# RE01 1500KB グループ、256KB グループ R_GDT ドライバ カラー液晶向けサンプルコード(using CMSIS Driver Package)

## CMSIS Driver Package R_GDT サンプルコード

### 要旨

本アプリケーションノートでは RE01 1500KB グループ、および RE01 256KB グループ CMSIS Driver Package を使用したサンプルコードについて説明します。サンプルコードは同梱されたプロジェクトをご参照ください。

下記に本サンプルコードの概要を示します。

**表　サンプルコードの概要**

| サンプルコードの動作概要 | 主となる周辺機能使用する | 主として使用するドライバ |
|---|---|---|
| 画像処理：<br>GDT 機能を使用し、回転等の画像処理を実施します。<br>GDT へのデータの入力・出力には DMAC を使用します。 | GDT および DMAC を使用 | R_GDT、R_DMAC |
| 画像出力：<br>R_SMIP ドライバを用いて、画像データを LCD に出力します。 | SPI および DMAC を使用 | R_SMIP(R_SPI、R_DMAC) |

サンプルコードのビルド条件によってはプログラムサイズが 32KB を超えます。IAR 社の EWARM をご使用の場合、評価版コンパイラではビルドサイズ制限に該当します。この場合、最適化レベルを高くするか、一部のデモを削除してください。デモを削除する方法は、「3 ソフトウェア説明」をご参照ください。

### 対象デバイス

RE01 1500KB グループ

RE01 256KB グループ

### ご注意

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。

### 関連ドキュメント

RE01 1500KB、256KB グループ CMSIS Package を用いた開発スタートアップガイド(R01AN4660)

# 目次

# 1. 仕様

## 1.1 プロジェクト説明

本アプリケーションノートには以下のサンプルコードプロジェクトが同梱されています。

RE01 1500KB グループ用サンプルコードプロジェクト：an4810_cmsis_gdt_color_re

RE01 256KB グループ用サンプルコードプロジェクト：r01an4810_cmsis_gdt_color_re_256kb

an4810_cmsis_gdt_color_re は、Evaluation Kit RE01 1500KB 上で動作を確認したプロジェクトです。このプロジェクトの設定は Evaluation Kit RE01 1500KB に実装されている R7F0E015D2CFB に合わせています。

r01an4810_cmsis_gdt_color_re_256kb は、Evaluation Kit RE01 256KB 上で動作を確認したプロジェクトです。このプロジェクトの設定は、Evaluation Kit RE01 256KB に実装されている R7F0E01182CFP に合わせています。

その他のデバイスの場合は、プロジェクトの設定でデバイスを変更してご使用ください。

## 1.2 使用端子

以下にサンプルコードが使用する端子を示します。

**表 1-1 RE01 1500KB グループにて使用する端子**

| 使用端子 | 用途 |
|---|---|
| P011：ARDUINO-RSPCKA_B(注) | シリアルクロック |
| P010：ARDUINO-MOSIA_B(注) | シリアルデータ |
| P012：ARDUINO-SSLA0_B(注) | SCS 信号出力 |
| P409：ARDUINO-IO3 | COM 反転信号 |
| P202：ARDUINO-IO7 | I/O ポート　ディスプレイ ON/OFF 制御 |

注　上記の端子を使用する場合、EK ボードの改造が必要です。デフォルトでは接続されておりません。

**表 1-2 RE01 256KB グループにて使用する端子**

| 使用端子 | 用途 |
|---|---|
| P011：ARDUINO_RSPCKA_B | シリアルクロック |
| P010：ARDUINO_MOSIA_B | シリアルデータ |
| P015：ARDUINO_SSLA1_B | SCS 信号出力 |
| P603：ARDUINO_IO5 | COM 反転信号 |
| P302：ARDUINO_IO9 | I/O ポート　ディスプレイ ON/OFF 制御 |

## 1.3 フォルダ構成

サンプルコード、およびサンプルコードで使用しているドライバの、フォルダ構成を示します。

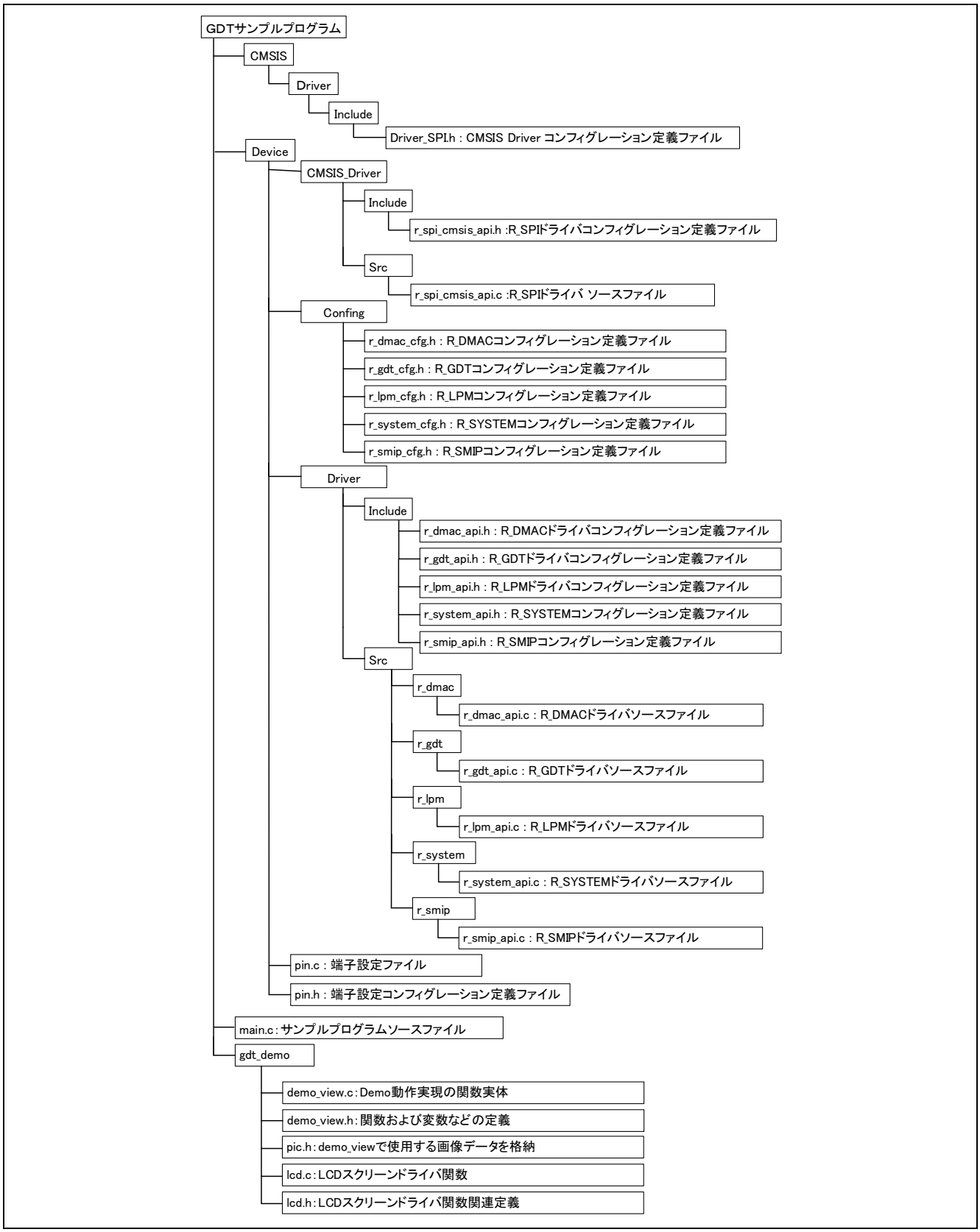



図 1.1 フォルダ構成

## 1.4 ファイル構成

表 1-3 にサンプルコードで追加・変更したファイルを示します。

**表 1-3 サンプルコードで追加・変更したファイル**

| ファイル名 | 処理・設定概要 | 備考 |
|---|---|---|
| main.c | メイン処理 | |
| r_system_cfg.h | システム設定 | 周辺機能の割り込み設定 |
| r_smip_cfg.h | SMIP コンフィグレーション設定 | 端子設定<br>COM 反転信号周期設定 |
| r_spi_cfg.h | SPI コンフィグレーション設定 | 送信制御（DMAC）設定 |
| r_dmac_cfg.h | DMAC コンフィグレーション設定 | DMAC1 の割り込み優先度設定 |
| pin.c | 端子設定 | SPI0 端子設定 |
| demo_view.c | Demo 動作実働部（r_GDT API 呼び出し部） | |
| demo_view.h | 関数および変数などの定義 | |
| pic.h | demo_view で使用する画像データを格納 | |
| lcd.c | LCD スクリーンドライバ関数 | |
| lcd.h | LCD スクリーンドライバ関数関連定義 | |

## 1.5 オプション設定メモリ

表 1-4 にサンプルコードで使用するオプション設定メモリの状態を示します。必要に応じて、お客様のシステムに最適な値を設定してください。

**表 1-4 サンプルコードで使用するオプション設定メモリ**

| シンボル | アドレス | 設定値 | 内容 |
|---|---|---|---|
| AWS | 0100A164h～0100A167h | FFFF FFFFh | アクセスウィンド設定無し |
| OSIS | 0100A150h～0100A15Fh | FFFF FFFFh | ID コードプロテクト無し（ALL FFh） |
| SECMPUxxx | 00000408h~0000043Bh | FFFF FFFFh | MPU 無効 |
| OFS1 | 00000404h~00000407h | FFFF FFFFh | リセット後、電圧監視 0 リセット無効<br>リセット後、HOCO 発振無効 |
| OFS0 | 00000400h~00000403h | FFFF FFFFh | IWDT 自動起動無効<br>WDT 自動起動無効 |

## 2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記（表 2-1、表 2-2）の条件で動作を確認しています。

**表 2-1 動作確認条件（RE01 1500KB グループ）**

| 項目 | | 内容 |
|---|---|---|
| 使用マイコン | | R7F0E015D2CFB 144pin |
| 動作周波数 | システムクロック: HOCO を選択 | • システムクロック(ICLK) : 32MHz (HOCO 1 分周)<br>• 周辺モジュールクロック A(PCLKA) : 32MHz (HOCO 1 分周)<br>• 周辺モジュールクロック B(PCLKB) : 32MHz (HOCO 1 分周) |
| AGT タイマ動作クロック | | サブクロック発振器(SOSC) : 32.768kHz |
| 動作電圧 | | 3.3V |
| 統合開発環境 | IAR | IAR Embedded Workbench for ARM Version 8.40.2<br>C コンパイラ：IAR C/C++ Compiler for ARM Version 8.40.2 |
| | e² studio | Renesas e² studio Version 7<br>C コンパイラ：GCC ARM Embedded Version 6.3.1.20170620<br>GNU 6-2017-q2-update |
| デバッガ | | Segger J-Link OB |
| 最適化設定 | | 最適化レベル "中" |
| ターゲットボード | | Evaluation Kit RE01 1500KB（型名：RTK70E015DSXXXXXBE） |
| MIP-LCD 接続用ボード | | スイッチサイエンス製：MIP 反射型液晶中継ボード |
| MIP LCD | | JDI 製 : LPM013M126A |
| CMSIS Driver Package のバージョン | | Rev1.10 |
| サンプルコードのバージョン | | Rev1.02 |

**表 2-2 動作確認条件（RE01 256KB グループ）**

| 項目 | | 内容 |
|---|---|---|
| 使用マイコン | | R7F0E01182CFP 100pin |
| 動作周波数 | システムクロック: HOCO を選択 | • システムクロック(ICLK) : 32MHz (HOCO 1 分周)<br>• 周辺モジュールクロック A(PCLKA) : 32MHz (HOCO 1 分周)<br>• 周辺モジュールクロック B(PCLKB) : 32MHz (HOCO 1 分周) |
| AGT タイマ動作クロック | | サブクロック発振器(SOSC) : 32.768kHz |
| 動作電圧 | | 3.3V |
| 統合開発環境 | IAR | IAR Embedded Workbench for ARM Version 8.40.2<br>C コンパイラ：IAR C/C++ Compiler for ARM Version 8.40.2 |
| | e² studio | Renesas e² studio 2020-07<br>C コンパイラ：GCC ARM Embedded Version 6.3.1.20170620<br>GNU 6-2017-q2-update |
| デバッガ | | Segger J-Link OB |
| 最適化設定 | | 最適化レベル "中" |
| ターゲットボード | | Evaluation Kit RE01 256KB（型名：RTK70E0118CXXXXXBJ） |
| MIP-LCD 接続用ボード | | スイッチサイエンス製：MIP 反射型液晶中継ボード |
| MIP LCD | | JDI 製 : LPM013M126A |
| CMSIS Driver Package のバージョン | | Rev1.00 |
| サンプルコードのバージョン | | Rev1.03 |

# 3. ソフトウェア説明

本サンプルコードは、R_GDTドライバを用いた画像変換を行い、カラーLCDに画像を出力します。
GDTへの画像入力・出力にはDMACを使用しています。カラーLCDへのデータ出力にはR_SMIPドライバを使用しています。LCDには以下のデモを繰り返し表示します。(注1)

- INTRO
- FUNCTION
- CALENDAR
- GRAPH

サンプルコードの動作を以下に示します。

□初期化処理

関連の初期化処理を実施します。

1)システム初期化処理

2)システムクロック設定

3)IO 電源制御（全 IO 電源供給）

4) R_SMIP ドライバによる MLCD 表示初期設定(API 関数 R_SMIP_Open()、R_SMIP_PowerOn()を実行)

□API 関数 R_GDT_Open()を実行し、GDT のクロックを ON にする。

□DMAC チャネル選択

API 関数 R_GDT_DmacChSel()を実施し、DMAC 使用するチャネルを設定

□繰り返し下記の 4 つの機能を実施。(注 1)

1) INTRO

2) FUNCTION

3) CALENDAR

4) GRAPH

□API 関数 R_GDT_Close()を実行し、GDT のクロックを OFF にする。

**表 3-1 サンプルプログラムの動作情報**

| 項目 | 設定値 |
|---|---|
| SPI チャネル選択 | SPI CH0 |
| SPI ビットレート | 1Mbps |
| SMIP COM 反転信号 | 125Hz |
| SMIP AGT チャネル設定 | CH0 |
| SMIP AGT 動作クロック | SOCO (32.768kHz) |
| ディスプレイ ON/OFF 制御 | ON |
| GDT 用の DMAC チャネル設定 | DMAC CH1:RAM 上の画像データを GDT に転送<br>DMAC CH2:GDT の変換データを RAM に転送<br>(CH0～CH3 から任意のチャネルを設定可能) |
| SMIP(SPI)用の DMAC チャネル設定 | DMAC CH0<br>(CH0～CH3 から任意のチャネルを設定可能) |

注1. サンプルコードのビルド条件によってはプログラムサイズが32KBを超えます。IAR社のEWARMをご使用の場合、評価版コンパイラではビルドサイズ制限に該当します。この場合、最適化レベルを高くするか、一部のデモを削除してください。表示するデモは以下の定義をコメントアウトすることで削除できます。

| 定義名 | デモ | 初期値 |
|---|---|---|
| FUNCTION_DEMO_ENABLE | FUNCTION | 有効（デモを表示する） |
| CALENDAR_DEMO_ENABLE | CALENDAR | 無効（デモを表示しない） |
| GRAPH_DEMO_ENABLE | GRAPH | 無効（デモを表示しない） |

## 3.1 システム構成図

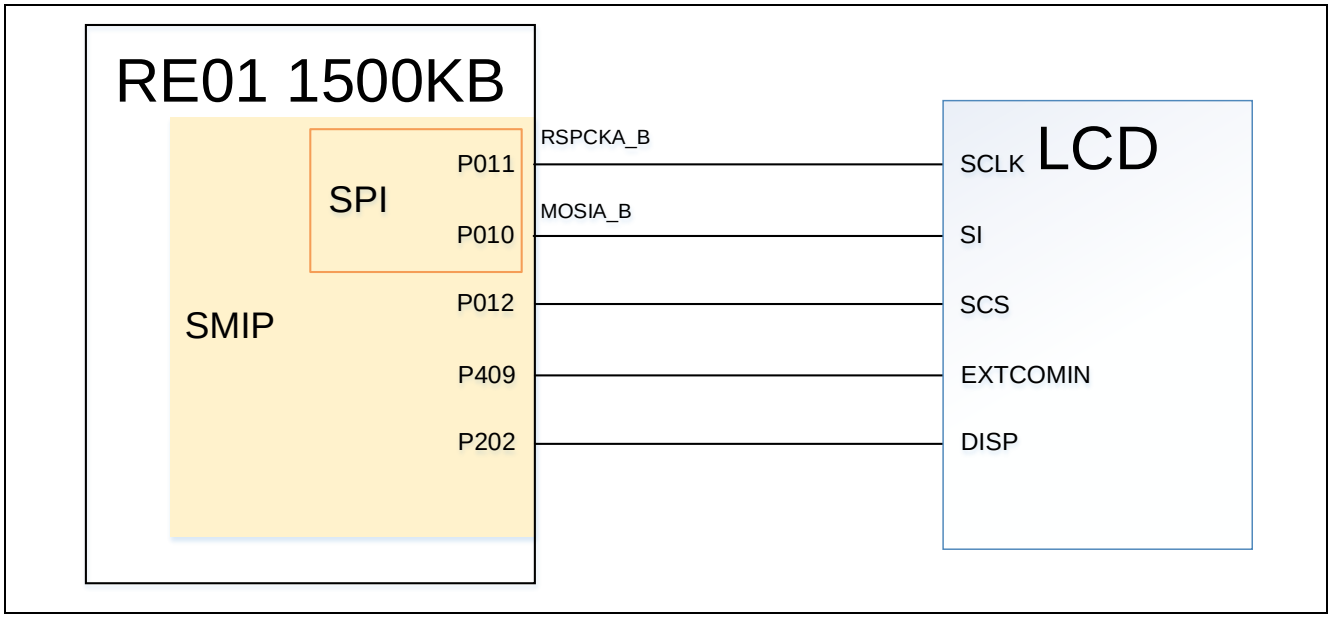



図 3.1 システム構成図(RE01 1500KB グループ)

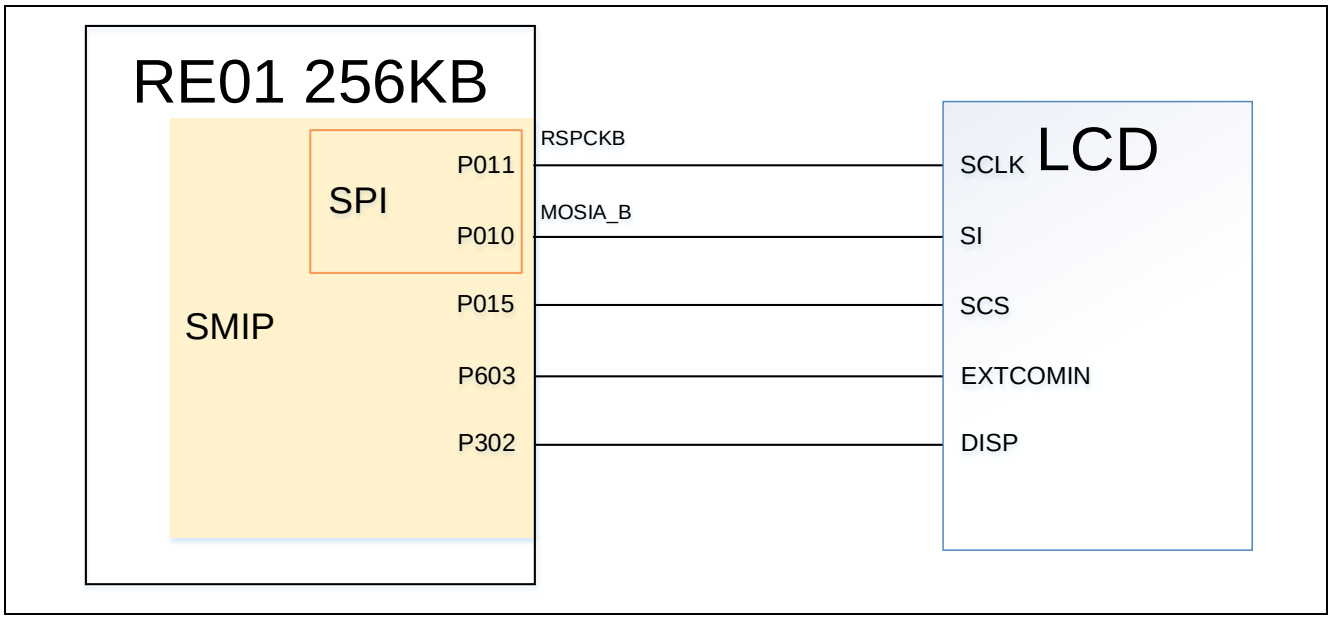
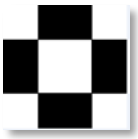
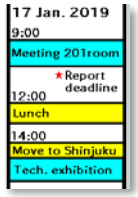
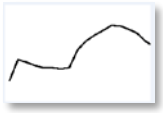
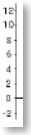



図 3.2 システム構成図(RE01 256KB グループ)

## 3.2 関数一覧

サンプルコードで追加した関数を紹介する前に、まず関数で使用するROMの配列と内部保存するデータのイメージを示します。

| ROM の配列名 | データイメージ |
|---|---|
| img_renesas_176x176[3][3872] |  |
| img_menu_176x176[3][3872] |  |
| img_car_176x176[3872] |  |
| img_sqr_176x176[3872] |  |
| img_calendar_176x128[2816] |  |
| img_calendar_top_176x48[3][1056] |  |
| img_calendar_tip_176x263[3][5808] |  |
| img_graph_256x176_1[5632] |  |
| img_graph_256x176_2[5632] |  |
| img_graph_256x176_3[5632] |  |

| | |
|---|---|
| img_graph_144x32[3][576] |  |
| img_graph_144x16[288] |  |
| img_graph_32x176[704] |  |

サンプルコードの中の関数で使用する RAM の配列を以下に示します。

| RAM 配列名 | 用途 |
|---|---|
| gdt_img_buf0[3][GDT_IMG_BUF_H*GDT_IMG_BUF_V] | データー時保存用 |
| gdt_img_buf1[3][GDT_IMG_BUF_H*GDT_IMG_BUF_V] | データー時保存用 |
| gdt_img_buf2[3][GDT_IMG_BUF_H*GDT_IMG_BUF_V] | データー時保存用 |
| gdt_img_color_lcd_out[RAM_LCD_MEM_H*RAM_LCD_MEM_V] | LCD 出力データ保存用配列 |

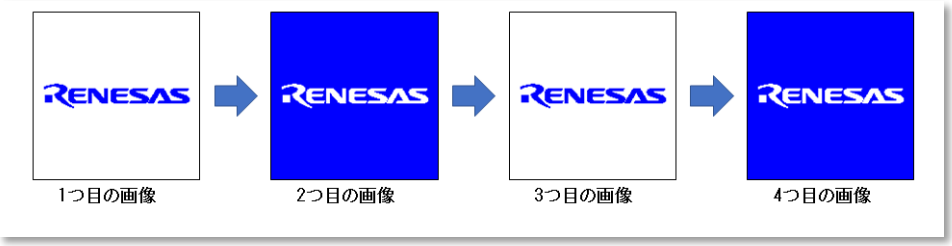
サンプルコードで追加した関数について説明します。必要に応じて各関数の内容修正してください。

### main

概　要　　メイン処理
ヘッダ　　なし
宣　言　　void main(void)
説　明　　システムクロック設定や R_SMIP ドライバ、DMAC などの転送設定を行います。
　　INTRO、FUNCTION、CALENDAR、GRAPH 繰り返して実施します。
引　数　　なし
リターン値　　なし

### intro

概　要　　イントロ機能のデモ実現
ヘッダ　　なし
宣　言　　void view_int(void)
説　明　　下記の機能を実現します。




詳細の実現方法を以下に示します。

1. 入力画像保存：
   GDT に入力する画像（img_renesas_176x176[3][3872]）の RGB データを一旦 2 つの GDT イメージバッファに保存。(2 つのバッファの保存内容は同じ)。

2. 1 つ目の画像の出力処理：入力画像そのまま出力
GDT イメージバッファに保存した RGB データをそのまま GDT に入力し、GDT がカラーデータ整列処理(Img_Out_Coloralign)を行い、SPI で画像を LCD に出力(LCD_Output_3bit)

3. 2 つ目の画像の出力処理：入力画像を反転→カラーデータ整列→出力
GDT イメージバッファに保存した RGB データを GDT に入力し、GDT が反転処理(gdt_iflp_256x176)を行ってからデータを、再度 GDT に入力しカラーデータ整列処理(Img_Out_Coloralign)を行う。最後 SPI で画像を LCD に出力(LCD_Output_3bit)

4. 3 つ目の画像の出力処理：2 つ目の画像を反転→カラーデータ整列→出力
処理内容は「3. 2 つ目の画像の出力処理」と同じ、ただし、GDT に入力する RGB データは「3. 2 つ目の画像の出力処理」でカラーデータ整列処理後のデータになる。

5. 4 つ目の画像の出力処理：3 つ目の画像を反転→カラーデータ整列→出力
処理内容は「4. 3 つ目の画像の出力処理」と同じ、ただし、GDT に入力する RGB データは「4. 3 つ目の画像の出力処理」でカラーデータ整列処理後のデータになる。

| | |
|---|---|
| 引　数 | なし |
| リターン値 | なし |

## function

| | |
|---|---|
| 概　要 | ファクションデモ機能を実現<br>カラー整列のデモになる。GDT の反転、縮小、合成、回転、スクロール機能を使用して画像処理を行う。 |
| ヘッダ | なし |
| 宣　言 | void function(void) |
| 説　明 | 下記の機能を実現します。<br>・ファクションのアイコンを 90° 右回転 1 周<br>・入力画像(車)を 1 回反転実施。<br>・入力画像(車)を縮小(8/8(元画像)→6/8→4/8→2/8)<br>・入力画像(車)と背景画像と合成<br>・入力画像(車)を回転（1/4 元画像→右回転→左回転→上下反転）<br>・入力画像(車)を 7 ビットスクロール 3 回実施 |

詳細の実現方法を以下に示します。

1. 関数のアイコンを 90° 右回転 1 周


1)入力画像保存：
GDT に入力する画像（img_menu_176x176[3][3872]）の RGB データを一旦 2 つの GDT イメージバッファに保存。(2 つのバッファの保存内容は同じ)。
2)入力画像そのまま出力

GDT イメージバッファに保存した RGB データを GDT に入力し、GDT がカラーデータ整列処理(Img_Out_Coloralign)を行い、SPI で画像を LCD に出力(LCD_Output_3bit)

3) アイコンを右 90 度回転してから出力
GDT イメージバッファに保存した RGB データを GDT に入力し、GDT が回転処理(menu_rotate)及びカラーデータ整列処理(Img_Out_Coloralign)を行ってから、SPI で画像を LCD に出力(LCD_Output_3bit)

4) アイコンをさらに右 90 度（元画像の 180 度）回転してから出力
処理方法は上記の”2)”番と同じ。

5) アイコンをさらに右 90 度（元画像の 270 度）回転してから出力
処理方法は上記の”2)”番と同じ。

6) アイコンをさらに右 90 度（元画像の 360 度）回転してから出力
処理方法は上記の”2)”番と同じ。

2. 入力画像(車)を 1 回反転実施

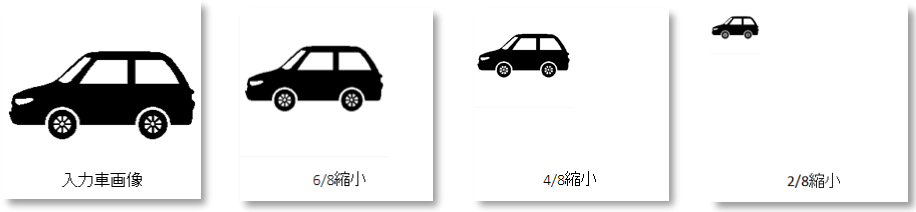
反転前　　反転後

1) 入力画像保存：
GDT の入力画像車（img_car_176x176[3872]）と背景画像(img_sqr_176x176[3872])の RGB データを一旦 2 つの GDT イメージバッファに保存。
補足：背景画像は後記の「4. 入力画像(車)と背景画像と合成」で使用。

2) 車画像そのまま出力
GDT イメージバッファに保存した車画像データをそのまま SPI で画像を LCD に出力(LCD_Output_1bit)

3) 車画像反転してから出力
GDT イメージバッファに保存した車画像データを GDT に入力し、GDT が反転処理(gdt_iflp_256x176)を行ってから、SPI で画像を LCD に出力(LCD_Output_1bit)

3. 入力画像(車)を縮小(8/8(元画像)→6/8→4/8→2/8)




1) 車画像そのまま出力
GDT イメージバッファに保存した車画像データをそのまま SPI で画像を LCD に出力(LCD_Output_1bit)

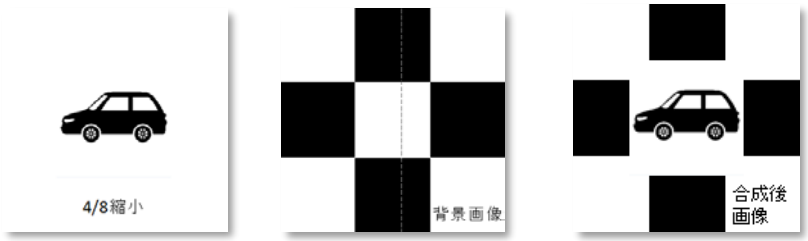
2) 車画像 6/8 縮小してから出力

GDT イメージバッファに保存した車画像データを GDT に入力し、GDT で 6/8 縮小処理(Gdt_shrink_func)を行ってから SPI で画像を LCD に出力(LCD_Output_1bit)

3)車画像 4/8 縮小してから出力
6/8 縮小と同じ、ただし縮小倍率は 4/8。

4)車画像 2/8 縮小してから出力
6/8 縮小と同じ、ただし縮小倍率は 2/8。

4. 入力画像(車)と背景画像と合成




1)4/8 縮小した車画像を背景画像と合成
「3. 入力画像(車)を縮小」で 4/8 縮小した車画像データと背景画像を GDT に入力し、合成処理(gdt_monochrome_uniflp)を行ってから SPI で画像を LCD に出力 (LCD_Output_1bit)

5. 入力画像(車)を回転 (1/2 元画像→右回転→左回転→上下反転)


1)2/8 縮小した車画像を出力
「3. 入力画像(車)を縮小」で 2/8 縮小した車画像データを SPI で画像を LCD に出力 (LCD_Output_1bit)

2)2/8 縮小した車画像を右回転して出力に加える
2/8 縮小した車画像データを右回転(gdt_menu_icon_rotate)して SPI で画像を LCD に出力 (LCD_Output_1bit)

3)2/8 縮小した車画像を左回転して出力に加える
2/8 縮小した車画像データを左回転(gdt_menu_icon_rotate)して SPI で画像を LCD に出力 (LCD_Output_1bit)

4)2/8 縮小した車画像を上下反転して出力に加える
2/8 縮小した車画像データを左回転(gdt_menu_icon_rotate)して SPI で画像を LCD に出力 (LCD_Output_1bit)

6. 入力画像(車)を 7 ビットスクロール 3 回実施

1)車画像そのまま出力
GDT イメージバッファに保存した車画像データをそのまま SPI で画像を LCD に出力(LCD_Output_1bit)

2)左右スクロール実施
ステップ 1：左スクロール実施
GDT のスクロール機能(gdt_scroll_uniflp)を使って、車画像を 7bit ずつ左シフト実施。 三回ごとに、シフトした画像を SPI で LCD に出力(LCD_Output_1bit)。
ステップ 2：右スクロール実施

GDT のスクロール機能(gdt_scroll_uniflp)を使って、車画像を 7bit ずつ右シフトするように、下記の順番で左シフトを行う。三回ごとに、シフトした画像を SPI で LCD に出力(LCD_Output_1bit)。

- 車画像の始点座標（168,0）からサイズ 8x176 の左 1bit スクロール
- 車画像の始点座標（160,0）からサイズ 16x176 の左 2bit スクロール
- 車画像の始点座標（152,0）からサイズ 24x176 の左 3bit スクロール出力
- 車画像の始点座標（144,0）からサイズ 32x176 の左 4bit スクロール
- 車画像の始点座標（136,0）からサイズ 40x176 の左 5bit スクロール
- 車画像の始点座標（128,0）からサイズ 48x176 の左 6bit スクロール出力
- 車画像の始点座標（120,0）からサイズ 56x176 の左 7bit スクロール
- 車画像の始点座標（112,0）からサイズ 64x176 を直接取出し
- 車画像の始点座標（104,0）からサイズ 72x176 の左 1bit スクロール出力
- 車画像の始点座標（96,0）からサイズ 80x176 の左 4bit スクロール
- 車画像の始点座標（88,0）からサイズ 88x176 の左 5bit スクロール
- 車画像の始点座標（80,0）からサイズ 96x176 の左 6bit スクロール出力
- 車画像の始点座標（72,0）からサイズ 104x176 の左 4bit スクロール
- 車画像の始点座標（64,0）からサイズ 112x176 の左 5bit スクロール
- 車画像の始点座標（56.0）からサイズ 120x176 の左 6bit スクロール出力
- 車画像の始点座標（48,0）からサイズ 128x176 の左 4bit スクロール
- 車画像の始点座標（40,0）からサイズ 136x176 の左 5bit スクロール
- 車画像の始点座標（32,0）からサイズ 144x176 の左 6bit スクロール出力
- 車画像の始点座標（24,0）からサイズ 152x176 の左 4bit スクロール
- 車画像の始点座標（16,0）からサイズ 160x176 の左 5bit スクロール
- 車画像の始点座標（8,0）からサイズ 168x176 の左 6bit スクロール出力

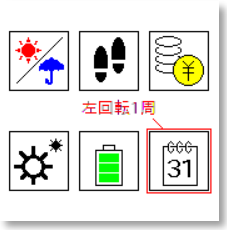
| | |
|---|---|
| 引　数 | なし |
| リターン値 | なし |

## calendar

| | |
|---|---|
| 概　要 | カレンダー関数(カラー化のデモ) |
| ヘッダ | なし |
| 宣　言 | void calendar(void) |
| 説　明 | 下記の機能を実現します。<br>・カレンダーアイコンを 90° 左回転 1 周<br>・カレンダー表示（カラー化、整列）<br>・スケジュール表示(縮小、整列)<br>・スケジュール拡大表示（整列） |

詳細の実現方法を以下に示します。

1. カレンダーのアイコンを 90° 左回転 1 周




1) 入力画像保存：
GDT に入力する画像（img_menu_176x176[3][3872]）の RGB データを一旦 2 つの GDT イメージバッファに保存。(2 つのバッファの保存内容は同じ)。

2) 入力画像そのまま出力
GDT イメージバッファに保存した RGB データを GDT に入力し、GDT がカラーデー

タ整列処理(Img_Out_Coloralign)を行い、SPI で画像を LCD に出力(LCD_Output_3bit)

3) アイコンを左 90 度回転してから出力
GDT イメージバッファに保存した RGB データを GDT に入力し、GDT が回転処理(menu_rotate)及びカラーデータ整列処理(Img_Out_Coloralign)を行ってから、SPI で画像を LCD に出力(LCD_Output_3bit)

4) アイコンをさらに左 90 度（元画像の 180 度）回転してから出力
処理方法は上記の”2)”番と同じ。

5) アイコンをさらに左 90 度（元画像の 270 度）回転してから出力
処理方法は上記の”2)”番と同じ。

6) アイコンをさらに左 90 度（元画像の 360 度）回転してから出力
処理方法は上記の”2)”番と同じ。

2. カレンダー表示（カラー化、整列）
カレンダーの年月画像と日付画像と組み合わせて、カラー化及び整列実施


年月画像　　日付画像　　カラー化整列後

1) 入力画像保存：
日付画像(img_calendar_176x128[2816])の RGB データを一旦 GDT イメージバッファに保存。(開始座標：(48,0)、範囲：176×128)。
年月画像(img_calendar_top_176x48[3][1056])の RGB データを一旦 GDT イメージバッファに保存。(開始座標：(0,0)、範囲：176×48)。

2) カレンダーのカラー化：
イメージバッファに保存した年月画像と日付画像データを GDT に入力して、カラー化(gdt_color)を実施
・カラー化(赤)
日曜日の列の数字を赤くする
開始座標：(0,48)、範囲：24×128
・カラー化(青)
土曜日の列の数字を青くする
開始座標：(152,48)、範囲：24×128
・カラー化(黒)
平日の列の数字を黒くする
開始座標：(24,48)、範囲：128×128

3) カレンダーの整列化：
イメージバッファに保存したカレンダーデータを GDT に入力して、整列処理(Img_Out_Coloralign)をしてから、SPI で画像を LCD に出力(LCD_Output_3bit)。

3. スケジュール表示(縮小、整列)


スケジュールデータ(出力しない)


処理後のデータ(LCD 出力)

1) 入力画像保存：
スケジュールデータ(縦長 2 画面分、img_calendar_tip_176x263[3][5808])の RGB を一旦 GDT の 2 個のイメージバッファに保存。(イメージバッファはそれぞれ 1 画面分を保存)。
2) スケジュール画像 5/8 縮小してから整列、出力
GDT イメージバッファに保存したスケジュール画像データを GDT に入力し、GDT で 5/8 縮小処理(Gdt_shrink_func)を行ってから、RGB 整列(Img_Out_Coloralign)後、SPI で画像を LCD に出力(LCD_Output_3bit)

4. スケジュール拡大表示（整列）
イメージバッファに保存したスケジュールデータを 3 つの画面として出力。


1 回目


2 回目


3 回目

1) 1 回スケジュール目出力
GDT イメージバッファに保存したスケジュール画像データを 1 回目分切り出し、GDT に入力し RGB 整列(Img_Out_Coloralign)後、SPI で画像を LCD に出力(LCD_Output_3bit)
2) 2 回スケジュール目出力
GDT イメージバッファに保存したスケジュール画像データを 2 回目分切り出し、GDT に入力し RGB 整列(Img_Out_Coloralign)後、SPI で画像を LCD に出力(LCD_Output_3bit)
3) 3 回スケジュール目出力
GDT イメージバッファに保存したスケジュール画像データを 3 回目分切り出し、GDT に入力し RGB 整列(Img_Out_Coloralign)後、SPI で画像を LCD に出力(LCD_Output_3bit)

引　数　　なし
リターン値　　なし

---

graph

---

概　要　　グラフ表示関数(カラー合成のデモ)
ヘッダ　　なし

宣　言　　void graph(void)

説　明　　下記の機能を実現します。

・天気のアイコンを左右反転 4 回
・グラフ画像/横軸画像カラー化、合成
・凡例画像/横軸タイトル画像/縦軸画像カラー化、合成
・合成後グラフ画像/横軸画像のスクロール

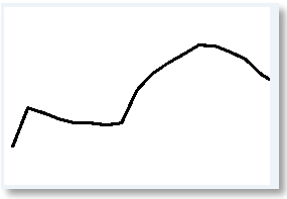
詳細の実現方法を以下に示します。

1. 天気のアイコンを左右反転 4 回

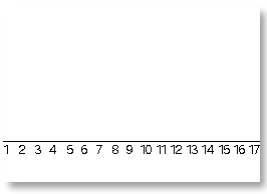

1)入力画像保存：
GDT に入力する画像（img_menu_176x176[3][3872]）の RGB データを一旦 2 つの GDT イメージバッファに保存。(2 つのバッファの保存内容は同じ)。
2)入力画像そのまま整列して出力
GDT イメージバッファに保存した RGB データを GDT に入力し、GDT がカラーデータ整列処理(Img_Out_Coloralign)を行い、SPI で画像を LCD に出力(LCD_Output_3bit)
3)アイコンを左右反転してから出力（1 回目）
GDT イメージバッファに保存した RGB データを GDT に入力し、GDT が GRAPH アイコンを左右反転処理(menu_rotate)及びカラーデータ整列処理(Img_Out_Coloralign)を行ってから、SPI で画像を LCD に出力(LCD_Output_3bit)
4) アイコンを左右反転してから出力（2 回目）
1 回目と同じ。
5) アイコンを左右反転してから出力（3 回目）
1 回目と同じ。
6) アイコンを左右反転してから出力（4 回目）
1 回目と同じ。

2. グラフ画像/横軸画像カラー化、合成

グラフ画像(風速)



グラフ画像(温度)



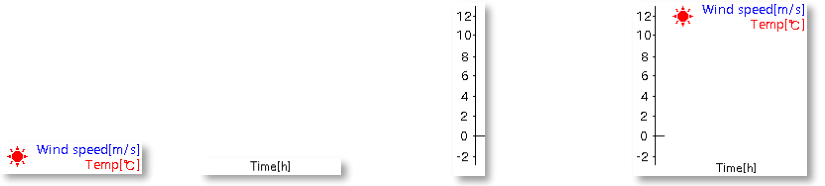

横軸画像

1) 風速グラフ画像カラー化
風速グラフ画像（img_graph_256x176_1[5632]）のデータを一旦 GDT イメージバッファに保存し、青カラー化(gdt_color)を行う。
2) 温度グラフ画像カラー化後、風速グラフとカラー合成
温度グラフ画像（img_graph_256x176_2[5632]）のデータを一旦 GDT イメージ

バッファに保存し、赤カラー化(gdt_color)を行う。
カラー化後の風速グラフ画像を温度グラフ画像とカラー合成(gdt_colorsync)を行う。

3) 横軸画像カラー化後、グラフ画像とカラー合成
横軸画像(img_graph_256x176_3[5632])のデータを一旦 GDT イメージバッファに保存し、黒カラー化(gdt_color)を行う。
カラー化後の横軸画像をグラフ画像(温度と風速で合成したもの)とカラー合成(gdt_colorsync)を行う。

3. 凡例画像/横軸タイトル画像/縦軸画像カラー化、合成




凡例画像　横軸タイトル画像　縦軸画像　合成後画像イメージ

1) 凡例画像データ保存
凡例画像（img_graph_144x32[3][576]）のデータを一旦 GDT イメージバッファに保存。
2) 横軸タイトル画像保存、黒カラー化
横軸タイトル画像(img_graph_144x16[288])のデータを一旦 GDT イメージバッファに保存し、黒カラー化(gdt_color)。
3) 縦軸画像カラー化実施
縦軸画像（img_graph_32x176[704）のデータを一旦 GDT イメージバッファに保存し、黒カラー化(gdt_color)を行う。
4) 凡例・横軸タイトル・縦軸合成後の出力
凡例画像、横軸タイトル画像、縦軸画像をカラー合成(gdt_colorsync)してから、カラーデータ整列処理(Img_Out_Coloralign)を行ってから、SPI で画像を LCD に出力(LCD_Output_3bit)

4. 合成後グラフ画像/横軸画像のスクロール：
1) グラフと横軸画像スクロール
「2. グラフ画像/横軸画像カラー化、合成」で合成したグラフ画像/横軸画像を左 7bit スクロール(gdt_scroll_uniflp)実施。
2)グラフ・横軸及び凡例・時間・縦軸カラー合成
スクロール後のグラフと横軸画像を凡例・時間・縦軸の画像とカラー合成(gdt_colorsync)してから、カラーデータ整列処理(Img_Out_Coloralign)を行ってから、SPI で画像を LCD に出力(LCD_Output_3bit)

| | |
|---|---|
| 引　数 | なし |
| リターン値 | なし |

## menu_rotate

| | |
|---|---|
| 概　要 | メニューの中のアイコンを回転用の関数<br>開始座標と回転モードを設定しアイコンの回転を実施 |
| ヘッダ | なし |
| 宣　言 | void menu_rotate(icon_rotate_info_t *icon_rotate_info) |
| 説　明 | 1. 入力画像座標設定 |

2. 出力画像座標設定
3. gdt_menu_icon_rotate を使って回転を実施。入力画像アドレス、出力画像アドレス、座標、回転モードの設定。
4. カラーデータ整列処理 (Img_Out_Coloralign())
5. SPI で画像を LCD に出力 (LCD_Output_3bit())

| | | |
|---|---|---|
| 引　数 | icon_rotate_info | アイコン設定パラメータ：入力画像アドレス、出力画像アドレス、座標、回転モード |
| リターン値 | なし | |

## gdt_iflp_256x176

| | |
|---|---|
| 概　要 | 256x176 全画面反転機能関数(デモ用)<br>エンディアン変換処理の API 関数(R_GDT_Nochange)の引数を設定し、R_GDT_Nochange 関数を呼び出します |
| ヘッダ | なし |
| 宣　言 | void gdt_iflp_256x176(uint32_t src_img_addr,uint32_t dest_img_addr) |
| 説　明 | 1. エンディアン変換入力画像のサイズ設定<br>2. エンディアン変換出力画像のサイズ設定<br>3. エンディアン変換入力画像および出力画像の開始座標を(0,0)に固定、サイズを 256 x 176 に固定。<br>4. エンディアン変換処理の設定<br>1) 反転機能 ON<br>5. R_GDT_ Nochange 関数でエンディアン変換実施<br>6. エンディアン変換処理の終了フラグ(GDT_end_flag)を待つ<br>7. エンディアン変換処理の終了フラグ(GDT_end_flag)をクリア |
| 引　数 | uint32_t src_img_addr　入力画像アドレス<br>uint32_t dest_img_addr　出力画像アドレス |
| リターン値 | なし |

## gdt_nochange_256x176

| | |
|---|---|
| 概　要 | 256x176 全画面反転機能関数(デモ用)<br>エンディアン変換処理の API 関数(R_GDT_Nochange)の引数を設定し、R_GDT_Nochange 関数を呼び出します |
| ヘッダ | なし |
| 宣　言 | void gdt_nochange_256x176(nochange_info_t nochange_info) |
| 説　明 | 1. エンディアン変換入力画像のサイズ設定<br>2. エンディアン変換出力画像のサイズ設定<br>3. 入力画面の開始点座標と出力画面の開始点座標<br>4. エンディアン変換処理の設定<br>1) 反転機能 OFF<br>5. R_GDT_ Nochange 関数でエンディアン変換実施<br>6. エンディアン変換処理の終了フラグ(GDT_end_flag)を待つ<br>7. エンディアン変換処理の終了フラグ(GDT_end_flag)をクリア |
| 引　数 | nochange_info |
| リターン値 | なし |

## gdt_menu_icon_rotate

| | |
|---|---|
| 概　要 | 全画面回転機能関数(デモ用) |

| | |
|---|---|
| | 回転機能の API 関数(R_GDT_Rotate)の引数を設定し、R_GDT_Rotate 関数を呼び出します |
| ヘッダ | なし |
| 宣　言 | gdt_menu_icon_rotate(uint32_t src_img_addr,uint32_t dest_img_addr,st_blk_conv_info_t *blk_conv_info,uint32_t rotate_mode) |
| 説　明 | 1. 回転入力画像のサイズ設定<br>2. 回転出力画像のサイズ設定<br>3. 回転入力画像の開始座標、回転する部分のサイズを設定、回転出力画像の開始座標を設定<br>4. 回転処理の設定<br>1) 反転機能 OFF<br>2) 回転機能 ON<br>3) 回転機能選択ビットの設定<br>5. R_GDT_Rotate 関数で回転処理を実施<br>6. 回転処理の終了フラグ(GDT_end_flag)を待つ<br>回転処理の終了フラグ(GDT_end_flag)をクリア |
| 引　数 | src_img_addr 入力画像アドレス<br>dest_img_addr 出力画像アドレス<br>blk_conv_info ブロック情報<br>rotate_mode GDTCR.RTTFC[1:0](回転機能選択ビット)の設定<br>0：90° 右回転<br>1：90° 左回転<br>2：上下反転<br>3：左右反転 |
| リターン値 | なし |

gdt_shrink_func

| | |
|---|---|
| 概　要 | 縮小機能処理の API 関数(R_GDT_Shrink)の引数を設定し、R_GDT_Shrink 関数を呼び出します |
| ヘッダ | なし |
| 宣　言 | Gdt_shrink_func(uint32_t src_img_addr,uint32_t dest_img_addr,uint32_t dest_x,uint32_t dest_y,uint32_t blk_y,uint8_t shrink_size) |
| 説　明 | 1. 縮小入力画像のサイズ設定<br>2. 縮小出力画像のサイズ設定<br>3. 入力画像の開始座標および縮小する部分のサイズを設定、縮小出力画像の開始座標を設定<br>4. 縮小処理の設定<br>1) 反転機能 OFF<br>2) 縮小サイズの設定（本関数の引数として入力される）<br>5. R_GDT_Shrink 関数で縮小実施<br>6. 縮小処理の終了フラグ(GDT_end_flag)を待つ<br>7. 縮小処理の終了フラグ(GDT_end_flag)をクリア |
| 引　数 | src_img_addr 入力画像アドレス<br>dest_img_addr 出力画像アドレス<br>dest_x 出力画像横座標<br>dest_y 出力画像縦座標<br>blk_y 縮小画像ブロック縦サイズ<br>shrink_size 縮小サイズ |
| リターン値 | なし |

## gdt_monochrome_uniflp

| | | |
|---|---|---|
| 概　要 | 全画面モノクロ合成機能関数(デモ用)<br>エンディアン変換処理の API 関数(R_GDT_Monochrome)の引数を設定し、R_GDT_Monochrome 関数を呼び出します | |
| ヘッダ | なし | |
| 宣　言 | void gdt_monochrome_uniflp(uint32_t img_foreg_addr,uint32_t img_back_addr,uint32_t img_edge_addr,uint32_t img_dest_addr) | |
| 説　明 | 1. 縁取りがある場合、入力画像 3 枚(前景、背景、縁取り)に設定<br>2. モノクロ合成機能出力画像のサイズ設定<br>3. 縁取りがある場合(入力画像 3 枚)の入力画像前景､縁取り開始座標(0,0)固定, 背景開始座標(40,44)固定<br>4. および合成部分のサイズ(88,88)固定合成出力画像開始座標(40,44)固定<br>5. モノクロ合成処理の設定<br>6. モノクロ合成機能 ON<br>7. 反転機能 OFF<br>8. R_GDT_Monochrome 関数でエンディアン変換実施<br>9. モノクロ合成処理の終了フラグ(GDT_end_flag)を待つ<br>モノクロ合成処理の終了フラグ(GDT_end_flag)をクリア | |
| 引　数 | img_foreg_addr<br>img_back_addr<br>img_edge_addr<br>img_dest_addr | 前景アドレス、<br>背景アドレス<br>縁取りアドレス<br>出力画像アドレス |
| リターン値 | なし | |

## gdt_scroll_uniflp

| | | |
|---|---|---|
| 概　要 | 全画面スクロール機能関数(デモ用)<br>スクロール機能の API 関数(R_GDT_Scroll)の引数を設定し、R_GDT_Scroll 関数を呼び出します | |
| ヘッダ | なし | |
| 宣　言 | void gdt_scroll_uniflp(scroll_info_t *scroll_info_in) | |
| 説　明 | 1. スクロール入力画像のサイズ設定<br>2. スクロール出力画像のサイズ設定<br>3. スクロール入力画像の開始座標およびスクロールする部分のサイズを設定、スクロール出力画像の開始座標を設定<br>4. スクロール処理の設定<br>画像スクロール機能有効化ビットの設定<br>5. R_GDT_Scroll 関数でスクロール処理を実施<br>6. スクロール処理の終了フラグ(GDT_end_flag)を待つ<br>スクロール処理の終了フラグ(GDT_end_flag)をクリア | |
| 引　数 | scroll_info_in | 入力画像アドレス、出力画像アドレス、エンディアン変換する部分のサイズを設定 |
| リターン値 | なし | |

## gdt_colorsync

| | |
|---|---|
| 概　要 | カラー合成機能関数(デモ用) |

| | | |
|---|---|---|
| | カラー合成機能の API 関数(R_GDT_Colorsyn)の引数を設定し、R_GDT_Colorsyn 関数を呼び出します | |
| ヘッダ | なし | |
| 宣　言 | void gdt_colorsync(st_img_in_info_t *colorsyn_img_src_info_in,st_img_out_info_t *colorsyn_img_dest_info_in,st_blk_conv_info_t *colorsyn_blk_conv_info_in,uint32_t cpts, uint32_t cdcs) | |
| 説　明 | 1. カラー合成入力画像(前景、背景の 2 枚)のアドレスの設定。<br>2. カラー合成入力画像(前景、背景の 2 枚)のサイズの設定。<br>3. カラー合成入力画像目標アドレスの設定<br>4. カラー合成入力画像、出力画像の座標の設定<br>5. カラー合成パラメータ設定彩色合成参数设定<br>① カラー画像優先・透過設定ビット<br>② 指定色設定ビット<br>6. R_GDT_Colorsyn 関数でスクロール処理を実施<br>7. カラー合成の終了フラグ(GDT_end_flag)を待つ<br>カラー合成の終了フラグ(GDT_end_flag)をクリア | |
| 引　数 | colorsyn_img_src_info_in | 入力画像アドレス |
| | colorsyn_img_dest_info_in | 出力画像アドレス |
| | colorsyn_blk_conv_info_in | ブロック情報 |
| | cpts | カラー画像優先・透過設定 |
| | cdcs | 指定色設定 |
| リターン値 | なし | |

gdt_color

| | | |
|---|---|---|
| 概　要 | カラー化機能の API 関数(gdt_color)の引数を設定し、gdt_color 関数を呼び出します | |
| ヘッダ | なし | |
| 宣　言 | void gdt_color(uint32_t src_img_addr,uint32_t dest_img_addr,st_blk_conv_info_t *color_blk_conv_info,uint32_t rgb_0,uint32_t rgb_1 ) | |
| 説　明 | 1. カラー化入力画像のサイズ設定<br>2. カラー化出力画像のサイズ設定<br>3. カラー化入力画像の開始座標およびスクロールする部分のサイズを設定、スクロール出力画像の開始座標を設定<br>4. カラー化処理の設定<br>① 0 データの色指定<br>② 1 データの色指定<br>5. R_GDT_Scroll 関数でスクロール処理を実施<br>6. スクロール処理の終了フラグ(GDT_end_flag)を待つ<br>スクロール処理の終了フラグ(GDT_end_flag)をクリア | |
| 引　数 | src_img_addr | 入力画像アドレス |
| | dest_img_addr | 出力画像アドレス |
| | color_blk_conv_info | ブロック情報 |
| | rgb_0 | 0 データの色指定 |
| | rgb_1 | 1 データの色指定 |

リターン値　なし

## Img_Out_Coloralign

| | |
|---|---|
| 概　要 | カラーデータ整列機能関数(デモ用)<br>カラーデータ整列機能の API 関数(R_GDT_Coloralign)の引数を設定し、<br>R_GDT_Coloralign 関数を呼び出します |
| ヘッダ | なし |
| 宣　言 | void　Img_Out_Coloralign(uint32_t buf_addr,uint32_t buf_out) |
| 説　明 | 1.　カラーデータ整列入力画像のサイズ設定<br>2.　カラーデータ整列出力画像のサイズ設定<br>3.　カラーデータ整列画像および出力画像の開始座標を(0,0)に固定、サイズを 176 ｘ176ｘ3 に固定スクロール処理の設定<br>4.　R_GDT_Coloralign 関数でスクロール処理を実施<br>5.　スクロール処理の終了フラグ(GDT_end_flag)を待つ<br>スクロール処理の終了フラグ(GDT_end_flag)をクリア |
| 引　数 | buf_addr　入力画像アドレス<br>buf_out　出力画像アドレス |
| リターン値 | なし |

## LCD_Output_3bit

| | |
|---|---|
| 概　要 | GDT 処理した画面を SPI よりカラーLCD へ出力（カラー3bit 画面用） |
| ヘッダ | なし |
| 宣　言 | void LCD_Output_3bit(void) |
| 説　明 | 1.　SPI より画像データを出力<br>2.　全画像データ出力完了待ち<br>3.　1 秒 Delay |
| 引　数 | なし |
| リターン値 | なし |

## LCD_Output_1bit

| | |
|---|---|
| 概　要 | GDT 処理した画面を SPI よりカラーLCD へ出力（モノクロ画面用） |
| ヘッダ | なし |
| 宣　言 | void LCD_Output_1bit(unsigned char *ram_addr) |
| 説　明 | 1.　SPI より画像データを出力<br>2.　全画像データ出力完了待ち<br>3.　1 秒 Delay |
| 引　数 | ram_addr　出力画面 Buffer の開始アドレス |
| リターン値 | なし |

## 3.3 イベントリンク設定

表 3-2 に ICU イベントリンク設定の内容を示します。

**表 3-2 ICU イベントリンク設定内容**

| NVICにリンクする割り込み信号 | 変更箇所 | 変更内容 |
|---|---|---|
| AGT_AGTI | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTI | □設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER11) |
| DMAC0_INT | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT | □ 設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER0) |
| DMAC1_INT | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_DMAC1_INT | □ 設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER5) |
| DMAC2_INT | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_DMAC2_INT | □ 設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER14) |
| GDT_DATOI | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_GDT_DATOI | □ 設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER4) |
| GDT_DATII | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_GDT_DATII | □ 設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER3) |
| SPI0_SPII | [r_system_cfg.h]<br>SYSTEM_CFG_EVENT_NUMBER_SPI0_SPII | □ 設定値変更<br>(SYSTEM_IRQ_EVENT_NUMBER10) |

## 3.4 フローチャート

本章では、メイン処理フローのみを紹介します。他の関数の処理については、各関数の説明欄を参照してください。

図 3.2 にメイン処理のフローチャートを示します。

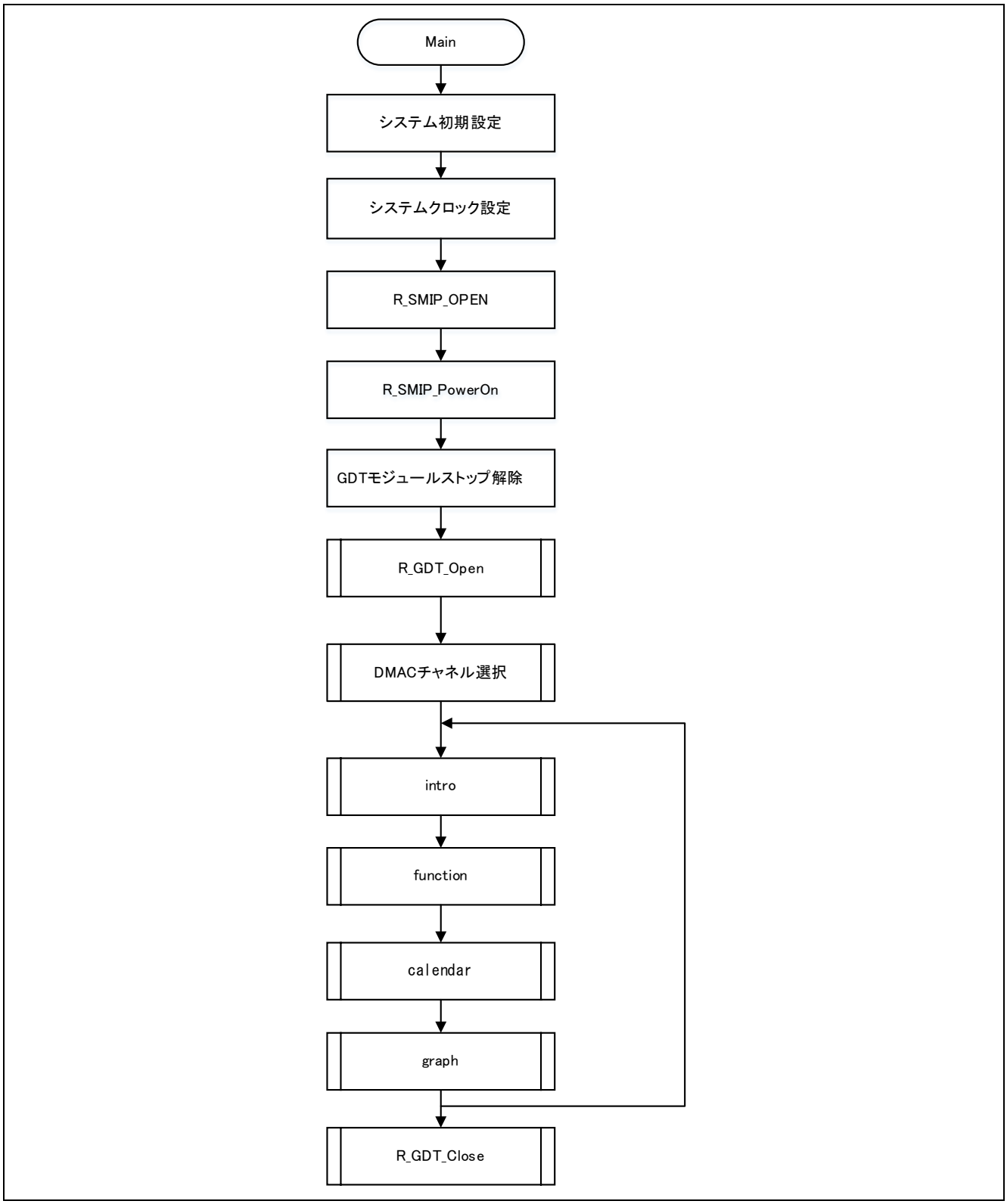



図 3.3 メイン処理

## 3.5 カラー画像データ作成方法

デモで使用される画像データ作成の手順を以下に示します。

### 3.5.1 準備

対象の BMP 形式の画像を LCD 出力形式に変換します。

1. BMP 形式画像
2. 画像形式変換ツール：bmp2cs12（BMP 形式→CSV 形式）
3. LCD 出力形式変換：r01an4810xx0070/可変式 LCD 出力用シート.xlsx
   （CSV 形式→LCD 出力形式）

### 3.5.2 使用上の制限事項

本アプリケーションノートで使用した画像形式変換ツールはフリーソフトを使用しています。任意のフリーソフトで画像形式変換をしてください。

### 3.5.3 手順

図 3.3 を例として、BMP 形式の画像を LCD 出力形式に変換する手順を紹介します。


図 3.4 カラーBMP 画像

1. 図 3.3 のようなカラーBMP 画像を用意してください。全画面に画像を表示する場合のサイズは 176x176(bit)です。
2. bmp2cs12.lzh の中の bmp2csv.exe を実行(ダブルクリック)すると、図 3.4 の画面が出力されます。「実行」ボタンをダブルクリックしてください。

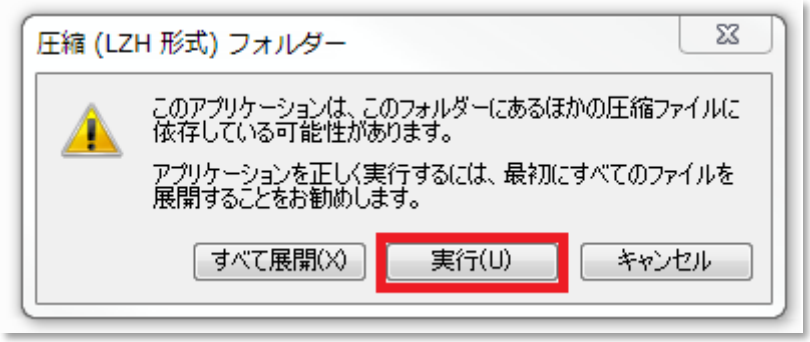



図 3.5 bmp2csv 実行確認画面

3. 図 3.5 の画面で BMP ファイルを選択すると、入力画像のイメージが左下に表示されます。

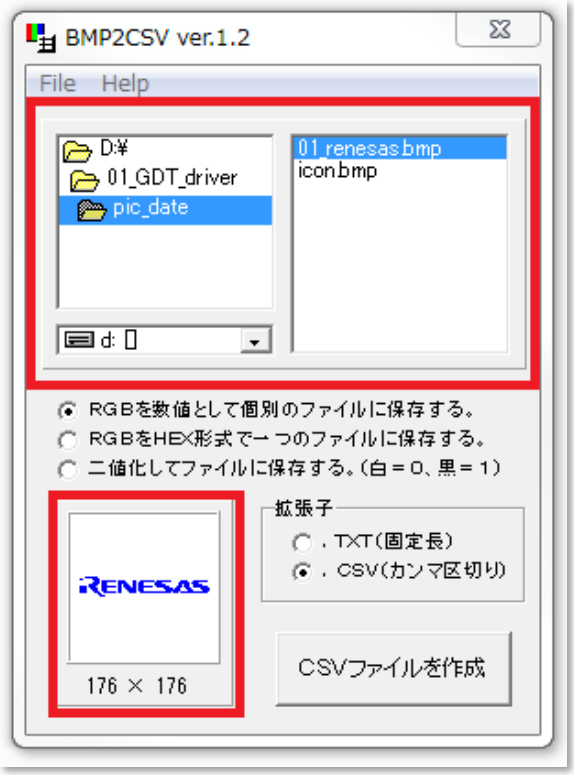



図 3.6 bmp2csv の UI 画面

4. 図 3.6 に示すように、変換形式は「RGB を数値として個別のファイルに保存する。」を選択し、拡張子は「.CSV(カンマ区切り)」を選択してください。

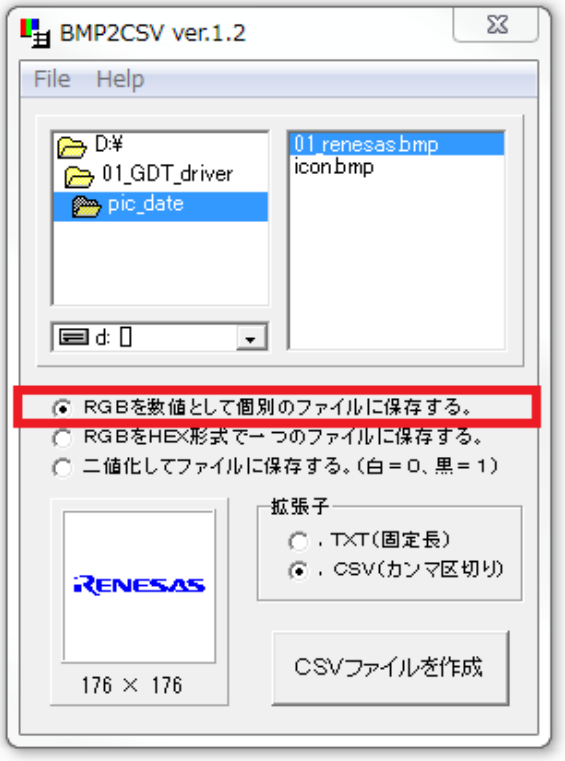



図 3.7 bmp2csv の出力形式選択

5. 図 3.7 に示すように、「CSV ファイル作成」をクリックして、CSV ファイルを作成します。
   入力の BMP 画像のフォルダの下に、RGB それぞれの CSV ファイル「iconW_R.CSV」、「iconW_G.CSV」、「iconW_B.CSV」が生成されます。CSV の中のすべての“255”を “1”に置き換える必要があります。

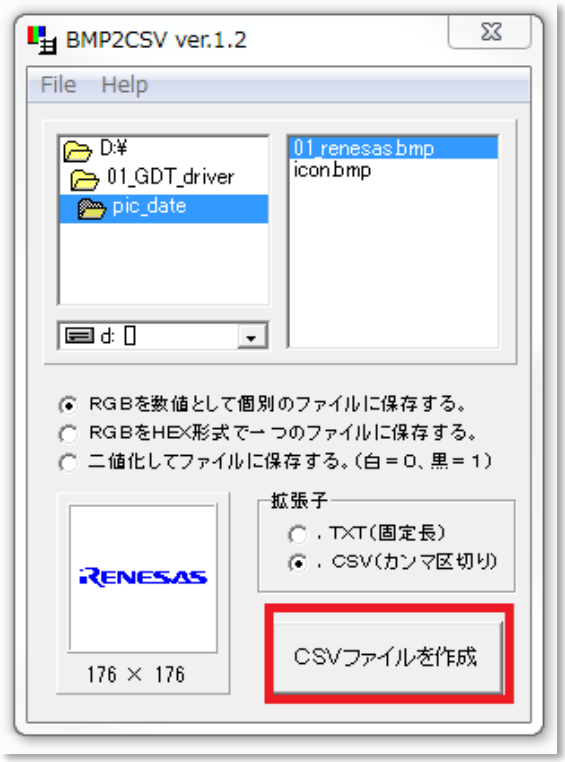



図 3.8 bmp2csv の CVS ファイル作成の実行

6. ツール「可変式 LCD 出力用シート.xlsx」内の、sheet「表紙_説明書兼一部設定」にツールの手順説明を記載していますので、よく読んでからご利用ください。

7. sheet「表紙_説明書兼一部設定」の下にある「●以下設定」を図 3.8 のように設定してください。

| | |
|---|---|
| ●以下設定 | |
| 画像X幅(単位：ピクセル、8～512) | 176 |
| 画像Y幅(単位：ピクセル,1～256) | 176 |
| アドレス部サイズ(バイト単位) | 1 |
| ダミーサイクル部サイズ(バイト単位 | 4 |
| ダミーサイクル部データ(FFまでの16 | 00 |

図 3.9 表紙_説明書兼一部設定の抜粋

8. 図 3.9 に示すように、sheet「データ打ち込み」を表示し、step「5」で生成したデータを「データ打ち込み」のセル B3 から貼り付けます。

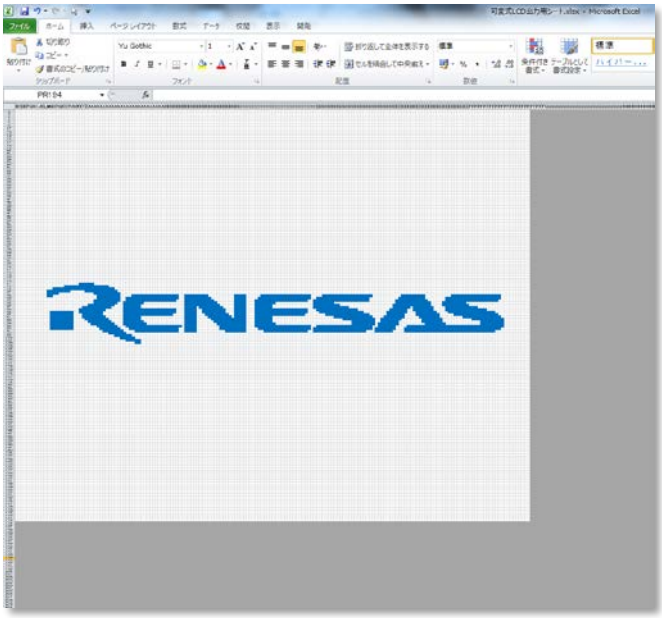


図 3.10　データ打ち込み

9. 図 3.10 に示すように、ツール「可変式 LCD 出力用シート.xlsx」の sheet「output」で LCD 出力形式データが自動的に表示されます。上 6 行の設定（黄色塗りつぶし）は状況によって変更してください。

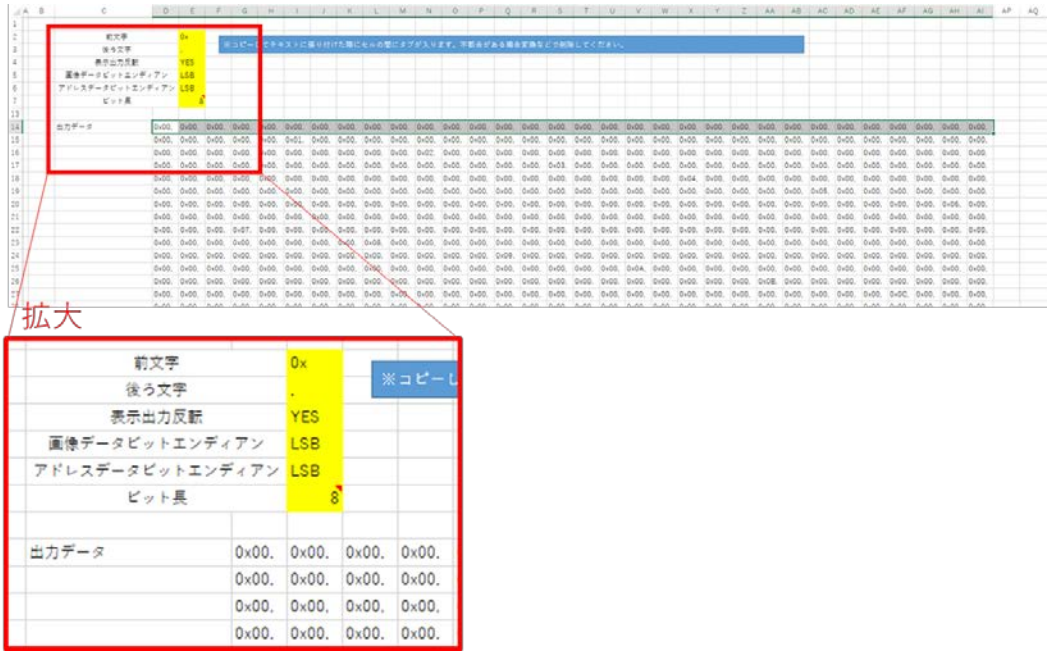



図 3.11　出力データの設定

10. Sheet「output」の中のデータをセル D14 から任意の*.h にコピーします。本例では、an4810_cmsis_gdt_color_re/Renesas_RE_DFP/gdt_demo/pic.h の/* R-data */、/* G-data */、/* B-data */の箇所に、それぞれ R 画像、G 画像、B 画像をコピーします。

11. Sheet「output」の中のデータについて、用途により、sheet「表紙_説明書兼一部設定」の下にある「●以下設定」の設定が異なるため、詳細を以下に示します。

1) LCD に出力する際は、図 3.11 のように“アドレス部サイズ(バイト単位)”と“ダミーサイクル部サイズ(バイト単位)”を設定してください。Sheet「output」に出力されるイメージを図 3.12 に示します。

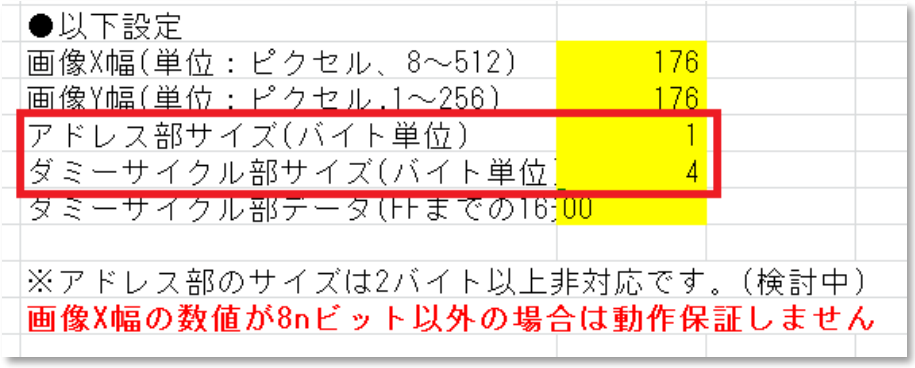



図 3.12 LCD に出力する際の設定


図 3.13 LCD に出力する際のデータ出力イメージ

2) 画像処理などに使用する際は、図 3.13 のように“アドレス部サイズ(バイト単位)”と“ダミーサイクル部サイズ(バイト単位)”を“0”に設定してください。Sheet「output」に出力されるイメージを図 3.14 に示します。

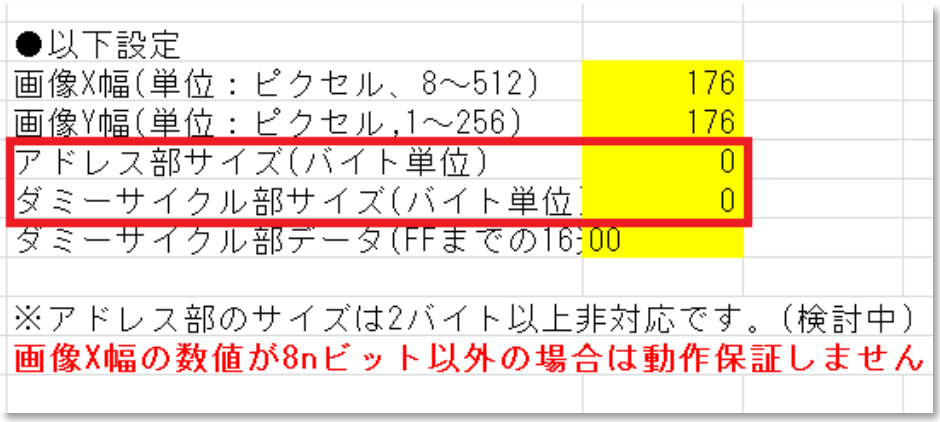



図 3.14 画像処理などに使用する際の設定


図 3.15 画像処理などに使用する際のデータ出力イメージ

# 4. ドライバの API 仕様

## 4.1 外部仕様書

本ドライバには API の外部仕様を記したドキュメントを同包しています。

Documents フォルダの直下にある Driver Specification フォルダに格納されています。

# 5. GDT ドライバを使用する上での注意事項

本章では GDT ドライバに関する主だった注意点を紹介します。すべての注意点を紹介しきれていません。注意点について『4.1 外部仕様書』をご参照ください。

## 5.1 画像データサイズ

画像データの水平サイズおよび垂直サイズは、8 の倍数(ビット)としてください。また、画像の水平サイズは、水平メモリサイズ以下に設定してください。

## 5.2 水平メモリサイズ

本ドライバはワーク領域のサイズを引数で指定する必要があります。加工する画像サイズ同等もしくは以上のサイズを次の中から選択してください。

水平メモリサイズは、32、64、128、または 256 を設定してください。ただし、カラーデータ整列機能の 3 ビットモード使用時は、96、192、384、または 768、4 ビットモード使用時は 128、256、512、または 1024 を設定してください。

## 5.3 画像処理単位

回転機能(R_GDT_Rotate)、モノクロ合成機能(R_GDT_Monochrome)、およびカラー合成機能(R_GDT_Colorsyn)は、画像処理単位を 2 種類(8×8 ビットまたは 16×16 ビット)から選択できます。各関数の画像処理データサイズ選択変数(gdtdsz)で設定してください。

回転機能を画像処理単位 16×16 ビットで使用する際、処理前および処理後ブロックの開始アドレスは偶数番地に設定してください。

モノクロ合成機能およびカラー合成機能では、画像処理単位 8×8 ビットのみ対応しています。これらの関数使用時は gdtdsz=1 に設定してください。

## 5.4 DMAC の割り込み設定および優先順位の制限

本ドライバは、必ず計 2 チャネルの DMAC を使用します。他の機能で使用する DMAC のチャネルと重複しないように、使用チャネルを設定してください。

また、本ドライバで使用する DMAC の割り込み優先順位に制限があります。使用する DMAC のチャネルは、R_GDT_DmacChSel 関数で設定します。この関数では、”GDT への入力データ転送用 DMAC”と、”出力データ転送用 DMAC”を設定します。上記で選択した DMAC のチャネルは、以下制限を満たすよう DMAC の割り込み優先順位を設定してください。

＊制限：使用する DMAC 割り込みの優先順位
入力データ転送に使用するチャネル ＞ 出力データ転送に使用するチャネル

DMAC の各チャネルの割り込み優先順位は、
/Device/Config/r_dmac_cfg.h のマクロ定義”DMACn_INT_PRIORITY(n=0～3)”で設定できます。

例：入力にチャネル 1、出力にチャネル 2 を使う場合

```
#define DMAC1_INT_PRIORITY              (0)     ///  (set to 0 to 3, 0 is highest priority.)
#define DMAC2_INT_PRIORITY              (3)     ///  (set to 0 to 3, 0 is highest priority.)
```

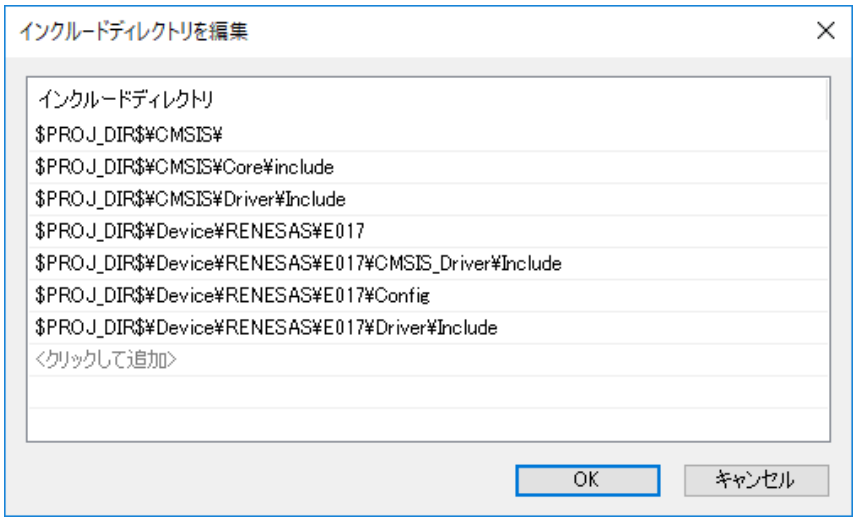
# 6. トラブルシューティング

## 6.1 ビルドエラーが発生する

A-1）インクルードディレクトリが設定されていますか？

EWARM をご使用の場合、下記の例の様にインクルードディレクトリを設定することを推奨します。

IDE のオプション［C/C++コンパイラ］->［プリプロセッサ]から設定ができます。




## 6.2 CMSIS ドライバの API をコールすると HardFault Error が発生する

A） API の RAM 展開ができていない可能性があります。

RAM 上に配置した API をコールする前に R_SYS_CodeCopy 関数にて API を RAM 展開しているか確認してください。詳細は関連ドキュメント No. R01AN4660 をご参照ください。

## 6.3 API を呼び出しているが周辺機能が動作しない

A）API の設定が問題無くできていますか？

API の戻り値を確認し、エラー値が返っていないかをご確認ください。

特に r_system_cfg.h の割り込み設定がされていないことでエラー値が返っている事例が多く発生しています。詳細は関連ドキュメント No. R01AN4660 をご参照ください。

## 6.4 API の戻り値は正常であるが、周辺機能から端子出力が行われない

A） 端子設定は正しいでしょうか？

Pin.c の中にある関数にて端子設定が正しく行えているか確認してください。

詳細は関連ドキュメント No. R01AN4660 をご参照ください。

## 6.5 周辺機能の入力または出力が期待通り動作しない

A)周辺機能を初期設定する前に VOCR レジスタの設定が行えているか確認してください。

詳細は関連ドキュメント No. R01AN4660 をご参照ください。

## 7. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

## 8. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RE01 1500KB グループユーザーズマニュアル ハードウェア編 R01UH0796

RE01 256KB グループユーザーズマニュアル ハードウェア編 R01UH0894

（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。）

RE01 1500KB, 256KB CMSIS Package スタートアップガイド

RE01 1500KB、256KB グループ CMSIS パッケージを用いた開発スタートアップガイド R01AN4660

（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

（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。）

## 改訂記録

| Rev. | 発行日 | 改訂内容 | |
|---|---|---|---|
| | | ページ | ポイント |
| 1.00 | Aug.19.2019 | 全ページ | 初版発行 |
| 1.01 | Jan.15.2020 | プログラム | GDT ドライバを Ver.1.00b に更新<br>内部関数 v_gdt_cpuline_byte_rd_gdt_limit の不備を暫定的に修正 |
| 1.02 | Mar.19.2020 | -<br>プログラム | RE01 256KB グループを追加<br>CMSIS Driver Package を差し替え<br>- RE01 1500KB: CMSIS Driver Package Rev.1.10<br>- RE01 256KB: CMSIS Driver Package Rev.0.80 |
| 1.03 | Jun.01.2020 | 3<br>6<br>プログラム<br>(256KB) | サンプルコードプロジェクト名を変更<br>動作確認条件を更新<br>CMSIS Driver Package を差し替え<br>- RE01 256KB: CMSIS Driver Package Rev.1.00 |

# 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。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

## 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。