RZ/A1H グループ

ディスプレイアウトコンペアユニット（DISCOM）サンプルプログラム

R01AN1825JJ0100
Rev.1.00
2014.05.27

要旨

本アプリケーションノートでは、RZ/A1H のディスプレイアウトコンペアユニット（以下 DISCOM）を使用し、グラフィックス表示モジュール ビデオディスプレイコントローラ 5(以下 VDC5)から出力されているデータが、期待するグラフィックスデータであるかチェックするサンプルプログラムになります。

ディスプレイアウトコンペアユニットサンプルプログラムの特長を以下に示します。

・グラフィックス表示モジュールのグラフィックスプレーンはレイヤ 2 を使用して CRC コード比較。
・DISCOM に設定し比較する CRC はソフトウェアで計算(DISCOM を使用しても計算可能)。
・ピクセルフォーマットは、16bit/pixel(RGB565)を使用。
・比較対象となる矩形領域については、画面領域全体を指定。
・比較結果が不一致の場合、割り込みが発生。
・画面を一定間隔で変更(例 赤→青→白等)しながら DISCOM で CRC の比較を行い結果をシリアル出力。

対象デバイス

RZ/A1H

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 3

2. 動作確認条件 4

3. 関連アプリケーションノート 5

4. 周辺機能説明 5

5. ハードウェア説明 6
5.1 ハードウェア構成例 6

6. ソフトウェア説明 7
6.1 動作概要 7
6.1.1 シリアル出力 7
6.2 メモリマップ 8
6.2.1 サンプルコードのセクション配置 9
6.2.2 MMU の設定 12
6.3 割り込み 13
6.4 固定幅整数一覧 13
6.5 定数/エラーコード一覧 14
6.6 変数一覧 15
6.7 関数一覧 15
6.8 関数仕様 16
6.8.1 R_DISCOM_Initialize 16
6.8.2 R_DISCOM_Terminate 16
6.8.3 R_DISCOM_Configure 17
6.8.4 R_DISCOM_SetInterrupt 19
6.8.5 R_DISCOM_Start 19
6.8.6 R_DISCOM_Stop 19
6.8.7 R_DISCOM_GetCRC 20
6.8.8 DISCOMDRV_Init 20
6.8.9 DISCOMDRV _Term 20
6.8.10 DISCOMDRV _ConvChannel 21
6.8.11 DISCOMDRV _ConvGrTypeSelPlanel 22
6.8.12 DISCOMDRV _ConvPixRdFormat 23
6.8.13 DISCOMDRV _CalcCRC 24
6.9 フローチャート 25
6.9.1 メイン処理 25
6.9.2 DISCOM の CRC エラー検出処理 26

7. サンプルコード 27

8. 参考ドキュメント 27

1. 仕様

本アプリケーションノートのサンプルコードは、RZ/A1H に搭載された DISCOM を使用しグラフィックス表示モジュール VDC5 から出力されているデータが、期待するグラフィックスデータであるかチェックするサンプルプログラムになります。表 1.1 に使用する周辺機能と用途を以下に示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
DISCOM	グラフィックスデータの CRC コード比較
INTC (割り込み ID：CMP1 (129))	ディスプレイアウトコンペア相違検出
VDC5 (ch1) (レイヤ 2)	液晶表示

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	RZ/A1H
動作周波数（注）	CPU クロック（Iφ）: 400MHz 画像処理クロック（Gφ）: 266.67MHz 内部バスクロック（Bφ）: 133.33MHz 周辺クロック（P1φ）: 66.67MHz 周辺クロック（P0φ）: 33.33MHz
動作電圧	電源電圧（I/O）: 3.3V 電源電圧（内部）: 1.18V
統合開発環境	ARM®統合開発環境 ARM Development Studio 5（DS-5TM）Version 5.16
C コンパイラ	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
	コンパイラオプション --O3 -Ospace --cpu=Cortex-A9 --littleend --arm --apcs=/interwork --no_unaligned_access --fpu=vfpv3_fp16 -g --asm
動作モード	ブートモード 0 （CS0 空間 16 ビットブート）
使用ボード	GENMAI ボード ・R7S72100 CPU ボード RTK772100BC00000BR ・R7S72100 CPU ボード用オプションボード RTK7721000B00000BR
使用デバイス （ボード上で使用する機能）	Display Out (Analog RGB D-sub15) (オプションボード:J15)
	シリアルインタフェース(Dsub-9 コネクタの接続)

【注】 クロックモード 0（EXTAL 端子からの 13.33MHz のクロック入力）で使用時の動作周波数です。

3.　関連アプリケーションノート

本サンプルソフトウェア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/A1H グループ　初期設定例 (R01AN1864JJ)

RZ/A1H グループ　ビデオディスプレイコントローラ(VDC5)サンプルドライバ(R01AN1822JJ)

RZ/A1H グループ レジスタ定義ヘッダ・ファイル iodefine.h（R01AN1860JJ）

4.　周辺機能説明

DISCOM、VDC5 についての基本的な内容は、RZ/A1H グループ・ユーザーズマニュアル ハードウエ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

図 5.1 にハードウェア構成例を示します。

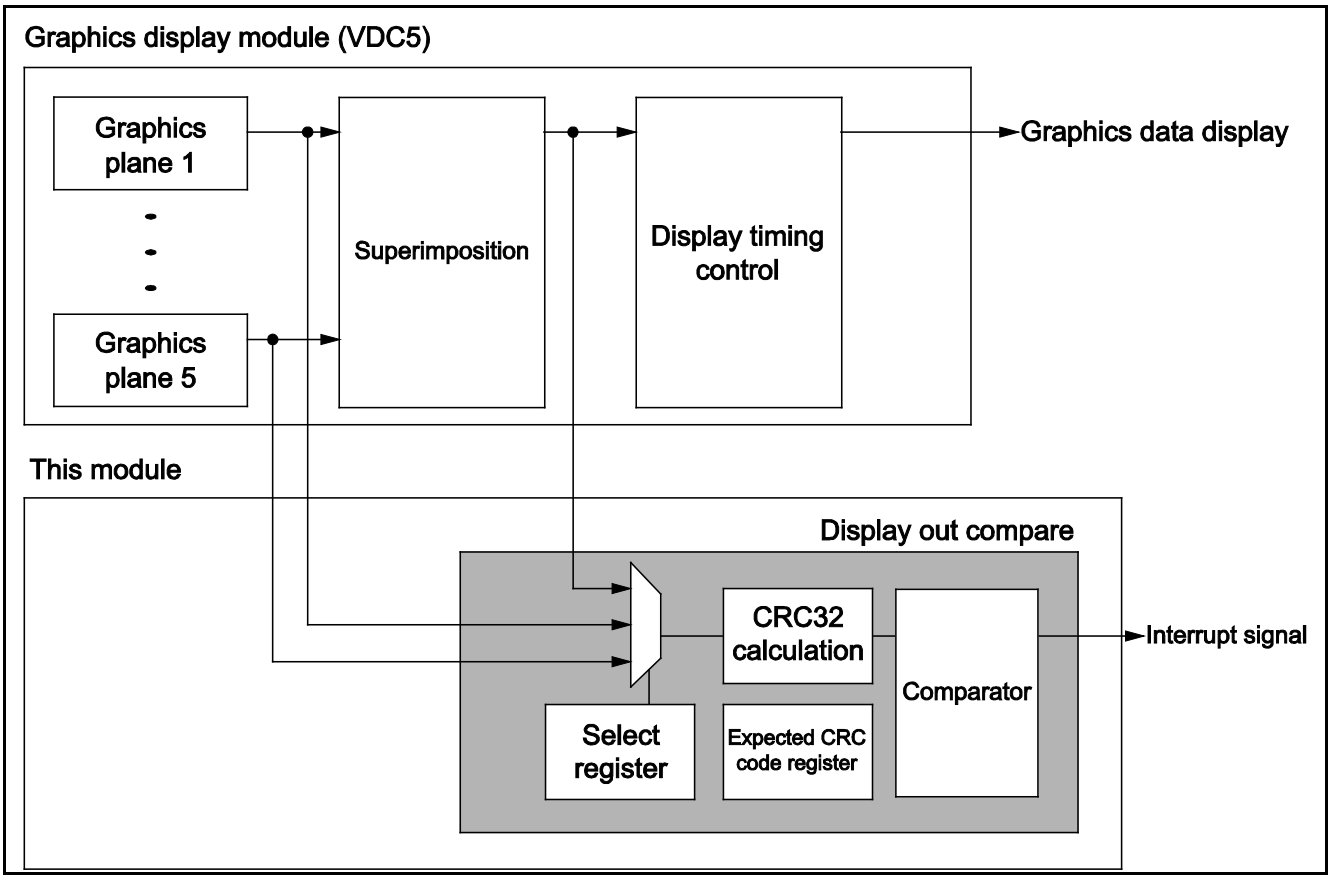

図 5.1　ハードウェア構成例

6. ソフトウェア説明

6.1 動作概要

本サンプルソフトウェアは、グラフィックス表示モジュール VDC5 から出力されているデータが、期待するグラフィックスデータであるか DISCOM を使用しチェックするサンプルプログラムになります。

VDC5 で表示する画面は、一定周期で変更し、その度、DISCOM を使用し比較を行っています。比較する画面の CRC 計算は、ソフトウェアで行い DISCOM に設定していますが、DISCOM を使用し CRC 計算を行うことも可能になります。比較結果が合わない場合、DISCOM のディスプレイアウトコンペア相違検出割り込みにて検出します。比較結果についてはシリアルを使用し出力しています。

6.1.1 シリアル出力

シリアルを使用した出力結果を記載します。括弧内は、CRC の値になります。

シリアルのビットレートは、115200bps になります。

```
RZ/A1H CPU Board Sample Program. Ver.1.00
Copyright (C) 2013 Renesas Electronics Corporation. All rights reserved.

_/_/_/_/  Aragon Discom Sample Ver.1.00  _/_/_/_/

        CRC OK (0x209C86E3)
        CRC OK (0x571E9393)
        CRC OK (0x0E34838C)
        CRC OK (0xC4F7C315)
        CRC OK (0x5278291D)
        CRC OK (0x98BB6984)
        CRC OK (0xC191799B)
        CRC OK (0x2B4702B4)
        CRC OK (0xD3E181F3)
        CRC OK (0x9DDDD30A)
        CRC OK (0xBDB85D8F)
        CRC OK (0x0B523902)
        CRC OK (0xA7833023)
```

図 6.1　DISCOM 出力結果

6.2　メモリマップ

図 6.2 に RZ/A1H グループのアドレス空間と GENMAI ボードのメモリマップを示します。

サンプルコードでは、ROM 領域を使用するコードおよびデータを CS0 空間に接続した NOR フラッシュメモリに配置し、RAM 領域を使用するコードおよびデータを大容量内蔵 RAM に配置するようにしています。

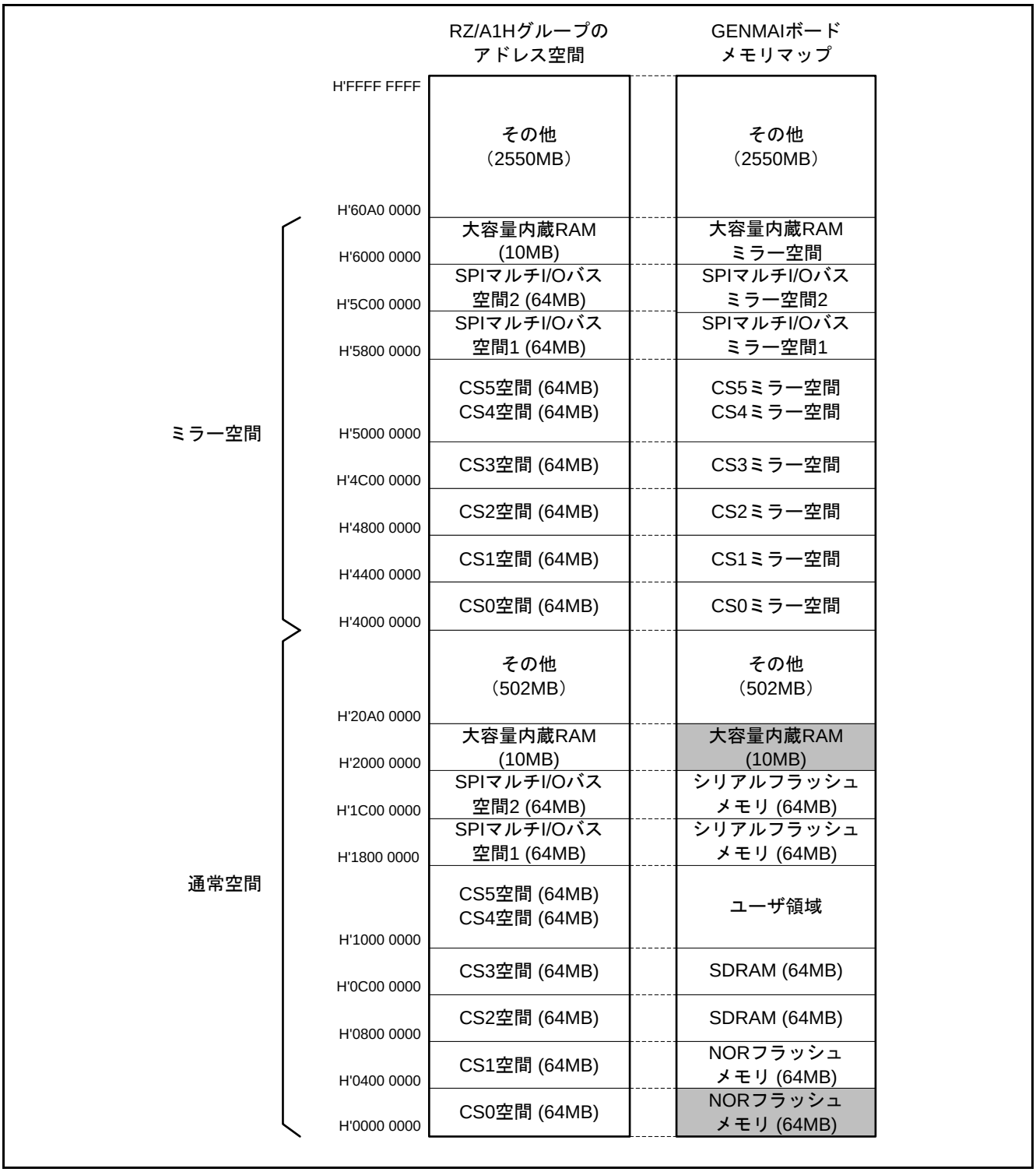

図 6.2　のメモリマップ

6.2.1　サンプルコードのセクション配置

サンプルコードでは、割り込み処理の高速化のため、例外処理ベクタテーブルと IRQ 割り込みハンドラを大容量内蔵 RAM 上に配置して、これらの処理を大容量内蔵 RAM 上で実行するようにしています。例外処理ベクタテーブルおよび IRQ 割り込みハンドラのプログラムコードの NOR フラッシュメモリ領域から大容量内蔵 RAM 領域への転送処理、初期値なしデータセクションのゼロクリア処理、および初期値ありデータセクションの初期化処理は、スキャッタローディング機能を使用して行っています。スキャッタローディング機能の詳細は、ARM より提供される「ARM コンパイラツールチェーン リンカの使用」の「イメージの構造と生成」を参照してください。

表 6.1 および表 6.2 にサンプルコードで使用するセクションを示し、図 6.3 にサンプルコードの初期状態のセクション配置（ロードビュー）と、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）を示します。

表 6.1　使用するセクション（1/2）

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_TABLE	例外処理ベクタテーブル	Code	FALSH	FLASH
RESET_HANDLER	リセットハンドラ処理のプログラムコード領域 　この領域は以下のセクションから構成されています ・INITCA9CACHE（L1 キャッシュ設定） ・INIT_TTB（MMU 設定） ・RESET_HANDLER（リセットハンドラ）	Code	FALSH	FLASH
CODE_BASIC_SETUP	動作周波数とフラッシュメモリ最適化のためのプログラムコード領域	Code	FLASH	FLASH
InRoot	この領域は C 標準ライブラリなどのルート領域に配置するセクションから構成されています	Code および RO Data	FLASH	FLASH
CODE_FPU_INIT	NEON および VFP 初期設定のプログラムコード領域。この領域は以下のセクションから構成されています ・CODE_FPU_INIT ・FPU_INIT	Code	FLASH	FLASH
CODE_RESET	ハードウェア初期設定のプログラムコード領域。この領域は以下のセクションから構成されています ・CODE_RESET（スタートアップ処理） ・INIT_VBAR（ベクタベース設定）	Code	FLASH	FLASH
CODE_IO_REGRW	IO レジスタのリード/ライト関数のプログラムコード領域	Code	FLASH	FLASH
CODE	デフォルトのプログラムコード領域 　C ソースでセクション名を定義しない Code タイプのセクションは、すべてこの領域に配置されます	Code	FLASH	FLASH
CONST	デフォルトの定数データ領域 　C ソースでセクション名を定義しない RO Data タイプのセクションは、すべてこの領域に配置されます	RO Data	FLASH	FLASH

表 6.2　使用するセクション（2/2）

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_MIRROR_TABLE	例外処理ベクタテーブル （大容量内蔵 RAM に転送して実行するためのセクション）	Code	FLASH	LRAM
CODE_HANDLER_JMPTBL	IRQ 割り込みハンドラのユーザ定義関数のプログラムコード領域	Code	FLASH	LRAM
CODE_HANDLER	IRQ 割り込みハンドラのプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_HANDLER ・IRQ_FIQ_HANDLER	Code	FLASH	LRAM
DATA_HANDLER_JMPTBL	IRQ 割り込みハンドラのユーザ定義関数の登録テーブルデータ領域	RW Data	FLASH	LRAM
ARM_LIB_STACK	アプリケーションスタック領域	ZI Data	―	LRAM
IRQ_STACK	IRQ モードのスタック領域	ZI Data	―	LRAM
FIQ_STACK	FIQ モードのスタック領域	ZI Data	―	LRAM
SVC_STACK	スーパバイザ（SVC）モードのスタック領域	ZI Data	―	LRAM
ABT_STACK	アボート（ABT）モードのスタック領域	ZI Data	―	LRAM
TTB	MMU 変換テーブル領域	ZI Data	―	LRAM
ARM_LIB_HEAP	アプリケーションヒープ領域	ZI Data	―	LRAM
DATA	デフォルトの初期値ありデータ領域 C ソースでセクション名を定義しない RW Data タイプのセクションは、すべてこの領域に配置されます	RW Data	FLASH	LRAM
BSS	デフォルトの初期値なしデータ領域 C ソースでセクション名を定義しない ZI Data タイプのセクションは、すべてこの領域に配置されます	ZI Data	―	LRAM
VRAM	表示バッファ	Data	―	LRAM

【注】 1. 表中のロード領域および実行領域において、FLASH は NOR フラッシュメモリの領域を、LRAM は大容量内蔵 RAM の領域を表します。
2. セクションの名前は基本的に領域と同じ名前にしていますが、RESET_HANDLER、InRoot、CODE_FPU_INIT、CODE_RESET、CODE、CONST、CODE_HANDLER、DATA、BSS の各領域は複数のセクションから構成されています。領域とセクションについては、ARM コンパイラツールチェーンのマニュアルを参照してください。

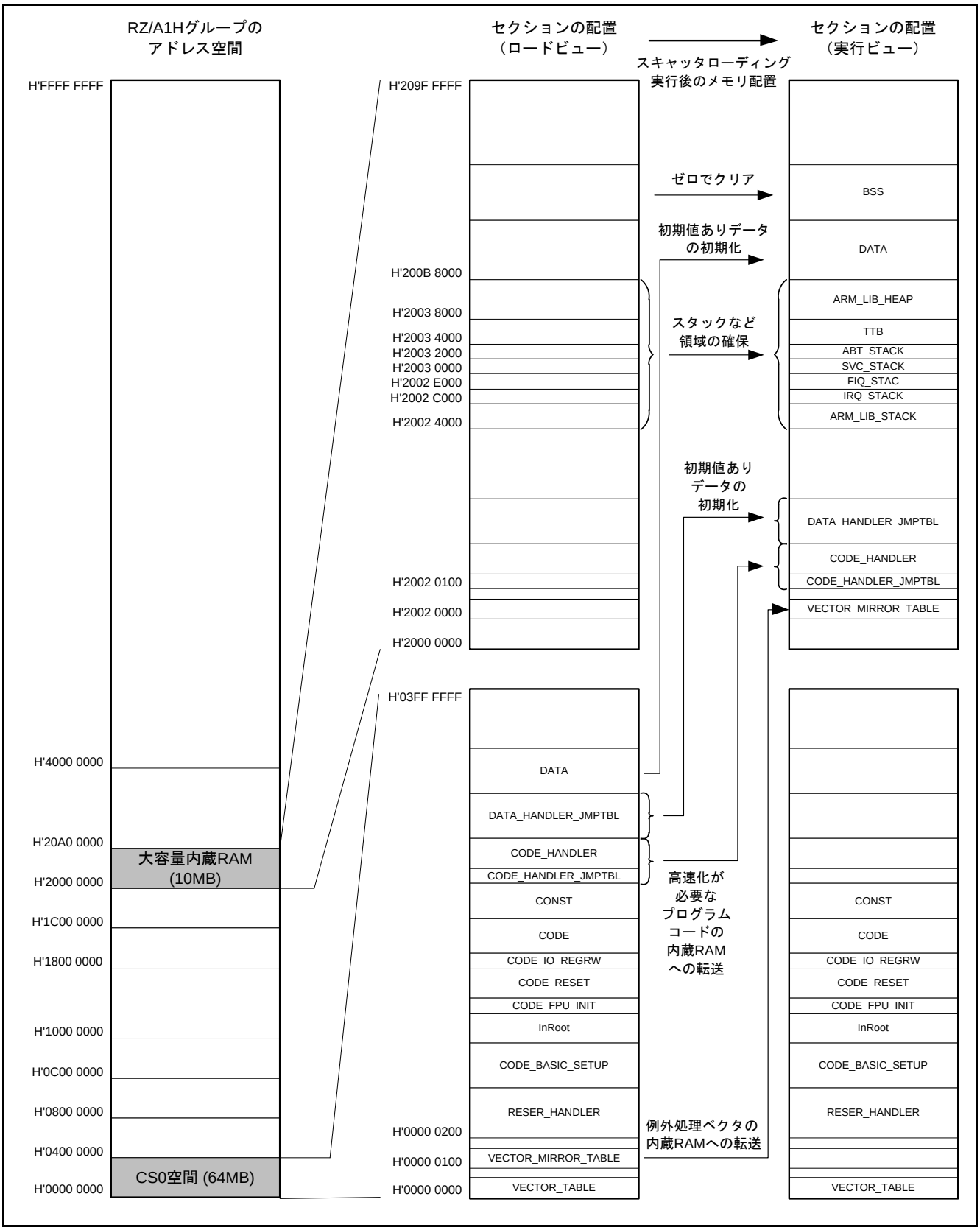
RZ/A1Hグループの
アドレス空間
セクションの配置
（ロードビュー）
スキャッタローディング
実行後のメモリ配置
セクションの配置
（実行ビュー）
H'FFFF FFFF
H'4000 0000
H'20A0 0000
大容量内蔵RAM
(10MB)
H'2000 0000
H'1C00 0000
H'1800 0000
H'1000 0000
H'0C00 0000
H'0800 0000
H'0400 0000
CS0空間 (64MB)
H'0000 0000
H'209F FFFF
H'200B 8000
H'2003 8000
H'2003 4000
H'2003 2000
H'2003 0000
H'2002 E000
H'2002 C000
H'2002 4000
H'2002 0100
H'2002 0000
H'2000 0000
H'03FF FFFF
H'0000 0200
H'0000 0100
H'0000 0000
DATA
DATA_HANDLER_JMPTBL
CODE_HANDLER
CODE_HANDLER_JMPTBL
CONST
CODE
CODE_IO_REGRW
CODE_RESET
CODE_FPU_INIT
InRoot
CODE_BASIC_SETUP
RESER_HANDLER
VECTOR_MIRROR_TABLE
VECTOR_TABLE
ゼロでクリア
初期値ありデータ
の初期化
スタックなど
領域の確保
初期値あり
データの
初期化
高速化が
必要な
プログラム
コードの
内蔵RAM
への転送
例外処理ベクタの
内蔵RAMへの転送
BSS
DATA
ARM_LIB_HEAP
TTB
ABT_STACK
SVC_STACK
FIQ_STAC
IRQ_STACK
ARM_LIB_STACK
DATA_HANDLER_JMPTBL
CODE_HANDLER
CODE_HANDLER_JMPTBL
VECTOR_MIRROR_TABLE
CONST
CODE
CODE_IO_REGRW
CODE_RESET
CODE_FPU_INIT
InRoot
CODE_BASIC_SETUP
RESER_HANDLER
VECTOR_TABLE

図 6.3　セクション配置

6.2.2　MMU の設定

RZ/A1H CPU ボード RTK772100BC00000BR で使用するハードウェア資源のメモリマップにあわせて、H'0000 0000 番地から 1MB 単位で 4GB の領域を MMU で管理するように設定しています（ttb_init.s ファイルで設定しています）。システムにあわせてカスタマイズする場合は、最少単位を 1MB としてください。

表 6.3 に、参考プログラムの MMU の設定を示します。

表 6.3　MMU の設定

定義名	内容	アドレス	サイズ	メモリタイプ
M_SIZE_NOR	CS0、CS1 空間 （NOR フラッシュメモリ）	H'0000 0000 ～ H'07FFFFFF	128MB	L1 キャッシュ有効、 ノーマルメモリ
M_SIZE_SDRAM	CS2、CS3 空間 （SDRAM）	H'0800 0000 ～ H'0FFF FFFF	128MB	L1 キャッシュ有効、 ノーマルメモリ
M_SIZE_CS45	CS4、CS5 空間	H'1000 0000 ～ H'17FF FFFF	128MB	ストロングリオーダメモリ （L1 キャッシュ無効）
M_SIZE_SPI	SPI マルチ IO バス空間 1 および 2（シリアルフラッシュメモリ）	H'1800 0000 ～ H'1FFF FFFF	128MB	L1 キャッシュ有効、 ノーマルメモリ
M_SIZE_RAM	大容量内蔵 RAM 空間	H'2000 0000 ～ H'209F FFFF	10MB	L1 キャッシュ有効、 ノーマルメモリ
M_SIZE_IO_1	内蔵周辺モジュールおよび 予約エリア	H'20A0 0000 ～ H'3FFF FFFF	502MB	ストロングリオーダメモリ （L1 キャッシュ無効）
M_SIZE_NOR_M	CS0、CS1 ミラー空間	H'4000 0000 ～ H'47FF FFFF	128MB	L1 キャッシュ無効、 ノーマルメモリ
M_SIZE_SDRAM_M	CS2、CS3 ミラー空間	H'4800 0000 ～ H'4FFF FFFF	128MB	L1 キャッシュ無効、 ノーマルメモリ
M_SIZE_CS45_M	CS4、CS5 ミラー空間	H'5000 0000 ～ H'57FF FFFF	128MB	ストロングリオーダメモリ （L1 キャッシュ無効）
M_SIZE_SPI_M	SPI マルチ IO バス ミラー空間 1 および 2	H'5800 0000 ～ H'5FFF FFFF	128MB	L1 キャッシュ無効、 ノーマルメモリ
M_SIZE_RAM_M	大容量内蔵 RAM ミラー空間	H'6000 0000 ～ H'609F FFFF	10MB	L1 キャッシュ無効、 ノーマルメモリ
M_SIZE_IO_2	内蔵周辺モジュールおよび 予約エリア	H'60A0 0000 ～ H'FFFF FFFF	2550MB	ストロングリオーダメモリ （L1 キャッシュ無効）

6.3　割り込み

表 6.4 にサンプルコードで使用する割り込みを示します。

表 6.4　サンプルコードで使用する割り込み

割り込み(要因 ID)	優先度	処理概要
CMP1	5	ディスプレイアウトコンペア相違検出割り込み
GR3_VLINE1	5	グラフィックス（3）パネル出力の指定ライン信号

6.4　固定幅整数一覧

表 6.5 にサンプルコードで使用する固定幅整数を示します。

表 6.5　サンプルコードで使用する固定幅整数

シンボル	内容
char_t	8 ビット文字
bool_t	論理型。値は true（1）, false（0）
int_t	高速な整数、符号あり、本サンプルコードでは 32 ビット整数。
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）
float32_t	32 ビット浮動小数 （"__ARM_NEON__"を指定時、標準ライブラリにて定義）
float64_t	64 ビット浮動小数（標準ライブラリにて定義） （"__ARM_NEON__"を指定時、標準ライブラリにて定義）
float128_t	128 ビット浮動小数

6.5　定数/エラーコード一覧

表 6.6 にサンプルコードで使用する定数、表 6.7 にサンプルコードのエラーコードを示します。

表 6.6　サンプルコードで使用する定数

定数名	設定値	内容
DISCOM_CHANNEL_0	0	DISCOM チャンネル 0
DISCOM_CHANNEL_1	1	DISCOM チェンネル 1
DISCOM_OFF	0	Off
DISCOM_ON	1	On
DISCOM_PIX_FORMAT_ARGB8888	0	ピクセルフォーマット ARGB8888
DISCOM_PIX_FORMAT_RGB888	1	ピクセルフォーマット RGB888
DISCOM_PIX_FORMAT_RGB565	2	ピクセルフォーマット RGB565
DISCOM_CMPSELP_NO	0	非選択
DISCOM_CMPSELP_1_GR0	1	グラフィックス（0）
DISCOM_CMPSELP_2_GR1	2	グラフィックス（1）
DISCOM_CMPSELP_3_GR2	3	グラフィックス（2）
DISCOM_CMPSELP_4_GR3	4	グラフィックス（3）
DISCOM_CMPSELP_5_OIR	5	グラフィックス（OIR）
DISCOM_CMPSELP_AB	6	αブレンド後のグラフィックスデータ

表 6.7　サンプルコードのエラーコード

定数名	設定値	内容
DISCOM_OK	0	正常終了
DISCOM_ERR_PARAM_CHANNEL	1	パラメータエラー：指定されたチャンネル番号が正しくありません。
DISCOM_ERR_PARAM_BIT_WIDTH	2	パラメータエラー：指定された値は、パラメータのビット幅より大きい。
DISCOM_ERR_PARAM_UNDEFINED	3	パラメータエラー：指定された値定義されていません。.
DISCOM_ERR_PARAM_NULL	4	パラメータエラー：NULL ポインタに設定することはできません。
DISCOM_ERR_UPDATE_TIMING	5	タイミングエラー：このタイミングでレジスタを更新した場合、意図しないCRCコード計算結果となります。
DISCOM_ERR_STATUS	6	ステータスエラー：この状態では、API 関数の呼出しが許可されていない。

6.6　変数一覧

表 6.8 に static 型変数を示します。

表 6.8　static 型変数

型	変数名	内容	使用関数
int32_t	crc_error	DISCOM の割り込み検出フラグ	CrcErrorDetection ()

6.7　関数一覧

表 6.9 に関数を示します。

表 6.9　関数一覧

関数名	ページ番号
R_DISCOM_Initialize	16
R_DISCOM_Terminate	16
R_DISCOM_Configure	17
R_DISCOM_SetInterrupt	19
R_DISCOM_Start	19
R_DISCOM_Stop	19
R_DISCOM_GetCRC	20
DISCOMDRV_Init	20
DISCOMDRV _Term	20
DISCOMDRV _ConvChannel	21
DISCOMDRV _ConvGrTypeSelPlanel	22
DISCOMDRV _ConvPixRdFormat	23
DISCOMDRV _CalcCRC	24

6.8　関数仕様

サンプルコードの関数仕様を示します。

6.8.1　R_DISCOM_Initialize

R_DISCOM_Initialize		
概　要	DISCOM ドライバ初期化処理	
ヘッダ	r_discom.h	
宣　言	discom_error_t R_DISCOM_Initialize(const discom_channel_t ch, void (* const init_func)(uint32_t), const uint32_t user_num)	
説　明	この関数では、以下の処理を行っています。 - 指定されたユーザ定義関数の実行 - DISCOM 内部変数の初期化 - パネルクロックのイネーブル	
引　数	ch	: チャンネル l
	init_func	: ユーザ定義関数のポインタ
	user_num	: ユーザ定義変数
リターン値	エラーコード	

6.8.2　R_DISCOM_Terminate

R_DISCOM_Terminate		
概　要	DISCOM ドライバ終了処理	
ヘッダ	r_discom.h	
宣　言	discom_error_t R_DISCOM_Terminate(const discom_channel_t ch, void (* const quit_func)(uint32_t), const uint32_t user_num)	
説　明	この関数では、以下の処理を行っています。 - 指定されたユーザ定義関数の実行 - DISCOM の割込み停止	
引　数	ch	: チャンネル
	quit_func	: ユーザ定義関数のポインタ
	user_num	: ユーザ定義変数
リターン値	エラーコード	

6.8.3　R_DISCOM_Configure

R_DISCOM_Configure

概　要	DISCOM ドライバ実行パラメータの設定処理
ヘッダ	r_discom.h
宣　言	discom_error_t R_DISCOM_Configure(const discom_channel_t ch, const discom_config_t * const config, const discom_rect_area_t * const rect_area)
説　明	この関数では、以下の処理を行っています。 - 実行パラメータの設定 - CRC コードを計算する為の矩形領域を設定する
引　数	ch　：チャンネル config　：実行パラメータ rect_area　：CRC コードが計算される矩形領域
リターン値	エラーコード

discom_config_t 構造体のメンバは以下の通りです。

```
typedef struct
{
    discom_pix_format_t  pix_format;
    uint8_t              cmpdfa;
    discom_onoff_t       cmpdauf;
    discom_cmpselp_t     cmpselp;
} discom_config_t;
```

型 引数名	入出力	説明
discom_pix_format_t pix_format	in	ピクセルフォーマット • DISCOM_PIX_FORMAT_ARGB8888: ARGB8888 • DISCOM_PIX_FORMAT_RGB888: RGB888 • DISCOM_PIX_FORMAT_RGB565: RGB565
uint8_t cmpdfa	in	ディスプレイアウトコンペアデフォルトα値
discom_onoff_t cmpdauf	in	ディスプレイアウトコンペアデフォルトα値の利用 • DISCOM_OFF: デフォルトα値を使用しない • DISCOM_ON: デフォルトα値を使用する
discom_cmpselp_tdly cmpselp	in	ディスプレイアウトコンペア選択プレーン • DISCOM_CMPSELP_NO: • DISCOM_CMPSELP_1_GR0: グラフィックス（0） • DISCOM_CMPSELP_2_GR1: グラフィックス（1） • DISCOM_CMPSELP_3_GR2: グラフィックス（2） • DISCOM_CMPSELP_4_GR3: グラフィックス（3） • DISCOM_CMPSELP_5_OIR: グラフィックス（OIR） • DISCOM_CMPSELP_AB: αブレンド後のグラフィックスデータ

discom_rect_area_t 構造体のメンバは以下の通りです。

```
typedef struct
{
    uint16_t    cmpspx;
    uint16_t    cmpspy;
    uint16_t    cmpszx;
    uint16_t    cmpszy;
} discom_rect_area_t;
```

型 引数名	入出力	説明
uint16_t cmpspx	in	ディスプレイアウトコンペア水平方向開始位置 CRC コード計算対象となる矩形領域の水平方向開始位置を指定します。グラフィックスデータの水平方向サイズ以下の値を設定してください。
uint16_t cmpspy	in	ディスプレイアウトコンペア垂直方向開始位置 CRC コード計算対象となる矩形領域の垂直方向開始位置を指定します。グラフィックスデータの垂直方向サイズ以下の値を設定してください。
uint16_t cmpszx	in	ディスプレイアウトコンペア水平方向サイズ CRC コード計算対象となる矩形領域の水平方向サイズを指定します。グラフィックスデータ水平方向サイズ≧水平方向開始位置(cmpspx)＋水平方向サイズ(cmpszx) となるように値を設定してください。
uint16_t cmpszy	in	ディスプレイアウトコンペア垂直方向サイズ CRC コード計算対象となる矩形領域の垂直方向サイズを指定します。グラフィックスデータ垂直方向サイズ≧垂直方向開始位置(cmpspy)＋垂直方向サイズ(cmpszy) なるように値を設定してください。

6.8.4　R_DISCOM_SetInterrupt

R_DISCOM_SetInterrupt		
概　要	DISCOM ドライバ割込み設定処理	
ヘッダ	r_discom.h	
宣　言	discom_error_t R_DISCOM_SetInterrupt(const discom_channel_t ch, void (* const callback)(void))	
説　明	この関数では、以下の処理を行っています。 - DISCOM の割込み設定. - コールバック関数の設定	
引　数	ch	: チャンネル
	callback	: ユーザ定義関数のポインタ
リターン値	エラーコード	

6.8.5　R_DISCOM_Start

R_DISCOM_Start		
概　要	DISCOM ドライバ処理開始	
ヘッダ	r_discom.h	
宣　言	discom_error_t R_DISCOM_Start(const discom_channel_t ch, const uint32_t cmpecrc)	
説　明	この関数では、以下の処理を行っています。 - DISCOM 処理開始 - 比較する CRC コードの設定	
引　数	ch	: チャンネル
	cmpecrc	: 比較 CRC コード
リターン値	エラーコード	

6.8.6　R_DISCOM_Stop

R_DISCOM_Stop		
概　要	DISCOM ドライバ停止処理	
ヘッダ	r_discom.h	
宣　言	discom_error_t R_DISCOM_Stop(const discom_channel_t ch)	
説　明	この関数では、以下の処理を行っています。 - DISCOM 処理の停止	
引　数	ch	: チャンネル
リターン値	エラーコード	

6.8.7　R_DISCOM_GetCRC

R_DISCOM_GetCRC	
概　要	DISCOM ドライバ CRC コードの取得
ヘッダ	r_discom.h
宣　言	discom_error_t R_DISCOM_GetCRC(const discom_channel_t ch, const uint32_t * crc_code);
説　明	この関数では、以下の処理を行っています。 - CRC コードの取得
引　数	ch　：Channel crc_code　：CRC code
リターン値	エラーコード

6.8.8　DISCOMDRV_Init

DISCOMDRV _Init	
概　要	DISCOM ドライバ初期化処理
ヘッダ	discom_drv_utility.h
宣　言	discom_error_t DISCOMDRV _Init(const discom_channel_t channel, void (* const IntCallbackFunc)(void))
説　明	この関数では、以下の処理を行っています。 - R_DISCOM_Initialize ドライバ関数呼出し. - R_DISCOM_SetInterrupt ドライバ関数呼出し
引　数	channel　：チャンネル IntCallbackFunc　：ユーザ定義関数のポインタ
リターン値	エラーコード

6.8.9　DISCOMDRV _Term

DISCOMDRV _Term	
概　要	DISCOM ドライバ終了処理
ヘッダ	discom_drv_utility.h
宣　言	discom_error_t DISCOMDRV _Term(const discom_channel_t channel)
説　明	この関数では、以下の処理を行っています。 - R_DISCOM_Terminate ドライバ関数呼出し. - R_DISCOM_SetInterrupt ドライバ関数呼出し.
引　数	channel　：チャンネル
リターン値	エラーコード

6.8.10　DISCOMDRV _ConvChannel

DISCOMDRV _ConvChannel

概　要	チャンネル番号の変更
ヘッダ	discom_drv_utility.h
宣　言	discom_channel_t DISCOMDRV _ConvChannel (vdc5_channel_t channel)
説　明	この関数では、以下の処理を行っています。 - VDC5 のチャンネル番号から DISCOM のチャンネル番号に変更
引　数	Channel　　：チャンネル
リターン値	エラーコード

r_vdc5.h vdc5_channel_t 列挙方を以下に記載します。

```
typedef enum
{
    VDC5_CHANNEL_0 = 0,     /*!< VDC5 channel 0 */
    VDC5_CHANNEL_1,         /*!< VDC5 channel 1 */
    VDC5_CHANNEL_NUM        /*!< The number of VDC5 channels */
} vdc5_channel_t;
```

r_discom.h discom_channel_t 列挙型を以下に記載します。

```
typedef enum
{
    DISCOM_CHANNEL_0 = 0,   /*!< DISCOM channel 0 */
    DISCOM_CHANNEL_1,       /*!< DISCOM channel 1 */
    DISCOM_CHANNEL_NUM      /*!< The number of DISCOM channels */
} discom_channel_t;
```

6.8.11　DISCOMDRV _ConvGrTypeSelPlanel

DISCOMDRV _ConvGrTypeSelPlanel

概　要	グラフィックスタイプの変更.
ヘッダ	discom_drv_utility.h
宣　言	discom_cmpselp_t DISCOMDRV_ConvGrTypeSelPlane(const vdc5_graphics_type_t graphics_type)
説　明	この関数では、以下の処理を行っています。 - VDC5 のグラフィックスタイプを DISCOM のグラフィックスタイプに変更
引　数	graphics_type　　：VDC5 Graphics type ID
リターン値	エラーコード.

r_vdc5.h vdc5_graphics_type_t 列挙方を以下に記載します。

```
typedef enum
{
    VDC5_GR_TYPE_GR0 = 0,  /*!< Graphics 0 */
    VDC5_GR_TYPE_GR1,      /*!< Graphics 1 */
    VDC5_GR_TYPE_GR2,      /*!< Graphics 2 */
    VDC5_GR_TYPE_GR3,      /*!< Graphics 3 */
    VDC5_GR_TYPE_VIN,      /*!< VIN */
    VDC5_GR_TYPE_OIR,      /*!< OIR */
    VDC5_GR_TYPE_NUM
} vdc5_graphics_type_t;
```

r_discom.h discom_cmpselp_t 列挙方を以下に記載します。

```
typedef enum
{
    DISCOM_CMPSELP_NO    = 0,  /*!< No data */
    DISCOM_CMPSELP_1_GR0 = 1,  /*!< Graphics data of plane 1 (VDC5 graphics 0) */
    DISCOM_CMPSELP_2_GR1 = 2,  /*!< Graphics data of plane 2 (VDC5 graphics 1) */
    DISCOM_CMPSELP_3_GR2 = 3,  /*!< Graphics data of plane 3 (VDC5 graphics 2) */
    DISCOM_CMPSELP_4_GR3 = 4,  /*!< Graphics data of plane 4 (VDC5 graphics 3) */
    DISCOM_CMPSELP_5_OIR = 5,  /*!< Graphics data of plane 5 (VDC5 OIR)*/
    DISCOM_CMPSELP_AB    = 9   /*!< Graphics data after alpha blending */
} discom_cmpselp_t;
```

6.8.12　DISCOMDRV _ConvPixRdFormat

DISCOMDRV _ConvPixRdFormat	
概　要	ピクセルフォーマットタイプ変換
ヘッダ	discom_drv_utility.h
宣　言	vdc5_gr_format_t DISCOMDRV _ConvPixRdFormat(const discom_pix_format_t pix_format)
説　明	この関数では、以下の処理を行っています。 - DISCOM のピクセルフォーマットから VDC5 のピクセルフォーマットに変更
引　数	pix_format
リターン値	エラーコード

r_discom.h discom_pix_format_t 列挙方を以下に記載します。

```
typedef enum
{
    DISCOM_PIX_FORMAT_ARGB8888 = 0,
    DISCOM_PIX_FORMAT_RGB888,
    DISCOM_PIX_FORMAT_RGB565,
    DISCOM_PIX_FORMAT_NUM
} discom_pix_format_t;
```

r_vdc5.h vdc5_gr_format_t 列挙方を以下に記載します。

```
typedef enum
{
    VDC5_GR_FORMAT_RGB565 = 0, /*!< RGB565 */
    VDC5_GR_FORMAT_RGB888,     /*!< RGB888 */
    VDC5_GR_FORMAT_ARGB8888,   /*!< ARGB8888 */
                .
                .
    VDC5_GR_FORMAT_NUM         /*!< The number of signal formats */
} vdc5_gr_format_t;
```

6.8.13　DISCOMDRV _CalcCRC

DISCOMDRV _CalcCRC

概　要	CRC 計算
ヘッダ	discom_drv_utility.h
宣　言	uint32_t DISCOMDRV _CalcCRC (void * const framebuff, const uint32_t fb_stride, const discom_pix_format_t pix_format, const uint32_t crc_ini, const discom_rect_area_t * const rect_area)
説　明	この関数では、以下の処理を行っています。 -CRC calculation of the rect area.
引　数	framebuff : バッファポインタ fb_stride : バッファストライド pix_format : ピクセルフォーマット crc_ini : CRC の初期値 rect_area : CRC 計算を行う矩形エリア
リターン値	CRC コード

6.9　フローチャート

6.9.1　メイン処理

図 6.4 にメイン処理のフローチャートを示します。

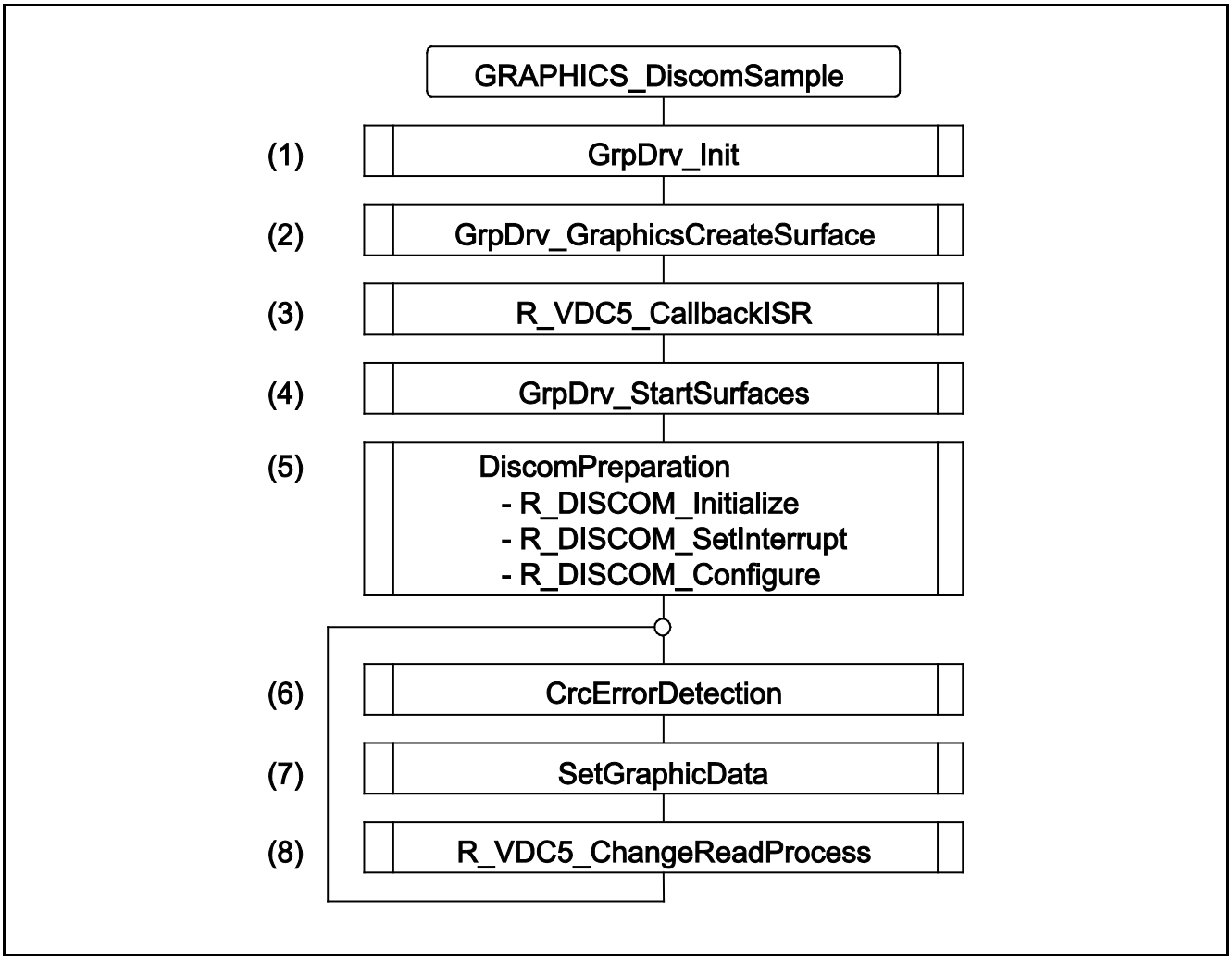

図 6.4　メイン処理

(1)～(4)：VDC5 初期化及び表示設定処理

(5)　　：DISCOM の初期化及び動作パラメータの設定

(6)　　：表示グラフィックの CRC 計算及び DISCOM 開始、CRC 比較結果確認処理

(7)　　：表示グラフィックデータの作成

(8)　　：表示グラフィックの切替処理

6.9.2　DISCOM の CRC エラー検出処理

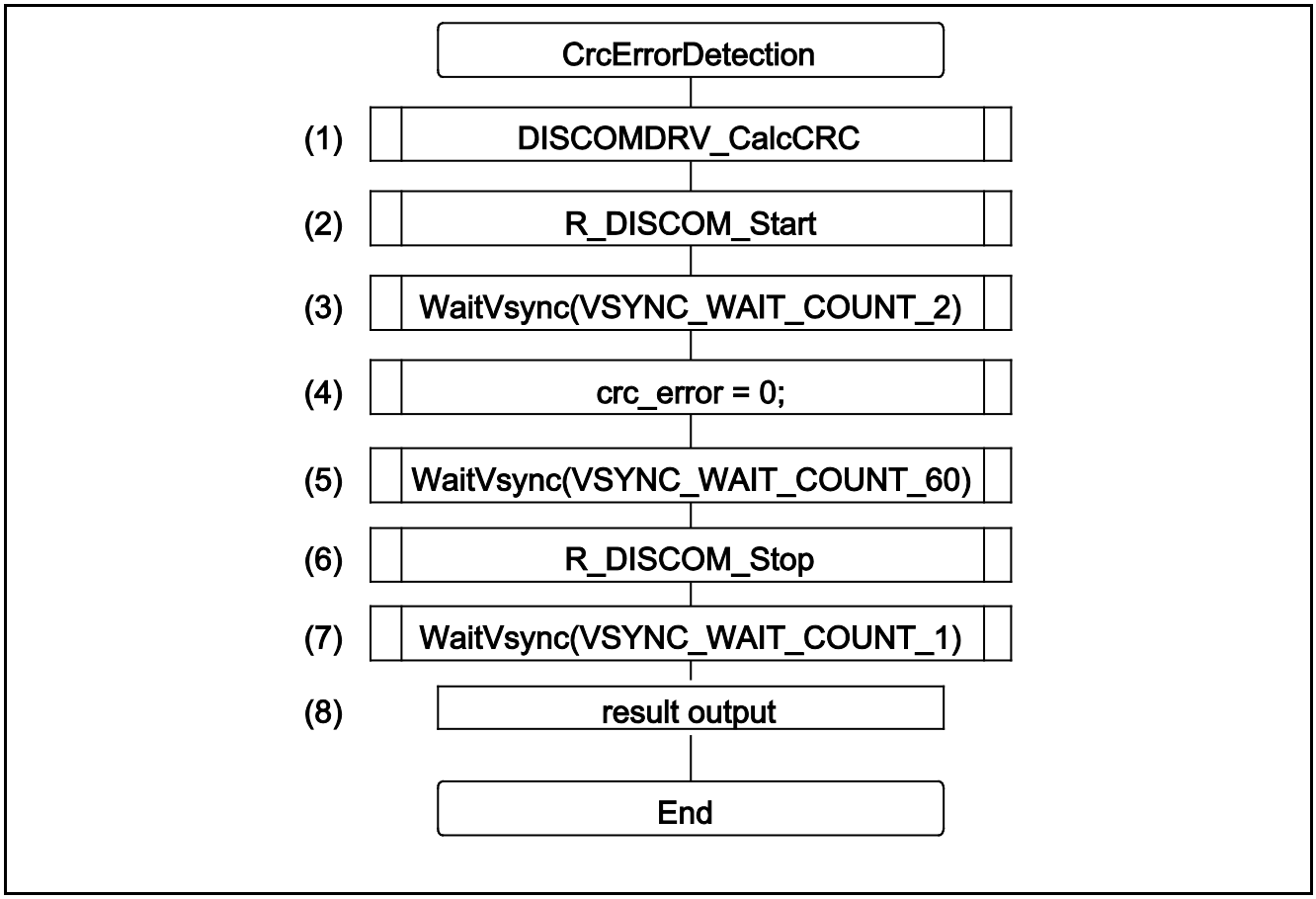

図 6.5　DISCOM エラー検出処理

(1)　：ソフトウェアによる表示グラフィックスの CRC 計算処理

(2)　：DISCOM を開始

(3)　：2 Vsync 待ち処理

　グラフィックスデータ有効期間の終わりで DISOCM のレジスタ設定が更新される為、レジスタ設定更新で 1 Vsync 間待ちます。また、有効なグラフィックスデータの初めを検出してディスプレイアウトコンペアが開始される為、1 Vsymc 待ちます。合計 2Vsync の待ちを行っています。

(4)　：DISCOM の割り込み検出フラグをクリア

　グラフィックスデータが異なった場合、ディスプレイアウトコンペア相違検出割り込みが入り本フラグに 1 がセットされます。

(5)　：ディスプレイアウトコンペア処理を VSYNC_WAIT_COUNT 間行います。

(6)　：DISCOM を停止

(7)　：1 Vsync 待ち処理

　DISCOM のレジスタ設定更新で 1 Vsync 間待ちます。

(8)　：ディスプレイアウトコンペアの結果

　DISCOM の割り込み検出フラグの状態をみて、結果をシリアルで出力します。

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/A1H 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。）

R7S72100 RTK772100BC00000BR (GENMAI) 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。）

R7S72100 CPU ボード用オプションボード RTK7721000B00000BR（GENMAI）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。）

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
（最新版を ARM ホームページから入手してください。）

ARM Generic Inerrupt Controller Architecure Specification Architecture version 1.0
（最新版を ARM 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

ARM ソフトウェア開発ツール（ARM Compiler toolchain、ARM DS-5 等）に関しては、ARM ホームページから入手してください。
（最新版をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録	RZ/A1H グループ ディスプレイアウトコンペアユニット(DISCOM)サンプルプログラム

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.05.27	—	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. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は､切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

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

ルネサス エレクトロニクス株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

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