RX ファミリ

グラフィック LCD コントローラモジュール

Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT)を使用したグラフィック LCD コントローラモジュールについて説明します。本モジュールはグラフィック LCD コントローラ（以下、GLCDC と称す）を使用して、画像データを LCD パネルに表示します。以降、本モジュールを GLCDC FIT モジュールと表記します。

対象デバイス

- RX65N グループ、RX651 グループ　　ROM 容量：1.5MB　～　2MB
- RX72M グループ
- RX72N グループ
- RX66N グループ

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

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については 6.1 動作確認環境を参照してください。

関連ドキュメント

Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)

目次

1. 概要

1.1 GLCDC FIT モジュールとは

GLCDC FIT モジュールは API として、プロジェクトに組み込んで使用します。GLCDC FIT モジュールの組み込み方については、「2.12 FIT モジュールの追加方法」を参照してください。

1.2 GLCDC FIT モジュールの概要

GLCDC FIT モジュールは GLCDC を使用して、メモリから読み出した画像データを LCD パネルに出力する手段を提供します。以下に、GLCDC FIT モジュールでサポートする機能を示します。

- 32bpp、16bpp のデータフォーマット、8 ビット、4 ビット、1 ビットの CLUT（カラールックアップテーブル）データフォーマットを選択可能。
- 3 面の重ね合わせ機能（グラフィック画面 2 面に対してはアルファブレンドが可能）
- 出力する LCD パネル用に輝度補正、コントラスト補正、RGB ガンマ補正が可能。
- RGB（888）、RGB（666）、RGB（565）のパラレルデータ出力を選択可能。出力データフォーマットに対してディザ処理が可能。

1.3 API の概要

表 1.1 に GLCDC FIT モジュールに含まれる API 関数を示します。

表 1.1　API 関数一覧

関数	関数説明
R_GLCDC_Open	GLCDC FIT モジュールの初期化を行います。 コンフィグレーションオプション “GLCDC_CFG_CONFIGURATION_MODE”または QE for Display [RX] V2.0.0 以降を使用する場合 （define 定義“QE_DISPLAY_CONFIGURATION”が宣言されている場合）によって動作が異なります。 詳細は 3.API 関数 を参照してください。
R_GLCDC_Close	GLCDC FIT モジュールの終了処理を行います。
R_GLCDC_Control	GLCDC FIT モジュールの制御処理を行います。
R_GLCDC_LayerChange	GLCDC のグラフィック 1 とグラフィック 2 の動作を変更します。
R_GLCDC_BufferChange	GLCDC のグラフィック 1、グラフィック 2 のフレームバッファのアドレスを変更します。
R_GLCDC_ColorCorrection	GLCDC の輝度補正、コントラスト補正、ガンマ補正を変更します。
R_GLCDC_ClutUpdate	GLCDC の CLUT メモリを更新します。（本関数の処理が完了すると、更新した CLUT メモリは出力に反映されます。）
R_GLCDC_ClutUpdate_NoReflect	GLCDC の CLUT メモリを更新します。（本関数の処理が完了した際、更新した CLUT メモリは出力に反映されていません。 R_GLCDC_LayerChange 関数を実行し、出力に反映させてください。）
R_GLCDC_GetStatus	GLCDC のステータスを取得します。
R_GLCDC_GetVersion	GLCDC FIT モジュールのバージョン番号を返します。

1.4　状態遷移図

図 1.1 に GLCDC FIT モジュールの状態遷移図を示します。

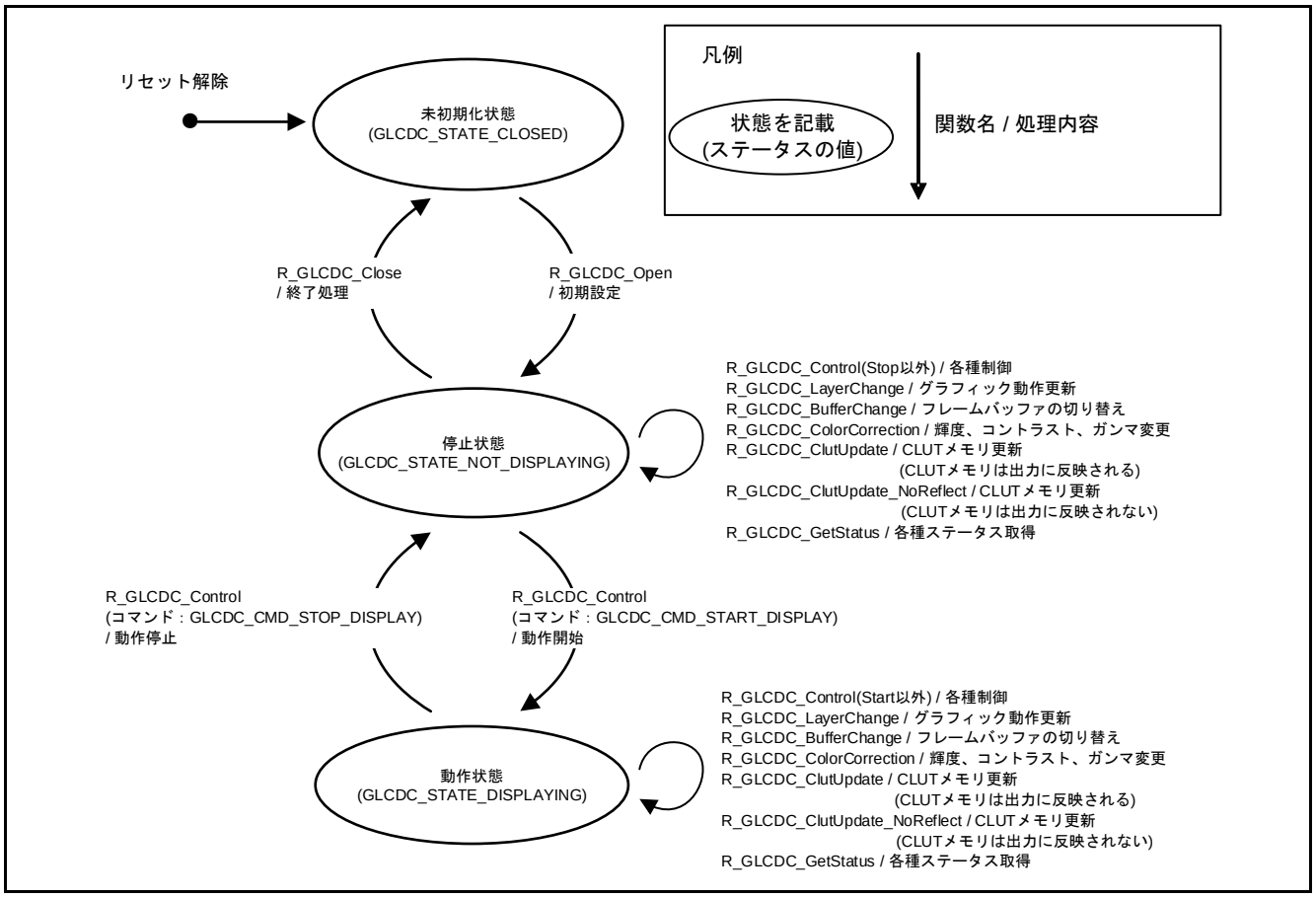

図 1.1　GLCDC FIT モジュールの状態遷移図

1.5　制限事項

GLCDC FIT モジュールは以下の制限事項があります。

- シリアル RGB のデータ出力に対応していません。
- 外部クロック（LCD_EXTCLK）の入力に対応していません。

1.6　RAM の配置に関する制限事項

FIT では、API 関数のポインタ引数に NULL と同じ値を設定すると、パラメータチェックにより戻り値がエラーとなる場合があります。そのため、API 関数に渡すポインタ引数の値は NULL と同じ値にしないでください。

ライブラリ関数の仕様で NULL の値は 0 と定義されています。そのため、API 関数のポインタ引数に渡す変数や関数が RAM の先頭番地(0x0 番地)に配置されていると上記現象が発生します。この場合、セクションの設定変更をするか、API 関数のポインタ引数に渡す変数や関数が 0x0 番地に配置されないように RAM の先頭にダミーの変数を用意してください。

なお、CCRX プロジェクト(e^2 studio V7.5.0)の場合、変数が 0x0 番地に配置されることを防ぐために RAM の先頭番地が 0x4 になっています。GCC プロジェクト(e^2 studio V7.5.0)、IAR プロジェクト(EWRX

V4.12.1)の場合は RAM の先頭番地が 0x0 になっていますので、上記対策が必要となります。

　IDE のバージョンアップによりセクションのデフォルト設定が変更されることがあります。最新の IDE を使用される際は、セクション設定をご確認の上、ご対応ください。

2. API 情報

GLCDC FIT モジュールは、下記の条件で動作を確認しています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- GLCDC

2.2 ソフトウェアの要求

GLCDC FIT モジュールは以下の FIT モジュールに依存しています。

- ボードサポートパッケージ (r_bsp) Rev.5.20 以降

2.3 サポートされているツールチェーン

GLCDC FIT モジュールは「6.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.4 使用する割り込みベクタ

R_GLCDC_Open 関数を実行すると引数の値に対応した VPOS 割り込み、GR1UF 割り込み、GR2UF 割り込みが有効になります。

表 2.1 に GLCDC FIT モジュールが使用する割り込みベクタを示します。

表 2.1　使用する割り込みベクター覧

デバイス	割り込みベクタ
RX65N RX72M RX72N RX66N	GROUPAL1 割り込み（ベクタ番号: 113） ● VPOS 割り込み（グループ割り込み要因番号：8） ● GR1UF 割り込み（グループ割り込み要因番号：9） ● GR2UF 割り込み（グループ割り込み要因番号：10）

2.5 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_glcdc_rx_if.h に記載しています。

2.6 整数型

GLCDC FIT モジュールは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.7　コンパイル時の設定

GLCDC FIT モジュールのコンフィグレーションオプションの設定は、r_glcdc_rx_config.h で行います。オプション名および設定値に関する説明を、下表に示します。

Configuration options in r_glcdc_rx_config.h	
GLCDC_CFG_PARAM_CHECKING_ENABLE ※デフォルト値は　“1”	パラメータチェック処理をコードに含めるか選択できます。 “0”を選択すると、パラメータチェック処理をコードから省略できるため、コードサイズが削減できます。 “0”の場合、パラメータチェック処理をコードから省略します。 “1”の場合、パラメータチェック処理をコードに含めます。
GLCDC_CFG_INTERRUPT_PRIORITY_LEVEL ※デフォルト値は　“5”	グループ AL1 割り込みの優先レベルを設定してください。 “0”　～　“15”の範囲で設定してください。
GLCDC_CFG_CONFIGURATION_MODE ※デフォルト値は“0”	GLCDC の設定方法を選択できます。 “0”の場合、GLCDC 設定データ構造体変数からパラメータを設定します。 “1”の場合、コンフィグレーションオプションからパラメータを設定します。

r_glcdc_rx_config.h で定義されている上記以外のコンフィグレーションオプションは、GLCDC_CFG_CONFIGURATION_MODE が“1”または QE for Display [RX] V2.0.0 以降を使用する場合（define 定義“QE_DISPLAY_CONFIGURATION”が宣言されている場合）に有効になります。また、それにより GLCDC の設定方法も表 2.2 のように変わります。

上記以外のコンフィグレーションオプションの詳細は、3 API 関数の R_GLCDC_Open () <コンフィグレーションオプションで設定する場合>を参照してください。

QE for Display[RX]については 5.6 QE for Dispay[RX]との連携を参照してください。

表 2.2　GLCDC の設定方法

QE for Display[RX]の使用 (QE_DISPLAY_CONFIGURATION)	GLCDC の設定方法の選択 (GLCDC_CFG_CONFIGURATION_MODE)	GLCDC の設定方法
QE for Display[RX]を使用しない (define 定義なし)	0	GLCDC 設定データ構造体変数からパラメータを設定
	1	コンフィグレーションオプションからパラメータを設定
QE for Display[RX]を使用する (define 定義あり)	0	コンフィグレーションオプションからパラメータを設定
	1	コンフィグレーションオプションからパラメータを設定

2.8　コードサイズ

GLCDC FIT モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します。代表して RX72N を掲載しています。

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.7 コンパイル時の設定」のコンフィグレーションオプションによって決まります。
下表の値は下記条件で確認しています。

モジュールリビジョン: r_glcdc_rx rev1.60

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.05.00
(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)
GCC for Renesas RX 8.3.0 202311
(統合開発環境のデフォルト設定に”-std=gnu99”オプションを追加)
IAR C/C++ Compiler for Renesas RX version 5.10.1
(統合開発環境のデフォルト設定)

ROM、RAM およびスタックのコードサイズ

デバイス	分類	使用メモリ					
		CCRX		GCC		IAR	
		パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
GLCDC_CFG_CONFIGURATION_MODE が“0”かつ QE for Display [RX] を使用しない場合							
RX72N	ROM	**6188 バイト**	**5052 バイト**	**8680 バイト**	**6936 バイト**	**8497 バイト**	**6597 バイト**
	RAM	**52 バイト**		**52 バイト**		**48 バイト**	
	スタック (注 1)	**160 バイト**		**-**		**184 バイト**	
GLCDC_CFG_CONFIGURATION_MODE が“1”または QE for Display [RX] を使用する場合							
RX72N	ROM	**6668 バイト**	**5532 バイト**	**9316 バイト**	**7540 バイト**	**9106 バイト**	**7206 バイト**
	RAM	**52 バイト**		**52 バイト**		**48 バイト**	
	スタック (注 1)	**160 バイト**		**-**		**184 バイト**	

注 1. 割り込み関数の最大使用スタックサイズを含みます。

2.9　引数

API 関数の引数である構造体を示します。この構造体は、API 関数のプロトタイプ宣言とともに r_glcdc_rx_if.h に記載されています。

```
/* GLCDC のメイン設定 */
typedef struct st_glcdc_cfg
{
   /** Generic configuration for display devices */
   glcdc_input_cfg_t input[GLCDC_FRAME_LAYER_NUM];            // GLCDC の入力画像設定
   glcdc_output_cfg_t output;                                 // GLCDC の出力設定
   glcdc_blend_t blend[GLCDC_FRAME_LAYER_NUM];                // ブレンド設定
   glcdc_chromakey_t chromakey[GLCDC_FRAME_LAYER_NUM];        // クロマキー設定
   glcdc_clut_cfg_t clut[GLCDC_FRAME_LAYER_NUM];              // CLUT メモリの設定

   /** 割り込みの設定 */
   glcdc_detect_cfg_t   detection;                            // GLCDC の検出の設定
   glcdc_interrupt_cfg_t   interrupt;                         // GLCDC の割り込みの設定

   /** GLCDC のイベント発生時の設定 */
   void (*p_callback)(void *);                                // コールバック関数へのポインタ

} glcdc_cfg_t;
```

```
/* GLCDC の入力画像設定*/
typedef struct st_glcdc_input_cfg
{
   uint32_t * p_base;                  // フレームバッファの先頭アドレス
   uint16_t hsize;                     // 画像データの水平ピクセルサイズ
   uint16_t vsize;                     // 画像データの垂直ピクセルサイズ
   int32_t offset;                     // 次のラインまでのオフセット値
   glcdc_in_format_t format;           // データフォーマットの設定
   bool frame_edge;                    // グラフィック領域の枠の表示、非表示の設定
   glcdc_coordinate_t coordinate;      // 画像データ表示を表示する開始位置
   glcdc_color_t bg_color;             // グラフィック背景色の設定

} glcdc_input_cfg_t;
```

```
/* GLCDC の出力設定 */
typedef struct st_glcdc_output_cfg
{
   glcdc_timing_t              htiming;              // 水平同期信号（HSYNC）のタイミング設定
   glcdc_timing_t              vtiming;              // 垂直同期信号（VSYNC）のタイミング設定
   glcdc_out_format_t          format;               // 出力データフォーマットの設定
   glcdc_endian_t              endian;               // 出力データのビットエンディアン設定
   glcdc_color_order_t         color_order;          // ピクセル順序の設定
   glcdc_sync_edge_t           sync_edge;            // HSYNC、VSYNC、データの出力位相の設定

   glcdc_color_t               bg_color;             // 背景色の設定

   glcdc_brightness_t          brightness;           // 輝度の設定
   glcdc_contrast_t            contrast;             // コントラストの設定
   glcdc_gamma_correction_t          gamma;          // ガンマ補正の設定
   glcdc_correction_proc_order_t     correction_proc_order;  // 補正処理の実行順序の設定

   glcdc_dithering_t           dithering;            // ディザ処理の設定

   glcdc_tcon_pin_t            tcon_hsync;           // 水平同期信号（HSYNC）の出力端子の設定
   glcdc_tcon_pin_t            tcon_vsync;           // 垂直同期信号（VSYNC）の出力端子の設定
   glcdc_tcon_pin_t            tcon_de;              // データイネーブル信号（DE）の出力端子の設定
   glcdc_signal_polarity_t     data_enable_polarity;   // データイネーブル信号（DE）の極性の設定
   glcdc_signal_polarity_t     hsync_polarity;         // 水平同期信号（HSYNC）の極性の設定
   glcdc_signal_polarity_t     vsync_polarity;         // 垂直同期信号（VSYNC）の極性の設定

   glcdc_clk_src_t             clksrc;                 // クロックソースの設定
   glcdc_panel_clk_div_t       clock_div_ratio;        // パネルクロックの分周比の設定

} glcdc_output_cfg_t;
```

```
/* ブレンド設定 */
typedef struct st_glcdc_blend
{
   glcdc_blend_control_t       blend_control;        // ブレンド処理の制御設定
   bool                        visible;              // 画像の表示、非表示の設定
   bool                        frame_edge;           // 矩形アルファブレンド領域の枠の表示、非表示設定
   uint8_t                     fixed_blend_value;    // アルファ値の設定
   uint8_t                     fade_speed;           // アルファ値の増減値の設定
   glcdc_coordinate_t          start_coordinate;     // ブレンド処理の開始位置の設定
   glcdc_coordinate_t          end_coordinate;       // ブレンド処理の終了位置の設定

} glcdc_blend_t;
```

```
/* クロマキー設定 */
typedef struct st_glcdc_chromakey
{
   bool              enable;          // RGB 参照クロマキー処理の有効/無効の選択
   glcdc_color_t     before;          // クロマキー処理対象 RGB 値の設定
   glcdc_color_t     after;           // クロマキー置き換え後の ARGB 値の設定

} glcdc_chromakey_t;
```

```
/* GLCDC の割り込みの設定 */
typedef struct st_glcdc_interrupt_cfg
{
   bool       vpos_enable;            // VPOS 割り込みの有効、無効の選択
   bool       gr1uf_enable;           // GR1UF 割り込みの有効、無効の選択
   bool       gr2uf_enable;           // GR2UF 割り込みの有効、無効の選択

} glcdc_interrupt_cfg_t;
```

```
/* GLCDC の検出の設定 */
typedef struct st_glcdc_detect_cfg
{
   bool       vpos_detect;            // VPOS 検出の有効、無効の選択
   bool       gr1uf_detect;           // GR1UF 検出の有効、無効の選択
   bool       gr2uf_detect;           // GR2UF 検出の有効、無効の選択

} glcdc_detect_cfg_t;
```

```
/* GLCDC のコールバック関数の引数 */
typedef struct st_glcdc_callback_args
{
   glcdc_event_t          event;      // イベントコード

} glcdc_callback_args_t;
```

```
/* GLCDC のステータス */
typedef struct st_glcdc_status
{
   glcdc_operating_status_t      state;                  // GLCDC FIT モジュールのステータス
   glcdc_detected_status_t       state_vpos;             // グラフィック 2 指定ライン通知ステータス
   glcdc_detected_status_t       state_gr1uf;            // グラフィック 1 アンダフロー検出ステータス
   glcdc_detected_status_t       state_gr2uf;            // グラフィック 2 アンダフロー検出ステータス
   glcdc_fade_status_t           fade_status[GLCDC_FRAME_LAYER_NUM];
                                                         // アルファブレンドステータス

} glcdc_status_t;
```

```
/* ディザ処理の設定 */
typedef struct st_glcdc_dithering
{
   bool                          dithering_on;           // ディザ処理の有効、無効の選択
   glcdc_dithering_mode_t        dithering_mode;         // ディザ処理のモード選択
   glcdc_dithering_pattern_t     dithering_pattern_a;    // 2x2 パターンディザのパターン値 A の設定
   glcdc_dithering_pattern_t     dithering_pattern_b;    // 2x2 パターンディザのパターン値 B の設定
   glcdc_dithering_pattern_t     dithering_pattern_c;    // 2x2 パターンディザのパターン値 C の設定
   glcdc_dithering_pattern_t     dithering_pattern_d;    // 2x2 パターンディザのパターン値 D の設定

} glcdc_dithering_t;
```

```
/* GLCDC の CLUT メモリの設定 */
typedef struct st_glcdc_clut_cfg
{
   bool              enable;          // CLUT メモリの更新の有効、無効の選択
   uint32_t          * p_base;        // CLUT の先頭アドレスへのポインタ
   uint16_t          start;           //  更新する CLUT メモリの開始エントリ番号
   uint16_t          size;            //  更新する CLUT メモリのサイズ

} glcdc_clut_cfg_t;
```

```
/* GLCDC 動作中の設定 */
typedef struct st_glcdc_runtime_cfg
{
   glcdc_input_cfg_t       input;          // GLCDC のグラフィックの設定
   glcdc_blend_t           blend;          //  ブレンド設定
   glcdc_chromakey_t       chromakey;      //  クロマキー設定

} glcdc_runtime_cfg_t;
```

```
/*  補正処理の設定 */
typedef struct st_glcdc_correction
{
   glcdc_brightness_t          brightness;     //  輝度の設定
   glcdc_contrast_t            contrast;       //  コントラストの設定
   glcdc_gamma_correction_t    gamma;          //  ガンマ補正の設定

} glcdc_correction_t;
```

```
/*  ガンマ補正の設定 */
typedef struct st_glcdc_gamma_correction
{
   bool                    enable;         //  ガンマ補正の有効、無効の選択
   gamma_correction_t      * p_r;          // R 値のガンマ補正テーブルの設定
   gamma_correction_t      * p_g;          // G 値のガンマ補正テーブルの設定
   gamma_correction_t      * p_b;          // B 値のガンマ補正テーブルの設定

} glcdc_gamma_correction_t;
```

```
/*  ガンマ補正テーブルの設定 */
typedef struct st_gamma_correction
{
   uint16_t       gain[GLCDC_GAMMA_CURVE_GAIN_ELEMENT_NUM];
                                                          //  ゲインの設定
   uint16_t       threshold[GLCDC_GAMMA_CURVE_THRESHOLD_ELEMENT_NUM];
                                                          //  しきい値の設定

} gamma_correction_t;
```

```
/* コントラストの設定 */
typedef struct st_glcdc_contrast
{
   bool          enable;          // コントラスト補正の有効、無効の選択
   uint8_t       r;               // R 信号のコントラスト調整値
   uint8_t       g;               // G 信号のコントラスト調整値
   uint8_t       b;               // B 信号のコントラスト調整値

} glcdc_contrast_t;
```

```
/* 輝度の設定 */
typedef struct st_glcdc_brightness
{
   bool          enable;          // 輝度補正の有効、無効の選択
   uint16_t      r;               // R 信号の輝度調整値
   uint16_t      g;               // G 信号の輝度調整値
   uint16_t      b;               // B 信号の輝度調整値

} glcdc_brightness_t;
```

```
/* 座標の設定 */
typedef struct st_glcdc_coordinate
{
   int16_t       x;               // X 座標
   int16_t       y;               // Y 座標

} glcdc_coordinate_t;
```

```
/* カラーの設定 */
typedef struct st_glcdc_color
{
   union
        {
             uint32_t       argb;
             struct
             {
                  uint32_t      a:8;          // A 値
                  uint32_t      r:8;          // R 値
                  uint32_t      g:8;          // G 値
                  uint32_t      b:8;          // B 値
             } byte;
        };

} glcdc_color_t;
```

```
/* 信号の出力タイミングの設定 */
typedef struct st_glcdc_timing
{
   uint16_t      display_cyc;     // データ有効期間のサイクル数
   uint16_t      front_porch;     // フロントポーチのサイクル数
   uint16_t      back_porch;      // バックポーチのサイクル数
   uint16_t      sync_width;      // アサート期間

} glcdc_timing_t;
```

```
/* R_GLCDC_ColorCorrection 関数のコマンド */
typedef enum e_glcdc_correction_cmd
{
  GLCDC_CORRECTION_CMD_SET_ALL,          // 全ての補正処理の設定を変更
  GLCDC_CORRECTION_CMD_BRIGHTNESS,       // 輝度補正の設定を変更
  GLCDC_CORRECTION_CMD_CONTRAST,         // コントラスト補正の設定を変更
  GLCDC_CORRECTION_CMD_GAMMA,            // ガンマ補正の設定を変更

} glcdc_correction_cmd_t;
```

```
/* R_GLCDC_Control 関数のコマンド */
typedef enum e_glcdc_control_cmd
{
  GLCDC_CMD_START_DISPLAY,               // GLCDC の動作開始
  GLCDC_CMD_STOP_DISPLAY,                // GLCDC の動作停止
  GLCDC_CMD_SET_INTERRUPT,               // 割り込みの設定
  GLCDC_CMD_CLR_DETECTED_STATUS,         // 検出ステータスのクリア
  GLCDC_CMD_CHANGE_BG_COLOR,             // バックグラウンド画面の背景色の変更

} glcdc_control_cmd_t;
```

```
/* グラフィック画面の定義 */
typedef enum e_glcdc_frame_layer
{
  GLCDC_FRAME_LAYER_1 = 0,               // グラフィック 1
  GLCDC_FRAME_LAYER_2 = 1                // グラフィック 2

} glcdc_frame_layer_t;
```

```
/* GLCDC FIT モジュールの動作モードの定義 */
typedef enum e_glcdc_state
{
  GLCDC_STATE_CLOSED = 0,                // 初期化前
  GLCDC_STATE_NOT_DISPLAYING = 1,        // GLCDC 動作停止
  GLCDC_STATE_DISPLAYING = 2             // GLCDC 動作中

} glcdc_operating_status_t;
```

```
/* イベントの定義 */
typedef enum e_glcdc_event
{
  GLCDC_EVENT_GR1_UNDERFLOW = 1,         // グラフィック 1 アンダフロー検出が発生
  GLCDC_EVENT_GR2_UNDERFLOW = 2,         // グラフィック 2 アンダフロー検出が発生
  GLCDC_EVENT_LINE_DETECTION = 3,        // グラフィック 2 指定ライン通知検出が発生

} glcdc_event_t;
```

```
/* フレームバッファの画像データのフォーマットの定義 */
typedef enum e_glcdc_in_format
{
   GLCDC_IN_FORMAT_16BITS_RGB565     = 0,   // RGB(565),          16 bits.
   GLCDC_IN_FORMAT_32BITS_RGB888     = 1,   // RGB(888),          32 bits.
   GLCDC_IN_FORMAT_16BITS_ARGB1555 = 2,   // ARGB(1555),        16 bits.
   GLCDC_IN_FORMAT_16BITS_ARGB4444 = 3,   // ARGB(4444),        16 bits.
   GLCDC_IN_FORMAT_32BITS_ARGB8888 = 4,   // ARGB(8888),        32 bits.
   GLCDC_IN_FORMAT_CLUT8                 = 5,     // CLUT(8),           8 bits.
   GLCDC_IN_FORMAT_CLUT4                 = 6,     // CLUT(4),           4 bits.
   GLCDC_IN_FORMAT_CLUT1                 = 7,     // CLUT(1),           1 bits.

} glcdc_in_format_t;
```

```
/* 出力データフォーマットの定義 */
typedef enum e_glcdc_out_format
{
   GLCDC_OUT_FORMAT_24BITS_RGB888 = 0,   // RGB(888),     24 bits.
   GLCDC_OUT_FORMAT_18BITS_RGB666 = 1,   // RGB(666),     18 bits.
   GLCDC_OUT_FORMAT_16BITS_RGB565 = 2,   // RGB(565),     16 bits.

} glcdc_out_format_t;
```

```
/* エンディアンの定義 */
typedef enum e_glcdc_endian
{
   GLCDC_ENDIAN_LITTLE = 0,              // 出力データはリトルエンディアン
   GLCDC_ENDIAN_BIG = 1,                 // 出力データはビッグエンディアン

} glcdc_endian_t;
```

```
/* ピクセル順序の定義 */
typedef enum e_glcdc_color_order
{
   GLCDC_COLOR_ORDER_RGB = 0,  // ピクセル順序は R-G-B 順
   GLCDC_COLOR_ORDER_BGR = 1   // ピクセル順序は B-G-R 順

} glcdc_color_order_t;
```

```
/* 極性の定義 */
typedef enum e_glcdc_signal_polarity
{
   GLCDC_SIGNAL_POLARITY_HIACTIVE = 0,          // ハイアクティブ
   GLCDC_SIGNAL_POLARITY_LOACTIVE = 1,          // ローアクティブ

} glcdc_signal_polarity_t;
```

```
/* 同期エッジの定義 */
typedef enum e_glcdc_sync_edge
{
   GLCDC_SIGNAL_SYNC_EDGE_RISING = 0,           // 立ち上がりに同期
   GLCDC_SIGNAL_SYNC_EDGE_FALLING = 1,          // 立ち下がりに同期

} glcdc_sync_edge_t;
```

```
/* アルファブレンド処理の定義 */
typedef enum e_glcdc_blend_control
{
   GLCDC_BLEND_CONTROL_NONE = 0,              // アルファブレンド処理無効
   GLCDC_BLEND_CONTROL_FADEIN = 1,            // フェードイン
   GLCDC_BLEND_CONTROL_FADEOUT = 2,           // フェードアウト
   GLCDC_BLEND_CONTROL_FIXED = 3,             // アルファ値固定
   GLCDC_BLEND_CONTROL_PIXEL = 4              // ピクセル単位アルファブレンド

} glcdc_blend_control_t;
```

```
/* フェード状態の定義 */
typedef enum e_glcdc_fade_status
{
   GLCDC_FADE_STATUS_NOT_UNDERWAY,         // フェードイン/フェードアウトは停止中
   GLCDC_FADE_STATUS_FADING_UNDERWAY,      // フェードイン/フェードアウトは実行中
   GLCDC_FADE_STATUS_UNCERTAIN             // グラフィックのレジスタ値設定中

} glcdc_fade_status_t;
```

```
/* クロックソースの定義 */
typedef enum e_glcdc_clk_src
{
   GLCDC_CLK_SRC_INTERNAL = 1,       // PLL クロックを使用

} glcdc_clk_src_t;
```

```
/* パネルクロックの分周比の定義 */
typedef enum e_glcdc_panel_clk_div
{
   GLCDC_PANEL_CLK_DIVISOR_1   = 1,       // 1 分周
   GLCDC_PANEL_CLK_DIVISOR_2   = 2,       // 2 分周
   GLCDC_PANEL_CLK_DIVISOR_3   = 3,       // 3 分周
   GLCDC_PANEL_CLK_DIVISOR_4   = 4,       // 4 分周
   GLCDC_PANEL_CLK_DIVISOR_5   = 5,       // 5 分周
   GLCDC_PANEL_CLK_DIVISOR_6   = 6,       // 6 分周
   GLCDC_PANEL_CLK_DIVISOR_7   = 7,       // 7 分周
   GLCDC_PANEL_CLK_DIVISOR_8   = 8,       // 8 分周
   GLCDC_PANEL_CLK_DIVISOR_9   = 9,       // 9 分周
   GLCDC_PANEL_CLK_DIVISOR_12 = 12,       // 12 分周
   GLCDC_PANEL_CLK_DIVISOR_16 = 16,       // 16 分周
   GLCDC_PANEL_CLK_DIVISOR_24 = 24,       // 24 分周
   GLCDC_PANEL_CLK_DIVISOR_32 = 32,       // 32 分周

} glcdc_panel_clk_div_t;
```

```
/* 信号の出力端子の定義 */
typedef enum e_glcdc_tcon_pin
{
   GLCDC_TCON_PIN_0 = 0,            // LCD_TCON0 端子
   GLCDC_TCON_PIN_1 = 1,            // LCD_TCON1 端子
   GLCDC_TCON_PIN_2 = 2,            // LCD_TCON2 端子
   GLCDC_TCON_PIN_3 = 3,            // LCD_TCON3 端子
   GLCDC_TCON_PIN_NON = 4                  // 出力端子を指定しない

} glcdc_tcon_pin_t;
```

```
/* 補正処理の実行順序の定義 */
typedef enum e_glcdc_correction_proc_order
{
   GLCDC_BRIGHTNESS_CONTRAST_TO_GAMMA = 0,   // 輝度、コントラスト -> ガンマ補正の順
   GLCDC_GAMMA_TO_BRIGHTNESS_CONTRAST = 1    // ガンマ補正 -> 輝度、コントラストの順

} glcdc_correction_proc_order_t;
```

```
/* ディザ処理のモードの定義 */
typedef enum e_glcdc_dithering_mode
{
   GLCDC_DITHERING_MODE_TRUNCATE    = 0,     // ディザ処理なし（切り捨て）
   GLCDC_DITHERING_MODE_ROUND_OFF   = 1,     // 0 捨 1 入
   GLCDC_DITHERING_MODE_2X2PATTERN = 2       // 2x2 パターンディザ

} glcdc_dithering_mode_t;
```

```
/* 2x2 パターンディザのパターン値の定義 */
typedef enum e_glcdc_dithering_pattern
{
   GLCDC_DITHERING_PATTERN_00 = 0,      // パターン '00'.
   GLCDC_DITHERING_PATTERN_01 = 1,      // パターン '01'.
   GLCDC_DITHERING_PATTERN_10 = 2,      // パターン '10'.
   GLCDC_DITHERING_PATTERN_11 = 3       // パターン '11'.

} glcdc_dithering_pattern_t;
```

```
/* 検出の定義 */
typedef enum e_glcdc_detected_status
{
   GLCDC_NOT_DETECTED,                  // 検出していない
   GLCDC_DETECTED                       // 検出した

} glcdc_detected_status_t;
```

2.10　戻り値

API 関数の戻り値を示します。この列挙型は、API 関数のプロトタイプ宣言とともに r_glcdc_rx_if.h に記載されています。

```
/* GLCDC の戻り値 */
typedef enum e_glcdc_err
{
   GLCDC_SUCCESS = 0,                      // 正常完了
   GLCDC_ERR_INVALID_PTR,                  // 引数が NULL ポインタ
   GLCDC_ERR_LOCK_FUNC,                    // GLCDC がロック済み
   GLCDC_ERR_INVALID_ARG,                  // 引数の値が不正
   GLCDC_ERR_INVALID_MODE,                 // 関数が実行できないモード
   GLCDC_ERR_NOT_OPEN,                     // R_GLCDC_Open 関数を実行していない
   GLCDC_ERR_INVALID_TIMING_SETTING,       // パネル出力信号のタイミング設定が不正
   GLCDC_ERR_INVALID_LAYER_SETTING,        // グラフィック画面の設定が不正
   GLCDC_ERR_INVALID_ALIGNMENT,            // フレームバッファの先頭アドレスが不正
   GLCDC_ERR_INVALID_GAMMA_SETTING,        // ガンマ補正の設定が不正
   GLCDC_ERR_INVALID_UPDATE_TIMING,        // レジスタ値の更新タイミングが不正
   GLCDC_ERR_INVALID_CLUT_ACCESS,          // CLUT メモリの設定が不正
   GLCDC_ERR_INVALID_BLEND_SETTING,        // ブレンドの設定が不正

} glcdc_err_t;
```

2.11　コールバック関数

GLCDC FIT モジュールでは、VPOS 割り込み、GR1UF 割り込み、GR2UF 割り込みが発生したタイミングで、ユーザが設定したコールバック関数を呼び出します。
コールバック関数は、「2.9 引数」に記載された構造体メンバ“p_callback” に、ユーザの関数のアドレスを格納することで設定されます。コールバック関数が呼び出されるとき、表 2.3 に示す定数が引数として渡されます。

引数の型は void ポインタ型で渡されるため、コールバック関数の引数は下記の例を参考に void 型のポインタ変数としてください。

コールバック関数内部で引数を使うときはキャストしてください。

なお、GLCDC ソフトウェアリセット解除後の初回のみ、意図しないグラフィック 2 指定ライン通知（VPOS フラグ）、グラフィック 1,2 アンダフロー（GR1UF フラグ、GR2UF フラグ）が検出されます。そのため、R_GLCDC_Open 関数実行後の初回の VPOS 割り込み処理では何もせず、次回の割り込み処理からユーザ処理を実行してください。

表 2.3　コールバック関数の引数一覧(enum glcdc_event_t)

定数定義	説明
GLCDC_EVENT_LINE_DETECTION	VPOS 割り込みの割り込み処理から呼ばれたコールバック関数
GLCDC_EVENT_GR1_UNDERFLOW	GR1UF 割り込みの割り込み処理から呼ばれたコールバック関数
GLCDC_EVENT_GR2_UNDERFLOW	GR2UF 割り込みの割り込み処理から呼ばれたコールバック関数

コールバック関数例：

```
bool first_interrupt_flag = false;

void my_glcdc_callback(void * pdata)
{
    if (false == first_interrupt_flag)
    {
        first_interrupt_flag = true;
        /* do nothing */
    }
    else
    {
        glcdc_callback_args_t     * pdecode;
        pdecode = (glcdc_callback_args_t *)pdata; //cast pointer to glcdc_callback_args_t
        ...
    }
}
```

2.12　FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e² studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e² studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: e² studio 編(R20AN0451)」を参照してください。

(2) e² studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e² studio の FIT コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: CS+編(R20AN0451)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.13　for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

「WAIT_LOOP」を記述している対象デバイス

・RX651, RX65N グループ

・RX72M グループ

・RX72N グループ

・RX66N グループ

以下に記述例を示します。

```
while 文の例：
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for 文の例：
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while 文の例：
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3.　API 関数

R_GLCDC_Open () < GLCDC 設定データ構造体で設定する場合>

この関数は、GLCDC FIT モジュールを初期化する関数です。この関数は他の API 関数を使用する前に実行される必要があります。

コンフィグレーションオプションの GLCDC_CFG_CONFIGURATION_MODE が“0”かつ QE for Display [RX] V2.0.0 以降を使用しない場合（define 定義“QE_DISPLAY_CONFIGURATION”が宣言されていない場合）の R_GLCDC_Open 関数の動作については本説明を参照してください。

Format

```
glcdc_err_t R_GLCDC_Open (
        glcdc_cfg_t * const       p_cfg       /* GLCDC 設定データ構造体のポインタ  */
)
```

Parameters

glcdc_cfg_t　　* p_cfg

GLCDC 設定データの構造体のポインタを設定してください。

参照する glcdc_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、本関数実行時に設定する必要はありません。

表 3.1　glcdc_cfg_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
output.htiming.back_porch	水平バックポーチ	5.1 画面の定義を参照	STHy 信号アサートタイミング、水平有効表示開始位置の設定
output.htiming.sync_width	水平アサート幅	5.1 画面の定義を参照	STHy 信号アサートタイミング、STHy 信号アサート幅、水平有効表示開始位置の設定
output.vtiming.back_porch	垂直バックポーチ	5.1 画面の定義を参照	STVy 信号アサートタイミング、垂直有効表示開始位置
output.vtiming.sync_width	垂直アサート幅	5.1 画面の定義を参照	STVy 信号アサートタイミング、STVy 信号アサート幅、垂直有効表示開始位置の設定
output.htiming.display_cyc	水平有効表示幅	5.1 画面の定義を参照	STHy 信号アサート幅、水平有効表示幅の設定
output.vtiming.display_cyc	垂直有効表示幅	5.1 画面の定義を参照	STVy 信号アサート幅、垂直有効表示幅の設定
output.htiming.front_porch	水平フロントポーチ	5.1 画面の定義を参照	水平有効表示幅、水平有効表示開始位置の設定
output.vtiming.front_porch	垂直フロントポーチ	5.1 画面の定義を参照	垂直有効表示幅、垂直有効表示開始位置の設定
p_callback	コールバック関数へのポインタ	コールバック関数へのアドレス	割り込み要因発生時にポインタが示すアドレスのコールバック関数を実行します
		FIT_NO_FUNC または NULL	要因が発生してもコールバック関数は実行されません

output.clksrc	クロックソース	GLCDC_CLK_SRC_INTERNAL	PLL クロックを使用
output.clock_div_ratio	クロックの分周比	1～32 分周(詳細は 2.9 引数 glcdc_panel_clk_div_t を参照してください)	LCD_CLK の分周比の設定
output.format	出力データフォーマット	GLCDC_OUT_FORMAT_24BITS_RGB888	出力データフォーマット、出力フォーマットを RGB(888)に設定ピクセルクロックを分周なしに設定
		GLCDC_OUT_FORMAT_18BITS_RGB666	出力データフォーマット、出力フォーマットを RGB(666)に設定ピクセルクロックを分周なしに設定
		GLCDC_OUT_FORMAT_16BITS_RGB565	出力データフォーマット、出力フォーマットを RGB(565)に設定ピクセルクロックを分周なしに設定
output.sync_edge	TCON、DATA の出力位相制御	GLCDC_SIGNAL_SYNC_EDGE_RISING	LCD_CLK の立ち上がりに同期して出力
		GLCDC_SIGNAL_SYNC_EDGE_FALLING	LCD_CLK の立ち下がりに同期して出力
output.tcon_hsync	水平同期信号(HSYNC)の出力端子	GLCDC_TCON_PIN_0	LCD_TCON0 端子を使用
		GLCDC_TCON_PIN_1	LCD_TCON1 端子を使用
		GLCDC_TCON_PIN_2	LCD_TCON2 端子を使用
		GLCDC_TCON_PIN_3	LCD_TCON3 端子を使用
		GLCDC_TCON_PIN_NON	HSYNC の出力を指定しない
output.hsync_polarity	水平同期信号(HSYNC)の極性	GLCDC_SIGNAL_POLARITY_LOACTIVE	極性をローアクティブに設定
		GLCDC_SIGNAL_POLARITY_HIACTIVE	極性をハイアクティブに設定
output.tcon_vsync	垂直同期信号(VSYNC)の出力端子	GLCDC_TCON_PIN_0	LCD_TCON0 端子を使用
		GLCDC_TCON_PIN_1	LCD_TCON1 端子を使用
		GLCDC_TCON_PIN_2	LCD_TCON2 端子を使用
		GLCDC_TCON_PIN_3	LCD_TCON3 端子を使用
		GLCDC_TCON_PIN_NON	VSYNC の出力を指定しない
output.vsync_polarity	垂直同期信号(VSYNC)の極性	GLCDC_SIGNAL_POLARITY_LOACTIVE	極性をローアクティブに設定
		GLCDC_SIGNAL_POLARITY_HIACTIVE	極性をハイアクティブに設定
output.tcon_de	データイネーブル信号(DE)の出力端子	GLCDC_TCON_PIN_0	LCD_TCON0 端子を使用
		GLCDC_TCON_PIN_1	LCD_TCON1 端子を使用
		GLCDC_TCON_PIN_2	LCD_TCON2 端子を使用
		GLCDC_TCON_PIN_3	LCD_TCON3 端子を使用
		GLCDC_TCON_PIN_NON	DE の出力を指定しない
output.data_enable_polarity	データイネーブル信号(DE)の極性	GLCDC_SIGNAL_POLARITY_LOACTIVE	極性をローアクティブに設定
		GLCDC_SIGNAL_POLARITY_HIACTIVE	極性をハイアクティブに設定
output.bg_color.byte.r	背景色 R 値	00h to FFh	背景色の R 値の設定
output.bg_color.byte.g	背景色 G 値	00h to FFh	背景色の G 値の設定

output.bg_color.byte.b	背景色 B 値	00h to FFh	背景色の B 値の設定
input.format	フレームバッファのデータフォーマット	GLCDC_IN_FORMAT_32BITS_ARGB8888	ARGB(8888)を使用
		GLCDC_IN_FORMAT_32BITS_RGB888	RGB(888)を使用
		GLCDC_IN_FORMAT_16BITS_RGB565	RGB(565)を使用
		GLCDC_IN_FORMAT_16BITS_ARGB1555	ARGB(1555)を使用
		GLCDC_IN_FORMAT_16BITS_ARGB4444	ARGB(4444)を使用
		GLCDC_IN_FORMAT_CLUT8	CLUT(8)を使用
		GLCDC_IN_FORMAT_CLUT4	CLUT(4)を使用
		GLCDC_IN_FORMAT_CLUT1	CLUT(1)を使用
input.p_base	フレームバッファの先頭アドレス	0000 0040h to FFFF FFC0h 下位 6 ビットは 0	フレームバッファの先頭アドレスの設定
		NULL	対象グラフィックを無効に設定（glcdc_cfg_t.input 以下の構造体メンバの設定値は無視されます）
input.bg_color.byte.r	グラフィック 1,2 の背景色 R 値	00h to FFh	グラフィック 1,2 の背景色の R 値の設定
input.bg_color.byte.g	グラフィック 1,2 の背景色 G 値	00h to FFh	グラフィック 1,2 の背景色の G 値の設定
input.bg_color.byte.b	グラフィック 1,2 の背景色 B 値	00h to FFh	グラフィック 1,2 の背景色の B 値の設定
input.hsize	画像データの横幅	5.1 画面の定義を参照	グラフィック 1,2 の画像の横幅の設定
input.vsize	画像データの高さ	5.1 画面の定義を参照	グラフィック 1,2 の画像の高さの設定
input.offset	マクロラインオフセット	-32768 to 32704 (64 の倍数)	グラフィック 1,2 のマクロラインオフセットの設定
input.frame_edge	グラフィック領域の枠の表示	true	グラフィック領域枠を表示に設定
		false	グラフィック領域枠を非表示に設定
input.coordinate.x	表示開始位置 x 座標	5.1 画面の定義を参照	グラフィック領域水平開始位置の設定
input.coordinate.y	表示開始位置 y 座標	5.1 画面の定義を参照	グラフィック領域垂直開始位置の設定
blend.blend_control	ブレンド処理の制御設定	GLCDC_BLEND_CONTROL_NONE	アルファブレンド処理を無効に設定
		GLCDC_BLEND_CONTROL_FADEIN	フェードインに設定
		GLCDC_BLEND_CONTROL_FADEOUT	フェードアウトに設定
		GLCDC_BLEND_CONTROL_FIXED	アルファ値固定に設定
		GLCDC_BLEND_CONTROL_PIXEL	ピクセル単位アルファブレンドに設定

blend.visible	画像の表示設定	true	画像を表示に設定
		false	画像を非表示に設定
blend. frame_edge	矩形アルファブレンド領域の枠の表示	true	矩形アルファブレンド領域の枠を表示に設定
		false	矩形アルファブレンド領域の枠を非表示に設定
blend.fixed_ blend_value	固定アルファ値	00h to FFh	固定アルファ値の設定（blend_control が GLCDC_BLEND_CONTROL_FIXED のときのみ有効）
blend. fade_speed	アルファ値の加減値	00h to FFh	アルファ値の加減値の設定（blend_control が GLCDC_BLEND_ CONTROL_FADEIN もしくは GLCDC_BLEND_CONTROL_FADEOUT のときのみ有効）
blend.start_ coordinate.x	ブレンド処理開始位置の x 座標	5.1 画面の定義を参照	矩形アルファブレンド領域水平幅、矩形アルファブレンド水平開始位置を設定
blend.end_ coordinate.x	ブレンド処理終了位置の x 座標	5.1 画面の定義を参照	
blend.start_ coordinate.y	ブレンド処理の開始位置の y 座標	5.1 画面の定義を参照	矩形アルファブレンド領域垂直幅、矩形アルファブレンド垂直開始位置を設定
blend.end_ coordinate.y	ブレンド処理の終了位置の y 座標	5.1 画面の定義を参照	
chromakey. enable	クロマキー 処理の有効、無効	true	クロマキー処理を有効に設定
		false	クロマキー処理を無効に設定（glcdc_cfg_t.chromakey 以下の構造体メンバの設定値は無視されます）
chromakey. before.byte.r	クロマキー処理対象 R 値	00h to FFh	クロマキー処理対象 R 値の設定
chromakey. before.byte.g	クロマキー処理対象 G 値	00h to FFh	クロマキー処理対象 G 値の設定
chromakey. before.byte.b	クロマキー処理対象 B 値	00h to FFh	クロマキー処理対象 B 値の設定
chromakey. after.byte.a	クロマキー置き換え後 A 値	00h to FFh	クロマキー処理で置き換えた後の A 値を設定
chromakey. after.byte.r	クロマキー置き換え後 R 値	00h to FFh	クロマキー処理で置き換えた後の R 値を設定
chromakey. after.byte.g	クロマキー置き換え後 G 値	00h to FFh	クロマキー処理で置き換えた後の G 値を設定
chromakey. after.byte.b	クロマキー置き換え後 B 値	00h to FFh	クロマキー処理で置き換えた後の B 値を設定
output.endian	出力データのビットエンディアン	GLCDC_ENDIAN_LITTLE	リトルエンディアンに設定
		GLCDC_ENDIAN_BIG	ビッグエンディアンに設定
output. color_order	出力データのピクセル順序	GLCDC_COLOR_ ORDER_RGB	出力データのピクセル順序を R-G-B 順に設定
		GLCDC_COLOR_ ORDER_BGR	出力データのピクセル順序を B-G-R 順に設定

output.correction_proc_order	補正処理の実行順序	GLCDC_BRIGHTNESS_CONTRAST_TO_GAMMA	輝度/コントラスト補正の後にガンマ補正を実行するように設定
		GLCDC_GAMMA_TO_BRIGHTNESS_CONTRAST	ガンマ補正の後に輝度/コントラスト補正を実行するように設定
output.dithering.dithering_on	ディザ処理のモード選択	true	0 捨 1 入または 2x2 パターンディザに設定
		false	切り捨てモードに設定（glcdc_cfg_t.output.dithering 以下の構造体メンバの設定値は無視されます）
output.dithering.dithering_mode	ディザ処理のモード選択 2	GLCDC_DITHERING_MODE_TRUNCATE	切り捨てモードに設定
		GLCDC_DITHERING_MODE_ROUND_OFF	0 捨 1 入モードに設定
		GLCDC_DITHERING_MODE_2X2PATTERN	2x2 パターンディザに設定
output.dithering.dithering_pattern_a	ディザパターン値 A	GLCDC_DITHERING_PATTERN_00	2x2 パターンディザのパターン値 A の設定（dithering_mode が GLCDC_DITHERING_MODE_2X2PATTERN のときのみ有効）
		GLCDC_DITHERING_PATTERN_01	
		GLCDC_DITHERING_PATTERN_10	
		GLCDC_DITHERING_PATTERN_11	
output.dithering.dithering_pattern_b	ディザパターン値 B	GLCDC_DITHERING_PATTERN_00	2x2 パターンディザのパターン値 B の設定（dithering_mode が GLCDC_DITHERING_MODE_2X2PATTERN のときのみ有効）
		GLCDC_DITHERING_PATTERN_01	
		GLCDC_DITHERING_PATTERN_10	
		GLCDC_DITHERING_PATTERN_11	
output.dithering.dithering_pattern_c	ディザパターン値 C	GLCDC_DITHERING_PATTERN_00	2x2 パターンディザのパターン値 C の設定（dithering_mode が GLCDC_DITHERING_MODE_2X2PATTERN のときのみ有効）
		GLCDC_DITHERING_PATTERN_01	
		GLCDC_DITHERING_PATTERN_10	
		GLCDC_DITHERING_PATTERN_11	
output.dithering.dithering_pattern_d	ディザパターン値 D	GLCDC_DITHERING_PATTERN_00	2x2 パターンディザのパターン値 D の設定（dithering_mode が GLCDC_DITHERING_MODE_2X2PATTERN のときのみ有効）
		GLCDC_DITHERING_PATTERN_01	
		GLCDC_DITHERING_PATTERN_10	
		GLCDC_DITHERING_PATTERN_11	
output.brightness.enable	輝度補正の有効、無効	true	輝度補正を有効に設定
		false	輝度補正を無効に設定（glcdc_cfg_t.output.brightness 以下の構造体メンバの設定値に依ら

			ず、 RGB 信号の輝度調整値に 0 が設定されます）
output. brightness.r	R 信号の輝度調整値	000h：-512 : 200h：0 : 3FFh：+511	R 信号の輝度調整値の設定
output. brightness.g	G 信号の輝度調整値		G 信号の輝度調整値の設定
output. brightness.b	B 信号の輝度調整値		B 信号の輝度調整値の設定
output.contrast. enable	コントラスト補正有効、無効	true	コントラスト補正を有効に設定
		false	コントラスト補正を無効に設定 （glcdc_cfg_t.output.contrast 以下の構造体メンバの設定値に依らず、 RGB 信号のコントラスト調整値に 1.000 が設定されます）
output. contrast.r	R 信号のコントラスト調整値	00h：0/128 = 0.000 : 80h：128/128 = 1.000 : FFh：255/128 = 1.992	R 信号のコントラスト調整値を設定
output. contrast.g	G 信号のコントラスト調整値		G 信号のコントラスト調整値を設定
output. contrast.b	B 信号のコントラスト調整値		B 信号のコントラスト調整値を設定
output.gamma. enable	ガンマ補正の有効、無効	true	ガンマ補正を有効に設定
		false	ガンマ補正を無効に設定 （glcdc_cfg_t.output.gamma 以下の構造体メンバの設定値は無視されます）
output.gamma. p_r	R 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	R 信号の各領域のゲインと開始しきい値の設定
output.gamma. p_g	G 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	G 信号の各領域のゲインと開始しきい値の設定
output.gamma. p_b	B 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	B 信号の各領域のゲインと開始しきい値の設定
clut.enable	CLUT メモリの更新の有効、無効の選択	true	CLUT メモリを更新します
		false	CLUT メモリを更新しません （glcdc_cfg_t.clut 以下の構造体メンバの設定値は無視されます）
clut.p_base	CLUT メモリの先頭アドレスへのポインタ	NULL 以外	ポインタが指し示すアドレスから値を読み出し CLUT メモリにコピーします
clut.start	更新する CLUT メモリの開始エントリ番号	0 to 255 (ただし、 start + size < 257)	指定したエントリ番号から CLUT メモリの更新を開始します
clut.size	更新する CLUT メモリのエントリサイズ	1 to 256 (ただし、 start + size < 257)	指定したサイズ分の CLUT メモリを更新します
detection.vpos_ detect	VPOS 検出の許可、禁止	true	VPOS 検出を許可に設定
		false	VPOS 検出を禁止に設定
detection.gr1uf_ detect	GR1UF 検出の許可、禁止	true	GR1UF 検出を許可に設定
		false	GR1UF 検出を禁止に設定
		true	GR2UF 検出を許可に設定

detection.gr2uf_detect	GR2UF 検出の許可、禁止	false	GR2UF 検出を禁止に設定
interrupt.vpos_enable	VPOS 割り込みの許可、禁止	true	VPOS 割り込みを許可に設定
		false	VPOS 割り込みを禁止に設定
interrupt.gr1uf_enable	GR1UF 割り込みの許可、禁止	true	GR1UF 割り込みを許可に設定
		false	GR1UF 割り込みを禁止に設定
interrupt.gr2uf_enable	GR2UF 割り込みの許可、禁止	true	GR2UF 割り込みを許可に設定
		false	GR2UF 割り込みを禁止に設定

Return Values

```
GLCDC_SUCCESS                   /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR           /* 引数 p_cfg が NULL ポインタの場合 */
GLCDC_ERR_LOCK_FUNC             /* GLCDC がロック済みの場合 */
GLCDC_ERR_INVALID_ARG           /* GLCDC 設定データのパラメータが不正の場合 */
GLCDC_ERR_INVALID_MODE          /* 関数が実行できないモードである場合 */
GLCDC_ERR_INVALID_TIMING_SETTING  /* パネル出力信号のタイミングの設定が不正の場合 */
GLCDC_ERR_INVALID_LAYER_SETTING   /* グラフィック画面の設定が不正の場合 */
GLCDC_ERR_INVALID_ALIGNMENT       /* フレームバッファの先頭アドレスが不正の場合 */
GLCDC_ERR_INVALID_GAMMA_SETTING   /* ガンマの設定が不正の場合 */
GLCDC_ERR_INVALID_CLUT_ACCESS     /* CLUT メモリの設定が不正の場合 */
GLCDC_ERR_INVALID_BLEND_SETTING   /* ブレンドの設定が不正の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC を使用するために、GLCDC のモジュールストップ、ソフトウェアリセットを解除します。その後、パネルクロック、パネル出力信号のタイミング、バックグラウンド画面、グラフィック画面、CLUT メモリ、出力データフォーマット、補正処理、GLCDC で使用する割り込みを設定します。

本関数はモード「GLCDC_STATE_CLOSED」のときに実行できます。本関数の処理が正常に完了した場合、モード「GLCDC_STATE_NOT_DISPLAYING」に遷移します。

Example

```
volatile glcdc_err_t     ret_glcdc;
glcdc_cfg_t      p_cfg;

p_cfg.htiming.back_porch = 2;
...    // Set arguments parameter.
p_cfg.interrupt.gr2uf_enable = true;

ret_glcdc = R_GLCDC_Open(&p_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

- 本関数で p_base に NULL を設定して対象グラフィック画面を無効に設定した場合

R_GLCDC_LayerChange 関数でのグラフィック画面の設定、R_GLCDC_ClutUpdate 関数での CLUT メモリの更新は無効になります。無効にしたグラフィック画面を有効にする場合は、再度 R_GLCDC_Open 関数を実行して対象グラフィック画面を有効に設定してください。

- マクロラインオフセット設定時の注意事項

ハードウェアの仕様上、フレームバッファから 64 バイトごとにデータを読み出しているため、構造体メンバ input.offset（マクロラインオフセット）は、64 の倍数に設定してください。制限を守ることができない場合、「5.5 マクロラインオフセットの制限を守ることができない場合」を参照してください。

- glcdc_cfg_t 構造体変数使用時の注意事項

glcdc_cfg_t 構造体変数は static 宣言の付加またはグローバル変数として定義してください。Auto 変数にしてしまうとスタックが不足する可能性があります。

R_GLCDC_Open () <コンフィグレーションオプションで設定する場合>

この関数は、GLCDC FIT モジュールを初期化する関数です。この関数は他の API 関数を使用する前に実行される必要があります。

コンフィグレーションオプションの GLCDC_CFG_CONFIGURATION_MODE が“1”または QE for Display [RX] V2.0.0 以降を使用する場合（define 定義“QE_DISPLAY_CONFIGURATION”が宣言されている場合）の R_GLCDC_Open 関数の動作については本説明を参照してください。

Format

```
glcdc_err_t R_GLCDC_Open (
        glcdc_cfg_t * const     p_cfg       /* GLCDC 設定データ構造体のポインタ */
)
```

Parameters

glcdc_cfg_t　　* p_cfg

GLCDC 設定データ構造体のポインタを設定してください。本関数実行時にコンフィグレーションオプションの設定値が格納されます。

Return Values

```
GLCDC_SUCCESS                   /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR           /* 引数 p_cfg が NULL ポインタの場合 */
GLCDC_ERR_LOCK_FUNC             /* GLCDC がロック済みの場合 */
GLCDC_ERR_INVALID_ARG           /* GLCDC 設定データのパラメータが不正の場合 */
GLCDC_ERR_INVALID_MODE          /* 関数が実行できないモードである場合 */
GLCDC_ERR_INVALID_TIMING_SETTING  /* パネル出力信号のタイミングの設定が不正の場合 */
GLCDC_ERR_INVALID_LAYER_SETTING   /* グラフィック画面の設定が不正の場合 */
GLCDC_ERR_INVALID_ALIGNMENT       /* フレームバッファの先頭アドレスが不正の場合 */
GLCDC_ERR_INVALID_GAMMA_SETTING   /* ガンマの設定が不正の場合 */
GLCDC_ERR_INVALID_CLUT_ACCESS     /* CLUT メモリの設定が不正の場合 */
GLCDC_ERR_INVALID_BLEND_SETTING   /* ブレンドの設定が不正の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

コンフィグレーションオプションの GLCDC_CFG_CONFIGURATION_MODE が“1”の場合、r_glcdc_rx_config.h に定義されているコンフィグレーションオプションを参照して GLCDC FIT モジュールの設定を行います。

QE for Display[RX] V2.0.0 以降を使用する場合（define 定義“QE_DISPLAY_CONFIGURATION”が宣言されている場合）は、r_glcdc_rx_config.h と QE for Display[RX] が生成したヘッダファイル（r_lcd_timing.h、r_image_config.h）に定義されているコンフィグレーションオプションを参照して GLCDC FIT モジュールの設定を行います。r_glcdc_rx_config.h と QE for Display[RX] が生成したヘッダファイル（r_lcd_timing.h、r_image_config.h）の両方に存在する定義については、QE for Display[RX] が生

成したヘッダファイル（r_lcd_timing.h、r_image_config.h）の定義が有効になります。

コンフィグレーションオプションの設定は GLCDC 設定データの構造体メンバにそれぞれ対応（LCD_CH0_CALLBACK_ENABLE を除きます）しています。本関数が実行されるとコンフィグレーションオプションの設定値が引数（p_cfg）で指定された構造体メンバに格納されます。

表 3.2　GLCDC 設定データの構造体メンバとコンフィグレーションオプションの対応

概略	構造体メンバ	QE for Display[RX]が生成する define 定義	r_glcdc_rx_config.h の define 定義
水平バックポーチ	output.htiming.back_porch	LCD_CH0_W_HBP	LCD_CH0_W_HBP ※デフォルト値は “62”
水平アサート幅	output.htiming.sync_width	LCD_CH0_W_HSYNC	LCD_CH0_W_HSYNC ※デフォルト値は “25”
垂直バックポーチ	output.vtiming.back_porch	LCD_CH0_W_VBP	LCD_CH0_W_VBP ※デフォルト値は “7”
垂直アサート幅	output.vtiming.sync_width	LCD_CH0_W_VSYNC	LCD_CH0_W_VSYNC ※デフォルト値は “1”
水平有効表示幅	output.htiming.display_cyc	LCD_CH0_DISP_HW	LCD_CH0_DISP_HW ※デフォルト値は “480”
垂直有効表示幅	output.vtiming.display_cyc	LCD_CH0_DISP_VW	LCD_CH0_DISP_VW ※デフォルト値は “272”
水平フロントポーチ	output.htiming.front_porch	LCD_CH0_W_HFP	LCD_CH0_W_HFP ※デフォルト値は “17”
垂直フロントポーチ	output.vtiming.front_porch	LCD_CH0_W_VFP	LCD_CH0_W_VFP ※デフォルト値は “8”
クロックソース	output.clksrc	— ※FIT モジュール内で固定値（GLCDC_CLK_SRC_INTERNAL）が設定されます。	
クロックの分周比	output.clock_div_ratio	LCD_CH0_OUT_CLK_DIV_RATIO	LCD_CH0_OUT_CLK_DIV_RATIO ※デフォルト値は “GLCDC_PANEL_CLK_DIVISOR_24”
出力データフォーマット	output.format	LCD_CH0_OUT_FORMAT	LCD_CH0_OUT_FORMAT ※デフォルト値は “GLCDC_OUT_FORMAT_16BITS_RGB565”
出力データのビットエンディアン	output.endian	LCD_CH0_OUT_ENDIAN	LCD_CH0_OUT_ENDIAN ※デフォルト値は “GLCDC_ENDIAN_LITTLE”
出力データのピクセル順序	output.color_order	LCD_CH0_OUT_COLOR_ORDER	LCD_CH0_OUT_COLOR_ORDER ※デフォルト値は “GLCDC_COLOR_ORDER_RGB”
TCON、DATA の出力位相制御	output.sync_edge	LCD_CH0_OUT_EDGE	LCD_CH0_OUT_EDGE ※デフォルト値は “GLCDC_SIGNAL_SYNC_EDGE_RISING”
水平同期信号(HSYNC)の出力端子	output.tcon_hsync	LCD_CH0_TCON_PIN_HSYNC	LCD_CH0_TCON_PIN_HSYNC

			※デフォルト値は“GLCDC_TCON_PIN_2”
水平同期信号(HSYNC)の極性	output.hsync_polarity	LCD_CH0_TCON_POL_HSYNC	LCD_CH0_TCON_POL_HSYNC ※デフォルト値は“GLCDC_SIGNAL_POLARITY_LOACTIVE”
垂直同期信号(VSYNC)の出力端子	output.tcon_vsync	LCD_CH0_TCON_PIN_VSYNC	LCD_CH0_TCON_PIN_VSYNC ※デフォルト値は“GLCDC_TCON_PIN_0”
垂直同期信号(VSYNC)の極性	output.vsync_polarity	LCD_CH0_TCON_POL_VSYNC	LCD_CH0_TCON_POL_VSYNC ※デフォルト値は“GLCDC_SIGNAL_POLARITY_LOACTIVE”
データイネーブル信号(DE)の出力端子	output.tcon_de	LCD_CH0_TCON_PIN_DE	LCD_CH0_TCON_PIN_DE ※デフォルト値は“GLCDC_TCON_PIN_3”
データイネーブル信号(DE)の極性	output.data_enable_polarity	LCD_CH0_TCON_POL_DE	LCD_CH0_TCON_POL_DE ※デフォルト値は“GLCDC_SIGNAL_POLARITY_HIACTIVE”
背景色	output.bg_color.argb	LCD_CH0_OUT_BG_COLOR	LCD_CH0_OUT_BG_COLOR ※デフォルト値は“0x00000000”
フレームバッファの画像フォーマット	input.format	LCD_CH0_IN_GR2_FORMAT LCD_CH0_IN_GR1_FORMAT	LCD_CH0_IN_GR2_FORMAT LCD_CH0_IN_GR1_FORMAT ※デフォルト値は“GLCDC_IN_FORMAT_16BITS_RGB565”
フレームバッファの先頭アドレス	input.p_base	LCD_CH0_IN_GR2_PBASE LCD_CH0_IN_GR1_PBASE	LCD_CH0_IN_GR2_PBASE ※デフォルト値は“0x00800000” LCD_CH0_IN_GR1_PBASE ※デフォルト値は “NULL”
グラフィック 1,2 の背景色 RGB 値	input.bg_color.argb	—	LCD_CH0_IN_GR2_BG_COLOR LCD_CH0_IN_GR1_BG_COLOR ※デフォルト値は“0x00000000”
画像データの横幅	input.hsize	LCD_CH0_IN_GR2_HSIZE LCD_CH0_IN_GR1_HSIZE	LCD_CH0_IN_GR2_HSIZE LCD_CH0_IN_GR1_HSIZE ※デフォルト値は “480”
画像のデータの高さ	input.vsize	LCD_CH0_IN_GR2_VSIZE LCD_CH0_IN_GR1_VSIZE	LCD_CH0_IN_GR2_VSIZE LCD_CH0_IN_GR1_VSIZE ※デフォルト値は “272”
マクロラインオフセット	input.offset	LCD_CH0_IN_GR2_LINEOFFSET LCD_CH0_IN_GR1_LINEOFFSET	LCD_CH0_IN_GR2_LINEOFFSET LCD_CH0_IN_GR1_LINEOFFSET ※デフォルト値は “960”
グラフィック領域の枠の表示	input.frame_edge	—	LCD_CH0_IN_GR2_FRAME_EDGE LCD_CH0_IN_GR1_FRAME_EDGE

			※デフォルト値は “false”
表示開始位置 x 座標	input. coordinate.x	LCD_CH0_IN_GR2_COORD_X LCD_CH0_IN_GR1_COORD_X	LCD_CH0_IN_GR2_COORD_X LCD_CH0_IN_GR1_COORD_X ※デフォルト値は “0”
表示開始位置 y 座標	input. coordinate.y	LCD_CH0_IN_GR2_COORD_Y LCD_CH0_IN_GR1_COORD_Y	LCD_CH0_IN_GR2_COORD_Y LCD_CH0_IN_GR1_COORD_Y ※デフォルト値は “0”
ブレンド処理の制御設定	blend. blend_control	—	LCD_CH0_BLEND_GR2_BLEND_CONTROL LCD_CH0_BLEND_GR1_BLEND_CONTROL ※デフォルト値は “GLCDC_BLEND_CONTROL_NONE”
画像の表示設定	blend. visible	—	LCD_CH0_BLEND_GR2_VISIBLE LCD_CH0_BLEND_GR1_VISIBLE ※デフォルト値は “true”
矩形アルファブレンド領域の枠の表示	blend. frame_edge	—	LCD_CH0_BLEND_GR2_FRAME_EDGE LCD_CH0_BLEND_GR1_FRAME_EDGE ※デフォルト値は “false”
固定アルファ値	blend. fixed_blend_value	—	LCD_CH0_BLEND_GR2_FIXED_BLEND_VALUE LCD_CH0_BLEND_GR1_FIXED_BLEND_VALUE ※デフォルト値は “255”
アルファ値の加減値	blend. fade_speed	—	LCD_CH0_BLEND_GR2_FADE_SPEED LCD_CH0_BLEND_GR1_FADE_SPEED ※デフォルト値は “255”
ブレンド処理開始位置の x 座標	blend. start_coordinate.x	—	LCD_CH0_BLEND_GR2_START_COORD_X LCD_CH0_BLEND_GR1_START_COORD_X ※デフォルト値は “0”
ブレンド処理終了位置の x 座標	blend. end_coordinate.x	—	LCD_CH0_BLEND_GR2_END_COORD_X LCD_CH0_BLEND_GR1_END_COORD_X ※デフォルト値は “0”
ブレンド処理の開始位置の y 座標	blend. start_coordinate.y	—	LCD_CH0_BLEND_GR2_START_COORD_Y LCD_CH0_BLEND_GR1_START_COORD_Y ※デフォルト値は “0”
ブレンド処理の終了位置の y 座標	blend. end_coordinate.y	—	LCD_CH0_BLEND_GR2_END_COORD_Y LCD_CH0_BLEND_GR1_END_COORD_Y

			※デフォルト値は “0”
クロマキー 処理の有効、無効	chromakey. enable	—	LCD_CH0_CHROMAKEY_GR2 _ENABLE LCD_CH0_CHROMAKEY_GR1 _ENABLE ※デフォルト値は “false”
クロマキー処理対象	chromakey. before.argb	—	LCD_CH0_CHROMAKEY_GR2 _BEFORE_ARGB LCD_CH0_CHROMAKEY_GR1 _BEFORE_ARGB ※デフォルト値は “0x00000000”
クロマキー置き換え後	chromakey. after.argb	—	LCD_CH0_CHROMAKEY_GR2 _AFTER_ARGB LCD_CH0_CHROMAKEY_GR1 _AFTER_ARGB ※デフォルト値は “0x00000000”
補正処理の実行順序	output. correction_proc_order	IMGC_OUTCTL_CALIB_ROU TE	IMGC_OUTCTL_CALIB_ROUT E ※デフォルト値は “GLCDC_BRIGHTNESS_CON TRAST_TO_GAMMA”
ディザ処理のモード選択	output.dithering. dithering_on	IMGC_DITHER_ACTIVE	IMGC_DITHER_ACTIVE ※デフォルト値は “false”
ディザ処理のモード選択 2	output.dithering. dithering_mode	IMGC_DITHER_MODE	IMGC_DITHER_MODE ※デフォルト値は “GLCDC_DITHERING_MODE _TRUNCATE”
ディザパターン値 A	output.dithering. dithering_pattern_a	IMGC_DITHER_2X2_PA	IMGC_DITHER_2X2_PA ※デフォルト値は “GLCDC_DITHERING_PATTE RN_11”
ディザパターン値 B	output.dithering. dithering_pattern_b	IMGC_DITHER_2X2_PB	IMGC_DITHER_2X2_PB ※デフォルト値は “GLCDC_DITHERING_PATTE RN_00”
ディザパターン値 C	output.dithering. dithering_pattern_c	IMGC_DITHER_2X2_PC	IMGC_DITHER_2X2_PC ※デフォルト値は “GLCDC_DITHERING_PATTE RN_10”
ディザパターン値 D	output.dithering. dithering_pattern_d	IMGC_DITHER_2X2_PD	IMGC_DITHER_2X2_PD ※デフォルト値は “GLCDC_DITHERING_PATTE RN_01”
輝度補正の有効、無効	output. brightness.enable	IMGC_BRIGHT_OUTCTL_AC TIVE ※QE for Display [RX]では、常に true が設定されます。	IMGC_BRIGHT_OUTCTL_ACTI VE ※デフォルト値は “true”
R 信号の輝度調整値	output. brightness.r	IMGC_BRIGHT_OUTCTL_OF FSET_R	IMGC_BRIGHT_OUTCTL_OFF SET_R ※デフォルト値は “512”
G 信号の輝度調整値	output. brightness.g	IMGC_BRIGHT_OUTCTL_OF FSET_G	IMGC_BRIGHT_OUTCTL_OFF SET_G ※デフォルト値は “512”

B 信号の輝度調整値	output. brightness.b	IMGC_BRIGHT_OUTCTL_OFFSET_B	IMGC_BRIGHT_OUTCTL_OFFSET_B ※デフォルト値は “512”
コントラスト補正有効、無効	output. contrast.enable	IMGC_CONTRAST_OUTCTL_ACTIVE ※QE for Display [RX]では、常に true が設定されます。	IMGC_CONTRAST_OUTCTL_ACTIVE ※デフォルト値は “true”
R 信号のコントラスト調整値	output. contrast.r	IMGC_CONTRAST_OUTCTL_GAIN_R	IMGC_CONTRAST_OUTCTL_GAIN_R ※デフォルト値は “128”
G 信号のコントラスト調整値	output. contrast.g	IMGC_CONTRAST_OUTCTL_GAIN_G	IMGC_CONTRAST_OUTCTL_GAIN_G ※デフォルト値は “128”
B 信号のコントラスト調整値	output. contrast.b	IMGC_CONTRAST_OUTCTL_GAIN_B	IMGC_CONTRAST_OUTCTL_GAIN_B ※デフォルト値は “128”
ガンマ補正の有効、無効	output. gamma.enable	IMGC_GAMMA_ACTIVE ※QE for Display [RX]では、常に true が設定されます。	IMGC_GAMMA_ACTIVE ※デフォルト値は “true”
R 信号のガンマ補正テーブル	output. gamma.p_r	・gain[16] IMGC_GAMMA_R_GAIN_00 ～ IMGC_GAMMA_R_GAIN_15 ・Threshold[15] IMGC_GAMMA_R_TH_01～ IMGC_GAMMA_R_TH_15	・gain[16] IMGC_GAMMA_R_GAIN_00～ IMGC_GAMMA_R_GAIN_15 ・Threshold[15] IMGC_GAMMA_R_TH_01～ IMGC_GAMMA_R_TH_15 ※デフォルト値はガンマ補正 1.1 の値（5.2 ガンマ補正値の計算方法 参照）
G 信号のガンマ補正テーブル	output. gamma.p_g	・gain[16] IMGC_GAMMA_G_GAIN_00 ～ IMGC_GAMMA_G_GAIN_15 ・Threshold[15] IMGC_GAMMA_G_TH_01～ IMGC_GAMMA_G_TH_15	・gain[16] IMGC_GAMMA_G_GAIN_00～ IMGC_GAMMA_G_GAIN_15 ・Threshold[15] IMGC_GAMMA_G_TH_01～ IMGC_GAMMA_G_TH_15 ※デフォルト値はガンマ補正 1.1 の値（5.2 ガンマ補正値の計算方法 参照）
B 信号のガンマ補正テーブル	output. gamma.p_b	・gain[16] IMGC_GAMMA_B_GAIN_00 ～ IMGC_GAMMA_B_GAIN_15 ・Threshold[15] IMGC_GAMMA_B_TH_01～ IMGC_GAMMA_B_TH_15	・gain[16] IMGC_GAMMA_B_GAIN_00～ IMGC_GAMMA_B_GAIN_15 ・Threshold[15] IMGC_GAMMA_B_TH_01～ IMGC_GAMMA_B_TH_15 ※デフォルト値はガンマ補正 1.1 の値（5.2 ガンマ補正値の計算方法 参照）
CLUT メモリの更新の有効、無効の選択	clut.enable	—	LCD_CH0_CLUT_GR2_ENABLE LCD_CH0_CLUT_GR1_ENABLE ※デフォルト値は “false”
CLUT メモリの先頭アドレスへのポインタ	clut.p_base	—	LCD_CH0_CLUT_GR2_PBASE LCD_CH0_CLUT_GR1_PBASE ※デフォルト値は “FIT_NO_PTR”

			※CLUT メモリを使用する場合は本定義の設定と合わせて、LCD_CH0_CLUT_GRx_ENABLE を“true”に設定してください。 その際、本定義に FIT_NO_PTR を設定しないでください。
更新する CLUT メモリの開始エントリ番号	clut.start	—	LCD_CH0_CLUT_GR2_START LCD_CH0_CLUT_GR1_START ※デフォルト値は “0”
更新する CLUT メモリのエントリサイズ	clut.size	—	LCD_CH0_CLUT_GR2_SIZE LCD_CH0_CLUT_GR1_SIZE ※デフォルト値は “256”
VPOS 検出の許可、禁止	detection. vpos_detect	LCD_CH0_DETECT_VPOS	LCD_CH0_DETECT_VPOS ※デフォルト値は “false”
GR1UF 検出の許可、禁止	detection. gr1uf_detect	—	LCD_CH0_DETECT_GR1UF ※デフォルト値は “false”
GR2UF 検出の許可、禁止	detection. gr2uf_detect	—	LCD_CH0_DETECT_GR2UF ※デフォルト値は “false”
VPOS 割り込みの許可、禁止	interrupt. vpos_enable	LCD_CH0_INTERRUPT_VPOS_ENABLE	LCD_CH0_INTERRUPT_VPOS_ENABLE ※デフォルト値は “false”
GR1UF 割り込みの許可、禁止	interrupt. gr1uf_enable	—	LCD_CH0_INTERRUPT_GR1UF_ENABLE ※デフォルト値は “false”
GR2UF 割り込みの許可、禁止	interrupt. gr2uf_enable	—	LCD_CH0_INTERRUPT_GR2UF_ENABLE ※デフォルト値は “false”
コールバック関数へのポインタ	p_callback	LCD_CH0_PCALLBACK ※コールバック関数を使用する場合は本定義の設定と合わせて、LCD_CH0_CALLBACK_ENABLE に“true”が設定されます。	LCD_CH0_PCALLBACK ※デフォルト値は “glcdc_callback” ※コールバック関数を使用する場合は本定義の設定と合わせて、LCD_CH0_CALLBACK_ENABLE に“true”を設定してください。

- ガンマ補正テーブルについて

コンフィグレーションオプションの GLCDC_CFG_CONFIGURATION_MODE が“1”または QE for Display[RX] V2.0.0 以降を使用する場合（define 定義“QE_DISPLAY_CONFIGURATION”が宣言されている場合）、以下の RGB のガンマ補正テーブルが GLCDC FIT モジュール内で定義されます。ガンマ補正テーブルの各値には、QE for Display[RX]が生成した define 定義が反映されており、r_glcdc_rx_if.h ファイルのインクルードによってユーザからも参照できます。また、R_GLCDC_Open 関数実行後に構造体メンバ output.gamma.p_r、output.gamma.p_g、output.gamma.p_b には、各ガンマテーブルへのポインタが格納されます。

<r_glcdc_rx_if.h ファイルで extern 宣言されているガンマ補正テーブル>
```
extern const gamma_correction_t g_glcdc_gamma_table_r;
extern const gamma_correction_t g_glcdc_gamma_table_g;
extern const gamma_correction_t g_glcdc_gamma_table_b;
```

- コールバック関数の設定（LCD_CH0_CALLBACK_ENABLE と LCD_CH0_PCALLBACK）について

コンフィグレーションオプションの GLCDC_CFG_CONFIGURATION_MODE が“1”かつ QE for Display[RX] V2.0.0 以降を使用しない場合（define 定義“QE_DISPLAY_CONFIGURATION”が宣言されていない場合）、コンフィグレーションオプションの LCD_CH0_CALLBACK_ENABLE と LCD_CH0_PCALLBACK でコールバック関数を設定します。

LCD_CH0_CALLBACK_ENABLE を“true”に設定すると、LCD_CH0_PCALLBACK の設定が有効になりますので、LCD_CH0_PCALLBACK にコールバック関数名を設定してください。コールバック関数名に FIT_NO_FUNC または NULL を設定しないでください。

LCD_CH0_CALLBACK_ENABLE を“false”に設定すると、LCD_CH0_PCALLBACK の設定が無効になります。その場合、コールバック関数へのポインタ（p_callback）には、FIT_NO_FUNC が格納されます。

<コールバック関数の設定>
```
#define LCD_CH0_CALLBACK_ENABLE    (true)
#define LCD_CH0_PCALLBACK          (my_glcdc_callback)
```

Example
```
volatile glcdc_err_t     ret_glcdc;
glcdc_cfg_t      p_cfg;

// Parameter settings are made using configuration options.

ret_glcdc = R_GLCDC_Open(&p_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
// After executing the R_GLCDC_Open function, the value set by the configuration option
// is stored in p_cfg.
```

Special Notes:
GLCDC 設定データ構造体で設定する場合の R_GLCDC_Open 関数と同等です。

R_GLCDC_Close ()

GLCDC の動作を終了する関数です。

Format

```
glcdc_err_t R_GLCDC_Close (
        void
)
```

Parameters

なし

Return Values

GLCDC_SUCCESS	*/* 問題なく処理が完了した場合 */*
GLCDC_ERR_NOT_OPEN	*/* R_GLCDC_Open 関数が実行されていない場合 */*
GLCDC_ERR_INVALID_MODE	*/* 関数が実行できないモードである場合 */*

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC の動作を終了するために、GLCDC で使用する割り込みを禁止に設定します。その後、GLCDC のソフトウェアリセットを実行して、モジュールストップに遷移させます。

本関数はモード「GLCDC_STATE_NOT_DISPLAYING」の時に実行できます。本関数の処理が正常に完了した場合、モード「GLCDC_STATE_CLOSED」に遷移します。

Example

```
volatile glcdc_err_t     ret_glcdc;

ret_glcdc = R_GLCDC_Close();
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

本関数を実行すると、CLUT メモリ以外のレジスタが初期化されます。再度 GLCDC を動作させる場合は R_GLCDC_Open 関数実行時に、必要な値を再設定してください。

R_GLCDC_Control ()

コントロールコマンドに応じた処理行う関数です。

Format

```
glcdc_err_t R_GLCDC_Control (
        glcdc_control_cmd_t     cmd         /* コントロールコマンド */
        void const * const      p_args      /* 設定パラメータの構造体のポインタ */
)
```

Parameters

glcdc_control_cmd_t　　cmd

コントロールコマンドを指定してください。

void const * const　　p_args

設定パラメータの構造体のポインタを設定してください。

指定できるコントロールコマンドの一覧表を示します。引数に設定した void 型ポインタは、各コマンドに応じて適切な型に変換して処理されます。

表 3.3　R_GLCDC_Control 関数のコントロールコマンド一覧

コマンドの定義	コマンドの内容	p_args に設定する型
GLCDC_CMD_START_DISPLAY	GLCDC の動作を許可にして LCD パネルに画像データを出力します。本コマンドはモード「GLCDC_STATE_NOT_DISPLAYING」のときに実行できます。本コマンド処理が正常に完了した場合、モード「GLCDC_STATE_DISPLAYING」に遷移します。	使用しません。NULL または FIT_NO_FUNC を設定してください。
GLCDC_CMD_STOP_DISPLAY	GLCDC の動作を停止します。本コマンドはモード「GLCDC_STATE_DISPLAYING」のときに実行できます。本コマンド処理が正常に完了した場合、モード「GLCDC_STATE_NOT_DISPLAYING」に遷移します。	使用しません。NULL または FIT_NO_FUNC を設定してください。
GLCDC_CMD_SET_INTERRUPT	GLCDC で使用する割り込みを設定します。本コマンド処理は R_GLCDC_Open 関数実行後、どのタイミングでも呼び出し可能です。本コマンド処理が完了しても、現在のモードから遷移しません。	glcdc_interrupt_cfg_t *
GLCDC_CMD_CLR_DETECTED_STATUS	グラフィック 2 指定ライン通知検出ステータス、グラフィック 1 アンダフロー検出ステータス、グラフィック 2 アンダフロー検出ステータスの検出フラグをクリアします。本コマンド処理は R_GLCDC_Open 関数実行後、どのタイミングでも呼び出し可能です。本コマンド処理が完了しても、現在のモードから遷移しません。	glcdc_detect_cfg_t *
GLCDC_CMD_CHANGE_BG_COLOR	バックグラウンド画面の背景色を設定します。本コマンド処理が完了しても、現在のモードから遷移しません。	glcdc_color_t *

参照する glcdc_interrupt_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CMD_SET_INTERRUPT」コマンド実行時に設定する必要はありません。

表 3.4　glcdc_interrupt_cfg_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
vpos_enable	VPOS 割り込みの許可、禁止	true	VPOS 割り込みを許可に設定
		false	VPOS 割り込みを禁止に設定
gr1uf_enable	GR1UF 割り込みの許可、禁止	true	GR1UF 割り込みを許可に設定
		false	GR1UF 割り込みを禁止に設定
gr2uf_enable	GR2UF 割り込みの許可、禁止	true	GR2UF 割り込みを許可に設定
		false	GR2UF 割り込みを禁止に設定

参照する glcdc_detect_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CMD_CLR_DETECTED_STATUS」コマンド実行時に設定する必要はありません。

表 3.5　glcdc_detect_cfg_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
vpos_detect	VPOS 検出フラグのクリア	true	VPOS 検出フラグをクリアする
		false	VPOS 検出フラグをクリアしない
gr1uf_detect	GR1UF 検出フラグのクリア	true	GR1UF 検出フラグをクリアする
		false	GR1UF 検出フラグをクリアしない
gr2uf_detect	GR2UF 検出フラグのクリア	true	GR2UF 検出フラグをクリアする
		false	GR2UF 検出フラグをクリアしない

参照する glcdc_color_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CMD_CHANGE_BG_COLOR」コマンド実行時に設定する必要はありません。

表 3.6　glcdc_color_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
byte.r	背景色 R 値	00h to FFh	背景色の R 値の設定
byte.g	背景色 G 値	00h to FFh	背景色の G 値の設定
byte.b	背景色 B 値	00h to FFh	背景色の B 値の設定

Return Values

```
GLCDC_SUCCESS                /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR        /* 引数 p_args が NULL ポインタの場合 */
GLCDC_ERR_INVALID_ARG        /* 設定パラメータが不正の場合 */
GLCDC_ERR_INVALID_MODE       /* 関数が実行できないモードである場合 */
GLCDC_ERR_NOT_OPEN           /* R_GLCDC_Open 関数が実行されていない場合 */
GLCDC_ERR_INVALID_UPDATE_TIMING    /* レジスタの更新タイミングが無効の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

コントロールコマンドに応じて、GLCDC の制御処理を行います。

Example

```
/* GLCDC を動作させる場合 */
volatile glcdc_err_t     ret_glcdc;

ret_glcdc = R_GLCDC_Control(GLCDC_CMD_START_DISPLAY, NULL);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

```
/* GLCDC を停止させる場合 */
volatile glcdc_err_t     ret_glcdc;

ret_glcdc = R_GLCDC_Control(GLCDC_CMD_STOP_DISPLAY, NULL);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

```
/* GLCDC の割り込み許可/禁止を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_interrupt_cfg_t     int_cfg;

int_cfg.vpos_enable = true;
int_cfg.gr1uf_enable = true;
int_cfg.gr2uf_enable = true;

ret_glcdc = R_GLCDC_Control(GLCDC_CMD_SET_INTERRUPT, (void *)&int_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

```
/* GLCDC の検出ステータスをクリアする場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_detect_cfg_t     detect_cfg;

detect_cfg.vpos_detect = true;
detect_cfg.gr1uf_detect = true;
detect_cfg.gr2uf_detect = true;

ret_glcdc = R_GLCDC_Control(GLCDC_CMD_CLR_DETECTED_STATUS, (void *)&detect_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

```
/* GLCDC の背景色を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_color_t     bg_color;

bg_color.byte.r = 0xFFh;
bg_color.byte.g = 0xFFh;
bg_color.byte.b = 0xFFh;

ret_glcdc = R_GLCDC_Control(GLCDC_CMD_CHANGE_BG_COLOR, (void *)&bg_color);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

「GLCDC_CMD_STOP_DISPLAY」コマンドを実行すると、GLCDC はバックグラウンド画面生成部のフレームエンドまで待ってから動作を停止します。再び GLCDC を動作させる場合は、LCD パネルへの出力信号のフレームエンドまで待ってから GLCDC を動作させてください。フレームエンド前に GLCDC を動作させた場合、LCD パネルによっては動作に影響が生じる場合があります。

R_GLCDC_LayerChange ()

グラフィック 1、グラフィック 2 の動作を変更する関数です。

Format

```
glcdc_err_t R_GLCDC_LayerChange (
        glcdc_frame_layer_t       frame                    /* 動作を変更するグラフィック画面 */
        glcdc_runtime_cfg_t const * const        p_args    /* 設定パラメータの構造体のポインタ */
)
```

Parameters

glcdc_frame_layer_t　　frame

動作を変更するグラフィック画面を設定してください。

void const * const　　p_args

設定パラメータの構造体のポインタを設定してください。

参照する glcdc_runtime_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、本関数実行時に設定する必要はありません。

表 3.7　glcdc_runtime_cfg_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
input.format	フレームバッファのデータフォーマット	GLCDC_IN_FORMAT_32BITS_ARGB8888	ARGB(8888)を使用
		GLCDC_IN_FORMAT_32BITS_RGB888	RGB(888)を使用
		GLCDC_IN_FORMAT_16BITS_RGB565	RGB(565)を使用
		GLCDC_IN_FORMAT_16BITS_ARGB1555	ARGB(1555)を使用
		GLCDC_IN_FORMAT_16BITS_ARGB4444	ARGB(4444)を使用
		GLCDC_IN_FORMAT_CLUT8	CLUT(8)を使用
		GLCDC_IN_FORMAT_CLUT4	CLUT(4)を使用
		GLCDC_IN_FORMAT_CLUT1	CLUT(1)を使用
input.p_base	フレームバッファの先頭アドレス	0000 0040h to FFFF FFC0h 下位 6 ビットは 0	フレームバッファの先頭アドレスの設定
input.bg_color.byte.r	グラフィック 1,2 の背景色 R 値	00h to FFh	グラフィック 1,2 の背景色の R 値の設定
input.bg_color.byte.g	グラフィック 1,2 の背景色 G 値	00h to FFh	グラフィック 1,2 の背景色の G 値の設定
input.bg_color.byte.b	グラフィック 1,2 の背景色 B 値	00h to FFh	グラフィック 1,2 の背景色の B 値の設定
input.hsize	画像データの横幅	5.1 画面の定義を参照	グラフィック 1,2 の画像の横幅の設定
input.vsize	画像データの高さ	5.1 画面の定義を参照	グラフィック 1,2 の画像の高さの設定

input.offset	マクロラインオフセット	-32768 to 32704 (64 の倍数)	グラフィック 1,2 のマクロラインオフセットの設定
input. frame_edge	グラフィック領域の枠の表示	true	グラフィック領域枠を表示に設定
		false	グラフィック領域枠を非表示に設定
input. coordinate.x	表示開始位置 x 座標	5.1 画面の定義を参照	グラフィック領域水平開始位置の設定
input. coordinate.y	表示開始位置 y 座標	5.1 画面の定義を参照	グラフィック領域垂直開始位置の設定
blend. blend_control	ブレンド処理の制御設定	GLCDC_BLEND_ CONTROL_NONE	アルファブレンド処理を無効に設定
		GLCDC_BLEND_ CONTROL_FADEIN	フェードインに設定
		GLCDC_BLEND_ CONTROL_FADEOUT	フェードアウトに設定
		GLCDC_BLEND_ CONTROL_FIXED	アルファ値固定に設定
		GLCDC_BLEND_ CONTROL_PIXEL	ピクセル単位アルファブレンドに設定
blend.visible	画像の表示設定	true	画像を表示に設定
		false	画像を非表示に設定
blend. frame_edge	矩形アルファブレンド領域の枠の表示	true	矩形アルファブレンド領域の枠を表示に設定
		false	矩形アルファブレンド領域の枠を非表示に設定
blend.fixed_ blend_value	固定アルファ値	00h to FFh	固定アルファ値の設定 （blend_control が GLCDC_BLEND_CONTROL_FIXED のときのみ有効）
blend. fade_speed	アルファ値の加減値	00h to FFh	アルファ値の加減値の設定 （blend_control が GLCDC_BLEND_CONTROL_FADEIN もしくは GLCDC_BLEND_CONTROL_FADEOUT のときのみ有効）
blend.start_ coordinate.x	ブレンド処理開始位置の x 座標	5.1 画面の定義を参照	矩形アルファブレンド領域水平幅、矩形アルファブレンド水平開始位置を設定
blend.end_ coordinate.x	ブレンド処理終了位置の x 座標	5.1 画面の定義を参照	
blend.start_ coordinate.y	ブレンド処理の開始位置の y 座標	5.1 画面の定義を参照	矩形アルファブレンド領域垂直幅、矩形アルファブレンド垂直開始位置を設定
blend.end_ coordinate.y	ブレンド処理の終了位置の y 座標	5.1 画面の定義を参照	
chromakey. enable	クロマキー処理の有効、無効	true	クロマキー処理を有効に設定
		false	クロマキー処理を無効に設定 （glcdc_runtime_cfg_t.chromakey 以下の構造体メンバの設定値は無視されます）
chromakey. before.byte.r	クロマキー処理対象 R 値	00h to FFh	クロマキー処理対象 R 値の設定
chromakey. before.byte.g	クロマキー処理対象 G 値	00h to FFh	クロマキー処理対象 G 値の設定
chromakey. before.byte.b	クロマキー処理対象 B 値	00h to FFh	クロマキー処理対象 B 値の設定

chromakey.after.byte.a	クロマキー置き換え後 A 値	00h to FFh	クロマキー処理で置き換えた後の A 値を設定
chromakey.after.byte.r	クロマキー置き換え後 R 値	00h to FFh	クロマキー処理で置き換えた後の R 値を設定
chromakey.after.byte.g	クロマキー置き換え後 G 値	00h to FFh	クロマキー処理で置き換えた後の G 値を設定
chromakey.after.byte.b	クロマキー置き換え後 B 値	00h to FFh	クロマキー処理で置き換えた後の B 値を設定

Return Values

```
GLCDC_SUCCESS                   /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR           /* 引数 p_args が NULL ポインタの場合 */
GLCDC_ERR_INVALID_ARG           /* 設定パラメータが不正の場合 */
GLCDC_ERR_NOT_OPEN              /* R_GLCDC_Open 関数が実行されていない場合 */
GLCDC_ERR_INVALID_UPDATE_TIMING /* レジスタの更新タイミングが無効の場合 */
GLCDC_ERR_INVALID_LAYER_SETTING /* グラフィック画面の設定が不正の場合 */
GLCDC_ERR_INVALID_ALIGNMENT     /* フレームバッファの先頭アドレスが不正の場合 */
GLCDC_ERR_INVALID_BLEND_SETTING /* ブレンドの設定が不正の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC のグラフィック 1、グラフィック 2 の動作を変更します。

本関数はモード「GLCDC_STATE_DISPLAYING」および「GLCDC_STATE_NOT_DISPLAYING」の時に実行できます。本関数の処理が完了しても現在のモードから遷移しません。

Example

```
/* グラフィック 1 の設定を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_frame_layer_t     frame;
glcdc_runtime_cfg_t     runtime_cfg;

frame = GLCDC_FRAME_LAYER_1;

runtime_cfg.input.format = GLCDC_IN_FORMAT_CLUT8;
runtime_cfg.input.p_base = (uint32_t *)0x00800000;
runtime_cfg.input.hsize = 448;
runtime_cfg.input.vsize = 253;
runtime_cfg.input.offset = 448;
runtime_cfg.input.frame_edge = false;
runtime_cfg.input.bg_color.byte.r = 0xCC;
runtime_cfg.input.bg_color.byte.g = 0xCC;
runtime_cfg.input.bg_color.byte.b = 0xCC;
runtime_cfg.input.coordinate.x = 16;
runtime_cfg.input.coordinate.y = 9;

runtime_cfg.blend.blend_control = GLCDC_BLEND_CONTROL_NONE;
runtime_cfg.blend.visible = true;
runtime_cfg.blend.frame_edge = false;
runtime_cfg.blend.fixed_blend_value = 0x00;
runtime_cfg.blend.fade_speed = 0x00;
runtime_cfg.blend.start_coordinate.x = 0;
runtime_cfg.blend.start_coordinate.y = 0;
runtime_cfg.blend.end_coordinate.x = 0;
runtime_cfg.blend.end_coordinate.y = 0;

runtime_cfg.chromakey.enable = false;
runtime_cfg.chromakey.before.byte.g = 0x00;
runtime_cfg.chromakey.before.byte.b = 0x00;
runtime_cfg.chromakey.before.byte.r = 0x00;
runtime_cfg.chromakey.after.byte.a = 0x00;
runtime_cfg.chromakey.after.byte.g = 0x00;
runtime_cfg.chromakey.after.byte.b = 0x00;
runtime_cfg.chromakey.after.byte.r = 0x00;

ret_glcdc = R_GLCDC_LayerChange(frame, &runtime_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

なし

R_GLCDC_BufferChange ()

グラフィック 1、グラフィック 2 のフレームバッファのアドレスを変更する関数です。

Format

```
glcdc_err_t R_GLCDC_BufferChange (
        glcdc_frame_layer_t      frame                    /* 対象グラフィックレイヤ */
        uint32_t const * const                p_base      /* 切り替え先のアドレス */
)
```

Parameters

glcdc_frame_layer_t　　frame

対象グラフィックレイヤを設定してください。

uint32_t const * const　　p_base

切り替え先のアドレスを設定してください。

Return Values

```
GLCDC_SUCCESS                   /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR           /* 引数 p_base が NULL ポインタの場合 */
GLCDC_ERR_INVALID_ARG           /* 設定パラメータが不正の場合 */
GLCDC_ERR_NOT_OPEN              /* R_GLCDC_Open 関数が実行されていない場合 */
GLCDC_ERR_INVALID_UPDATE_TIMING /* レジスタの更新タイミングが無効の場合 */
GLCDC_ERR_INVALID_ALIGNMENT     /* フレームバッファの先頭アドレスが不正の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC のグラフィック 1、グラフィック 2 のフレームバッファのアドレスを変更します。

本関数はモード「GLCDC_STATE_DISPLAYING」および「GLCDC_STATE_NOT_DISPLAYING」の時に実行できます。本関数の処理が完了しても現在のモードから遷移しません。

Example

```
/* グラフィック 1 のフレームバッファのアドレスを 0x00800000 に変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_frame_layer_t     frame;

frame = GLCDC_FRAME_LAYER_1;


ret_glcdc = R_GLCDC_BufferChange(frame, 0x00800000);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

なし

R_GLCDC_ColorCorrection ()

GLCDC の輝度補正、コントラスト補正、ガンマ補正の設定を変更する関数です。

Format

```
glcdc_err_t R_GLCDC_ColorCorrection (
        glcdc_correction_cmd_t      cmd         /* 変更する設定のコマンド */
        void const * const          p_args      /* 設定パラメータの構造体のポインタ */
)
```

Parameters

glcdc_correction_cmd_t　　cmd

変更する設定をコマンドで指定してください。

void const * const　　p_args

設定パラメータの構造体のポインタを設定してください。

指定できるコントロールコマンドの一覧表を示します。引数に設定した void 型ポインタは、各コマンドに応じて適切な型に変換して処理されます。

表 3.8　R_GLCDC_ColorCorrection 関数のコントロールコマンド一覧

コマンドの定義	コマンドの内容	p_args に設定する型
GLCDC_CORRECTION_CMD_SET_ALL	輝度補正、コントラスト補正、ガンマ補正の設定を行います。	glcdc_correction_t *
GLCDC_CORRECTION_CMD_BRIGHTNESS	輝度補正の設定を行います。	glcdc_brightness_t *
GLCDC_CORRECTION_CMD_CONTRAST	コントラスト補正の設定を行います。	glcdc_contrast_t *
GLCDC_CORRECTION_CMD_GAMMA	ガンマ補正の設定を行います。	glcdc_gamma_correction_t *

参照する glcdc_correction_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CORRECTION_CMD_SET_ALL」コマンド実行時に設定する必要はありません。

表 3.9　glcdc_correction_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
brightness.enable	輝度補正の有効、無効	true	輝度補正を有効に設定
		false	輝度補正を無効に設定（glcdc_correction_t.brightness 以下の構造体メンバの設定値に依らず、RGB 信号の輝度調整値に 0 が設定されます）
brightness.r	R 信号の輝度調整値	000h：-512 : 200h：0 : 3FFh：+511	R 信号の輝度調整値の設定
brightness.g	G 信号の輝度調整値		G 信号の輝度調整値の設定
brightness.b	B 信号の輝度調整値		B 信号の輝度調整値の設定

contrast.enable	コントラスト補正有効、無効	true	コントラスト補正を有効に設定
		false	コントラスト補正を無効に設定（glcdc_correction_t.contrast 以下の構造体メンバの設定値に依らず、RGB 信号のコントラスト調整値に 1.000 が設定されます）
contrast.r	R 信号のコントラスト調整値	00h：0/128 = 0.000 : 80h：128/128 = 1.000 : FFh：255/128 = 1.992	R 信号のコントラスト調整値を設定
contrast.g	G 信号のコントラスト調整値		G 信号のコントラスト調整値を設定
contrast.b	B 信号のコントラスト調整値		B 信号のコントラスト調整値を設定
gamma.enable	ガンマ補正の有効、無効	true	ガンマ補正を有効に設定
		false	ガンマ補正を無効に設定（glcdc_correction_t.gamma 以下の構造体メンバの設定値は無視されます）
gamma.p_r	R 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	R 信号の各領域のゲインと開始しきい値の設定
gamma.p_g	G 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	G 信号の各領域のゲインと開始しきい値の設定
gamma.p_b	B 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	B 信号の各領域のゲインと開始しきい値の設定

参照する glcdc_brightness_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CORRECTION_CMD_BRIGHTNESS」コマンド実行時に設定する必要はありません。

表 3.10　glcdc_brightness_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
enable	輝度補正の有効、無効	true	輝度補正を有効に設定
		false	輝度補正を無効に設定（glcdc_brightness_t 以下の構造体メンバの設定値に依らず、RGB 信号の輝度調整値に 0 が設定されます）
r	R 信号の輝度調整値	000h：-512 : 200h：0 : 3FFh：+511	R 信号の輝度調整値の設定
g	G 信号の輝度調整値		G 信号の輝度調整値の設定
b	B 信号の輝度調整値		B 信号の輝度調整値の設定

参照する glcdc_contrast_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CORRECTION_CMD_CONTRAST」コマンド実行時に設定する必要はありません。

表 3.11　glcdc_contrast_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
enable	コントラスト補正有効、無効	true	コントラスト補正を有効に設定
		false	コントラスト補正を無効に設定（glcdc_contrast_t 以下の構造体メンバの設定値に依らず、RGB 信号のコントラスト調整値に 1.000 が設定されます）
r	R 信号のコントラスト調整値	00h：0/128 = 0.000 : 80h：128/128 = 1.000 : FFh：255/128 = 1.992	R 信号のコントラスト調整値を設定
g	G 信号のコントラスト調整値		G 信号のコントラスト調整値を設定
b	B 信号のコントラスト調整値		B 信号のコントラスト調整値を設定

参照する glcdc_gamma_correction_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、「GLCDC_CORRECTION_CMD_GAMMA」コマンド実行時に設定する必要はありません。

表 3.12　glcdc_gamma_correction_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
enable	ガンマ補正の有効、無効	true	ガンマ補正を有効に設定
		false	ガンマ補正を無効に設定（glcdc_gamma_correction_t 以下の構造体メンバの設定値は無視されます）
p_r	R 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	R 信号の各領域のゲインと開始しきい値の設定
p_g	G 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	G 信号の各領域のゲインと開始しきい値の設定
p_b	B 信号のガンマ補正テーブル	5.2 ガンマ補正値の計算方法を参照	B 信号の各領域のゲインと開始しきい値の設定

Return Values

```
GLCDC_SUCCESS                   /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR           /* 引数 p_args が NULL ポインタの場合 */
GLCDC_ERR_INVALID_ARG           /* 設定パラメータが不正の場合 */
GLCDC_ERR_NOT_OPEN              /* R_GLCDC_Open 関数が実行されていない場合 */
GLCDC_ERR_INVALID_UPDATE_TIMING /* レジスタの更新タイミングが無効の場合 */
GLCDC_ERR_INVALID_GAMMA_SETTING /* ガンマの設定が不正の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC の輝度、コントラスト、ガンマ補正の設定を変更します。本関数の第 1 引数のコマンドに応じて設定する内容が決まります。
本関数はモード「GLCDC_STATE_DISPLAYING」および「GLCDC_STATE_NOT_DISPLAYING」の時に実行できます。本関数の処理が完了しても現在のモードから遷移しません。

Example

```
/* 全ての設定を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_correction_t     correction_cfg;

correction_cfg.brightness.enable = true;
correction_cfg.brightness.r = 0x200;
correction_cfg.brightness.g = 0x200;
correction_cfg.brightness.b = 0x200;

correction_cfg.contrast.enable = true;
correction_cfg.contrast.r = 0x80;
correction_cfg.contrast.g = 0x80;
correction_cfg.contrast.b = 0x80;

correction_cfg.gamma.enable = true;
correction_cfg.gamma.p_r = (gamma_correction_t *)&g_gamma_table;
correction_cfg.gamma.p_g = (gamma_correction_t *)&g_gamma_table;
correction_cfg.gamma.p_b = (gamma_correction_t *)&g_gamma_table;

ret_glcdc = R_GLCDC_ColorCorrection(GLCDC_CORRECTION_CMD_SET_ALL,
                                    (void *)&correction_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

```
/* 輝度補正の設定を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_brightness_t     brightness_cfg;

brightness_cfg.enable = true;
brightness_cfg.r = 0x200;
brightness_cfg.g = 0x200;
brightness_cfg.b = 0x200;

ret_glcdc = R_GLCDC_ColorCorrection(GLCDC_CORRECTION_CMD_BRIGHTNESS,
                                    (void *)&brightness_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

```
/* コントラスト補正の設定を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_contrast_t     contrast_cfg;

contrast_cfg.enable = true;
contrast_cfg.r = 0x80;
contrast_cfg.g = 0x80;
contrast_cfg.b = 0x80;

ret_glcdc = R_GLCDC_ColorCorrection(GLCDC_CORRECTION_CMD_CONTRAST,
                                    (void *)&contrast_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
    /* error processing */
}
```

```
/* ガンマ補正の設定を変更する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_gamma_correction_t     gamma_cfg;

gamma_cfg.enable = true;
gamma_cfg.p_r = (gamma_correction_t *)&g_gamma_table;
gamma_cfg.p_g = (gamma_correction_t *)&g_gamma_table;
gamma_cfg.p_b = (gamma_correction_t *)&g_gamma_table;

ret_glcdc = R_GLCDC_ColorCorrection(GLCDC_CORRECTION_CMD_GAMMA,
                                    (void *)&gamma_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
    /* error processing */
}
```

Special Notes:

なし

R_GLCDC_ClutUpdate ()

GLCDC の CLUT メモリを更新する関数です。更新した CLUT メモリは出力に反映されます。

Format

```
glcdc_err_t R_GLCDC_ClutUpdate (
        glcdc_frame_layer_t                 frame       /* 動作を変更するグラフィック画面 */
        glcdc_clut_cfg_t const * const      p_clut_cfg  /* CLUT メモリの構造体のポインタ */
)
```

Parameters

glcdc_frame_layer_t frame

動作を変更するグラフィック画面を設定してください。

glcdc_clut_cfg_t　　p_clut_cfg

CLUT メモリの構造体のポインタを設定してください。

参照する glcdc_clut_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、本関数実行時に設定する必要はありません。

表 3.13　glcdc_clut_cfg_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
enable	CLUT メモリの更新の有効、無効の選択	true	CLUT メモリを更新します
		false	関数を実行しても、CLUT メモリは更新されません
p_base	CLUT メモリの先頭アドレスへのポインタ	NULL 以外	ポインタが指し示すアドレスから値を読み出し CLUT メモリにコピーします
start	更新する CLUT メモリの開始エントリ番号	0 to 255 (ただし、 start + size < 257)	指定したエントリ番号から CLUT メモリの更新を開始します
size	更新する CLUT メモリのエントリサイズ	1 to 256 (ただし、 start + size < 257)	指定したサイズ分の CLUT メモリを更新します

Return Values

```
GLCDC_SUCCESS                   /* 問題なく処理が完了した場合 */
GLCDC_ERR_INVALID_PTR           /* 引数 p_clut_cfg が NULL ポインタの場合 */
GLCDC_ERR_INVALID_ARG           /* 設定パラメータが不正の場合 */
GLCDC_ERR_NOT_OPEN              /* R_GLCDC_Open 関数が実行されていない場合 */
GLCDC_ERR_INVALID_UPDATE_TIMING /* レジスタの更新タイミングが不正の場合 */
GLCDC_ERR_INVALID_CLUT_ACCESS   /* CLUT メモリの設定が不正の場合 */
```

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC の CLUT メモリを更新します。
本関数はモード「GLCDC_STATE_DISPLAYING」および「GLCDC_STATE_NOT_DISPLAYING」の時に実行できます。本関数の処理が完了しても、現在のモードから遷移しません。

Example

```
/* グラフィック 1 の CLUT メモリを全て更新する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_clut_cfg_t     clut_cfg;

clut_cfg.enable = true;
clut_cfg.p_base = (uint32_t *)g_gr_clut_table;
clut_cfg.size = 256;
clut_cfg.start = 0;

ret_glcdc = R_GLCDC_ClutUpdate(GLCDC_FRAME_LAYER_1, &clut_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
    /* error processing */
}
```

Special Notes:

なし

R_GLCDC_ClutUpdate_ NoReflect ()

GLCDC の CLUT メモリを更新します。ただし、更新した CLUT メモリは出力に反映されません。
なお、フォーマット、パラメータ、リターンバリューは R_GLCDC_ClutUpdate 関数と同じです。

Description

本関数は、画像毎に CLUT メモリを更新する際などに、R_GLCDC_LayerChange 関数と組み合わせて使うことで CLUT メモリとグラフィック変更を同時に行うことが出来ます。

使用方法は、Example を参考にしてください。
なお、本関数はモード「GLCDC_STATE_DISPLAYING」および「GLCDC_STATE_NOT_DISPLAYING」の時に実行できます。本関数の処理が完了しても、現在のモードから遷移しません。

Example

```
/* グラフィック 1 の CLUT メモリを更新し、直後にグラフィック 1 を変更する。*/
volatile glcdc_err_t     ret_glcdc;
glcdc_clut_cfg_t     clut_cfg;
glcdc_frame_layer_t     frame;
glcdc_runtime_cfg_t     runtime_cfg;

/* R_GLCDC_ClutUpdate_ NoReflect 関数で使用する値の設定 */
clut_cfg.enable = true;
clut_cfg.p_base = (uint32_t *)g_gr_clut_table;
clut_cfg.size = 256;
clut_cfg.start = 0;
frame = GLCDC_FRAME_LAYER_1;

/* R_GLCDC_LayerChange 関数で使用する値の設定 */
runtime_cfg.input.format = GLCDC_IN_FORMAT_CLUT8;
runtime_cfg.input.p_base = (uint32_t *)0x00800000;
runtime_cfg.input.hsize = 448;
runtime_cfg.input.vsize = 253;
runtime_cfg.input.offset = 448;
runtime_cfg.input.frame_edge = false;
runtime_cfg.input.bg_color.byte.r = 0xCC;
runtime_cfg.input.bg_color.byte.g = 0xCC;
runtime_cfg.input.bg_color.byte.b = 0xCC;
runtime_cfg.input.coordinate.x = 16;
runtime_cfg.input.coordinate.y = 9;
runtime_cfg.blend.blend_control = GLCDC_BLEND_CONTROL_NONE;
runtime_cfg.blend.visible = true;
runtime_cfg.blend.frame_edge = false;
runtime_cfg.blend.fixed_blend_value = 0x00;
runtime_cfg.blend.fade_speed = 0x00;
runtime_cfg.blend.start_coordinate.x = 0;
runtime_cfg.blend.start_coordinate.y = 0;
runtime_cfg.blend.end_coordinate.x = 0;
runtime_cfg.blend.end_coordinate.y = 0;
runtime_cfg.chromakey.enable = false;
runtime_cfg.chromakey.before.byte.g = 0x00;
runtime_cfg.chromakey.before.byte.b = 0x00;
runtime_cfg.chromakey.before.byte.r = 0x00;
runtime_cfg.chromakey.after.byte.a = 0x00;
runtime_cfg.chromakey.after.byte.g = 0x00;
runtime_cfg.chromakey.after.byte.b = 0x00;
runtime_cfg.chromakey.after.byte.r = 0x00;

ret_glcdc = R_GLCDC_ClutUpdate_ NoReflect (frame, &clut_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}

/* CLUT メモリとグラフィック設定が同時に出力に反映されます。*/
ret_glcdc = R_GLCDC_LayerChange (frame, &runtime_cfg);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

なし

R_GLCDC_GetStatus ()

GLCDC の各ステータスを取得する関数です。

Format

```
glcdc_err_t R_GLCDC_GetStatus (
        glcdc_status_t * const      p_status        /* 取得ステータスを格納する構造体のポインタ */
)
```

Parameters

glcdc_status_t * const　p_status

取得ステータスを格納する構造体のポインタを設定してください。

表 3.14　glcdc_status_t 構造体メンバと設定値

構造体メンバ	概略	設定値	設定内容
state	GLCDC FIT モジュールの遷移状態	GLCDC_STATE_NOT_DISPLAYING	GLCDC は停止中
		GLCDC_STATE_DISPLAYING	GLCDC は動作中
state_vpos	グラフィック 2 指定ライン通知の検出有無	GLCDC_NOT_DETECTED	検出していない
		GLCDC_DETECTED	検出した
state_gr1uf	グラフィック 1 アンダフローの検出有無	GLCDC_NOT_DETECTED	検出していない
		GLCDC_DETECTED	検出した
state_gr2uf	グラフィック 2 アンダフローの検出有無	GLCDC_NOT_DETECTED	検出していない
		GLCDC_DETECTED	検出した
fade_status	グラフィック 1,2 のフェード状態	GLCDC_FADE_STATUS_NOT_UNDERWAY	フェードイン/フェードアウトは停止中
		GLCDC_FADE_STATUS_FADING_UNDERWAY	フェードイン/フェードアウトは実行中
		GLCDC_FADE_STATUS_UNCERTAIN	グラフィックのレジスタ値を設定中

Return Values

GLCDC_SUCCESS　*/* 問題なく処理が完了した場合 */*

GLCDC_ERR_INVALID_PTR　*/* 引数 p_status が NULL ポインタの場合 */*

GLCDC_ERR_NOT_OPEN　*/* R_GLCDC_Open 関数が実行されていない場合 */*

Properties

r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description

GLCDC の各ステータスを取得します。取得したステータスは引数で渡された構造体 p_status に書き込みます。

本関数は R_GLCDC_Open 関数実行後、どのタイミングでも呼び出し可能です。本関数の処理が完了しても、現在のモードから遷移しません。

Example

```
/* GLCDC の各ステータスを取得する場合 */
volatile glcdc_err_t     ret_glcdc;
glcdc_status_t     status;

ret_glcdc = R_GLCDC_GetStatus(&status);
if (GLCDC_SUCCESS != ret_glcdc)
{
      /* error processing */
}
```

Special Notes:

なし

R_GLCDC_GetVersion ()

API のバージョンを返す関数です。

Format
uint32_t R_GLCDC_GetVersion (void)

Parameters
なし

Return Values
バージョン番号

Properties
r_glcdc_rx_if.h にプロトタイプ宣言されています。

Description
現在インストールされている GLCDC FIT モジュールのバージョンを返します。バージョン番号はコード化されています。最初の 2 バイトがメジャーバージョン番号で、後の 2 バイトがマイナーバージョン番号です。例えば、バージョンが 4.25 の場合、戻り値は‘0x00040019’となります。

Example
```
/* GLCDC FIT モジュールのバージョンを取得する場合 */
volatile uint32_t     version;

version = R_GLCDC_GetVersion();
```

Special Notes:
なし

4. 端子設定

GLCDC FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。端子設定は、R_GLCDC_Open 関数を呼び出した後に行ってください。

e² studio の場合は「FIT コンフィグレータ」または「スマート・コンフィグレータ」の端子設定機能を使用することができます。FIT コンフィグレータ、スマート・コンフィグレータの端子設定機能を使用すると、端子設定画面で選択したオプションに応じて、ソースファイルが出力されます。そのソースファイルで定義された関数を呼び出すことにより端子を設定できます。詳細は表 4.1 を参照してください。

表 4.1　FIT コンフィグレータが出力する関数一覧

使用マイコン	出力される関数名	備考
RX65N RX72M RX72N RX66N	R_GLCDC_PinSet()	

5.　使用方法

5.1　　画面の定義

GLCDC FIT モジュールでは、R_GLCDC_Open 関数、R_GLCDC_LayerChange 関数の引数の値に従って、各画面の基準点、有効表示領域、表示開始位置などが決まります。使用する LCD パネルの仕様から「図 5.1 画面の定義」と「表 5.1 各引数の設定範囲」に従って、引数の値を設定してください。

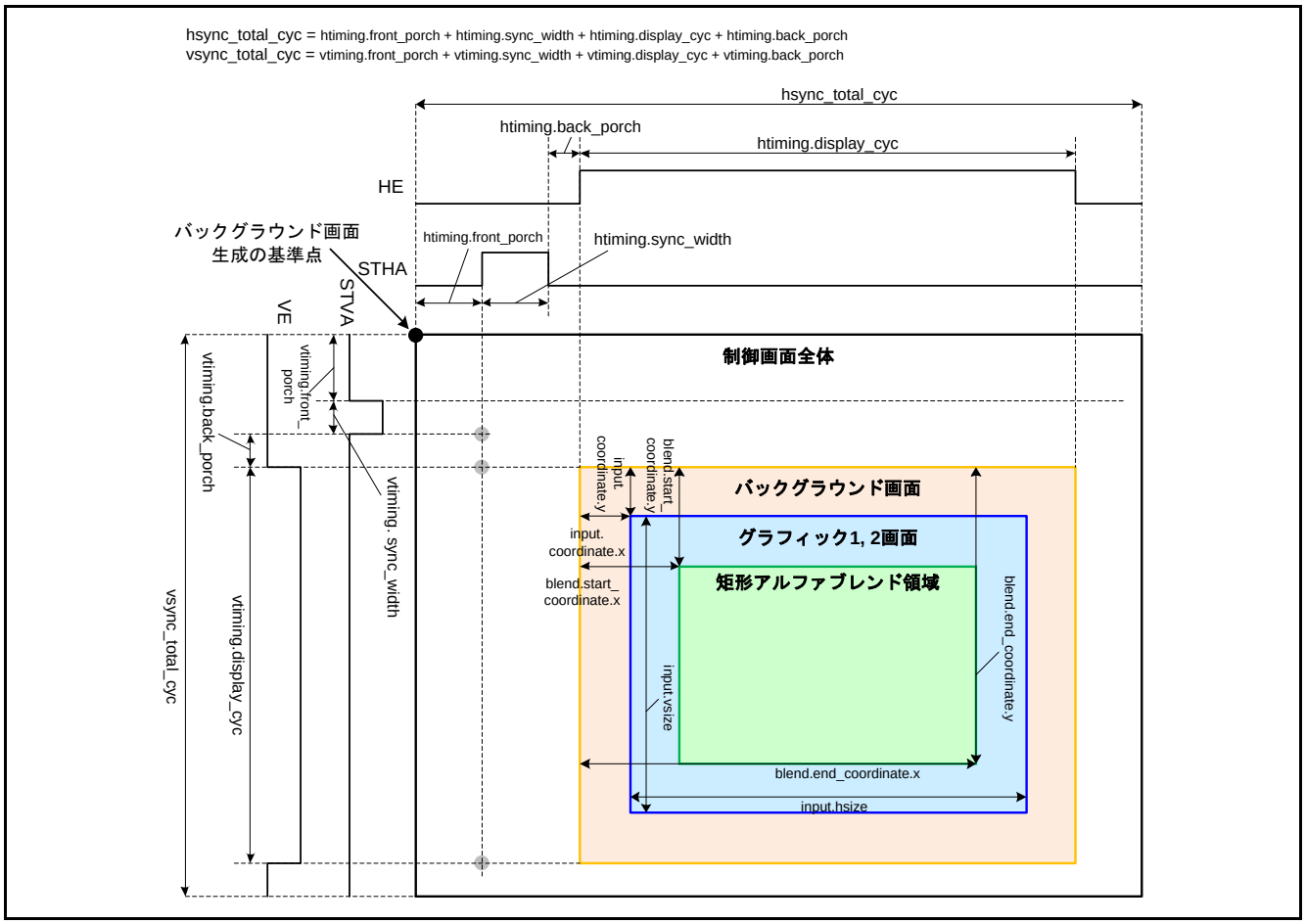

図 5.1　画面の定義

表 5.1　各引数の設定範囲

引数名称	設定値	備考
htiming.front_porch	2 < htiming.front_porch	hsync_total_cyc = (htiming.front_porch + htiming.back_porch + htiming.display_cyc + htiming.sync_width)としたとき、 23 < hsync_total_cyc < 1025 の範囲で設定してください 5 < ((htiming.front_porch - 2) + htiming.back_porch + htiming.sync_width) の範囲で設定してください 2x2 パターンディザを使用する場合、htiming.display_cyc を 4 の倍数で設定してください
htiming.back_porch	0 < htiming.back_porch	
htiming.display_cyc	15 < htiming.display_cyc	
htiming.sync_width	0 ≦ htiming.sync_width	
vtiming.front_porch	1 < vtiming.front_porch	vsync_total_cyc = (vtiming.front_porch + vtiming.back_porch + vtiming.display_cyc + vtiming_syncwidth)としたとき、 19 < vsync_total_cyc < 1025 の範囲で設定してください 2 < ((vtiming.front_porch - 1) + vtiming.back_porch + vtiming.sync_width) の範囲で設定してください 2x2 パターンディザを使用する場合、vtiming.display_cyc を 2 の倍数で設定してください
vtiming.back_porch	0 < vtiming.back_porch	
vtiming.display_cyc	15 < vtiming.display_cyc	
vtiming.sync_width	0 ≦ vtiming.sync_width	
input.hsize	15 < input.hsize < (htiming.display_cyc + 1)	偶数値で設定してください
input.coordinate.x	0 ≦ input.coordinate.x < (htiming.display_cyc - 15)	(input.coordinate.x + input.hsize) < (htiming.display_cyc + 1)の範囲で設定してください
input.vsize	15 < input.vsize < (vtiming.display_cyc + 1)	
input.coordinate.y	0 ≦ input.coordinate.y < (vtiming.display_cyc - 15)	(input.coordinate.y + input.vsize) < (vtiming.display_cyc + 1)の範囲で設定してください
blend.start_coordinate.x	0 ≦ blend.start_coordinate.x < blend.end_coordinate.x < htiming.display_cyc かつ 0 ≦ blend.start_coordinate.x < blend.end_coordinate.x < 1017	(htiming.back_porch + htiming.sync_width + blend.start_coordinate.x) < 1006 の範囲で設定してください 水平座標位置 100~200 の範囲に設定したい場合は、blend.start_coordinate.x = 100, blend.end_coordinate.x = (200 + 1) としてください
blend.end_coordinate.x		
blend.start_coordinate.y	0 ≦ blend.start_coordinate.y < blend.end_coordinate.y < vtiming.display_cyc かつ 0 ≦ blend.start_coordinate.y < blend.end_coordinate.y < 1021	(vtiming.back_porch + vtiming.sync_width + blend.start_coordinate.y) < 1007 の範囲で設定してください 垂直座標位置 100~200 の範囲に設定したい場合は、blend.start_coordinate.y = 100, blend.end_coordinate.y = (200 + 1) としてください
blend.end_coordinate.y		

5.2　ガンマ補正値の計算方法

GLCDC におけるガンマ補正値の計算方法について示します。

GLCDC FIT モジュールは、ガンマ補正を使用することで、LCD パネルの特性に合わせて明るさを調整できます。ガンマ補正を適切に実施するには、GAMxLUTn (n =1…8)レジスタにゲイン値を、GAMxAREAn (n = 1…5)レジスタに領域のしきい値を設定してください。

各領域のゲイン値の計算方法の例を以下に示します。

$$Dout = \left(\frac{Din}{pixel}\right)^{\frac{1}{\gamma}} \times pixel$$

上記の計算式において、γはガンマ値、pixel は画素数、Din は補正前の輝度値、Dout は補正後の輝度値を表します。ただし、GLCDC は入出力信号を 10 ビットで計算しているため、pixel は 1023 になります。

例えば、各領域の幅が均等に 64 になるように設定します。上記の計算式から、ガンマ値をγ=0.7 としたとき、Din=0 の場合の Dout は 0 になり、Din=64 の場合の Dout は 19.512 になります。

$$gain = \frac{Dout_{m+1} - Dout_m}{width} \quad (m = 0\sim15)$$

上記の計算式において、領域 0 のゲイン値を求める場合、Dout(m+1)は領域 1 の補正後の輝度値、Dout(m)は領域 0 の補正後の輝度値、width は領域 0 の幅を表します。

上記の計算式から領域 0(m = 0)のゲイン値は gain=0.304875 になります。また、領域 0 のゲイン値のレジスタへの設定値は 0.304875 x 1024 = 312（小数点第一位を四捨五入）になります。上記手順を 16 領域分繰り返してガンマ補正テーブルを設定します。

各領域の幅を設定するためのしきい値は、TH(k) < TH(k+1)になるように設定してください。ただし、TH(k) = 0x3FF の場合に限り、TH(k) = TH(k+1)としても構いません。

以下に各ガンマ補正値における設定例を示します。

```
/*  ガンマ補正値 0.5 におけるガンマ補正テーブルの設定例  */
const gamma_correction_t g_gamma_table =
{
      /* gain (r = 0.5) */
      { 64, 192, 320, 448, 577, 705, 833, 961, 1089, 1217, 1345, 1473, 1602, 1730, 1858, 1954 },
      /* threshold */
      { 64, 128, 192, 256, 320, 384, 448, 512, 576, 640, 704, 768, 832, 896, 960 }
};
/*  ガンマ補正値 0.7 におけるガンマ補正テーブルの設定例  */
const gamma_correction_t g_gamma_table =
{
      /* gain (r = 0.7) */
      { 312, 528, 659, 762, 849, 926, 995, 1057, 1116, 1170, 1222, 1270, 1316, 1361, 1403, 1421 },
      /* threshold */
      {    64,   128,   192,   256,   320,   384,   448,   512,   576,   640,   704,   768,   832,   896,   960 }
};
/*  ガンマ補正値 0.9 におけるガンマ補正テーブルの設定例  */
const gamma_correction_t g_gamma_table =
{
      /* gain (r = 0.9) */
      { 753, 873, 925, 961, 988, 1010, 1029, 1046, 1061, 1074, 1086, 1097, 1107, 1117, 1126, 1116 },
      /* threshold */
      {    64,   128,   192,   256,   320,   384,   448,   512,   576,   640,   704,   768,   832,   896,   960 }
};
/*  ガンマ補正値 1.1 におけるガンマ補正テーブルの設定例  */
const gamma_correction_t g_gamma_table =
{
      /* gain (r = 1.1) */
      { 1317, 1157, 1103, 1069, 1045, 1026, 1010, 997, 986, 976, 967, 959, 952, 945, 939, 919 },
      /* threshold */
      {    64,   128,   192,   256,   320,   384,   448,   512,   576,   640,   704,   768,   832,   896,   960 }
};
/*  ガンマ補正値 1.3 におけるガンマ補正テーブルの設定例  */
const gamma_correction_t g_gamma_table =
{
      /* gain (r = 1.3) */
      { 1941, 1367, 1211, 1119, 1056, 1008, 970, 938, 911, 888, 868, 850, 834, 819, 806, 781 },
      /* threshold */
      {    64,   128,   192,   256,   320,   384,   448,   512,   576,   640,   704,   768,   832,   896,   960 }
};
```

5.3　ブレンド設定における注意事項

画像の表示設定、ブレンド処理の制御設定、クロマキー処理の有効、無効について、設定値の組み合わせに制限事項があります。設定可能な組み合わせを「表 5.2 設定値の組み合わせ」に示します。また、記載してある設定値の組み合わせ以外では使用しないでください。

表 5.2　設定値の組み合わせ

画像の表示設定 (blend.visible)	ブレンド処理の制御設定 (blend.blend_control)	クロマキー処理の有効、無効 (chromakey.enable)	表示内容
false	GLCDC_BLEND_CONTROL_NONE	false	下層画面
false	GLCDC_BLEND_CONTROL_PIXEL	false	下層画面
true	GLCDC_BLEND_CONTROL_NONE	false	カレント画面
true	GLCDC_BLEND_CONTROL_FADEIN	true	矩形領域内は、カレント画面と下層画面のフェードイン 矩形領域外は、カレント画面をクロマキー処理したデータと下層画面のピクセル単位アルファブレンド表示
		false	矩形領域内は、カレント画面と下層画面のフェードイン 矩形領域外は、カレント画面と下層画面のピクセル単位アルファブレンド表示
true	GLCDC_BLEND_CONTROL_FADEOUT	true	矩形領域内は、カレント画面と下層画面のフェードアウト 矩形領域外は、カレント画面をクロマキー処理したデータと下層画面のピクセル単位アルファブレンド表示
		false	矩形領域内は、カレント画面と下層画面のフェードアウト 矩形領域外は、カレント画面と下層画面のピクセル単位アルファブレンド表示
true	GLCDC_BLEND_CONTROL_FIXED *1	true	矩形領域内は、カレント画面と下層画面の矩形アルファブレンド表示 矩形領域外は、カレント画面をクロマキー処理したデータと下層画面のピクセル単位アルファブレンド表示
		false	矩形領域内は、カレント画面と下層画面の矩形アルファブレンド表示 矩形領域外は、カレント画面と下層画面のピクセル単位アルファブレンド表示
true	GLCDC_BLEND_CONTROL_PIXEL	true	カレント画面をクロマキー処理したデータと下層画面のピクセル単位アルファブレンド表示
		false	カレント画面と下層画面のピクセル単位アルファブレンド表示

【注】 *1 本値を設定しているグラフィック画面は、R_GLCDC_GetStatus 関数実行時の取得ステータス fade_status が必ず GLCDC_FADE_STATUS_FADING_UNDERWAY になります。

5.4　内部メインバス 2 優先順位設定について

GLCDC が利用する内部メインバス 2 に優先順位の設定があります。リセット解除後の優先順位はグラフィック 1 > グラフィック 2 となっているため、グラフィック 1 のデータの読み出しが優先されます。優先度設定はボードサポートパッケージモジュール（BSP モジュール）で行うことができます。ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)「3.2.10 拡張バスマスタ優先度設定」を参照してください。

5.5　マクロラインオフセットの制限を守ることができない場合

フレームバッファのデータフォーマットや横幅によってマクロラインオフセットの制限を守ることができない場合、画像の横幅を拡張し余白を作ることで、マクロラインオフセットの制限を満たす画像を作成してください。

例としてフレームバッファのデータフォーマットを CLUT(8)かつ横幅 480px の画像を LCD に表示させる方法を説明します。本来マクロラインオフセットを 480（1 ピクセルあたりのバイト数 × 画像の横幅 = 1 × 480）と設定します。しかし、480 はマクロラインオフセットの制限である 64 の倍数ではありません。そのため条件を満たすように余白を含め横幅 512px の画像に拡張し、拡張した画像をフレームバッファに書き込んでください。その後、画像データの横幅（input.hsize）を 480px と設定することで、画像を任意の横幅で表示することができます。拡張することでフレームバッファに冗長を持つことになり、その分のメモリ使用量が増加します。

詳しくは各デバイスのユーザーズマニュアル　ハードウェア編　グラフィック LCD コントローラ（GLCDC）の章を参照してください。

以下の「図 5.2 」は画像の加工例を示します。拡張画像に示している赤線が拡張した部分です。

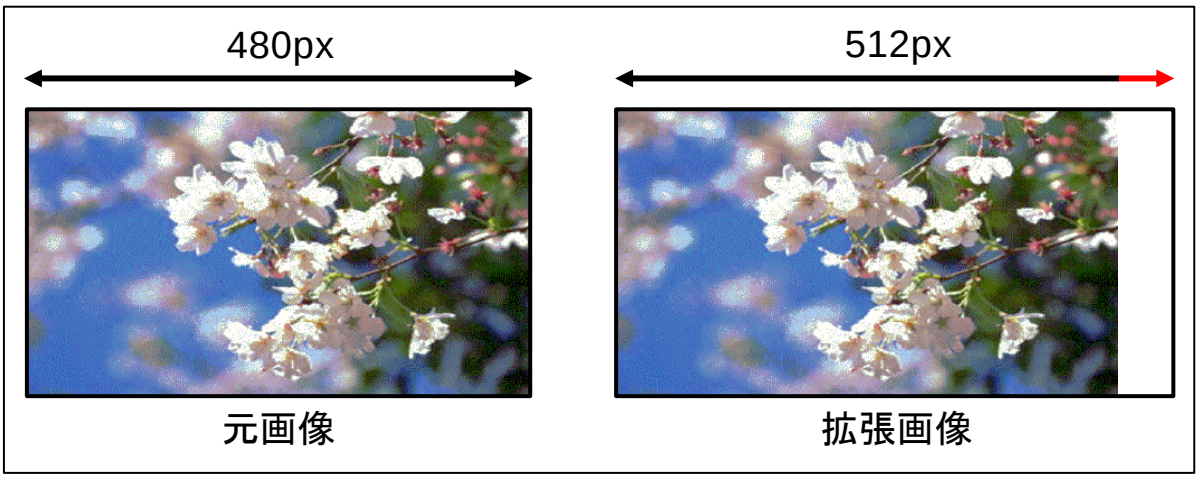

図 5.2　拡張画像例

5.6　QE for Dispay[RX]との連携

QE for Display[RX]はルネサス製 RX マイコンに対応する統合開発環境 e² studio 用のプラグインです。QE for Display[RX]では、表示制御を GUI 上で設定することができます。使用する表示機器の情報を入力することで、表示制御に必要な情報を含んだヘッダファイルを生成します。また、ツールにはリアルタイムでタイミングを調整する機能があり、使用する表示機器を接続したまま微調整を行ってから、ヘッダファイルを生成する事も可能です。

QE for Display[RX] V2.0.0 以降では、QE for Display[RX]がコンパイラオプション（-define）に“QE_DISPLAY_CONFIGURATION”の定義を追加します。GLCDC FIT モジュールは、その定義を確認することにより、r_glcdc_rx_config.h や QE for Display[RX]が生成したヘッダファイル（r_lcd_timing.h、r_image_config.h）のコンフィグレーションオプションを参照して GLCDC の設定を行います。

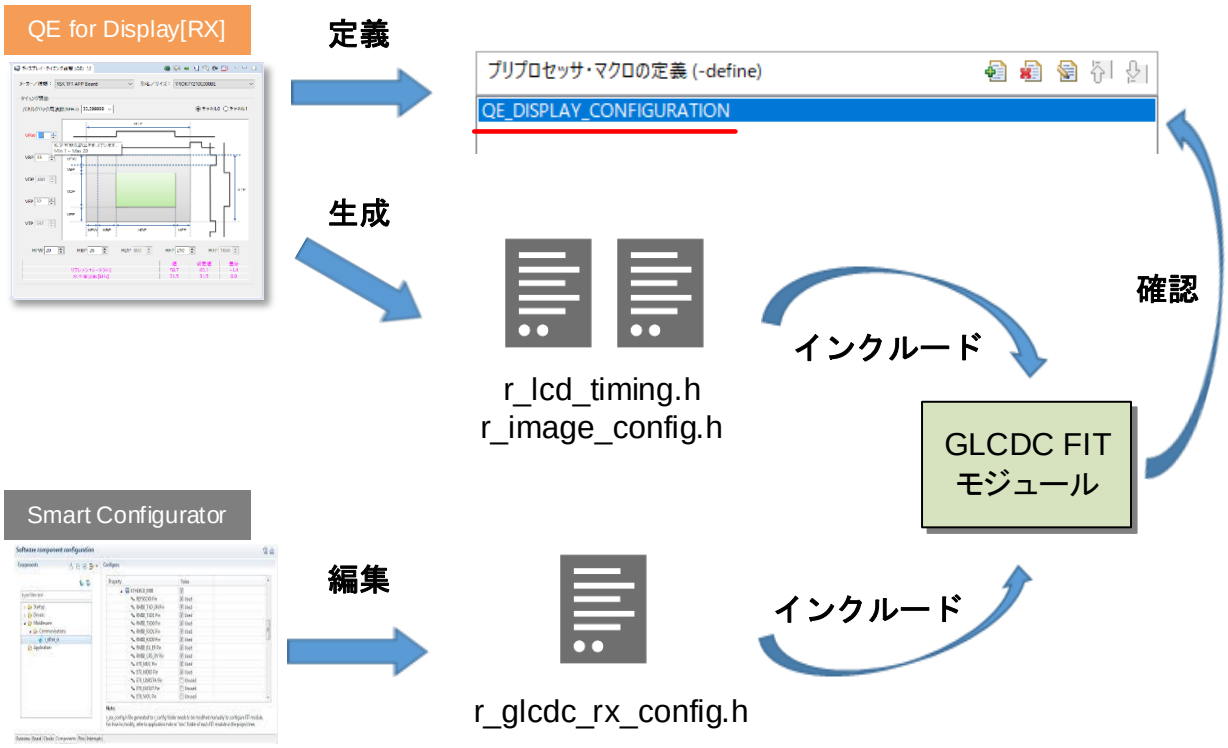

図 5.3　GLCDC FIT モジュールと QE for Display[RX]との連携

6. 付録

6.1 動作確認環境

GLCDC FIT モジュールの動作確認環境を以下に示します。

表 6.1　動作確認環境 (Rev.1.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V6.0.0.001
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.07.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6.2　動作確認環境 (Rev.1.01)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.01

表 6.3　動作確認環境(Rev.1.10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.8.4.201801 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.10
使用ボード	Renesas Starter Kit for RX65N-2MB（型名：RTK500565Nxxxxxxxxx）

表 6.4　動作確認環境(Rev.1.20)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.8.4.201801 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.20

表 6.5　動作確認環境(Rev.1.30)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.8.4.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.30
使用ボード	Renesas Starter Kit for RX72N（型名：RTK5572NNxxxxxxxx）

表 6.6　動作確認環境(Rev.1.40)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2020-04 IAR Embedded Workbench for Renesas RX 4.14.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.02.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0 201904 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.14.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.40
使用ボード	Renesas Envision KIT RPBRX72N (型名: RTK5RX72N0CxxxxxBJ)

表 6.7　動作確認環境(Rev.1.50)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio 2020-10 IAR Embedded Workbench for Renesas RX 4.14.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.02.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0 202004 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.14.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.50
使用ボード	Renesas Envision KIT RPBRX72N (型名: RTK5RX72N0CxxxxxBJ)

表 6.8　動作確認環境(Rev.1.60)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio 2023-10 IAR Embedded Workbench for Renesas RX 5.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.05.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0 202311 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.60
使用ボード	Renesas Envision KIT RPBRX72N (型名: RTK5RX72N0CxxxxxBJ)

表 6.9　動作確認環境(Rev.1.61)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio 2025-01 IAR Embedded Workbench for Renesas RX 5.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.07.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0 202411 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.61
使用ボード	-

6.2　トラブルシューティング

(1) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A：FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_glcdc_rx module.」エラーが発生します。

A：追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

(3) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「… - setting values is out of range defined in r_glcdc_rx_config.h.」エラーが発生します。

A：“r_glcdc_rx_config.h”ファイルの設定値が間違っている可能性があります。
“r_glcdc_rx_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

(4) Q：本 FIT モジュールは Vsync、Hsync と DE の 3 つの信号を設定していますが、Vsync と Hsync のみや、DE のみの LCD にも対応していますか？

A：対応しています。
「4. 端子設定」の方法により、使用する信号の端子設定（MPC の設定）を行うことで、各 LCD パネルへの対応が可能です。端子設定されていない信号は出力されません。
加えて API 関数 R_GLCDC_Open ()で、同期信号の出力端子（output.tcon_hsync, output.tcon_vsync, output.tcon_de）を設定してください。使用しない同期信号には、"GLCDC_TCON_PIN_NON" (出力端子を指定しない)を指定してください。

また DE 信号のみの LCD の場合、DE の表示領域以外の領域（ブランキング期間）は HSYNC と VSYNC のタイミングを調整することで設定してください。計算式は以下の通りです。

水平ブランキング期間 =	htiming.front_porch + htiming.back_porch + htiming.sync_width,
垂直ブランキング期間 =	vtiming.front_porch + vtiming.back_porch + vtiming.sync_width

(5) Q：ライン検出（VPOS 割り込み）の機能がありますが、ライン検出の発生タイミングを教えて下さい。

A： 「5.1 画面の定義」 をご覧ください。

「図 5.1 画面の定義」 で、「制御画面全体」の最終ラインにおいて STHA 信号がアサートされるタイミングで発生します。

(6) Q：画像が期待通り表示されません。

Q-1：LCD パネルに画像が表示されません。
A-1：正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「4 端子設定」を参照してください。

Q-2：画像のデータフォーマット（32bpp、16bpp、8bpp など）を変更すると、期待したように表示できません。

A-2：以下の 3 点のパラメータを確認してください。

1. フレームバッファのデータフォーマット （input.format）

   画像データに応じたデータフォーマットを指定してください
2. 画像の横幅（input.hsize）
3. マクロラインオフセット （input.offset）

マクロラインオフセット（1 ピクセル当たりのバイト数×横幅）が 64 の倍数となるように設定してください。条件を満たすことができない場合、「5.5 マクロラインオフセットの制限を守ることができない場合」をご覧ください。

Q-3：RX MCU をビッグエンディアンに設定した場合、画像がうまく表示されません。

A-3：画像データに対してエンディアン変換を行ってください。エンディアン変換の方法はデータフォーマットによって異なります。ユーザーズマニュアル ハードウェア編「グラフィック LCD コントローラ」、「グラフィック、カラールックアップテーブル（CLUT）のデータフォーマット」を参照してください。

Q-4：画像の色合いが正常ではないです。

A-4：フレームバッファのピクセル順序が ARGB（アルファ値、赤値、緑値、青値）であることを確認してください。また、出力データのピクセル順序（output.color_order）もご確認ください。

(7) Q：コンフィグレーションオプション（r_glcdc_rx_config.h）で設定したとおりに動作しません。

A：コンフィグレーションオプション“GLCDC_CFG_CONFIGURATION_MODE”が“1”であること。または、QE for Display[RX] V2.0.0 以降を使用する場合には define 定義“QE_DISPLAY_CONFIGURATION”が宣言されていることを確認してください。なお、QE for Display[RX]が define 定義“QE_DISPLAY_CONFIGURATION”を宣言するのはヘッダファイルを生成するのと同時です。また、r_glcdc_rx_config.h と QE for Display[RX] が生成したヘッダファイル（r_lcd_timing.h、r_image_config.h）の両方に存在する定義については、QE for Display[RX] が生成したヘッダファイル（r_lcd_timing.h、r_image_config.h）の定義が有効になります。詳細は 5.6 QE for Dispay[RX]との連携 を参照してください。

(8) Q：QE for Display[RX] V2.0.0 以降を使用しなくても、コンフィグレーションオプション（r_glcdc_rx_config.h）で GLCDC FIT を設定することができますか？

A：できます。コンフィグレーションオプション“GLCDC_CFG_CONFIGURATION_MODE”を“1”に設定してください。r_glcdc_rx_config.h に定義されている GLCDC FIT を設定のためのコンフィグレーションオプションが有効になります。詳細は 2.7 コンパイル時の設定 を参照してください。

7. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RX ファミリ RX65N グループ、RX651 グループ ユーザーズマニュアル ハードウェア編
（R01UH0590）
（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。）

RX ファミリ RX72M グループ ユーザーズマニュアル ハードウェア編 （R01UH0804）
（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。）

RX ファミリ RX72N グループ ユーザーズマニュアル ハードウェア編 （R01UH0824）
（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。）

RX ファミリ RX66N グループ ユーザーズマニュアル ハードウェア編 （R01UH0825）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

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

ユーザーズマニュアル：開発環境

RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（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の対応について

GLCDC FIT モジュールが対応しているテクニカルアップデートはありません。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Oct.1.17	-	初版発行
1.01	Feb.1.19	52	表 6.2　動作確認環境 (Rev.1.01)　追加
		-	機能関連 Smart Configurator での GUI によるコンフィグオプション設定機能に対応 • 内容 GUI によるコンフィグオプション設定機能に対応するため、設定ファイルを追加。
1.10	May.31.19	-	GCC、IAR コンパイラのサポートを追加
		1	対象コンパイラを追加
		8	コードサイズのフォーマット変更
		21	「2.13 for 文、while 文、do while 文について」を追加
		28	Special Note にマクロラインオフセット設定時の注意事項を追加
		69	「5.4 内部メインバス 2 優先順位設定について」、 「5.5 マクロラインオフセットの制限を守ることができない場合」追加
		71	表 6.3　動作確認環境 (Rev.1.10)　追加
		74	6.3 トラブルシューティング • (4)～(6)を追加
1.20	May.31.19	-	RX72M グループのサポート追加
1.30	Sep.20.19	-	RX72N, RX66N グループのサポート追加
		4	「1.6 RAM の配置に関する制限事項」を追加
		16	2.9 引数　信号出力端子の定義 • 出力端子を指定しない GLCDC_TCON_PIN_NON を追加
		22	3 API 関数 • 構造体メンバ"output.tcon_hsync", "output.tcon_vsync", "output.tcon_de"の設定値に"GLCDC_TCON_PIN_NON”を追加
		22~48	3 API 関数 • 各 API 関数にある Reentrant の記載を削除。
		64	5.1 画面の定義 • htiming.front_porch と vtiming.front_porch の設定値範囲拡大
		74	6.2 トラブルシューティング • (4)の回答を修正
		ソースコード	• API 関数のヘッダに Doxygen コメントを追加 • モジュールストップ設定ビットの設定(MSTP())とグループ割り込み要求許可/禁止(EN())の前後に、割り込み制御処理の追加
1.40	Jun.30.20	7	2.7 コンパイル時の設定 • QE for Display[RX]に関する情報追加 • GLCDC の設定方法のオプション（GLCDC_CFG_CONFIGURATION_MODE）を追加
		8	2.8 コードサイズ • コードサイズの更新

		20	2.12 FIT モジュールの追加方法 • (5)を追加 • 表記の修正
		22	3. API 関数 • R_GLCDC_Open () < GLCDC 設定データ構造体で設定する場合> を修正
		26	3. API 関数 • 表 3.1 のディザ処理のモード選択 2 に GLCDC_DITHERING_MODE_TRUNCATE を追加
		31	3. API 関数 • R_GLCDC_Open () <コンフィグレーションオプションで設定する場合> を追加
		65	5.6 QE for Dispay[RX]との連携 • 追加
		67	6.1 動作確認環境 • 表 6.6　動作確認環境(Rev.1.40) を追加
		69	6.2 トラブルシューティング • (7)、(8)を追加
		ソースコード	• R_GLCDC_Open 関数 QE for Display[RX]との連携に伴い、インタフェースの仕様を変更 • r_glcdc_rx_config.h GLCDC の設定方法のオプション（GLCDC_CFG_CONFIGURATION_MODE）を追加 • r_glcdc_rx65n.h、r_glcdc_rx66n.h、r_glcdc_rx72m.h、r_glcdc_rx72n.h、r_glcdc_private.c ファイル プリプロセッサの条件式を修正
1.50	Mar.9.21	3	1.3　API の概要 • 表 1.1 API 関数一覧に R_GLCDC_ClutUpdate_ NoReflect 関数を追加
		4	1.4 状態遷移図　更新
		8	2.8 コードサイズ コードサイズの更新
		55	3. API 関数 • R_GLCDC_ClutUpdate_ NoReflect 関数の追加
		71	6.1 動作確認環境 表 6.7　動作確認環境(Rev.1.50) を追加
		ソースコード	• R_GLCDC_ClutUpdate_ NoReflect 関数 新規関数を追加 • r_glcdc_rx_config.h プリプロセッサの条件式を修正
1.60	Jan.30.24	3	1.3　API の概要 表 1.1 API 関数一覧に R_GLCDC_BufferChange 関数を追加
		4	1.4 状態遷移図　更新
		8	2.8 コードサイズ コードサイズの更新

		30	Special Notes:に glcdc_cfg_t 構造体変数使用時の注意事項を追加
		46	• R_GLCDC_LayerChange 関数 Return Values の GLCDC_ERR_INVALID_MODE /*関数が実行できないモードである場合*/を削除 Description の内容を修正
		48,49	• R_GLCDC_BufferChange 関数を追加
		52,53	• R_GLCDC_ColorCorrection 関数 Return Values の GLCDC_ERR_INVALID_MODE /*関数が実行できないモードである場合*/を削除 Description の内容を修正
		55,56	• R_GLCDC_ClutUpdate 関数 Return Values の GLCDC_ERR_INVALID_MODE /*関数が実行できないモードである場合*/を削除 Description の内容を修正
		57	• R_GLCDC_ClutUpdate_NoReflect 関数 Description の内容を修正
		ソースコード	• R_GLCDC_BufferChange 関数 新規関数を追加 • R_GLCDC_ClutUpdate 関数 停止状態(GLCDC_STATE_NOT_DISPLAYING)でも実行できる仕様に変更 • R_GLCDC_ClutUpdate_NoReflect 関数 停止状態(GLCDC_STATE_NOT_DISPLAYING)でも実行できる仕様に変更 • R_GLCDC_LayerChange 関数 停止状態(GLCDC_STATE_NOT_DISPLAYING)でも実行できる仕様に変更 • R_GLCDC_ColorCorrection 関数 停止状態(GLCDC_STATE_NOT_DISPLAYING)でも実行できる仕様に変更 • r_glcdc_rx.if.h #include <stdbool.h>から#include <platform.h>に変更
1.61	Mar.20.25	73	6.1 動作確認環境 表 6.9　動作確認環境(Rev.1.61) を追加
		ソースコード	プログラムの免責事項を変更

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. 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。