み込み時の注意事項

本サンプルコードを組み込む場合は、以下をインクルードしてください。

- r_iic_drv_api.h
- r_iic_drv_sub.h
- r_iic_drv_sfr.h
- r_iic_drv_int.h
- r_iic_drv_os.h

8.2 初期化時の注意事項

対象チャンネルを始めて初期化する場合、チャンネル状態フラグ `g_iic_ChStatus[]` とデバイス状態フラグ `*(pRIic_Info.pDevStatus)` に “R_IIC_NO_INIT” を設定してください。以降、このフラグは本サンプルコードで管理するため、設定は禁止です。

8.3 チャンネル状態フラグとデバイス状態フラグについて

本サンプルコードでは、チャンネル状態フラグとデバイス状態フラグにより、通信状態の整合性を保っています。最初の初期化以外で変更された場合、その後の通信は保証しませんのでご注意ください。

8.4 同一チャンネル上の複数スレーブデバイス制御時のアドバンス関数のコール方法

同一チャンネル上の複数のスレーブデバイスを制御するためには、以下の手順で処理を行ってください。

下記①項の処理により、非通信状態のデバイスに対して通信が行われることを防止できます。

- ①アドバンス関数コール対象のスレーブデバイスの I²C 通信情報構造体のデバイス状態フラグが “R_IIC_COMMUNICATION”（通信中）であることを確認する。
- ②アドバンス関数をコールする。
- ③通信が終了するまで①～②を繰り返す。
- ④通信終了。以降、開始関数をコールすることで通信が可能です。

アドバンス関数は、スレーブデバイスを識別せず、通信中のスレーブデバイスの処理を進めるものです。したがって、デバイス状態フラグが “R_IIC_COMMUNICATION”（通信中）でない場合であっても、アドバンス関数コールにより通信中のスレーブデバイスの処理が進みます。

8.5 アドバンス関数を割り込み内で処理する場合と OS 制御について

アドバンス関数を割り込み内で処理する場合と OS 制御（※）については、未検証です。使用される場合は十分に評価を行い、必要に応じて修正してください。

※：本サンプルコードの OS 制御は、 μ ITRON4.0 を想定しています。

8.6 転送速度設定

対象チャンネル毎に設定する必要があります。最大 400kHz までの設定可能です。

ただし、標準モード・デバイスとファーストモード・デバイスが混載されるバスでは、標準モードの最大 100kHz に設定する必要があります。転送レートは、表 6-14 に定義する R_IIC_CHx_LCLK/ R_IIC_CHx_LCLK (x はチャンネル番号) で設定してください。

8.7 #define 定義「IICAx_ENABLE」と「MAX_IIC_CH_NUM」の設定について

例として、チャンネル 2 のみを使用する場合の設定方法を示します。

#define 定義「IICAx_ENABLE」は「IICA2_ENABLE」のみ有効にします。これにより、チャンネル 0 とチャンネル 1 のソースコードをマスクします。

```
/*-----*/
/*  Select channels to enable.                                */
/*-----*/
/* #define IICA0_ENABLE */
/* #define IICA1_ENABLE */
#define IICA2_ENABLE
```

#define 定義「MAX_IIC_CH_NUM」には“3”を設定します。使用するチャンネル数は“1”ですが、ここで設定する値は「使用する最大チャンネル番号+1」になりますので、ご注意ください。

```
/*-----*/
/*  Define channel No.(max) + 1.                            */
/*-----*/
#define MAX_IIC_CH_NUM      (uint8_t) (3)
```

8.8 割り込み関数 #pragma interrupt の定義について

使用する I²C チャンネルの割り込み関数 #pragma interrupt を定義してください。使用しないチャンネルはコメントアウトしてください。チャンネルが存在しない MCU に対して定義を行うと、コンパイルエラーが発生します。

8.9 ユーザ API コール時の注意事項

本アプリケーションノートのユーザ API はメイン処理上からのみコールしてください。割り込みハンドラから発行した場合、正常に動作しない可能性があります。

8.10 型宣言の重複によるワーニングについて

本ドライバでは stdint.h で宣言される intN_t、uintN_t (N はビット長) を定義しています。stdint.h をインクルードした場合や、既に型を定義している場合、コンパイル時にワーニングが発生する可能性があります。本ドライバにある型宣言が不要な場合には、定義を削除してください。

8.11 コンパイル時の注意事項

CC-RL コンパイラでコンパイルした場合、警告メッセージ「W0520111:文は実行されません。」が出力されます。

これは `break` 文の直前で `return` しているため `break` 文が実行されないという警告メッセージです。動作に影響がないため無視して問題ありません。

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/>

お問い合わせ先

<http://japan.renesas.com/contact/>

すべての商標および登録商標は、それぞれの所有者に帰属します。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2012.12.28	—	初版発行
1.01	2013.03.07	—	アプリケーション番号を R01AN1074JJ0101 に変更した。
		—	Rev.1.00 から Rev.1.01 に変更した。
		P6	表 2-1 総合開発環境欄の CubeSuite+の Ver.を V1.03.00 に変更した。
		P6	表 2-1 C コンパイラ欄の Ver.を V1.60 に変更した。
		P6	表 2-1 サンプルコードのバージョンを Ver.1.01 に変更した。
		P6	表 2-1 評価に使用したソフトウェア 誤記を訂正した。
		P6	3.関連アプリケーションノート 誤記を訂正した。
		P8	図 5-2 内容を変更した。
		P8	5.4 最大転送速度 内容を変更した。
		P16	6.5 スレーブデバイス制御の実現方法 内容を変更した。
		P19	図 6-12 内容を変更した。
		P20	図 6-13 内容を変更した。
		P21	図 6-14 内容を変更した。
		P22	表 6-4 誤記を訂正した。
		P27	表 6-6 ROM サイズと最大ユーザスタックサイズを変更した。
		P27	表 6-7 サンプルコードのフォルダとアプリケーションノートの番号を r01an1074jj0101_rl78g14 に変更した。
		P29	表 6-10 内容を変更した。
		P30	表 6-12 内容を変更した。
		P43	図 6-23 誤記を訂正した。
		P49	6.18.3 マスタ送信開始関数 備考 内容を変更した。
		P50	図 6-26 文章を追加した。
		P53	6.18.4 マスタ受信開始関数 備考 内容を変更した。
		P54	図 6-28 文章を追加した。
		P57	6.18.5 マスタ複合開始関数 備考 内容を変更した。
		P58	図 6-30 文章を追加した。
		P61	6.18.6 アドバンス関数 備考 内容を変更した。
		P65	6.18.7 SCL 疑似クロック生成関数 備考 内容を変更した。
		P73	7.3 項を新規追加した。
		P76	8.4 同一チャネル上の複数スレーブデバイス制御時のアドバンス関数のコール方法 内容を変更した。
		P77	8.8 項、8.9 項を新規追加した。
A-1	製品ご使用上の注意事項 を改定した。		
—	ご注意書き を改定した。		
1.02	2014.2.1	6	2. 動作確認条件 に、見出しとして(1) RL78/G14 SAU 統合開発環境 CubeSuite+の場合 を追加した。 統合環境 元は総合環境であった。
		7	2. 動作確認条件 に、(2) RL78/G14 SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		28	6.10 必要メモリサイズ に、見出しとして(1) RL78/G14 SAU 統合開発環境 CubeSuite+の場合 を追加した。 (2) RL78/G14 SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		29	6.11 ファイル構成 に、アプリケーションノートの更新に伴い、ファイル構成を更新した。

		42	6.18.1 関数共通処理 図 6-20 にて、関数開始直後に、パラメータチェック処理を追加した。
		43	6.18.1 関数共通処理 図 6-21 にて、関数開始直後に、パラメータチェック処理を追加した。
		44	6.18.1 関数共通処理 図 6-22 にて、関数開始直後に、パラメータチェック処理を追加した。
		45	6.18.1 関数共通処理 図 6-23 にて、関数開始直後に、パラメータチェック処理を追加した。
		48	6.18.2 I ² C ドライバ初期化関数 図 6-24 にて、関数開始直後のパラメータチェック処理を削除した。
		52	6.18.3 マスタ送信開始関数 図 6-26 にて、関数開始直後のパラメータチェック処理を削除した。
		56	6.18.4 マスタ受信開始関数 図 6-28 にて、関数開始直後のパラメータチェック処理を削除した。
		60	6.18.5 マスタ複合開始関数 図 6-30 にて、関数開始直後のパラメータチェック処理を削除した。
		64	6.18.6 アドバンス関数 図 6-32 にて、関数開始直後のパラメータチェック処理を削除した。
		69	6.18.7 SCL 疑似クロック生成関数 図 6-35 にて、関数開始直後のパラメータチェック処理を削除した。
		71	6.18.8 I ² C ドライバリセット関数 図 6-36 にて、関数開始直後のパラメータチェック処理を削除した。
		76	7.4 r_iic_drv_sfr.c 新規に追加した。
1.03	2014.9.30	—	RL78/G1C、RL78/L12、RL78/L13、RL78/L1C グループを追加した。 統合環境 元は総合環境であった。
		—	複数グループの MCU をサポートしたため、記載上、「RL78/G14」を「RL78 ファミリ MCU」に変更した。
		1	要旨 に最新のスレーブデバイス制御ソフトウェアの組み合わせ情報 URL を追加した。
		6	2. 動作確認条件 に、(1) RL78/G14 IICA 統合開発環境 CubeSuite+の場合と(2) RL78/G14 IICA 統合開発環境 IAR embedded Workbench の場合 の「RL78/G14 IICA」 元は「RL78/G14 SAU」であった。
		7	2. 動作確認条件 に、(3) RL78/G1C IICA 統合開発環境 CubeSuite+の場合と(4) RL78/G1C SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		8	2. 動作確認条件 に、(5) RL78/L12 IICA 統合開発環境 CubeSuite+の場合と(6) RL78/L12 SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		9	2. 動作確認条件 に、(7) RL78/L13 IICA 統合開発環境 CubeSuite+の場合と(8) RL78/L13 SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		10	2. 動作確認条件 に、(9) RL78/L1C IICA 統合開発環境 CubeSuite+の場合と(10) RL78/L1C SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		13	6.3 ソフトウェア動作 (2) 通常制御 (OS 有り) にて、「本制御の処理は、組み込まれていません。使用される場合、必要に応じて修正してください。」 元は「本制御は動作未確認のため、使用される場合、十分に評価して必要に応じて修正してください。」であった。
		33	6.10 必要メモリサイズ に、(3) RL78/L13 IICA 統合開発環境 CubeSuite+の場合 を追加した。

			(4) RL78/L13 SAU 統合開発環境 IAR embedded Workbench の場合 を追加した。
		34	6.11 ファイル構成 に、アプリケーションノートの更新に伴い、ファイル構成を更新した。
		46	6.18.1 関数共通処理 に文章を追加した。
		46	6.18.1 関数共通処理 に文章を追加した。
		81-82	7.3 r_iic_drv_int.c に 7.3.1 統合開発環境 CubSuite+の場合、7.3.2 統合開発環境 IAR Embedded Workbench の場合を追加した。
		83	7.5 r_iic_drv_os.c を追加した。
1.04	2016.3.31	6	2. 動作確認条件 (1) RL78/G14 IICA 統合開発環境 CS+ for CA,CX の場合（コンパイラ：CA78K0R） で情報を更新した。 (2) RL78/G14 IICA 統合開発環境 CS+ for CC の場合（コンパイラ：CC-RL） を追加した。
		33	6.10 必要メモリサイズ (1) RL78/G14 IICA 統合開発環境 CS+ for CA,CX の場合（コンパイラ：CA78K0R） で情報を更新した。 (2) RL78/G14 IICA 統合開発環境 CS+ for CC の場合（コンパイラ：CC-RL） を追加した。
		36	6.11 ファイル構成 アプリケーションノート番号を更新した。
		38, 80	定義 SCLCHK_CNT を追加した。
		83	7.3 r_iic_drv_int.c 7.3.1 のタイトルを変更した。 7.3.2 を追加した。 7.3.3 (旧 7.3.2) - (2)を更新した。
		89-90	8.10 型宣言の重複によるワーニングについて を追加した。 8.11 コンパイル時の注意事項 を追加した。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理

【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

【注意】電源投入時は、製品の状態は不定です。

電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。

外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレス（予約領域）のアクセス禁止

【注意】リザーブアドレス（予約領域）のアクセスを禁止します。

アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

【注意】リセット時は、クロックが安定した後、リセットを解除してください。

プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後、切り替えてください。

リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

同じグループのマイコンでも型名が違っていると、内部ROM、レイアウトパターンの相違などにより、電氣的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 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・システムの設計において、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て、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。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りが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。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
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、産業用ロボット等
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、各種安全装置等
当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しており、使用することはできません。たとえ、意図し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損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
9. 本資料に記載されている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・システムに使用することはできません。また、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用用途に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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禁じます。

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

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



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 エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

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/contact/>