RZ/T1グループ

DMAサンプルプログラム

R01AN2598JJ0140
Rev.1.40
2018.06.07

要旨

本アプリケーションノートでは、RZ/T1 グループ MCU を搭載した RSK RZ/T1 評価ボード対応の DMA（Direct Memory Access）サンプルプログラムについて説明します。

DMA 機能を使用して、内蔵 RAM 転送元エリア 1, 2, 3（各 256byte）の内容を、内蔵 RAM 転送先エリア 1, 2, 3（各 256byte）へコピーします。

- DMAC0 のチャネル 0 を使用
- DMAC 起動要因としてソフトウェアリクエストを使用
- 転送データ単位は 8bit 指定
- 転送モードはブロック転送を指定
- DMA 転送完了割り込み、DMA エラー割り込みを使用

対象デバイス

RZ/T1

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 4
2. 動作環境 5
3. 関連アプリケーションノート 6
4. 周辺機能説明 7
5. ハードウェア説明 8
5.1 ハードウェア構成例 8
5.2 使用端子一覧 8
6. ソフトウェア説明 9
6.1 動作概要 9
6.1.1 プロジェクト設定 9
6.1.2 使用準備 9
6.2 メモリマップ 10
6.2.1 サンプルプログラムのセクション配置 10
6.2.2 MPU の設定 10
6.2.3 例外処理ベクタテーブル 10
6.3 使用割り込み一覧 10
6.4 固定幅整数一覧 10
6.5 定数／エラーコード一覧 11
6.6 構造体／共用体／列挙型一覧 14
6.7 大域変数一覧 25
6.8 関数一覧 26
6.9 関数仕様 27
6.9.1 main 27
6.9.2 dma_err_callback 28
6.9.3 dma_dmac0_soft_trg_isr 29
6.9.4 R_DMA_Open 31
6.9.5 R_DMA_Control 34
6.9.6 R_DMA_Close 35
6.9.7 R_DMA_GetVersion 35
6.9.8 R_DMA_SoftTrgCommon 36
6.10 フローチャート 37
6.10.1 main 処理 37
6.10.2 dma_err_callback 処理 38
6.10.3 dma_dmac0_soft_trg_isr 処理 38
6.10.4 R_DMA_Open 処理 39
6.10.5 R_DMA_Control 処理 40
6.10.6 R_DMA_Close 処理 43
6.10.7 R_DMA_GetVersion 処理 43
6.10.8 R_DMA_SoftTrgCommon 処理 44
6.11 R_DMA_Control コマンド一覧 45

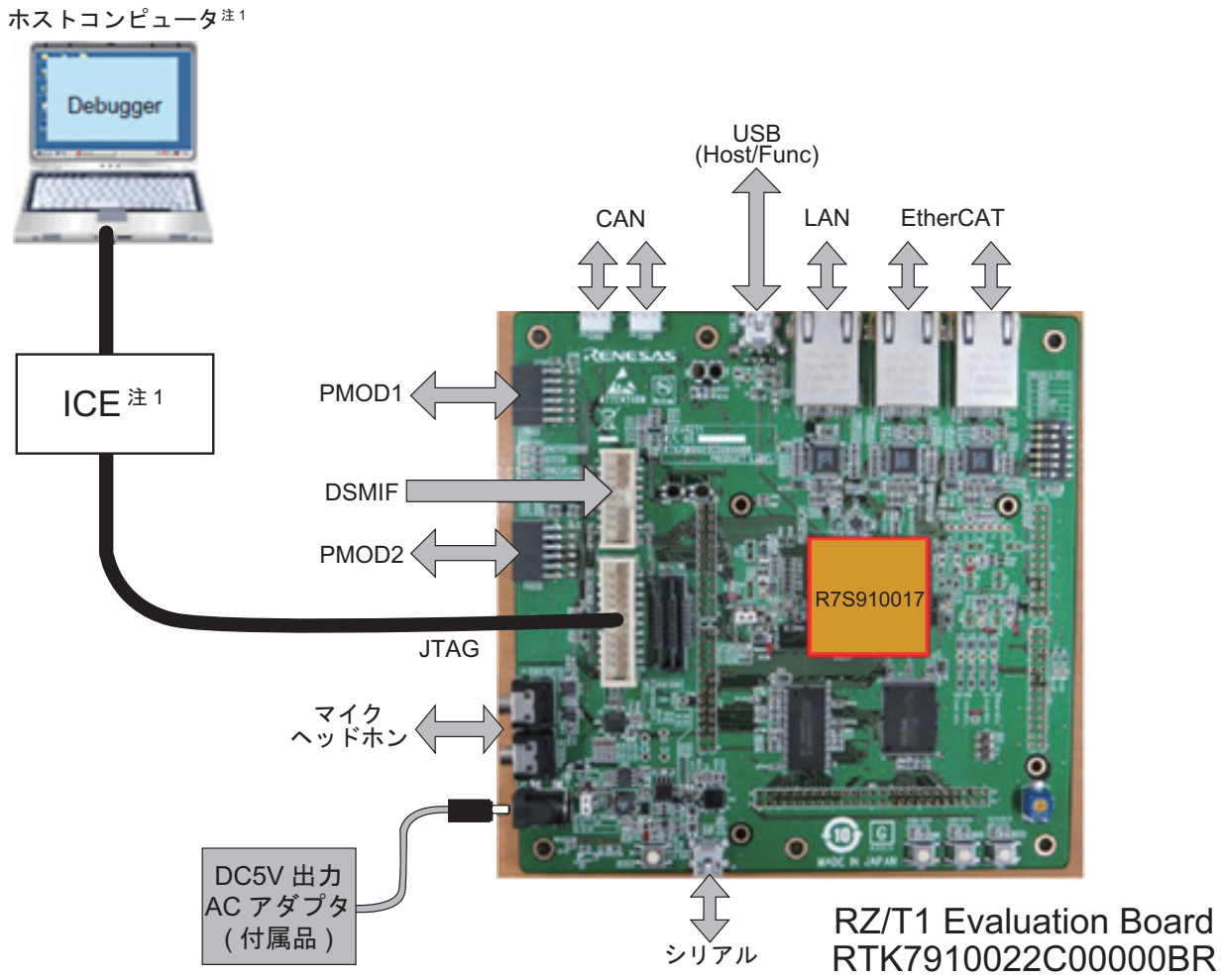
6.11.1 DMA_CMD_LINK_SET 46
6.11.2 DMA_CMD_SKIP_SET 48
6.11.3 DMA_CMD_DSCITVL_SET 49
7. サンプルコード 55
8. 参考ドキュメント 56

1.　仕様

周辺機能と動作環境を以下に示します。

表1.1　使用する周辺機能と用途

周辺機能	用途
RZ/T1内蔵DMAコントローラ（DMACAa）	DMAC0（Unit0）のチャネル0のDMA転送
RZ/T1内蔵割り込みコントローラ（ICUA）	割り込み制御 • DMAC0のソフトウェア起動によるDMA転送完了（ベクタ番号251） • DMAC0のDMA転送エラー（ベクタ番号293）



注 1.　別途お客様でご用意いただく必要があります。

図 1.1　動作環境

2. 動作環境

本アプリケーションノートのサンプルコードは、以下の環境を想定しています。

表2.1 動作環境

項目	内容
使用マイコン	RZ/T1グループ
動作周波数	CPUCLK = 450MHz
動作電圧	3.3V
統合開発環境	IARシステムズ製 Embedded Workbench® for Arm Version 8.20.2 Arm製 DS-5™ 5.26.2 RENESAS製 e2studio 6.1.0
動作モード	SPIブートモード 16ビットバスブートモード
使用ボード	RZ/T1 Evaluation Board （RTK7910022C00000BR）
使用デバイス （ボード上で使用する機能）	• NORフラッシュメモリ（CS0、CS1空間に接続） メーカ名：Macronix International Co..、型名：MX29GL512FLT2I-10Q • SDRAM（CS2、CS3空間に接続） メーカ名：Integrated Silicon Solution Inc、型名：IS42S16320D-7TL • シリアルフラッシュメモリ メーカ名：Macronix International Co..、型名：MX25L51245G

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。

- RZ/T1 グループ 初期設定 アプリケーションノート（R01AN2554JJ）

注 . 本アプリケーションノートで記載しないレジスタに関しては、RZ/T1 グループ 初期設定 アプリケーションノートで設定した値のまま使用します。

4. 周辺機能説明

動作モード、DMA コントローラ（DMACAa）、割り込みコントローラ（ICUA）についての基本的な内容は、RZ/T1 グル ― プ・ユーザーズマニュアル ハードウェア編に記載しています。

5. ハードウェア説明

5.1 ハードウェア構成例

DMA 機能である外部リクエスト、外部割り込みの機能は使用されないため、DMA 端子は外部と接続されていません。

5.2 使用端子一覧

DMA サンプルドライバが使用する端子はありません。

6.　ソフトウェア説明

6.1　動作概要

DMA サンプルプログラムの機能概要を**表 6.1　動作概要**に示します。また、**図 6.1** にシステムブロックを示します。

表 6.1　動作概要

機能	概要
処理概要	• 内蔵 RAM 転送元エリア 1, 2, 3（各 256byte）の内容を、DMA 機能を使用して内蔵 RAM 転送先エリア 1, 2, 3（各 256byte）へコピーする。
DMA 設定	• チャネル：DMAC0 チャネル 0 使用 • DMAC 起動要因：ソフトウェアリクエスト使用 • 転送データ単位：8bit 指定 • 転送モード：ブロック転送指定 • 割り込み要求：DMAC0 チャネル 0 の転送完了割り込み、DMAC0 の転送エラー割り込みを使用
割り込みハンドラ設定	• DMA 転送完了割り込みハンドラのサンプルプログラムへの実装 • ハンドラの処理内容は、DMA 転送完了回数のカウント
コールバック関数	• DMA エラー用コールバック関数のサンプルプログラムへの実装と、DMA ドライバへの登録 • コールバック関数の処理内容は、メインへエラーを通知するためのフラグセット
結果確認	• ソース、ディスティネーションのデータを比較することで、データコピーされていることを確認する。

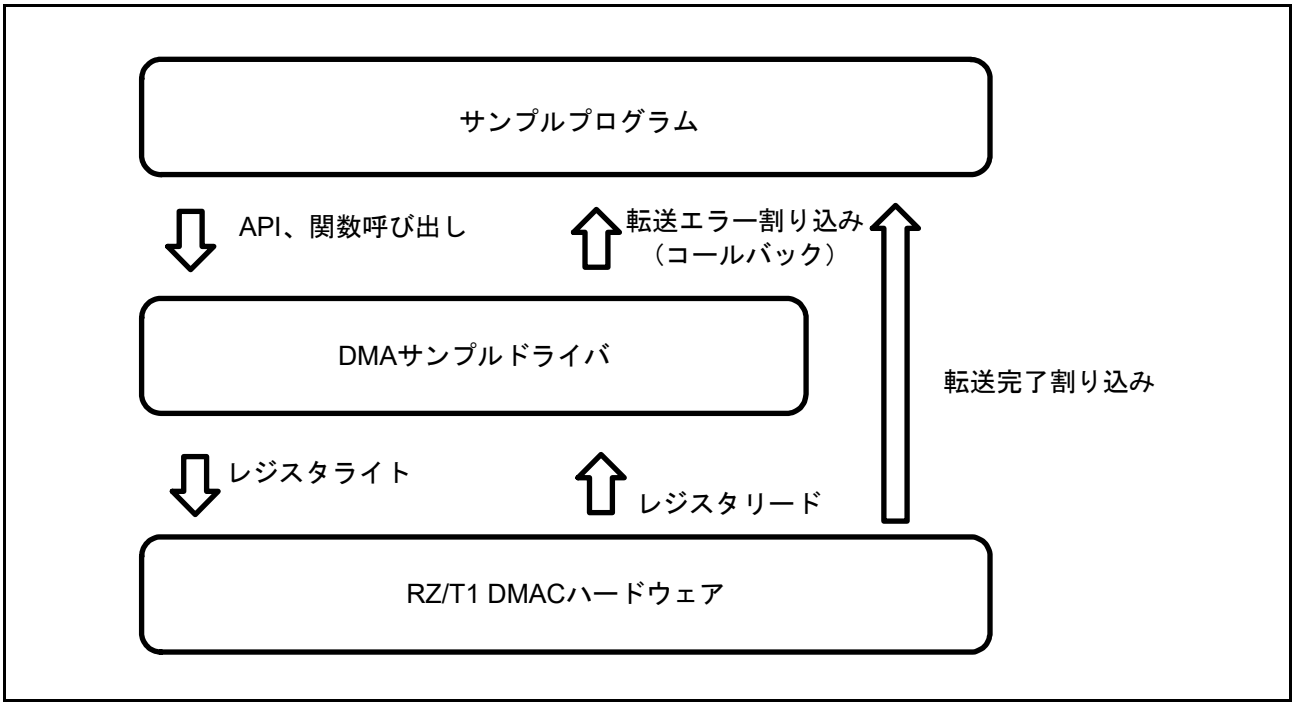


図 6.1　システムブロック

6.1.1　プロジェクト設定

開発環境となる EWARM 上で使用されるプロジェクト設定については、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.1.2　使用準備

本サンプルプログラムの実行準備は必要ありません。

6.2 メモリマップ

RZ/T1 グループのアドレス空間と RZ/T1 評価ボードのメモリマッピングについては、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.2.1 サンプルプログラムのセクション配置

サンプルプログラムで使用するセクションおよびサンプルプログラムの初期状態のセクション配置（ロードビュー）、スキャッタローディング機能を使用後のセクション配置（実行ビュー）は、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.2.2 MPU の設定

MPU の設定は、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.2.3 例外処理ベクタテーブル

例外処理ベクタテーブルについては、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.3 使用割り込み一覧

サンプルプログラムで使用する割り込みを以下に示します。

表6.2 サンプルプログラムで使用する割り込み

割り込み（要因ID）	優先度	処理概要
DMA転送ソフトウェア起動Unit0 （DMASRQ0）	3	DMA転送のソフトウェア起動によるDMA転送完了割り込み 転送完了時に、DMA転送完了回数をカウントする処理
DMA転送 転送エラー Unit0 （DMAERR0）	3	DMA転送の転送エラー 転送エラー発生時に、メイン処理へ通知するための情報を設定

6.4 固定幅整数一覧

表 6.3 にサンプルプログラムで使用する固定幅整数を示します。

表6.3 サンプルプログラムで使用する固定幅整数

シンボル	内容
int8_t	8ビット整数、符号あり（標準ライブラリにて定義）
int16_t	16ビット整数、符号あり（標準ライブラリにて定義）
int32_t	32ビット整数、符号あり（標準ライブラリにて定義）
int64_t	64ビット整数、符号あり（標準ライブラリにて定義）
uint8_t	8ビット整数、符号なし（標準ライブラリにて定義）
uint16_t	16ビット整数、符号なし（標準ライブラリにて定義）
uint32_t	32ビット整数、符号なし（標準ライブラリにて定義）
uint64_t	64ビット整数、符号なし（標準ライブラリにて定義）

6.5　定数／エラーコード一覧

表 6.4 に DMA サンプルプログラムにて使用する定数、表 6.5、表 6.6 に定数、表 6.7 にエラーコードを示します。

表6.4　DMAサンプルプログラム定数

定数名	設定値	内容
DMA_DATA_LENGTH	0x00000100	DMA転送データの全バイト長
DMA_DESCRIPTOR_NUM	3	設定するディスクリプタ数
DMA_CHITVL_VAL	0x0000	チャネルインターバルレジスタの設定値 DMA転送間隔を設定
DMA_DSCITVL_VAL	0x00	ディスクリプタインターバルレジスタの設定値 ディスクリプタ・リード間隔を設定

表6.5　DMAサンプルドライバ定数（コンフィグ可能）

定数名	設定値	内容
DMA_LINK_BUF_NUM	3	リンクモードのディスクリプタバッファ数
DMA_CMNCR_INIT	0x00001010	CMNCR レジスタ初期値 DMAバースト転送中にリフレッシュ要求を受け付ける。 DACK0/1/2をロウレベルアクティブ出力。 TEND0/1/2をロウレベルアクティブ出力。
DMA_DMAC0_DCTRL_A_INIT	0x00000000	DCTRL_A レジスタ初期値 DMAC0 チャネル0～7：固定優先順位モード
DMA_DMAC0_DCTRL_B_INIT	0x00000000	DCTRL_B レジスタ初期値 DMAC0 チャネル8～15：固定優先順位モード
DMA_DMAC1_DCTRL_A_INIT	0x00000000	DCTRL_A レジスタ初期値 DMAC1 チャネル0～7：固定優先順位モード
DMA_DMAC1_DCTRL_B_INIT	0x00000000	DCTRL_B レジスタ初期値 DMAC1 チャネル8～15：固定優先順位モード
DMA_CHITVL_INIT	0x00000000	CHITVL_(ch) (ch = 1～15) レジスタ初期値 DMA トランスファ間隔(0)を設定。
DMA_SCNT_INIT	0x00000000	SCNT_(ch) (ch = 1～15) レジスタ初期値 DMA転送元の連続アクセス空間サイズ(0)を設定
DMA_SSKP_INIT	0x00000000	SSKP_(ch) (ch = 1～15) レジスタ初期値 DMA転送元のスキップ空間サイズ(0)を設定
DMA_DCNT_INIT	0x00000000	DCNT_(注1) レジスタ初期値 DMA転送先の連続アクセス空間サイズ(0)を設定
DMA_DSKP_INIT	0x00000000	DSKP_(注1) レジスタ初期値 DMA転送先のスキップ空間サイズ(0)を設定
DMA_CFG_PARAM_CHECKING_ENABLE	1	DMAパラメータチェック許可スイッチ 0:禁止／1:許可
DMA_IR_PRIORITY_293_DMAERR_U0	3	ベクタ293（DMA Unit0のエラー）の割り込み優先レベル
DMA_IR_PRIORITY_294_DMAERR_U1	3	ベクタ294（DMA Unit1のエラー）の割り込み優先レベル

注1.　1～15

表6.6 DMAサンプルドライバ定数（1 / 2）

定数名	設定値	内容
DMA_HVA_WRITE_DATA	0x00000000u	HVAへの書き込みデータ
DMA_NUM_UNITS	2	DMA全ユニット数
DMA_NUM_CHANNELS	16	DMAユニット毎のチャネル数
DMA_NUM_REG_TBL	11	レジスタテーブル数
DMA_UNIT0	0	DMAユニット0
DMA_UNIT1	1	DMAユニット1
DMA_CHANNEL0	0	DMAユニット0
DMA_CHANNEL1	1	DMAユニット1
DMA_CHANNEL2	2	DMAユニット2
DMA_CHANNEL3	3	DMAユニット3
DMA_CHANNEL4	4	DMAユニット4
DMA_CHANNEL5	5	DMAユニット5
DMA_CHANNEL6	6	DMAユニット6
DMA_CHANNEL7	7	DMAユニット7
DMA_CHANNEL8	8	DMAユニット8
DMA_CHANNEL9	9	DMAユニット9
DMA_CHANNEL10	10	DMAユニット10
DMA_CHANNEL11	11	DMAユニット11
DMA_CHANNEL12	12	DMAユニット12
DMA_CHANNEL13	13	DMAユニット13
DMA_CHANNEL14	14	DMAユニット14
DMA_CHANNEL15	15	DMAユニット15
BITSHIFT_4	4	4ビットシフト数
BITSHIFT_16	16	16ビットシフト数
CHANNEL_MASK_4	0x0F	チャネルマスク4ビット
REGISTER_MASK_3	0x7	レジスタマスク3ビット
REGISTER_SET_BIT3	0x8	レジスタのbit3をセットするための値
REG_INDEX_CHSTAT	0	CHSTATレジスタのテーブルインデックス番号
REG_INDEX _CHCTRL	1	CHCTRLレジスタのテーブルインデックス番号
REG_INDEX _CHCFG	2	CHCFGレジスタのテーブルインデックス番号
REG_INDEX _CHITVL	3	CHITVLレジスタのテーブルインデックス番号
REG_INDEX _NXLA	4	NXLAレジスタのテーブルインデックス番号
REG_INDEX _CRLA	5	CRLAレジスタのテーブルインデックス番号
REG_INDEX _SCNT	6	SCNTレジスタのテーブルインデックス番号
REG_INDEX _SSKP	7	SSKPレジスタのテーブルインデックス番号
REG_INDEX _DCNT	8	DCNTレジスタのテーブルインデックス番号
REG_INDEX _DSKP	9	DSKPレジスタのテーブルインデックス番号
REG_INDEX _DMASEL	10	DMASELレジスタのテーブルインデックス番号
DATARAM_MIRR_STR_ADR	0x22000000	Data RAMエリアの開始アドレス
DATARAM_MIRR_END_ADR	0x22080000	Data RAMエリアの終了アドレス
DATARAM_MIRR_ADR_OFS	0x02000000	Data RAMエリアの非キャッシュエリアとミラー（キャッシュ）エリアのアドレスオフセット
INSTRAM_MIRR_STR_ADR	0x24000000	Instruction RAMエリアの開始アドレス
INSTRAM_MIRR_END_ADR	0x24080000	Instruction RAMエリアの終了アドレス

表6.6 DMAサンプルドライバ定数（2 / 2）

定数名	設定値	内容
INSTRAM_MIRR_ADR_OFS	0x20000000	Instruction RAMエリアの非キャッシュエリアとミラー（キャッシュ）エリアのアドレスオフセット
SPIBSC_MIRR_STR_ADR	0x30000000	SPIマルチI/Oバス空間の開始アドレス
SPIBSC_MIRR_END_ADR	0x34000000	SPIマルチI/Oバス空間の終了アドレス
SPIBSC_MIRR_ADR_OFS	0x20000000	SPIマルチI/Oバス空間の非キャッシュエリアとミラー（キャッシュ）エリアのアドレスオフセット
EX_MIRR_STR_ADR	0x40000000	外部アドレス空間の開始アドレス
EX_MIRR_END_ADR	0x58000000	外部アドレス空間の終了アドレス
EX_MIRR_ADR_OFS	0x20000000	外部アドレス空間の非キャッシュエリアとミラー（キャッシュ）エリアのアドレスオフセット
DMA_RZT1_VERSION_MAJOR	1	Major Version
DMA_RZT1_VERSION_MINOR	0	Minor Version

表6.7 DMAサンプルドライバ（エラーコード）

定数名	設定値	内容
DMA_SUCCESS	0	成功
DMA_ERR_BAD_UNIT	1	ユニット番号が不正
DMA_ERR_BAD_CHAN	2	チャネル番号が不正
DMA_ERR_CH_NOT_OPENED	3	オープンされていない
DMA_ERR_CH_NOT_CLOSED	4	クローズできない（すでにオープンされている）
DMA_ERR_UNKNOWN_CMD	5	不明なコマンド
DMA_ERR_INVALID_ARG	6	パラメータの内容が不正
DMA_ERR_NULL_PTR	7	パラメータが空（NULL）
DMA_ERR_BUFF_FULL	8	ディスクリプタ設定バッファに空きがない

6.6 構造体／共用体／列挙型一覧

図 6.2 に DMA サンプルドライバで使用する構造体／共用体を、図 6.3 に列挙型を示します。

```
/* vector setting */
typedef struct dma_vector_para_s
{
    uint16_t vector_no;                         /* vector no. */
    uint32_t priority;                          /* priority */
    void   (__irq __arm * pcallback_isr)(void); /* pointer to user callback interrupt function. */
} dma_vector_para_t;

/* link desctiptor format. */
typedef struct dma_link_fmt_s
{
    union {
        struct{
            uint16_t        sts         :16;    /* transfer size at the time of DSCFM = 3. */
            uint8_t                     :7;
            uint8_t         d           :1;     /* descriptor status. */
            uint8_t         lv          :1;     /* descriptor valid/invalid status. */
            uint8_t         le          :1;     /* link status of descriptor. */
            uint8_t         wbd         :1;     /* mask of write back. */
            uint8_t                     :1;
            uint8_t         dscfm       :4;     /* format of descriptor */
        } BIT;
        uint32_t LONG;
    } hd;
    uint32_t sa;                                /* source address. */
    uint32_t da;                                /* destination address. */
    uint32_t tb;                                /* transaction byte. */
    uint32_t chcfg;                             /* channel config. */
    uint32_t chitvl;                            /* channel interval. */
    uint32_t ex;                                /* channel extension. */
    uint32_t nla;                               /* next link address. */
} dma_link_fmt_t;

/* DMA handle */
typedef struct dma_handle_s
{
    uint8_t unit;
    uint8_t channel;
    dma_vector_para_t   vector_para;            /* vector parameter. */
    dma_ch_sts_t dma_ch_sts;                    /* channel state */
    void   (*pcallback_err)(void);              /* pointer to user callback function. */
    uint8_t write_index;                        /* write index of link buffer */
    dma_link_fmt_t link_buf[DMA_LINK_BUF_NUM];  /* link buffer */
    dma_mod_skip_t sskip_mode;                  /* source skip mode */
    dma_mod_skip_t dskip_mode;                  /* destionation skip mode */
} dma_handle_t;
```

```
/* link parameter of R_DMA_Control Function. */
typedef struct dma_control_link_para_s
{
    uint32_t sa;                                /* source address. */
    uint32_t da;                                /* destination address. */
    uint32_t tb;                                /* transaction byte. */
    union {
        struct{
            chcfg_sel_t         sel         :3; /* Channel Select */
            chcfg_reqd_t        reqd        :1; /* DMA Activation Request Source Select. */
            chcfg_reqmtd_t      reqmtd      :3; /* Method of Detecting DMA Transfer Request Signals. */
            uint8_t                         :1;
            chcfg_am_t          am          :3; /* ACK Mode. */
            chcfg_drrp_t        drrp        :1; /* Descriptor Reload Enable. */
            chcfg_sds_t         sds         :4; /* Source Data Size. */
            chcfg_dds_t         dds         :4; /* Destination Data Size. */
            chcfg_sad_t         sad         :1; /* Source Address Count Direction. */
            chcfg_dad_t         dad         :1; /* Destination Address Count Direction. */
            chcfg_tm_t          tm          :1; /* Transfer Mode. */
            chcfg_wonly_t       wonly       :1; /* Write-Only Mode. */
            chcfg_dem_t         dem         :1; /* Transfer Completion Interrupt Mask. */
            uint8_t                         :1;
            chcfg_dim_t         dim         :1; /* Descriptor Interrupt Mask. */
            chcfg_sbe_t         sbe         :1; /* Buffer Flush Enable. */
            chcfg_rsel_t        rsel        :1; /* Next Register Select. */
            chcfg_rsw_t         rsw         :1; /* RSEL Reverse. */
            chcfg_ren_t         ren         :1; /* Register Set Enable. */
            chcfg_dms_t         dms         :1; /* DMA Mode Select. */
        } BIT;
        unsigned long LONG;
    } chcfg;
    uint16_t chitvl;                            /* channel interval. */
} dma_control_link_para_t;

/* skip parameter of R_DMA_Control Function. */
typedef struct dma_control_skip_para_s
{
    /* SCNT register */
    uint32_t        scnt;                       /* Source Continuous Access Size. */

    /* SSKP register */
    uint32_t        sskp;                       /* Source Skip Size. */

    /* DCNT register */
    uint32_t        dcnt;                       /* Destination Continuous Access Size. */

    /* DSKP register */
    uint32_t        dskp;                       /* Destination Skip Size. */
} dma_control_skip_para_t;
```

```
/* descriptor interval parameter of R_DMA_Control Function. */
typedef struct dma_control_dscitvl_para_s
{
    /* DSCITVL register */
    uint8_t        dscitvl;                    /* Descriptor Interval. */
} dma_control_dscitvl_para_t;

/* CHSTAT register */
typedef union dma_chstat_reg_u
{
    struct{
        uint8_t         en          :1;     /* DMA Activation Enable. */
        uint8_t         rqst        :1;     /* DMA Transfer Request. */
        uint8_t         tact        :1;     /* DMAC Operating Status. */
        uint8_t         sus         :1;     /* Suspend. */
        uint8_t         er          :1;     /* DMA Error. */
        uint8_t         end         :1;     /* DMA Transfer Completion Interrupt. */
        uint8_t                     :1;
        uint8_t         sr          :1;     /* Next Register Select. */
        uint8_t         dl          :1;     /* Descriptor Load. */
        uint8_t         dw          :1;     /* Descriptor Write Back. */
        uint8_t         der         :1;     /* Descriptor Error. */
        uint8_t         mode        :1;     /* DMA Mode. */
        uint8_t                     :4;
        uint8_t         intm        :1;     /* Interrupt Request Mask. */
        uint8_t         dmarqm      :1;     /* DMA Activation Request Mask. */
        uint8_t         swprq       :1;     /* Forced Ejection Request. */
        uint8_t                     :5;
        uint8_t         dnum        :8;     /* Amount of Data in the Buffer. */
    } BIT;
    unsigned long LONG;
} dma_chstat_reg_t;

/* CHCFG register */
typedef union dma_chcfg_reg_u
{
    /* CHCFG register */
    struct{
        chcfg_sel_t         sel         :3; /* Channel Select */
        chcfg_reqd_t        reqd        :1; /* DMA Activation Request Source Select. */
        chcfg_reqmtd_t      reqmtd      :3; /* Method of Detecting DMA Transfer Request Signals. */
        uint8_t                         :1;
        chcfg_am_t          am          :3; /* ACK Mode. */
        chcfg_drrp_t        drrp        :1; /* Descriptor Reload Enable. */
        chcfg_sds_t         sds         :4; /* Source Data Size. */
        chcfg_dds_t         dds         :4; /* Destination Data Size. */
        chcfg_sad_t         sad         :1; /* Source Address Count Direction. */
        chcfg_dad_t         dad         :1; /* Destination Address Count Direction. */
```

```
        chcfg_tm_t          tm          :1; /* Transfer Mode. */
        chcfg_wonly_t       wonly       :1; /* Write-Only Mode. */
        chcfg_dem_t         dem         :1; /* Transfer Completion Interrupt Mask. */
        uint8_t                         :1;
        chcfg_dim_t         dim         :1; /* Descriptor Interrupt Mask. */
        chcfg_sbe_t         sbe         :1; /* Buffer Flush Enable. */
        chcfg_rsel_t        rsel        :1; /* Next Register Select. */
        chcfg_rsw_t         rsw         :1; /* RSEL Reverse. */
        chcfg_ren_t         ren         :1; /* Register Set Enable. */
        chcfg_dms_t         dms         :1; /* DMA Mode Select. */
    } BIT;
    unsigned long LONG;
} dma_chcfg_reg_t;

/* CHCTRL register */
typedef union dma_chctrl_reg_u
{
    struct{
        chctrl_seten_t      seten       :1; /* DMA Activation Enable. */
        chctrl_clren_t      clren       :1; /* DMA Activation Enable Clear. */
        uint8_t                         :1;
        chctrl_swrst_t      swrst       :1; /* Software Reset. */
        chctrl_clrrq_t      clrrq       :1; /* DMA Transfer Request Clear. */
        chctrl_clrend_t     clrend      :1; /* END Clear. */
        uint8_t                         :1;
        chctrl_clrder_t     clrder      :1; /* DER Clear. */
        chctrl_setsus_t     setsus      :1; /* Suspend Request. */
        chctrl_clrsus_t     clrsus      :1; /* Suspend Clear. */
        uint8_t                         :2;
        chctrl_setren_t     setren      :1; /* REN Set Enable. */
        uint8_t                         :1;
        chctrl_setsswprq_t  setsswprq   :1; /* Software Forced Ejection Request. */
        uint8_t                         :1;
        chctrl_setintm_t    setintm     :1; /* Interrupt Request Mask. */
        chctrl_clrintm_t    clrintm     :1; /* Interrupt Request Mask Clear. */
        chctrl_setdmarqm_t  setdmarqm   :1; /* DMA Activation Request Mask. */
        chctrl_clrdmarqm_t  clrdmarqm   :1; /* DMA Activation Request Mask Clear. */
        uint16_t                        :12;
    } BIT;
    unsigned long LONG;
} dma_chctrl_reg_t;
```

図 6.2　構造体／共用体

```
/* DMAC0/1 channel 0-15 */
typedef enum dma_unit_channel_s
{
    DMA_DMAC0_CH0           = (0),  /* Channel 0 of DMAC0. */
    DMA_DMAC0_CH1,                  /* Channel 1 of DMAC0. */
    DMA_DMAC0_CH2,                  /* Channel 2 of DMAC0. */
    DMA_DMAC0_CH3,                  /* Channel 3 of DMAC0. */
    DMA_DMAC0_CH4,                  /* Channel 4 of DMAC0. */
    DMA_DMAC0_CH5,                  /* Channel 5 of DMAC0. */
    DMA_DMAC0_CH6,                  /* Channel 6 of DMAC0. */
    DMA_DMAC0_CH7,                  /* Channel 7 of DMAC0. */
    DMA_DMAC0_CH8,                  /* Channel 8 of DMAC0. */
    DMA_DMAC0_CH9,                  /* Channel 9 of DMAC0. */
    DMA_DMAC0_CH10,                 /* Channel 10 of DMAC0. */
    DMA_DMAC0_CH11,                 /* Channel 11 of DMAC0. */
    DMA_DMAC0_CH12,                 /* Channel 12 of DMAC0. */
    DMA_DMAC0_CH13,                 /* Channel 13 of DMAC0. */
    DMA_DMAC0_CH14,                 /* Channel 14 of DMAC0. */
    DMA_DMAC0_CH15,                 /* Channel 15 of DMAC0. */
    DMA_DMAC1_CH0,                  /* Channel 0 of DMAC1. */
    DMA_DMAC1_CH1,                  /* Channel 1 of DMAC1. */
    DMA_DMAC1_CH2,                  /* Channel 2 of DMAC1. */
    DMA_DMAC1_CH3,                  /* Channel 3 of DMAC1. */
    DMA_DMAC1_CH4,                  /* Channel 4 of DMAC1. */
    DMA_DMAC1_CH5,                  /* Channel 5 of DMAC1. */
    DMA_DMAC1_CH6,                  /* Channel 6 of DMAC1. */
    DMA_DMAC1_CH7,                  /* Channel 7 of DMAC1. */
    DMA_DMAC1_CH8,                  /* Channel 8 of DMAC1. */
    DMA_DMAC1_CH9,                  /* Channel 9 of DMAC1. */
    DMA_DMAC1_CH10,                 /* Channel 10 of DMAC1. */
    DMA_DMAC1_CH11,                 /* Channel 11 of DMAC1. */
    DMA_DMAC1_CH12,                 /* Channel 12 of DMAC1. */
    DMA_DMAC1_CH13,                 /* Channel 13 of DMAC1. */
    DMA_DMAC1_CH14,                 /* Channel 14 of DMAC1. */
    DMA_DMAC1_CH15                  /* Channel 15 of DMAC1. */
} dma_unit_channel_t;

/* DMA API error codes */
typedef enum dma_err_e
{
    DMA_SUCCESS             = (0),  /* success */
    DMA_ERR_BAD_UNIT,               /* Invalid unit number. */
    DMA_ERR_BAD_CHAN,               /* Invalid channel number. */
    DMA_ERR_CH_NOT_OPENED,          /* Channel not yet opened. */
    DMA_ERR_CH_NOT_CLOSED,          /* Channel still open from previous open. */
    DMA_ERR_UNKNOWN_CMD,            /* Control command is not recognized. */
    DMA_ERR_INVALID_ARG,            /* Argument is not valid for parameter. */
    DMA_ERR_NULL_PTR,               /* Received null pointer; missing required argument. */
    DMA_ERR_BUFF_FULL               /* LINK mode buffer full. */
} dma_err_t;
```

```
/* DMA API command codes */
typedef enum dma_cmd_e
{
    DMA_CMD_LINK_SET        = (0),  /* setting of link descriptor. */
    DMA_CMD_SKIP_SET        = (1),  /* setting of skip parameter. */
    DMA_CMD_DSCITVL_SET      = (2),  /* setting of descriptor interval. */
} dma_cmd_t;

/* channel state */
typedef enum
{
    DMA_CH_STS_CLOSE        = (0),  /* channel close state. */
    DMA_CH_STS_OPEN         = (1)   /* channel open state. */
} dma_ch_sts_t;

/* skip mode Select */
typedef enum
{
    DMA_MOD_SKIP_INVALID    = (0),  /* skip mode invalid. */
    DMA_MOD_SKIP_VALID      = (1)   /* skip mode valid. */
} dma_mod_skip_t;

/* DMA Activation Enable (CHCTRL register) */
typedef enum
{
    CHCTRL_SETEN_NOP        = (0),  /* Operation is not affected. */
    CHCTRL_SETEN_EN         = (1)   /* DMA Activation Enable. */
} chctrl_seten_t;

/* DMA Activation Enable Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLREN_NOP        = (0),  /* Operation is not affected. */
    CHCTRL_CLREN_STP        = (1)   /* DMA Activation Enable Clear. */
} chctrl_clren_t;

/* Software Reset (CHCTRL register) */
typedef enum
{
    CHCTRL_SWRST_NOP        = (0),  /* Operation is not affected. */
    CHCTRL_SWRST_RST        = (1)   /* DMA Software Reset. */
} chctrl_swrst_t;

/* DMA Transfer Request Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLRRQ_NOP        = (0),  /* Operation is not affected. */
    CHCTRL_CLRRQ_CLR        = (1)   /* DMA Transfer Request Clear. */
} chctrl_clrrq_t;

/* END Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLREND_NOP       = (0),  /* Operation is not affected. */
    CHCTRL_CLREND_CLR       = (1)   /* DMA END Clear. */
} chctrl_clrend_t;
```

```
/* DER Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLRDER_NOP       = (0),  /* Operation is not affected. */
    CHCTRL_CLRDER_CLR       = (1)   /* DMA DER clear. */
} chctrl_clrder_t;

/* Suspend Request (CHCTRL register) */
typedef enum
{
    CHCTRL_SETSUS_NOP       = (0),  /* Operation is not affected. */
    CHCTRL_SETSUS_SUS       = (1)   /* DMA Suspend Request. */
} chctrl_setsus_t;

/* Suspend Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLRSUS_NOP       = (0),  /* Operation is not affected. */
    CHCTRL_CLRSUS_SUS       = (1)   /* DMA Suspend Clear. */
} chctrl_clrsus_t;

/* REN Set Enable (CHCTRL register) */
typedef enum
{
    CHCTRL_SETREN_NOP       = (0),  /* Operation is not affected. */
    CHCTRL_SETREN_SET       = (1)   /* REN Set Enable. (Continue to run the DMA transferred.) */
} chctrl_setren_t;

/* Software Forced Ejection Request (CHCTRL register) */
typedef enum
{
    CHCTRL_SETSSWPRQ_NOP    = (0),  /* Operation is not affected. */
    CHCTRL_SETSSWPRQ_OUT    = (1)   /* Software Forced Ejection Request set. */
} chctrl_setsswprq_t;

/* Interrupt Request Mask (CHCTRL register) */
typedef enum
{
    CHCTRL_SETINTM_NOP      = (0),  /* Operation is not affected. */
    CHCTRL_SETINTM_MSK      = (1)   /* Interrupt Request Mask. */
} chctrl_setintm_t;

/* Interrupt Request Mask Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLRINTM_NOP      = (0),  /* Operation is not affected. */
    CHCTRL_CLRINTM_CLR      = (1)   /* Interrupt Request Mask Clear. */
} chctrl_clrintm_t;

/* DMA Activation Request Mask (CHCTRL register) */
typedef enum
{
    CHCTRL_SETDMARQM_NOP    = (0),  /* Operation is not affected. */
    CHCTRL_SETDMARQM_MSK    = (1)   /* DMA Activation Request Mask. */
} chctrl_setdmarqm_t;
```

```
/* DMA Activation Request Mask Clear (CHCTRL register) */
typedef enum
{
    CHCTRL_CLRDMARQM_NOP    = (0),  /* Operation is not affected. */
    CHCTRL_CLRDMARQM_CLR    = (1)   /* DMA Activation Request Mask Clear. */
} chctrl_clrdmarqm_t;

/* Channel Select (CHCFG register) */
typedef enum
{
    CHCFG_SEL_CH0           = (0),  /* channel 0 select */
    CHCFG_SEL_CH1           = (1),  /* channel 1 select */
    CHCFG_SEL_CH2           = (2),  /* channel 2 select */
    CHCFG_SEL_CH3           = (3),  /* channel 3 select */
    CHCFG_SEL_CH4           = (4),  /* channel 4 select */
    CHCFG_SEL_CH5           = (5),  /* channel 5 select */
    CHCFG_SEL_CH6           = (6),  /* channel 6 select */
    CHCFG_SEL_CH7           = (7),  /* channel 7 select */
    CHCFG_SEL_CH8           = (0),  /* channel 8 select */
    CHCFG_SEL_CH9           = (1),  /* channel 9 select */
    CHCFG_SEL_CH10          = (2),  /* channel 10 select */
    CHCFG_SEL_CH11          = (3),  /* channel 11 select */
    CHCFG_SEL_CH12          = (4),  /* channel 12 select */
    CHCFG_SEL_CH13          = (5),  /* channel 13 select */
    CHCFG_SEL_CH14          = (6),  /* channel 14 select */
    CHCFG_SEL_CH15          = (7)   /* channel 15 select */
} chcfg_sel_t;

/* DMA Activation Request Source Select (CHCFG register) */
typedef enum
{
    CHCFG_REQD_SRC          = (0),  /* Requested by a transfer source module. */
    CHCFG_REQD_DES          = (1)   /* Requested by a transfer destination module. */
} chcfg_reqd_t;

/* Method of Detecting DMA Transfer Request Signals (CHCFG register) */
typedef enum
{
    CHCFG_REQMTD_INVALID    = (0),  /* detection invalid */
    CHCFG_REQMTD_EDG_FAL    = (1),  /* Detects the falling edge */
    CHCFG_REQMTD_EDG_RIS    = (2),  /* Detects the rising edge */
    CHCFG_REQMTD_LVL_LOW    = (5),  /* Detects the low level */
    CHCFG_REQMTD_LVL_HI     = (6)   /* Detects the high level */
} chcfg_reqmtd_t;

/* ACK Mode (CHCFG register) */
typedef enum
{
    CHCFG_AM_LVLMODE        = (1),  /* Level mode. */
    CHCFG_AM_BUSMODE        = (2),  /* Bus cycle mode. */
    CHCFG_AM_MSK            = (4)   /* Masks DACK/TEND output. */
} chcfg_am_t;
```

```
/* Descriptor Reload Enable (CHCFG register) */
typedef enum
{
    CHCFG_DRRP_END          = (0),  /* stops the operation. */
    CHCFG_DRRP_REP          = (1)   /* Continues reading the same descriptor. */
} chcfg_drrp_t;

/* Source Data Size (CHCFG register) */
typedef enum
{
    CHCFG_SDS_8BIT          = (0),  /* 8bit */
    CHCFG_SDS_16BIT         = (1),  /* 16bit */
    CHCFG_SDS_32BIT         = (2),  /* 32bit */
    CHCFG_SDS_128BIT        = (4),  /* 128bit */
    CHCFG_SDS_256BIT        = (5),  /* 256bit */
    CHCFG_SDS_512BIT        = (6),  /* 512bit */
    CHCFG_SDS_8BIT_SKP      = (8),  /* 8bit (skip mode) */
    CHCFG_SDS_16BIT_SKP     = (9),  /* 16bit (skip mode) */
    CHCFG_SDS_32BIT_SKP     = (10), /* 32bit (skip mode) */
    CHCFG_SDS_128BIT_SKP    = (12), /* 128bit (skip mode) */
    CHCFG_SDS_256BIT_SKP    = (13), /* 256bit (skip mode) */
    CHCFG_SDS_512BIT_SKP    = (14)  /* 512bit (skip mode) */
} chcfg_sds_t;

/* Destination Data Size (CHCFG register) */
typedef enum
{
   CHCFG_DDS_8BIT          = (0),  /* 8bit */
    CHCFG_DDS_16BIT         = (1),  /* 16bit */
    CHCFG_DDS_32BIT         = (2),  /* 32bit */
    CHCFG_DDS_128BIT        = (4),  /* 128bit */
    CHCFG_DDS_256BIT        = (5),  /* 256bit */
    CHCFG_DDS_512BIT        = (6),  /* 512bit */
    CHCFG_DDS_8BIT_SKP      = (8),  /* 8bit (skip mode) */
    CHCFG_DDS_16BIT_SKP     = (9),  /* 16bit (skip mode) */
    CHCFG_DDS_32BIT_SKP     = (10), /* 32bit (skip mode) */
    CHCFG_DDS_128BIT_SKP    = (12), /* 128bit (skip mode) */
    CHCFG_DDS_256BIT_SKP    = (13), /* 256bit (skip mode) */
    CHCFG_DDS_512BIT_SKP    = (14)  /* 512bit (skip mode) */
} chcfg_dds_t;

/* Source Address Count Direction (CHCFG register) */
typedef enum
{
    CHCFG_SAD_INC           = (0),  /* Increment. */
    CHCFG_SAD_FIX           = (1)   /* Fixed. */
} chcfg_sad_t;

/* Destination Address Count Direction (CHCFG register) */
typedef enum
{
    CHCFG_DAD_INC           = (0),  /* Increment. */
    CHCFG_DAD_FIX           = (1)   /* Fixed. */
} chcfg_dad_t;
```

```
 /* Transfer Mode (CHCFG register) */
typedef enum
{
    CHCFG_TM_SIN            = (0),  /* Single transfer mode. */
    CHCFG_TM_BLK            = (1)   /* Block transfer mode. */
} chcfg_tm_t;

/* Write-Only Mode (CHCFG register) */
typedef enum
{
    CHCFG_WONLY_INVALID     = (0),  /* write-only mode invalid. */
    CHCFG_WONLY_VALID       = (1)   /* write-only mode valid. */
} chcfg_wonly_t;

/* Transfer Completion Interrupt Mask (CHCFG register) */
typedef enum
{
    CHCFG_DEM_NMSK          = (0),  /* Does not mask. */
    CHCFG_DEM_MSK           = (1)   /* Masks. */
} chcfg_dem_t;

/* Descriptor Interrupt Mask (CHCFG register) */
typedef enum
{
    CHCFG_DIM_NMSK          = (0),  /* Does not mask a DMA transfer completion interrupt. */
    CHCFG_DIM_MSK           = (1)   /* Masks a DMA transfer completion interrupt. */
} chcfg_dim_t;

/* Buffer Flush Enable (CHCFG register) */
typedef enum
{
    CHCFG_SBE_NOUT          = (0),  /* Stops transfer without flushing data in the buffer. */
    CHCFG_SBE_OUT           = (1)   /* Stops transfer after flushing data in the buffer. */
} chcfg_sbe_t;

/* Next Register Select (CHCFG register) */
typedef enum
{
    CHCFG_RSEL_R0           = (0),  /* Executes Next0 Register Set. */
    CHCFG_RSEL_R1           = (1)   /* Executes Next1 Register Set. */
} chcfg_rsel_t;
```

```
/* RSEL Reverse (CHCFG register) */
typedef enum
{
    CHCFG_RSW_NML           = (0),  /* Does not reverse the RSEL bit when DMA transfer completes. */
    CHCFG_RSW_REV           = (1)   /* Reverses the RSEL bit when DMA transfer completes. */
} chcfg_rsw_t;

/* Register Set Enable (CHCFG register) */
typedef enum
{
    CHCFG_REN_NEXE          = (0),  /* Does not perform DMA transfer accordingly. */
    CHCFG_REN_EXE           = (1)   /* Performs DMA transfer accordingly. */
} chcfg_ren_t;

/* DMA Mode Select (CHCFG register) */
typedef enum
{
    CHCFG_DMS_REG           = (0),  /* Register mode. */
    CHCFG_DMS_LNK           = (1)   /* Link mode. */
} chcfg_dms_t;

/* Priority Control Select (DCTRL register) */
typedef enum
{
    DCTRL_PR_FIX            = (0),  /* Fixed priority mode. */
    DCTRL_PR_RNDRBN         = (1)   /* Round-robin mode. */
} dctrl_pr_t;

/* DMA power setting value. */
typedef enum
{   /* Values will be uses as bit flags. */
    DMA_POWER_ON        = (0),      /* power on */
    DMA_POWER_OFF       = (1)       /* power off */
} dma_power_t;

/* interrupt enable/disable parameter. */
typedef enum
{
    DMA_ICU_INTR_DISABLE,           /* disable */
    DMA_ICU_INTR_ENABLE             /* enable */
} dma_icu_intr_para_t;

/* Status of the vector number search */
typedef enum
{
    DMA_SEARCH_CONTINUE,            /* Search continues. */
    DMA_SEARCH_EXIT                 /* Search exit. */
} dma_search_sts_t;
```

図 6.3　　列挙型

6.7 大域変数一覧

表 6.8 に DMA のサンプルプログラム／ドライバの大域変数一覧を示します。

表6.8 大域変数一覧

型	変数名	内容	使用関数
volatile uint8_t	g_dma_cmp_dsc_cnt	DMA転送完了のディスクリプタカウンタ	main dma_dmac0_soft_trg_isr
volatile uint8_t	g_dma_err_flag	DMAエラーフラグ	main dma_err_callback
static uint8_t	des_data0[256]	DMA転送先エリア0	main
static uint8_t	des_data1[256]	DMA転送先エリア1	main
static uint8_t	des_data2[256]	DMA転送先エリア2	main
static const uint8_t	src_data0[256]	DMA転送元エリア0	main
static const uint8_t	src_data1[256]	DMA転送元エリア1	main
static const uint8_t	src_data2[256]	DMA転送元エリア2	main
static dma_handle_t	gb_dma_handles[2][16]	DMA全チャネルのハンドル情報 （2ユニット、16チャネル分）	R_DMA_Open R_DMA_Control R_DMA_Close

6.8 関数一覧

表 6.9 に DMA のサンプルプログラム、表 6.10 に DMA サンプルドライバ関数一覧を示します。

表 6.9 DMA サンプルプログラム関数一覧

関数名	ページ番号
main	27
dma_err_callback	28
dma_dmac0_soft_trg_isr	29

表 6.10 DMA サンプルドライバ関数一覧

関数名	ページ番号
R_DMA_Open	31
R_DMA_Control	34
R_DMA_Close	35
R_DMA_GetVersion	35
R_DMA_SoftTrgCommon	36

6.9 関数仕様

サンプルプログラム、サンプルドライバの関数仕様を示します。

6.9.1 main

main	
概　要	サンプルプログラムのメイン処理
ヘッダ	—
宣　言	int main(void)
説　明	転送元エリア src_data0、src_data1、src_data2（各 256byte）の内容を、DMA 機能を使用して転送先エリア des_data0、des_data1、des_data2（各 256byte）へコピーします。 〇主な処理内容 • 変数、エリアの初期化 • DMAC0 チャネル 0 のオープン（R_DMA_Open） - チャネル：DMAC0 チャネル 0 を指定 - DMAC 起動要因：DMAC0 ソフトウェアリクエストを指定 - DMA 転送完了割り込み定義 ・割り込みハンドラ定義 ・ベクタ番号設定 ・割り込み優先レベル設定 - DMA エラー割り込みのコールバック関数を登録 • DMA リンクディスクリプタの設定＆転送開始（R_DMA_Control） - 転送元アドレスの指定（src_data0、src_data1、src_data2） - 転送先アドレスの指定（des_data0、des_data1、des_data2） - 転送バイト数の指定（DMA_DATA_LENGTH） - 転送要求元の指定（CHCFG_REQD_SRC：転送元） - 転送要求の検出方法を指定（CHCFG_REQMTD_EDG_RIS：立ち上がりエッジ） - ACK モードを指定（CHCFG_AM_BUSMODE：バス・サイクル・モード） - 転送元のデータサイズを指定（CHCFG_SDS_8BIT：8 ビット） - 転送先のデータサイズを指定（CHCFG_DDS_8BIT：8 ビット） - 転送元アドレスのカウント方向を指定（CHCFG_SAD_INC：インクリメント） - 転送先アドレスのカウント方向を指定（CHCFG_DAD_INC：インクリメント） - 転送モードを指定（CHCFG_TM_BLK：ブロック転送モード） ライトオンリーモードの設定を指定（CHCFG_WONLY_INVALID：ライトオンリーモード無効） - ディスクリプタのリード間隔を指定（1） • DMA 転送完了、DMA エラー監視 DMA 転送完了、DMA エラー割り込みの監視を行い、どちらかが発生するまで監視を継続し、発生した場合はクローズ処理を実行 • DMAC0 チャネル 0 のクローズ
引　数	なし
リターン値	0

6.9.2 dma_err_callback

dma_err_callback	
概　要	DMA エラーのコールバック
ヘッダ	—
宣　言	static void dma_err_callback(void)
説　明	DMA エラーのハンドラから呼ばれるコールバック関数 メイン通知用のエラー発生フラグをセットしている。
引　数	なし
リターン値	なし

6.9.3 dma_dmac0_soft_trg_isr

dma_dmac0_soft_trg_isr	
概　要	DMAC0 のソフトウェア要求要因の割り込みハンドラ
ヘッダ	—
宣　言	static void dma_dmac0_soft_trg_isr(void)
説　明	DMAC0 のソフトウェア要求要因の割り込みハンドラ処理。 • 割り込みエッジ検出のクリア • ディスクリプタ実行回数のカウント • 割り込みシーケンスの終了 サンプルプログラムにてハンドラ処理を記述する必要があるため、以下の例 1、2 に従い記述して下さい。（各要求要因の詳細は表 6.12 を参照）

例 1）ソフトウェア要求要因（ベクタ番号 = 251）のハンドラ

```
#ifdef __ICCARM__
#pragma type_attribute=__irq __arm
#endif  /* __ICCARM__ */
static void dma_dmac0_soft_trg_isr(void)
{
    /* Clear interrupt edge detection */
    VIC.PIC7.BIT.PIC251 = 1;
    R_DMA_SoftTrgCommon(ICU_VEC_NUM_251);

    g_dma_cmp_dsc_cnt++;

    /* End interrupt sequence */
    VIC.HVA0.LONG = DMA_HVA_WRITE_DATA;
}
```

例 2）シリアル通信ユニット 2 要因（ベクタ番号 = 111）のハンドラ

```
#ifdef __ICCARM__
#pragma type_attribute=__irq __arm
#endif  /* __ICCARM__ */
static void dma_scif2_isr(void)
{
    /* Clear interrupt edge detection */
    VIC.PIC3.BIT.PIC111 = 1;

    g_dma_cmp_dsc_cnt++;

    /* End interrupt sequence */
    VIC.HVA0.LONG = DMA_HVA_WRITE_DATA;
}
```

注 . 記述中の 251、111 の番号は、割り込み要因のベクタ番号に合わせて変更してください。

注 . R_DMA_SoftTrgCommon 関数は、割り込み要因のベクタ番号が 251 or 252 の場合のみコールしてください。

引　数	なし
リターン値	なし

6.9.4　R_DMA_Open

R_DMA_Open	
概　要	DMA モジュールのオープン
ヘッダ	r_dma_rzt1_if.h
宣　言	dma_err_t R_DMA_Open(const dma_unit_channel_t unit_channel, const dma_vector_para_t * const vector_para, void (* const pcallback_err)(void), dma_handle_t * const phandle)
説　明	DMA の指定ユニット、チャネル（第 1 引数）に対して、DMA モジュールのオープン、DMA 転送完了割り込みハンドラ（第 2 引数）の設定、DMA エラー時のコールバック関数（第 3 引数）の登録を行い、オープンした情報をハンドル（第 4 引数）へ設定して返す。 なお、関数の実行結果をリターン値として返す。 〇主な処理内容 • 引数チェック - ユニット番号（第 1 引数）の範囲チェック - DMA 転送完了割り込みハンドラ（第 2 引数）のポインタ NULL チェック - ベクタ番号（第 2 引数）の範囲チェック - ベクタ番号（第 2 引数）の重複使用チェック - ハンドル（第 4 引数）のポインタ NULL チェック - チャネルのオープン状態チェック すでにチャネルがオープンされているとエラー （すでにオープンされているチャネルはオープンできない。再度オープンする場合は R_DMA_Close 関数にて一度クローズする必要あり ） • 情報設定 - オープン状態設定 - ユニット情報設定 - チャネル情報設定 - DMA 転送完了割り込みのベクタ番号、割り込み優先レベル、ハンドラ設定 - DMA エラーコールバック設定 - SKIP モードの設定（SKIP モード無効） • DMA ユニット 0/1 モジュールストップ状態の解除（消費電力低減機能 OFF） • DMA 機能設定 - CMNCR レジスタ設定 - DCTRL_(unit) レジスタ設定（固定優先順位モード） チャネル 0 ～ 7 内、チャネル 8 ～ 15 内は固定優先順位。 チャネル 0 ～ 7、チャネル 8 ～ 15 間はラウンドロビン。 - NXLA_(ch) レジスタ設定 （リンクディスクリプタ領域の先頭アドレスを設定）

- CHCFG_(ch) レジスタ設定
  - ・チャネルを設定
  - ・転送要求元（CHCFG_REQD_SRC：転送元）
  - ・転送要求の検出方法（CHCFG_REQMTD_EDG_RIS：立ち上がりエッジ）
  - ・ACK モード（CHCFG_AM_BUSMODE：バス・サイクル・モード）
  - ・ディスクリプタ再読み込み動作（CHCFG_DRRP_REP：ディスクリプタ・リード継続）
  - ・転送元のデータサイズ（CHCFG_SDS_8BIT：8 ビット）
  - ・転送先のデータサイズ（CHCFG_DDS_8BIT：8 ビット）
  - ・転送元アドレスのカウント方向（CHCFG_SAD_INC：インクリメント）
  - ・転送先アドレスのカウント方向（CHCFG_DAD_INC：インクリメント）
  - ・転送モード（CHCFG_TM_BLK：ブロック転送モード）
  - ・ライトオンリーモード（CHCFG_WONLY_INVALID：ライトオンリーモード無効）
  - ・転送完了割り込みマスク（CHCFG_DEM_NMSK：マスクしない）
  - ・ディスクリプタ割り込みマスク（CHCFG_DIM_NMSK：マスクしない）
  - ・バッファ掃出し設定（CHCFG_SBE_NOUT：バッファ掃出ししない）
  - ・Next レジスタ（CHCFG_RSEL_R0：Next0 Register Set を実行）
  - ・RSEL 反転（CHCFG_RSW_NML：DMA 転送完了後に RSEL 反転しない）
  - ・レジスタ・セット動作（CHCFG_REN_NEXE：続けて DMA 転送を実行しない）
  - ・DMA モード（CHCFG_DMS_LNK：リンクモード）
- CHITVL_(ch) レジスタ設定（DMA_CHITVL_INIT：0x00000000）
- SCNT_(ch) レジスタ設定（DMA_SCNT_INIT：0x00000000）
- SSKP_(ch) レジスタ設定（DMA_SSKP_INIT：0x00000000）
- DCNT_(ch) レジスタ設定（DMA_DCNT_INIT：0x00000000）
- DSKP_(ch) レジスタ設定（DMA_DSKP_INIT：0x00000000）

- ループ構成のディスクリプタ初期化
  - ディスクリプタヘッダ設定
    - ・STS ビットに 0 設定
    - ・D ビットに 0 設定（ディスクリプタ・エラーなし）
    - ・LV ビットに 0 設定（ディスクリプタ無効）
    - ・LE ビットに 0 設定（リンク継続）
    - ・WBD ビットに 0 設定（LV ビット 0 を書き戻す）
    - ・DSCFM ビットに 1 設定（ディスクリプタフォーマット 1）
  - ネクストリンクアドレス設定

    ループ構成の各ディスクリプタのネクストリンクアドレスにアドレスを設定

- ハンドル（第 4 引数）への情報設定
- ベクタ番号、割り込み優先レベル（第 2 引数）の ICU への登録および割り込み許可
- DMA エラー割り込みの ICU への登録および割り込み許可
- DMA 転送開始
  - DMA ソフトウェアリセット
  - DMA 転送許可
  - DMA(unit)SEL(ch) レジスタ設定
    DMA チャネル要因設定

注 . (unit) = 0/1、(ch) = 0 ～ 15

引　数	const dma_unit_channel_t unit_channel	：ユニット、チャネル指定 DMA のユニットを上位 4bit、チャネルを下位 4bit で指定。（図 6.3 の dma_unit_channel_t にて指定）
	const dma_vector_para_t * const vector_para	：ベクタパラメータ指定 DMA 転送完了割り込みのベクタ番号（表 6.12 参照）、割り込み優先レベル、ハンドラ関数を指定。
	void (*pcallback_err)(void)	：DMA エラーコールバック指定 DMA エラー割り込み発生時に実行するコールバック関数を指定する。コールバック関数未使用の場合は NULL を指定する。
	dma_handle_t * const phandle	：DMA のハンドルのポインタを指定 phandle のポインタ指定されたエリアへオープンした DMA のチャネル情報を返す。 phandle のポインタが指定するエリアは、R_DMA_Open を呼び出す処理にて確保が必要。
リターン値	DMA_SUCCESS	：成功
	DMA_ERR_BAD_UNIT	：失敗：ユニット番号が不正
	DMA_ERR_BAD_CHAN	：失敗：チャネル番号が不正
	DMA_ERR_CH_NOT_CLOSED	：失敗：クローズできない
	DMA_ERR_INVALID_ARG	：失敗：パラメータの内容が不正
	DMA_ERR_NULL_PTR	：失敗：パラメータが空（NULL）
補足	１つの DMA 転送完了割り込み要因（第 2 引数の vector_para にて指定するベクタ番号）に対して、複数の DMAC チャネル（第 1 引数）を割り当てることはできません。 ソフトウェアリクエストの場合も同様で、ソフトウェアリクエストによる転送完了割り込み要因は 2 つであるため、最大 2 チャネルまでの同時使用が可能です。 本 API にて重複使用をチェックしており、該当した場合は DMA_ERR_INVALID_ARG エラーを返します。	

6.9.5　R_DMA_Control

R_DMA_Control		
概　要	DMA モジュールのコマンド実行	
ヘッダ	r_dma_rzt1_if.h	
宣　言	dma_err_t R_DMA_Control(const dma_handle_t * const phandle, const uint8_t cmd, const void * const param)	
説　明	ハンドル（第 1 引数）により指定されたチャネルに対して、指定されたコマンド（第 2 引数）を実行し、パラメータ（第 3 引数）の設定を行う。関数の実行結果をリターン値として返す。 〇主な処理内容 • 引数チェック - 指定ハンドル（第 1 引数）のポインタ NULL チェック - 指定パラメータ（第 3 引数）のポインタ NULL チェック - チャネルのオープン状態チェック チャネルがオープンされていないとエラー （R_DMA_Control 関数は、事前に R_DMA_Open 関数にてチャネルをオープンした状態で実行する必要あり） • コマンド処理 第 2 引数により指定されたコマンドにより、各処理を実行する。 該当しないコマンドの場合は不明なコマンドとしてエラーを返す。 コマンド内容に関しては、表 6.11 を参照。	
引　数	const dma_handle_t * const phandle	: DMA のハンドルのポインタを指定 R_DMA_Open にてオープン済みのハンドルを指定する。
	const uint8_t cmd	: 実行するコマンドを指定 （詳細は表 6.11 を参照）
	void * const param	: パラメータのポインタを指定 （詳細は表 6.11、表 6.12 を参照）
リターン値	DMA_SUCCESS	: 成功
	DMA_ERR_CH_NOT_OPENED	: 失敗：オープンされていない
	DMA_ERR_UNKNOWN_CMD	: 失敗：不明なコマンド
	DMA_ERR_INVALID_ARG	: 失敗：パラメータの内容が不正
	DMA_ERR_NULL_PTR	: 失敗：パラメータが空（NULL）
	DMA_ERR_BUFF_FULL	: 失敗：ディスクリプタ設定バッファに空きがない
補足	本 API を複数同時に実行することは禁止です。 使用するユーザーにて排他で実行するようにして下さい。	

6.9.6 R_DMA_Close

R_DMA_Close		
概　要	DMA モジュールのクローズ	
ヘッダ	r_dma_rzt1_if.h	
宣　言	dma_err_t R_DMA_Close(const dma_handle_t * const phandle)	
説　明	ハンドル（第 1 引数）により指定されたチャネルをクローズする。 関数の実行結果をリターン値として返す。 〇主な処理内容 • 引数チェック - 指定ハンドル（第 1 引数）のポインタ NULL チェック - チャネルのオープン状態チェック チャネルがオープンされていないとエラー （R_DMA_Open 関数にてオープンされた状態のチャネルしかクローズできない） • 指定チャネル（第 1 引数）の割り込みを ICU にて禁止 • オープン状態の解除 • DMA ユニット 0/1 モジュールストップ状態への遷移（消費電力低減機能 ON） ユニット中のすべてのチャネルがクローズされた場合にのみ消費電流低減機能を ON する。ユニット中のチャネルが 1 つでもオープンされていれば消費電力機能は ON しない。	
引　数	const dma_handle_t * const phandle	：DMA のハンドルのポインタを指定 R_DMA_Open にてオープン済みのハンドルを指定する。
リターン値	DMA_SUCCESS DMA_ERR_CH_NOT_OPENED DMA_ERR_NULL_PTR	：成功 ：失敗：オープンされていない ：失敗：パラメータが空（NULL）

6.9.7 R_DMA_GetVersion

R_DMA_GetVersion	
概　要	DMA モジュールのバージョン情報取得
ヘッダ	r_dma_rzt1_if.h
宣　言	uint32_t R_DMA_GetVersion(void)
説　明	DMA モジュールのバージョン情報をリターン値として返す
引　数	なし
リターン値	DMA サンプルドライバのバージョン情報（32bit） 16-31bit：Major Version 0-15bit：Minor Version

6.9.8 R_DMA_SoftTrgCommon

R_DMA_SoftTrgCommon		
概　要	ソフトウェア要求要因を発行する	
ヘッダ	r_dma_rzt1_if.h	
宣　言	void R_DMA_SoftTrgCommon(const uint16_t vector_no)	
説　明	ソフトウェア要求要因（ベクタ番号 251 or 252）による DMA 転送完了後にコールすることで、転送データが存在する場合に DMA 転送要求を発行します。 注 . サンプルプログラムにて記述されるソフトウェア要求要因の割り込みハンドラからコールして下さい。	
引　数	const uint16_t vector_no	割り込み要因のベクタ番号 ICU_VEC_NUM_251、ICU_VEC_NUM_252 の何れかを指定してください。
リターン値	なし	

6.10 フローチャート

6.10.1 main 処理

図 6.4 に main 処理のフローチャートを示します。

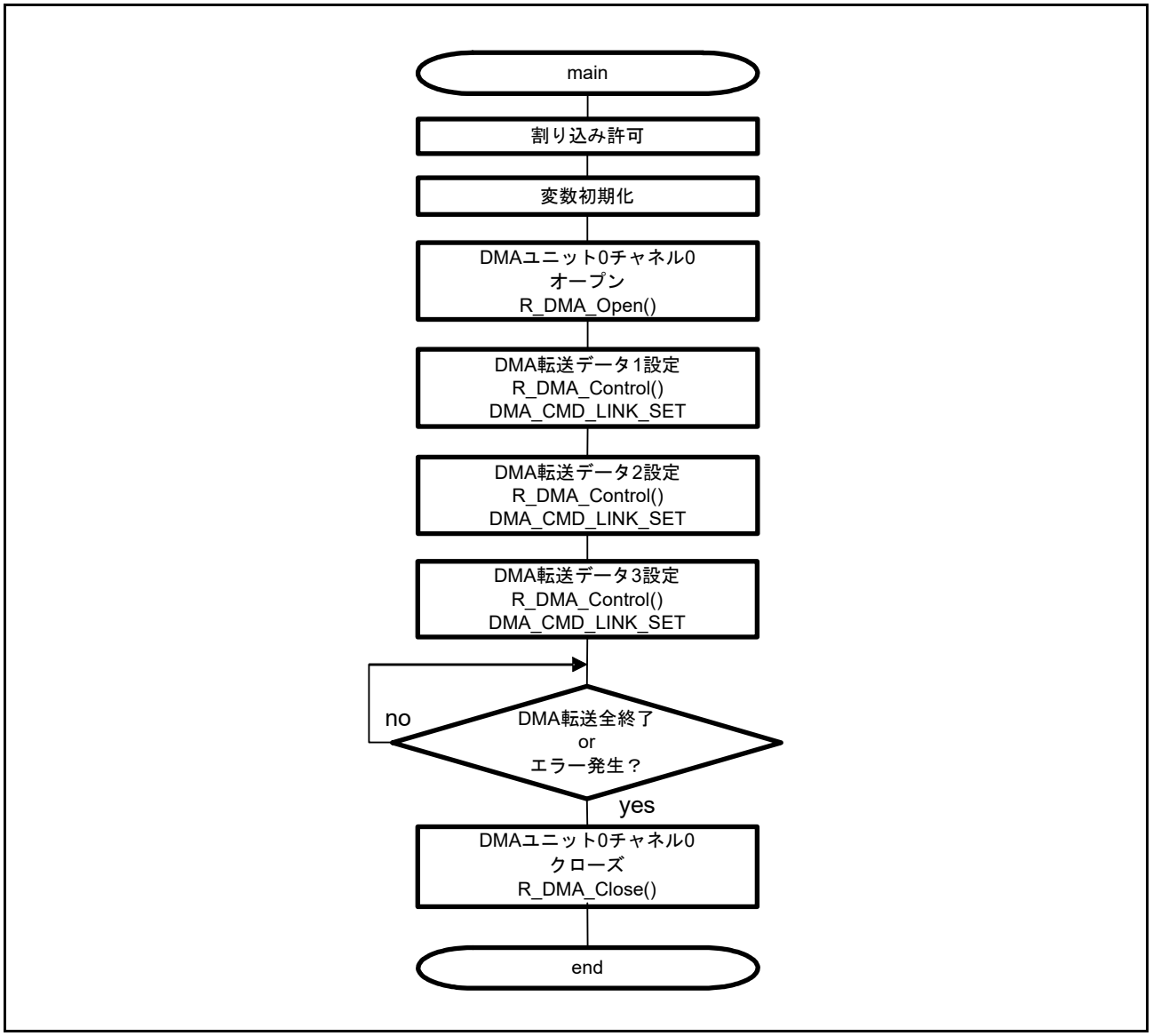


図 6.4 main 処理

6.10.2 dma_err_callback 処理

図 6.5 に dma_err_callback 処理のフローチャートを示します。

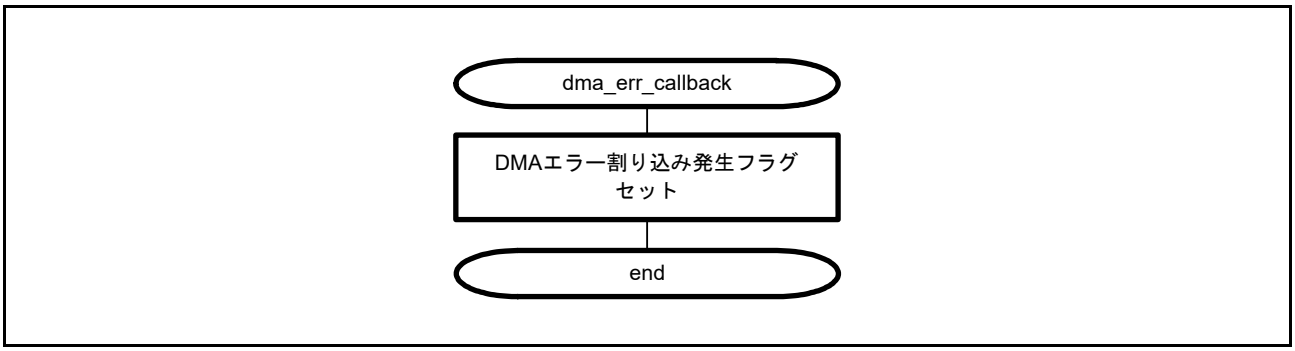


図 6.5 dma_err_callback 処理

6.10.3 dma_dmac0_soft_trg_isr 処理

図 6.6 に dma_dmac0_soft_trg_isr 処理のフローチャートを示します。

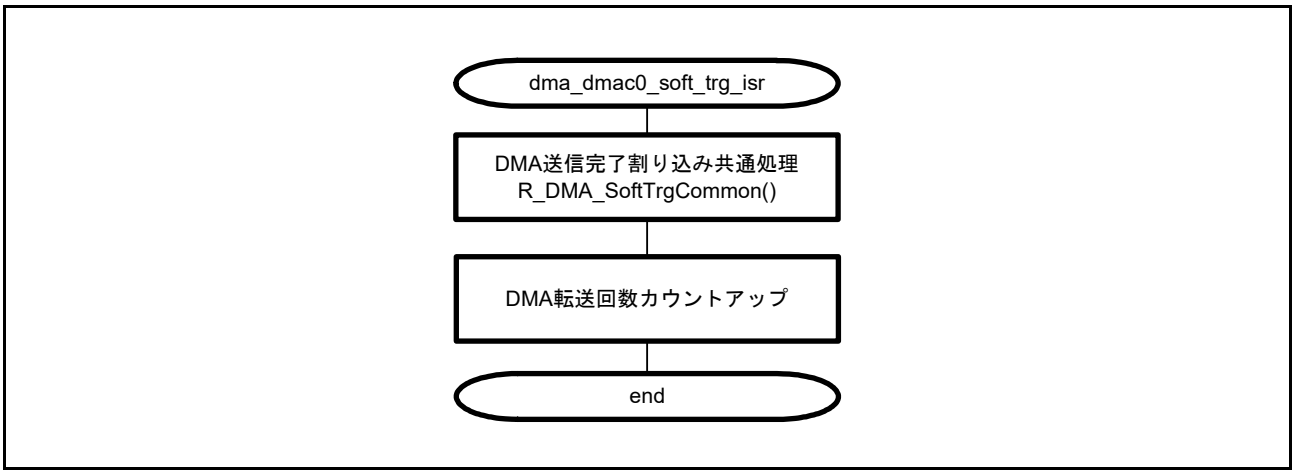


図 6.6 dma_dmac0_soft_trg_isr 処理

6.10.4 R_DMA_Open 処理

図 6.7 に R_DMA_Open 処理のフローチャートを示します。

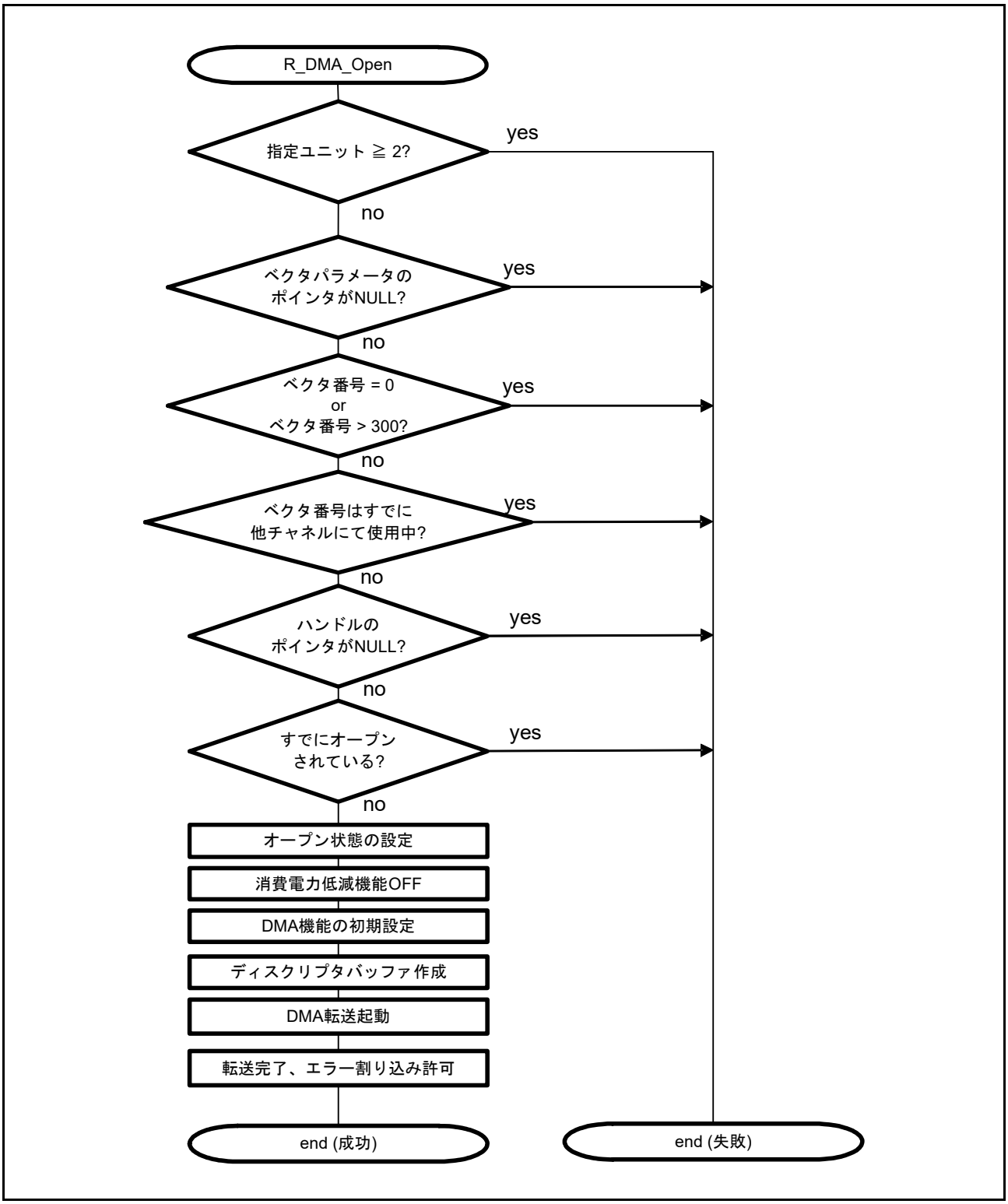


図 6.7 R_DMA_Open 処理

6.10.5 R_DMA_Control 処理

図 6.8、図 6.9、図 6.10 に R_DMA_Control 処理のフローチャートを示します。

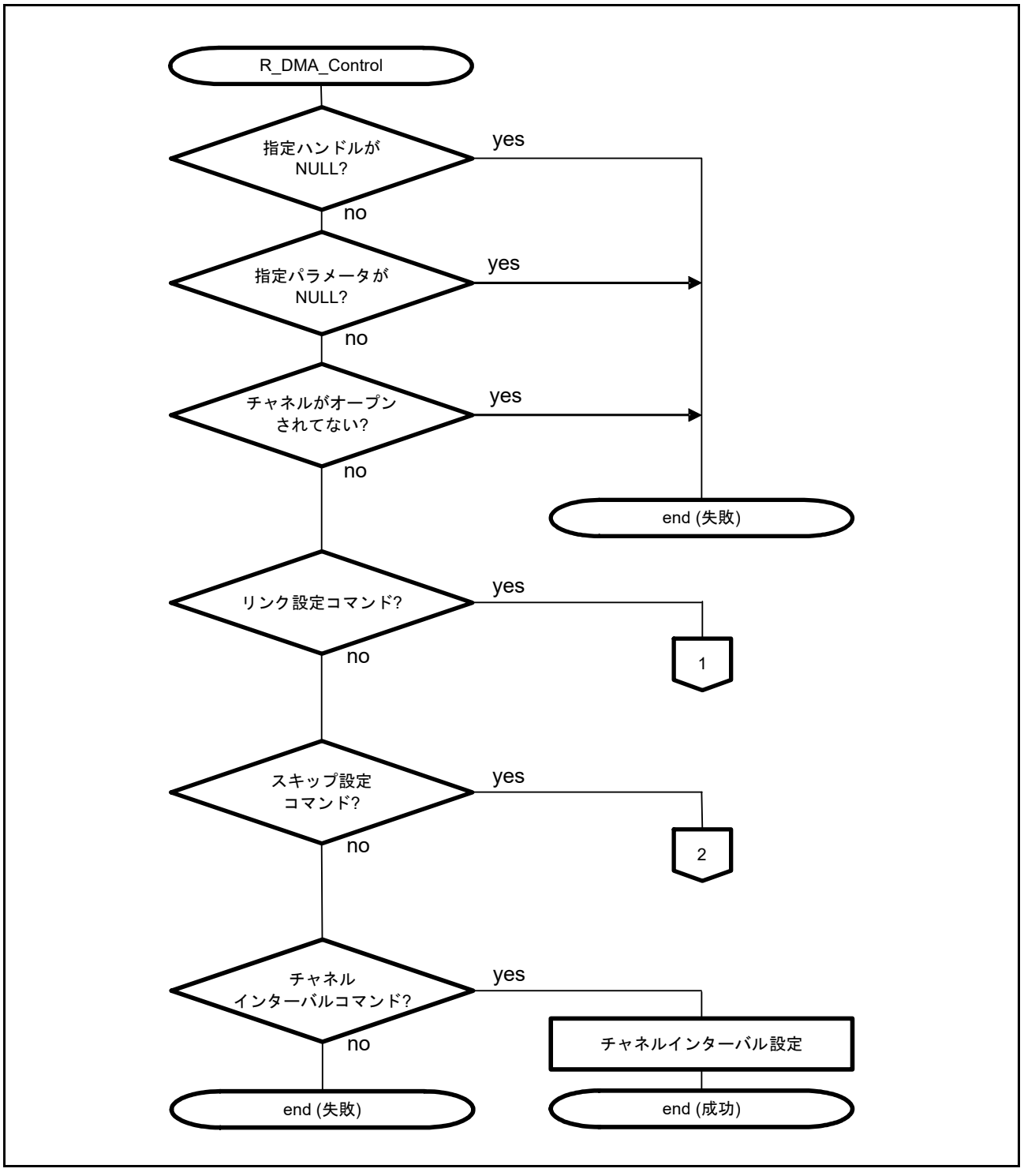


図 6.8 R_DMA_Control 処理

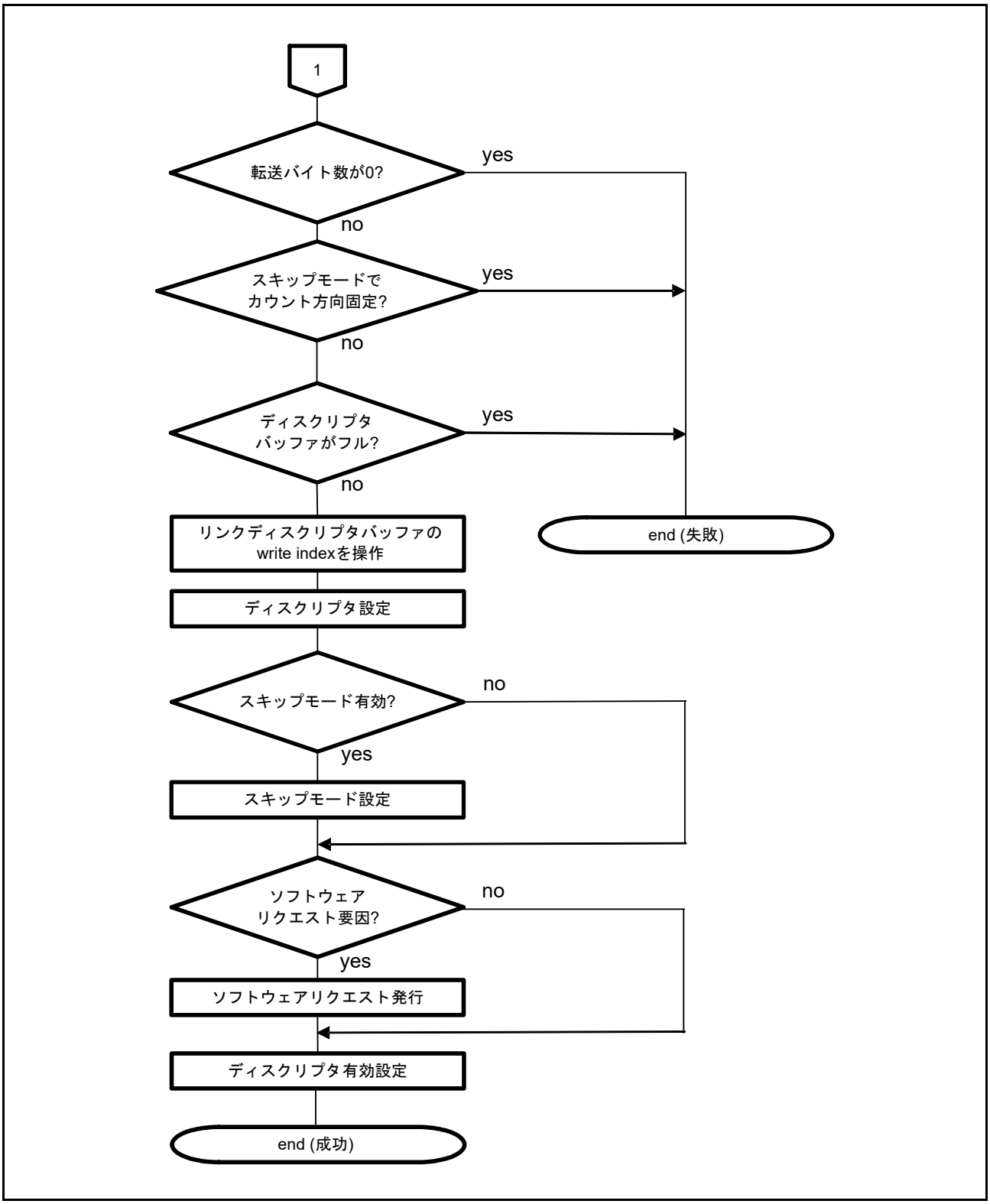


図 6.9 R_DMA_Control 処理

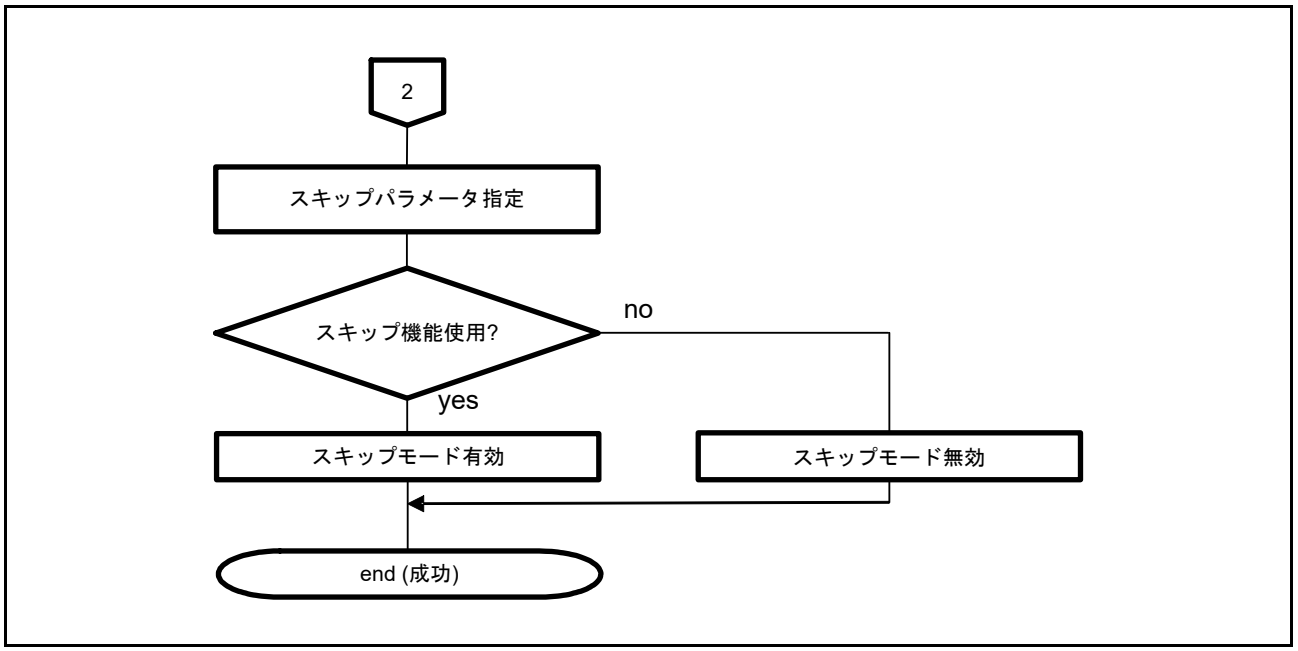


図 6.10　R_DMA_Control 処理

6.10.6 R_DMA_Close 処理

図 6.11 に R_DMA_Close 処理のフローチャートを示します。

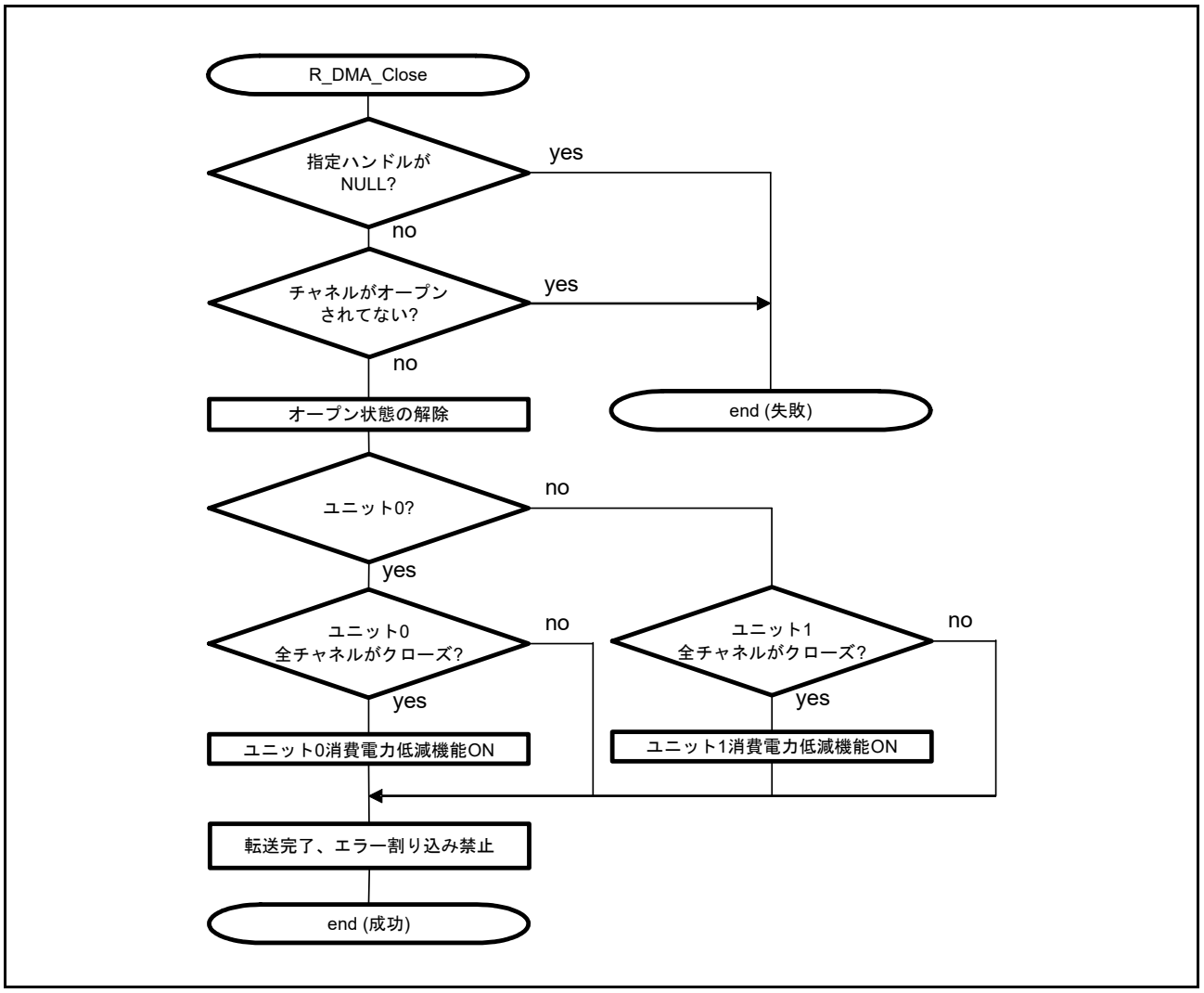


図 6.11 R_DMA_Close 処理

6.10.7 R_DMA_GetVersion 処理

図 6.12 に R_DMA_GetVersion 処理のフローチャートを示します。

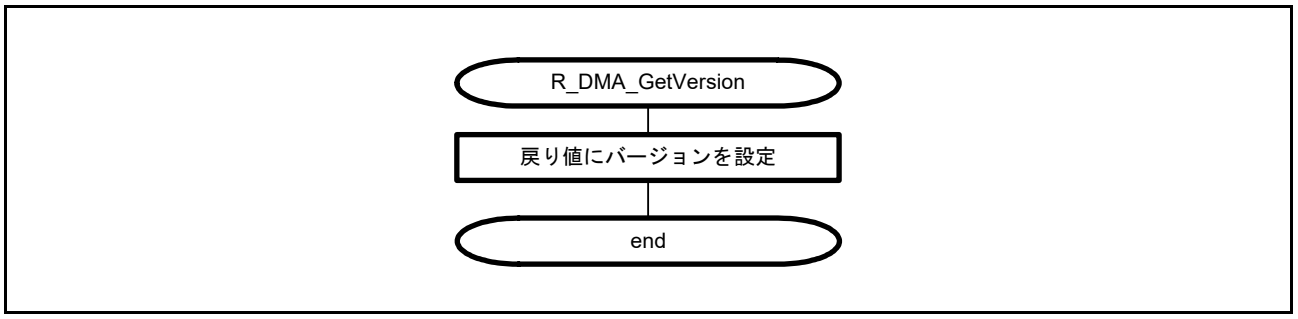


図 6.12 R_DMA_GetVersion 処理

6.10.8 R_DMA_SoftTrgCommon 処理

図 6.13 に R_DMA_SoftTrgCommon 処理のフローチャートを示します。

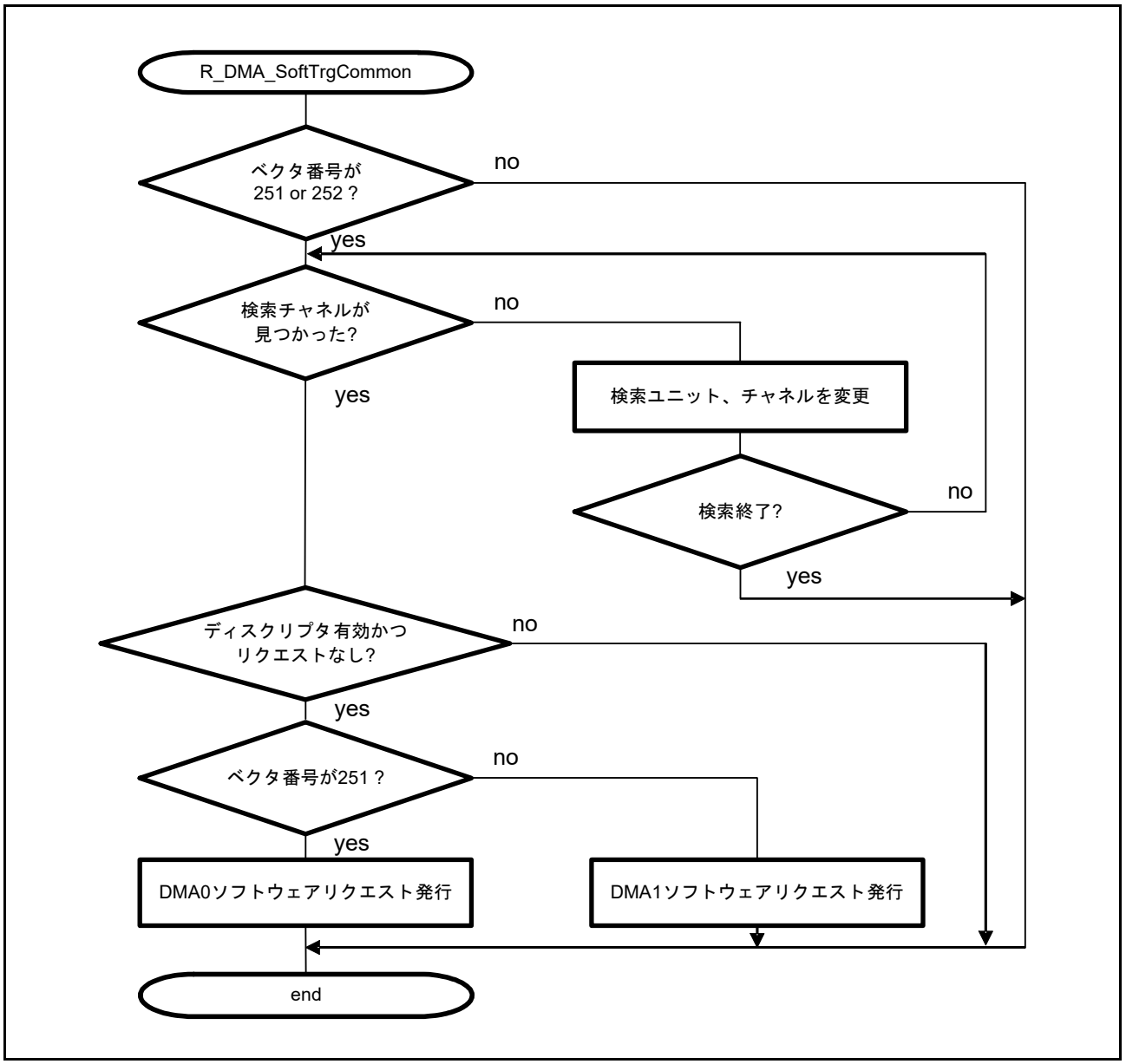


図 6.13 R_DMA_SoftTrgCommon 処理

6.11 R_DMA_Control コマンド一覧

R_DMA_Control 関数で使用するコマンド一覧を以下に示します。

表6.11 R_DMA_Controlコマンド一覧

コマンド名	内容
DMA_CMD_LINK_SET	リンクディスクリプタ（転送パラメータ）の設定を行い、DMA転送を開始します。
DMA_CMD_SKIP_SET	スキップ機能のパラメータを設定します。
DMA_CMD_DSCITVL_SET	ディスクリプタのリード間隔のパラメータを設定します。

6.11.1　DMA_CMD_LINK_SET

DMA_CMD_LINK_SET		
概　要	リンクディスクリプタの設定を行い、DMA 転送を開始します。	
ヘッダ	r_dma_rzt1_if.h	
説　明	第 1 引数のハンドルにて指定されるユニット、チャネルに対して、第 3 引数にて指定されるリンクディスクリプタ設定を行い、DMA 転送を開始します。	
パラメータ	dma_control_link_para_t	第 3 引数のパラメータ構造体
	uint32_t sa	転送元のアドレスを指定します。 注．　ライトオンリーモード (wonly = CHCFG_WONLY_VALID) を指定した場合は、ここには転送するデータパターンを指定して下さい。データサイズが 16/8 ビットの場合は下位から 16・8 ビットが有効とります。
	uint32_t da	転送先のアドレスを指定します。
	uint32_t tb	転送バイト数を指定します。 注．　0 は指定しないで下さい。（エラーを返します）
	chcfg_reqd_t　reqd	転送要求元を指定します。 : CHCFG_REQD_SRC：転送元を指定します。 : CHCFG_REQD_DES：転送先を指定します。
	chcfg_reqmtd_t reqmtd	転送要求の検出方法を指定します。 : CHCFG_REQMTD_INVALID：検出無効 : CHCFG_REQMTD_EDG_FAL：立ち下りエッジ : CHCFG_REQMTD_EDG_RIS：立ち上がりエッジ : CHCFG_REQMTD_LVL_LOW：ロウレベル : CHCFG_REQMTD_LVL_HI：ハイレベル 注．　転送要因毎に設定が決まっている場合があります。（表 6.12 参照）
	chcfg_am_t　am	ACK モードを指定します。 : CHCFG_AM_LVLMODE：レベルモード : CHCFG_AM_BUSMODE：バス・サイクル・モード : CHCFG_AM_MUSK：ACK をマスク 注．　転送要因毎に設定が決まっています。（表 6.12 参照）
	chcfg_sds_t　sds	転送元のデータサイズを指定します。 : CHCFG_SDS_8BIT：8 ビット : CHCFG_SDS_16BIT：16 ビット : CHCFG_SDS_32BIT：32 ビット : CHCFG_SDS_128BIT：128 ビット : CHCFG_SDS_256BIT：256 ビット : CHCFG_SDS_512BIT：512 ビット

	chcfg_dds_t　dds	転送先のデータサイズを指定します。 ：CHCFG_DDS_8BIT：8 ビット ：CHCFG_DDS_16BIT：16 ビット ：CHCFG_DDS_32BIT：32 ビット ：CHCFG_DDS_128BIT：128 ビット ：CHCFG_DDS_256BIT：256 ビット ：CHCFG_DDS_512BIT：512 ビット
	chcfg_sad_t　sad	転送元アドレスのカウント方向を指定します。 ：CHCFG_SAD_INC：インクリメント ：CHCFG_SAD_FIX：固定 注．SKIP モードを使用する場合、固定は指定しないで下さい。（エラーを返します） SKIP モードの使用は、DMA_CMD_SKIP_SET コマンドにて選択が可能です。
	chcfg_dad_t　dad	転送先アドレスのカウント方向を指定します。 ：CHCFG_DAD_INC：インクリメント ：CHCFG_DAD_FIX：固定 注．SKIP モードを使用する場合、固定は指定しないで下さい。（エラーを返します） SKIP モードの使用は、DMA_CMD_SKIP_SET コマンドにて選択が可能です。
	chcfg_tm_t　tm	転送モードを指定します。 ：CHCFG_TM_SIN：シングル転送モード ：CHCFG_TM_BLK：ブロック転送モード
	chcfg_wonly_t　wonly	ライトオンリーモードを指定します。 ：CHCFG_WONLY_INVALID：ライトオンリーモード無効 ：CHCFG_WONLY_VALID：ライトオンリーモード有効 注．ライトオンリーモード有効とした場合は、転送するデータパターンを本構造体の sa で指定して下さい。
	uint16_t chitvl	DMA の転送間隔を設定します。 reqd で指定された側の転送 1 回毎にインターバルを挿入し、バスを解放します。 転送間隔は、設定値 × 6.67ns です。 注．設定可能な値は 0 ～ 0xFFFF 注．6.67ns ≒ 1/ICLK (ICLK = 150MHz)
リターン値	DMA_SUCCESS	：成功
	DMA_ERR_NULL_PTR	：失敗：パラメータが指定されていない
	DMA_ERR_CH_NOT_OPENED	：失敗：オープンされていない
	DMA_ERR_BUFF_FULL	：失敗：ディスクリプタ設定バッファに空きがない
	DMA_ERR_INVALID_ARG	：失敗：パラメータの内容が不正

補足	本コマンドは、サンプルドライバ内部にて確保しているリンクバッファエリアに対し、コマンド実行毎に 1 つのリンクディスクリプタエリアを確保します。 確保されたディスクリプタエリアは、ディスクリプタの実行が完了すると解放されますが、完了するまでエリアを確保し続けます。 そのため、未完了のディスクリプタでリンクバッファエリアがいっぱいになり、新規ディスクリプタエリアの確保ができない場合、DMA_ERR_BUFF_FULL エラーを返します。 リンクバッファエリアのサイズ（DMA_LINK_BUF_NUM）を、使用する条件に合わせて変更して下さい。 DMA_LINK_BUF_NUM の値を 1 つ増やす毎に、1024byte の内蔵 RAM を消費します。 （ディスクリプタサイズ 32byte × 16 チャネル × 2 ユニット）

6.11.2　DMA_CMD_SKIP_SET

DMA_CMD_SKIP_SET		
概　要	DMA 転送のスキップ機能のパラメータを設定します。	
ヘッダ	r_dma_rzt1_if.h	
説　明	第 1 引数のハンドルにて指定されるユニット、チャネルに対して、第 3 引数にて指定されるスキップ機能のパラメータ設定を行います。 scnt/dcnt にて指定した連続転送する空間サイズ分を転送後、sskp/dskp にて指定したスキップする空間サイズ分をスキップして転送します。 本コマンドにて各パラメータ値に 0 以外を設定すると、以降の DMA_CMD_LINK_SET コマンドにてスキップ機能が有効となります。 また、全パラメータ値に 0 を設定すると、以降の DMA_CMD_LINK_SET コマンドにてスキップ機能が無効となります。	
パラメータ	dma_control_skip_para_t	第 3 引数のパラメータ構造体
	uint32_t scnt	転送元のアドレスにて、連続転送する空間サイズを指定します。
	uint32_t sskp	転送元のアドレスにて、スキップする空間サイズを指定します。
	uint32_t dcnt	転送先のアドレスにて、連続転送する空間サイズを指定します。
	uint32_t dskp	転送先のアドレスにて、スキップする空間サイズを指定します。
リターン値	なし	
補足	—	

6.11.3 DMA_CMD_DSCITVL_SET

DMA_CMD_DSCITVL_SET		
概　要	ディスクリプタのリード間隔を設定します。	
ヘッダ	r_dma_rzt1_if.h	
説　明	第 1 引数のハンドルにて指定されるユニット、チャネルに対して、第 3 引数にて指定されるディスクリプタのリード間隔の設定を行います。 ディスクリプタのリード間隔の設定は以下の 4 グループのチャネル群で共有しています。本コマンドで設定を行った場合、同じグループに属するすべてのチャネルのディスクリプタのリード間隔が変更になります。 • ユニット 0、チャネル 0 ～ 7 • ユニット 0、チャネル 8 ～ 15 • ユニット 1、チャネル 0 ～ 7 • ユニット 1、チャネル 8 ～ 15	
パラメータ	dma_control_dscitvl_para_t	第 3 引数のパラメータ構造体
	uint8_t dscitvl	ディスクリプタのリード間隔を指定します。 設定値 × 256 × 6.67ns の間隔でディスクリプタの再リードを行います。 注 . 設定可能な値は 0 ～ 0xFF 注 . 6.67ns ≒ 1/ICLK (ICLK = 150MHz)
リターン値	なし	
補足	—	

表6.12　DMA転送要求の検出動作指定（1 / 5）

転送要求元	要求要因	ベクタ番号	am設定値	reqmtd設定値
外部リクエスト	DREQ0	248	CHCFG_AM_LVLMODE： レベルモード CHCFG_AM_BUSMODE： バス・サイクル・モード CHCFG_AM_MUSK： DACK/TEND出力マスク 注．　DMA転送要求元の仕様に合わせて設定してください。	CHCFG_REQMTD_EDG_FAL： 立ち下がりエッジ検出 CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出 CHCFG_REQMTD_LVL_LOW： ロウレベル検出 CHCFG_REQMTD_LVL_HI： ハイレベル検出
	DREQ1	249		
	DREQ2	250		
外部割り込み	IRQ0	4	CHCFG_AM_BUSMODE： バス・サイクル・モード	CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出 CHCFG_REQMTD_LVL_HI： ハイレベル検出 注．　ここでは、レベルorエッジの指定のみを行うため、レベルの場合はCHCFG_REQMTD_EDG_RISを、エッジの場合はCHCFG_REQMTD_LVL_HIを設定して下さい。実際のH/Lレベルは IRQCR(ch = 1～15)レジスタにより決まります。
	IRQ1	5		
	IRQ2	6		
	IRQ3	7		
	IRQ4	8		
	IRQ5	9		
	IRQ6	10		
	IRQ7	11		
	IRQ8	12		
	IRQ9	13		
	IRQ10	14		
	IRQ11	15		
	IRQ12	16		
	IRQ13	17		
	IRQ14	18		
	IRQ15	19		
CMT Unit0	コンペアマッチ0	21		CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出
	コンペアマッチ1	22		
CMT Unit1	コンペアマッチ0	23		
	コンペアマッチ1	24		
CMTW Unit0	コンペアマッチ	25		
	インプットキャプチャ 0	26		
	インプットキャプチャ 1	27		
	アウトプットキャプチャ 0	28		
	アウトプットキャプチャ 1	29		
CMTW Unit1	コンペアマッチ	30		
	インプットキャプチャ 0	31		
	インプットキャプチャ 1	32		
	アウトプットキャプチャ 0	33		
	アウトプットキャプチャ 1	34		

表6.12 DMA転送要求の検出動作指定（２／５）

転送要求元	要求要因	ベクタ番号	am設定値	reqmtd設定値
S12ADC Unit0	AD変換完了	35	CHCFG_AM_BUSMODE： バス・サイクル・モード	CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出
	グループB変換完了	36		
S12ADC Unit1	AD変換完了	38		
	グループB変換完了	39		
DMAC0	DMAC0ソフトウェアトリガ	251		
DMAC1	DMAC1ソフトウェアトリガ	252		
USB	FuncDMA要求1	43		
	FuncDMA要求1	44		
Ether Switch with IEEE1588	Ether SWITCH割り込み	45		
	Ether SWITCH DLR割り込み	46		
	Ether SWITCH SYNCOUT割り込み	47		
Ether PHY	Ether PHY割り込み0	48		
	Ether PHY割り込み1	49		
	Ether PHY割り込み2	50		
Ether MAC	Ether MAC DMA受信完了	51		
	Ether MAC DMA送信完了	52		
	受信フレーム正常割り込み	53		
MTU3a	TGI7A	59		
	TGI7B	60		
	TGI7C	61		
	TGI7D	62		
	TCIV7	63		
Ether MAC	Ether MIIマネージメント・アクセス完了割り込み	64		
	Etherポーズ・パケット送信完了割り込み	65		
	Ether送信完了割り込み	66		
Ether CAT slave（R-IN Engine搭載製品のみ）	Sync0割り込み	73		
	Sync1割り込み	74		
	Ether CAT割り込み	75		
	SOF割り込み	76		
	EOF割り込み	77		

表6.12　DMA転送要求の検出動作指定（3 / 5）

転送要求元	要求要因	ベクタ番号	am設定値	reqmtd設定値
RSPI Unit0	受信バッファフル	80	CHCFG_AM_BUSMODE： バス・サイクル・モード	CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出
	送信バッファエンプティ	81		
RSPI Unit1	受信バッファフル	84		
	送信バッファエンプティ	85		
RSPI Unit2	受信バッファフル	88		
	送信バッファエンプティ	89		
RSPI Unit3	受信バッファフル	92		
	送信バッファエンプティ	93		
SCIFA Unit0	受信バッファフル	97		CHCFG_REQMTD_LVL_HI： ハイレベル検出
	送信バッファエンプティ	98		
SCIFA Unit1	受信バッファフル	101		
	送信バッファエンプティ	102		
SCIFA Unit2	受信バッファフル	110		
	送信バッファエンプティ	111		
SCIFA Unit3	受信バッファフル	114		
	送信バッファエンプティ	115		
SCIFA Unit4	受信バッファフル	118		
	送信バッファエンプティ	119		
RIIC Unit0	データ受信終了	122		CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出
	送信データエンプティ	123		
RIIC Unit1	データ受信終了	125		
	送信データエンプティ	126		
MTU3a	TGIA0	145		
	TGIB0	146		
	TGIC0	147		
	TGID0	148		
	TGIA1	152		
	TGIB1	153		
	TGIA2	156		
	TGIB2	157		
	TGIA3	160		
	TGIB3	161		
	TGIC3	162		
	TGID3	163		
	TGIA4	165		
	TGIB4	166		
	TGIC4	167		
	TGID4	168		

表6.12 DMA転送要求の検出動作指定（4 / 5）

転送要求元	要求要因	ベクタ番号	am設定値	reqmtd設定値
MTU3a	TGIV4	169	CHCFG_AM_BUSMODE： バス・サイクル・モード	CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出
	TGIU5	170		
	TGIV5	171		
	TGIW5	172		
	TGIA8	173		
	TGIB8	174		
	TGIC8	175		
	TGID8	176		
GPT	GTCIA0	178		
	GTCIB0	179		
	GTCIC0	180		
	GTCID0	181		
	GTCIE0	182		
	GTCIF0	183		
	GDTE0	184		
	CTCIV0	185		
	GTCIU0	186		
	GTCIA1	187		
	GTCIB1	188		
	GTCIC1	189		
	GTCID1	190		
	GTCIE1	191		
	GTCIF1	192		
	CDTE1	193		
	GTCIV1	194		
	GTCIU1	195		
	GTCIA2	196		
	GTCIB2	197		
	GTCIC2	198		
	GTCID2	199		
	GTCIE2	200		
	GTCIF2	201		
	CDTE2	202		
	GTCIV2	203		
	GTCIU2	204		
	GTCIA3	205		
	GTCIB3	206		
	GTCIC3	207		
	GTCID3	208		
	GTCIE3	209		
	GTCIF3	210		
	CDTE3	211		
	GTCIV3	212		
	GTCIU3	213		
	ETGIN	214		

表6.12 DMA転送要求の検出動作指定（５／５）

転送要求元	要求要因	ベクタ番号	am設定値	reqmtd設定値
GPT	ETGIP	215	CHCFG_AM_BUSMODE： バス・サイクル・モード	CHCFG_REQMTD_EDG_RIS： 立ち上がりエッジ検出
TPUa Unit0	TGI0A	216		
	TGI0B	217		
	TGI1A	221		
	TGI1B	222		
	TGI2A	225		
	TGI2B	226		
	TGI3A	229		
	TGI3B	230		
	TGI4A	234		
	TGI4B	235		
	TGI5A	238		
	TGI5B	239		
ELC	割り込み要求0	242		
	割り込み要求1	243		

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
  RZ/T1 グループ ユーザーズマニュアル ハードウェア編
  （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。）

  RZ/T1 Evaluation Board RTK7910022C00000BR ユーザーズマニュアル
  （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
  IAR 統合開発環境（IAR Embedded Workbench® for Arm）に関しては、IAR ホームページから入手してください。
  （最新版を IAR 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	DMAサンプルプログラム アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
0.10	2015.03.06	—	初版発行
1.00	2015.04.10	—	Web掲載に際しRevのみ変更
1.10	2015.08.06	2. 動作環境	
		5	表2.1 動作環境　統合開発環境　表記一部修正、追加
		6. ソフトウェア説明	
		10	6.2.4 説明文 参照を追加
		10	表6.2 タイトル、サイズを一部修正
		11	表6.3 追加
		11	表6.4 追加
		—	ディスクリプタインターバルとチャネルインターバルの設定パラメータが逆転していた不具合を修正するため、以下を変更
		13	表6.7 DMAサンプルプログラム定数 DMA_CHITVL_VALマクロ追加 DMA_DSCITVL_VALの値を変更(1->0)
		16 ～ 18	図6.2　構造体/共用体 dma_link_fmt_tのメンバdscitvlをchitvlに変更 dma_control_link_para_tのメンバdscitvlをchitvlに変更 dma_control_chitvl_para_tをdma_control_dscitvl_para_tに変更し、メンバのchitvlをdscitvlに変更
		21	図6.3　列挙型 dma_cmd_tのメンバDMA_CMD_CHITVL_SETをDMA_CMD_DSCITVL_SETに変更
		48	表6.14　R_DMA_Controlコマンド一覧 DMA_CMD_CHITVL_SETをDMA_CMD_DSCITVL_SETに変更
		50	6.11.1 DMA_CMD_LINK_SET パラメータdscitvlをchitvlに変更し、説明を差し替え
		52	6.11.3 DMA_CMD_DSCITVL_SET DMA_CMD_CHITVL_SETをDMA_CMD_DSCITVL_SETに置き換えたため、内容を変更
1.20	2015.12.03	2. 動作環境	
		5	表2.1 動作環境　統合開発環境　一部修正
1.30	2017.04.05	2. 動作環境	
		5	表2.1 動作環境　統合開発環境の内容変更
		6. ソフトウェア説明	
		—	6.2.4 必要メモリサイズ　削除
1.40	2018.06.07	2.動作環境	
		5	表2.1 動作環境　統合開発環境の内容変更
		8. 参考ドキュメント	
		56	IAR 統合開発環境名変更

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、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、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。
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。
6. 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。
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、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。
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。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. 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。
10. 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。
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。
12. 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.4.0-1 2017.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

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

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/