RL78/G23

ELCL、SPI による I²S 通信

要旨

本アプリケーションノートでは、ロジック＆イベント・リンク・コントローラ（ELCL）とシリアル・アレイ・ユニット（SAU）を利用して I²S 通信を実現する方法を説明します。

動作確認デバイス

RL78/G23

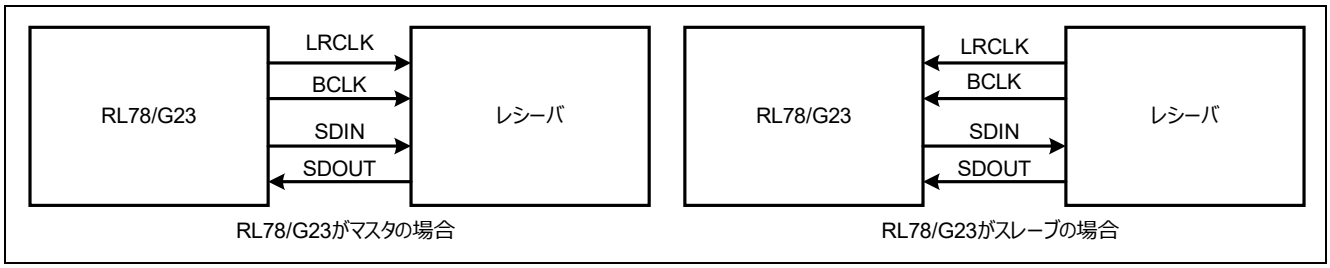
本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様に合わせて変更し、十分評価してください。

目次

1. 仕様

本アプリケーションノートでサポートする I²S 通信の構成を図 1-1、仕様を表 1-1 に示します。

図 1-1 RL78/G23 とレシーバの構成



補足 LRCLK：LR クロック（ワード・セレクト）
L チャネル、R チャネルの区別をする信号で Low が L チャネル、High が R チャネル
サンプリング周波数と一致
BCLK：I²S ビットクロック
SDIN：サウンド・データをレシーバが受信
SDOUT：サウンド・データをレシーバが送信

表 1-1 I²S 通信フォーマット

項目	内容
音声データフォーマット	PCM
機能	マスタ、スレーブどちらでも動作可能
	BCLK 周波数：32fs、48fs、64fs（fs：サンプリング周波数）
	サンプリング周波数：8～96kHz 注 1
	データサイズ：16、24、32 ビット

注 1. 表のデータサイズとサンプリング周波数の組み合わせで動作可能です。
ただし、サンプリング周波数 96kHz の時、データサイズ 24 または 32 ビットは、電気的特性を満足できないので、サポートできません。

図 1-2 に ELCL を使った I²S 通信（マスタ）の構成を示します。

BCLK と LRCLK はタイマ・アレイ・ユニット 0（TAU0）を使用して生成します。TAU00 のインターバル・タイマ・モードで作成した BCLK は ELCL でタイミングを調整し、SAU01 の SCK01 に接続されます。また、TAU01 のイベント・カウンタ・モードで TAU00 を外部イベントとして使用し、LRCLK として端子から出力されます。

図 1-2 ELCL を使った I²S 通信（マスタ）の構成

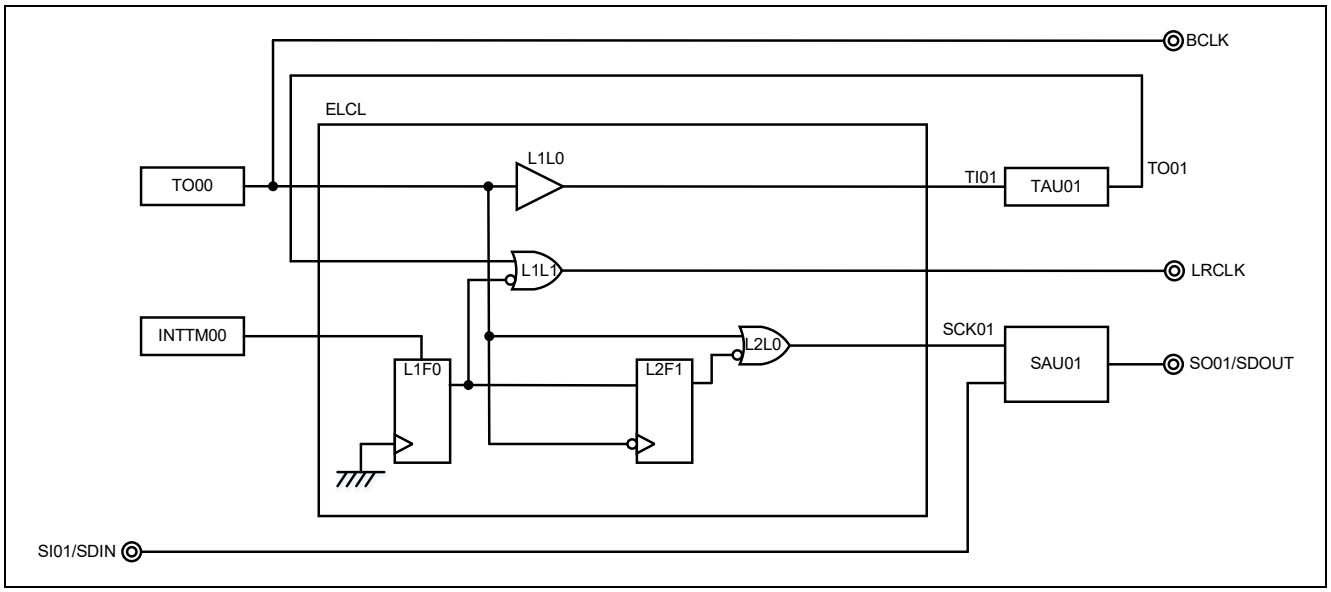


図 1-3 に ELCL を使った I²S 通信（マスタ）のタイミングチャート（データサイズ 16bit）を示します。

(1) TAU00 はインターバル・タイマ・モード、TAU01 はイベント・カウンタ・モード、SAU01 は連続送信モード（スレーブ）に設定します
(2) TAU00、TAU01 が動作を開始します
(3) INTTM00 を ELCL の L1F0 でラッチし、LRCLK を Low にします
(4) (3)でラッチした信号で BCLK の 1 クロック分遅らせ、SCK01 を生成します
(5) SCK01 に同期して、SO01 が動作を開始します

図 1-3 ELCL を使った I²S 通信（マスタ）のタイミングチャート

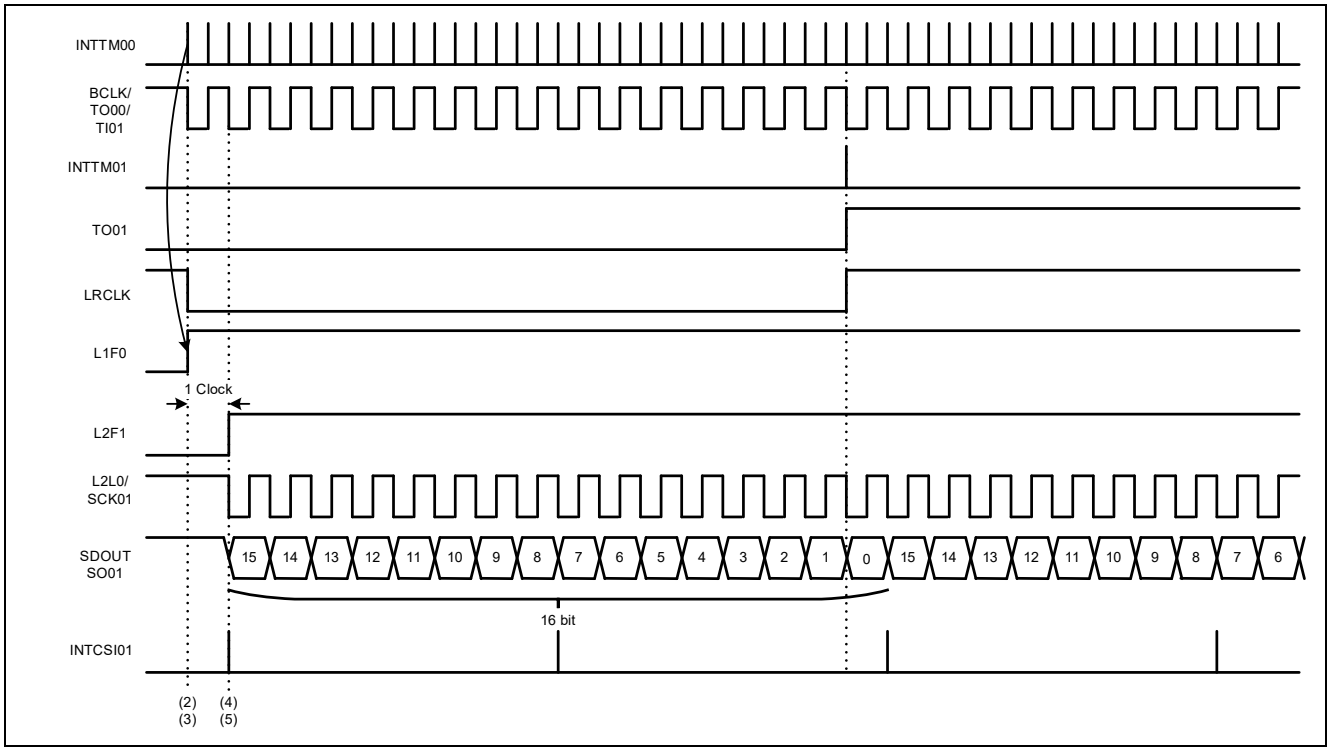


図 1-4 に ELCL を使った I²S 通信（スレーブ）の構成を示します。

BCLK は ELCL でタイミングを調整し SAU01 の SCK01 に接続されます。LRCLK は ELCL で SCK01 のタイミングを生成する信号として使用します。

図 1-4 ELCL を使った I²S 通信（スレーブ）の構成

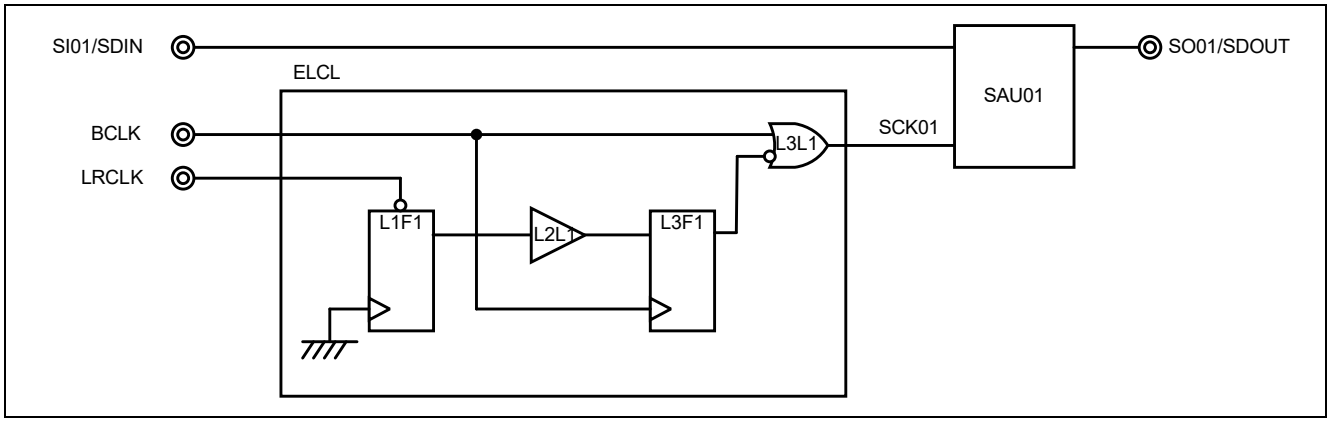
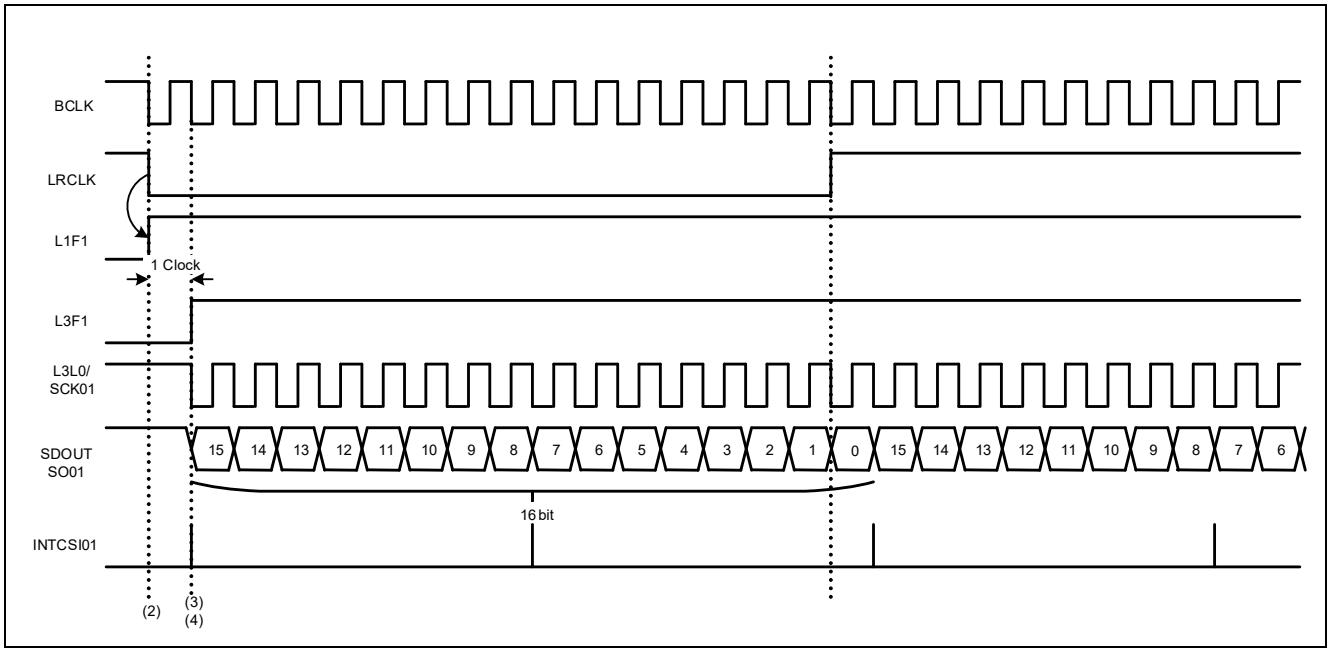


図 1-5 に ELCL を使った I²S 通信（スレーブ）のタイミングチャート（データサイズ 16bit）を示します。

(1) SAU01 は連続送信モード（スレーブ）に設定します
(2) LRCLK が High→Low の変化を ELCL の L1F1 でラッチします
(3) (2)でラッチした信号で BCLK を 1 クロック分遅らせ、SCK01 を生成します
(4) SCK01 に同期して、SO01 が動作を開始します

図 1-5 ELCL を使った I²S 通信（スレーブ）のタイミングチャート



補足 BCLK と LRCLK の初期値は CODEC モジュールによって異なります。

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

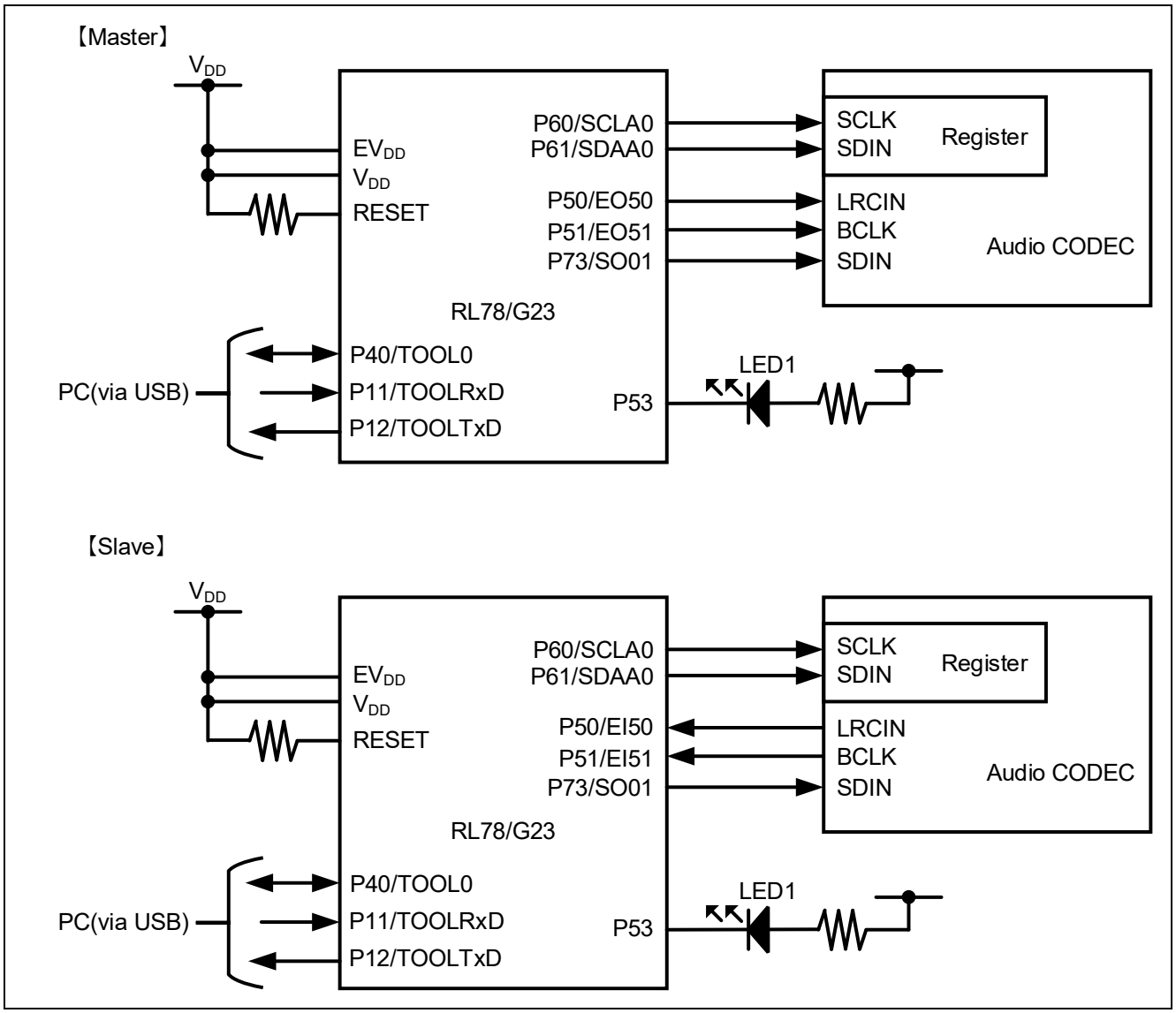
項目	内容
使用マイコン	RL78/G23（R7F100GLG）
動作周波数	• 高速オンチップ・オシレータ・クロック：32MHz • CPU／周辺ハードウェア・クロック：32MHz
動作電圧	• 3.3V • LVD0 動作（V_{LVD0}）：リセット・モード 立ち上がり時 TYP. 1.90V 立ち下がり時 TYP. 1.86V
統合開発環境（CS+）	ルネサスエレクトロニクス製 CS+ for CC V8.07.00
C コンパイラ（CS+）	ルネサスエレクトロニクス製 CC-RL V1.11
統合開発環境（e² studio）	ルネサスエレクトロニクス製 e² studio 2022-04 (22.04.0)
C コンパイラ（e² studio）	ルネサスエレクトロニクス製 CC-RL V1.11
統合開発環境（IAR）	IAR システム製 IAR Embedded Workbench for Renesas RL78 V4.21.1
C コンパイラ（IAR）	
スマート・コンフィグレータ	V.1.3.0
ボードサポートパッケージ（r_bsp）	V.1.20
エミュレータ	CS+、e² studio：COM ポート IAR：E2 エミュレータ Lite
使用ボード	RL78/G23 Fast Prototyping Board (RTK7RLG230CLG000BJ)

3. ハードウェア説明

3.1 ハードウェア構成例

図 3-1 に本アプリケーションのサンプルコードで使用するハードウェア構成例を示します。

図 3-1 ハードウェア構成例



補足 I²C 通信については 4.1.1 動作概要を参照してください。

注意 1. この回路イメージは接続の概要を示す為に簡略化しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して V_{DD} 又は V_{SS} に接続して下さい）。

注意 2. EV_{SS} で始まる名前の端子がある場合には V_{SS} に、EV_{DD} で始まる名前の端子がある場合には V_{DD} にそれぞれ接続してください。

注意 3. V_{DD} は LVD0 にて設定したリセット解除電圧（V_{LVD0}）以上にしてください。

3.2 使用端子一覧

表 3-1 に使用端子と機能を示します。

表 3-1 使用端子と機能

端子名	入出力	内容
P53	出力	LED1 点灯（Low Active）
P60/SCLA0	出力	シリアル・クロック
P61/SDAA0	出力	シリアル・データ
P50/EO50	出力（マスタ）	LRCLK
P50/EI50	入力（スレーブ）	
P51/EO51	出力（マスタ）	BCLK
P51/EI51	入力（スレーブ）	
P73/SO01	出力	SDIN

注意 本アプリケーションノートは、使用端子のみを端子処理しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください。

4. ELCL を使った I²S 通信（マスタ）

4.1 ソフトウェア説明

4.1.1 動作概要

本サンプルコードでは、RS 社の Audio CODEC 754-1974 を使用します。また、オーディオ CODEC 内のレジスタ設定に I²C 通信を、サウンド・データ送信処理に I²S 通信を使用します。

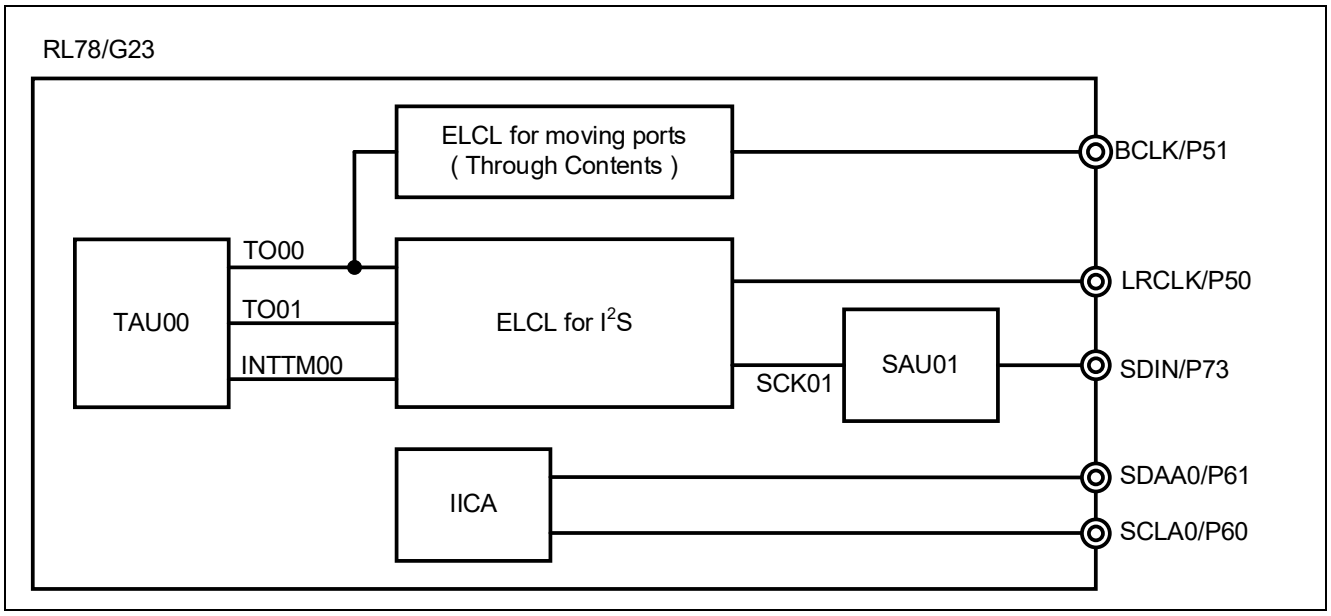
BCLK は ELCL の Through コンテンツを使用し、出力端子を P51 に変更しています。

サンプルコードは以下のような動作をします。

（1）I²C 通信用の IICA が動作を開始し、オーディオ CODEC 内のレジスタ設定を行います
（2）I²S 通信用の CSI01 が動作を開始します（SCK01 待ち状態）
（3）BCLK、LRCLK 用の TAU0 が動作を開始し、データ送信が開始します
（4）データ送信後、LED1 を点灯します
（5）CSI01、TAU0 が停止します

図 4-1 にサンプルコードのシステム構成を示します。

図 4-1 サンプルコードのシステム構成



4.1.2 フォルダ構成

表 4-1、表 4-2 にサンプルコードの使用しているソースファイル／ヘッダファイルの構成を示します。なお、統合開発環境で自動生成されるファイル、bsp 環境のファイルは除きます。

表 4-1 フォルダ構成 (1/2)

フォルダ、ファイル名	説明	スマート・コンフィグレータを使用
¥r01an6420_elcl_i2s_master<DIR>注 3	サンプルコードのフォルダ	
¥src<DIR>	プログラム格納用フォルダ	
main.c	サンプルコードソースファイル	
main.h	サンプルコードヘッダファイル	
r_codec.c 注 4	モジュール設定用ソースファイル	
r_codec.h 注 4	モジュール設定用ヘッダファイル	
sound_data.c	サウンド・データ用ソースファイル	
sound_data.h	サウンド・データ用ヘッダファイル	
¥csi01<DIR>	CSI01 用プログラム格納フォルダ	
csi01.c	CSI01 用ソースファイル	
csi01.h	CSI01 用ヘッダファイル	
¥elcl<DIR>	ELCL 用プログラム格納フォルダ	
elcl.c	ELCL 用ソースファイル	
elcl.h	ELCL 用ヘッダファイル	
¥iica0<DIR>注 4	IICA0 用プログラム格納フォルダ	
iica0.c	IICA0 用ソースファイル	
iica0.h	IICA0 用ヘッダファイル	
¥smc_gen<DIR>	スマート・コンフィグレータ生成フォルダ	√
¥Config_PORT<DIR>	ポート用プログラム格納フォルダ	√
Config_PORT.c	ポート用ソースファイル	√
Config_PORT.h	ポート用ヘッダファイル	√
Config_PORT_user.c	ポート用割り込みソースファイル	√注 1
¥Config_TAU0_0<DIR>	TAU00 用プログラム格納フォルダ	√
Config_TAU0_0.c	TAU00 用ソースファイル	√
Config_TAU0_0.h	TAU00 用ヘッダファイル	√
Config_TAU0_0_user.c	TAU00 用割り込みソースファイル	√注 2
¥Config_TAU0_1<DIR>	TAU01 用プログラム格納フォルダ	√
Config_TAU0_1.c	TAU01 用ソースファイル	√
Config_TAU0_1.h	TAU01 用ヘッダファイル	√
Config_TAU0_1_user.c	TAU01 用割り込みソースファイル	√注 1

補足 ”<DIR>” は、ディレクトリを意味します。
注 1. 本サンプルコードでは使用しません。
注 2. スマート・コンフィグレータで生成したファイルに初期設定を追加しています。
注 3. IAR 版のサンプルコードは r01an6420_elcl_i2s_master.ipcf を格納しています。ipcf ファイルについては、「RL78 スマート・コンフィグレータ ユーザーガイド：IAR 編（R20AN0581）」を確認してください。
注 4. RS 社の Audio CODEC 754-1974 のモジュール設定に必要なファイルです。

表 4-2 フォルダ構成 (2/2)

フォルダ、ファイル名	説明	スマート・コンフィグレータを使用
¥Config_TAU0_7<DIR>注4	TAU07 用プログラム格納フォルダ	√
Config_TAU0_7.c	TAU07 用ソースファイル	√
Config_TAU0_7.h	TAU07 用ヘッダファイル	√
Config_TAU0_7_user.c	TAU07 用割り込みソースファイル	√注5
¥Config_Through<DIR>	Through 用プログラム格納フォルダ	√
Config_Through.c	Through 用ソースファイル	√
Config_Through.h	Through 用ヘッダファイル	√
Config_Through _user.c	Through 用割り込みソースファイル	√
¥general<DIR>	初期化、共通プログラム格納フォルダ	√
¥r_bsp<DIR>	BSP 用プログラム格納フォルダ	√
¥r_config<DIR>	プログラム格納フォルダ	√

補足 ”<DIR>” は、ディレクトリを意味します。
注 4. RS 社の Audio CODEC 754-1974 のモジュール設定に必要なファイルです。
注 5. スマート・コンフィグレータで生成したファイルに割り込み処理ルーチンを追加しています。

I²S 通信は CSI を使用するため、本来の CSI の規格に沿わず、オーバーランエラーが発生します。スマート・コンフィグレータで生成したコードからエラー検出処理を削除しています。

また、IICA で多重割り込みを有効にするため、スマート・コンフィグレータで生成されたコードを修正し、不要な関数を削除しています。

4.1.3 オプション・バイトの設定一覧

表 4-3 にオプション・バイト設定を示します。

表 4-3 オプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	1110 1111B (EFH)	ウォッチドッグ・タイマ動作停止 （リセット解除後、カウント停止）
000C1H/040C1H	1111 1110B (FEH)	LVD0 リセット・モード 検出電圧：立ち上がり 1.90V／立下り 1.86V
000C2H/040C2H	1110 1000B (E8H)	フラッシュ動作モード：高速メインモード 高速オンチップ・オシレータの周波数：32MHz
000C3H/040C3H	1000 0101B (85H)	オンチップ・デバッグ動作許可

4.1.4 定数一覧

表 4-4、表 4-5 にサンプルコードで使用する定数を示します。

表 4-4 サンプルコードで使用する定数 (1/2)

定数名	設定値	内容	ファイル
LED1	P5_bit.no3	P53	csi01.c
LED_ON	0	LED を ON するための設定値	csi01.c
LED_OFF	1	LED を OFF するための設定値	csi01.c
DATA_48K_32B	1	再生するサウンド・データを選択する。（1：有効、0：無効）	sound_data.h
DATA_44K_24B	0	再生するサウンド・データを選択する。（1：有効、0：無効）	sound_data.h
DATA_8K_16B	0	再生するサウンド・データを選択する。（1：有効、0：無効）	sound_data.h
DECODE_PCM_SIZE	32000 (DATA_48K_32B)	サウンド・データのデータ数。 再生するサウンド・データによって設定値が変化。	sound_data.h
	22791 (DATA_44K_24B)		
	20481 (DATA_8K_16B)		
g_tx_buf[]	サンプルコード参照	サウンド・データ	sound_data.c
SENSOR_ADD	0x34	センサのアドレス	r_codec.h
WAIT_TIME	100	IICA0 通信待ち時間	r_codec.h
g_cmd_l_vol[2]	{ 0x00, 0x17 }	左ライン・入力チャンネル・ボリューム・制御コマンド	r_codec.c
g_cmd_r_vol[2]	{ 0x02, 0x17 }	右ライン・入力チャンネル・ボリューム・制御コマンド	r_codec.c
g_cmd_bypass[2]	{ 0x08, 0x12 }	アナログ・オーディオパス制御コマンド	r_codec.c
g_cmd_deempha[2]	{ 0x0A, 0x00 }	デジタル・オーディオパス制御コマンド	r_codec.c
g_cmd_line_adc[2]	{ 0x0C, 0x42 }	パワーダウン制御コマンド	r_codec.c

表 4-5 サンプルコードで使用する定数 (2/2)

定数名	設定値	内容	ファイル
g_cmd_set_rate[2]	{0x10, 0x00} (48kHz)	サンプリング周波数制御コマンド	r_codec.c
	{ 0x10, 0x20 } (44kHz)		
	{0x10, 0x0C} (8kHz)		
g_cmd_set_format[2]	{0x0E, 0x0E} (32bit)	デジタル・オーディオインターフェース・フォーマット設定コマンド	r_codec.c
	{0x10, 0x0A} (24bit)		
	{0x0E, 0x02} (16bit)		
g_cmd_activate[2]	{ 0x12, 0x01 }	CODEC モジュール・アクティブコマンド	r_codec.c
g_cmd_inactivate[2]	{ 0x12, 0x00 }	CODEC モジュール・非アクティブコマンド	r_codec.c
g_cmd_reset[2]	{ 0x1E, 0x00 }	レジスタ・リセットコマンド	r_codec.c

4.1.5 変数一覧

表 4-6 に本サンプルコードで使用するグローバル変数を示します。

表 4-6 サンプルコードで使用するグローバル変数

型	変数名	内容	使用関数
volatile uint16_t	g_ms_timer	ウエイト処理のカウント値	r_ms_delay, r_Config_TAU0_7_interrupt
volatile uint8_t	g_tx_done_flag	送信完了フラグ	main r_csi01_callback_sendend
volatile uint8_t	g_sample_mode	I^2C 通信ステータス	r_codec_init r_codec_start r_codec_stop r_iica0_callback_master_sende nd

4.1.6 関数一覧

表 4-7 にサンプルコードで使用する関数を示します。ただし、スマート・コンフィグレータで生成された関数の内、変更を行っていないものは除きます。

表 4-7 関数一覧

関数名	概要	ソースファイル
main	メイン処理	main.c
r_csi01_create	CSI01 初期設定処理	csi.c
r_csi01_start	CSI01 動作開始処理	csi.c
r_csi01_stop	CSI01 停止処理	csi.c
r_csi01_send	CSI01 送信開始処理	csi.c
r_csi01_callback_sendend	CSI01 送信終了のコールバック処理	csi.c
r_csi01_interrupt	CSI01 割り込み処理	csi.c
r_elcl_create	ELCL 初期設定処理	elcl.c
r_elcl_start	ELCL 出力開始処理	elcl.c
r_elcl_stop	ELCL 停止処理	elcl.c
r_elcl_reset_flipflop	ELCL のフリップフロップリセット処理	elcl.c
r_elcl_start_port_move	ELCL の端子出力先変更開始処理	elcl.c
r_elcl_stop_port_move	ELCL の端子出力先変更停止処理	elcl.c
r_iica0_create	IICA0 初期設定処理	Iica0.c
r_iica0_stop	IICA0 停止処理	Iica0.c
r_iica0_stopcondition	IICA0 ストップ・コンディション処理	Iica0.c
r_iica0_master_send	IICA0 マスタ送信開始処理	Iica0.c
r_iica0_callback_master_sendend	IICA0 送信完了コールバック処理	Iica0.c
r_iica0_master_handler	IICA0 割り込みハンドラ	Iica0.c
r_iica0_interrupt	IICA0 割り込み処理	Iica0.c
r_codec_init	CODEC モジュールのセットアップ処理	r_codec.c
r_codec_start	CODEC モジュールの開始処理	r_codec.c
r_codec_stop	CODEC モジュールの停止処理	r_codec.c
r_tau0_start	TAU0 のカウンタ開始処理	r_cg_tau_common.c
R_Config_TAU0_0_Create_UserInit	TAU0 の初期化ユーザーコード追加	Config_TAU0_0_user.c
r_ms_delay	IICA0 ウエイト処理	Config_TAU0_7_user.c
r_Config_TAU0_7_interrupt	TAU チャネル 7 割り込み処理	Config_TAU0_7_user.c

4.1.7 関数仕様

サンプルコードの関数仕様を示します。

[関数名] main

概要	メイン処理
ヘッダ	r_smc_entry.h, main.h, elcl.h,csi01.h, r_codec.h, sound_data.h
宣言	void main (void);
説明	ELCL、CSI01 の初期化、CODEC モジュールの設定、I²S 通信を行います
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_create

概要	CSI01 初期設定処理
ヘッダ	csi01.h, elcl.h
宣言	void r_csi01_create (void);
説明	CSI01 の初期設定を行います
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_start

概要	CSI01 動作開始処理
ヘッダ	csi01.h, elcl.h
宣言	void r_csi01_start (void);
説明	CSI01 の動作を許可します
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_stop

概要	CSI01 停止処理
ヘッダ	csi01.h, elcl.h, Config_TAU0_7.h
宣言	void r_csi01_stop (void);
説明	CSI01 の動作を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_send

概要	CSI01 送信開始処理
ヘッダ	csi01.h, elcl.h
宣言	MD_STATUS r_csi01_send (uint8_t *const tx_buf, uint16_t tx_num);
説明	引数 tx_buf で指定したアドレスから引数 tx_num 個のデータを送信します エラー検出は行いません
引数	tx_buf, tx_num
リターン値	Status
備考	なし

[関数名] r_csi01_callback_sendend

概要	CSI01 送信終了のコールバック処理
ヘッダ	csi01.h, elcl.h
宣言	static void r_csi01_callback_sendend (void);
説明	ELCL の出力を停止し、g_tx_done_flag を立て、LED1 を点灯します
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_interrupt

概要	CSI01 割り込み処理
ヘッダ	csi01.h, elcl.h
宣言	#pragma interrupt r_csi01_interrupt (vect=INTCSI01)
説明	次の送信データを SIO01 にセットします
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_create

概要	ELCL の初期設定処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_create (void);
説明	ELCL の初期設定を行います
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_start

概要	ELCL 動作開始処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_start (void);
説明	ELCL の出力を開始します
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_stop

概要	ELCL 動作停止処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_stop (void);
説明	ELCL の出力を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_reset_flipflop

概要	ELCL のフリップフロップリセット処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_reset_flipflop (void);
説明	ELOMONI フラグをリセットするためにフリップフロップをリセットします
引数	なし
リターン値	なし
備考	ELLnCTL のビット 6 とビット 7 に 0 を設定するとフリップフロップはリセットされます

[関数名] r_elcl_start_port_move

概要	ELCL の端子出力先変更開始処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_start_port_move (void);
説明	R_Config_Through_Start を実行し、端子出力先変更処理を開始します
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_stop_port_move

概要	ELCL の端子出力先変更停止処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_stop_port_move (void);
説明	R_Config_Through_Stop を実行し、端子出力先変更処理を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_create

概要	IICA0 初期設定処理
ヘッダ	iica.h, r_codec.h
宣言	void r_csi01_create (void);
説明	IICA0 の初期設定をします
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_stop

概要	IICA0 停止処理
ヘッダ	iica.h, r_codec.h
宣言	void r_iica0_stop (void);
説明	IICA0 の動作を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_stopcondition

概要	IICA0 ストップ・コンディション
ヘッダ	iica.h, r_codec.h
宣言	void r_iica0_stopcondition (void);
説明	ストップ・コンディションを生成します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_master_send

概要	IICA0 マスタ送信開始処理
ヘッダ	iica.h, r_codec.h
宣言	MD_STATUS r_iica0_master_send (uint8_t adr, uint8_t *const tx_buf, uint16_t tx_num, uint8_t wait);
説明	(1) I²C バスが解放されている場合、スタート・コンデションを生成します (2) スタート・コンデション生成後、スレーブ・アドレスを送信モードにし、データ送信を開始します (3) 通信完了を待ちます
引数	adr, tx_buf, tx_num, wait
リターン値	status
備考	なし

[関数名] r_iica0_callback_master_sendend

概要	IICA0 送信完了コールバック処理
ヘッダ	iica.h, r_codec.h
宣言	static void r_iica0_callback_master_sendend (void);
説明	ストップ・コンディションを生成し、IIC 通信ステータスを FINISH にします
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_master_handler

概要	IICA0 割り込みハンドラ
ヘッダ	iica.h, r_codec.h
宣言	static void r_iica0_master_handler (void);
説明	スレーブ・アドレスに対して ACK を検出した場合、データを送信します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_interrupt

概要	IICA0 割り込み処理
ヘッダ	iica.h, r_codec.h
宣言	#pragma interrupt r_iica0_interrupt (vect=INTIICA0,enable=true)
説明	IICA0 の割り込み要求の受付処理を行います 多重割り込みを許可します
引数	なし
リターン値	なし
備考	なし

[関数名] r_codec_init

概要	CODEC モジュールのセットアップ処理
ヘッダ	sound_data.h, iica0.h, r_codec.h, Config_TAU0_7.h
宣言	void r_codec_init (void);
説明	(1) IICA0 の初期設定処理を行います (2) CODEC モジュールのセットアップを行います (3) 設定完了を待ちます(100ms)
引数	なし
リターン値	なし
備考	なし

[関数名] r_codec_start

概要	CODEC モジュールの開始処理
ヘッダ	sound_data.h, iica0.h, r_codec.h, Config_TAU0_7.h
宣言	void r_codec_start (void);
説明	CODEC モジュールの動作を開始します
引数	なし
リターン値	なし
備考	なし

[関数名] r_codec_stop

概要	CODEC モジュールの停止処理
ヘッダ	sound_data.h, iica0.h, r_codec.h, Config_TAU0_7.h
宣言	void r_codec_stop (void);
説明	CODEC モジュールの動作を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_tau0_start

概要	TAU0 カウンタ開始処理
ヘッダ	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_0.h, Config_TAU0_1.h Config_TAU0_7.h,r_cg_tau_common.h
宣言	void r_tau0_start (void);
説明	TAU00、TAU01 のカウンタを開始します
引数	なし
リターン値	なし
備考	なし

[関数名] R_Config_TAU0_0_Create_UserInit

概要	TAU0 の初期化ユーザーコード追加
ヘッダ	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_0.h
宣言	void R_Config_TAU0_0_Create_UserInit (void);
説明	TAU00 の初期出力を 1、TAU01 の初期出力を 0 に設定します
引数	なし
リターン値	なし
備考	なし

[関数名] r_ms_delay

概要	ウエイト処理
ヘッダ	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_7.h
宣言	void r_ms_delay (uint16_t msec);
説明	引数 msec で指定した時間（ms）ウエイトします TAU0 チャネル 7 を使ってカウントします。g_ms_timer が引数 msec 未満の場合はポーリングし、msec 以上の場合はウエイト処理を完了します
引数	msec
リターン値	なし
備考	なし

[関数名] r_Config_TAU0_7_interrupt

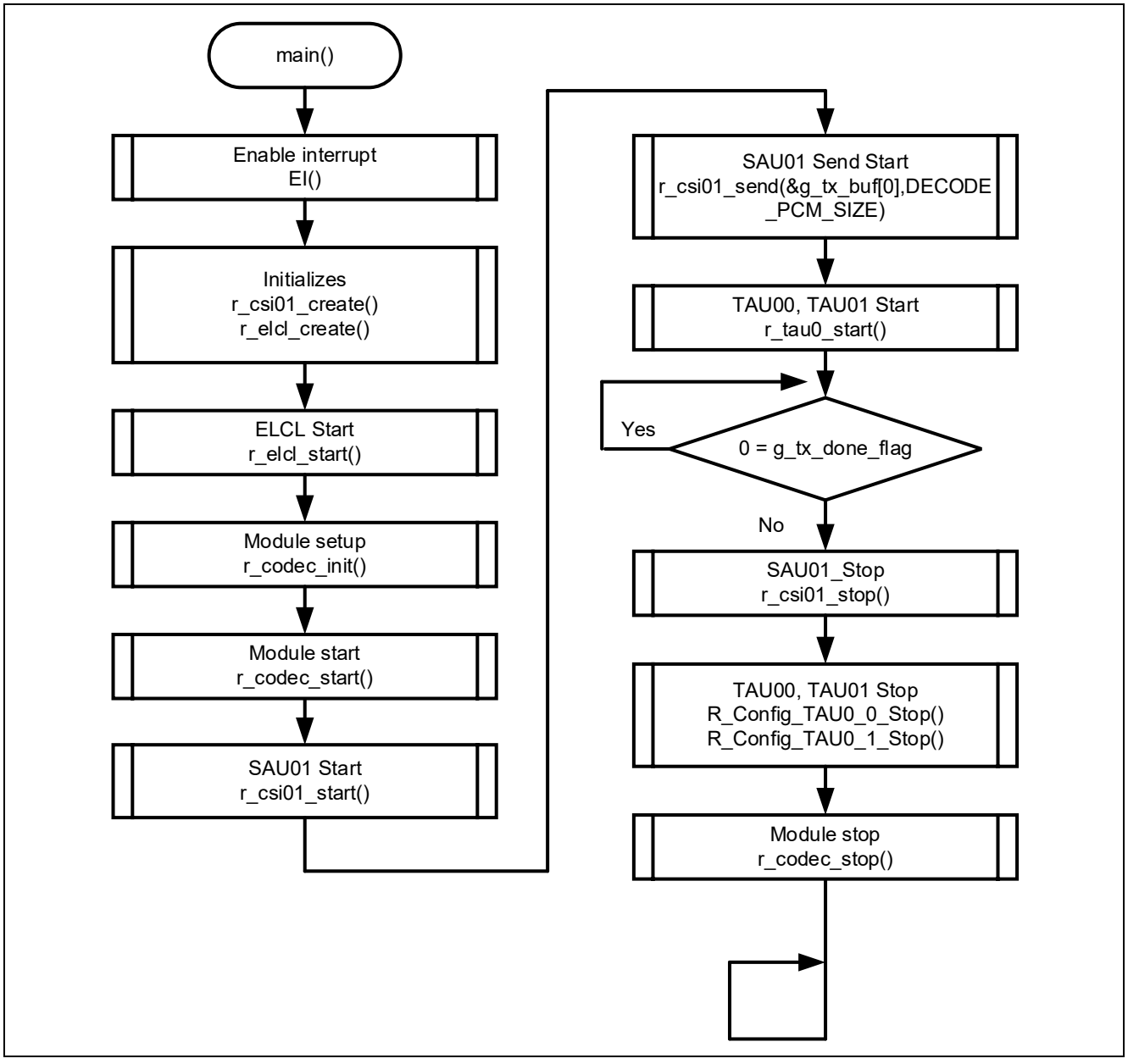
概要	TAU07 割り込み処理
ヘッダ	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_7.h
宣言	#pragma interrupt r_Config_TAU0_7_interrupt (vect=INTTM07)
説明	TAU0 チャネル 7 の INTTM07 による割り込み処理です g_ms_timer をカウントアップします
引数	なし
リターン値	なし
備考	なし

4.1.8 フローチャート

4.1.8.1 メイン処理

図 4-2 にメイン処理のフローチャートを示します。

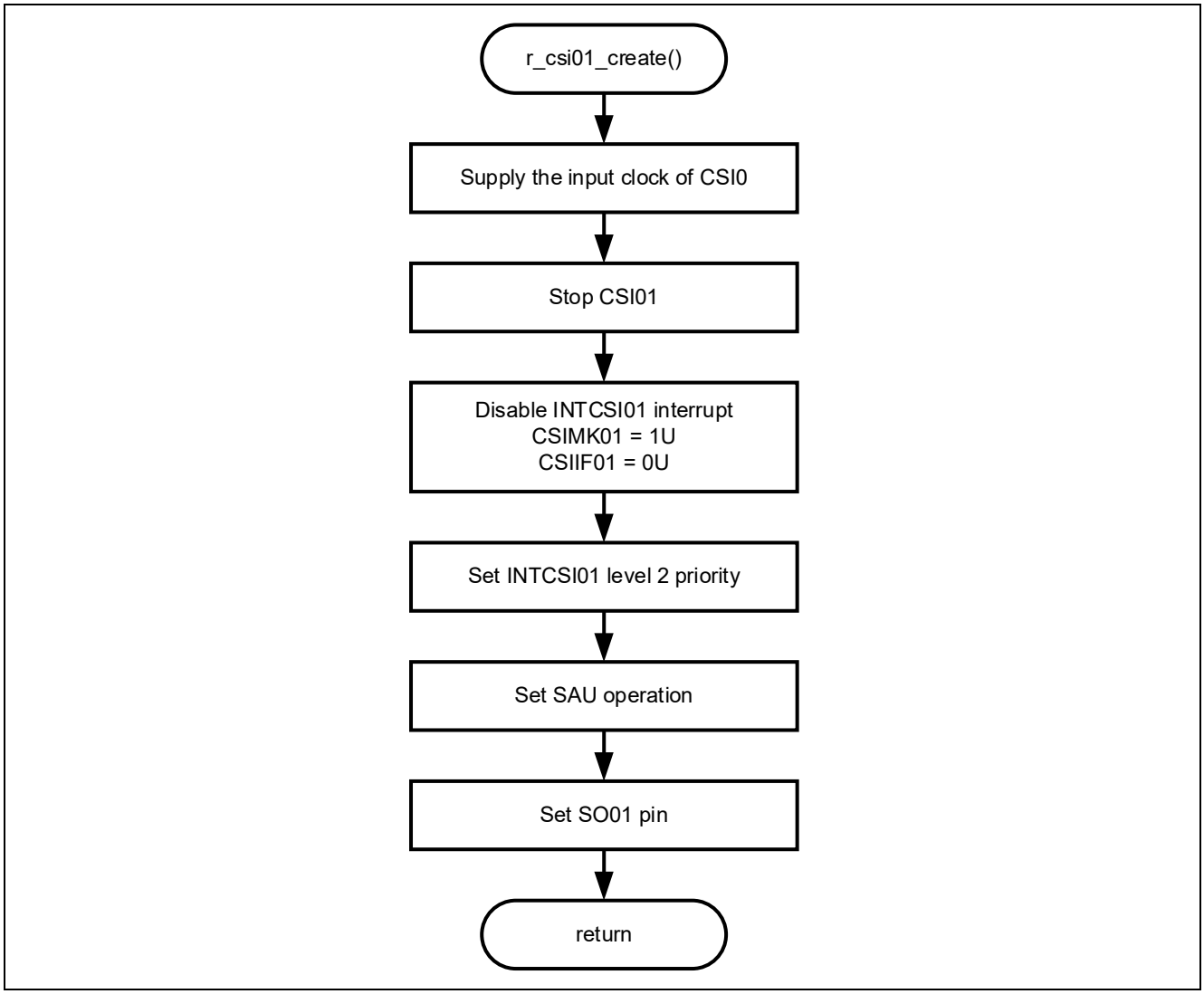
図 4-2 メイン処理



4.1.8.2 CSI01 初期設定処理

図 4-3 に CSI01 初期設定処理のフローチャートを示します。

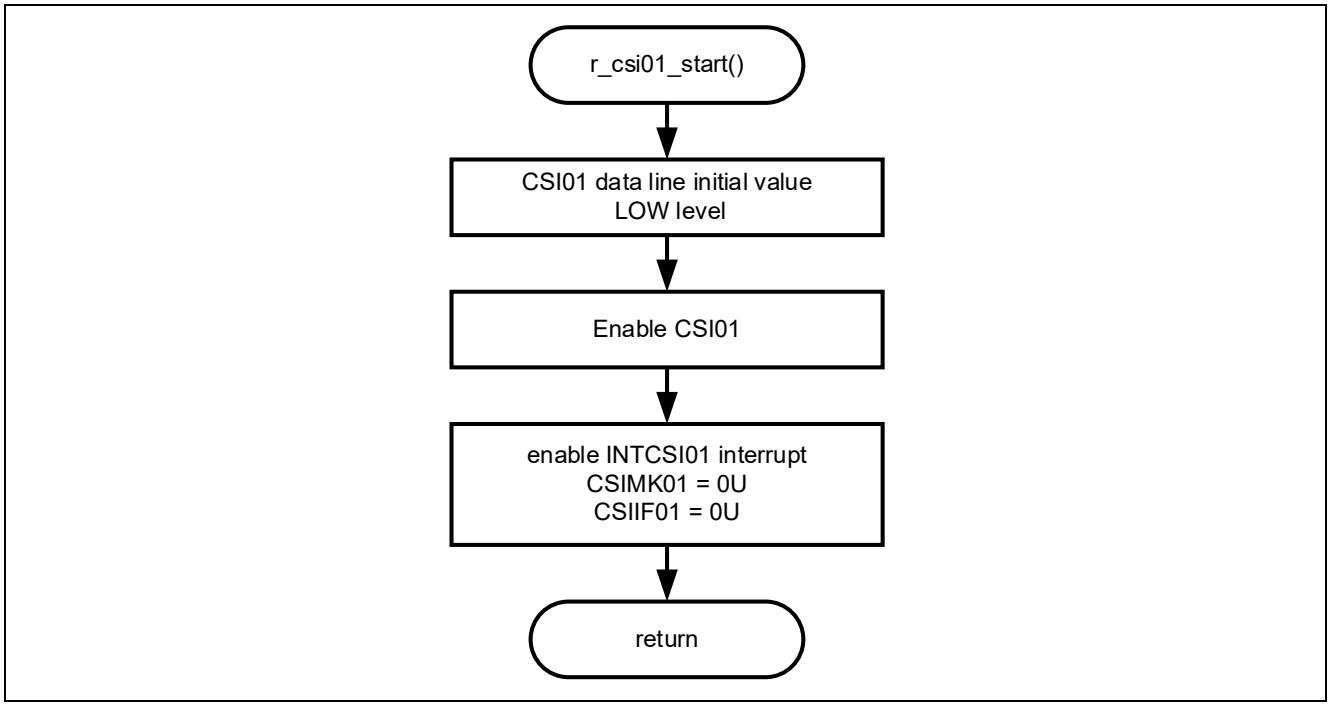
図 4-3 CSI01 初期設定処理



4.1.8.3 CSI01 動作開始処理

図 4-4 に CSI01 動作開始処理のフローチャートを示します。

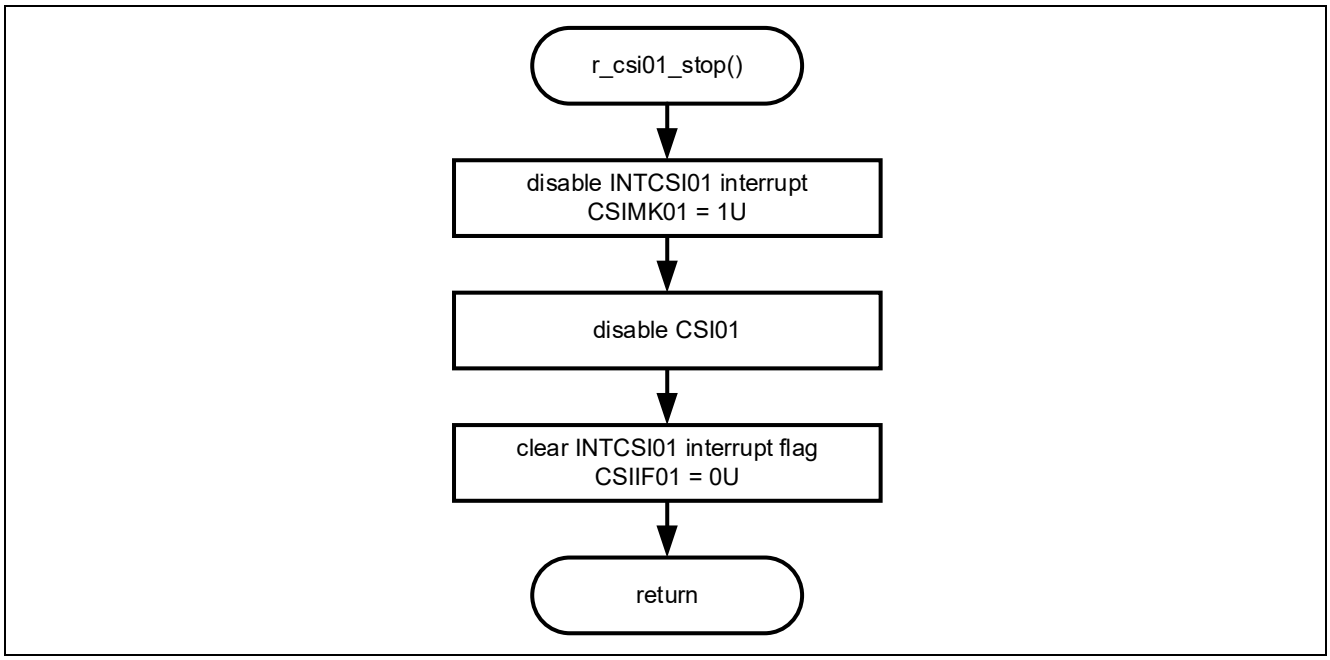
図 4-4 CSI01 動作開始処理



4.1.8.4 CSI01 停止処理

図 4-5 に CSI01 停止処理のフローチャートを示します。

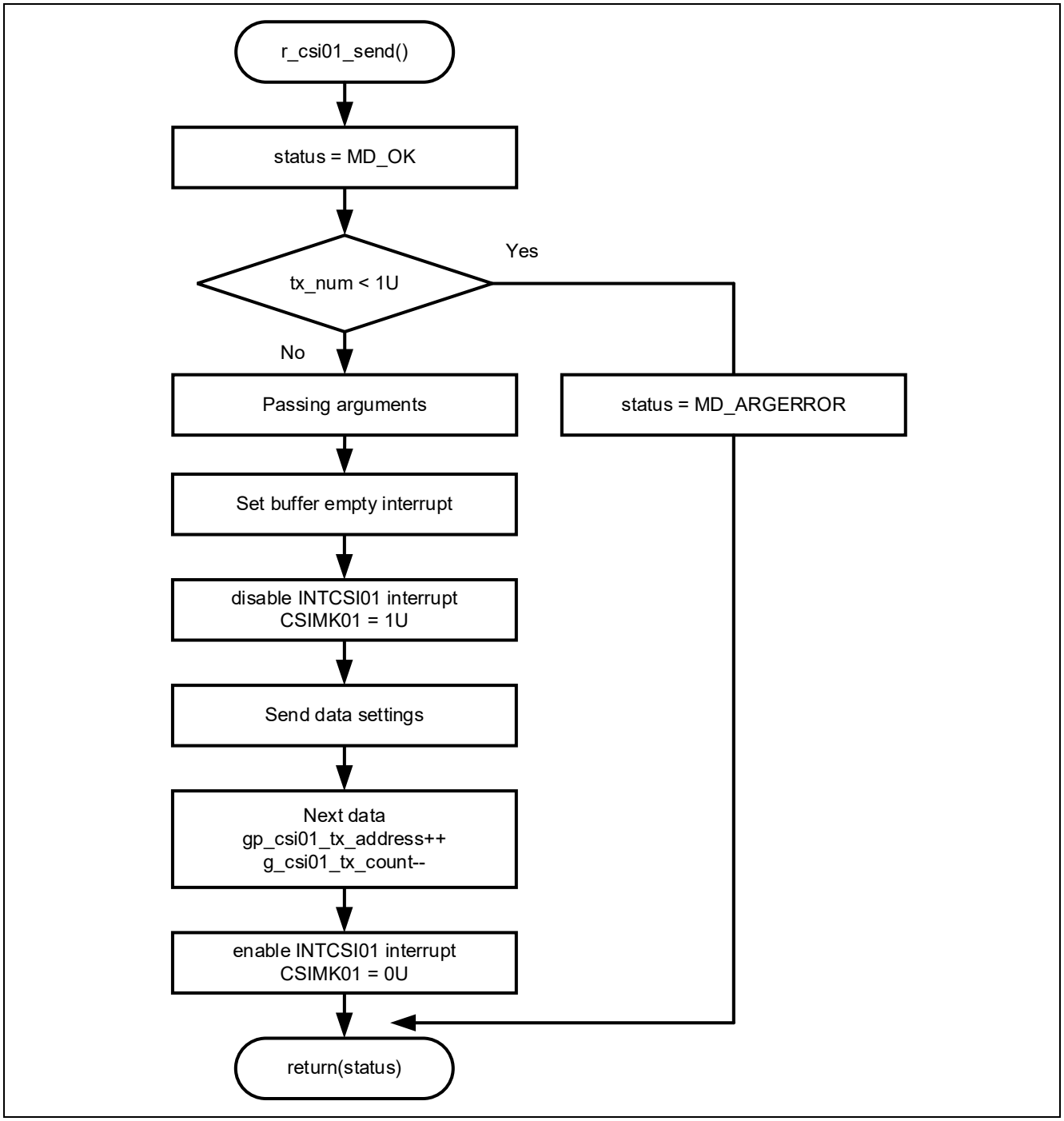
図 4-5 CSI01 停止処理



4.1.8.5 CSI01 送信開始処理

図 4-6 に CSI01 送信開始処理のフローチャートを示します。

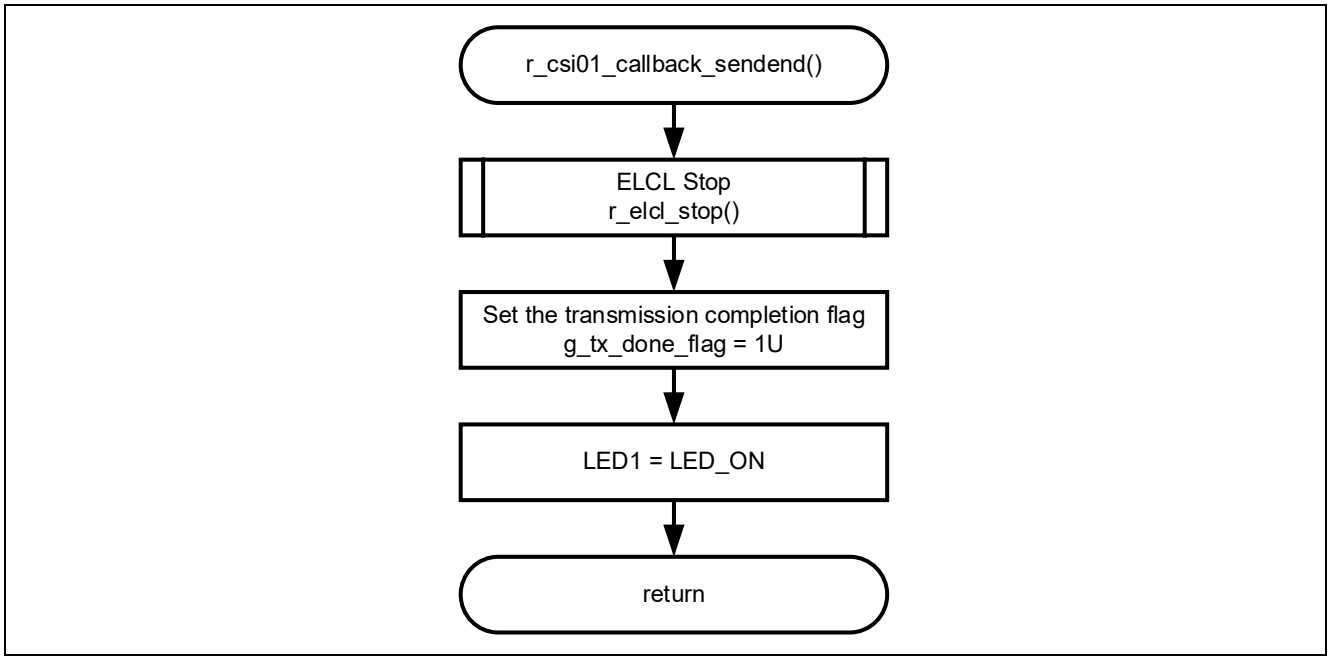
図 4-6 CSI01 送信開始処理



4.1.8.6 CSI01 送信終了のコールバック処理

図 4-7 に CSI01 送信終了のコールバック処理のフローチャートを示します。

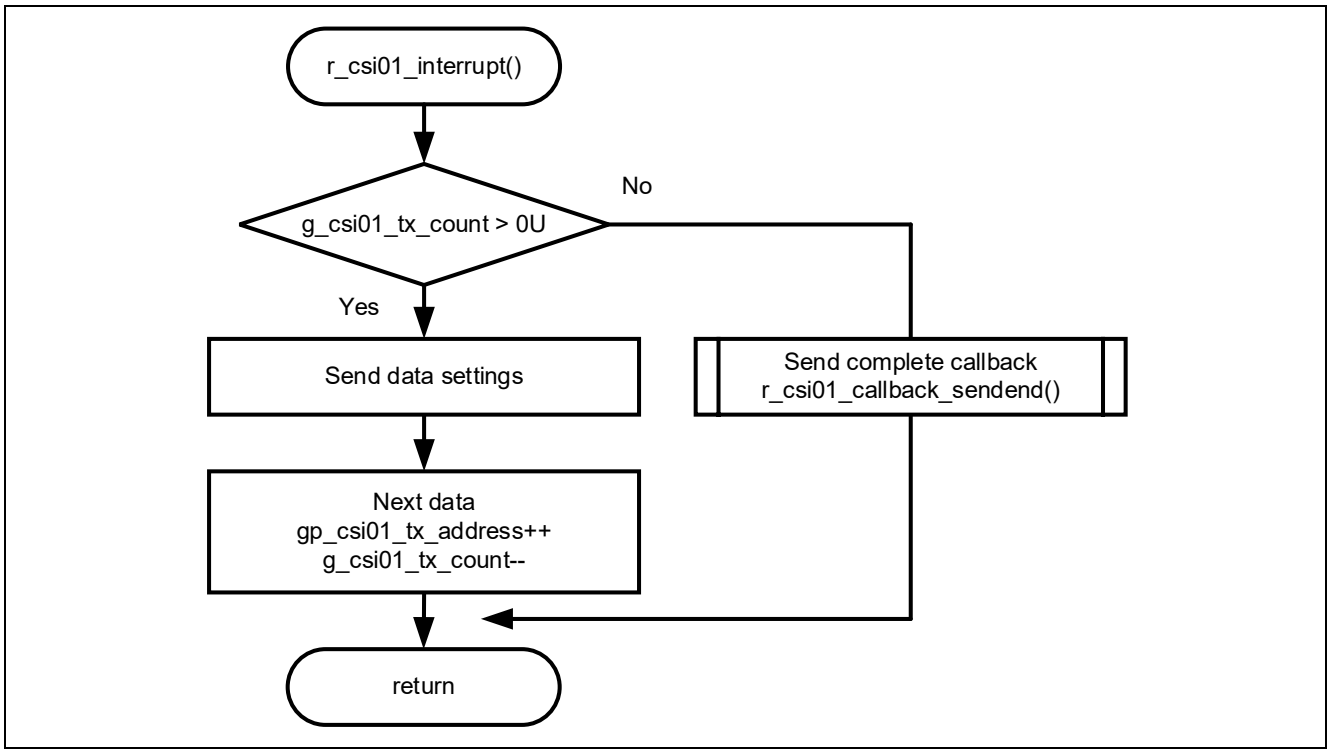
図 4-7 CSI01 送信完了のコールバック処理



4.1.8.7 CSI01 割り込み処理

図 4-8 に CSI01 割り込み処理のフローチャートを示します。

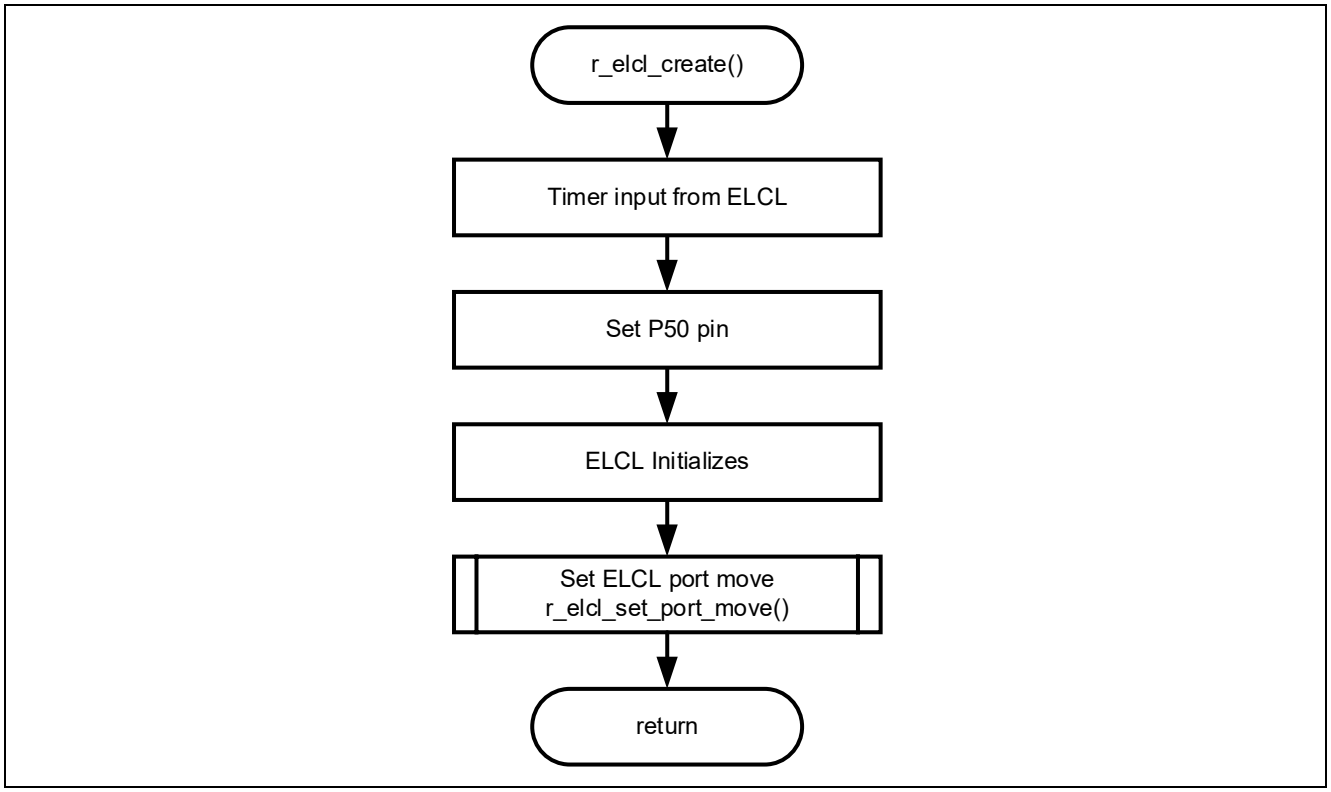
図 4-8 CSI01 割り込み処理



4.1.8.8 ELCL 初期設定処理

図 4-9 に ELCL 初期設定処理のフローチャートを示します。

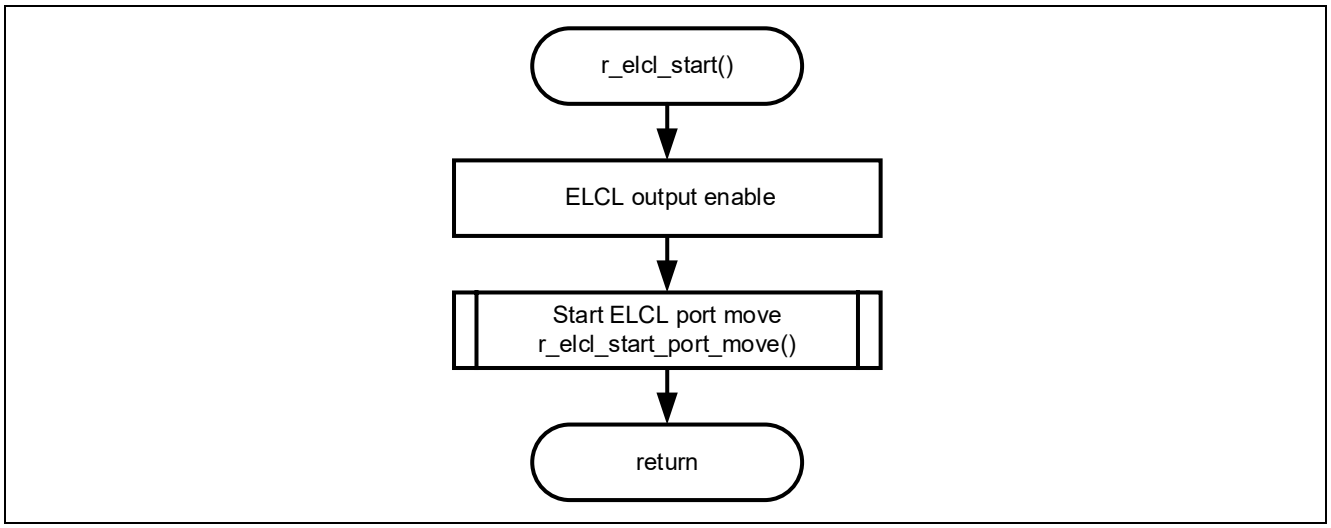
図 4-9 ELCL 初期設定処理



4.1.8.9 ELCL 出力開始処理

図 4-10 に ELCL 出力開始処理のフローチャートを示します。

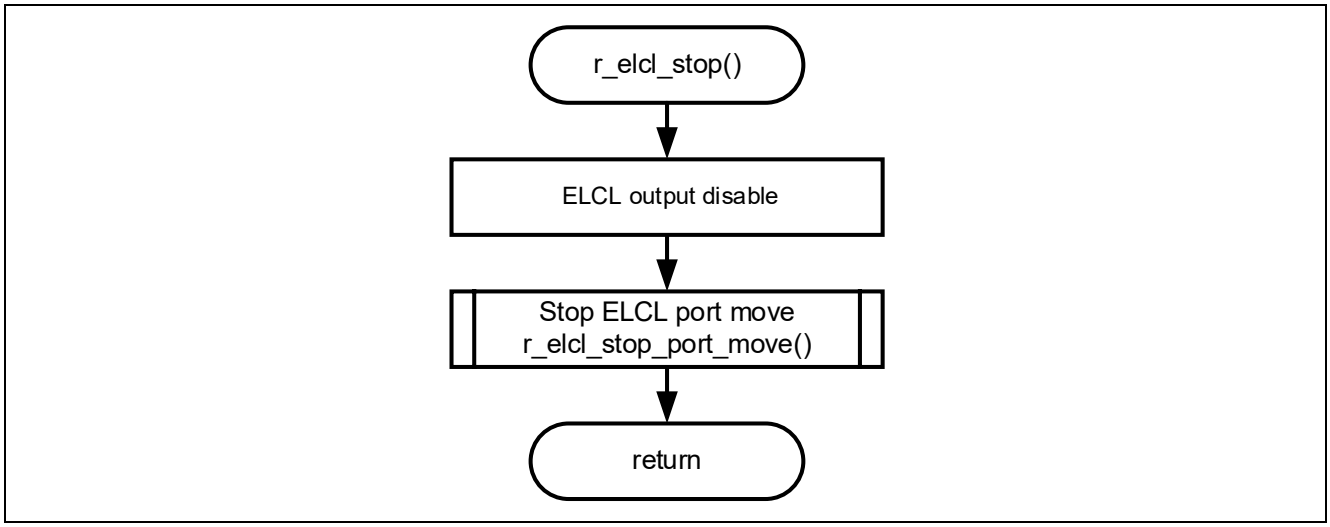
図 4-10 ELCL 出力開始処理



4.1.8.10 ELCL 停止処理

図 4-11 に ELCL 停止処理のフローチャートを示します。

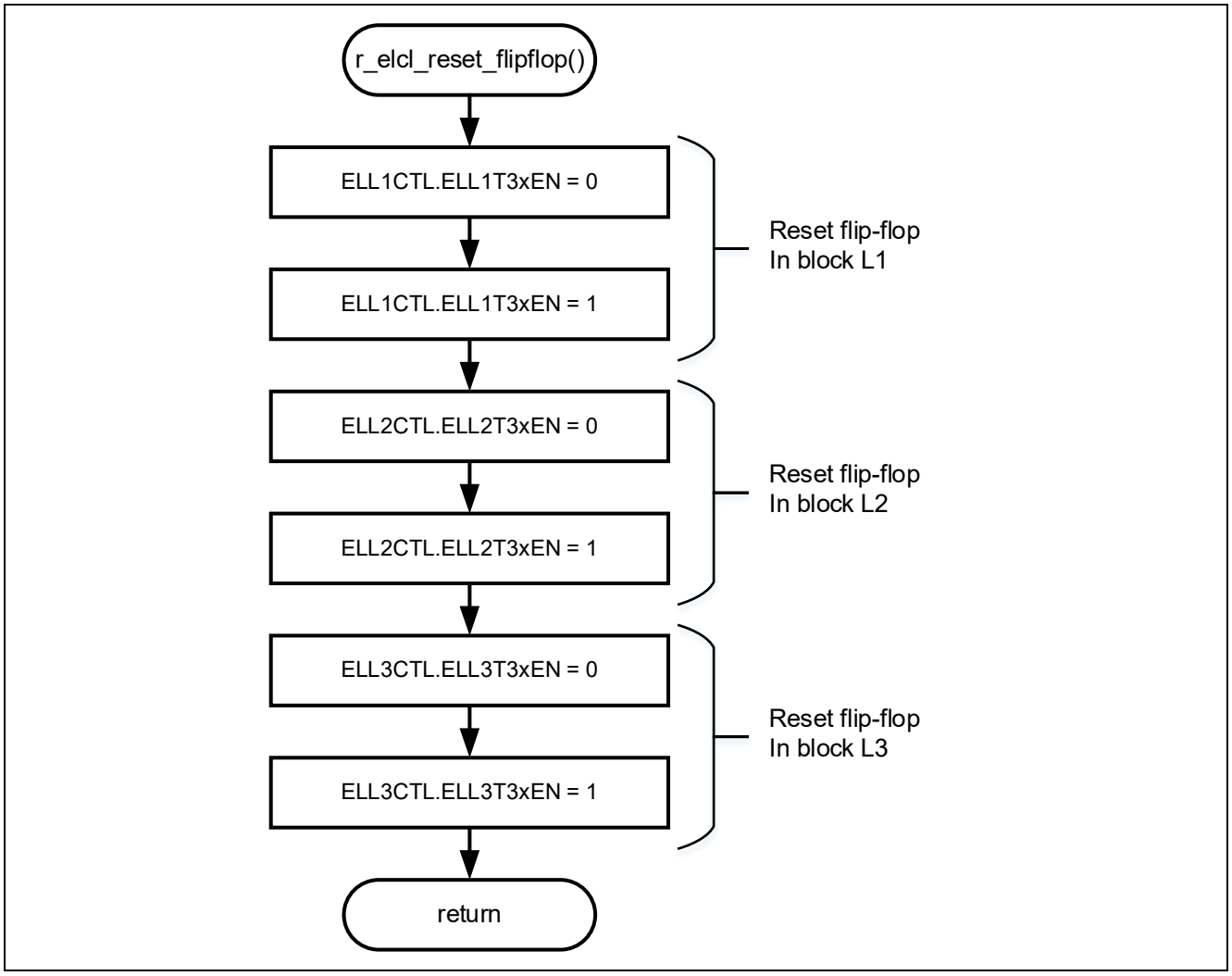
図 4-11　ELCL 停止処理



4.1.8.11 ELCL のフリップフロップリセット処理

図 4-12 に ELCL のフリップフロップリセット処理のフローチャートを示します。

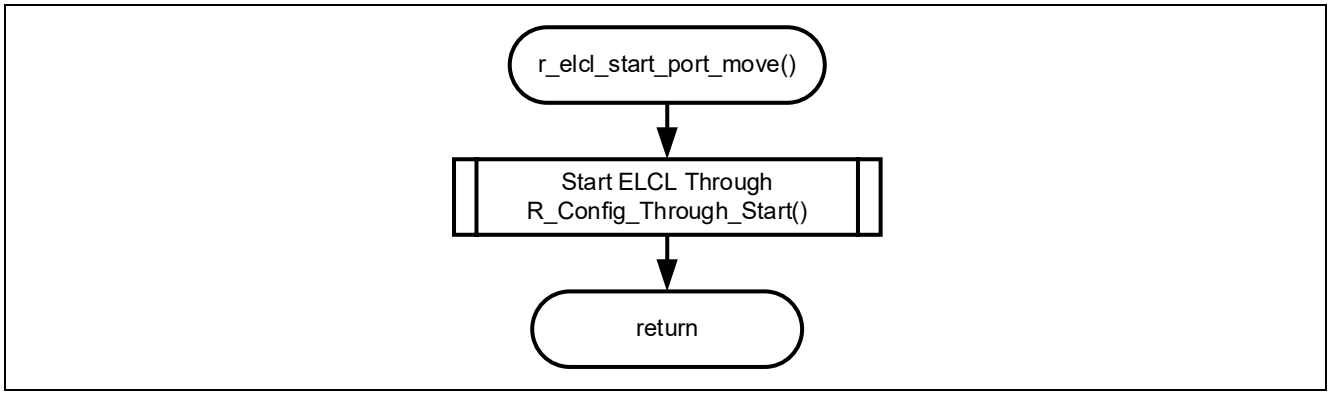
図 4-12　ELCL のフリップフロップリセット処理



4.1.8.12 ELCL の端子出力先変更開始処理

図 4-13 に ELCL の端子出力先変更開始処理のフローチャートを示します。

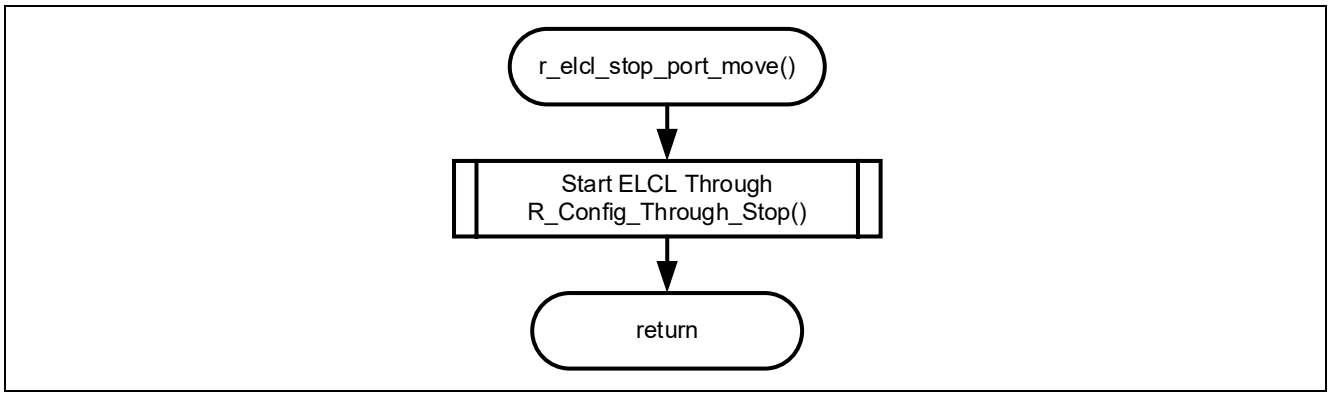
図 4-13　ELCL の端子出力先変更開始処理



4.1.8.13 ELCL の端子出力先変更停止処理

図 4-14 に ELCL の端子出力先変更停止処理のフローチャートを示します。

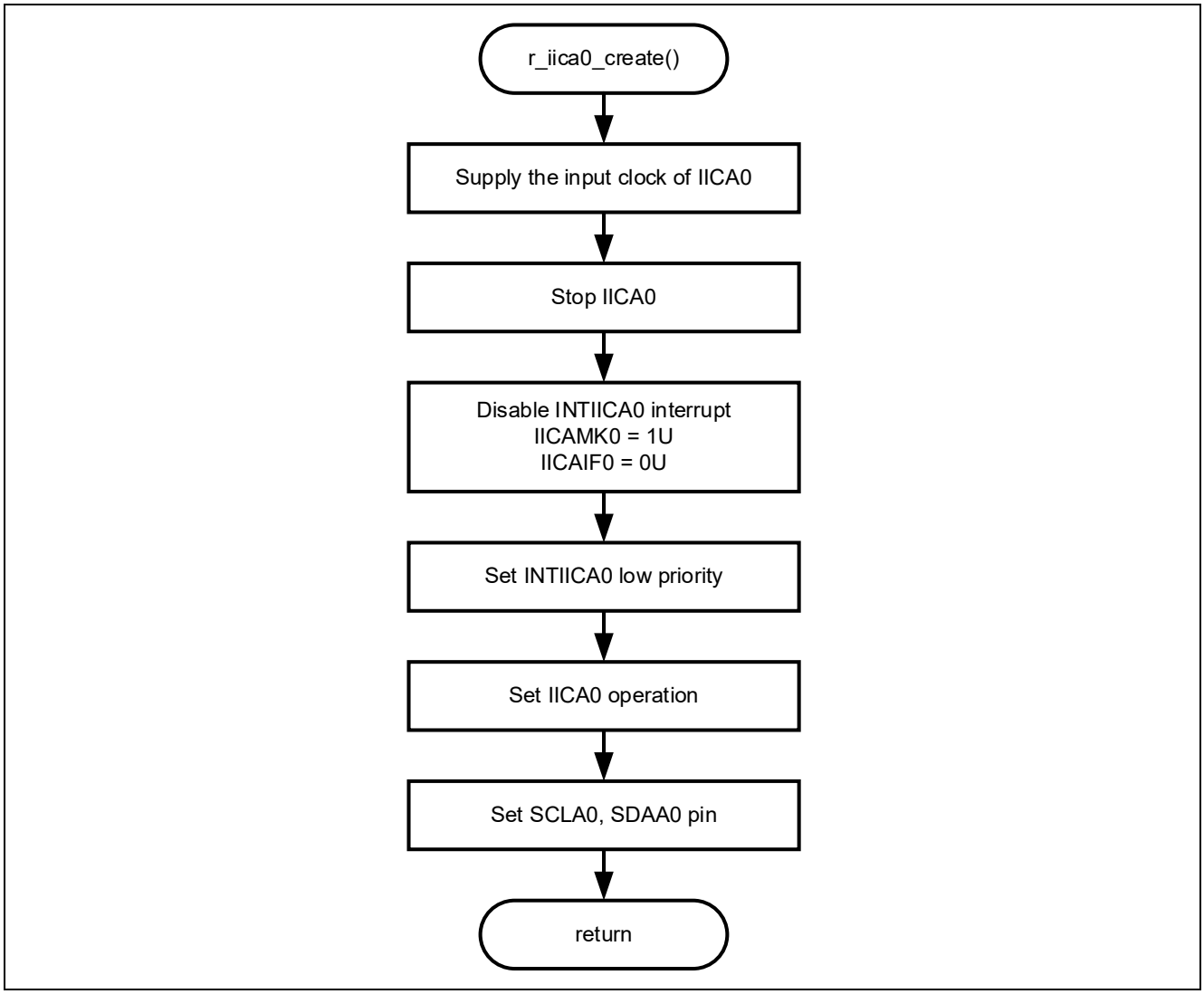
図 4-14　ELCL の端子出力先変更停止処理



4.1.8.14 IICA0 初期設定処理

図 4-15 に IICA0 初期設定処理のフローチャートを示します。

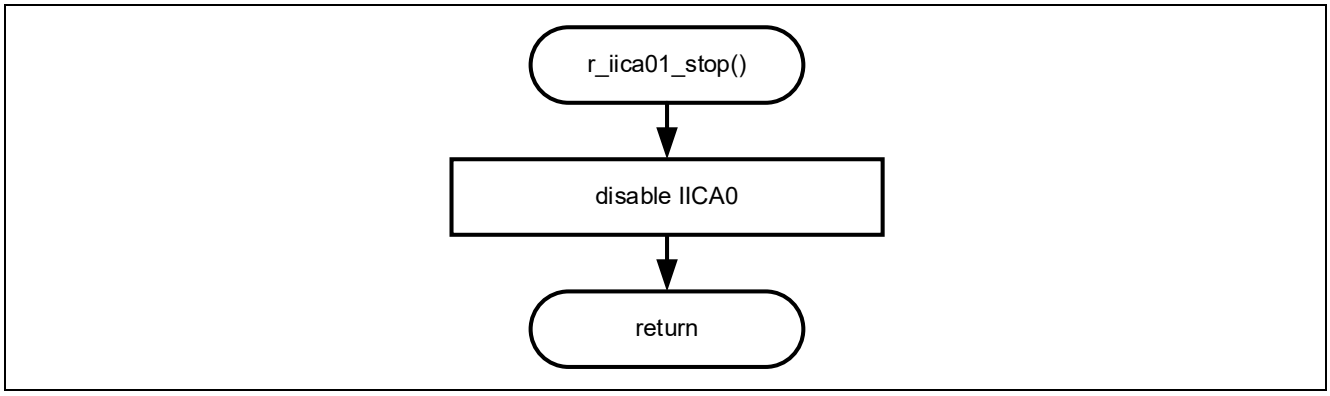
図 4-15　IICA0 初期設定処理



4.1.8.15 IICA0 停止処理

図 4-16 に IICA0 停止処理のフローチャートを示します。

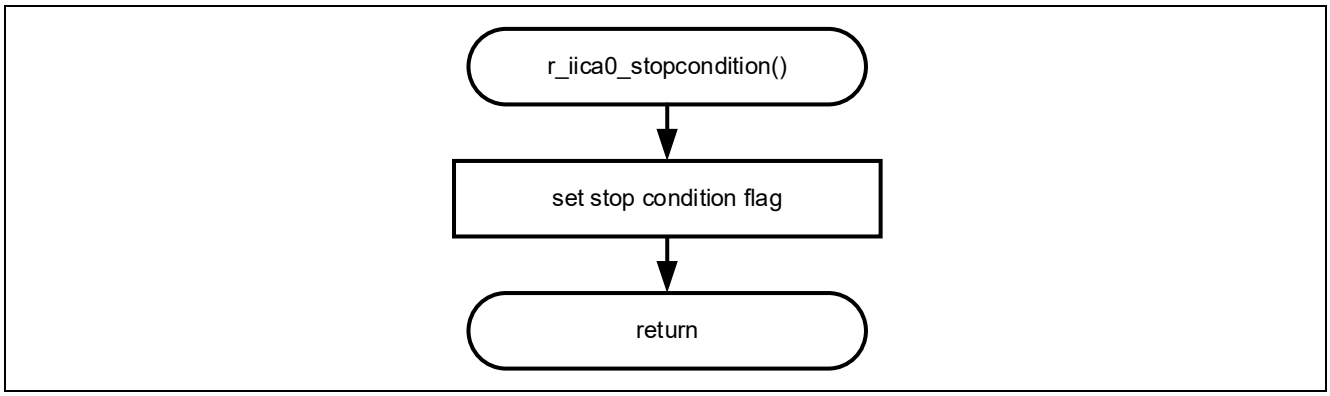
図 4-16　IICA0 停止処理



4.1.8.16 IICA0 ストップ・コンディション処理

図 4-17 に IICA0 ストップ・コンディション処理のフローチャートを示します。

図 4-17　IICA0 ストップ・コンディション処理



4.1.8.17 IICA0 マスタ送信開始処理

図 4-18、図 4-19 に IICA0 マスタ送信開始処理のフローチャートを示します。

図 4-18 IICA0 マスタ送信開始処理 (1/2)

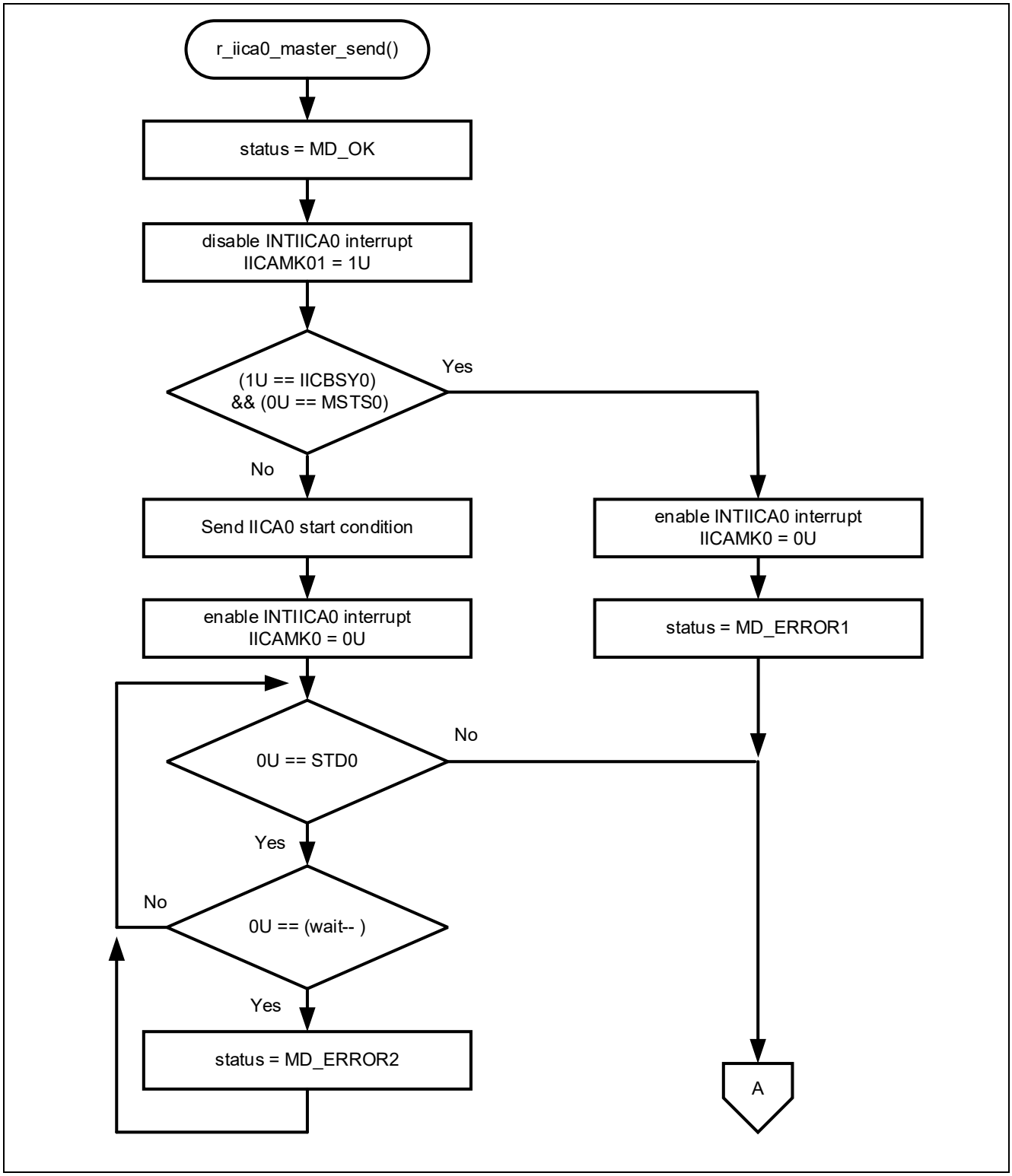
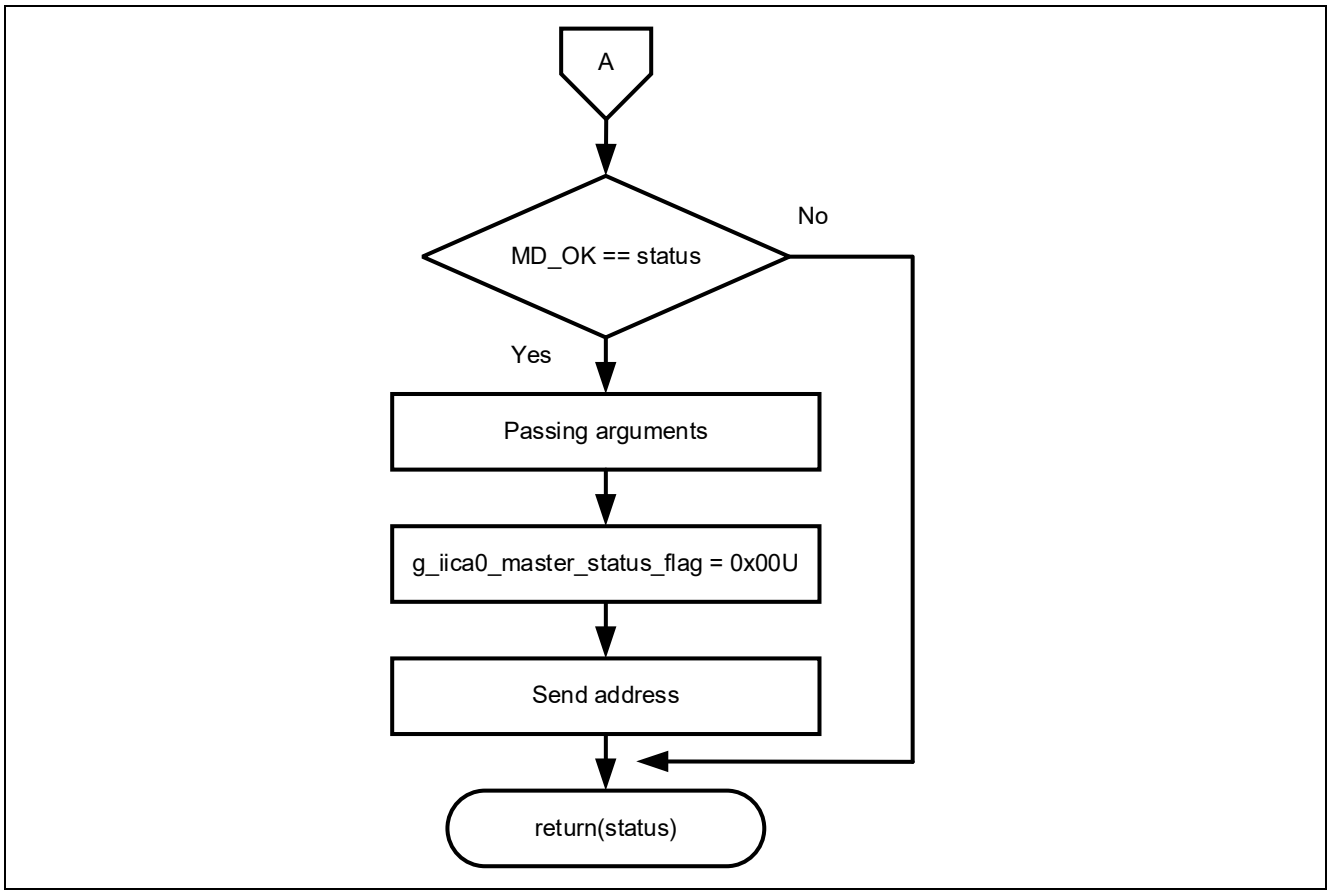


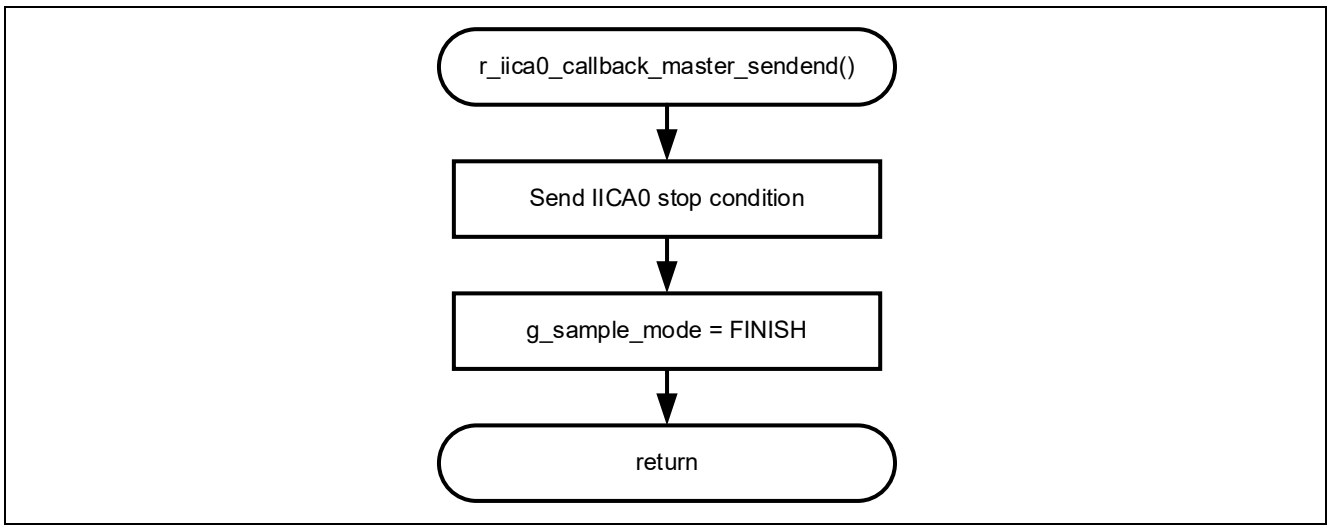
図 4-19 IICA0 マスタ送信開始処理 (2/2)



4.1.8.18 IICA0 送信完了コールバック処理

図 4-20 に IICA 送信完了コールバック処理のフローチャートを示します。

図 4-20 IICA0 送信完了コールバック処理



4.1.8.19 IICA0 割り込みハンドラ

図 4-21、図 4-22 に IICA 割り込みハンドラのフローチャートを示します。

図 4-21　IICA0 割り込みハンドラ (1/2)

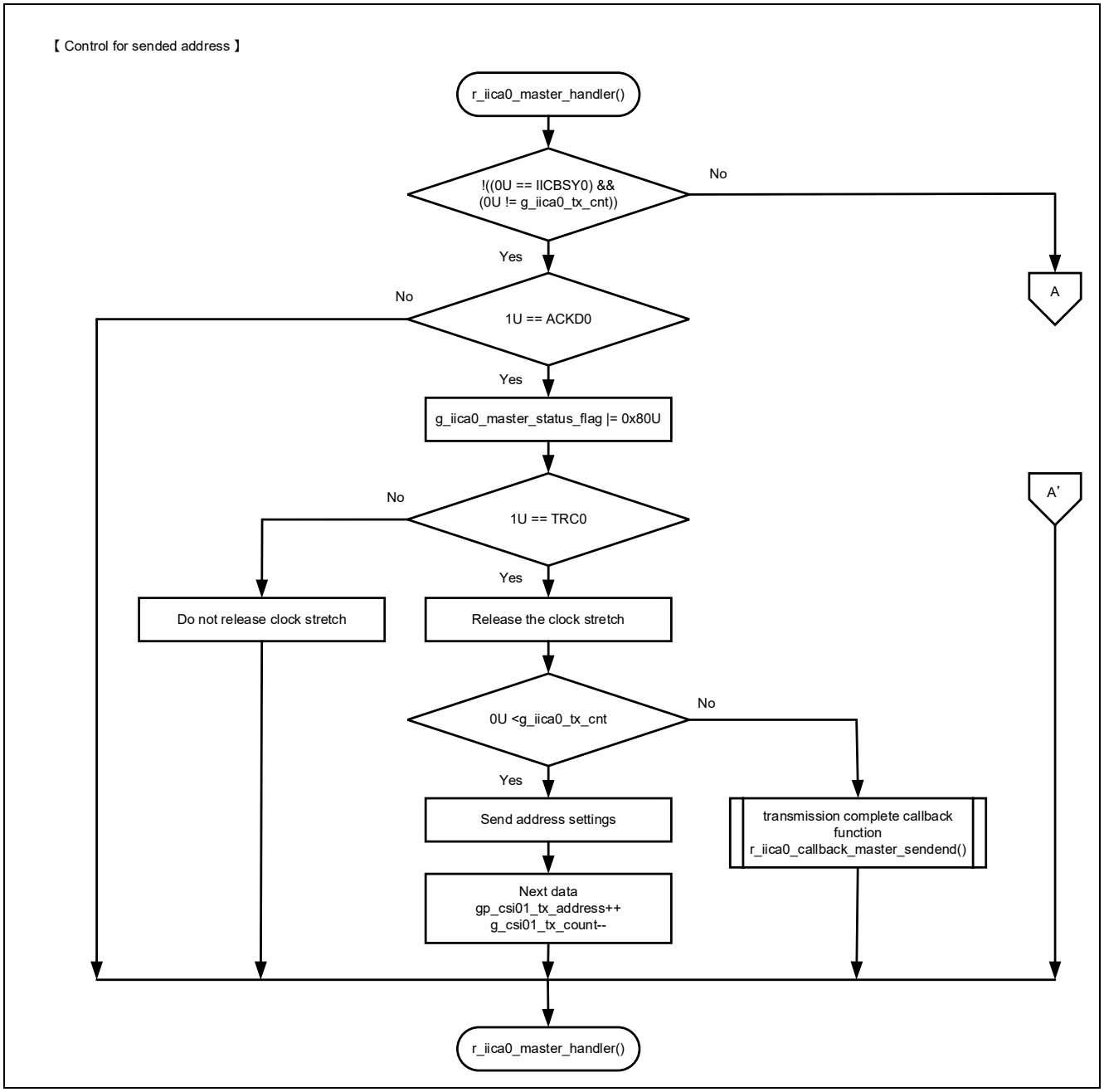
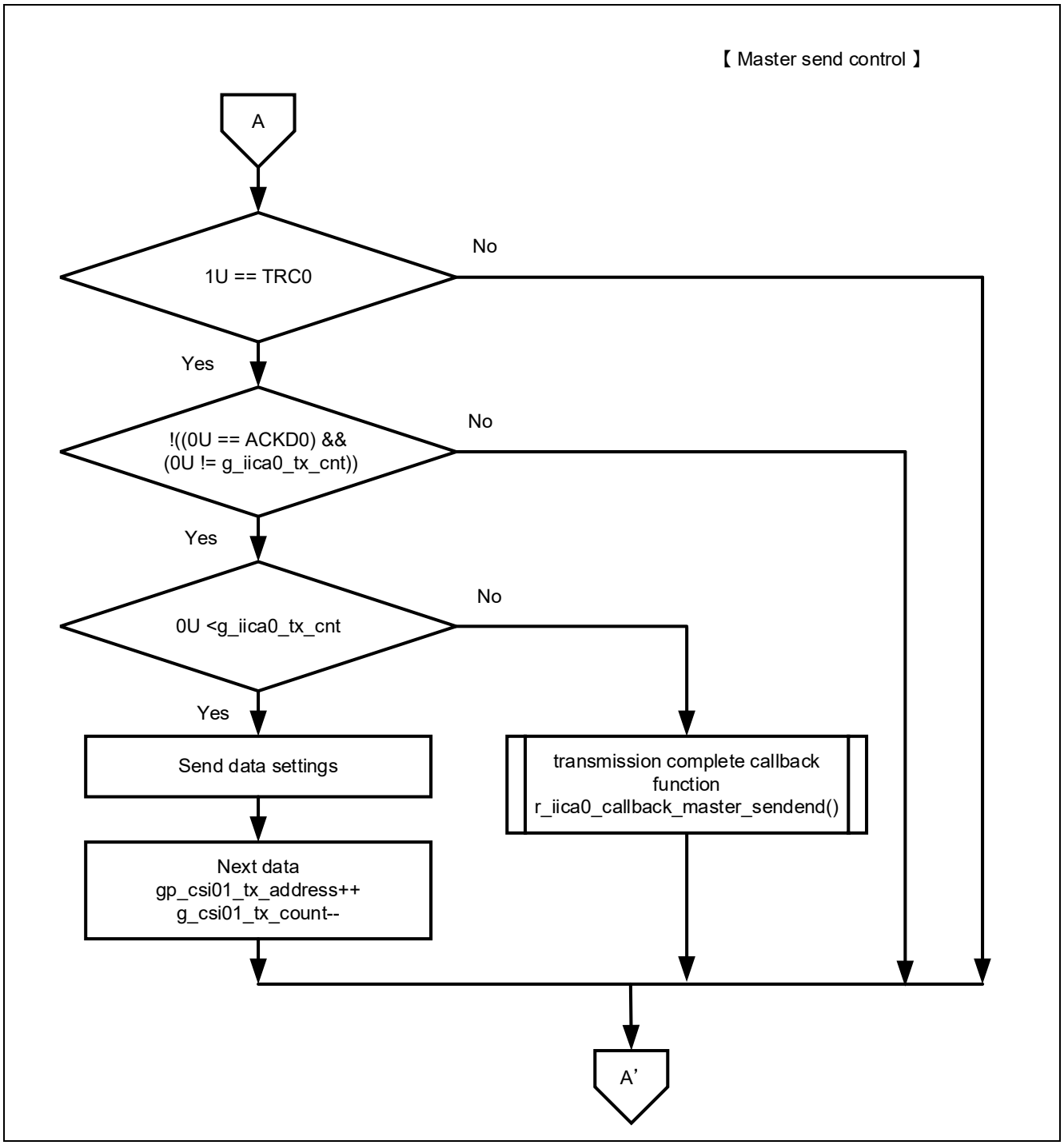


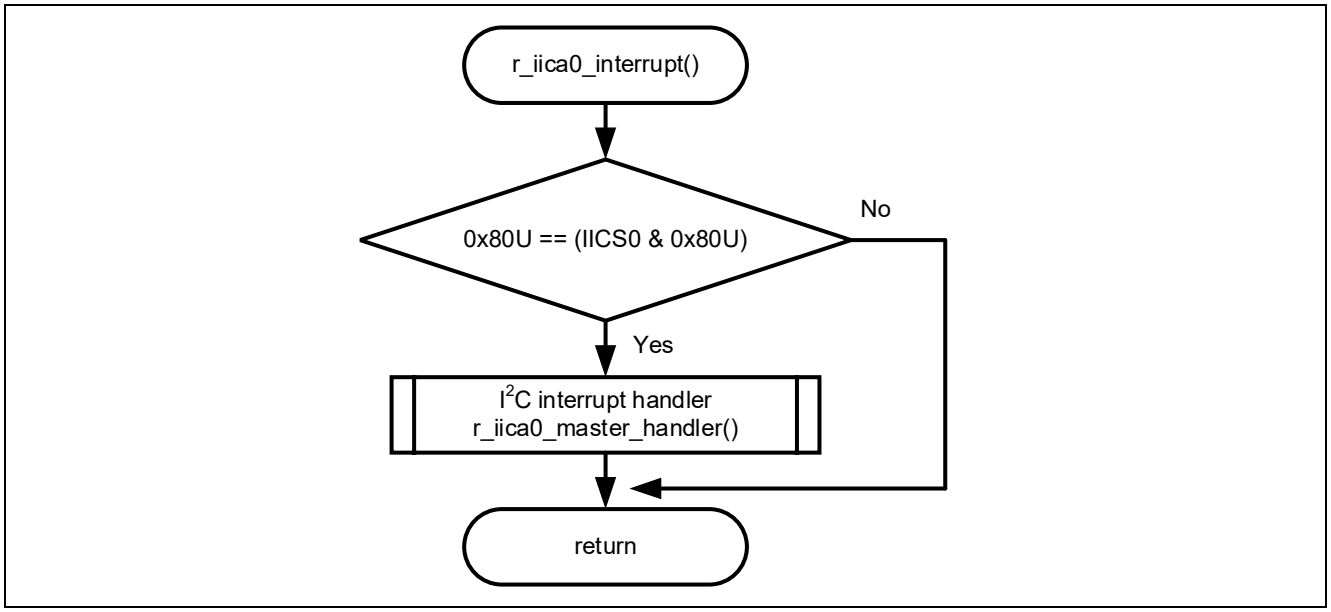
図 4-22 IICA0 割り込みハンドラ (2/2)



4.1.8.20 IICA0 割り込み処理

図 4-23 に IICA0 割り込み処理のフローチャートを示します。

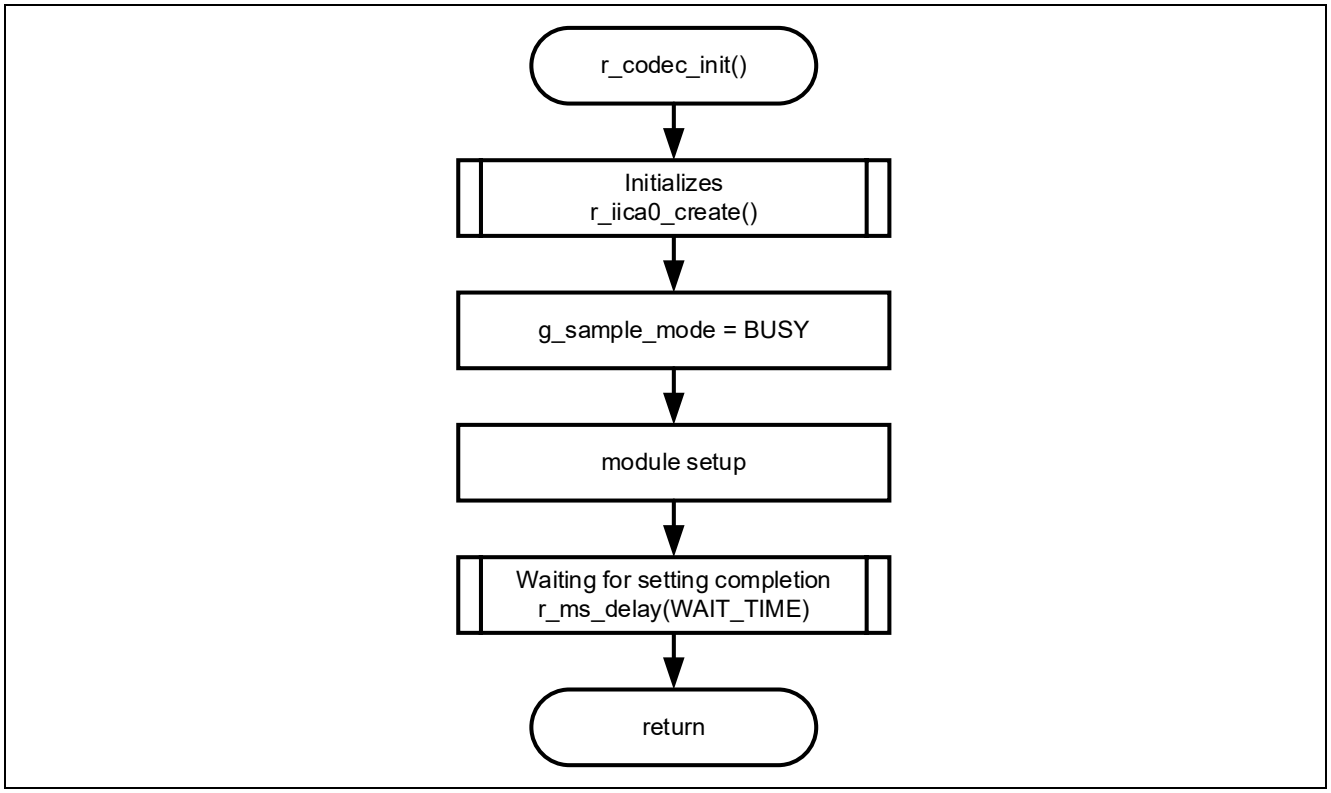
図 4-23　IICA0 割り込み処理



4.1.8.21 CODEC モジュールのセットアップ処理

図 4-24 に CODEC モジュールのセットアップ処理のフローチャートを示します。

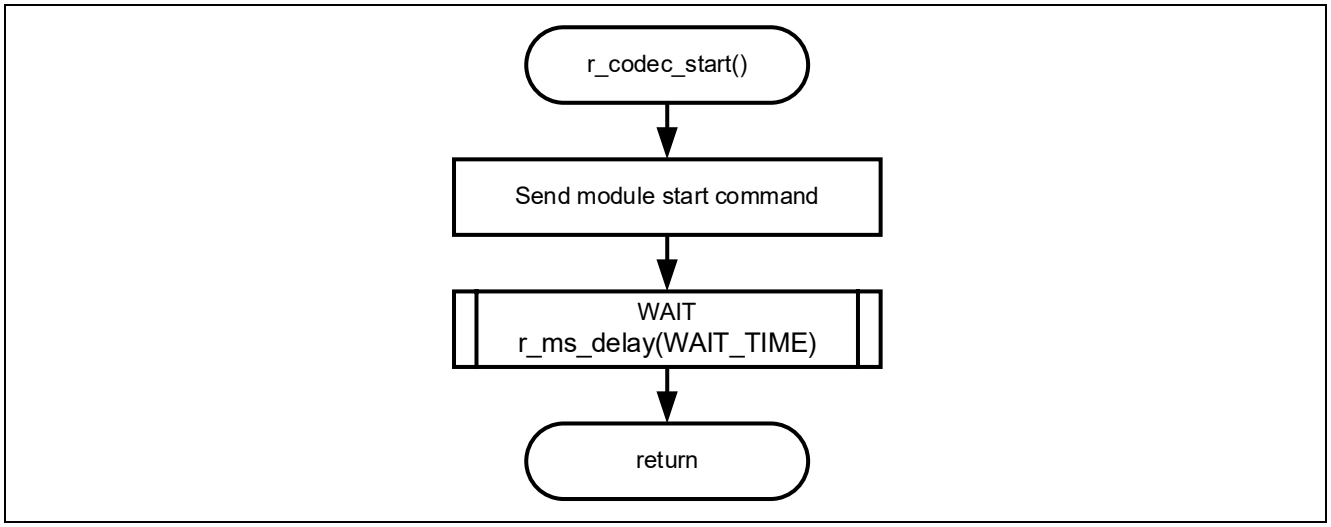
図 4-24　CODEC モジュールのセットアップ処理



4.1.8.22 CODEC モジュールの開始処理

図 4-25 に CODEC モジュールの開始処理のフローチャートを示します。

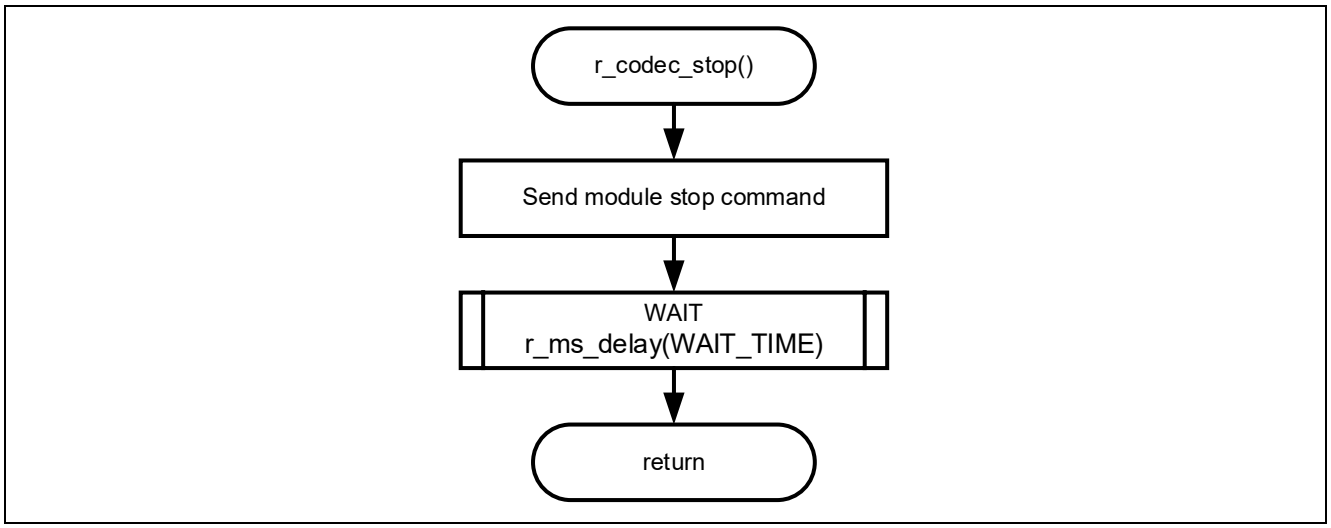
図 4-25 CODEC モジュールの開始処理



4.1.8.23 CODEC モジュールの停止処理

図 4-26 に CODEC モジュールの停止処理のフローチャートを示します。

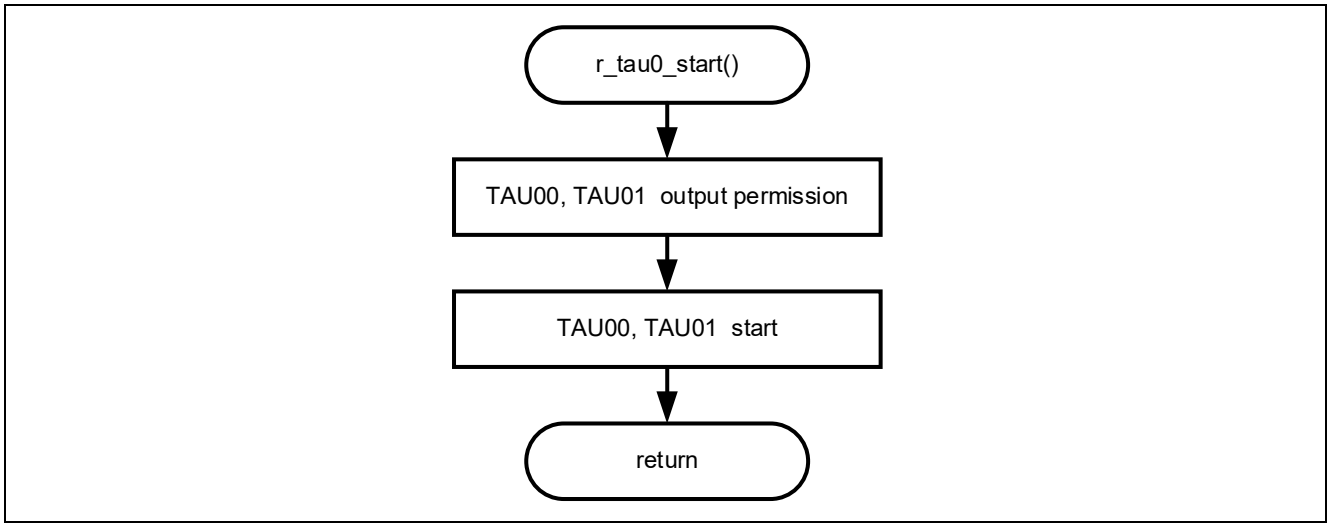
図 4-26 CODEC モジュールの停止処理



4.1.8.24 TAU0 カウンタ開始処理

図 4-27 に TAU0 カウンタ開始処理のフローチャートを示します。

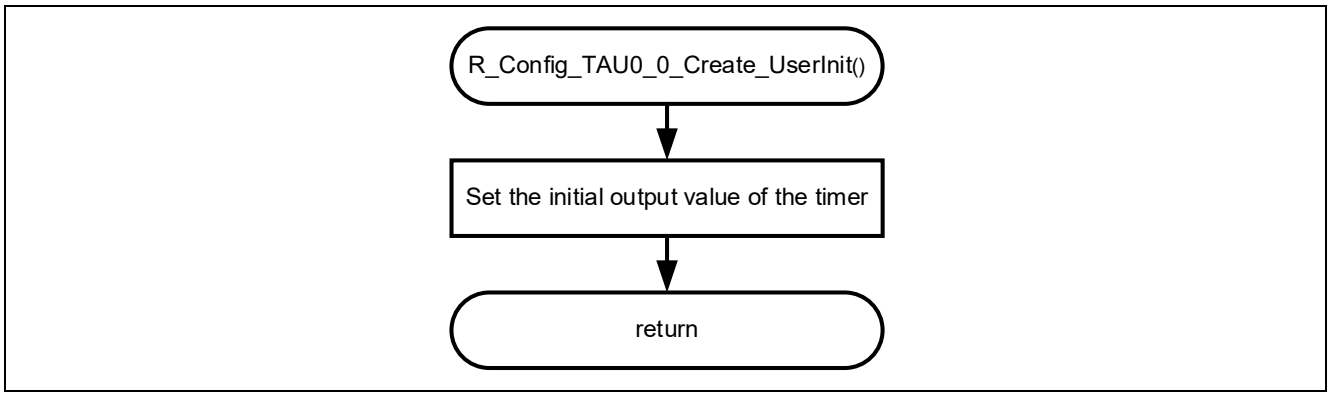
図 4-27　TAU0 カウンタ開始処理



4.1.8.25 TAU0 の初期化ユーザーコード追加

図 4-28 に TAU0 の初期化ユーザーコード追加のフローチャートを示します。

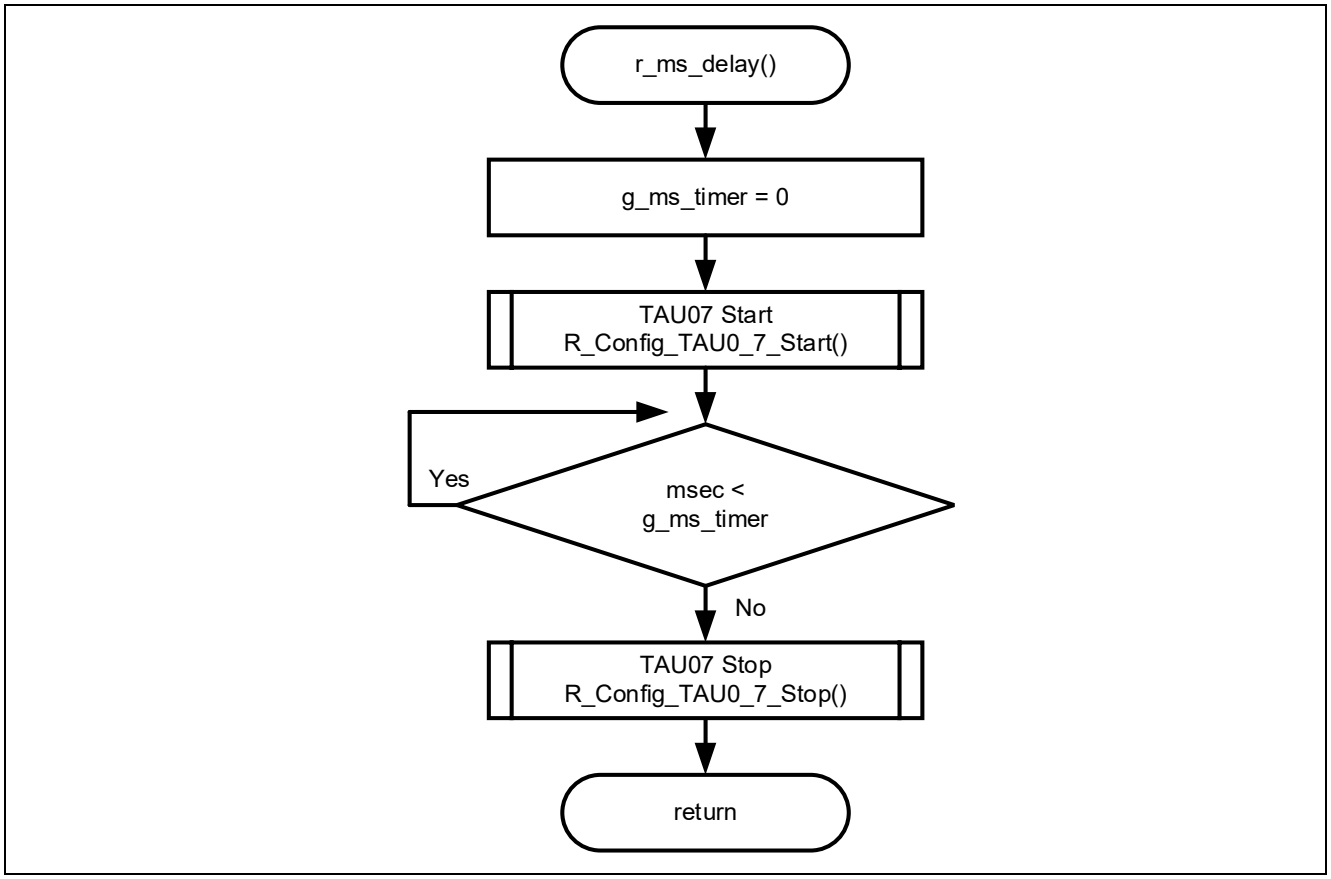
図 4-28　TAU0 の初期化ユーザーコード追加



4.1.8.26 IICA0 ウエイト処理

図 4-29 に IICA0 ウエイト処理のフローチャートを示します。

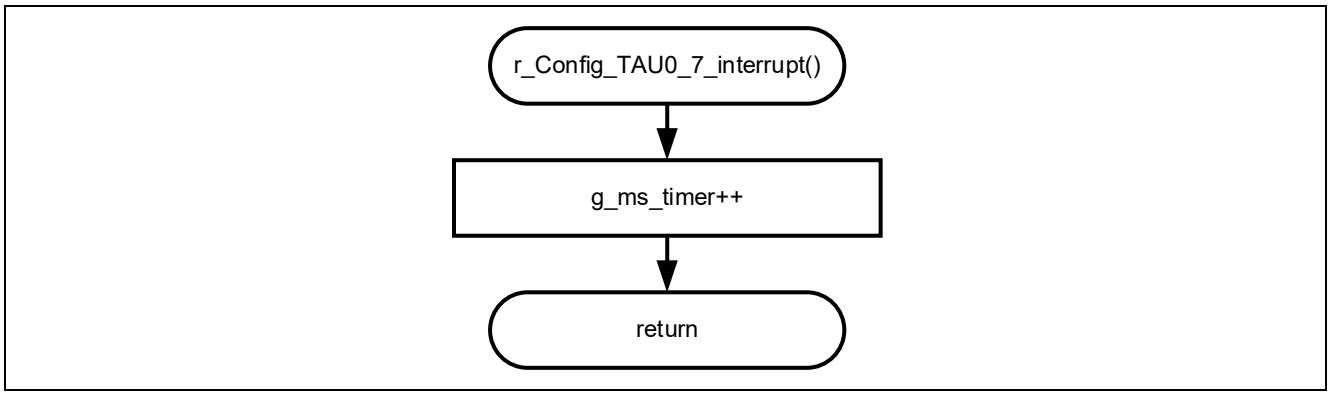
図 4-29　IICA0 ウエイト処理



4.1.8.27 TAU0 チャネル 7 割り込み処理

図 4-30 に TAU0 チャネル 7 割り込み処理のフローチャートを示します。

図 4-30　TAU0 チャネル 7 割り込み処理



4.2 応用例

本アプリケーションノートは、サンプルコードの他に以下のスマート・コンフィグレータの設定ファイルを格納しています。

r01an6420_elcl_i2s_master.scfg

ファイルの説明と使用する上での設定例および注意事項を以下に示します。

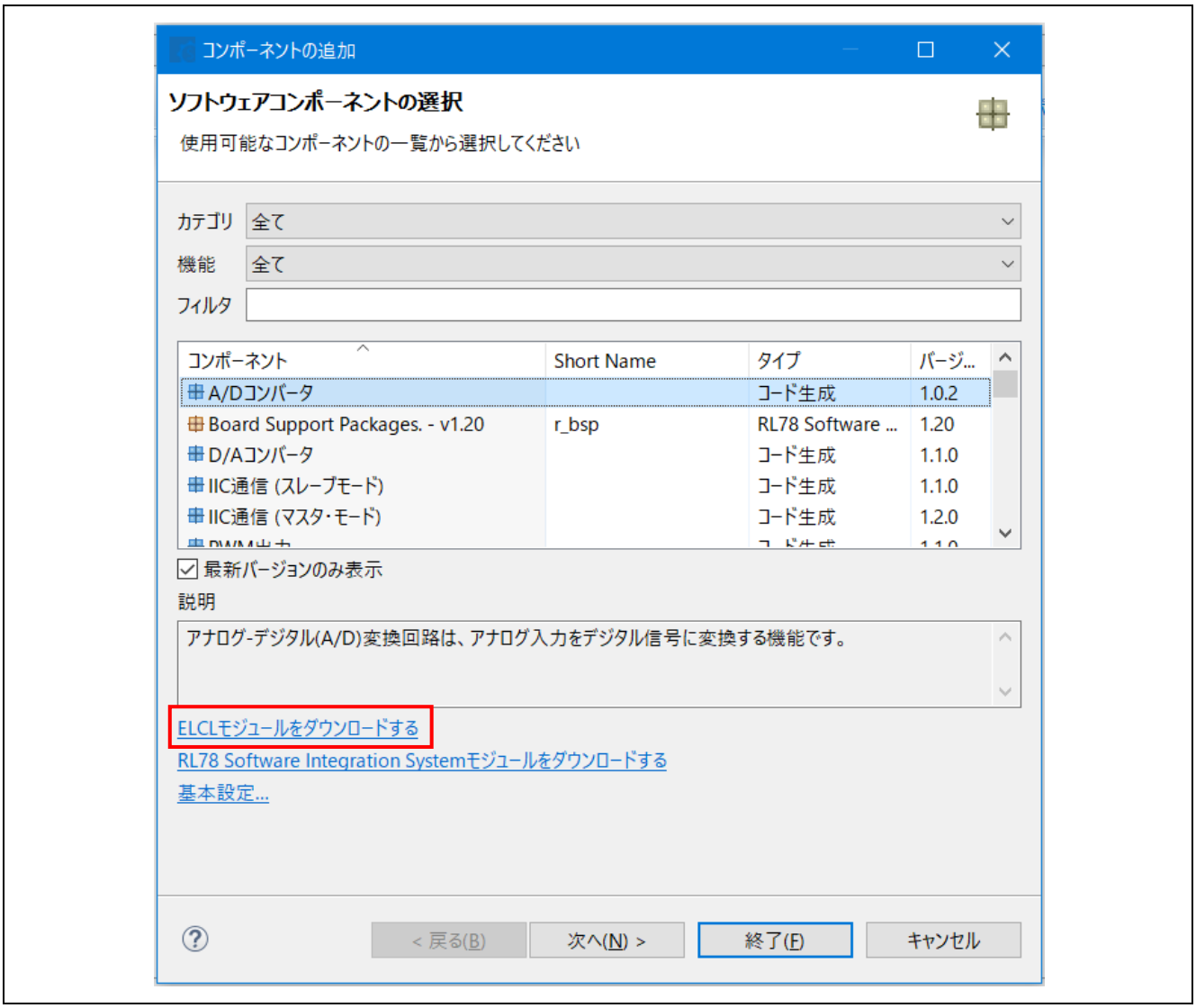
4.2.1 ELCL のコンポーネントの設定

ELCL コンポーネントを使用するためには ELCL コンテンツファイルのインストールが必要です。

手順を以下に示します。

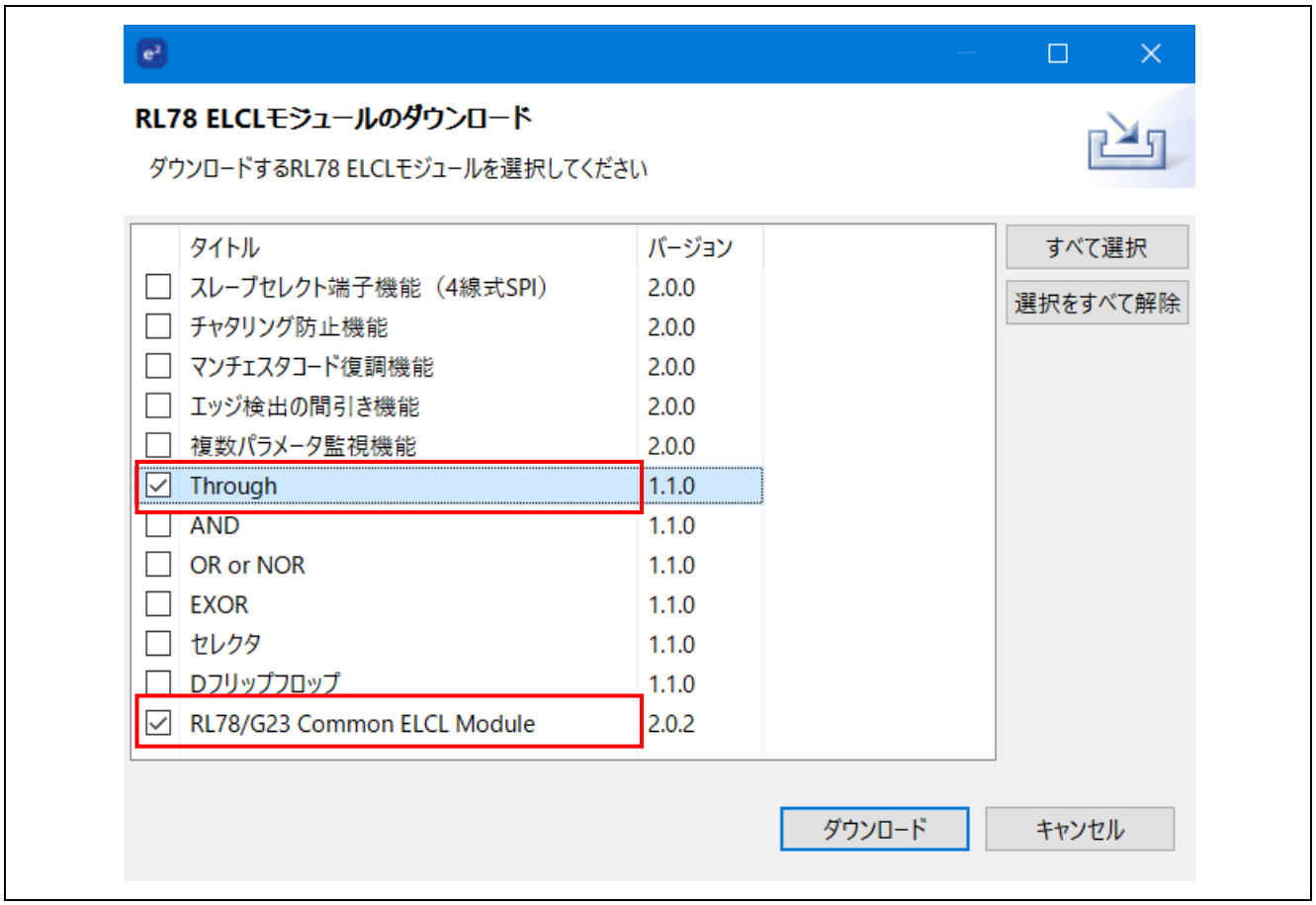
1. スマート・コンフィグレータを起動してください。
2. 「コンポーネント」タグをクリックし、「コンポーネントの追加」をクリックしてください。
3. 図 4-31 に示す「コンポーネントの追加」のウィンドウが開きますので、「ELCL モジュールをダウンロードする」をクリックしてください。

図 4-31 コンポーネントの選択



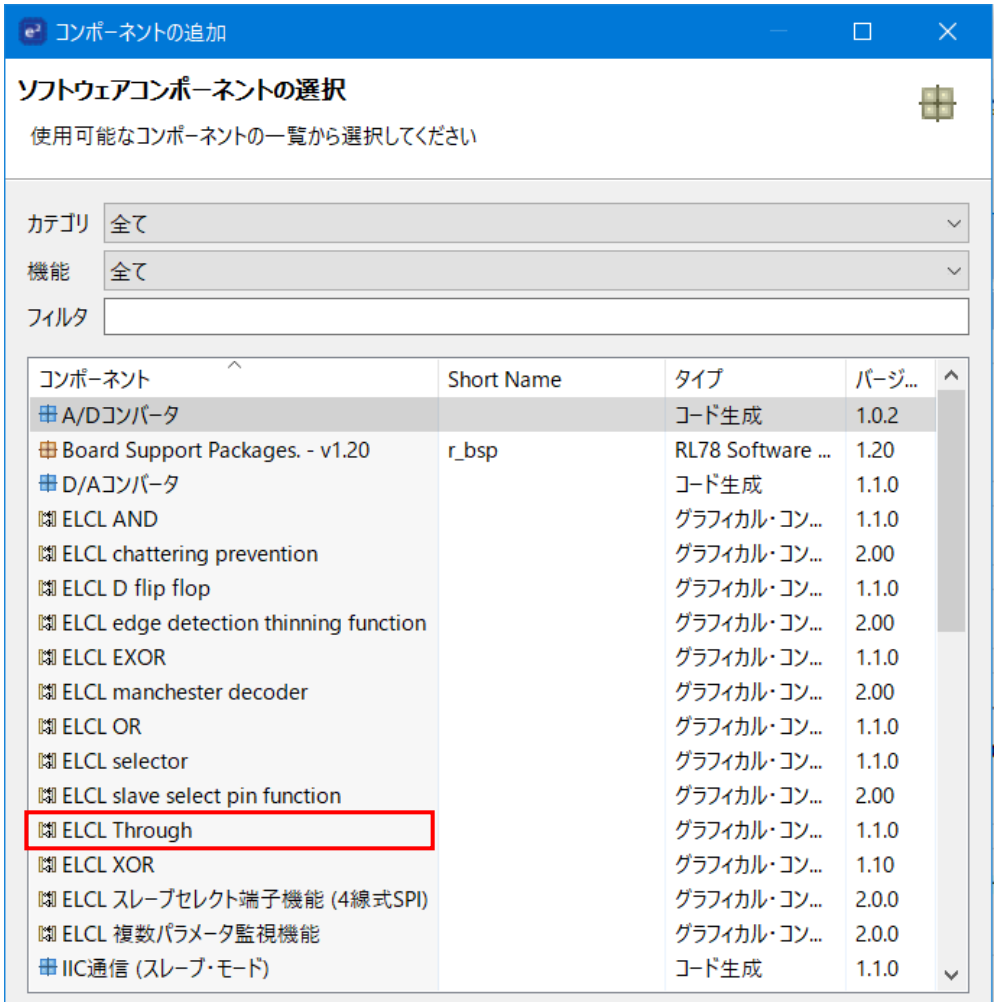
4. 「Through」を選択し、ダウンロードしてください。共通設定ファイル「RL78/G23 Common ELCL Module」もダウンロードしてください。

図 4-32 モジュールのダウンロード



5. ダウンロード完了後、「ELCL Through」が選択できることを確認してください。

図 4-33 モジュールの選択



4.2.2 r01an6420_elcl_i2s_master.scfg

サンプルコードで使用しているスマート・コンフィグレータの設定ファイルです。スマート・コンフィグレータで設定されている全ての機能が含まれています。サンプルコードの設定は以下の通りです。

表 4-8 スマート・コンフィグレータの設定値

タグ名	コンポーネント	内容
クロック	-	動作モード：高速メインモード 2.4 (V) ~5.5 (V) EV_{DD} 設定：1.8V≦EV_{DD0}＜5.5V 高速オンチップ・オシレータ：32MHz f_{IHP}：32MHz f_{CLK}：32000kHz（高速オンチップ・オシレータ） f_{SXP}：32.768kHz（低速オンチップ・オシレータ）
システム	-	オンチップ・デバッグ動作設定：COM ポート[注 1] 疑似 RRM/DMM 機能設定：使用する Start/Stop 関数機能設定：使用しない トレース機能設定：使用する セキュリティ ID 設定：セキュリティ ID を設定する セキュリティ ID：0x00000000000000000000 セキュリティ ID 認証失敗時の設定：フラシュ・メモリのデータを消去しない
コンポーネント	r_bsp	Start up select : Enable (use BSP startup) Control of invalid memory access detection : Disable RAM guard space (GRAM0-1) : Disabled Guard of control registers of port function (GPORT) : Disabled Guard of registers of interrupt function (GINT) : Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC) : Disabled Data flash access control (DFLEN) : Disables Initialization of peripheral functions by Code Generator/Smart Configurator : Enable API functions disable : Enable Parameter check enable : Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode : High-speed Enable user warm start callback (PRE) : Unused Enable user warm start callback (POST) : Unused Watchdog Timer refresh enable : Unused
	Config_LVD0	動作モード設定：リセット・モード 電圧検出設定：リセット発生電圧 (V_{LVD0})：1.86 (V)

表 4-9　スマート・コンフィグレータの設定値

タグ名	コンポーネント	内容
コンポーネント	Config_TAU0_0	コンポーネント：インターバル・タイマ 動作モード：16 ビット・カウンタ・モード リソース：TAU0_0 動作クロック：CK00 クロックソース：f_{CLK} インターバル時間：0.163us カウント開始時に INTTM00 割り込みを発生する 割り込み設定：使用しない
	Config_TAU0_1	コンポーネント：外部イベント・カウント リソース：TAU0_1 動作クロック：CK00 クロックソース：f_{CLK} 入力ソース設定：ELCL 動作モード設定：16 ビット 外部イベント・エッジ選択（TI01）：立ち上がりエッジ カウント値：32 割り込み設定：使用しない
	Config_TAU0_7	コンポーネント：インターバル・タイマ 動作モード：16 ビット・カウンタ・モード リソース：TAU0_7 動作クロック：CK01 クロックソース：$f_{CLK}/2^8$ インターバル時間：1ms 割り込み設定：使用する 優先順位：レベル 2
	Config_Through	コンポーネント：ELCL Through Common setting：L3L0 Detail setting：L3L0 Input signal selector：ELISEL_5 , TO00 Application：Through Output signal selector：P51
	Config_PORT	コンポーネント：ポート ポート選択：PORT5 P53：出力（1 を出力）

4.2.2.1　クロック

サンプルコードで使用するクロックの設定を行います。

4.2.2.2 システム

サンプルコードのオンチップ・デバッグ設定を行います。

「オンチップ・デバッグ動作設定」、「セキュリティ ID 認証失敗時の設定」は、「表 4-3 オプション・バイト設定」の「オンチップ・デバッグ動作許可」に影響を与えます。設定を変更する際は注意してください。

4.2.2.3 r_bsp

サンプルコードのスタートアップの設定を行います。

4.2.2.4 Config_LVD0

サンプルコードの電源管理の設定を行います。

「表 4-3 オプション・バイト設定」の「LVD0 の設定」に影響を与えます。設定を変更する際は注意してください。

4.2.2.5 Config_TAU00

サンプルコードの TAU00 の設定を行います。

サンプルコードでは、「インターバル・タイマ」として使用し、BCLK を生成します。サンプリング周波数とデータサイズに合わせて、図 4-34 のインターバル時間に、表 4-10 の値を設定してください。赤字はサンプルコードでデータが準備されている設定です。

表 4-10 インターバル時間の設定値

データサイズ	サンプリング周波数（kHz）									
	8	11	12	16	22.1	24	32	44.1	48	96
16bit	1.953	1.417	1.302	0.977	0.709	0.651	0.488	0.354	0.326	0.163
24bit	1.302	0.945	0.868	0.651	0.472	0.434	0.326	0.236	0.217	—
32bit	0.977	0.709	0.651	0.488	0.354	0.326	0.244	0.177	0.163	—

図 4-34 TAU0 チャネル 0 設定（サンプリング周波数 48kHz、データサイズ 32bit の場合）

4.2.2.6 Config_TAU01

サンプルコードの TAU01 の設定を行います。

サンプルコードでは、「外部イベント・カウント」として使用し、LRCLK を生成します。データサイズに合わせて、図 4-35 のカウント値には、表 4-11 の値を設定してください。

表 4-11 カウント値

データサイズ	カウント値
16bit	16
24bit	24
32bit	32

図 4-35 TAU0 チャネル 1 設定（データサイズ 32bit の場合）

4.2.2.7 Config_TAU07

サンプルコードの TAU07 の設定を行います。

サンプルコードでは 1ms のインターバル・タイマとして使用します。

4.2.2.8 Config_Through

サンプルコードの BCLK の出力端子を変更するために設定を行います。

BCLK として使用する TO00 の出力先を P51 に設定します。

4.2.2.9 Config_PORT

サンプルコードのポートの設定を行います。

サンプルコードでは LED1 の制御に P53 を使用します。

4.2.3 サウンド・データの変更方法

4.2.3.1 サンプルコード内のサウンド・データを変更する場合

本サンプルコードは以下の条件のサウンド・データが準備されています。

サンプリング周波数：48kHz、データ数：32bit

サンプリング周波数：44.1kHz、データ数：24bit

サンプリング周波数：8kHz 、データ数：16bit

サンプルコードの sound_data.h で再生するサウンド・データを変更できます。

初期設定では、以下のようにサンプリング周波数：48kHz、データ数：32bit のサウンド・データが設定されています。

```
#define DATA_48K_32B (1)
#define DATA_44K_24B (0)
#define DATA_8K_16B  (0)
```

再生したいサウンド・データのみ 1、それ以外を 0 に設定することでサウンド・データを変更できます。設定した条件に合わせて、表 4-10、表 4-11 を参考に TAU00 と TAU01 を変更してください。

以上の設定を行うことでサンプルコード内のサウンド・データを変更することができます。

4.2.3.2 サンプルコードに新規のサウンド・データを追加する場合

本サンプルコードでは、新規のサウンド・データを追加できます。

(1) サンプルコードの sound_data.c の以下の g_tx_buf[]に新規のサウンド・データを追加してください。

```
#else    /* Not DATA_8K_16B_DATA */
/* For user setting */
const uint8_t g_tx_buf[] =
       {
              /* Add sound data here */
       };

#endif /* DATA_48K_32B */
```

(2) サンプルコードの sound_data.h で再生するサウンド・データを以下のようにすべて 0 に設定してください。

```
#define DATA_48K_32B (0)
#define DATA_44K_24B (0)
#define DATA_8K_16B  (0)
```

(3) 同ファイルの送信データ数定数「DECODE_PCM_SIZE」を設定してください。

```
#else /* For user setting */
#define DECODE_PCM_SIZE    (0)

#endif
```

(4) サンプルコードの r_codec.c の以下の部分にサンプリング周波数とデータ数の設定を行ってください。

```
#else /* For user setting */
uint8_t g_cmd_set_rate[2] =
       { 0x10, 0x00 }; /* Set sampling frequency */
uint8_t g_cmd_set_format[2] =
       { 0x0E, 0x0E }; /* Set input bit length */
#endif
```

表 4-12 サンプリング周波数の設定値

サンプリング周波数	g_cmd_set_rate の設定値
96kHz	0x10, 0x1C
48kHz	0x10, 0x00
44.1kHz	0x10, 0x20
32kHz	0x10, 0x18
8kHz	0x10, 0x0C

表 4-13 データ数の設定値

データ数	g_cmd_set_format の設定値
16bit	0x0E, 0x02
24bit	0x0E, 0x0A
32bit	0x0E, 0x0E

(5) 設定した条件に合わせて、表 4-10、表 4-11 を参考に TAU00 と TAU01 を変更してください。

以上の設定を行うことでサンプルコードに新規のサウンド・データを追加することができます。

4.2.4 他の CODEC モジュールを使用する場合

RS 社の Audio CODEC 754-1974 以外の CODEC モジュールを使用する場合、表 4-1、表 4-2 の RS 社の Audio CODEC 754-1974 モジュールの設定に必要なファイルを削除してください。また、main 関数内の CODEC モジュールセットアップ処理、CODEC モジュール開始処理を削除し、使用する CODEC モジュールに合わせて各種設定を追加してください。

5. ELCL を使った I²S 通信（スレーブ）

5.1 ソフトウェア説明

5.1.1 動作概要

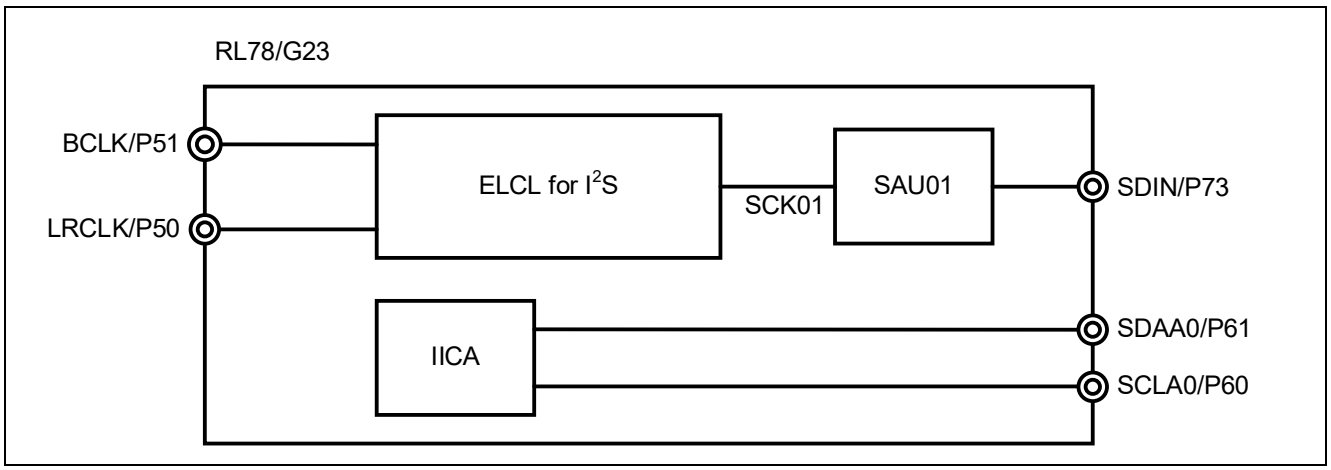
本サンプルコードでは、RS 社の Audio CODEC 754-1974 を使用します。また、オーディオ CODEC 内のレジスタ設定に I²C 通信を、サウンド・データ送信処理に I²S 通信を使用します。

サンプルコードは以下のような動作をします。

（1）I²C 通信用の IICA が動作を開始し、オーディオ CODEC 内のレジスタ設定を行います
（2）I²S 通信用の CSI01 が動作を開始します（SCK01 待ち状態）
（3）オーディオ CODEC から BCLK、LRCLK が入力され、データ送信が開始します
（4）データ送信後、LED1 を点灯します
（5）CSI01 が停止します

図 5-1 にサンプルコードのシステム構成を示します。

図 5-1 サンプルコードのシステム構成



5.1.2 フォルダ構成

表 5-1 にサンプルコードの使用しているソースファイル／ヘッダファイルの構成を示します。なお、統合開発環境で自動生成されるファイル、bsp 環境のファイルは除きます。

表 5-1 フォルダ構成

フォルダ、ファイル名	説明	スマート・コンフィグレータを使用
¥r01an6420_elcl_slave<DIR>注 3	サンプルコードのフォルダ	
¥src<DIR>	プログラム格納用フォルダ	
main.c	サンプルコードソースファイル	
main.h	サンプルコードヘッダファイル	
r_codec.c 注 4	CODEC モジュール設定用ソースファイル	
r_codec.h 注 4	CODEC モジュール設定用ヘッダファイル	
sound_data.c	サウンド・データ用ソースファイル	
sound_data.h	サウンド・データ用ヘッダファイル	
¥csi01<DIR>	CSI01 用プログラム格納フォルダ	
csi01.c	CSI01 用ソースファイル	
csi01.h	CSI01 用ヘッダファイル	
¥elcl<DIR>	ELCL 用プログラム格納フォルダ	
elcl.c	ELCL 用ソースファイル	
elcl.h	ELCL 用ヘッダファイル	
¥iica0<DIR>注 4	IICA0 用プログラム格納フォルダ	
iica0.c	IICA0 用ソースファイル	
iica0.h	IICA0 用ヘッダファイル	
¥smc_gen<DIR>	スマート・コンフィグレータ生成フォルダ	√
¥Config_PORT<DIR>	ポート用プログラム格納フォルダ	√
Config_PORT.c	ポート用ソースファイル	√
Config_PORT.h	ポート用ヘッダファイル	√
Config_PORT_user.c	ポート用割り込みソースファイル	√注 1
¥Config_TAU0_7<DIR>注 4	TAU07 用プログラム格納フォルダ	√
Config_TAU0_7.c	TAU07 用ソースファイル	√
Config_TAU0_7.h	TAU07 用ヘッダファイル	√
Config_TAU0_7_user.c	TAU07 用割り込みソースファイル	√注 2
¥general<DIR>	初期化、共通プログラム格納フォルダ	√
¥r_bsp<DIR>	BSP 用プログラム格納フォルダ	√
¥r_config<DIR>	プログラム格納フォルダ	√

補足 ”<DIR>” は、ディレクトリを意味します。
注 1. 本サンプルコードでは使用しません。
注 2. スマート・コンフィグレータで生成したファイルに割り込み処理ルーチンを追加しています。
注 3. IAR 版のサンプルコードは r01an6420_elcl_i2s_slave.ipcf を格納しています。ipcf ファイルについては、「RL78 スマート・コンフィグレータ ユーザーガイド：IAR 編（R20AN0581）」を確認してください。
注 4. RS 社の Audio CODEC 754-1974 の CODEC モジュール設定に必要なファイルです。

I²S 通信は CSI を使用するため、本来の CSI の規格に沿わず、オーバーランエラーが発生します。スマート・コンフィグレータで生成したコードからエラー検出処理を削除しています。

また、IICA で多重割り込みを有効にするため、スマート・コンフィグレータで生成されたコードを修正し、不要な関数を削除しています。

5.1.3 オプション・バイトの設定一覧

表 5-2 にオプション・バイト設定を示します。

表 5-2 オプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	1110 1111B (EFH)	ウォッチドッグ・タイマ動作停止 （リセット解除後、カウント停止）
000C1H/040C1H	1111 1110B (FEH)	LVD0 リセット・モード 検出電圧：立ち上がり 1.90V／立下り 1.86V
000C2H/040C2H	1110 1000B (E8H)	フラッシュ動作モード：高速メインモード 高速オンチップ・オシレータの周波数：32MHz
000C3H/040C3H	1000 0101B (85H)	オンチップ・デバッグ動作許可

5.1.4 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	ファイル
LED1	P5_bit.no3	P53	csi01.c
LED_ON	0	LED を ON するための設定値	csi01.c
LED_OFF	1	LED を OFF するための設定値	csi01.c
DATA_48K_32B	1	再生するサウンド・データを選択する。（1：有効、0：無効）	sound_data.h
DECODE_PCM_SIZE	32000	サウンド・データのデータ数。 再生するサウンド・データによって設定値が変化。	sound_data.h
g_tx_buf[]	サンプルコード参照	サウンド・データ	sound_data.c
SENSOR_ADD	0x34	センサのアドレス	r_codec.h
WAIT_TIME	100	IICA0 通信待ち時間	r_codec.h
g_cmd_l_vol[2]	{ 0x00, 0x17 }	左ライン・入力チャンネル・ボリューム・制御コマンド	r_codec.c
g_cmd_r_vol[2]	{ 0x02, 0x17 }	右ライン・入力チャンネル・ボリューム・制御コマンド	r_codec.c
g_cmd_bypass[2]	{ 0x08, 0x12 }	アナログ・オーディオパス制御コマンド	r_codec.c
g_cmd_deempha[2]	{ 0x0A, 0x00 }	デジタル・オーディオパス制御コマンド	r_codec.c
g_cmd_line_adc[2]	{ 0x0C, 0x42 }	パワーダウン制御コマンド	r_codec.c
g_cmd_set_rate[2]	{0x10, 0x00}	サンプリング周波数制御コマンド	r_codec.c
g_cmd_set_format[2]	{0x0E, 0x0E}	デジタル・オーディオインターフェース・フォーマット設定コマンド	r_codec.c
g_cmd_activate[2]	{ 0x12, 0x01 }	CODEC モジュール・アクティブコマンド	r_codec.c
g_cmd_inactivate[2]	{ 0x12, 0x00 }	CODEC モジュール・非アクティブコマンド	r_codec.c
g_cmd_reset[2]	{ 0x1E, 0x00 }	レジスタ・リセットコマンド	r_codec.c

5.1.5 変数一覧

表 5-4 に本サンプルコードで使用するグローバル変数を示します。

表 5-4 変数一覧

型	変数名	内容	使用関数
volatile uint16_t	g_ms_timer	ウエイト処理のカウント値	r_ms_delay, r_Config_TAU0_7_interrupt
volatile uint8_t	g_tx_done_flag	送信完了フラグ	main r_csi01_callback_sendend
volatile uint8_t	g_sample_mode	I²C 通信ステータス	r_codec_init r_codec_start r_codec_stop r_iica0_callback_master_sende nd

5.1.6 関数一覧

表 5-5 にサンプルコードで使用する関数を示します。ただし、スマート・コンフィグレータで生成された関数の内、変更を行っていないものは除きます。

表 5-5 関数一覧

関数名	概要	ソースファイル
main	メイン処理	main.c
r_csi01_create	CSI01 初期設定処理	csi.c
r_csi01_start	CSI01 動作開始処理	csi.c
r_csi01_stop	CSI01 停止処理	csi.c
r_csi01_send	CSI01 送信開始処理	csi.c
r_csi01_callback_sendend	CSI01 送信終了のコールバック処理	csi.c
r_csi01_interrupt	CSI01 割り込み処理	csi.c
r_elcl_create	ELCL 初期設定処理	elcl.c
r_elcl_start	ELCL 出力開始処理	elcl.c
r_elcl_stop	ELCL 停止処理	elcl.c
r_elcl_reset_flipflop	ELCL のフリップフロップリセット処理	elcl.c
r_iica0_create	IICA0 初期設定処理	Iica0.c
r_iica0_stop	IICA0 停止処理	Iica0.c
r_iica0_stopcondition	IICA0 ストップ・コンディション処理	Iica0.c
r_iica0_master_send	IICA0 マスタ送信開始処理	Iica0.c
r_iica0_callback_master_sendend	IICA0 送信完了コールバック処理	Iica0.c
r_iica0_master_handler	IICA0 割り込みハンドラ	Iica0.c
r_iica0_interrupt	IICA0 割り込み処理	Iica0.c
r_codec_init	CODEC モジュールのセットアップ処理	r_codec.c
r_codec_start	CODEC モジュールの開始処理	r_codec.c
r_codec_stop	CODEC モジュールの停止処理	r_codec.c
r_ms_delay	IICA0 ウエイト処理	Config_TAU0_7_user.c
r_Config_TAU0_7_interrupt	TAU チャネル 7 割り込み処理	Config_TAU0_7_user.c

5.1.7 関数仕様

サンプルコードの関数仕様を示します。

[関数名] main

概要	メイン処理
ヘッダ	r_smc_entry.h, main.h, elcl.h, csi01.h, r_codec.h, sound_data.h
宣言	void main (void);
説明	ELCL、CSI01 の初期化、CODEC モジュールの設定、I²S 通信を行います
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_create

概要	CSI01 初期設定処理
ヘッダ	csi01.h, elcl.h
宣言	void r_csi01_create (void);
説明	CSI01 の初期設定を行います
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_start

概要	CSI01 動作開始処理
ヘッダ	csi01.h, elcl.h
宣言	void r_csi01_start (void);
説明	CSI01 の動作を許可します
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_stop

概要	CSI01 停止処理
ヘッダ	csi01.h, elcl.h
宣言	void r_csi01_stop (void);
説明	CSI01 の動作を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_send

概要	CSI01 送信開始処理
ヘッダ	csi01.h, elcl.h
宣言	MD_STATUS r_csi01_send (uint8_t *const tx_buf, uint16_t tx_num);
説明	引数 tx_buf で指定したアドレスから引数 tx_num 個のデータを送信します エラー検出は行いません
引数	tx_buf, tx_num
リターン値	Status
備考	なし

[関数名] r_csi01_callback_sendend

概要	CSI01 送信終了のコールバック処理
ヘッダ	csi01.h, elcl.h
宣言	static void r_csi01_callback_sendend (void);
説明	ELCL の出力を停止し、g_tx_done_flag を立て、LED1 を点灯します
引数	なし
リターン値	なし
備考	なし

[関数名] r_csi01_interrupt

概要	CSI01 割り込み処理
ヘッダ	csi01.h, elcl.h
宣言	#pragma interrupt r_csi01_interrupt (vect=INTCSI01)
説明	次の送信データを SIO01 にセットします
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_create

概要	ELCL の初期設定処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_create (void);
説明	ELCL の初期設定を行います
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_start

概要	ELCL 動作開始処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_start (void);
説明	ELCL の出力を開始します
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_stop

概要	ELCL 動作停止処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_stop (void);
説明	ELCL の出力を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_elcl_reset_flipflop

概要	ELCL のフリップフロップリセット処理
ヘッダ	elcl.h, Config_Throgh.h, platform.h
宣言	void r_elcl_reset_flipflop (void);
説明	ELOMONI フラグをリセットするためにフリップフロップをリセットします
引数	なし
リターン値	なし
備考	ELLnCTL のビット 6 とビット 7 に 0 を設定するとフリップフロップはリセットされます

[関数名] r_iica0_create

概要	IICA0 初期設定処理
ヘッダ	iica.h, r_codec.h
宣言	void r_csi01_create (void);
説明	IICA0 の初期設定をします
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_stop

概要	IICA0 停止処理
ヘッダ	iica.h, r_codec.h
宣言	void r_iica0_stop (void);
説明	IICA0 の動作を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_stopcondition

概要	IICA0 ストップ・コンディション
ヘッダ	iica.h, r_codec.h
宣言	void r_iica0_stopcondition (void);
説明	ストップ・コンディションを生成します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_master_send

概要	IICA0 マスタ送信開始処理
ヘッダ	iica.h, r_codec.h
宣言	MD_STATUS r_iica0_master_send (uint8_t adr, uint8_t *const tx_buf, uint16_t tx_num, uint8_t wait);
説明	(1) I²C バスが解放されている場合、スタート・コンデションを生成します (2) スタート・コンデション生成後、スレーブ・アドレスを送信モードにし、データ送信を開始します (3) 通信完了を待ちます
引数	adr, tx_buf, tx_num, wait
リターン値	status
備考	なし

[関数名] r_iica0_callback_master_sendend

概要	IICA0 送信完了コールバック処理
ヘッダ	iica.h, r_codec.h
宣言	static void r_iica0_callback_master_sendend (void);
説明	ストップ・コンディションを生成し、IIC 通信ステータスを FINISH にします
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_master_handler

概要	IICA0 割り込みハンドラ
ヘッダ	iica.h, r_codec.h
宣言	void r_iica0_master_handler (void);
説明	スレーブ・アドレスに対して ACK を検出した場合、データを送信します
引数	なし
リターン値	なし
備考	なし

[関数名] r_iica0_interrupt

概要	IICA0 割り込み処理
ヘッダ	iica.h, r_codec.h
宣言	#pragma interrupt r_iica0_interrupt (vect=INTIICA0,enable=true)
説明	IICA0 の割り込み要求の受付処理を行います 多重割り込みを許可しています
引数	なし
リターン値	なし
備考	なし

[関数名] r_codec_init

概要	CODEC モジュールのセットアップ処理
ヘッダ	sound_data.h, iica0.h, r_codec.h, Config_TAU0_7.h
宣言	void r_codec_init (void);
説明	(1) IICA0 の初期設定処理を行います (2) CODEC モジュールのセットアップを行います (3) 設定完了を待ちます(100ms)
引数	なし
リターン値	なし
備考	なし

[関数名] r_codec_start

概要	CODEC モジュールの開始処理
ヘッダ	sound_data.h, iica0.h, r_codec.h, Config_TAU0_7.h
宣言	void r_codec_start (void);
説明	CODEC モジュールの動作を開始します
引数	なし
リターン値	なし
備考	なし

[関数名] r_codec_stop

概要	CODEC モジュールの停止処理
ヘッダ	sound_data.h, iica0.h, r_codec.h, Config_TAU0_7.h
宣言	void r_codec_stop (void);
説明	CODEC モジュールの動作を停止します
引数	なし
リターン値	なし
備考	なし

[関数名] r_ms_delay

概要	ウエイト処理
ヘッダ	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_7.h
宣言	void r_ms_delay (uint16_t msec);
説明	引数 msec で指定した時間（ms）ウエイトします TAU0 チャネル 7 を使ってカウントします。g_ms_timer が引数 msec 未満の場合はポーリングし、msec 以上の場合はウエイト処理を完了します
引数	msec
リターン値	なし
備考	なし

[関数名] r_Config_TAU0_7_interrupt

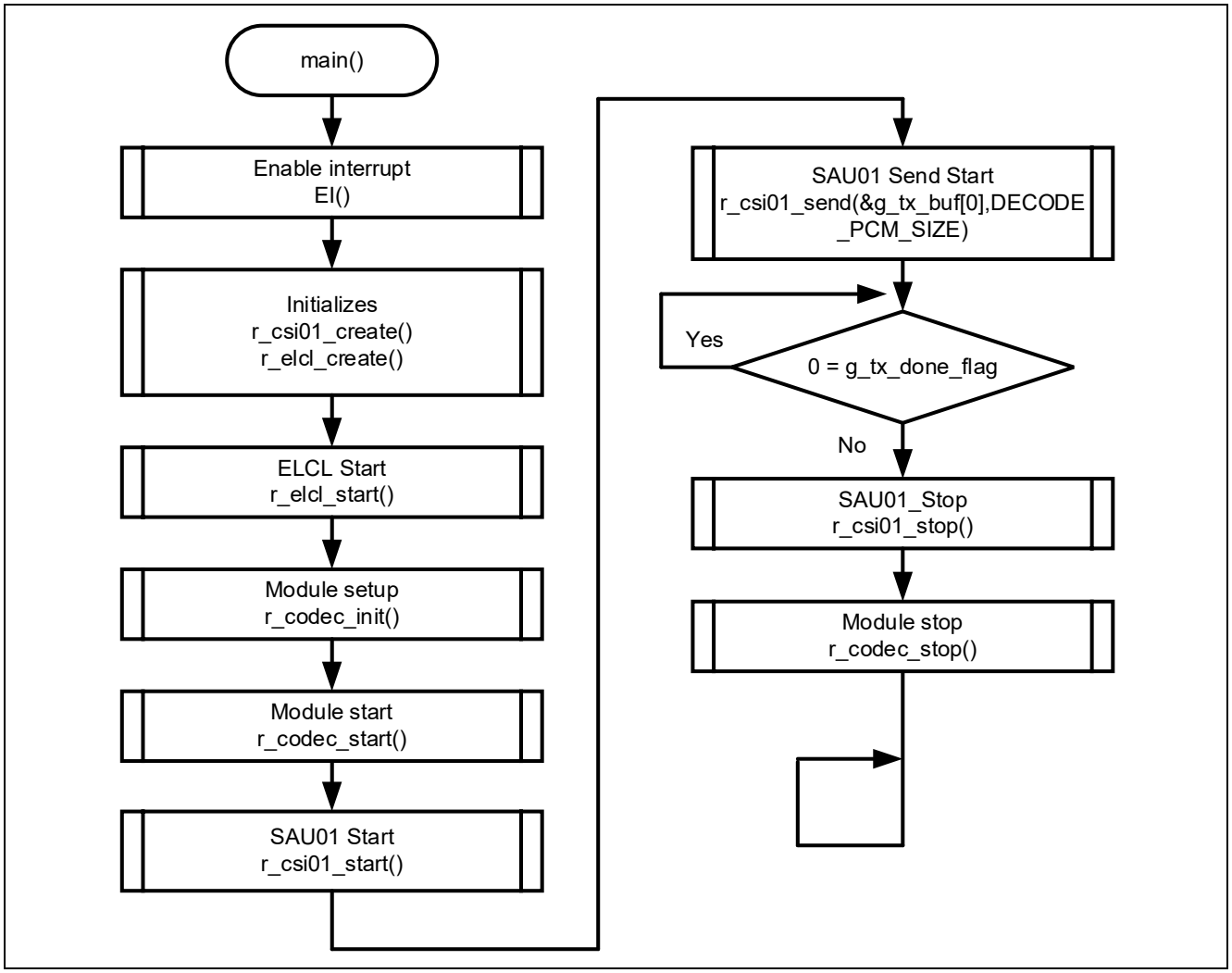
概要	TAU07 割り込み処理
ヘッダ	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_7.h
宣言	#pragma interrupt r_Config_TAU0_7_interrupt (vect=INTTM07)
説明	TAU0 チャネル 7 の INTTM07 による割り込み処理です g_ms_timer をカウントアップします
引数	なし
リターン値	なし
備考	なし

5.1.8 フローチャート

5.1.8.1 メイン処理

図 5-2 にメイン処理のフローチャートを示します。

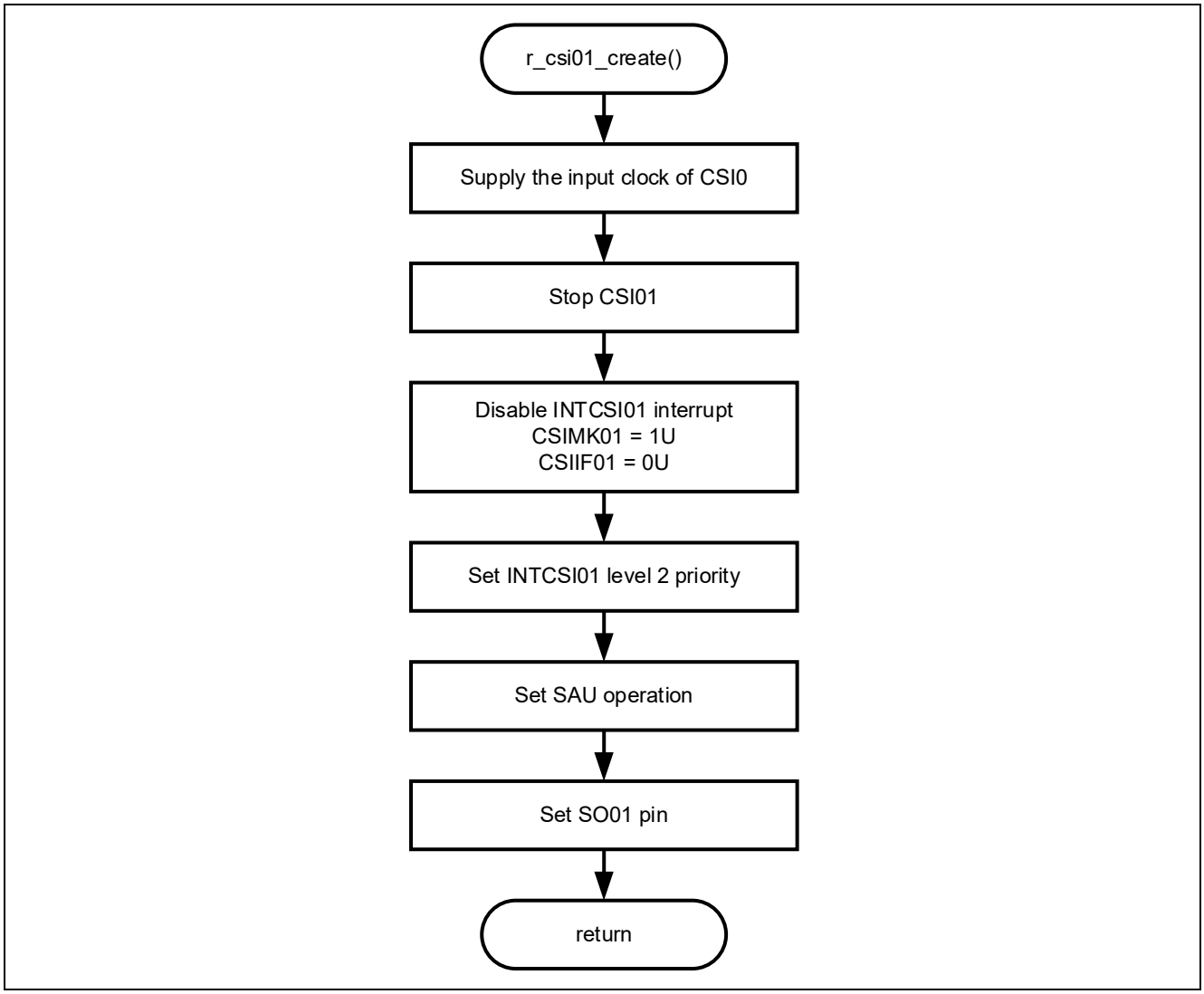
図 5-2 メイン処理



5.1.8.2 CSI01 初期設定処理

図 5-3 に CSI01 初期設定処理のフローチャートを示します。

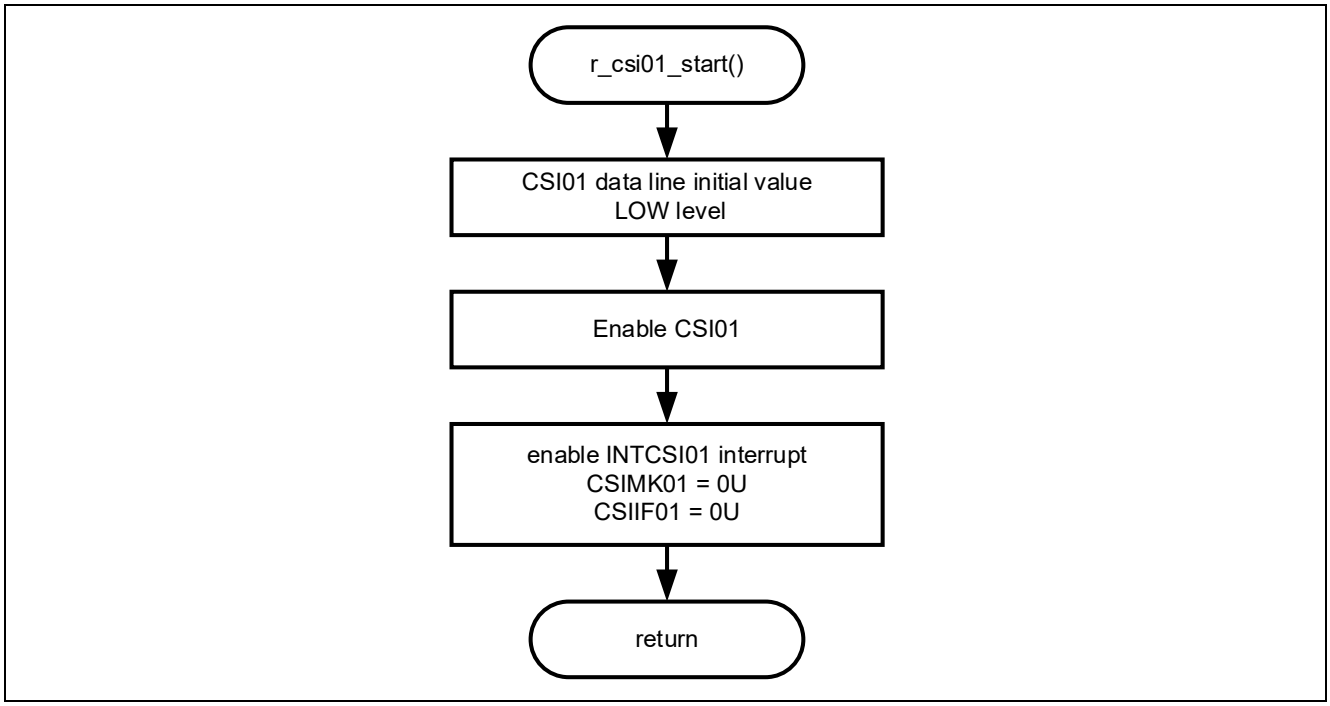
図 5-3 CSI01 初期設定処理



5.1.8.3 CSI01 動作開始処理

図 5-4 に CSI01 動作開始処理のフローチャートを示します。

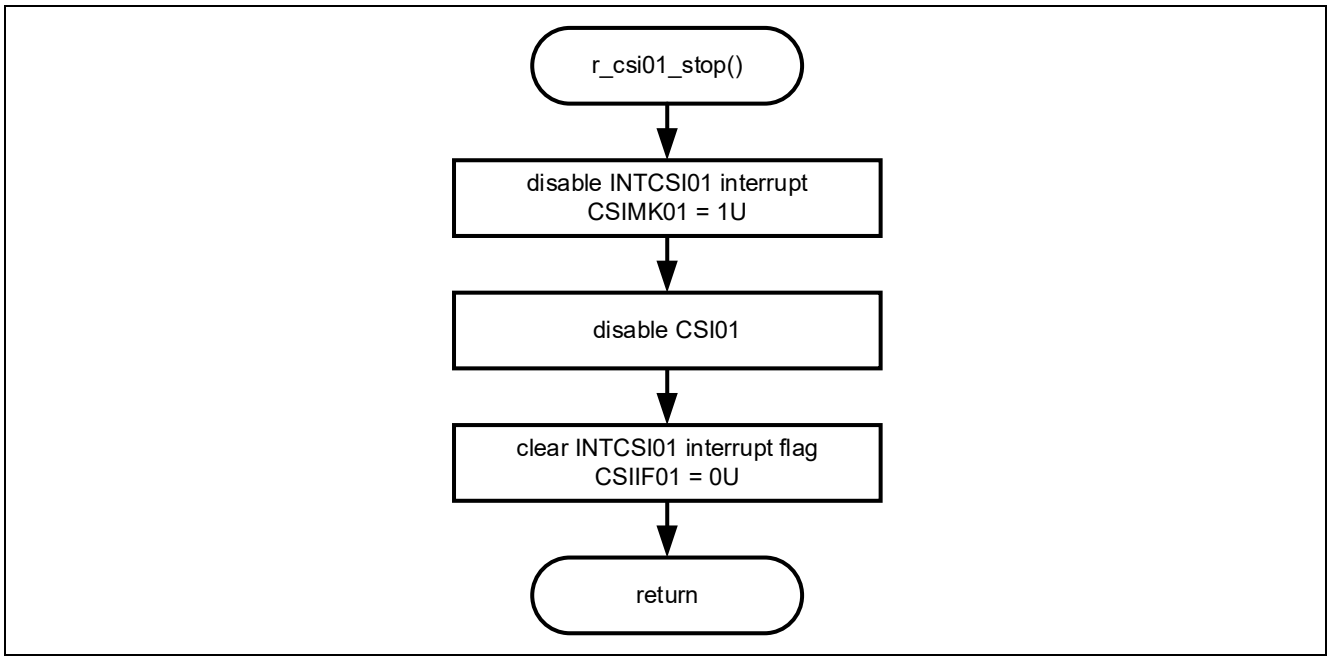
図 5-4 CSI01 動作開始処理



5.1.8.4 CSI01 停止処理

図 5-5 に CSI01 停止処理のフローチャートを示します。

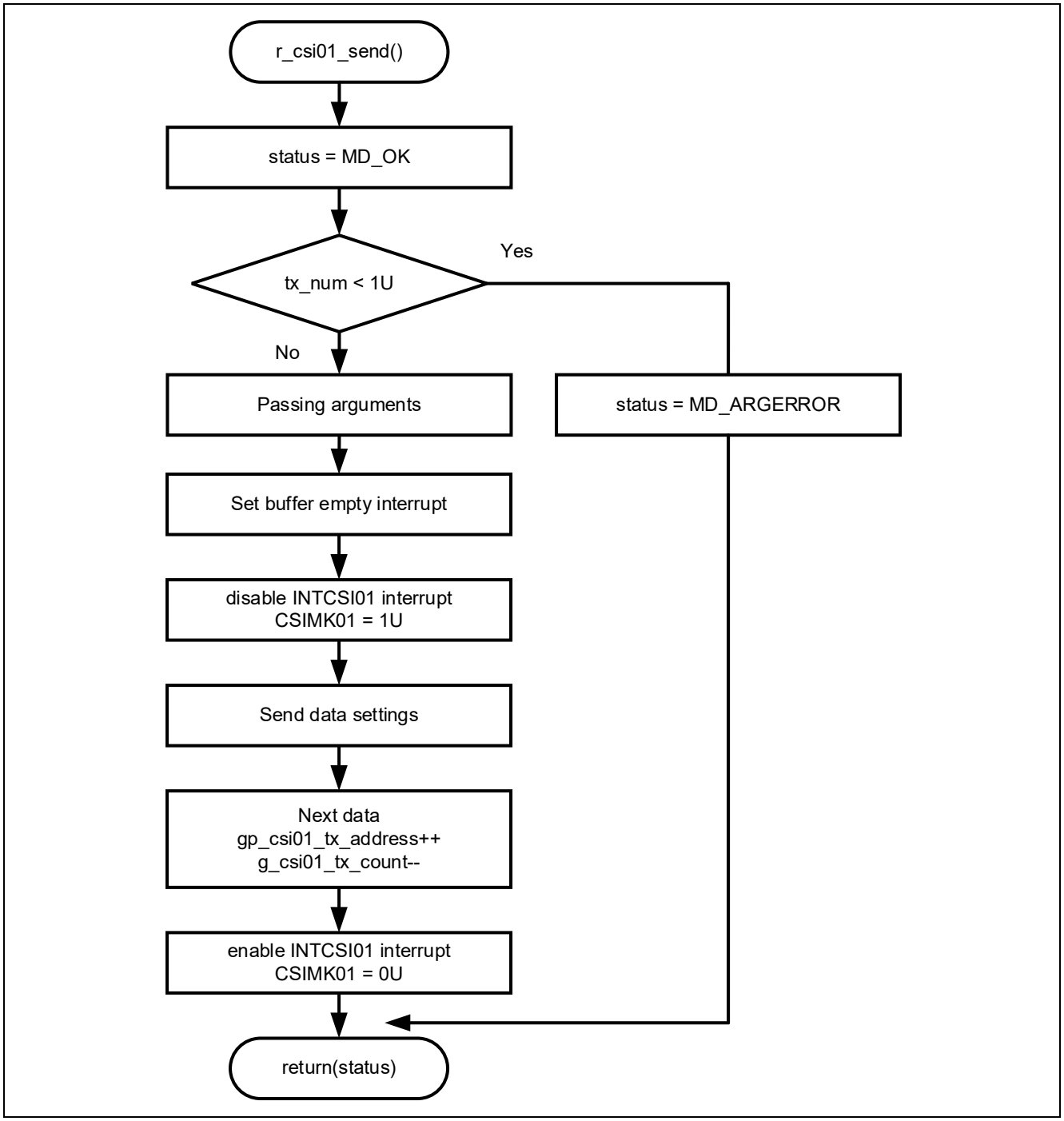
図 5-5 CSI01 停止処理



5.1.8.5 CSI01 送信開始処理

図 5-6 に CSI01 送信開始処理のフローチャートを示します。

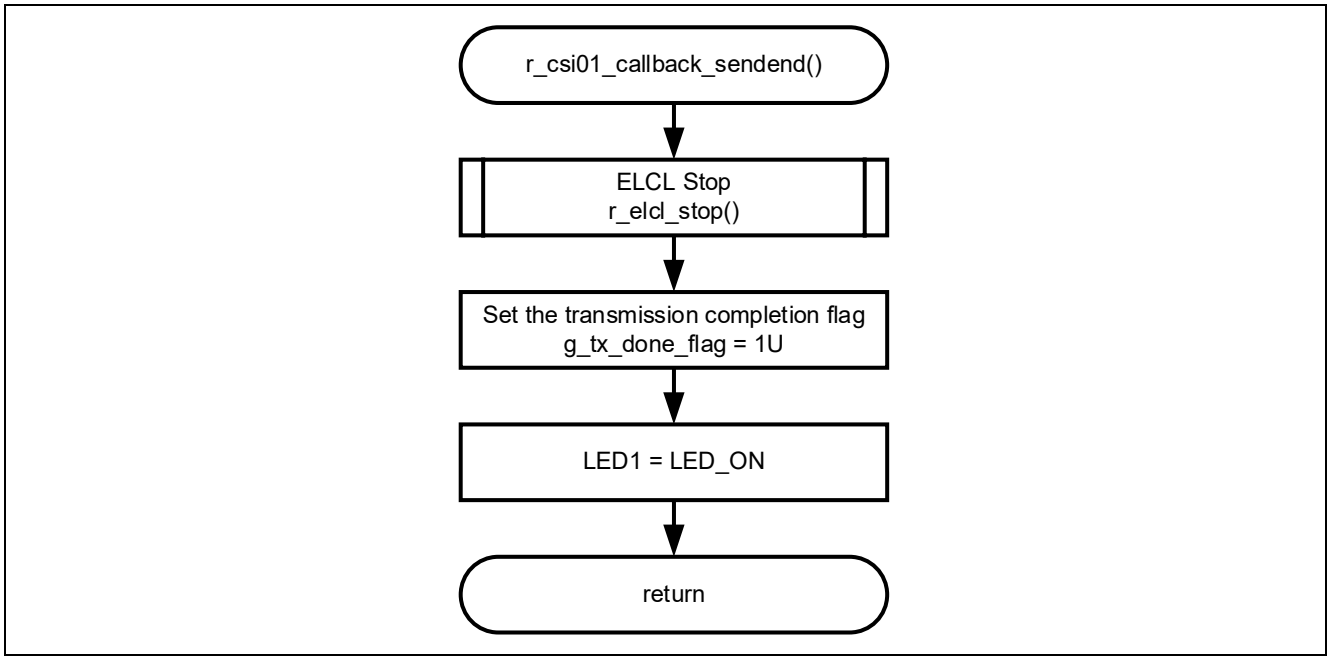
図 5-6 CSI01 送信開始処理



5.1.8.6 CSI01 送信終了のコールバック処理

図 5-7 に CSI01 送信終了のコールバック処理のフローチャートを示します。

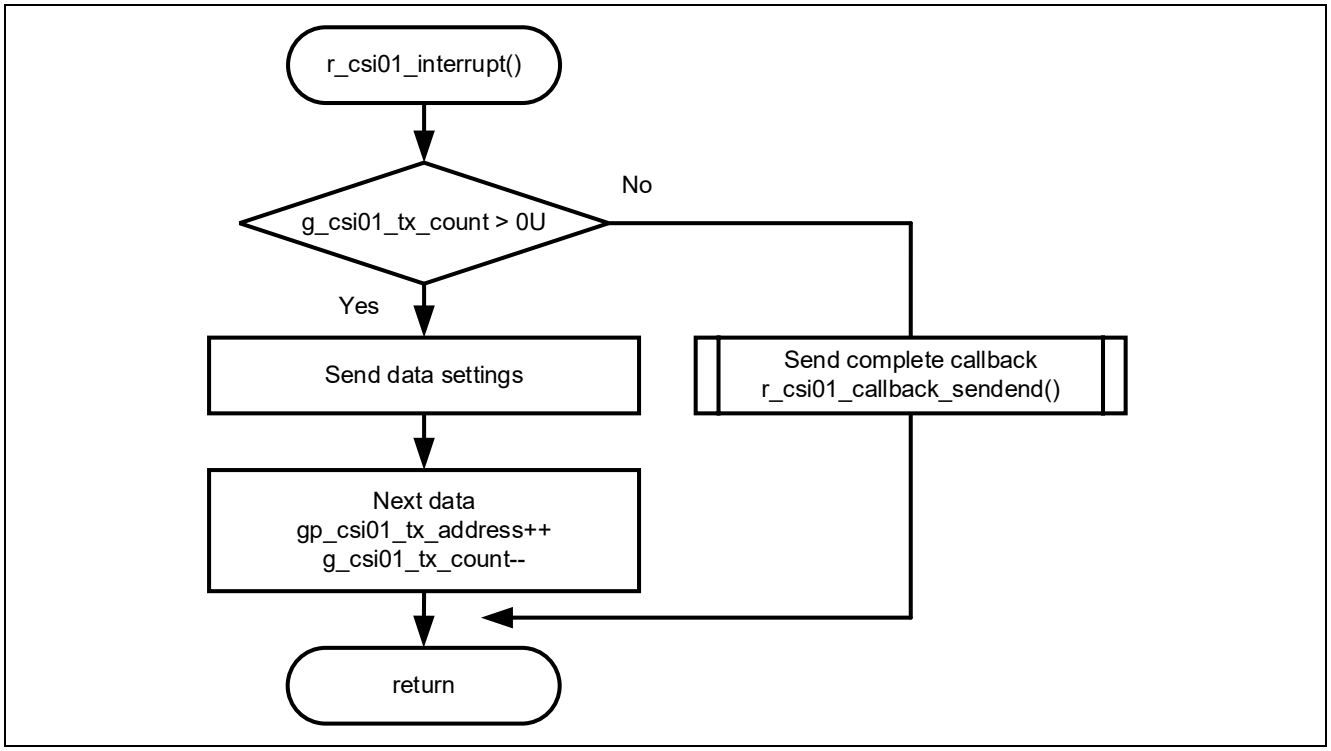
図 5-7 CSI01 送信完了のコールバック処理



5.1.8.7 CSI01 割り込み処理

図 5-8 に CSI01 割り込み処理のフローチャートを示します。

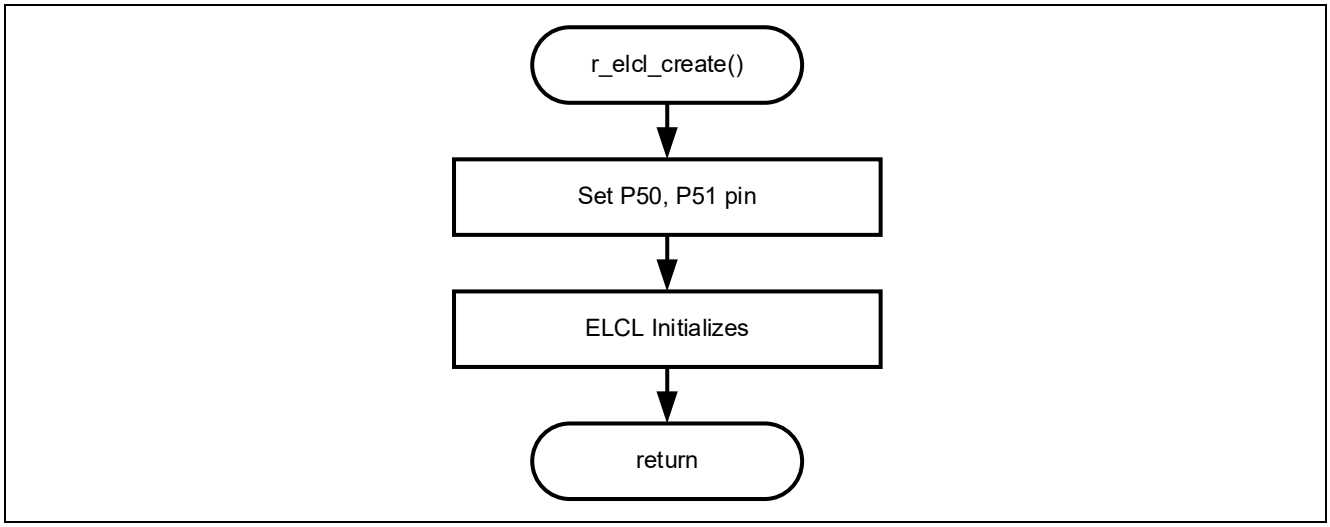
図 5-8 CSI01 割り込み処理



5.1.8.8 ELCL 初期設定処理

図 5-9 に ELCL 初期設定処理のフローチャートを示します。

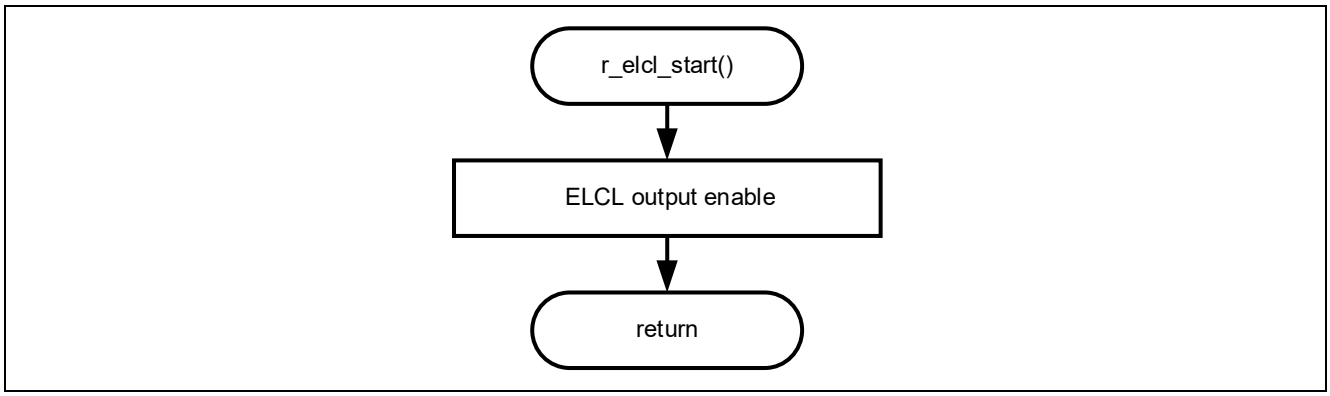
図 5-9 ELCL 初期設定処理



5.1.8.9 ELCL 出力開始処理

図 5-10 に ELCL 出力開始処理のフローチャートを示します。

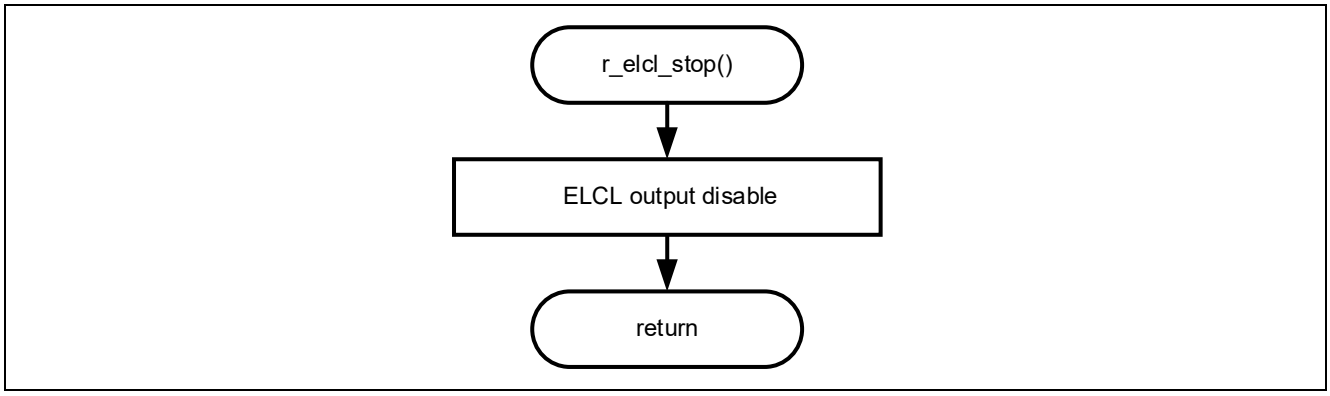
図 5-10 ELCL 出力開始処理



5.1.8.10 ELCL 停止処理

図 5-11 に ELCL 停止処理のフローチャートを示します。

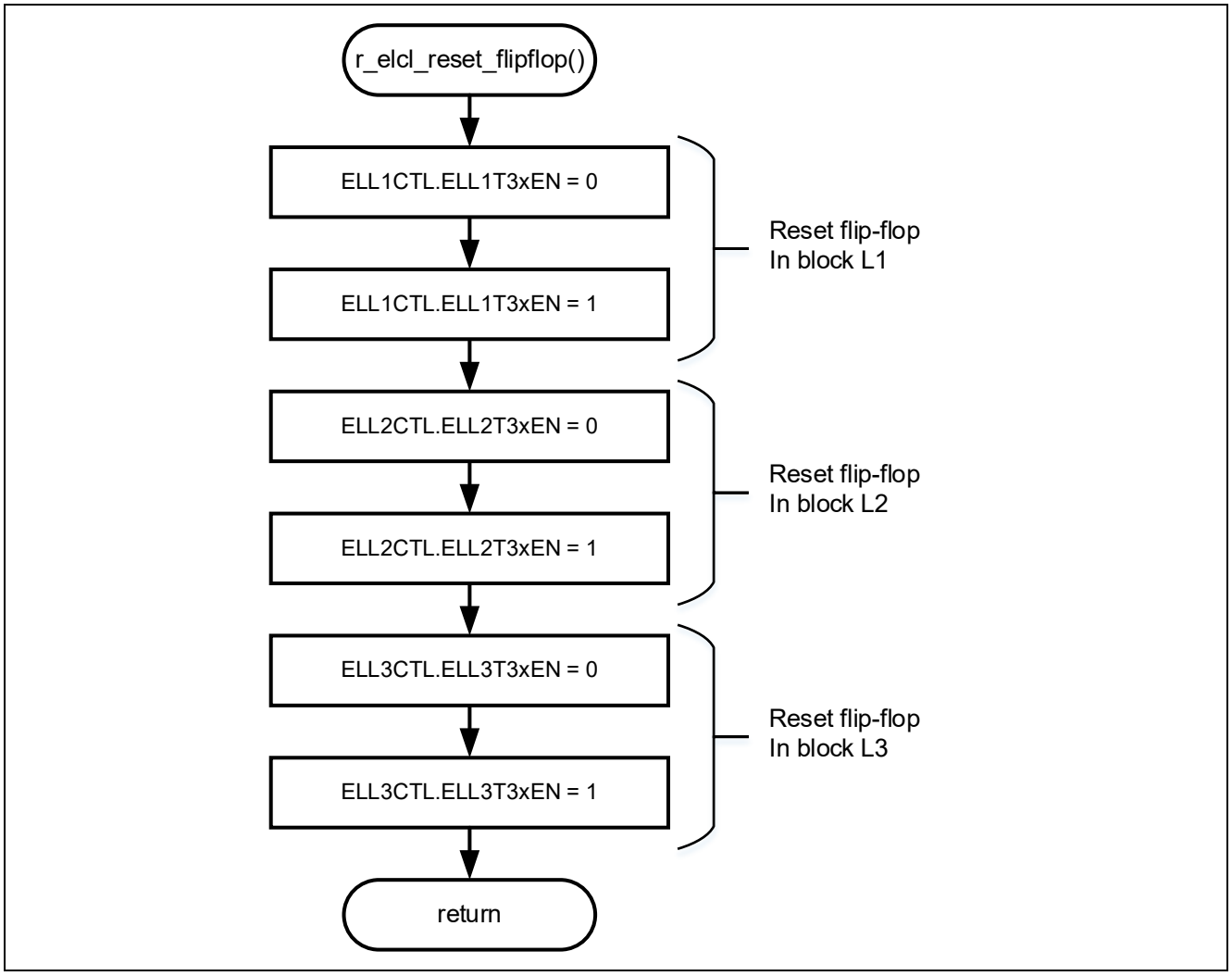
図 5-11　ELCL 停止処理



5.1.8.11 ELCL のフリップフロップリセット処理

図 5-12 に ELCL のフリップフロップリセット処理のフローチャートを示します。

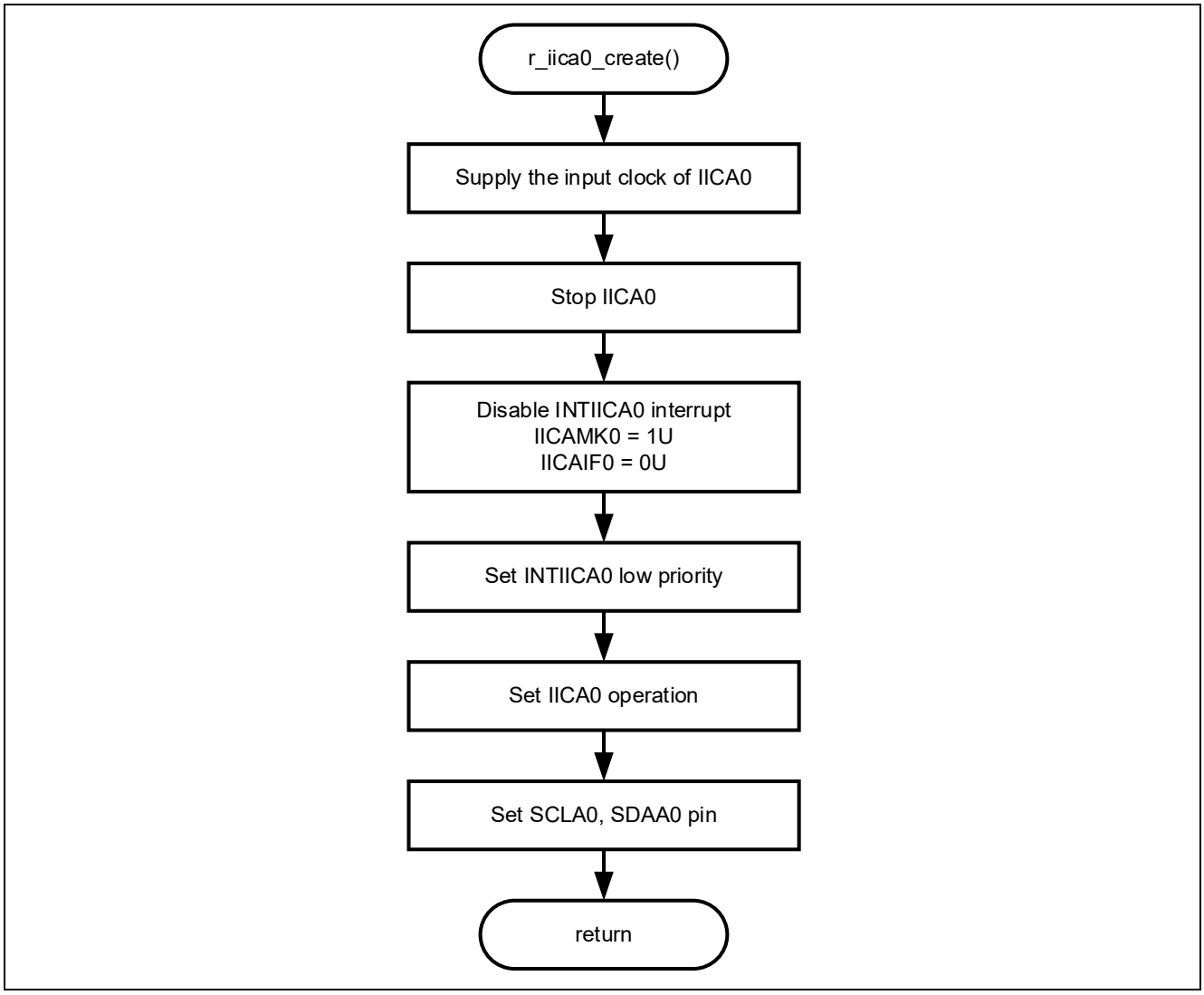
図 5-12　ELCL のフリップフロップリセット処理



5.1.8.12 IICA0 初期設定処理

図 5-13 に IICA0 初期設定処理のフローチャートを示します。

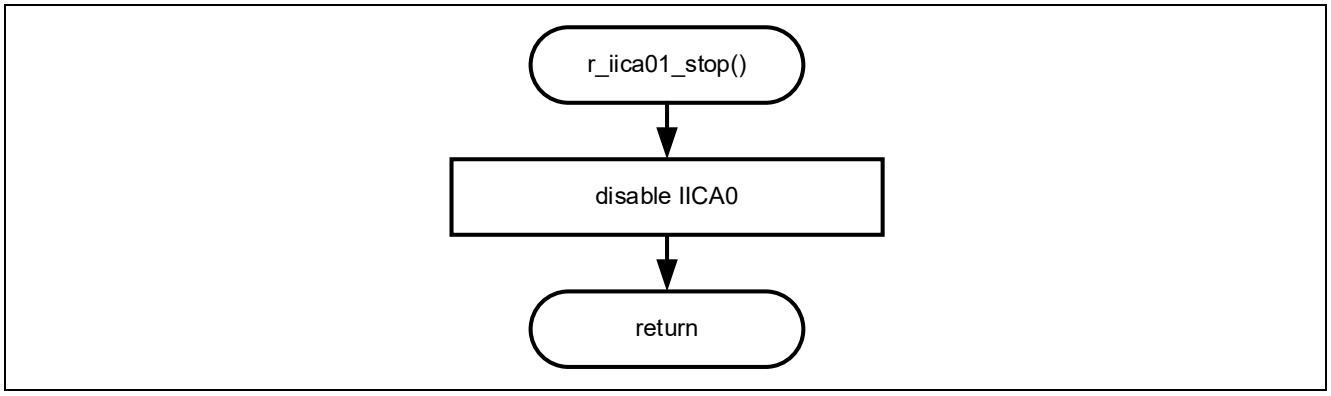
図 5-13　IICA0 初期設定処理



5.1.8.13 IICA0 停止処理

図 5-14 に IICA0 停止処理のフローチャートを示します。

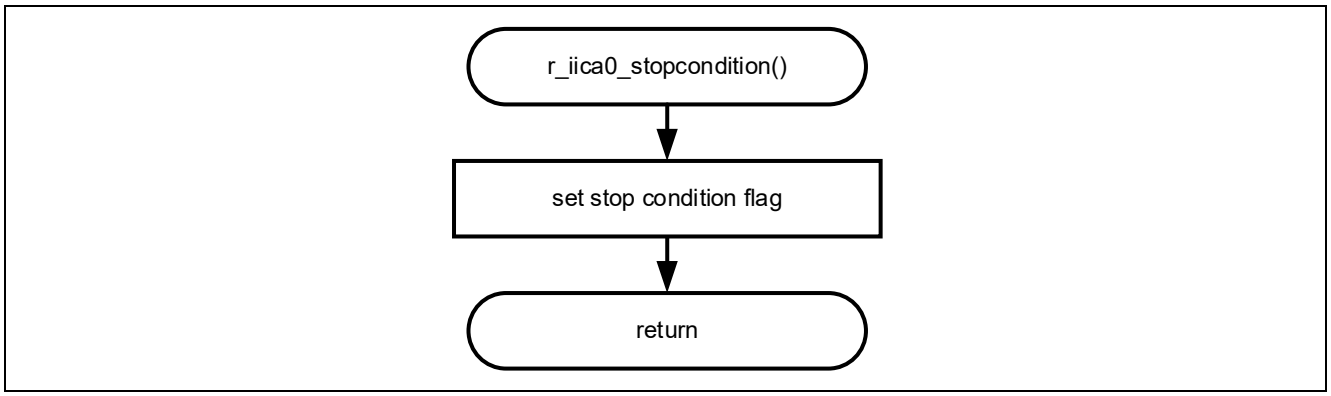
図 5-14　IICA0 停止処理



5.1.8.14 IICA0 ストップ・コンディション処理

図 5-15 に IICA0 ストップ・コンディション処理のフローチャートを示します。

図 5-15　IICA0 ストップ・コンディション処理



5.1.8.15 IICA0 マスタ送信開始処理

図 5-16、図 5-17 に IICA0 マスタ送信開始処理のフローチャートを示します。

図 5-16 IICA0 マスタ送信開始処理 (1/2)

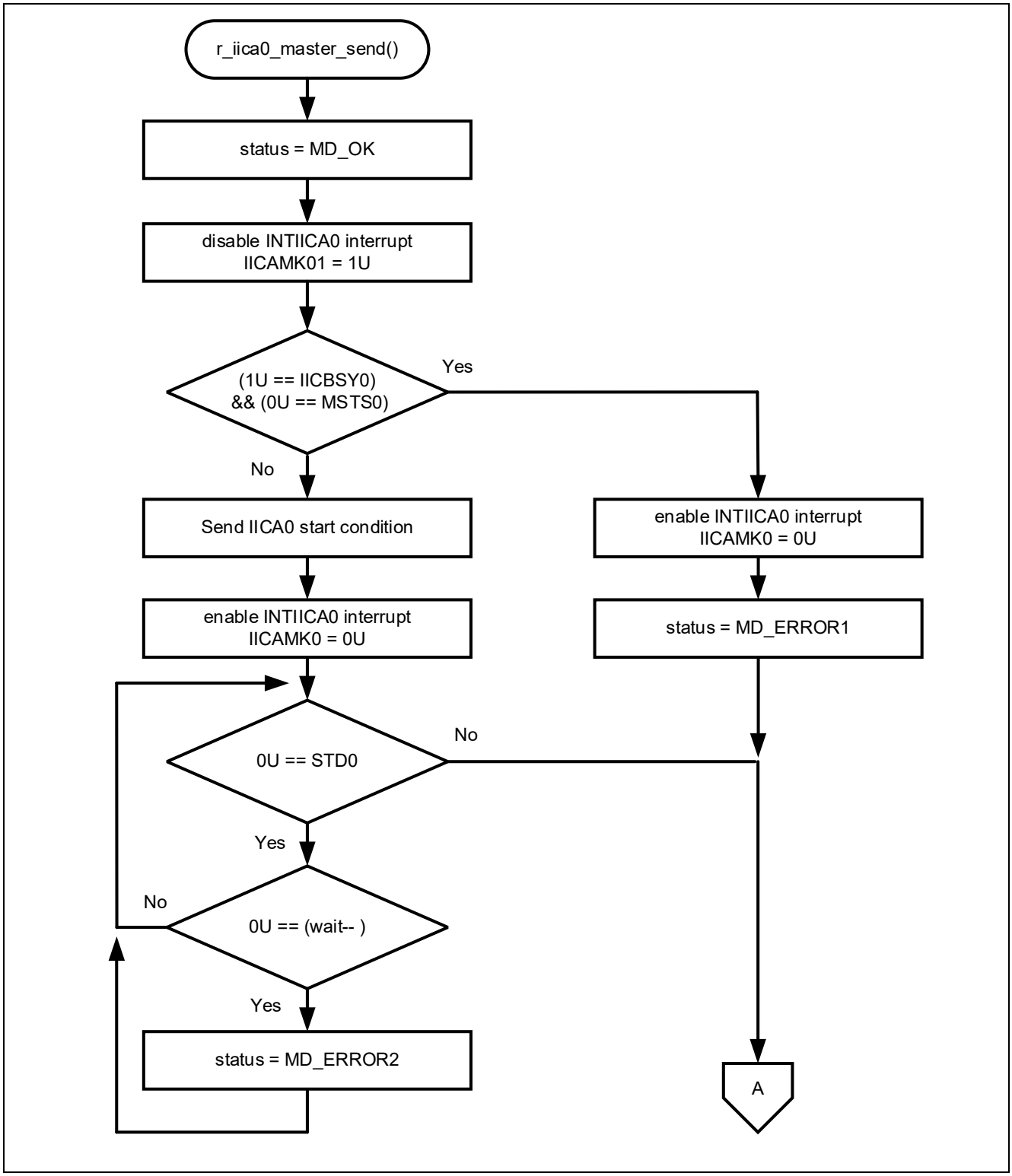
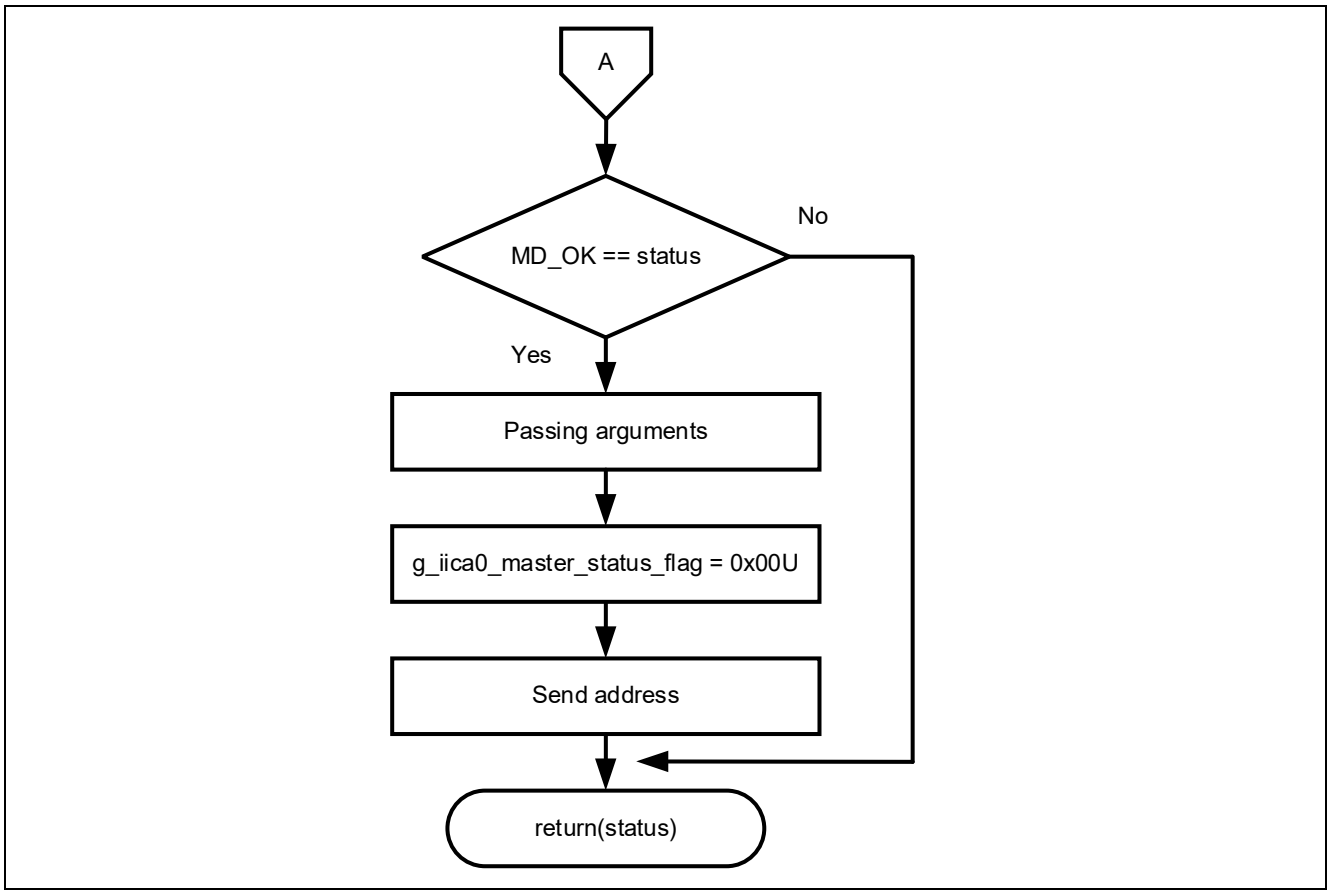


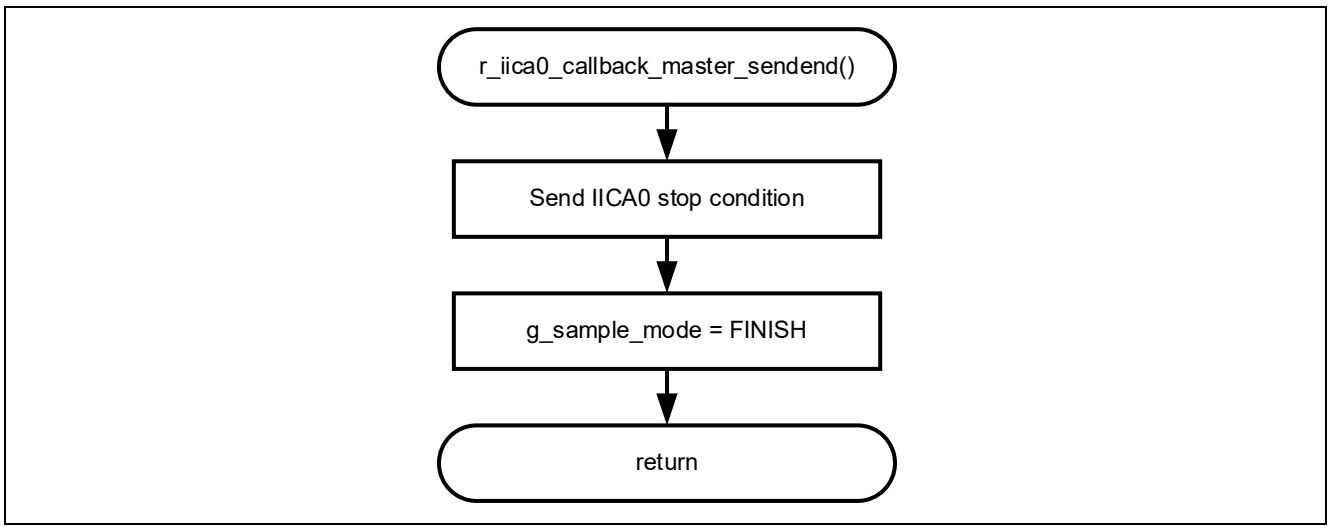
図 5-17 IICA0 マスタ送信開始処理 (2/2)



5.1.8.16 IICA0 送信完了コールバック処理

図 5-18 に IICA 送信完了コールバック処理のフローチャートを示します。

図 5-18 IICA0 送信完了コールバック処理



5.1.8.17 IICA0 割り込みハンドラ

図 5-19、図 5-20 に IICA 割り込みハンドラのフローチャートを示します。

図 5-19 IICA0 割り込みハンドラ (1/2)

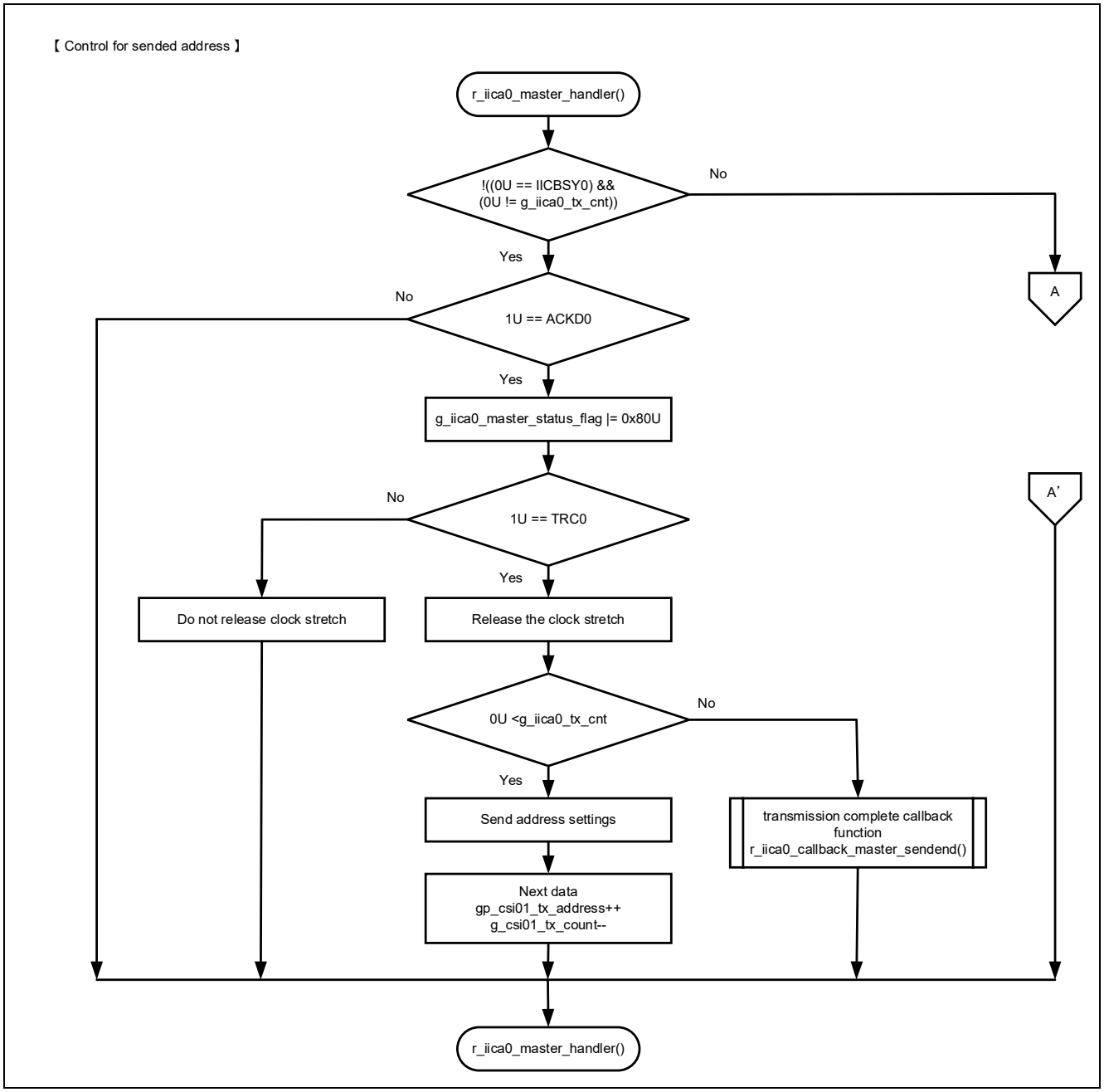
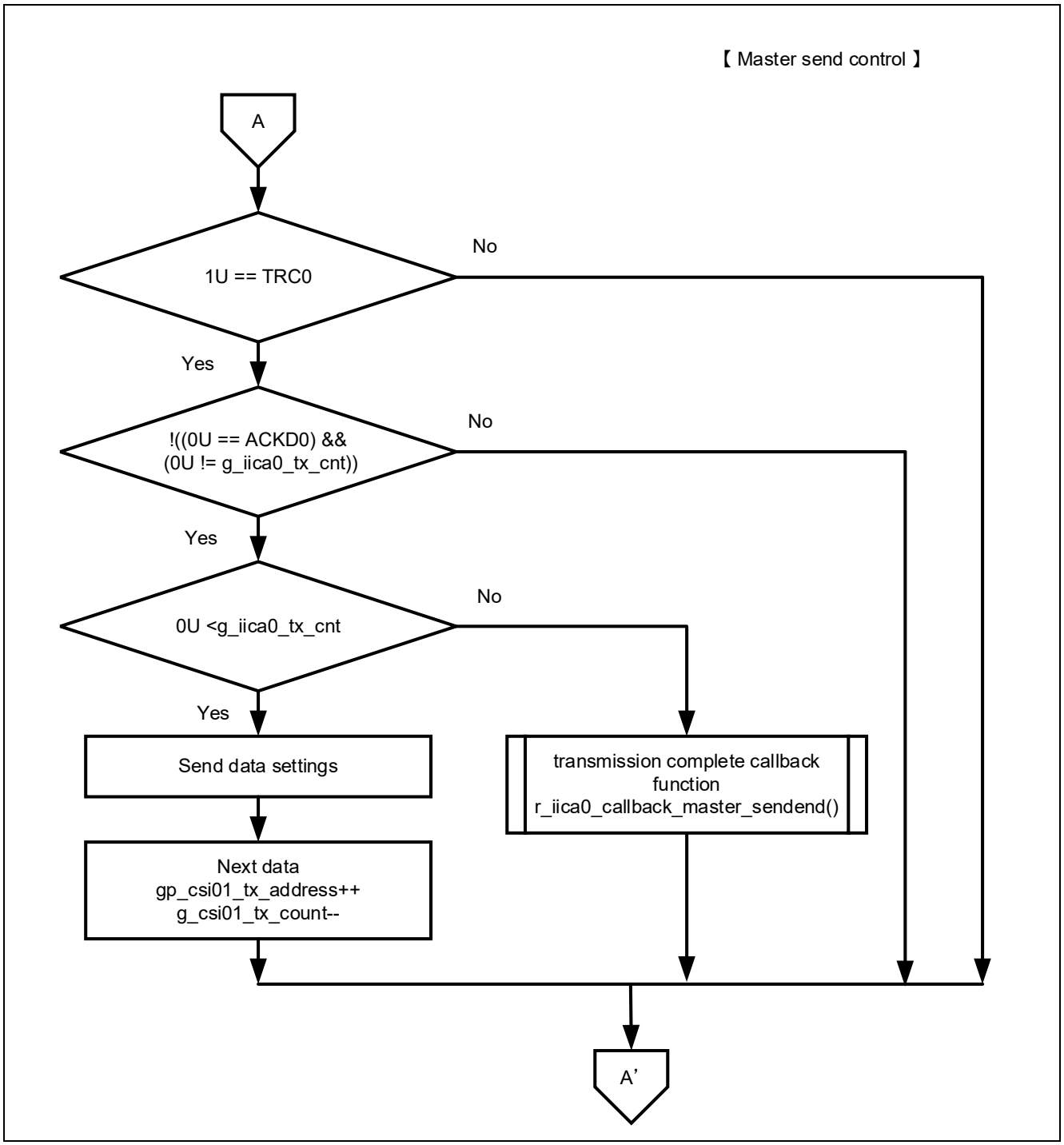


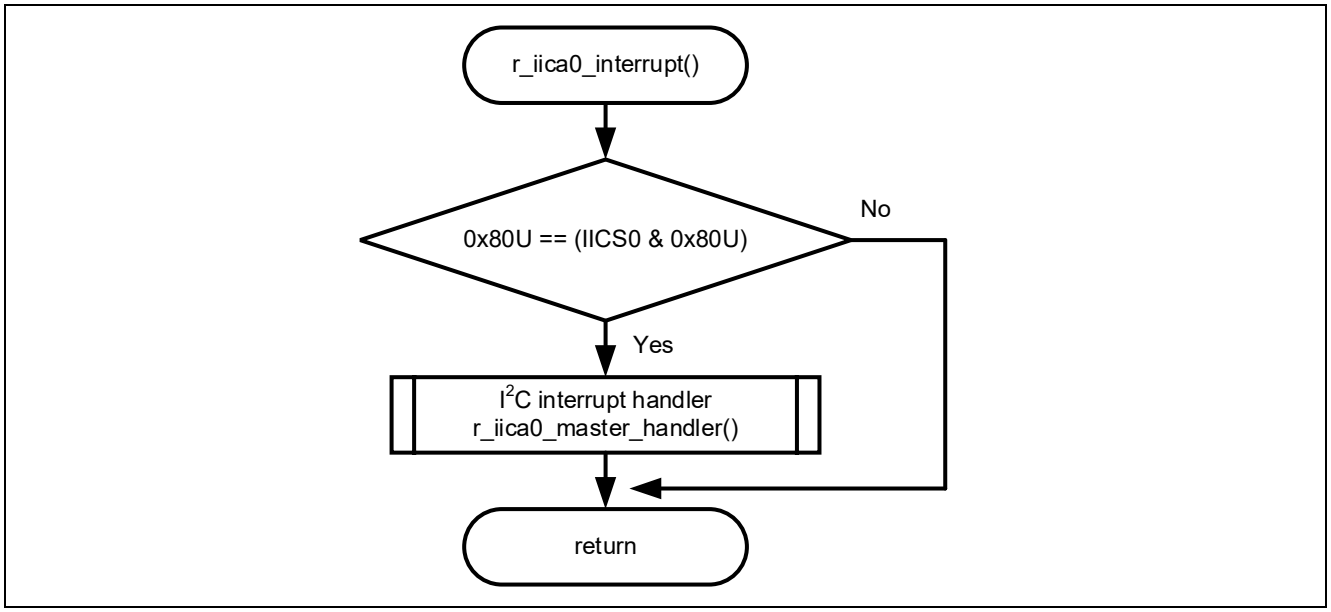
図 5-20 IICA0 割り込みハンドラ (2/2)



5.1.8.18 IICA0 割り込み処理

図 5-21 に IICA0 割り込み処理のフローチャートを示します。

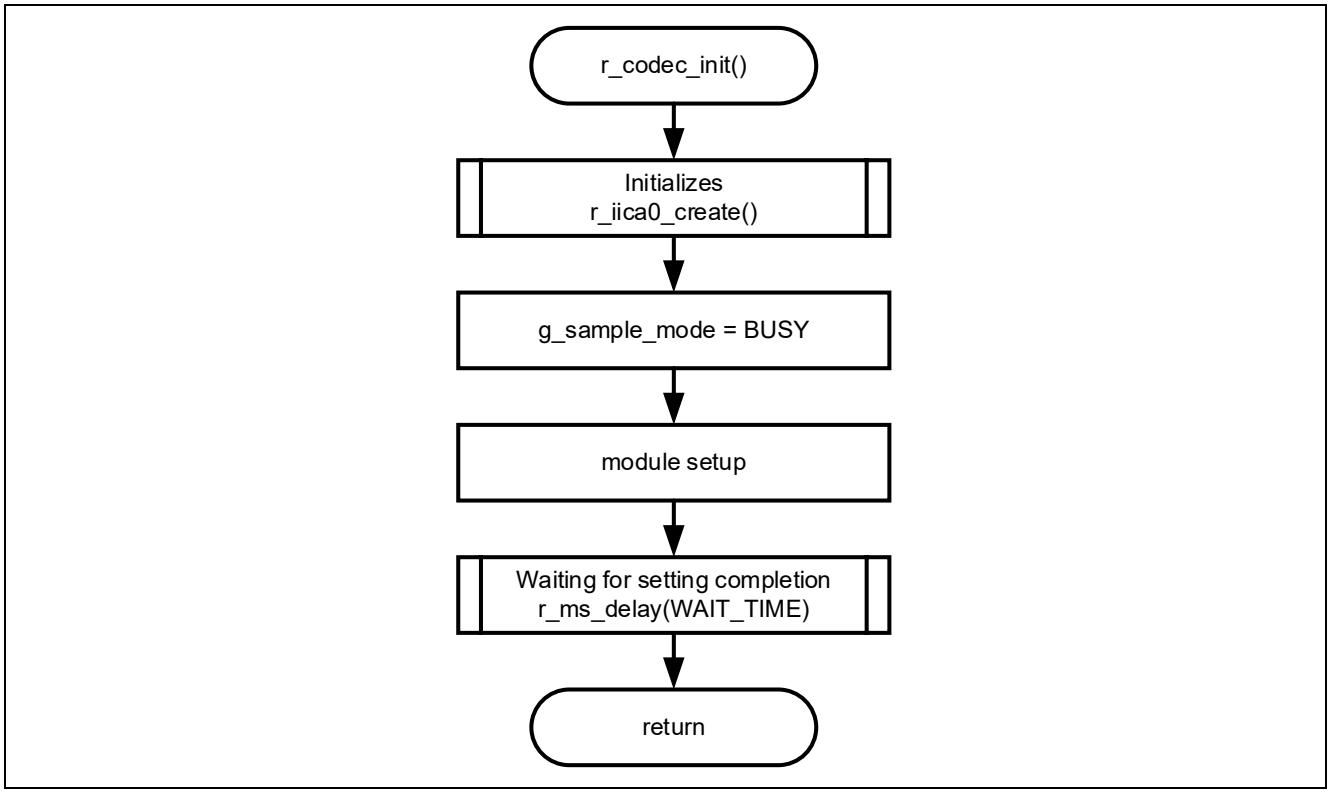
図 5-21 IICA0 割り込み処理



5.1.8.19 CODEC モジュールのセットアップ処理

図 5-22 に CODEC モジュールのセットアップ処理のフローチャートを示します。

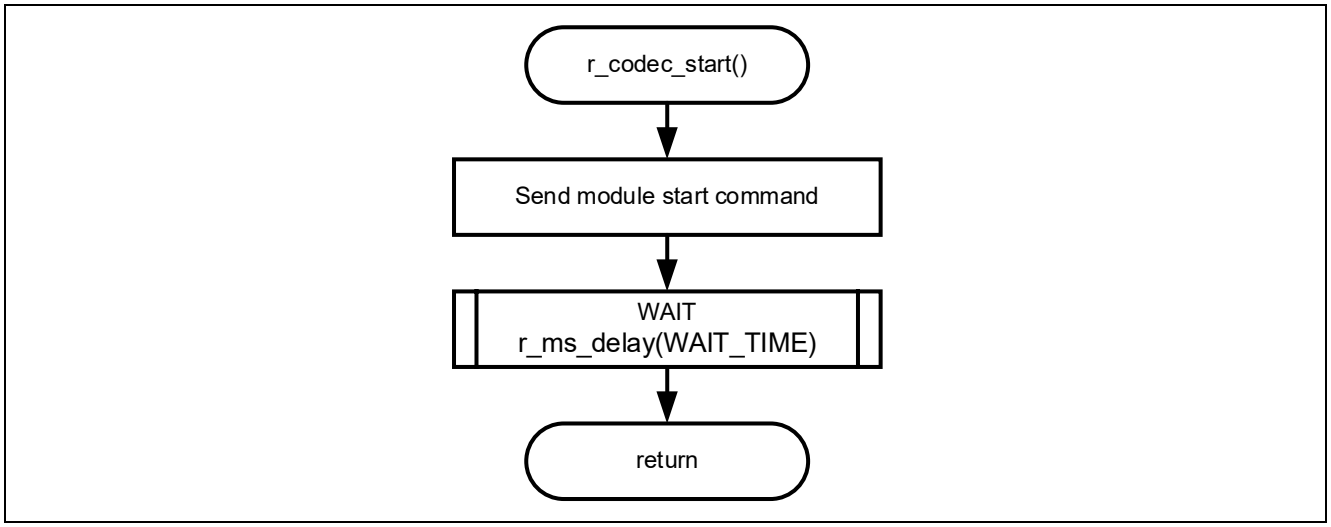
図 5-22 CODEC モジュールのセットアップ処理



5.1.8.20 CODEC モジュールの開始処理

図 5-23 に CODEC モジュールの開始処理のフローチャートを示します。

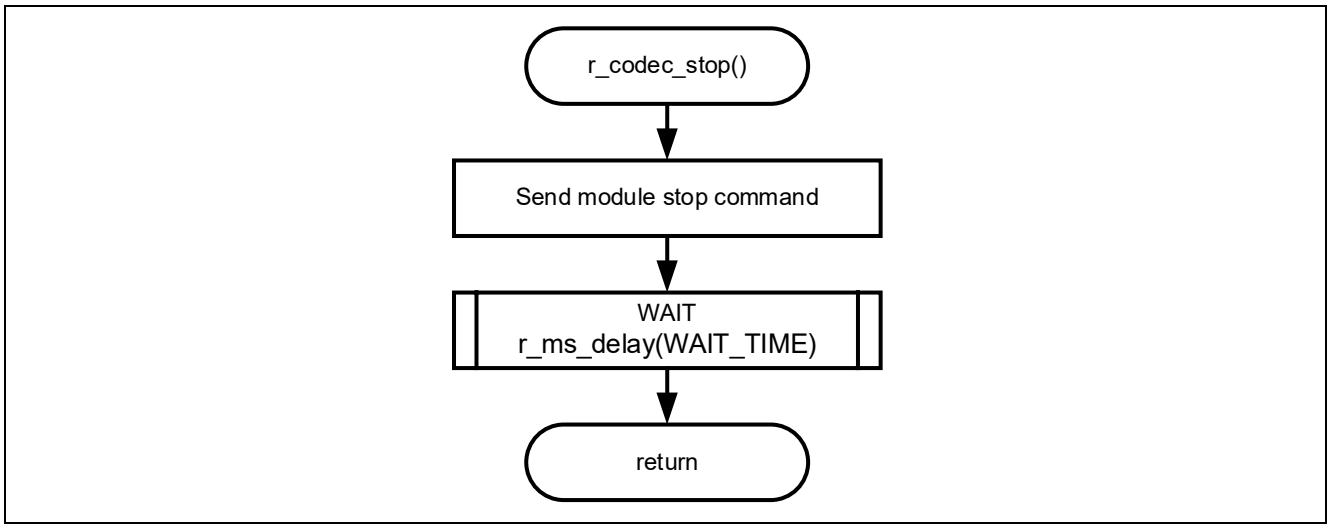
図 5-23 CODEC モジュールの開始処理



5.1.8.21 CODEC モジュールの停止処理

図 5-24 に CODEC モジュールの停止処理のフローチャートを示します。

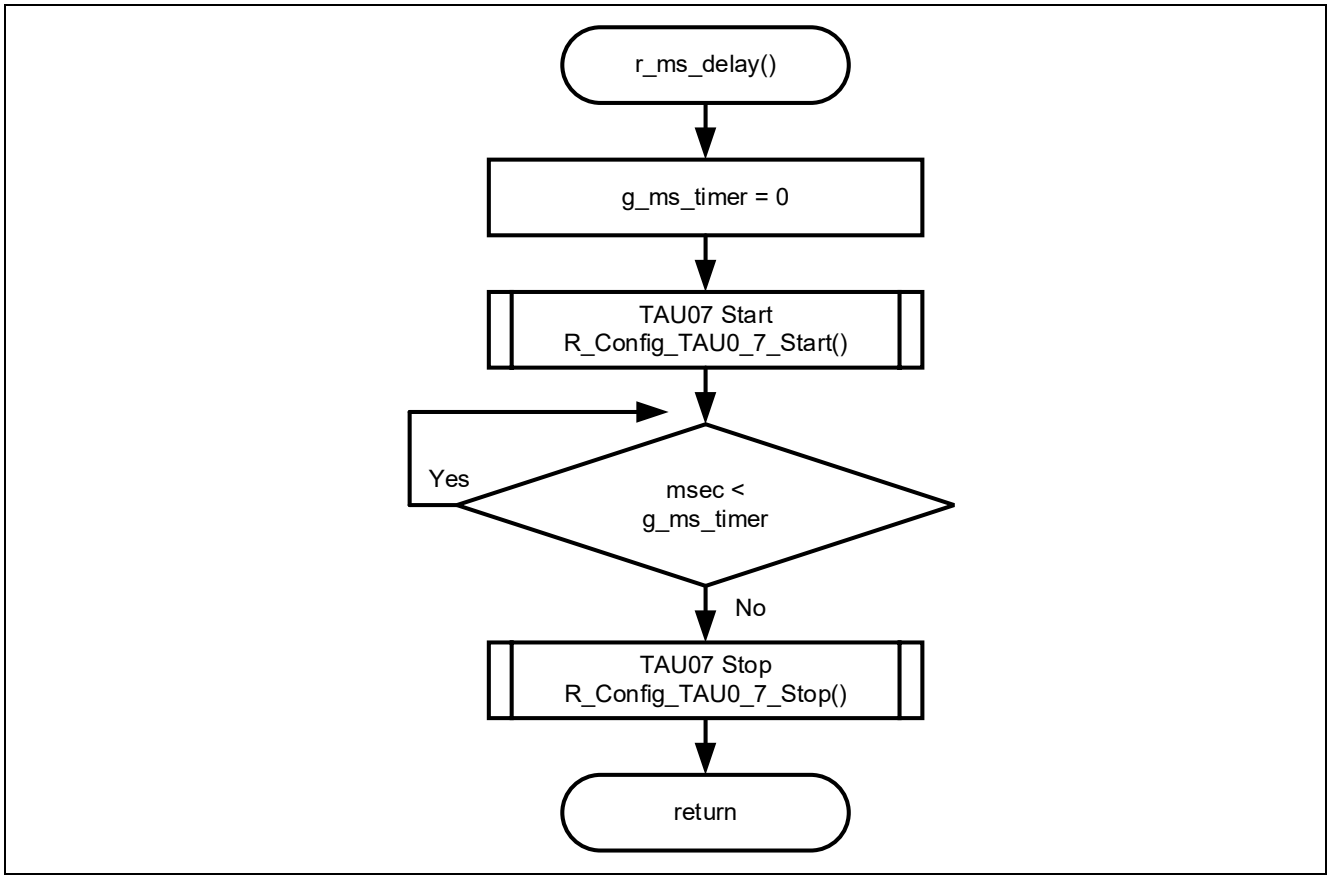
図 5-24 CODEC モジュールの停止処理



5.1.8.22 IICA0 ウエイト処理

図 5-25 に IICA0 ウエイト処理のフローチャートを示します。

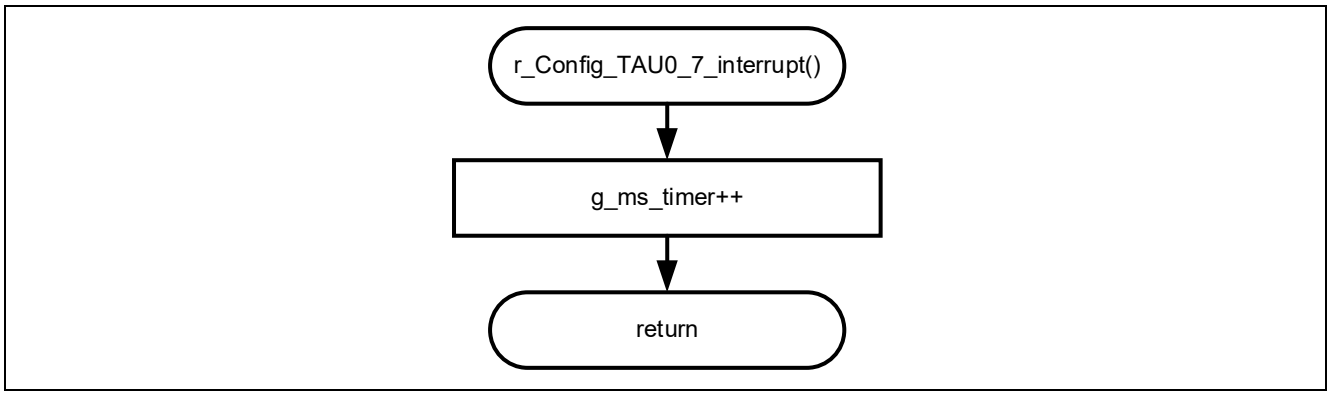
図 5-25　IICA0 ウエイト処理



5.1.8.23 TAU0 チャネル 7 割り込み処理

図 5-26 に TAU0 チャネル 7 割り込み処理のフローチャートを示します。

図 5-26　TAU0 チャネル 7 割り込み処理



5.2 応用例

本アプリケーションノートは、サンプルコードの他に以下のスマート・コンフィグレータの設定ファイルを格納しています。

r01an6420_elcl_i2s_slave.scfg

ファイルの説明と使用する上での設定例および注意事項を以下に示します。

5.2.1 r01an6420_elcl_i2s_slave.scfg

サンプルコードで使用しているスマート・コンフィグレータの設定ファイルです。スマート・コンフィグレータで設定されている全ての機能が含まれています。サンプルコードの設定は以下の通りです。

表 5-6 スマート・コンフィグレータの設定値

タグ名	コンポーネント	内容
クロック	-	動作モード：高速メインモード 2.4 (V) ~5.5 (V) EV_{DD} 設定：1.8V≦EV_{DD0}＜5.5V 高速オンチップ・オシレータ：32MHz f_{IHP}：32MHz f_{CLK}：32000kHz（高速オンチップ・オシレータ） f_{SXP}：32.768kHz（低速オンチップ・オシレータ）
システム	-	オンチップ・デバッグ動作設定：COM ポート[注1] 疑似 RRM/DMM 機能設定：使用する Start/Stop 関数機能設定：使用しない トレース機能設定：使用する セキュリティ ID 設定：セキュリティ ID を設定する セキュリティ ID：0x00000000000000000000 セキュリティ ID 認証失敗時の設定：フラシュ・メモリのデータを消去しない
コンポーネント	r_bsp	Start up select : Enable (use BSP startup) Control of invalid memory access detection : Disable RAM guard space (GRAM0-1) : Disabled Guard of control registers of port function (GPORT) : Disabled Guard of registers of interrupt function (GINT) : Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC) : Disabled Data flash access control (DFLEN) : Disables Initialization of peripheral functions by Code Generator/Smart Configurator : Enable API functions disable : Enable Parameter check enable : Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode : High-speed Enable user warm start callback (PRE) : Unused Enable user warm start callback (POST) : Unused Watchdog Timer refresh enable : Unused
	Config_LVD0	動作モード設定：リセット・モード 電圧検出設定：リセット発生電圧 (V_{LVD0})：1.86 (V)

表 5-7 スマート・コンフィグレータの設定値

タグ名	コンポーネント	内容
コンポーネント	Config_TAU0_7	コンポーネント：インターバル・タイマ 動作モード：16 ビット・カウンタ・モード リソース：TAU0_7 動作クロック：CK01 クロックソース：$f_{CLK}/2^8$ インターバル時間：1ms 割り込み設定：使用する 優先順位：レベル 2
	Config_PORT	コンポーネント：ポート ポート選択：PORT5 P53：出力（1 を出力）

5.2.1.1 クロック

サンプルコードで使用するクロックの設定を行います。

5.2.1.2 システム

サンプルコードのオンチップ・デバッグ設定を行います。

「オンチップ・デバッグ動作設定」、「セキュリティ ID 認証失敗時の設定」は、「表 4-2 オプション・バイト設定」の「オンチップ・デバッグ動作許可」に影響を与えます。設定を変更する際は注意してください。

5.2.1.3 r_bsp

サンプルコードのスタートアップの設定を行います。

5.2.1.4 Config_LVD0

サンプルコードの電源管理の設定を行います。

「表 4-2 オプション・バイト設定」の「LVD0 の設定」に影響を与えます。設定を変更する際は注意してください。

5.2.1.5 Config_TAU07

サンプルコードの TAU07 の設定を行います。

サンプルコードでは 1ms のインターバル・タイマとして使用します。

5.2.1.6 Config_PORT

サンプルコードのポートの設定を行います。

サンプルコードでは LED1 の制御に P53 を使用します。

5.2.2 サウンド・データの変更方法

5.2.2.1 サンプルコードに新規のサウンド・データを追加する場合

本サンプルコードでは、サンプリング周波数：48kHz、データ数：32bit のデータに限り新規のサウンド・データを追加できます。

(1) サンプルコードの sound_data.c の以下の g_tx_buf[]に新規のサウンド・データを追加してください。

```
#else /*  Not DATA_48K_32B */
/* For user setting */
const uint8_t g_tx_buf[] =
       {
              /* Add sound data here */
       };

#endif /* DATA_48K_32B *
```

(2) サンプルコードの sound_data.h で再生するサウンド・データを以下のように 0 に設定してください。

```
#define DATA_48K_32B (0)
```

(3) 同ファイルの送信データ数定数「DECODE_PCM_SIZE」を設定してください。

```
  #else /* For user setting */
  #define DECODE_PCM_SIZE    (0)

  #endif
```

以上の設定を行うことでサンプルコードに新規のサウンド・データを追加することができます。

5.2.3 他の CODEC モジュールを使用する場合

RS 社の Audio CODEC 754-1974 以外の CODEC モジュールを使用する場合、表 5-1 の RS 社の Audio CODEC 754-1974 モジュールの設定に必要なファイルを削除してください。また、main 関数内の CODEC モジュールセットアップ処理、CODEC モジュール開始処理を削除し、使用する CODEC モジュールに合わせて各種設定を追加してください。

6. 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。

以下の 2 種類のプロジェクト・サンプルコードを用意しています。

RL78/G23 がマスタの場合のサンプルコード：¥Master フォルダ以下のサンプルプロジェクト

RL78/G23 がスレーブの場合のサンプルコード：¥Slave フォルダ以下のサンプルプロジェクト

7. 参考ドキュメント

RL78/G23 ユーザーズマニュアル　ハードウェア編（R01UH0896J）

RL78 ファミリ　ユーザーズマニュアル　ソフトウェア編（R01US0015J）

RL78 スマート・コンフィグレータ　ユーザーガイド：CS+編（R20AN0580J）

RL78 スマート・コンフィグレータ　ユーザーガイド：e^2 studio 編（R20AN0579J）

RL78 スマート・コンフィグレータ　ユーザーガイド：IAR 編（R20AN0581J）

（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

（最新版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。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jun.24.22	-	初版発行

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。
10. 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。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配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。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。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。