R_GDT Driver Sample Code (Using CMSIS Driver Package) for RE01 1500KB Group, 256KB Group

R_GDT Sample Code Using CMSIS Driver Package

Summary
This application note describes the R_GDT driver sample code using the CMSIS driver package for the RE01 1500KB Group and RE01 256KB Group. The sample code can be found in the projects delivered with this application note.

The overview of this sample code is shown in the table below.

Table  Overview of Sample Code

<table>
<thead>
<tr>
<th>Overview of Sample Code Operation</th>
<th>Peripheral Modules Mainly Used</th>
<th>Driver Modules Mainly Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image processing:</td>
<td>GDT and DMAC</td>
<td>R_GDT, R_DMAC</td>
</tr>
<tr>
<td>Performs image processing, such</td>
<td></td>
<td></td>
</tr>
<tr>
<td>as rotation, using GDT functions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The DMAC is used for data input to the GDT or data output from the GDT.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image output:</td>
<td>SPI and DMAC</td>
<td>R_SMIP (R_SPI, R_DMAC)</td>
</tr>
<tr>
<td>Outputs image data to an LCD using the R_SMIP driver.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Target Device
RE01 1500KB Group
RE01 256KB Group

Note
When applying the sample code covered in this application note to another microcontroller, please modify the code according to the specifications for the target microcontroller and conduct an extensive evaluation of the modified program.

Related Document
RE01 1500KB, 256KB Group Startup Guide to Development Using CMSIS Package (R01AN4660)
Contents

1. Specifications ................................................................................................................................. 3
1.1 Description of Projects .................................................................................................................. 3
1.2 Pins Used ...................................................................................................................................... 4
1.3 Folder Structure ............................................................................................................................ 5
1.4 File Configuration ......................................................................................................................... 6
1.5 Option-Setting Memory ................................................................................................................ 6

2. Operating Conditions .................................................................................................................. 7

3. Description of Software ................................................................................................................ 9
3.1 Sample Code of Multiple Functions for Monochrome LCDs ........................................................ 10
3.1.1 System Configuration ............................................................................................................. 11
3.1.2 List of Functions ................................................................................................................... 12
3.1.3 Event Link Settings ................................................................................................................ 26
3.1.4 Flowcharts ............................................................................................................................ 27
3.2 Sample Code of Font Unfold Function for Monochrome LCDs ..................................................... 33
3.2.1 System Configuration ............................................................................................................. 34
3.2.2 List of Functions ................................................................................................................... 35
3.2.3 Event Link Settings ................................................................................................................ 38
3.2.4 Flowcharts ............................................................................................................................ 39
3.3 Method for Generating Monochrome Image Data ........................................................................ 41
3.3.1 Preparation ........................................................................................................................ 41
3.3.2 Restriction on Use .................................................................................................................. 41
3.3.3 Procedure ............................................................................................................................ 41

4. Specifications of Driver APIs ....................................................................................................... 47
4.1 External Specification Document ............................................................................................... 47

5. Usage Notes of R_GDT Driver ..................................................................................................... 48
5.1 Sizes of Image Data ..................................................................................................................... 48
5.2 Horizontal Size of Memory ........................................................................................................ 48
5.3 Image Processing Unit ................................................................................................................ 48
5.4 DMAC Interrupt Settings and Restriction on Priority ................................................................... 48

6. Troubleshooting ............................................................................................................................. 49
6.1 Occurrence of a Build Error with the IAR Compiler .................................................................... 49
6.2 Occurrence of HardFault Error when API of CMSIS Driver Is Called ........................................ 49
6.3 Peripheral Function Fails to Operate when API Is Called .......................................................... 49
6.4 Normal API Return Value But No Pin Output from Peripheral Function ...................................... 49
6.5 Peripheral Function’s Input or Output Does Not Operate as Expected ........................................ 49

7. Sample Code .................................................................................................................................. 50

8. Reference Documents .................................................................................................................. 50

Revision History .............................................................................................................................. 51
1. Specifications

1.1 Description of Projects

The following two sample code projects are provided with this application note.

[Sample code project for RE01 1500KB group]

r01an4755_hal_gdt_mono_re_1500kb.zip: Sample code of the reversal, shrink (this operation is called “scaling down” in User’s Manual: Hardware), monochrome synthesis, rotation, and scroll functions for LCDs capable of displaying monochrome images

r01an4755_hal_gdt_font_re_1500kb.zip: Sample code of the font unfold (this operation is called “conversion of glyph data into image data” in User’s Manual: Hardware) function

The sample code project for RE01 1500KB group have been tested using the Evaluation Kit RE01 1500KB. These projects are configured to match the settings of R7F0E015D2CFB mounted on the Evaluation Kit RE01 1500KB. When using another device, change the device settings in the projects to those of the target device.

[Sample code project for RE01 256KB group]

r01an4755_hal_gdt_mono_re_256kb.zip: Sample code of the reversal, shrink (this operation is called “scaling down” in User’s Manual: Hardware), monochrome synthesis, rotation, and scroll functions for LCDs capable of displaying monochrome images

r01an4755_hal_gdt_font_re_256kb.zip: Sample code of the font unfold (this operation is called “conversion of glyph data into image data” in User’s Manual: Hardware) function

The sample code project for RE01 256KB group have been tested using the Evaluation Kit RE01 256KB. These projects are configured to match the settings of R7F0E01182CFP mounted on the Evaluation Kit RE01 256KB. When using another device, change the device settings in the projects to those of the target device.
1.2 Pins Used

The pins used by the sample code are shown below.

Table 1-1 The pins used by the sample code for RE01 1500KB group

<table>
<thead>
<tr>
<th>Pin Used</th>
<th>Purpose of Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>P605</td>
<td>RST</td>
</tr>
<tr>
<td>P606</td>
<td>VCOM</td>
</tr>
<tr>
<td>P607</td>
<td>RSPCKB_B(SPI-SCLK)</td>
</tr>
<tr>
<td>P608</td>
<td>MISOB_B(DNF)</td>
</tr>
<tr>
<td>P609</td>
<td>MOSIB_B(SPI-SI)</td>
</tr>
<tr>
<td>P610</td>
<td>SCS</td>
</tr>
<tr>
<td>P302</td>
<td>GPIO</td>
</tr>
<tr>
<td>P303</td>
<td>GPIO</td>
</tr>
<tr>
<td>P100</td>
<td>SW1 (Not used in font unfolding)</td>
</tr>
<tr>
<td>P508</td>
<td>SW2 (Not used in font unfolding)</td>
</tr>
<tr>
<td>P410</td>
<td>SW3 (Not used in font unfolding)</td>
</tr>
</tbody>
</table>

Table 1-2 The pins used by the sample code for RE01 256KB group

<table>
<thead>
<tr>
<th>Pin Used</th>
<th>Purpose of Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>P813</td>
<td>RST</td>
</tr>
<tr>
<td>P814</td>
<td>VCOM</td>
</tr>
<tr>
<td>P011</td>
<td>RSPCKA_B(SPI-SCLK)</td>
</tr>
<tr>
<td>P815</td>
<td>MISOA_B(DNF)</td>
</tr>
<tr>
<td>P010</td>
<td>MOSIA_B(SPI-SI)</td>
</tr>
<tr>
<td>P015</td>
<td>SCS</td>
</tr>
<tr>
<td>P014</td>
<td>GPIO</td>
</tr>
<tr>
<td>P806</td>
<td>GPIO</td>
</tr>
<tr>
<td>P509</td>
<td>SW1 (Not used in font unfolding)</td>
</tr>
<tr>
<td>P508</td>
<td>SW2 (Not used in font unfolding)</td>
</tr>
<tr>
<td>P501</td>
<td>SW3 (Not used in font unfolding) (*)</td>
</tr>
</tbody>
</table>

Note. SW3 is not mounted on Evaluation Kit RE01 256KB. Mount SW with external circuit.
1.3 Folder Structure

The folder structure of the sample code and drivers used by the sample code are shown below.

![Folder Structure Diagram]

Figure 1.1   Folder Structure
1.4 File Configuration
Table 1-3 shows the files that are added or modified for this sample code.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Overview of Processing or Configuration</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>Main processing</td>
<td></td>
</tr>
<tr>
<td>r_system_cfg.h</td>
<td>System configuration</td>
<td></td>
</tr>
<tr>
<td>r_spi_cfg.h</td>
<td>SPI configuration setting</td>
<td></td>
</tr>
<tr>
<td>r_smip_cfg.h</td>
<td>SMIP configuration setting</td>
<td></td>
</tr>
<tr>
<td>r_dmac_cfg.h</td>
<td>DMAC configuration setting</td>
<td></td>
</tr>
<tr>
<td>pin.c</td>
<td>Pin setting</td>
<td></td>
</tr>
<tr>
<td>demo_view.c</td>
<td>Actual operation part of demonstration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(r_GDT API calling part)</td>
<td></td>
</tr>
<tr>
<td>demo_view.h</td>
<td>Definitions of functions and variables</td>
<td></td>
</tr>
<tr>
<td>pic.h</td>
<td>Storage of image data to be used in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>demo_view</td>
<td></td>
</tr>
<tr>
<td>pic_icon.h</td>
<td>Storage of image data to be used in the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>main function</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not used in the project of the font unfold function.</td>
</tr>
</tbody>
</table>

1.5 Option-Setting Memory
Table 1-4 shows the option-setting memory setting for sample code. Set suitable values for a user system if required.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>0100A164h to 0100A167h</td>
<td>FFFF FFFFh</td>
<td>No access window settings</td>
</tr>
<tr>
<td>OSIS</td>
<td>0100A150h to 0100A15Fh</td>
<td>FFFF FFFFh</td>
<td>No ID code protection (All FFh)</td>
</tr>
<tr>
<td>SECMPUxxx</td>
<td>000000408h to 0000043Bh</td>
<td>FFFF FFFFh</td>
<td>MPU is disabled.</td>
</tr>
<tr>
<td>OFS1</td>
<td>00000404h to 00000407h</td>
<td>FFFF FFFFh</td>
<td>After a reset, the voltage monitor 0 reset is disabled. After a reset, HOCO oscillation is disabled.</td>
</tr>
<tr>
<td>OFS0</td>
<td>00000400h to 00000403h</td>
<td>FFFF FFFFh</td>
<td>Automatic activation of IWDT is disabled. Automatic activation of WDT is disabled.</td>
</tr>
</tbody>
</table>
2. Operating Conditions

The operation of the sample code provided with this application note has been tested under the following conditions (Table 2-1, Table 2-2).

Table 2-1 Operating Conditions for RE01 1500KB group

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>R7F0E015D2CFB (144-pin)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>System clock: HOCO is selected</td>
</tr>
<tr>
<td></td>
<td>• System clock (ICLK): 32 MHz (HOCO frequency is divided by 1)</td>
</tr>
<tr>
<td></td>
<td>• Peripheral module clock A (PCLKA): 32 MHz (HOCO frequency is divided by 1)</td>
</tr>
<tr>
<td></td>
<td>• Peripheral module clock B (PCLKB): 32 MHz (HOCO frequency is divided by 1)</td>
</tr>
<tr>
<td>AGT timer operation clock</td>
<td>Sub-clock oscillator (SOSC): 32.768 kHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>IAR Embedded Workbench for ARM Version 8.40.2</td>
</tr>
<tr>
<td></td>
<td>C compiler: IAR C/C++ Compiler for ARM Version 8.40.2</td>
</tr>
<tr>
<td></td>
<td>e² studio Renesas e² studio Version 7</td>
</tr>
<tr>
<td></td>
<td>• C compiler: GCC ARM Embedded Version 6.3.1.20170620 GNU 6-2017-q2-update</td>
</tr>
<tr>
<td>Debugger</td>
<td>SEGGER J-Link OB</td>
</tr>
<tr>
<td>IAR optimization setting</td>
<td>Optimization level &quot;high (size)&quot;</td>
</tr>
<tr>
<td>(sample code of multiple functions for monochrome LCDs)</td>
<td></td>
</tr>
<tr>
<td>IAR optimization setting</td>
<td>Optimization level &quot;low&quot;</td>
</tr>
<tr>
<td>(sample code of the font unfold function for monochrome LCDs)</td>
<td></td>
</tr>
<tr>
<td>Target board</td>
<td>Evaluation Kit RE01 1500KB</td>
</tr>
<tr>
<td></td>
<td>(Product type number: RTK70E015DSXXXXXBE)</td>
</tr>
<tr>
<td>Version of CMSIS driver package</td>
<td>Rev. 1.10A</td>
</tr>
<tr>
<td>Version of sample code</td>
<td>Rev. 1.05</td>
</tr>
</tbody>
</table>
## Table 2-2  Operating Conditions for RE01 256KB group

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microcontroller used</strong></td>
<td>R7F0E01182CFP (100pin)</td>
</tr>
<tr>
<td><strong>Operating frequency</strong></td>
<td><strong>System clock: HOCO is selected</strong></td>
</tr>
<tr>
<td></td>
<td>• System clock (ICLK): 32 MHz (HOCO frequency is divided by 1)</td>
</tr>
<tr>
<td></td>
<td>• Peripheral module clock A (PCLKA): 32 MHz (HOCO frequency is divided by 1)</td>
</tr>
<tr>
<td></td>
<td>• Peripheral module clock B (PCLKB): 32 MHz (HOCO frequency is divided by 1)</td>
</tr>
<tr>
<td><strong>AGT timer operation clock</strong></td>
<td>Sub-clock oscillator (SOSC): 32.768 kHz</td>
</tr>
<tr>
<td><strong>Operating voltage</strong></td>
<td>3.3 V</td>
</tr>
<tr>
<td><strong>Integrated development environment</strong></td>
<td><strong>IAR</strong></td>
</tr>
<tr>
<td></td>
<td>IAR Embedded Workbench for ARM  Version 8.40.2</td>
</tr>
<tr>
<td></td>
<td>C compiler: IAR C/C++ Compiler for ARM Version 8.40.2</td>
</tr>
<tr>
<td></td>
<td><strong>e² studio</strong></td>
</tr>
<tr>
<td></td>
<td>Renesas e² studio 2020-07</td>
</tr>
<tr>
<td></td>
<td>C compiler: GCC ARM Embedded Version 6.3.1.20170620</td>
</tr>
<tr>
<td></td>
<td>GNU 6-2017-q2-update</td>
</tr>
<tr>
<td><strong>Debugger</strong></td>
<td>SEGGER J-Link OB</td>
</tr>
<tr>
<td><strong>IAR optimization setting (sample code of multiple functions for monochrome LCDs)</strong></td>
<td>Optimization level &quot;high (size)&quot;</td>
</tr>
<tr>
<td><strong>IAR optimization setting (sample code of the font unfold function for monochrome LCDs)</strong></td>
<td>Optimization level &quot;low&quot;</td>
</tr>
<tr>
<td><strong>Target board</strong></td>
<td>Evaluation Kit RE01 256KB</td>
</tr>
<tr>
<td></td>
<td>(Product type number: RTK70E0118CXXXXXBJ)</td>
</tr>
<tr>
<td><strong>Version of CMSIS driver package</strong></td>
<td>Rev.1.00</td>
</tr>
<tr>
<td><strong>Version of sample code</strong></td>
<td>Rev.1.05</td>
</tr>
</tbody>
</table>
3. Description of Software

This sample code uses the R_GDT driver to convert images and outputs them to an LCD. The DMAC is used to input or output images for the GDT and to input or output data for the SPI, which serves as the communication interface with the LCD.

<table>
<thead>
<tr>
<th>Item</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI channel selection</td>
<td>Sample code project for RE01 1500KB group : SPI CH1</td>
</tr>
<tr>
<td></td>
<td>Sample code project for RE01 256KB group : SPI CH0</td>
</tr>
<tr>
<td>SPI bit rate</td>
<td>500kbps</td>
</tr>
<tr>
<td>SMIP AGT channel setting</td>
<td>CH0</td>
</tr>
<tr>
<td>SMIP AGT operating clock</td>
<td>SOCO (32.768kHz)</td>
</tr>
<tr>
<td>DMAC channel settings for the GDT</td>
<td>DMAC CH0: Transfers image data from RAM to GDT.</td>
</tr>
<tr>
<td></td>
<td>DMAC CH1: Transfers converted data from GDT to RAM.</td>
</tr>
<tr>
<td></td>
<td>(Any channel from CH0 to CH3 can be selected.)</td>
</tr>
<tr>
<td>DMAC channel settings for the SPI</td>
<td>DMAC CH2</td>
</tr>
<tr>
<td></td>
<td>(Any channel from CH0 to CH3 can be selected.)</td>
</tr>
</tbody>
</table>
3.1 Sample Code of Multiple Functions for Monochrome LCDs

The sample code of multiple functions for monochrome LCDs provides the user with four different demonstrations; the user can select one of the demonstrations from the displayed menu by pressing buttons on the board.

- **Up-SW (\(^{\ast}1\))**: Selects the demonstration displayed above the current one on the menu.
- **Down-SW (SW2)**: Selects the demonstration displayed below the current one on the menu.
- **Enter-SW (\(^{\ast}2\))**: Determines the selection.

Note1. For RE01 1500KB group, use SW1. For RE01 256KB group, use SW3.

Note2. For RE01 1500KB group, use SW3. For RE01 256KB group, use SW1.

The sample code operates as follows.

- **Initialization**
  Necessary initialization processing is performed.
  1) System initialization
  2) System clock settings
  3) I/O power control (all I/O power supplies are turned on)
  4) Initial settings of MLCD display by the R_SMIP driver (API functions R_SMIP_Open() and R_SMIP_PowerOn() are executed)
  5) Initial settings of the menu to be displayed

- **Preparing a 1/2 image (icon) to be used in GDT processing**
  The demonstration of a monochrome synthesis function and rotation function uses an image having half (128 bits \( \times \) 128 bits) the size of the input image (256 bits \( \times \) 256 bits). The image is prepared before the menu is displayed.
  1) The API function R_GDT_Open() is executed to activate the GDT.
  2) The endian conversion and shrink functions are used to create an image with half the size of the input image.
  3) After creation of the 1/2 image (icon) to be used in GDT processing, the API function R_GDT_Close() is executed to disable the GDT operation. To perform demonstration (GDT processing), the API function R_GDT_Open() should be executed again.

- **Controlling SW**
  1) **Up-SW**
     Pressing Up-SW selects the demonstration displayed above the current one on the menu (executes view_up()).
  2) **Down-SW**
     Pressing Down-SW selects the demonstration displayed below the current one on the menu (executes view_down()).
  3) **Enter-SW**
     Pressing Enter-SW executes the menu selected on the menu (executes view_enter()).
3.1.1 System Configuration

Figure 3.1 System Configuration for RE01 1500KB group

RE01
1500KB group
(R7F0E015D2CFB)

PMOD1

P100(Pin99)
P508(Pin107)
P410(Pin23)

SW1
SW2
SW3

LCD screen

Note. SW3 is not mounted on Evaluation Kit RE01 256KB.
Mount SW with external circuit.

Figure 3.2 System Configuration for RE01 256KB group

RE01
256KB group
(R7F0E01182CFP)

PMOD

P509(Pin74)
P508(Pin75)
P501(Pin76)

SW1
SW2
SW3(*)

LCD screen

Note. SW3 is not mounted on Evaluation Kit RE01 256KB.
Mount SW with external circuit.
3.1.2 List of Functions
The functions added to the sample code are described here. Modify them as necessary.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| main     | Main processing<br>Header: None<br>Declaration: void main(void)<br>Description: This function sets up the system clock and SMIP (SPI and DMAC) transfer. The sample code waits for a button to be pressed within the main loop. |}
| view_int | Menu initialization function<br>Header: None<br>Declaration: void view_int(void)<br>Description: 1. This function combines the images of individual demonstration names to create the menu image in the initial state (the menu_merge_image function). The sample code displays the following four items on the menu.<br>1) Function: Displayed in white (1); selected state<br>2) Speed: Displayed in black (0); unselected state<br>3) Alarm: Displayed in black (0); unselected state<br>4) Navigation: Displayed in black (0); unselected state<br>2. The combined image is output to the LCD (the display_img function). |}
| Argument | None<br>Return Value | None

| Argument | None<br>Return Value | None |
### view_down

<table>
<thead>
<tr>
<th>Overview</th>
<th>Menu option function (move down)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void view_down (void)</td>
</tr>
<tr>
<td>Description</td>
<td>1. This function calculates the key position (Key_position) when Down-SW is pressed to select the menu item displayed below the current one. According to the Key_position value, the displayed image is changed.</td>
</tr>
<tr>
<td></td>
<td>2. According to the Key_position value, the displayed image is changed.</td>
</tr>
<tr>
<td></td>
<td>1) Key_position is 0: Changes the color of &quot;Function&quot; to white and &quot;Navigation&quot; to black. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>2) Key_position is 1: Changes the color of &quot;Speed&quot; to white and &quot;Function&quot; to black. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>3) Key_position is 2: Changes the color of &quot;Alarm&quot; to white and &quot;Speed&quot; to black. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>4) Key_position is 3: Changes the color of &quot;Navigation&quot; to white and &quot;Alarm&quot; to black. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>5) Key_position is 4: Changes the color of &quot;Function&quot; to white and &quot;Navigation&quot; to black. The other menu items are not changed.</td>
</tr>
</tbody>
</table>

#### Argument
- None

#### Return Value
- None

### view_up

<table>
<thead>
<tr>
<th>Overview</th>
<th>Menu option function (move up)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void view_up (void)</td>
</tr>
<tr>
<td>Description</td>
<td>1. This function calculates the key position (Key_position) when Up-SW is pressed to select the menu item displayed below the current one. According to the Key_position value, the displayed image is changed.</td>
</tr>
<tr>
<td></td>
<td>2. The changed image is output to the LCD (the display_img function).</td>
</tr>
<tr>
<td></td>
<td>1) Key_position is 0: Changes the color of &quot;Function&quot; to black and &quot;Navigation&quot; to white. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>2) Key_position is 1: Changes the color of &quot;Speed&quot; to black and &quot;Function&quot; to white. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>3) Key_position is 2: Changes the color of &quot;Alarm&quot; to black and &quot;Speed&quot; to white. The other menu items are not changed.</td>
</tr>
<tr>
<td></td>
<td>4) Key_position is 3: Changes the color of &quot;Navigation&quot; to black and &quot;Alarm&quot; to white. The other menu items are not changed.</td>
</tr>
</tbody>
</table>

#### Argument
- None

#### Return Value
- None
## view_enter

### Overview
Menu option function (select an item)

### Header
None

### Declaration
void view_enter (void)

### Description
This function switches the display on the LCD between the menu screen and the demonstration screen every time Enter-SW is pressed.

1. When Enter-SW is pressed while the LCD panel displays the menu, the selected demonstration is performed.

   1) When the key position is set to "Function" (Key_position = 0), the following processing is performed. As the sample code waits for the specified time after outputting the processed image from the GDT to the LCD, the LCD display update interval depends on the specified time.
      - The API function R_GDT_Open() is executed (GDT becomes active).
      - After the reversal process, the resultant image is output to the LCD (gdt_iflp_triple_seq).
      - After the shrink process, the resultant image is output to the LCD (gdt_shrink_big2small2big_seq).
      - After the monochrome synthesis process, the resultant image is output to the LCD (gdt_monochrome_icon_seq).
      - After the rotation process, the resultant image is output to the LCD (gdt_rotate_icon_seq).
      - After the scroll process, the resultant image is output to the LCD (gdt_scroll_left_7bit_seq).
      - The API function R_GDT_Close() is executed (GDT is disabled).

   2) When the key position is set to "Speed" (Key_position = 1), the processing 1) is performed without delay.

   3) When the key position is set to "Alarm" (Key_position = 2)
      - The API function R_GDT_Open() is executed (GDT becomes active).
      - The alarm function is executed (gdt_alarm_function).
      - The API function R_GDT_Close() is executed (GDT is disabled).

   4) When the key position is set to "Navigation" (Key_position = 3)
      - The API function R_GDT_Open() is executed (GDT becomes active).
      - Initial settings of navigation
      - Navigation function demonstration 1 (shrink the map size)
      - Navigation function demonstration 2 (move from map_aW to map_bW)
      - Navigation function demonstration 3 (move from map_bW to map_dW)
      - The API function R_GDT_Close() is executed (GDT is disabled).

2. When Enter-SW is pressed after the selected demonstration is completed, the display is switched from the demonstration screen to the menu screen.

### Argument
None

### Return Value
None
gdt_iflp_triple_seq

Overview
Reversal function demonstration

Header
None

Declaration
void gdt_iflp_triple_seq (uint16_t delay_times)

Description
This function reverses the input image with a size of 256 bits × 256 bits three times and outputs the result to the LCD after every reversal process.

1. The prepared input image (256 bits × 256 bits) is temporarily stored and then output to the LCD (the display_img function).
2. The following steps are repeated three times.
   1) The image is reversed (gdt_iflp_256x256).
   2) The result is output to the LCD (the display_img function).
      The reversed image is input to the next reversal process.

Argument
uint16_t delay_times

Delay time before the reversed image is displayed.
(This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)

Return Value
None

gdt_shrink_big2small2big_seq

Overview
Shrink function demonstration

Header
None

Declaration
void gdt_shrink_big2small2big_seq (uint16_t delay_times)

Description
This function shrinks the input image with a size of 256 bits × 256 bits, gradually changes the image size in the order of large → small → large, and outputs the result to the LCD. The shrunk image is displayed at the center of the LCD panel.

1. The array for temporarily storing the processed image is cleared (gdt_img_clr).
2. The prepared input image (256 bits × 256 bits) is temporarily stored and then output to the LCD (the display_img function).
3. The common settings for the shrink function are made.
4. The following steps are repeated seven times (the shrink size (shrink_ratio) is changed in the order of 7/8 → ... → 1/8 in steps of 1/8).
   1) The array for temporarily storing the processed image is cleared (gdt_img_clr).
   2) The shrink size is changed (the value of the shrink size storage variable shrink_size is decremented).
   3) The start coordinates of the output block is calculated (the horizontal start coordinate blk_start_pix_h and vertical start coordinate blk_start_pix_v are incremented by one block (16 bits)).
   4) The shrink function is executed (gdt_shrink_uniflp_black_bakgrd).
   5) The result is output to the LCD (the display_img function).
5. The following steps are repeated six times (the shrink function is executed six times with the shrink size changed from small to large).
   1) The array for temporarily storing the processed image is cleared (gdt_img_clr)
   2) The shrink size is changed (the value of the shrink size storage variable shrink_size is incremented).
   3) The start coordinates of the output block is calculated (the horizontal start coordinate blk_start_pix_h, and vertical start coordinate blk_start_pix_v are decremented by one block (16 bits)).
   4) The shrink function is executed (gdt_shrink_uniflp_black_bakgrd).
   5) The result is output to the LCD (the display_img function).

Argument
uint16_t delay_times

Delay time before the shrunk image is displayed.
(This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)
Return Value
None

<table>
<thead>
<tr>
<th>gdt_monochrome_icon_seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
</tr>
<tr>
<td>Header</td>
</tr>
<tr>
<td>Declaration</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

1. Endian conversion for the background and foreground images is set up.
2. Endian conversion is performed (gdt_endian_uniflp_one_page).
3. The foreground image shrunk to 4/8 the original size is placed at the center of the display area.
4. Parameters for the first monochrome synthesis process are specified.
   - Size: 256 bits × 256 bits
   - Trimming: Disabled
5. The first monochrome synthesis process is performed.
   1) The background image is displayed (display_img).
   2) The foreground image is displayed (display_img).
   3) Monochrome synthesis is executed (gdt_monochrome_uniflp).
   4) The image generated by monochrome synthesis is displayed (display_img).
   5) The image is cleared (gdt_img_clr).
   6) A white screen is displayed after the image is cleared (display_img).
6. Parameters for the second monochrome synthesis process are specified.
   1) The foreground image for the first process is used for the trimming image for the second process.
   2) Trimming is enabled.
7. The second monochrome synthesis process is performed.
   1) The foreground image is displayed (display_img).
   2) The background image is displayed (display_img).
   3) The trimming image is displayed (display_img).
   4) Monochrome synthesis is executed (gdt_monochrome_uniflp).
   5) The image generated by monochrome synthesis is displayed (display_img).

<table>
<thead>
<tr>
<th>Argument</th>
<th>uint16_t delay_times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay time before the image generated by monochrome synthesis is displayed. (This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Value</th>
<th>None</th>
</tr>
</thead>
</table>

R01AN4755EJ0105  Rev.1.05  Page 16 of 51  Jun.01. 2020
gdt_rotate_icon_seq

<table>
<thead>
<tr>
<th>Overview</th>
<th>Rotation function demonstration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void gdt_rotate_icon_seq (uint16_t delay_times)</td>
</tr>
<tr>
<td>Description</td>
<td>This function rotates the input image and outputs the result to the LCD. The function uses the input image prepared in advance according to &quot;Preparing a 1/2 image (icon) to be used in GDT processing&quot; in section 3.1 — that is, an image with a size of 128 bits × 128 bits obtained by shrinking the input image to 4/8 the original size.</td>
</tr>
</tbody>
</table>

1. The array for temporarily storing the processed image is cleared (gdt_img_clr).
2. The input data having a size of 128 bits × 128 bits is output to the coordinates (0,0) of the LCD.
3. The rotation process is performed with necessary parameters specified (gdt_rotate_uniflp_one_page) and the result is output to the LCD.
   1) Rotation direction: 90-degree clockwise
      Output destination coordinates: (128,0)
   2) Rotation is performed by the gdt_rotate_uniflp_one_page function.
   3) The result is output to the LCD (the display_img function).
4. The rotation process is performed with parameters updated (gdt_rotate_uniflp_one_page) and the result is output to the LCD.
   1) Rotation direction: 90-degree counterclockwise
      Output destination coordinates: (0,128)
   2) Rotation is performed by the gdt_rotate_uniflp_one_page function.
   3) The result is output to the LCD (the display_img function).
5. The data displayed on the LCD is cleared and the input image is output to the coordinates (0,0) of the LCD in the same way as step 2.
6. The rotation process is performed with parameters updated (gdt_rotate_uniflp_one_page) and the result is output to the LCD.
   1) Rotation direction: Horizontal flip
      Output destination coordinates: (128,0)
   2) Rotation is performed by the gdt_rotate_uniflp_one_page function.
   3) The result is output to the LCD (the display_img function).
7. The rotation process is performed with parameters updated (gdt_rotate_uniflp_one_page) and the result is output to the LCD.
   1) Rotation direction: Vertical flip
      Output destination coordinates: (0,128)
   2) Rotation is performed by the gdt_rotate_uniflp_one_page function.
   3) The result is output to the LCD (the display_img function).

<table>
<thead>
<tr>
<th>Argument</th>
<th>uint16_t delay_times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Delay time before the rotated image is displayed. (This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
gdt_scroll_left_7bit_seq

**Overview**
Scroll function demonstration

**Header**
None

**Declaration**
void gdt_scroll_left_7bit_seq (uint16_t delay_times)

**Description**
This function scrolls the input image and outputs the result to the LCD.

1. The prepared input image having a size of 256 bits × 256 bits is output to the LCD (the display_img function).
2. The following steps (scroll to the left by 7 bits) are repeated 30 times.
   1) The array for temporarily storing the processed image is cleared (gdt_img_clr)/
   2) The block size for scrolling is updated.
   3) The scroll process is performed (gdt_scroll_uniflp).
   4) The scrolled image is output to the LCD (the display_img function).
   5) The offset for scrolling is updated (8 is added to the offset).
3. The offset for scrolling is specified (240).
4. The following steps (scroll to the right by 7 bits) are repeated 30 times.
   1) The array for temporarily storing the processed image is cleared (gdt_img_clr).
   2) The block size for scrolling is updated.
   3) The scroll process is performed (gdt_scroll_uniflp).
   4) The scrolled image is output to the LCD (the display_img function).
   5) The offset for scrolling is updated (8 is subtracted from the offset).

**Argument**
uint16_t delay_times

Delay time before the scrolled image is displayed.
(This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)

**Return Value**
None

gdt_alarm_function

**Overview**
Alarm function demonstration

**Header**
None

**Declaration**
void gdt_alarm_function (void)

**Description**
This function uses the monochrome synthesis and reversal processes to implement an alarm.

1) Eleven alarm images are stored (alarm01 to alarm11).
   Background image (alarm01): 256 bits × 256 bits
   Foreground image (alarm06): 144 bits × 32 bits
   Foreground images (alarm02 to alarm05 and alarm07 to alarm11): 32 bits × 32 bits
2) Endian conversion for the alarm background is set up.
3) Endian conversion is performed (gdt_endian_uniflp_one_page).
4) The data after endian conversion is output to the LCD (the display_img function).
5) Parameters for monochrome synthesis are specified.
6) Ten foreground images and a background image are combined by monochrome synthesis and the result is output to the LCD.
   • Loop = 4 (alarm06)
     A foreground image with a size of 144 bits × 32 bits is combined with the background and output to the LCD.
   • Loop = 5 and Loop = 7 (alarm07 and alarm09)
     A foreground image with a size of 32 bits × 32 bits is reversed in monochrome, combined with the background, and output to the LCD.
   • Others
     A foreground image with a size of 32 bits × 32 bits is combined with the background without any other processing performed and is output to the LCD.
<table>
<thead>
<tr>
<th>Argument</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

gdt_navigation_int

**Overview**
Navigation input image initialization

**Header**
None

**Declaration**
void gdt_navigation_int (void)

**Description**
This function places the starting point and destination on the initial navigation screen and outputs the screen image to the LCD. The input image should be prepared in advance.

1. Endian conversion is applied to the initial navigation screen and the result is output to the LCD.
2. Endian conversion is applied to the screen image containing the text that indicates the starting point and destination of navigation.
3. The synthesis process is performed with necessary parameters specified and the result is output to the LCD.
   
   **Background image:** Initial navigation screen
   
   **Foreground image:** Image containing the text that indicates the starting point and destination of navigation

4. Endian conversion is applied to the following seven images to be used in navigation (gdt_endian_uniflp_one_page).
   
   1) map_aW: Map A
   2) map_bW: Map B
   3) map_cW: Map C
   4) map_dW: Map D
   5) human: Human icon
   6) human_border: Border of human icon
   7) compassW: Compass icon

<table>
<thead>
<tr>
<th>Argument</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

gdt_navigation_1

**Overview**
Navigation demonstration 1 (enlarge the map)

**Header**
None

**Declaration**
void gdt_navigation_1 (void)

**Description**
This function gradually enlarges the whole map with the starting point kept at the center of the map and outputs the results to the LCD.

1. Four input maps are shrunk (aW: 4/8, bW: 4/8, cW: 4/8, and dW: 4/8) and combined with the compass, and the results are output to the LCD.
   
   1) Input map A is shrunk with necessary parameters specified (size: 4/8, display position: top-left).
      (gdt_shrink_uniflp_black_bakgrd)
   2) Input map B is shrunk with necessary parameters specified (size: 4/8, display position: top-right).
      (gdt_shrink_uniflp_black_bakgrd)
   3) Input map C is shrunk with necessary parameters specified (size: 4/8, display position: bottom-left).
      (gdt_shrink_uniflp_black_bakgrd)
   4) Input map D is shrunk with necessary parameters specified (size: 4/8, display position: bottom-right).
      (gdt_shrink_uniflp_black_bakgrd)
   5) The compass is combined with the images 1) to 4)
      (gdt_monochrome_compassW(1)) and the results are output to the LCD.
2. The four input maps are shrunk in the same way as step 1 and combined with the compass, and the results are output to the LCD. Note that the aW map size is changed to 5/8 and the other map sizes are the same as those in step 1.

3. The four input maps are shrunk in the same way as step 1 and combined with the compass, and the results are output to the LCD. Note that the aW map size is changed to 6/8 and the other map sizes are the same as those in step 1.

4. The four input maps are shrunk in the same way as step 1 and combined with the compass, and the results are output to the LCD. Note that the aW map size is changed to 7/8 and the other map sizes are the same as those in step 1.

5. Map A is combined with the compass with the original map size (256 × 256) kept unchanged and the result is output to the LCD.

6. Both map A with the original size (256 × 256) and the compass are rotated counterclockwise and then combined, and the result is output to the LCD.

---

**gdt_navigation_2**

- **Overview**: Navigation demonstration 2 (move from map_aW to map_bW)
- **Header**: None
- **Declaration**: void gdt_navigation_2 (void)
- **Description**: This function scrolls the navigation map from map_aW to map_bW and outputs the result to the LCD.
  1. An image is generated from the second half of map A and the first half of map B and the image data is temporarily stored (map AandB).
  2. Map A is rotated counterclockwise with necessary parameters specified, monochrome synthesis is performed to combine the map, human icon, and compass icon, and the result is output to the LCD.
  3. Map A is scrolled to the left and the valid image data is temporarily stored. To fill the right-hand blank space of the scrolled map, data is fetched from map AandB stored in step 1 and a complete map image is composed.
  4. The image data obtained in step 3 is combined with the human icon and compass icon and the result is output to the LCD.
  5. Steps 3 and 4 are repeated 15 times.
  6. Map AandB stored in step 1 is scrolled to the left and the valid image data is temporarily stored. To fill the right-hand blank space of the scrolled map, data is fetched from map B and a complete map image is composed.
  7. The image data obtained in step 6 is combined with the human icon and compass icon and the result is output to the LCD.

- **Argument**: None
- **Return Value**: None
gdt_navigation_3

Overview
Navigation demonstration 3 (move from map_bW to map_dW)

Header
None

Declaration
void gdt_navigation_3 (void)

Description
This function scrolls the navigation map from map_bW to map_dW and outputs the result to the LCD.

1. Maps B and D are rotated counterclockwise, and then an image is generated from the second half of map B and the first half of map D. The image data is temporarily stored (map BandD).
2. Map B is rotated counterclockwise with necessary parameters specified, monochrome synthesis is performed to combine the map, human icon, and compass icon, and the result is output to the LCD.
3. Map B is scrolled to the left and the valid image data is temporarily stored. To fill the right-hand blank space of the scrolled map, data is fetched from map BandD stored in step 1 and a complete map image is composed.
4. The image data obtained in step 3 is combined with the human icon and compass icon and the result is output to the LCD.
5. Steps 3 and 4 are repeated 15 times.
6. Map BandD stored in step 1 is scrolled to the left and the valid image data is temporarily stored. To fill the right-hand blank space of the scrolled map, data is fetched from map D and a complete map image is composed.
7. The image data obtained in step 6 is combined with the human icon and compass icon and the result is output to the LCD.

Argument
None

Return Value
None

gdt_shrink_uniflp_black_bakgrd

Overview
Shrink function with the reversal function disabled and the color of the excess background area set to white (for demonstration)
This function sets up the arguments of the shrink API function (R_GDT_Shrink) and calls the function.

Header
None

Declaration
void gdt_shrink_uniflp_black_bakgrd (uint8_t shrink_size, shrink_in_info_t in_info, shrink_out_info_t out_info)

Description
1. The input image size for the shrink function is specified.
2. The output image size for the shrink function is specified.
3. The start coordinates of the input image and the size of the area to be shrunk are specified. The start coordinates of output image after the shrink process are specified.
4. Parameters for the shrink processing are specified.
   1) Reversal is disabled.
   2) The shrink size is specified (passed through an argument of this function).
   3) The color of the excess background area is set to white (1).
5. The shrink function R_GDT_Shrink is executed.
6. Execution waits for the shrink end flag (shrink_normal_end) to be set.
7. The shrink end flag (shrink_normal_end) is cleared.

Argument
uint8_t shrink_size
shrink_in_info_t in_info
shrink_out_info_t out_info

Shrink size
Information of the input image for the shrink function (size) and the block to be processed (start coordinates and processing size)
Information of the output image (start coordinates and
**Return Value**  None

---

### gdt_iflp_256x256

**Overview**  256 × 256 entire screen reversal function (for demonstration)
This function sets up the arguments of the endian conversion API function (R_GDT_Endian) and calls the function.

**Header**  None

**Declaration**  void gdt_iflp_256x256(uint32_t src_img_addr,uint32_t dest_img_addr)

**Description**
1. The input image size for endian conversion is specified.
2. The output image size for endian conversion is specified.
3. The start coordinates of the input and output images for endian conversion are fixed to (0,0) and the image size is fixed to 256 × 256.
4. Parameters for endian conversion are specified.
   1) Endian conversion is disabled.
   2) Reversal is enabled.
5. The endian conversion function R_GDT_Endian is executed.
6. Execution waits for the endian conversion end flag (Endian_normal_end) to be set.
7. The endian conversion end flag (Endian_normal_end) is cleared.

**Argument**
- uint32_t src_img_addr  Input image address
- uint32_t dest_img_addr  Output image address

**Return Value**  None

---

### gdt_endian_uniflp_one_page

**Overview**  Entire screen endian conversion function (for demonstration)
This function sets up the arguments of the endian conversion API function (R_GDT_Endian) and calls the function.

**Header**  None

**Declaration**  void gdt_endian_uniflp_one_page(endian_info_t endian_info)

**Description**
1. The input image size for endian conversion is specified.
2. The output image size for endian conversion is specified.
3. The start coordinates of the input and output images for endian conversion are fixed to (0,0) and the input image size for endian conversion is specified.
4. Parameters for endian conversion are specified.
5. Endian conversion is disabled.
6. Reversal is enabled.
7. The endian conversion function R_GDT_Endian is executed.
8. Execution waits for the endian conversion end flag (Endian_normal_end) to be set.
9. The endian conversion end flag (Endian_normal_end) is cleared.

**Argument**
- endian_info_t endian_info  Input image address, output image address, and the size of the target area of endian conversion

**Return Value**  None
### gdt_monochrome_uniflp

**Overview**
Entire screen monochrome synthesis function (for demonstration)
This function sets up the arguments of the endian conversion API function (R_GDT_Monochrome) and calls the function.

**Header**
None

**Declaration**
void gdt_monochrome_uniflp(monochrome_info_t cfg_info, monochrome_range_info_t range_info)

**Description**
1. The number of input images for monochrome synthesis is specified
   1) With trimming enabled: Three input images (foreground, background, and trimming)
   2) With trimming disabled: Two input images (foreground and background)
2. The output image size for monochrome synthesis is specified.
3. The start coordinates of the input image and the size of the target area of synthesis are specified when trimming is enabled (three images are input). The start coordinates of the output image after synthesis are fixed to (0,0).
4. Parameters for monochrome synthesis are specified.
5. Monochrome synthesis is enabled.
6. Reversal is disabled.
7. The monochrome synthesis function R_GDT_Monochrome is executed.
8. Execution waits for the monochrome synthesis end flag (Monochrome_normal_end) to be set.
9. The monochrome synthesis end flag (Monochrome_normal_end) is cleared.

**Argument**
- monochrome_info_t cfg_info
  - Foreground, background, and trimming addresses, and synthesis mode
- monochrome_range_info_t range_info
  - Input image size information

**Return Value**
None

### gdt_rotate_uniflp_one_page

**Overview**
Entire screen rotation function (for demonstration)
This function sets up the arguments of the rotation API function (R_GDT_Rotate) and calls the function.

**Header**
None

**Declaration**
void gdt_rotate_uniflp_one_page(rotate_info_t rotate_info)

**Description**
1. The input image size for rotation is specified.
2. The output image size for rotation is specified.
3. The start coordinates of the input image for rotation are fixed to (0,0) and the size of the target area of rotation are specified. The start coordinates of the output image after rotation are specified.
4. Parameters for rotation are specified.
   1) Reversal is disabled.
   2) Rotation is enabled.
   3) The rotation type select bits are set up.
5. The rotation function R_GDT_Rotate is executed.
6. Execution waits for the rotation end flag (rotate_normal_end) to be set.
   The rotation end flag (rotate_normal_end) is cleared.

**Argument**
- rotate_info_t rotate_info
  - Input image address, output image address, and the size of the target area of endian conversion

**Return Value**
None
### gdt_scroll_uniflp

**Overview**
Entire screen scroll function (for demonstration)
This function sets up the arguments of the scroll API function (R_GDT_Scroll) and calls the function.

**Header**
None

**Declaration**

```c
void gdt_scroll_uniflp(scroll_info_t scroll_info,uint8_t scroll_iscren)
```

**Description**

1. The input image size for scrolling is specified.
2. The output image size for scrolling is specified.
3. The start coordinates of the input image for scrolling and the size of the target area of scrolling are specified. The start coordinates of the output image after scrolling are specified.
4. Parameters for scrolling are specified.
   1) Reversal is disabled.
   2) The image scrolling enable bits are set up.
5. The scroll function R_GDT_Scroll is executed.
6. Execution waits for the scroll end flag (Scroll_normal_end) to be set. The scroll end flag (Scroll_normal_end) is cleared.

**Argument**

- `scroll_info_t scroll_info`, Input image address, output image address, and the size of the target area of endian conversion
- `uint8_t scroll_iscren`, Image scrolling enable bit setting

**Return Value**
None

### menu_merge_image

**Overview**
Function for updating the image displayed as the menu

**Header**
None

**Declaration**

```c
void menu_merge_image(uint32_t View_X,uint32_t View_Y,const unsigned char image[1792])
```

**Description**
This function updates the image data displayed as the menu. The update size is set to the size of the image for each demonstration name (256 bits × 56 bits).

**Argument**

- `uint32_t View_X`, X coordinate of the update start address
- `uint32_t View_Y`, Y coordinate of the update start address
- `const unsigned char image[1792]`, New image

**Return Value**
None

### display_img

**Overview**
LCD output function

**Header**
None

**Declaration**

```c
void display_img(unsigned char img[],uint16_t img_h_size,uint16_t img_v_size,uint16_t delay_times,uint8_t reverse_en)
```

**Description**
This function outputs the input image to the LCD.

1. The array for temporarily storing the input image is cleared.
2. Endian conversion is enabled or disabled.
3. The input image is temporarily stored (after endian conversion when conversion is enabled).
4. The image stored in the array is output to the LCD through the SPI.
5. Execution ends after waiting for the delay time.

**Argument**

- `unsigned char img[]`, Input image array
- `uint16_t img_h_size`, Horizontal size (bits) of the input image
- `uint16_t img_v_size`, Vertical size (bits) of the input image
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint16_t</code> delay_times</td>
<td>Delay time after the output to the LCD (This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)</td>
</tr>
<tr>
<td><code>uint8_t</code> reverse_en</td>
<td>Endian conversion enable or disable setting</td>
</tr>
</tbody>
</table>

Return Value: None
### Event Link Settings

Table 3-2 shows the ICU event link settings.

<table>
<thead>
<tr>
<th>Interrupt Linked with NVIC</th>
<th>Location of Change</th>
<th>Details of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAC0_INT</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER0)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT</td>
<td></td>
</tr>
<tr>
<td>DMAC1_INT</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER5)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_DMAC1_INT</td>
<td></td>
</tr>
<tr>
<td>DMAC2_INT</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER14)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_DMAC2_INT</td>
<td></td>
</tr>
<tr>
<td>GDT_DATOI</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER4)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_GDT_DATOI</td>
<td></td>
</tr>
<tr>
<td>GDT_DATII</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER3)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_GDT_DATII</td>
<td></td>
</tr>
<tr>
<td>AGT_AGTI</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER11)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTI</td>
<td></td>
</tr>
<tr>
<td>SPI1_SPIII (RE01 1500KB Group only)</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER10)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_SPI1_SPIII</td>
<td></td>
</tr>
<tr>
<td>SPI0_SPIII (RE01 256KB Group only)</td>
<td>[r_system_cfg.h]</td>
<td>Setting change (SYSTEM_IRQ_EVENT_NUMBER2)</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_SPI0_SPIII</td>
<td></td>
</tr>
</tbody>
</table>
3.1.4 Flowcharts
Figure 3.3 shows a flowchart of the main processing.

![Flowchart of Main Processing](image-url)
Figure 3.4 to Figure 3.8 show flowcharts of individual image processes when Key_position is calculated as 1 or 2 by the view_enter function.

Figure 3.4 shows a flowchart of the reversal processing.

```
Start

Output an input image to LCD

Specify parameters for reversal process

If lp_cnt = 0

lp_cnt ++

lp_cnt < 3

End

Output reversed result to LCD
```

Figure 3.4 Reversal Processing
Figure 3.5 shows a flowchart of the shrink processing.

Start

Output an input image to LCD

Specify parameters for shrink process

Shrink_cnt=0

NO

Shrink_cnt<7

Execute shrink function 6 times.

Shrink_size ~
Shrink_cnt++
pix_h=pix_h+16   pix_v=pix_v+16

YES

End

Output an input image to LCD

Shrink_cnt<6

Execute shrink function 5 times.

Shrink_size ++
Shrink_cnt++
pix_h=pix_h-16   pix_v=pix_v-16

NO

Output an input image to LCD

gdt_shrink_uniflp_black_bakgrd
Output shrunk image to LCD.

Figure 3.5 Shrink Processing
Figure 3.6 shows a flowchart of the monochrome synthesis processing.

```
Start

Specify parameters for endian conversion
gdt_endian_uniflp_one_page

Specify parameters for the first monochrome synthesis process

Display background and foreground images

gdt_monochrome_uniflp
Output the image generated by monochrome synthesis to LCD

Clear image
Output the white screen after the clear to LCD

Display background, foreground, and trimming images

Specify parameters for the second monochrome synthesis process
Use the previous foreground image for current trimming
Enable trimming

gdt_monochrome_uniflp
Output the image generated by monochrome synthesis to LCD

End
```

Figure 3.6   Monochrome Synthesis Processing
Figure 3.7 shows a flowchart of the rotation processing.

Start

Output an input image to LCD

Specify parameters for rotation process
90-degree clockwise rotation
Output destination coordinates: (128,0)

\texttt{gdt\_rotate\_uniflp\_one\_page}
Output the result of rotation to LCD

Update parameters for rotation process
90-degree counterclockwise rotation
Output destination coordinates: (0,128)

\texttt{gdt\_rotate\_uniflp\_one\_page}
Output the result of rotation to LCD

Output an input image to LCD

Specify parameters for rotation process
Horizontal flip rotation
Output destination coordinates: (128,0)

\texttt{gdt\_rotate\_uniflp\_one\_page}
Output the result of rotation to LCD

Update parameters for rotation process
Vertical flip rotation
Output destination coordinates: (0,128)

\texttt{gdt\_rotate\_uniflp\_one\_page}
Output the result of rotation to LCD

Output an input image to LCD

End

Figure 3.7 Rotation Processing
Figure 3.8 shows a flowchart of the scroll processing.

![Flowchart of Scroll Processing]

- **Start**
- Output an input image to LCD
- Specify parameters for scrolling
- scroll_times = 0
- scroll_offset % 16
  - NO: src_blk_pix_h = 0 + scroll_offset, blk_size_h = 248 - scroll_offset
  - YES: src_blk_pix_h = 0 + scroll_offset, blk_size_h = 256 - scroll_offset
- gdt_scroll_uniflp (Output scrolled image to LCD)
- scroll_times < 30
  - NO: scroll_offset = scroll_offset - 8
  - YES: scroll_offset = scroll_offset + 8
- scroll_times < 60
  - NO: scroll_times++
  - YES: End
3.2 Sample Code of Font Unfold Function for Monochrome LCDs

The sample code of the font unfold function for monochrome LCDs operates as follows.

- Initialization
  
  Necessary initialization processing is performed.
  
  1) System initialization
  2) System clock settings
  3) I/O power control (all I/O power supplies are turned on)
  4) Initial settings of MLCD display by the R_SMIP driver (API functions R_SMIP_Open() and R_SMIP_PowerOn() are executed)

- The API function R_GDT_Open() is executed to activate the GDT.

- Executing the font unfold functions
  
  1) Font conversion when the input size is 63 bits \times 64 bits
  2) Font conversion when the input size is 33 bits \times 31 bits

- Executing the API function R_GDT_Close() to disable the GDT
3.2.1 System Configuration

**Figure 3.9** System Configuration for RE01 1500KB group

**Figure 3.10** System Configuration for RE01 256KB group
3.2.2 List of Functions
The functions added to the sample code are described here. Modify them as necessary.

### main

<table>
<thead>
<tr>
<th>Overview</th>
<th>Main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void main(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This function sets up the system clock and SMIP (SPI and DMAC) transfer. The sample code repeats the following steps within the main loop. 1) The API function R_GDT_Open() is executed to activate the GDT. 2) Font data with an output image size of 63 bits × 64 bits is unfolded. 3) Font data with an output image size of 33 bits × 31 bits is unfolded. 4) The API function R_GDT_Close() is executed to disable the GDT.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### gdt_fount_63x64_seq

<table>
<thead>
<tr>
<th>Overview</th>
<th>Font unfolding demonstration (output image size: 63 bits × 64 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void gdt_fount_63x64_seq (uint16_t delay_times)</td>
</tr>
<tr>
<td>Description</td>
<td>This function unfolds the font data of the input image and outputs the result to the LCD. 1. The information of font unfolding (input and output image addresses and output image size) is specified. 2. The output image array is cleared. 3. The prepared input image is output to the LCD (the display_img function). 4. Font unfolding is performed (gdt_fount_uniflp_nofill). 5. The image after font unfolding is output to the LCD.</td>
</tr>
<tr>
<td>Argument</td>
<td>uint16_t delay_times</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

Sample input image: 64 bits × 63 bits

![Sample input image](image)

Sample output image after font unfolding: 63 bits × 64 bits

![Sample output image](image)
### gdt_fount_33x31_seq

**Overview**  
Font unfolding demonstration (output image size: 33 bits × 31 bits)

**Header**  
None

**Declaration**  
void gdt_fount_33x31_seq (uint16_t delay_times)

**Description**  
This function unfolds the font data of the input image and outputs the result to the LCD.

1. The information of font unfolding (input and output image addresses and output image size) is specified.
2. The output image array is cleared.
3. The prepared input image is output to the LCD (the display_img function).
4. Font unfolding is performed (gdt_fount_uniflp_nofill).
5. The image after font unfolding is output to the LCD.

Sample input image: 64 bits × 16 bits

![Sample input image](image)

Sample output image after font unfolding: 33 bits × 31 bits

![Sample output image](image)

**Argument**  
uint16_t delay_times  
Delay time before the resultant image of font unfolding is displayed  
(This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)

**Return Value**  
None

### gdt_fount_uniflp_nofill

**Overview**  
Font unfold function (for demonstration)

This function sets up the arguments of the font unfold API function R_GDT_Fount and calls the function.

**Header**  
None

**Declaration**  
void gdt_fount_uniflp_nofill(fount_info_t fount_info)

**Description**  
1. The input image size for font unfolding is specified.
2. The output image size for font unfolding is specified.
3. The coordinates of the input and output images for font unfolding are fixed to (0, 0) and the block size is fixed to 0.
4. The mode for font unfolding is specified.
   1) Unused bit: Fixed to 0.
   2) Start address change bit: Fixed to 0.
   3) The horizontal size of font data is specified.
   4) The vertical size of font data is specified.
   5) Reversal is disabled.
5. The font unfold API function (R_GDT_Fount) is executed.
6. Execution waits for the font unfold end flag (gdt_normal_end) to be set. After font unfolding ends, the flag (gdt_normal_end) is cleared.
### display_img

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fount_info_t</td>
<td>Input image address, output image address, horizontal size of font data, and vertical size of font data</td>
</tr>
<tr>
<td>fount_info</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

#### Overview
LCD output function

#### Header
None

#### Declaration
```c
void display_img(unsigned char img[], uint16_t img_h_size, uint16_t img_v_size, uint16_t delay_times, uint8_t reverse_en)
```

#### Description
1. This function outputs the input image to the LCD.
2. The array for temporarily storing the input image is cleared.
3. Endian conversion is enabled or disabled.
4. The input image is temporarily stored (after endian conversion when conversion is enabled).
5. The image stored in the array is output to the LCD through the SPI.
6. Execution ends after waiting for the delay time.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char img[]</td>
<td>Input image array</td>
</tr>
<tr>
<td>uint16_t img_h_size</td>
<td>Horizontal size (pixels) of the input image</td>
</tr>
<tr>
<td>uint16_t img_v_size</td>
<td>Vertical size (pixels) of the input image</td>
</tr>
<tr>
<td>uint16_t delay_times</td>
<td>Delay time after the output to the LCD (This argument can be specified in 1-ms units. When 2 is specified, the delay time is 2 ms.)</td>
</tr>
<tr>
<td>uint8_t reverse_en</td>
<td>Endian conversion enable or disable setting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
### 3.2.3 Event Link Settings

Table 3-3 shows the ICU event link settings.

#### Table 3-3 ICU Event Link Settings

<table>
<thead>
<tr>
<th>Interrupt Linked with NVIC</th>
<th>Location of Change</th>
<th>Details of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAC0_INT</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER0)</td>
</tr>
<tr>
<td>DMAC1_INT</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_DMAC1_INT</td>
<td>(SYSTEM_IRQEVENT_NUMBER5)</td>
</tr>
<tr>
<td>DMAC2_INT</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_DMAC2_INT</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER14)</td>
</tr>
<tr>
<td>GDT_DATOII</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_GDT_DATOII</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER4)</td>
</tr>
<tr>
<td>GDT_DATII</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_GDT_DATII</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER3)</td>
</tr>
<tr>
<td>AGT_AGTI</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTI</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER11)</td>
</tr>
<tr>
<td>SPI1_SPIII (RE01 1500KB Group only)</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_SPI1_SPIII</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER10)</td>
</tr>
<tr>
<td>SPI0_SPIII (RE01 256KB Group only)</td>
<td>[r_system_cfg.h]</td>
<td>• Setting change</td>
</tr>
<tr>
<td></td>
<td>SYSTEM_CFG_EVENT_NUMBER_SPI0_SPIII</td>
<td>(SYSTEM_IRQ_EVENT_NUMBER2)</td>
</tr>
</tbody>
</table>
3.2.4 Flowcharts

Figure 3.11 shows a flowchart of the main processing.

![Flowchart of Main Processing](image-url)
Figure 3.12 shows a flowchart of the font unfold processing.

```
Start

Specify parameters for font unfolding

Output an input image to LCD.

gdt_fount_uniflp_nofill

Output the image after font unfolding to LCD.

End
```

Figure 3.12   Font Unfold Processing
3.3 Method for Generating Monochrome Image Data

The procedure for generating image data to be used in a demonstration is shown in the following.

3.3.1 Preparation
Convert the format of the target image from the BMP format to the LCD output.

1. BMP-format image
2. Image format conversion tool: bmp2cs12 (BMP format → CSV format)
3. LCD output format conversion tool:
   r01an4755xx0100-re-gdt/Data_creation_sheet_for_LCD_output.xlsx
   (CSV format → LCD output format)

3.3.2 Restriction on Use
The image format conversion tool (bmp2cs12) used in this application note is free software. Convert the image format using any free software you wish to use.

3.3.3 Procedure
The procedure for converting the format of an image from the BMP format into the LCD output format is introduced here with Figure 3.13 as an example.

![Figure 3.13 icon.bmp](icon.bmp)

1. Prepare a monochrome BMP image such as that shown in Figure 3.13. The size for a full-screen display of the image is 256 × 256 (bits).

2. Execute (double-click on) bmp2csv.exe in bmp2cs12.lzh, and the message box in Figure 3.14 appears. Click on the [Execute] button.

![Figure 3.14 Message Box for Confirming bmp2csv Execution](message_box.png)
3. Select a BMP file in the dialog box in Figure 3.15, and a visualization of the input image is displayed in the lower left of the dialog box.

![Figure 3.15  UI Screen of bmp2csv](image)

4. Select "Save in file after binarization (white = 0, black = 1)." as the conversion format and select ".CSV (comma-separated values)" as the extension, as shown in Figure 3.16.

![Figure 3.16  Selecting Output Format of bmp2csv](image)
5. Click on "Create CSV file" to create a CSV file, as shown in Figure 3.17. In this example, "iconW.CSV" is generated in the same folder as the input image.

![Figure 3.17 Creating a CSV File of bmp2csv](image)

6. Before using the "Data_creation_sheet_for_LCD_output.xlsx" tool, be sure to thoroughly read the explanation of the procedure for using the tool in the "Cover_and_Setting" sheet of the tool.

7. Set values in "-Setting image size and additional data" at the bottom of the "Cover_and_Setting" sheet, as shown in Figure 3.18.

![Figure 3.18 Excerpt from "Cover_and_Setting" Sheet](image)
8. Open the "Data entry" sheet and paste the data generated in step 5 from the B3 cell of the "Data entry" sheet, as shown in Figure 3.19.

![Figure 3.19 Data Entry](image)

9. Data in the LCD output format is automatically displayed in the "Output" sheet of the "Data_creation_sheet_for_LCD_output.xlsx" tool, as shown in Figure 3.20. Change the settings in the first six rows (marked in yellow) according to the situation.

![Figure 3.20 Setting Output Data](image)
10. Copy the data from the D14 cell in the "Output" sheet to a desired "*.h" file. In this example, the data is copied to "an4755_hal_gdt_mono_re/demo_view/pic_icon.h".

11. Since the settings of "Setting image size and additional data" at the bottom of the "Cover_and_Setting" sheet differ depending on the purpose of using the data in the "Output" sheet, details are shown below.

1) When the data is output to an LCD screen, set "Address part size (in bytes)" and "Dummy cycle part size (in bytes)", as shown in Figure 3.21. The visualization that is output to the "Output" sheet is shown in Figure 3.22.

![Figure 3.21 Settings for Output to an LCD](image1)

![Figure 3.22 Visualization of Data Output for Output to an LCD](image2)

2) When the data is used in image processing, set 0 in "Address part size (in bytes)" and "Dummy cycle part size (in bytes)", as shown in Figure 3.23. The visualization that is output to the "Output" sheet is shown in Figure 3.24.

![Figure 3.23 Settings for Use in Image Processing](image3)
Figure 3.24   Visualization of Data Output for Use in Image Processing
4. Specifications of Driver APIs

4.1 External Specification Document

This driver package contains a document that describes the API external specifications. It is stored in the Driver Specification folder that is directly below the Documents folder.
5. Usage Notes of R_GDT Driver

This chapter introduces the main points to concern regarding the R_GDT driver. Note that not all the usage notes are given here.

For other notes, see "4.1 External Specification Document".

5.1 Sizes of Image Data

The horizontal size and vertical size of image data must be a multiple of 8 (bits). Also, the horizontal size of an image must be equal to or smaller than the horizontal size of memory.

5.2 Horizontal Size of Memory

In this driver, the size of the work area has to be specified with an argument. Select from the following choices a size that is at least equal to the size of the image to be processed.

Set the horizontal size of memory as 32, 64, 128, or 256 bits. However, set 96, 192, 384, or 768 bits when the 3-bit data mode of the color data sorting function is used and set 128, 256, 512, or 1024 bits when the 4-bit data mode is used.

5.3 Image Processing Unit

The image processing unit for the rotation function (R_GDT_Rotate), monochrome synthesis function (R_GDT_Monochrome), or color synthesis function (R_GDT_Colorsyn) can be selected from two types: 8 bits \( \times \) 8 bits or 16 bits \( \times \) 16 bits. Specify it using the image processing data size select variable (gdtdsz) of each function.

When using the rotation function with an image processing unit of 16 bits \( \times \) 16 bits, the start address of a block before and after processing should be set to an even address.

The monochrome synthesis function and color synthesis function support only an image processing unit of 8 bits \( \times \) 8 bits. When using these functions, set gdtdsz = 1.

5.4 DMAC Interrupt Settings and Restriction on Priority

This driver always uses a total of two DMAC channels. Specify the channels to be used so that they do not overlap with other DMAC channels used by other functions.

The interrupt priority of the DMAC channels used by this driver is limited. Specify the DMAC channels to be used in the R_GDT_DmacChSel function. "DMAC for transferring input data to GDT" and "DMAC for transferring output data" are to be set in this function. Set the interrupt priority of the DMAC channels selected above to satisfy the following restriction.

* Restriction: Interrupt priority of used DMAC channels
  "Channel used for transferring input data" > "Channel used for transferring output data"

The interrupt priority of each DMAC channel can be set in the respective macro definition "DMAc_n_INT_PRIORITY (n = 0 to 3)" of "Device/Config/r_dmac_cfg.h".

Example: When channel 1 is used for input and channel 2 is used for output

```
#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. Troubleshooting

6.1 Occurrence of a Build Error with the IAR Compiler

A-1) Have the include directories been specified correctly?

When using EWARM, we recommend that the include directories be specified as shown in the example below.

The include directories can be specified from IDE Options [C/C++ Compiler] → [Preprocessor].

6.2 Occurrence of HardFault Error when API of CMSIS Driver Is Called

A) The API has possibly not been copied to RAM.

Before calling an API function that is mapped to RAM, make sure that it has been copied to RAM by the R_SYS_CodeCopy function. For details, refer to the related document No. R01AN4660.

6.3 Peripheral Function Fails to Operate when API Is Called

A) Has the API been set up correctly?

Check the API's return value to see if an error has occurred.

Errors are often caused by problems related to interrupts not being set in r_system_cfg.h. For details, refer to the related document No. R01AN4660.

6.4 Normal API Return Value But No Pin Output from Peripheral Function

A) Are the pin settings correct?

Check to make sure the pins have been set up correctly by the functions in pin.c.

For details, refer to the related document No. R01AN4660.

6.5 Peripheral Function’s Input or Output Does Not Operate as Expected

A) Check to make sure the VOCR register has been set up correctly before making the initial settings for peripheral functions.

For details, refer to the related document No. R01AN4660.
7. Sample Code
Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents
User’s Manual: Hardware
   RE01 1500KB Group User’s Manual: Hardware R01UH0796
   RE01 256KB Group User’s Manual: Hardware R01UH0894
   (The latest version can be downloaded from the Renesas Electronics website.)

RE01 1500KB, 256KB CMSIS Package Startup Guide
   RE01 1500KB, 256KB Group Startup Guide to Development Using CMSIS Package R01AN4660
   (The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News
   (The latest version can be downloaded from the Renesas Electronics website.)

User’s Manual: Development Tools
   (The latest version can be downloaded from the Renesas Electronics website.)
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Oct. 18, 2019</td>
<td>-</td>
<td>-</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Jan. 15, 2020</td>
<td>GDT driver updated to Ver.1.00b</td>
<td>Program</td>
<td>GDT driver updated to Ver.1.00b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temporarily corrected deficiency of internal function</td>
<td></td>
<td>&quot;v_gdt_cpuline_byte_rd_gdt_limit&quot;</td>
</tr>
<tr>
<td>1.03</td>
<td>Apr. 17, 2020</td>
<td>Replaced CMSIS Driver Package</td>
<td>Program</td>
<td>Replaced CMSIS Driver Package</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMSIS Driver Package Rev.1.10A : Fixed ToolNews R20TS0574</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.04</td>
<td>Apr. 17, 2020</td>
<td>Add RE01 256KB group.</td>
<td>Program</td>
<td>Add RE01 256KB group.</td>
</tr>
<tr>
<td>1.05</td>
<td>Jun. 1, 2020</td>
<td>Change the name of the sample code project</td>
<td>3, 7, 8, 33, 35, 36, 39 Program (256KB)</td>
<td>Change the name of the sample code project</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated &quot;Operating Conditions&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified the description of the font expansion process to match the program</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replaced CMSIS Driver Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- RE01 256KB: CMSIS Driver Package Rev.1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Change function name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- gdt_fount_33x32_seq to gdt_fount_33x31_seq</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Change the Font unfolding size.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 33x32 to 33x31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.
   Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements.
   Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between VIL (Max.) and VIH (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between VIL (Max.) and VIH (Min.).

7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product or system depend on the product’s quality grade as indicated below.

   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks
Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

© 2020 Renesas Electronics Corporation. All rights reserved.