RX210、RX21A、RX220 グループ

RIIC を用いたマルチマスタ I2C-bus

R01AN1348JJ0101
Rev.1.01
2014.07.01

要旨

本アプリケーションノートでは、RX210、RX21A、RX220 グループの I^2C バスインタフェース(以下、RIIC)を使用して、マルチマスタで通信する方法について説明します。

対象デバイス

RX210、RX21A、RX220 グループ

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あわせて変更し、十分評価してください。

目次

1. 仕様

RIIC を使用して、I²C バスでマルチマスタ通信を行います。

リセット解除後、マスタ送信とマスタ受信を 1 度だけ行います。マスタ送信で 00h～09h の 10 バイトを送信し、その後、マスタ受信で 10 バイトを受信します。

マスタ送信中またはマスタ受信中にアービトレーションロストを検出した場合は、他のマスタデバイスの通信を優先し、スレーブ動作を行います。

- 転送速度　　: 100kbps
- アドレスフォーマット　　: 7 ビットアドレスフォーマット
- マスタ/スレーブ動作　　: マスタ送信、マスタ受信、スレーブ送信、スレーブ受信

I²C バスの通信フォーマットについては「RX210 グループ ユーザーズマニュアル ハードウェア編」、および I²C バスの規格書を参照してください。

表 1.1に使用する周辺機能と用途を、図 1.1に動作概要を示します。

表1.1　使用する周辺機能と用途

周辺機能	用途
RIIC	マルチマスタ I²C バス

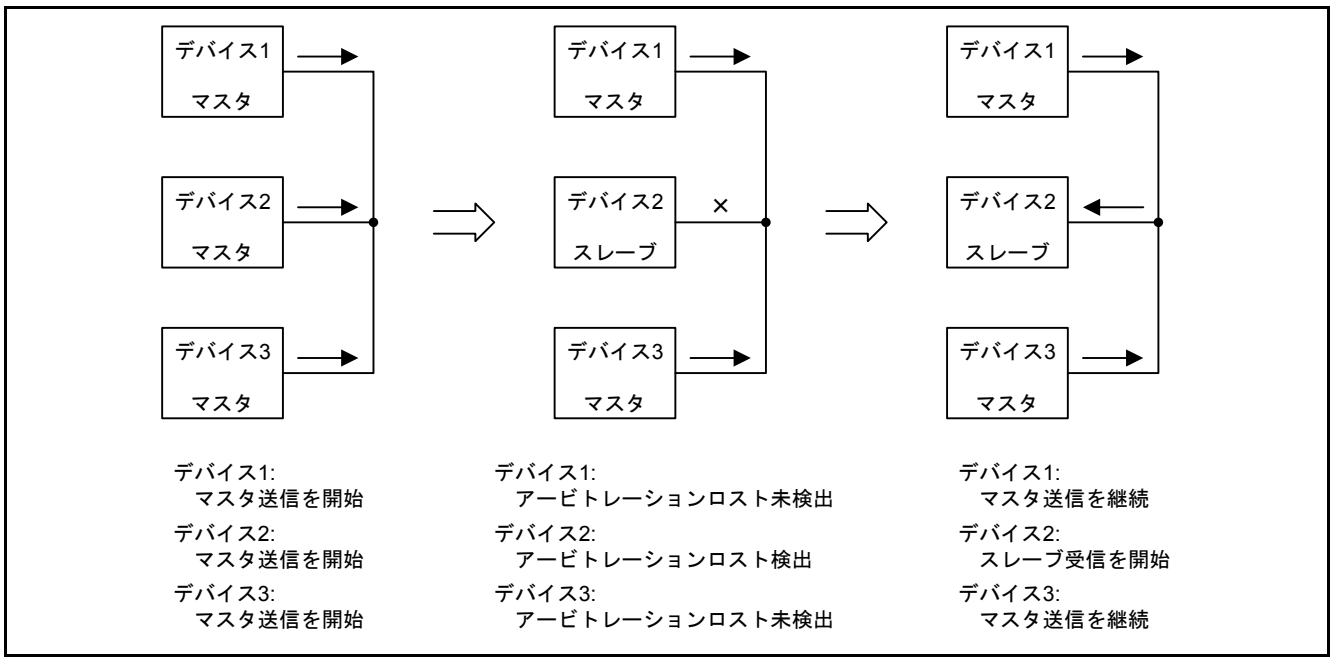


図1.1　動作概要

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、下記の条件で動作を確認しています。

表2.1　動作確認条件

項目	内容
使用マイコン	R5F52108ADFP (RX210 グループ)
動作周波数	• メインクロック: 20MHz • PLL: 100MHz (メインクロック 2 分周 10 逓倍) • システムクロック(ICLK): 50MHz (PLL 2 分周) • 周辺モジュールクロック B (PCLKB): 25MHz (PLL 4 分周)
動作電圧	5.0V
統合開発環境	ルネサスエレクトロニクス製 High-performance Embedded Workshop Version 4.09.01
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V.1.02 Release 01
	コンパイルオプション -cpu=rx200 -output=obj="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -nologo (統合開発環境のデフォルト設定を使用しています)
iodefine.h のバージョン	Version 1.2A
エンディアン	リトルエンディアン
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
サンプルコードのバージョン	Version 1.00
使用ボード	Renesas Starter Kit for RX210 (製品型名: R0K505210C000BE)

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。併せて参照してください。

- RX210グループ　初期設定例　Rev.2.00（R01AN1002JJ）
- RX21A グループ　初期設定例　Rev.1.10（R01AN1486JJ）
- RX220 グループ　初期設定例　Rev.1.10（R01AN1494JJ）

上記アプリケーションノートの初期設定関数を、本アプリケーションノートのサンプルコードで使用しています。Rev は本アプリケーションノート作成時点のものです。
最新版がある場合、最新版に差し替えて使用してください。最新版はルネサスエレクトロニクスホームページで確認および入手してください。

4.　ハードウェア説明

4.1　　ハードウェア構成例

図 4.1に接続例を示します。

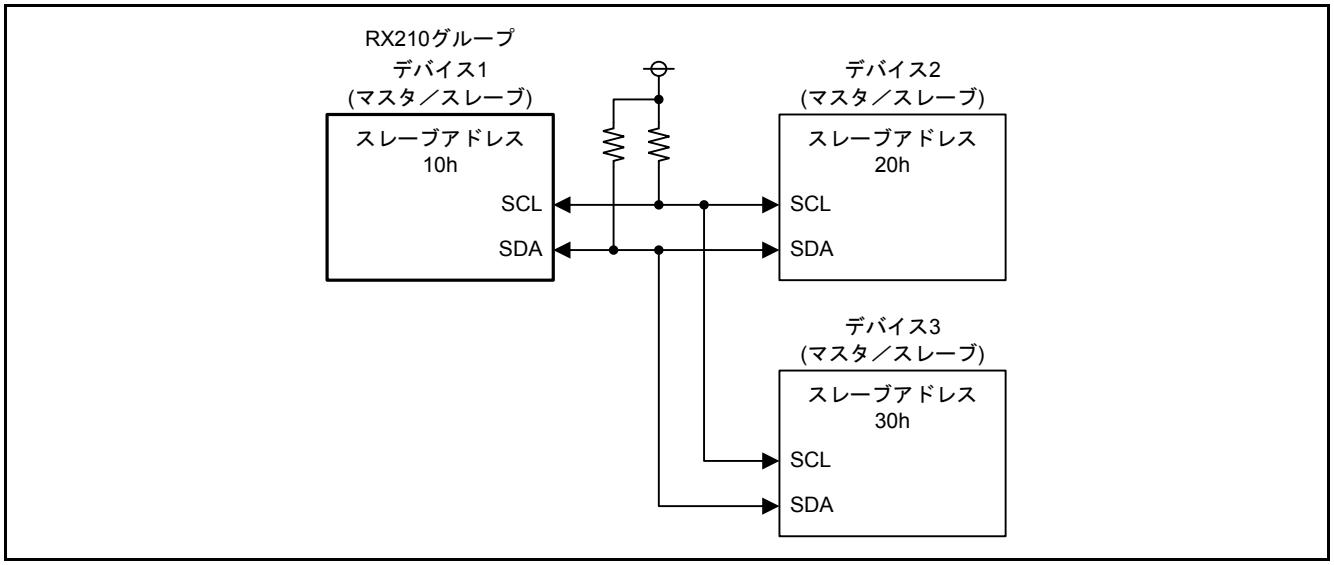


図4.1　接続例

4.2　　使用端子一覧

表 4.1に使用端子と機能を示します。

使用端子は 100 ピン版の製品を想定しています。100 ピン版未満の製品を使用する場合は、使用する製品に合わせて端子を選択してください。

表4.1　使用端子と機能

端子名	入出力	内容
P12/SCL	入出力	シリアルクロック入出力端子
P13/SDA	入出力	シリアルデータ入出力端子

5.　ソフトウェア説明

リセット解除後、初期設定を行って RIIC の動作を許可し、スレーブ動作を開始します。

バスフリー状態であることを確認すると、10 バイトのマスタ送信を開始します。スレーブアドレスは 20h、R/W ビットは“0”、送信データは 00h～09h を送信します。マスタ送信が終了すると、バスフリー状態であることを確認して、10 バイトのマスタ受信を開始します。スレーブアドレスは 20h、R/W ビットは“1”を送信します。マスタ受信が終了すると、バスフリー状態であることを確認して、RIIC の動作を禁止します。

マスタアービトレーションロストを検出した場合は、自動的にスレーブ受信モードになります。このとき、スレーブアドレスが一致していると、R/W ビットに従ってスレーブ受信、またはスレーブ送信を開始します。

NACK を検出した場合は、自動的に転送動作が中断されますので、ストップコンディションを発行して通信を終了します。

マスタ通信開始後、以下の条件が成立するとコールバック関数を呼び出します。

- アービトレーションロスト検出
- マスタ送信完了
- マスタ受信完了

使用する周辺機能の設定を以下に示します。

＜RIIC＞

- マスタ/スレーブ動作　: マスタ送信、マスタ受信、スレーブ受信、スレーブ送信
- アドレスフォーマット　: 7 ビットアドレスフォーマット
- スレーブアドレス　: 10h
- 転送速度　: 100kbps
- アービトレーションロスト検出　: マスタアービトレーションロスト検出
- 割り込み　: 送信データエンプティ割り込み(ICTXI)を許可
  : 送信終了割り込み(ICTEI)を許可
  : 受信データフル割り込み(ICRXI)を許可
  : NACK 受信割り込み(NAKI)を許可
  : ストップコンディション検出割り込み(SPI)を許可
  : アービトレーションロスト割り込み(ALI)を許可

5.1　動作概要

5.1.1　マスタ送信

(1)　マスタ送信開始
ICCR2.BBSY フラグが“0”であることを確認して ICCR2.ST ビットを“1”(スタートコンディションの発行を要求する)にします。

(2)　スタートコンディション発行
スタートコンディションが発行されると ICSR2.TDRE フラグが“1”になり、TXI0 割り込み要求が発生します。TXI0 割り込み処理で ICDRT レジスタにスレーブアドレスと R/W#ビットを書きます。

(3)　データ送信
ICDRT レジスタから ICDRS レジスタにデータが転送されると、再び TDRE フラグが“1”になり、TXI0 割り込み要求が発生します。TXI0 割り込み処理で ICDRT レジスタにマスタ送信バッファの値を書きます。最終データを書いたら、次に発生する TXI0 割り込みで ICIER.TIE ビットを“0”(ICTXI 割り込み要求を禁止)、ICIER.TEIE ビットを“1”(ICTEI 割り込み要求を許可)にします。

(4)　送信終了
最終データの送信が終了すると、ICSR2.TEND フラグが“1”になり、TEI0 割り込み要求が発生します。TEI0 割り込み処理で ICCR2.SP ビットを“1”(ストップコンディションの発行を要求する)にします。

(5)　ストップコンディション発行
ストップコンディションが発行されると ICSR2.STOP フラグが“1”になり、EEI0 割り込み要求が発生します。EEI0 割り込み処理で TIE ビットを“1”、TEIE ビットを“0”にして、コールバック関数(マスタ送信/受信完了)を呼び出します。

図 5.1にマスタ送信のタイミング図を示します。

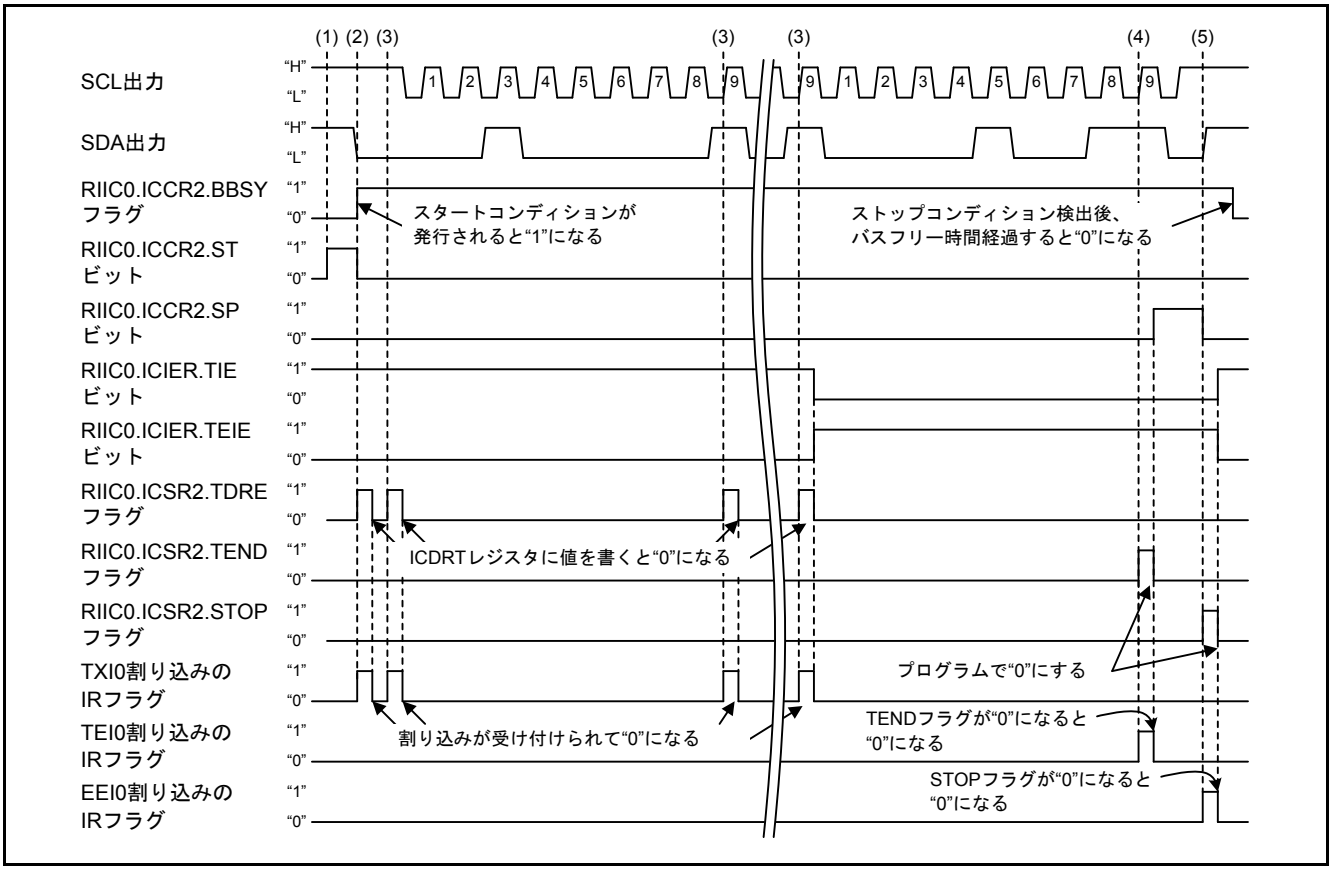


図5.1　マスタ送信のタイミング図

5.1.2　マスタ受信

(1)　マスタ受信開始
ICCR2.BBSY フラグが“0”であることを確認して ICCR2.ST ビットを“1”(スタートコンディションの発行を要求する)にします。

(2)　スタートコンディション発行
スタートコンディションが発行されると ICSR2.TDRE フラグが“1”になり、TXI0 割り込み要求が発生します。TXI0 割り込み処理で ICDRT レジスタにスレーブアドレスと R/W#ビットを書きます。ICIER.TIE ビットを“0”(ICTXI 割り込み要求を禁止)、TXI0 割り込みの IR フラグを“0”にします。

(3)　スレーブアドレス送信完了
スレーブアドレスの送信が完了すると ICSR2.RDRF フラグが“1”になり、RXI0 割り込み要求が発生します。RXI0 割り込み処理で ICDRR レジスタをダミーリードします。

(4)　データ受信完了
データの受信が完了すると RDRF フラグが“1”になり、RXI0 割り込み要求が発生します。RXI0 割り込み処理で ICDRR レジスタの値をマスタ受信バッファに格納します。(最終データ－2)バイト目の受信完了時は、ICMR3.WAIT ビットを“1”にします。(最終データ－1)バイト目の受信完了時は、ICMR3.RDRFS ビットを“1”、ICMR3.ACKBT ビットを“1”にします。最終データの受信完了時は、ICCR2.SP ビットを“1”(ストップコンディションの発行を要求する)、ACKBT ビットを“1”、WAIT ビットを“0”にします。

(5)　ストップコンディション発行
ストップコンディションが発行されると ICSR2.STOP フラグが“1”になり、EEI0 割り込み要求が発生します。EEI0 割り込み処理で TIE ビットを“1”にして、コールバック関数(マスタ送信/受信完了)を呼び出します。

図 5.2にマスタ受信のタイミング図を示します。

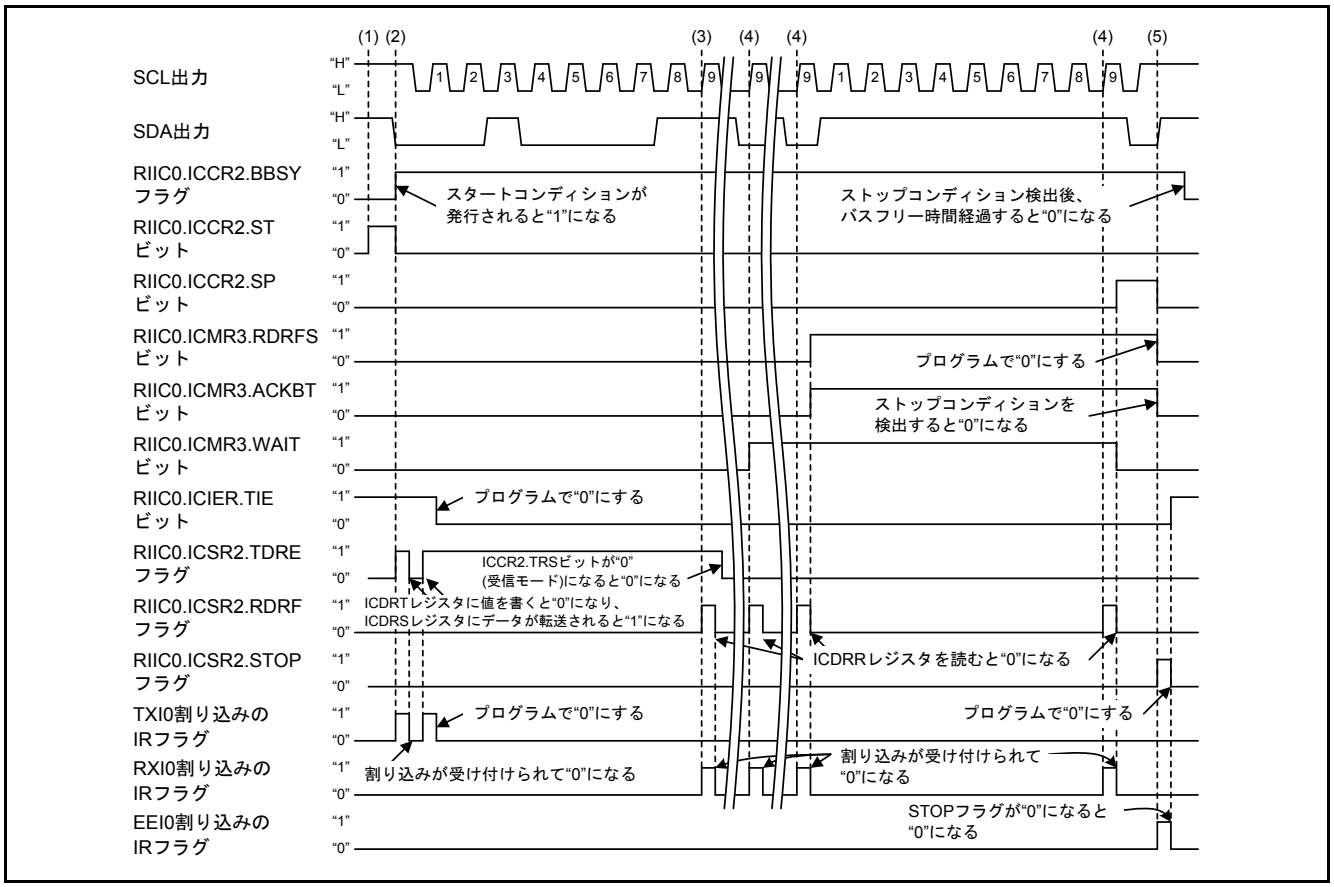


図5.2　マスタ受信のタイミング図

5.1.3　スレーブ動作

スレーブ動作中にスレーブアドレスが一致した場合は、R/W ビットに従ってスレーブ送信、またはスレーブ受信を開始します。

スレーブ送信、またはスレーブ受信が完了すると、スレーブ動作を停止して、コールバック関数を呼び出します。スレーブ動作を継続する場合は、ユーザ I/F 関数(スレーブ動作開始)を再度コールしてください。サンプルコードでは、コールバック関数でユーザ I/F 関数(スレーブ動作開始)をコールして、スレーブ動作を継続させています。

5.1.3.1　スレーブ送信

- スレーブ送信開始後は、NACK を受信するまでスレーブ送信バッファの先頭からデータを送信します。
- 送信数が設定しているスレーブ送信バイト数を超える場合は、FFh を送信します。
- 送信が完了すると、コールバック関数を呼び出します。ただし、スレーブ送信バイト数未満のデータ送信中に NACK を受信した場合はコールバック関数を呼び出さず、再びスレーブ送信が開始されたときにスレーブ送信バッファの先頭から送信します。

5.1.3.2　スレーブ受信

- スレーブ受信開始後は、ストップコンディションを検出するまでスレーブ受信バッファの先頭から受信データを格納します。
- 設定しているスレーブ受信バイト数を超えた受信データは、スレーブ受信バッファには格納せずに読み捨てます。
- 受信が完了すると、コールバック関数を呼び出します。ただし、スレーブ受信バイト数未満のデータ受信中にストップコンディションを検出した場合はコールバック関数を呼び出さず、再びスレーブ受信が開始されたときにスレーブ受信バッファの先頭から格納(上書き)します。

5.2　ファイル構成

表 5.1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されるファイルは除きます。

表5.1　サンプルコードで使用するファイル

ファイル名	概要	備考
main.c	メイン処理	
r_init_stop_module.c	リセット後に動作している周辺機能の停止	
r_init_stop_module.h	r_init_stop_module.c のヘッダファイル	
r_init_non_existent_port.c	存在しないポートの初期設定	
r_init_non_existent_port.h	r_init_non_existent_port.c のヘッダファイル	
r_init_clock.c	クロック初期設定	
r_init_clock.h	r_init_clock.c のヘッダファイル	
riic.c	RIIC 処理	
riic_int.c	RIIC 割り込み処理	
riic.h	riic.c のヘッダファイル	

5.3　オプション設定メモリ

表 5.2にサンプルコードで使用するオプション設定メモリの状態を示します。必要に応じて、お客様のシステムに最適な値を設定してください。

表5.2　サンプルコードで使用するオプション設定メモリ

シンボル	アドレス	設定値	内容
OFS0	FFFF FF8Fh～FFFF FF8Ch	FFFF FFFFh	リセット後、IWDT は停止 リセット後、WDT は停止
OFS1	FFFF FF8Bh～FFFF FF88h	FFFF FFFFh	リセット後、電圧監視 0 リセット無効 リセット後、HOCO 発振が無効
MDES	FFFF FF83h～FFFF FF80h	FFFF FFFFh	リトルエンディアン

5.4　定数一覧

表 5.3にサンプルコードで使用する定数を示します。

表5.3　サンプルコードで使用する定数

定数名	設定値	内容
SELF_ADDRESS	10h	自己スレーブアドレス
SLV_ADDRESS	20h	スレーブアドレス
MST_DATA_NUM	(10+1)	マスタ送信/受信バイト数 (バイト数+スレーブアドレス)
SLV_DATA_NUM	10	スレーブ送信/受信バイト数
MST_WRITE	00h	R/W ビットの設定値: 0: Write
MST_READ	01h	R/W ビットの設定値: 1: Read
RIIC_OK	0	関数 RiicMstStart のリターン値
RIIC_NG	1	関数 RiicMstStart のリターン値
RIIC_BUS_BUSY	2	関数 RiicMstStart のリターン値
RIIC_ST_MST_STOP	0	関数 RiicGetMstState のリターン値
RIIC_ST_MST_BUSY	1	関数 RiicGetMstState のリターン値
RIIC_ST_MST_NACK	2	関数 RiicGetMstState のリターン値
RIIC_ST_MST_AL	3	関数 RiicGetMstState のリターン値
RIIC_ST_MST_COMPLETE	4	関数 RiicGetMstState のリターン値
RIIC_SET	1	フラグのセット
RIIC_CLEAR	0	フラグのクリア
RIIC_ENABLE	1	RIIC 許可
RIIC_DISABLE	0	RIIC 禁止
RIIC_RXI	01h	関数 RiicIntEna、関数 RiicIntDis の引数
RIIC_TXI	02h	関数 RiicIntEna、関数 RiicIntDis の引数
RIIC_TEI	04h	関数 RiicIntEna、関数 RiicIntDis の引数

5.5　変数一覧

表 5.4にグローバル変数を、表 5.5、表 5.6にstatic 型変数を示します。

表5.4　グローバル変数

型	変数名	内容	使用関数
uint8_t	MstTrmBuff[256]	マスタ送信バッファ	main
uint8_t	MstRcvBuff[256]	マスタ受信バッファ	main
uint8_t	SlvTrmBuff[256]	スレーブ送信バッファ	main
uint8_t	SlvRcvBuff[256]	スレーブ受信バッファ	main

表5.5　static 型変数(1)

型	変数名	内容	使用関数
static uint8_t	RiicTrmAddr	スレーブアドレス	RiicMstStart RiicTDRE
static uint8_t *	RiicMstBuff	マスタ送信/受信バッファへのポインタ	RiicMstStart RiicRDRF RiicTDRE RiicTEND
static uint32_t	RiicMstCnt	マスタ送信/受信バイト数のカウンタ	RiicIni RiicMstStart RiicRDRF RiicTDRE RiicTEND
static uint32_t	RiicMstNum	マスタ送信/受信バイト数	
static uint8_t *	RiicSlvTrmBuff	スレーブ送信バッファへのポインタ	RiicSlvStart RiicTDRE RiicSTOP
static uint8_t *	RiicSlvTrmStartBuff	スレーブ送信バッファへのポインタ	RiicSlvStart RiicSTOP
static uint32_t	RiicSlvTrmCnt	スレーブ送信バイト数のカウンタ	RiicIni RiicSlvStart RiicTDRE RiicTEND RiicSTOP
static uint8_t *	RiicSlvRcvBuff	スレーブ受信バッファへのポインタ	RiicSlvStart RiicRDRF RiicSTOP
static uint8_t *	RiicSlvRcvStartBuff	スレーブ受信バッファへのポインタ	RiicSlvStart RiicSTOP
static uint32_t	RiicSlvRcvCnt	スレーブ受信バイト数のカウンタ	RiicIni RiicSlvStart RiicRDRF RiicSTOP
static uint32_t	RiicSlvNum	スレーブ送信/受信バイト数	RiicIni RiicSlvStart RiicRDRF RiicTDRE RiicTEND RiicSTOP

表5.6　static 型変数(2)

型	変数名	内容	使用関数
static volatile uint8_t	RiicStartFlg	通信中フラグ　0: 通信中以外 1: 通信中	RiicIni RiicRDRF RiicTDRE RiicSTOP RiicAL
static volatile uint8_t	RiicMstFlg	マスタ動作フラグ　0: スレーブ動作 1: マスタ動作	RiicMstStart RiicRDRF RiicTDRE RiicTEND RiicSTOP RiicNACK RiicAL
static volatile uint8_t	RiicMstState	マスタ状態 0: 停止 1: ビジー(通信中) 2: NACK 検出 3: アービトレーションロスト検出 4: 通信完了	RiicIni RiicMstStart RiicGetMstState RiicSTOP RiicNACK RiicAL

5.6　関数一覧

表 5.7にサンプルコードで使用する関数を示します。

表5.7　サンプルコードで使用する関数

関数名	概要
main	メイン処理
port_init	ポート初期設定
R_INIT_StopModule	リセット後に動作している周辺機能の停止
R_INIT_NonExistentPort	存在しないポートの初期設定
R_INIT_Clock	クロック初期設定
CbSlaveTrm	コールバック関数(スレーブ送信完了)
CbSlaveRcv	コールバック関数(スレーブ受信完了)
CbMaster	コールバック関数(マスタ送信/受信完了)
RiicIni	ユーザ I/F 関数(RIIC 初期設定)
RiicSlvStart	ユーザ I/F 関数(スレーブ動作開始)
RiicMstStart	ユーザ I/F 関数(マスタ動作開始)
RiicGetMstState	ユーザ I/F 関数(マスタ状態取得)
RiicEnable	RIIC 許可
RiicDisable	RIIC 禁止
RiicIntEna	RIIC 割り込み許可
RiicIntDis	RIIC 割り込み禁止
RiicRDRF	受信データフル割り込み
RiicTDRE	送信データエンプティ割り込み
RiicTEND	送信終了割り込み
RiicSTOP	ストップコンディション検出割り込み
RiicNACK	NACK 検出割り込み
RiicAL	アービトレーションロスト検出割り込み
RiicTMO	タイムアウト検出割り込み
RiicSTART	スタートコンディション検出割り込み
Excep_RIIC0_EEI0	RIIC0.EEI0 割り込み処理
Excep_RIIC0_RXI0	RIIC0.RXI0 割り込み処理
Excep_RIIC0_TXI0	RIIC0.TXI0 割り込み処理
Excep_RIIC0_TEI0	RIIC0.TEI0 割り込み処理

5.7　関数仕様

サンプルコードの関数仕様を示します。

main	
概　要	メイン処理
ヘッダ	なし
宣　言	void main(void)
説　明	初期設定後、10 バイトのマスタ送信を行って、10 バイトのマスタ受信を行います。
引　数	なし
リターン値	なし

port_init	
概　要	ポート初期設定
ヘッダ	なし
宣　言	void port_init(void)
説　明	ポートの初期設定を行います。
引　数	なし
リターン値	なし

R_INIT_StopModule	
概　要	リセット後に動作している周辺機能の停止
ヘッダ	r_init_stop_module.h
宣　言	void R_INIT_StopModule(void)
説　明	モジュールストップ状態へ遷移する設定を行います。
引　数	なし
リターン値	なし
備　考	サンプルコードでは、モジュールストップ状態への遷移は行っていません。 本関数の詳細は、各グループのアプリケーションノート「初期設定例」を参照してください。

R_INIT_NonExistentPort	
概　要	存在しないポートの初期設定
ヘッダ	r_init_non_existent_port.h
宣　言	void R_INIT_NonExistentPort(void)
説　明	100 ピン未満の製品に対して、存在しないポートの端子に対応するポート方向レジスタの初期設定を行います。
引　数	なし
リターン値	なし
備　考	サンプルコードでは、100 ピン版(PIN_SIZE=100)に設定しています。 本関数をコールした後に、存在しないポートを含む PDR、PODR レジスタへバイト単位で書き込む場合、存在しないポートの方向制御ビットには“1”、ポート出力データ格納ビットには“0”を設定してください。 本関数の詳細は、各グループのアプリケーションノート「初期設定例」を参照してください。

R_INIT_Clock	
概　要	クロック初期設定
ヘッダ	r_init_clock.h
宣　言	void R_INIT_Clock(void)
説　明	クロックの初期設定を行います。
引　数	なし
リターン値	なし
備　考	サンプルコードでは、システムクロックを PLL とし、サブクロックを使用しない処理を選択しています。 本関数の詳細は、各グループのアプリケーションノート「初期設定例」を参照してください。

CbSlaveTrm	
概　要	コールバック関数(スレーブ送信完了)
ヘッダ	なし
宣　言	void CbSlaveTrm(void)
説　明	スレーブ送信完了時に呼び出されます。
引　数	なし
リターン値	なし

CbSlaveRcv	
概　要	コールバック関数(スレーブ受信完了)
ヘッダ	なし
宣　言	void CbSlaveRcv(void)
説　明	スレーブ受信完了時に呼び出されます。
引　数	なし
リターン値	なし

CbMaster	
概　要	コールバック関数(マスタ送信/受信完了)
ヘッダ	なし
宣　言	void CbMaster(void)
説　明	マスタ通信開始後、以下の条件が成立すると呼び出されます。 ・アービトレーションロスト検出 ・マスタ送信完了 ・マスタ受信完了
引　数	なし
リターン値	なし
備　考	サンプルコードでは処理を行っていません。必要に応じてプログラムを追加してください。

RiicIni			
概　要	ユーザ I/F 関数(RIIC 初期設定)		
ヘッダ	riic.h		
宣　言	void RiicIni(uint8_t in_SelfAddr, uint8_t in_Enable)		
説　明	RIIC の初期設定を行います。		
引　数	uint8_t in_SelfAddr		: 自己アドレス(ビット 0 は“0”に設定)
	uint8_t in_Enable		: RIIC 許可/禁止
		RIIC_ENABLE	: RIIC 許可
		RIIC_DISABLE	: RIIC 禁止
リターン値	なし		

RiicSlvStart			
概　要	ユーザ I/F 関数(スレーブ動作開始)		
ヘッダ	riic.h		
宣　言	void RiicSlvStart(uint8_t * in_RcvAddr, uint8_t * in_TrmAddr, uint32_t in_num, CallBackFunc cbTrm, CallBackFunc cbRcv)		
説　明	スレーブ動作を開始します。		
引　数	uint8_t * in_RcvAddr		: スレーブ受信データ格納ポインタ
	uint8_t * in_TrmAddr		: スレーブ送信データ格納ポインタ
	uint32_t	in_num	: スレーブ送信/受信バイト数
	CallBackFunc	cbTrm	: コールバック関数(スレーブ送信完了)
	CallBackFunc	cbRcv	: コールバック関数(スレーブ受信完了)
リターン値	なし		

RiicMstStart			
概　要	ユーザ I/F 関数(マスタ動作開始)		
ヘッダ	riic.h		
宣　言	uint8_t RiicMstStart(uint8_t in_addr, uint8_t * in_buff, uint32_t in_num, CallBackFunc cb)		
説　明	マスタ動作を開始します。		
引　数	uint8_t	in_addr	: スレーブアドレス(ビット 0 は R/W ビット)
	uint8_t *	in_buff	: マスタ送信/受信データ格納ポインタ
	uint32_t	in_num	: マスタ送信/受信バイト数
	CallBackFunc	cb	: コールバック関数(マスタ送信/受信完了)
リターン値	RIIC_OK		: 正常終了
	RIIC_NG		: 引数エラー(送信/受信バイト数が 2 未満)
	RIIC_BUS_BUSY		: バスビジー

RiicGetMstState		
概　要	ユーザ I/F 関数(マスタ状態取得)	
ヘッダ	riic.h	
宣　言	uint8_t RiicGetMstState(void)	
説　明	マスタ状態を返します。	
引　数	なし	
リターン値	RIIC_ST_MST_STOP	：停止
	RIIC_ST_MST_BUSY	：ビジー(通信中)
	RIIC_ST_MST_NACK	：NACK 検出
	RIIC_ST_MST_AL	：アービトレーションロスト検出
	RIIC_ST_MST_COMPLETE	：通信完了

RiicEnable			
概　要	RIIC 許可		
ヘッダ	なし		
宣　言	void RiicEnable(uint8_t addr)		
説　明	RIIC 動作を許可します。		
引　数	uint8_t	addr	：自己アドレス
リターン値	なし		

RiicDisable	
概　要	RIIC 禁止
ヘッダ	なし
宣　言	void RiicDisable(void)
説　明	RIIC 動作を禁止します。
引　数	なし
リターン値	なし

RiicIntEna		
概　要	RIIC 割り込み許可	
ヘッダ	なし	
宣　言	void RiicIntEna(uint8_t req)	
説　明	RIIC の割り込み要求を許可します。	
引　数	uint8_t req	：許可要求
	RIIC_RXI	: RXI 割り込み要求を許可
	RIIC_TXI	: TXI 割り込み要求を許可
	RIIC_TEI	: TEI 割り込み要求を許可
リターン値	なし	

RiicIntDis	
概　要	RIIC 割り込み禁止
ヘッダ	なし
宣　言	void RiicIntDis(uint8_t req)
説　明	RIIC の割り込み要求を禁止します。
引　数	uint8_t req　　: 禁止要求 RIIC_RXI　: RXI 割り込み要求を禁止 RIIC_TXI　: TXI 割り込み要求を禁止 RIIC_TEI　: TEI 割り込み要求を禁止
リターン値	なし

RiicRDRF	
概　要	受信データフル割り込み
ヘッダ	riic.h
宣　言	void RiicRDRF(void)
説　明	RIIC0.RXI0 割り込み処理関数から呼び出されます。受信データを読み出します。
引　数	なし
リターン値	なし

RiicTDRE	
概　要	送信データエンプティ割り込み
ヘッダ	riic.h
宣　言	void RiicTDRE(void)
説　明	RIIC0.TXI0 割り込み処理関数から呼び出されます。送信データを書き込みます。
引　数	なし
リターン値	なし

RiicTEND	
概　要	送信終了割り込み
ヘッダ	riic.h
宣　言	void RiicTEND(void)
説　明	RIIC0.TEI0 割り込み処理関数から呼び出されます。ストップコンディションを発行します。
引　数	なし
リターン値	なし

RiicSTOP	
概　要	ストップコンディション検出割り込み
ヘッダ	riic.h
宣　言	void RiicSTOP(void)
説　明	RIIC0.EEI0 割り込み処理関数から呼び出されます。コールバック関数を呼び出します。
引　数	なし
リターン値	なし

RiicNACK	
概　要	NACK 検出割り込み
ヘッダ	riic.h
宣　言	void RiicNACK(void)
説　明	RIIC0.EEI0 割り込み処理関数から呼び出されます。ストップコンディションを発行します。
引　数	なし
リターン値	なし

RiicAL	
概　要	アービトレーションロスト検出割り込み
ヘッダ	riic.h
宣　言	void RiicAL(void)
説　明	RIIC0.EEI0 割り込み処理関数から呼び出されます。コールバック関数を呼び出します。
引　数	なし
リターン値	なし

RiicTMO	
概　要	タイムアウト検出割り込み
ヘッダ	riic.h
宣　言	void RiicTMO(void)
説　明	RIIC0.EEI0 割り込み処理関数から呼び出されます。
引　数	なし
リターン値	なし
備　考	サンプルコードでは処理を行っていません。必要に応じてプログラムを追加してください。

RiicSTART	
概　要	スタートコンディション検出割り込み
ヘッダ	なし
宣　言	void RiicSTART(void)
説　明	RIIC0.EEI0 割り込み処理関数から呼び出されます。
引　数	なし
リターン値	なし
備　考	サンプルコードでは処理を行っていません。必要に応じてプログラムを追加してください。

Excep_RIIC0_EEI0	
概　要	RIIC0.EEI0 割り込み処理
ヘッダ	なし
宣　言	void Excep_RIIC0_EEI0(void)
説　明	通信エラー/イベント発生割り込み処理を行います。
引　数	なし
リターン値	なし

Excep_RIIC0_RXI0	
概　要	RIIC0.RXI0 割り込み処理
ヘッダ	なし
宣　言	void Excep_RIIC0_RXI0(void)
説　明	受信データフル割り込み処理を行います。
引　数	なし
リターン値	なし

Excep_RIIC0_TXI0	
概　要	RIIC0.TXI0 割り込み処理
ヘッダ	なし
宣　言	void Excep_RIIC0_TXI0(void)
説　明	送信データエンプティ割り込み処理を行います。
引　数	なし
リターン値	なし

Excep_RIIC0_TEI0	
概　要	RIIC0.TEI0 割り込み処理
ヘッダ	なし
宣　言	void Excep_RIIC0_TEI0(void)
説　明	送信終了割り込み処理を行います。
引　数	なし
リターン値	なし

5.8　フローチャート

5.8.1　メイン処理

図 5.3にメイン処理のフローチャートを示します。

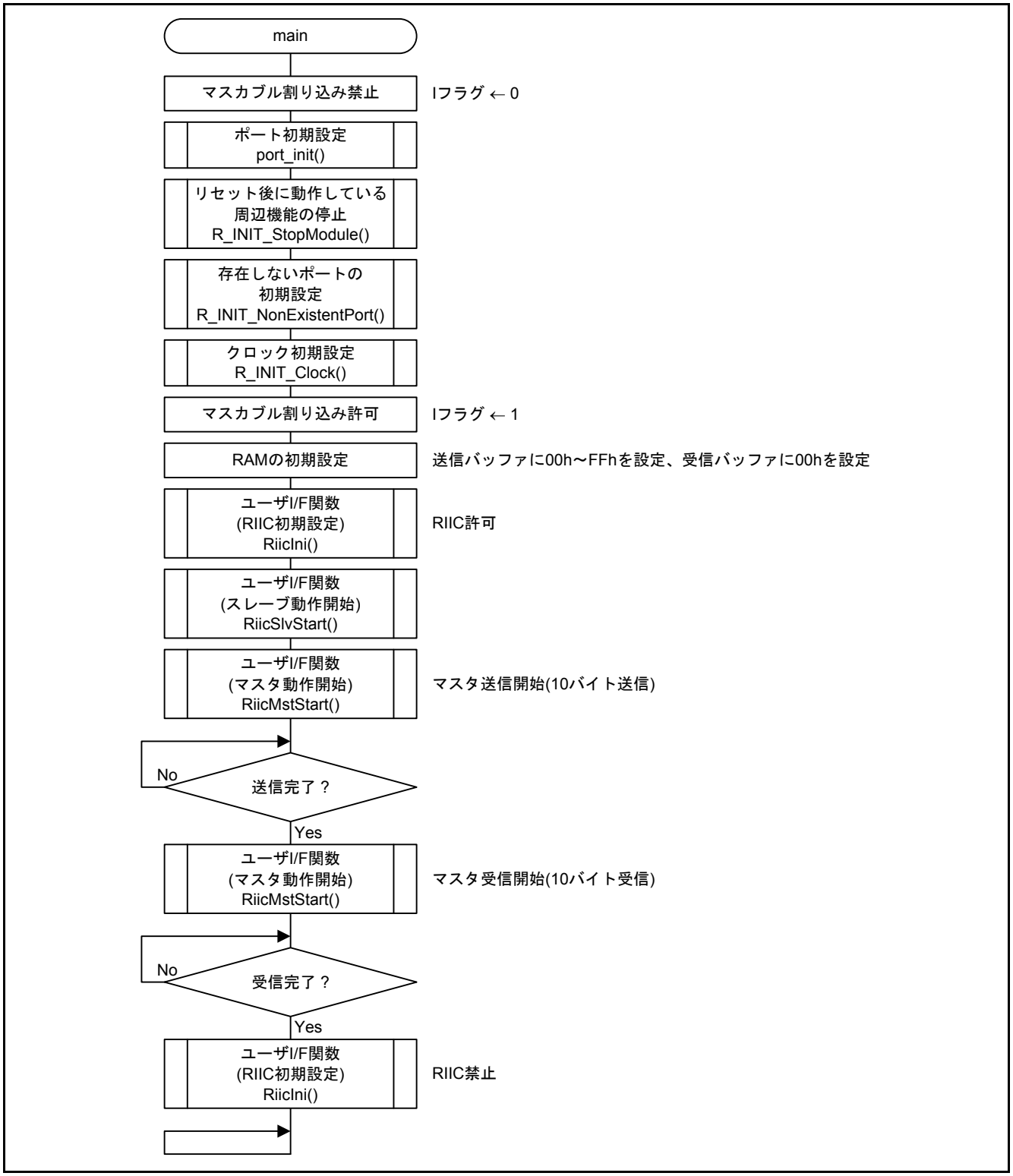


図5.3　メイン処理

5.8.2　ポート初期設定

図 5.4にポート初期設定のフローチャートを示します。

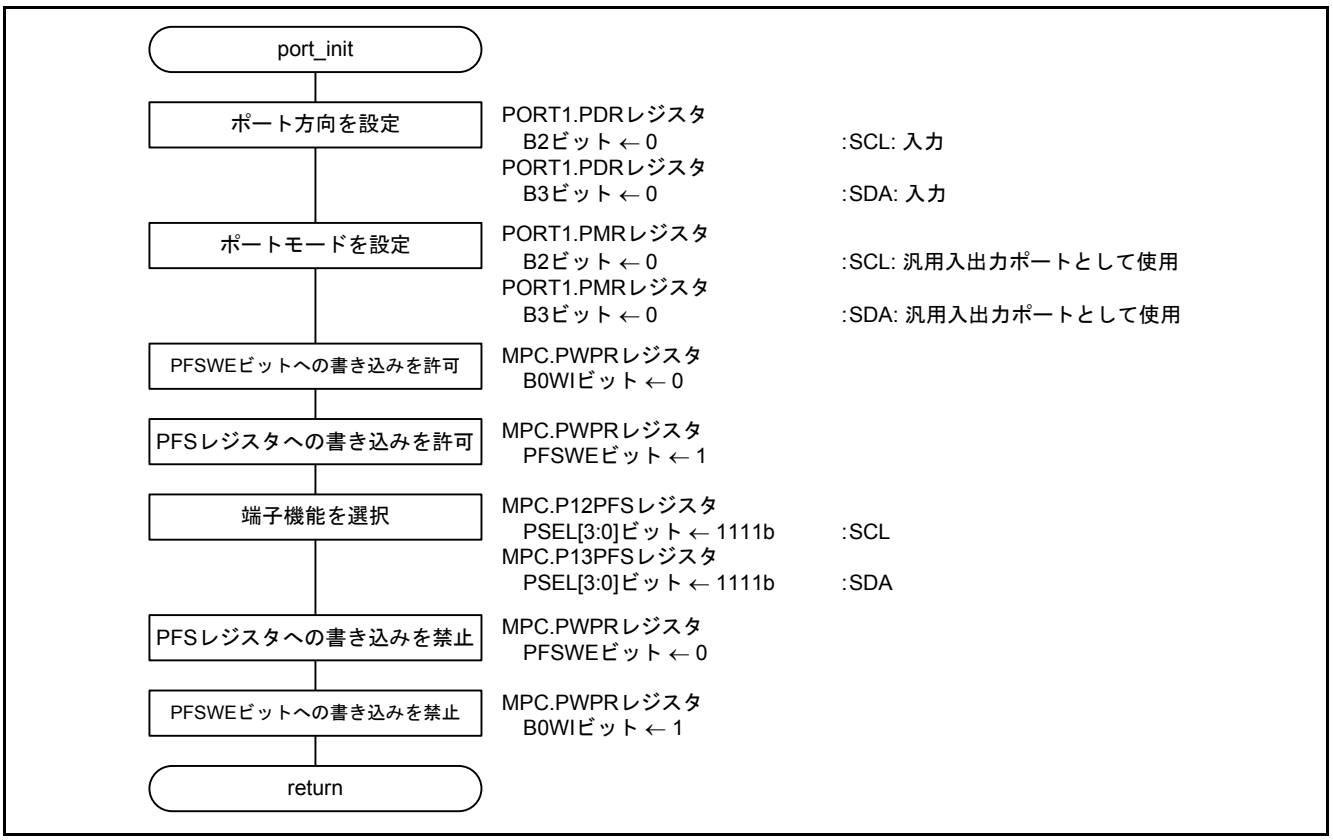


図5.4　ポート初期設定

5.8.3　コールバック関数(スレーブ送信完了)

図 5.5にコールバック関数(スレーブ送信完了)のフローチャートを示します。

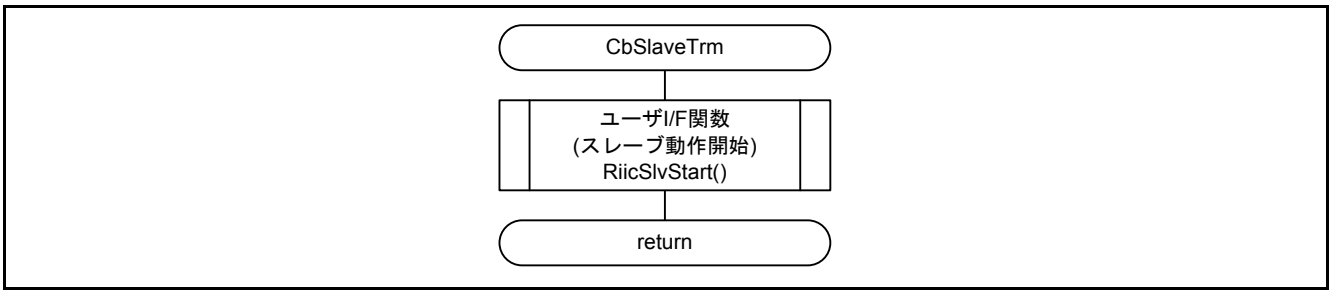


図5.5　コールバック関数(スレーブ送信完了)

5.8.4　コールバック関数(スレーブ受信完了)

図 5.6にコールバック関数(スレーブ受信完了)のフローチャートを示します。

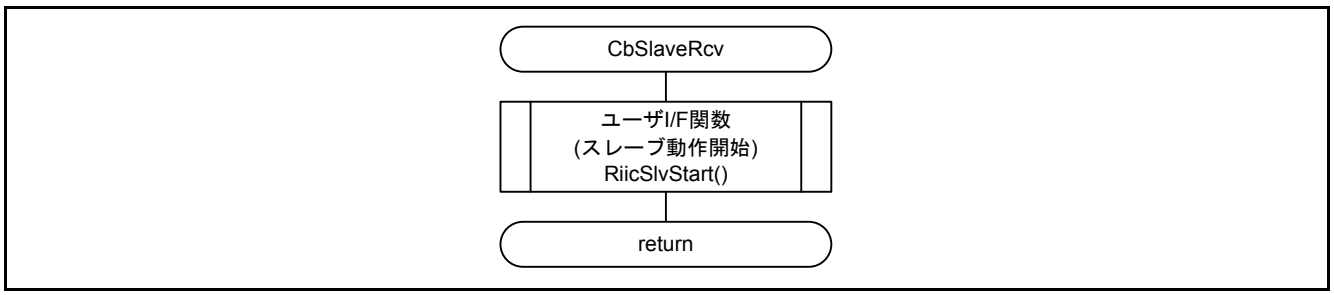
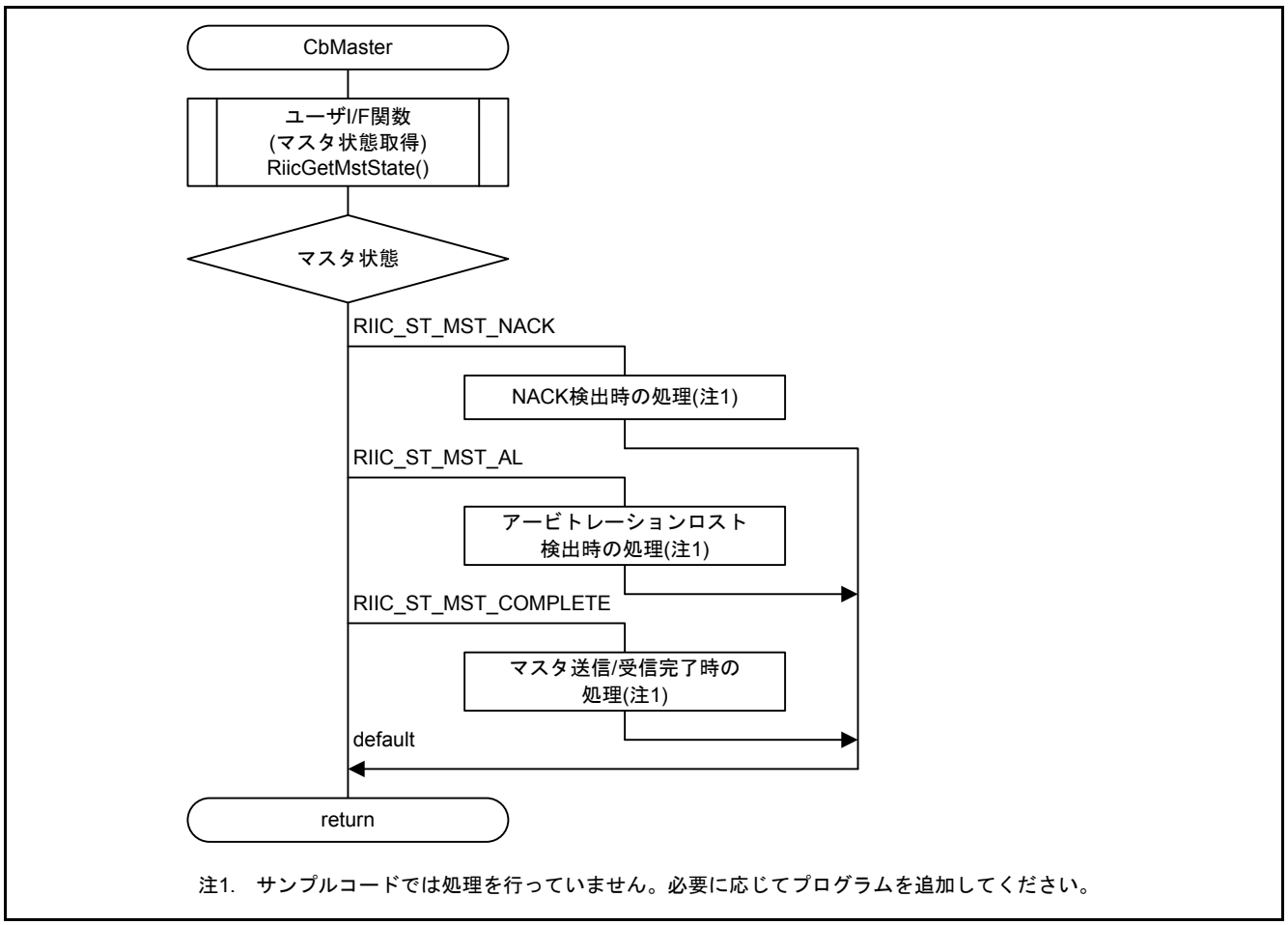


図5.6　コールバック関数(スレーブ受信完了)

5.8.5　コールバック関数(マスタ送信/受信完了)

図 5.7にコールバック関数(マスタ送信/受信完了)のフローチャートを示します。



注1.　サンプルコードでは処理を行っていません。必要に応じてプログラムを追加してください。

図5.7　コールバック関数(マスタ送信/受信完了)

5.8.6　ユーザ I/F 関数(RIIC 初期設定)

図 5.8にユーザ I/F 関数(RIIC 初期設定)のフローチャートを示します。

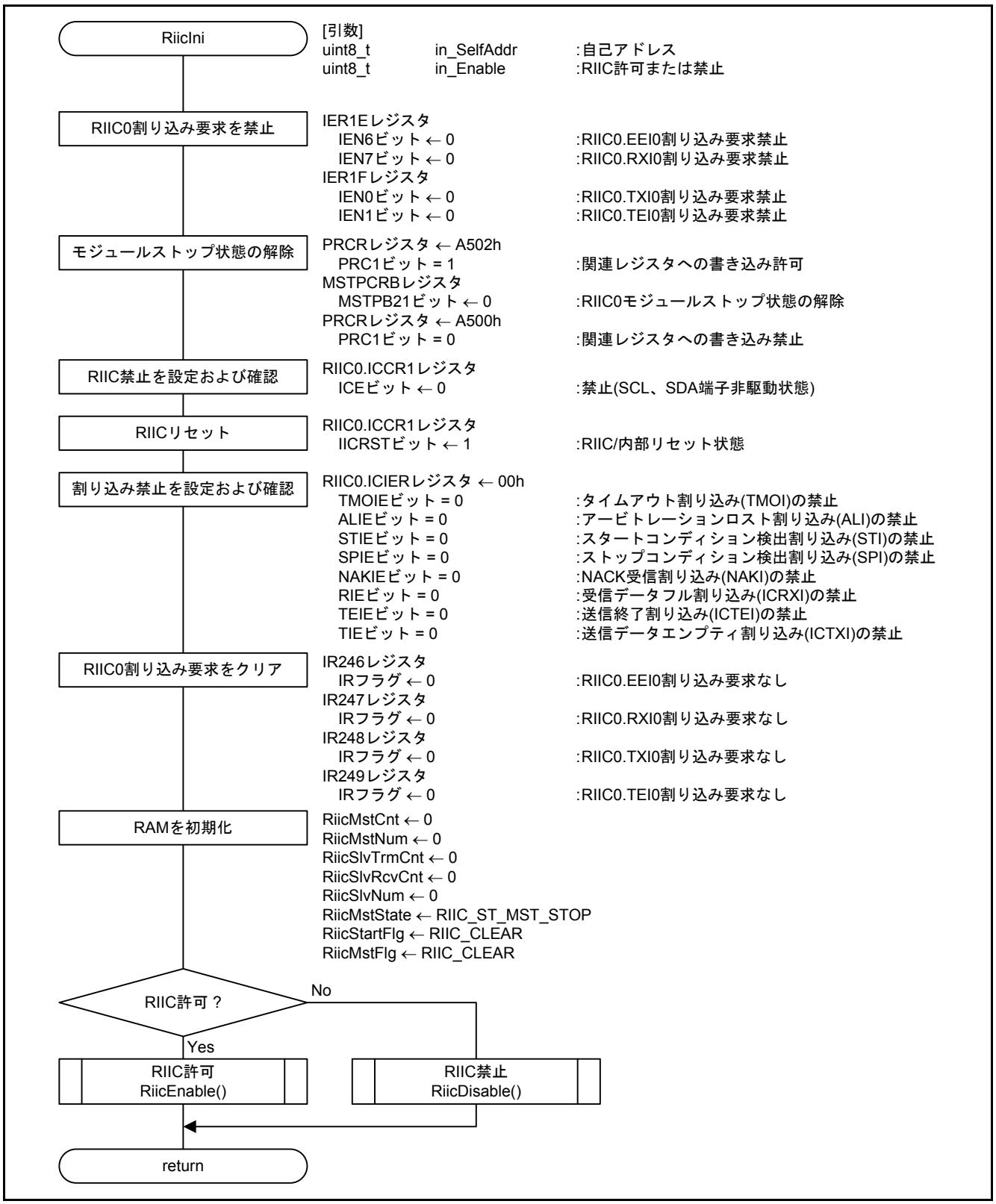


図5.8　ユーザ I/F 関数(RIIC 初期設定)

5.8.7　ユーザ I/F 関数(スレーブ動作開始)

図 5.9にユーザ I/F 関数(スレーブ動作開始)のフローチャートを示します。

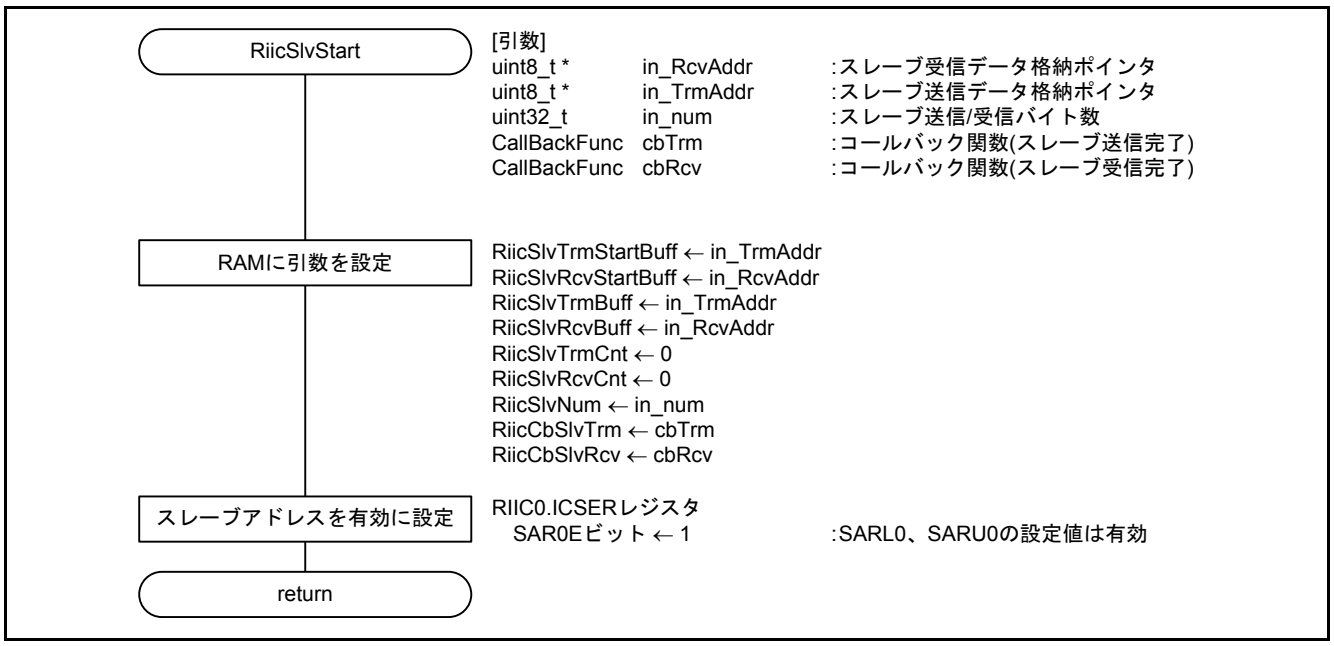


図5.9　ユーザ I/F 関数(スレーブ動作開始)

5.8.8　ユーザ I/F 関数(マスタ動作開始)

図 5.10にユーザ I/F 関数(マスタ動作開始)のフローチャートを示します。

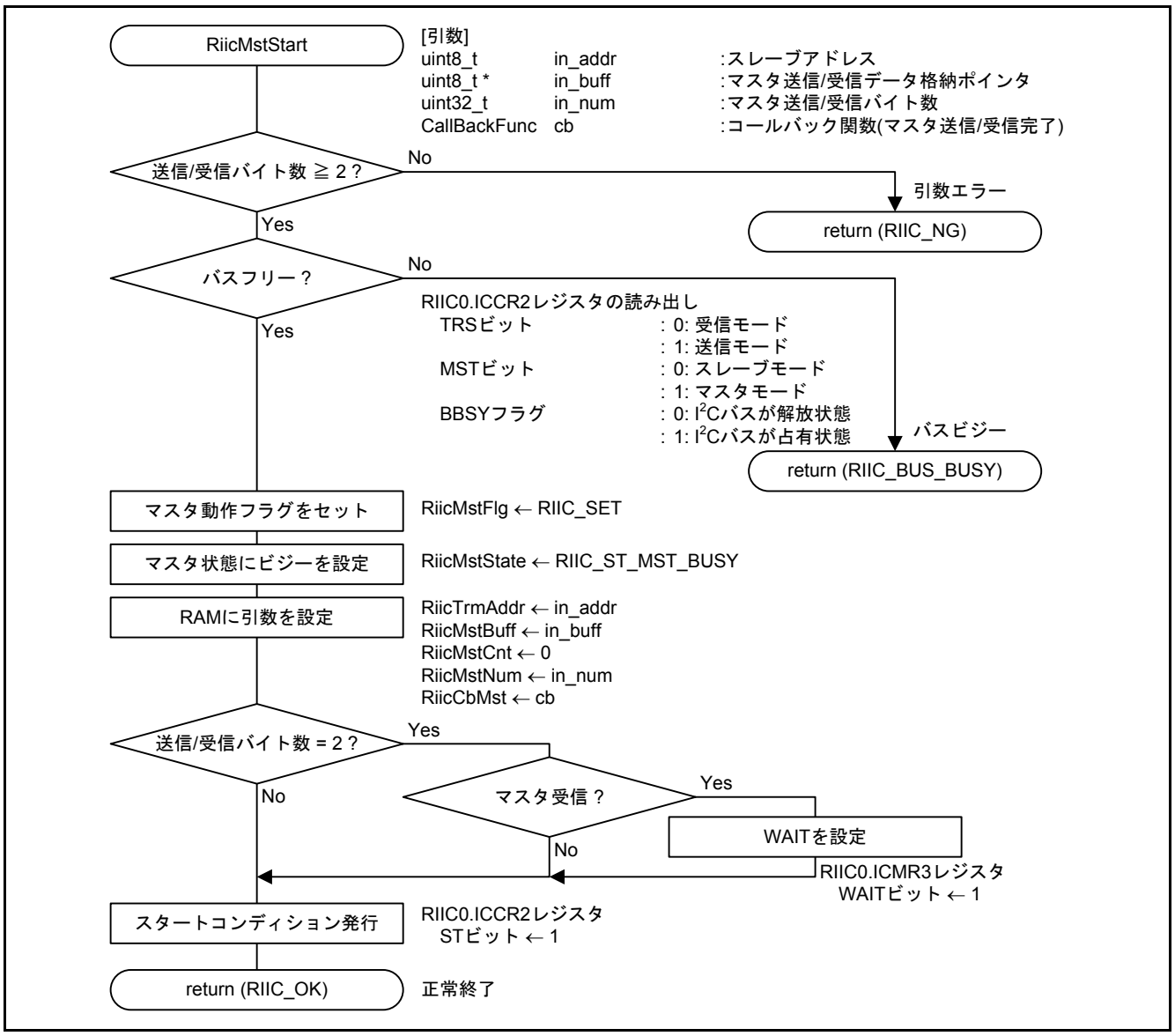


図5.10　ユーザ I/F 関数(マスタ動作開始)

5.8.9　ユーザ I/F 関数(マスタ状態取得)

図 5.11にユーザ I/F 関数(マスタ状態取得)のフローチャートを示します。

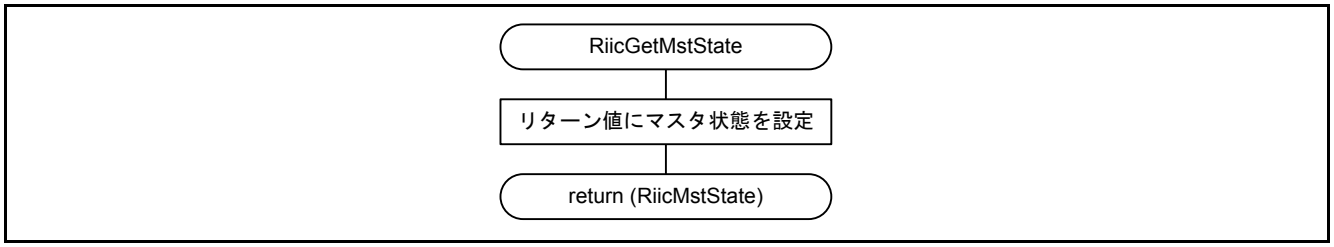


図5.11　ユーザ I/F 関数(マスタ状態取得)

5.8.10　RIIC 許可

図 5.12にRIIC 許可のフローチャートを示します。

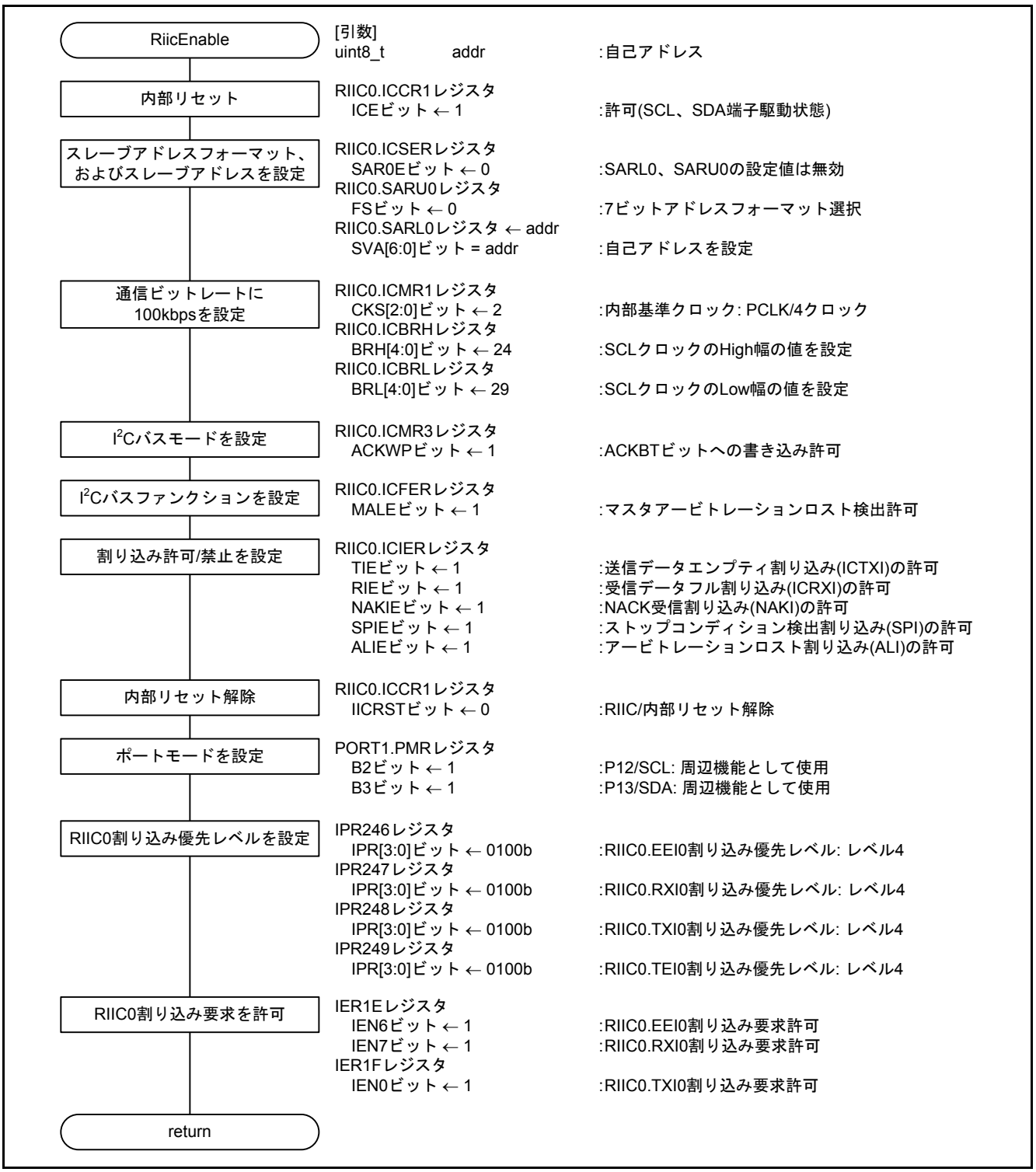


図5.12　RIIC 許可

5.8.11　RIIC 禁止

図 5.13にRIIC 禁止のフローチャートを示します。

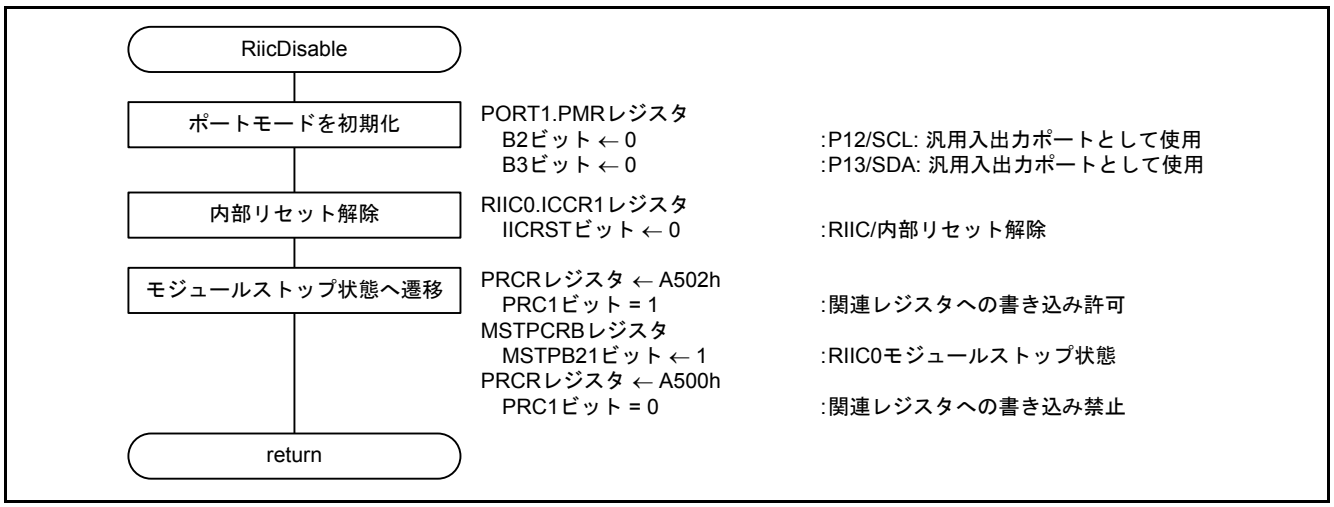


図5.13　RIIC 禁止

5.8.12　RIIC 割り込み許可

図 5.14にRIIC 割り込み許可のフローチャートを示します。

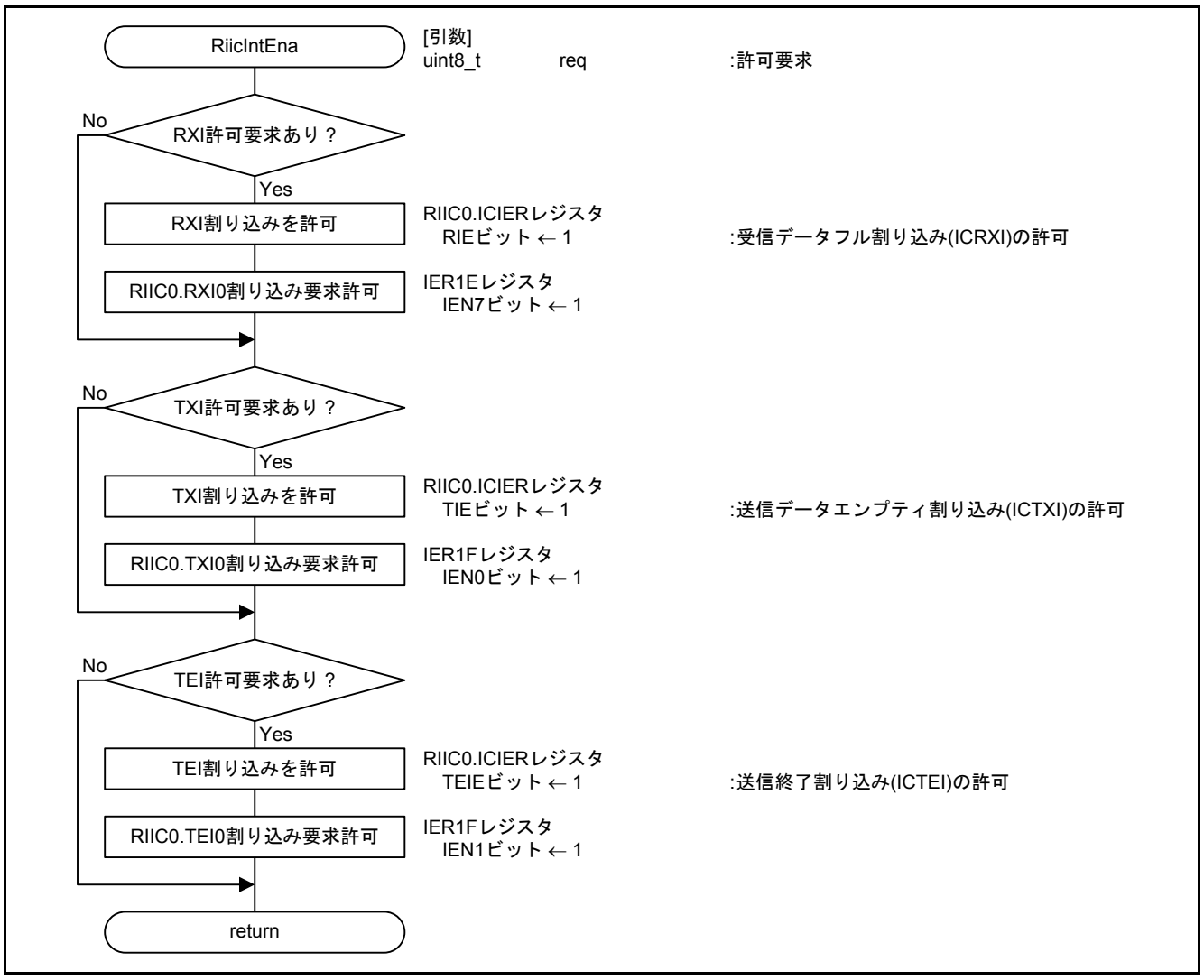


図5.14　RIIC 割り込み許可

5.8.13　RIIC 割り込み禁止

図 5.15にRIIC 割り込み禁止のフローチャートを示します。

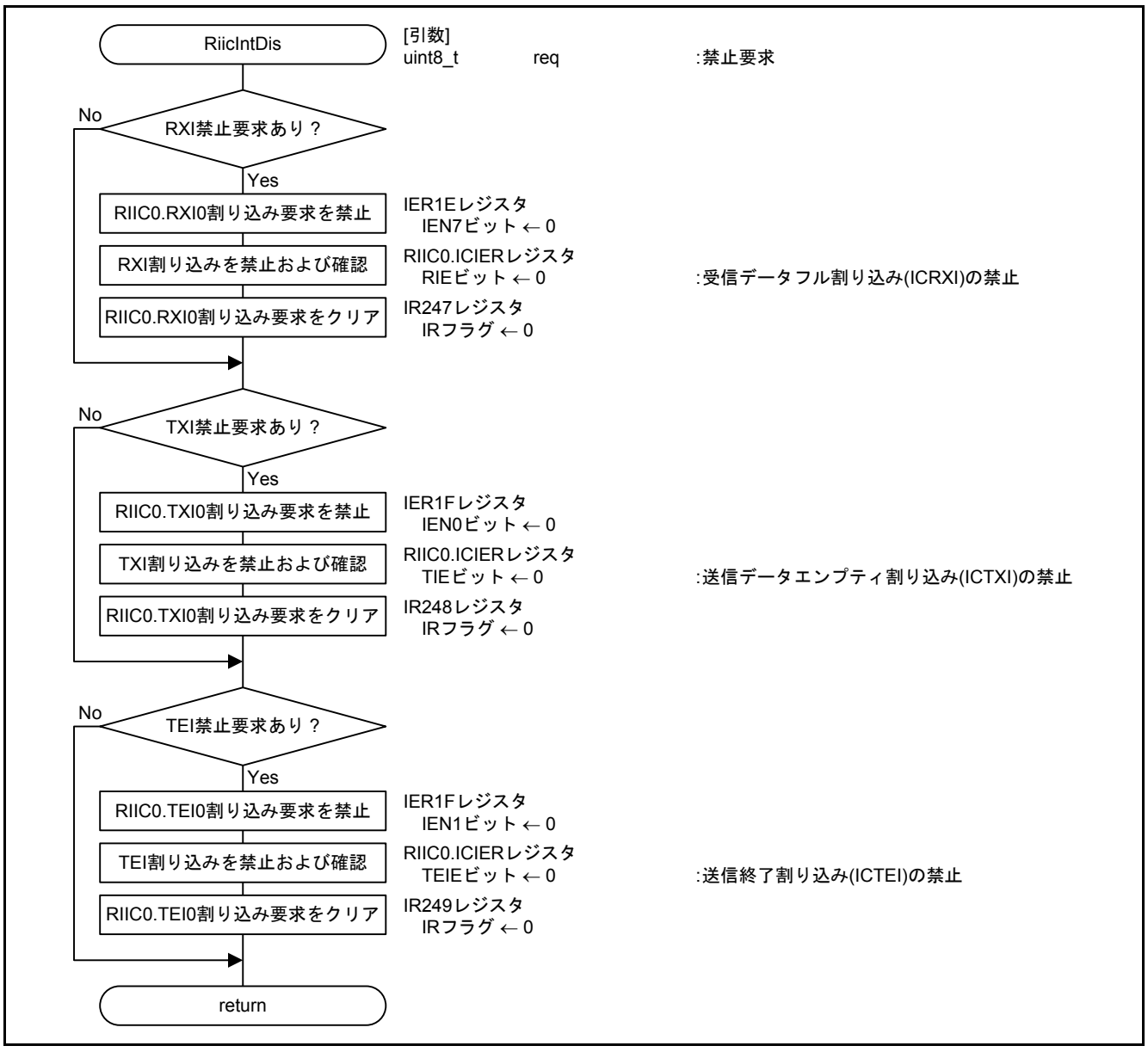


図5.15　RIIC 割り込み禁止

5.8.14　受信データフル割り込み

図 5.16、図 5.17に受信データフル割り込みのフローチャートを示します。

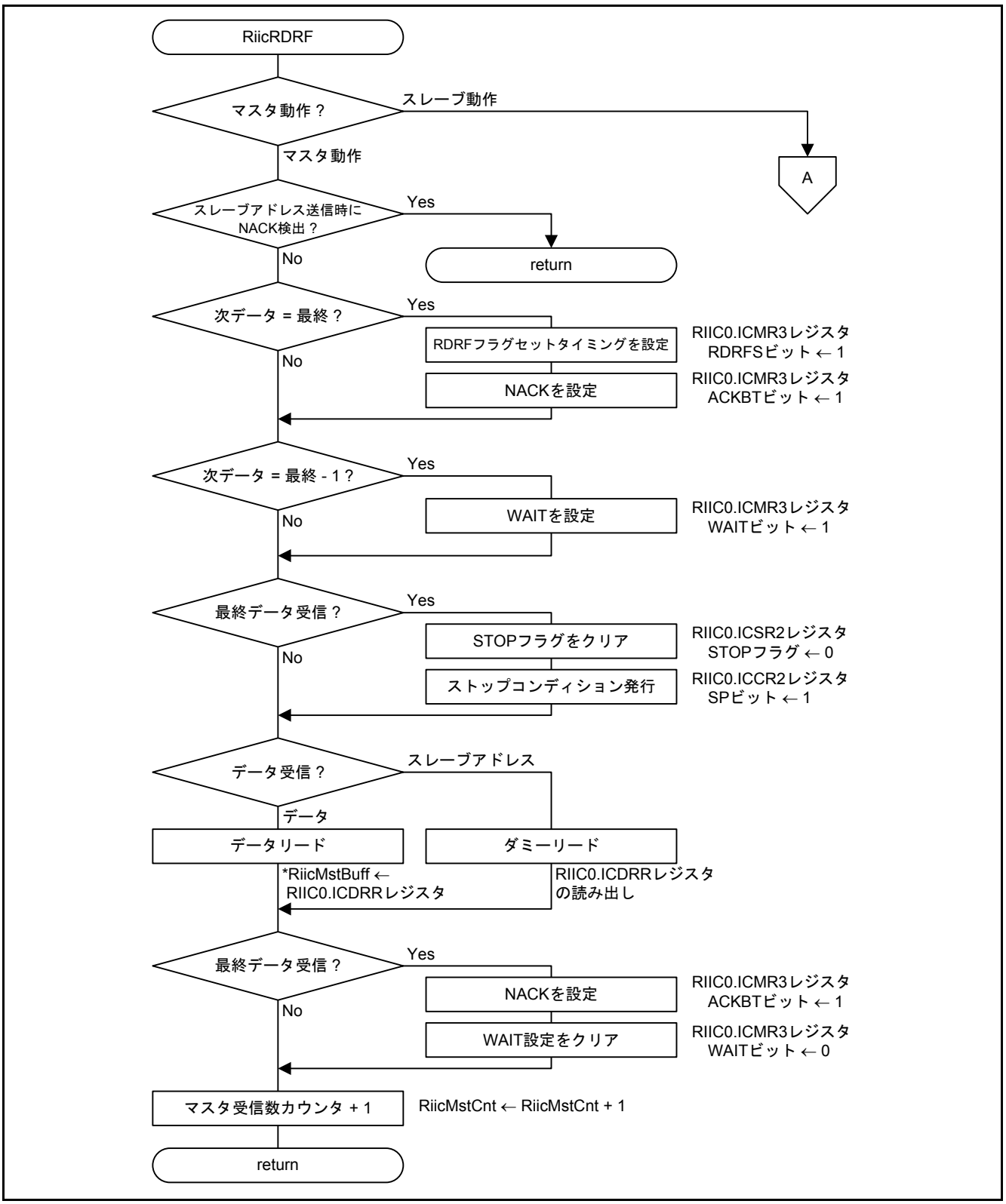


図5.16　受信データフル割り込み(1)

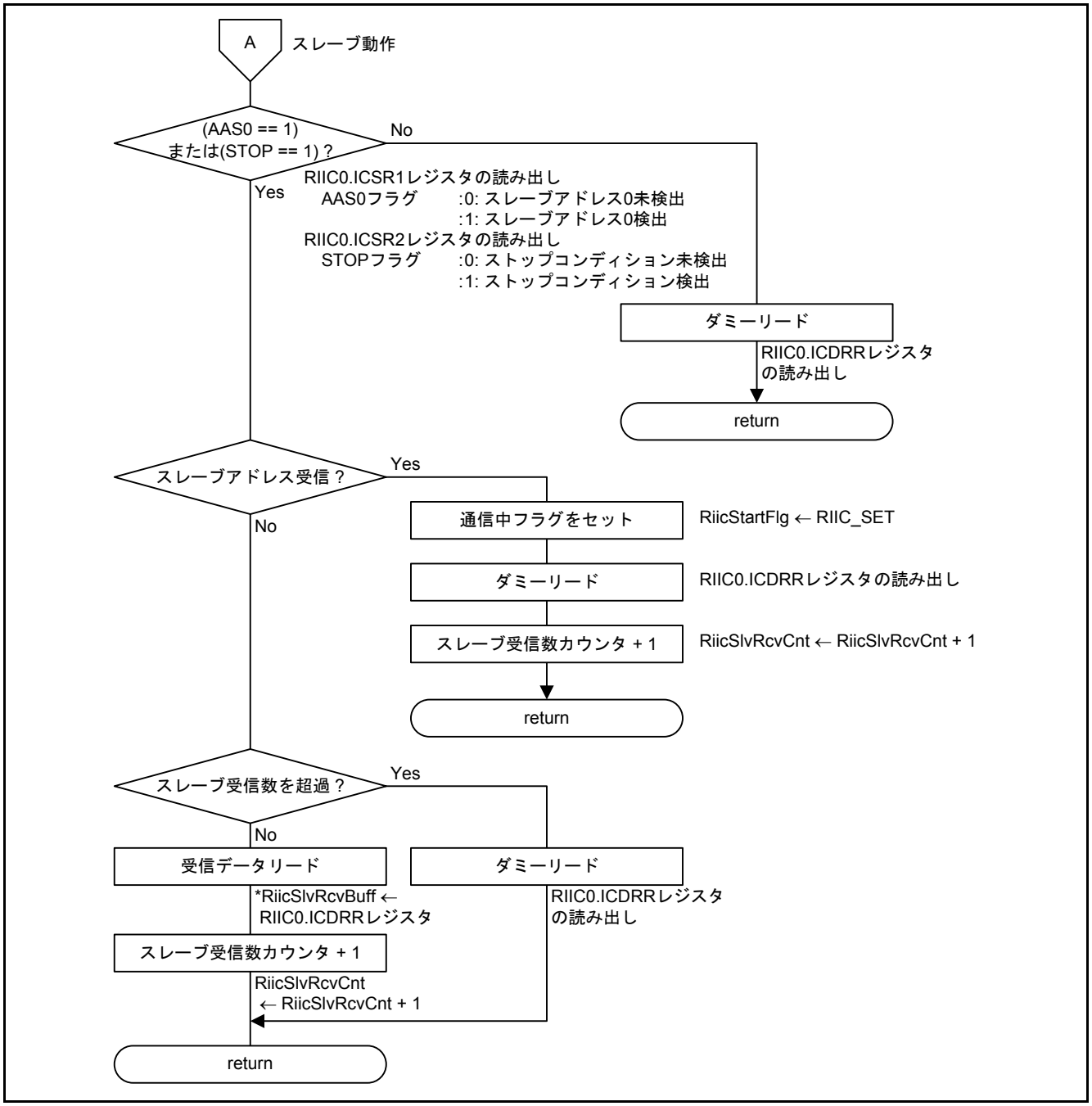


図5.17　受信データフル割り込み(2)

5.8.15　送信データエンプティ割り込み

図 5.18に送信データエンプティ割り込みのフローチャートを示します。

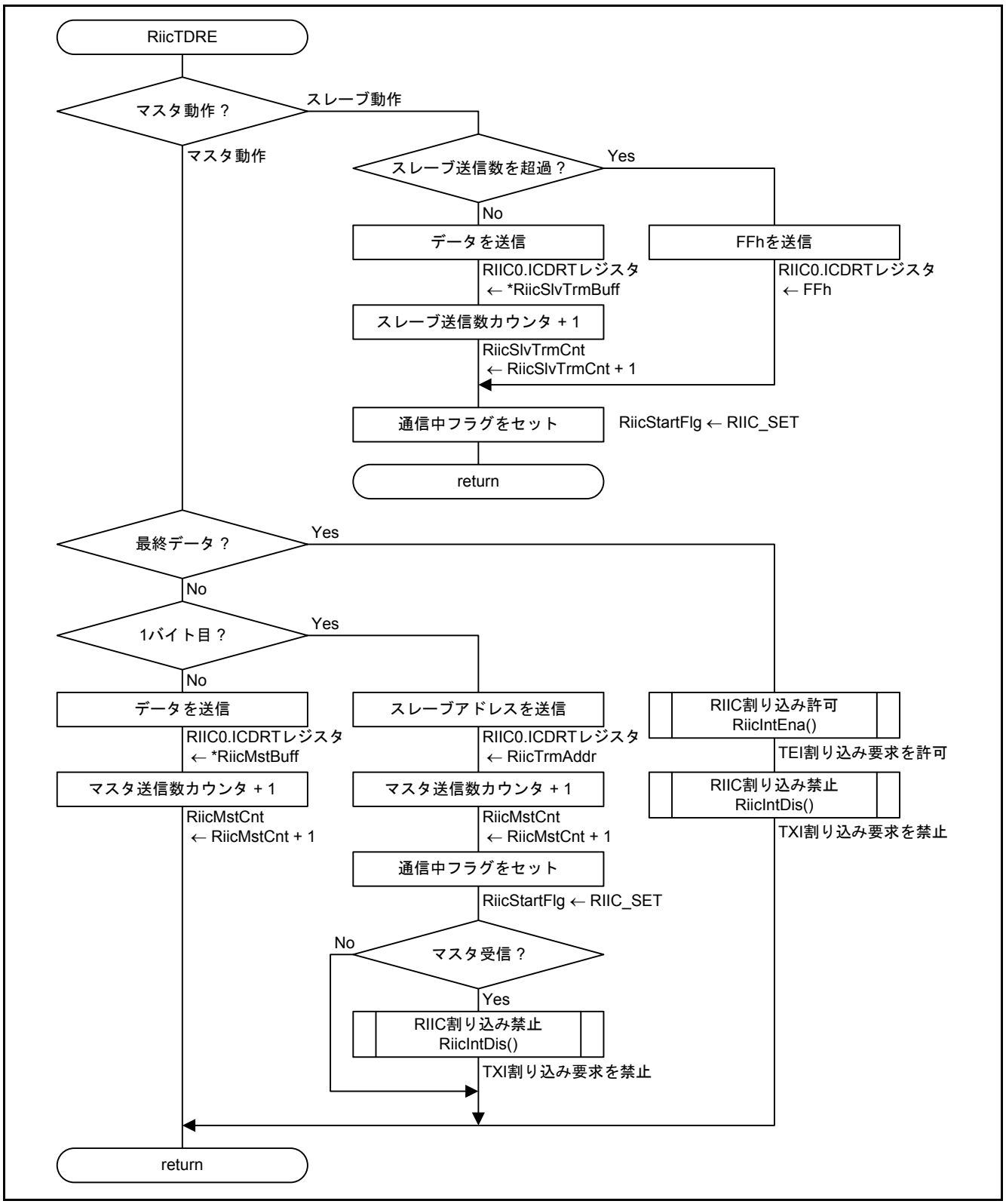


図5.18　送信データエンプティ割り込み

5.8.16　送信終了割り込み

図 5.19に送信終了割り込みのフローチャートを示します。

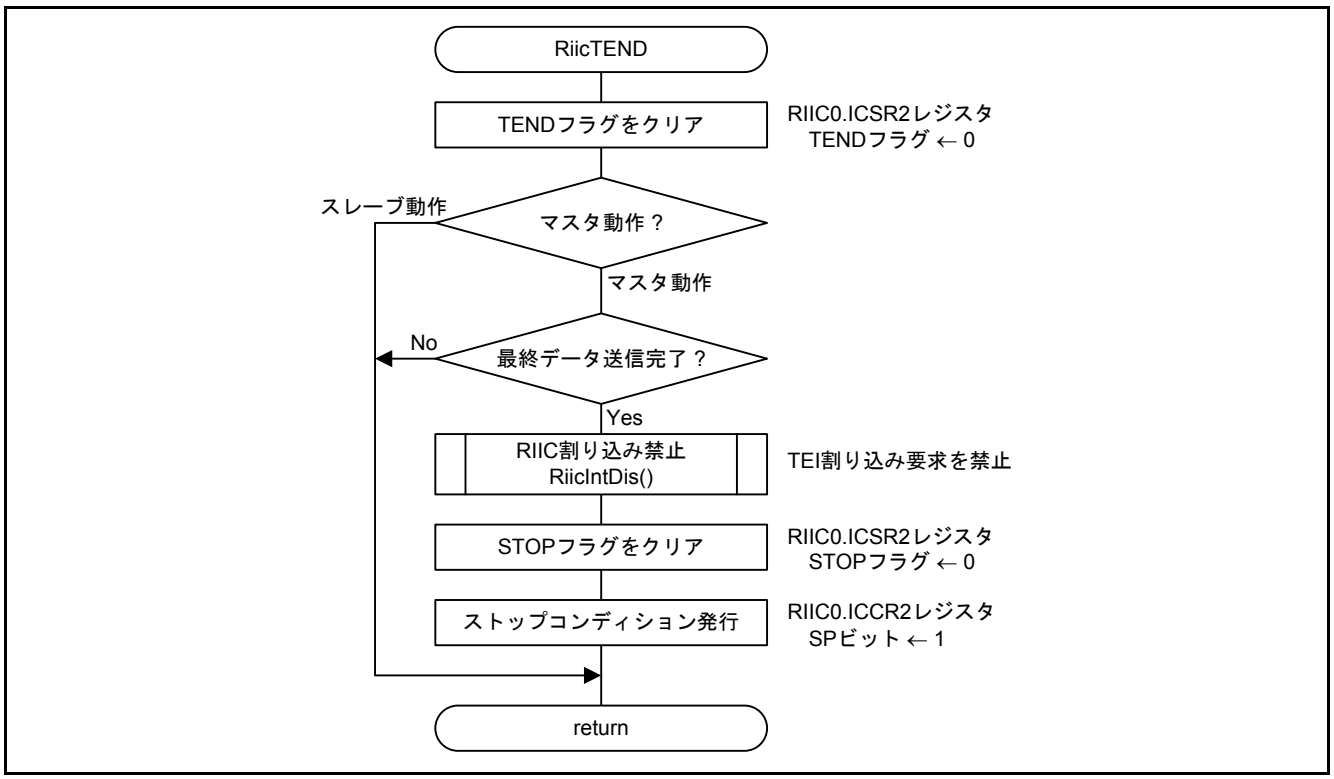


図5.19　送信終了割り込み

5.8.17　ストップコンディション検出割り込み

図 5.20にストップコンディション検出割り込みのフローチャートを示します。

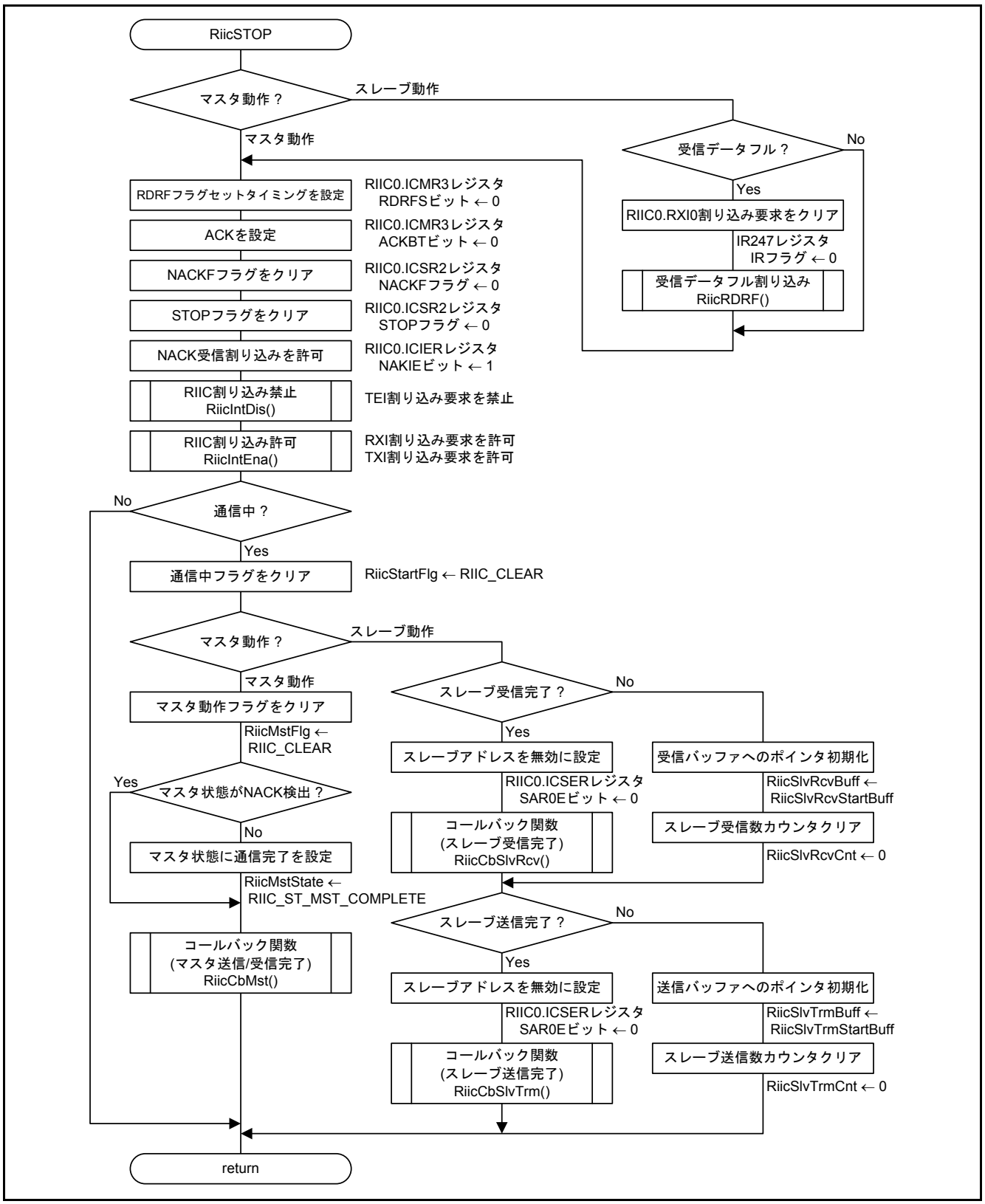


図5.20　ストップコンディション検出割り込み

5.8.18　NACK 検出割り込み

図 5.21にNACK 検出割り込みのフローチャートを示します。

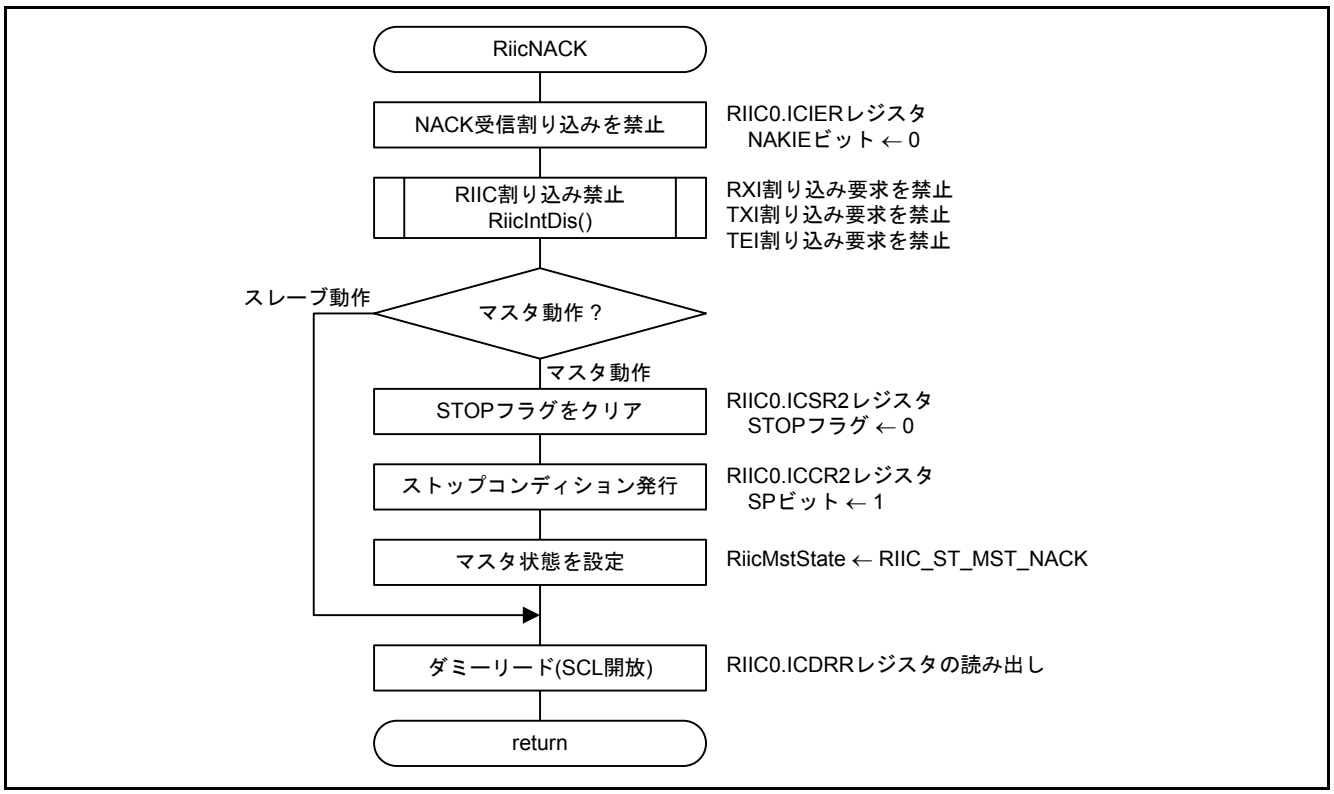


図5.21　NACK 検出割り込み

5.8.19　アービトレーションロスト検出割り込み

図 5.22にアービトレーションロスト検出割り込みのフローチャートを示します。

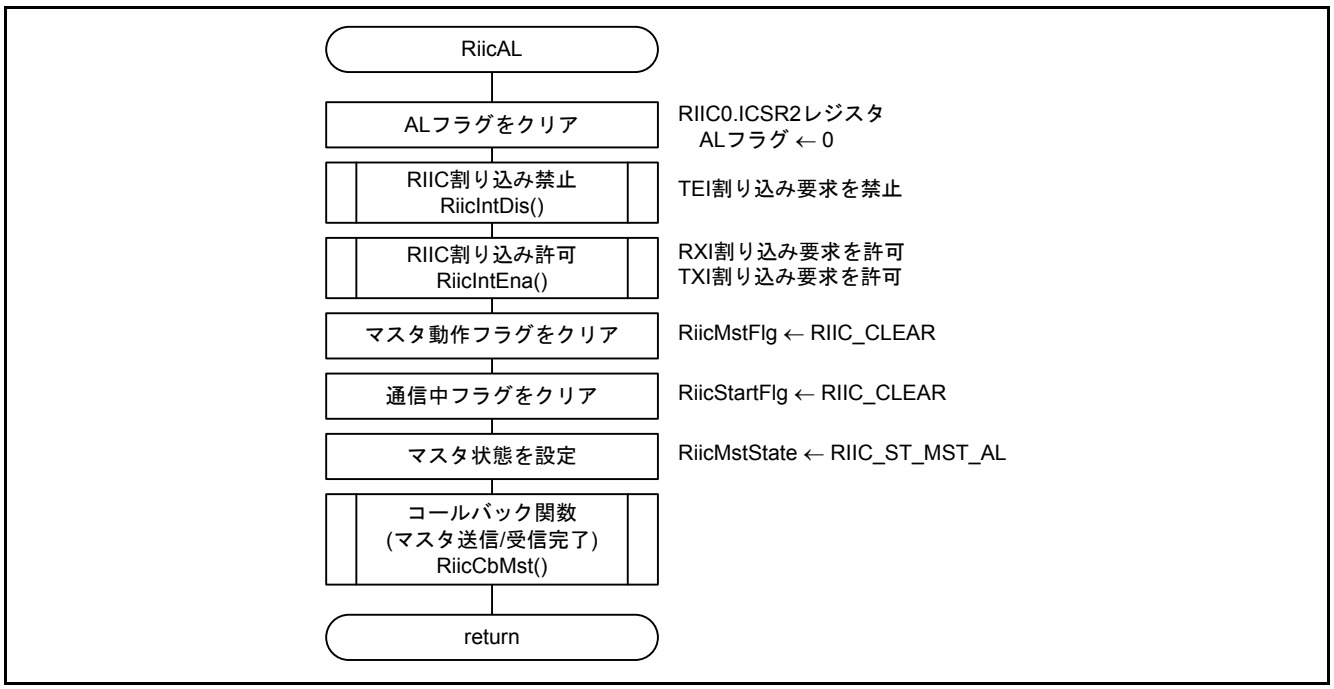


図5.22　アービトレーションロスト検出割り込み

5.8.20　タイムアウト検出割り込み

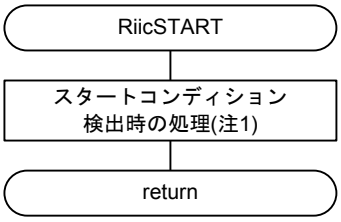
図 5.23にタイムアウト検出割り込みのフローチャートを示します。

注1.　サンプルコードでは処理を行っていません。必要に応じてプログラムを追加してください。

図5.23　タイムアウト検出割り込み

5.8.21　スタートコンディション検出割り込み

図 5.24にスタートコンディション検出割り込みのフローチャートを示します。



注1.　サンプルコードでは処理を行っていません。必要に応じてプログラムを追加してください。

図5.24　スタートコンディション検出割り込み

5.8.22　RIIC0.EEI0 割り込み処理

図 5.25にRIIC0.EEI0 割り込み処理のフローチャートを示します。

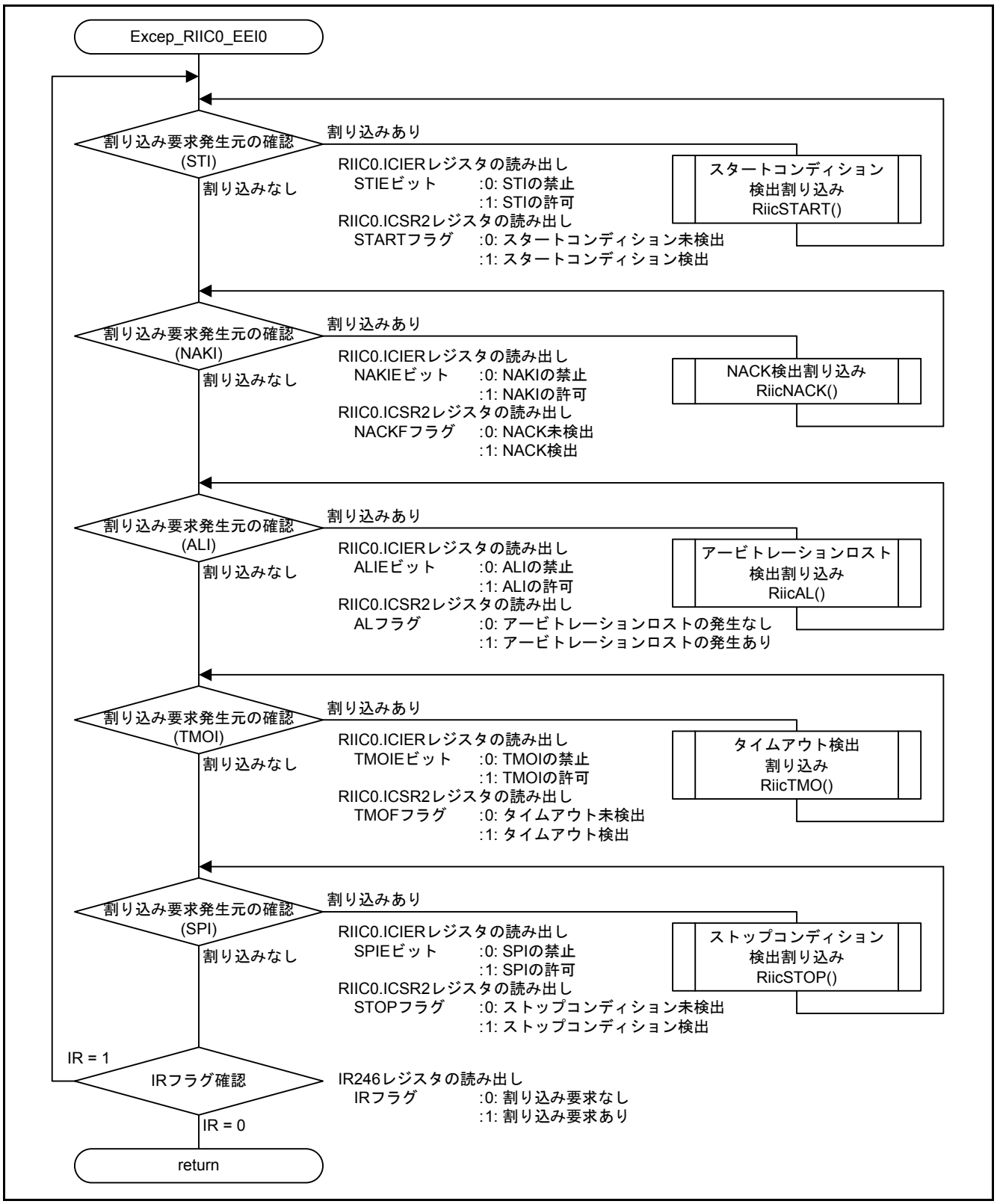


図5.25　RIIC0.EEI0 割り込み処理

5.8.23　RIIC0.RXI0 割り込み処理

図 5.26にRIIC0.RXI0 割り込み処理のフローチャートを示します。

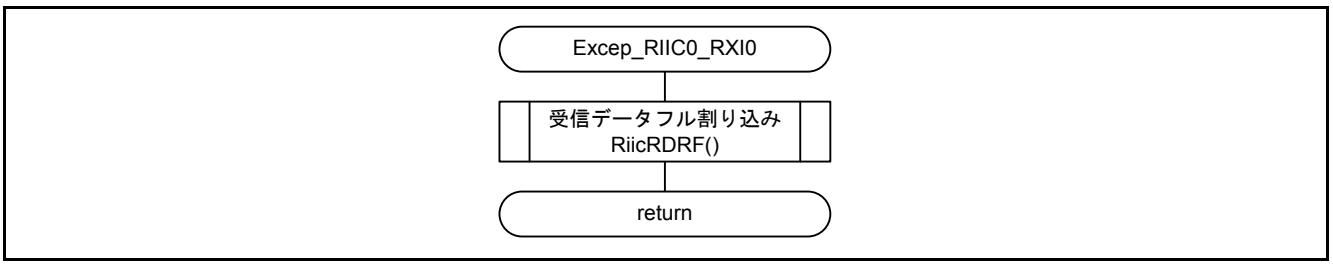


図5.26　RIIC0.RXI0 割り込み処理

5.8.24　RIIC0.TXI0 割り込み処理

図 5.27にRIIC0.TXI0 割り込み処理のフローチャートを示します。

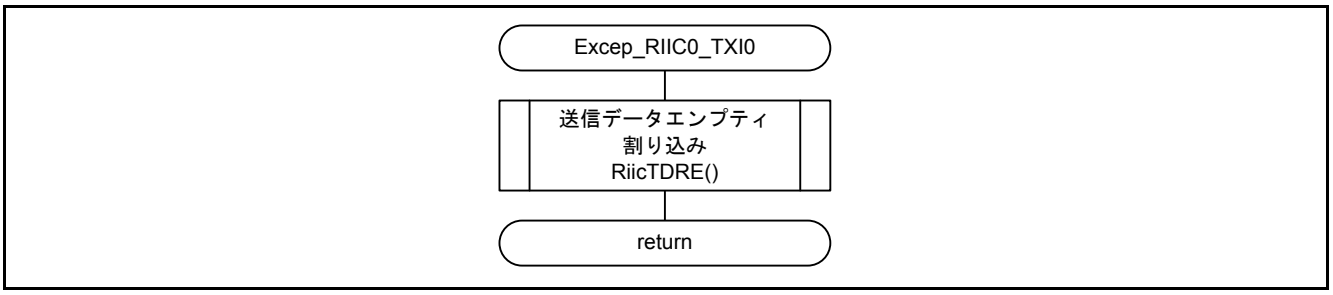


図5.27　RIIC0.TXI0 割り込み処理

5.8.25　RIIC0.TEI0 割り込み処理

図 5.28にRIIC0.TEI0 割り込み処理のフローチャートを示します。

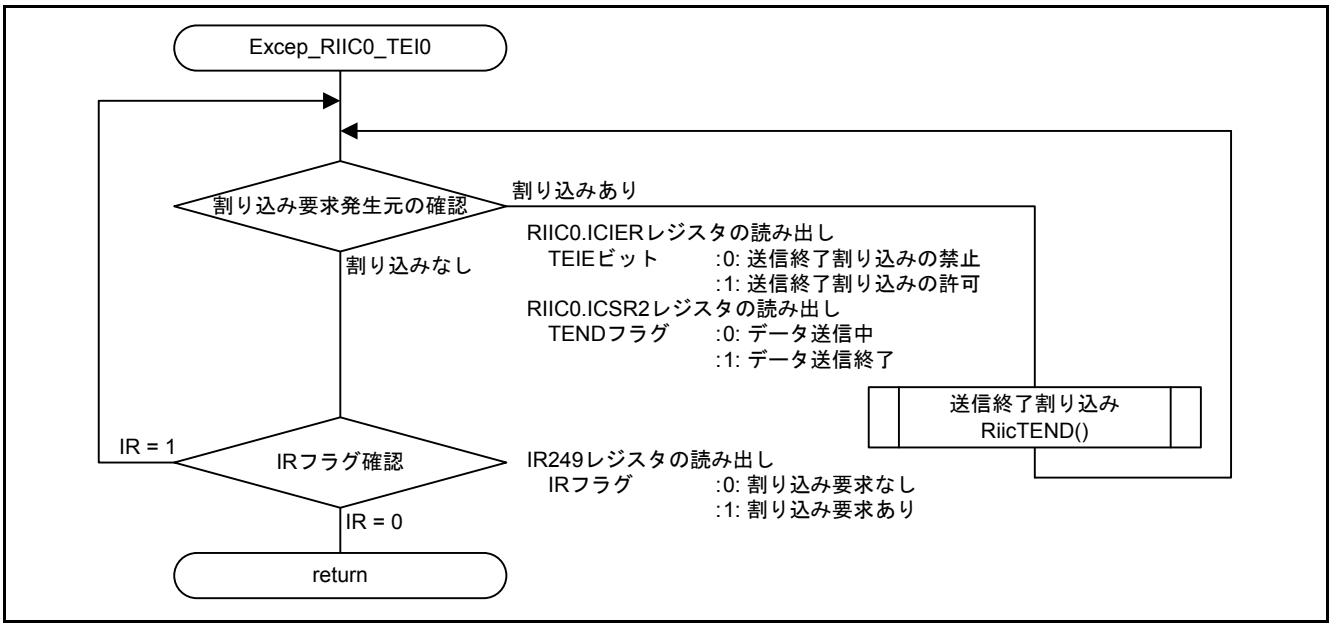


図5.28　RIIC0.TEI0 割り込み処理

6.　RX21A、RX220 グループ 初期設定例 アプリケーションノートとの組み合わせ方

本アプリケーションノートのサンプルコードは、RX210 グループで動作することを確認しています。RX21A グループや RX220 グループで動作させるには、それぞれの初期設定例のアプリケーションノートと組み合わせてください。

手順は、初期設定例のアプリケーションノート 「5. RX210 グループのアプリケーションノートを RX21A グループに適用する方法」、「4. RX210 グループのアプリケーションノートを RX220 グループに適用する方法」を参照ください。

7. サンプルコード

サンプルコードは、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。

8. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RX210グループ ユーザーズマニュアル ハードウェア編 Rev.1.50（R01UH0037JJ）
RX21A グループ ユーザーズマニュアル ハードウェア編 Rev.1.00（R01UH0251JJ）
RX220 グループ ユーザーズマニュアル ハードウェア編 Rev.1.10（R01UH0292JJ）
（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。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

RX ファミリ C/C++コンパイラパッケージ V.1.01 ユーザーズマニュアル Rev.1.00（R20UT0570JJ）
（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/contact/

改訂記録	RX210、RX21A、RX220 グループ RIIC を用いたマルチマスタ I2C-bus

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.03.01	—	初版発行
1.01	2014.07.01	1	対象デバイスに RX21A、RX220 グループを追加
		4	関連アプリケーションノートに RX21A、RX220 グループ 初期設定例のアプリケーションノートを追加
		15、16	参照するアプリケーションノートを各グループのアプリケーションノート初期設定例に変更
		41	RX21A、RX220 グループ 初期設定例と組み合わせる方法の参照先を追加
		42	参考ドキュメントに RX21A、RX220 グループのユーザーズマニュアルを追加

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. リザーブアドレスのアクセス禁止

【注意】リザーブアドレス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に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、事前に問題ないことをご確認下さい。

同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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。
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らず、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て、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事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　〒100-0004　千代田区大手町2-6-2（日本ビル）

■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/