SH7753 グループ
RIIC ドライバソフトウェア

R01AN1608JJ0101
Rev.1.01
2013.10.23

要旨

本仕様書は SH7753 グループの RIIC（I2C Bus Interface）ドライバについて説明します。

対象デバイス

SH7753

動作環境

本仕様書に示す RIIC ドライバの動作環境を以下に示します。

- 評価ボード : SH7753 グループ EVB ボード
- ソフトウェア : High-Performance Embedded Workshop - Ver 4.09.01.007
  : Toolchain - Ver 9.4.1.0
  : OptLinker - Ver 10.01.00
  : SH アセンブラ - Ver 7.01.02
  : SH C/C++コンパイラ- Ver 9.04.01
  : SH C/C++ライブラリジェネレータ - Ver 3.00.03
- エミュレータ : E10A エミュレータ Ver 3.03.00

目次

1. 機能概要

本ドライバは SH7753 に搭載されている RIIC を使用し、I2C インタフェースを持つデバイスと通信する為の関数群で構成されています。

1.1 主な機能

RIIC ドライバの主な機能を以下に示します。

・オペレーション設定、ステータスフラグ情報取得

・マスタ送信／受信

・スレーブ送信／受信

1.2 関連ドキュメント

SH7753 グループ　ユーザーズマニュアル：ハードウェア編

SH7753 グループ　アプリケーションノート　DMAC ドライバソフトウェア

2. ドライバ仕様

2.1 関数一覧

表 1に、RIIC ドライバの関数一覧を示します。

表 1. RIIC ドライバ関数一覧

分類	関数名	機能概要	スタック サイズ
設定関数	R_RIIC_SetOpr	オペレーション設定処理	76 バイト
	R_RIIC_GetStatus	ステータス情報取得処理	4 バイト
	R_RIIC_ClearStatusFlag	ステータスフラグクリア処理	36 バイト
通信関数	R_RIIC_Start	通信開始処理	16 バイト
	R_RIIC_TxDataSet	送信データ設定処理	4 バイト
	R_RIIC_RxDataGet	受信データ取得処理	4 バイト
	R_RIIC_GenerateStart	スタートコンディション生成処理	4 バイト
	R_RIIC_GenerateStop	ストップコンディション生成処理	4 バイト
	R_RIIC_NackEnable	NACK 生成有効処理	4 バイト
	R_RIIC_NackDisable	NACK 生成無効処理	4 バイト
	R_RIIC_Stop	通信停止処理	8 バイト
ISQ 通信関数	R_RIIC_ISQ_Start	ISQ 通信開始処理	16 バイト
	R_RIIC_ISQ_MasterTx	ISQ マスタ送信処理	100 バイト
	R_RIIC_ISQ_MasterRx	ISQ マスタ受信処理	104 バイト
	R_RIIC_ISQ_SlaveTx	ISQ スレーブ送信処理	36 バイト
	R_RIIC_ISQ_SlaveRx	ISQ スレーブ受信処理	28 バイト
	R_RIIC_ISQ_SlaveDetectDisable	ISQ スレーブアドレス検出無効処理	8 バイト
	R_RIIC_ISQ_SlaveDetectEnable	ISQ スレーブアドレス検出無効処理	16 バイト
	R_RIIC_ISQ_Stop	ISQ 通信停止処理	8 バイト
割り込みハンドラ関数	R_RIIC_TEIn_Interrupt(n=0 to 9)	送信終了割り込みハンドラ	4 バイト
	R_RIIC_EEIn_Interrupt(n=0 to 9)	イベント割り込みハンドラ	8 バイト

2.2 エラーコード一覧

表 2に RIIC ドライバのエラーコード一覧を示します。

表 2. RIIC ドライバエラーコード一覧

値	エラーコード（マクロ定義）	エラー内容
0	RET_NORMAL	正常終了
-1	RET_ERR_PARAM1	第 1 引数不正
-2	RET_ERR_PARAM2	第 2 引数不正
-3	RET_ERR_PARAM3	第 3 引数不正
-4	RET_ERR_PARAM4	第 4 引数不正

2.3 コールバック関数仕様一覧

表 3に RIIC ドライバのコールバック関数仕様一覧を示す。RIIC ドライバのコールバック関数は各ドライバ関数の処理完了時に通知としてコールする。

表 3. RIIC ドライバコールバック関数仕様一覧

関数名	コールバック名	仕様内容
R_RIIC_Start	void (*pv_event_ntf_callback)(char_t c_event_flag)	イベント通知コールバック （NULL の場合、コールバックなし）
R_RIIC_ISQ_Start	void (*pv_isq_event_ntf_callback)(char_t c_event_flag)	ISQ イベント通知コールバック
R_RIIC_ISQ_MasterTx	void (*pv_isq_mstr_tx_end_callback)(uchar_t uc_status, ulong_t ul_tx_size)	ISQ マスタ送信完了コールバック
R_RIIC_ISQ_MasterRx	void (*pv_ isq_mstr_rx_end_callback)(uchar_t uc_status, ulong_t ul_rx_size)	ISQ マスタ受信完了コールバック
R_RIIC_ISQ_SlaveTx	void (*pv_isq_slv_tx_end_callback)(uchar_t uc_status, ulong_t ul_tx_size)	ISQ スレーブ送信完了コールバック
R_RIIC_ISQ_SlaveRx	void (*pv_isq_slv_rx_end_callback)(uchar_t uc_status, ulong_t ul_rx_size)	ISQ スレーブ受信完了コールバック

3. 関数

本章では、RIIC ドライバの各関数仕様詳細を示します。各関数詳細の読み方は以下のとおりです。

関数名 機能概要	分類
書式	関数の呼び出し形式を示します。#include “ヘッダファイル”で示すヘッダファイルは、この関数の実行に必要な標準ヘッダファイルで、必ずインクルードします。
引数	I,O は、引数がそれぞれ入力データ、出力データであることを意味します。
戻り値	関数の戻り値を示します。
解説	関数の仕様について説明します。
注意	注意事項があればここに示します。

R_RIIC_SetOpr

設定関数

オペレーション設定処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_SetOpr( uchar_t uc_ch, riic_opr_t *pst_opr, riic_slave_ctrl_t *pst_slave_ctrl_t );
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
riic_opr_t *pst_opr	I	オペレーション設定処理管理構造体へのポインタ
riic_slave_ctrl_t *pst_slave_ctrl_t	I	スレーブ制御処理管理構造体へのポインタ ※マスタモードの設定時、NULL を設定してください

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正
RET_ERR_PARAM2	第 2 引数不正
RET_ERR_PARAM3	第 3 引数不正

解説

本関数は、オペレーション設定処理管理構造体及び、スレーブ制御処理管理構造体の情報に基づき、指定 RIIC チャネルのオペレーション設定を行います。

【RIIC オペレーション処理管理構造体】riic_opr_t

型	名称	I/O	機能説明
uchar_t	uc_smbs	I	バスモード設定（0：I2C、1：SMBus）
uchar_t	uc_cks	I	内部参照クロック分周 0：Pck／1 1：Pck／2 2：Pck／4 3：Pck／8 4：Pck／16 5：Pck／32 6：Pck／64 7：Pck／128
uchar_t	uc_brh	I	ビットレート High レベル周期（H’00～H’1F）
uchar_t	uc_brl	I	ビットレート Low レベル周期（H’00～H’1F）
uchar_t	uc_sddl	I	SDA 出力遅延カウンタ 0：遅延なし 1：1 サイクル 2：2 サイクル 3：3 サイクル 4：4 サイクル 5：5 サイクル 6：6 サイクル 7：7 サイクル
uchar_t	uc_dlcs	I	SDA 出力遅延クロックソース 0：内部参照クロック 1：内部参照クロックの 2 分周

uchar_t	uc_sdid	I	SDA 入力内部遅延[*1] 0：1 サイクル 1：2 サイクル 2：4 サイクル 3：遅延なし
uchar_t	uc_fmpe	I	ファストモードプラス（0：無効、1：有効）
uchar_t	uc_scle	I	SCL 同期回路[*2]（0：無効、1：有効）
uchar_t	uc_sale	I	スレーブアービトレーションロスト検出[*2] （0：無効、1：有効）
uchar_t	uc_nale	I	NACK 送信アービトレーションロスト検出[*2] （0：無効、1：有効）
uchar_t	uc_male	I	マスタ送信アービトレーションロスト検出[*2] （0：無効、1：有効）
uchar_t	uc_tmoe	I	タイムアウト機能[*2]（0：無効、1：有効）
uchar_t	uc_tmoh	I	タイムアウト H カウント制御[*2] 0：SCL ラインが HIGH レベル時、カウントが無効 1：SCL ラインが HIGH レベル時、カウントが有効
uchar_t	uc_tmol	I	タイムアウト L カウント制御[*2] 0：SCL ラインが LOW レベル時、カウントが無効 1：SCL ラインが LOW レベル時、カウントが有効
uchar_t	uc_tmos	I	タイムアウト検出タイムモード （0：LONG モード、1：SHORT モード）
uchar_t	uc_nfe	I	ノイズフィルタ（0：無効、1：有効）
uchar_t	uc_nf	I	ノイズフィルタステージ 0：1 ステージ 1：2 ステージ 2：3 ステージ 3：4 ステージ

[*1] ISQ マスタ／スレーブ送受信関数使用時は、遅延なし（’3’）を設定してください。

[*2] ISQ マスタ／スレーブ送受信関数使用時は、有効（’1’）を設定してください。

【RIIC スレーブ制御処理管理構造体】riic_slave_ctrl_t

型	名称	I/O	機能説明
uchar_t	uc_slave0_ctrl	I	スレーブ 0 制御[*3]（0：無効、1：7-bit、2：10-bit）
uchar_t	uc_slave1_ctrl	I	スレーブ 1 制御[*3]（0：無効、1：7-bit、2：10-bit）
uchar_t	uc_slave2_ctrl	I	スレーブ 2 制御[*3]（0：無効、1：7-bit、2：10-bit）
ushort_t	us_slave0_addr	I	スレーブ 0 アドレス[*4]
ushort_t	us_slave1_addr	I	スレーブ 1 アドレス[*4]
ushort_t	us_slave2_addr	I	スレーブ 2 アドレス[*4]

[*3] ISQ スレーブ送受信関数使用時は、10-bit（’2’）を設定しないでください。

[*4] 7-bit モード：bit[7:1] 、10-bit モード：bit[10:1] 、無効モード：H’0000。

注意　指定 RIIC チャネルが通信停止状態で、本関数をコールすること。

R_RIIC_GetStatus

設定関数

ステータス情報取得処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
long_t R_RIIC_GetStatus( uchar_t uc_ch );
```

引数			
	uchar_t uc_ch	I	チャネル番号（0～9）

戻り値		
	H’00000000～	ステータスフラグ情報
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルのステータスフラグビットの読み出し処理を行います。

なお、ISQ マスタ／スレーブ送受信中に、本関数をコールする場合、以下ステータスフラグの読み出し処理を行わない。

・MST、TRS、HOA、DID、GCA（読んだ場合、”0”が読めます）

【ステータスフラグ情報詳細】

bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24
0	SENDF	0	0	0	0	0	0
bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
BBSY	MST	TRS	-	-	-	-	-
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
TDRE	TEND	RDRF	NACKF	STOP	START	AL	TMOF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
HOA	-	DID	-	GCA	AAS2	AAS1	AAS0

“-”：リザーブビット。リードした値は不定。

注意

なし

R_RIIC_ClearStatusFlag

設定関数

ステータスフラグクリア処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ClearStatusFlag( uchar_t uc_ch, ulong_t ul_status_flag );
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
ulong_t ul_status_flag	I	ステータスフラグ

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、引数“ul_status_flag” の情報に基づき、指定 RIIC チャネルのステータスフラグビットのクリア処理を行います。

【ステータスフラグ情報詳細】

bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24
-	SENDF	-	-	-	-	-	-
bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
-	-	-	-	-	-	-	-
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
-	TEND	RDRF	NACKF	STOP	START	AL	TMOF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
HOA	-	DID	-	GCA	AAS2	AAS1	AAS0

“0” ：フラグクリアなし
“1” ：フラグクリアあり
“-” ：リザーブビット。“0”を設定してください。

注意

マスタ送受信中または、ISQ マスタ／スレーブ送受信中に本関数を使用しないこと。

R_RIIC_Start

通信関数

通信開始処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_Start( uchar_t uc_ch, void (*pv_event_ntf_callback )( char_t c_event_flag ) );
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
void (*pv_event_ntf_callback) (char_t c_event_flag)	I	イベント通知コールバックアドレス （NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの通信開始処理を行います。

本関数コール後､接続デバイスとの通信によりイベントを検出した場合､イベント情報を引数として、“pv_event_ntf_callback”に指定した定義関数をコールすることで通知します。

本関数をコール後、データ送受信関数（R_RIIC_TxDataSet、R_RIIC_RxDataGet）及び、DMAC データ転送関数（R_DMAC_SetOpr、R_DMAC_TransferStart）をコールすることで、接続デバイスとのデータ送受信ができます。

【イベント情報詳細】

イベント	引数値
スタートコンディション検出	H’01
ストップコンディション検出	H’02
NACK 検出	H’04
アービトレーションロスト検出	H’08
タイムアウト検出	H’10
送信終了検出	H’20

※イベント通知コールバック関数の NULL 設定で本関数コール後、ステータスフラグビット（TEND、START、STOP 、NACKF、AL、TMOF）をクリアまたは確認する場合、R_RIIC_ClearStatusFlag、R_RIIC_GetStatus関数を使用してください。

注意

指定 RIIC チャネルが通信停止状態で、本関数をコールすること。

本関数をコールする前に、R_RIIC_SetOpr関数をコールし、指定 RIIC チャネルのオペレーション設定を実行すること。

DMAC ドライバについては、「SH7753 グループ アプリケーションノート DMAC ドライバソフトウェア 」を参照のこと。

R_RIIC_TxDataSet

通信関数

送信データ設定処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_TxDataSet(uchar_t uc_ch, uchar_t uc_data );
```

引数			
	uchar_t uc_ch	I	チャネル番号（0～9）
	uchar_t uc_data	I	送信データ

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの送信データ設定処理を行います。
引数”uc_data”に指定したデータを送信データレジスタ（ICDRT）へ設定します。
本関数コール後、データの送信完了を確認する場合、イベント通知コールバック関数の送信終了検出イベント（H’20）または、R_RIIC_GetStatusを使用し、送信データエンプティ（TDRE）／送信終了（TEND）状態を確認してください。

注意

本関数をコールする前に、R_RIIC_Start関数をコールし、指定 RIIC チャネルを通信状態にしておくこと。
指定 RIIC チャネルが送信データエンプティ状態（TDRE = 1）または、送信終了（TEND = 1）状態で、本関数を使用してください。

R_RIIC_RxDataGet

通信関数

受信データ取得処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
uchar_t R_RIIC_RxDataGet( uchar_t uc_ch );
```

引数			
	uchar_t uc_ch	I	チャネル番号（0～9）

戻り値		
	H’00～H’FF	受信データ

解説

本関数は、指定 RIIC チャネルの受信データ取得処理を行います。
受信データレジスタ（ICDRR）よりデータを取得し、戻り値として通知します。

注意

本関数をコールする前に、R_RIIC_Start関数をコールし、指定 RIIC チャネルを通信状態にしておくこと。
指定 RIIC チャネルが受信データフル状態（RDRF = 1 ）で、本関数を使用してください。
本関数では引数のチェックは実行しない。

R_RIIC_GenerateStart

通信関数

スタートコンディション生成処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_GenerateStart( uchar_t uc_ch );
```

引数			
	uchar_t uc_ch	I	チャネル番号（0～9）

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルから接続スレーブデバイスへのスタートコンディション生成処理を行います。なお、指定 RIIC チャネルがバスビジー状態（BBSY=1））場合は、リスタートコンディション生成処理を行います。

本関数コール後、スタートコンディション生成完了を確認する場合、イベント通知コールバック関数のスタートコンディション検出イベント（H’01）または、R_RIIC_GetStatusをコールし、スタートコンディション検出（START）状態を確認してください。

注意

指定 RIIC チャネルが通信開始状態で、本関数をコールすること。

R_RIIC_GenerateStop

通信関数

ストップコンディション生成処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_GenerateStop( uchar_t uc_ch );
```

引数			
	uchar_t uc_ch	I	チャネル番号（0～9）

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルから接続スレーブデバイスへのストップコンディション生成処理を行います。

本関数コール後、ストップコンディション生成完了を確認する場合、イベント通知コールバック関数のストップコンディション検出イベント（H’02）または、R_RIIC_GetStatusをコールし、ストップコンディション検出（STOP）状態を確認してください。

注意

指定 RIIC チャネルが通信開始状態で、本関数をコールすること。

R_RIIC_NackEnable

通信関数

NACK 生成有効設定処理

書式
```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_NackEnable( uchar_t uc_ch );
```

引数				
	uchar_t uc_ch	I	チャネル番号（0～9）	

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説 本関数は、指定 RIIC チャネルの NACK 生成有効設定処理を行います。
本関数コール後、データ受信毎に NACK を生成します。

注意 指定 RIIC チャネルが通信開始状態及び、受信モード状態で、本関数をコールすること。

R_RIIC_NackDisable

通信関数

NACK 生成無効設定処理

書式
```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_NackDisable( uchar_t uc_ch );
```

引数				
	uchar_t uc_ch	I	チャネル番号（0～9）	

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説 本関数は、指定 RIIC チャネルの NACK 生成無効設定処理を行います。

注意 指定 RIIC チャネルが通信開始状態及び、受信モード状態で、本関数をコールすること。

R_RIIC_Stop

通信関数

通信停止処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_Stop( uchar_t uc_ch );
```

引数				
	uchar_t uc_ch	I	チャネル番号（0～9）	

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの通信停止処理を行います。
データ通信の終了及び、中断する時、本関数を使用してください。

注意

指定 RIIC チャネルが通信状態で、本関数をコールすること。

R_RIIC_ISQ_Start

ISQ 通信関数

ISQ 通信開始処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_Start( uchar_t uc_ch, void (*pv_isq_event_ntf_callback )
( char_t c_event_flag ) );
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
void (*pv_isq_event_ntf_callback) (char_t c_event_flag)	I	イベント通知コールバックアドレス （NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ 通信開始処理を行います。

本関数コール後､接続デバイスとの通信によりイベントを検出した場合､イベント情報を引数として、“pv_isq_event_ntf_callback”に指定した定義関数をコールすることで通知します。

本関数をコール後、ISQ 送受信関数（R_RIIC_ISQ_MasterTx、R_RIIC_ISQ_MasterRx、R_RIIC_ISQ_SlaveTx、R_RIIC_ISQ_SlaveRx）をコールすることで、接続デバイスとのデータ送受信ができます。

【イベント情報詳細】

イベント	引数値
スタートコンディション検出	H’01
ストップコンディション検出	H’02
NACK 検出	H’04
アービトレーションロスト検出	H’08
タイムアウト検出	H’10
送信終了検出	H’20

※イベント通知コールバック関数の NULL 設定で本関数コール後、ステータスフラグビットを確認またはクリアする場合、R_RIIC_GetStatus、R_RIIC_ClearStatusFlag、関数を使用してください。

注意

指定 RIIC チャネルが通信停止状態で、本関数をコールすること。

本関数をコールする前に、R_RIIC_SetOpr関数をコールし、指定 RIIC チャネルのオペレーション設定を実行すること。

DMAC ドライバについては、「SH7753 グループ　アプリケーションノート　DMAC ドライバソフトウェア 」を参照のこと。

R_RIIC_ISQ_MasterTx

ISQ 通信関数

ISQ マスタ送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_MasterTx(uchar_t uc_ch, riic_mstr_ctrl_t *pst_mstr_tx,
void (*pv_isq_mstr_tx_end_callback )( uchar_t uc_status, ulong_t ul_tx_size ));
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
riic_mstr_ctrl_t *pst_mstr_tx	I	マスタ送受信制御管理構造体へのポインタ
void (*pv_isq_mstr_tx_end_callback) (uchar_t uc_status, ulong_t ul_tx_size)	I	ISQ マスタ送信完了時のコールバックアドレス（NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ マスタ送信処理を行います。

スタートコンディション生成※、スレーブアドレス＋ライトビットの送信を実行し、データ格納領域の先頭アドレスから、データサイズ数を送信します。データ送信完了後、ストップコンディション生成（ストップコンディション生成有効の場合）を発行し、送信済みデータ数及びステータス情報を引数として、“pv_isq_mstr_tx_end_callback”に指定した定義関数をコールすることで通知します。

本関数はデータ送信に汎用 DMAC を使用するため、本関数をコール前に、DMAC データ転送関数（R_DMAC_SetOpr、R_DMAC_TransferStart）を実行してください。

※前回の送受信処理でストップコンディションが生成されていない（バスビジー状態（BBSY=1））場合は、リスタートコンディションを生成します。

【マスタ送受信制御管理構造体】riic_mstr_ctrl_t

型	名称	I/O	機能説明
uchar_t	uc_slave_addr_length	I	スレーブアドレス幅（0 : 7-bit 固定）
ushort_t	us_slave_addr	I	送信先スレーブアドレス（bit[7:1]） ※その他ビットは、“0”を設定してください。
uchar_t *	puc_data_addr	I	送信データ格納領域アドレス*
ulong_t	ul_data_size	I	送信データサイズ*
uchar_t	uc_stop_generate	I	ストップコンディション生成（0 : 無効、1 : 有効）

（*）使用 DMAC チャネルで設定する値と同じ値を設定してください。

【ステータス情報詳細】

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	SENDF	0	TMOF	AL	0	0	0

※ISQ マスタ送信完了コールバック関数の NULL 設定で本関数コール後、ISQ 送受信完了（SENDF = 1）を確認する場合、R_RIIC_GetStatus関数を使用してください。また、ISQ マスタ送信完了後に、R_RIIC_ISQ_Stop関数をコールし、指定 RIIC チャネルの通信を停止しください。

注意 本関数は、10-bit モードのスレーブアドレスをサポートしていない。
本関数をコールする前に、R_RIIC_ISQ_Start関数をコールし、指定 RIIC チャネルを通信状態にしておくこと。
本関数コール前から、送信処理の完了まで、指定 RIIC チャネルの TXI と RXI 割り込みを割り込み要因禁止状態にしておくこと。
DMAC ドライバについては、「SH7753 グループ DMAC ドライバソフトウェア編」を参照のこと。

R_RIIC_ISQ_MasterRx

ISQ 通信関数

ISQ マスタ受信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_MasterRx( riic_mstr_ctrl_t *pst_mstr_rx,
void (*pv_isq_mstr_rx_end_callback )( uchar_t uc_status, ulong_t ul_rx_size) );
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
riic_mstr_ctrl_t *pst_mstr_rx	I	マスタ送受信制御管理構造体へのポインタ
void (*pv_isq_mstr_rx_end_callback) (uchar_t uc_status ,ulong_t ul_rx_size)	I	ISQ マスタ受信完了時のコールバックアドレス（NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ マスタ受信処理を行います。

スタートコンディション生成※、スレーブアドレス＋リードビット送信を実行し、データ格納領域の先頭アドレスから、受信データを格納します。データ受信完了後、NACK 生成とストップコンディション生成を実行し、受信済みデータ数及びステータス情報を引数として、“pv_isq_mstr_rx_end_callback”に指定した定義関数をコールすることで通知します。

本関数はデータ受信に汎用 DMAC を使用するため、本関数をコール前に、DMAC データ転送関数（R_DMAC_SetOpr、R_DMAC_TransferStart）を実行してください。

※前回の送受信処理でストップコンディションが生成されていない（バスビジー状態（BBSY=1））場合は、リスタートコンディションを生成します。

【マスタ送受信制御管理構造体】riic_mstr_ctrl_t

型	名称	I/O	機能説明
uchar_t	uc_slave_addr_length	I	スレーブアドレス幅（0 : 7-bit 固定）
ushort_t	us_slave_addr	I	受信先スレーブアドレス（bit[7:1]） ※その他ビットは、“0”を設定してください。
uchar_t *	puc_data_addr	O	受信データ格納領域アドレス*
ulong_t	ul_data_size	I	受信データサイズ*
uchar_t	uc_stop_generate	I	ストップコンディション生成（1 : 有効固定）

（*）使用 DMAC チャネルで設定する値と同じ値を設定してください。

【ステータス情報詳細】

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	SENDF	0	TMOF	AL	0	0	0

※ISQ マスタ受信完了コールバック関数の NULL 設定で本関数コール後、ISQ 送受信完了（SENDF = 1）を確認する場合、R_RIIC_GetStatus関数を使用してください。また、ISQ マスタ受信完了後に、R_RIIC_ISQ_Stop関数をコールし、指定 RIIC チャネルの通信を停止しください。

注意 本関数は、10-bit モードのスレーブアドレスをサポートしていない。
本関数をコールする前に、R_RIIC_ISQ_Start関数をコールし、指定 RIIC チャネルを通信状態にしておくこと。
本関数コール前から、受信処理の完了まで、指定 RIIC チャネルの TXI と RXI 割り込みを割り込み要因禁止状態にしておくこと。
DMAC ドライバについては、「SH7753 グループ DMAC ドライバソフトウェア編」を参照のこと。

R_RIIC_ISQ_SlaveTx

ISQ 通信関数

ISQ スレーブ送信処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_SlaveTx( uchar_t uc_ch, ulong_t ul_data_size,
  void (*pv_isq_slv_tx_end_callback )( uchar_t uc_status, ulong_t ul_tx_size));
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
ulong_t ul_data_size	I	送信データサイズ （使用 DMAC チャネルで設定する転送バイト数と同じ値を設定してください）
void (*pv_isq_slv_tx_end_callback) (uchar_t uc_status, ulong_t ul_tx_size)	I	ISQ スレーブ送信完了時のコールバックアドレス （NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ スレーブ送信処理を行います。

データ格納領域の先頭アドレスから、データサイズ数を送信し、データ送信後に、送信済みデータ数及びステータス情報を引数として、“pv_isq_slv_tx_end_callback”に指定した定義関数をコールすることで通知します。

本関数コール後、接続マスタデバイスが送信データ数を上回るデータが要求された場合、指定 RIIC チャネルの SCL 端子を”Low”に保持し、タイムアウトを検出します。

本関数はデータ送信に汎用 DMAC を使用するため、本関数をコール前に、DMAC データ転送関数（R_DMAC_SetOpr、R_DMAC_TransferStart）を実行してください。

【ステータス情報詳細】

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	SENDF	0	TMOF	AL	0	0	0

※ISQ スレーブ送信完了コールバック関数の NULL 設定で本関数コール後、ISQ 送受信完了（SENDF = 1）を確認する場合、R_RIIC_GetStatus関数を使用してください。また、ISQ スレーブ送信完了後に、R_RIIC_ISQ_Stop関数をコールし、指定 RIIC チャネルの通信を停止しください。

注意

本関数は、10-bit モードのスレーブアドレスをサポートしていない。

本関数をコールする前に、R_RIIC_ISQ_Start関数をコールし、指定 RIIC チャネルを通信状態にしておくこと。また、接続マスタデバイスからスタートコンディションを検出し、アドレスマッチ検出（AAS0/1/2 = 1）及びスレーブ送信状態（MST = 0, TRS = 1）で、本関数を使用すること。また、R_RIIC_ISQ_SlaveDetectDisable関数をコールし、スレーブアドレス検出を無効にしておくこと。

本関数コール前から、送信処理の完了まで、指定 RIIC チャネルの TXI と RXI 割り込みを割り込み要因禁止状態にしておくこと。

DMAC ドライバについては、「SH7753 グループ DMAC ドライバソフトウェア編」を参照のこと。

R_RIIC_ISQ_SlaveRx

ISQ 通信関数

ISQ スレーブ受信処理

書式

```
#include  “r_common.h”
#include  “iodefine.h”
#include  “r_riic_if.h”
char_t R_RIIC_ISQ_SlaveRx( uchar_t uc_ch, ulong_t ul_data_size,
  void (*pv_isq_slv_rx_end_callback )( uchar_t uc_status, ulong_t ul_rx_size));
```

引数

uchar_t uc_ch	I	チャネル番号（0～9）
ulong_t ul_data_size	I	受信データサイズ （使用 DMAC チャネルで設定する転送バイト数と同じ値を設定してください）
void (*pv_isq_slv_rx_end_callback) (uchar_t uc_status, ulong_t ul_rx_size)	I	ISQ スレーブ受信完了時のコールバックアドレス （NULL の場合、コールバックなし）

戻り値

RET_NORMAL	正常終了
RET_ERR_PARAM1	第 1 引数不正
RET_ERR_PARAM2	第 2 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ スレーブ受信処理を行います。

データ格納領域の先頭アドレスから、データサイズ数を受信し、受信完了後に受信済みデータ数及びステータス情報を引数として、“pv_isq_slv_rx_end_callback”に指定した定義関数をコールすることで通知します。

本関数コール後、接続マスタデバイスが受信データ数を上回るデータが要求された場合、指定 RIIC チャネルの SCL 端子を”Low”に保持し、タイムアウトを検出します。

本関数はデータ送信に汎用 DMAC を使用するため、本関数をコール前に、DMAC データ転送関数（R_DMAC_SetOpr、R_DMAC_TransferStart）を実行してください。

【ステータス情報詳細】

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	SENDF	0	TMOF	AL	0	0	0

※ISQ スレーブ受信コールバック関数の NULL 設定で本関数コール後、ISQ 送受信完了（SENDF = 1）を確認する場合、R_RIIC_GetStatus関数を使用してください。また、ISQ スレーブ受信完了後に、R_RIIC_ISQ_Stop関数をコールし、指定 RIIC チャネルの通信を停止しください。

注意

本関数は、10-bit モードのスレーブアドレスをサポートしていない。

本関数をコールする前に、R_RIIC_ISQ_Start関数をコールし、指定 RIIC チャネルを通信状態にしておくこと。また、接続マスタデバイスからスタートコンディションを検出し、アドレスマッチ検出（AAS0/1/2 = 1）及びスレーブ受信状態（MST = 0, TRS = 0）で、本関数を使用すること。また、R_RIIC_ISQ_SlaveDetectDisable関数をコールし、スレーブアドレス検出を無効にしておくこと。

本関数コール前から、受信処理の完了まで、指定 RIIC チャネルの TXI と RXI 割り込みを割り込み要因禁止状態にしておくこと。

DMAC ドライバについては、「SH7753 グループ DMAC ドライバソフトウェア編」を参照のこと。

R_RIIC_ISQ_SlaveDetectDisable

ISQ 通信関数

ISQ スレーブアドレス検出無効処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_SlaveDetectDisable( uchar_t uc_ch );
```

引数				
	uchar_t uc_ch		I	チャネル番号（0～9）

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ スレーブアドレス検出無効処理を行う。
ISQ スレーブ送受信の開始前に、本関数を使用してください。ISQ スレーブ送受信完了後、スレーブアドレス検出を有効する場合、R_RIIC_ISQ_SlaveDetectEnable関数を使用してください。

注意

指定 RIIC チャネルが接続マスタデバイスからスタートコンディションを検出し、アドレスマッチ検出（AAS0/1/2 = 1）及びスレーブ状態（MST = 0）で、本関数を使用すること。

R_RIIC_ISQ_SlaveDetectEnable

ISQ 通信関数

ISQ スレーブアドレス検出有効処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_SlaveDetectEnable( uchar_t uc_ch );
```

引数				
	uchar_t uc_ch		I	チャネル番号（0～9）

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ スレーブアドレス検出有効処理を行う。
ISQ スレーブ送受信の完了後に、本関数を使用してください。

注意

指定 RIIC チャネルが通信状態、及びバスアイドル状態（BBSY = 0）で、本関数を使用すること。

R_RIIC_ISQ_Stop

ISQ 通信関数

ISQ 通信停止処理

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
char_t R_RIIC_ISQ_Stop( uchar_t uc_ch );
```

引数			
	uchar_t uc_ch	I	チャネル番号（0～9）

戻り値		
	RET_NORMAL	正常終了
	RET_ERR_PARAM1	第 1 引数不正

解説

本関数は、指定 RIIC チャネルの ISQ 通信停止処理を行います。
ISQ データ通信の終了及び、中断する時、本関数を使用してください。

注意

指定 RIIC チャネルが ISQ 通信状態で、本関数をコールすること。

R_RIIC_TEIn_Interrupt(n=0 to 9)

割り込みハンドラ関数

送信終了割り込みハンドラ

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
void R_TEIn_Interrupt( void );
```

引数　なし

戻り値　なし

解説　本関数は、TEIn（n=0 to 9）割り込み内での処理を行います。
TEIn 割り込みハンドラから、本割り込みハンドラ関数をコールすることにより、R_RIIC_Start関数で指定したイベント通知コールバック関数をコールし、送信終了イベントを通知します。

注意　本関数は、TEIn 割り込みのハンドラ内からコールすること。

R_RIIC_EEIn_Interrupt(n=0 to 9)

割り込みハンドラ関数

イベント割り込みハンドラ

書式

```
#include “r_common.h”
#include “iodefine.h”
#include “r_riic_if.h”
void R_RIIC_EEIn_Interrupt( void );
```

引数　なし

戻り値　なし

解説　本関数は、EEIn（n=0 to 9）割り込み内での処理を行います。
EEIn 割り込みハンドラから、本割り込みハンドラ関数をコールすることにより、R_RIIC_Start関数で指定したイベント通知コールバック関数をコールし、該当イベント情報（スタート／ストップコンディション検出、NACK 検出、アービトレーションロスト検出、タイムアウト検出）を通知します。また、ISQ 通信関数で指定した ISQ 送受信完了コールバック関数をコールし、ISQ 送受信完了を通知します。

注意　本関数は、EEIn 割り込みのハンドラ内からコールすること。

4. 動作条件

4.1 ポートについて

本デバイスドライバ使用時は、使用する RIIC チャネルのポートの有効状態で使用すること。表 4 に有効の設定が必要ポートを示します。

表 4. ポート設定一覧

レジスタ	ビット	設定
PMCR	PM0MD	00 : SCL7 機能を設定します
	PM1MD	00 : SDA7 機能を設定します
	PM2MD	00 : SCL6 機能を設定します
	PM3MD	00 : SDA6 機能を設定します
PRCR	PR0MD	00 : SCL0 機能を設定します
	PR1MD	00 : SDA0 機能を設定します
	PR2MD	00 : SCL1 機能を設定します
	PR3MD	00 : SDA1 機能を設定します
	PR4MD	00 : SCL2 機能を設定します
	PR5MD	00 : SDA2 機能を設定します
	PR6MD	00 : SCL8 機能を設定します
	PR7MD	00 : SDA8 機能を設定します
PSCR	PS0MD	00 : SCL3 機能を設定します
	PS1MD	00 : SDA3 機能を設定します
	PS2MD	00 : SCL4 機能を設定します
	PS3MD	00 : SDA4 機能を設定します
	PS4MD	00 : SCL5 機能を設定します
	PS5MD	00 : SDA5 機能を設定します
	PS6MD	00 : SCL9 機能を設定します
	PS7MD	00 : SDA9 機能を設定します

注. 上記設定と併せて、H'FFEC 006E 番地に H'0C を設定してください。

4.2 割り込みについて

本デバイスドライバ時は、使用 RIIC チャネルの割り込み要因許可状態で使用すること。また、割り込みベクタテーブルに使用 RIIC チャネルの各割り込みハンドラ関数を登録すること。表 1に登録が必要な割り込みハンドラ関数を示します。なお、使用 RIIC チャネルの TXI と RXI の割り込みについては、割り込み要因禁止状態に設定してください。

表 5. 割り込みベクタテーブル設定関数一覧

割り込み要因番号	割り込み要因	割り込みハンドラ関数
H'1420	TEI0	R_RIIC_TEI0_Interrupt
H'1460	EEI0	R_RIIC_EEI0_Interrupt
H'14E0	TEI1	R_RIIC_TEI1_Interrupt
H'1520	EEI1	R_RIIC_EEI1_Interrupt
H'1560	TEI2	R_RIIC_TEI2_Interrupt
H'1600	EEI2	R_RIIC_EEI2_Interrupt
H'1640	TEI3	R_RIIC_TEI3_Interrupt
H'1700	EEI3	R_RIIC_EEI3_Interrupt
H'1800	TEI4	R_RIIC_TEI4_Interrupt
H'1840	EEI4	R_RIIC_EEI4_Interrupt
H'1880	TEI5	R_RIIC_TEI5_Interrupt
H'18C0	EEI5	R_RIIC_EEI5_Interrupt
H'1900	TEI6	R_RIIC_TEI6_Interrupt
H'1980	EEI6	R_RIIC_EEI6_Interrupt
H'1A00	TEI7	R_RIIC_TEI7_Interrupt
H'1A40	EEI7	R_RIIC_EEI7_Interrupt
H'1A80	TEI8	R_RIIC_TEI8_Interrupt
H'1B40	EEI8	R_RIIC_EEI8_Interrupt
H'1B80	TEI9	R_RIIC_TEI9_Interrupt
H'1C20	EEI9	R_RIIC_EEI9_Interrupt

4.3 使用メモリ領域について

本デバイスドライバで使用するメモリ領域を表 6に示します。

表 6. メモリ使用量一覧表

内容	セクション	属性	バイト数	備考
プログラムコード	P	code, align=4	12226 バイト	
定数データ	C	data, align=4	0 バイト	
初期値ありデータ	D	data, align=4	0 バイト	
初期値なしデータ	B	data, align=4	132 バイト	

5. 注意事項

下記の RIIC 仕様に対して、本デバイスドライバでは非サポートであります。

- ホストアドレス検出（HOA）
- デバイス ID コマンド検出（DID）
- ジェネラルコールアドレス検出（GCA）

6. 付録

6.1 ドライバ関数の使用例

RIIC ドライバの使用例を図 1～図 6に示します。

6.1.1 マスタ送信時の使用例（10bit アドレスモード）

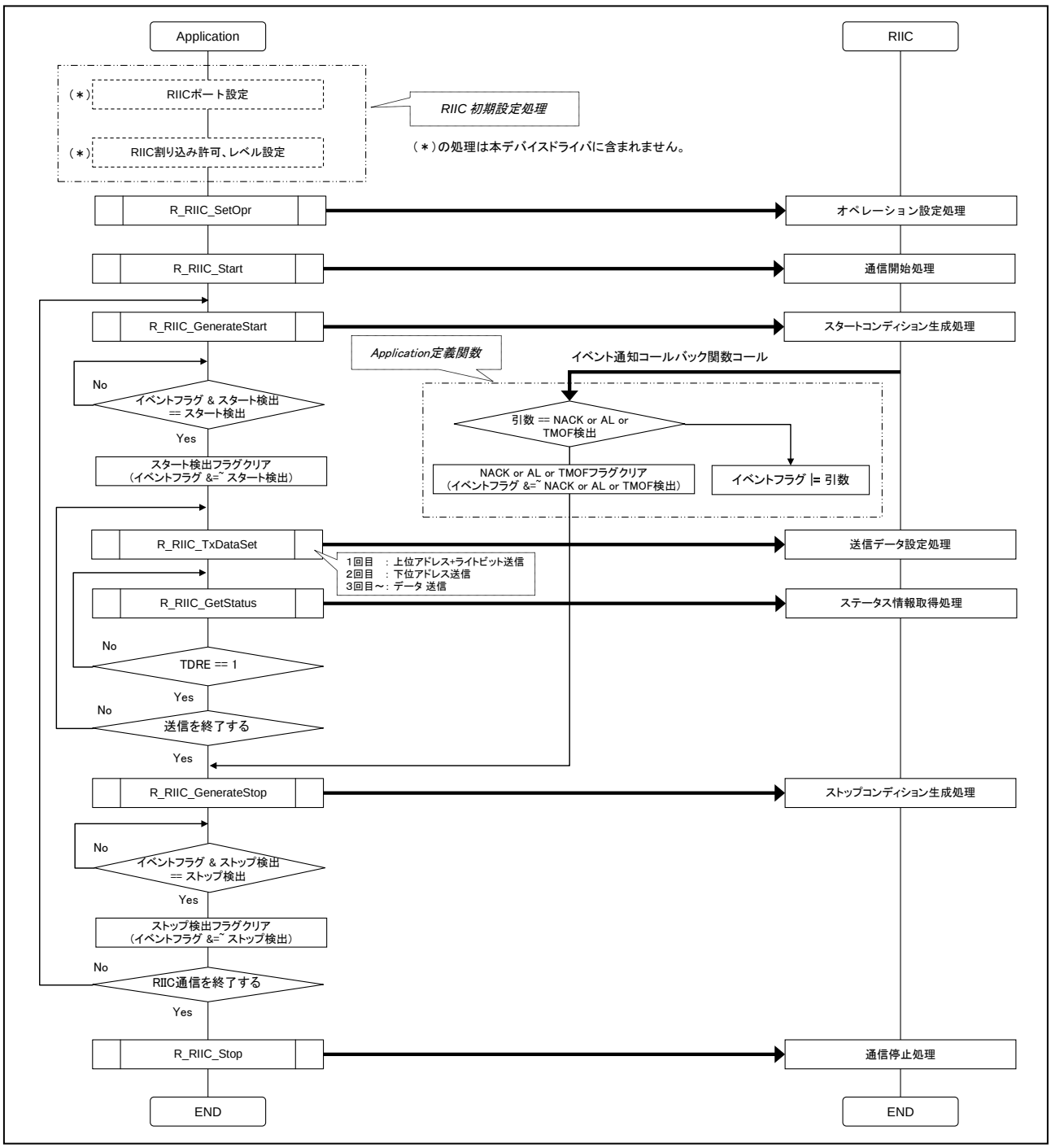

図 1. マスタモード送信時の使用例（10bit アドレスモード）

6.1.2 マスタ受信時の使用例（10bit アドレスモード）

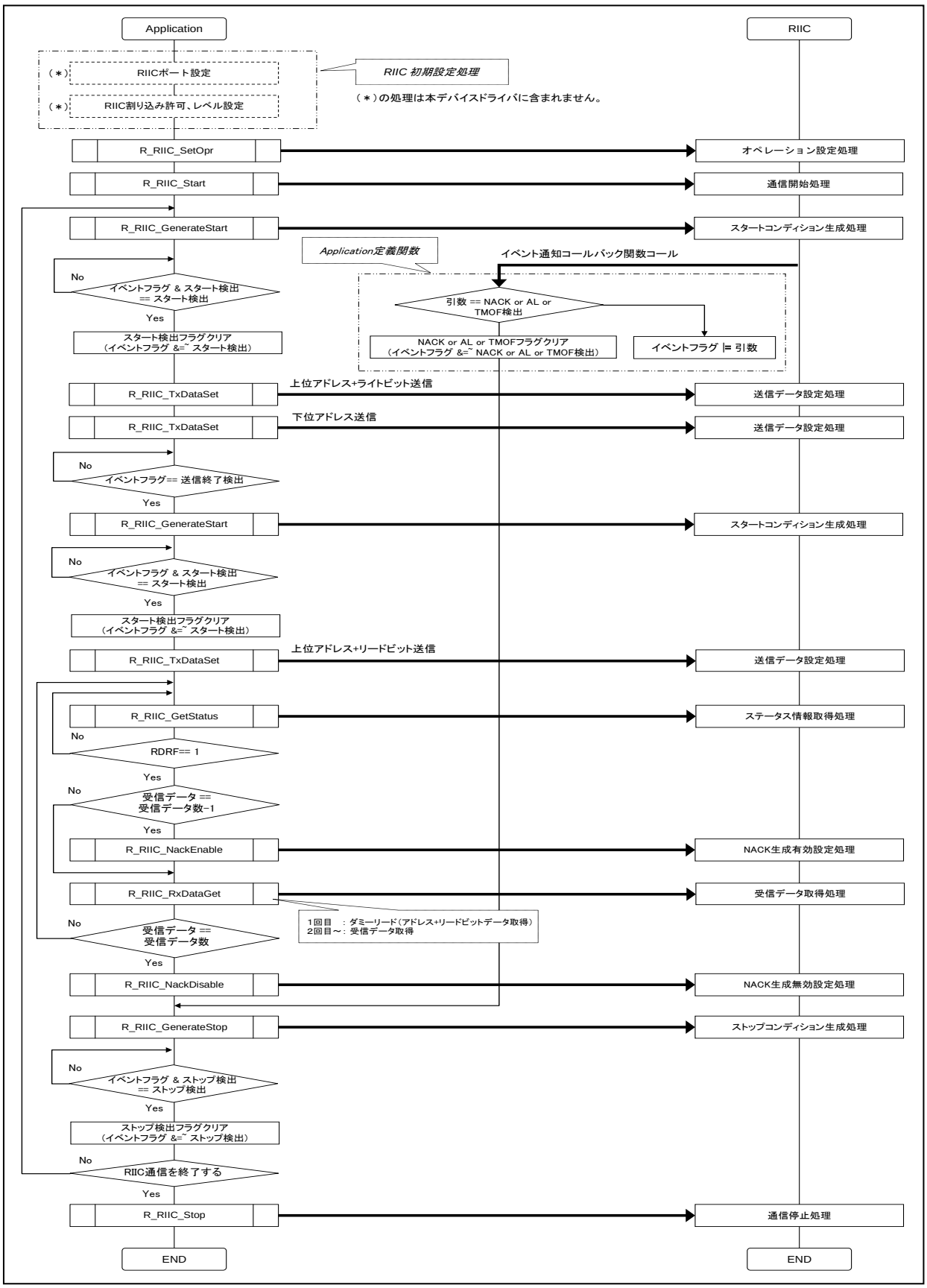

図 2. マスタ受信時の処理例（10bit アドレスモード）

6.1.3 マスタ送信時の処理例（7bit アドレスモード、DMAC 連動）

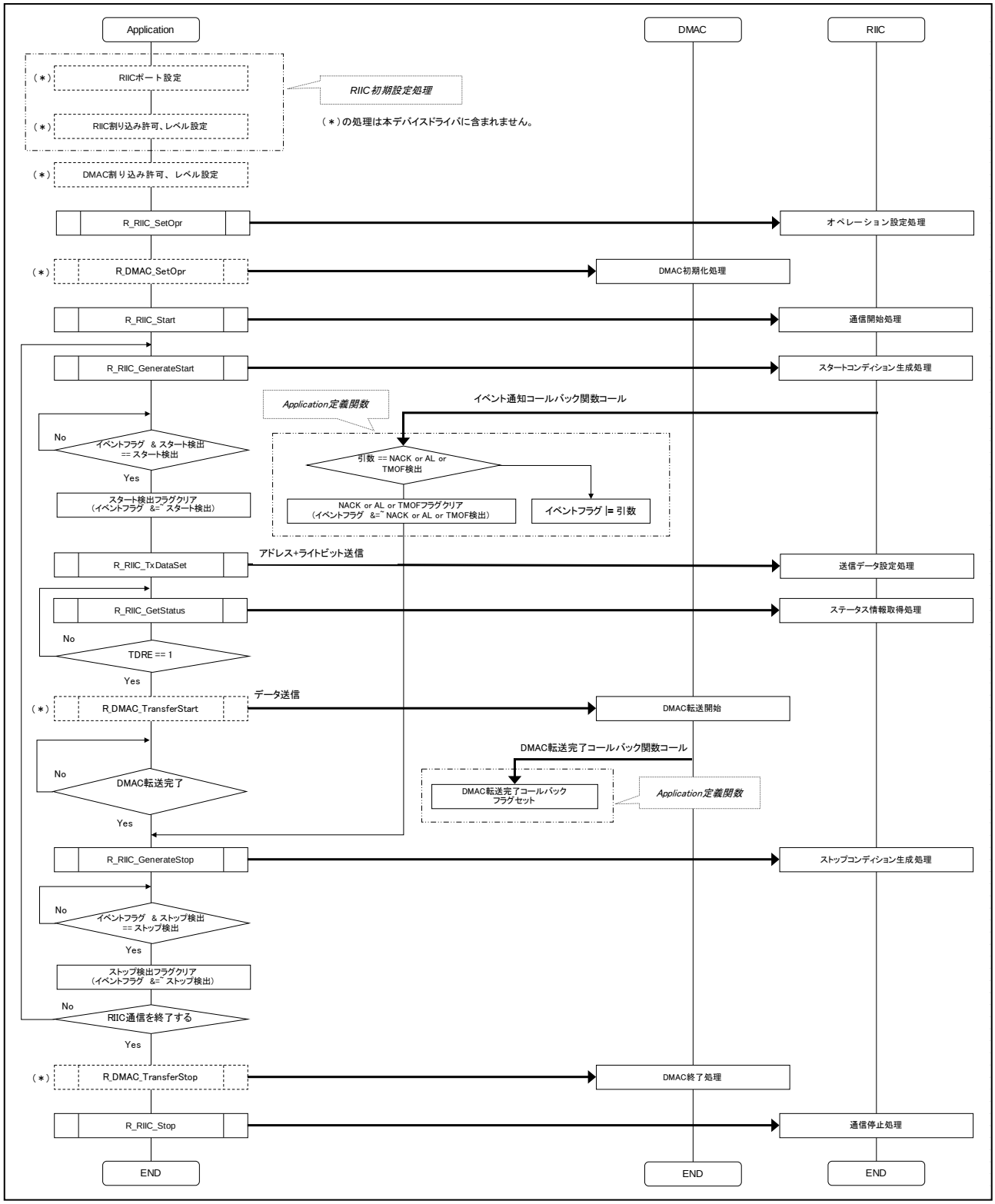

図 3. マスタ送信時の処理例（7bit アドレスモード、DMAC 連動）

6.1.4 マスタ受信時の処理例（7bit アドレスモード、DMAC 連動）

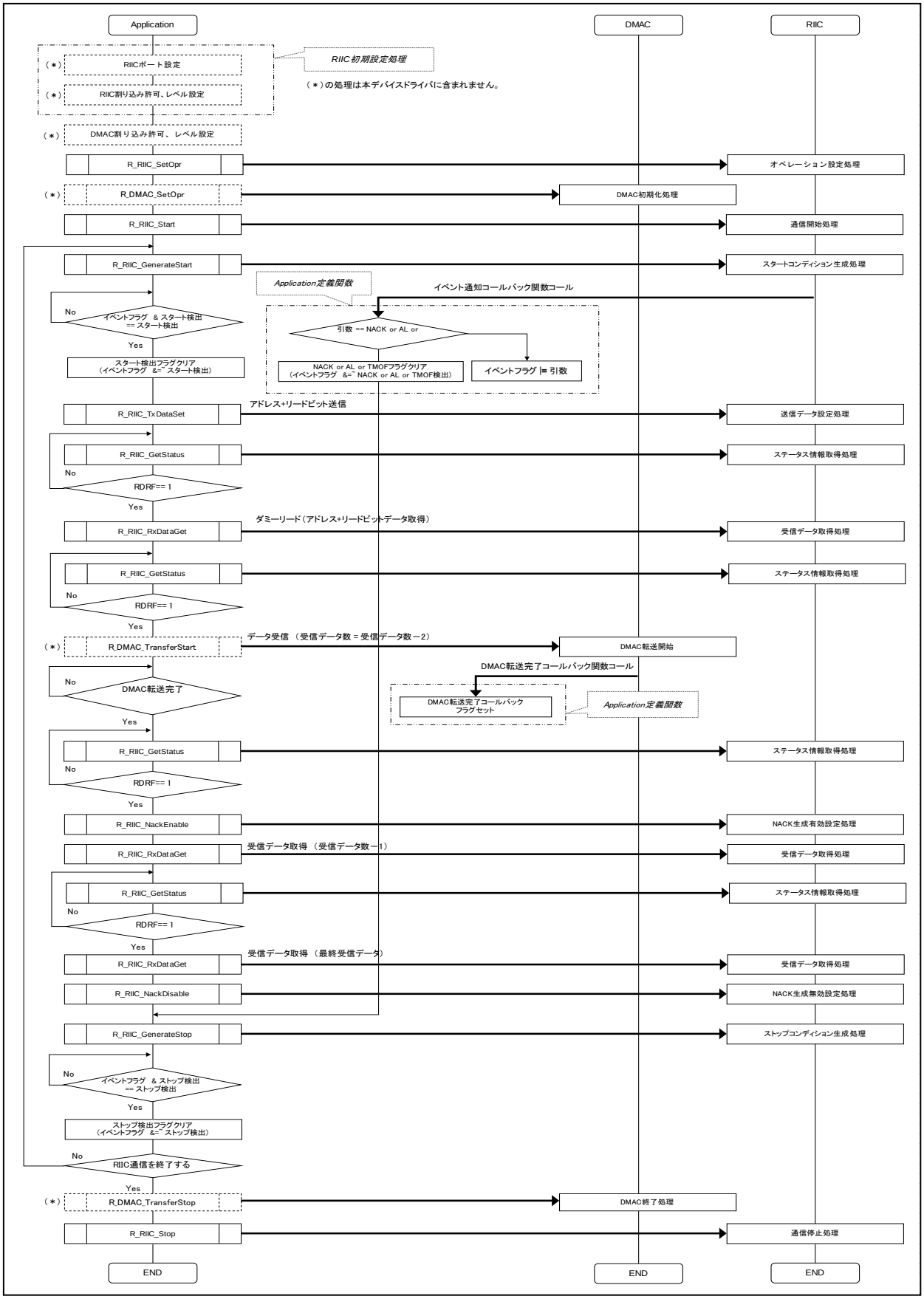

図 4. マスタ受信時の処理例（7bit アドレスモード、DMAC 連動）

6.1.5 スレーブ送信時の処理例

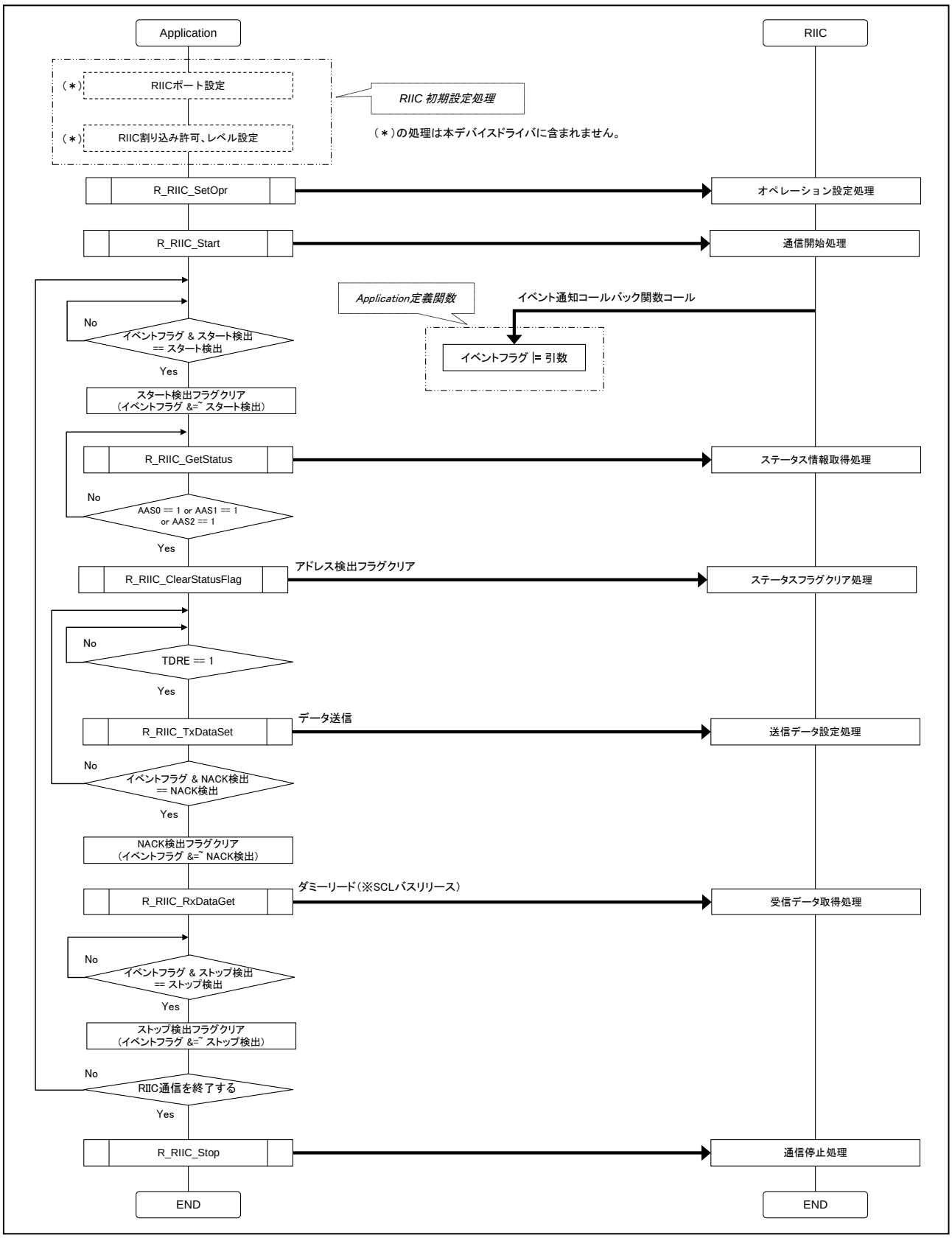

図 5. スレーブ送信時の処理例

6.1.6 スレーブ受信時の処理例

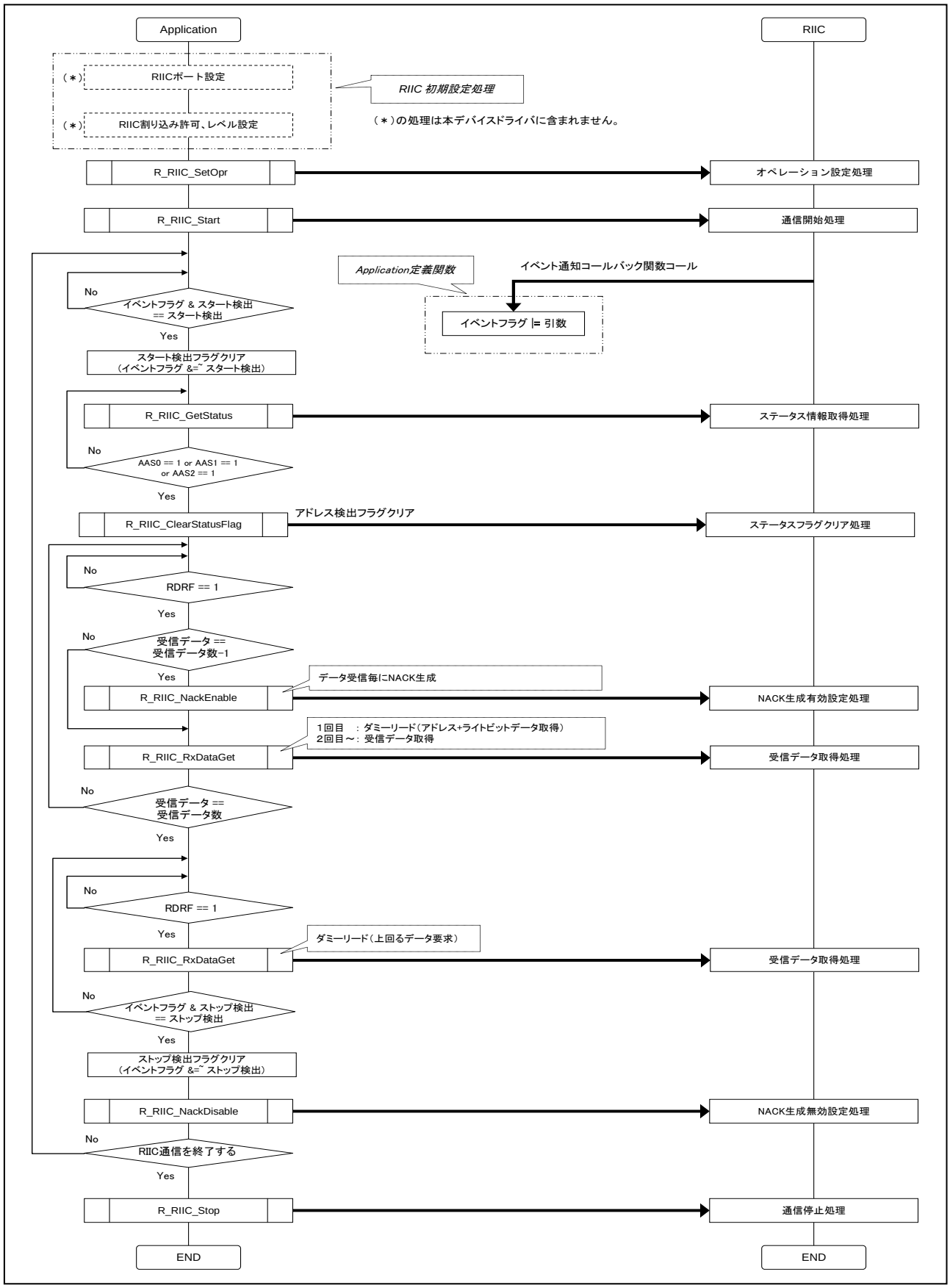

図 6. スレーブ受信時の処理例

6.1.7 ISQ マスタ送信時の処理例

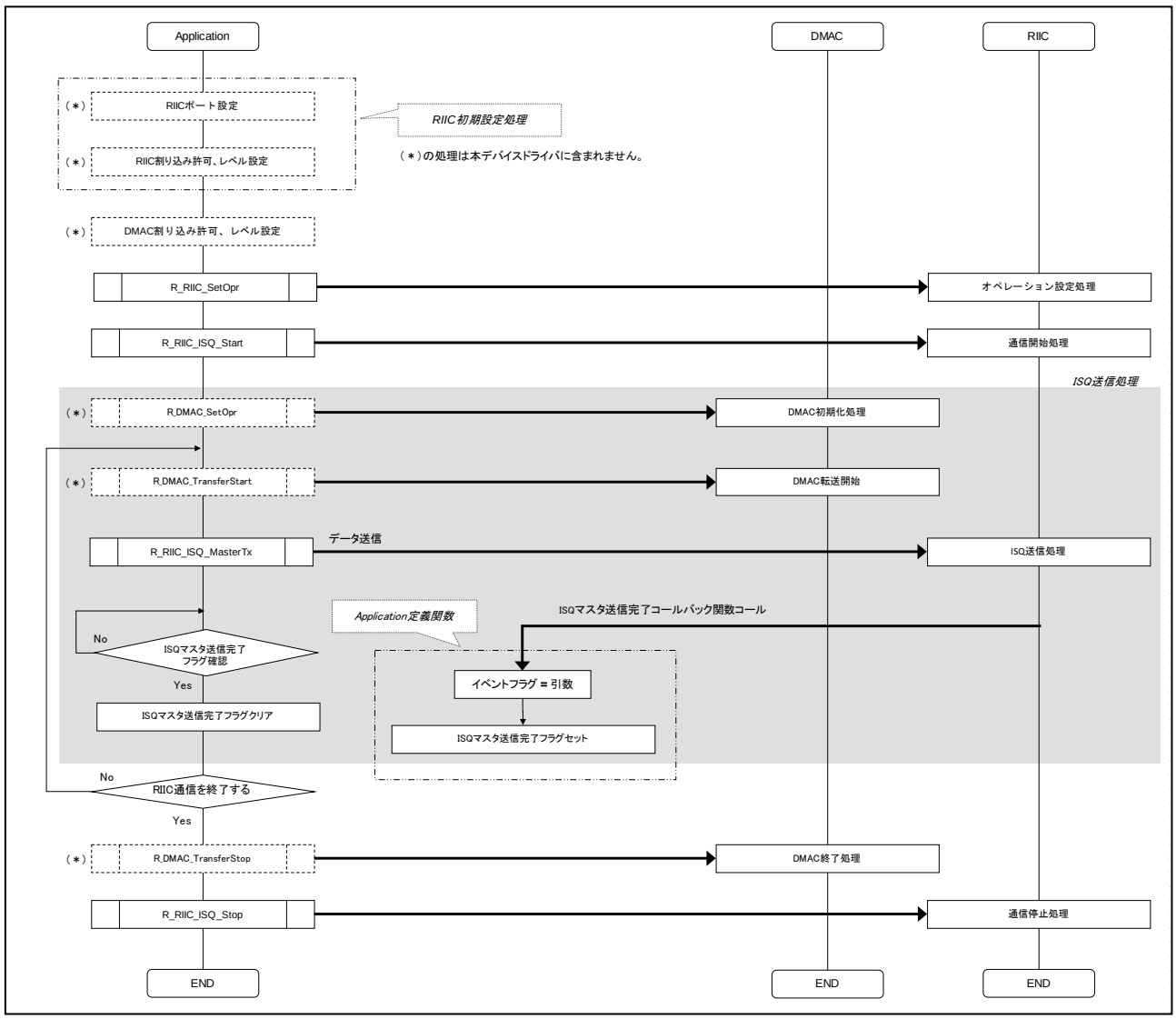

図 7. ISQ マスタ送信時の処理例

6.1.8 ISQ マスタ受信時の処理例

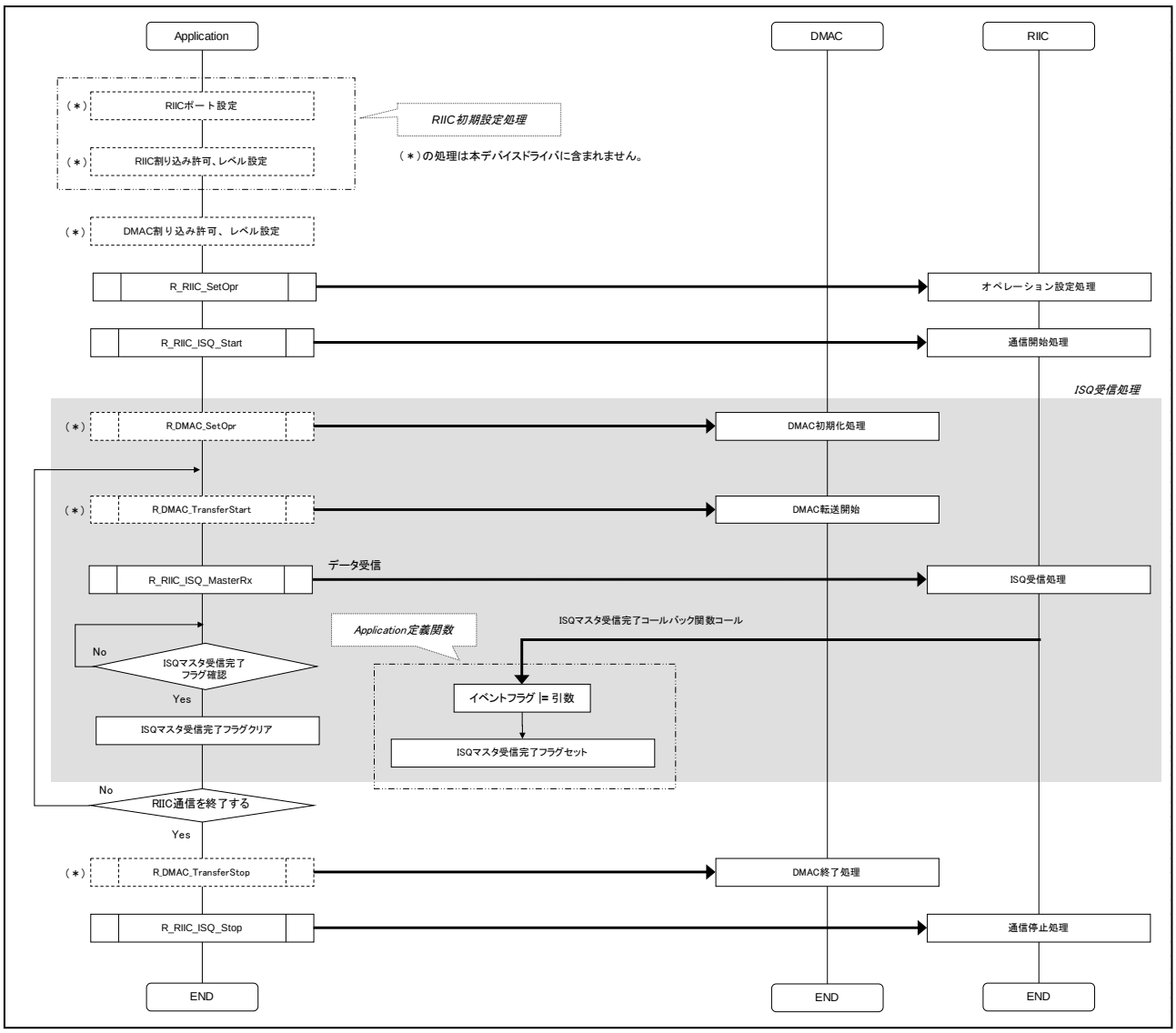

図 8. ISQ マスタ受信時の処理例

6.1.9 ISQ スレーブ送信時の処理例

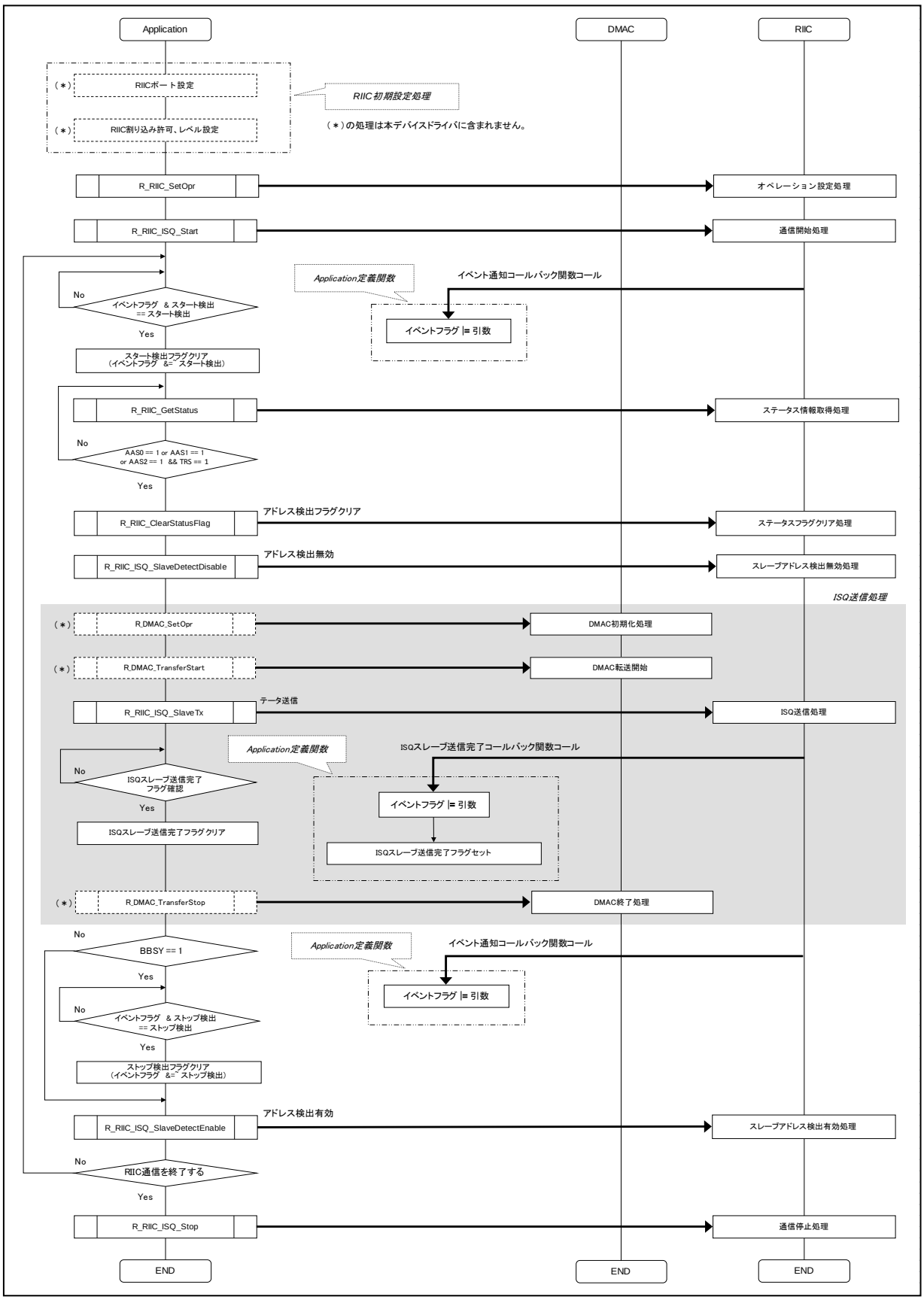

図 9. ISQ スレーブ送信時の処理例

6.1.10 ISQ スレーブ受信時の処理例

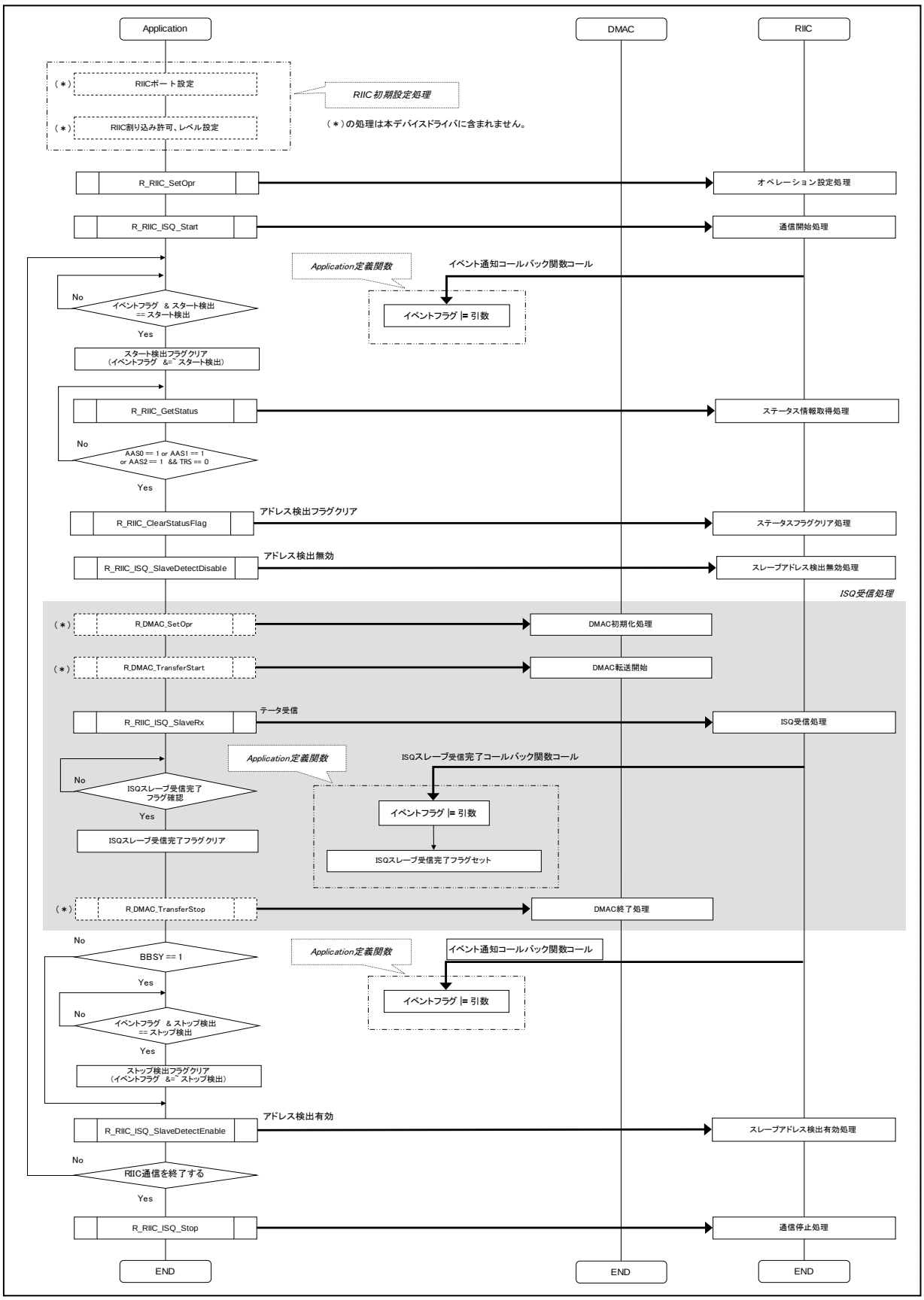

図 10. ISQ スレーブ受信時の処理例

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

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.7.1	—	初版発行
1.01	2013.10.23	—	サンプルコードの修正

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、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うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、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・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒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/