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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends 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; and industrial robots etc.

- **“High Quality”**: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

11. This document may not be 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, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.
How to Use This Manual

This manual describes the role of the CubeSuite+ integrated development environment for developing applications and systems for 78K0 microcontrollers, and provides an outline of its features.

CubeSuite+ is an integrated development environment (IDE) for 78K0 microcontrollers, integrating the necessary tools for the development phase of software (e.g. design, implementation, and debugging) into a single platform.

By providing an integrated environment, it is possible to perform all development using just this product, without the need to use many different tools separately.

Readers

This manual is intended for users who wish to understand the functions of the CubeSuite+ and design software and hardware application systems.

Purpose

This manual is intended to give users an understanding of the functions of the CubeSuite+ to use for reference in developing the hardware or software of systems using these devices.

Organization

This manual can be broadly divided into the following units.

CHAPTER 1    GENERAL
CHAPTER 2    FUNCTIONS (Pin Configurator)
CHAPTER 3    FUNCTIONS (Code Generator)
APPENDIX A   WINDOW REFERENCE
APPENDIX B   OUTPUT FILES
APPENDIX C   API FUNCTIONS
APPENDIX D   INDEX

How to Read This Manual

It is assumed that the readers of this manual have general knowledge of electricity, logic circuits, and microcontrollers.

Conventions

Data significance: Higher digits on the left and lower digits on the right
Active low representation: XXX (overscore over pin or signal name)
Note: Footnote for item marked with Note in the text
Caution: Information requiring particular attention
Remark: Supplementary information
Numeric representation: Decimal ... XXXX
Hexadecimal ... 0xXXXX
## Related Documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Document No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeSuite+</td>
<td>R20UT2133E</td>
</tr>
<tr>
<td>Integrated Development Environment</td>
<td>R20UT2134E</td>
</tr>
<tr>
<td>User's Manual</td>
<td>R20UT2135E</td>
</tr>
<tr>
<td>V850 Design</td>
<td>R20UT2136E</td>
</tr>
<tr>
<td>R8C Design</td>
<td>R20UT2137E</td>
</tr>
<tr>
<td>RL78 Design</td>
<td>R20UT2138E</td>
</tr>
<tr>
<td>78K0R Design</td>
<td>R20UT2139E</td>
</tr>
<tr>
<td>78K0 Design</td>
<td>This manual</td>
</tr>
<tr>
<td>RX Coding</td>
<td>R20UT0767E</td>
</tr>
<tr>
<td>V850 Coding</td>
<td>R20UT0553E</td>
</tr>
<tr>
<td>Coding for CX Compiler</td>
<td>R20UT2139E</td>
</tr>
<tr>
<td>R8C Coding</td>
<td>R20UT0576E</td>
</tr>
<tr>
<td>RL78, 78K0R Coding</td>
<td>R20UT2140E</td>
</tr>
<tr>
<td>78K0 Coding</td>
<td>R20UT2141E</td>
</tr>
<tr>
<td>RX Build</td>
<td>R20UT0768E</td>
</tr>
<tr>
<td>V850 Build</td>
<td>R20UT0557E</td>
</tr>
<tr>
<td>Build for CX Compiler</td>
<td>R20UT2142E</td>
</tr>
<tr>
<td>R8C Build</td>
<td>R20UT0575E</td>
</tr>
<tr>
<td>RL78, 78K0R Build</td>
<td>R20UT2143E</td>
</tr>
<tr>
<td>78K0 Build</td>
<td>R20UT0783E</td>
</tr>
<tr>
<td>RX Debug</td>
<td>R20UT2175E</td>
</tr>
<tr>
<td>V850 Debug</td>
<td>R20UT2144E</td>
</tr>
<tr>
<td>R8C Debug</td>
<td>R20UT0770E</td>
</tr>
<tr>
<td>RL78 Debug</td>
<td>R20UT2145E</td>
</tr>
<tr>
<td>78K0R Debug</td>
<td>R20UT0732E</td>
</tr>
<tr>
<td>78K0 Debug</td>
<td>R20UT0731E</td>
</tr>
<tr>
<td>Analysis</td>
<td>R20UT2146E</td>
</tr>
<tr>
<td>Message</td>
<td>R20UT2147E</td>
</tr>
</tbody>
</table>

**Caution**  The related documents listed above are subject to change without notice. Be sure to use the latest edition of each document when designing.

All trademarks or registered trademarks in this document are the property of their respective owners.
TABLE OF CONTENTS

CHAPTER 1 GENERAL ... 7

1.1 Overview ... 7
1.2 Features ... 7

CHAPTER 2 FUNCTIONS (Pin Configurator) ... 8

2.1 Overview ... 8
2.2 Open Device Pin List Panel ... 10
   2.2.1 Select item ... 11
   2.2.2 Change display order ... 12
   2.2.3 Add column ... 13
   2.2.4 Delete column ... 13
2.3 Open Device Top View Panel ... 14
   2.3.1 Select shape of microcontroller ... 15
   2.3.2 Select color ... 16
   2.3.3 Select popup information ... 17
   2.3.4 Select additional information ... 18
2.4 Enter Information ... 19
2.5 Output Report Files ... 20
   2.5.1 Output device pin list ... 20
   2.5.2 Output device top view ... 21

CHAPTER 3 FUNCTIONS (Code Generator) ... 22

3.1 Overview ... 22
3.2 Open Code Generator Panel ... 23
3.3 Enter Information ... 24
   3.3.1 Input rule ... 24
   3.3.2 Icon indicating incorrect entry ... 25
   3.3.3 Icon indicating pin conflict ... 26
3.4 Confirm Source Code ... 27
3.5 Output Source Code ... 28
   3.5.1 Setting that determines whether or not to generate source code ... 29
   3.5.2 Change file name ... 30
   3.5.3 Change API function name ... 31
   3.5.4 Change output mode ... 32
   3.5.5 Change output destination folder ... 33
3.6 Output Report Files ... 34
   3.6.1 Change output format ... 36
   3.6.2 Change output destination ... 37
APPENDIX A  WINDOW REFERENCE ... 38
   A.1  Description ... 38

APPENDIX B  OUTPUT FILES ... 84
   B.1  Overview ... 84
   B.2  Output File ... 84

APPENDIX C  API FUNCTIONS ... 89
   C.1  Overview ... 89
   C.2  Output Function ... 89
   C.3  Function Reference ... 95
       C.3.1  System ... 97
       C.3.2  Port ... 108
       C.3.3  Interrupt ... 115
       C.3.4  Serial ... 126
       C.3.5  Operational Amplifier ... 166
       C.3.6  Comparator ... 176
       C.3.7  A/D Converter ... 181
       C.3.8  Timer ... 191
       C.3.9  Watchdog Timer ... 222
       C.3.10  Real-time Clock ... 224
       C.3.11  Clock Output ... 258
       C.3.12  LVI ... 264

APPENDIX D  INDEX ... 272
CubeSuite+ is an integrated development environment used to carry out tasks such as design, coding, build and debug for developing application systems.

This chapter gives an overview of the design tool (Pin Configurator/Code Generator).

1.1 Overview

The design tool, which is one of the components provided by CubeSuite+, enables you to output the pin assignment of the microcontroller (device pin list and device top view), and the source code (device driver programs, C source files and header files) necessary to control the peripheral functions provided by the microcontroller (clock generator, port functions, etc.) by configuring various information using the GUI.

1.2 Features

The design tool (Pin Configurator/Code Generator) has the following features.

- Code generating function
  The Code Generator can output not only device driver programs in accordance with the information configured using the GUI, but also a build environment such as sample programs containing main functions and link directive files.

- Reporting function
  You can output configured information using Pin Configurator/Code Generator as files in various formats for use as design documents.

- Renaming function
  The user can change default names assigned to the files output by Code Generator and the API functions contained in the source code.
CHAPTER 2 FUNCTIONS (Pin Configurator)

This chapter describes the key functions provided by the design tool (Pin Configurator) along with operation procedures.

2.1 Overview

The Pin Configurator is used to output report files such as a device pin list and a device top view by entering pin assignment information of the microcontroller.

The following sections describe the operation procedures for Pin Configurator.

(1) **Start CubeSuite+**

Launch CubeSuite+ from the [Start] menu of Windows.


(2) **Create/Open project**

Create a new project (that defines a kind of project, microcontroller to be used, build tools to be used, etc.) or load an existing project.


(3) **Open Device Pin List Panel**

Open the Device Pin List panel, where you enter information on the pins of the microcontroller.

(a) **Select item**

Allows you to select items displayed in the device pin list.

(b) **Change display order**

Allows you to change the order in which items are displayed in the device pin list.

(c) **Add column**

Allows you to add columns to the device pin list.

(d) **Delete column**

Allows you to delete columns from the device pin list.

(4) **Open Device Top View Panel**

Open the Device Top View panel, where you can confirm the information entered for the pins.

(a) **Select shape of microcontroller**

Allows you to select the shape of the microcontroller displayed in the Device Top View panel.

(b) **Select color**

Allows you to select colors used to distinguish the type of pins (power pins, special pins, used pins, etc.) whose information is displayed in the Device Top View panel.
(c) **Select popup information**  
Allows you to select the type of information that popups when you move the mouse cursor over each pin in the Device Top View panel.

(d) **Select additional information**  
Select the type of information to display in Pin area of the Device Top View panel.

(5) **Enter Information**  
Enter information on the pins of the microcontroller in the Device Pin List panel.

(6) **Output Report Files**  
Output report files (files containing configured information using Pin Configurator: device pin list and device top view) to the specified folder.

(a) **Output device pin list**  
Output a device pin list.

(b) **Output device top view**  
Output a device top view.

(7) **Save project**  
Save a project.

**Remark**  
2.2 Open Device Pin List Panel

Open the Device Pin List panel, where you enter information on the pins of the microcontroller.

To open the Device Pin List panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List] in the Project Tree panel.

Figure 2-1. Open Device Pin List Panel

Remarks 1. If an unsupported microcontroller is defined in the project for Pin Configurator, then “[Pin Configurator (Design Tool)] node” will hide under [Project name (Project)] in the Project Tree panel.

2. The Device Pin List panel consists of three tabs. Selecting one of the tabs changes the order in which “information on each pin of the microcontroller” is displayed.
   - [Pin Number] tab
     Information on each pin of the microcontroller is displayed in the order of pin number.
   - [Macro] tab
     Information on each pin of the microcontroller is displayed in the order it was grouped into peripheral functions.
   - [External Peripheral] tab
     Information about the pins connected to external peripherals is displayed in order grouped at the external-peripheral component level.
The Pin Configurator is used to select items to be displayed in the device pin list using the button in the upper left corner of the device pin list.

To select the item to be displayed, use the Column Chooser dialog box that opens by pressing the button in the upper left corner of the device pin list.

**Figure 2-2. Select Item**

![Column Chooser dialog box](image.png)

**Remark** To select the item to be displayed, check the check box that corresponds to the item.

**Table 2-1. Select Item**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checked</td>
<td>Displays the selected item in the device pin list.</td>
</tr>
<tr>
<td>Not checked</td>
<td>Hides the selected item in the device pin list.</td>
</tr>
</tbody>
</table>
2.2.2 Change display order

In Pin Configurator, you can change the display order of columns in the device pin list (move columns) by dragging and dropping columns.

Figure 2-3. Change Display Order

Remark To change the display order, click the button in the upper left of the device pin list. The Column Chooser dialog box opens. Drag an item displayed in the dialog's select Items to display area, and drop it to the desired destination in the device pin list. This will change the display order.
2.2.3 Add column

The Pin Configurator is used to add the "user's own column" to the device pin list using the [New Column...] button in the Column Chooser dialog box that opens by pressing the button in the upper left corner of the device pin list.

To add a column, use the New Column dialog box that opens by pressing the [New Column...] button in the Column Chooser dialog box.

![Figure 2-4. Add Column](image)

**Remark** On the device pin list, adding columns to the first level of [Macro] tab, [External Peripheral] tab is restricted.

2.2.4 Delete column

The Pin Configurator is used to delete the "user's own column" from the device pin list using the [Delete Column] button in the Column Chooser dialog box that opens by pressing the button in the upper left corner of the device pin list.

To delete a column, select the column you want to delete in the displayed item selection area of the Column Chooser dialog box, and press the [Delete Column] button.

![Figure 2-5. Delete Column](image)

**Remark** You can only delete the column which you added using the New Column dialog box.
2.3 Open Device Top View Panel

Open the Device Top View panel, where you can confirm the information entered for the pins of the microcontroller. To open the Device Top View panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Top View] in the Project Tree panel.

Remark In the Property panel, on the [Pin Configurator Settings] tab, if "BGA" is selected for the Package type, then Device Top View panel cannot be opened.
2.3.1 Select shape of microcontroller

Select the shape of the microcontroller displayed in the Device Top View panel which is opened as described in "2.3 Open Device Top View Panel".

To select the shape of the microcontroller, click [Pin Configurator Settings] tab >> [Package type] in the Property panel and select the desired shape.

Remark Selection of the shape of the microcontroller is made using the order name (such as GC and GF).
2.3.2 Select color

Select the colors used to distinguish the type of pins (power pins, special pins, unused pins, etc.) whose information is displayed in the **Device Top View panel** which is opened as described in "2.3 Open Device Top View Panel".

To select the color to be displayed, select the desired color in the color palette that opens by clicking [Device Top View Settings] tab >> [Color] in the **Property panel**.

![Select Color](image)

**Remark** Select the colors to be displayed for the following eight types of items.

<table>
<thead>
<tr>
<th>Item</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power pins</td>
<td>Selects the display color for power pins (pins whose use is limited to power).</td>
</tr>
<tr>
<td>Special pins</td>
<td>Selects the display color for special pins (pins with specified uses).</td>
</tr>
<tr>
<td>Unused pins</td>
<td>Selects the display color for unused pins (dual-use pins with no use set in the <strong>Device Pin List panel</strong>).</td>
</tr>
<tr>
<td>Used pins</td>
<td>Selects the display color for used pins (dual-use pins with a use set in the <strong>Device Pin List panel</strong>).</td>
</tr>
<tr>
<td>Device</td>
<td>Selects the display color of the microcontroller.</td>
</tr>
<tr>
<td>Highlight color for a selected pin</td>
<td>Selects the background color of a pin selected in the <strong>Device Pin List panel</strong>, on the [Pin Number] tab.</td>
</tr>
<tr>
<td>Highlight color for macro pins</td>
<td>Selects the background color of pins selected in the <strong>Device Pin List panel</strong>, on the [Macro] tab.</td>
</tr>
<tr>
<td>Highlight color for external peripheral pins</td>
<td>Selects the background color of pins selected in the <strong>Device Pin List panel</strong>, on the [External Peripheral] tab.</td>
</tr>
</tbody>
</table>
2.3.3 Select popup information

Select the type of information that popups when you move the mouse cursor over each pin in the Device Top View panel which is opened as described in "2.3 Open Device Top View Panel".

To select the popup information, click [Device Top View Settings] tab >> [Tool tip] in the Property panel and select the desired type of information.

**Remark** Pop-up information is selected from the following four types.

<table>
<thead>
<tr>
<th>Popup Information</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display all</td>
<td>Displays the &quot;Description&quot;, &quot;Recommend Connection for Unused&quot;, and &quot;Attention&quot; strings for the device pin list.</td>
</tr>
<tr>
<td>Description / recommended connection for unused pin only</td>
<td>Displays the &quot;Description&quot;, and &quot;Recommend Connection for Unused&quot; strings for the device pin list.</td>
</tr>
<tr>
<td>Attention only</td>
<td>Displays the &quot;Attention&quot; string for the device pin list.</td>
</tr>
<tr>
<td>Not display</td>
<td>Hides tooltips when the mouse cursor hovers over a pin.</td>
</tr>
</tbody>
</table>
2.3.4 Select additional information

Select the type of information to display in Pin area, in the Device Top View panel opened in "2.3 Open Device Top View Panel".

Note that additional information is selected from the Property panel, on the [Device Top View Settings] tab, by selecting the corresponding information under [Pin Name Display].

![Figure 2-10. Select Additional Information](image)

**Remarks 1.** Select one of the following two types for Define name (whether to display the "Define Name" string of the Device Pin List in appended format).

<table>
<thead>
<tr>
<th>Display</th>
<th>Displays the &quot;Define Name&quot; string of the device pin list in appended format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not display</td>
<td>Hides the &quot;Define Name&quot; string of the device pin list.</td>
</tr>
</tbody>
</table>

2. Select one of the following two types for Pin function (whether to display it whether or not a function is selected for "Function" on the Device Pin List).

<table>
<thead>
<tr>
<th>Display all</th>
<th>Displays functions selected via the device pin list's &quot;Function&quot; feature in parentheses.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected function only</td>
<td>Only display functions selected via the device pin list's &quot;Function&quot; feature in the device top view.</td>
</tr>
</tbody>
</table>
2.4 Enter Information

Enter information on the pins of the microcontroller in the Device Pin List panel which is opened as described in "2.2 Open Device Pin List Panel".

Remarks 1. You cannot add information in the "Pin Number" column, "Pin Name" column, "Description" column, "Recommend Connection for Unused" column and "Attention" column because they contain fixed information.

2. If the "Free" in the "Function" column is changed to a specific pin name, color of the corresponding pin in the Device Top View panel changes from the "color representing the unused pins" to the "color representing the used pins" selected by clicking [Device Top View Settings] tab >> [Color] in the Property panel.

Figure 2-11. Change in Displayed Color
2.5 Output Report Files

Output report files (files containing information configured using Pin Configurator: device pin list and device top view) to the specified folder.

2.5.1 Output device pin list

Select [File] menu >> [Save Pin List As...] to output a report file (a file containing information configured using Pin Configurator: device pin list).

The destination folder for the device pin list is specified in the Save As dialog box which opens by selecting [File] menu >> [Save Pin List As ...].

Remarks 1. If a device pin list has been already output, that list will be overwritten by selecting [File] menu >> [Save Pin List].

2. The output format for the device pin list is limited to Microsoft Office Excel Book.
2.5.2 Output device top view

Select [File] menu >> [Save Top View As...] to output a report file (a file containing information configured using Pin Configurator: device top view).

The destination folder for the device top view is specified in the Save As dialog box which opens by selecting [File] menu >> [Save Top View As ...].

**Figure 2-13. Output Device Top View**

![Save As dialog box](image)

**Remark** If a device top view has been already output, that view will be overwritten by selecting [File] menu >> [Save Top View].
CHAPTER 3 FUNCTIONS (Code Generator)

This chapter describes the key functions provided by the design tool (Code Generator) along with operation procedures.

3.1 Overview

The Code Generator outputs source code (device driver programs) based on information selected/entered on CubeSuite+ panels that is needed to control peripheral functions provided by the microcontroller (clock generator, port functions, etc.).

The following sections describe the operation procedures for Code Generator.

(1) Start CubeSuite+
Launch CubeSuite+ from the [Start] menu of Windows.


(2) Create/Open project
Create a new project (that defines a kind of project, microcontroller to be used, build tools to be used, etc.) or load an existing project.


(3) Open Code Generator Panel
Open the Code Generator panel used to configure the information necessary to control the peripheral functions (clock generator, port functions, etc.).

(4) Enter Information
Configure the information necessary to control the peripheral functions in the Code Generator panel.

(5) Confirm Source Code
Confirm the source code (device driver program) that reflects the information configured in the Code Generator panel.

(6) Output Source Code
Output the source code (device driver program) to the specified folder.

(7) Output Report Files
Output report files (a file containing information configured using Code Generator and a file containing information regarding the source code) to the specified folder.

(8) Save project
Save a project.

3.2 Open Code Generator Panel

Open the Code Generator panel to configure the information necessary to control the peripheral functions (clock generator, port functions, etc.).

To open the Code Generator panel, double-click [Project name (Project)] >> [Code Generator (Design Tool)] >> Peripheral function node "[System], [Port], etc." in the Project Tree panel.

**Figure 3-1. Open Code Generator Panel**

![Code Generator Panel](image)

**Remark** If an unsupported microcontroller is defined in the project for Code Generator, then "[Code Generator (Design Tool)] node" will hide under [Project name (Project)] in the Project Tree panel.
### 3.3 Enter Information

Configure the information necessary to control the peripheral functions in the information setting area of the Code Generator panel which is opened as described in "3.2 Open Code Generator Panel".

**Remark** When controlling multiple peripheral functions, repeat the procedures described in "3.2 Open Code Generator Panel" through "3.3 Enter Information".

#### 3.3.1 Input rule

Following is the rules for input to the Code Generator panel.

1. **Character set**
   - Character sets that are allowed to input are as follows.

   **Table 3-1. List of Character Set**

<table>
<thead>
<tr>
<th>Character Set</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>1-byte alphabet, number, symbol</td>
</tr>
<tr>
<td>Shift-JIS</td>
<td>2-byte alphabet, number, symbol, Hiragana, Katakana, Kanji and 1-byte Katakana</td>
</tr>
<tr>
<td>EUC-JP</td>
<td>2-byte alphabet, number, symbol, Hiragana, Katakana, Kanji and 1-byte Katakana</td>
</tr>
<tr>
<td>UTF-8</td>
<td>2-byte alphabet, number, symbol, Hiragana, Katakana, Kanji (include Chinese character) and 1-byte Katakana</td>
</tr>
</tbody>
</table>

2. **Number**
   - Notations allowed when entering numbers are as follows.

   **Table 3-2. List of Notation**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal number</td>
<td>A numeric value that starts with a number between 1 and 9 and followed by numbers between 0 and 9, and the numeric value 0</td>
</tr>
<tr>
<td>Hex number</td>
<td>A numeric value that starts with 0x and followed by a combination of numbers from 0 to 9 and characters from A to F (characters are not case sensitive)</td>
</tr>
</tbody>
</table>
3.3.2 Icon indicating incorrect entry

When performing code generation, if you enter an invalid string in the Code Generator panel, or a required input is missing, then a icon displays next to the incorrect input, and the text is displayed in red to warn that there is a problem with the input.

Remark If the mouse cursor is moved over the icon, information regarding the string that should be entered (tips for correcting the entry) popups.

Figure 3-2. Icon Indicating Incorrect Entry
3.3.3 Icon indicating pin conflict

If a conflict occurs between the pins while setting various peripheral functions in the Code Generator panel, the icon is displayed at the location where the conflict occurs to warn the user of a conflict between the pins.

**Remark**  If the mouse cursor is moved over the icon, information regarding the conflict between the pins (tips for avoiding the conflict) popups.

**Figure 3-3. Icon Indicating Pin Conflict**
3.4 Confirm Source Code

Confirm the source code (device driver program) that reflects the information configured as described in "3.3 Enter Information".

To confirm the source code, use the Code Generator Preview panel that opens by selecting [View] menu >> [Code Generator Preview].

Figure 3-4. Confirm Source Code

Remarks 1. You can change the source code to be displayed by selecting the source file name or API function name in the Code Generator Preview panel.

2. The following table displays the meaning of the color of the source code text displayed in the Code Generator Preview panel.

Table 3-3. Color of Source Code

<table>
<thead>
<tr>
<th>Color</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Comment</td>
</tr>
<tr>
<td>Blue</td>
<td>Reserved word for C compiler</td>
</tr>
<tr>
<td>Red</td>
<td>Numeric value</td>
</tr>
<tr>
<td>Black</td>
<td>Code section</td>
</tr>
<tr>
<td>Gray</td>
<td>File name</td>
</tr>
</tbody>
</table>

3. You cannot edit the source code within the Code Generator Preview panel.

4. For some of the API functions (such as API functions for serial array units), values such as the SFR register value are calculated and finalized when the source code is generated (when the Generate Code button on the Code Generator panel is pressed). For this reason, the source code displayed in the Code Generator Preview panel may not be the same as that would actually be generated.
3.5 Output Source Code

Output the source code (device driver program) by pressing the button on the Code Generator panel.

The destination folder for the source code is specified by clicking [Generation] tab >> [Output folder] in the Property panel.

**Figure 3-5. Output Source Code**

![Figure 3-5. Output Source Code](image)

**Remark** In order to both output source files and add them to the project (display the corresponding source file names in the Project Tree panel) when you click the button, you must open the Property panel, and under [Generation] tab >> [Register files], specify "Output files to project".

**Figure 3-6. Configure Whether to Register**

![Figure 3-6. Configure Whether to Register](image)
3.5.1 Setting that determines whether or not to generate source code

You can set the type of output API functions (all API functions or only initialization API functions) by selecting [Output all API function according to the setting/Output only initialization API function] from [Generation] tab >> [Output control of API function] in the Property panel.

![Figure 3-7. Setting That Determines Type of API Functions](image)

You can set whether or not to generate the corresponding source code on a per-API function basis by selecting [Generate code/Not generate code] from the context menu displayed by right clicking the API function name in the Code Generator Preview panel.

![Figure 3-8. Setting That Determines Whether or Not to Generate Source Code](image)
Remark You can confirm the current setting for the generation of source code by checking the type of icon in the Code Generator Preview panel.

Table 3-4. Setting That Determines Whether or Not to Generate Source Code

<table>
<thead>
<tr>
<th>Type of Icon</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td>Source code for the currently selected API function is generated. If this icon is displayed next to the API function, the corresponding source code must be generated (it is impossible to change the icon to <img src="image2.png" alt="Icon" />).</td>
</tr>
<tr>
<td><img src="image3.png" alt="Icon" /></td>
<td>Source code for the currently selected API function is generated.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Icon" /></td>
<td>Source code for the currently selected API function is not generated.</td>
</tr>
</tbody>
</table>

3.5.2 Change file name

The Code Generator is used to change the file name by selecting [Rename] from the context menu displayed by right clicking the file name in the Code Generator Preview panel.

Figure 3-9. Change File Name

Remark To restore the default file name defined by Code Generator, select [Default] from the context menu.
3.5.3 Change API function name

The Code Generator is used to change the name of the API function by selecting [Rename] from the context menu displayed by right clicking the API function name in the Code Generator Preview panel.

Figure 3-10. Change API Function Name

Remark To restore the default name of the API function defined by Code Generator, select [Default] from the context menu.
3.5.4 Change output mode

The Code Generator is used to change the output mode (Do nothing if file exists, Merge file, Overwrite file) for the source code by selecting [Generation] tab >> [Generate file] in the Property panel.

Figure 3-11. Change Output Mode

Remark

The output mode is selected from the following three types.

Table 3-5. Output Mode of Source Code

<table>
<thead>
<tr>
<th>Output Mode</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do nothing if file exists</td>
<td>If a file with the same name exists, a new file will not be output.</td>
</tr>
<tr>
<td>Merge file</td>
<td>If a file with the same name exists, a new file is merged with the existing file.</td>
</tr>
<tr>
<td></td>
<td>Only the section between &quot;/* Start user code ... Do not edit comment generated here <em>/&quot; and &quot;/</em> End user code. Do not edit comment generated here */&quot; will be merged.</td>
</tr>
<tr>
<td>Overwrite file</td>
<td>If a file with the same name exists, the existing file is overwritten by a new file.</td>
</tr>
</tbody>
</table>
3.5.5 Change output destination folder

The Code Generator is used to change the output destination folder for the source code by selecting [Generation] tab >> [Output folder] in the Property panel.

To change the output destination, use the Browse For Folder dialog box which opens by pressing the [...] button in the [Output folder].

Figure 3-12. Change Output Destination Folder
3.6 Output Report Files

Output report files (a file containing information configured using Code Generator and a file containing information regarding the source code) by first activating the Code Generator panel or Code Generator Preview panel, then selecting [File] menu >> [Save Code Generator Report].


Remarks 1. You can only use "macro" or "function" as a name of the report file.

Table 3-6. Output Report Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>macro</td>
<td>A file that contains the information configured using Code Generator</td>
</tr>
<tr>
<td>function</td>
<td>A file that contains the information regarding the source code</td>
</tr>
</tbody>
</table>

2. The output mode for the report file is fixed to "Overwrite file".

Figure 3-13. Output Example of Report File "macro"
Figure 3-14. Output Example of Report File "function"

<table>
<thead>
<tr>
<th>Module</th>
<th>File</th>
<th>Macro</th>
<th>Function</th>
<th>Default</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG_main.c</td>
<td>void main(void)</td>
<td>CG_main.c</td>
<td>main</td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R_MAIN_UserInit</td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td>CG_systeminit.c</td>
<td>void R_MAIN_UserInit(void)</td>
<td>CG_systeminit.c</td>
<td>Used</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>void systeminit(void)</td>
<td>systeminit</td>
<td></td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>void hwinit(void)</td>
<td>hwinit</td>
<td></td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td>CG_macrodriver.h</td>
<td></td>
<td>CG_macrodriver.h</td>
<td>Used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG_userdefine.h</td>
<td></td>
<td>CG_userdefine.h</td>
<td>Used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG_Ik.dr</td>
<td></td>
<td>CG_Ik.dr</td>
<td>Used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG_option.asm</td>
<td></td>
<td>CG_option.asm</td>
<td>Used</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG_system.c</td>
<td>void CLOCK_Init(void)</td>
<td>CG_system.c</td>
<td>CLOCK_Init</td>
<td>Used</td>
</tr>
</tbody>
</table>
3.6.1 Change output format

The Code Generator is used to change the output format (HTML file or CSV file) of the report file by selecting [Generation] tab >> [Report type] in the Property panel.

Figure 3-15. Change Output Format

### Table 3-7. Output Mode of Source Code

<table>
<thead>
<tr>
<th>Report Type</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML file</td>
<td>Outputs a report file in HTML format.</td>
</tr>
<tr>
<td>CSV file</td>
<td>Outputs a report file in CSV format.</td>
</tr>
</tbody>
</table>
3.6.2 Change output destination

The Code Generator is used to change the output destination folder for the report file by selecting [Generation] tab >> [Output folder] in the Property panel.

To change the output destination, use the Browse For Folder dialog box which opens by pressing the [...] button in the [Output folder].

Figure 3-16. Change Output Destination
APPENDIX A  WINDOW REFERENCE

This appendix explains in detail the functions of the windows, panels and dialog boxes of the design tool.

A.1 Description

The design tool has the following windows, panels and dialog boxes.

Table A-1. Window/Panel/Dialog Box List

<table>
<thead>
<tr>
<th>Window/Panel/Dialog Box Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main window</td>
<td>This is the first window to open when CubeSuite+ is launched. This window is used to operate various components (design tool, build tool, etc.) provided by CubeSuite+.</td>
</tr>
<tr>
<td>Project Tree panel</td>
<td>This panel displays the components of the project (microcontroller, design tool, build tool, etc.) in a tree structure.</td>
</tr>
<tr>
<td>Property panel</td>
<td>This panel allows you to view the information and change the setting for the node selected in the Project Tree panel, the peripheral function button pressed in the Code Generator panel or the file selected in the Code Generator Preview panel.</td>
</tr>
<tr>
<td>Device Pin List panel</td>
<td>This panel allows you to enter information on each pin of the microcontroller.</td>
</tr>
<tr>
<td>Device Top View panel</td>
<td>This panel displays the information entered in the Device Pin List panel.</td>
</tr>
<tr>
<td>Code Generator panel</td>
<td>This panel allows you to configure the information necessary to control the peripheral functions provided by the microcontroller.</td>
</tr>
<tr>
<td>Code Generator Preview panel</td>
<td>This panel allows you to confirm or configure on a per-API function basis the setting that determines whether or not the source code (device driver program) is generated when the &quot;Generate Code&quot; button is pressed in the Code Generator panel. It also allows you to confirm the source code that reflects the information configured in the Code Generator panel.</td>
</tr>
<tr>
<td>Output panel</td>
<td>This panel displays operation logs for various components (design tool, build tool, etc.) provided by CubeSuite+.</td>
</tr>
<tr>
<td>Column Chooser dialog box</td>
<td>This dialog box allows you to choose whether or not to display the item listed in this dialog box in the device pin list, and add columns to or delete columns from the device pin list.</td>
</tr>
<tr>
<td>New Column dialog box</td>
<td>This dialog box allows you to add your own column to the device pin list.</td>
</tr>
<tr>
<td>Browse For Folder dialog box</td>
<td>This dialog box allows you to specify the output destination for files (source code, report file, etc.).</td>
</tr>
<tr>
<td>Save As dialog box</td>
<td>This dialog box allows you to name and save a file (such as a report file).</td>
</tr>
</tbody>
</table>
Main window

This is the first window to open when CubeSuite+ is launched. This window is used to operate various components (design tool, build tool, etc.) provided by CubeSuite+.

Figure A-1. Main Window

The following items are explained here.
- [How to open]
- [Description of each area]

[How to open]
- From the [start] menu, select [All Programs] >> [Renesas Electronics CubeSuite+] >> [CubeSuite+].

[Description of each area]

(1) Menu bar
This area consists of the following menu items.
### (a) [File] menu

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Pin List</td>
<td>Device Pin List panel-dedicated item Saves a report file (a file containing information configured using Pin Configurator: device pin list) overwriting the existing file.</td>
</tr>
<tr>
<td>Save Pin List As...</td>
<td>Device Pin List panel-dedicated item Opens the Save As dialog box for naming and saving a report file (a file containing information configured using Pin Configurator: device pin list).</td>
</tr>
<tr>
<td>Save Top View</td>
<td>Device Top View panel-dedicated item Saves a report file (a file containing information configured using Pin Configurator: device top view) overwriting the existing file.</td>
</tr>
<tr>
<td>Save Top View As...</td>
<td>Device Top View panel-dedicated item Opens the Save As dialog box for naming and saving a report file (a file containing information configured using Pin Configurator: device top view).</td>
</tr>
<tr>
<td>Save Code Generator Report</td>
<td>Code Generator panel/Code Generator Preview panel-dedicated item Outputs report files (a file containing information configured using Code Generator and a file containing information regarding the source code).</td>
</tr>
<tr>
<td></td>
<td>- The output format for the report file (either HTML or CSV) is selected by clicking [Generation] tab &gt;&gt; [Report type] in the Property panel.</td>
</tr>
<tr>
<td>Save Output-Tab Name</td>
<td>Output panel-dedicated item Saves the message corresponding to the specified tab overwriting the existing file.</td>
</tr>
<tr>
<td>Save Output-Tab Name As...</td>
<td>Output panel-dedicated item Opens the Save As dialog box for naming and saving the message corresponding to the specified tab.</td>
</tr>
</tbody>
</table>

### (b) [Edit] menu

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Property panel-dedicated item Cancels the effect of an edit operation to restore the previous state.</td>
</tr>
<tr>
<td>Cut</td>
<td>Property panel-dedicated item Sends the character string or lines selected with range selection to the clipboard and deletes them.</td>
</tr>
<tr>
<td>Copy</td>
<td>Property panel/Output panel-dedicated item Sends the character string or lines selected with range selection to the clipboard.</td>
</tr>
<tr>
<td>Paste</td>
<td>Property panel-dedicated item Inserts the contents of the clipboard at the caret position.</td>
</tr>
<tr>
<td>Delete</td>
<td>Property panel-dedicated item Deletes the character string or the lines selected with the range selection.</td>
</tr>
<tr>
<td>Select All</td>
<td>Property panel/Output panel-dedicated item Selects all the strings displayed in the item being edited or all the strings displayed in the Message area.</td>
</tr>
</tbody>
</table>
### (c) [Help] menu

<table>
<thead>
<tr>
<th>Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search...</td>
<td>Device Pin List panel/Code Generator Preview panel/Output panel-dedicated item</td>
</tr>
<tr>
<td>Replace...</td>
<td>Output panel-dedicated item</td>
</tr>
</tbody>
</table>

- Opens the Search and Replace dialog box for searching strings with the [Quick Search] tab selected.
- Opens the Search and Replace dialog box for replacing strings with the [Whole Replace] tab selected.

### (2) Panel display area

This area consists of multiple panels, each dedicated to a different purpose. See the following sections for details on this area.

- Project Tree panel
- Property panel
- Device Pin List panel
- Device Top View panel
- Code Generator panel
- Code Generator Preview panel
- Output panel
This panel displays components of the project (microcontroller, design tool, build tool, etc.) in a tree structure.

**Figure A-2. Project Tree Panel**

The following items are explained here.
- [How to open]
- [Description of each area]
- [Help] menu (Project Tree panel-dedicated items]
- [Context menu]

**[How to open]**
- From the [View] menu, select [Project Tree].

**[Description of each area]**

1. **Project tree area**
   This area displays components of the project (microcontroller, design tool, build tool, etc.) in a tree structure.

   (a) **Pin Configurator (Design Tool)**
   This node consists of the following pin nodes.
(b) **Code Generator (Design Tool)**
This node consists of the following peripheral function nodes.
When there is peripheral function target microcontroller is not supporting, peripheral function button is not disokayed.

<table>
<thead>
<tr>
<th>Device Pin List</th>
<th>Opens the [Device Pin List] panel for entering information on the pins of the microcontroller.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Top View</td>
<td>Opens the [Device Top View] panel that displays the information entered in the Device Pin List panel.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Opens the [System] for configuring the information necessary to control the functions of clock generator, on-chip debug function and etc. provided by the microcontroller.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Opens the [Port] for configuring the information necessary to control the port functions provided by the microcontroller.</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Opens the [Interrupt] for configuring the information necessary to control the interrupt functions and the key interrupt function provided by the microcontroller.</td>
</tr>
<tr>
<td>Serial</td>
<td>Opens the [Serial] for configuring the information necessary to control the functions of serial array unit and functions of serial interface provided by the microcontroller.</td>
</tr>
<tr>
<td>Operational Amplifier</td>
<td>Opens the [Operational Amplifier] for configuring the information necessary to control the functions of comparator/programmable gain amplifier provided by the microcontroller.</td>
</tr>
<tr>
<td>Comparator</td>
<td>Opens the [Comparator] for configuring the information necessary to control the functions of comparator provided by the microcontroller.</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Opens the [A/D Converter] for configuring the information necessary to control the function of A/D converter provided by the microcontroller.</td>
</tr>
<tr>
<td>Timer</td>
<td>Opens the [Timer] for configuring the information necessary to control the functions of timer array unit provided by the microcontroller.</td>
</tr>
<tr>
<td>Watchdog Timer</td>
<td>Opens the [Watchdog Timer] for configuring the information necessary to control the functions of watchdog timer provided by the microcontroller.</td>
</tr>
<tr>
<td>Real-time Clock</td>
<td>Opens the [Real-time Clock] for configuring the information necessary to control the functions of real-time counter provided by the microcontroller.</td>
</tr>
<tr>
<td>Clock Output</td>
<td>Opens the [Clock Output] for configuring the information necessary to control the functions of clock output controller provided by the microcontroller.</td>
</tr>
<tr>
<td>LVI</td>
<td>Opens the [LVI] for configuring the information necessary to control the functions of low-voltage detector provided by the microcontroller.</td>
</tr>
</tbody>
</table>

(c) **Icons**
The table below displays the meaning of the icon displayed to the left of the string representing the peripheral function node.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Operation in the corresponding Code Generator panel has been carried out.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operation in the corresponding Code Generator panel has not been carried out.</td>
</tr>
</tbody>
</table>
The problem occurs on the settings became the manipulation to the other peripheral function node influences.

### [[Help] menu (Project Tree panel-dedicated items)]

| Open Help for Project Tree Panel | Displays the help of this panel. |

### [Context menu]

The following context menu items are displayed by right clicking the mouse.

<table>
<thead>
<tr>
<th>Return to Reset Value</th>
<th>Restores the information for the selected peripheral function node to its default state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Opens the Property panel containing the information for the selected node ([Pin Configurator (Design Tool)], [Device Pin List], [Device Top View], [Code Generator (Design Tool)], peripheral function node “[System], [Port], etc.”).</td>
</tr>
</tbody>
</table>
This panel allows you to view the information on and change the setting for the node selected in the Project Tree panel, the peripheral function button pressed in the Code Generator panel or the file selected in the Code Generator Preview panel.

**Figure A-3. Property Panel (Selected [Pin Configurator (Design Tool)])**

The following items are explained here.
- [How to open]
- [Description of each area]
- [[Edit] menu (Property panel-dedicated items)]
- [Context menu]

**How to open**

- On the Project Tree panel, select a node ([Pin Configurator (Design Tool)], [Device Pin List], [Device Top View], [Code Generator (Design Tool)], peripheral function node "[System], [Port], etc."), and then select [Property] from the [View] menu.
- On the Project Tree panel, select a node ([Pin Configurator (Design Tool)], [Device Pin List], [Device Top View], [Code Generator (Design Tool)], peripheral function node "[System], [Port], etc."), and then select [Property] from the context menu.
- On the Code Generator Preview panel, select a file, and then select [Property] from the [View] menu.
- On the Code Generator Preview panel, select a file, and then select [Property] from the context menu.

**Remarks 1.** If this panel is already open, selecting a different node ([Pin Configurator (Design Tool)], [Device Pin List], [Device Top View], [Code Generator (Design Tool)] or peripheral function node "[System], [Port], etc." in the Project Tree panel changes the content displayed in the Detail information display/change area and explanation area accordingly.

**Remarks 2.** If this panel is already open, pressing a different peripheral function button "", "", etc." in the Code Generator panel changes the content displayed in the Detail information display/change area and explanation area accordingly.

**Remarks 3.** If this panel is already open, selecting a different file in the Code Generator Preview panel changes the content displayed in the Detail information display/change area and explanation area accordingly.
[Description of each area]

(1) **Detail information display/change area**

This area allows you to view the information on and change the setting for the node ([Pin Configurator (Design Tool)], [Device Pin List], [Device Top View], [Code Generator (Design Tool)] or peripheral function node “[System], [Port], etc.”) selected in the Project Tree panel, the peripheral function button “ , ”, etc.” pressed in the Code Generator panel, or the file selected in the Code Generator Preview panel.

The content displayed in this area differs depending on the node selected in the Project Tree panel, the peripheral function button pressed in the Code Generator panel or the file selected in the Code Generator Preview panel.

The following table displays the meaning of and displayed to the left of each category.

<table>
<thead>
<tr>
<th></th>
<th>Indicates that the items within the category are displayed as a “collapsed view”.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that the items within the category are displayed as an “expanded view”.</td>
</tr>
</tbody>
</table>

**Remark** To switch between and , click this mark or double-click the category name.

(2) **Tab selection area**

Categories for the display of the detailed information are changed when each tab is selected.

In this panel, following tabs are contained (see the section explaining each tab for details on the display/setting on the tab).

- [Pin Configurator Settings] tab
- [Device Pin List Information] tab
- [Device Top View Settings] tab
- [Generation] tab
- [Macro Setting] tab
- [File Setting] tab

**[[Edit] menu (Property panel-dedicated items)]**

<table>
<thead>
<tr>
<th></th>
<th>Indicates that the items within the category are displayed as a “collapsed view”.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that the items within the category are displayed as an “expanded view”.</td>
</tr>
</tbody>
</table>

**[Context menu]**

The following context menu items are displayed by right clicking the mouse.

(1) **While the item is being edited**

<table>
<thead>
<tr>
<th></th>
<th>Indicates that the items within the category are displayed as a “collapsed view”.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that the items within the category are displayed as an “expanded view”.</td>
</tr>
</tbody>
</table>
(2) **While the item is not being edited**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Sends the character string or lines selected with range selection to the clipboard.</td>
</tr>
<tr>
<td>Paste</td>
<td>Inserts the contents of the clipboard at the caret position.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the character string or the lines selected with the range selection.</td>
</tr>
<tr>
<td>Select All</td>
<td>Selects all strings displayed in the item being edited.</td>
</tr>
<tr>
<td>Property Reset to Default</td>
<td>Restores the selected item to its default state.</td>
</tr>
<tr>
<td>Property Reset All to Default</td>
<td>Restores all items to their default state.</td>
</tr>
</tbody>
</table>
[Pin Configurator Settings] tab

This tab displays information (Product Information and Package) on the [Pin Configurator (Design Tool)] selected in the Project Tree panel.

**Figure A-4.  [Pin Configurator Settings] Tab**

The following items are explained here.
- [How to open]
- [Description of each area]

**[How to open]**

- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)], and then select [Property] from the [View] menu.
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)], and then select [Property] from the context menu.

**Remark**  If this panel is already open, selecting a different [Pin Configurator (Design Tool)] in the Project Tree panel changes the content displayed accordingly.

**[Description of each area]**

(1) **[Product Information] category**

This area displays product information (Version and Release date) on Pin Configurator.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Displays the version of Pin Configurator (Pin Configurator Plug-in).</td>
</tr>
<tr>
<td>Release date</td>
<td>Displays the release date of Pin Configurator (Pin Configurator Plug-in).</td>
</tr>
</tbody>
</table>

(2) **[Package] category**

Change the shape (Package type) and settings of the microcontroller to display as the device top view in the Device Top View panel.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package type</td>
<td>Selects the shape of the microcontroller displayed in the device top view.</td>
</tr>
</tbody>
</table>
[Device Pin List Information] tab

This tab displays information (Product Information) on the [Device Pin List] selected in the Project Tree panel.

Figure A-5. [Device Pin List Information] Tab

The following items are explained here.
- [How to open]
- [Description of each area]

[How to open]
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List], and then select [Property] from the [View] menu.
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List], and then select [Property] from the context menu.

Remark  If this panel is already open, selecting a different [Device Pin List] in the Project Tree panel changes the content displayed accordingly.

[Description of each area]

(1) [Product Information] category
This area displays product information (Version and Release date) on Pin Configurator.

<table>
<thead>
<tr>
<th>Version</th>
<th>Displays the version of Pin Configurator (Pin Configurator Plug-in).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release date</td>
<td>Displays the release date of Pin Configurator (Pin Configurator Plug-in).</td>
</tr>
</tbody>
</table>
**[Device Top View Settings] tab**

This tab allows you to view the information (Color, Tool Tip and Pin Name Display) on and change the setting for the [Device Top View] selected in the Project Tree panel.

![Figure A-6. [Device Top View Settings] Tab](image)

The following items are explained here.
- [How to open]
- [Description of each area]

[How to open]

- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Top View], and then select [Property] from the [View] menu.
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Top View], and then select [Property] from the context menu.

**Remark** If this panel is already open, selecting a different [Device Top View] in the Project Tree panel changes the content displayed accordingly.

[Description of each area]

(1) **[Color] category**

Select the display colors to differentiate the pin groups (Power pins, Special pins, etc.) in the device top view.

<table>
<thead>
<tr>
<th>Power pins</th>
<th>Selects the display color for power pins (pins whose use is limited to power).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special pins</td>
<td>Selects the display color for special pins (pins with specified uses).</td>
</tr>
</tbody>
</table>
To change the setting of the color, use the following color palette which opens by making a selection from the dropdown list in this area.

### Figure A-7. Color Palette

![Color Palette](image)

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unused pins</strong></td>
<td>Selects the display color for unused pins (dual-use pins with no use set in the Device Pin List panel).</td>
</tr>
<tr>
<td><strong>Used pins</strong></td>
<td>Selects the display color for used pins (dual-use pins with a use set in the Device Pin List panel).</td>
</tr>
<tr>
<td><strong>Device</strong></td>
<td>Selects the display color of the microcontroller.</td>
</tr>
<tr>
<td><strong>Highlight color for a selected pin</strong></td>
<td>Selects the background color of a pin selected in the Device Pin List panel, on the [Pin Number] tab.</td>
</tr>
<tr>
<td><strong>Highlight color for macro pins</strong></td>
<td>Selects the background color of pins selected in the Device Pin List panel, on the [Macro] tab.</td>
</tr>
<tr>
<td><strong>Highlight color for external peripheral pins</strong></td>
<td>Selects the background color of pins selected in the Device Pin List panel, on the [External Peripheral] tab.</td>
</tr>
</tbody>
</table>

**Remark**

To change the setting of the color, use the following color palette which opens by making a selection from the dropdown list in this area.

### Figure A-7. Color Palette

![Color Palette](image)

<table>
<thead>
<tr>
<th>Tool Tip</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Display all</strong></td>
<td>Displays the &quot;Description&quot;, &quot;Recommend Connection for Unused&quot;, and &quot;Attention&quot; strings for the device pin list.</td>
</tr>
<tr>
<td><strong>Description / recommended connection for unused pin only</strong></td>
<td>Displays the &quot;Description&quot;, and &quot;Recommend Connection for Unused&quot; strings for the device pin list.</td>
</tr>
<tr>
<td><strong>Attention only</strong></td>
<td>Displays the &quot;Attention&quot; string for the device pin list.</td>
</tr>
<tr>
<td><strong>Not display</strong></td>
<td>Hides tooltips when the mouse cursor hovers over a pin.</td>
</tr>
</tbody>
</table>

### (2) [Tool Tip] category

Select whether to display a tooltip with information about a pin when the mouse cursor is moved over the pin in the device top view.

<table>
<thead>
<tr>
<th>Tool Tip</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Display all</strong></td>
<td>Displays the &quot;Description&quot;, &quot;Recommend Connection for Unused&quot;, and &quot;Attention&quot; strings for the device pin list.</td>
</tr>
<tr>
<td><strong>Description / recommended connection for unused pin only</strong></td>
<td>Displays the &quot;Description&quot;, and &quot;Recommend Connection for Unused&quot; strings for the device pin list.</td>
</tr>
<tr>
<td><strong>Attention only</strong></td>
<td>Displays the &quot;Attention&quot; string for the device pin list.</td>
</tr>
<tr>
<td><strong>Not display</strong></td>
<td>Hides tooltips when the mouse cursor hovers over a pin.</td>
</tr>
</tbody>
</table>

### (3) [Pin Name Display] category

Select whether to display additional information about the pin in the device top view.
<table>
<thead>
<tr>
<th>Define name</th>
<th>Selects whether to display the &quot;Define Name&quot; string of the device pin list appended to the pin in the device top view.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td>Displays the &quot;Define Name&quot; string of the device pin list in appended format.</td>
</tr>
<tr>
<td>Not display</td>
<td>Hides the &quot;Define Name&quot; string of the device pin list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin function</th>
<th>Selects whether to also display unselected functions in the device top view when a function has been selected from the device pin list's &quot;Function&quot; feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display all</td>
<td>Displays functions selected via the device pin list's &quot;Function&quot; feature in parentheses.</td>
</tr>
<tr>
<td>Selected function only</td>
<td>Only display functions selected via the device pin list's &quot;Function&quot; feature in the device top view.</td>
</tr>
</tbody>
</table>
[Generation] tab

This tab allows you to view the information (Product Information, Generate File Mode and Pin Configurator Reflect Mode) on and change the setting for the [Code Generator (Design Tool)] selected in the Project Tree panel.

![Figure A-8. [Generation] Tab](image)

The following items are explained here.

- **[How to open]**
- **[Description of each area]**

**[How to open]**

- On the Project Tree panel, select [Project name (Project)] >> [Code Generator (Design Tool)], and then select [Property] from the [View] menu.
- On the Project Tree panel, select [Project name (Project)] >> [Code Generator (Design Tool)], and then select [Property] from the context menu.

**Remark** If this panel is already open, selecting a different [Code Generator (Design Tool)] in the Project Tree panel changes the content displayed accordingly.

**[Description of each area]**

1. **[Product Information] category**
   This area displays product information (Version and Release date) on Code Generator.

<table>
<thead>
<tr>
<th>Version</th>
<th>Displays the version of Code Generator (Code Library).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release date</td>
<td>Displays the release date of Code Generator (Code Library).</td>
</tr>
</tbody>
</table>
(2) [Generate File Mode] category
This area allows you to view and change the setting for the file generation mode (Output control of API function, Generate file, etc.) of Code Generator.

<table>
<thead>
<tr>
<th>Output control of API function</th>
<th>Views or Selects the type of output API functions (all API functions or only initialization API functions) when the Generate Code button is pressed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output all API functions</td>
<td>Outputs all API functions.</td>
</tr>
<tr>
<td>according to the setting</td>
<td></td>
</tr>
<tr>
<td>Output only initialization</td>
<td>Outputs only initialization API functions.</td>
</tr>
<tr>
<td>API function</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generate file</th>
<th>Views or selects the operation mode applied when the Generate Code button is pressed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do nothing if file exists</td>
<td></td>
</tr>
<tr>
<td>Merge file</td>
<td>If a file with the same name exists, a new file will not be output.</td>
</tr>
<tr>
<td>Overwrite file</td>
<td>If a file with the same name exists, the existing file is overwritten by a new file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output folder</th>
<th>Views or selects the destination folder for various files (source code and report files) which are output when the Generate Code button is pressed or when [File] menu &gt;&gt; [Save Code Generator Report] is selected.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Report type</th>
<th>Views or selects the format of the report files (a file containing information configured using Code Generator and a file containing information regarding the source code) which are output when [File] menu &gt;&gt; [Save Code Generator Report] is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML file</td>
<td>Outputs a report file in HTML format.</td>
</tr>
<tr>
<td>CSV file</td>
<td>Outputs a report file in CSV format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register files</th>
<th>Selects whether source code generated by pressing the Generate Code button should be added to the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output files to project</td>
<td>Adds output source code to the project. The source code will be added to the Project Tree panel, under the [File] - [Code Generator] node.</td>
</tr>
<tr>
<td>Not output files to project</td>
<td>Does not add output source code to the project.</td>
</tr>
</tbody>
</table>

**Remark** To change the output destination, use the Browse For Folder dialog box which opens by pressing the [...] button in this area.

(3) [Pin Configurator Reflect Mode] category
Configure the information linking (Mode) between Code Generator and Pin Configurator.
<table>
<thead>
<tr>
<th>Mode</th>
<th>Selects whether to reflect the settings made in the Code Generator panel in the Device Pin List panel when the Reflect in Pin button is pressed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflected</td>
<td>Reflects Code Generator panel settings in the Device Pin List panel.</td>
</tr>
<tr>
<td>Not reflected</td>
<td>Does not reflect Code Generator panel settings in the Device Pin List panel.</td>
</tr>
</tbody>
</table>

**Remark**  If "Not reflected" is selected, then the Reflect in Pin button will be grayed out (deselected).
[Macro Setting] tab

This tab allows you to view the information (Macro Information) on and change the setting for the peripheral function node "[System], [Port], etc." selected in the Project Tree panel, or the peripheral function button " , , etc." pressed in the Code Generator panel.

Figure A-9. [Macro Setting] Tab

The following items are explained here.
- [How to open]
- [Description of each area]

[How to open]
- On the Project Tree panel, select [Project name (Project)] >> [Code Generator (Design Tool)] >> Peripheral function node "[System], [Port], etc.", and then select [Property] from the [View] menu.
- On the Project Tree panel, select [Project name (Project)] >> [Code Generator (Design Tool)] >> Peripheral function node "[System], [Port], etc.", and then select [Property] from the context menu.

Remarks 1. If this panel is already open, selecting a different peripheral function node "[System], [Port], etc." in the Project Tree panel changes the content displayed accordingly.
2. If this panel is already open, pressing a different type of peripheral function button " , , etc." in the Code Generator panel changes the content displayed accordingly.

[Description of each area]

(1) [Macro Information] category
This area allows you to view the information (Macro name) on and change the setting for the peripheral function node "[System], [Port], etc." selected in the Project Tree panel, or the peripheral function button " , , etc." pressed in the Code Generator panel.

| Macro name | Displays the type of peripheral function node selected in the Project Tree panel or the type of peripheral function button pressed in the Code Generator panel. |
[File Setting] tab

This tab allows you to view the information (File Information) on and change the setting for the file selected in the Code Generator Preview panel.

Figure A-10. [File Setting] Tab

The following items are explained here.
- [How to open]
- [Description of each area]

[How to open]
- On the Code Generator Preview panel, select a file, and then select [Property] from the [View] menu.
- On the Code Generator Preview panel, select a file, and then select [Property] from the context menu.

[Description of each area]

(1) [File Information] category
This area allows you to view the information (Default name and File name) on and change the setting for the file selected in the Code Generator Preview panel.

<table>
<thead>
<tr>
<th>Default name</th>
<th>Views or selects the setting that determines whether the name of the file selected in the Code Generator Preview panel is a default name or not.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>The file name is a default name. Changing this area from &quot;No&quot; to &quot;Yes&quot; changes the name of the file to its default name.</td>
</tr>
<tr>
<td>No</td>
<td>The file name is not a default name.</td>
</tr>
<tr>
<td>File name</td>
<td>Displays or change the name of the file selected on the Code Generator Preview panel.</td>
</tr>
</tbody>
</table>
CubeSuite+ V1.03.00

APPENDIX A  WINDOW REFERENCE

Device Pin List panel

This panel allows you to enter information on each pin of the microcontroller.

Remark  The Device pin list area can be zoomed in and out by \(100\%\) in the tool bar, or by operating the mouse wheel while holding down the [Ctrl] key.

Figure A-11.  Device Pin List Panel

The following items are explained here.
- [How to open]
- [Description of each area]
- [File] menu (Device Pin List panel-dedicated items)
- [Help] menu (Device Pin List panel-dedicated items)

[How to open]
- On the Project Tree panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List].
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List], and then press the [Enter] key.
- From the [View] menu, select [Pin Configurator] >> [Device Pin List].

[Description of each area]

(1) Toolbar
This area consists of the following buttons.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Expanded View" /></td>
<td>Displays the information in the Device pin list area in an expanded view.</td>
</tr>
<tr>
<td><img src="image" alt="Folded View" /></td>
<td>Displays the information in the Device pin list area in a folded view only.</td>
</tr>
<tr>
<td><img src="image" alt="Auto Configure" /></td>
<td>Clicks this button to automatically process the configuration information in the selected function, I/O, N-ch, and other fields after selecting one of the peripheral functions displayed in the first level on the [Macro] tab.</td>
</tr>
</tbody>
</table>
Remarks 1. Click the \( \text{button} \) to add the information in question as a choice in the "External Parts" column of the [Macro] tab and the [Pin Number] tab.

2. Click the \( \text{button} \) to remove the external peripheral component in question from the Device Top View panel.

(2) Device pin list area
This area displays the "device pin list" for entering information on the pins of the microcontroller.

(3) Tab selection area
Selecting the tab changes the order in which "information on each pin of the microcontroller" is displayed. This panel has the following tabs:
- [Pin Number] tab
  This tab displays information on each pin of the microcontroller in the order of pin number.
- [Macro] tab
  This tab displays information on each pin of the microcontroller in the order it was grouped into peripheral functions.
- [External Peripheral] tab
  This tab displays information about the pins connected to external peripherals in order grouped at the external-peripheral component level.

[[File] menu (Device Pin List panel-dedicated items)]

<table>
<thead>
<tr>
<th>Save Pin List</th>
<th>Saves a report file (a file containing information configured using Pin Configurator: device pin list) overwriting the existing file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Pin List As...</td>
<td>Opens the Save As dialog box for naming and saving a report file (a file containing information configured using Pin Configurator: device pin list).</td>
</tr>
</tbody>
</table>

[[Help] menu (Device Pin List panel-dedicated items)]

| Open Help for Device Pin List Panel | Displays the help of this panel. |
**[Pin Number] tab**

This tab displays information on each pin of the microcontroller in the order of pin number.

![Figure A-12. [Pin Number] Tab](image)

The following items are explained here.
- [How to open]
- [Description of each area]

**[How to open]**

- On the Project Tree panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List].
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List], and then press the [Enter] key.
- From the [View] menu, select [Pin Configurator] >> [Device Pin List].

**[Description of each area]**

(1) Device pin list area

This area displays the "device pin list" for entering information on the pins of the microcontroller.
The device pin list in this area is organized in the order of pin number.
The following are the columns comprising the device pin list.

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Number</td>
<td>Displays the pin number of the pin.</td>
</tr>
<tr>
<td>Pin Name</td>
<td>This area allows you to select &quot;which function to use&quot; when the pin has more than one functions.</td>
</tr>
<tr>
<td>Function</td>
<td>This area allows you to select &quot;which function to use&quot; when the pin has more than one functions.</td>
</tr>
<tr>
<td>I/O</td>
<td>This area allows you to select the I/O mode of the pin.</td>
</tr>
<tr>
<td>N-ch</td>
<td>This area allows you to select &quot;which output mode to apply&quot; when using the pin in the output mode.</td>
</tr>
</tbody>
</table>
### Remarks

1. You cannot add information in the "Pin Number" column, "Pin Name" column, "Description" column, "Recommend Connection for Unused" column and "Attention" column because they contain fixed information.

2. If the "Free" in the "Function" column is changed to a specific pin name, color of the corresponding pin in the Device Top View panel changes from the "color representing the unused pins" to the "color representing the used pins" selected by clicking [Device Top View Settings] tab >> [Color] in the Property panel.

3. To move columns (change the display order) in the device pin list, drag and drop the desired column to the desired location.

4. To add the "user's own column", use the New Column dialog box which opens by pressing the [New Column...] button in the Column Chooser dialog box which opens by pressing the button in the upper left corner of the device pin list.

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Name</td>
<td>This area allows you to assign a &quot;user-defined pin name&quot; to the pin.</td>
</tr>
<tr>
<td></td>
<td>Within 256 characters can be entered in the [Define Name].</td>
</tr>
<tr>
<td>Description</td>
<td>Displays the summary of function of the pin.</td>
</tr>
<tr>
<td>Recommend Connection for Unused</td>
<td>Displays instructions on how to handle the pin when it is not used.</td>
</tr>
<tr>
<td></td>
<td>This column displays information only when the &quot;Free&quot; is selected in the &quot;Function&quot; column.</td>
</tr>
<tr>
<td>Attention</td>
<td>Displays the precaution on using the pin.</td>
</tr>
<tr>
<td>External Parts</td>
<td>This area is for selecting which external peripheral controller to connect the pin to.</td>
</tr>
</tbody>
</table>
**[Macro] tab**

This tab displays information on each pin of the microcontroller in the order it was grouped into peripheral functions.

![Figure A-13. [Macro] Tab](image)

The following items are explained here.
- [How to open]
- [Description of each area]

**[How to open]**
- On the Project Tree panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List].
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List], and then press the [Enter] key.
- From the [View] menu, select [Pin Configurator] >> [Device Pin List].

**[Description of each area]**

**(1) Device pin list area**

This area displays the "device pin list" for entering information on the pins of the microcontroller. The device pin list in this area is organized in the order the pins were grouped into peripheral functions.

**(a) First layer**

The following are the columns comprising the device pin list.

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro Name</td>
<td>Displays the name of the peripheral function.</td>
</tr>
<tr>
<td>Total</td>
<td>Displays the total number of pins assigned to the peripheral function.</td>
</tr>
<tr>
<td>Used</td>
<td>Displays the total number of pins for which the purpose has been set.</td>
</tr>
<tr>
<td>Used in Other Macro</td>
<td>Displays the total number of pins for which the purpose has been set by other peripheral functions.</td>
</tr>
</tbody>
</table>
(b) Second layer

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Number</td>
<td>Displays the pin number of the pin.</td>
</tr>
<tr>
<td>Pin Name</td>
<td>Displays the pin name of the pin.</td>
</tr>
<tr>
<td>Function</td>
<td>This area allows you to select &quot;which function to use&quot; when the pin has more than one functions.</td>
</tr>
<tr>
<td>I/O</td>
<td>This area allows you to select the I/O mode of the pin.</td>
</tr>
<tr>
<td>N-ch</td>
<td>This area allows you to select &quot;which output mode to apply&quot; when using the pin in the output mode.</td>
</tr>
<tr>
<td>Define Name</td>
<td>This area allows you to assign a &quot;user-defined pin name&quot; to the pin. Within 256 characters can be entered in the [Define Name].</td>
</tr>
<tr>
<td>Description</td>
<td>Displays the summary of function of the pin.</td>
</tr>
<tr>
<td>Recommend Connection for Unused</td>
<td>Displays instructions on how to handle the pin when it is not used. This column displays information only when the &quot;Free&quot; is selected in the &quot;Function&quot; column.</td>
</tr>
<tr>
<td>Attention</td>
<td>Displays the precaution on using the pin.</td>
</tr>
<tr>
<td>External Parts</td>
<td>This area is for selecting which external peripheral controller to connect the pin to.</td>
</tr>
</tbody>
</table>

Remarks 1. You cannot add information in the "Macro Name", "Total", "Used", "Used by other function", "Pin Number", "Pin Name", "Description", "Recommend Connection for Unused" and "Attention" columns because they contain fixed information.

2. If the "Free" in the "Function" column is changed to a specific pin name, color of the corresponding pin in the Device Top View panel changes from the "color representing the unused pins" to the "color representing the used pins" selected by clicking [Device Top View Settings] tab >> [Color] in the Property panel.

3. To move columns (change the display order) in the device pin list, drag and drop the desired column to the desired location.

4. To add the "user's own column", use the New Column dialog box which opens by pressing the [New Column...] button in the Column Chooser dialog box which opens by pressing the button in the upper left corner of the device pin list.
[External Peripheral] tab

This tab displays information about the pins connected to external peripherals in order grouped at the external-peripheral component level.

Figure A-14. [External Peripheral] Tab

The following items are explained here.
- [How to open]
- [Description of each area]

[How to open]
- On the Project Tree panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List].
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Pin List], and then press the [Enter] key.
- From the [View] menu, select [Pin Configurator] >> [Device Pin List].

[Description of each area]

(1) Device pin list area
This area displays the "device pin list" for entering information on the pins connected to external peripheral parts. Note that items in this area's device pin list are sorted by groups at the external peripheral controller level.

(a) First layer
The following are the columns comprising the device pin list.

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Peripheral</td>
<td>Displays the name of the external peripheral controller. To change the name, select this field and then press the [F2] key.</td>
</tr>
<tr>
<td>Total</td>
<td>Displays the total number of pins allocated for connection with the microcontroller.</td>
</tr>
</tbody>
</table>
(b) Second layer

| Column Heading                  | Outline                                                                 
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Number</td>
<td>Displays the pin number of the pin.</td>
</tr>
<tr>
<td>Pin Name</td>
<td>Displays the pin name of the pin.</td>
</tr>
<tr>
<td>Function</td>
<td>This area allows you to select &quot;which function to use&quot; when the pin has more than one functions.</td>
</tr>
<tr>
<td>I/O</td>
<td>This area allows you to select the I/O mode of the pin.</td>
</tr>
<tr>
<td>N-ch</td>
<td>This area allows you to select &quot;which output mode to apply&quot; when using the pin in the output mode.</td>
</tr>
<tr>
<td>Define Name</td>
<td>This area allows you to assign a &quot;user-defined pin name&quot; to the pin.</td>
</tr>
<tr>
<td></td>
<td>Within 256 characters can be entered in the [Define Name].</td>
</tr>
<tr>
<td>Description</td>
<td>Displays the summary of function of the pin.</td>
</tr>
<tr>
<td>Recommend Connection for Unused</td>
<td>Displays instructions on how to handle the pin when it is not used. The column displays information only when the &quot;Free&quot; is selected in the &quot;Function&quot; column.</td>
</tr>
<tr>
<td>Attention</td>
<td>Displays the precaution on using the pin.</td>
</tr>
</tbody>
</table>

**Remarks**

1. You cannot add information in the "External Peripheral Name", "Connected Pins", "Pin Number", "Pin Name", "Description", "Recommend Connection for Unused" and "Attention" columns because they contain fixed information.

2. If the "Free" in the "Function" column is changed to a specific pin name, color of the corresponding pin in the Device Top View panel changes from the "color representing the unused pins" to the "color representing the used pins" selected by clicking [Device Top View Settings] tab >> [Color] in the Property panel.

3. To move columns (change the display order) in the device pin list, drag and drop the desired column to the desired location.

4. To add the "user's own column", use the New Column dialog box which opens by pressing the [New Column...] button in the Column Chooser dialog box which opens by pressing the button in the upper left corner of the device pin list.
Device Top View panel

This panel displays the information entered in the Device Pin List panel.

**Remark**  The Device top view area can be zoomed in and out by 100% in the tool bar.

![Device Top View Panel](image)

The following items are explained here.
- [How to open]
- [Description of each area]
- [[File] menu (Device Top View panel-dedicated items)]
- [[Help] menu (Device Top View panel-dedicated items)]
- [Context menu]

**[How to open]**
- On the Project Tree panel, double-click [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Top View].
- On the Project Tree panel, select [Project name (Project)] >> [Pin Configurator (Design Tool)] >> [Device Top View] and then press the [Enter] key.
- From the [View] menu, select [Pin Configurator] >> [Device Top View].
Remark In the Property panel, on the [Pin Configurator Settings] tab, if "BGA" is selected for the Package type, then this panel cannot be opened.

[Description of each area]

(1) Toolbar
This area consists of the following buttons.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Button]</td>
<td>Clicks this button to enable changing of the display in the Device top view area by drag and drop. By pressing this button, the shape of the mouse cursor in the Device top view area changes from the arrow to the hand.</td>
</tr>
<tr>
<td>![Button]</td>
<td>Clicks this button to enable moving external peripheral components in the Device top view area to arbitrary locations, and select pins. By pressing this button, the shape of the mouse cursor which has changed into the hand by pressing the button reverts back to the arrow.</td>
</tr>
<tr>
<td>![Button]</td>
<td>Rotates the content in the Device top view area 90 degrees counter-clockwise.</td>
</tr>
<tr>
<td>![Button]</td>
<td>Rotates the content in the Device top view area 90 degrees clockwise.</td>
</tr>
<tr>
<td>![Button]</td>
<td>Expands or reduces the content in the Device top view area.</td>
</tr>
</tbody>
</table>

(2) [User Define] area
Drag and drop the button from this area to the Device top view area to create and display an external peripheral controller.

(3) Device top view area
This area displays the pin assignment of the microcontroller. Settings of the pin assignment are displayed using the colors specified by selecting [Device Top View Settings] tab >> [Color] in the Property panel.

Remark If the pin name in the diagram is double-clicked, the Device Pin List panel opens and the focus moves to the clicked pin in the list.

[[File] menu (Device Top View panel-dedicated items)]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Top View</td>
<td>Saves a report file (a file containing information configured using Pin Configurator: device top view) overwriting the existing file.</td>
</tr>
<tr>
<td>Save Top View As...</td>
<td>Opens the Save As dialog box for naming and saving a report file (a file containing information configured using Pin Configurator: device top view).</td>
</tr>
</tbody>
</table>

[[Help] menu (Device Top View panel-dedicated items)]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Help for Device Top View Panel</td>
<td>Displays the help of this panel.</td>
</tr>
</tbody>
</table>

[Context menu]
When you right click on a pin or external peripheral controller in the Device top view area, the following context menu displays.
(1) When a pin is right clicked

<table>
<thead>
<tr>
<th>Use as</th>
<th>If the pin has multiple functions, select which function to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to External Peripheral</td>
<td>Selects which external peripheral controller to connect the pin to.</td>
</tr>
</tbody>
</table>

(2) When an external peripheral controller is right clocked

<table>
<thead>
<tr>
<th>Disconnect Pin</th>
<th>Disconnects from the pin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete External Peripheral</td>
<td>Removes the external peripheral controller.</td>
</tr>
</tbody>
</table>
This panel allows you to configure the information necessary to control the peripheral functions provided by the microcontroller.

**Figure A-16. Code Generator Panel: [System]**

![Code Generator Panel](image)

The following items are explained here.

- **[How to open]**
- **[Description of each area]**
- **[[File] menu (Code Generator panel-dedicated items)]**
- **[[Help] menu (Code Generator panel-dedicated items)]**

**[How to open]**

- On the **Project Tree panel**, double-click **[Project name (Project)]** >> **[Code Generator (Design Tool)]** >> Peripheral function node "[System], [Port], etc."
- On the **Project Tree panel**, select **[Project name (Project)]** >> **[Code Generator (Design Tool)]** >> Peripheral function node "[System], [Port], etc.", and then press the [Enter] key.

**Remark** If this panel is already open, pressing a different peripheral function button "[ ], [ ], etc." changes the content displayed in the **Information setting area** accordingly.
[Description of each area]

(1) Toolbar

This area consists of the following "peripheral function buttons". When there is peripheral function target microcontroller is not supporting, peripheral function button is not disokayed.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Reflect in Pin" /></td>
<td>Reflects settings made on this panel in the Device Pin List panel, and then output the changed contents to the Output panel. This button will be grayed out (disabled) if &quot;Not reflected&quot; is selected in the [PinPart Combination Mode] category of the [Generation] tab.</td>
</tr>
<tr>
<td><img src="image" alt="Generate Code" /></td>
<td>Outputs the source code (device driver program) to the folder specified by selecting [Generation] tab &gt;&gt; [Output folder] in the Property panel.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[System] for configuring the information necessary to control the functions of clock generator, on-chip debug function and etc. provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[System] for configuring the information necessary to control the functions of clock generator, on-chip debug function and etc. provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Port] for configuring the information necessary to control the port functions provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Port] for configuring the information necessary to control the port functions provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Interrupt] for configuring the information necessary to control the interrupt functions and the key interrupt function provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Interrupt] for configuring the information necessary to control the interrupt functions and the key interrupt function provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Serial] for configuring the information necessary to control the functions of serial array unit and functions of serial interface provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Serial] for configuring the information necessary to control the functions of serial array unit and functions of serial interface provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Operational Amplifier] for configuring the information necessary to control the functions of comparator/programmable gain amplifier provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Operational Amplifier] for configuring the information necessary to control the functions of comparator/programmable gain amplifier provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Comparator] for configuring the information necessary to control the function of comparator provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Comparator] for configuring the information necessary to control the function of comparator provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[A/D Converter] for configuring the information necessary to control the function of A/D converter provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[A/D Converter] for configuring the information necessary to control the function of A/D converter provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Timer] for configuring the information necessary to control the functions of timer array unit provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Timer] for configuring the information necessary to control the functions of timer array unit provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Watchdog Timer] for configuring the information necessary to control the functions of watchdog timer provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Watchdog Timer] for configuring the information necessary to control the functions of watchdog timer provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Real-time Clock] for configuring the information necessary to control the functions of real-time counter provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Real-time Clock] for configuring the information necessary to control the functions of real-time counter provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[Clock Output] for configuring the information necessary to control the functions of clock output controller provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[Clock Output] for configuring the information necessary to control the functions of clock output controller provided by the microcontroller&quot;.</td>
</tr>
<tr>
<td><img src="image" alt="Changes the content displayed in the Information setting area to the &quot;[LVI] for configuring the information necessary to control the functions of low-voltage detector provided by the microcontroller&quot;." /></td>
<td>Changes the content displayed in the Information setting area to the &quot;[LVI] for configuring the information necessary to control the functions of low-voltage detector provided by the microcontroller&quot;.</td>
</tr>
</tbody>
</table>
(2) Information setting area

The content displayed in this area differs depending on the "peripheral function node" or "peripheral function button" selected or pressed when opening this panel.

See User's Manual for Microcontroller for details on the items to be set.

[[File] menu (Code Generator panel-dedicated items)]

| Save Code Generator Report | Outputs report files (a file containing information configured using Code Generator and a file containing information regarding the source code). |

Remarks 1. The output format for the report file (either HTML or CSV) is selected by clicking [Generation] tab >> [Report type] in the Property panel.


[[Help] menu (Code Generator panel-dedicated items)]

| Open Help for Code Generator Panel | Displays the help of this panel. |
This panel allows you to confirm or configure on a per-API function basis the setting that determines whether or not the source code (device driver program) is generated when the button is pressed in the Code Generator panel. It also allows you to confirm the source code that reflects the information configured in the Code Generator panel.

**Figure A-17. Code Generator Preview Panel**

The following items are explained here.

- [How to open]
- [Description of each area]
- [[File] menu (Code Generator Preview panel-dedicated items)]
- [[Help] menu (Code Generator Preview panel-dedicated items)]
- [Context menu]

**[How to open]**

- From the [View] menu, select [Code Generator Preview].

**[Description of each area]**

1. **Preview tree**

   This area allows you to confirm or configure on a per-API function basis the setting that determines whether or not the source code (device driver program) is generated when the button is pressed in the Code Generator panel.

   **Remarks 1.**
   
   You can change the source code to be displayed by selecting the source file name or API function name in this tree.

   2. To select whether or not to generate the source code, use the context menu (Generate code/Not generate code) which is displayed by right-clicking the mouse while the mouse cursor is on the desired icon in the tree.
3. You can confirm the current setting that determines whether or not to generate the source code by checking the type of icon.

Table A-2. Setting That Determines Whether or Not to Generate the Source Code

<table>
<thead>
<tr>
<th>Type of Icon</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Source Code Icon" /></td>
<td>Source code for the currently selected API function is generated. If this icon is displayed next to the API function, the corresponding source code must be generated (it is impossible to change the icon to <img src="image" alt="Source Code Icon" />).</td>
</tr>
<tr>
<td><img src="image" alt="Source Code Icon" /></td>
<td>Source code for the currently selected API function is generated.</td>
</tr>
<tr>
<td><img src="image" alt="Source Code Icon" /></td>
<td>Source code for the currently selected API function is not generated.</td>
</tr>
</tbody>
</table>

(2) Source code display area

This area allows you to confirm the source code (device driver program) that reflects the information configured in the Code Generator panel.

The following table displays the meaning of the color of the source code text displayed in this area.

Table A-3. Color of Source Code

<table>
<thead>
<tr>
<th>Color</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Comment</td>
</tr>
<tr>
<td>Blue</td>
<td>Reserved word for C compiler</td>
</tr>
<tr>
<td>Red</td>
<td>Numeric value</td>
</tr>
<tr>
<td>Black</td>
<td>Code section</td>
</tr>
<tr>
<td>Gray</td>
<td>File name</td>
</tr>
</tbody>
</table>

Remarks 1. You cannot edit the source code within this panel.
2. For some of the API functions (such as API functions for serial array units), values such as the SFR register value are calculated and finalized when the source code is generated (when the ![Generate Code](image) button on the Code Generator panel is pressed). For this reason, the source code displayed in this panel may not be the same as that would actually be generated.
3. You can change the source code to be displayed by selecting the source file name or API function name in the preview tree.

[[File] menu (Code Generator Preview panel-dedicated items)]

| Save Code Generator Report | Outputs report files (a file containing information configured using Code Generator and a file containing information regarding the source code). |

Remarks 1. The output format for the report file (either HTML or CSV) is selected by clicking [Generation tab >> Report type] in the Property panel.
2. The destination folder for the report file is specified by clicking [Generation tab >> Output folder] in the Property panel.
## [Help] menu (Code Generator Preview panel-dedicated items)]

| Open Help for Code Generator Preview Panel | Displays the help of this panel. |

## [Context menu]

The following context menu items are displayed by right clicking the mouse.

| Generate code | Makes a setting so that the source code of the currently selected API function is generated to the folder specified by selecting [Generation] tab >> [Output folder] in the Property panel. Selecting this context menu item changes the icon of the currently selected API function from to . This item will be grayed out (disabled) if the currently selected API function is not initialization API function, and "Output only initialization API function" is selected [Generation] tab >> [Output control of API function] in the Property panel. |
| Not generate code | Makes a setting so that the source code of the currently selected API function is not generated when the button is pressed in the Code Generator panel. Selecting this context menu item changes the icon of the currently selected API function from to . |
| Rename | Selecting this menu item changes the name portion of the currently selected file or API function into an edit box for editing the name. You can change the name of the file or API function by editing its name in the edit box. |
| Default | Reverts the file name or API function name to its original name before it was edited. |
| Property | Opens the Property panel that contains the information for the currently selected file. |
This panel is used to display operating logs for various components (design tool, build tool, debug tool, etc.) provided by CubeSuite+.

The messages are classified by the message origination tool and displayed on the individual tabs.

**Remark**  The **Message area** can be zoomed in and out by \[100%\] in the tool bar, or by operating the mouse wheel while holding down the [Ctrl] key.

**Figure A-18.  Output Panel**

The following items are explained here.
- [How to open]
- [Description of each area]
- [[File] menu (Output panel-dedicated items)]
- [[Edit] menu (Output panel-dedicated items)]
- [Context menu]

**[How to open]**
- From the [View] menu, select [Output].

**[Description of each area]**

1. **Message area**
   The output messages of each tool are displayed.
   The colors of message display differ with the type of message as shown below (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box).

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Display Example (Default)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal message</td>
<td><strong>ABCDEF abcdef 0123</strong></td>
<td>Character color Black</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background color White</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displayed with information notices.</td>
</tr>
<tr>
<td>Warning message</td>
<td>ABCDEF abcdef 0123</td>
<td>Character color Blue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background color Standard color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displayed with warnings about operations.</td>
</tr>
<tr>
<td>Error message</td>
<td><strong>ABCDEF abcdef 0123</strong></td>
<td>Character color Red</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background color Light gray</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displayed when there is a critical error, or when execution is not possible due to a operational mistake.</td>
</tr>
</tbody>
</table>
(2) Tab selection area
Select the tab that indicates the origin of message.
The following tabs are available for the debug tool.

<table>
<thead>
<tr>
<th>Tab Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Messages</td>
<td>Displays operation logs for all components (design tool, build tool, debug tool, etc.) provided by CubeSuite+.</td>
</tr>
<tr>
<td>Code Generator</td>
<td>Display only operation logs for the Code Generator out of those for various components (design tool, build tool, debug tool, etc.) provided by CubeSuite+.</td>
</tr>
</tbody>
</table>

Caution Even if a new message is output on a deselected tab, tab selection will not automatically switch. In this case, " * " mark will be added in front of the tab name, indicating that a new message has been output.

[[File] menu (Output panel-dedicated items)]

| Save Output-Tab Name | Saves the message corresponding to the specified tab overwriting the existing file. |
| Save Output-Tab Name As... | Opens the Save As dialog box for naming and saving the message corresponding to the specified tab. |

[[Edit] menu (Output panel-dedicated items)]

| Copy | Sends the character string or lines selected with range selection to the clipboard. |
| Select All | Selects all the messages displayed on the Message area. |
| Search... | Opens the Search and Replace dialog box for searching strings with the [Quick Search] tab selected. |
| Replace... | Opens the Search and Replace dialog box for replacing strings with the [Whole Replace] tab selected. |

[Context menu]
The following context menu items are displayed by right clicking the mouse.

| Copy | Sends the character string or lines selected with range selection to the clipboard. |
| Select All | Selects all the messages displayed on the Message area. |
| Clear | Deletes all the messages displayed on the Message area. |
| Stop Searching | Cancels the search currently being executed. This is invalid when a search is not being executed. |
| Open Help for Message | Displays help for the message on the current caret location. This only applies to warning messages and error messages. |
Column Chooser dialog box

This dialog box allows you to choose whether or not to display the item listed in this dialog box in the device pin list, and add columns to or delete columns from the device pin list.

Figure A-19. Column Chooser Dialog Box

The following items are explained here.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]
- In the [Pin Number] tab of the Device Pin List panel, click the button.
- In the [Macro] tab of the Device Pin List panel, click the button.
- In the [External Peripheral] tab of the Device Pin List panel, click the button.

[Description of each area]

(1) Operational object selection area
This area allows you to select the device pin list to be configured in this dialog box.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Number</td>
<td>Configures the device pin list corresponding to the [Pin Number] tab.</td>
</tr>
<tr>
<td>Macro</td>
<td>Configures the device pin list belonging to the first layer of the [Macro] tab.</td>
</tr>
<tr>
<td>Macro - Pin</td>
<td>Configures the device pin list belonging to the second layer of the [Macro] tab.</td>
</tr>
<tr>
<td>External Peripheral</td>
<td>Configures the device pin list belonging to the first layer of the [External Peripheral] tab.</td>
</tr>
<tr>
<td>External Peripheral - Pin</td>
<td>Configures the device pin list belonging to the second layer of the [External Peripheral] tab.</td>
</tr>
</tbody>
</table>
Figure A-20. Operational Object ([Pin Number] Tab)

Figure A-21. Operational Object ([Macro] Tab: First Layer)

Figure A-22. Operational Object ([Macro] Tab: Second Layer)
Figure A-23. Operational Object ([External Peripheral] Tab: First Layer)

Figure A-24. Operational Object ([External Peripheral] Tab: Second Layer)

(2) Displayed item selection area
Select whether or not to display the item selected in the Operational object selection area in the device pin list.

<table>
<thead>
<tr>
<th>Checked</th>
<th>Displays the selected item in the device pin list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not checked</td>
<td>Hides the selected item in the device pin list.</td>
</tr>
</tbody>
</table>

[Function buttons]

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Column...</td>
<td>Opens the New Column dialog box for adding columns to the device pin list.</td>
</tr>
<tr>
<td>Delete Column</td>
<td>Deletes the selected columns from the device pin list.</td>
</tr>
<tr>
<td></td>
<td>You can only delete the column which you added using the New Column dialog box.</td>
</tr>
<tr>
<td>Default</td>
<td>Restores the column order to the default settings.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes this dialog box.</td>
</tr>
</tbody>
</table>
New Column dialog box

This dialog box allows you to add your own column to the device pin list.

Figure A-25. New Column Dialog Box

The following items are explained here.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- Click the [New Column...] button in the Column Chooser dialog box.

[Description of each area]

(1) [Name]
This area allows you to enter column headings of the columns added to the device pin list.
Within 256 characters can be entered in the [Name].

(2) [Type]
Select the input format of the column to add to the device pin list.

<table>
<thead>
<tr>
<th>Text</th>
<th>Only character strings can be entered in the column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check box</td>
<td>Adds a column of check boxes.</td>
</tr>
<tr>
<td>Whole number</td>
<td>Only integers can be entered in the column.</td>
</tr>
<tr>
<td>Real number</td>
<td>Only real numbers can be entered in the column.</td>
</tr>
<tr>
<td>Date</td>
<td>Only dates in YYYYMMDD format can be entered in the column.</td>
</tr>
</tbody>
</table>

[Function buttons]

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Adds a column that has the column heading specified in the [Name] to the right end of the device pin list.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Ignores the setting and closes this dialog box.</td>
</tr>
</tbody>
</table>
### Browse For Folder dialog box

This dialog box allows you to specify the output destination for files (source code, report file, etc.).

#### Figure A-26. Browse For Folder Dialog Box

![Browse For Folder Dialog Box](image)

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

#### How to open

- In the [Generation] tab of the Property panel, click the [...] button in [Output folder].

#### Description of each area

1. **Folder location**
   
   Select the folder to which the files (source code, report file, etc.) are output.

#### Function buttons

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make New Folder</td>
<td>Creates a &quot;New Folder&quot; below the folder selected in the Folder location.</td>
</tr>
<tr>
<td>OK</td>
<td>Specifies the folder selected in the Folder location as the destination for the files.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Ignores the setting and closes this dialog box.</td>
</tr>
</tbody>
</table>
**Save As dialog box**

This dialog box allows you to name and save a file (such as a report file).

![Save As Dialog Box](image)

**Figure A-27. Save As Dialog Box**

The following items are explained here.
- [How to open]
- [Description of each area]
- [Function buttons]

**[How to open]**
- From the [File] menu, select [Save <object> As...].

**[Description of each area]**

(1) **[Save in]**
   Select the folder to which the files (report files, etc.) are output.

(2) **List of files**
   This area displays a list of files matching the conditions selected in [Save in] and [Save as type].
(3) [File name]
Specify the name of the file to be output.

(4) [Save as type]
Select the type of the file to be output.

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Office Excel Book (*.xls)</td>
<td>Microsoft Office Excel Book format</td>
</tr>
<tr>
<td>Bitmap (*.bmp)</td>
<td>Bitmap format</td>
</tr>
<tr>
<td>PNG (*.png)</td>
<td>PNG format</td>
</tr>
<tr>
<td>JPEG (*.jpg)</td>
<td>JPEG format</td>
</tr>
<tr>
<td>EMF (*.emf)</td>
<td>EMF format</td>
</tr>
</tbody>
</table>

[Function buttons]

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Outputs a file having the name specified in the [File name] and [Save as type] to the folder specified in the [Save in].</td>
</tr>
<tr>
<td>Cancel</td>
<td>Ignores the setting and closes this dialog box.</td>
</tr>
</tbody>
</table>
APPENDIX B OUTPUT FILES

This appendix describes the files output by Code Generator.

B.1 Overview

Below is a list of files output by Code Generator.

<table>
<thead>
<tr>
<th>Table B-1. File List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of Output</td>
</tr>
<tr>
<td>Peripheral function</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Project</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

B.2 Output File

Below are the files (peripheral function) output by Code Generator.

<table>
<thead>
<tr>
<th>Table B-2. File List (Peripheral Function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peripheral Function</td>
</tr>
</tbody>
</table>
| System | CG_system.c | CLOCK_Init  
CG_ChangeClockMode  
CG_ChangeFrequency  
CG_SelectPowerSaveMode  
CG_SelectStabTime  
CG_ChangePllMode  
CG_system_user.c | CLOCK_UserInit  
CG_ReadResetSource  
CG_system.h | - |
| Port | CG_port.c | PORT_Init  
PORT_ChangePmnInput  
PORT_ChangePmnOutput  
CG_port_user.c | PORT_UserInit  
CG_port.h | - |
| Interrupt | CG_int.c | INTP_Init  
KEY_Init  
INT_MaskableInterruptEnable  
INTPn_Disable |
<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Source File Name</th>
<th>Names of API Functions Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt</td>
<td>CG_int.c</td>
<td>INTPn_Enable, KEY_Disable, KEY_Enable</td>
</tr>
<tr>
<td></td>
<td>CG_int_user.c</td>
<td>INTP_UserInit, KEY_UserInit, MD_INTPn, MD_INTKR</td>
</tr>
<tr>
<td></td>
<td>CG_int.h</td>
<td>-</td>
</tr>
<tr>
<td>Serial</td>
<td>CG_serial.c</td>
<td>UART6_Init, UART6_Start, UART6_Stop, UART6_SendData, UART6_ReceiveData, CSI1n_Init, CSI1n_Start, CSI1n_Stop, CSI1n_ReceiveData, CSI1n_SendReceiveData, IICA_Init, IICA_Stop, IICA_MasterSendStart, IICA_MasterReceiveStart, IICA_StopCondition, IICA_SlaveSendStart, IICA_SlaveReceiveStart</td>
</tr>
<tr>
<td></td>
<td>CG_serial_user.c</td>
<td>UART6_UserInit, UART6_SendEndCallback, UART6_ReceiveEndCallback, UART6_SoftOverRunCallback, UART6_ErrorCallback, CSI1n_UserInit, CSI1n_SendEndCallback, CSI1n_ReceiveEndCallback, IICA_UserInit, IICA_MasterSendEndCallback, IICA_MasterReceiveEndCallback, IICA_MasterErrorCallback, IICA_SlaveSendEndCallback, IICA_SlaveReceiveEndCallback, IICA_SlaveErrorCallback, IICA_GetStopConditionCallback, MD_INTSR6, MD_INTSRE6, MD_INTST6, MD_INTCSI1n, MD_INTIICA0</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>Source File Name</td>
<td>Names of API Functions Included</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Serial</td>
<td>CG_serial.h</td>
<td>-</td>
</tr>
<tr>
<td>Operational Amplifier</td>
<td>CG_opamp.c</td>
<td>OPAMP_Init, PGA_Start, PGA_Stop, PGA_ChangePGAFactor, AMP_Start, AMP_Stop, AMPn_Start, AMPn_Stop</td>
</tr>
<tr>
<td></td>
<td>CG_opamp_user.c</td>
<td>OPAMP_UserInit</td>
</tr>
<tr>
<td></td>
<td>CG_opamp.h</td>
<td>-</td>
</tr>
<tr>
<td>Comparator</td>
<td>CG_comparator.c</td>
<td>Comparator_Init, Comparatorn_Start, Comparatorn_Stop</td>
</tr>
<tr>
<td></td>
<td>CG_comparator_user.c</td>
<td>Comparator_UserInit, MD_INTCMPn</td>
</tr>
<tr>
<td></td>
<td>CG_comparator.h</td>
<td>-</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>CG_ad.c</td>
<td>AD_Init, AD_ComparatorOn, AD_ComparatorOff, AD_Start, AD_Stop, AD_SelectADChannel, AD_Read, AD_ReadByte</td>
</tr>
<tr>
<td></td>
<td>CG_ad_user.c</td>
<td>AD_UserInit, MD_INTAD</td>
</tr>
<tr>
<td></td>
<td>CG_ad.h</td>
<td>-</td>
</tr>
<tr>
<td>Timer</td>
<td>CG_timer.c</td>
<td>TMX_Init, TMXn_Start, TMXn_Stop, TMXn_ChangeDuty, TMXn_ChangeDualDuty, TMX_EnableHighImpedanceState, TMX_DisableHighImpedanceState, TM00_Init, TM00_Start, TM00_Stop, TM00_ChangeTimerCondition, TM00_GetFreeRunningValue, TM00_SoftwareTriggerOn, TM00_ChangeDuty, TM00_GetPulseWidth, TM5n_Init, TM5n_Start</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>Source File Name</td>
<td>Names of API Functions Included</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Timer</td>
<td>CG_timer.c</td>
<td>TM5n_Stop&lt;br&gt;TM5n_ChangeTimerCondition&lt;br&gt;TM5n_ChangeDuty&lt;br&gt;TMHn_Init&lt;br&gt;TMHn_Start&lt;br&gt;TMHn_Stop&lt;br&gt;TMHn_ChangeTimerCondition&lt;br&gt;TMHn_ChangeDuty&lt;br&gt;TMH1_CarrierOutputEnable&lt;br&gt;TMH1_CarrierOutputDisable</td>
</tr>
<tr>
<td></td>
<td>CG_timer_user.c</td>
<td>TM00_UserInit&lt;br&gt;TM5n_UserInit&lt;br&gt;TMHn_UserInit&lt;br&gt;MD_INTTMXn&lt;br&gt;MD_INTTM0n0&lt;br&gt;MD_INTTM5n&lt;br&gt;MD_INTMnHn</td>
</tr>
<tr>
<td></td>
<td>CG_timer.h</td>
<td>-</td>
</tr>
<tr>
<td>Watchdog Timer</td>
<td>CG_wdt.c</td>
<td>WDT_Restart</td>
</tr>
<tr>
<td></td>
<td>CG_wdt.h</td>
<td>-</td>
</tr>
<tr>
<td>Real-time Clock</td>
<td>CG_rtc.c</td>
<td>RTC_Init&lt;br&gt;RTC_PowerOff&lt;br&gt;RTC.CounterEnable&lt;br&gt;RTC.CounterDisable&lt;br&gt;RTC.SetHourSystem&lt;br&gt;RTC.CounterSet&lt;br&gt;RTC.CounterGet&lt;br&gt;RTC.ConstPeriodInterruptEnable&lt;br&gt;RTC.ConstPeriodInterruptDisable&lt;br&gt;RTC.AlarmEnable&lt;br&gt;RTC.AlarmDisable&lt;br&gt;RTC.AlarmSet&lt;br&gt;RTC.AlarmGet&lt;br&gt;RTC.IntervalStart&lt;br&gt;RTC.IntervalStop&lt;br&gt;RTC.IntervalInterruptEnable&lt;br&gt;RTC.IntervalInterruptDisable&lt;br&gt;RTC_RTC1HZ_OutputEnable&lt;br&gt;RTC_RTC1HZ_OutputDisable&lt;br&gt;RTC_RTCCL_OutputEnable&lt;br&gt;RTC_RTCCL_OutputDisable&lt;br&gt;RTC_RTCDIV_OutputEnable&lt;br&gt;RTC_RTCDIV_OutputDisable&lt;br&gt;RTC.ChangeCorrectionValue</td>
</tr>
<tr>
<td></td>
<td>CG_rtc_user.c</td>
<td>RTC_UserInit&lt;br&gt;RTC.ConstPeriodInterruptCallback</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>Source File Name</td>
<td>Names of API Functions Included</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td><strong>Real-time Clock</strong></td>
<td>CG_rtc_user.c</td>
<td>RTC_AlarmInterruptCallback</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_INTRTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_INTRTCI</td>
</tr>
<tr>
<td>CG_rtc.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock Output</strong></td>
<td>CG_pcl.c</td>
<td>PCL_Init</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCL_Start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCL_Stop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCL_ChangeFreq</td>
</tr>
<tr>
<td>CG_pcl_user.c</td>
<td></td>
<td>PCL_UserInit</td>
</tr>
<tr>
<td>CG_pcl.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LVI</strong></td>
<td>CG_lvi.c</td>
<td>LVI_Init</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVI InterruptModeStart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVI_ResetModeStart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVI_Stop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LVI_SetLVILevel</td>
</tr>
<tr>
<td>CG_lvi_user.c</td>
<td></td>
<td>LVI_UserInit</td>
</tr>
<tr>
<td>CG_lvi.h</td>
<td></td>
<td>MD_INTRVI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This appendix describes the API functions output by Code Generator.

C.1 Overview

Below are the naming conventions for API functions output by Code Generator.

- Macro names are in ALL CAPS. The number in front of the macro name is a hexadecimal value; this is the same value as the macro value.
- Local variable names are in all lower case.
- Global variable names start with a "g" and use Camel Case.
- Names of pointers to global variables start with a "gp" and use Camel Case.
- Names of elements in enum statements are in ALL CAPS.

C.2 Output Function

Below is a list of API functions output by Code Generator.

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>CLOCK_Init</td>
<td>Performs initialization required to control the clock generator, on-chip debug, and etc. .</td>
</tr>
<tr>
<td></td>
<td>CLOCK_UserInit</td>
<td>Performs user-defined initialization relating to the clock generator, on-chip debug, and etc. .</td>
</tr>
<tr>
<td></td>
<td>CG_ReadResetSource</td>
<td>Performs processing in response to RESET signal.</td>
</tr>
<tr>
<td></td>
<td>CG_ChangeClockMode</td>
<td>Changes the CPU clock/peripheral hardware clock.</td>
</tr>
<tr>
<td></td>
<td>CG_ChangeFrequency</td>
<td>Changes the division ratio of the CPU clock/peripheral hardware clock.</td>
</tr>
<tr>
<td></td>
<td>CG_SelectPowerSaveMode</td>
<td>Configures the CPU's standby function.</td>
</tr>
<tr>
<td></td>
<td>CG_SelectStabTime</td>
<td>Configures the oscillation stabilization time of the X1 clock.</td>
</tr>
<tr>
<td></td>
<td>CG_ChangePllMode</td>
<td>Controls the operation of PLL function.</td>
</tr>
<tr>
<td>Port</td>
<td>PORT_Init</td>
<td>Performs initialization necessary to control port functions.</td>
</tr>
<tr>
<td></td>
<td>PORT_UserInit</td>
<td>Performs user-defined initialization relating to the port.</td>
</tr>
<tr>
<td></td>
<td>PORT_ChangePmnInput</td>
<td>Switches the pin's I/O mode from output mode to input mode.</td>
</tr>
<tr>
<td></td>
<td>PORT_ChangePmnOutput</td>
<td>Switches the pin's I/O mode from input mode to output mode.</td>
</tr>
<tr>
<td>Interrupt</td>
<td>INTP_Init</td>
<td>Performs initialization necessary to control the external interrupt INTPn functions.</td>
</tr>
<tr>
<td></td>
<td>INTP_UserInit</td>
<td>Performs user-defined initialization relating to the external interrupt INTPn functions.</td>
</tr>
<tr>
<td></td>
<td>KEY_Init</td>
<td>Performs initialization necessary to control the key interrupt INTKR functions.</td>
</tr>
<tr>
<td></td>
<td>KEY_UserInit</td>
<td>Performs user-defined initialization relating to the key interrupt INTKR functions.</td>
</tr>
<tr>
<td></td>
<td>INT_MaskableInterruptEnable</td>
<td>Disables/enables the acceptance of the maskable interrupts.</td>
</tr>
<tr>
<td></td>
<td>INTPn_Disable</td>
<td>Disables the acceptance of the maskable interrupts INTPn (external interrupt requests).</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>API Function Name</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Interrupt</td>
<td>INTPn_Enable</td>
<td>Enables the acceptance of the maskable interrupts INTPn (external requests).</td>
</tr>
<tr>
<td></td>
<td>KEY_Disable</td>
<td>Disables the acceptance of the key interrupts INTKR.</td>
</tr>
<tr>
<td></td>
<td>KEY_Enable</td>
<td>Enables the acceptance of the key interrupts INTKR.</td>
</tr>
<tr>
<td>Serial</td>
<td>UART6_Init</td>
<td>Performs initialization of the serial interface (UART6) channel.</td>
</tr>
<tr>
<td></td>
<td>UART6_UserInit</td>
<td>Performs user-defined initialization of the serial interface (UART6).</td>
</tr>
<tr>
<td></td>
<td>UART6_Start</td>
<td>Sets UART communication to standby mode.</td>
</tr>
<tr>
<td></td>
<td>UART6_Stop</td>
<td>Ends UART communication.</td>
</tr>
<tr>
<td></td>
<td>UART6_SendData</td>
<td>Starts UART data transmission.</td>
</tr>
<tr>
<td></td>
<td>UART6_ReceiveData</td>
<td>Starts UART data reception.</td>
</tr>
<tr>
<td></td>
<td>UART6_SendEndCallback</td>
<td>Performs processing in response to the UART transmission complete interrupt INTST6.</td>
</tr>
<tr>
<td></td>
<td>UART6_ReceiveEndCallback</td>
<td>Performs processing in response to the UART reception complete interrupt INTSR6.</td>
</tr>
<tr>
<td></td>
<td>UART6_SoftOverRunCallback</td>
<td>Performs processing in response to the UART reception complete interrupt INTSR6.</td>
</tr>
<tr>
<td></td>
<td>UART6_ErrorCallback</td>
<td>Performs processing in response to the UART communication error interrupt INTSRE6.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_Init</td>
<td>Performs initialization of the serial interface (CSI1n) channel.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_UserInit</td>
<td>Performs user-defined initialization of the serial interface (CSI1n).</td>
</tr>
<tr>
<td></td>
<td>CSI1n_Start</td>
<td>Sets CSI1n communication to standby mode.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_Stop</td>
<td>Ends CSI1n communication.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_ReceiveData</td>
<td>Starts CSI1n data reception.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_SendReceiveData</td>
<td>Starts CSI1n data transmission/reception.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_SendEndCallback</td>
<td>Performs processing in response to the CSI1n communication complete interrupt INTCSI1n.</td>
</tr>
<tr>
<td></td>
<td>CSI1n_ReceiveEndCallback</td>
<td>Performs processing in response to the CSI1n communication complete interrupt INTCSI1n.</td>
</tr>
<tr>
<td></td>
<td>IICA_Init</td>
<td>Performs initialization of the serial interface (IICA).</td>
</tr>
<tr>
<td></td>
<td>IICA_UserInit</td>
<td>Performs user-defined initialization of the serial interface (IICA).</td>
</tr>
<tr>
<td></td>
<td>IICA_Stop</td>
<td>Ends IICA communication.</td>
</tr>
<tr>
<td></td>
<td>IICA_MasterSendStart</td>
<td>Starts IICA master transmission.</td>
</tr>
<tr>
<td></td>
<td>IICA_MasterReceiveStart</td>
<td>Starts IICA master reception.</td>
</tr>
<tr>
<td></td>
<td>IICA_StopCondition</td>
<td>Generates stop conditions.</td>
</tr>
<tr>
<td></td>
<td>IICA_MasterSendEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td></td>
<td>IICA_MasterReceiveEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>API Function Name</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Serial</td>
<td>IICA_MasterErrorCallback</td>
<td>Performs processing in response to detection of error in IICA master communication.</td>
</tr>
<tr>
<td></td>
<td>IICA_SlaveSendStart</td>
<td>Starts IICA slave transmission.</td>
</tr>
<tr>
<td></td>
<td>IICA_SlaveReceiveStart</td>
<td>Starts IICA slave reception.</td>
</tr>
<tr>
<td></td>
<td>IICA_SlaveSendEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td></td>
<td>IICA_SlaveReceiveEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td></td>
<td>IICA_SlaveErrorCallback</td>
<td>Performs processing in response to detection of error in IICA slave communication.</td>
</tr>
<tr>
<td></td>
<td>IICA_GetStopConditionCallback</td>
<td>Performs processing in response to detection of stop condition in IICA slave communication.</td>
</tr>
<tr>
<td>Operational Amplifier</td>
<td>OPAMP_Init</td>
<td>Performs initialization necessary to control operational amplifier functions.</td>
</tr>
<tr>
<td></td>
<td>OPAMP_UserInit</td>
<td>Performs user-defined initialization relating to the operational amplifier.</td>
</tr>
<tr>
<td></td>
<td>PGA_Start</td>
<td>Starts the operation of operational amplifier (PGA mode).</td>
</tr>
<tr>
<td></td>
<td>PGA_Stop</td>
<td>Ends the operation of operational amplifier (PGA mode).</td>
</tr>
<tr>
<td></td>
<td>PGA_ChangePGAFactor</td>
<td>Sets the input voltage amplification factor of a operational amplifier (PGA mode).</td>
</tr>
<tr>
<td></td>
<td>AMP_Start</td>
<td>Starts the operation of operational amplifier (single AMP mode).</td>
</tr>
<tr>
<td></td>
<td>AMP_Stop</td>
<td>Ends the operation of operational amplifier (single AMP mode).</td>
</tr>
<tr>
<td></td>
<td>AMPn_Start</td>
<td>Starts the operation of operational amplifier n (single AMP mode).</td>
</tr>
<tr>
<td></td>
<td>AMPn_Stop</td>
<td>Ends the operation of operational amplifier n (single AMP mode).</td>
</tr>
<tr>
<td>Comparator</td>
<td>Comparator_Init</td>
<td>Performs initialization necessary to control comparator functions.</td>
</tr>
<tr>
<td></td>
<td>Comparator_UserInit</td>
<td>Performs user-defined initialization relating to the comparator.</td>
</tr>
<tr>
<td></td>
<td>Comparatorn_Start</td>
<td>Starts the operation of comparator n.</td>
</tr>
<tr>
<td></td>
<td>Comparatorn_Stop</td>
<td>Ends the operation of comparator n.</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>AD_Init</td>
<td>Performs initialization necessary to control A/D converter functions.</td>
</tr>
<tr>
<td></td>
<td>AD_UserInit</td>
<td>Performs user-defined initialization relating to the A/D converter.</td>
</tr>
<tr>
<td></td>
<td>AD_ComparatorOn</td>
<td>Enables operation of voltage converter.</td>
</tr>
<tr>
<td></td>
<td>AD_ComparatorOff</td>
<td>Disables operation of voltage converter.</td>
</tr>
<tr>
<td></td>
<td>AD_Start</td>
<td>Starts A/D conversion.</td>
</tr>
<tr>
<td></td>
<td>AD_Stop</td>
<td>Ends A/D conversion.</td>
</tr>
<tr>
<td></td>
<td>AD_SelectADChannel</td>
<td>Configures the analog voltage input pin for A/D conversion.</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>API Function Name</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>AD_Read</td>
<td>Reads the results of A/D conversion (10 bits).</td>
</tr>
<tr>
<td></td>
<td>AD_ReadByte</td>
<td>Reads the results of A/D conversion (8 bits; most significant 8 bits of 10-bit resolution).</td>
</tr>
<tr>
<td>Timer</td>
<td>TMX_Init</td>
<td>Performs initialization necessary to control 16-bit timer Xn functions.</td>
</tr>
<tr>
<td></td>
<td>TMXn_Start</td>
<td>Starts the count for 16-bit timer Xn.</td>
</tr>
<tr>
<td></td>
<td>TMXn_Stop</td>
<td>Ends the count for 16-bit timer Xn.</td>
</tr>
<tr>
<td></td>
<td>TMXn_ChangeDuty</td>
<td>Changes the duty ratio of the PWM signal single-output to the TOX0n pin.</td>
</tr>
<tr>
<td></td>
<td>TMXn_ChangeDualDuty</td>
<td>Changes the duty ratio of the PWM signal dual-output to the TOX0n pin.</td>
</tr>
<tr>
<td></td>
<td>TMX_EnableHighImpedanceState</td>
<td>Begins high impedance output of the 16-bit timer Xn.</td>
</tr>
<tr>
<td></td>
<td>TMX_DisableHighImpedanceState</td>
<td>Ends high impedance output of the 16-bit timer Xn.</td>
</tr>
<tr>
<td></td>
<td>TM00_Init</td>
<td>Performs initialization necessary to control 16-bit timer/event counter 00 functions.</td>
</tr>
<tr>
<td></td>
<td>TM00_UserInit</td>
<td>Performs user-defined initialization relating to the 16-bit timer/event counter 00.</td>
</tr>
<tr>
<td></td>
<td>TM00_Start</td>
<td>Starts the count for 16-bit timer/event counter 00.</td>
</tr>
<tr>
<td></td>
<td>TM00_Stop</td>
<td>Ends the count for 16-bit timer/event counter 00.</td>
</tr>
<tr>
<td></td>
<td>TM00_ChangeTimerCondition</td>
<td>Changes the value of capture/compare control register 00 (CRC00).</td>
</tr>
<tr>
<td></td>
<td>TM00_GetFreeRunningValue</td>
<td>Captures the content of the capture register (CR0n0).</td>
</tr>
<tr>
<td></td>
<td>TM00_SoftwareTriggerOn</td>
<td>Generates the trigger (software trigger) for one-shot pulse output.</td>
</tr>
<tr>
<td></td>
<td>TM00_ChangeDuty</td>
<td>Changes the duty ratio of the signal output to the TO00 pin.</td>
</tr>
<tr>
<td></td>
<td>TM00_GetPulseWidth</td>
<td>Captures the high/low-level width measured for the signal (pulses) input to the TI0n0 pin.</td>
</tr>
<tr>
<td></td>
<td>TM5n_Init</td>
<td>Performs initialization necessary to control 8-bit timer/event counter 5n functions.</td>
</tr>
<tr>
<td></td>
<td>TM5n_UserInit</td>
<td>Performs user-defined initialization relating to the 8-bit timer/event counter 5n.</td>
</tr>
<tr>
<td></td>
<td>TM5n_Start</td>
<td>Starts the count for 8-bit timer/event counter 5n.</td>
</tr>
<tr>
<td></td>
<td>TM5n_Stop</td>
<td>Ends the count for 8-bit timer/event counter 5n.</td>
</tr>
<tr>
<td></td>
<td>TM5n_ChangeTimerCondition</td>
<td>Changes the value of 8-bit timer compare register 5n (CR5n).</td>
</tr>
<tr>
<td></td>
<td>TM5n_ChangeDuty</td>
<td>Changes the duty ratio of the PWM signal output to the TO5n pin.</td>
</tr>
<tr>
<td></td>
<td>TMHn_Init</td>
<td>Performs initialization necessary to control 8-bit timer Hn functions.</td>
</tr>
<tr>
<td></td>
<td>TMHn_UserInit</td>
<td>Performs user-defined initialization relating to the 8-bit timer Hn.</td>
</tr>
<tr>
<td></td>
<td>TMHn_Start</td>
<td>Starts the count for 8-bit timer Hn.</td>
</tr>
<tr>
<td></td>
<td>TMHn_Stop</td>
<td>Ends the count for 8-bit timer Hn.</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>API Function Name</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Timer</td>
<td>TMHn_ChangeTimerCondition</td>
<td>Changes the value of 8-bit timer H compare register 0n/1n (CMP0n/CMP1n).</td>
</tr>
<tr>
<td></td>
<td>TMHn_ChangeDuty</td>
<td>Changes the duty ratio of the PWM signal output to the TOHn pin.</td>
</tr>
<tr>
<td></td>
<td>TMH1_CarrierOutputEnable</td>
<td>Begins carrier pulse output of the 8-bit timer H1 (carrier generator mode).</td>
</tr>
<tr>
<td></td>
<td>TMH1_CarrierOutputDisable</td>
<td>Ends carrier pulse output of the 8-bit timer H1 (carrier generator mode).</td>
</tr>
<tr>
<td>Watchdog Timer</td>
<td>WDT_Restart</td>
<td>Clears the watchdog timer counter and resumes counting.</td>
</tr>
<tr>
<td>Real-time Clock</td>
<td>RTC_Init</td>
<td>Performs initialization necessary to control real-time counter functions.</td>
</tr>
<tr>
<td></td>
<td>RTC_UserInit</td>
<td>Performs user-defined initialization relating to the real-time counter.</td>
</tr>
<tr>
<td></td>
<td>RTC_PowerOff</td>
<td>Halts the clock supplied to the real-time counter.</td>
</tr>
<tr>
<td></td>
<td>RTC_CounterEnable</td>
<td>Starts the count of the real-time counter (year, month, weekday, day, hour, minute, second).</td>
</tr>
<tr>
<td></td>
<td>RTC_CounterDisable</td>
<td>Ends the count of the real-time counter (year, month, weekday, day, hour, minute, second).</td>
</tr>
<tr>
<td></td>
<td>RTC_SetHourSystem</td>
<td>Sets the clock type (12-hour or 24-hour clock) of the real-time counter.</td>
</tr>
<tr>
<td></td>
<td>RTC_CounterSet</td>
<td>Sets the counter value (year, month, weekday, day, hour, minute, second) of the real-time counter.</td>
</tr>
<tr>
<td></td>
<td>RTC_CounterGet</td>
<td>Reads the counter value (year, month, weekday, day, hour, minute, second) of the real-time counter.</td>
</tr>
<tr>
<td></td>
<td>RTC_ConstPeriodInterruptEnable</td>
<td>Sets the cycle of the interrupts INTRTC, then starts the cyclic interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_ConstPeriodInterruptDisable</td>
<td>Ends the cyclic interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_ConstPeriodInterruptCallback</td>
<td>Performs processing in response to the cyclic interrupt INTRTC.</td>
</tr>
<tr>
<td></td>
<td>RTC_AlarmEnable</td>
<td>Starts the alarm interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_AlarmDisable</td>
<td>Ends the alarm interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_AlarmSet</td>
<td>Sets the alarm conditions (weekday, hour, minute).</td>
</tr>
<tr>
<td></td>
<td>RTC_AlarmGet</td>
<td>Reads the alarm conditions (weekday, hour, minute).</td>
</tr>
<tr>
<td></td>
<td>RTC_AlarmInterruptCallback</td>
<td>Performs processing in response to the alarm interrupt INTRTC.</td>
</tr>
<tr>
<td></td>
<td>RTC_IntervalStart</td>
<td>Starts the interval interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_IntervalStop</td>
<td>Ends the interval interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_IntervalInterruptEnable</td>
<td>Sets the cycle of the interrupts, then starts the interval interrupt INTRTCI function.</td>
</tr>
<tr>
<td></td>
<td>RTC_IntervalInterruptDisable</td>
<td>Ends the interval interrupt function.</td>
</tr>
<tr>
<td></td>
<td>RTC_RTC1HZ_OutputEnable</td>
<td>Enables output of the real-time counter correction clock (1 Hz) to the RTC1HZ pin.</td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>API Function Name</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Real-time Clock</td>
<td>RTC_RTC1HZ_OutputDisable</td>
<td>Disables output of the real-time counter correction clock (1 Hz) to the RTC1HZ pin.</td>
</tr>
<tr>
<td></td>
<td>RTC_RTCCL_OutputEnable</td>
<td>Enables output of the real-time counter clock (32 kHz source) to the RTCCL pin.</td>
</tr>
<tr>
<td></td>
<td>RTC_RTCCL_OutputDisable</td>
<td>Disables output of the real-time counter clock (32 kHz source) to the RTCCL pin.</td>
</tr>
<tr>
<td></td>
<td>RTC_RTCDIV_OutputEnable</td>
<td>Enables output of the real-time counter clock (32 kHz cycle) to the RTCDIV pin.</td>
</tr>
<tr>
<td></td>
<td>RTC_RTCDIV_OutputDisable</td>
<td>Disables output of the real-time counter clock (32 kHz cycle) to the RTCDIV pin.</td>
</tr>
<tr>
<td></td>
<td>RTC_ChangeCorrectionValue</td>
<td>Changes the timing and correction value for correcting clock errors.</td>
</tr>
<tr>
<td>Clock Output</td>
<td>PCL_Init</td>
<td>Performs initialization necessary to control clock output control circuit functions.</td>
</tr>
<tr>
<td></td>
<td>PCL_UserInit</td>
<td>Performs user-defined initialization relating to the clock output control circuits.</td>
</tr>
<tr>
<td></td>
<td>PCL_Start</td>
<td>Starts clock output.</td>
</tr>
<tr>
<td></td>
<td>PCL_Stop</td>
<td>Ends clock output.</td>
</tr>
<tr>
<td></td>
<td>PCL_ChangeFreq</td>
<td>Changes the output clock to the PCL pin.</td>
</tr>
<tr>
<td>LVI</td>
<td>LVI_Init</td>
<td>Performs initialization necessary to control low-voltage detector functions.</td>
</tr>
<tr>
<td></td>
<td>LVI_UserInit</td>
<td>Performs user-defined initialization relating to the low-voltage detector.</td>
</tr>
<tr>
<td></td>
<td>LVI_interruptModeStart</td>
<td>Starts low-voltage detection (when in interrupt generation mode).</td>
</tr>
<tr>
<td></td>
<td>LVI_ResetModeStart</td>
<td>Starts low-voltage detection (when in internal reset mode).</td>
</tr>
<tr>
<td></td>
<td>LVI_Stop</td>
<td>Stops low-voltage detection.</td>
</tr>
<tr>
<td></td>
<td>LVI_SetLVILevel</td>
<td>Sets the low-voltage detection level.</td>
</tr>
</tbody>
</table>
C.3 Function Reference

This section describes the API functions output by Code Generator, using the following notation format.

Figure C-1. Notation Format of API Functions

(1) **Name**
   Indicates the name of the API function.

(2) **Outline**
   Outlines the functions of the API function.

(3) **[Classification]**
   Indicates the name of the C source file to which the API function is output.

(4) **[Syntax]**
   Indicates the format to be used when describing an API function to be called in C language.

(5) **[Argument(s)]**
   
<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

(6) **[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>......</td>
<td>...</td>
</tr>
</tbody>
</table>

(7) **[Example]**

............................

............................
(5) [Argument(s)]
API function arguments are explained in the following format.

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
</tr>
</tbody>
</table>

(a) I/O
Argument classification
I ... Input argument
O ... Output argument

(b) Argument
Argument data type

(c) Description
Description of argument

(6) [Return value]
API function return value is explained in the following format.

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
</tr>
</tbody>
</table>

(a) Macro
Macro of return value

(b) Description
Description of return value

(7) [Example]
Shows an example of the API function in use.
C.3.1 System

Below is a list of API functions output by Code Generator for system use.

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOCK_Init</td>
<td>Performs initialization required to control the clock generator, on-chip debug, and etc. .</td>
</tr>
<tr>
<td>CLOCK_UserInit</td>
<td>Performs user-defined initialization relating to the clock generator, on-chip debug, and etc. .</td>
</tr>
<tr>
<td>CG_ReadResetSource</td>
<td>Performs processing in response to RESET signal.</td>
</tr>
<tr>
<td>CG_ChangeClockMode</td>
<td>Changes the CPU clock/peripheral hardware clock.</td>
</tr>
<tr>
<td>CG_ChangeFrequency</td>
<td>Changes the division ratio of the CPU clock/peripheral hardware clock.</td>
</tr>
<tr>
<td>CG_SelectPowerSaveMode</td>
<td>Configures the CPU's standby function.</td>
</tr>
<tr>
<td>CG_SelectStabTime</td>
<td>Configures the oscillation stabilization time of the X1 clock.</td>
</tr>
<tr>
<td>CG_ChangePllMode</td>
<td>Controls the operation of PLL function.</td>
</tr>
</tbody>
</table>
**CLOCK_Init**

Performs initialization required to control the clock generator, on-chip debug and etc.

**[Classification]**

CG_system.c

**[Syntax]**

```
void CLOCK_Init ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
CLOCK_UserInit

Performs user-defined initialization relating to the clock generator, on-chip debug, and etc.

Remark This API function is called as the CLOCK_Init callback routine.

[Classification]
CG_system_user.c

[Syntax]

```c
void CLOCK_UserInit ( void );
```

[Argument(s)]
None.

[Return value]
None.
CG_ReadResetSource

Performs processing in response to RESET signal.

[Classification]
CG_system_user.c

[Syntax]

```c
void CG_ReadResetSource ( void );
```

[Argument(s)]
None.

[Return value]
None.

[Example]
Below are examples of the different processes executing depending on the RESET signal trigger.

[CG_Systeminit.c]

```c
void systeminit ( void ) {
    CG_ReadResetSource ();      /* Processes executed by RESET signal trigger */
    ......                      
}
```

[CG_system_user.c]

```c
#include "CG_macrodriver.h"
void CG_ReadResetSource ( void ) {
    UCHAR flag = RESF;        /* Reset control flag register: Obtain RESF contents */
    if ( flag & 0x1 ) {       /* Trigger identification: Check LVIRF flag */
        ......                   /* Internal reset request by low-voltage detector */
    } else if ( flag & 0x10 ) { /* Trigger identification: Check WDTRF flag */
        ......                   /* Internal reset request by watchdog timer */
    }
    ......                      
}
```
Changes the CPU clock/peripheral hardware clock.

[Classification]

CG_system.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_ChangeClockMode ( enum ClockMode mode );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum ClockMode mode;</td>
<td>CPU clock (fCPU)/peripheral hardware clock (fPRS) type</td>
</tr>
</tbody>
</table>

- HIOCLK: fCPU/fPRS >> Internal high-speed oscillation clock
- HIOSYSCLK: fCPU >> Internal high-speed oscillation clock, fPRS >> High-speed system clock
- SYSX1CLK: fCPU/fPRS >> X1 clock
- SYSEXTCCLK: fCPU/fPRS >> External main system clock
- SUBXT1CLK: fCPU >> XT1 clock
- SUBEXTCLK: fCPU >> External subsystem clock

[Remark]

SUBXT1CLK and SUBEXTCLK can only be specified when the target device is a 78K0/KC2-L.

[Return value]

| Macro           | Description                                                        |
|-----------------|                                                                  |
| MD_OK           | Normal completion                                                 |
| MD_ERROR1       | Exit with error (abend)                                           |
|                 | - Cannot change fCPU/fPRS to the internal high-speed oscillation clock. |
| MD_ERROR2       | Exit with error (abend)                                           |
|                 | - Cannot change fCPU to the internal high-speed oscillation clock.  |
|                 | - Cannot change fPRS to the high-speed system clock.               |
| MD_ERROR3       | Exit with error (abend)                                           |
|                 | - Cannot change fCPU/fPRS to the X1 clock.                         |
| MD_ERROR4       | Exit with error (abend)                                           |
|                 | - Cannot change fCPU/fPRS to the external main system clock.       |
| MD_ERROR5       | Exit with error (abend) [Kx2-L]                                  |
|                 | - Cannot change fCPU to the XT1 clock.                             |
| MD_ERROR6       | Exit with error (abend) [Kx2-L]                                  |
|                 | - Cannot change fCPU to the external subsystem clock.              |
| MD_ARGERROR     | Invalid argument specification                                    |
Remark  The values MD_ERROR5 and MD_ERROR6 will only be returned when the target device is a 78K0/KC2-L.
CG_ChangeFrequency

Changes the division ratio of the CPU clock/peripheral hardware clock.

[Classification]
CG_system.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_ChangeFrequency ( enum CPUClock clock );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum CPUClock</td>
<td>Division ratio type</td>
</tr>
<tr>
<td></td>
<td>clock</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEMCLOCK:</td>
<td>fMAIN</td>
</tr>
<tr>
<td></td>
<td>SYSONEHALF:</td>
<td>fMAIN/2</td>
</tr>
<tr>
<td></td>
<td>SYSONEFOURTH:</td>
<td>fMAIN/4</td>
</tr>
<tr>
<td></td>
<td>SYSONEIGHTH:</td>
<td>fMAIN/8</td>
</tr>
<tr>
<td></td>
<td>SYSONESIXTEENTH:</td>
<td>fMAIN/16</td>
</tr>
</tbody>
</table>

Remark  "fMAIN" signifies the frequency of the main system clock.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
**CG_SelectPowerSaveMode**

Configures the CPU's standby function.

**[Classification]**

CG_system.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_SelectPowerSaveMode ( enum PSLevel level );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum PSLevel level;</td>
<td>Standby function type</td>
</tr>
</tbody>
</table>

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR</td>
<td>Exit with error (abend) [Kx2-L]</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

- If the CPU is operating by XT1 clock, then STOP mode cannot be specified.

**Remark**  The value MD_ERROR will only be returned when the target device is a 78K0/KC2-L.

**[Example]**

Below is an example of changing the standby function to "STOP mode".

**[CG_main.c]**

```c
#include "CG_macrodriver.h"
#include "CG_system.h"

void main ( void ) {
    MD_STATUS ret;
    .......
    TM00_Stop (); /* Stop count */
    ret = CG_SelectPowerSaveMode ( PSSTOP ); /* Change to STOP mode */
    if ( ret != MD_OK ) {
        while ( 1 );
    }
}
```
TM00_Init ();    /* Initialize TM00 */
TM00_Start ();   /* Start count */

.......

}
CG_SelectStabTime

Configures the oscillation stabilization time of the X1 clock.

[Classification]
CG_system.c

[Syntax]

```
#include    "CG_macrodriver.h"
#include    "CG_system.h"
MD_STATUS   CG_SelectStabTime ( enum StabTime waittime );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum StabTime waittime;</td>
<td>Oscillation stabilization time type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STLEVEL0: 2^11/fx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STLEVEL1: 2^13/fx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STLEVEL2: 2^14/fx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STLEVEL3: 2^15/fx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STLEVEL4: 2^16/fx</td>
</tr>
</tbody>
</table>

Remark  "fx" signifies the frequency of the X1 clock.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
CG_ChangePllMode

Controls the operation of PLL function.

[Classification]
CG_system.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_system.h"

MD_STATUS CG_ChangePllMode ( enum PllMode pllmode );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum PllMode pllmode;</td>
<td>Control of operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSPLLON: Enable operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSPLLOFF: Stop operation</td>
</tr>
</tbody>
</table>

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
C.3.2 Port

Below is a list of API functions output by Code Generator for port use.

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT_Init</td>
<td>Performs initialization necessary to control port functions.</td>
</tr>
<tr>
<td>PORT_UserInit</td>
<td>Performs user-defined initialization relating to the port.</td>
</tr>
<tr>
<td>PORT_ChangePmnInput</td>
<td>Switches the pin's I/O mode from output mode to input mode.</td>
</tr>
<tr>
<td>PORT_ChangePmnOutput</td>
<td>Switches the pin's I/O mode from input mode to output mode.</td>
</tr>
</tbody>
</table>
PORT_Init

Performs initialization necessary to control port functions.

[Classification]
CG_port.c

[Syntax]

```c
void PORT_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
PORT_UserInit

Performs user-defined initialization relating to the port.

Remark This API function is called as the PORT_Init callback routine.

[Classification]
CG_port_user.c

[Syntax]

```c
void PORT_UserInit ( void );
```

[Argument(s)]
None.

[Return value]
None.
Switches the pin’s I/O mode from output mode to input mode.

[Classification]

CG_port.c

[Syntax]

The format for specifying this API function differs according to whether the target pin has built-in pull-up resistance/a SMBus input buffer.

- Built-in pull-up resistance: none; SMBus input buffer: none

```c
void PORT_ChangePmnInput ( void );
```

- Built-in pull-up resistance: yes; SMBus input buffer: none

```c
#include    "CG_macrodriver.h"
void    PORT_ChangePmnInput ( BOOL enablepu );
```

- Built-in pull-up resistance: yes; SMBus input buffer: yes

```c
#include    "CG_macrodriver.h"
void    PORT_ChangePmnInput ( BOOL enablepu, BOOL enablesmbus );
```

Remark  \( mn \) is the port number.

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>BOOL enablepu;</td>
<td>Built-in pull-up resistance used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_TRUE: Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_FALSE: No</td>
</tr>
<tr>
<td>I</td>
<td>BOOL enablesmbus;</td>
<td>Input buffer type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_TRUE: SMBus input buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_FALSE: Normal input buffer</td>
</tr>
</tbody>
</table>

[Return value]

None.

[Example 1]

Below is shown an example where pin P00 (built-in pull-up resistance: yes; SMBus input buffer: none) is changed as follows:

- I/O mode type: Input mode
- Built-in pull-up resistance used: Yes
Below is shown an example where pin P00 (built-in pull-up resistance: yes; SMBus input buffer: none) is changed as follows:

I/O mode type: Input mode
Built-in pull-up resistance used: No

```c
#include    "CG_macrodriver.h"
void main ( void ) {
        //......
        PORT_ChangeP00Input ( MD_TRUE );    /* Switch I/O mode */
        //......
}
```

[Example 2]

Below is shown an example where pin P04 (built-in pull-up resistance: yes; SMBus input buffer: yes) is changed as follows:

I/O mode type: Input mode
Built-in pull-up resistance used: No
Input buffer type: SMBus input buffer

```c
#include    "CG_macrodriver.h"
void main ( void ) {
        //......
        PORT_ChangeP04Input ( MD_FALSE, MD_TRUE );  /* Switch I/O mode */
        //......
}
```

[Example 3]
Switches the pin's I/O mode from input mode to output mode.

[Classification]
CG_port.c

[Syntax]
The format for specifying this API function differs according to whether the target pin conducts N-ch open drain output.

- N-ch open drain output: none

```c
#include "CG_macrodriver.h"
void PORT_ChangePmnOutput ( BOOL initialValue );
```

- N-ch open drain output: yes

```c
#include "CG_macrodriver.h"
void PORT_ChangePmnOutput ( BOOL enableNch, BOOL initialValue );
```

Remark  

nm is the port number.

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>BOOL enableNch</td>
<td>Output mode type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_TRUE: N-ch open drain output (VDD withstand voltage) mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_FALSE: Normal output mode</td>
</tr>
<tr>
<td>I</td>
<td>BOOL initialValue</td>
<td>Initial output value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_SET: Output HIGH level &quot;1&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_CLEAR: Output LOW level &quot;0&quot;</td>
</tr>
</tbody>
</table>

[Return value]

None.

[Example 1]

Below is shown an example where pin P00 (N-ch open drain output: none) is changed as follows:

I/O mode type: Output mode
Initial output value: Output HIGH level "1"

[CG_main.c]

```c
#include "CG_macrodriver.h"
void main ( void ) {
    ......
    PORT_ChangeP00Output ( MD_SET ); /* Switch I/O mode */
}
```
[Example 2]  
Below is shown an example where pin P04 (N-ch open drain output: yes) is changed as follows:
- I/O mode type:  Output mode
- Output mode type: N-ch open drain output (VDD withstand voltage) mode
- Initial output value: Output LOW level "0"

[CG_main.c]

#include "CG_macrodriver.h"
void main ( void ) {
    .......
    PORT_ChangeP04Output ( MD_TRUE, MD_CLEAR ); /* Switch I/O mode */
    .......
}
### Interrupt

Below is a list of API functions output by Code Generator for interrupt and key interrupt use.

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTP_Init</td>
<td>Performs initialization necessary to control the external interrupt INTPn functions.</td>
</tr>
<tr>
<td>INTP_UserInit</td>
<td>Performs user-defined initialization relating to the external interrupt INTPn functions.</td>
</tr>
<tr>
<td>KEY_Init</td>
<td>Performs initialization necessary to control the key interrupt INTKR functions.</td>
</tr>
<tr>
<td>KEY_UserInit</td>
<td>Performs user-defined initialization relating to the key interrupt INTKR functions.</td>
</tr>
<tr>
<td>INT_MaskableInterruptEnable</td>
<td>Disables/enables the acceptance of the maskable interrupts.</td>
</tr>
<tr>
<td>INTPn_Disable</td>
<td>Disables the acceptance of the maskable interrupts INTPn (external interrupt requests).</td>
</tr>
<tr>
<td>INTPn_Enable</td>
<td>Enables the acceptance of the maskable interrupts INTPn (external interrupt requests).</td>
</tr>
<tr>
<td>KEY_Disable</td>
<td>Disables the acceptance of the key interrupts INTKR.</td>
</tr>
<tr>
<td>KEY_Enable</td>
<td>Enables the acceptance of the key interrupts INTKR.</td>
</tr>
</tbody>
</table>
INTP_Init

Performs initialization necessary to control the external interrupt INTPn functions.

[Classification]
CG_int.c

[Syntax]

```c
void INTP_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
**INTP_UserInit**

Performs user-defined initialization relating to the external interrupt INTPn functions.

**Remark**  This API function is called as the INTP_Init callback routine.

**[Classification]**

CG_int_user.c

**[Syntax]**

```c
void INTP_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
KEY_Init

Performs initialization necessary to control the key interrupt INTKR functions.

[Classification]
CG_int.c

[Syntax]

```c
void KEY_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
KEY_UserInit

Performs user-defined initialization relating to the key interrupt INTKR functions.

Remark  This API function is called as the KEY_Init callback routine.

[Classification]
CG_int_user.c

[Syntax]

```c
void KEY_UserInit ( void );
```

[Argument(s)]
None.

[Return value]
None.
**INT_MaskableInterruptEnable**

Disables/enables the acceptance of the maskable interrupts.

**[Classification]**

CG_int.c

**[Syntax]**

```c
#include    "CG_macrodriver.h"
#include    "CG_int.h"

MD_STATUS   INT_MaskableInterruptEnable ( enum MaskableSource name, BOOL enableflag );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum MaskableSource name;</td>
<td>Maskable interrupt type</td>
</tr>
<tr>
<td></td>
<td>INT_xxx: Maskable interrupt</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>BOOL enableflag;</td>
<td>Acceptance enabled/disabled</td>
</tr>
<tr>
<td></td>
<td>MD_TRUE: Acceptance enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MD_FALSE: Acceptance disabled</td>
<td></td>
</tr>
</tbody>
</table>

**[Remark]**

See the header file CG_int.h for details about the maskable interrupt type INT_xxx.

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

**[Example 1]**

Below is an example of disabling acceptance of the maskable interrupt INTP0.

**[CG_main.c]**

```c
#include    "CG_macrodriver.h"
#include    "CG_int.h"

void main ( void ) {
    ........
    INT_MaskableInterruptEnable ( INT_INTP0, MD_FALSE ); /* Disable acceptance of maskable interrupt INTP0 */
    ........
}
```
[Example 2]

Below is an example of enabling acceptance of the maskable interrupt INTP0.

[CG_main.c]

```c
#include "CG_macrodriver.h"
#include "CG_int.h"

void main ( void ) {
    ...... 
    INT_MaskableInterruptEnable ( INT_INTP0, MD_TRUE ); /* Enable acceptance of maskable interrupt INTP0 */ 
    ...... 
}
```
Disables the acceptance of the maskable interrupts INTPn (external interrupt requests).

[Classification]
CG_int.c

[Syntax]

```c
void INTPn_Disable ( void );
```

Remark \( n \) is the interrupt factor number.

[Argument(s)]
None.

[Return value]
None.
**INTPn_Enable**

Enables the acceptance of the maskable interrupts INTPn (external interrupt requests).

**[Classification]**

CG_int.c

**[Syntax]**

```c
void INTPn_Enable ( void );
```

**Remark**  

n is the interrupt factor number.

**[Argument(s)]**

None.

**[Return value]**

None.
<table>
<thead>
<tr>
<th><strong>KEY_Disable</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Disables the acceptance of the key interrupts INTKR.</td>
</tr>
</tbody>
</table>

**[Classification]**

CG_int.c

**[Syntax]**

```c
void KEY_Disable ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**KEY_Enable**

Enables the acceptance of the key interrupts INTKR.

**[Classification]**

CG_int.c

**[Syntax]**

```c
void KEY_Enable ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
C.3.4 Serial

Below is a list of API functions output by Code Generator for serial array unit and serial interface use.

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART6_Init</td>
<td>Performs initialization of the serial interface (UART6) channel.</td>
</tr>
<tr>
<td>UART6_UserInit</td>
<td>Performs user-defined initialization of the serial interface (UART6).</td>
</tr>
<tr>
<td>UART6_Start</td>
<td>Sets UART communication to standby mode.</td>
</tr>
<tr>
<td>UART6_Stop</td>
<td>Ends UART communication.</td>
</tr>
<tr>
<td>UART6_SendData</td>
<td>Starts UART data transmission.</td>
</tr>
<tr>
<td>UART6_ReceiveData</td>
<td>Starts UART data reception.</td>
</tr>
<tr>
<td>UART6_SendEndCallback</td>
<td>Performs processing in response to the UART transmission complete interrupt INTST6.</td>
</tr>
<tr>
<td>UART6_ReceiveEndCallback</td>
<td>Performs processing in response to the UART reception complete interrupt INTSR6.</td>
</tr>
<tr>
<td>UART6_SoftOverRunCallback</td>
<td>Performs processing in response to the serial transfer end interrupt INTSR6.</td>
</tr>
<tr>
<td>UART6_ErrorCallback</td>
<td>Performs processing in response to the UART communication error interrupt INTSRE6.</td>
</tr>
<tr>
<td>CSI1n_Init</td>
<td>Performs initialization of the serial interface (CSI1n) channel.</td>
</tr>
<tr>
<td>CSI1n_UserInit</td>
<td>Performs user-defined initialization of the serial interface (CSI1n).</td>
</tr>
<tr>
<td>CSI1n_Start</td>
<td>Sets CSI1n communication to standby mode.</td>
</tr>
<tr>
<td>CSI1n_Stop</td>
<td>Ends CSI1n communication.</td>
</tr>
<tr>
<td>CSI1n_ReceiveData</td>
<td>Starts CSI1n data reception.</td>
</tr>
<tr>
<td>CSI1n_SendReceiveData</td>
<td>Starts CSI1n data transmission/reception.</td>
</tr>
<tr>
<td>CSI1n_SendEndCallback</td>
<td>Performs processing in response to the CSI1n communication complete interrupt INTCSI1n.</td>
</tr>
<tr>
<td>CSI1n_ReceiveEndCallback</td>
<td>Performs processing in response to the CSI1n communication complete interrupt INTSCSI1n.</td>
</tr>
<tr>
<td>IICA_Init</td>
<td>Performs initialization of the serial interface (IICA).</td>
</tr>
<tr>
<td>IICA_UserInit</td>
<td>Performs user-defined initialization of the serial interface (IICA).</td>
</tr>
<tr>
<td>IICA_Stop</td>
<td>Ends IICA communication.</td>
</tr>
<tr>
<td>IICA_MasterSendStart</td>
<td>Starts IICA master transmission.</td>
</tr>
<tr>
<td>IICA_MasterReceiveStart</td>
<td>Starts IICA master reception.</td>
</tr>
<tr>
<td>IICA_StopCondition</td>
<td>Generates stop conditions.</td>
</tr>
<tr>
<td>IICA_MasterSendEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td>IICA_MasterReceiveEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td>IICA_MasterErrorCallback</td>
<td>Performs processing in response to detection of error in IICA master communication.</td>
</tr>
<tr>
<td>IICA_SlaveSendStart</td>
<td>Starts IICA slave transmission.</td>
</tr>
<tr>
<td>IICA_SlaveReceiveStart</td>
<td>Starts IICA slave reception.</td>
</tr>
<tr>
<td>API Function Name</td>
<td>Function</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IICA_SlaveSendEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td>IICA_SlaveReceiveEndCallback</td>
<td>Performs processing in response to the IICA communication complete interrupt INTIICA0.</td>
</tr>
<tr>
<td>IICA_SlaveErrorCallback</td>
<td>Performs processing in response to detection of error in IICA slave communication.</td>
</tr>
<tr>
<td>IICA_GetStopConditionCallback</td>
<td>Performs processing in response to detection of stop condition in IICA slave communication.</td>
</tr>
</tbody>
</table>
**UART6_Init**

Performs initialization of the serial interface (UART6) channel.

**[Classification]**

CG_serial.c

**[Syntax]**

```c
void UART6_Init ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**UART6_UserInit**

Performs user-defined initialization of the serial interface (UART6).

**Remark** This API function is called as the UART6_Init callback routine.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void UART6_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
UART6_Start

Sets UART communication to standby mode.

[Classification]
CG_serial.c

[Syntax]

```c
void UART6_Start ( void );
```

[Argument(s)]
None.

[Return value]
None.
**UART6_Stop**

Ends UART communication.

**[Classification]**

CG_serial.c

**[Syntax]**

```c
void UART6_Stop ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
UART6_SendData

Starts UART data transmission.

**Remarks**

1. This API function repeats the byte-level UART transmission from the buffer specified in parameter `txbuf` the number of times specified in parameter `txnum`.
2. When performing a UART transmission, `UART6_Start` must be called before this API function is called.

**[Classification]**

CG_serial.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
MD_STATUS UART6_SendData ( UCHAR *txbuf, USHORT txnum );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR *txbuf;</td>
<td>Pointer to a buffer storing the transmission data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT txnum;</td>
<td>Total amount of data to send</td>
</tr>
</tbody>
</table>

**Remark**

You can only set 1 for the total `txnum` of the sending data in case the serial interface (UART6) operates in DALI mode.

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

**[Example]**

Below is an example of sending a UART transmission of four bytes of fixed-length data one time.

```c
#include "CG_macrodriver.h"
BOOL gFlag; /* Transmission complete flag */
void main ( void ) {
  UCHAR txbuf[] = "ABCD";
  USHORT txnum = 4;
  gFlag = 1; /* Initialize transmission complete flag */
  ......
  UART6_Start (); /* Start UART communication*/
  UART6_SendData ( &txbuf, txnum ); /* Start UART data transmission */
}```
while ( gFlag ); /* Wait for txnum transmissions */
......
}

#include    "CG_macrodriver.h"
extern  BOOL                      gFlag;                  /* Transmission complete flag */
__interrupt void MD_INTST6 ( void ) {   /* Interrupt processing for INTST6 */
    if ( gUart6TxCnt > 0 ) {
        ......
        ......
        UART6_SendEndCallback ();       /* Call callback routine */
    } else {
        UART6_SendEndCallback ();       /* Call callback routine */
    }
}

void UART6_SendEndCallback ( void ) {   /* Callback routine for INTST6 */
    gFlag = 0;                          /* Set transmission complete flag */
}
### UART6_ReceiveData

Starts UART data reception.

**Remarks**

1. This API function performs byte-level UART reception the number of times specified by the parameter `rxnum`, and stores the data in the buffer specified by the parameter `rxbuf`.
2. Actual UART reception starts after this API function is called, and `UART6_Start` is then called.

### Classification

CG_serial.c

### Syntax

```c
#include    "CG_macrodriver.h"
MD_STATUS   UART6_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

### Argument(s)

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>UCHAR    *rxbuf;</td>
<td>Pointer to a buffer to store the received data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT   rxnum;</td>
<td>Total amount of data to receive</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

**Remark**

You can only set 2 for the total `rxnum` of the receiving data in case the serial interface (UART6) operates in DALI mode.

### Example

Below is an example of UART reception of four bytes of fixed-length data one time.

```c
#include    "CG_macrodriver.h"

BOOL   gFlag; /* Reception complete flag */

void main ( void ) {
    UCHAR   rxbuf[10];
    USHORT  rxnum = 4;
    gFlag = 1; /* Initialize reception complete flag */
    ......
    UART6_ReceiveData ( &rxbuf, rxnum ); /* Start UART data reception */
    UART6_Start (); /* Start UART communication */
}
```
while ( gFlag );                        /* Wait for rxnum receptions */
......
}

#include    "CG_macrodriver.h"
extern  BOOL  gFlag;                      /* Reception complete flag */
__interrupt void MD_INTSR6 ( void ) {       /* Interrupt processing for INTSR6 */
......
if ( gUart6RxLen > gUart6RxCnt ) {
......
if ( gUart6RxLen == gUart6RxCnt ) {
    UART6_ReceiveEndCallback ();    /* Call callback routine */
}
}
void UART6_ReceiveEndCallback ( void ) {   /* Callback routine for INTSR6 */
    gFlag = 0;                        /* Set reception complete flag */
}
**UART6_SendEndCallback**

Performs processing in response to the UART transmission complete interrupt INTST6.

**Remark**  This API function is called as the callback routine of interrupt process MD_INTST6 corresponding to the UART transmission complete interrupt INTST6 (performed when number of transmission data specified by UART6_SendData parameter `txnum` has been completed).

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void UART6_SendEndCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**UART6_ReceiveEndCallback**

Performs processing in response to the UART reception complete interrupt INTSR6.

**Remark** This API function is called as the callback routine of interrupt process MD_INTSR6 corresponding to the UART reception complete interrupt INTSR6 (performed when number of received data specified by UART6_ReceiveData parameter `rxnum` has been completed).

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void UART6_ReceiveEndCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**UART6_SoftOverRunCallback**

Performs processing in response to the UART reception complete interrupt INTSR6.

**Remark** This API function is called as the callback routine of interrupt process MD_INTSR6 corresponding to the UART reception complete interrupt INTSR6 (process performed when the amount of data received is greater than the parameter `rxnum` specified for `UART6_ReceiveData`).

**[Classification]**
CG_serial_user.c

**[Syntax]**
- [Ix2]

  ```c
  void UART6_SoftOverRunCallback ( USHORT rx_data );
  ```

- [Kx2-L]

  ```c
  void UART6_SoftOverRunCallback ( UCHAR rx_data );
  ```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>USHORT</td>
<td><code>rx_data</code>; Received data (data received greater than the number specified in the parameter <code>rxnum</code> for <code>UART6_ReceiveData</code>)</td>
</tr>
<tr>
<td>O</td>
<td>UCHAR</td>
<td><code>rx_data</code>; Received data (data received greater than the number specified in the parameter <code>rxnum</code> for <code>UART6_ReceiveData</code>)</td>
</tr>
</tbody>
</table>

**[Return value]**

None.
**UART6_ErrorCallback**

Performs processing in response to the UART communication error interrupt INTSRE6.

**Remark** This API function is called as the callback routine of interrupt process MD_INTSRE6 corresponding to the UART communication error interrupt INTSRE6.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
#include    "CG_macrodriver.h"
void    UART6_ErrorCallback ( UCHAR err_type );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>UCHAR err_type;</td>
<td>Trigger for UART communication error interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000xx1B: Overrun error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000x1xB: Framing error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000001xxB: Parity error</td>
</tr>
</tbody>
</table>

**[Return value]**

None.

**[Example]**

Below are examples of callback processing by the trigger for the UART communication error interrupt.

**[CG_serial_user.c]**

```c
#include    "CG_macrodriver.h"
__interrupt void MD_INTSRE6 ( void ) {          /* Interrupt processing for INTSRE6 */
    UCHAR err_type;
    ......
    UART6_ErrorCallback ( err_type );           /* Call callback routine */
}

void UART6_ErrorCallback ( UCHAR err_type ) {   /* Callback routine for INTSRE6 */
    if ( err_type & 0x1 ) {                     /* Determine trigger */
        ......
        /* Callback processing in response to overrun error */
    } else if ( err_type & 0x2 ) {              /* Determine trigger */
        ......
        /* Callback processing in response to framing error */
    } else if ( err_type & 0x4 ) {              /* Determine trigger */
        ......
        /* Callback processing in response to parity error */
    }
}
```
CSI1n_Init

Performs initialization of the serial interface (CSI1n) channel.

[Classification]
CG_serial.c

[Syntax]

    void    CSI1n_Init ( void );

[Remark]  
  \( n \) is the channel number.

[Argument(s)]

  None.

[Return value]

  None.
### CSI1n_UserInit

Performs user-defined initialization of the serial interface (CSI1n).

**Remark**  This API function is called as the CSI1n_Init callback routine.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void CSI1n_UserInit ( void );
```

**Remark**  \( n \) is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
CSI1n_Start

Sets CSI1n communication to standby mode.

[Classification]
CG_serial.c

[Syntax]

```c
void CSI1n_Start ( void );
```

Remark  $n$ is the channel number.

[Argument(s)]
None.

[Return value]
None.
CSI1n_Stop

Ends CSI1n communication.

[Classification]
CG_serial.c

[Syntax]

```c
void CSI1n_Stop ( void );
```

Remark  $n$ is the channel number.

[Argument(s)]
None.

[Return value]
None.
CSI1n_ReceiveData

Starts CSI1n data reception.

**Remarks**

1. This API function performs byte-level CSI1n reception the number of times specified by the parameter `rxnum`, and stores the data in the buffer specified by the parameter `rxbuf`.
2. When performing a CSI1n reception, `CSI1n_Start` must be called before this API function is called.

**[Classification]**

CG_serial.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
MD_STATUS CSI1n_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**Remark**

`n` is the channel number.

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>UCHAR *rxbuf;</td>
<td>Pointer to a buffer to store the received data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT rxnum;</td>
<td>Total amount of data to receive</td>
</tr>
</tbody>
</table>

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

**[Example]**

Below is an example of receiving a CSI10 transmission of four bytes of fixed-length data from channel 10 one time.

```c
#include "CG_macrodriver.h"

BOOL gFlag;    /* Reception complete flag */

void main ( void ) {
    UCHAR rxbuf[10];
    USHORT rxnum = 4;
    gFlag = 1; /* Initialize reception complete flag */
    .......
    CSI10_Start (); /* Start CSI10 communication */
    CSI10_ReceiveData ( &rxbuf, rxnum ); /* Start CSI10 reception */
    while ( gFlag ); /* Wait for rxnum receptions */
}```
......

}{

[CG_serial_user.c]

#include "CG_macrodriver.h"
extern BOOL gFlag;                      /* Reception complete flag */
__interrupt void MD_INTCSI10 ( void ) {     /* Interrupt processing for INTCSI10 */
    if ( gCsi10RxCnt < gCsi10RxLen ) {
        ......
        if ( gCsi10RxCnt == gCsi10RxLen ) {
            CSI10_ReceiveEndCallback ();    /* Call callback routine */
        } else {
            ......
        }
    } else {
    }
}

void CSI10_ReceiveEndCallback ( void ) { /* Callback routine for INTCSI10 */
    gFlag = 0;                              /* Set reception complete flag */
}
CSI1n_SendReceiveData

Starts CSI1n data transmission/reception.

Remarks 1. This API function repeats the byte-level CSI1n transmission from the buffer specified in parameter txbuf the number of times specified in parameter txnum.
2. This API function performs byte-level CSI1n reception the number of times specified by the parameter txnum, and stores the data in the buffer specified by the parameter rxbuf.
3. When performing a CSI1n reception, CSI1n_Start must be called before this API function is called.

[Classification]
CG_serial.c

[Syntax]

```c
#include "CG_macrodriver.h"
MD_STATUS CSI1n_SendReceiveData ( UCHAR *txbuf, USHORT txnum, UCHAR *rxbuf );
```

Remark $n$ is the channel number.

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR *txbuf;</td>
<td>Pointer to a buffer storing the transmission data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT txnum;</td>
<td>Total amount of data to send/receive</td>
</tr>
<tr>
<td>O</td>
<td>UCHAR *rxbuf;</td>
<td>Pointer to a buffer to store the received data</td>
</tr>
</tbody>
</table>

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

[Example]

Below is an example of sending and receiving a CSI10 transmission of four bytes of fixed-length data from channel 10 one time.

[CG_main.c]

```c
#include "CG_macrodriver.h"
BOOL gSflag;            /* Transmission complete flag */
void main ( void ) {    
   UCHAR txbuf[] = "0123";
   USHORT txnum = 4;
   UCHAR rxbuf[10];     
```
```c
#include "CG_macrodriver.h"
extern BOOL gSflag; /* Transmission complete flag */
__interrupt void MD_INTCSI10 ( void ) { /* Interrupt processing for INTCSI10 */
  if ( gCsi10TxCnt > 0 ) {
    ......
    ......
    CSI10_SendEndCallback (); /* Call callback routine */
  } else {
    ......
  }
}

void CSI10_SendEndCallback ( void ) { /* Callback routine for INTCSI10 */
  gSflag = 0; /* Set transmission complete flag */
}
```

```c
gSflag = 1; /* Initialize flag */
......
CSI10_Start (); /* Start CSI10 communication */
CSI10_SendReceiveData ( &txbuf, txnum, &rxbuf ); /* Start CSI10 send/receive */
*/
  while ( gSflag ); /* Wait for txnum transmissions/receptions */
  ......
}
```

[CG_serial_user.c]
### CSI1n_SendEndCallback

Performs processing in response to the CSI1n communication complete interrupt INTCSI1n.

**Remark**  
This API function is called as the callback routine of interrupt process MD_INTCSI1n corresponding to the CSI1n communication complete interrupt INTCSI1n (performed when number of transmission data specified by CSI1n_SendReceiveData parameter txnum has been completed).

**[Classification]**  
CG_serial_user.c

**[Syntax]**

```c
void CSI1n_SendEndCallback ( void );
```

**Remark**  
$n$ is the channel number.

**[Argument(s)]**  
None.

**[Return value]**  
None.
**CSI1\_ReceiveEndCallback**

Performs processing in response to the CSI1\_n communication complete interrupt INTCSI1\_n.

**Remark**  This API function is called as the callback routine of interrupt process MD\_INTCSI1\_n corresponding to the CSI1\_n communication complete interrupt INTCSI1\_n (performed when number of received data specified by CSI1\_n\_ReceiveData parameter rxnum has been completed).

**[Classification]**

CG\_serial\_user.c

**[Syntax]**

```
void CSI1\_n\_ReceiveEndCallback ( void );
```

**Remark**  \( n \) is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
IICA_Init

Performs initialization of the serial interface (IICA).

[Classification]
CG_serial.c

[Syntax]

```c
void IICA_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
IICA_UserInit

Performs user-defined initialization of the serial interface (IICA).

Remark  This API function is called as the IICA_Init callback routine.

[Classification]
CG_serial_user.c

[Syntax]

void IICA_UserInit ( void );

[Argument(s)]
None.

[Return value]
None.
IICA_Stop

Ends IICA communication.

[Classification]
CG_serial.c

[Syntax]

```c
void IICA_Stop ( void );
```

[Argument(s)]
None.

[Return value]
None.
IICA_MasterSendStart

Starts IICA master transmission.

**Remark**  This API function repeats the byte-level IICA master transmission from the buffer specified in parameter `txbuf` the number of times specified in parameter `txnum`.

**[Classification]**
- CG_serial.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
MD_STATUS IICA_MasterSendStart ( UCHAR adr, UCHAR *txbuf, USHORT txnum, UCHAR wait);
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR</td>
<td>Slave address</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR *</td>
<td>Pointer to a buffer storing the transmission data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT</td>
<td>Total amount of data to send</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR</td>
<td>Setup time of start conditions</td>
</tr>
</tbody>
</table>

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR1</td>
<td>Bus communication status</td>
</tr>
<tr>
<td>MD_ERROR2</td>
<td>Bus not released status</td>
</tr>
</tbody>
</table>
**IICA_MasterReceiveStart**

Starts IICA master reception.

**Remark**  This API function performs byte-level IICA master reception the number of times specified by the parameter `rxnum`, and stores the data in the buffer specified by the parameter `rxbuf`.

**[Classification]**

CG_serial.c

**[Syntax]**

```c
#include    "CG_macrodriver.h"

MD_STATUS   IICA_MasterReceiveStart ( UCHAR adr, UCHAR *rxbuf, USHORT rxnum, UCHAR wait);
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR adr</td>
<td>Slave address</td>
</tr>
<tr>
<td>O</td>
<td>UCHAR *rxbuf</td>
<td>Pointer to a buffer to store the received data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT rxnum</td>
<td>Total amount of data to receive</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR wait</td>
<td>Setup time of start conditions</td>
</tr>
</tbody>
</table>

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR1</td>
<td>Bus communication status</td>
</tr>
<tr>
<td>MD_ERROR2</td>
<td>Bus not released status</td>
</tr>
</tbody>
</table>
IICA_StopCondition

Generates stop conditions.

[Classification]
CG_serial.c

[Syntax]

```c
void IICA_StopCondition ( void );
```

[Argument(s)]
None.

[Return value]
None.
**IICA_MasterSendEndCallback**

Performs processing in response to the IICA communication complete interrupt INTIICA0.

**Remark**  This API function is called as the callback routine of interrupt process MD_INTIICA0 corresponding to the IICA communication complete interrupt INTIICA0.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void IICA_MasterSendEndCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**IICA_MsterReceiveEndCallback**

Performs processing in response to the IICA communication complete interrupt INTIICA0.

**Remark**  This API function is called as the callback routine of interrupt process MD_INTIICA0 corresponding to the IICA communication complete interrupt INTIICA0.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void IICA_MasterReceiveEndCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**IICA_MasterErrorCallback**

Performs processing in response to detection of error in IICA master communication.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
#include "CG_macrodriver.h"

void IICA_MasterErrorCallback ( MD_STATUS flag );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>MD_STATUS flag;</td>
<td>Cause of communication error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_SPT: Stop condition detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_NACK: Acknowledge not detected</td>
</tr>
</tbody>
</table>

**[Return value]**

None.
**IICA_SlaveSendStart**

Starts IICA slave transmission.

**Remark**  
This API function repeats the byte-level IICA slave transmission from the buffer specified in parameter `txbuf` the number of times specified in parameter `txnum`.

**[Classification]**  
CG_serial.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
void IICA_SlaveSendStart ( UCHAR *txbuf, USHORT txnum );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR *txbuf;</td>
<td>Pointer to a buffer storing the transmission data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT txnum;</td>
<td>Total amount of data to send</td>
</tr>
</tbody>
</table>

**[Return value]**

None.
**IICA_SlaveReceiveStart**

Starts IICA slave reception.

**Remark**  This API function performs byte-level IICA slave reception the number of times specified by the parameter `rxnum`, and stores the data in the buffer specified by the parameter `rxbuf`.

[Classification]  
CG_serial.c

[Syntax]  
```c
#include "CG_macrodriver.h"
void IICA_SlaveReceiveStart ( UCHAR *rxbuf, USHORT rxnum );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>UCHAR *rxbuf;</td>
<td>Pointer to a buffer to store the received data</td>
</tr>
<tr>
<td>I</td>
<td>USHORT rxnum;</td>
<td>Total amount of data to receive</td>
</tr>
</tbody>
</table>

[Return value]  
None.
### IICA_SlaveSendEndCallback

Performs processing in response to the IICA communication complete interrupt INTIICA0.

**Remark**  This API function is called as the callback routine of interrupt process MD_INTIICA0 corresponding to the IICA communication complete interrupt INTIICA0.

**[Classification]**
CG_serial_user.c

**[Syntax]**

```c
void IICA_SlaveSendEndCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
## IICA_SlaveReceiveEndCallback

Performs processing in response to the IICA communication complete interrupt INTIICA0.

**Remark**  This API function is called as the callback routine of interrupt process MD_INTIICA0 corresponding to the IICA communication complete interrupt INTIICA0.

### [Classification]

CG_serial_user.c

### [Syntax]

```c
void IICA_SlaveReceiveEndCallback ( void );
```

### [Argument(s)]

None.

### [Return value]

None.
IICA_SlaveErrorCallback

Performs processing in response to detection of error in IICA slave communication.

[Classification]
CG_serial_user.c

[Syntax]
#include    "CG_macrodriver.h"

void    IICA_SlaveErrorCallback ( MD_STATUS flag );

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>MD_STATUS</td>
<td>Cause of communication error</td>
</tr>
<tr>
<td></td>
<td>flag;</td>
<td>MD_ERROR: Address mismatch detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD_NACK: Acknowledge not detected</td>
</tr>
</tbody>
</table>

[Return value]
None.
**IICA_GetStopConditionCallback**

Performs processing in response to detection of stop condition in IICA slave communication.

**[Classification]**

CG_serial_user.c

**[Syntax]**

```c
void IICA_GetStopConditionCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
C.3.5 Operational Amplifier

Below is a list of API functions output by Code Generator for operational amplifiers use.

Table C-6. API Functions: [Operational Amplifier]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPAMP_Init</td>
<td>Performs initialization necessary to control operational amplifier functions.</td>
</tr>
<tr>
<td>OPAMP_UserInit</td>
<td>Performs user-defined initialization relating to the operational amplifier.</td>
</tr>
<tr>
<td>PGA_Start</td>
<td>Starts the operation of operational amplifier (PGA mode).</td>
</tr>
<tr>
<td>PGA_Stop</td>
<td>Ends the operation of operational amplifier (PGA mode).</td>
</tr>
<tr>
<td>PGA_ChangePGAFactor</td>
<td>Sets the input voltage amplification factor of a operational amplifier (PGA mode).</td>
</tr>
<tr>
<td>AMP_Start</td>
<td>Starts the operation of operational amplifier (single AMP mode).</td>
</tr>
<tr>
<td>AMP_Stop</td>
<td>Ends the operation of operational amplifier (single AMP mode).</td>
</tr>
<tr>
<td>AMPn_Start</td>
<td>Starts the operation of operational amplifier n (single AMP mode).</td>
</tr>
<tr>
<td>AMPn_Stop</td>
<td>Ends the operation of operational amplifier n (single AMP mode).</td>
</tr>
</tbody>
</table>
OPAMP_Init

Performs initialization necessary to control operational amplifier functions.

[Classification]
CG_opamp.c

[Syntax]

```c
void OPAMP_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
OPAMP_UserInit

Performs user-defined initialization relating to the operational amplifier.

**Remark**  This API function is called as the OPAMP_Init callback routine.

**[Classification]**
CG_opamp_user.c

**[Syntax]**

```c
void OPAMP_UserInit ( void );
```

**[Argument(s)]**
None.

**[Return value]**
None.
PGA_Start

Starts the operation of operational amplifier (PGA mode).

[Classification]
CG_opamp.c

[Syntax]

```c
void PGA_Start ( void );
```

[Argument(s)]
None.

[Return value]
None.
PGA_Stop

Ends the operation of operational amplifier (PGA mode).

[Classification]
CG_opamp.c

[Syntax]

```c
void PGA_Stop ( void );
```

[Argument(s)]
None.

[Return value]
None.
Sets the input voltage amplification factor of a operational amplifier (PGA mode).

**Remark**  The value specified in parameter `factor` is set to operational amplifier control register (AMP0M).

### [Classification]

CG_opamp.c

### [Syntax]

```c
#include    "CG_macrodriver.h"
#include    "CG_opamp.h"
MD_STATUS   PGA_ChangePGAFactor ( enum PGAFactor factor );
```

### [Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| I   | enum PGAFactor `factor`; | Input voltage amplification factor  
PGAFACTOR0: x4  
PGAFACTOR1: x8  
PGAFACTOR2: x16  
PGAFACTOR3: x32 |

### [Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
**AMP_Start**

Starts the operation of operational amplifier (single AMP mode).

**[Classification]**

CG_opamp.c

**[Syntax]**

```c
void AMP_Start ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**AMP_Stop**

Ends the operation of operational amplifier (single AMP mode).

**[Classification]**

CG_opamp.c

**[Syntax]**

```c
void AMP_Stop ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**AMPn_Start**

Starts the operation of operational amplifier $n$ (single AMP mode).

[Classification]

CG_opamp.c

[Syntax]

```c
void AMPn_Start ( void );
```

Remark $n$ is the channel number.

[Argument(s)]

None.

[Return value]

None.
**AMPn_Stop**

Ends the operation of operational amplifier n (single AMP mode).

**[Classification]**

CG_opamp.c

**[Syntax]**

```c
void AMPn_Stop ( void );
```

**Remark**  

n is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
C.3.6 Comparator

Below is a list of API functions output by Code Generator for comparator use.

Table C-7. API Functions: [Comparator]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparator_Init</td>
<td>Performs initialization necessary to control comparator functions.</td>
</tr>
<tr>
<td>Comparator_UserInit</td>
<td>Performs user-defined initialization relating to the comparator.</td>
</tr>
<tr>
<td>Comparator_Start</td>
<td>Starts the operation of comparator n.</td>
</tr>
<tr>
<td>Comparator_Stop</td>
<td>Ends the operation of comparator n.</td>
</tr>
</tbody>
</table>
**Comparator_Init**

Performs initialization necessary to control comparator functions.

**[Classification]**

CG_comparator.c

**[Syntax]**

```c
void Comparator_Init ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**Comparator_UserInit**

Performs user-defined initialization relating to the comparator.

**Remark**  This API function is called as the Comparator_Init callback routine.

**[Classification]**

CG_comparator_user.c

**[Syntax]**

```c
void Comparator_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
Comparator\textsubscript{n} \_Start

Starts the operation of comparator \textit{n}.

[Classification]

CG\_comparator.c

[Syntax]

\begin{verbatim}
void Comparator\textsubscript{n} \_Start ( void );
\end{verbatim}

Remark \textit{n} is the channel number.

[Argument(s)]

None.

[Return value]

None.
Ends the operation of comparator.

[Classification]

CG_comparatorn.c

[Syntax]

```c
void Comparator_Stop ( void );
```

Remark  

$n$ is the channel number.

[Argument(s)]

None.

[Return value]

None.
C.3.7 A/D Converter

Below is a list of API functions output by Code Generator for A/D converter use.

Table C-8. API Functions: [A/D Converter]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD_Init</td>
<td>Performs initialization necessary to control A/D converter functions.</td>
</tr>
<tr>
<td>AD_UserInit</td>
<td>Performs user-defined initialization relating to the A/D converter.</td>
</tr>
<tr>
<td>AD_ComparatorOn</td>
<td>Enables operation of voltage converter.</td>
</tr>
<tr>
<td>AD_ComparatorOff</td>
<td>Disables operation of voltage converter.</td>
</tr>
<tr>
<td>AD_Start</td>
<td>Starts A/D conversion.</td>
</tr>
<tr>
<td>AD_Stop</td>
<td>Ends A/D conversion.</td>
</tr>
<tr>
<td>AD_SelectADChannel</td>
<td>Configures the analog voltage input pin for A/D conversion.</td>
</tr>
<tr>
<td>AD_Read</td>
<td>Reads the results of A/D conversion (10 bits).</td>
</tr>
<tr>
<td>AD_ReadByte</td>
<td>Reads the results of A/D conversion (8 bits; most significant 8 bits of 10-bit resolution).</td>
</tr>
</tbody>
</table>
**AD_Init**

Performs initialization necessary to control A/D converter functions.

[Classification]
CG_ad.c

[Syntax]

```c
void AD_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
AD_UserInit

Performs user-defined initialization relating to the A/D converter.

Remark: This API function is called as the AD_Init callback routine.

[Classification]
CG_ad_user.c

[Syntax]

```c
void AD_UserInit ( void );
```

[Argument(s)]

None.

[Return value]

None.
**AD_ComparatorOn**

Enables operation of voltage converter.

**Remark**  
About 1 microsecond of stabilization time is required when changing the voltage converter from operation stopped to operation enabled status. Consequently, about 1 micro second must be left free between the call to this API function and the call to AD_Start.

**[Classification]**

CG_ad.c

**[Syntax]**

```c
void AD_ComparatorOn ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**AD_ComparatorOff**

Disables operation of voltage converter.

**[Classification]**

CG_ad.c

**[Syntax]**

```c
void AD_ComparatorOff ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**AD_Start**

Starts A/D conversion.

**Remark**  About 1 micro second of stabilization time is required when changing the voltage converter from operation stopped to operation enabled status. Consequently, about 1 micro second must be left free between the call to `AD_ComparatorOn` and the call to this API function.

**[Classification]**

CG_ad.c

**[Syntax]**

```c
void AD_Start ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**AD_Stop**

Ends A/D conversion.

**Remark**  The voltage converter continues to operate after the process of this API function completes. Consequently, to stop the operation of the voltage converter, you must call `AD_ComparatorOff` after the process of this API function completes.

**[Classification]**

CG_ad.c

**[Syntax]**

```c
void AD_Stop ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
AD_SelectADChannel

Configures the analog voltage input pin for A/D conversion.

**Remark**  The value specified in parameter `channel` is set to analog input channel specification register (ADS).

**[Classification]**

CG_ad.c

**[Syntax]**

```c
#include "CG_ad.h"
MD_STATUS AD_SelectADChannel ( enum ADChannel channel );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum ADChannel channel;</td>
<td>Analog voltage input pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADCHANNELn: Input pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADCHANNELPGAIN: Operational amplifier output pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADCHANNEL12V: Internal voltage (1.2 V)</td>
</tr>
</tbody>
</table>

**Remark**  See the header file CG_ad.h for details about the analog voltage input pin ADCHANNELn.

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
## AD_Read

Reads the results of A/D conversion (10 bits).

**Remark**  The contents of the 10-bit A/D conversion result register (ADCR) are stored in the area specified by parameter `buffer`.

### [Classification]

CG_ad.c

### [Syntax]

```c
#include "CG_macrodriver.h"

void AD_Read ( USHORT *buffer );
```

### [Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>USHORT *buffer</td>
<td>Pointer to area in which to store read results of A/D conversion</td>
</tr>
</tbody>
</table>

### [Return value]

None.
**AD_ReadByte**

Reads the results of A/D conversion (8 bits; most significant 8 bits of 10-bit resolution).

**Remark** The contents of the 8-bit A/D conversion result register H (ADCRH) are stored in the area specified by parameter *buffer*.

**[Classification]**

CG_ad.c

**[Syntax]**

```c
#include "CG_macrodriver.h"

void AD_ReadByte ( UCHAR *buffer );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>UCHAR *buffer</td>
<td>Pointer to area in which to store the results of A/D conversion</td>
</tr>
</tbody>
</table>

**[Return value]**

None.
C.3.8 Timer

Below is a list of API functions output by Code Generator for timer array unit use.

Table C-9. API Functions: [Timer]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMX_Init</td>
<td>Performs initialization necessary to control 16-bit timer Xn functions.</td>
</tr>
<tr>
<td>TMXn_Start</td>
<td>Starts the count for 16-bit timer Xn.</td>
</tr>
<tr>
<td>TMXn_Stop</td>
<td>Ends the count for 16-bit timer Xn.</td>
</tr>
<tr>
<td>TMXn_ChangeDuty</td>
<td>Changes the duty ratio of the PWM signal single-output to the TOX0n pin.</td>
</tr>
<tr>
<td>TMXn_ChangeDualDuty</td>
<td>Changes the duty ratio of the PWM signal dual-output to the TOX0n pin.</td>
</tr>
<tr>
<td>TMX_EnableHighImpedanceState</td>
<td>Begins high impedance output of the 16-bit timer Xn.</td>
</tr>
<tr>
<td>TMX_DisableHighImpedanceState</td>
<td>Ends high impedance output of the 16-bit timer Xn.</td>
</tr>
<tr>
<td>TM00_Init</td>
<td>Performs initialization necessary to control 16-bit timer/event counter 00 functions.</td>
</tr>
<tr>
<td>TM00_UserInit</td>
<td>Performs user-defined initialization relating to the 16-bit timer/event counter 00.</td>
</tr>
<tr>
<td>TM00_Start</td>
<td>Starts the count for 16-bit timer/event counter 00.</td>
</tr>
<tr>
<td>TM00_Stop</td>
<td>Ends the count for 16-bit timer/event counter 00.</td>
</tr>
<tr>
<td>TM00_ChangeTimerCondition</td>
<td>Changes the value of capture/compare control register 00 (CRC00).</td>
</tr>
<tr>
<td>TM00_GetFreeRunningValue</td>
<td>Captures the content of the capture register (CR0n).</td>
</tr>
<tr>
<td>TM00_SoftwareTriggerOn</td>
<td>Generates the trigger (software trigger) for one-shot pulse output.</td>
</tr>
<tr>
<td>TM00_ChangeDuty</td>
<td>Changes the duty ratio of the signal output to the TO00 pin.</td>
</tr>
<tr>
<td>TM00_GetPulseWidth</td>
<td>Captures the high/low-level width measured for the signal (pulses) input to the TI0n0 pin.</td>
</tr>
<tr>
<td>TM5n_Init</td>
<td>Performs initialization necessary to control 8-bit timer/event counter 5n functions.</td>
</tr>
<tr>
<td>TM5n_UserInit</td>
<td>Performs user-defined initialization relating to the 8-bit timer/event counter 5n.</td>
</tr>
<tr>
<td>TM5n_Start</td>
<td>Starts the count for 8-bit timer/event counter 5n.</td>
</tr>
<tr>
<td>TM5n_Stop</td>
<td>Ends the count for 8-bit timer/event counter 5n.</td>
</tr>
<tr>
<td>TM5n_ChangeTimerCondition</td>
<td>Changes the value of 8-bit timer compare register 5n (CR5n).</td>
</tr>
<tr>
<td>TM5n_ChangeDuty</td>
<td>Changes the duty ratio of the PWM signal output to the TO5n pin.</td>
</tr>
<tr>
<td>TMHn_Init</td>
<td>Performs initialization necessary to control 8-bit timer Hn functions.</td>
</tr>
<tr>
<td>TMHn_UserInit</td>
<td>Performs user-defined initialization relating to the 8-bit timer Hn.</td>
</tr>
<tr>
<td>TMHn_Start</td>
<td>Starts the count for 8-bit timer Hn.</td>
</tr>
<tr>
<td>TMHn_Stop</td>
<td>Ends the count for 8-bit timer Hn.</td>
</tr>
<tr>
<td>TMHn_ChangeTimerCondition</td>
<td>Changes the value of 8-bit timer H compare register 0n/1n (CMP0n/CMP1n).</td>
</tr>
<tr>
<td>TMHn_ChangeDuty</td>
<td>Changes the duty ratio of the PWM signal output to the TOHn pin.</td>
</tr>
<tr>
<td>TMH1_CarrierOutputEnable</td>
<td>Begins carrier pulse output of the 8-bit timer H1 (carrier generator mode).</td>
</tr>
<tr>
<td>TMH1_CarrierOutputDisable</td>
<td>Ends carrier pulse output of the 8-bit timer H1 (carrier generator mode).</td>
</tr>
</tbody>
</table>
**TMX_Init**

Performs initialization necessary to control 16-bit timer Xn functions.

**[Classification]**
CG_timer.c

**[Syntax]**

```c
void TMX_Init ( void );
```

**[Argument(s)]**
None.

**[Return value]**
None.
**TMXn_Start**

Starts the count for 16-bit timer Xn.

**Remark**  The time from the call to this API function to the start of counting depends on the type of the function in question (e.g. PWM output, or A/D conversion start timing signal output).

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TMXn_Start ( void );
```

**Remark**  \( n \) is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
**TMXn_Stop**

Ends the count for 16-bit timer Xn.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TMXn_Stop ( void );
```

**Remark**  
$n$ is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
TMXn_ChangeDuty

Changes the duty ratio of the PWM signal single-output to the TOXn pin.

Remark  This API function can only be called when the 16-bit timer Xn is being used for single-output.

[Classification]

CG_timer.c

[Syntax]

```c
#include "CG_macrodriver.h"
void TMXn_ChangeDuty ( UCHAR ratio );
```

Remark  n is the channel number.

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR ratio;</td>
<td>Duty ratio (0 to 100, unit: %)</td>
</tr>
</tbody>
</table>

Remark  The value set to duty ratio ratio must be in base 10 notation.

[Return value]

None.

[Example]

The example below shows changing the duty ratio to 25%.

[CG_main.c]

```c
#include "CG_macrodriver.h"
void main ( void ) {
    UCHAR ratio = 25;
    ......
    TMX0_Start (); /* Start count */
    ......
    TMX0_ChangeDuty ( ratio ); /* Change duty ratio */
    ......
}
```
TMXn_ChangeDualDuty

Changes the duty ratio of the PWM signal dual-output to the TOXn pin.

**Remark**  This API function can only be called when the 16-bit timer Xn is being used for dual-output.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
#include "CG_macrodriver.h"

void TMXn_ChangeDualDuty ( UCHAR ratio, UCHAR ratio1, UCHAR delay );
```

**Remark**  n is the channel number.

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR ratio;</td>
<td>Duty ratio of TOXn0 (0 to 100, unit: %)</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR ratio1;</td>
<td>Duty ratio of TOXn1 (0 to 100, unit: %)</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR delay;</td>
<td>Delay time of TOXn0 (0 to 100, unit: %)</td>
</tr>
</tbody>
</table>

**Remarks**

1. The value set to duty ratio *ratio*, *ratio1* and *delay* must be in base 10 notation.
2. The following figure displays the meaning of each argument.

![Diagram](image_url)

**[Return value]**

None.
**TMX_EnableHighImpedanceState**

Begins high impedance output of the 16-bit timer Xn.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TMX_EnableHighImpedanceState ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**TMX_DisableHighImpedanceState**

Ends high impedance output of the 16-bit timer \( X_n \).

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TMX_DisableHighImpedanceState ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
TM00_Init

Performs initialization necessary to control 16-bit timer/event counter 00 functions.

[Classification]
CG_timer.c

[Syntax]

    void    TM00_Init ( void );

[Argument(s)]
None.

[Return value]
None.
**TM00_UserInit**

Performs user-defined initialization relating to the 16-bit timer/event counter 00.

**Remark**   This API function is called as the `TM00_Init` callback routine.

**[Classification]**

CG_timer_user.c

**[Syntax]**

```
void TM00_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**TM00_Start**

Starts the count for 16-bit timer/event counter 00.

**Remark**  The time from the call to this API function to the start of counting depends on the type of the function in question (e.g. interval timer, square-wave output, or external event counter).

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TM00_Start ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**TM00_Stop**

Ends the count for 16-bit timer/event counter 00.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TM00_Stop ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**TM00_ChangeTimerCondition**

Changes the value of 16-bit timer capture/compare control register \( 0n0 \) (CR\(0n0\)).

**Remark**  To change the contents of CR\(0n0\), you must call TM00_Stop before calling this API function.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TM00_ChangeTimerCondition ( USHORT *array_reg, USHORT array_num );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>USHORT *array_reg;</td>
<td>Pointer to the area storing the value to set in the target register</td>
</tr>
<tr>
<td>I</td>
<td>USHORT array_num;</td>
<td>The register to change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CR000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: CR000, CR010</td>
</tr>
</tbody>
</table>

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR</td>
<td>Invalid argument &quot;array_num&quot; specification</td>
</tr>
</tbody>
</table>
Captures the content of the capture register (CR0n0).

**Remark**  This API function can only be called when the 16-bit timer/event counter 00 is running in free-running timer mode, and the 16-bit timer capture/compare control register 0n0 (CR0n0) is being used as a capture register.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
#include    "CG_macrodriver.h"
#include    "CG_timer.h"
MD_STATUS   TM00_GetFreeRunningValue ( ULONG *count, enum TMChannel channel);
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>ULONG *count;</td>
<td>Pointer to area in which to store the captured value</td>
</tr>
<tr>
<td>I</td>
<td>enum TMChannel channel;</td>
<td>The pin to capture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMCHANNEL0: TI000 pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMCHANNEL1: TI010 pin</td>
</tr>
</tbody>
</table>

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR</td>
<td>Exit with error (abend)</td>
</tr>
<tr>
<td></td>
<td>- CR0n0 is operating as a compare register.</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
Generates the trigger (software trigger) for one-shot pulse output.

**Remark** This API function can only be called when the 16-bit timer/event counter 00 is being used for one-shot pulse output.

**[Classification]**
CG_timer.c

**[Syntax]**

```c
void TM00_SoftwareTriggerOn ( void );
```

**[Argument(s)]**
None.

**[Return value]**
None.
TM00_ChangeDuty

Changes the duty ratio of the signal output to the TO00 pin.

**Remark**  This API function can only be called when the 16-bit timer/event counter 00 is being used for PPG output.

**[Classification]**
CG_timer.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
void TM00_ChangeDuty ( UCHAR ratio );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR ratio;</td>
<td>Duty ratio (0 to 100, unit: %)</td>
</tr>
</tbody>
</table>

**Remark**  The value set to duty ratio `ratio` must be in base 10 notation.

**[Return value]**

None.

**[Example]**
The example below shows changing the duty ratio to 25%.

**[CG_main.c]**

```c
#include "CG_macrodriver.h"
void main ( void ) {
   UCHAR ratio = 25;
   ......
   TM00_Start (); /* Start count */
   ......
   TM00_ChangeDuty ( ratio ); /* Change duty ratio */
   ......
}
```
**TM00_GetPulseWidth**

Captures the high/low-level width measured for the signal (pulses) input to the TI0n0 pin.

**Remark**  This API function can only be called when the 16-bit timer/event counter 00 is being used for pulse width measurement.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
#include "CG_timer.h"

void TM00_GetPulseWidth ( ULONG *highwidth, ULONG *lowwidth, enum TMChannel channel );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>ULONG *highwidth</td>
<td>Pointer to area storing the high-level measurement width (0x0 to 0xffff)</td>
</tr>
<tr>
<td>O</td>
<td>ULONG *lowwidth</td>
<td>Pointer to area storing the low-level measurement width (0x0 to 0xffff)</td>
</tr>
<tr>
<td>I</td>
<td>enum TMChannel</td>
<td>The pin to measure</td>
</tr>
<tr>
<td></td>
<td>channel</td>
<td>TMCHANNEL0: TI000 pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMCHANNEL1: TI010 pin</td>
</tr>
</tbody>
</table>

**[Return value]**

None.
TM5n_Init

Performs initialization necessary to control 8-bit timer/event counter 5n functions.

[Classification]
CG_timer.c

[Syntax]

```c
void TM5n_Init ( void );
```

Remark  \( n \) is the channel number.

[Argument(s)]
None.

[Return value]
None.
Performs user-defined initialization relating to the 8-bit timer/event counter $5n$.

**Remark**  This API function is called as the $TM5n\_Init$ callback routine.

**[Classification]**

CG\_timer\_user.c

**[Syntax]**

```c
void TM5n\_UserInit ( void ) ;
```

**Remark**  $n$ is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
TM5\textsubscript{\textit{n}}_Start

Starts the count for 8-bit timer/event counter \textit{n}.

Remark  The time from the call to this API function to the start of counting depends on the type of the function in question (e.g. interval timer, or external event counter).

[Classification]
CG\_timer\_c

[Syntax]

\begin{verbatim}
void TM5\textsubscript{\textit{n}}_Start ( void );
\end{verbatim}

Remark  \textit{n} is the channel number.

[Argument(s)]
None.

[Return value]
None.
TM5n_Stop

Ends the count for 8-bit timer/event counter 5n.

[Classification]
CG_timer.c

[Syntax]

```c
void TM5n_Stop ( void );
```

Remark  \( n \) is the channel number.

[Argument(s)]
None.

[Return value]
None.
**TM5n_ChangeTimerCondition**

Changes the value of 8-bit timer compare register 5\(n\) (CR5\(n\)).

**Remark** To change the contents of CR5\(n\), you must call TM5n_Stop before calling this API function.

[**Classification**]

CG_timer.c

[**Syntax**]

```c
#include "CG_macrodriver.h"
void TM5n_ChangeTimerCondition ( UCHAR value);
```

**Remark** \(n\) is the channel number.

[**Argument(s)**]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR value;</td>
<td>The value to set in CR5(n)</td>
</tr>
</tbody>
</table>

[**Return value**]

None.
**TM5n_ChangeDuty**

Changes the duty ratio of the PWM signal output to the TO5n pin.

**Remark**  This API function can only be called when the 8-bit timer/event counter 5n is being used for PWM output.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
void TM5n_ChangeDuty ( UCHAR ratio );
```

**Remark**  \( n \) is the channel number.

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR ratio;</td>
<td>Duty ratio (0 to 100, unit: %)</td>
</tr>
</tbody>
</table>

**Remark**  The value set to duty ratio \( ratio \) must be in base 10 notation.

**[Return value]**

None.

**[Example]**

The example below shows changing the duty ratio to 25%.

**[CG_main.c]**

```c
#include "CG_macrodriver.h"
void main ( void ) {
    UCHAR  ratio = 25;
    ......  
    TM50_Start ();  /* Start count */
    ......  
    TM50_ChangeDuty ( ratio );  /* Change duty ratio */
    ......  
}
```
**TMHn_Init**

Performs initialization necessary to control 8-bit timer Hn functions.

[Classification]
CG_timer.c

[Syntax]

```c
void TMHn_Init ( void );
```

**Remark**

$n$ is the channel number.

[Argument(s)]

None.

[Return value]

None.
**TMHn_UserInit**

Performs user-defined initialization relating to the 8-bit timer Hn.

**Remark**  This API function is called as the TMHn_Init callback routine.

**[Classification]**

CG_timer_user.c

**[Syntax]**

```c
void TMHn_UserInit ( void );
```

**[Syntax]**

Remark  \( n \) is the channel number.

**[Argument(s)]**

None.

**[Return value]**

None.
TMHn_Start

Starts the count for 8-bit timer Hn.

**Remark** The time from the call to this API function to the start of counting depends on the type of the function in question (e.g. interval timer, square-wave output, or PWM output).

**[Classification]**
CG_timer.c

**[Syntax]**

```c
void TMHn_Start ( void );
```

**Remark** \( n \) is the channel number.

**[Argument(s)]**
None.

**[Return value]**
None.
TMHn_Stop

Ends the count for 8-bit timer Hn.

[Classification]
CG_timer.c

[Syntax]

```c
void TMHn_Stop ( void );
```

Remark  \( n \) is the channel number.

[Argument(s)]
None.

[Return value]
None.
Changes the value of 8-bit timer H compare register 0n/1n (CMP0n/CMP1n).

**Remark** To change the contents of CMP0n/CMP1n, you must call **TMHn_Stop** before calling this API function.

### [Classification]

CG_timer.c

### [Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_timer.h"

MD_STATUS   TMHn ChangeTimerCondition ( UCHAR * array_reg, UCHAR array_num );
```

**Remark**  

\( n \) is the channel number.

### [Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR *array_reg;</td>
<td>Pointer to the area storing the value to set in the target register</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR array_num;</td>
<td>The register to change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CMP0n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: CMP0n, CMP1n</td>
</tr>
</tbody>
</table>

### [Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
Changes the duty ratio of the PWM signal output to the TOHn pin.

**Remark**  This API function can only be called when the 8-bit timer Hn is being used for PWM output.

**[Classification]**

CG_timer.c

**[Syntax]**

```c
#include "CG_macrodriver.h"
void TMHn_ChangeDuty ( UCHAR ratio );
```

**Remark**  $n$ is the channel number.

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>UCHAR ratio;</td>
<td>Duty ratio (0 to 100, unit: %)</td>
</tr>
</tbody>
</table>

**Remark**  The value set to duty ratio ratio must be in base 10 notation.

**[Return value]**

None.

**[Example]**

The example below shows changing the duty ratio to 25%.

**[CG_main.c]**

```c
#include "CG_macrodriver.h"
void main ( void ) {
    UCHAR ratio = 25;
    ......
    TMH0_Start (); /* Start count */
    ......
    TMH0_ChangeDuty ( ratio ); /* Change duty ratio */
    ......
}
```
**TMH1_CarrierOutputEnable**

Begins carrier pulse output of the 8-bit timer H1 (carrier generator mode).

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TMH1_CarrierOutputEnable ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**TMH1_CarrierOutputDisable**

Ends carrier pulse output of the 8-bit timer H1 (carrier generator mode).

**[Classification]**

CG_timer.c

**[Syntax]**

```c
void TMH1_CarrierOutputDisable ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
C.3.9 Watchdog Timer

Below is a list of API functions output by Code Generator for watchdog timer use.

Table C-10. API Functions: [Watchdog Timer]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDT_Restart</td>
<td>Clears the watchdog timer counter and resumes counting.</td>
</tr>
</tbody>
</table>
**WDT_Restart**

Clears the watchdog timer counter and resumes counting.

**[Classification]**

CG_wdt.c

**[Syntax]**

```c
void WDT_Restart ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
### C.3.10 Real-time Clock

Below is a list of API functions output by Code Generator for real-time counter use.

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_Init</td>
<td>Performs initialization necessary to control real-time counter functions.</td>
</tr>
<tr>
<td>RTC_UserInit</td>
<td>Performs user-defined initialization relating to the real-time counter.</td>
</tr>
<tr>
<td>RTC_PowerOff</td>
<td>Halts the clock supplied to the real-time counter.</td>
</tr>
<tr>
<td>RTC_CounterEnable</td>
<td>Starts the count of the real-time counter (year, month, weekday, day, hour, minute, second).</td>
</tr>
<tr>
<td>RTC_CounterDisable</td>
<td>Ends the count of the real-time counter (year, month, weekday, day, hour, minute, second).</td>
</tr>
<tr>
<td>RTC_SetHourSystem</td>
<td>Sets the clock type (12-hour or 24-hour clock) of the real-time counter.</td>
</tr>
<tr>
<td>RTC_CounterSet</td>
<td>Sets the counter value (year, month, weekday, day, hour, minute, second) of the real-time counter.</td>
</tr>
<tr>
<td>RTC_CounterGet</td>
<td>Reads the counter value (year, month, weekday, day, hour, minute, second) of the real-time counter.</td>
</tr>
<tr>
<td>RTC_ConstPeriodInterruptEnable</td>
<td>Sets the cycle of the interrupts INTRTC, then starts the cyclic interrupt function.</td>
</tr>
<tr>
<td>RTC_ConstPeriodInterruptDisable</td>
<td>Ends the cyclic interrupt function.</td>
</tr>
<tr>
<td>RTC_ConstPeriodInterruptCallback</td>
<td>Performs processing in response to the cyclic interrupt INTRTC.</td>
</tr>
<tr>
<td>RTC_AlarmEnable</td>
<td>Starts the alarm interrupt function.</td>
</tr>
<tr>
<td>RTC_AlarmDisable</td>
<td>Ends the alarm interrupt function.</td>
</tr>
<tr>
<td>RTC_AlarmSet</td>
<td>Sets the alarm conditions (weekday, hour, minute).</td>
</tr>
<tr>
<td>RTC_AlarmGet</td>
<td>Reads the alarm conditions (weekday, hour, minute).</td>
</tr>
<tr>
<td>RTC_AlarmInterruptCallback</td>
<td>Performs processing in response to the alarm interrupt INTRTC.</td>
</tr>
<tr>
<td>RTC_IntervalStart</td>
<td>Starts the interval interrupt function.</td>
</tr>
<tr>
<td>RTC_IntervalStop</td>
<td>Ends the interval interrupt function.</td>
</tr>
<tr>
<td>RTC_IntervalInterruptEnable</td>
<td>Sets the cycle of the interrupts INTRTCI, then starts the interval interrupt function.</td>
</tr>
<tr>
<td>RTC_IntervalInterruptDisable</td>
<td>Ends the interval interrupt function.</td>
</tr>
<tr>
<td>RTC_RTC1HZ_OutputEnable</td>
<td>Enables output of the real-time counter correction clock (1 Hz) to the RTC1HZ pin.</td>
</tr>
<tr>
<td>RTC_RTC1HZ_OutputDisable</td>
<td>Disables output of the real-time counter correction clock (1 Hz) to the RTC1HZ pin.</td>
</tr>
<tr>
<td>RTC_RTCCL_OutputEnable</td>
<td>Enables output of the real-time counter clock (32 kHz source) to the RTCCL pin.</td>
</tr>
<tr>
<td>RTC_RTCCL_OutputDisable</td>
<td>Disables output of the real-time counter clock (32 kHz source) to the RTCCL pin.</td>
</tr>
<tr>
<td>RTC_RTCDIV_OutputEnable</td>
<td>Enables output of the real-time counter clock (32 kHz cycle) to the RTCDIV pin.</td>
</tr>
<tr>
<td>RTC_RTCDIV_OutputDisable</td>
<td>Disables output of the real-time counter clock (32 kHz cycle) to the RTCDIV pin.</td>
</tr>
<tr>
<td>RTC_ChangeCorrectionValue</td>
<td>Changes the timing and correction value for correcting clock errors.</td>
</tr>
</tbody>
</table>
RTC_Init

Performs initialization necessary to control real-time counter functions.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
**RTC_UserInit**

Performs user-defined initialization relating to the real-time counter.

**Remark**  This API function is called as the RTC_Init callback routine.

**[Classification]**

CG rtc_user.c

**[Syntax]**

```c
void RTC_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
RTC_PowerOff

Halts the clock supplied to the real-time counter.

Remark	Calling this API function changes the real-time counter to reset status. For this reason, writes to the control registers (e.g. real-time counter control register 0: RTCC0) after this API function is called are ignored.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_PowerOff ( void );
```

[Argument(s)]
None.

[Return value]
None.
RTC_CounterEnable

Starts the count of the real-time counter (year, month, weekday, day, hour, minute, second).

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_CounterEnable ( void );
```

[Argument(s)]
None.

[Return value]
None.
RTC_CounterDisable

Ends the count of the real-time counter (year, month, weekday, day, hour, minute, second).

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_CounterDisable ( void );
```

[Argument(s)]
None.

[Return value]
None.
Sets the clock type (12-hour or 24-hour clock) of the real-time counter.

[Classification]
CG_rtc.c

[Syntax]

```c
#include    "CG_macrodriver.h"
#include    "CG_rtc.h"
MD_STATUS   RTC_SetHourSystem ( enum RTCHourSystem hoursystem );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum  RTCHourSystem hoursystem;</td>
<td>Clock type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HOUR12: 12-hour clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HOUR24: 24-hour clock</td>
</tr>
</tbody>
</table>

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_BUSY1</td>
<td>Executing count process (before change to setting)</td>
</tr>
<tr>
<td>MD_BUSY2</td>
<td>Stopping count process (after change to setting)</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

Remark   If MD_BUSY1 or MD_BUSY2 is returned, it may be because the counter-operation is stopped, or the counter operation start wait time is too short, so make the value of the RTC_WAITTIME macro defined in the header file "CG_rtc.h" larger.

[Example]

Below is an example of setting the clock type to the 24-hour clock.

[CG_main.c]

```c
#include    "CG_rtc.h"
void main ( void ) {
    ......
    RTC_CounterEnable ();  /* Start count */
    ......
    RTC_SetHourSystem ( HOUR24 );  /* Set clock type */
    ......
```
RTC_CounterSet

Sets the counter value (year, month, weekday, day, hour, minute, second) of the real-time counter.

[Classification]
CG_rtc.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_CounterSet ( struct RTCCounterValue counterwriteval );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>struct RTCCounterValue counterwriteval;</td>
</tr>
</tbody>
</table>

[Remark] Below is an example of the structure RTCCounterValue (counter value) for the real-time counter.

```c
struct RTCCounterValue {
    UCHAR Sec;    /* second */
    UCHAR Min;    /* Minute */
    UCHAR Hour;   /* Hour */
    UCHAR Day;    /* Day */
    UCHAR Week;   /* Weekday (0: Sunday, 6: Saturday) */
    UCHAR Month;  /* Month */
    UCHAR Year;   /* Year */
};
```

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_BUSY1</td>
<td>Executing count process (before change to setting)</td>
</tr>
<tr>
<td>MD_BUSY2</td>
<td>Stopping count process (after change to setting)</td>
</tr>
</tbody>
</table>

[Remark] If MD_BUSY1 or MD_BUSY2 is returned, it may be because the counter-operation is stopped, or the counter operation start wait time is too short, so make the value of the RTC_WAITTIME macro defined in the header file "CG_rtc.h" larger.

[Example]
The example below shows the counter value of the real-time counter being set to "2008/12/25 (Thu.) 17:30:00".
```c
#include "CG_rtc.h"
void main ( main ) {
    struct RTCCounterValue counterwriteval;
    .......
    RTC_CounterEnable (); /* Start count */
    .......
    counterwriteval.Year = 0x08;
    counterwriteval.Month = 0x12;
    counterwriteval.Day = 0x25;
    counterwriteval.Week = 0x05;
    counterwriteval.Hour = 0x17;
    counterwriteval.Min = 0x30;
    counterwriteval.Sec = 0;
    RTC_SetHourSystem ( HOUR24 ); /* Set clock type */
    RTC_CounterSet ( counterwriteval ); /* Set counter value */
    .......
}
```
RTC.CounterGet

Reads the counter value (year, month, weekday, day, hour, minute, second) of the real-time counter.

[Classification]
CG_rtc.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC.CounterGet ( struct RTCounterValue *counterreadval );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>struct RTCounterValue *counterreadval</td>
<td>Pointer to structure in which to store the counter value being read</td>
</tr>
</tbody>
</table>

[Remark] See RTC.CounterSet for details about the RTCCounterValue counter value.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_BUSY1</td>
<td>Executing count process (before reading)</td>
</tr>
<tr>
<td>MD_BUSY2</td>
<td>Stopping count process (after reading)</td>
</tr>
</tbody>
</table>

[Remark] If MD_BUSY1 or MD_BUSY2 is returned, it may be because the counter-operation is stopped, or the counter operation start wait time is too short, so make the value of the RTC_WAITTIME macro defined in the header file "CG_rtc.h" larger.

[Example]

Below is an example of reading the counter value of the real-time counter.

[CG_main.c]

```c
#include "CG_rtc.h"
void main ( void ) {  
    struct RTCounterValue counterreadval;
    ......
    RTC.CounterEnable (); /* Start count */
    ......
    RTC.CounterGet ( &counterreadval ); /* Read count value */
    ......
```
Sets the cycle of the interrupts INTRTC, then starts the cyclic interrupt function.

[Classification]
CG_rtc.c

[Syntax]

```c
#include    "CG_rtc.h"
MD_STATUS   RTC_ConstPeriodInterruptEnable ( enum RTCINTPeriod  period );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum RTCINTPeriod  period;</td>
<td>Interrupt INTRTC cycle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HALFSEC: 0.5 seconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONESEC: 1 second</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONEMIN: 1 minute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONEHOUR: 1 hour</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONEDAY: 1 day</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONEMONTH: 1 month</td>
</tr>
</tbody>
</table>

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

[Example]

Below is an example of setting the cycle of the interrupts INTRTC, then starting the cyclic interrupt function.

[CG_main.c]

```c
#include    "CG_rtc.h"
void main ( void ) {
    .......
    RTC_ConstPeriodInterruptDisable ();    /* End of cyclic interrupt function */
    .......
    RTC_ConstPeriodInterruptEnable ( HALFSEC ); /* Start of cyclic interrupt function */
    .......
}
```
RTC_ConstPeriodInterruptDisable

Ends the cyclic interrupt function.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_ConstPeriodInterruptDisable ( void );
```

[Argument(s)]
None.

[Return value]
None.
**RTC_ConstPeriodInterruptCallback**

Performs processing in response to the cyclic interrupt INTRTC.

**Remark**  This API function is called as the callback routine of interrupt process MD_INTRTC corresponding to the cyclic interrupt INTRTC.

**[Classification]**

CG_rtc_user.c

**[Syntax]**

```c
void RTC_ConstPeriodInterruptCallback ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
RTCAlarmEnable

Starts the alarm interrupt function.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_AlarmEnable ( void );
```

[Argument(s)]
None.

[Return value]
None.
RTC_AlarmDisable

Ends the alarm interrupt function.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_AlarmDisable ( void );
```

[Argument(s)]

None.

[Return value]

None.
RTC_AlarmSet

Sets the alarm conditions (weekday, hour, minute).

[Classification]
CG_rtc.c

[Syntax]

```c
#include "CG_rtc.h"
void RTC_AlarmSet ( struct RTCAlarmValue alarmval );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>struct RTCAlarmValue</td>
<td>Alarm conditions (weekday, hour, minute)</td>
</tr>
</tbody>
</table>

[Remark] Below is shown the structure RTCAlarmValue (alarm conditions).

```c
struct RTCAlarmValue {
    UCHAR Alarmwm;    /* Minute */
    UCHAR Alarmwh;    /* Hour */
    UCHAR Alarmww;    /* Weekday */
};
```

- Alarmwm (Minute)
  Below are shown the meanings of each bit of the structure member Alarmwm.

```
<table>
<thead>
<tr>
<th>7</th>
<th>4</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

BCD code (minute: digit 1)

BCD code (minute: digit 10)

- Alarmwh (Hour)
  Below are shown the meanings of each bit of the structure member Alarmwh. If the real-time counter is set to the 12-hour clock, then bit 5 has the following meaning.

  | 0: | 1: |
  | AM | PM |
- Alarmww (Weekday)
  Below are shown the meanings of each bit of the structure member Alarmww.

\[
\begin{array}{cccccc}
7 & 6 & 5 & 4 & 3 & 0 \\
\hline
\text{BCD code (hour: digit 1)} & \text{BCD code (hour: digit 10)} \\
00: & \text{Fixed} \\
\end{array}
\]

[Return value]
None.

[Example 1]
The example below shows the alarm conditions being set to "Monday/Tuesday/Wednesday at 17:30".

[CG_main.c]
```c
#include    "CG_rtc.h"

void main ( void ) {
  struct  RTCAlarmValue alarmval;
  ....
  RTC_AlarmEnable ();    /* Start alarm interrupt function */
  RTC_CounterEnable ();  /* Start count */
  ....
  RTC_SetHourSystem ( HOUR24 );  /* Set clock type */
  alarmval.Alarmww = 0xe;
  alarmval.Alarmwh = 0x17;
  alarmval.Alarmwm = 0x30;
  RTC_AlarmSet ( alarmval );  /* Set conditions */
  ....
}
```
[Example 2]

The example below shows the alarm conditions being set to "Saturday/Sunday (time left unchanged)".

[CG_main.c]

```c
#include "CG_rtc.h"
void main ( void ) {
    struct RTCAlarmValue alarmval;
    ......
    RTC_AlarmEnable (); /* Start alarm interrupt function */
    ......
    RTC_AlarmSet ( alarmval ); /* Change conditions */
    ......
}
```
RTC_AlarmGet

Reads the alarm conditions (weekday, hour, minute).

[Classification]
CG_rtc.c

[Syntax]

```c
#include "CG_rtc.h"
void RTC_AlarmGet ( struct RTCAlarmValue *alarmval );
```

[Remark]  See RTC_AlarmSet for details about RTCAlarmValue (alarm conditions).

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>struct RTCAlarmValue *alarmval;</td>
<td>Pointer to structure in which to store the conditions being read</td>
</tr>
</tbody>
</table>

[Return value]

None.

[Example]

The example below shows the alarm conditions being read.

[CG_main.c]

```c
#include "CG_rtc.h"
void main ( void ) {
    struct RTCAlarmValue alarmval;
    ........
    RTC_AlarmEnable (); /* Start alarm interrupt function */
    ........
    RTC_AlarmGet ( &alarmval ); /* Read conditions */
    ........
}
```
RTC_AlarmInterruptCallback

Performs processing in response to the alarm interrupt INTRTC.

Remark This API function is called as the callback routine of interrupt process MD_INTRTC corresponding to the alarm interrupt INTRTC.

[Classification]
CG_rtc_user.c

[Syntax]
void RTC_AlarmInterruptCallback ( void );

[Argument(s)]
None.

[Return value]
None.
### RTC_IntervalStart

Starts the interval interrupt function.

**Remark**  After setting the cycle of the interrupts INTRTCI, call `RTC_IntervalInterruptEnable` to start the interval interrupt function.

**[Classification]**

CG_rtc.c

**[Syntax]**

```c
void RTC_IntervalStart ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
RTC_IntervalStop

Ends the interval interrupt function.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_IntervalStop ( void );
```

[Argument(s)]
None.

[Return value]
None.
Sets the cycle of the interrupts INTRTCI, then starts the interval interrupt function.

**Remark** Call **RTC_IntervalStart** to start the interval interrupt function without setting the cycle of the interrupts INTRTCI.

[Classification]
CG_rtc.c

[Syntax]

```c
#include    "CG_rtc.h"
MD_STATUS   RTC_IntervalInterruptEnable ( enum RTCINTInterval interval );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| I   | enum RTCINTInterval interval; | Interrupt INTRTCI cycle INTERVAL0: 2^6/fSUB  
INTERVAL1: 2^7/fSUB  
INTERVAL2: 2^8/fSUB  
INTERVAL3: 2^9/fSUB  
INTERVAL4: 2^10/fSUB  
INTERVAL5: 2^11/fSUB  
INTERVAL6: 2^12/fSUB |

**Remark** fSUB is the frequency of the subsystem clock.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>

[Example]

Below is an example of changing the interval, the restarting the interval interrupt function.

[CG_main.c]

```c
#include    "CG_rtc.h"
void main ( void ) {
       ......
       RTC_IntervalStart ();                /* Start interval interrupt function */
       ......
```
RTC_IntervalStop (); /* End interval interrupt function */
......
RTC_IntervalInterruptEnable ( INTERVAL6 ); /* Start interval interrupt function */
......
}
RTC_IntervalInterruptDisable

Ends the interval interrupt function.

[Classification]
CG_RTC.c

[Syntax]

```c
void RTC_IntervalInterruptDisable ( void );
```

[Argument(s)]
None.

[Return value]
None.
RTC_RTC1HZ_OutputEnable

Enables output of the real-time counter correction clock (1 Hz) to the RTC1HZ pin.

[Classification]
CG rtc.c

[Syntax]

```c
void RTC_RTC1HZ_OutputEnable ( void );
```

[Argument(s)]
None.

[Return value]
None.
RTC_RTC1HZ_OutputDisable

Disables output of the real-time counter correction clock (1 Hz) to the RTC1HZ pin.

[Classification]

CG_rtc.c

[Syntax]

```c
void RTC_RTC1HZ_OutputDisable ( void );
```

[Argument(s)]

None.

[Return value]

None.
**RTC_RTCCL_OutputEnable**

Enables output of the real-time counter clock (32 kHz source) to the RTCL pin.

**[Classification]**

CG_rtc.c

**[Syntax]**

```c
void RTC_RTCCL_OutputEnable ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
RTC_RTCCL_OutputDisable

Disables output of the real-time counter clock (32 kHz source) to the RTCCL pin.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_RTCCL_OutputDisable ( void );
```

[Argument(s)]
None.

[Return value]
None.
RTC_RTCDIV_OutputEnable

Enables output of the real-time counter clock (32 kHz cycle) to the RTCDIV pin.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_RTCDIV_OutputEnable ( void );
```

[Argument(s)]
None.

[Return value]
None.
Disables output of the real-time counter clock (32 kHz cycle) to the RTCDIV pin.

[Classification]
CG_rtc.c

[Syntax]

```c
void RTC_RTCDIV_OutputDisable ( void );
```

[Argument(s)]
None.

[Return value]
None.
Changes the timing and correction value for correcting clock errors.

[Classification]
CG_rtc.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_ChangeCorrectionValue ( enum RTCCorectionTiming timing, UCHAR correctval );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum RTCCorectionTiming</td>
<td>When clock errors are corrected</td>
</tr>
<tr>
<td></td>
<td>timing;</td>
<td>EVERY20S: When the seconds digits are 00, 20 or 40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EVERY60S: When the seconds digits are 00</td>
</tr>
<tr>
<td>I</td>
<td>UCHAR correctval;</td>
<td>Clock error correction value</td>
</tr>
</tbody>
</table>

Remark This API function does not correct clock errors if correction value `correctVal` is set to 0x0, 0x1, 0x40 or 0x41.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
C.3.11 Clock Output

Below is a list of API functions output by Code Generator for clock output use.

Table C-12. API Functions: [Clock Output]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCL_Init</td>
<td>Performs initialization necessary to control clock output control circuit functions.</td>
</tr>
<tr>
<td>PCL_UserInit</td>
<td>Performs user-defined initialization relating to the clock output control circuits.</td>
</tr>
<tr>
<td>PCL_Start</td>
<td>Starts clock output.</td>
</tr>
<tr>
<td>PCL_Stop</td>
<td>Ends clock output.</td>
</tr>
<tr>
<td>PCL_ChangeFreq</td>
<td>Changes the output clock to the PCL pin.</td>
</tr>
</tbody>
</table>
PCL_Init

Performs initialization necessary to control clock output control circuit functions.

[Classification]
CG_pcl.c

[Syntax]

```c
void PCL_Init ( void );
```

[Argument(s)]
None.

[Return value]
None.
**PCL_UserInit**

Performs user-defined initialization relating to the clock output control circuits.

**Remark**  This API function is called as the PCL_Init callback routine.

**[Classification]**

CG_pcl_user.c

**[Syntax]**

```c
void PCL_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
### PCL_Start

Starts clock output.

**[Classification]**

CG pcl.c

**[Syntax]**

```c
void PCL_Start ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
**PCL_Stop**

Ends clock output.

**[Classification]**
CG_pcl.c

**[Syntax]**

```c
void PCL_Stop ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
PCL_ChangeFreq

Changes the output clock to the PCL pin.

**Remark**  The value specified in parameter `clock` is set to clock output select register (CKS).

**[Classification]**

CG_pcl.c

**[Syntax]**

```c
#include "CG_pclbuz.h"
MD_STATUS PCL_ChangeFreq ( enum PCLclock clock );
```

**[Argument(s)]**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum PCLclock clock;</td>
<td>Output clock type</td>
</tr>
<tr>
<td></td>
<td>FPRS: fPRS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS2: fPRS/2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS4: fPRS/4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS8: fPRS/8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS16: fPRS/16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS32: fPRS/2048</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS64: fPRS/4096</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPRS128: fPRS/8192</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUBCLOCK: fSUB</td>
<td></td>
</tr>
</tbody>
</table>

**Remark**  fPRS is the main system clock frequency; fSUB is the subsystem clock frequency.

**[Return value]**

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
### C.3.12 LVI

Below is a list of API functions output by Code Generator for low-voltage detector use.

#### Table C-13. API Functions: [LVI]

<table>
<thead>
<tr>
<th>API Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVI_Init</td>
<td>Performs initialization necessary to control low-voltage detector functions.</td>
</tr>
<tr>
<td>LVI_UserInit</td>
<td>Performs user-defined initialization relating to the low-voltage detector.</td>
</tr>
<tr>
<td>LVI_INTERRUPTMODESTART</td>
<td>Starts low-voltage detection (when in interrupt generation mode).</td>
</tr>
<tr>
<td>LVI_ResetModeStart</td>
<td>Starts low-voltage detection (when in internal reset mode).</td>
</tr>
<tr>
<td>LVI_Stop</td>
<td>Stops low-voltage detection.</td>
</tr>
<tr>
<td>LVI_SetLVILevel</td>
<td>Sets the low-voltage detection level.</td>
</tr>
</tbody>
</table>
**LVI_Init**

Performs initialization necessary to control low-voltage detector functions.

[Classification]

CG_lvi.c

[Syntax]

```c
void LVI_Init ( void );
```

[Argument(s)]

None.

[Return value]

None.
**LVI_UserInit**

Performs user-defined initialization relating to the low-voltage detector.

**Remark** This API function is called as the LVI_Init callback routine.

**[Classification]**

CG_lvi_user.c

**[Syntax]**

```c
void LVI_UserInit ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
LVIInterruptModeStart

Starts low-voltage detection (when in interrupt generation mode).

[Classification]
CG_lvi.c

[Syntax]

```c
void LVIInterruptModeStart ( void );
```

[Argument(s)]
None.

[Return value]
None.

[Example]
The example below shows the detection of low voltage when the operation mode is interrupt generation mode (generate the interrupt INTLVI).

[CG_main.c]

```c
void main ( void ) {
    ......
    LVIInterruptModeStart ( ); /* Start low-voltage detection */
    ......
}
```

[CG_lvi_user.c]

```c
__interrupt void MD_INTLVI ( void ) { /* Interrupt processing for INTLVI */
    if ( LVIF == 1 ) { /* Trigger identification: Check LVIF flag */
        ...... /* Handle case when "power voltage (VDD) < detected voltage (VLVI)" detected */
    } else {
        ...... /* Handle case when "power voltage (VDD) >= detected voltage (VLVI)" detected */
    }
}
```
LVI_ResetModeStart

Starts low-voltage detection (when in internal reset mode).

[Classification]
CG_lvi.c

[Syntax]
MD_STATUS LVI_ResetModeStart ( void );

[Argument(s)]
None.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR</td>
<td>Exit with error (abend)</td>
</tr>
<tr>
<td></td>
<td>- The object of low voltage detection is external voltage (VDD), and power voltage (VDD) &lt;= detected voltage (VLVI).</td>
</tr>
<tr>
<td></td>
<td>- The object of low voltage detection is external input voltage (EXLVI), and external input voltage (EXLVI) &lt;= detected voltage (VEXLVI).</td>
</tr>
</tbody>
</table>
**[Function]**

**LVI_Stop**

Stops low-voltage detection.

**[Classification]**

CG_lvi.c

**[Syntax]**

```c
void LVI_Stop ( void );
```

**[Argument(s)]**

None.

**[Return value]**

None.
LVI_SetLVILevel

Sets the low-voltage detection level.

Remarks 1. To change the low-voltage detection level, you must call LVI_Stop before calling this API function.
2. The value specified in parameter level is set to low-voltage detection level select register (LVIS).

[Classification]
CG_lvi.c

[Syntax]

```c
#include "CG_macrodriver.h"
#include "CG_lvi.h"
MD_STATUS LVI_SetLVILevel ( enum LVILevel level );
```

[Argument(s)]

<table>
<thead>
<tr>
<th>I/O</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>enum LVILevel level;</td>
<td>Voltage level to detect as low voltage</td>
</tr>
</tbody>
</table>

| LVILEVEL0: | 4.22 V ± 0.1 V |
| LVILEVEL1: | 4.07 V ± 0.1 V |
| LVILEVEL2: | 3.92 V ± 0.1 V |
| LVILEVEL3: | 3.76 V ± 0.1 V |
| LVILEVEL4: | 3.61 V ± 0.1 V |
| LVILEVEL5: | 3.45 V ± 0.1 V |
| LVILEVEL6: | 3.30 V ± 0.1 V |
| LVILEVEL7: | 3.15 V ± 0.1 V |
| LVILEVEL8: | 2.99 V ± 0.1 V |
| LVILEVEL9: | 2.84 V ± 0.1 V |
| LVILEVEL10: | 2.68 V ± 0.1 V |
| LVILEVEL11: | 2.53 V ± 0.1 V |
| LVILEVEL12: | 2.38 V ± 0.1 V |
| LVILEVEL13: | 2.22 V ± 0.1 V |
| LVILEVEL14: | 2.07 V ± 0.1 V |
| LVILEVEL15: | 1.91 V ± 0.1 V |

Remark  LVILEVEL10 to LVILEVEL15 can only be specified when the target device is a 78K0/Kx2-L.

[Return value]

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD_OK</td>
<td>Normal completion</td>
</tr>
<tr>
<td>MD_ERROR</td>
<td>Exit with error (abend)</td>
</tr>
<tr>
<td>- The target of low-voltage detection is external input voltage (EXLVI) from the external input pin.</td>
<td></td>
</tr>
<tr>
<td>MD_ARGERROR</td>
<td>Invalid argument specification</td>
</tr>
</tbody>
</table>
Remark  The value MD_ERROR will only be returned when the target device is a 78K0/KB2-L or 78K0/KC2-L.
## APPENDIX D INDEX

### A
- AD_ComparatorOff  ...  185
- AD_ComparatorOn  ...  184
- A/D Converter  ...  181
  - AD_ComparatorOff  ...  185
  - AD_ComparatorOn  ...  184
  - AD_Init  ...  182
  - AD_Read  ...  189
  - AD_ReadByte  ...  190
  - AD_SelectADChannel  ...  188
  - AD_Start  ...  186
  - AD_Stop  ...  187
  - AD_UserInit  ...  183
  - AD_Init  ...  182
  - AD_Read  ...  189
  - AD_ReadByte  ...  190
  - AD_SelectADChannel  ...  188
  - AD_Start  ...  186
  - AD_Stop  ...  187
  - AD_UserInit  ...  183
  - AMPn_Start  ...  174
  - AMPn_Stop  ...  175
  - AMP_Start  ...  172
  - AMP_Stop  ...  173
  - API functions  ...  89
    - A/D Converter  ...  181
    - Clock Output  ...  258
    - Comparator  ...  176
    - Interrupt  ...  115
    - LVI  ...  264
    - Operational Amplifier  ...  166
    - Port  ...  108
    - Real-time Clock  ...  224
    - Serial  ...  126
    - System  ...  97
    - Timer  ...  191
    - Watchdog Timer  ...  222

### B
- Browse For Folder dialog box  ...  81

### C
- CG_ChangeClockMode  ...  101
- CG_ChangeFrequency  ...  103
- CG_ChangePllMode  ...  107
- CG_ReadResetSource  ...  100
- CG>SelectPowerSaveMode  ...  104
- CG_SelectStabTime  ...  106
- CLOCK_Init  ...  98
- Clock Output  ...  258
  - PCL_ChangeFreq  ...  263
  - PCL_Init  ...  259
  - PCL_Start  ...  261
  - PCL_Stop  ...  262
  - PCL_UserInit  ...  260
- CLOCK_UserInit  ...  99
- Code Generator panel  ...  69
- Code Generator Preview panel  ...  72
- Column Chooser dialog box  ...  77
- Comparator  ...  176
  - Comparator_Init  ...  177
  - Comparator_Start  ...  179
  - Comparator_Stop  ...  180
  - Comparator_UserInit  ...  178
- Comparator_Init  ...  177
- Comparator_Start  ...  179
- Comparator_Stop  ...  180
- Comparator_UserInit  ...  178
- CSI1n_Init  ...  141
  - CSI1n_ReceiveData  ...  145
  - CSI1n_ReceiveEndCallback  ...  150
  - CSI1n_SendEndCallback  ...  149
  - CSI1n_SendReceiveData  ...  147
  - CSI1n_Start  ...  143
  - CSI1n_Stop  ...  144
  - CSI1n_UserInit  ...  142
D
[Device Pin List Information] tab ... 49
Device Pin List panel ... 58
[External Peripheral] tab ... 64
[Macro] tab ... 62
[Pin Number] tab ... 60
Device Top View panel ... 66
[Device Top View Settings] tab ... 50

E
[External Peripheral] tab ... 64

F
[File Setting] tab ... 57
Functions ... 8, 22
  Code Generator ... 22
  Pin Configurator ... 8

G
[Generation] tab ... 53

I
IICA_GetStopConditionCallback ... 165
IICA_Init ... 151
IICA_MasterErrorCallback ... 159
IICA_MasterReceiveEndCallback ... 158
IICA_MasterReceiveStart ... 155
IICA_MasterSendEndCallback ... 157
IICA_MasterSendStart ... 154
IICA_SlaveErrorCallback ... 164
IICA_SlaveReceiveEndCallback ... 163
IICA_SlaveReceiveStart ... 161
IICA_SlaveSendEndCallback ... 162
IICA_SlaveSendStart ... 160
IICA_Stop ... 153
IICA_StopCondition ... 156
IICA_UserInit ... 152
Interrupt ... 115
  INT_MaskableInterruptEnable ... 120
  INTP_Init ... 116
  INTPn_Disable ... 122
  INTPn_Enable ... 123

K
KEY_Disable ... 124
KEY_Enable ... 125
KEY_Init ... 118
KEY_UserInit ... 119

L
LVI ... 264
  LVI_Init ... 265
  LVI_InterruptModeStart ... 267
  LVI_ResetModeStart ... 268
  LVI_SetLVILevel ... 270
  LVI_Stop ... 269
  LVI_UserInit ... 266
  LVI_Init ... 265
  LVI_InterruptModeStart ... 267
  LVI_ResetModeStart ... 268
  LVI_SetLVILevel ... 270
  LVI_Stop ... 269
  LVI_UserInit ... 266

M
[Macro Setting] tab ... 56
[Macro] tab ... 62
Main window ... 39

N
New Column dialog box ... 80

O
OPAMP_Init ... 167
OPAMP_UserInit ... 168
Operational Amplifier ... 166
  AMPn_Start ... 174
  AMPn_Stop ... 175
  AMP_Start ... 172
  AMP_Stop ... 173
  OPAMP_Init ... 167
  OPAMP_UserInit ... 168
  PGA_ChangePGAFactor ... 171
  PGA_Start ... 169
  PGA_Stop ... 170
Output panel ... 75

P
PCL_ChangeFreq ... 263
PCL_Init ... 259
PCL_Start ... 261
PCL_Stop ... 262
PCL_UserInit ... 260
PCL_ChangeFreq ... 263
PCL_Init ... 259
PCL_Start ... 261
PCL_Stop ... 262
[Pin Configurator Settings] tab ... 48
[Pin Number] tab ... 60
Port ... 108
  PORT_ChangePmnInput ... 111
  PORT_ChangePmnOutput ... 113
  PORT_Init ... 109
  PORT_UserInit ... 110
PORT_ChangePmnInput ... 111
PORT_ChangePmnOutput ... 113
PORT_Init ... 109
PORT_UserInit ... 110
Property panel ... 45
  [Device Pin List Information] tab ... 49
  [Device Top View Settings] tab ... 50
  [File Setting] tab ... 57
  [Generation] tab ... 53
  [Macro Setting] tab ... 56
  [Pin Configurator Settings] tab ... 48
R
Real-time Clock ... 224
  RTC_AlarmDisable ... 240
  RTC_AlarmEnable ... 239
  RTC_AlarmGet ... 244
  RTC_AlarmInterruptCallback ... 245
  RTC_AlarmSet ... 241
  RTC_ConstPeriodInterruptCallback ... 238
  RTC_ConstPeriodInterruptValue ... 257
  RTC_ConstPeriodInterruptDisable ... 237
  RTC_ConstPeriodInterruptEnable ... 236
  RTC.CounterDisable ... 229
  RTC.CounterEnable ... 228
  RTC.CounterGet ... 234
  RTC.CounterSet ... 232
  RTC_Init ... 225
  RTC.IntervalInterruptDisable ... 250
  RTC.IntervalInterruptEnable ... 248
  RTC.IntervalStart ... 246
  RTC.IntervalStop ... 247
  RTC_RTC1HZ_OutputDisable ... 252
  RTC_RTC1HZ_OutputEnable ... 251
  RTC_RTCCL_OutputDisable ... 254
  RTC_RTCCL_OutputEnable ... 253
  RTC_RTCDIV_OutputDisable ... 256
  RTC_RTCDIV_OutputEnable ... 255
  RTC.SetHourSystem ... 230
  RTC.UserInit ... 226
  RTC.PowerOff ... 227
  RTC_AlarmDisable ... 240
  RTC_AlarmEnable ... 239
  RTC_AlarmGet ... 244
  RTC_AlarmInterruptCallback ... 245
  RTC.AlarmSet ... 241
  RTC.ChangeCorrectionValue ... 257
  RTC.ConstPeriodInterruptCallback ... 238
  RTC.ConstPeriodInterruptDisable ... 237
  RTC.ConstPeriodInterruptEnable ... 236
  RTC.CounterDisable ... 229
  RTC.CounterEnable ... 228
  RTC.CounterGet ... 234
RTC.CounterSet ... 232
RTC.Init ... 225
RTC.IntervalInterruptDisable ... 250
RTC.IntervalInterruptEnable ... 248
RTC.IntervalStart ... 246
RTC.IntervalStop ... 247
RTC.PowerOff ... 227
RTC_RTC1HZ_OutputDisable ... 252
RTC_RTC1HZ_OutputEnable ... 251
RTC_RTCCL_OutputDisable ... 254
RTC_RTCCL_OutputEnable ... 253
RTC_RTCDIV_OutputDisable ... 256
RTC_RTCDIV_OutputEnable ... 255
RTC.SetHourSystem ... 230
RTC.UserInit ... 226

S
Save As dialog box ... 82
Serial ... 126
  CSI1n_Init ... 141
  CSI1n_ReceiveData ... 145
  CSI1n_ReceiveEndCallback ... 150
  CSI1n_SendEndCallback ... 149
  CSI1n_SendReceiveData ... 147
  CSI1n_Start ... 143
  CSI1n_Stop ... 144
  CSI1n_UserInit ... 142
  IICA_GetStopConditionCallback ... 165
  IICA_Init ... 151
  IICA_MasterErrorCallback ... 159
  IICA_MasterReceiveEndCallback ... 158
  IICA_MasterReceiveStart ... 155
  IICA_MasterSendEndCallback ... 157
  IICA_MasterSendStart ... 154
  IICA_SlaveErrorCallback ... 164
  IICA_SlaveReceiveEndCallback ... 163
  IICA_SlaveReceiveStart ... 161
  IICA_SlaveSendEndCallback ... 162
  IICA_SlaveSendStart ... 160
  IICA_Stop ... 153
  IICA_StopCondition ... 156

IICA_UserInit ... 152
UART6_ErrorCallback ... 139
UART6_Init ... 128
UART6.ReceiveData ... 134
UART6.ReceiveEndCallback ... 137
UART6.SendData ... 132
UART6.SendEndCallback ... 136
UART6_SoftOverRunCallback ... 138
UART6.Start ... 130
UART6_Stop ... 131
UART6_UserInit ... 129
System ... 97
  CG.ChangeClockMode ... 101
  CG.ChangeFrequency ... 103
  CG.ChangePllMode ... 107
  CG.ReadResetSource ... 100
  CG.SelectPowerSaveMode ... 104
  CG.SelectStabTime ... 106
  CLOCK_Init ... 98
  CLOCK_UserInit ... 99

T
Timer ... 191
  TM00.ChangeDuty ... 206
  TM00.ChangeTimerCondition ... 203
  TM00.GetFreeRunningValue ... 204
  TM00.GetPulseWidth ... 207
  TM00.Init ... 199
  TM00.SoftwareTriggerOn ... 205
  TM00_Start ... 201
  TM00_Stop ... 202
  TM00_UserInit ... 200
  TM5n.ChangeDuty ... 213
  TM5n.ChangeTimerCondition ... 212
  TM5n_Init ... 208
  TM5n_Start ... 210
  TM5n_Stop ... 211
  TM5n_UserInit ... 209
  TMH1.CarrierOutputDisable ... 221
  TMH1.CarrierOutputEnable ... 220
  TMHn.ChangeDuty ... 219
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep 01, 2012</td>
<td>First Edition issued</td>
</tr>
</tbody>
</table>