RX220 Group Peripheral Driver Generator Reference Manual

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corporation without notice. Please review the latest information published by Renesas Electronics Corporation through various means, including the Renesas Electronics Corporation website (http://www.renesas.com).

Renesas Electronics www.renesas.com

Rev.1.02 May 2014

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; anticrime 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 majorityowned subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Introduction

This manual was written to explain how to make the peripheral I/O drivers on the Peripheral Driver Generator for RX220. For the basic information about the Peripheral Driver Generator, refer to the Peripheral Driver Generator user's manual.



Table of Contents

In	troductio	on	3
Та	able of C	ontents	4
1.	Over	view	11
	1.1	Supported peripheral modules	11
	1.2	Tool requirements	
2.	Creat	ing a new project	12
3.	Settir	g Up the Peripheral Modules	13
	3.1	Main Window	
	3.2	Pin Functions (Multifunction Pin Controller)	
	3.2.1	[Pin function] Sheet	
	3.2.2		
	3.2.3	Peripheral-Module Settings Shared between the [Pin function] and [Peripheral pin usage] Sheets	
	3.2.4		
	3.3	Endian	
4.		ial	
	4.1	An LED blinking on a 8-bit timer (TMR) interrupt	26
	4.2	An LED blinking on the PWM output of the multi-function timer pulse unit 2 (MTU2a)	
	4.3	Continuously scanning on 12-Bit A/D converter (S12ADb)	
	4.4	Triggering DTCa by ICUb	
	4.5	Data transfer between SCIe channels 1 and 5	
5.		fication of Generated Functions	
	5.1	Clock-Generation Circuit	
	5.1.1	R PG Clock Set	
	5.1.2		
	5.1.2	R_PG_Clock_Start_MAIN	
	5.1.4		
	5.1.5		
	5.1.6	R_PG_Clock_Stop_SUB	
	5.1.7	R_PG_Clock_Start_LOCO	
	5.1.8		
	5.1.9	-	
	5.1.1	0 R_PG_Clock_Stop_HOCO	84
	5.1.1	-	
	5.1.1		
	5.1.1	3 R_PG_Clock_Enable_MAIN_StopDetection	87
	5.1.1	-	
	5.1.1	-	
	5.1.1	-	
	5.1.1		
	5.1.1		

5.1.19	R_PG_Clock_GetHOCOPowerStatus	
5.2	Voltage Detection Circuit (LVDAa)	
5.2.1	R_PG_LVD_Set	
5.2.2	R_PG_LVD_GetStatus	
5.2.3	R_PG_LVD_ClearDetectionFlag_LVD <voltage circuit="" detection="" number=""></voltage>	
5.2.4	R_PG_LVD_Disable_LVD <voltage circuit="" detection="" number=""></voltage>	
5.3	Clock Frequency Accuracy Measurement Circuit (CAC)	
5.3.1	R_PG_CAC_Set	
5.3.2	R_PG_CAC_ClearFlag_FrequencyError	
5.3.3	R_PG_CAC_ClearFlag_MeasurementEnd	
5.3.4	R_PG_CAC_ClearFlag_Overflow	
5.3.5	R_PG_CAC_StartMeasurement	
5.3.6	R_PG_CAC_StopMeasurement	
5.3.7	R_PG_CAC_GetStatusFlags	
5.3.8	R_PG_CAC_GetCounterBufferRegister	
5.3.9	R_PG_CAC_StopModule	
5.4	Low Power Consumption	
5.4.1	R_PG_LPC_Set	
5.4.2	R_PG_LPC_Sleep	
5.4.3	R_PG_LPC_AllModuleClockStop	
5.4.4	R_PG_LPC_SoftwareStandby	
5.4.5	R_PG_LPC_ChangeOperatingPowerControl	
5.4.6	R_PG_LPC_ChangeSleepModeReturnClock	
5.4.7	R_PG_LPC_GetPowerOnResetFlag	
5.4.8	R_PG_LPC_GetLVDDetectionFlag	
5.4.9	R_PG_LPC_GetOperatingPowerControlFlag	
5.4.10	R_PG_LPC_GetStatus	
5.5	Register Write Protection Function	
5.5.1	R_PG_RWP_RegisterWriteCgc	
5.5.2	R_PG_RWP_RegisterWriteModeLpcReset	
5.5.3	R_PG_RWP_RegisterWriteLvd	
5.5.4	R_PG_RWP_RegisterWriteMpc	
5.5.5	R_PG_RWP_GetStatusCgc	
5.5.6	R_PG_RWP_GetStatusModeLpcReset	
5.5.7	R_PG_RWP_GetStatusLvd	
5.5.8	R_PG_RWP_GetStatusMpc	
5.6	Interrupt Controller (ICUb)	126
5.6.1	R_PG_ExtInterrupt_Set_< <i>interrupt type</i> >	
5.6.2	R_PG_ExtInterrupt_Disable_ <interrupt type=""></interrupt>	
5.6.3	R_PG_ExtInterrupt_GetRequestFlag_ <interrupt type=""></interrupt>	
5.6.4	R_PG_ExtInterrupt_ClearRequestFlag_ <interrupt type=""></interrupt>	
5.6.5	R_PG_ExtInterrupt_EnableFilter_< <i>interrupt type</i> >	
5.6.6	R_PG_ExtInterrupt_DisableFilter_< <i>interrupt type</i> >	
5.6.7	R_PG_SoftwareInterrupt_Set	
5.6.8	R_PG_SoftwareInterrupt_Generate	

5.6.9	R_PG_FastInterrupt_Set	. 135
5.6.10	R_PG_Exception_Set	. 136
5.7 B	uses	. 137
5.7.1	R_PG_ExtBus_PresetBus	. 137
5.7.2	R_PG_ExtBus_SetBus	. 138
5.7.3	R_PG_ExtBus_GetErrorStatus	. 139
5.7.4	R_PG_ExtBus_ClearErrorFlags	. 140
5.8 D	MA controller (DMACA)	. 141
5.8.1	R_PG_DMAC_Set_C< <i>channel number</i> >	. 141
5.8.2	R_PG_DMAC_Activate_C< <i>channel number</i> >	. 144
5.8.3	R_PG_DMAC_StartTransfer_C <channel number=""></channel>	. 145
5.8.4	R_PG_DMAC_StartContinuousTransfer_C <channel number=""></channel>	. 146
5.8.5	R_PG_DMAC_StopContinuousTransfer_C <channel number=""></channel>	. 147
5.8.6	R_PG_DMAC_Suspend_C <channel number=""></channel>	. 148
5.8.7	R_PG_DMAC_GetTransferCount_C <channel number=""></channel>	. 149
5.8.8	R_PG_DMAC_SetTransferCount_C< <i>channel number</i> >	. 150
5.8.9	R_PG_DMAC_GetRepeatBlockSizeCount_C <channel number=""></channel>	. 151
5.8.10	R_PG_DMAC_SetRepeatBlockSizeCount_C <channel number=""></channel>	. 152
5.8.11	R_PG_DMAC_ClearInterruptFlag_C <channel number=""></channel>	. 153
5.8.12	R_PG_DMAC_GetTransferEndFlag_C <channel number=""></channel>	. 154
5.8.13	R_PG_DMAC_ClearTransferEndFlag_C <channel number=""></channel>	. 155
5.8.14	R_PG_DMAC_GetTransferEscapeEndFlag_C <channel number=""></channel>	. 156
5.8.15	R_PG_DMAC_ClearTransferEscapeEndFlag_C <channel number=""></channel>	. 157
5.8.16	R_PG_DMAC_SetSrcAddress_C <channel number=""></channel>	. 158
5.8.17	R_PG_DMAC_SetDestAddress_C< <i>channel number</i> >	. 159
5.8.18	R_PG_DMAC_SetAddressOffset_C< <i>channel number</i> >	. 160
5.8.19	R_PG_DMAC_SetExtendedRepeatSrc_C< <i>channel number</i> >	. 161
5.8.20	R_PG_DMAC_SetExtendedRepeatDest_C <channel number=""></channel>	. 162
5.8.21	R_PG_DMAC_StopModule_C< <i>channel number</i> >	. 163
5.9 D	Pata Transfer Controller (DTCa)	. 164
5.9.1	R_PG_DTC_Set	. 164
5.9.2	R_PG_DTC_Set_< <i>trigger source</i> >	. 165
5.9.3	R_PG_DTC_Activate	. 167
5.9.4	R_PG_DTC_SuspendTransfer	. 168
5.9.5	R_PG_DTC_GetTransmitStatus	. 169
5.9.6	R_PG_DTC_StopModule	. 170
5.10 E	vent Link Controller (ELC)	. 171
5.10.1	R_PG_ELC_Set	. 171
5.10.2	R_PG_ELC_SetLink_ <peripheral module=""></peripheral>	. 172
5.10.3	R_PG_ELC_DisableLink_ <peripheral module=""></peripheral>	. 173
5.10.4	R_PG_ELC_Set_PortGroup <pre>port group number></pre>	
5.10.5	R_PG_ELC_Set_SinglePort <single-port number=""></single-port>	. 175
5.10.6	R_PG_ELC_AllEventLinkEnable	
5.10.7	R_PG_ELC_AllEventLinkDisable	. 177
5.10.8	R_PG_ELC_Generate_SoftwareEvent	. 178

5.10.9	R_PG_ELC_GetPortBufferValue_Group <port-group number=""></port-group>	179
5.10.10	R_PG_ELC_SetPortBufferValue_Group <port-group number=""></port-group>	180
5.10.11	R_PG_ELC_StopModule	181
5.11 I/	O Ports	182
5.11.1	R_PG_IO_PORT_Set_P <port number=""></port>	182
5.11.2	R_PG_IO_PORT_Set_P <port number=""><pin number=""></pin></port>	183
5.11.3	R_PG_IO_PORT_Read_P <pre>port number></pre>	184
5.11.4	R_PG_IO_PORT_Read_P <pre>port number><pin number=""></pin></pre>	185
5.11.5	R_PG_IO_PORT_Write_P <port number=""></port>	186
5.11.6	R_PG_IO_PORT_Write_P <port number=""><pin number=""></pin></port>	187
5.11.7	R_PG_IO_PORT_SetPortNotAvailable	188
5.12 N	Iulti-Function Timer Pulse Unit 2 (MTU2a)	189
5.12.1	R_PG_Timer_Set_MTU_U <unit number="">_<channels></channels></unit>	189
5.12.2	R_PG_Timer_StartCount_MTU_U <unit number="">_C<channel number="">(_<phase>)</phase></channel></unit>	191
5.12.3	R_PG_Timer_SynchronouslyStartCount_MTU_U <unit number=""></unit>	192
5.12.4	R_PG_Timer_HaltCount_MTU_U <unit number="">_C<channel number="">(_<phase>)</phase></channel></unit>	193
5.12.5	R_PG_Timer_GetCounterValue_MTU_U <unit number="">_C<channel number=""></channel></unit>	194
5.12.6	R_PG_Timer_SetCounterValue_MTU_U <unit number="">_C<channel number="">(_<phase>)</phase></channel></unit>	195
5.12.7	R_PG_Timer_GetRequestFlag_MTU_U <unit number="">_C<channel number=""></channel></unit>	196
5.12.8	R_PG_Timer_StopModule_MTU_U <unit number=""></unit>	198
5.12.9	R_PG_Timer_GetTGR_MTU_U <unit number="">_C<channel number=""></channel></unit>	199
5.12.10	R_PG_Timer_SetTGR_ <general register="">_MTU_U<unit number="">_C<channel number=""></channel></unit></general>	201
5.12.11	R_PG_Timer_SetBuffer_AD_MTU_U <unit number="">_C<channel number=""></channel></unit>	202
5.12.12	R_PG_Timer_SetBuffer_CycleData_MTU_U <unit number="">_<channels></channels></unit>	203
5.12.13	R_PG_Timer_SetOutputPhaseSwitch_MTU_U <unit number="">_<channels></channels></unit>	204
5.12.14	R_PG_Timer_ControlOutputPin_MTU_U <unit number="">_<channels></channels></unit>	205
5.12.15	R_PG_Timer_SetBuffer_PWMOutputLevel_MTU_U <unit number="">_<channels></channels></unit>	206
5.12.16	R_PG_Timer_ControlBufferTransfer_MTU_U <unit number="">_<channels></channels></unit>	207
5.13 P	ort Output Enable 2 (POE2a)	208
5.13.1	R_PG_POE_Set	208
5.13.2	R_PG_POE_SetHiZ_< <i>Timer channels</i> >	209
5.13.3	R_PG_POE_GetRequestFlagHiZ_< <i>Timer channels/flag</i> >	210
5.13.4	R_PG_POE_GetShortFlag_ <timer channels=""></timer>	211
5.13.5	R_PG_POE_ClearFlag_< <i>Timer channels/flag</i> >	212
5.14 8	-Bit Timer (TMR)	213
5.14.1	R_PG_Timer_Start_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	213
5.14.2	R_PG_Timer_HaltCount_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	215
5.14.3	R_PG_Timer_ResumeCount_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	216
5.14.4	R_PG_Timer_GetCounterValue_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	217
5.14.5	R_PG_Timer_SetCounterValue_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	218
5.14.6	R_PG_Timer_GetRequestFlag_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	219
5.14.7	R_PG_Timer_HaltCountElc_TMR_U <unit number="">_C<channel number=""></channel></unit>	220
5.14.8	$R_PG_Timer_GetCountStateElc_TMR_U<\!\mathit{unit\ number}>_C<\!\mathit{channel\ number}>\ldots$	221
5.14.9	R_PG_Timer_StopModule_TMR_U <unit number=""></unit>	222
5.15 C	ompare Match Timer (CMT)	223

R_PG_Timer_Set_CMT_U <unit number="">_C<channel number=""></channel></unit>	223
R_PG_Timer_StartCount_CMT_U <unit number="">_C<channel number=""></channel></unit>	224
R_PG_Timer_HaltCount_CMT_U <unit number="">_C<channel number=""></channel></unit>	225
R_PG_Timer_GetCounterValue_CMT_U <unit number="">_C<channel number=""></channel></unit>	
R_PG_Timer_SetCounterValue_CMT_U <unit number="">_C<channel number=""></channel></unit>	
R_PG_Timer_StopModule_CMT_U <unit number=""></unit>	
Realtime Clock (RTCc)	229
R_PG_RTC_Start	229
R_PG_RTC_WarmStart	
R_PG_RTC_Stop	
R_PG_RTC_Restart	
R_PG_RTC_SetCurrentTime	
R_PG_RTC_GetStatus	
R_PG_RTC_Adjust30sec	
R_PG_RTC_ManualErrorAdjust	
R_PG_RTC_Set24HourMode	
R_PG_RTC_Set12HourMode	
R_PG_RTC_AutoErrorAdjust_Enable	
R_PG_RTC_AutoErrorAdjust_Disable	
-	
R_PG_RTC_SetAlarmTime	
R_PG_RTC_SetPeriodicInterrupt	
-	
-	
R_PG_RTC_SetCurrentTimeBinary	
-	
•	
-	
ndependent Watchdog Timer (IWDTa)	
R_PG_Timer_GetStatus_IWDT	
R_PG_SCI_Set_C< <i>channel number</i> >	
R_PG_SCI_StartSending_C< <i>channel number</i> >	
	R_PG_Timer_StartCount_CMT_U <unit number="">_C<channel number=""> R_PG_Timer_HaltCount_CMT_U<unit number="">_C<channel number=""> R_PG_Timer_GetCounterValue_CMT_U<unit number="">_C<channel number=""> R_PG_Timer_StopModul_CMT_U<unit number="">_C<channel number=""> R_PG_Timer_StopModul_CMT_U<unit number=""> R_PG_RTC_StopModul_CMT_U<unit number=""> R_PG_RTC_Start R_PG_RTC_Start R_PG_RTC_Start R_PG_RTC_Start R_PG_RTC_SetCurrentTime R_PG_RTC_GetStatus R_PG_RTC_GetStatus R_PG_RTC_GetStatus R_PG_RTC_GetStatus R_PG_RTC_Set2HourMode P_G_RTC_Set2HourMode P_G_RTC_Set2HourMode R_PG_RTC_Set2HourMode R_PG_RTC_AutoErrorAdjust_Enable. 2 R_PG_RTC_AutoErrorAdjust_Enable 3 R_PG_RTC_SetAlarmTime 5 R_PG_RTC_SetAlarmTime 5 R_PG_RTC_SetAlarmTime 5 R_PG_RTC_SetAlarmTime 5 R_PG_RTC_StartBinary 9 R_PG_RTC_StartBin</unit></unit></channel></unit></channel></unit></channel></unit></channel></unit>

5.18.5	R_PG_SCI_I2CMode_Send_C< <i>channel number</i> >	268
5.18.6	R_PG_SCI_I2CMode_SendWithoutStop_C< <i>channel number</i> >	269
5.18.7	R_PG_SCI_I2CMode_GenerateStopCondition_C< <i>channel number</i> >	270
5.18.8	R_PG_SCI_I2CMode_Receive_C< <i>channel number</i> >	271
5.18.9	R_PG_SCI_I2CMode_RestartReceive_C< <i>channel number</i> >	272
5.18.10	R_PG_SCI_I2CMode_ReceiveLast_C <channel number=""></channel>	274
5.18.11	R_PG_SCI_I2CMode_GetEvent_C< <i>channel number</i> >	276
5.18.12	R_PG_SCI_SPIMode_Transfer_C< <i>channel number</i> >	277
5.18.13	R_PG_SCI_SPIMode_GetErrorFlag_C <channel number=""></channel>	278
5.18.14	R_PG_SCI_GetSentDataCount_C <channel number=""></channel>	279
5.18.15	R_PG_SCI_ReceiveStationID_C< <i>channel number</i> >	280
5.18.16	R_PG_SCI_StartReceiving_C< <i>channel number</i> >	281
5.18.17	R_PG_SCI_ReceiveAllData_C <channel number=""></channel>	282
5.18.18	R_PG_SCI_ControlClockOutput_C< <i>channel number</i> >	283
5.18.19	R_PG_SCI_StopCommunication_C< <i>channel number</i> >	284
5.18.20	R_PG_SCI_GetReceivedDataCount_C <channel number=""></channel>	285
5.18.21	R_PG_SCI_GetReceptionErrorFlag_C< <i>channel number</i> >	286
5.18.22	R_PG_SCI_ClearReceptionErrorFlag_C <channel number=""></channel>	287
5.18.23	R_PG_SCI_GetTransmitStatus_C <channel number=""></channel>	288
5.18.24	R_PG_SCI_StopModule_C< <i>channel number</i> >	289
5.19 I ²	C Bus Interface (RIIC)	290
5.19.1	R_PG_I2C_Set_C< <i>channel number</i> >	290
5.19.2	R_PG_I2C_MasterReceive_C <channel number=""></channel>	291
5.19.3	R_PG_I2C_MasterReceiveLast_C <channel number=""></channel>	293
5.19.4	R_PG_I2C_MasterSend_C< <i>channel number</i> >	295
5.19.5	R_PG_I2C_MasterSendWithoutStop_C< <i>channel number</i> >	297
5.19.6	R_PG_I2C_GenerateStopCondition_C <channel number=""></channel>	299
5.19.7	R_PG_I2C_GetBusState_C< <i>channel number</i> >	300
5.19.8	R_PG_I2C_SlaveMonitor_C< <i>channel number</i> >	. 301
5.19.9	R_PG_I2C_SlaveSend_C< <i>channel number</i> >	303
5.19.10	R_PG_I2C_GetDetectedAddress_C< <i>channel number</i> >	304
5.19.11	R_PG_I2C_GetTR_C< <i>channel number</i> >	305
5.19.12	R_PG_I2C_GetEvent_C< <i>channel number</i> >	306
5.19.13	R_PG_I2C_GetReceivedDataCount_C< <i>channel number</i> >	. 307
5.19.14	R_PG_I2C_GetSentDataCount_C< <i>channel number</i> >	. 308
5.19.15	R_PG_I2C_Reset_C< <i>channel number</i> >	. 309
5.19.16	R_PG_I2C_StopModule_C< <i>channel number</i> >	310
5.20 Se	erial Peripheral Interface (RSPI)	. 311
5.20.1	R_PG_RSPI_Set_C< <i>channel number</i> >	. 311
5.20.2	R_PG_RSPI_SetCommand_C< <i>channel number</i> >	
5.20.3	R_PG_RSPI_StartTransfer_C< <i>channel number</i> >	
5.20.4	R_PG_RSPI_TransferAllData_C< <i>channel number</i> >	
5.20.5	R_PG_RSPI_GetStatus_C< <i>channel number</i> >	
5.20.6	R_PG_RSPI_GetError_C< <i>channel number</i> >	
5.20.7	R_PG_RSPI_GetCommandStatus_C< <i>channel number</i> >	

5.20.8 R_PG_RSPI_LoopBack <loopback mode="">_C<channel number=""></channel></loopback>	
5.20.9 R_PG_RSPI_StopModule_C <channel number=""></channel>	
5.21 CRC Calculator (CRC)	
5.21.1 R_PG_CRC_Set	
5.21.2 R_PG_CRC_InputData	323
5.21.3 R_PG_CRC_GetResult	
5.21.4 R_PG_CRC_StopModule	325
5.22 12-Bit A/D Converter (S12ADb)	
5.22.1 R_PG_ADC_12_Set_S12AD0	326
5.22.2 R_PG_ADC_12_StartConversionSW_S12AD0	327
5.22.3 R_PG_ADC_12_StopConversion_S12AD0	
5.22.4 R_PG_ADC_12_GetResult_S12AD0	
5.22.5 R_PG_ADC_12_GetResult_DblTrigger_S12AD0	
5.22.6 R_PG_ADC_12_GetResult_SelfDiag_S12AD0	
5.22.7 R_PG_ADC_12_StopModule_S12AD0	
5.23 Comparator A (CMPA)	
5.23.1 R_PG_CPA_Set_CP< <i>comparator circuit number</i> >	335
5.23.2 R_PG_CPA_Disable_CP< <i>comparator circuit number</i> >	
5.23.3 R_PG_CPA_GetStatus	
5.24 Data Operation Circuit (DOC)	
5.24.1 R_PG_DOC_Set	
5.24.2 R_PG_DOC_GetStatusFlag	339
5.24.3 R_PG_DOC_GetResult	
5.24.4 R_PG_DOC_InputData	
5.24.5 R_PG_DOC_UpdateData	
5.24.6 R_PG_DOC_StopModule	
5.25 Notes on Notification Functions	
5.25.1 Interrupts and processor mode	
5.25.2 Interrupts and DSP instructions	
6. Registering Files with the IDE and Building Them	
Appendix 1. Pin Functions for which the Allocation Can be Changed	



1. Overview

1.1 Supported peripheral modules

The Peripheral Driver Generator supports the following products of RX220 group, peripheral modules and endian.

(1) Products

Part No.	Package	Part No.	Package	
R5F52206BxFP	PLQP0100KB	R5F52203BxFP	PLQP0100KB	
R5F52206BxFM	PLQP0064KB	R5F52203BxFM	PLQP0064KB	
R5F52206BxFL	PLQP0048KB	R5F52203BxFL	PLQP0048KB	
R5F52205BxFP	PLQP0100KB	R5F52201BxFM	PLQP0064KB	
R5F52205BxFM	PLQP0064KB	R5F52201BxFL	PLQP0048KB	
R5F52205BxFL	PLQP0048KB			

(2) Peripheral Modules

Voltage Detection Circuit (LVDAa)	Port Output Enable 2 (POE2a)
Clock Generation Circuit	8-Bit Timer (TMR)
Clock Frequency Accuracy Measurement Circuit (CAC)	Compare Match Timer (CMT)
Low Power Consumption	Realtime Clock (RTCc)
Register Write Protection Function	Independent Watchdog Timer (IWDTa)
Interrupt Controller (ICUb), Exceptions	Serial Communications Interface (SCIe,SCIf)
Buses	I ² C Bus Interface (RIIC)
DMA Controller (DMACA)	Serial Peripheral Interface (RSPI)
Data Transfer Controller (DTCa)	CRC Calculator (CRC)
Event Link Controller (ELC)	12-Bit A/D Converter (S12ADb)
I/O Ports	Comparator A (CMPA)
Multifunction Pin Controller (MPC)	Data Operation Circuit (DOC)
Multi-Function Timer Pulse Unit 2 (MTU2a)	

The IrDA Interface is not supported.

The binary-count mode of Realtime Clock (RTCc) is not supported.

(3) Endian

Little Big

1.2 Tool requirements

The following tools are required for this version of RX220 group Peripheral Driver Generator.

- RX Family C/C++ Compiler Package V.1.02 Release 01
- RX220 Group Renesas Peripheral Driver Library V.1.10 (Bundled in Peripheral Driver Generator)



2. Creating a new project

To create the new project file, select the menu [File] -> [New Project]. New project dialog box will open.

Project new	—
Project name:	
default	
Directory:	
c:\renesas\PDG2_pro	Browse
Device selection	
Series:	RX200 -
Group:	RX220 -
Part No.:	R5F52206BxFP -
Package:	PLQP0100KB-A
ROM capacity:	256K byte(s)
RAM capacity:	16K byte(s)
	OK Cancel

Fig 2.1 New project dialog box

For RX220 group, select [RX200] as a series and select [RX220] as a group. The package type, ROM capacity and RAM capacity of selected product are displayed.

By clicking [OK], new project is created and opened.

The EXTAL input clock frequency is not set after opening a new project. Therefore an error icon is displayed. For error display, refer to the user's manual.

😵 PDG2 - [default.pd2]		- • •
Eile View Tool Help		_ 8 ×
] 🗅 🚅 🖬 [🎦 🐌 🐼 [🎦 ?		
System	Internal clock source: Main clock oscillator Main clock oscillator settings Use the main clock oscillator Main clock oscillation source: Resonator Main clock (EXTAL input) frequency: MHz Oscillator drive capability: 16-MHz to 20-MHz (lead type ceramic resonator) Main clock oscillation stop detection: Disable Sub-clock oscillator drive ability: Use RTC-dedicated clock (supplied by the sub-clock oscillator) Sub-clock oscillator drive ability: Drive ability for low CL Resonator frequency: 0.032768	E
SYSTEM LVDAa CAC LPC ICUb Buses D	Resonator frequency: MACA DTCa ELC I/O MTUZa POEZa TMR CMT RTCC IV/DTa SCI RIIC RSPI CRC S12ADb CMPA DOC	-
Generated Source File Information		Φ×
Ready		CAP NUM SCRL

Fig 2.2 Error display of new project

Set the frequency of the clock to be used here.



3. Setting Up the Peripheral Modules

3.1 Main Window

Figure 3.1 shows the main window for setting up peripheral modules.

System Clock Generation Circuit Clock Generation Circuit Option setting Register Write Protection Function Resource pane	Internal clock source: Main clock oscillator Main clock oscillator settings Use the main clock oscillator Main clock oscillation source: Resonator Main clock (EXTAL input) frequency: 20 MHz Oscillator drive capability: 16MHz to 20MHz Gscillator drive capability: 16MHz to 20MHz Gscillator drive scapability: 16MHz to 20MHz (16MHz to 20MHz (lead type ceramic resonator) Main clock oscillation stop detection: Disable	E
SYSTEM LVDAa CAC LPC ICUb Buses DMA	Sub-clock oscillator settings Use RTC-dedicated clock (supplied by the sub-clock oscillator) Sub-clock oscillator drive ability: Resonator frequency: CA DTCa ELC 1/0 Peripheral-module selection tabs Doc	4

Figure 3.1 Display in the Main Window (Example)

Table 3.1 shows the correspondence between the peripheral-module selection tabs, items in the resource pane, and peripheral modules to be set up.

Tab	Resource pane	Corresponding Peripheral Module or Function
SYSTEM	Clock Generation Circuit	Clock Generation Circuit
	Pin(Multifunction pin controller)	Pinfunctions (Multifunction Pin Controller (MPC))
	Option setting	Endian setting
	Register Write Protection Function	Register Write Protection Function
LVDAa	Voltage monitoring 0 to 2	Voltage monitoring 0 to 2
CAC	Clock frequency accuracy	Clock Frequency Accuracy Measurement Circuit (CAC)
	measurement circuit (CAC)	
LPC	Low Power Consumption	Low Power Consumption
ICUb	Interrupts	Interrupt Control Unit (ICUb)
		(Fastinterrupt, Software Interrupt, External Interrupt (NMI, IRQ0 to
		IRQ7))
	Exceptions	Exceptions
Buses	Common settings	Bus Priority and Bus Error Monitoring
DMACA	DMACA0 to DMACA3	DMA Controller (DMACA) Channel 0 to 3
DTCa	Data transfer controller (DTCa)	Data Transfer Controller (DTCa)
ELC	Event link settings	Event Link Controller (ELC) event link settings
	Port group and single port settings	Event Link Controller (ELC) port group and single port settings
I/O	Port 0 to Port J	I/O Port 0 to J
MTU2a	Unit0 (MTU0 to MTU5)	Multi-Function Timer Pulse Unit 2 (MTU2a) Channlel 0 to 5
POE2a	POE2a	Port Output Enable 2 (POE2a)

 Table 3.1
 Peripheral-Module Selection Tabs, Items in the Resource Pane, and Peripheral Modules



TMR	Unit0 (TMR0 and TMR1)	8-Bit Timer (TMR) Unit 0 (Channlel 0 and 1)
	Unit1 (TMR2 and TMR3)	8-Bit Timer (TMR) Unit 1 (Channlel 2 and 3)
CMT	Unit0 (CMT0 and CMT1)	Compare Match Timer (CMT) Unit 0 (Channlel 0 and 1)
	Unit1 (CMT2 and CMT3)	Compare Match Timer (CMT) Unit 1 (Channlel 2 and 3)
RTCc	Realtime Clock (RTCc)	Realtime Clock (RTCc)
IWDTa	Independent Watchdog Timer	Independent Watchdog Timer (IWDTa)
	(IWDTa)	
SCI	SCI 1, 5, 6, 9, 12	Serial Communications Interface SCIe(SCI1,5,6,9), SCIf(SCI12)
RIIC	RIICO	I ² C Bus Interface (RIIC)
RSPI	RSPI0	Serial Peripheral Interface (RSPI)
CRC	CRC Calculator (CRC)	CRC Calculator (CRC)
S12ADb	S12AD0	12-Bit A/D Converter (S12ADb)
CMPA	Comparator A (CMPA)	Comparator A (CMPA)
DOC	Data Operation Circuit (DOC)	Data Operation Circuit (DOC)

For how to set up the peripheral modules, refer to the user's manual. For details on the setting of pin functions, refer to section 3.2, Pin Functions.



3.2 Pin Functions (Multifunction Pin Controller)

The multifunction pin controller (MPC) in RX220-group MCUs selects the functions to be assigned to individual pins. The Peripheral Driver Generator provides a pin-function pane through which settings for the MPC can be made.

Select the [SYSTEM] tab from the peripheral-module selection tabs and click on [Pin (Multi function pin controller)] in the resource pane to open the pin-function pane.

B- System	Pin No.	Pin name	Selected function	Direc	State		-
Clock Generation Circuit	1	NC					=
Pin (Multifunction pin controller)	2	P03	Not assigned				
Option security Option security	3	NC					
Register Write Protection Function	4	PJ3/MTIOC3C/CTS6#/RTS6#/SS6#	Not assigned				
	5	VCL	VCL				
	6	PJ1/MTIOC3A	Not assigned				
	7	MD/FINED	MD	Input			
	8	XCIN					
	9	XCOUT					
	10	RES#	RES#	Input			
	11	XTAL/P37	XTAL	Output			
	12	VSS	VSS	Input			
	13	EXTAL/P36	EXTAL	Input			
	14	VCC	VCC	Input			-
_	Pin functi	on Peripheral pin usage Pin layout				_	4 0
YSTEM LVDAa CAC LPC ICUb Buses DMA	CA DTCa	ELC I/O MTU2a POE2a TMR CMT RTCc I	WDTa SCI RIIC	RSPI CF	RC S12ADb CMPA		4 1

Figure 3.2 Opening the Pin-Function Pane

The pin-function pane has [Pin function] and [Peripheral pin usage] sheets. The two sheets are linked, so that settings can be made in either of them.

3.2.1 [Pin function] Sheet

(1) Configuration

The [Pin function] sheet shows all of the MCU pins in order and the functions that have been assigned to those pins. This sheet can be used to select functions for each of the pins with multiplexed functions.

Pin No.	Pin name	Selected function	Direc	State	
1	NC				
2	P03	Not assigned			
3	NC				
4	PJ3/MTIOC3C/CTS6#/RTS6#/SS6#	Not assigned			
5	VCL	VCL			
6	PJ1/MTIOC3A	Not assigned			
7	MD/FINED	MD	Input		
8	XCIN				
9	XCOUT				
10	RES#	RES#	Input		
11	XTAL/P37	XTAL	Output		
12	VSS	VSS	Input		
13	EXTAL/P36	EXTAL	Input		
14	VCC	VCC	Input		

Figure 3.3 Pin-Function Pane ([Pin function] Sheet)

The contents of each column are shown in table 3.2.

Column	Description
Pin No.	Pin number
Pin name	Name of the pin (which shows all of the functions assigned to that pin)
Selected function	Currently allocated pin function
Direction	Whether the pin function is an input or output
State	Warning or error message, if any

Table 3.2 Columns on the [Pin function] Sheet

(2) Default State

By default (i.e. when no pins have been set up for use with peripheral modules), "Not assigned" is shown in the [Selected function] column for each port pin, indicating that no function has yet been selected (figure 3.4).

[Pin No.	Pin name	Selected function	Direction	State
	15	P35/NMI	Not assigned		



Note:

Port pins of RX220-group MCUs are general-purpose input port pins by default. Even though "Not assigned" is shown in the [Selected function] column for each port pin by default (i.e. when no pins have been set up for use with peripheral modules), the pin will act as a general-purpose input port pin. When you designate a pin as a general-purpose input port pin in the [I/O] pane, the name of the general-purpose input port pin will appear in the [Selected function] column (figure 3.5(b)).

Pin No.	Pin name	Selected function	Direction	State
100	P05	Not assigned		

(a) Default State

Pin No.	Pin name	Selected function	Direction	State
100	P05	P05	Input	

(b) After Designating P05 as a General-Purpose Input Port Pin in the [I/O] Pane

Figure 3.5 Display for Pin P05 (100-Pin LQFP Package)

(3) Selecting a Pin Function

When a pin has multiplexed functions, placing the mouse pointer on the [Selected function] column in the row for that pin brings up a drop-down button. Clicking on the button brings up a list of selectable pin functions (figure 3.6).

Pin No.	Pin name	Selected function	Direction	State
15	P35/NMI	Not assigned 💌		
		Not assigned		
		P35	ľ	
		NMI		

Figure 3.6 Selectable Pin Functions



In the default state (i.e. when no pins have been set up for use with peripheral modules), if [Selected function] is changed from "Not assigned" to another pin function, the warning [<Name of the pin function> has not been configured in the peripheral settings.] appears. For example, when [Selected function] for P35/NMI is changed from "Not assigned" to NMI despite the interrupt controller (ICUA) not being set up, a warning appears as shown in figure 3.7.

Pin No.	Pin name	Selected function	Direction	State
15	P35/NMI	NMI		NMI has not been configured in the peripheral settings.

Figure 3.7 Warning on Changing [Selected function] in the Default State

When the NMI has been set up in the [ICUA] pane, the warning disappears and "NMI" appears in the [Selected function] column.

Pin No.	Pin name	Selected function	Direction	State
15	P35/NMI	NMI	Input	

Figure 3.8 After Setting the NMI up

Note:

The generation of source files is still possible when the warning shown in figure 3.7 is being displayed, but the pin will not act as an NMI. For details, refer to section 3.2.4, Error Messages and Warnings on Pin Settings.

(4) Selecting a Pin Function before Setting up the Associated Peripheral Module

When a peripheral module is set up after selecting the pin functions on the [Pin function] sheet, the selected pin functions are automatically allocated to the pins.

IRQ5, for example, can be assigned to P15, PA4, PD5, or PE5 (P15, PA4, or PE5 for products in 64-pin packages, P15, or PA4 for productions in 48-pin packages). To assign IRQ5 to PE5, IRQ5 should be selected as the [Selected function] for PE5 on the [Pin function] sheet (figure 3.9).

Pin No.	Pin name	Selected function	Direction	State
13	PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	IRQ5		IRQ5 has not been configured in the peripheral settings.

Figure 3.9 IRQ5 Selected for PE5 (with the ICUA Not Set up)

When IRQ5 is set up in the [ICUA] pane, IRQ5 is actually assigned to PE5 (figure 3.10).

Pin No.	Pin name	Selected function	Direction	State
73	PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	IRQ5	Input	

Figure 3.10 IRQ5 Selected for PE5 (after the ICUA Has been Set up)



3.2.2 [Peripheral pin usage] Sheet

The [Peripheral pin usage] sheet shows which pins are used by the corresponding peripheral module. The pin functions associated with the peripheral module selected in the left section and where those functions are assigned are listed in the right section. If multiple pins are selectable for a specific function, the allocation can be changed through this sheet.

ALL Clock	*	Pin name	Pin function	Assignment	Pin No.	Direction	State	A
LIOCK CAC		XTAL	Main clock oscillator connection	XTAL/P37	11	Output		
nterrupts	=	EXTAL	Main clock oscillator connection	EXTAL/P36	13	Input		
ITU		XCIN						
1TUO 1TU1		XCOUT						
ITU2		CACREF						
ITU3		NMI						
1TU4 1TU5		IRQO						
'OE		IRQ1						
MRO		IRQ2						
MB1		IRQ3						
MR2 MR3		IRQ4						
	-	IDOE						-
function Peri	ipheral p	in usage Pin Ia	yout					4 1

Figure 3.11 Pin-Function Pane ([Peripheral pin usage] Sheet)

Table 3.3 lists the columns on the [Peripheral pin usage] sheet.

Column	Contents
Pin Name	Names of pins used by the peripheral module selected in the left section
Pin Function	Pin function
Assignment	Full name of the MCU pin, showing all of the functions assigned to that pin
Pin No.	Pin number
Direction	Input or output
State	Warning or error message, if any

Table 3.3 Columns on the [Peripheral pin usage] Sheet

(1) Default State

By default (i.e. when no pins have been set up for use with peripheral modules), the [Pin Function] and [Assignment] columns are blank (figure 3.12).

Pin name	Pin function	Assignment	Pin No.	Direction	State
IRQ0					

Figure 3.12 [Peripheral pin usage] Sheet in the Default State

(2) Assigning a Pin Function to a Port Pin

When a peripheral module associated with input to or output from pins has been set up, the pin functions to be used by that peripheral module are assigned to the corresponding port pins and the current settings are shown on the [Peripheral pin usage] sheet. If you have set up external interrupt IRQ0 in the detailed settings pane, for example, pin IRQ0 is assigned to P30 and the [Peripheral pin usage] sheet shows the setting of IRQ0 as follows.

Pin name	Pin function	Assignment	Pin No.	Direction	State
IRQ0	External interrupt	P30/MTIOC4B/TMRI3/	20	Input	

Figure 3.13 Display of a Pin Function Assigned to a Port Pin (Example)

Note:

When a peripheral module is set up in the default state (i.e. when no pin functions have been selected on the [Pin function] or [Peripheral pin usage] sheet), the pin functions for that peripheral module are assigned to the port pins listed in the "Allocation in the Default State" section of appendix 1, Pin Functions for which the Allocation Can be Changed. When the allocation of pin functions has been designated on the [Pin function] sheet before a peripheral module is set up, the pin functions are assigned to the selected port pins.

Subsequently setting up general-purpose I/O port pin P30, which uses the same pin as IRQ0, in the [I/O] pane will cause a conflict and a warning will be output as shown in figure 3.14.

Pin name	Pin function	Assignment	Pin No.	Direction	State
🔥 IRQO	External interrupt	P30/MTIOC4B/TMRI3/P0E8#/	20	Input	Conflicting with another pin function.

Figure 3.14 Warning of a Conflict between Pin Functions

Note:

Even if two or more pin functions are assigned to a single pin (as in figure 3.14), generating source files is still possible. You can switch between the functions, although more than one cannot be in use at the same time. For details, refer to section 3.2.4, Error Messages and Warnings on Pin Settings.

The allocation of IRQ0 can be changed. Other pins to which IRQ0 can be assigned are selectable from a drop-down list box. Placing the mouse pointer on the [Assignment] column brings up a drop-down button.

Pin name	Pin function	Assignment	Pin No.	Direction	State
🔥 IRQO	External interrupt	P30/MTIOC4B/TMRI3/POE8 👻	20	Input	Conflicting with another pin function.
		Figure 3.15 Dr	- on-Down	Button	

Click on the drop-down button and select one of the options displayed in the list box.

Pin name	Pin function	Assignment	Pin No.	Direction	State
🔥 IRQO	External interrupt	P30/MTIOC4B/TMRI3/POE8 💌	20	Input	Conflicting with another pin function.
		P30/MTIOC48/TMRI3/P0E8#/R PD0/IRQ0 PH1/TM00/IRQ0	XD1/SMIS	01/SSCL1/	RQ0-DS/RTCIC0

Figure 3.16 Changing the Allocation of a Pin Function

If IRQ0 is assigned to PH1 and that pin is not being used for any other peripheral module, the conflict between P30 and IRQ0 can be resolved.

Pin name	Pin function	Assignment	Pin No.	Direction	State
IRQO	External interrupt	PH1/TM00/IRQ0	37	Input	

Figure 3.17 Display after Changing the Allocation

The pin functions for which you can select the assignment are listed in appendix 1, Pin Functions for which the Allocation Can be Changed.

Note:

When the peripheral module has not been set up (as in figure 3.12), the allocation of pin functions cannot be changed through this sheet.



3.2.3 Peripheral-Module Settings Shared between the [Pin function] and [Peripheral pin usage] Sheets

A change to a setting on either the [Pin function] or [Peripheral pin usage] sheet is reflected on the other sheet. When the allocation of a pin function is changed on the [Pin function] sheet, that change also applies to the [Peripheral pin usage] sheet, and vice versa (figure 3.18).

[Pin function] sheet				[Peripheral pin usage] sheet				
Pin No.	Pin name	Selected function		Pin name	Pin function	Assignment	Pin No	
71	PE7/IRQ7/AN015	IRQ7		IRQ2				
72	PE6/IRQ6/AN014	Not assigned		IRQ3				
73	PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	Not assigned						
74	PE4/MTIOC4D/MTIOC1A/AN012/CMPA2	Not assigned		IRQ4				
75	PE3/MTIOC4B/POE8#/CTS12#/RTS12#/SS12#/A	Not assigned		IRQ5				
76	PE2/MTIOC4A/RXD12/RXDX12/SMIS012/SSCL12/	Not assigned		IRQ6				
77	PE1/MTIOC4C/TXD12/TXDX12/SIOX12/SMOSI12/	Not assigned		IRQ7	External interrupt	PE7/IRQ7/AN015	71	
78	PE0/SCK12/AN008	Not assigned					\sim	
79	PD7/MTIC5U/POE0#/IRQ7	IRQ7					1	
				IRQ7	External interrupt	PD7/MTIC5U/POE0#/IRQ7	79	
	Change	he pin function	<u> </u>			Change the	e allocat	
	Change				Reflect	-		

Figure 3.18 Linking of the [Pin function] and [Peripheral pin usage] Sheets

The current settings for each peripheral module are reflected on the [Pin function] and [Peripheral pin usage] sheets. When IRQn is set up in the [ICUA] pane, for example, the [Peripheral pin usage] sheet shows that IRQn is in use and the allocation of IRQn is displayed on the [Pin function] and [Peripheral pin usage] sheets. When the setting for IRQn in the [ICUA] pane is canceled, the allocation of IRQn is canceled on the [Pin function] and [Peripheral pin usage] sheets.



Figure 3.19 Setting up a Peripheral Module and Allocating Pin Functions



On the other hand, a change made on the [Pin function] or [Peripheral pin usage] sheet is not reflected on the detailed-settings pane for the peripheral module. Even if [Selected function] for IRQn is changed to "Not assigned" on the [Pin function] sheet after IRQn has been set up in the [ICUA] pane, for example, the setting of IRQn in the [ICUA] pane is not canceled. Since no pin is assigned to IRQn in this case, an error message appears. For details on the error messages, refer to section 3.2.4, Error Messages and Warnings on Pin Settings.





3.2.4 Error Messages and Warnings on Pin Settings

When an incorrect setting is made, an error message or warning is displayed on the [Pin function] or [Peripheral pin usage] sheet. The errors and warnings are listed in table 3.4.

Cause	Туре	Message
A single pin function has been	Error	"The same function is assigned to <i><pin numbers=""></pin></i> ."
selected for multiple pins.		([Pin function] sheet)
		"Do not assign a single function to multiple pins."
		([Peripheral pin usage] sheet)
The pin function has not been	Error	"Not assigned" ([Peripheral pin usage] sheet)
allocated.		
Multiple pin functions have been	Warning	"Conflicting between different functions." ([Pin function] sheet)
selected for a single pin.		"Conflicting with another pin function." ([Peripheral pin usage] sheet)
Conflict with use of a pin by a	Warning	"Conflicting with an on-chip emulator pin."
debugger		([Pin function] sheet)
		"Conflicting between a peripheral module pin and an on-chip emulator
		pin."
		([Peripheral pin usage] sheet)
The peripheral module has not	Warning	" <pin function=""> has not been configured in the peripheral settings."</pin>
been set up.		([Pin function] sheet)

Table 3.4 Errors and Warnings



Details of the errors and warnings are given below.

(1) A single pin function has been selected for multiple pins.

Selecting a single pin function for multiple pins leads to an error that prevents the generation of source files.

In this case, allocate another pin function to either of the pins, change the entry on the [Pin function] sheet to "Not assigned", or re-select the allocation of the pin function on the [Peripheral pin usage] sheet.

Pin No.	Pin name	Selected function	Direction	State			
20	P30/MTIOC4B/TMRI3/P0E8#/RXD1/SMIS01/SSC	IRQO	Input	The same function is assigned to 20/86.			
86 🔇	PD0/IRQ0	IRQO	Input	The same function is assigned to 20/86.			
(a) [Pin function] Sheet							

Pin name	Pin function	Assignment	Pin No.	Direction	State			
😣 IRQO	External interrupt	Conflicted	20/86	Input	Do not assign a single function to multiple pins.			
(b) [Dorinhard nin years] Sheat								

(b) [Peripheral pin usage] Sheet

Figure 3.21 Example of an Error (Selection of a Single Function for Multiple Pins)

(2) The pin function has not been allocated.

Failure to allocate a pin function required by a peripheral module leads to an error and prevents the generation of source files.

Select the pin function for a corresponding pin on the [Pin function] sheet or designate the allocation of the pin function on the [Peripheral pin usage] sheet.

Pin name	Pin function	Assignment	Pin No.	Direction	State
😣 IRQO 👘	External interrupt	Not assigned	Not assigned	Input	Not assigned.

[Peripheral pin usage] Sheet

Figure 3.22 Example of an Error (Pin Function not Allocated)

(3) Multiple pin functions have been selected for a single pin.

A warning appears when two or more pin functions have been assigned to a single pin (as in figure 3.23), but generating source files is still possible. You can switch between the functions, although they cannot be used at the same time.

To switch between pin functions, make the initial setting for the peripheral module using that pin function, since the individual pin functions are set by the initial-setting function for the given peripheral module. However, RTCOUT and RTCIC2 cannot be assigned to the same pin.



Pin No.	Pin name		Selected function	n Direction	State				
10 🔨	P30/MTIOC4B/TMRI3/PO	E8#/RXD1/SMISO1/SSC	P30/IRQ0		Conflicting between different functions.				
(a) [Pin function] Sheet									
Pin name	Pin function	Assignment	Pin No.	Direction	State				
🔥 IRQO	External interrupt	P30/MTIOC4B/TMRI3	37 20	Input	Conflicting with another pin function.				
Pin name	Pin function	Assignment	Pin No.	Direction	State				
🔥 P30	General input port	P30/MTIOC4B/TMRI3	37 20	Input	Conflicting with another pin function.				
	(b) [Peripheral pin usage] Sheet								

Figure 3.23 Example of a Warning (Multiple Pin Functions Selected for a Single Pin)

(4) Conflict with use of a pin by a debugger

A warning appears when a pin function for a peripheral module has been allocated to a pin for use by an on-chip debugger. Generating source files is still possible. Note, however, that the other pin function allocated to the pin may not be usable while the on-chip debugger is in use.

Pin No.	Pin name	Selected function	Direction	State
1 21	P27/MTIOC2B/TMCI3/SCK1	ТМСІЗ		Conflicting with an on-chip emulator pin.

(a) [Pin function] Sheet

Pin name	Pin function	Assignment	Pin No.	Direction	State
🔥 тмсіз	Counter clock input	P27/MTIOC	21	Input	Conflicting between a peripheral module pin and an on-chip emulator pin.
				• 1 1	

(b) [Peripheral pin usage] Sheet

Figure 3.24 Example of a Warning (Conflict with Use of a Pin by a Debugger)

(5) The peripheral module has not been set up.

A warning appears when a pin function is selected on the [Pin function] sheet but the corresponding peripheral module has not been set up. Although generating source files is still possible, the selected pin function will not be usable. To enable the selected pin function, set up the peripheral module that is to use the function and call the initial-setting function, which sets the registers to change the pin function.

Pin No.	Pin name	Selected function	Direction	State
1 20	P30/MTIOC4B/TMRI3/POE8	IRQO		IRQ0 has not been configured in the peripheral settings.

[Pin function] Sheet

Figure 3.25 Example of a Warning (Peripheral Module Not Set up)



3.3 Endian

Select the [SYSTEM] tab from the peripheral-module selection tabs and click on [Option setting] in the resource pane to open the endian setting pane.

	Clock Clock Pin (M Option Regist	ultifur 1 setti	nction ng	pin co	ntroller			n settin an sele	- -	_ittle endia	in	•]			
SYSTEM	LVDAa	CAC	LPC	ICUb	Buses	DMACA	DTCa	ELC	I/O	MTU2a	POE2a	TMR	CMT	RTCc	IWDTa	SI A D

Figure 3.26 The setting method of endian

Select endian to be used here. This setting is only used for selecting Renesas Peripheral Driver Library files (xxx_little.lib or xxx_big.lib) to be linked and thus does not affect the output source code.



4. Tutorial

This section introduces the usage of the Peripheral Driver Generator by giving instructions on how to use the Peripheral Driver Generator and High-performance Embedded Workshop to create a tutorial program that implements the following operations on the Renesas Starter Kit board for the RX220.

- An LED blinking on a 8-bit timer (TMR) interrupt
- An LED blinking on the PWM output of the multi-function timer pulse unit 2 (MTU2a)
- Continuously scanning on 12-Bit A/D converter (S12ADb)
- Triggering DTCa by ICUb
- Data transfer between SCIe channels 0 and 5

The labels given below respectively indicate operations to take place in the Peripheral Driver Generator and in the High-performance Embedded Workshop.





4.1 An LED blinking on a 8-bit timer (TMR) interrupt

The LED2 on RSK board is connected to P16. In this tutorial, 8-bit Timer and I/O port will be set up to blink this LED as follows.

Note : If there is a switch that enables/disables P16 on the RSK board, enable it.



(1) Making the Peripheral Driver Generator project

_	_	
	_	

- 1. Start the Peripheral Driver Generator.
- 2. Select [File]->[New Project] menu.

PDG2				
File View	Tool Help			
New Pr	oject	Ctrl+N(N)		
Open		Ctrl+O(O)		
1 c:\ren	esas\\default.pd2			
Exit				
			·	
×				
Gener				
Create a new	project			

3. Specify "rx220_demo1" as the project name.

Set the CPU type as follows.

Series : RX200 Group : RX220 Part No. : R5F52206BxFP

Note: If another type of chip is mounted on your RSK board, select corresponding CPU type.

Pro	ject new	—
Pr	roject name:	
D	(220_demo1	
_	irectory:	
c	:\renesas\PDG2_pro	Browse
Г	Device selection-	
	Series:	RX200 💌
	Group:	RX220 💌
	Part No.:	R5F522068xFP
	Package:	PLQP0100KB-A
	ROM capacity:	256K byte(s)
	RAM capacity:	16K byte(s)
		OK Cancel



PDG



-The clock setting window opens and the error icons are displayed in the initial state.

PDG2 - [rx220_demo1.pd2]		
Ele View Iool Help		- 8 ×
System Cock Generation Circuit Pn (Muffrancis pin controller) Orgital cetting Tréglater Write Protection Function	Internal clock source: Main clock oscillator Main clock oscillator settings Use the main clock oscillator Main clock oscillator source: Main clock oscillator source: Main clock oscillator drive capability: 15-MHz to 20-MHz (Lead type ceramic resonator) Main clock oscillator settings Use RTC-dedicated clock (supplied by the sub-clock oscillator) Sub-clock oscillator drive ability: Drive ability for low CL Resonator frequency:	A III
SYSTEM LVDAa CAC LPC ICUb B	High-speed on-chip oscillator (HOCD) settings Use the high-speed on-chip oscillator (HOCO) uses DMACA_DTCa_ELC_VO_MTU2a_POE2a_TMR_CMT_RTCC_IWDTa_SCI_RIIC_REPI_CRC_S12ADb_CMPA_DOC	-
Ready	Clock setting window	

Place the mouse pointer on the error icon, then the contents of error is displayed.



There are 3 types of icons in Peripheral Driver Generator	
😣 Error	
The setting is not allowed.	
The source filese cannot be generated if there is an error setting.	
🕂 Warning	
The setting is possible but may be wrong.	
Source files can be generated.	
Information	
Additional information for the complex setting.	
Only icons on the setting window can display the tooltip.	



(3) Clock setting

PDG

It is necessary to set the main (EXTAL) clock frequency first.
 External clock frequency of the RSK board is 20 MHz. Set 20 to the edit box.

- 2. ICLK, PCLKB, PCLKD and FCLK are used in 20 MHz.
 - Set 20 to the edit box.

Main clock oscillator settings	
Use the main clock oscillator	
Main clock oscillation source:	Resonator 1
Main clock (EXTAL input) frequency:	20 MHz
Oscillator drive capability:	16MHz to 20MHz
🔲 16-MHz to 20-MHz (Lead type ce	ramic resonator)
Main clock oscillation stop detection:	Disable

Frequency settings					
Internal clock source frequency:	20.	000000	MHz		
	Frequency	2	Actual value		Internal clock source frequency division ratio
System clock (ICLK):	20	MHz	20.000000	MHz	1
Peripheral module clock B (PCLKB): 🤫	20	MHz	20.000000	MHz	1
Peripheral module clock D (PCLKD):	20	MHz	20.000000	MHz	1
FlashIF clock (FCLK):	20	MHz	20.000000	MHz	1
RTC-dedicated clock (RTCSCLK):	0.032768	MHz			
IWDT-dedicated clock (IWDTCLK):	0.125000	MHz			

(4) Endian setting

PDG

For the endian setting, refer to section 3.3, Endian.



(5) I/O Port setting

PDG

The LED2 on RSK is connected to P16 so set P16 to output port.

- 1. Select "I/O" tab
- 2. Select "Port 1"
- 3. Check "Pn6"
- 4. Select "Output"

PDG2 - [rx220_demo1.pd2 *] S File View								
	2							
VO Ports Port 0 Port 1 Port 2	Port	1						
🖬 Port 3		Used as an I/O port	Direction	Enable input pull-up resistor	Output type		High-drive output	
Port 4	Pn0		Ţ	Γ		Ŧ		
Port A	Pn1	Г	~			Ŧ		
Port C	Pn2		Input 💌	Γ	CMOS output	Ŧ	Γ	
Port D	Pn3		Input 💌		CMOS output	Ŧ		
Port H	Pn4	3	Input 💌	Δ	CMOS output	Ŧ		
	Pn5	- ا	Input 👻		CMOS output	~	Γ	
	Pn6		Output 💌		CMOS output	•		
	Pn7		Input 💌	Γ	CMOS output	Ŧ		
				1				
							C RSPI CRC S12ADb CMPA DOC	
×				/				
Ready								



(6) TMR setting-1

PDG

In this tutorial, TMR (8-bit timer) Unit0 is used in 16 bit mode (two 8-bit timers cascade connection)

- 1. Select "TMR" tab
- 2. Select "Unit0"
- 3. Select "16 bit timer mode"
- 4. Check "Use this channel"

S - Bt Timer (TIR) - Timer TIR) - Timer TI	E
Count source: Internal clock (PCLK)	
Specify the external clock frequency	
Count source frequency: 20.000000 MHz	
Counter clearing source: Disable counter clear	
Specify the timer operating period and duty cycle Timer operating period: 3.276800 msec Actual value:	
Duty cycle: 2 Actual of Error:	
SYSTEM LVDAa CAC LPC ICUb Buses DMACA DTCa ELC 10 MTU2a OE2a STMR CMT RTCC IWDT	a SCI RIIC RSPI CRC S12ADb CMPA DOC
Ready	



(7)	TMR	setting	r-2
· · /			, —

PDG

Set the other items as follows.

Count settings	
Count source:	Internal clock (PCLK/8192)
	Specify the external clock frequency
Count source frequency:	0.002441 MHz
Counter clearing source:	Compare match B
Specify the timer operat	ting period and duty cycle -Count source :
Timer operating period:	1000 Internal clock(PCLK/8192)
Duty cycle:	-Counter clearing source : Compare match B -Interval : 1000 ms -Duty cycle : 50%
Compare match A value (T Compare match B value (T	automatically calculated

(8) TMR setting-3

PDG

Set the interrupt notification functions.

These functions are called when the interrupt occurs.

	Interrupt settings Use overflow interrupt (OVIr Interrupt request destinatio	-Check compare match A interrupt Notification function name is "Tmr0CmAIntFunc" -Check compare match B interrupt Notification function name is "Tmr0CmBIntFunc"	
(🛛 🔽 Use compare match A interr	арссилиял	
	nterrupt request destinatio	n: CPU 🔽	
	Interrupt notification function	n name: Tmr0CmAIntFunc	
(🔽 Use compare match B interr	upt (CMIBn)	
	Interrupt request destinatio	n: CPU 🔽	
	Interrupt notification function	n name: Tmr0CmBIntFunc	
	CPU interrupt priority level (Shar	ed with OVIn, CMIAn and CMIBn):	



- (9) Generating source files PDG
- 1. To generate source files, click **P** on the tool bar.
- 2. Save confirmation dialog box is displayed. Click [Yes].



3. Click [OK] on the message box.



4. Generated functions are listed in lower pane.

By double clicking the line of function, source file can be opened.

PDG2 - [rx220_demo1.pd2] S File View Tool			- 8 ×
D 🛩 🖬 🔯 🗞 🖄 🦿			
	e 🗸 🚱		E
Vuse this channel Count settings Count source: Count source frequen SYSTEM LVDAa CAC LPC ICUb Buses DMAG		SCI RIIC RSPI CRC S12ADD DA CMPA C	*
× Source file name	Generated function name	Functional explanation of functions	
C(vrenesas\PDG2_proj\n220_demo1\TMR\R_ c\vrenesas\PDG2_proj\n220_demo1\TMR\R_ c\vrenesas\PDG2_proj\n220_demo1\TMR\R_ c\vrenesas\PDG2_proj\n220_demo1\TMR\R_ c\vrenesas\PDG2_proj\n220_demo1\TMR\R_ c\vrenesas\PDG2_proj\n220_demo1\TMR\R_ c\vrenesas\PDG2_proj\n220_demo1\TMR\R_	, bool R_PG_Timer_Start_TMR_U0(void) . bool R_PG_Timer_HatkCount_TMR_U0(void) . bool R_PG_Timer_ResumeCountr/MR_U0(void) . bool R_PG_Timer_SetCounter/value_TMR_U0(uint16_t * counter_val) . bool R_PG_Timer_SetCounter/value_TMR_U0(uint16_t counter_val) . bool R_PG_Timer_SetRequestFlag_TMR_U0(void) * cma, bool * cmb, bool * ov) . bool R_PG_Timer_SetSpModule_TMR_U0(void)	Set up the TMR and start the count Halt the TMR count Resume the TMR count Acquire the TMR counter value Set the TMR counter value Acquire and clear the TMR interrupt flags Shut down the TMR unit	Others
Ready			1.



(10) Preparing the High-performance Embedded Workshop project

HEW

Start the High-performance Embedded Workshop and make RX220 workspace.





е

New Project-8/10-Se	□RX200	/E20 SYSTEM Simulator	cify the targ	et emulato	r.
	<pre> Target type Target CPU < Back Next > </pre>	8	Cancel	ect is com	
rx220_demo1 - High-performance ile Edit View Project Build		/indow Help	FIU		JIELE
Image: Second secon	1. 1.1			Lebug	▼ Defaul
🚭 P. 💂 T. 🔍 N. [🖸					
9 01 01 AL AT 21 21 21 0	₽ ₽ ?				

Ready



Default1 desktop

Build \bigwedge Debug \bigwedge Find in Files 1 \bigwedge Find in Files 2 \bigwedge Macro \bigwedge Test \bigwedge Version Control \bigwedge

INS

(11) Adding the generated source files to the High-performance Embedded Workshop project



2. Click [OK] on the confirmation dialog box.

PDG2		×
<u>^</u>	The generated source files will be registered in HEW project. Make sure the destination project of source registration has been opened as an active project. Make sure that the Hew TargetServer has be set up. If two or more workspaces are opened, close the workspaces which does not include target project.	
	Click OK if registration is ready.	
	OK Cance	1

- 3. This is a linkage setting of Renesas Peripheral Driver Library.
 - When using multiple lib files, linkage order can be set in this dialog box.

riority high	y in which order libraries are linked. [C:\renesas\PDG2\lib\RX220\RX220_library_100_little.lib	_
Priority low		Up
	ОК	Cancel

put in "AddFromPDG" folder.	Image: Second secon	
	Pr ■ T ■ N □ ■	

Make sure that the HEW Target Server has been set up before executing registration.

For details, refer Peripheral Driver Generator user's manual.
(12) Making the program on High-performance Embedded Workshop

HEW

By changing the part of "main" function, make the following program on High-performance Embedded Workshop.

```
//Include "R_PG_<project name>.h"
#include "R_PG_rx220_demo1.h"
void main(void)
{
    //Configure I/O port pins that are not available
//
    R PG IO PORT SetPortNotAvailable();
    //Set up the clock
    R_PG_Clock_Set();
    //Set up port P16
    R_PG_IO_PORT_Write_P16(1);
    R_PG_IO_PORT_Set_P1();
    //Set up TMR Unit0 and start count
    R_PG_Timer_Start_TMR_U0();
    while(1);
}
// Compare match A interrupt notification function
void Tmr0CmAIntFunc(void)
{
    // Turn on the LED
    R_PG_IO_PORT_Write_P16(0);
}
// Compare match B interrupt notification function
void Tmr0CmBIntFunc(void)
{
    // Turn off the LED
    R_PG_IO_PORT_Write_P16(1);
}
```



(13) Connecting to the emulator, building the program and executing

HEW

1. Connect to the emulator



2. Just by clicking [Build] button, program can be built because Renesas Peripheral Driver Library and include directory are automatically registered in build setting.



- 3. Download the program
- 4. Execute the program and see the LED on RSK board.



Reset go button





4.2 An LED blinking on the PWM output of the multi-function timer pulse unit 2 (MTU2a)

The LED0 on RSK board is connected to P14. This port can also be used as PWM output pin (MTIOC3A) of the multi-function timer pulse unit 2. In this tutorial, the multi-function timer pulse unit 2 will be set up to operate in PWM mode 1 and the PWM output will blink the LED0 as follows.

Note : If there is a switch that enables/disables P14(MTIOC3A) on the RSK board, enable it.



ΟK

Cancel

PDG (1)Making the Peripheral Driver Generator project Project new x Make the new Peripheral Driver Generator project Project name: "rx220_demo2". For details on how to make the new rx220_demo2 Peripheral Driver Generator project, refer to section 4.1 Directory: (1), Making the Peripheral Driver Generator project. c:\renesas\PDG2_proj Browse ... Set the CPU type as follows. Device selection Series: RX200 Ŧ Series : RX200 RX220 Group: RX220 Group: • Part No. : R5F52206BxFP R5F52206BxFP Part No.: Ŧ PLQP0100KB-A Package: Note: If another type of chip is mounted on your 256K RSK board, select corresponding CPU type. ROM capacity: byte(s) 16K RAM capacity: byte(s)

(2) Clock setting

PDG

- 1. The clock setting window opens and the error icons are displayed in the initial state. For icons such as and displayed on window, refer to section 4.1 (2), Initial state.
- 2. For the clock setting, refer to section 4.1 (3), Clock setting.
- (3) Endian setting

PDG

For the endian setting, refer to section 3.3, Endian.



(4) MTU2a setting-1

PDG

Opening MTU2a channel 3(MTU3) setting window

- 1. Select "MTU2a" tab.
- 2. Select "MTU3" on tree view.
- 3. Check "Use this channel".

File View Tool Help Selectable modes and the corresponding channels are given below. NullH-function T With Hundle Time and the corresponding channels are given below. Null Hundle Time and the corresponding channels are given below. Null Hundle Time and the corresponding channels are given below. Selectable modes and the corresponding channels are given below. Null Hundle Time and the corresponding channels are given below. Selectable modes and the corresponding channels are given below. Null Hundle Time and the corresponding channels are given below. Selectable modes and the corresponding channels are given below. Null Hundle Time and the corresponding channels are given below. Null A mitual and thus a selectable modes and the corresponding channels are given below. Null Hundle Hundle Time and the corresponding channels are given below. Null A mitual A mi		- • •
Multi-Function Ti 2 (MTU2a) Selectable modes and the corresponding channels are given below. Normal mode (free-running, periodic-counter, and external-event count operations): MTU0 to MTU4 NVM mode 1: MTU0 to MTU4 NVM mode 1: MTU0 to MTU4 NVM mode (T to 4) MTU and MTU2 NVM mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 mode (T to 4) MTU3 mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 mode (T to 4) MTU3 mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 mode (T to 4) MTU3 mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 mode (T to 4) MTU3 NVUS NVM mode (T to 4) MTU3 NVM mode (T to 4) MTU3 NVM mode (T to 4) MTU3 NVM NVM NVM mode (T to 4) MTU3 NVM NVM NVM NVM NVM mode (T to 4) MTU3 NVM		- 8 ×
Selectable mode: and the corresponding channels are given below. MTU1 MTU2 MTU2 MTU2 MTU2 MTU2 MTU3		
	Multi-function Ti Multi-function Ti Multi-function Ti MUD (MTU2a) Selectable modes and the corresponding channels are given below. Normal mode [free-running, periodic-counter, and extendevent count operations]: MTU0 to MTU4 PVM mode 11 to 10 MTU4 PVM mode 11 to 10 MTU2 Phase counting mode [1 to 4) MTU and MTU2 Phase counting mode [1 to 4) MTU3 MTU4 MTU5 MTU5 MTU5 MTU5 MTU2 Phase counting mode [1 to 4) MTU3 THU1 and MTU2 Phase counting mode [1 to 4) MTU3 THU1 and MTU2 Phase counting mode [1 to 4) MTU3 MTU4 MTU5 MTU5 MTU2 Phase counting mode [1 to 4) MTU3 Thu4 MTU5 Phase counting mode [1 to 4) MTU5 Phase counting mode [1 to 4) MTU3 Phase	
Ready	Ready	

(5) MTU2a setting-2

Select "PWM mode 1" for the operation mode.

Operation mode Mode selection: PW/M mode 1	Explanation of selected operation mode
	ut from pins MTIOCnA and MTIOCnC when the corresponding RC and TGRD), respectively, are in use. When the values of the the output from the pins is controlled to generate PWM waveforms.

(6) MTU2a setting-3

PDG

PDG

The counter setting is as follows.

- 1. Select "TGRA register compare match" for a counter clearing source.
- 2. Select "Internal clock (PCLK/256)" for a count source.
- 3. Set "500msec" to timer operation period.

Count settings		_
Counter clearing source:	TGRA compare match (use TGRA as a cycle register)	
Count source:	Internal clock (PCLK/256)	
	Specify the external clock frequency	
Count source frequency:	0.078125 MHz	
Timer operating period:	500 msec 🖌 Actual value: 500.006400msec	
Cycle register value:	3 0.001280%	



(7) MTU2a setting-4

PDG

General register setting is as follows.

- 1. The TGRA is selected as a counter clearing source in the counter setting. Then the TGRA value is calculated from the count source frequency and the timer operating period.
- 2. Select "Initial output of MTIOCnA pin is high: High output at compare match" for TGRA output compare operation.
- 3. Set "33000" to TGRB initial value.
- 4. Select "Low output from MTIOCnA pin at compare match" for TGRB output compare operation.
- 5. The MTIOCnC output is not used in this tutorial. Select "MTIOCnC pin output is disabled" for TGRD output compare operation.

-General register and input/ou	tput settings
TGRA	
Function:	Output compare register
	A compare match with the counter value caus frupt request to be issued and the signal output from the pin to be controlled.
Initial value of the register:	39062
-	2
Input capture/output compa	are operation:
Use the noise filter for th	
TGBB	
Function:	Output compare register
	A compare match with the counter value caus from the pin to be controlled. 3
Initial value of the register:	33000
Input capture/output comp	are operation: 4
Low output from MTIOCnA	pin at compare match
Use the noise filter for th	ne MTIOCnB pin
TGRC	
Function:	Output compare register A compare match with the counter value causes an interrupt request to be issued and the signal output
	from the pin to be controlled.
Initial value of the register:	0
Input capture/output comp	are operation:
MTIOCnC pin output is disa	abled
Buffer transfer timing:	When compare match A occurs
🔲 Use the noise filter for th	ne MTIOCnC pin
TGRD	
Function:	Output compare register
	A compare match with the counter value causes an interrupt request to be issued and the signal output
	from the pin to be controlled.
Initial value of the register:	
Input capture/output comp	are operation: 5
MTIOCnC pin output is disa	abled
Buffer transfer timing:	When compare match B occurs
🔲 Use the noise filter for th	ne MTIOCnD pin



(8) MTU2a setting-5

The compare match timing and the output waveform are displayed in a diagram.

PDG



(9) Generating source files **PDG**

To generate source files, click in on the tool bar. For details on generating source files, refer to section 4.1 (9), Generating source files.

(10) Preparing the High-performance Embedded Workshop project

Start the High-performance Embedded Workshop and make RX220 workspace. For details on making High-performance Embedded Workshop project, refer to section 4.1 (10), Preparing the High-performance Embedded Workshop project.

(11) Adding the generated source files to the High-performance Embedded Workshop project

To add the generated source files to High-performance Embedded Workshop, click in on the tool bar. For details on adding the source files to High-performance Embedded Workshop project, refer to section 4.1 (11), Adding the generated source files to the High-performance Embedded Workshop project.



PDG

HEW

(12) Making the program on High-performance Embedded Workshop

HEW

By changing the part of "main" function, make the following program on High-performance Embedded Workshop.



(13) Connecting to the emulator, building the program and executing

HEW

Execute the program and see the LED blinking on RSK board. For details on connecting to the emulator, building the program, and executing the program, refer to section 4.1 (13), connecting to the emulator, building the program and executing.





4.3 Continuously scanning on 12-Bit A/D converter (S12ADb)

In RX220 RSK board, the potentiometer is connected to AN000 analog input. In this tutorial, the 12-Bit A/D converter (S12ADb) will be set up to execute A/D conversion continuously. And the result of A/D conversion will be monitored on High-performance Embedded Workshop.



Potentiometer

Note : If there is a switch that enables/disables AN000 on the RSK board, enable it.

Making the Peripheral Driver Generator project
 Make the new Peripheral Driver Generator project
 "rx220_demo3". For details on how to make the new
 Peripheral Driver Generator project, refer to section 4.1
 (1), Making the Peripheral Driver Generator project.
 Set the CPU type as follows.

Series : RX200 Group : RX220 Part No. : R5F52206BxFP

Note: If another type of chip is mounted on your RSK board, select corresponding CPU type.

PDG

P	roject new	— ו
ſ	Project name:	
l	rx220_demo3	
	Directory:	
	c:\renesas\PDG2_pr	oj Browse
	Device selection	
	Series:	RX200 -
	Group:	R×220 -
	Part No.:	R5F52206BxFP
	Package:	PLQP0100KB-A
	ROM capacity:	256K byte(s)
	RAM capacity:	16K byte(s)
		OK Cancel



PDG

- 1. The clock setting window opens and the error icons are displayed in the initial state. For icons such as and displayed on window, refer to section 4.1 (2), Initial state.
- 2. For the clock setting, refer to section 4.1 (3), Clock setting.
- (3) Endian setting

PDG

For the endian setting, refer to section 3.3, Endian.

(4) A/D converter setting-1

PDG

Select "S12ADb" tab and click S12AD0 on tree view.

	- • •
S File View Iool Help	- 8 ×
Image: Standown of the	Â
Use this unit	-
Operation settings	
Conversion target: Analog input channel	
Mode: Single scan mode	
Internal reference voltage	
Add A/D-converted value	
Analog input channel	
Double trigger mode	
Convert Convert Add Use dedicated (Group A) (Group B) A/D-converted value sample-and-hold circuit	
ANOOI 🔽 🗖 🗖	
	Ψ.
SYSTEM LVDAa CAC LPC ICUb Buses DMACA DTCA ELC 1/0 MTUZA POEZA TMR CMT RTCC INDTA SCI RIC RSPI CRC S12AD6 CAPA DOC	
×J	
kedy Carter Cart	



(5) A/D converter setting-2

PDG

Make the following setting for S12AD0.

- 1. Check "Use this unit".
- 2. Select "Analog input channel" for the conversion target.
- 3. Select "Continuous scan mode" for the operation mode.
- 4. Check "AN000" for the analog input channel.
- 5. Select "Software trigger only" for the conversion start trigger (Group A).
- 6. Select "Right-alignment" for the data placement.
- 7. Select "Disables automatic clearing" for the automatic clearing of A/D data register.

Unit: S	1						
🗸 Use this un	nit	_					
- Operation set	ttings —			2			
Conversion	target:	Analog input chann	el 🗾				
Mode:		Continuous scan m	ode 💽				
Internal refe	erence v	oltage		3			
🗖 Add	A/D-cor	nverted value					
Analog inpu	it chann	el					
🗖 Dout	ble trigge	er mode					
	Conve (Group		Add A/D-converted value	Use dedicated sample-and-hold circuit	Q		
AN000	v	Г	Γ				
AN001		Г	Γ	□ 4			
AN002	\checkmark	Г					
AN003	~						
AN004	\checkmark	Γ					
AN005	~	Г					
AN006	◄						
AN007	✓						
AN008	~	Г					
AN009	◄						
AN010							
AN011	V						
AN012	•						
AN013	•						
AN014	হ						
AN015	N.	I	I				
Conversion	start trig	ger (Group A):					5
Software tri	igger on	ly .				•	Т
Conversion	start trig	ger (Group B):					
Compare-m	hatch/Inj	put-capture A signal f	rom MTUO (TRGQAN)			v	
A/D-conver	rted valu	e addition count:	2-time conversion (addition)	once)		Ŧ	6
Data placer	ment:		Right-alignment			•	
Automatic c	learing	of A/D data register:	Disables automatic clearing			-	
							7



(6) A/D converter setting-3

PDG

Make the following setting for S12AD0.

8. Check "Use A/D conversion end interrupt (S12ADI0)".

nterrupt settings	rupt (S12ADIO) 🧳			
Interrupt request destination:	CPU	8	•	
CPU interrupt priority level:	15 -	Interrupt notification function name:	S12ad0AIntFunc	
Use A/D conversion end inter	rupt for group B (GBA	.DI)		
Interrupt request destination:	CPU		v	
CPU interrupt priority level:	15	Interrupt notification function name:	S12ad0BIntFunc	

(7) Checking the pin usage

PDG

- It is possible to check the usage of pins on the pin function windows

1. After setting up the S12ADb, select "SYSTEM" tab and click "Pin (Multifunction pin controller)" on the tree view.

2. On the Pin function window, you can see that No.95 pin is used as AN000.

System	Pin No.	Pin name	Selected function	Direction	State		
Pin (Multifunction pin controller)	82	PD4/POE3#/IRQ4	Not assigned				
Option setting	83	PD3/POE8#/IRQ3	Not assigned				
Register Write Protection Function	84	PD2/MTIOC4D/IRQ2	Not assigned				
	85	PD1/MTIOC4B/IRQ1	Not assigned				
	86	PD0/IRQ0	Not assigned				
	87	P47/AN007	AN007	Input			
	88	P46/AN006	AN006	Input			
	89	P45/AN005	AN005	Input			
	90	P44/AN004	AN004	Input			
	91	P43/AN003	AN003	Input			
	92	P42/AN002	AN002	Input			
	93	P41/AN001	AN001	Input			
		VREFLO	VREFLO	Input		<u> </u>	
	95	P40/AN000	AN000	Input			
	96	VREFH0	VREFH0	Input			
	97	AVCC0	AVCC0	Input			
	98	P07/ADTRG0#	Not assigned				
	99	AVSS0	AVSS0	Input			
	100	P05	Not assigned				
	<u> </u>						
	Pin func	tion Peripheral pin usage Pin layo	ut				



- State of pin usage for each peripheral module is displayed in the peripheral pin usage window.

Select peripheral pin usage sheet and click S12AD0 to check the usage of AN000 pin.

System	POE	 Pin name 	Pin function	Assignment	Pin No.	Direction State	T
Clock Generation Circuit Pin (Multifunction pin controller)	TMR0	AN000	Analog input	P40/AN000	95	Input	
Option setting	TMR1 TMR2	AN001	Analog input	P41/AN001	93	Input	7
Register Write Protection Function	TMB3	AN002	Analog input	P42/AN002	92	Input	
	RTCc	AN003	Analog input	P43/AN003	91	Input	
	SCI1	AN004	Analog input	P44/AN004	90	Input	
	SCI5	AN005	Analog input	P45/AN005	89	Input	
	SCI6 SCI9	AN006	Analog input	P46/AN006	88	Input	
	SCI12	AN007	Analog input	P47/AN007	87	Input	
	BILCO	AN008	Analog input	PE0/D8[A8/D8]/SCK12/AN008	78	Input	
	RSPI0	E AN009/CMPB0	Analog input(AN009)	PE1/D9[A9/D9]/MTI0C4C/TXD12/TXDX12/SI0X12/S	77	Input	
	S12AD0	AN010/CVREFB0	Analog input(AN010)	PE2/D10[A10/D10]/MTIOC4A/RXD12/RXDX12/SMIS	76	Input	
	CMPA/U/D General port P0	AN011/CMPA1	Analog input(AN011)	PE3/D11[A11/D11]/MTIOC4B/POE8#/CTS12#/RTS12.	75	Input	
	General port P1	AN012/CMPA2	Analog input(AN012)	PE4/D12[A12/D12]/MTIOC4D/MTIOC1A/AN012/CMP	74	Input	
	General port P2	AN013	Analog input	PE5/D13[A13/D13]/MTIOC4C/MTIOC2B/IRQ5/AN013	73	Input	
	General port P3	AN014	Analog input	PE6/D14[A14/D14]/IRQ6/AN014	72	Input	
	General port P4	AN015	Analog input	PE7/D15[A15/D15]/IRQ7/AN015	71	Input	
	General port P5	ADTRG0#					
	General port PA General port PB						
	General port PC						
	General ner PD	*					
	Pin function Periphe	eral pin usage Pin lavo	ut				

(8) Generating source files

To generate source files, click in the tool bar. For details on generating source files, refer to section 4.1 (9), Generating source files.

(9) Preparing the High-performance Embedded Workshop project

PDG

Start the High-performance Embedded Workshop and make RX220 workspace. For details on making High-performance Embedded Workshop project, refer to section 4.1 (10), Preparing the High-performance Embedded Workshop project.

HEW

(10) Adding the generated source files to the High-performance Embedded Workshop project

To add the generated source files to High-performance Embedded Workshop, click on the tool bar. For details on adding the source files to High-performance Embedded Workshop project, refer to section 4.1 (11), Adding the generated source files to the High-performance Embedded Workshop project.



(11) Making the program on High-performance Embedded Workshop

HEW

By changing the part of "main" function, make the following program on High-performance Embedded Workshop.

```
//Include "R PG <project name>.h"
#include "R_PG_rx220_demo3.h"
void main(void)
{
    //Configure I/O port pins that are not available
    R_PG_IO_PORT_SetPortNotAvailable();
//
    //Set up the clock
    R_PG_Clock_Set();
    //Set up A/D converter
    R_PG_ADC_12_Set_S12AD0();
    //Start A/D conversion
    R_PG_ADC_12_StartConversionSW_S12AD0();
    while(1);
}
//Variable to store the result
uint16_t result;
//A/D conversion end interrupt notification function
void S12ad0AIntFunc(void)
{
    //Get the result of conversion
    R_PG_ADC_12_GetResult_S12AD0(&result);
}
```



🗗 🛃 🖊 🏫 🗙 📑 📌 🖓

Value

✓ Watch1 (Watch2) Watch3) Watch4 /

Address

Watch

R

(12) Connecting to the emulator, building the program and downloading

Build the program and download it. For details on connecting to the emulator, building the program, and downloading refer to section 4.1 (13), connecting to the emulator, building the program and executing.

(13) Adding the variable of A/D conversion result to the watch window

Open the Watch window and add the variable "result". Set "result" to the real time update to monitor the variable change during execution.

Scope

(14)	Executing the program and monitoring the A/D conversion result

Туре

Start the execution and screw the potentiometer to change the analog input voltage. The value of "result" on the watch window will change.

Name Value Walue H'Offf

HEW

HEW

HEW

Tutorial

4.4 Triggering DTCa by ICUb

In RX220 RSK board, switch 1 (SW1) is connected to IRQ1. In this tutorial, the data transfer controller (DTCa) and ICUb will be set up and DTC transfer triggered by IRQ1 will be performed.



Note : If there is a switch that enables/disables IRQ1 on the RSK board, enable it.

(1) Making the Peripheral Driver Generator project

Make the new Peripheral Driver Generator project "rx220_demo4". For details on how to make the new Peripheral Driver Generator project, refer to section 4.1 (1), Making the Peripheral Driver Generator project. Set the CPU type as follows.

Series : RX200 Group : RX220 Part No. : R5F52206BxFP

Note: If another type of chip is mounted on your RSK board, select corresponding CPU type.

Proj	ect new		x
Pro	oject name:		
rx.	220_demo4		
Dir	ectory:		
C:	\renesas\PDG2_pr	oj Browse	
)evice selection—		_
	Series:	R×200 -	
	Group:	RX220 💌	
	Part No.:	R5F52206BxFP	
	Package:	PLQP0100KB-A	J
	ROM capacity:	256K byte(s)
	RAM capacity:	16K byte(s)
		OK Cancel	



PDG

- 1. The clock setting window opens and the error icons are displayed in the initial state. For icons such as and displayed on window, refer to section 4.1 (2), Initial state.
- 2. For the clock setting, refer to section 4.1 (3), Clock setting.
- (3) Endian setting

PDG

For the endian setting, refer to section 3.3, Endian.

(4) DTCa setting-1

PDG

1. Select "DTCa" tab to open the DTCa setting window.

2. Check "Use data transfer controller".

3. The DTCa vector table will be allocated from 2000h. Set "2000".

😵 PDG2 - [n;220_demo4.pd2 *]	
🛱 Eile View Iool Help	- 8 ×
Data Transfer Controller (DTCa)	<u>^</u>
I v Use bata transfer controller	
Dure settings	
Transfer data read skip enable: Disable 🚽 🖌 Address mode:	
	E
Base address for the DTC vector table address: 2000 h	
Add transfer data settings Add transfer data	
Delete transfer data	
Chain transfer activating source:	
Chain transfer data num:	
Transfer dala start address: h	
Transfer mode:	
Block/Repeat area:	
Transfer unit size.	
Block transfer size:	-
SYSTEM LVDAA CAC LPC ICUB BUSSS DWACA DTCA ELCINO MTUZA POEZA TMR CMT RTCC INDTA SCI RIC RSPI CRC S12ADb CMPA DOC	
Ready 1	

(5) DTCa setting-2



- 1. Click [Add transfer data] to add the transfer data.
- 2. Select "IRQ1 (external pin interrupt)" for the activating source.
- 3. Set "2400" to the transfer data start address.
- 4. Select "Normal transfer mode" for the transfer mode.
- 5. Set "1" to the transfer unit size.
- 6. Set "10" to the transfer count.
- 7. Set "2410" to the source start address.
- 8. Select "Increment" for the source address mode.
- 9. Set "2420" to the destination start address.
- 10. Select "Increment" for the destination address mode.

Address mode:				
	Full-address mode (32 bits) 💌		
Base address for the DTC vector tab	le address:	2000 h		
1	- Transfer data settings			
Add transfer data	Activating source:		2	2
Dalaha kawafaa daha	IRQ1 (external pin interrupt)		-	
Delete ransfer data	Chain transfer activating source	2		
IRQ1 Transfer data				
IIIII Transfer data	Chain transfer data num:			
	Transfer data start address:		2400 h	3
	Transfer mode:	Normal transfer mode	-	4
	Block/Repeat area:	Source side	-	
	Transfer unit size:	1	💌 byte	ə(s)
	Block transfer size:			
	Transfer data size:		1 byte	e(s)
	Transfer count:		10	6
	Total transfer data size:		10 byte	e(s)
	Source start address:		2410 h	7
	Source address mode:	Increment	•	8
	Destination start address:		2420 h	9
	Destination address mode:	Increment		10
	Interrupt control:			_
	Request is transferred to CF	2 2 when specified transfer is	completed	
	C Request is transferred to CF	PU each time DTC transfer is	performed	
				—
	Enable chain transfer			

Tutorial



(6) ICUb setting

PDG

- 1. Select "ICUb" tab to open the ICUb setting window.
- 2. Click "Interrupts" on the tree view.
- 3. Check "Use IRQ1".
- 4. Select "Falling edge" for the detection method of IRQ1.
- 5. Select "CPU (After activating DTC and data transfer completion)".
- 6. CPU interrupt will not be used then set "0" to the CPU interrupt priority level.

PDG2 - [n220_demo4.pd2 *]	
S Elle View Iool Help	- 8 ×
Interrupt Control Unt (ICUb) CPU interrupt priority level: 15	<u> </u>
Detection method: Faling edge	
Interrupt request destination CPU (Alter activating DTC and data transfer completion)	
CPU interrupt priority levet 0 🕂 🔊 Interrupt notification function name: Irg1In/Func 5	-
Exable dinital filter	=
Digital filter sampling clock: PCLK 6	
☐ UseIRQ2	
Detection method: Low level	
Interrupt request destination: CPU 💌	
CPU interrupt priority level: 15-1 Interrupt notification function name: Irg2IntFunc	
	Ŧ
SYSTEM LVDAA CAC LPC A ICUB BUSES DMACA DTCA ELC 10 MTU2A POE2A TMR OMT RTCC IWDTA SCI RIC RSPI CRC S12ADb CMPA DOC	
Ready	

(7) Generating source files **PDG**

To generate source files, click in the tool bar. For details on generating source files, refer to section 4.1 (9), Generating source files.

HEW

(8) Preparing the High-performance Embedded Workshop project

Start the High-performance Embedded Workshop and make RX220 workspace. For details on making High-performance Embedded Workshop project, refer to section 4.1 (10), Preparing the High-performance Embedded Workshop project.

(9) Adding the generated source files to the High-performance Embedded Workshop project

To add the generated source files to High-performance Embedded Workshop, click in on the tool bar. For details on adding the source files to High-performance Embedded Workshop project, refer to section 4.1 (11), Adding the generated source files to the High-performance Embedded Workshop project.



(10) Making the program on High-performance Embedded Workshop

HEW

By changing the part of "main" function, make the following program on High-performance Embedded Workshop.

//Include "R_PG_ <project name="">.h" #include "R_PG_rx220_demo4.h"</project>
//DTC vector table
<pre>#pragma address dtc_vector_table = 0x00002000 uint32_t dtc_vector_table [256];</pre>
<pre>//DTC transfer data storage area (IRQ1) #pragma address dtc_transfer_data_IRQ1 = 0x00002400 uint32_t dtc_transfer_data_IRQ1 [4];</pre>
//Transfer source
<pre>#pragma address dtc_src_data = 0x00002410 uint8_t dtc_src_data [10] = "ABCDEFGHIJ";</pre>
//Transfer destination
<pre>#pragma address dtc_dest_data = 0x00002420</pre>
uint8_t dtc_dest_data [10];
void main(void)
{
//initialize transfer destination
int i; for(i=0; i=10; i+1)
for(i=0; i<10; i++){
}
//Configure I/O port pins that are not available // R_PG_IO_PORT_SetPortNotAvailable();
R_PG_Clock_Set(); // Set up the clock
// Set up the DTC (e.g. vector table address) R_PG_DTC_Set();
// Set up the DTC (transfer data of IRQ1) R_PG_DTC_Set_IRQ1();
R_PG_ExtInterrupt_Set_IRQ1(); // Set up IRQ1
R_PG_DTC_Activate(); // Make the DTC be ready to the trigger while(1);
}

(11) Connecting to the emulator, building the program and downloading

Build the program and download it. For details on connecting to the emulator, building the program, and downloading refer to section 4.1 (13), connecting to the emulator, building the program and executing.

Open the Watch window and add the variable "dtc_dest_data". Expand the array and set it to the real time update to monitor the variable change during execution.

Watch					X	
Name	Value	Address	Туре	Scope		
<pre>Itc_dest_data</pre>		{ 00002420 }	(uint8_t[10])	[Current Scope]		
R [0]	н'00 '.'	{ 00002420 }	(uint8_t)			
R [1]	н'00 '.'	{ 00002421 }	(uint8_t)			
R [2]	н'00 '.'	{ 00002422 }	(uint8_t)			
R [3]	н'00 '.'	{ 00002423 }	(uint8_t)			
R [4]	н'00 '.'	{ 00002424 }	(uint8_t)			
R [5]	н'00 '.'	{ 00002425 }	(uint8_t)			
R [6]	н'00 '.'	{ 00002426 }	(uint8_t)			
R [7]	н'00 '.'	{ 00002427 }	(uint8_t)			
R [8]	н'00 '.'	{ 00002428 }	(uint8_t)			
R [9]	н'00 '.'	{ 00002429 }	(uint8_t)			
Watch1 (Watch2) Watch3) Watch4 /						

(13) Executing the program and monitoring the result of the transfer

HEW

HEW

Start the execution and push the SW1. The value of "dtc_dest_data" on the watch window will change.







4.5 Data transfer between SCIe channels 1 and 5

In this tutorial, SCI channel 1 and 5 will be set up to transfer data in asynchronous mode. Connect the transmission pin of channel 1 (TXD1) and the reception pin of channel 5 (RXD5) on the RSK board as follows.



Note : If there are switches that enables/disables TXD1 and RXD5 on the RSK board, enable it.

Making the Peripheral Driver Generator project
 Make the new Peripheral Driver Generator project
 "rx220_demo5". For details on how to make the new
 Peripheral Driver Generator project, refer to section 4.1

Making the Peripheral Driver Generator project.
 Set the CPU type as follows.

Series : RX200 Group : RX220 Part No. : R5F52206BxFP

Note: If another type of chip is mounted on your RSK board, select corresponding CPU type.

PDG

Pro	ject new		x
Pr	oject name:		ר
D	220_demo5		J
Di	rectory:		
c	:\renesas\PDG2_proj		Browse
Г	Device selection		
	Series:	R×200	•
	Group:	RX220	-
	Part No.:	R5F52206B>	(FP 👤
	Package:	PLQP0100K	3-A
	ROM capacity:		256K byte(s)
	RAM capacity:		16K byte(s)
		OK	Cancel



PDG

- 1. The clock setting window opens and the error icons are displayed in the initial state. For icons such as and displayed on window, refer to section 4.1 (2), Initial state.
- 2. For the clock setting, refer to section 4.1 (3), Clock setting.
- (3) Endian setting

PDG

PDG

For the endian setting, refer to section 3.3, Endian.

(4) SCIe setting

Select "SCI" tab to open the SCIe setting window.

👺 <u>F</u> ile <u>V</u> iew <u>T</u> ool <u>H</u> elp	_ <i>B</i> ×
Senal Communications Interface (SCIe, SCIf) SCI5 SCI5 SCI5 SCI6 SCI9 SCI12 Mode settings Mode: V Function selection: SPI mode: Enable multi-processor communications function Noise filter clock:	D of receiving station:
Data format Data length: Parity bit: Stop bit: SYSTEM LVDAa CAC LPC ICUb Buses DMACA DTCa ELC JO MTU2a POE2a TMR CMT	TCC NDTa SCI RIIC REPI CRC S12ADb CMPA DOC



PDG

1. Select SCI1 on the tree view.

Serial Com	municatio	ns Interface (SCIe, SCIf)
🗖 SCI1		
- SCI5	1	
🔲 SCI6	•	
SCI9		
🔲 SCI12		

- 2. Check "Use this channel".
- 3. Select "Asynchronous mode".
- 4. Select "Transmission" for the function.
- 5. Leave the data format settings at the default.

Channel: SCI1	2
Use this channel	
Mode settings	3
Mode:	Asynchronous mode
Function selection:	Transmission
SPI mode:	Master mode 4
Enable multi-proc	essor communications function ID of receiving station:
🔲 Enable noise can	celling function
Noise filter clock:	Clock signal divided by 1
Data format	5
Data length:	8 bits
Parity bit:	None
Stop bit:	1 bit
Data direction:	LSB-first
Data inversion:	Disable

6. Set the bit rate to "9600bps".

Transfer speed		
		20.000000 MHz
Module operating clock (PCLK) frequency:	1	20.000000 MHz
Transfer clock:	Internal clock	•
6	,	_
Bit rate: 9600 bps	Actual value:	9615.384615bps
	Error:	0.160256%

7. Select "Notify the transmission completion of all data by function call" for the data transmission method.

Г	Data transmission method, Data reception method				
ſ	Data transmission method:	Notify the transmission completion of all data by function call 📃 🧣			
l		Transmit end notification function name:	Sci1TrFunc	J	



(6) SCI5 (receptor) setting

PDG

1. Select SCI5 on the tree view.



- 2. Check "Use this channel".
- 3. Select "Asynchronous mode".
- 4. Select "Reception" for the function.
- 5. Leave the data format settings at the default.

Channel: SCI5	2
Mode settings	3
Mode:	Asynchronous mode
Function selection:	Reception
SPI mode:	Master mode 4
Enable multi-proc	essor communications function ID of receiving station:
🔲 Enable noise can	celling function
Noise filter clock:	Clock signal divided by 1
Data format	5
Data length:	8 bits
Parity bit:	None
Stop bit:	1 bit 💌
Data direction:	LSB-first
Data inversion:	Disable

6. Set the bit rate to "9600bps".

- Transfer speed		
Hansiel speeu		
Module operating clock (PCLK) frequency:		20.000000 MHz
Transfer clock:	Internal clock	•
Bit rate: 9600 bps	Actual value:	9615.384615bps
	Error:	0.160256%

7. Select "Notify the reception completion of all data by function call" for the data reception method.

Data transmission method, Data reception method			
Data transmission method:	Wait at the transmission function until all data has been transmitted		
	Transmit end notification function name:	Sci5TrFunc	7
Data reception method:	Notify the reception completion of all data by function call 📃 🔮		
	Receive end notification function name:	Sci5ReFunc	
	🔲 Notify receive error detection by funct	ion call	
	Receive error notification function name:	Sci5ErFunc	



PDG

The RXD5 can be assigned to RXD5 (PA2) or RXD5 (PA3) or RXD5 (PC2). Select the pin function assignment as follows.

- 1. Select "SYSTEM" tab.
- 2. Select "Pin (Multifunction pin controller)" on tree view.
- 3. Select "Peripheral pin usage" tab.
- 4. Select "SCI5" from the peripheral module list.
- 5. When the mouse pointer is placed on "Assignment" column of RXD5 line, a dropdown button is displayed. Select "PA2/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3" from the dropdown list.

Die View Iool Help	- 8 ×
System 2 U1 Pin name Pin function Assignment Pin No. Direction S Pin (Multifunction pin controller) MTU3 MTU4 SCK5/SSCL5///EXX/ Data receptort(Rk0m) PA2/Rx205/SSCL5///RX055/SSCL5//R005/SSCL5//RX055/SSCL5//R000//RX05//SSCL5//R005/SSCL5//R00//R00//	
General pot P0 General pot P2 General pot P2 General pot P3 Remeral pot P3 Pin function Peripheral pin usage Pin ayout SYSTEM L DAa CAC LPC ICUb Buses DMACA DTCa ELC W0 MT02a POE2a TMR CMT RTCc IWDTa SCI RIIC RSPI CRC S12ADb CMPA DOC Ready	

(8) Generating source files

PDG

To generate source files, click 2 on the tool bar. For details on generating source files, refer to section 4.1 (9), Generating source files.

(9) Preparing the High-performance Embedded Workshop project

Start the High-performance Embedded Workshop and make RX220 workspace. For details on making High-performance Embedded Workshop project, refer to section 4.1 (10), Preparing the High-performance Embedded Workshop project.

(10) Adding the generated source files to the High-performance Embedded Workshop project

To add the generated source files to High-performance Embedded Workshop, click in the tool bar. For details on adding the source files to High-performance Embedded Workshop project, refer to section 4.1 (11), Adding the generated source files to the High-performance Embedded Workshop project.

HEW

(11) Making the program on High-performance Embedded Workshop

HEW

By changing the part of "main" function, make the following program on High-performance Embedded Workshop.

```
//Include "R_PG_<project name>.h"
#include "R_PG_rx220_demo5.h"
//SCI1 transmission data
uint8_t tr_data[10] = "ABCDEFGHIJ";
//SCI5 reception data storage area
uint8_t re_data[10] = "-----";
void main(void)
{
    //Configure I/O port pins that are not available
//
    R_PG_IO_PORT_SetPortNotAvailable();
    // Set up the clock
    R_PG_Clock_Set();
    // Set up the SCI1
    R_PG_SCI_Set_C1();
    // Set up the SCI5
    R_PG_SCI_Set_C5();
    // Start SCI5 reception (number of data : 10)
    R_PG_SCI_StartReceiving_C5( re_data, 10 );
    // Start SCI1 transmission (number of data : 10)
    R_PG_SCI_StartSending_C1( tr_data, 10 );
    while(1);
}
//SCI1 transmission end notification function
void Sci1TrFunc(void)
{
    //Stop SCI1 communication
    R_PG_SCI_StopCommunication_C1();
}
//SCI5 reception end notification function
void Sci5ReFunc(void)
{
    //Stop SCI5 communication
    R_PG_SCI_StopCommunication_C5();
}
```





(12) Connecting to the emulator, building the program and downloading

Build the program and download it. For details on connecting to the emulator, building the program, and downloading refer to section 4.1 (13), connecting to the emulator, building the program and executing.

(13) Adding the variable of the reception data

Open the Watch window and add the variable "re_data". Expand the array and set it to the real time update to monitor the variable change during execution.

HEW

RR] ℓ † × ∌	r R 😳		(
lame	Value	Address	Туре	Scope
-R le_data		' { 00001011 }	(uint8_t[10])	[Current Scope]
R [0]	H'2d '-'	{ 00001011 }	(uint8_t)	
R [1]	H'2d '-'	{ 00001012 }	(uint8_t)	
R [2]	H'2d '-'	{ 00001013 }	(uint8_t)	
R [3]	H'2d '-'	{ 00001014 }	(uint8_t)	
- R [4]	H'2d '-'	{ 00001015 }	(uint8_t)	
R [5]	H'2d '-'	{ 00001016 }	(uint8_t)	
R [6]	H'2d '-'	{ 00001017 }	(uint8_t)	
R [7]	H'2d '-'	{ 00001018 }	(uint8_t)	
R [8]	H'2d '-'	{ 00001019 }	(uint8_t)	
R [9]	H'2d '-'	{ 0000101A }	(uint8_t)	
Watch1 / Watch2) Watch3) Watch4 /				

(14) Executing the program and monitoring the result of the transfer

HEW

HEW

Start the execution and check the value of "re_data" on the watch window.

Watch				X
r R 🗗 📝	🖊 🐴 🗙 🖃 🖬	• RP 😳		
Name	Value	Address	Туре	Scope
🖃 🤻 re_data	""	{ 00001011 }	(uint8_t[10])	[Current Scope]
R [0]	H'41 'A'	{ 00001011 }	(uint8_t)	
R [1]	н'42 'В'	{ 00001012 }	(uint8_t)	
R [2]	H'43 'C'	{ 00001013 }	(uint8_t)	
R [3]	H'44 'D'	{ 00001014 }	(uint8_t)	
R [4]	H'45 'E'	{ 00001015 }	(uint8_t)	
R [5]	H'46 'F'	{ 00001016 }	(uint8_t)	
R [6]	H'47 'G'	{ 00001017 }	(uint8_t)	
R [7]	н'48 'н'	{ 00001018 }	(uint8_t)	
R [8]	H'49 'I'	{ 00001019 }	(uint8_t)	
E. [9]	H'4a 'J'	{ 0000101A }	(uint8_t)	
≪ ▶ Watch1 / Watch2 / Watch3 / Watch4 /				



5. Specification of Generated Functions

Table 5.1 shows generated functions for the RX220.

Table 5.1Generated Functions for the RX220

Clock-generation circuit	
Generated Function	Description
R_PG_Clock_Set	Set up the clocks
R_PG_Clock_WaitSet	Set up the clocks (wait cycle insertion)
R_PG_Clock_Start_MAIN	Start the main clock oscillator
R_PG_Clock_Stop_MAIN	Stop the main clock oscillator
R_PG_Clock_Start_SUB	Start the sub-clock oscillator
R_PG_Clock_Stop_SUB	Stop the sub-clock oscillator
R_PG_Clock_Start_LOCO	Start the low-speed on-chip oscillator (LOCO)
R_PG_Clock_Stop_LOCO	Stop the low-speed on-chip oscillator (LOCO)
R_PG_Clock_Start_HOCO	Start the high-speed on-chip oscillator (HOCO)
R_PG_Clock_Stop_HOCO	Stop the high-speed on-chip oscillator (HOCO)
R_PG_Clock_PowerON_HOCO	Turn on the high-speed on-chip oscillator (HOCO) power supply
R_PG_Clock_PowerON_HOCO	Turn off the high-speed on-chip oscillator (HOCO) power supply
R_PG_Clock_Enable_MAIN_StopDetection	Enable the main clock oscillation stop detection function
R_PG_Clock_Disable_MAIN_StopDetection	Disable the main clock oscillation stop detection function
R_PG_Clock_GetFlag_MAIN_StopDetection	Acquire the main clock oscillation stop detection flag
R_PG_Clock_ClearFlag_MAIN_StopDetection	Clear the main clock oscillation stop detection flag
R_PG_Clock_GetSelectedClockSource	Acquire the current internal clock source
R_PG_Clock_GetClocksStatus	Acquire the status of the clocks
R_PG_Clock_GetHOCOPowerStatus	Acquire the status of high-speed on-chip oscillator (HOCO) power supply

Voltage Detection Circuit (LVDAa)

Generated Function	Description	
R_PG_LVD_Set	Set up the voltage detection circuit	
	(Voltage-monitoring 1 and 2)	
R_PG_LVD_GetStatus	Get the status flag of Voltage Detection Circuit	
R_PG_LVD_ClearDetectionFlag_LVD <voltage circuit<="" detection="" td=""><td>Clear Voltage Monitoring n Voltage Change</td></voltage>	Clear Voltage Monitoring n Voltage Change	
number>	Detection Flag n: 1 or 2	
R_PG_LVD_Disable_LVD <voltage circuit="" detection="" number=""></voltage>	Disable Voltage Monitoring n n: 1 or 2	



Clock Frequency Accuracy Measurement Circuit (CAC)

Generated Function	Description
R_PG_CAC_Set	Set up the CAC and start the measurement
R_PG_CAC_ClearFlag_FrequencyError	Clear the frequency error flag
R_PG_CAC_ClearFlag_MeasurementEnd	Clear the measurement end flag
R_PG_CAC_ClearFlag_Overflow	Clear the overflow flag
R_PG_CAC_StartMeasurement	Start the measurement
R_PG_CAC_StopMeasurement	Stop the measurement
R_PG_CAC_GetStatusFlags	Acquire the CAC status flags
R_PG_CAC_GetCounterBufferRegister	Acquire the counter buffer register (CACNTBR) value
R_PG_CAC_StopModule	Shut down the CAC

Low Power Consumption

Generated Function	Description
R_PG_LPC_Set	Set up the low power consumption functions.
R_PG_LPC_Sleep	Enter sleep mode
R_PG_LPC_AllModuleClockStop	Enter all module clock stop mode
R_PG_LPC_SoftwareStandby	Enter software standby mode
R_PG_LPC_ChangeOperatingPowerControl	Change the operating power control mode
R_PG_LPC_ChangeSleepModeReturnClock	Change the sleep mode return clock source
R_PG_LPC_GetPowerOnResetFlag	Acquire the value of the power-on reset flag
R_PG_LPC_GetLVDDetectionFlag	Acquire the value of the LVD detection flags
R_PG_LPC_GetOperatingPowerControlFlag	Acquire the value of the operating power control mode transition flag
R_PG_LPC_GetStatus	Get the status of the low power consumption functions

Register Write Protection Function

Generated Function	Description
R_PG_RWP_RegisterWriteCgc	Enables or disables writing to registers
	associated with the clock generation circuit
R_PG_RWP_RegisterWriteModeLpcReset	Enables or disables writing to registers
	associated with the operating mode, low
	power comsumption, and software reset
R_PG_RWP_RegisterWriteLvd	Enables or disables writing to registers
	associated with LVD
R_PG_RWP_RegisterWriteMpc	Enables or disables writing to pin-function
	selection registers
R_PG_RWP_GetStatusCgc	Acquires a value indicating whether writing to
	registers associated with the clock
	generation circuit is enabled or disabled
R_PG_RWP_GetStatusModeLpcReset	Acquires a value indicating whether writing to
	registers associated with the operating



	mode, low power comsumption, and
	software reset is enabled or disabled
R_PG_RWP_GetStatusLvd	Acquires a value indicating whether writing to
	registers associated with LVD is enabled or
	disabled
R_PG_RWP_GetStatusMpc	Acquires a value indicating whether writing to
	pin-function selection registers is enabled or
	disabled

Interrupt controller (ICUb)

Generated Function	Description
R_PG_ExtInterrupt_Set_ <interrupt type=""></interrupt>	Set up an external interrupt
R_PG_ExtInterrupt_Disable_ <interrupt type=""></interrupt>	Disable the setting of an external interrupt
R_PG_ExtInterrupt_GetRequestFlag_ <interrupt type=""></interrupt>	Get an external interrupt request flag
R_PG_ExtInterrupt_ClearRequestFlag_ <interrupt type=""></interrupt>	Clear an external interrupt request flag
R_PG_ExtInterrupt_EnableFilter_< <i>interrupt type</i> >	Re-enable the digital filter
R_PG_ExtInterrupt_DisableFilter_ <interrupt type=""></interrupt>	Disable the digital filter
R_PG_SoftwareInterrupt_Set	Set up the software interrupt
R_PG_SoftwareInterrupt_Generate	Generate the software interrupt
R_PG_FastInterrupt_Set	Set an interrupt as the fast interrupt
R_PG_Exception_Set	Set exception handlers

Buses

Generated Function	Description
R_PG_ExtBus_PresetBus	Set the bus priority
R_PG_ExtBus_SetBus	Set the bus pins and the bus error monitoring
R_PG_ExtBus_GetErrorStatus	Acquire the status of bus error generation
R_PG_ExtBus_ClearErrorFlags	Clear the bus-error status registers

DMA controller (DMACA)

Generated Function	Description
R_PG_DMAC_Set_C <channel number=""></channel>	Set up a DMAC channel
R_PG_DMAC_Activate_C <channel number=""></channel>	Make the DMAC be ready for the start trigger
R_PG_DMAC_StartTransfer_C <channel number=""></channel>	Start the one transfer of DMAC (Software trigger)
R_PG_DMAC_StartContinuousTransfer_C <channel number=""></channel>	Start the continuous transfer of DMAC
	(Software trigger)
R_PG_DMAC_StopContinuousTransfer_C <channel number=""></channel>	Stop the software-triggered continuous
	transfer of DMAC
R_PG_DMAC_Suspend_C <channel number=""></channel>	Suspend the data transfer
R_PG_DMAC_GetTransferCount_C <channel number=""></channel>	Get the transfer counter value
R_PG_DMAC_SetTransferCount_C <channel number=""></channel>	Set the transfer counter
R_PG_DMAC_GetRepeatBlockSizeCount_C <channel number=""></channel>	Get the repeat/block size counter value
R_PG_DMAC_SetRepeatBlockSizeCount_C <channel number=""></channel>	Set the repeat/block size count
R_PG_DMAC_ClearInterruptFlag_C <channel number=""></channel>	Get and clear the interrupt request flag
R_PG_DMAC_GetTransferEndFlag_C <channel number=""></channel>	Get the transfer end flag
R_PG_DMAC_ClearTransferEndFlag_C <channel number=""></channel>	Clear the transfer end flag
R_PG_DMAC_GetTransferEscapeEndFlag_C <channel number=""></channel>	Get the transfer escape end flag



R_PG_DMAC_ClearTransferEscapeEndFlag_C <channel number=""></channel>	Clear the escape transfer end flag
R_PG_DMAC_SetSrcAddress_C <channel number=""></channel>	Set the source address
R_PG_DMAC_SetDestAddress_C <channel number=""></channel>	Set the destination address
R_PG_DMAC_SetAddressOffset_C <channel number=""></channel>	Set the address offset
R_PG_DMAC_SetExtendedRepeatSrc_C <channel number=""></channel>	Set the source address extended repeat value
R_PG_DMAC_SetExtendedRepeatDest_C <channel number=""></channel>	Set the destination address extended repeat value
R_PG_DMAC_StopModule_C <channel number=""></channel>	Stop the DMAC channel

Data Transfer Controller (DTCa)

Generated Function	Description
R_PG_DTC_Set	Set up the DTC
R_PG_DTC_Set_ <trigger source=""></trigger>	Set the DTC transfer data
R_PG_DTC_Activate	Make DTC be ready for the trigger
R_PG_DTC_SuspendTransfer	Stop transfer data
R_PG_DTC_GetTransmitStatus	Get transfer data status
R_PG_DTC_StopModule	Shut down the DTC

Event Link Controller (ELC)

Generated Function	Description
R_PG_ELC_Set	Sets the ELC
R_PG_ELC_SetLink_ <peripheral module=""></peripheral>	Sets an event link
R_PG_ELC_DisableLink_ <peripheral module=""></peripheral>	Disables an event link
R_PG_ELC_Set_PortGroup <port group="" number=""></port>	Sets a port group
R_PG_ELC_Set_SinglePort <single-port number=""></single-port>	Sets a single-port pin
R_PG_ELC_AllEventLinkEnable	Enables all event links
R_PG_ELC_AllEventLinkDisable	Disables all event links
R_PG_ELC_Generate_SoftwareEvent	Generates a software event
R_PG_ELC_GetPortBufferValue_Group <port-group number=""></port-group>	Acquires the value of a port buffer register
R_PG_ELC_SetPortBufferValue_Group <port-group number=""></port-group>	Sets a value for a port buffer register
R_PG_ELC_StopModule	Stops the ELC

I/O port

Generated Function	Description
R_PG_IO_PORT_Set_P <port number=""></port>	Set the I/O ports
R_PG_IO_PORT_Set_P <port number=""><pin number=""></pin></port>	Set an I/O port (one pin)
R_PG_IO_PORT_Read_P <port number=""></port>	Read data from Port Input Register
R_PG_IO_PORT_Read_P <port number=""><pin number=""></pin></port>	Read 1-bit data from Port Input Register
R_PG_IO_PORT_Write_P <port number=""></port>	Write data to Port Output Data Register
R_PG_IO_PORT_Write_P <port number=""><pin number=""></pin></port>	Write 1-bit data to Port Output Data Register
R_PG_IO_PORT_SetPortNotAvailable	Handle unavailable pins

Multi-Function Timer Pulse Unit 2 (MTU2a)

Generated Function	Description
R_PG_Timer_Set_MTU_U <unit number="">_<channels></channels></unit>	Set up the MTU
R_PG_Timer_StartCount_MTU_U <unit number="">_C<channel number></channel </unit>	Start the MTU count operation



R_PG_Timer_SynchronouslyStartCount_MTU_U <unit number=""></unit>	Start the MTU count operation of two or more channels simultaneously
R_PG_Timer_HaltCount_MTU_U< <i>unit number</i> >_C< <i>channel</i> number>	Halt the MTU count operation
R_PG_Timer_GetCounterValue_MTU_U< <i>unit</i> number>_C <channel number=""></channel>	Acquire the MTU counter value
R_PG_Timer_SetCounterValue_MTU_U< <i>unit</i> number>_C <channel number="">(_<phase>)</phase></channel>	Set the MTU counter value
R_PG_Timer_GetRequestFlag_MTU_U< <i>unit number</i> >_C< <i>channel</i> <i>number</i> >	Acquire and clear the MTU interrupt flags
R_PG_Timer_StopModule_MTU_U <unit number=""></unit>	Shut down the MTU unit
R_PG_Timer_GetRequestFlag_MTU_U< <i>unit number>_</i> C< <i>channel</i> <i>number></i>	Acquire the general register value
R_PG_Timer_SetTGR_ <general register="">_MTU_U<unit number>_C<channel number=""></channel></unit </general>	Set the general register value
R_PG_Timer_SetBuffer_AD_MTU_U <unit number="">_C<channel number></channel </unit>	Set A/D converter start request cycle set buffer registers (TADCOBRA and TADCOBRB)
R_PG_Timer_SetBuffer_CycleData_MTU_U< <i>unit</i> number>_ <channels></channels>	Set the cycle buffer register
R_PG_Timer_SetOutputPhaseSwitch_MTU_U< <i>unit</i> number>_ <channels></channels>	Switch PWM output level
R_PG_Timer_ControlOutputPin_MTU_U< <i>unit</i> number>_ <channels></channels>	Enable or disable the PWM output
R_PG_Timer_SetBuffer_PWMOutputLevel_MTU_U <unit< td=""><td>Set the PWM output level in the buffer</td></unit<>	Set the PWM output level in the buffer
number>_ <channels></channels>	register
R_PG_Timer_ControlBufferTransfer_MTU_U <unit< td=""><td>Enable or disable buffer transfer from the</td></unit<>	Enable or disable buffer transfer from the
number>_ <channels></channels>	buffer registers to the temporary registers

Port Output Enable 2 (POE2a)

Generated Function	Description
R_PG_POE_Set	Set up the POE
R_PG_POE_SetHiZ_< <i>Timer channels></i>	Place the timer output pins in high-impedance state
R_PG_POE_GetRequestFlagHiZ_ <timer channels="" flag=""></timer>	Acquire the high-impedance request flags
R_PG_POE_GetShortFlag_ <timer channels=""></timer>	Acquire the MTU output short flags
R_PG_POE_ClearFlag_ <timer channels="" flag=""></timer>	Clear the high-impedance request flags and the output short flags

8-bit timer (TMR)

	1
Generated Function	Description
R_PG_Timer_Start_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	Set a TMR and start it counting
R_PG_Timer_HaltCount_TMR_U <unit number="">(_C<channel number="">)</channel></unit>	Halt counting by a TMR
R_PG_Timer_ResumeCount_TMR_U <unit number="">(_C<channel< td=""><td>Resume counting by a TMR</td></channel<></unit>	Resume counting by a TMR
number>)	
R_PG_Timer_GetCounterValue_TMR_U <unit number="">(_C<channel< td=""><td>Get the counter value of a TMR</td></channel<></unit>	Get the counter value of a TMR
number>)	
R_PG_Timer_SetCounterValue_TMR_U <unit number="">(_C<channel< td=""><td>Set the counter value of a TMR</td></channel<></unit>	Set the counter value of a TMR



number>)	
R_PG_Timer_GetRequestFlag_TMR_U <unit number="">(_C<channel< td=""><td>Acquire and clear the TMR interrupt flags</td></channel<></unit>	Acquire and clear the TMR interrupt flags
number>)	
R_PG_Timer_HaltCountElc_TMR_U <unit number="">_C<channel number=""></channel></unit>	Stop the TMR operation that was started by
	the ELC
R_PG_Timer_GetCountStateElc_TMR_U <unit number="">_C<channel< td=""><td>Acquire the state of the TMR operation that</td></channel<></unit>	Acquire the state of the TMR operation that
number>	was started by the ELC
R_PG_Timer_StopModule _TMR_U <unit number=""></unit>	Stop a TMR unit

Compare Match Timer (CMT)	
Generated Function	Description
R_PG_Timer_Set_CMT_U <unit number="">_C<channel number=""></channel></unit>	Set up the CMT
R_PG_Timer_StartCount_CMT_U <unit number="">_C<channel number=""></channel></unit>	Start or resume the CMT count operation
R_PG_Timer_HaltCount_CMT_U <unit number="">_C<channel number=""></channel></unit>	Halt the CMT count
R_PG_Timer_GetCounterValue_CMT_U <unit number="">_C<channel< td=""><td>Acquire the CMT counter value</td></channel<></unit>	Acquire the CMT counter value
number>	
R_PG_Timer_SetCounterValue_CMT_U <unit number="">_C<channel< td=""><td>Set the CMT counter value</td></channel<></unit>	Set the CMT counter value
number>	
R_PG_Timer_StopModule _CMT_U <unit number=""></unit>	Shut down the CMT unit

Shut down the CMT unit
Description
Sets up the RTC and starts its counter
Sets up the RTC of warm start and starts its
counter
Suspends counting by the RTC
Restarts counting by the RTC
Sets the current time
Acquires information on the current state of
the RTC
Performs 30-second unit adjustment
Corrects an error of the timer
Places the RTC in 24-hour mode
Places the RTC in 12-hour mode
Enables automatic correction of errors of the timer
Disables automatic correction of errors of the timer
Enables or disables alarms
Sets the time for an alarm
Specifies the cycle for generating the cyclic
interrupt
Enables the clock output
Disables the clock output
Sets u the RTC and starts its counter
(binary count mode)
Suspends counting by the RTC (binary count mode)



R_PG_RTC_RestartBinary	Restarts counting by the RTC (binary count mode)
R_PG_RTC_SetCurrentTimeBinary	Sets the current time (binay count mode)
R_PG_RTC_GetStatusBinary	Acquires information on the current state of
	the RTC (binary count mode)
R_PG_RTC_ManualErrorAdjustBinary	Corrects an error of the timer
	(binary count mode)
R_PG_RTC_AutoErrorAdjustBinary_Enable	Enables automatic correction of errors of the
	timer (binary count mode)
R_PG_RTC_AutoErrorAdjustBinary_Disable	Disables automatic correction of errors of the
	timer (binary count mode)
R_PG_RTC_SetAlarmTimeBinary	Sets the time for an alarm
	(binary count mode)
R_PG_RTC_SetPeriodicInterruptBinary	Specifies the cycle for generating the cyclic
	interrupt (binary count mode)
R_PG_RTC_ClockOutBinary_Enable	Enables the clock output
	(binary count mode)
R_PG_RTC_ClockOutBinary_Disable	Disables the clock output
	(binary count mode)

Independent Watchdog Timer (IWDTa)

Generated Function	Description
R_PG_Timer_Start_IWDT	Sets up the IWDT and starts its timer
R_PG_Timer_RefreshCounter_IWDT	Refresh the counter
R_PG_Timer_GetStatus_IWDT	Acquires the status flag and count value of
	IWDT

Serial Communications Interface (SCIe, SCIf)

Generated Function	Description
R_PG_SCI_Set_C <channel number=""></channel>	Set a SCI channel
R_PG_SCI_SendTargetStationID_C <channel number=""></channel>	Transmits the ID code of the receiving
	station
R_PG_SCI_StartSending_C <channel number=""></channel>	Start the data transmission
R_PG_SCI_SendAllData_C <channel number=""></channel>	Transmit all data
R_PG_SCI_I2CMode_Send_C <channel number=""></channel>	Transmit data by simple I ² C bus interface
R_PG_SCI_I2CMode_SendWithoutStop_C <channel number=""></channel>	Transmit data by simple I ² C bus interface (no
	stop condition)
R_PG_SCI_I2CMode_GenerateStopCondition_C <channel< td=""><td>Generate a stop condition</td></channel<>	Generate a stop condition
number>	
R_PG_SCI_I2CMode_Receive_C <channel number=""></channel>	Receive data by simple I ² C bus interface
R_PG_SCI_I2CMode_RestartReceive_C <channel number=""></channel>	Receive data by simple I ² C bus interface
	(RE-START condition)
R_PG_SCI_I2CMode_ReceiveLast_C <channel number=""></channel>	Making reception complete in simple I ² C bus
	interface
R_PG_SCI_I2CMode_GetEvent_C <channel number=""></channel>	Get the detected event in the simple I ² C
	mode


R_PG_SCI_SPIMode_Transfer_C <channel number=""></channel>	Transmit data by simple SPI mode
R_PG_SCI_SPIMode_GetErrorFlag_C <channel number=""></channel>	Get the serial reception error flag in the
	simple SPI mode
R_PG_SCI_GetSentDataCount_C <channel number=""></channel>	Acquire the number of transmitted data
R_PG_SCI_ReceiveStationID_C <channel number=""></channel>	Receives the ID code matches the ID of the
	receiving station itself
R_PG_SCI_StartReceiving_C <channel number=""></channel>	Start the data reception
R_PG_SCI_ReceiveAllData_C <channel number=""></channel>	Receive all data
R_PG_SCI_ControlClockOutput_C <channel number=""></channel>	Control the output from the SCKn pin (n: 0,
	1, 5, 6, 8, 9, or 12)
R_PG_SCI_StopCommunication_C <channel number=""></channel>	Stop transmission and reception
R_PG_SCI_GetReceivedDataCount_C <channel number=""></channel>	Acquire the number of received data
R_PG_SCI_GetReceptionErrorFlag_C <channel number=""></channel>	Get the serial reception error flag
R_PG_SCI_ClearReceptionErrorFlag_C <channel number=""></channel>	Clear the serial reception error flag
R_PG_SCI_GetTransmitStatus_C <channel number=""></channel>	Get the state of transmission
R_PG_SCI_StopModule_C <channel number=""></channel>	Shut down a SCI channel

I²C Bus Interface (RIIC)

Generated Function	Description
R_PG_I2C_Set_C <channel number=""></channel>	Set up the I ² C bus interface channel
R_PG_I2C_MasterReceive_C <channel number=""></channel>	Master data reception
R_PG_I2C_MasterReceiveLast_C <channel number=""></channel>	Complete a master reception process
R_PG_I2C_MasterSend_C <channel number=""></channel>	Master data transmission
R_PG_I2C_MasterSendWithoutStop_C <channel number=""></channel>	Master data transmission (No stop condition)
R_PG_I2C_GenerateStopCondition_C <channel number=""></channel>	Generate the stop condition
R_PG_I2C_GetBusState_C <channel number=""></channel>	Get the bus state
R_PG_I2C_SlaveMonitor_C <channel number=""></channel>	Slave bus monitor
R_PG_I2C_SlaveSend_C <channel number=""></channel>	Slave data transmission
R_PG_I2C_GetDetectedAddress_C <channel number=""></channel>	Get the detected address
R_PG_I2C_GetTR_C <channel number=""></channel>	Get the transmit/receive mode
R_PG_I2C_GetEvent_C <channel number=""></channel>	Get the detected event
R_PG_I2C_GetReceivedDataCount_C <channel number=""></channel>	Acquires the count of transmitted data
R_PG_I2C_GetSentDataCount_C <channel number=""></channel>	Acquires the count of received data
R_PG_I2C_Reset_C <channel number=""></channel>	Reset the bus
R_PG_I2C_StopModule_C <channel number=""></channel>	Shut down the I ² C bus interface channel

Serial Peripheral Interface (RSPI)

Generated Function	Description
R_PG_RSPI_Set_C <channel number=""></channel>	Set up a RSPI channel
R_PG_RSPI_SetCommand_C <channel number=""></channel>	Set commands
R_PG_RSPI_StartTransfer_C <channel number=""></channel>	Start the data transfer
R_PG_RSPI_TransferAllData_C <channel number=""></channel>	Transfer all data
R_PG_RSPI_GetStatus_C <channel number=""></channel>	Acquire the transfer status
R_PG_RSPI_GetError_C <channel number=""></channel>	Acquire the error flags
R_PG_RSPI_GetCommandStatus_C <channel number=""></channel>	Acquire the command status
R_PG_RSPI_LoopBack <loopback mode="">_C<channel number=""></channel></loopback>	Set loopback mode



R_PG_RSPI_StopModule_C <channel number=""></channel>	Shut down a RSPI channel
CPC Calculator (CPC)	

CRC Calculator (CRC)	
Generated Function	Description
R_PG_CRC_Set	Set up CRC calculator
R_PG_CRC_InputData	Input a data to CRC calculator
R_PG_CRC_GetResult	Get the the result of calculation
R_PG_CRC_StopModule	Shut down CRC Calculator

12-Bit A/D Converter (S12ADb)

Generated Function	Description
R_PG_ADC_12_Set_S12AD0	Sets up the 12-bit A/D converter
R_PG_ADC_12_StartConversionSW_S12AD0	Starts A/D conversion (by a software trigger)
R_PG_ADC_12_StopConversion_S12AD0	Stops A/D conversion
R_PG_ADC_12_GetResult_S12AD0	Gets the result of A/D conversion of an
	analog input or internal reference voltage
R_PG_ADC_12_GetResult_DblTrigger_S12AD0	Gets the result of A/D conversion in
	response to the second trigger in the
	double-trigger mode
R_PG_ADC_12_GetResult_SelfDiag_S12AD0	Gets the result of A/D conversion as part of
	self diagnosis by the A/D converter
R_PG_ADC_12_StopModule_S12AD0	Shuts down the 12-bit A/D converter

Comparator A (CMPA)

Generated Function	Description	
R_PG_CPA_Set_CP <comparator circuit="" number=""></comparator>	Sets up comparator n	n: A1 or A2
R_PG_CPA_Disable_CP <comparator circuit="" number=""></comparator>	Disable comparator n circuit	n: A1 or A2
R_PG_CPA_GetStatus	Get comparator A status flag	

Data Operation Circuit (DOC)

Generated Function	Description
R_PG_DOC_Set	Set up the Data Operation Circuit
R_PG_DOC_GetStatusFlag	Acquire the status of the data operation
	circuit
R_PG_DOC_GetResult	Acquire the result of data operation
R_PG_DOC_InputData	Input data
R_PG_DOC_UpdateData	Update data
R_PG_DOC_StopModule	Disable the data operation circuit



5.1 Clock-Generation Circuit

5.1.1 R_PG_Clock_Set

Definition	bool R_PG_Clock_Set(void)		
Description	Set up the clocks		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Set		
Doming	 Sets up each clock source and starts the oscillation. Switches the internal clock source to the clock which is specified on GUI. Sets the frequency of the system clock (ICLK), the peripheral module clocks (PCLKB and PCLKD), the FlashIF clock (FCLK), and the external bus clock (BCLK). 		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Set the clock-generation circuit. R_PG_Clock_Set(); }</project></pre>		



5.1.2 R_PG_Clock_WaitSet

Definition	<pre>bool R_PG_Clock_WaitSet(double wait_time)</pre>		
Description	Set up the clocks		
Parameter_	double wait_time	Oscillation stabilization waiting time (in seconds)	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Set	R_CGC_Set	
<u>Details</u>	 Sets up each clock source and starts the oscillation. Switches the internal clock source to the clock which is specified on GUI. This function inserts wait cycles before switching the internal clock source. If wait cycles are not required, use R_PG_Clock_Set. 		
 The actual waiting time may be different from the specified value. 			
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	<pre>void func(void) { //Set the clock- R_PG_Clock_V }</pre>	generation circuit and switch the clock source after waiting 0.5 seconds. VaitSet(0.5);	



5.1.3 R_PG_Clock_Start_MAIN

Definition	bool R_PG_Clock_Start_MAIN(void)	
Description	Start the main clock oscillator	
<u>Conditions for</u> output	The main clock or PLL circuit is set to be used on GUI.	
Parameter_	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Starts the main cloc	k oscillator.
•	• If the main clock is set to be used on GUI, the main clock will start the oscillation in	
	R_PG_Clock_Set.	
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	<pre>void func(void) { //Start the main clock oscillator. R_PG_Clock_Start_MAIN(); }</pre>	



5.1.4 R_PG_Clock_Stop_MAIN

Definition	<pre>bool R_PG_Clock_Stop_MAIN(void)</pre>	
Description	Stop the main clock oscillator	
<u>Conditions for</u> output	The main clock or PLL circuit is set to be used on GUI.	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Stops the main cloc	k oscillator.
	• The main clock oscillator cannot be stopped when the main clock or PLL circuit is used as the internal clock source.	
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	<pre>void func(void) { //Stop the main clock oscillator. R_PG_Clock_Stop_MAIN(); }</pre>	



5.1.5 R_PG_Clock_Start_SUB

Definition	bool R_PG_Clock_Start_SUB(void)	
Description	Start the sub-clock oscillator	
Parameter_	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Starts the sub-clock oscillator.	
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Start the sub-clock oscillator. R_PG_Clock_Start_SUB(); }</project></pre>	



5.1.6 R_PG_Clock_Stop_SUB

Definition	bool R_PG_Clock_Stop_SUB(void)		
Description	Stop the sub-clock oscillator		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Control		
<u>Details</u>	• Stops the sub-clock	Stops the sub-clock oscillator.	
	The sub-clock oscillator cannot be stopped when the sub-clock is used as the internal		
	clock source.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	<pre>void func(void) { //Stop the sub-clock oscillator. R_PG_Clock_Stop_SUB(); }</pre>		



5.1.7 R_PG_Clock_Start_LOCO

Definition	bool R_PG_Clock_Start_LOCO(void)	
Description	Start the low-speed on-chip oscillator (LOCO)	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Starts the low-speed on-chip oscillator (LOCO).	
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Start the low-speed on-chip oscillator (LOCO). R_PG_Clock_Start_LOCO(); }</project></pre>	



5.1.8 R_PG_Clock_Stop_LOCO

Definition	bool R_PG_Clock_Stop_LOCO(void)		
Description	Stop the low-speed of	Stop the low-speed on-chip oscillator (LOCO)	
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Control		
Details	• Stops the low-speed	d on-chip oscillator (LOCO).	
	The low-speed on-chip oscillator (LOCO) cannot be stopped when the LOCO is used as		
	the internal clock source.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	<pre>void func(void) { //Stop the low-speed on-chip oscillator (LOCO). R_PG_Clock_Stop_LOCO(); }</pre>		



5.1.9 R_PG_Clock_Start_HOCO

Definition	bool R_PG_Clock_Start_HOCO(void)		
Description	Start the high-speed on-chip oscillator (HOCO)		
<u>Conditions for</u> output	The high-speed on-ch	nip oscillator (HOCO) is set to be used on GUI.	
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Control		
Details	• Starts the high-speed on-chip oscillator (HOCO).		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func(void) { //Start the high-speed on-chip oscillator (HOCO). R_PG_Clock_Start_HOCO(); }</pre>		



5.1.10 R_PG_Clock_Stop_HOCO

Definition	bool R_PG_Clock_Stop_HOCO(void)	
Description	Stop the high-speed on-chip oscillator (HOCO)	
<u>Conditions for</u> output	The high-speed on-chip oscillator (HOCO) is set to be used on GUI.	
Parameter_	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Stops the high-spee	d on-chip oscillator (HOCO).
	The high-speed on-chip oscillator (HOCO) cannot be stopped when the HOCO is used as the internal clock source.	
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	<pre>void func(void) { //Stop the high-speed on-chip oscillator (HOCO). R_PG_Clock_Stop_HOCO(); }</pre>	



5.1.11 R_PG_Clock_PowerON_HOCO

Definition	bool R_PG_Clock_PowerON_HOCO(void)		
Description	Turn on the high-speed on-chip oscillator (HOCO) power supply		
<u>Conditions for</u> output	The high-speed on-ch	The high-speed on-chip oscillator (HOCO) is set to be used on GUI.	
Parameter_	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Control		
Details	• Turns on the power supply of the high-speed on-chip oscillator (HOCO)		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func(void) { //Turn on the HOCO power supply R_PG_Clock_PowerON_HOCO(); }</pre>		



5.1.12 R_PG_Clock_PowerOFF_HOCO

Definition	bool R_PG_Clock_PowerON_HOCO(void)		
Description	Turn off the high-speed on-chip oscillator (HOCO) power supply		
<u>Conditions for</u> output	The high-speed on-cl	hip oscillator (HOCO) is set to be used on GUI.	
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_Control		
Details	• Turns off the power supply of the high-speed on-chip oscillator (HOCO)		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func(void) { //Turn off the HOCO power supply R_PG_Clock_PowerOFF_HOCO(); }</pre>		



5.1.13 R_PG_Clock_Enable_MAIN_StopDetection

Definition	bool R_PG_Clock_Enable_MAIN_StopDetection(void)	
Description	Enable the main clock oscillation stop detection function	
<u>Conditions for</u> output	The main clock oscillation stop detection function has been set on GUI.	
Parameter_	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Enables the main clock oscillation stop detection function.	
	• If the main clock oscillation stop detection function has been set on GUI, the function is set up and enabled in R_PG_Clock_Set.	
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	<pre>void func(void) { //Enable main clock oscillation stop detection function R_PG_Clock_Enable_MAIN_StopDetection(); }</pre>	



5.1.14 R_PG_Clock_Disable_MAIN_StopDetection

Definition	bool R_PG_Clock_Disable_MAIN_StopDetection(void)	
Description	Disable the main clock oscillation stop detection function	
<u>Conditions for</u> output	The main clock oscil	lation stop detection function has been set on GUI.
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_Control	
Details	Disables the main clock oscillation stop detection function.	
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	<pre>void func(void) { //Disable main clock oscillation stop detection function R_PG_Clock_Disable_MAIN_StopDetection(); }</pre>	



5.1.15 R_PG_Clock_GetFlag_MAIN_StopDetection

Definition	bool R_PG_Clock_GetFlag_MAIN_StopDetection (bool* stop)	
Description	Acquire the main clock oscillation stop detection flag	
<u>Conditions for</u> output	The main clock oscillation stop detection function has been set on GUI.	
Parameter	bool* stop	The address of storage area for the main clock oscillation stop detection flag
		detection mag
Return value	true	Acquisition of the flag succeeded
	false	Acquisition of the flag failed
File for output	R_PG_Clock.c	
RPDL function	R_CGC_GetStatus	
Details	Acquires the main clock oscillation stop detection flag.	
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	bool stop;	
	<pre>void func(void) { //Acquire the main clock oscillation stop detection flag R_PG_Clock_GetFlag_MAIN_StopDetection(&stop); }</pre>	



5.1.16 R_PG_Clock_ClearFlag_MAIN_StopDetection

Definition	bool R_PG_Clock_ClearFlag_MAIN_StopDetection (void)		
Description	Clear the main clock	Clear the main clock oscillation stop detection flag	
<u>Conditions for</u> output	The main clock oscil	lation stop detection function has been set on GUI.	
Parameter	None		
Return value	true	Clearing succeeded	
	false	Clearing failed	
File for output	R_PG_Clock.c	R_PG_Clock.c	
RPDL function	R_CGC_Control		
Details	Clears the main clock oscillation stop detection flag.		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func(void) { //Clear the main clock oscillation stop detection flag R_PG_Clock_ClearFlag_MAIN_StopDetection(); }</pre>		



Definition	<pre>bool R_PG_Clock_GetSelectedClockSource (uint8_t* clock)</pre>	
<u>Description</u>	Acquire the current internal clock source	
<u>Parameter</u>	uint8_t* clock	The address of storage area for the value that corresponds to current internal clock source Correspondence between clock sources and stored values 0:Low-speed on-chip oscillator 1:High-speed on-chip oscillator 2:Main clock 3:Sub-clock 4:PLL circuit
<u>Return value</u>	true false	Acquisition succeeded Acquisition failed
<u>File for output</u> <u>RPDL function</u> <u>Details</u> <u>Example</u>	false Acquisition failed R_PG_Clock.c R_CGC_GetStatus • Acquires the current internal clock source //Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" uint8_t clock; void func(void) { //Acquire the current internal clock source R_PG_Clock_GetSelectedClockSource(&clock);</project>	

5.1.17 R_PG_Clock_GetSelectedClockSource



5.1.18 R_PG_Clock_GetClocksStatus

<u>Definition</u>	bool R_PG_Clock_GetClocksStatus [] (bool* main,	bool* sub,	bool* loco,	bool* iwdt,
	bool* hoco)			
Description	Acquire the status of the clocks			

Parameter	bool* main	The address of storage area for the value of the main clock stop bit (0:Operating 1:Stopped)	
	bool* sub	The address of storage area for the value of the sub-clock stop bit (0:Operating 1:Stopped)	
	bool* loco	The address of storage area for the value of the low-speed on-chip oscillator stop bit	
	bool* iwdt	 (0:Operating 1:Stopped) The address of storage area for the value of the IWDT-dedicated low-speed on-chip oscillator stop bit (0:Operating 1:Stopped) 	
	bool* hoco	The address of storage area for the value of the high-speed on-chip oscillator stop bit (0:Operating 1:Stopped)	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_GetStatus		
<u>Details</u>	 Acquire the oscillation status of the clocks Specify the address of storage area for the item to be acquired. Specify 0 for a item that is not required. 		
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
<pre>bool loco; void func(void) { //Acquire the status of the the low-speed on-chip oscillator R_PG_Clock_GetClocksStatus (0, 0, &loco, 0, 0);</pre>			



5.1.19	R_PG_Clock	_GetHOCOPowerStatus
--------	------------	---------------------

Definition	bool R_PG_Clock_GetHOCOPowerStatus (bool* power)		
Description	Acquire the status of high-speed on-chip oscillator (HOCO) power supply		
Parameter	bool* power	The address of storage area for the value of the HOCO power supply bit (0:ON 1:OFF)	
Return value	true	Acquisition of the flag succeeded	
	false	Acquisition of the flag failed	
File for output	R_PG_Clock.c		
RPDL function	R_CGC_GetStatus		
<u>Details</u>	• Acquires the status of high-speed on-chip oscillator (HOCO) power supply.		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	bool power;		
	<pre>void func(void) { //Acquire the status of HOCO power supply R_PG_Clock_GetHOCOPowerStatus (& power); }</pre>		



5.2 Voltage Detection Circuit (LVDAa)

5.2.1 R_PG_LVD_Set

Definition	bool R_PG_LVD_Set (void)	
Description	Set up the voltage detection circuit (Voltage-monitoring 1 and Voltage-monitoring 2)	
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_LVD.c	
RPDL function	R_LVD_Create	
<u>Details</u> •	Both Voltage-monitoring 1 and Voltage-monitoring 2 can be set up in one function call.	
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) {</project></pre>	
	}	



5.2.2 R_PG_LVD_GetStatus

<u>Definition</u>	<pre>bool R_PG_LVD_GetStatus (bool * lvd1_detect, bool * lvd1_monitor, bool * lvd2_detect, bool * lvd2_monitor)</pre>		
Description		Voltage Detection Circuit	
Parameter	bool * lvd1_detect	The address of storage area for Voltage Monitoring 1 Voltage Change Detection Flag	
	bool *	The address of storage area for Voltage Monitoring 1 Signal Monitor	
	lvd1_monitor	Flag	
	bool * lvd2_detect	The address of storage area for Voltage Monitoring 2 Voltage Change Detection Flag	
	bool *	The address of storage area for Voltage Monitoring 2 Signal Monitor	
	lvd2_monitor	Flag	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_LVD.c		
RPDL function	R_LVD_GetStatus		
Details	• This function acqu	uires the status flag of Voltage Detection Circuit.	
	• Specify 0 for a flag	that is not required.	
Example	//Include "R_PG_ <pt< td=""><td>roject name>.h" to use this function.</td></pt<>	roject name>.h" to use this function.	
	#include "R_PG_def	ault.h"	
	bool lvd1_det, lvd2_det;		
	bool lvd1_mon, lvd2	_mon;	
	void func(void)		
	{		
	// Get the status	flag of Voltage Detection Circuit.	
	R_PG_LVD_G	etStatus(&lvd1_detect, &lvd1_monitor, &lvd2_detect,	
	&lvd2_monitor);		
	if(lvd1_det){		
	//Processin	g when Voltage Monitoring 1 Voltage Change is detected	
	}		
	if(lvd2_det){		
	//Processin	g when Voltage Monitoring 2 Voltage Change is detected	
	}		
	}		



5.2.3 R_PG_LVD_ClearDetectionFlag_LVD<Voltage Detection Circuit number>

Definition	bool R_PG_LVD_ClearDetectionFlag_LVD <voltage circuit="" detection="" number=""> (void) <voltage circuit="" detection="" number="">: 1 or 2</voltage></voltage>	
Description	Clear Voltage Monito	bring n Voltage Change Detection Flag n: 1 or 2
Parameter	None	
Return value	true	Clearing succeeded
	false	Clearing failed
File for output	R_PG_LVD.c	
RPDL function	R_LVD_Control	
Details •	• This function clears Voltage Monitoring n Voltage Change Detection Flag. n: 1 or 2	
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	void func(void)	
	{	
	// Clear Voltage Monitoring 1 Voltage Change Detection Flag.	
	R_PG_LVD_Cle	earDetectionFlag_LVD1();
	}	



5.2.4 R_PG_LVD_Disable_LVD<Voltage Detection Circuit number>

<u>Definition</u>	bool R_PG_LVD_Disable_LVD <voltage circuit="" detection="" number=""> (void) <voltage circuit="" detection="" number="">: 1 or 2</voltage></voltage>		
Description	Disable Voltage Mon	itoring n n: 1 or 2	
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_LVD.c		
RPDL function	R_LVD_Control		
<u>Details</u> •	• This function disables Voltage Monitoring n. n: 1 or 2		
Example	//Include "R_PG_ <project name="">.h" to use this function.</project>		
	#include "R_PG_default.h"		
	void func(void)		
	{		
	// Disable Voltage Monitoring 1.		
	R_PG_LVD_Disable_LVD1();		
	}		



5.3 Clock Frequency Accuracy Measurement Circuit (CAC)

5.3.1 R_PG_CAC_Set

Definition	bool R_PG_CAC_Set(void)	
Description	Set up the CAC and start the measurement	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_CAC.c	
RPDL function	R_CAC_Create	
<u>2 • • • • • • •</u>	 Sets up the clock frequency accuracy measurement circuit (CAC) and starts the measurement. Call R_CGC_Set to set up the clocks before calling this function. 	
<u>Example</u>	Call R_CGC_Set to set up the clocks before calling this function. //Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Set up the clocks R_PG_Clock_Set(); //Set up the CAC and start the measurement R_PG_CAC_Set (void); }</project>	



5.3.2 R_PG_CAC_ClearFlag_FrequencyError

Definition	bool R_PG_CAC_ClearFlag_FrequencyError(void)		
Description	Clear the frequency error flag		
<u>Conditions for</u> output	The frequency error i	The frequency error interrupt (FERRF) is set to be enabled on GUI.	
Parameter_	None		
<u>Return value</u>	true	Clearing succeeded	
	false	Clearing failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_Control		
Details	• Clear the frequency	error flag	
Example	A case where the settin	g has been made in the GUI as follows.	
	• The frequency error	r interrupt (FERRF) has been set	
	• CacErrIntFunc has been specified as the frequency error interrupt (FERRF) notification		
	function name		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void CacErrIntFunc(void) { //Operation when the frequency error interrupt occurs func2();</pre>		
	//Clear the frequ		
	R_PG_CAC_ClearFlag_FrequencyErro(); }		
	void func1(void)		
	{ //Set up the cloc	ks	
	R_PG_Clock_S		
	<pre>//Set up the CAC and start the measurement R_PG_CAC_Set (void);</pre>		
	}		



5.3.3 R_PG_CAC_ClearFlag_MeasurementEnd

Definition	bool R_PG_CAC_ClearFlag_MeasurementEnd(void)		
Description	Clear the measurement end flag		
<u>Conditions for</u> output	The measurement end interrupt (MENDF) is set to be enabled on GUI.		
Parameter	None		
Return value	true	Clearing succeeded	
	false	Clearing failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_Control		
Details	• Clear the measurem	ent end flag	
-	 R_CAC_Control Clear the measurement end flag A case where the setting has been made in the GUI as follows. The measurement end interrupt (MENDF) has been set CacEndIntFunc has been specified as the measurement end interrupt (MENDF) notification function name //Include "R_PG_<project name="">.h" to use this function.</project> #include "R_PG_default.h" void CacEndIntFunc(void) { //Operation when the frequency error interrupt occurs func2(); //Clear the measurement end flag R_PG_CAC_ClearFlag_MeasurementEnd(); //Set up the clocks R_PG_Clock_Set(); //Set up the clocks R_PG_CAC_and start the measurement R_PG_CAC_Set (void); 		



5.3.4 R_PG_CAC_ClearFlag_Overflow

Definition	bool R_PG_CAC_ClearFlag_Overflow(void)		
Description	Clear the overflow flag		
<u>Conditions for</u> output	The overflow interrupt (OVFF) is set to be enabled on GUI.		
Parameter	None		
Return value	true	Clearing succeeded	
	false	Clearing failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_Control		
Details	Clear the overflow flag		
Example	A case where the setting has been made in the GUI as follows.		
•	• The overflow interrupt (OVFF) has been set		
	• CacOvIntFunc has been specified as the overflow interrupt (OVFF) notification function		
	name		
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void CacOvIntFunc(void) { //Operation when the overflow interrupt occurs func2(); //Clear the overflow flag R_PG_CAC_ClearFlag_Overflow(); }</project></pre>		
	<pre>void func1(void) { //Set up the clocks R_PG_Clock_Set(); //Set up the CAC and start the measurement R_PG_CAC_Set (void);</pre>		
	}		



5.3.5 R_PG_CAC_StartMeasurement

Definition	bool R_PG_CAC_StartMeasurement(void)		
Description	Start the measurement		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_Control		
<u>Details</u> •	Resumes the measurement which has been stopped by R_PG_CAC_StopMeasurement.		
<u>Example</u>			



5.3.6 R_PG_CAC_StopMeasurement

Definition	bool R_PG_CAC_StopMeasurement(void)	
Description	Stop the measurement	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_CAC.c	
RPDL function	R_CAC_Control	
Details	Stops the measurement	
Example	Refer to the example of R_PG_CAC_StartMeasurement.	



5.3.7 R_PG_CAC_GetStatusFlags

Definition	bool R_PG_CAC_GetStatusFlags(bool *err, bool *end, bool *ov)		
Description	Acquire the CAC status flags		
Parameter	bool *err	The address of storage area for the frequency error flag	
	bool *end	The address of storage area for the measurement end flag	
	bool *ov	The address of storage area for the overflow flag	
Return value	true	Acquisition of the flags succeeded	
	false	Acquisition of the flags failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_GetStatus		
Details	• Acquires the frequency error flag, the measurement end flag and the overflow flag.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" bool g_err; bool g_end; bool g_ov; void func(void) { //Acquire the CAC status flags bool R_PG_CAC_GetStatusFlags(&g_err, &g_end, &g_ov); }</project></pre>		



Definition	<pre>bool R_PG_CAC_GetCounterBufferRegister(uint16_t *cacntbr_val)</pre>		
Description	Acquire the counter	Acquire the counter buffer register (CACNTBR) value	
Parameter	uint16_t *cacntbr_val	The address of storage area for the counter buffer register (CACNTBR) value	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_GetStatus		
<u>Details</u>	• Acquires the counter buffer register (CACNTBR) value		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint16_t cacntbr_val;		
	<pre>void func(void) { //Acquire the counter buffer register value R_PG_CAC_GetCounterBufferRegister(&cacntbr_val); }</pre>		

5.3.8 R_PG_CAC_GetCounterBufferRegister



5.3.9 R_PG_CAC_StopModule

Definition	<pre>bool R_PG_CAC_StopModule(void)</pre>		
Description	Shut down the CAC		
Parameter_	None		
Return value	true	Stopping succeeded	
	false	Stopping failed	
File for output	R_PG_CAC.c		
RPDL function	R_CAC_Destroy		
Details	Shuts down the clock frequency accuracy measurement circuit (CAC).		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Shut down the CAC R_PG_CAC_StopModule(); }</project></pre>		



5.4 Low Power Consumption

5.4.1 R_PG_LPC_Set

Definition	bool R_PG_LPC_Set (void)		
Description	Set up the low power consumption functions.		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_Create		
<u>Details</u> •	• This function configures the low power conditions.		
•	Call this function l	before starting the clock source for which you have set the oscillation	
	settling time throu	gh the GUI.	
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	void func(void)		
	{		
	//Stop the sub-cl	ock oscillator.	
	R_PG_Clock_Stop_SUB(); // Set up the low power consumption functions. R_PG_LPC_Set (void); //Start the sub-clock oscillator.		
	R_PG_Clock_St	tart_SUB();	
	//Set the clock-generation circuit and switch the clock source after waiting 2 seconds.		
	R_PG_Clock_W	/aitSet(2);	
	}		



5.4.2 R_PG_LPC_Sleep

Definition	bool R_PG_LPC_Sleep (void)		
Description	Enter sleep mode.		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_Control		
<u>Details</u> •	• This function set the system to sleep mode.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { // Enter sleep mode. R_PG_LPC_Sleep(void); }</project></pre>		


5.4.3 R_PG_LPC_AllModuleClockStop

<u>Definition</u>	bool R_PG_LPC_AllModuleClockStop (void)		
Description	Enter all module clock stop mode.		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_Control		
<u>Details</u>	allowed to operate while all module clock stop mode.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { // Enter all module clock stop mode. R_PG_LPC_AllModuleClockStop (void); }</project></pre>		



5.4.4 R_PG_LPC_SoftwareStandby

Definition	bool R_PG_LPC_SoftwareStandby(void)		
Description	Enter software standby mode.		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_Control		
<u>Details</u> •	 This function set the system to software standby mode. Call R_PG_LPC_Set before calling this function to set the operation during software standby mode. 		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { // Set up the low power consumption functions. R_PG_LPC_Set (void); // Enter software standby mode. R_PG_LPC_SoftwareStandby (void); }</project></pre>		



5.4.5 R_PG_LPC_ChangeOperatingPowerControl

Definition	<pre>bool R_PG_LPC_ChangeOperatingPowerControl(uint8_t mode)</pre>		
Description	Change the operating power control mode		
<u>Parameter</u>	uint8_t mode	Operating power control mode 0 : High-speed operating mode 1 : Middle-speed operating mode A 2 : Middle-speed operating mode B 3 : Low-speed operating mode 1 4 : Low-speed operating mode 2	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_Control		
Details	• Changes the operating power control mode.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { // Change the operating power control mode to middle-speed operating mode A R_PG_LPC_ChangeOperatingPowerControl(1); }</project></pre>		



5.4.6 R_PG_LPC_ChangeSleepModeReturnClock

Definition bool R_PG_LPC_ChangeSleepModeReturnClock(uint8_t return_clock)

<u>Description</u> Change the sleep mode return clock source

Γ		
uint8_t return_clock	Sleep mode return clock source	
	0:Switching is disabled 1:HOCO 2:Main clock oscillator)	
true	Setting was made correctly	
false	Setting failed	
R_PG_LPC.c		
D. I.D.C. Constant		
R_LPC_Control		
• Changes the sleep mode return clock source.		
//Include "R_PG_ <project< td=""><td>ct name>.h" to use this function.</td></project<>	ct name>.h" to use this function.	
#include "R_PG_default.	h"	
void func(void)		
{		
// Change the sleep	mode return clock source to HOCO	
• •		
R_PG_LPC_Change	eSleepModeReturnClock(1);	
}		
	true false R_PG_LPC.c R_LPC_Control Changes the sleep mo //Include "R_PG_ <project #include "R_PG_default void func(void) { // Change the sleep</project 	



5.4.7 R_PG_LPC_GetPowerOnResetFlag

Definition	bool R_PG_LPC_GetPowerOnResetFlag (bool *reset)			
Description	Acquire the value of the power-on reset flag.			
Parameter_	bool *reset The address of storage area for the power-on reset flag			
Return value	true	Acquisition succeeded		
	false	Acquisition failed		
File for output	R_PG_LPC.c			
RPDL function	R_LPC_GetStatus			
<u>Details</u> •	 The reset detection flags and the deep software standby cancel request flags are cleared by calling this function. Use R_PG_LPC_GetStatus instead of this function to get these flags simultaneously if needed. 			
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" bool reset; void func(void) { // Acquire the power-on reset flags. R_PG_LPC_GetPowerOnResetFlag(&reset); if(reset){ // Processing when the power-on reset is detected } }</project></pre>			



5.4.8 R_PG_LPC_GetLVDDetectionFlag

Definition	bool R_PG_LPC_GetLVDDetectionFlag (bool * lvd0, bool * lvd1, bool * lvd2)		
Description	Acquire the value of the LVD detection flags.		
Parameter	bool * lvd0	The address of storage area for the LVD0 detection flag	
	bool * lvd1	The address of storage area for the LVD1 detection flag	
	bool * lvd2	The address of storage area for the LVD2 detection flag	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_GetStatus		
<u>Details</u>	• This function acqu	ires the value of the LVD detection flags.	
	• Specify the address	s of storage area for the flags to be acquired.	
	• Specify 0 for a flag that is not required.		
	• The reset detection flags and the deep software standby cancel request flags are cleared		
		ction. Use R_PG_LPC_GetStatus instead of this function to get these	
	flags simultaneously if needed.		
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function.</project>		
	#include "R_PG_defa	ault.h"	
	bool lvd1;		
	bool lvd2;		
	void func(void)		
	{		
	// Acquire the L	VD1 and LVD2 flags.	
	R_PG_LPC_Ge	tLVDDetectionFlag (0, &lvd1, &lvd2);	
	if(lvd1){		
	//Processing	g when the LVD1 is detected	
	}		
	if(lvd2){		
	//Processin	g when the LVD2 is detected	
	3		
	,		



Definition	bool R_PG_LPC_GetOperatingPowerControlFlag(bool * during_transition)		
Description	Acquire the value of the operating power control mode transition flag		
Parameter	bool * during_transition	The address of the storage area for the operating power control mode transition flag	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_LPC.c		
RPDL function	R_LPC_GetStatus		
<u>Details</u>	 This function acquires the value of the operating power control mode transition flag. The reset detection flags and the deep software standby cancel request flags are cleared by calling this function. Use R_PG_LPC_GetStatus instead of this function to get these flags simultaneously if needed. 		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" bool during_transition;</project></pre>		
		ing power control mode transition flag eratingPowerControlFlag (&during_transition);	

5.4.9 R_PG_LPC_GetOperatingPowerControlFlag



5.4.10 R_PG_LPC_GetStatus

Definition	<pre>bool R_PG_LPC_GetStatus(uint16_t *data)</pre>					
Description	Get the status of the low power consumption functions.					
Parameter	uint16_t *data	uint16_t *data The address of storage area for the status data				
Return value	true	Acquisition succeeded				
	false	Acquisition failed				
File for output	R_PG_LPC.h					
RPDL function	R_LPC_GetStatus					
Details	• This function acq	• This function acquires the reset status.				
	-	function, the function of	RPDL R_P	G_LPC_G	etStatus is	s called
	directly.					
	•	hall be stored in the forma	it below.		1.0	
		B15-b9	1		b8	
	0 Operating Power Control Mode transition fla 0: Transition completed 1: During Transition			ransition flag		
	b7-b4		b3	b2	b1	b0
	Reset status (RST	Reset status (RSTSR0) (0: not detected; 1: detected)			00	
		0 LVD2		LVD1	LVD0	Power-on
						reset
	• The RSTSR(LVI	D detection flags) are clea	red by callin	ng this fun	ction.	
	RSTSR.PORF(po	ower-on reset flag) is only	v initialized	by a pin re	eset.	
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>					
	uint16_t data;					
	void func(void)					
	{					
	// Acquire the L	PC status				
	R_PG_LPC_Ge	etStatus(&data);				
	}					



5.5 Register Write Protection Function

5.5.1 R_PG_RWP_RegisterWriteCgc

Definition Description	bool R_PG_RWP_RegisterWriteCgc (bool enable) Enables or disables writing to registers associated with the clock generation circuit			
Parameter	bool enable	Whether writing to registers is enabled or disabled (1: enabled, 0: disabled)		
Return value	true	Setting was made correctly.		
	false	Setting failed.		
File for output	R_PG_RWP.c			
RPDL function	R_RWP_Control			
Details	Enables or disables v	vriting to registers associated with the clock generation circuit.		
Example	// Include "R_PG_ <pro #include "R_PG_defau</pro 	ject name>.h" to use this function. lt.h"		
	bool cgc; bool mode_lpc_reset; bool lvd;			
	<pre>bool b0wi,pfswe; void func1(void) {</pre>			
	<pre>// Enable writing to registers associated with the clock generation circuit. R_PG_RWP_RegisterWriteCgc(1);</pre>			
	 // Enable writing to registers associated with the operating mode, // low power comsumption, and software reset. R_PG_RWP_RegisterWriteModeLpcReset(1); 			
	<pre>// Enable writing to registers associated with LVD. R_PG_RWP_RegisterWriteLvd(1);</pre>			
	<pre>// Enable writing to pin-function selection registers. R_PG_RWP_RegisterWriteMpc(1); }</pre>			
	void func2(void)			
	<pre>{ // Disable writing to registers associated with the clock generation circuit. R_PG_RWP_RegisterWriteCgc(0);</pre>			
	 // Disable writing to registers associated with the operating mode, // low power comsumption, and software reset. R_PG_RWP_RegisterWriteModeLpcReset(0); 			
	<pre>// Disable writing to registers associated with LVD. R_PG_RWP_RegisterWriteLvd(0);</pre>			
	<pre>// Disable writing to pin-function selection registers. R_PG_RWP_RegisterWriteMpc(0);</pre>			



}

{

void func3(void)

// Acquire the value indicating whether writing to registers associated with the clock
// generation circuit is enabled or disabled.

R_PG_RWP_GetStatusCgc(&cgc);

// Acquire the value indicating whether writing to registers associated with

 $\prime\prime$ the operating mode, low power comsumption, and software reset is enabled or $\prime\prime$ disabled.

R_PG_RWP_GetStatusModeLpcReset(&mode_lpc_reset);

// Acquire the value indicating whether writing to registers associated with LVD is // enabled or disabled.

R_PG_RWP_GetStatusLvd(&lvd);

// Acquire the value indicating whether writing to pin-function selection registers is
// enabled or disabled.

R_PG_RWP_GetStatusMpc(&b0wi, &pfswe);

}



5.5.2 R_PG_RWP_RegisterWriteModeLpcReset

Definition Description	bool R_PG_RWP_RegisterWriteModeLpcReset (bool enable) Enables or disables writing to registers associated with the operating mode, low power comsumption, and software reset		
Parameter	bool enable Whether writing to registers is enabled or disabled (1: enabled, 0:		
		disabled)	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RWP.c		
RPDL function	R_RWP_Control		
Details	• Enables or disables writing to registers associated with the operating mode, low power		
	comsumption, and so	ftware reset.	
Example	Refer to the example of	FR_PG_RWP_RegisterWriteCgc.	



5.5.3 R_PG_RWP_RegisterWriteLvd

Definition	bool R_PG_RWP_RegisterWriteLvd (bool enable)		
Description	Enables or disables writing to registers associated with LVD		
Parameter	bool enable Whether writing to registers is enabled or disabled (1: enabled, 0: disabled)		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RWP.c		
RPDL function	R_RWP_Control		
<u>Details</u>	• Enables or disables writing to registers associated with LVD.		
<u>Example</u>	Refer to the example of R_PG_RWP_RegisterWriteCgc.		



5.5.4 R_PG_RWP_RegisterWriteMpc

Definition	bool R_PG_RWP_RegisterWriteMpc (bool enable)		
Description	Enables or disables writing to pin-function selection registers		
Parameter	bool enable Whether writing to registers is enabled or disabled (1: enabled, 0: disabled)		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RWP.c		
RPDL function	R_RWP_Control		
Details	Enables or disables writing to pin-function selection registers.		
Example	Refer to the example of R_PG_RWP_RegisterWriteCgc.		



5.5.5 R_PG_RWP_GetStatusCgc

<u>Definition</u>	<pre>bool R_PG_RWP_GetStatusCgc (bool * cgc)</pre>	
<u>Description</u>	Acquires a value indicating whether writing to registers associated with the clock generation circuit is enabled or disabled	
Parameter	bool * cgc	Whether writing to registers associated with the clock generation circuit is enabled or disabled (1: enabled, 0: disabled)
Return value	true	The value of the flag was successfully acquired.
	false	Acquisition of the value of the flag failed.
File for output	R_PG_RWP.c	
RPDL function	R_RWP_GetStatus	
<u>Details</u> •	Acquires a value indic generation circuit is en	cating whether writing to registers associated with the clock nabled or disabled.
Example	Refer to the example of R_PG_RWP_RegisterWriteCgc.	



5.5.6 R_PG_RWP_GetStatusModeLpcReset

Definition Description	bool R_PG_RWP_GetStatusModeLpcReset (bool * mode_lpc_reset) Acquires a value indicating whether writing to registers associated with the operating mode, low power comsumption, and software reset is enabled or disabled		
Parameter	bool *	Whether writing to registers associated with the operating mode,	
	mode_lpc_reset	low power consumption, and software reset is enabled or disabled	
		(1: enabled, 0: disabled)	
Return value	true	The value of the flag was successfully acquired.	
	false	Acquisition of the value of the flag failed.	
File for output	R_PG_RWP.c		
RPDL function	R_RWP_GetStatus		
Details	• Acquires a value indicating whether writing to registers associated with the operating		
	mode, low power con	msumption, and software reset is enabled or disabled.	
Example [Variable]	Refer to the example of	Refer to the example of R_PG_RWP_RegisterWriteCgc.	



5.5.7 R_PG_RWP_GetStatusLvd

Definition Description	bool R_PG_RWP_GetStatusLvd (bool * lvd) Acquires a value indicating whether writing to registers associated with LVD is enabled or disabled	
Parameter	bool * lvd	Whether writing to registers associated with LVD is enabled or disabled (1: enabled, 0: disabled)
<u>Return value</u>	true false	The value of the flag was successfully acquired. Acquisition of the value of the flag failed.
File for output RPDL function Details •	or disabled.	cating whether writing to registers associated with LVD is enabled
Example	Refer to the example of	R_PG_RWP_RegisterWriteCgc.



5.5.8 R_PG_RWP_GetStatusMpc

Definition Description	bool R_PG_RWP_GetStatusMpc (bool * b0wi, bool * pfswe) Acquires a value indicating whether writing to pin-function selection registers is enabled or disabled	
Parameter	bool * b0wi	Whether writing to the PFSWE bit in the PWPR register is enabled or disabled (1: enabled, 0: disabled)
	bool * pfswe	Whether writing to the PFS register is enabled or disabled (1: enabled, 0: disabled)
Return value	true	The value of the flag was successfully acquired.
	false	Acquisition of the value of the flag failed.
File for output	R_PG_RWP.c	
RPDL function	R_RWP_GetStatus	
<u>Details</u>	• Acquires a value indi or disabled.	cating whether writing to pin-function selection registers is enabled
Example	Refer to the example of	f R_PG_RWP_RegisterWriteCgc.



5.6 Interrupt Controller (ICUb)

5.6.1 R_PG_ExtInterrupt_Set_<interrupt type>

Definition	bool R_PG_ExtInterrupt_Set_< <i>interrupt type</i> > (void)			
Description	Set up an external inter	<interrupt type="">: IRQ0 to IRQ7 or NMI Set up an external interrupt</interrupt>		
Parameter	None	1		
<u>Return value</u>	true	Setting was made correctly		
	false	Setting failed		
File for output	R_PG_ExtInterrupt_< <i>i</i> < <i>interrupt type</i> >: IRG			
RPDL function		upt, R_INTC_CreateExtInterrupt		
		• •		
Details	 The Multifunction Pin Control registers are modified to enable each selected IRQ pin and the I/O Port PMR and PDR registers are modified to set the pin as an input. For IRQn, the pin to be used is set according to the selection in the [Peripheral Pin Usage] window. When the name of the interrupt notification function has been specified in the GUI, if an interrupt occurs in the CPU, the function having the specified name will be called. Create the interrupt notification function as follows: void <<i>name of the interrupt notification function></i> (void) For the interrupt notification function, note the contents of this chapter end, Notes on Notification Functions. If the interrupt propriety level is set to 0 in the GUI, an interrupt handler will not be called even when the external interrupt is input. The request flag can be acquired by calling R_PG_ExtInterrupt_GetRequestFlag_<<i>interrupt type></i>. If [Enable digital filter] is specified in the GUI, the digital filter is enabled when called this function. 			
Example1	A case where Irq0IntFunc has been specified as the name of an interrupt notification			
	function: //Include "R_PG_ <project name="">.h" to use this function.</project>			
	#include "R_PG_default.h"			
<pre>#include "R_PG_default.h" void func(void) { //Set IRQ0. R_PG_ExtInterrupt_Set_IRQ0(); } //IRQ0 notification function void Irq0IntFunc (void) { func_irq0(); //Processing of IRQ0 }</pre>		pt_Set_IRQ0(); ction)		



Example2

A case where the interrupt propriety level is set to 0: //Include "R_PG_<project name>.h" to use this function. #include "R_PG_default.h" void func(void) { bool flag; //Set IRQ0. R_PG_ExtInterrupt_Set_IRQ0(); do{ //Acquire the interrupt request flag for IRQ0. R_PG_ExtInterrupt_GetRequestFlag_IRQ0(&flag); }while(! flag); func_irq0(); //Processing of IRQ0 //Clear the interrupt request flag for IRQ0. R_PG_ExtInterrupt_ClearRequestFlag_IRQ0(); }



5.6.2 R_PG_ExtInterrupt_Disable_<*interrupt type>*

Definition	<pre>bool R_PG_ExtInterrupt_Disable_<interrupt type=""> (void)</interrupt></pre>	
	<interrupt type="">: IRQ</interrupt>	
Description	Disable an external inte	errupt
Parameter	None	
Return value	true	Disabling was made correctly
	false	Disabling failed
File for output	R_PG_ExtInterrupt_< <i>i</i>	
	<interrupt type="">: IRQ</interrupt>	
RPDL function	R_INTC_ControlExtIn	terrupt
Details	• Disables an external	interrupt (IRQ0 to IRQ7).
	• Settings of MPC and signal are retained.	I/O ports registers for the pin being used for the external interrupt
	• When disabling an IF	Qn pin, the Interrupt Request flag will be cleared automatically.
	• When the name of the	e interrupt notification function has been specified in the GUI, the
	function having the s	pecified name may be called once more if a valid event occurs just
	before the interrupt pin is disabled.	
Example	A case where Irq0IntFunc has been specified as the name of an interrupt notification function: //Include "R_PG_ <project name="">.h" to use this function.</project>	
	#include "R_PG_defau	lt.h"
	void func(void)	
	{	
	//Set IRQ0.	
	R_PG_ExtInterrup	ot_Set_IRQ0();
	}	
	//External interrupt (IR	Q0) notification function
	void Irq0IntFunc (void)	
	{	
	//Disable IRQ0.	
	R_PG_ExtInterrup	ot_Disable_IRQ0();
	<pre>func_irq0(); //Processing of IRQ0 }</pre>	
	,	



5.6.3 R_PG_ExtInterrupt_GetRequestFlag_<interrupt type>

Definition	<pre>bool R_PG_ExtInterrupt_GetRequestFlag_<interrupt type=""> (bool * flag)</interrupt></pre>		
	<interrupt type="">: IRC</interrupt>	<interrupt type="">: IRQ0 to IRQ7 or NMI</interrupt>	
Description	Get an external interrup	Get an external interrupt request flag	
Parameter	bool * flag	The address of storage area for the interrupt request flag	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_ExtInterrupt_< <i>interrupt type</i> >.c < <i>interrupt type</i> >: IRQ0 to IRQ7 or NMI		
RPDL function	R_INTC_GetExtInterruptStatus		
<u>Details</u>	• Acquires the interrupt request flag for an external interrupt (IRQ0 to IRQ7 or the NMI). When an interrupt is requested, 'true' is entered in the specified destination for storage of the flag's value.		
Example	Refer to the Example2 of R_PG_ExtInterrupt_Set_ <interrupt type=""></interrupt>		



5.6.4 R_PG_ExtInterrupt_ClearRequestFlag_<interrupt type>

Definition	<pre>bool R_PG_ExtInterrupt_ClearRequestFlag_<interrupt type=""> (void) <interrupt type="">: IRQ0 to IRQ7 or NMI</interrupt></interrupt></pre>		
Description		Clear an external interrupt request flag	
Parameter	None		
Return value	true	Clearing flag succeeded	
	false	Clearing flag failed	
File for output	R_PG_ExtInterrupt_< <i>interrupt type</i> >.c < <i>interrupt type</i> >: IRQ0 to IRQ7 or NMI		
RPDL function	R_INTC_ControlExtInterrupt		
<u>Details</u>	 Clears the interrupt request flag for an external interrupt (IRQ0 to IRQ7 or NMI). If the level-sensitive interrupt is selected, the interrupt request flag is cleared when high-level is input to the interrupt pin. The request flag of level-sensitive interrupt cannot be cleared by this function. 		
Example	Refer to the Example2	Refer to the Example2 of R_PG_ExtInterrupt_Set_ <interrupt type=""></interrupt>	



5.6.5 R_PG_ExtInterrupt_EnableFilter_<*interrupt type>*

<u>Definition</u>	<pre>bool R_PG_ExtInterrupt_EnableFilter_<interrupt type=""> (uint32_t div) <interrupt type="">: IRQ0 to IRQ7 or NMI</interrupt></interrupt></pre>	
Description	Re-enable the digital filter	
Conditions for	When [Enable digital f	ilter] is specified in the GUI.
<u>output</u>		
Parameter	uint32_t div	Peripheral module clock division values
		1: digital filter sampling clock = PCLK
		8: digital filter sampling clock = PCLK/8
		32: digital filter sampling clock = PCLK/32
		64: digital filter sampling clock = PCLK/64
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_ExtInterrupt_< <i>i</i> < <i>interrupt type</i> >: IRC	Q0 to IRQ7 or NMI
RPDL function	R_INTC_ControlExtIn	terrupt
<u>Details</u>	• The digital filter disabled by R_PG_ExtInterrupt_DisableFilter_< <i>interrupt type></i> is	
	enabled, and digital filter sampling clock is set again.	
<u>Example</u>	When [Use IRQ0] is specified in the GUI ([Enable digital filter] is specified)	
		ect name>.h" to use this function.
	#include "R_PG_default.h"	
	void func1(void)	
	{	
		(); //The clock-generation circuit has to be set first.
	R_PG_ExtInterrup	pt_Set_IRQ0(); //Set IRQ0 (Enabling digital filter)
	}	
	void func2(void)	
	{	
	R_PG_ExtInterrup	pt_DisableFilter_IRQ0(); //Disabling digital filter
	R_PG_ExtInterrupt_EnableFilter_IRQ0(1); //Re-enabling the digital filter }	



5.6.6 R_PG_ExtInterrupt_DisableFilter_<interrupt type>

Definition Description	bool R_PG_ExtInterrupt_DisableFilter_< <i>interrupt type></i> (void) < <i>interrupt type></i> : IRQ0 to IRQ7 or NMI Disable the digital filter		
	Ũ		
Conditions for	When [Enable digital fi	lter] is specified in the GUI.	
<u>output</u>			
Parameter	None		
D . 1			
Return value	true	Disabling was made correctly	
	false	Disabling failed	
File for output	R_PG_ExtInterrupt_< <i>interrupt type</i> >.c		
	<interrupt type="">: IRQ0 to IRQ7 or NMI</interrupt>		
RPDL function	R_INTC_ControlExtInterrupt		
<u>Details</u>	• The digital filter is disabled.		
	• Disable the digital filter before transition to Software Standby Mode. To use the digital		
	filter again after retur	filter again after return from software standby mode, call	
	R_PG_ExtInterrupt_EnableFilter_< <i>interrupt type</i> >.		
<u>Example</u>	Refer to the example of	R_PG_ExtInterrupt_EnableFilter_< <i>interrupt type</i> >	



5.6.7 R_PG_SoftwareInterrupt_Set

Definition	<pre>bool R_PG_SoftwareInterrupt_Set(void)</pre>	
Description	Set up the software interrupt	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_SoftwareInterru	pt.c
RPDL function	R_INTC_CreateSoftwa	reInterrupt
Details	• Sets up the software i	nterrupt.
	• The software interrup	t cannot be generated by calling this function. To generate the
	software interrupt, ca	ll R_PG_SoftwareInterrupt_Generate.
<u>Example</u>	A case where SwIntFur	c was specified as the name of the software interrupt notification
	function in the GUI.	
	//Include "R_PG_ <proj< td=""><td>ect name>.h" to use this function.</td></proj<>	ect name>.h" to use this function.
	#include "R_PG_default.h"	
	void SwIntFunc(void);	
	void func(void)	
	{	
	//Set up the software interrupt	
	R_PG_SoftwareInterrupt_Set();	
	//Generate the software interrupt	
	R_PG_SoftwareInterrupt_Generate();	
	}	
	void SwIntFunc(void)	
	{	
	//Processing of sof	tware interrupt
	}	
	<u> </u>	



5.6.8 R_PG_SoftwareInterrupt_Generate

Definition	<pre>bool R_PG_SoftwareInterrupt_Generate(void)</pre>		
Description	Generate the software interrupt		
Parameter	None		
Return value	true	Generating was made correctly	
	false	Generating failed	
File for output	R_PG_SoftwareInterru	R_PG_SoftwareInterrupt.c	
RPDL function	R_INTC_Write		
<u>Details</u>	 Generates the software interrupt. Call R_PG_SoftwareInterrupt_Set before calling this function to set up the software 		
Example	interrupt. Refer to the example of R_PG_SoftwareInterrupt_Set		



5.6.9 R_PG_FastInterrupt_Set

Definition	bool R_PG_FastInterrupt_Set (void)	
Description	Set up the fast interrupt	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_FastInterrupt.c	
RPDL function	R_INTC_CreateFastInterrupt	
<u></u>	 Sets the interrupt source specified in the GUI as the fast interrupt. The specified interrupt source is not set or enabled. The interrupt source to be set as the fast interrupt must be set and enabled by the functions for the peripheral module. This function uses an unconditional trap instruction (BRK) to set the fast-interrupt vector register (FINTV). If interrupts are disabled (the interrupt enable bit (I) of the processor status word is 0), this function will be locked. The interrupt handler that is specified as a fast interrupt will be compiled as a fast 	
<u>Example</u>	<pre>interrupt handler by specifying fint in #pragma interrupt declaration. A case where IRQ0 has been specified as the fast interrupt in the GUI: //Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Set IRQ0 as the fast interrupt. R_PG_FastInterrupt_Set (); //Set IRQ0. R_PG_ExtInterrupt_Set_IRQ0(); } </project></pre>	



5.6.10 R_PG_Exception_Set

Definition	bool R_PG_Exception_Set (void)		
Description	Set the exception handlers		
Parameter	None		
Return value	true Setting was made correctly		
	false	Setting failed	
File for output	R_PG_Exception.c		
RPDL function	R_INTC_CreateExcept	ionHandlers	
<u>Details</u>	 Sets the exception notification functions. If an exception for which the name of the exception notification function was specified in the GUI occurs after this function is called, the function with the specified name will be called. Create the exception notification function as follows: void <<i>name of the exception notification function</i>> (void) For the exception notification function, note the contents of this chapter end, Notes on 		
<u>Example</u>	Notification Functions. A case where the following exception notification functions have been set in the GUI: Privileged instruction exception: PrivInstExcFunc Undefined instruction exception: UndefInstExcFunc //Include "R_PG_ <project name=""> h" to use this function</project>		
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Set the exception handlers. R_PG_Exception_Set(); } void PrivInstExcFunc(){ func_pi_excep(); //Processing in response to a privileged instruction exception } void UndefInstExcFunc (){ func_ui_excep(); //Processing in response to an undefined instruction exception }</project></pre>		



5.7 Buses

5.7.1 R_PG_ExtBus_PresetBus

Definition	bool R_PG_ExtBus_PresetBus(void)	
Description	Set the bus priority	
Conditions for	The bus priority has been set on GUI	
<u>output</u>		
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_ExtBus.c	
RPDL function	R_BSC_Set	
Details	Sets the bus priority.If required, call this f	unction before calling R_PG_ExtBus_SetBus.
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" void func(void) {</project>	



5.7.2 R_PG_ExtBus_SetBus

Definition	bool R_PG_ExtBus_SetBus(void)		
Description	Set up the bus error monitoring		
Parameter	None		
Return value	true Setting was made correctly		
	false	Setting failed	
<u>File for output</u> <u>RPDL function</u> <u>Details</u>	• The bus error interrup enabled on GUI, the occurs. Create the int void <i><name i="" i<="" of="" the=""></name></i>	R_BSC_Create Sets up the bus error monitoring. The bus error interrupt is set by this function. If the bus error interrupt has been set to be enabled on GUI, the function having the specified name will be called when an interrupt occurs. Create the interrupt notification function as follows: void < <i>name of the interrupt notification function</i> > (void) For the interrupt notification function, note the contents of the section Notes on	
	 The status of bus error generation can be acquired by calling R_PG_ExtBus_GetErrorStatus. If required, call R_PG_ExtBus_PresetBus before calling this function. 		
<u>Example</u>			



5.7.3 R_PG_ExtBus_GetErrorStatus

<u>Definition</u>	<pre>bool R_PG_ExtBus_GetErrorStatus (bool * addr_err, uint8_t * master, uint16_t * err_addr)</pre>		
Description	Acquire the status of bus error generation		
<u>Parameter</u>	bool * addr_err uint8_t * master uint16_t * err_addr	The address of storage area for the illegal address access error flag The address of storage area for ID code of bus master that accessed a bus when a bus error occurred ID code of bus master: 0:CPU 3:DMAC/DTC The address of storage area for upper 13 bits of an address that was accessed when a bus error occurred	
Return value	true	Acquisition succeeded.	
	false	Acquisition failed.	
File for output	R_PG_ExtBus.c		
RPDL function	R_BSC_GetStatus		
<u>Details</u>	 Acquires the status of bus error generation from the bus error status registers. Specify the address of storage area for an item to be acquired. Specify 0 for an item that is not required. 		
Example	A case where BusErrFunc has been specified as the name of the bus error interrupt notification function.		
	//Include "R_PG_ <proj #include "R_PG_defau</proj 	ect name>.h" to use this function. lt.h"	
	<pre>void func(void) { //Set up the bus er R_PG_ExtBus_Se } //Bus error notification</pre>	etBus();	
	<pre>void BusErrFunc(void) { bool addr_err; uint8_t master; uint16_t err_addr;</pre>		
	<pre>//Aquire bus error status R_PG_ExtBus_GetErrorStatus(&addr_err, &master, &err_addr); if(addr_err){ //Processing when illegal address access error occurs }</pre>		
	//Clear the bus err R_PG_ExtBus_Cl }		



5.7.4 R_PG_ExtBus_ClearErrorFlags

Definition	bool R_PG_ExtBus_ClearErrorFlags(void)		
Description	Clear the bus-error status registers		
Parameter_	None		
Return value	true	Clearing succeeded	
	false	Clearing failed	
File for output	R_PG_ExtBus.c		
RPDL function	R_BSC_Control		
Details	• Clears the bus-error status registers (illegal address access error flag, ID code of bus master and a value of accessed address).		
<u>Example</u>	A case where BusErrFu notification function.	inc has been specified as the name of the bus error interrupt	
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func(void) { //Set up the bus error monitoring. R_PG_ExtBus_SetBus(); } //Bus error notification function void BusErrFunc(void) { bool addr_err; }</pre>		
	<pre>uint8_t master; uint16_t err_addr; //Aquire bus error status R_PG_ExtBus_GetErrorStatus(&addr_err, &master, &err_addr); if(addr_err){ //Processing when illegal address access error occurs } //Clear the bus error status registers R_PG_ExtBus_ClearErrorFlags(); }</pre>		



5.8 DMA controller (DMACA)

5.8.1 R_PG_DMAC_Set_C<channel number>

Definition	<pre>bool R_PG_DMAC_Set_C<channel <channel="" num="" number="">: 0 to 3</channel></pre>	bber> (void)	
Description	Set up a DMAC channel		
Parameter	None		
Return value	true Setting was mad	e correctly.	
	false Setting failed.		
File for output	R_PG_DMAC_C < <i>channel number</i> >.c		
	<i><unit number=""></unit></i> : 0 to 3		
<u>RPDL function</u> Details	R_DMAC_Create		
	 Releases the DMAC from the module-stop and makes initial settings. If an interrupt was selected as a transfer start trigger, the DMAC channel will be ready for the interrupt signal by calling R_PG_DMAC_Activate_C<<i>channel number</i>> after calling this function. If the software trigger was selected as a transfer start trigger, DMAC channel will start the data transfer when calling R_PG_DMAC_StartTransfer_C<<i>channel number</i>> or R_PG_DMAC_StartContinuousTransfer_C<<i>channel number</i>> after calling this function. The DMAC interrupt is set by this function. When the name of the interrupt notification function having the specified name will be called. Create the interrupt notification function as follows: void <<i>name of the interrupt notification function</i>> (void) For the interrupt notification function, note the contents of this chapter end, Notes on Notification Functions. To transfer the SCI transmission data by DMAC, make the following settings. 		
	DMAC settings Transfer request source	: TXI1 (SCI1 transmit data empty interrupt)	
	Operation when the transfer completes	: Clear the interrupt flag of the activation source	
	Destination start address	: Address of Transmit Data Register (TDR)	
		also from the program. Refer the usage example 2 and 3.	
	Destination address update mode	: Fixed	
	Length of a single data	: 1 byte	
	SCIe setting		
	Data transmission method : Transfer the transmitted ser		
	 For usage of function, refer to example 2. To transfer the SCI reception data by DMAC, make the following settings. DMAC settings 		
	Transfer request source	: RXI1 (SCI1 receive data full interrupt)	
	Operation when the transfer completes	: Clear the interrupt flag of the activation source	
	Source start address	: Address of Receive Data Register (RDR)	
		also from the program. Refer the usage example 2 and 3.	
	Source address update mode	: Fixed	
	Length of a single data	: 1 byte	



SCIe setting
Data transmission method : Transfer the received serial data by DMAC
For usage of function, refer to example 3.
A case where IRQ0 activates DMA transfer
• IRQ0 interrupt was selected as a transfer start trigger of DMAC0 in GUI.
• Dmac0IntFunc was specified as the DMA interrupt notification function name in the GUI.
DMAC was selected as an interrupt request destination for IRQ0.
#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>
void func(void)
R_PG_DMAC_Set_C0(); //Set up DMAC0
R_PG_ExtInterrupt_Set_IRQ0(); //Set up IRQ0
R_PG_DMAC_Activate_C0(); //Make DMAC0 be ready for the transfer start trigger
}
//DMA interrupt notification function
void Dmac0IntFunc (void)
R_PG_DMAC_StopModule_C0(); //Stop DMAC
}
A case where the SCI transmission data is transferred by DMAC
 DmacOIntFunc was specified as the DMA interrupt notification function name in the GUI.
• The SCI1 transmit data empty interrupt is selected as a DMA transfer trigger.
#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>
volatile bool sci_dma_transfer_complete; //DMA transfer end flag
uint8_t tr[]="ABCDEFG"; //Data source
<pre>void func(void) {</pre>
<pre>{ //Initialize DMA transfer end flag sci_dma_transfer_complete = false; }</pre>
R_PG_Clock_Set(); //The clock-generation circuit has to be set first.
R_PG_SCI_Set_C1(); //Set up SCI1
R_PG_DMAC_Set_C0(); //Set up DMAC0
//Set source address, destination address and transfer counter
R_PG_DMAC_SetSrcAddress_C0(tr);
R_PG_DMAC_SetDestAddress_C0((void*)&(SCI1.TDR)); R_PG_DMAC_SetTransferCount_C0(8);
//Make DMAC0 be ready for the transfer start trigger
R_PG_DMAC_Activate_C0();
//Enable the SCI1 transmission (TXI interrupt occurs and DMA transfer starts)
R_PG_SCI_SendAllData_C1(
PDL_NO_PTR, PDL_NO_DATA
);
<pre>// Wait for the DMAC to complete the transfer while (sci_dma_transfer_complete == false);</pre>
}

//DMA interrupt notification function

RENESAS

void Dmac0IntFunc (void)
{
 //SCI transmit end flag
 bool sci_transfer_complete;
 sci_transfer_complete = false;
 // Wait for the SCI to complete the transmission
 do{
 R_PG_SCI_GetTransmitStatus_C1(&sci_transfer_complete);
 } while(! sci_transfer_complete);
 //Stop the SCI
 R_PG_SCI_StopCommunication_C1();
 //Stop the DMAC
 R_PG_DMAC_StopModule_C0();
 sci_dma_transfer_complete = true;
}

Example 3

A case where the SCI reception data is transferred by DMAC

Dmac0IntFunc was specified as the DMA interrupt notification function name in the GUI.

```
The SCI1 receive data empty interrupt is selecited as a DMA transfer trigger.
```

#inc	lude "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>
	tile bool sci_dma_transfer_complete; //DMA transfer end flag 8_t re[]=""; //Data destination
	func(void)
{	<pre>//Initialize DMA transfer end flag sci_dma_transfer_complete = false;</pre>
	R_PG_SCI_Set_C1(); //Set up SCI1
	R_PG_DMAC_Set_C0(); //Set up DMAC0
	//Set source address, destination address and transfer counter R_PG_DMAC_SetSrcAddress_C0((void*)&(SCI1.RDR)); R_PG_DMAC_SetDestAddress_C0(re); R_PG_DMAC_SetTransferCount_C0(8);
	//Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0();
	<pre>//Enable the SCI1 reception R_PG_SCI_ReceiveAllData_C1(PDL_NO_PTR, PDL_NO_DATA).</pre>
});
	MA interrupt notification function I Dmac0IntFunc (void)
l	//Stop the SCI reception R_PG_SCI_StopCommunication_C1();
}	//Stop the DMAC R_PG_DMAC_StopModule_C0();



5.8.2 R_PG_DMAC_Activate_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_Activate_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Make the DMAC be ready for the start trigger		
<u>Conditions for</u> output Parameter	An interrupt is selected as a transfer start trigger		
<u>Return value</u>			
<u>Return value</u>	true Setting was made correctly. false Setting failed.		
File for output	R_PG_DMAC_C < <i>channel number</i> >.c < <i>channel number</i> >: 0 to 3		
RPDL function	R_DMAC_Control		
<u>Details</u>	 This function makes the DMAC channel be ready for the transfer start trigger. This function is genetarted when an interrupt is selected as a transfer start trigger. Call R_PG_DMAC_Set_C<<i>channel number></i> to set up a DMAC channel before calling this function. 		
Example	A case where the setting is made as follows.		
	• IRQ0 was selected as a transfer start trigger of DMAC0 in normal transfer mode		
	Dmac0IntFunc was specified as the DMA0 interrupt notification function name		
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>		
	<pre>void func(void) { //Set up DMAC0</pre>		
	R_PG_DMAC_Set_C0();		
	//Set IRQ0 R_PG_ExtInterrupt_Set_IRQ0();		
	<pre>//Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); }</pre>		
	<pre>//DMA interrupt notification function void Dmac0IntFunc (void) {</pre>		
	//Stop the DMAC R_PG_DMAC_StopModule_C0(); }		


5.8.3 R_PG_DMAC_StartTransfer_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_StartTransfer_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Start the one transfer of DMAC (Software trigger)		
Conditions for	The software trigger is s	elected as a transfer start trigger	
<u>output</u>			
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_DMAC_C <char <channel number="">: 0</channel></char 		
RPDL function	R_DMAC_Control		
<u>Details</u>	• This function starts DM	A transfer of the channel specified the software trigger as a	
	transfer start trigger.		
	• A DMA transfer request is cleared automatically when data transfer is started.		
<u>Example</u>	A case where the setting is	s made as follows.	
	• The software trigger was selected as a transfer start trigger of DMAC0 in normal transfer		
	mode		
	Dmac0IntFunc was sp	ecified as the DMA interrupt notification function name	
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>		
	volatile bool transferred	;	
	void func(void)		
	{ transferred = false;		
	//Set up DMAC0 R_PG_DMAC_Set_C0();		
	<pre>while(transferred == false){ //Start the DMA transfer of DMAC0 R_PG_DMAC_StartTransfer_C0();</pre>		
	<pre>} //Stop the DMAC R_PG_DMAC_StopModule_C0(); }</pre>		
	//DMA interrupt notification function void Dmac0IntFunc (void)		
	{ transferred = true;		
	}		



5.8.4 R_PG_DMAC_StartContinuousTransfer_C<*channel number*>

<u>Definition</u>	<pre>bool R_PG_DMAC_StartContinuousTransfer_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Start the continuous transfer of DMAC (Software trigger)		
<u>Conditions for</u> output	The software trigger is s	elected as a transfer start trigger	
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_DMAC_C <chan <channel number="">: 0</channel></chan 		
RPDL function	R_DMAC_Control		
<u>Details</u>	 This function starts DMA transfer of the channel specified the software trigger as a transfer start trigger. This function enables continuous DMA transfer because a DMA transfer request is generated again after completion of a transfer. 		
<u>Example</u>	A case where the setting is	-	
	 The software trigger was selected as a transfer start trigger of DMAC0 in normal transfer mode Dmac0IntFunc was specified as the DMA interrupt notification function name #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); //Start the DMA transfer of DMAC0 R_PG_DMAC_StartContinuousTransfer_C0(); { 		
	<pre>//DMA interrupt notification void Dmac0IntFunc (void) { //Stop the DMAC R_PG_DMAC_Stop }</pre>	id)	



5.8.5 R_PG_DMAC_StopContinuousTransfer_C<*channel number*>

<u>Definition</u>	<pre>bool R_PG_DMAC_StopContinuousTransfer_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Stop the software-triggered continuous transfer of DMAC		
Conditions for	The software trigger is s	elected as a transfer start trigger	
<u>output</u>			
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_DMAC_C < <i>char</i>	inel number>.c	
	<channel number="">: 0 t</channel>	to 3	
RPDL function	R_DMAC_Control		
<u>Details</u>	• This function clears DMA transfer request of the channel specified the software trigger as a transfer start trigger.		
Example	A case where the setting is	s made as follows.	
	 The software trigger was selected as a transfer start trigger of DMAC0 in normal transfer mode #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> 		
	void func1(void)	//include K_1 O_ <project name="">.n to use uns function.</project>	
	{		
	//Set up DMAC0 R_PG_DMAC_Set_	C0();	
	<pre>//Start the DMA transfer of DMAC0 R_PG_DMAC_StartContinuousTransfer_C0(); }</pre>		
	void func2(void)		
	<pre>{ //Clear DMA transfer request by software R_PG_DMAC_StopContinuousTransfer_C0(); }</pre>		



5.8.6 R_PG_DMAC_Suspend_C<channel number>

Definition	<pre>bool R_PG_DMAC_Suspend_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Suspend the data transfer		
Parameter	None		
Return value	true	Suspending succeeded.	
	false	Suspending failed.	
File for output	R_PG_DMAC_C <chan <channel number="">: 0</channel></chan 		
RPDL function	R_DMAC_Control		
<u>Details</u>	 This function suspends(disables) the DMA transfer. This function can suspend the DMA transfer triggered by hardware. To resume the transfer, when interrupt is selected as a transfer start trigger, clear the interrupt request flag of trigger source and call R_PG_DMAC_Activate_C<<i>channel number></i> to make the DMAC channel be ready for the transfer start trigger. 		
Example			

RENESAS

5.8.7 R_PG_DMAC_GetTransferCount_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_GetTransferCount_C<channel number=""> (uint16_t * count) < channel number > : 0 to 3</channel></pre>	
Description	Get the transfer counter	value
Parameter_	uint16_t * count	The address of storage area for the counter value
<u>Return value</u>	true false	Acquisition succeeded Acquisition failed.
File for output	R_PG_DMAC_C <chan <channel number="">: 0</channel></chan 	
RPDL function	R_DMAC_GetStatus	
<u>Details</u>	 This function gets the current transfer counter value. The DMA interrupt request flag (IR flag) is cleared in this function. Call R_PG_DMAC_ClearInterruptFlag_C<<i>channel number></i> to get the DMA interrupt request flag before calling this function if needed. 	
<u>Example</u>	A case where the setting is made as follows. • The transfer start trigger of DMAC0 is interrupt #include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function. void func(void) { uint16_t count; //Set up DMAC0 R_PG_DMAC_Set_C0(); //Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); //Wait for the transfer counter to become lower than 10 do{ R_PG_DMAC_GetTransferCount_C0(& count); } while(count >= 10); //Suspend the DMA transfer R_PG_DMAC_Suspend_C0(); }</project>	



5.8.8 R_PG_DMAC_SetTransferCount_C<channel number>

<pre>bool R_PG_DMAC_SetTransferCount_C<channel number="">(uint16_t count) < channel number > : 0 to 3</channel></pre>		
Set the transfer counter		
uint16_t count	Value to be set to the transfer counter	
true	Setting was made correctly	
 R_PG_DMAC_C<<i>channel number</i>>.c <i>channel number</i>>: 0 to 3 R_DMAC_Control This function sets the transfer counter. The valid range of the counter value is from 0 to 65535 (0 : free running mode) in normal 		
 The valid range of the counter value is from 0 to 65535 (0 : free running mode) in normal transfer mode, 0 to 1023 (0 = 1024 units) in repeat transfer mode and block transfer mode. A case where the setting is made as follows. IRQ0 interrupt was selected as a transfer start trigger of DMAC0 Dmac0IntFunc was specified as the DMA interrupt notification function name #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> void func(void) (//Set up DMAC0 R_PG_DMAC_Set_C0(); //Set IRQ0 R_PG_DMAC_Set_C0(); //Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); //DMA interrupt notification function void Dmac0IntFunc (void) ((/Suspend the DMA transfer R_PG_DMAC_Suspend_C0(); //Change the DMAC0 settings R_PG_DMAC_SetDestAddress_C0(dest_address); //Source address R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter // Make DMAC0 be ready for the transfer start trigger 		
	< channel number > : 0 Set the transfer counter uint16_t count true false R_PG_DMAC_C <chan <channel number="">: 0 : R_DMAC_Control This function sets the t The valid range of the transfer mode, 0 to 102 A case where the setting is IRQ0 interrupt was sel Dmac0IntFunc was sp #include "R_PG_default.h" void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0 //Set IRQ0 R_PG_ExtInterrupt_S // Make DMAC0 be ra R_PG_DMAC_Set_C0 //Set IRQ0 R_PG_DMAC_Activa } //DMA interrupt notification void Dmac0IntFunc (void) { //Suspend the DMA tra R_PG_DMAC_SetSra R_PG_DMAC_SetSra R_PG_DMAC_SetDe R_PG_DMAC_SetTra</channel></chan 	



5.8.9 R_PG_DMAC_GetRepeatBlockSizeCount_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_GetRepeatBlockSizeCount_C<channel number=""> (uint16_t * count) < channel number > : 0 to 3</channel></pre>	
Description	Get the repeat/block size counter value	
<u>Conditions for</u> output	Repeat transfer mode or block transfer mode is selected for the transfer mode.	
Parameter	uint16_t * count The address of storage area for the counter value	
Return value	true Acquisition succeeded	
	false Acquisition failed.	
File for output	R_PG_DMAC_C < <i>channel number</i> >.c < <i>channel number</i> >: 0 to 3	
RPDL function	R_DMAC_GetStatus	
<u>Details</u>	 This function gets the current repeat/block size counter value. The DMA interrupt request flag (IR flag) is cleared in this function. Call R_PG_DMAC_ClearInterruptFlag_C<<i>channel number></i> to get the DMA interrupt request flag before calling this function if needed. 	
Example	A case where the setting is made as follows.	
	• DMAC0 is set to repeat transfer mode	
	The transfer start trigger is interrupt	
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>	
	void func(void) {	
	uint16_t count;	
	//Set up DMAC0 R_PG_DMAC_Set_C0();	
	<pre>//Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0();</pre>	
	<pre>//Wait for the repeat size counter to become lower than 10 do{ R_PG_DMAC_GetRepeatBlockSizeCount_C0(& count); } while(count >= 10);</pre>	
	<pre>//Suspend the DMA transfer R_PG_DMAC_Suspend_C0(); }</pre>	



5.8.10 R_PG_DMAC_SetRepeatBlockSizeCount_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_SetRepeatBlockSizeCount_C<channel number=""> (uint16_t count) < channel number > : 0 to 3</channel></pre>			
Description	Set the repeat/block size counter value			
Conditions for output	Repeat transfer mode or	Repeat transfer mode or block transfer mode is selected for the transfer mode.		
Parameter_	uint16_t count	Value to be set to the repeat/block size counter		
Return value	true	Setting was made correctly		
	false	Setting failed		
File for output	R_PG_DMAC_C <char <channel number="">: 0 t</channel></char 			
RPDL function	R_DMAC_GetStatus			
<u>Details</u>	 This function sets the repeat/block size counter. The valid range of the counter value is from 0 to 1023 (0 = 1024 units) in repeat transfer mode, 1 to 1023 in block transfer mode. 			
<u>Example</u>	 A case where the setting is made as follows. DMAC0 is set to repeat transfer mode IRQ0 interrupt was selected as a transfer start trigger Dmac0IntFunc was specified as the DMA interrupt notification function name 			
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); //Set IRQ0 R_PG_ExtInterrupt_Set_IRQ0(); // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); } //DMA interrupt notification function void Dmac0IntFunc (void) { //Suspend the DMA transfer R_PG_DMAC_Suspend_C0(); //Change the DMAC0 settings R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter R_PG_DMAC_SetRepeatBlockSizeCount_C0(repeat_count); //Repeat size counter // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); } </project></pre>			



5.8.11 R_PG_DMAC_ClearInterruptFlag_C<channel number>

Definition	<pre>bool R_PG_DMAC_ClearInterruptFlag_C<channel number=""> (bool * int_request) < channel number > : 0 to 3</channel></pre>	
Description	Get and clear the interrupt request flag	
Conditions for	DMA interrupt is enable	d
<u>output</u>		
Parameter	bool * int_request	The address of storage area for the interrupt request flag
Return value	true	Acquisition and clearing succeeded
	false	Acquisition and clearing failed
File for output	R_PG_DMAC_C < <i>char</i>	inel number>.c
	<channel number="">: 0</channel>	to 3
RPDL function	R_DMAC_GetStatus	
<u>Details</u>	• This function gets and	clears the DMA interrupt request flag (IR flag).
Example	 R_DMAC_GetStatus This function gets and clears the DMA interrupt request flag (IR flag). A case where the setting is made as follows. DMAC0 is set to mormal transfer mode The transfer start trigger is interrupt The DMA interrupt is enabled The DMA interrupt priority level is 0 #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> void func(void) tool int_request; //Set up DMAC0 R_PG_DMAC_Set_CO(); //Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_CO(); //Wait for the IR flag to become 1 do{ R_PG_DMAC_ClearInterruptFlag_C0(& int_request); while(int_request == false); 	



5.8.12 R_PG_DMAC_GetTransferEndFlag_C<channel number>

Definition	<pre>bool R_PG_DMAC_GetTransferEndFlag_C<channel number=""> (bool* end) < channel number > : 0 to 3</channel></pre>			
Description	Get the transfer end flag	Get the transfer end flag		
Parameter	bool* end	The address of storage area for the transfer end flag		
Return value	true	Acquisition succeeded		
	false	Acquisition failed.		
File for output	R_PG_DMAC_C <char <channel number="">: 0</channel></char 			
RPDL function	R_DMAC_GetStatus			
<u>Details</u>	 This function gets the transfer end flag. The DMA interrupt request flag (IR flag) is cleared in this function. Call R_PG_DMAC_ClearInterruptFlag_C<<i>channel number></i> to get the DMA interrupt request flag before calling this function if needed. The transfer end flag is not cleared in this function. Call R_PG_DMAC_ClearTransferEndFlag_C<<i>channel number></i> to clear the transfer end flag if needed. 			
Example	A case where the setting is made as follows. • DMAC0 is set to normal transfer mode • The transfer start trigger is interrupt • The DMA interrupt is not enabled #include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function. void func(void) { bool end; //Set up DMAC0 R_PG_DMAC_Set_C0(); //Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); //Wait for the transfer end flag to become 1 do{ R_PG_DMAC_GetTransferEndFlag_C0(& end); } while(end == false); //Clear the DMA transfer end flag R_PG_DMAC_ClearTransferEndFlag_C0(); }</project>			



5.8.13 R_PG_DMAC_ClearTransferEndFlag_C<channel number>

Definition	bool R_PG_DMAC_ClearTransferEndFlag_C <channel number=""> (void)</channel>			
	< channel number > : (0 to 3		
Description	Clear the transfer end fla	Clear the transfer end flag		
Parameter	None			
Return value	true	Clearing succeeded		
	false	Clearing failed		
File for output	R_PG_DMAC_C < <i>chan</i> < <i>channel number</i> >: 0			
RPDL function	R_DMAC_Control			
<u>Details</u>	• This function clears th	e transfer end flag.		
	• To get the transfer end flag, call R_PG_DMAC_GetTransferEndFlag_C< <i>channel number</i> >.			
Example	A case where the setting is made as follows.			
	• DMAC0 is set to morr	nal transfer mode		
	• The transfer start trigg	er is interrupt		
	• The DMA interrupt is not enabled			
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>			
	void func(void)			
	{ bool end;			
	//Set up DMAC0 R_PG_DMAC_Set_	_CO();		
	<pre>//Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); //Wait for the transfer end flag to become 1 do{ R_PG_DMAC_GetTransferEndFlag_C0(& end); } while(end == false); //Clear the DMA transfer end flag R_PG_DMAC_ClearTransferEndFlag_C0(); }</pre>			



5.8.14 R_PG_DMAC_GetTransferEscapeEndFlag_C<channel number>

<u>Definition</u>	bool R_PG_DMAC_GetTransferEscapeEndFlag_C <channel number=""> (bool* end) < channel number > : 0 to 3</channel>		
Description	Get the transfer escape end flag		
<u>Conditions for</u> output	[Completion of a 1-block/repeat size transfer], [Source address extended repeat area overflow] or [Destination address extended repeat area overflow] is selected as the interrupt output source		
Parameter	bool* end	The address of storage area for the transfer escape end flag	
<u>Return value</u>	true false	Acquisition succeeded Acquisition failed.	
File for output	R_PG_DMAC_C <chan <channel number="">: 0</channel></chan 		
RPDL function	R_DMAC_GetStatus		
<u>Details</u>	 This function gets the DMA transfer escape end flag (EDMSTS.ESIF). The DMA interrupt request flag (IR flag) is cleared in this function. Call R_PG_DMAC_ClearInterruptFlag_C<<i>channel number></i> to get the DMA interrupt request flag before calling this function if needed. The transfer escape end flag is not cleared in this function. Call R_PG_DMAC_ClearTransferEscapeEndFlag_C<<i>channel number></i> to clear the transfer escape end flag if needed. 		
<u>Example</u>	A case where the setting is made as follows. DMAC0 is set to repeat transfer mode The transfer start trigger is interrupt [Completion of a 1-block/repeat size transfer] is selected for the interrupt output source The DMA interrupt priority level is 0 #include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function. void func(void) { bool end; //Set up DMAC0 R_PG_DMAC_Set_C0(); //Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); //Wait for the transfer escape end flag to become 1 do{ R_PG_DMAC_GetTransferEscapeEndFlag_C0(& end); } while(end == false); //Clear the DMA transfer escape end flag R_PG_DMAC_ClearTransferEscapeEndFlag_C0(); }</project>		

RENESAS

5.8.15 R_PG_DMAC_ClearTransferEscapeEndFlag_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_ClearTransferEscapeEndFlag_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Clear the transfer escape end flag		
<u>Conditions for</u> output	[Completion of a 1-block/repeat size transfer], [Source address extended repeat area overflow] or [Destination address extended repeat area overflow] is selected as the interrupt output source		
Parameter	None		
<u>Return value</u>	trueClearing succeededfalseClearing failed		
File for output	R_PG_DMAC_C < <i>channel number</i> >.c < <i>channel number</i> >: 0 to 3		
RPDL function	R_DMAC_Control		
<u>Details</u>	 This function clears the transfer escape end flag. To get the transfer escape end flag, call R_PG_DMAC_GetTransferEscapeEndFlag_C<<i>channel number</i>>. 		
Example			



5.8.16 R_PG_DMAC_SetSrcAddress_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_SetSrcAddress_C<channel number="">(void * src_addr) < channel number > : 0 to 3</channel></pre>			
Description	Set the source address			
Parameter	void * src_addr	The source address to be set		
Return value	true	Setting was made correctly		
	false	Setting failed.		
File for output		R_PG_DMAC_C <channel number="">.c <channel number="">: 0 to 3</channel></channel>		
<u>RPDL function</u> <u>Details</u>	R_DMAC_ControlThis function sets the s	R_DMAC_ControlThis function sets the source address.		
Example	A case where the setting is	s made as follows.		
		ected as a transfer start trigger of DMAC0		
	 Dmac0IntFunc was specified as the DMA interrupt notification function name #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); } } 			
	//Set IRQ0 R_PG_ExtInterrupt_Set_IRQ0();			
	<pre>// Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); } //DMA interrupt notification function void Dmac0IntFunc (void)</pre>			
	{ //Suspend the DMA transfer R_PG_DMAC_Suspend_C0();			
	<pre>//Change the DMAC0 settings R_PG_DMAC_SetSrcAddress_C0(src_address); //Source address R_PG_DMAC_SetDestAddress_C0(dest_address); //Destination address R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); }</pre>			



5.8.17 R_PG_DMAC_SetDestAddress_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_SetDestAddress_C<channel number="">(void * dest_addr) < channel number > : 0 to 3</channel></pre>		
Description	Set the source address		
Parameter	void * dest_addr	The destination address to be set	
Return value	true false	Setting was made correctly Setting failed.	
File for output	R_PG_DMAC_C< <i>channel number</i> >.c < <i>channel number</i> >: 0 to 3		
RPDL function Details	R_DMAC_ControlThis function sets the destination address.		
Example	A case where the setting is made as follows. • IRQ0 interrupt was selected as a transfer start trigger of DMAC0 • Dmac0IntFunc was specified as the DMA interrupt notification function name #include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function. void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); //Set IRQ0 R_PG_ExtInterrupt_Set_IRQ0(); // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); } //DMA interrupt notification function void Dmac0IntFunc (void) { //Suspend the DMA transfer R_PG_DMAC_Suspend_C0(); //Set up the DMAC and continue R_PG_DMAC_SetSrcAddress_C0(src_address); //Source address R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0();</project>		



5.8.18 R_PG_DMAC_SetAddressOffset_C<channel number>

Definition	<pre>bool R_PG_DMAC_SetAddressOffset_C<channel number="">(int32_t offset) < channel number > : 0 to 3</channel></pre>		
Description Conditions for output	Set the address offset [Offset addition] is selected for [Source address update mode] or [Destination address update mode].		
Parameter	int32_t offset	The offset value to be set	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_DMAC_C <chan <channel number="">: 0</channel></chan 		
RPDL function	R_DMAC_Control		
Details	 This function sets the a The range of the addression 	address offset. ess offset value is from +FFFFFh to -1000000h.	
English	_		
<u>Example</u>	 A case where the setting is made as follows. IRQ0 interrupt was selected as a transfer start trigger of DMAC0 Dmac0IntFunc was specified as the DMA interrupt notification function name [Offset addition] is selected. #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); //Set up DMAC0 		
	//Set IRQ0 R_PG_ExtInterrupt_S	Set_IRQ0();	
	<pre>// Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); }</pre>		
	//DMA interrupt notification function void Dmac0IntFunc (void) {		
	//Suspend the DMA transfer R_PG_DMAC_Suspend_C0();		
	<pre>//Set up the DMAC and continue R_PG_DMAC_SetSrcAddress_C0(src_address); //Source address R_PG_DMAC_SetDestAddress_C0(dest_address); //Destination address R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter R_PG_DMAC_SetAddressOffset_C0(offset); //Address offset // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); }</pre>		



5.8.19 R_PG_DMAC_SetExtendedRepeatSrc_C<channel number>

Definition	<pre>bool R_PG_DMAC_SetExtendedRepeatSrc_C<channel number="">(uint32_t area) < channel number > : 0 to 3</channel></pre>		
Description	Set the source address extended repeat value		
Conditions for	An extended repeat area	is specified for the transfer source.	
<u>output</u>			
Parameter	uint32_t area	The source address extended repeat value to be set	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_DMAC_C <channel number="">.c <channel number="">: 0 to 3</channel></channel>		
RPDL function	R_DMAC_Control		
Details		ource address extended repeat value.	
	• The value can be any p	power of 2, from 2^1 to 2^{27} .	
Example	A case where the setting is	s made as follows.	
		ected as a transfer start trigger of DMAC0	
	 Dmac0IntFunc was specified as the DMA interrupt notification function name An extended repeat area is specified for the transfer source and destination. 		
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func(void)</project></pre>		
	{		
	//Set up DMAC0 R_PG_DMAC_Set_C0();		
	//Set IRQ0		
R_PG_ExtInterrupt_Set_IRQ0(); // Make DMAC0 be ready for the transfer start trigger		et_IRQ0();	
	R_PG_DMAC_Activate_C0(); }		
	//DMA interrupt notification function void Dmac0IntFunc (void) {		
	//Suspend the DMA transfer R_PG_DMAC_Suspend_C0();		
	<pre>//Change the DMAC0 settings R_PG_DMAC_SetSrcAddress_C0(src_address); //Source address R_PG_DMAC_SetDestAddress_C0(dest_address); //Destination address R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter R_PG_DMAC_SetExtendedRepeatSrc_C0(src_repeat); //Source extended repeat size R_PG_DMAC_SetExtendedRepeatDest_C0(dest_repeat); //Destination extended repeat size // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); </pre>		
	}		



5.8.20 R_PG_DMAC_SetExtendedRepeatDest_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_SetExtendedRepeatDest_C<channel number="">(uint32_t area) < channel number > : 0 to 3</channel></pre>			
Description Conditions for output	Set the destination address extended repeat value An extended repeat area is specified for the transfer destination.			
Parameter	uint32_t area	The destination address extended repeat value to be set		
<u>Return value</u>	true false	Setting was made correctly Setting failed		
File for output	R_PG_DMAC_C <channel number="">.c <channel number="">: 0 to 3</channel></channel>			
<u>RPDL function</u> <u>Details</u>		 R_DMAC_Control This function sets the destination address extended repeat value. The value can be any power of 2, from 2¹ to 2²⁷. 		
<u>Example</u>	 A case where the setting is made as follows. IRQ0 interrupt was selected as a transfer start trigger of DMAC0 Dmac0IntFunc was specified as the DMA interrupt notification function name An extended repeat area is specified for the transfer source and destination. 			
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); //Set IRQ0 R_PG_ExtInterrupt_Set_IRQ0(); // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); } //DMA interrupt notification function void Dmac0IntFunc (void) { //Suspend the DMA transfer R_PG_DMAC_SetSrcAddress_C0(src_address); //Source address R_PG_DMAC_SetDestAddress_C0(dest_address); //Destination address R_PG_DMAC_SetTransferCount_C0(tr_count); //Transfer counter R_PG_DMAC_SetExtendedRepeatSrc_C0(dest_repeat); //Source extended repeat size R_PG_DMAC_SetExtendedRepeatDest_C0(dest_repeat); //Destination extended repeat size // Make DMAC0 be ready for the transfer start trigger R_PG_DMAC_Activate_C0(); </project></pre>			

5.8.21 R_PG_DMAC_StopModule_C<channel number>

<u>Definition</u>	<pre>bool R_PG_DMAC_StopModule_C<channel number=""> (void) < channel number > : 0 to 3</channel></pre>		
Description	Stop the DMAC channel		
Parameter	None		
Return value	true	Stopping succeeded.	
	false	Stopping failed.	
File for output	R_PG_DMAC_C< <i>channel number</i> >.c < <i>channel number</i> >: 0 to 3		
RPDL function	R_DMAC_Destroy		
<u>Details</u>	 Stops the DMAC channel. If all DMAC channels and DTC are stopped, DMAC and DTC shall be module-stop state. If another peripheral is being used to trigger a DMA transfer, stop the trigger sources before calling this function. 		
<u>Example</u>	A case where the setting is made as follows. • The software trigger was selected as a transfer start trigger of DMAC0 • Dmac0IntFunc was specified as the DMA interrupt notification function name #include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function. void func(void) { //Set up DMAC0 R_PG_DMAC_Set_C0(); //Start the DMA transfer of DMAC0 R_PG_DMAC_StartTransfer_C0(); //DMA interrupt notification function void Dmac0IntFunc (void) { //Stop the DMAC0 R_PG_DMAC_StopModule_C0(); }</project>		



5.9 Data Transfer Controller (DTCa)

5.9.1 R_PG_DTC_Set

Definition	bool R_PG_DTC_Set (void)		
Description	Set the common options for DTC		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_DTC.c		
RPDL function	R_DTC_Set		
<u>Details</u>	Releases DTC and	DMAC from the module-stop state.	
	• Before calling oth	er functions of DTC, call this function.	
	• This function conf	igures the read skip control, address mode and the DTC vector table	
base address.			
Example	 A case where the setting is made as follows. The DTC vector table address has been set to 2000h. The transfer setting of which the transfer start trigger is IRQ0 has been made. //Include "R_PG_<project name="">.h" to use this function.</project> #include "R_PG_default.h" //DTC vector table #pragma address dtc_vector_table = 0x00002000 uint32_t dtc_vector_table [256]; //Set up the DTC void func(void) 		
{ // Set the common options for R_PG_DTC_Set();		-	
	//Make the transfer setting of which the transfer start trigger is IRQ0 R_PG_DTC_Set_IRQ0();		
	<pre>//Make DTC be ready for the transfer start trigger R_PG_DTC_Activate();</pre>		
<pre>//Set up IRQ0 R_PG_ExtInterrupt_Set_IRQ0(); }</pre>		rupt_Set_IRQ0();	



5.9.2 R_PG_DTC_Set_<trigger source>

Definition

bool R_PG_DTC_Set_<trigger source> (void)

	< trigger source >	
	SWINT	Software interrupt
	CMI0 to 3	CMT0 to 3 compare match interrupt
	SPRI0	RSPI0 receive interrupt
	SPTI0	RSPI0 transmit interrupt
	IRQ0 to 7	External interrupts
	S12ADI0	A/D scan end interrupt
	GBADI	Group B scan end interrupt
	ELSR18I	ELC interrupt
	TGIA0 to D0	MTU0 input capture/compare match A to D interrupt
	TGIA1 or B1	MTU1 input capture/compare match A or B interrupt
	TGIA2 or B2	MTU2 input capture/compare match A or B interrupt
	TGIA3 to D3	MTU3 input capture/compare match A to D interrupt
	TGIA4 to D4	MTU4 input capture/compare match A to D interrupt
	TCIV4	MTU4 overflow/underflow interrupt
	TGIU5 to W5	MTU5 input capture/compare match U to W interrupt
	CMIA0 or B0	TMR0 compare match A or B interrupt
	CMIA1 or B1	TMR1 compare match A or B interrupt
	CMIA2 or B2	TMR2 compare match A or B interrupt
	CMIA3 or B3	TMR3 compare match A or B interrupt
	DMAC0I to 3I	DMACA0 to 3 interrupt
	RXI1, 5, 6, 9 and	SCI1, 5, 6, 9, 12 receive data full interrupt
	12	
	TXI1, 5, 6, 9 and	SCI1, 5, 6, 9, 12 transmit data empty interrupt
	12	
	ICRXI0	RIIC0 receive data full interrupt
	ICTXI0	RIIC0 transmit data empty interrupt
Description	Set the DTC transfer	data
Parameter	None	
Return value	true	Setting was made correctly

R_PG_DTC.c File for output

false

R_DTC_Create

RPDL function

Details

Store the transfer data that will be triggered by transfer start trigger in specified address.

The transfer data of the chain transfer will also be stored.

- If other transfer data has already been stored in the specified address, new data will be • overwritten.
- This function does not set any interrupts used for transfer start triggers. Set up interrupts

RENESAS

Setting failed

by each peripheral function.

- Select DTC as the request destination of interrupts used for the transfer start trigger.
- Call this function before configuring the peripherals that will be involved in the data transfer.

Example

A case where the setting is made as follows.

- The DTC vector table address has been set to 2000h.
- The transfer setting of which the transfer start trigger is IRQ0 has been made.
 - The transfer setting of which the transfer start trigger is IRQ1 has been made.

er is IRQ



5.9.3 R_PG_DTC_Activate

lling this		
lling this		
10		
le. een selected		
 "Request is transferred to CPU when specified transfer is completed" has been selected in the interrupt setting. The chain transfer has been disabled. Irq0IntFunc has been specified as an IRQ0 interrupt notification function name. 		
t.)		
a		



5.9.4 R_PG_DTC_SuspendTransfer

Definition	bool R_PG_DTC_SuspendTransfer (void)		
Description	Stop the data transfer		
Parameter	None		
Return value	true	Stopping succeeded	
	false	Stopping failed	
File for output	R_PG_DTC.c		
RPDL function	R_DTC_Control		
Details	• Stops the data transfer.		
	• If transfer is stopp	ed during data transfer, the accepted start request is active until the	
	processing is comp	pleted.	
	• Call R_DTC_Acti	vate to enable the transfer.	
<u>Example</u>		ing is made as follows.	
		able address has been set to 2000h. g of which the transfer start trigger is IRQ0 has been made.	
	//Include "R_PG_ <project name="">.h" to use this function.</project>		
	<pre>#include "R_PG_default.h" //DTC vector table #pragma address dtc_vector_table = 0x00002000 uint32_t dtc_vector_table [256]; //Set up the DTC void func(void) </pre>		
	{ // Set the commo R_PG_DTC_Se	on options for DTC t();	
	//Make the trans R_PG_DTC_Se	fer setting of which the transfer start trigger is IRQ0 t_IRQ0();	
	<pre>//Make DTC be ready for the transfer start trigger R_PG_DTC_Activate();</pre>		
<pre>//Set up IRQ0 R_PG_ExtInterrupt_Set_IRQ0();</pre>		upt_Set_IRQ0();	
	//Suspend the DTC transfer void func2(void)		
	<pre>{ R_PG_DTC_SuspendTransfer(); }</pre>		
	//Resume the DTC travoid func3(void)	ansfer	
	{ R_PG_DTC_Ac	tivate();	
	,		



5.9.5 R_PG_DTC_GetTransmitStatus

Definition	<pre>bool R_PG_DTC_GetTransmitStatus (uint8_t * vector, bool * active)</pre>	
Description	Get transfer status	
Parameter	uint8_t * vector	The address of storage area for the vector number of current data transfer (Valid when "* active" is 1)
	bool * active	The address of storage area for the progress flag. If this value is 1, the data transfer is processed.
Return value	true	Acquisition succeeded
	false	Acquisition failed
File for output	R_PG_DTC.c	
RPDL function	R_DTC_GetStatus	
Details	This function acqu	ires the active flag and the vector number of the current data transfer.
Example	 This function acquires the active flag and the vector number of the current data transfer. //Include "R_PG_<project name="">.h" to use this function.</project> #include "R_PG_default.h" uint8_t vector; bool active; void func(void)	

<u>Definition</u> bool R_PG_DTC_GetTransmitStatus (uint8_t * vector, bool * active)



5.9.6 R_PG_DTC_StopModule

Definition	bool R_PG_DTC_StopModule (void)		
Description	Shut down the DTC		
Parameter	None		
Return value	true	Shutting down succeeded	
	false	Shutting down failed	
File for output	R_PG_DTC.c		
RPDL function	R_DTC_Destroy		
<u>Details</u>	• Disable the interru	s down the DTC and places it in the module-stop state. upt used for transfer start trigger before calling this function. also shut down the DMAC.	
<u>Example</u>	 This function will also shut down the DMAC. A case where the setting is made as follows. The DTC vector table address has been set to 2000h. The transfer setting of which the transfer start trigger is IRQ0 has been made. The transfer setting of which the transfer start trigger is IRQ1 has been made. //Include "R_PG_<project name="">.h" to use this function.</project> #include "R_PG_default.h" 		
	<pre>//DTC vector table #pragma address dtc_ uint32_t dtc_vector_</pre>	_vector_table = 0x00002000 table [256];	
	<pre>void func(void) { // Set the common options for DTC R_PG_DTC_Set();</pre>		
	<pre>//Make the transfer setting of which the transfer start trigger is IRQ0 R_PG_DTC_Set_IRQ0();</pre>		
	//Make the trans R_PG_DTC_Se	sfer setting of which the transfer start trigger is IRQ1 t_IRQ1();	
	//Make DTC be R_PG_DTC_A	ready for the transfer start trigger ctivate();	
		nd IRQ1 rupt_Set_IRQ0(); rupt_Set_IRQ1();	
		rupt_Disable_IRQ0(); rupt_Disable_IRQ1(); DTC	



5.10 Event Link Controller (ELC)

5.10.1 R_PG_ELC_Set

Definition	bool R_PG_ELC_Set (void)			
Description	Sets the ELC			
Parameter	None			
Return value	true	Setting was made correctly.		
	false	Setting failed.		
File for output	R_PG_ELC.c	R_PG_ELC.c		
RPDL function	R_ELC_Create			
Details	Releases the ELC from the module-stop state.			
	 been specified, that for the CPU. The ir void <name li="" of="" the<=""> For notes on interrutive provided at the end </name>	with interrupt 1 has been set and an interrupt notification function has function is called in response to the generation of an interrupt request aterrupt notification function must be in the following format: the interrupt notification function> (void) upt notification functions, refer to "Notes on Notification Functions" of this section. be called before any other ELC functions.		
Example	ample The following settings have been made through the GUI.Set an event link with interrupt 1.			
*				
	Specify Elc1IntFun	c as the interrupt notification function.		
	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>			
	<pre>void func(void) { // Set up the event link controller (ELC). R_PG_ELC_Set(); // Set an event link. R_PG_ELC_SetLink_Interrupt1(); // Enable all event links. R_PG_ELC_AllEventLinkEnable(); } // Interrupt notification function void Elc1IntFunc(void) { // Interrupt handling</pre>			
	// Disable the eve R_PG_ELC_Disa }	nt link. ableLink_Interrupt1();		



5.10.2 R_PG_ELC_SetLink_<peripheral module>

D	efir	niti	on

bool R_PG_ELC_SetLink_peripheral module> (void)

	<peripheral module=""></peripheral>		
	MTU1 to 4	Multi-function timer pulse unit 2 (MTU2a) channel 1 to 4	
	TMR0 or 2	8-bit timer (TMR) channel 0 or 2	
	ADC12	12-bit A/D converter (S12ADb)	
	Interrupt1	Interrupt 1	
	Output_Group1	Output port group 1	
	Input_Group1	Input port group 1	
	SinglePort0 or 1	Single port 0 or 1	
Description	Sets an event link	Sets an event link	
<u>Conditions for</u> output	When a module to receive an event is specified, functions for that module are generated.		
Parameter	None	None	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_ELC.c		
RPDL function	R_ELC_Control		
Details	 Sets a link between an event and the module receiving the event signal. Specifies the action to be taken by the module on receiving the event signal. 		
	Call R_PG_ELC_AllEventLinkEnable to enable the event links.		
	• If you have selected the count start/event counter as [Operation when event i		
	TMR0/TMR2, call this function and then R_PG_Timer_Start_TMR_U <unit< td=""></unit<>		
<i>number>_C<channel number=""></channel></i> before enabling the event links		<i>nel number></i> before enabling the event links that have been set.	
Example	Refer to the example of R_PG_ELC_Set.		



5.10.3 R_PG_ELC_DisableLink_<peripheral module>

|--|

bool R_PG_ELC_DisableLink_peripheral module> (void)

	<pre><peripheral module=""></peripheral></pre>	
	MTU1 to 4	Multi-function timer pulse unit 2 (MTU2a) channel 1 to 4
	TMR0 or 2	8-bit timer (TMR) channel 0 or 2
	ADC12	12-bit A/D converter (S12ADb)
	Interrupt1	Interrupt 1
	Output_Group1	Output port group 1
	Input_Group1	Input port group 1
	SinglePort0 or 1	Single port 0 or 1
Description	Disables an event link	
Conditions for	When a module to receive an event is specified, functions for that module are generated.	
<u>output</u>		
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_ELC.c	
RPDL function	R_ELC_Control	
Details	Disables an event link that has been set.	
Example	Refer to the example of R_PG_ELC_Set.	



5.10.4 R_PG_ELC_Set_PortGroup<port group number>

Definition	<pre>bool R_PG_ELC_Set_PortGroup<port group="" number=""> (void)</port></pre>		
	<pre><port group="" number="">: 1</port></pre>		
Description	Sets a port group		
Conditions for		g been selected for [Include in port group] under [Output port group	
<u>output</u>	and Input port group]		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_ELC.c		
RPDL function	R_ELC_Control		
Details		rough the GUI will compose port group 1 (port B).	
		as for the port group.	
<u>Example</u>	The following setting	s have been made through the GUI.	
	PB0 to PB3 are sele	ected for input during the process of setting I/O ports.	
	The following item	is selected as the event signal for input port group 1:	
	[Software event signal]		
• The following item is selected as the action to be taken on input of the eve		is selected as the action to be taken on input of the event signal for	
	input port group 1:		
	[Transfer the signal value of the external pin to PDBFn]		
	• Input port group 1 includes the following bits:		
	[PB0 to PB3]		
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t pdbf1_val; // Destination for storage of the value of port buffer register 1		
	void func1(void)		
	{ R_PG_IO_PORT_Set_PB0(); // Set an I/O port pin (PB0). R_PG_IO_PORT_Set_PB1(); // Set an I/O port pin (PB1). R_PG_IO_PORT_Set_PB2(); // Set an I/O port pin (PB2). R_PG_IO_PORT_Set_PB3(); // Set an I/O port pin (PB3).		
	R_PG_ELC_Set(); // Set up the event link controller (ELC). R_PG_ELC_SetLink_Input_Group1(); // Set an event link. R_PG_ELC_Set_PortGroup1(); // Set the port group. R_PG_ELC_AllEventLinkEnable(); // Enable all event links. R_PG_ELC_Generate_SoftwareEvent(); // Generate a software event. }		
	void func2(void)		
	<pre>{ R_PG_ELC_GetPortBufferValue_Group1(&pdbf1_val); // Acquire the value of the // port buffer register.</pre>		
	R_PG_ELC_StopModule(); // Stop the ELC }		

5.10.5 R_PG_ELC_Set_SinglePort<single-port number>

Definition	<pre>bool R_PG_ELC_Set_SinglePort<single-port number=""> (void) <single-port number="">: 0 or 1</single-port></single-port></pre>		
Description	Sets a single-port pin		
<u>Conditions for</u> output	The single-port setting satisfies the following conditions:1. Any of port pins 0 or 1 is selected during the process of setting port pins.2. Event conditions can be selected.		
	Note: This function is	not usable when the port pins have been selected for output.	
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_ELC.c		
RPDL function	R_ELC_Control		
<u>Details</u>	This function sets the	fies the bit selected through the GUI as a single-port pin. he event condition for the single-port pin.	
•	This function is only usable when the port pins have been selected for input.		
Example	The following settings have been made through the GUI.		
	 PB0 is selected for input during the process of setting I/O port pins. The following item is selected as the event signal for interrupt 1: [Input edge detection signal of single input port 0] [CPU or CPU (After activating DMAC)] is selected as the action to be taken on input of the event signal for interrupt 1. [PB0] is selected as [Port settings] for single port pin 0. [Detect the falling edge] is selected as [Event generation condition] for single port pin 0. // Include "R_PG_<project name="">.h" to use this function.</project> 		
<pre>#include "R_PG_default.h" uint8_t elc1int_count=0; // Number of times interrupts are generated</pre>		ult.h"	
	<pre>void func(void) { R_PG_IO_PORT_Set_PB0(); // Set individual I/O port pins. R_PG_ELC_Set(); // Set up the event link controller (ELC). R_PG_ELC_SetLink_Interrupt1(); // Set an event link. R_PG_ELC_Set_SinglePort0(); // Set the single-port pin. R_PG_ELC_AllEventLinkEnable(); // Enable all event links. } // Interrupt notification function void Elc1IntFunc(void) { elc1int_count++; }</pre>		

RENESAS

5.10.6 R_PG_ELC_AllEventLinkEnable

Definition	bool R_PG_ELC_AllEventLinkEnable (void)	
Description	Enables all event links	
Conditions for	None	
<u>output</u>		
Return value	true	All event links were successfully enabled.
	false	Enabling of all event links failed.
File for output	R_PG_ELC.c	
RPDL function	R_ELC_Control	
Details	This function enables all event links that have been made.	
Example	Refer to the example of R_PG_ELC_Set.	



5.10.7 R_PG_ELC_AllEventLinkDisable

Definition	bool R_PG_ELC_AllEventLinkDisable (void)	
Description	Disables all event links	
Parameter	None	
Return value	true	All event links were successfully disabled.
	false	Disabling of all event links failed.
File for output	R_PG_ELC.c	
RPDL function	R_ELC_Control	
Details	This function disables all event links that have been made.	
Example	Refer to the example of R_PG_ELC_Set_SinglePort< <i>single-port number</i> >.	



5.10.8 R_PG_ELC_Generate_SoftwareEvent

<u>Definition</u>	bool R_PG_ELC_Generate_SoftwareEvent (void)		
Description	Generates a software	Generates a software event	
Conditions for	A software event sign	nal is selected as the event signal.	
<u>output</u>			
Parameter	None		
Return value	true A software event was successfully generated.		
	false	Generation of a software event signal failed.	
File for output	R_PG_ELC.c		
RPDL function	R_ELC_Control		
Details	This function generates a software event.		
Example	Refer to the example of R_PG_ELC_Set.		



5.10.9 R_PG_ELC_GetPortBufferValue_Group<port-group number>

Definition	bool R_PG_ELC_GetPortBufferValue_Group <port-group number=""> (uint8_t * reg_val)</port-group>		
	<pre><port-group number="">: 1</port-group></pre>		
Description	Acquires the value of a port buffer register		
Conditions for	Any of the bits having been selected for [Include in port group] under [Output port group n		
<u>output</u>	and Input port group n].		
	n: 1		
Parameter	uint8_t * reg_val	Destination for storage of the value of the port buffer register	
Return value	true	The value was successfully acquired.	
	false	Acquisition failed.	
File for output	R_PG_ELC.c		
RPDL function	R_ELC_Read		
Details	This function acquires the value of the port buffer register.		
	When <i><port-group number=""></port-group></i> is 1:		
	The value of PDBF1 (port buffer register 1) is acquired.		
Example	Refer to the example of R_PG_ELC_Set_PortGroup <port-group number="">.</port-group>		



5.10.10 R_PG_ELC_SetPortBufferValue_Group<port-group number>

Definition	bool R_PG_ELC_SetPortBufferValue_Group <port-group number=""> (uint8_t reg_val)</port-group>		
Description	<pre><port-group number="">: 1 Sets a value for a port buffer register</port-group></pre>		
Conditions for	Any of the bits having been selected for [Include in port group] under [Output port group n		
<u>output</u>	and Input port group n].		
oupur	n: 1		
Parameter	uint8_t reg_val	Value to be set for the port buffer register	
Return value	True	Setting was made correctly.	
	False	Setting failed.	
File for output	R_PG_ELC.c		
RPDL function	R_ELC_Write		
Details •	This function sets a value for the port buffer register.		
	When <i><port-group number=""></port-group></i> is 1:		
	The value is set in PDBF1 (port buffer register 1).		
Example	The following settings have been made through the GUI.		
	PB4 to PB7 are selected for output during the process of setting I/O ports.		
	 The following item is selected as an event signal for output port group 1: [Software event signal] The following item is selected as an action to be taken on input of the event signal for output port group 1: [Output the buffer value] 		
	 Output port group 1 includes the following bits: [PB4 to PB7] // Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { uint8_t pdbf1_val = 0xf0; // The value to be set for port buffer register 1</project> 		
	R_PG_IO_PORT_Set_PB4(); // Set an I/O port pin (PB4). R_PG_IO_PORT_Set_PB5(); // Set an I/O port pin (PB5). R_PG_IO_PORT_Set_PB6(); // Set an I/O port pin (PB6). R_PG_IO_PORT_Set_PB7(); // Set an I/O port pin (PB7).		
	R_PG_ELC_Set(); // Set up the event link controller (ELC). R_PG_ELC_SetLink_Output_Group1(); // Set an event link. R_PG_ELC_Set_PortGroup1(); // Set the port group. R_PG_ELC_SetPortBufferValue_Group1(pdbf1_val); // Set the value for the port // buffer register. R_PG_ELC_AllEventLinkEnable(); // Enable all event links.		
	R_PG_ELC_Generate_SoftwareEvent(); // Generate a software event.		


5.10.11 R_PG_ELC_StopModule

Definition	bool R_PG_ELC_StopModule (void)	
Description	Stops the ELC	
Parameter_	None	
Return value	true	The ELC was successfully stopped.
	false	Stopping the ELC failed.
File for output	R_PG_ELC.c	
RPDL function	R_ELC_Destroy	
Details	• This function disables all event links and places the ELC in the module-stop state.	
Example	Refer to the example of R_PG_ELC_Set_PortGroup <port-group number="">.</port-group>	



5.11 I/O Ports

5.11.1 R_PG_IO_PORT_Set_P<port number>

<u>Definition</u>	bool R_PG_IO_PORT_Set_P< <i>port number</i> > (void) < <i>port number</i> >: 0 to 5, A to E, H and J	
Description	Set up the I/O port	
Conditions for output	When [Used as an I/O port] of one or more pins are specified in the port in the GUI. However, when only P35 is specified in the PORT3, R_PG_IO_PORT_Set_P3 is not generated.	
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_IO_PORT_P <p< td=""><td>ort number>.c</td></p<>	ort number>.c
	<i><port number=""></port></i> : 0 to 5	5, A to E, H and J
RPDL function	R_IO_PORT_Set	
<u>Details</u>	• Selects the direction (input or output), input pull-up resistor, output type, and drive capacity for pins for which [Used as an I/O port] was specified in the GUI.	
	• This function is used to set all pins for which [Used as I/O port] has been selected in a	
When set as an output port (high-drive output), all bits change to normal		
English	entering deep softwar	-
<u>Example</u>	//Include "R_PG_ <project name="">.h" to use this function.</project>	
	#include "R_PG_default.h"	
	void func(void)	
	{	
	//Handle unavailat	ole pins
	R_PG_IO_PORT_	_SetPortNotAvailable();
	//Set P0.	
	R_PG_IO_PORT_	_Set_P0();
	}	



5.11.2 R_PG_IO_PORT_Set_P<port number><pin number>

<u>Definition</u>	<pre>bool R_PG_IO_PORT_Set_P<port number=""><pin number=""> (void) <port number="">: 0 to 5, A to E, H and J <pin number="">: 0 to 7</pin></port></pin></port></pre>	
Description	Set up the I/O port pin	
<u>Conditions for</u> output Parameter	When [Used as an I/O port] is specified in the GUI. However, R_PG_IO_PORT_Set_P35 is not generated. None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_IO_PORT_P< <i>port number</i> >.c < <i>port number</i> >: 0 to 5, A to E, H and J	
RPDL function	R_IO_PORT_Set	
<u>Details</u>	 Selects the direction (input or output), input pull-up resistor, output type, and drive capacity for pins for which [Used as an I/O port] was specified in the GUI. The setting only applies to one pin. 	
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Set P03. R_PG_IO_PORT_Set_P03(); //Set P05. R_PG_IO_PORT_Set_P05(); }</project></pre>	



5.11.3 R_PG_IO_PORT_Read_P<port number>

<u>Definition</u>	<pre>bool R_PG_IO_PORT_Read_P<port number=""> (uint8_t * data) <port number="">: 0 to 5, A to E, H and J</port></port></pre>	
Description	Read data from Port Inj	put Register
Conditions for	When [Used as an I/O]	port] of one or more pins are specified in the port in the GUI.
<u>output</u>		
Parameter	uint8_t * data	Destination for storage of the read pin state
Return value	true	Reading proceeded correctly.
	false	Reading failed.
File for output	R_PG_IO_PORT_P< <i>port number</i> >.c < <i>port number</i> >: 0 to 5, A to E, H and J	
RPDL function	R_IO_PORT_Read	
Details	• Reads Port Input Register to acquire the states of the pins. (Unit: Port)	
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" uint8_t data; void func(void) { //Acquire the states of P0 pins. R_PG_IO_PORT_Read_P0(&data); }</project></pre>	



5.11.4 R_PG_IO_PORT_Read_P<port number><pin number>

<u>Definition</u>	<pre>bool R_PG_IO_PORT_ <port number="">: 0 to 5 <pin number="">: 0 to 7</pin></port></pre>	_Read_P< <i>port number</i> >< <i>pin number</i> > (uint8_t * data) 5, A to E, H and J
Description	Read 1-bit data from Po	ort Input Register
<u>Conditions for</u> output	When [Used as an I/O port] of one or more pins are specified in the port in the GUI, the function of all existing pins in the port is generated.	
Parameter	uint8_t * data	Destination for storage of the read pin state
Return value	true	Reading proceeded correctly.
	false	Reading failed.
File for output	R_PG_IO_PORT_P< <i>port number</i> >.c (< <i>port number</i> >: 0 to 5, A to E, H and J)	
RPDL function	R_IO_PORT_Read	
Details	Reads Port Input Register to acquire the state of one pin.	
Example	The value is stored in the lowest-order bit of *data.	
Lixampie	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" uint8_t data_p03, data_p05; void func(void)</project></pre>	
	{	
	//Acquire the state	-
	R_PG_IO_PORT_	_Read_P03(& data_p03);
	//Acquire the state	of pin P05.
	R_PG_IO_PORT_	Read_P05(& data_p05);
	}	



5.11.5 R_PG_IO_PORT_Write_P<port number>

Definition	bool R_PG_IO_PORT_Write_P< <i>port number</i> > (uint8_t data) < <i>port number</i> >: 0 to 5, A to E, H and J	
Description	Write data to Port Output Data Register	
<u>Conditions for</u> output	When [Used as an I/O port] of one or more pins are specified in the port in the GUI. However, when only P35 is specified in the PORT3, R_PG_IO_PORT_Write_P3 is not generated.	
Parameter_	uint8_t data	Value to be written
<u>Return value</u>	true false	Writing proceeded correctly. Writing failed.
File for output	R_PG_IO_PORT_P< <i>port number</i> >.c < <i>port number</i> >: 0 to 5, A to E, H and J	
RPDL function	R_IO_PORT_Write	
<u>Details</u>	• Writes a value to Port Output Data Register. A value written to the register is output from the output port.	
<u>Example</u>		



5.11.6 R_PG_IO_PORT_Write_P<port number><pin number>

<u>Definition</u>	<pre>bool R_PG_IO_PORT_Write_P<port number=""><pin number=""> (uint8_t data) <port number="">: 0 to 5, A to E, H and J <pin number="">: 0 to 7</pin></port></pin></port></pre>	
Description	Write 1-bit data to Port	Output Data Register
<u>Conditions for</u> output	When [Used as an I/O port] is specified in the GUI. However, R_PG_IO_PORT_Write_P35 is not generated.	
Parameter	uint8_t data	Value to be written
<u>Return value</u>	true false	Writing proceeded correctly. Writing failed.
File for output	R_PG_IO_PORT_P	
RPDL function	R_IO_PORT_Write	
<u>Details</u>	• Writes a value to Por	t Output Data Register. A value written to an output port is output.
	Store the value in the	lowest-order bit of data.
Example	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) {</project></pre>	
	//Set P03.	
	R_PG_IO_PORT_Set_P03();	
	//Set P05.	
	R_PG_IO_PORT_	_Set_P05();
	//Output low level	from P03.
	R_PG_IO_PORT_	_Write_P03(0x00);
	//Output high leve	l from P05.
	R_PG_IO_PORT_	_Write_P05(0x01);
	}	



5.11.7 R_PG_IO_PORT_SetPortNotAvailable

Definition	bool R_PG_IO_PORT_SetPortNotAvailable (void)		
Description	Configure I/O port pins that are not available.		
Parameter	None	None	
Return value	true	Setting was made correctly	
File for output	R_PG_IO_PORT.c		
RPDL function	R_IO_PORT_NotAvailable		
<u>Details</u>	• All ports that are not available on smaller packages will be configured for CMOS-type low-level output.		
Example	 When using packages other than 100-pin, call this function first. Refer to the example of R_PG_IO_PORT_Set_P<port number="">.</port> 		
	rierer to the example of	The second	



5.12 Multi-Function Timer Pulse Unit 2 (MTU2a)

5.12.1 R_PG_Timer_Set_MTU_U<unit number>_<channels>

Definition bool R_PG_Timer_Set_MTU_U <unit number="">_<channels> (void) <unit number="">: 0 < <channels>: C0 to C5 C3_C4 (Complementary PWM mode or reset-synchronized PWM mode)</channels></unit></channels></unit>			
Description	Set up the MTU		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	<unit number="">: 0</unit>	J <unit number="">_C<channel number="">.c</channel></unit>	
RPDL function	R_MTU2_Set, R_MTU	J2_Create	
	 <channel number="">: 0 to 5</channel> R_MTU2_Set, R_MTU2_Create Releases the MTU from the module-stop and makes initial settings. Interrupts of the MTU are set by this function. When the name of the interrupt notification function has been specified in the GUI, if an interrupt occurs in the CPU, the function having the specified name will be called. Create the interrupt notification function as follows: void <name function="" interrupt="" notification="" of="" the=""> (void)</name> For the interrupt notification function, note the contents of section Notes on Notification Functions. If the interrupt propriety level is set to 0 in the GUI, a CPU interrupt does not occur. The state of a request flag can be acquired by calling R_PG_Timer_GetRequestFlag_MTU_U<unit number="">_C<channel number="">.</channel></unit> When counting driven by an externally input clock, the external reset signal, input capture, or pulse output is in use, the pin to be used is set in this function. To start the count operation, call R_PG_Timer_StartCount_MTU_U<<i>unit number></i>_C<channel number="">(<channel number="">(<cp>channel number>(<cp>color)</cp></cp></channel></channel> In complementary PWM mode or reset-synchronized PWM mode, paired channels are set up in the same time. Channels 3 and 4 are set up by R_PG_Timer_Set_MTU_U0_C3_C4. In complementary PWM mode or reset-synchronized PWM mode, PWM output is disabled in the initial state. To enable the pin output, call 		



Example 1	 A case where the setting is made as follows. MTU channel 1 was set up in normal mode Mtu1IcCmAIntFunc was specified as a compare match A interrupt notification function
	name
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>
	void func(void) { R DC Timer Set MTH H0 Close //Seture the MTH1
	R_PG_Timer_Set_MTU_U0_C1(); //Set up the MTU1 R_PG_Timer_StartCount_MTU_U0_C1(); // Start the count operation }
	void Mtu1IcCmAIntFunc(void)
	<pre>{ //Processing in response to a compare match A interrupt }</pre>
Example 2	A case where the setting is made as follows.
	MTU channel 3 and 4 were set up in complementary PWM mode
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>
	void func(void) {
	<pre>//Set up the MTU3 and MTU4 in complementary PWM mode R_PG_Timer_Set_MTU_U0_C3_C4 ();</pre>
	<pre>//Enable PWM output pin 1 positive and negative phase R_PG_Timer_ControlOutputPin_MTU_U0_C3_C4(</pre>
	1, //p1 : enable 1, //n1 : enable
	0, //p2 : disable
	0, //n2 : disable 0, //p3 : disable
	0 //n3 : disable);
	// Start the MTU3 and 4 count operation
	R_PG_Timer_SynchronouslyStartCount_MTU_U0(0, //ch0
	0, //ch1
	0, //ch2 1, //ch3
	1 //ch4);
	}



_	mber>(_ <phase>)</phase>			
<u>Definition</u>	<pre>bool R_PG_Timer_StartCount_MTU_U<unit number="">_C<channel number=""> (void) <unit number="">: 0 <channel number="">: 0 to 5</channel></unit></channel></unit></pre>			
	bool R_PG_Timer_StartCou <unit number="">: 0 <channel number="">: 5 <phase>: U, V or W</phase></channel></unit>	nnt_MTU_U <unit number="">_C<channel number="">_<phase> (void)</phase></channel></unit>		
Description	Start the MTU count of	Start the MTU count operation		
Parameter	None			
Return value	true	Setting was made correctly		
	false	Setting failed		
<u>File for output</u>	R_PG_Timer_MTU_U <unit number="">: 0 <channel number="">: 0</channel></unit>	V <unit number="">_C<channel number="">.c</channel></unit>		
RPDL function	R_MTU2_ControlCha	nnel		
<u>Details</u>	 before calling this fu In complementary P¹ operation of paired 2 R_PG_Timer_Synch 	et_MTU_U< <i>unit number</i> >_< <i>channels</i> > to make the initial settings		
<u>Example</u>	name #include "R_PG_defau void func(void) {	set up was specified as the compare match A interrupt notification function lt.h" //Include "R_PG_ <project name="">.h" to use this function. _MTU_U0_C1(); //Set up the MTU1 rtCount_MTU_U0_C1(); // Start the count operation</project>		

5.12.2 R_PG_Timer_StartCount_MTU_U<*unit number*>_C<*channel*

RENESAS

5.12.3 R_PG_Timer_SynchronouslyStartCount_MTU_U<unit number>

Definition	<pre>bool R_PG_Timer_SynchronouslyStartCount_MTU_U<unit number=""></unit></pre>		
	(bool ch0, bool ch1, bool ch2, bool ch3, bool ch4)		
	<unit number="">:</unit>	0	
Description	Start the MTU cou	nt operation of two or more channels simultaneously	
Parameter	bool ch0	Count operation of channel 0 (0:Do not start count 1:Start count)	
	bool ch1	Count operation of channel 1 (0:Do not start count 1:Start count)	
	bool ch2	Count operation of channel 2 (0:Do not start count 1:Start count)	
	bool ch3	Count operation of channel 3 (0:Do not start count 1:Start count)	
	bool ch4	Count operation of channel 4 (0:Do not start count 1:Start count)	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_Timer_MTU_U< <i>unit number</i> >.c < <i>unit number</i> >: 0		
RPDL function	R_MTU2_ControlUnit		
Details	• Starts the MTU count operation of two or more channels simultaneously.		
	• Call R_PG_Timer_Set_MTU_U< <i>unit number</i> >_< <i>channels</i> > to make the initial settings		
	before calling thi	-	
	e	y PWM mode or reset-synchronized PWM mode, start the count	
	-	ed 2 channels simultaneously by this function.	
Example	Refer to the example 2 of R_PG_Timer_Set_MTU_U <unit number="">_<channels></channels></unit>		



	mber>(_ <phase>)</phase>			
<u>Definition</u>	<unit number="">: 0 <channel number="">: 0 to 5</channel></unit>	nt_MTU_U< <i>unit number>_</i> C< <i>channel number></i> (void) nt_MTU_U< <i>unit number>_</i> C< <i>channel number>_</i> < <i>phase></i> (void)		
Description	Halt the MTU count op	eration		
Parameter	None			
Return value	true	Halting succeeded.		
	false	Halting failed.		
<u>File for output</u>	R_PG_Timer_MTU_U< <i>unit number>_</i> C< <i>channel number>.</i> c < <i>unit number></i> : 0 < <i>channel number></i> : 0 to 5			
RPDL function	R_MTU2_ControlChar	nel		
<u>Details</u>	 Halts the MTU count operation. To make the MTU resume counting, call R_PG_Timer_StartCount_MTU_U<<i>unit number></i>_C<<i>channel number></i>(_<<i>phase></i>) or R_PG_Timer_SynchronouslyStartCount_MTU_U<<i>unit number></i>. R_PG_Timer_HaltCount_MTU_U0_C5 can stop the count of U, V, and W phase simultaneously. 			
<u>Example</u>	name #include "R_PG_defaul void func(void) {	<pre>set up was specified as the compare match A interrupt notification function ht.h" //Include "R_PG_<project name="">.h" to use this function. MTU_U0_C1(); //Set up the MTU1 tCount_MTU_U0_C1(); // Start the count operation</project></pre>		

5.12.4 R_PG_Timer_HaltCount_MTU_U<*unit number*>_C<*channel number*>(_<*phase*>)



5.12.5 R_PG_Timer_GetCounterValue_MTU_U<unit number>_C<channel number>

<u>Definition</u>	<pre>(uint16_t * counter_val) <unit number="">: 0 <channel number="">: 0 to 4 bool R_PG_Timer_GetCounterVal</channel></unit></pre>	ue_MTU_U <unit number="">_C<channel number=""> ue_MTU_U<unit number="">_C<channel number=""> _t * counter_v_val, uint16_t * counter_w_val)</channel></unit></channel></unit>			
Description	Acquire the MTU counter value				
Parameter	-				
<u>r arameter</u>	For MTU0 to MTU4	Destinction for storage of the counter value			
	uint16_t * counter_val	Destination for storage of the counter value			
	For MTU5				
	uint16_t * counter_u_val	Destination for storage of the counter U value			
	uint16_t * counter_v_val	Destination for storage of the counter V value			
	uint16_t * counter_w_val	Destination for storage of the counter value			
Return value	true	Acquisition of the counter value succeeded.			
	false	Acquisition of the counter value failed.			
File for output	R_PG_Timer_MTU_U <unit number="">_C<channel number="">.c <unit number="">: 0 <channel number="">: 0 to 5</channel></unit></channel></unit>				
RPDL function	R_MTU2_ReadChannel				
<u>Details</u>	• Acquires the counter value of a MTU.				
<u>Example</u>	 A case where the setting is made as follows. MTU channel 0 was set up Set TGRA as an input capture register and enable an input capture A interrupt Mtu0IcCmAIntFunc was specified as the input capture A interrupt notification function name 				
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>				
	uint16_t counter_val;				
	void func(void)				
	<pre>{ R_PG_Timer_Set_MTU_U0_C0(); //Set up the MTU0 R_PG_Timer_StartCount_MTU_U0_C0(); // Start the count operation }</pre>				
	void Mtu0IcCmAIntFunc(void)				
	<pre>{ // Acquire the value of the MTU0 counter R_PG_Timer_GetCounterValue_MTU_U0_C0(& counter_val); }</pre>				

RENESAS

	6_Timer_SetCounterValue_MT nber>(_ <phase>)</phase>	U_U <unit number="">_C<channel< th=""></channel<></unit>		
<u>Definition</u>	<pre>bool R_PG_Timer_SetCounterValue_MTU_U<unit number="">_C<channel number=""> (uint16_t counter_val) <unit number="">: 0 <channel number="">: 0 to 4</channel></unit></channel></unit></pre>			
	bool R_PG_Timer_SetCounterValue_N (uint16_t counter_val) <unit number="">: 0 <channel num<="" td=""><td>MTU_U<unit number="">_C<channel number="">_<phase> nber>: 5 <phase>: U, V or W</phase></phase></channel></unit></td></channel></unit>	MTU_U <unit number="">_C<channel number="">_<phase> nber>: 5 <phase>: U, V or W</phase></phase></channel></unit>		
	<pre>bool R_PG_Timer_SetCounterValue_M (uint16_t counter_u_val, uint16_t coun</pre>			
Description	Set the MTU counter value			
Parameter	For MTU0 to MTU7			
	uint16_t counter_val	Value to be written to the counter		
	For MTU5 uint16_t counter_u_val	Value to be written to the counter U		
	uint16_t counter_v_val	Value to be written to the counter V		
	uint16_t counter_w_val	Value to be written to the counter W		
Return value	true	Setting of the counter value succeeded.		
	false	Setting of the counter value failed.		
File for output	R_PG_Timer_MTU_U <unit num<br=""><unit number="">: 0 <channel number="">: 0 to 5</channel></unit></unit>	ber>_C <channel number="">.c</channel>		
RPDL function	R_MTU2_ControlChannel			
Details	• Set the counter value of a MTU.			
<u>Example</u>	 A case where the setting is made as follows. MTU channel 1 was set up Set TGRA as an output compare register and enable a compare match A interrupt Mtu1IcCmAIntFunc was specified as the compare match A interrupt notification function name 			
		ude "R PG <pre>project name> h" to use this function</pre>		
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func (void)</project></pre>			
	<pre>{ R_PG_Timer_Set_MTU_U0 R_PG_Timer_StartCount_M } void Mtu1IcCmAIntFunc(void) {</pre>	D_C1(); //Set up the MTU1 (TU_U0_C1(); // Start the count operation lue_MTU_U0_C1(0); //Clear the counter		



5.12.7 R_PG_Timer_GetRequestFlag_MTU_U<unit number>_C<channel number>

<u>Definition</u>	bool R_PG_Time (bool* cm_ic_a, bool* cm_e, <unit number<br=""><channel num<="" th=""><th>bool* cm_ic_b, bool* bool* cm_f, bool* r>: 0</th><th>_U<unit number="">_C<channel number=""> cm_ic_c, bool*cm_ic_d, ov, bool*un);</channel></unit></th></channel></unit>	bool* cm_ic_b, bool* bool* cm_f, bool* r>: 0	_U <unit number="">_C<channel number=""> cm_ic_c, bool*cm_ic_d, ov, bool*un);</channel></unit>		
	bool R_PG_Time (bool* cm_ic_ <unit number<br=""><channel nur<="" td=""><td>u, bool* cm_ic_v, l r>: 0</td><td>_U<<i>unit number</i>>_C<<i>channel number</i>> pool* cm_ic_w);</td></channel></unit>	u, bool* cm_ic_v, l r>: 0	_U< <i>unit number</i> >_C< <i>channel number</i> > pool* cm_ic_w);		
Description	Acquire and clear the MTU interrupt flags				
Parameter	bool* cm_ic_a The address of storage area for the compare		rea for the compare match/input capture A flag		
	bool* cm_ic_b		rea for the compare match/input capture B flag		
	bool* cm_ic_c		rea for the compare match/input capture C flag		
	bool* cm_ic_d	The address of storage a	rea for the compare match/input capture D flag		
	bool* cm_e	The address of storage a	rea for the compare match E flag		
	bool* cm_f	The address of storage a	rea for the compare match F flag		
	bool* ov	The address of storage a	rea for the overflow flag		
	bool* un	The address of storage a	rea for the underflow flag		
	bool* cm_ic_u	The address of storage a	rea for the compare match/input capture U flag		
	bool* cm_ic_v	The address of storage a	rea for the compare match/input capture V flag		
	bool* cm_ic_w	The address of storage a	rea for the compare match/input capture W flag		
	Available flags for each channel are as follows.				
	MTU0		cm_ic_a to cm_ic_d, cm_e, cm_f, and ov		
	MTU1, 2		cm_ic_a, cm_ic_b, ov, and un		
	MTU3, 4		cm_ic_a to cm_ic_d, and ov		
	MTU5		cm_ic_u, cm_ic_v, and cm_ic_w		
	· •	PWM mode and ized PWM mode)	cm_ic_a to cm_ic_d		
	· •	MTU4 (complementary PWM mode and reset-synchronized PWM mode) cm_ic_a to cm_ic_d, and un			
Return value	true	Acquisition of the flags	succeeded		
	false	Acquisition of the flags			
File for output	R_PG_Timer_MTU_U <unit number="">_C<channel number="">.c</channel></unit>				
	<channel num<="" td=""><td></td><td></td></channel>				
RPDL function	R_MTU2_Read	Channel			
<u>Details</u>	 This function acquires the interrupt flags of MTU. All flags will be cleared in this function. Specify the address of storage area for the flags to be acquired. Specify 0 for a flag that is not required. 				

RENESAS

Example

A case where the setting is made as follows.
• MTU channel 1 was set up
• TGRA is set as an output compare register and the compare match interrupt is enabled
• The priority level of compare match interrupt is set to 0
#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>
bool cma_flag;
void func(void)
{
R_PG_Timer_Set_MTU_U0_C1(); //Set up the MTU1
R_PG_Timer_StartCount_MTU_U0_C1(); // Start the count operation
//Wait for the compare match A
do{
R_PG_Timer_GetRequestFlag_MTU_U0_C1(
& cma_flag, //a
0, //b
0, //c
0, //d
0, //e
0, //f
0, //e
0, //ov 0 //un
);
} while(!cma_flag);
//Processing in response to a compare match A
}



5.12.8 R_PG_Timer_StopModule_MTU_U<unit number>

Definition	<pre>bool R_PG_Timer_StopModule_MTU_U<unit number=""> (void) <unit number="">: 0</unit></unit></pre>			
Description	Shut down the MTU unit			
Parameter	None			
Return value	true	Shutting down succeeded		
	false	Shutting down failed		
File for output	R_PG_Timer_MTU_U- <unit number="">: 0</unit>	<unit number="">.c</unit>		
RPDL function	R_MTU2_Destroy			
<u>Details</u>	 Stops a MTU and places it in the module-stop state. If two or more channels are running when this function is called, all channels will be stopped. Call R_PG_Timer_HaltCount_MTU_U<<i>unit number></i>_C<<i>channel number></i>(_<<i>phase></i>) to stop a single channel. 			
Example	 A case where the setting is made as follows. MTU channel 1 was set up Set TGRA as an output compare register and enable a compare match A interrupt Mtu1IcCmAIntFunc was specified as the compare match A interrupt notification function name 			
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>			
	<pre>void func(void) { R_PG_Timer_Set_MTU_U0_C1(); //Set up the MTU1 R_PG_Timer_StartCount_MTU_U0_C1(); // Start the count operation }</pre>			
	void Mtu1IcCmAIntFu	nc(void)		
	{ // Stop the MTU u	nit 0		
	R_PG_Timer_StopModule_MTU_U0(); }			



5.12.9 R_PG_Timer_GetTGR_MTU_U<unit number>_C<channel number>

<u>Definition</u>	(uint16_t* tgr_a_val,	uint16_t* tgr_b_val, uir uint16_t* tgr_e_val, uir	-	
		l, uint16_t * tgr_v_val,	t number>_C <channel number=""> uint16_t * tgr_w_val);</channel>	
Description	Acquire the general register value			
Parameter	uint16_t* tgr_a_val The address of storage area for the general register A			
	uint16_t* tgr_b_val	The address of storage ar	rea for the general register B value	
	uint16_t* tgr_c_val	The address of storage ar	rea for the general register C value	
	uint16_t* tgr_d_val	The address of storage ar	rea for the general register D value	
	uint16_t* tgr_e_val The address of storage area for the general register E value			
	uint16_t* tgr_f_val The address of storage area for the general register F value			
	uint16_t* tgr_u_val	The address of storage ar	rea for the general register U value	
	uint16_t* tgr_v_val	The address of storage ar	rea for the general register V value	
	uint16_t* tgr_w_val The address of storage area for the general register W value			
	Available arguments for	r each channel are as follow	vs.	
	MTU0		tgr_a_val to tgr_f_val	
	MTU1, 2		tgr_a_val and tgr_b_val	
	MTU3, 4		tgr_a_val to tgr_d_val	
	MTU5		tgr_u_val to tgr_w_val	
Return value	true	Acquisition of the flags s	ucceeded	
	false	Acquisition of the flags f	ailed	
File for output	R_PG_Timer_MTU_U <unit number="">_C<channel number="">.c <unit number="">: 0 <channel number="">: 0 to 5</channel></unit></channel></unit>			
RPDL function	R_MTU2_ReadChanne	1		
<u>Details</u>	 This function acquires the general register value. Specify the address of storage area for an item to be acquired. Specify 0 for an item that is not required. 			



Example

A case where the setting is made as follows.

- MTU channel 0 was set up
- Set TGRA as an input capture register and enable an input capture A interrupt
- Mtu0IcCmAIntFunc was specified as the input capture A interrupt notification function name

#include "R_PG_default.h" //Include "R_PG_<project name>.h" to use this function.

uint16_t tgr_a_val;

void func(void)

```
{
    R_PG_Timer_Set_MTU_U0_C0();
                                         //Set up the MTU0
    R_PG_Timer_StartCount_MTU_U0_C0();
                                               // Start the count operation
}
void Mtu0IcCmAIntFunc(void)
{
    // Acquire the value of the TGRA
    R_PG_Timer_GetTGR_MTU_U0_C0(
         & tgr_a_val, //a
         0, //b
         0, //c
         0, //d
        0, //e
         0 //f
    );
}
```



5.12.10 R_PG_Timer_SetTGR_<general register>_MTU_U<unit number>_C<channel number>

Definition bool	ol R_PG_Timer_SetTGR_ <general register="">_MTU_U<unit number="">_C<channel number=""></channel></unit></general>				
(uin	t16_t value);				
<	general register>	:			
	MTU1, 2	: A or B			
	MTU3, 4	: A, B, C or D			
	MTU5	: U, V or W			
	unit number>: 0				
<	channel number>	: 0 to 5			
Description	Set the general r	egister value			
Parameter	uint16_t value	Value to be written to the general register			
Return value	true	Setting of the general register succeeded.			
	false	Setting of the general register failed.			
File for output	R_PG_Timer_M	ITU_U <unit number="">_C<channel number="">.c</channel></unit>			
	<unit number=""></unit>	>: 0			
	<channel num<="" td=""><td><i>ber></i>: 0 to 5</td></channel>	<i>ber></i> : 0 to 5			
RPDL function	R_MTU2_Cont	rolChannel			
Details	• This function sets the general register value.				
<u>Example</u>	A case where the setting is made as follows.				
	• MTU channel 1 was set up				
	• Set TGRA as an output compare register and enable a compare match A interrupt				
	Mtu1IcCmAIntFunc was specified as the compare match A interrupt notification function				
	name				
	#include "R_PG	_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>			
	void func (void)				
	{				
		er_Set_MTU_U0_C1(); //Set up the MTU1			
	K_PG_11m	er_StartCount_MTU_U0_C1(); // Start the count operation			
	void Mtu1IcCm	AIntrunc(void)			
	R PG Tim	uer_SetTGR_A_MTU_U0_C1(1000); //Set TGRA			
	}				



5.12.11 R_PG_Timer_SetBuffer_AD_MTU_U<unit number>_C<channel number>

<u>Definition</u>	<pre>bool R_PG_Timer_SetBuffer_AD_MTU_U<unit number="">_C<channel number=""> (uint16_t tadcobr_a_val, uint16_t tadcobr_b_val); <unit number="">: 0 <channel number="">: 4</channel></unit></channel></unit></pre>			
Description Conditions for output		ycle set buffer registers (TADCOBRA and TADCOBRB) rter start request cycle value is enabled.		
Parameter	uint16_t tadcobr_a_val	Value to be written to TADCOBRA		
	uint16_t tadcobr_b_val	Value to be written to TADCOBRB		
Return value	true	Setting of the counter value succeeded.		
	false	Setting of the counter value failed.		
File for output	R_PG_Timer_MTU_U< <i>unit number>_</i> C< <i>channel number>.</i> c < <i>unit number></i> : 0 < <i>channel number></i> : 3(*), 4 (* complementary PWM mode and reset-synchronized PWM mode)			
RPDL function	R_MTU2_ControlChannel			
Details	• This function sets the TADCOBRA and TADCOBRB values.			
Example	A case where the setting is made as follows.Buffer transfer of A/D converter start request cycle set register has been enabled			
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func (void) { R_PG_Timer_Set_MTU_U0_C4(); //Set up the MTU1 R_PG_Timer_StartCount_MTU_U0_C4(); // Start the count operation } void Mtu1IcCmAIntFunc(void) { // Set TADCOBRA and TADCOBRB R_PG_Timer_SetBuffer_AD_MTU_U0_C4(0x10, 0x20); }</project></pre>			



5.12.12 R_PG_Timer_SetBuffer_CycleData_MTU_U<unit number>_<channels>

Definition	bool R_PG_Timer_SetBuffer_Cy	cleData_MTU_U <unit number="">_<channels></channels></unit>	
	(uint16_t tcbr_val);		
	<unit number="">: 0</unit>		
	< <i>channels</i> >: C3_C4		
Description	Set the cycle buffer register		
Conditions for	MTU channels are set to complet	nentary PWM mode	
<u>output</u>			
Parameter_	uint16_t tcbr_val	Value to be written to the cycle buffer registerr	
Return value	true	Setting of the counter value succeeded.	
	false	Setting of the counter value failed.	
File for output	R_PG_Timer_MTU_U< <i>unit number>_</i> C< <i>channel number>.</i> c < <i>unit number></i> : 0 < <i>channel number></i> : 3		
RPDL function	R_MTU2_ControlUnit		
<u>Details</u>	• This function sets the cycle buffer register (TCBR).		
Example	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>		
	void func (void)		
	{		
		cleData_MTU_U0_C3_C4(0x1000);	
	}		
	<u> </u>		



5.12.13 R_PG_Timer_SetOutputPhaseSwitch_MTU_U<unit number>_<channels>

<u>Definition</u>	<pre>bool R_PG_Timer_SetOutputPhaseSwitch_MTU_U<unit number="">_<channels> (uint8_t output_level); <unit number="">: 0 <channels>: C3_C4</channels></unit></channels></unit></pre>							
Description Conditions for output	 Switch PWM output level The MTU channels are set to complementary PWM mode or reset-synchronized PWM mode The brushless DC motor control is enabled and the software is selected for the output control method 							
Parameter_		utput_level			VM output sett	ting (0 to 5)		
	The outp	ut level for each	ch value is	as f	follows			
	Value	MTIOC3B	MTIOC ₄	1A	MTIOC4B	MTIOC3D	MTIOC4C	MTIOC4D
		U phase	V phase	e	W phase	U phase	V phase	W phase
	0	OFF	OFF		OFF	OFF	OFF	OFF
	1	ON	OFF		OFF	OFF	OFF	ON
	2	OFF	ON		OFF	ON	OFF	OFF
	3	OFF	ON		OFF	OFF	OFF	ON
	4	OFF	OFF		ON	OFF	ON	OFF
	5	ON	OFF		OFF	OFF	ON	OFF
	6	OFF	OFF		ON	ON	OFF	OFF
	7	OFF	OFF		OFF	OFF	OFF	OFF
Return value	true			Se	tting of the co	unter value suc	cceeded.	
	false			Se	tting of the cou	unter value fai	led.	
File for output	R_PG_Timer_MTU_U <unit number="">_C<channel number="">.c <unit number="">: 0 <channel number="">: 3</channel></unit></channel></unit>							
RPDL function	R_MTU2_ControlUnit							
Details	• This function switches the PWM output level in brushless DC motor control							
Example	#include	"R_PG_defau	lt.h" //Inclu	ude	"R_PG_ <proj< td=""><td>ect name>.h" t</td><td>to use this fund</td><td>ction.</td></proj<>	ect name>.h" t	to use this fund	ction.
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func (void) {</project></pre>							



5.12.14 R_PG_Timer_ControlOutputPin_MTU_U<unit number>_<channels>

<u>Definition</u>	(bool p1_enable, bo	ntrolOutputPin_MTU_U< <i>unit number</i> >_< <i>channels></i> pol n1_enable, bool p2_enable, bool n2_enable, pol n3_enable) 4		
Description	Enable or disable the PWM output			
Conditions for output	MTU channels are set t	to complementary PWM mode or reset-synchronized PWM mode		
Parameter	bool p1_enable	U positive phase (MTIOCmB) output (0: Disable 1: Enable)		
	bool n1_enable	U negative phase (MTIOCmD) output (0: Disable 1: Enable)		
	bool p2_enable	V positive phase (MTIOCnA) output (0: Disable 1: Enable)		
	bool n2_enable	V negative phase (MTIOCnC) output (0: Disable 1: Enable)		
	bool p3_enable	W positive phase (MTIOCnB) output (0: Disable 1: Enable)		
	bool n3_enable	W negative phase (MTIOCnD) output (0: Disable 1: Enable)		
		m:3 n:4		
Return value	true	Setting of the counter value succeeded.		
	false	Setting of the counter value failed.		
File for output	R_PG_Timer_MTU_U< <i>unit number</i> >_C< <i>channel number</i> >.c < <i>unit number</i> >: 0 < <i>channel number</i> >: 3			
RPDL function	R_MTU2_ControlUnit			
Details	 This function enables or disables PWM output in complementary PWM mode or reset-synchronized PWM mode. In complementary PWM mode or reset-synchronized PWM mode, PWM output is disabled in the initial state. To enable the pin output, call this function before starting the count operation. 			
Example	Refer to the example 2 of R_PG_Timer_Set_MTU_U <unit number="">_<channels></channels></unit>			



5.12.15 R_PG_Timer_SetBuffer_PWMOutputLevel_MTU_U<unit number>_<channels>

<u>Definition</u>	<pre>bool R_PG_Timer_SetBuffer_PWMOutputLevel_MTU_U<unit number="">_<channels> (bool p1_high, bool n1_high, bool p2_high, bool n2_high, bool p3_high, bool n3_high) <unit number="">: 0 <channels>: C3_C4</channels></unit></channels></unit></pre>				
Description	Set the PV	VM output le	evel in the buffer register		
<u>Conditions for</u> output			set to complementary PWM n WM output level setting is en	•	nized PWM mode
Parameter	bool p1_h	igh	U positive phase (MTIOCm	nB) output	
	bool n1_h	igh	U negative phase (MTIOCn	nD) output	
	bool p2_h	igh	V positive phase (MTIOCn.	A) output	
	bool n2_h	igh	V negative phase (MTIOCn	nC) output	
	bool p3_h	igh	W positive phase (MTIOCn	nB) output	
	bool n3_h	igh	W negative phase (MTIOC)	nD) output	
			m:3 n:4		
	The outpu	it level in eac	h value is as follows		
	Value	Category		Positive phase	Negative phase
	0	Active lev	el	Low	Low
		Initial outp		Low	Low
		-	natch when up count	Low	High
			match when down count	High	Low
	1	Active lev		High	High
		Initial outp	out	High	High
			natch when up count	High	Low
		Compare r	match when down count	Low	High
Return value			·		
	folso		Setting of the counter value succeeded. Setting of the counter value failed.		
	false		Setting of the counter value	lalled.	
File for output	R_PG_Timer_MTU_U< <i>unit number>_</i> C< <i>channel number>.</i> c < <i>unit number></i> : 0 < <i>channel number></i> : 3				
RPDL function	R_MTU2	_ControlUnit	t		
<u>Details</u>	• This function sets the output level settings to the timer output level buffer register (TOLBR)				
Example	#include '	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>			
	<pre>whichde K_PG_default.n //metude K_PG_<pre>project name>.n to use this function. void func (void) {</pre></pre>				



5.12.16 R_PG_Timer_ControlBufferTransfer_MTU_U<unit number>_<channels>

<u>Definition</u>	bool R_PG_Timer_Con (bool enable) <unit number="">: 0 <channels>: C3_C</channels></unit>	ntrolBufferTransfer_MTU_U< <i>unit number>_<channels></channels></i>	
Description	Enable or disable buffe	r transfer from the buffer registers to the temporary registers	
<u>Conditions for</u> output	The MTU channels are set to complementary PWM modeInterrupt skipping function is set		
Parameter	bool enable	Buffer transfer control (0 :Disable 1 :Enable)	
Return value	true	Setting of the counter value succeeded.	
	false	Setting of the counter value failed.	
File for output	R_PG_Timer_MTU_U <unit number="">: 0 <channel number="">: 3</channel></unit>	<unit number="">_C<channel number="">.c</channel></unit>	
RPDL function	R_MTU2_ControlUnit		
<u>Details</u>	• This function enable. PWM mode to the te	s or disables transfer from the buffer registers used in complementary mporary registers.	
<u>Example</u>	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. void func (void) {</project></pre>		



5.13 Port Output Enable 2 (POE2a)

5.13.1 R_PG_POE_Set

Definition	bool R_PG_POE_Set (void)		
Description	Set up the POE		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_POE.c		
RPDL function	R_POE_Set, R_I	POE_Create	
	 request signal i The MTU mod Do not set pins When the name interrupt occur interrupt notifie void <<i>name</i> For the interrup Functions. 	put control of MTU0, 3 and 4 pins, the POE pins used for high-impedance nput, and the output enable interrupt. ule is not set up in this function. that are not used for MTU output. e of the interrupt notification function has been specified in the GUI, if an s in the CPU, the function having the specified name will be called. Create the cation function as follows: of the interrupt notification function> (void) ot notification function, note the contents of section Notes on Notification	
<u>Example</u>	Example • The output enable interrupt 2(OEI2) has been set		
<u>Brumpte</u>	-	c has been specified as an interrupt notification function name	
	//Include "R_PG #include "R_PG	_ <project name="">.h" to use this function. _default.h"</project>	
	<pre>void func(void) { R_PG_POE } void PoeOei2Int! { // Processin; }</pre>		



5.13.2 R_PG_POE_SetHiZ_<Timer channels>

Definition	bool R_PG_POE_SetHiZ_< <i>Timer channels</i> >(void)			
	< <i>Timer channels</i> >: MTU3_4, MTU0			
Description	Place the timer o	utput pins in high-impedance state		
Parameter	None			
Return value	true	Setting was made correctly		
	false	Setting failed		
File for output	R_PG_POE.c			
RPDL function	R_POE_Control			
Details	Places MTU0, 3, 4 output pins in high-impedance state.			
<u>Example</u>	A case where the s	etting is made as follows.		
		but has been set (Setting of MTU)		
	MTU0 output p	pins have been set to be controlled by the high impedance request		
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>			
	void func1(void)			
	{			
	R_PG_Time	er_Set_MTU_U0_C0(); //Set up the MTU0		
	R_PG_POE_Set(); // Set up the POE			
	R_PG_Timer_StartCount_MTU_U0_C0(); //Start the count operation of MTU0			
	}			
	void func2(void)			
	{			
	R_PG_POE	SetHiZ_MTU0(); // Place the MTU0 output pins in high-impedance state		
	}			



5.13.3 R_PG_POE_GetRequestFlagHiZ_<Timer channels/flag>

Definition	<pre>bool R_PG_POE_GetRequestFlagHiZ_MTU3_4 (bool * poe0, bool * poe1, bool * poe2, bool * poe3)</pre>			
	bool R_PG_PC	DE_GetRequestFlagHiZ_MTU0 (bool * poe8)		
	bool R_PG_POE_GetRequestFlagHiZ_OSTSTF (bool * oststf)			
Description	Acquire the hig	h-impedance request flags		
Parameter	bool* poe0	The address of storage area for POE0# high-impedance request flags		
	bool* poe1	The address of storage area for POE1# high-impedance request flags		
	bool* poe2	The address of storage area for POE2# high-impedance request flags		
	bool* poe3	The address of storage area for POE3# high-impedance request flags		
	bool* poe8	The address of storage area for POE8# high-impedance request flags		
	bool * oststf	The address of storage area for OSTST high-impedance flag		
Return value	true	Acquisition succeeded		
	false	Acquisition failed		
File for output	R_PG_POE.c			
RPDL function	R_POE_GetSta	atus		
<u>Details</u>	 Acquires the flags of high-impedance request signals input to POEn#pins (POEnF). (n:0 to 3 and 8) 			
	• Specify the address of storage area for the flags to be acquired. Specify 0 for a flag that is not required.			
	• The flag is va	alid only when the POE pin is set to a high-impedance request input in GUI.		
<u>Example</u>	A case where the setting is made as follows.			
	• MTU3 and 4 pin output has been set (Setting of MTU)			
	• MTU3 and 4 output pins have been set to be controlled by the high impedance request			
	POE0 has been selected as a high-impedance request signal input			
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>			
	bool poe0;			
	1 .			
	void func(void)			
	R_PG_Timer_Set_MTU_U0_C3(); //Set up the MTU R_PG_POE_Set(); // Set up the POE R_PG_Timer_StartCount_MTU_U0_C3(); //Start the count operation of MTU			
	//Wait for	the high-impedance request signal to be input		
	do{ R_P(}while(!]	G_POE_GetRequestFlagHiZ_MTU3_4(&poe0, 0, 0, 0); poe0);		
	//Processin	ng when the high-impedance request signal is input		
	R_PG_PC	DE_ClearFlag_MTU3_4(); //Clear high-impedance request flag		

5.13.4 R_PG_POE_GetShortFlag_<Timer channels>

Definition	bool R_PG_POE_GetShortFlag_< <i>Timer channels></i> (bool * detected) < <i>Timer channels></i> : MTU3_4			
Description	Acquire the MT	U output short flags		
Parameter	bool* detected	bool* detected The address of storage area for the output short flag (OSF1)		
Return value	true	Acquisition succeeded		
	false	Acquisition failed		
File for output	R_PG_POE.c			
RPDL function	R_POE_GetStat	us		
Details	• Acquires the M	ITU3 ,4 complementary PWM output short flags (OSF1).		
<u>Example</u>	 A case where the setting is made as follows. The output enable interrupt1(OEI1) has been set. PoeOei1IntFunc has been specified as the output enable interrupt 1 notification function name. 			
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) {</project></pre>			
	if(detected //Proce){ essing when MTU3,4 output short is detected		
	R_PG_ } }	_POE_ClearFlag_MTU3_4(); // Clear the output short flag(OSF1)		



5.13.5 R_PG_POE_ClearFlag_<Timer channels/flag>

<u>Definition</u>	bool R_PG_POE_Clear <timer channels="" flag<="" th=""><th>-</th><th></th></timer>	-		
Description	Clear the high-impedance request flags and the output short flags			
Parameter	None			
Return value	true	Clearing succeede	ed	
	false	Clearing failed		
File for output	R_PG_POE.c			
RPDL function	R_POE_Control			
<u>Details</u>	Clears the high-impedance request flags and the output short flags.The flags that shall be cleared by each function are as follows.			
	Timer channels / f	lag	Flags	
	MTU3, 4		POEn request flag (POEnF) (n:0 to 3)	
			MTU3,4 output short flag(OSF1)	
	MTU0		POE8 request flag (POE8F)	
	OSTSTF		OSTST high-impedance flag	
Fyample	Pafar to the axample of L	D DG DOE GotSh	ortElag <timer channels<="" td=""></timer>	

Example

Refer to the example of R_PG_POE_GetShortFlag_<*Timer channels*>



5.14 8-Bit Timer (TMR)

5.14.1 R_PG_Timer_Start_TMR_U<unit number>(_C<channel number>)

<u>Definition</u>	<unit number="">: 0 or <channel number="">: 0</channel></unit>		
Description	Set up the TMR and st	art the count operation	
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_Timer_TMR_U < <i>unit number</i> >: 0 and		
RPDL function	R_TMR_Set R_TMR_CreateChannel (8-bit mode) R_TMR_CreateUnit (16-bit mode)		
Details			



Example1	The 16-bit timer mode has been specified for TMR unit 1. In this case, the following interrupt notification functions have been set in the GUI.			
	Overflow interrupt: TmrOf2IntFunc			
	Compare match A interrupt: TmrCma2IntFunc			
	Compare match B interrupt: TmrCmb2IntFunc			
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>			
	<pre>void func(void) {</pre>			
	<pre>//Place TMR unit 1 in the 16-bit mode. R_PG_Timer_Start_TMR_U1(); }</pre>			
	<pre>void TmrOf2IntFunc(void) {</pre>			
	<pre>func_of(); //Processing in response to an overflow interrupt }</pre>			
	void TmrCma2IntFunc(void)			
	<pre>func_cma(); //Processing in response to a compare match A interrupt }</pre>			
	<pre>void TmrCmb2IntFunc(void) {</pre>			
	<pre>func_cmb(); //Processing in response to a compare match B interrupt }</pre>			
Example2	The 8-bit timer mode has been specified for TMR0 in the GUI.			
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>			
	<pre>void func1(void) {</pre>			
	bool cma_flag;			
	<pre>//Place TMR0 in the 8-bit mode and start it counting. R_PG_Timer_Start_TMR_U0_C0();</pre>			
	while(1){ bool flag; //Acquire the compare match A interrupt request flag. R_PG_Timer_GetRequestFlag_TMR_U0_C0(&cma_flag, 0, 0);			
	<pre>if(cma_flag){ func_cma0(); //Processing of IRQ0 } }</pre>			
	}			



5.14.2 R_PG_Timer_HaltCount_TMR_U<unit number>(_C<channel number>)

<u>Definition</u>	<pre>bool R_PG_Timer_HaltCount_TMR_U<unit number="">(_C<channel number="">) (void) <unit number="">: 0 or 1 <channel number="">: 0 to 3 ((_C<channel number="">) is added in the 8-bit mode.)</channel></channel></unit></channel></unit></pre>		
Description	Halt the TMR count op	eration	
Parameter	None		
Return value	true	Halting succeeded.	
	false	Halting failed.	
File for output	R_PG_Timer_TMR_U <unit number="">: 0 or</unit>		
RPDL function	R_TMR_ControlChan	nel (8-bit mode)	
	R_TMR_ControlUnit (16-bit mode)	
<u>Details</u>	 Halts the TMR count operation. To make the TMR resume counting, call the following function. R_PG_Timer_ResumeCount_TMR_U<<i>unit number</i>>(_C<<i>channel number</i>>) 		
<u>Example</u>	The 8-bit timer mode was specified for TMR0 in the GUI. TmrCma0IntFunc was specified as the name of the compare match A interrupt function in the GUI.		
	<pre>#include "R_PG_defau void func(void) { //Place TMR0 in t R_PG_Timer_Stat } void TmrCma0IntFunc { //Halt counting by R_PG_Timer_Hal func_cma(); // //Resume counting</pre>	he 8-bit mode. rt_TMR_U0_C0(); (void) TMR0. tCount_TMR_U0_C0(); /Processing in response to a compare match A interrupt	



5.14.3 R_PG_Timer_ResumeCount_TMR_U<unit number>(_C<channel number>)

<u>Definition</u>	<pre>bool R_PG_Timer_ResumeCount_TMR_U<unit number="">(_C<channel number="">) (void) <unit number="">: 0 or 1 <channel number="">: 0 to 3 ((_C<channel number="">) is added in the 8-bit mode.)</channel></channel></unit></channel></unit></pre>		
Description	Resume the TMR coun	t operation	
Parameter	None		
Return value	true	Resuming count succeeded.	
	false	Resuming count failed.	
File for output	R_PG_Timer_TMR_U< <i>unit number</i> >.c < <i>unit number</i> >: 0 or 1		
RPDL function	R_TMR_ControlChannel (8-bit mode) R_TMR_ControlUnit (16-bit mode)		
<u>Details</u>	• Resumes counting by a TMR that was halted by R_PG_Timer_HaltCount_TMR_U< <i>unit</i> number>(_C< <i>channel</i> number>).		
Example	Refer to the example of R_PG_Timer_HaltCount_TMR_U< <i>unit number</i> >(_C< <i>channel number</i> >)		


5.14.4 R_PG_Timer_GetCounterValue_TMR_U<unit number>(_C<channel number>)

<u>Definition</u>	 •8-bit mode bool R_PG_Timer_GetCounterValue_TMR_U<<i>unit number></i>_C<<i>channel number></i> (uint8_t * data) <i><unit number=""></unit></i>: 0 or 1 <i><channel number=""></channel></i>: 0 to 3 •16-bit mode bool R_PG_Timer_GetCounterValue_TMR_U<<i>unit number></i> (uint16_t * data) <i><unit number=""></unit></i>: 0 or 1 		
Description	Acquire the TMR counter value		
Parameter	uint8_t * data (8-bit mode)Destination for storage of the counter valueuint16_t * data (16-bit mode)Destination for storage of the counter value		
Return value	true Acquisition of the counter value succeeded.		
	false Acquisition of the counter value failed.		
File for output	<i><unit number=""></unit></i> : 0 or 1		
Details • Acquires the counter value of a TMR. The value of the 8-bit counter for the specified channel is stored if the TMR unit is in the 8-bit timer mode. The counter values for both channels are stored as follows if the TMR unit is in the 16-bit mode.			
	Unit b15 to b8 b7 to b0		
	0 TMR0 counter TMR1 counter		
	1TMR2 counterTMR3 counter		
<u>Example</u>	*When the TMR unit is in the 16-bit mode, the higher-order bits are in TMR0 (or TMR2). The 8-bit timer mode was selected for TMR0 in the GUI.		
	<pre>#include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. uint8_t counter_val; void func1(void) { R_PG_Timer_Start_TMR_U0_C0(); //Place TMR0 in the 8-bit mode. } void func2(void) { //Acquire the value of a counter of TMR0. R_PG_Timer_GetCounterValue_TMR_U0_C0(&counter_val); }</project></pre>		



5.14.5 R_PG_Timer_SetCounterValue_TMR_U<unit number>(_C<channel number>)

	 •8-bit mode bool R_PG_Timer_SetCounterValue_TMR_U<<i>unit number></i>_C<<i>channel number></i> (uint8_t data) <i><unit number=""></unit></i>: 0 or 1 <i><channel number=""></channel></i>: 0 to 3 •16-bit mode bool R_PG_Timer_SetCounterValue_TMR_U<<i>unit number></i> (uint16_t data) <i><unit number=""></unit></i>: 0 or 1 		
Description	Set the TMR counter value		
Parameter	uint8_t data (8-bit mode) uint16_t data (16-bit mode)Value to be set to the coun	ter	
Return value	true Setting of the counter valu	e succeeded.	
	false Setting of the counter valu	e failed.	
<u>File for output</u>	R_PG_Timer_TMR_U< <i>unit number</i> >.c < <i>unit number</i> >: 0 or 1		
RPDL function	R_TMR_ControlChannel (8-bit mode) R_TMR_ControlUnit (16-bit mode)		
<u>Details</u>	 Set the counter value of a TMR. The value of the 8-bit counter for the specified channel is stored if the TMR unit is in the 8-bit timer mode. The counter values for both channels are stored as follows if the TMR unit is in the 16-bit mode. 		
	Unit b15 to b8 b7 to b)	
	0 TMR0 counter TMR1	counter	
	1 TMR2 counter TMR3	counter	
	*When the TMR unit is in the 16-bit mode, the higher-order	bits are in TMR0 (or TMR2).	
Example	The 8-bit timer mode was selected for TMR0 in the GUI.		
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>		
	void func1(void)		
	{ //Place TMR0 in the 8-bit mode.		
	R_PG_Timer_Start_TMR_U0_C0();		
	}		
	void func2(void) {		
	<pre>//Set the value of a counter of TMR0. R_PG_Timer_SetCounterValue_TMR_U0_C0(0); }</pre>		



5.14.6 R_PG_Timer_GetRequestFlag_TMR_U<unit number>(_C<channel number>)

<u>Definition</u>	<pre>bool R_PG_Timer_GetRequestFlag_TMR_U<unit number="">_C<channel number=""> (bool* cma, bool* cmb, bool* ov); <unit number="">: 0 or 1 <channel number="">: 0 to 3 ((_C<channel number="">) is added in the 8-bit mode.)</channel></channel></unit></channel></unit></pre>	
Description	Acquire and clea	ar the TMR interrupt flags
Parameter Parameter	bool* cma	The address of storage area for the compare match A flag
	bool* cmb	The address of storage area for the compare match B flag
	bool* ov	The address of storage area for the overflow flag
Return value	true	Acquisition of the flags succeeded
	false	Acquisition of the flags failed
File for output	R_PG_Timer_T <unit number=""></unit>	MR_U< <i>unit number</i> >.c
RPDL function		hannel (8-bit mode) (nit (16-bit mode)
<u>Details</u>	 This function acquires the interrupt flags of TMR. All flags will be cleared in this function. Specify the address of storage area for the flags to be acquired. Specify 0 for a flag that is not required. 	
<u>Example</u>	The 8-bit timer mode was selected for TMR0 in the GUI.	
	<pre>#include "R_PG uint16_t counter void func(void) { //Place TM R_PG_Tim //Wait for th do{ R_PG_T</pre>	_default.h" //Include "R_PG_ <project name="">.h" to use this function. ; R0 in the 8-bit mode. er_Start_TMR_U0_C0(); e compare match A Cimer_GetRequestFlag_TMR_U0_C0(ma_flag,</project>



5.14.7 R_PG_Timer_HaltCountElc_TMR_U<unit number>_C<channel number>

Definition	bool R_PG_Timer_HaltCountElc_TMR_U <unit number="">_C<channel number=""> (void)</channel></unit>		
	<unit number="">: 0 or 1</unit>		
	<i><channel number=""></channel></i> : 0 or 2		
Description	Stop the TMR o	peration that was started by the ELC	
Parameter	None		
Return value	true	Halting succeeded	
	false	Halting failed	
File for output	R_PG_Timer_T	MR_U <unit number="">.c</unit>	
	<unit number=""></unit>	>: 0 or 1	
RPDL function	R_TMR_Contro	olChannel	
Details		stops the TMR operation that was started in response to an event signal	
		. The TMR restarts counting when it receives the event signal from the ELC	
	again.		
Example	[TMR]		
	Unit 0: 8-bit tim	er mode	
	[ELC]		
	TMR0 operation when event is input: Counting is started		
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>		
	void func1(void)		
	{		
	R_PG_Clo	ck_Set(); //The clock-generation circuit has to be set first.	
R_PG_ELC_Set(); //Set up the event link controller (ELC). R_PG_ELC_SetLink_TMR0(); //Set an event link. R_PG_Timer_Start_TMR_U0_C0(); //Set up TMR			
		r_Start_TMR_U0_C0(); //Set up TMR	
	R_PG_ELC	_AllEventLinkEnable(); //Enable all event links.	
	}		
	void func2(void)	
	{ //Stop the TMR operation that was started by the ELC		
R_PG_Timer_HaltCountElc_TMR_U0_C0();		er_HaltCountElc_TMR_U0_C0();	
	}		



5.14.8 R_PG_Timer_GetCountStateElc_TMR_U<unit number>_C<channel number>

Definition	<pre>bool R_PG_Timer_GetCountStateElc_TMR_U<unit number="">_C<channel number=""></channel></unit></pre>		
	(bool * count_state)		
	 <unit number="">: 0 or 1</unit> <channel number="">: 0 or 2</channel> 		
Description	Acquire the state	of the TMR operation that was started by the ELC	
Parameter	bool *	Destination for storage of the state of the TMR operation	
	count_state	0: Count stopped state in response to ELC.	
		1: Count start state in response to ELC.	
Return value	true	Acquisition succeeded.	
	false	Acquisition failed.	
File for output	R_PG_Timer_T	MR_U <unit number="">.c</unit>	
	<unit number=""></unit>	: 0 or 1	
RPDL function	R_TMR_ReadC	hannel	
<u>Details</u>	This function a	acquires the state of the TMR operation that was started by the ELC.	
Example	[TMR]		
	Unit 0: 8-bit timer mode		
	[ELC]		
	TMR0 operation when event is input: Counting is started		
	#include "R_PG_default.h" //Include "R_PG_ <project name="">.h" to use this function.</project>		
bool count_			
	void func1(void)		
{			
	R_PG_Clock_Set(); //The clock-generation circuit has to be set first.		
R_PG_ELC_Set(); //Set up the event link controller (ELC). R_PG_ELC_SetLink_TMR0(); //Set an event link.			
		r_Start_TMR_U0_C0(); //Set up TMR	
	R_PG_ELC_AllEventLinkEnable(); //Enable all event links.		
	void func2(void)		
	{ //Acquire the state of the TMR operation that was started by the ELC		
	*	r_GetCountStateElc_TMR_U0_C0(&count_state);	
	}		



5.14.9 R_PG_Timer_StopModule_TMR_U<unit number>

Definition	<pre>bool R_PG_Timer_StopModule_TMR_U<unit number=""> (void) <unit number="">: 0 or 1</unit></unit></pre>	
Description	Shut down a TMR unit	
Parameter	None	
Return value	true	Shutting down succeeded.
	false	Shutting down failed.
File for output	R_PG_Timer_TMR_U <unit number="">: 0 or 1</unit>	
RPDL function	R_TMR_Destroy	
<u>Details</u>	TMR1 of unit 0 (or b called, both channels	d places it in the module-stop state per unit. If both TMR0 and oth TMR2 and TMR3 of unit 1) are running when this function is are stopped. Call the following function to stop a single channel. punt_TMR_U< <i>unit number</i> >_C< <i>channel number</i> >
Example	The 8-bit timer mode was selected for TMR0 in the GUI.	
	TmrCma0IntFunc was specified as the name of the compare match A interrupt function in the GUI.	
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void)</project></pre>	
	{ //Place TMR0 in the second s	ne 8-bit mode. rt_TMR_U0_C0();
	//Stop TMR unit 0	Processing in response to a compare match A interrupt



5.15 Compare Match Timer (CMT)

5.15.1 R_PG_Timer_Set_CMT_U<unit number>_C<channel number>

<u>Definition</u>	bool R_PG_Timer_Set <unit number="">: 0 or <channel number="">: 0</channel></unit>	
Description	Set up the CMT	
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_Timer_CMT_U <unit number="">: 0 and</unit>	
RPDL function	R_CMT_Create	
	 Releases the CMT from the module-stop and makes initial settings. R_PG_Timer_StartCount_CMT_U<<i>unit number></i>_C<<i>channel number></i> can be used to start the count operation. Function R_PG_Clock_Set must be called before any use of this function. Interrupts of the CMT are set by this function. When the name of the interrupt notification function has been specified in the GUI, if an interrupt occurs in the CPU, the function having the specified name will be called. Create the interrupt notification function as follows: void <<i>name of the interrupt notification function></i> (void) For the interrupt notification function, note the contents of this chapter end, Notes on Notification Functions. 	
<u>Example</u>	-	

RENESAS

5.15.2 R_PG_Timer_StartCount_CMT_U<unit number>_C<channel number>

<u>Definition</u>	bool R_PG_Timer_Sta <unit number="">: 0 or <channel number="">: 0</channel></unit>	
Description	Start or resume the CM	T count operation
Parameter	None	
Return value	True	Starting or resuming count succeeded.
	False	Starting or resuming count failed.
File for output	R_PG_Timer_CMT_U <unit number="">: 0 or</unit>	
RPDL function	R_CMT_Control	
<u>Details</u>	 Starts counting by a CMT. Resumes counting by a CMT that was halted by R_PG_Timer_HaltCount_CMT_U<<i>unit number</i>>_C<<i>channel number</i>>. 	
Example	Refer to the example of I	R_PG_Timer_Set_CMT_U <unit number="">_C<channel number=""></channel></unit>



5.15.3 R_PG_Timer_HaltCount_CMT_U<unit number>_C<channel number>

<u>Definition</u>	bool R_PG_Timer_Hal <unit number="">: 0 or <channel number="">: 0</channel></unit>	
Description	Halt the CMT count op	eration
Parameter_	None	
Return value	true	Halting succeeded.
	false	Halting failed.
File for output	R_PG_Timer_CMT_U <unit number="">: 0 or</unit>	
RPDL function	R_CMT_Control	
<u>Details</u>	function.	operation. To make the CMT resume counting, call the following ount_CMT_U <unit number="">_C<channel number=""></channel></unit>
<u>Example</u>	Refer to the example of I	R_PG_Timer_Set_CMT_U <unit number="">_C<channel number=""></channel></unit>



5.15.4 R_PG_Timer_GetCounterValue_CMT_U<unit number>_C<channel number>

<u>Definition</u>	<pre>bool R_PG_Timer_GetCounterValue_CMT_U<unit number="">_C<channel number=""> (uint16_t * data) <unit number="">: 0 or 1 <channel number="">: 0 to 3</channel></unit></channel></unit></pre>	
Description	Acquire the CMT counter value	
<u>Parameter</u>	uint16_t * data	Destination for storage of the counter value
Return value	true	Acquisition of the counter value succeeded.
	false	Acquisition of the counter value failed.
File for output	R_PG_Timer_CMT_U< <i>unit number</i> >: 0 or 1	ber>.c
RPDL function	R_CMT_Read	
Details	• Acquires the counter value of a	CMT.
<u>Example</u>	A case where the setting is made as • CMT unit 0 channel 0 was set u	
	<pre>* CMT unit 0 channel 0 was set up #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function. uint16_t data; void func1(void) { R_PG_Clock_Set(); //The clock-generation circuit has to be set first. R_PG_Timer_Set_CMT_U0_C0(); //Set up the CMT0 R_PG_Timer_StartCount_CMT_U0_C0(); //Start the count operation } void func2(void) { //Acquire the value of a CMT0 counter R_PG_Timer_GetCounterValue_CMT_U0_C0(&data); }</project></pre>	



5.15.5 R_PG_Timer_SetCounterValue_CMT_U<unit number>_C<channel number>

<u>Definition</u>	bool R_PG_Timer_SetCounterValu (uint16_t data) < <i>unit number</i> >: 0 or 1 < <i>channel number</i> >: 0 to 3	ie_CMT_U <unit number="">_C<channel number=""></channel></unit>
Description	Set the CMT counter value	
Parameter	uint16_t data	Value to be set to the counter
Return value	true	Setting of the counter value succeeded.
File for output	false R_PG_Timer_CMT_U< <i>unit number</i> < <i>unit number</i> >: 0 or 1	Setting of the counter value failed.
RPDL function	R_CMT_Control	
Details	• Set the counter value of a CMT.	
<u>Example</u>	 Set the counter value of a CMT. A case where the setting is made as follows. CMT unit 0 channel 0 was set up #include "R_PG_default.h" //Include "R_PG_<project name="">.h" to use this function.</project> void func1(void) R_PG_Clock_Set(); //The clock-generation circuit has to be set first. R_PG_Timer_Set_CMT_U0_C0(); //Set up the CMT0 R_PG_Timer_StartCount_CMT_U0_C0(); //Start the count operation void func2(void) //Set the value of a CMT0 counter R_PG_Timer_SetCounterValue_CMT_U0_C0(0); 	



5.15.6 R_PG_Timer_StopModule_CMT_U<unit number>

Definition	<pre>bool R_PG_Timer_StopModule_CMT_U<unit number=""> (void) <unit number="">: 0 or 1</unit></unit></pre>	
Description	Shut down the CMT un	it
Parameter	None	
Return value	true	Shutting down succeeded.
	false	Shutting down failed.
File for output	R_PG_Timer_CMT_U< <unit number="">: 0 or 1</unit>	
RPDL function	R_CMT_Destroy	
<u>Details</u> •	CMT1 of unit 0 (or b called, both channels	d places it in the module-stop state per unit. If both CMT0 and oth CMT2 and CMT3 of unit 1) are running when this function is are stopped. Call the following function to stop a single channel. ount_CMT_U< <i>unit number>_</i> C< <i>channel number></i>
Example	<pre>#include "R_PG_defaul void func(void) { R_PG_Clock_Set(R_PG_Timer_Set_ R_PG_Timer_Star } void Cmt0IntFunc(void) { func_cmt0(); //Proof //Stop the CMT un </pre>	0 was set up becified as the compare match interrupt notification function name t.h" //Include "R_PG_ <project name="">.h" to use this function.); //The clock-generation circuit has to be set first. _CMT_U0_C0(); //Set up the CMT0 tCount_CMT_U0_C0(); //Start the count operation) cessing in response to a compare match interrupt</project>



5.16 Realtime Clock (RTCc)

5.16.1 R_PG_RTC_Start

Definition	bool R_PG_RTC_Start (void)	
Description	Sets up the RTC and starts its counter	
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Create	
<u>Details</u>	 R_RTC_Create Sets up the alarm interrupt, cyclic interrupt, and 1-Hz clock output from the RTCOUT pin and starts the RTC's counter. Before calling this function, call R_PG_Clock_Set to set the clock. This function does not set the current time. When the alarm interrupt is to be used, call this function before R_PG_RTC_SetCurrentTime, which sets the current time. 	
	R_PG_RTC_WarmStart(); //Set up the RTC of warm start and start its counter }	



5.16.2 R_PG_RTC_WarmStart

Definition	bool R_PG_RTC_WarmStart (void)	
Description	Sets up the RTC of warm start and starts its counter	
Parameter_	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_CreateWarm	
Details	• Sets up the alarm interrupt, cyclic interrupt and starts the RTC's counter.	
	• Before calling this function, call R_PG_Clock_Set to set the clock.	
Example	Refer to the example of R_PG_RTC_Start.	



5.16.3 R_PG_RTC_Stop

Definition	bool R_PG_RTC_Stop (void)	
Description	Suspends counting by the RTC	
Parameter	None	
Return value	true	Counting was successfully suspended.
	false	Suspension failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Control	
Details	Suspends counting by	the RTC.
	To restart counting, call R_PG_RTC_Restart.	
Example // Include "R_PG_ <project name="">.h" to use this function #include "R_PG_default.h"</project>		
	<pre>#include "R_PG_default.h" void func1(void) { R_PG_Clock_Set(); // Set the clock. R_PG_RTC_Start(); // Set up the RTC and start its counter. } void func2(void) { R_PG_RTC_Stop (); // Suspend counting. } void func3(void) { R_PG_RTC_Restart(); // Restart counting. }</pre>	



5.16.4 R_PG_RTC_Restart

Definition	bool R_PG_RTC_Restart (void)	
Description	Restarts counting by the RTC	
Parameter	None	
Return value	true	Counting was successfully restarted.
	false	Restarting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Control	
Details	Allows the RTC to restart counting that was suspended by R_PG_RTC_Stop.	
Example	Refer to the example of R_PG_RTC_Stop.	



5.16.5 R_PG_RTC_SetCurrentTime

Definition	bool R_PG_RTC_SetCurrentTime		
	(uint8_t seconds,	uint8_t minutes, bool pm, uint8_t hours,	
	uint8_t day,	uint8_t month, uint16_t year)	
Description	Sets the current time	e	
Parameter	uint8_t seconds	Seconds (valid range of values: 0x00 to 0x59, as BCD values)	
	uint8_t minutes	Minutes (valid range of values: 0x00 to 0x59, as BCD values)	
	bool pm	a.m./p.m.	
		0: a.m.	
		1: p.m.	
	uint8_t hours	Hours (valid range of values in 24-hour mode: 0x00 to 0x23, as BCD	
		values; in 12-hour mode: 0x01 to 0x12, as BCD values)	
	uint8_t day	Date (valid range of values: 0x01 to the number of days in the	
		specified month, as BCD values)	
	uint8_t month	Month (valid range of values: 0x01 to 0x12, as BCD values)	
	uint16_t year	Year (valid range of values: 0x0000 to 0x9999, as BCD values)	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_Control		
Details	• Sets the current t	• Sets the current time.	
		day-of-the-week counter is is figured out from the specified values for	
	date, month, and	-	
		• When this function is called during counting, counting is suspended while the current time is set and resumed on completion of the settings.	
		uses even for items that are not to be used for the alarm interrupt.	
		, specify 0 for p.m.	
Example	The following settings	s have been made through the GUI.	
	• Set up an alarm i	•	
	Specify RtcAlml	IntFunc as the alarm interrupt notification function.	
	// Include "R_PG_< #include "R_PG_de	project name>.h" to use this function.	
	void func(void)		
	R PG Clock S	Set(); // Set the clock.	
		art(); // Set up the RTC and start its counter.	
		etCurrentTime(// Set the current time (03:44:55 on Nov. 22, 2000)	
	0x55, // 55		
	0x44, // 44 0, // a.m.		
	0x03, // 03		
	0x22, // 22nd 0x11, // November		
	UX11, // NO	venioer	



```
0x2000 // 2000
   );
    R_PG_RTC_SetAlarmTime( // Set the alarm time (03:45:00 on Nov. 22, 2000)
       0x00, // 00 seconds
       0x45, // 45 minutes
       0, // a.m.
       0x03, // 03 o'clock
       0xff, // Day of the week (0xff: Automatically calculated from the date)
       0x22, // 22nd
       0x11, // November
       0x2000 // 2000
    );
    R_PG_RTC_AlarmControl( // Enable the year, month, date, day of the week, hour,
                                // minute, and second alarms.
        1, // Enable the seconds alarm
        1, // Enable the minutes alarm
        1, // Enable the hours alarm
        1, // Enable the day-of-the-week alarm
        1, // Enable the date alarm
        1, // Enable the month alarm
        1 // Enable the year alarm
   );
}
void RtcAlmIntFunc(void)
{
    // Alarm interrupt processing
}
```



5.16.6 R_PG_RTC_GetStatus

<u>Definition</u>	uint8_t * hours, uint8_ uint16_t * year, bool * bool * adjustment, bool *	nt8_t * seconds, uint8_t * minutes, bool * pm, t * day_of_week, uint8_t * day, uint8_t * month, carry, bool * alarm, bool * period, * reset, bool * running)	
Description	Acquires information on the	e current state of the RTC	
Parameter	bool * hour_mode24	Destination for storage of the hour-mode information (0:12-hour mode, 1: 24-hour mode)	
	uint8_t * seconds	Destination for storage of the current seconds-counter value	
	uint8_t * minutes	Destination for storage of the current minutes-counter value	
	bool * pm	Destination for storage of the a.m./p.m. value	
	uint8_t * hours	Destination for storage of the current hours-counter value	
	uint8_t * day_of_week	Destination for storage of the current day-of-the-week counter value	
	uint8_t * day	Destination for storage of the current date-counter value	
	uint8_t * month	Destination for storage of the current month-counter value	
	uint16_t * year	Destination for storage of the current year-counter value	
	bool * carry	Destination for storage of the incrementation interrupt flag	
	bool * alarm	Destination for storage of the alarm interrupt flag	
	bool * period	Destination for storage of the cyclic interrupt flag	
	bool * adjustment	Destination for storage of the 30-second unit adjustment bit	
		(0: normal operation, 1: adjustment in progress)	
	bool * reset	Destination for storage of the reset bit	
	bool * running	(0: normal operation, 1: resetting in progress) Destination for storage of the start bit	
	boor · running	(0: clock stopped, 1: clock operating)	
Return value			
<u>Return value</u>	true	Acquisition succeeded.	
	false	Acquisition failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_Read		
Details	• Acquires information on the current state of the RTC.		
	• For the parameter that corresponds to each of the items you wish to acquire, specify the		
	address where the value is to be stored. For the items you do not wish to acquire, on the		
	• other hand, specify 0.		
	 The interrupt flag is cleared within this function. When the value of the incrementation interrupt flag is 1, the current time will change 		
	while the information is being acquired. Read the value again in such cases.		
Example			
<u></u>	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	void func1(void)		
	<pre>{ R_PG_Clock_Set(); // Set the clock. R_PG_RTC_Start(); // Set up the RTC and start its counter. }</pre>		
	void func2(void)		

RENESAS

{	do{		
	// A	cquire the va	lues of the current time and incrementation interrupt flag.
		R_PG_RTC	C_GetStatus(
		&hour_mod	de24,//24-hour mode
		&seconds,	// Seconds
		&minutes,	// Minutes
		±,	// a.m./p.m.
		&hours,	// Hours
		0,	// Day of the week
		0,	// Date
		0,	// Month
		0,	// Year
		&carry,	// Incrementation interrupt flag
		0,	// Alarm interrupt flag
		0,	// Cyclic interrupt flag
		0,	// 30-second unit adjustment bit
		0,	// Reset bit
		0	// Start bit
);		
	} while(carry);	
}			



5.16.7 R_PG_RTC_Adjust30sec

Definition	bool R_PG_RTC_Adjust30sec (void)	
Description	Performs 30-second unit adjustment	
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Control	
<u>Details</u>	Performs 30-second	unit adjustment (29 or fewer seconds are rounded down to 00
	seconds while 30 or more seconds are treated as 1 minute).	
<u>Example</u>	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>	
	void func1(void)	
	<pre>{ R_PG_Clock_Set(); // Set the clock. R_PG_RTC_Start(); // Set up the RTC and start its counter. } void func2(void) { R_PG_RTC_Adjust30sec(); // Perform 30-second unit adjustment. }</pre>	



5.16.8 R_PG_RTC_ManualErrorAdjust

Definition	bool R_PG_RTC_ManualErrorAdjust (int8_t cycle)	
Description	Corrects an error of the timer	
Parameter	int8_t cycle	Value (i.e. a number of subclock cycles) for use in correcting an error of the timer
		-63 to -1 : Put the timer back 0 to 63 : Put the timer forward
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Control	
<u>Details</u>		



5.16.9 R_PG_RTC_Set24HourMode

Definition	bool R_PG_RTC_Set24HourMode (void)		
Description	Places the RTC in 24-hour mode		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_Control		
Details	Places the RTC in 24	-hour mode.	
<u>Example</u>	// Include "R_PG_ <proj #include "R_PG_defaul</proj 	ect name>.h" to use this function. t.h"	
	<pre>void func(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); // Places the RTC in 24-hour mode. R_PG_RTC_Set24HourMode(); }</pre>		



5.16.10 R_PG_RTC_Set12HourMode

Definition	bool R_PG_RTC_Set12HourMode (void)		
Description	Places the RTC in 12-hour mode		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_Control		
Details	Places the RTC in 12	2-hour mode.	
<u>Example</u>	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	void func(void) {		
	R_PG_Clock_Set()	; // Set the clock.	
<pre>// Set up the RTC and start its counter. R_PG_RTC_Start();</pre>			
	<pre>R_PG_RTC_Start(); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(</pre>		



5.16.11 R_PG_RTC_AutoErrorAdjust_Enable

Definition	bool R_PG_RTC_AutoErrorAdjust_Enable (int8_t cycle)	
Description	Enables automatic correction of errors of the timer	
<u>Conditions for</u> output	Automatic correction of	of errors of the timer has been set up.
Parameter	int8_t cycle	Value (i.e. a number of subclock cycles) for use in correcting an error of the timer
		-63 to -1: Put the timer back 0 to 63: Put the timer forward
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Control	
Details	Enables automatic c	correction of errors of the timer.
	-	ects errors (earliness or lateness) of the timer due to the precision of
Ensurals	subclock oscillation	over each adjustment cycle selected through the GUI.
<u>Example</u>	// Include "R_PG_ <pre>// Include "R_PG_defau</pre>	<pre>oject name>.h" to use this function. ult.h"</pre>
	int8_t cycle=-60;	
	<pre>void func(void) {</pre>	
	R_PG_Clock_Set(); // Set the clock.
	<pre>// Set up the RTC and start its counter. R_PG_RTC_Start();</pre>	
	<pre>// Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle);</pre>	
	<pre>// Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(</pre>	



5.16.12 R_PG_RTC_AutoErrorAdjust_Disable

Description Disables automatic correction of errors of the timer Conditions for Automatic correction of errors of the timer has been set up. output Parameter None Return value Irrue Setting was made correctly. false Setting failed. File for output R_PG_RTC.c RPDL function R_RTC_Control Details Disables automatic correction of errors of the timer. Example // Include "R_PG_cproject name>.h" to use this function. #int8_t cycle=-60; void func1(void) { R_PG_RTC_Start(); // Set up the RTC and start its counter. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_Start(); // Set the current Time(0x55, // 55 seconds 0x44, // 44 minutes 0, // am. 0, x03, // 3 o 'clock 0x210/ 2000 ; >; ; void func2(void) ; // Disable automatic correction of errors of the timer. R_PG_RTC_battoErrorAdjust_Enable(); ;	Definition	bool R_PG_RTC_AutoErrorAdjust_Disable (void)		
output Parameter None Return value true Setting was made correctly. false Setting failed. Eile for output R_PG_RTC.c RPDL function R_RTC_Control Details Disables automatic correction of errors of the timer. Example // Include "R_PG_sproject name>.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) {	Description	Disables automatic correction of errors of the timer		
Parameter None Return value Irrue Setting was made correctly. false Setting failed. File for output R_PG_RTC.c RPDL function R_RTC_Control Details Disables automatic correction of errors of the timer. Example // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) { R_PG_RTC_Start(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Set the clock. // Enable automatic correction of errors of the timer. R_PG_RTC_CeturentTime(0x53, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x3200 // 2000 \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</project>	Conditions for	Automatic correction of	errors of the timer has been set up.	
Idd Detring with indic correctly? false Setting failed. File for output R_PG_RTC.c RPDL function R_RTC_Control Details Disables automatic correction of errors of the timer. Example // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 0'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); ; void func2(void) { (// Disable automatic correction of errors of the timer.</project>	-	None		
File for output R_PG_RTC.c RPDL function R_RTC_Control Details • Disables automatic correction of errors of the timer. Example // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000 ;; ; void func2(void) { // Disable automatic correction of errors of the timer.</project>	Return value	true	Setting was made correctly.	
RPDL function R_RTC_Control Details • Disables automatic correction of errors of the timer. Example // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_StcUrrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd); ; void func2(void) { // Disable automatic correction of errors of the timer.</project>		false	Setting failed.	
Details • Disables automatic correction of errors of the timer. Example // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); } void func2(void) { // Disable automatic correction of errors of the timer.</project>	File for output	R_PG_RTC.c		
Example // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" int8_t cycle=-60; void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o' clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); } void func2(void) { // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer. // Disable automatic correction of errors of the timer.</project>				
<pre>// include 'R_PG_clefault.h" int8_t cycle=-60; void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(</pre>		Disables automatic c	orrection of errors of the timer.	
<pre>void func1(void) { R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(</pre>	Example			
<pre>{ R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(</pre>		int8_t cycle=-60;		
R_PG_Clock_Set(); // Set the clock. // Set up the RTC and start its counter. R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); } void func2(void) { // Disable automatic correction of errors of the timer.		void func1(void)		
R_PG_RTC_Start(); // Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); } void func2(void) { // Disable automatic correction of errors of the timer.			; // Set the clock.	
R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); } void func2(void) { // Disable automatic correction of errors of the timer.				
R_PG_RTC_SetCurrentTime(R_PG_RTC_AutoErrorAdjust_Enable(cycle); // Set the current time (03:44:55 on November 22, 2000) R_PG_RTC_SetCurrentTime(0x55, // 55 seconds 0x44, // 44 minutes 0, // a.m. 0x03, // 3 o'clock 0x22, // 22rd 0x11, // November 0x2000 // 2000); } void func2(void) { // Disable automatic correction of errors of the timer.		
R_PG_RTC_AutoErrorAdjust_Disable();				
}		<pre>k_PG_KIC_AutoErrorAdjust_Disable(); }</pre>		



5.16.13 R_PG_RTC_AlarmControl

Definition	bool R_PG_RTC_AlarmControl	
	(bool sec_enable, bo	ool min_enable, bool hour_enable, bool day_of_week_enable,
	bool day_enable, bo	ool month_enable, bool year_enable)
Description	Enables or disables alar	rms
Conditions for	An alarm interrupt has	been set up.
<u>output</u>		
Parameter	bool sec_enable	Seconds alarm (1: enabled, 0: disabled)
	bool min_enable	Minutes alarm (1: enabled, 0: disabled)
	bool hour_enable	Hours alarm (1: enabled, 0: disabled)
	bool	Day-of-the-week alarm (1: enabled, 0: disabled)
	day_of_week_enable	
	bool day_enable	Date alarm (1: enabled, 0: disabled)
	bool month_enable Month alarm (1: enabled, 0: disabled)	
	bool year_enable	Year alarm (1: enabled, 0: disabled)
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_Control	
Details	• Enables or disables the seconds, minutes, hours, day-of-the-week, date, month, or year	
	alarms.	
Example	Refer to the example of R_PG_RTC_SetCurrentTime.	



5.16.14 R_PG_RTC_SetAlarmTime

Definition	bool R_PG_RTC_SetAlarmTime			
	(uint8_t seconds, uint8_t minutes, bool pm, uint8_t hours,			
	uint8_t day_of_	week, uint8_t day, uint8_t month, uint16_t year)		
Description	Sets the time for an	n alarm		
Conditions for	An alarm interrupt	has been set up.		
output				
Parameter	uint8_t seconds	Seconds (valid range of values: 0x00 to 0x59, as BCD values)		
	uint8_t minutes	Minutes (valid range of values: 0x00 to 0x59, as BCD values)		
	bool pm	a.m./p.m.		
		0: a.m.		
		1: p.m.		
	uint8_t hours	Hours (valid range of values in 24-hour mode: 0x00 to 0x23, as BCD		
	values; in 12-hour mode: 0x01 to 0x12, as BCD values)uint8_tDay of the week (valid range of values: 0x00 for Sunday to 0x06 for			
	day_of_week	Saturday)		
	When 0xff is specified, the day of the week is figured out from the			
		values for day, month, and year.		
	uint8_t day	Date (valid range of values: 0x01 to the number of days in the		
		specified month, as BCD values)		
	uint8_t month	Month (valid range of values: 0x01 to 0x12, as BCD values)		
	uint16_t year	Year (valid range of values: 0x0000 to 0x9999, as BCD values)		
Return value	true	Setting was made correctly.		
	false	Setting failed.		
File for output	R_PG_RTC.c			
RPDL function	R_RTC_Control			
Details	 Sets the time for an alarm. 			
	• Specify valid values even for items that are not to be used for an alarm interrupt.			
		le, specify 0 for p.m.		
Example	Refer to the example of R_PG_RTC_SetCurrentTime.			



5.16.15 R_PG_RTC_SetPeriodicInterrupt

Definition	bool R_PG_RTC_SetPeriodicInterrupt (float frequency)		
Description	Specifies the cycle for generating the cyclic interrupt		
Conditions for	The cyclic interrupt	has been set up.	
<u>output</u>			
Parameter	float frequency	Frequency for the interrupt (Hz; valid values: 0.5, 1, 2, 4, 16, 64, and 256)	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_Control		
Details	Changes the cycl	le for generating the cyclic interrupt.	
Example 7	The following settings	s have been made through the GUI.	
	Setting up the cyclic interrupt.		
	• Specifying RtcPrdIntFunc as the cyclic interrupt notification function.		
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	void func(void)		
	R_PG_Clock_Set(); // Set the clock. R_PG_RTC_Start(); // Set up the RTC and start its counter.		
	<pre>void RtcAlmIntFunc(void) { // Cyclic interrupt processing R_PG_RTC_SetPeriodicInterrupt(4); // Specify 1/4 second as the cycle for</pre>		



5.16.16 R_PG_RTC_ClockOut_Enable

Definition	bool R_PG_RTC_ClockOut_Enable (void)		
Description	Enables the clock output		
<u>Conditions for</u> output	Output of a 1-Hz clock signal from the RTCOUT pin has been enabled.		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_Control		
Details	Starts 1-Hz cloc	k output from the RTCOUT pin.	
Example 7	The following setting	has been made through the GUI.	
-	Enable 1-Hz clock output from the RTCOUT pin.		
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	void func1(void)		
	<pre>{ R_PG_Clock_Set(); // Set the clock. R_PG_RTC_Start(); // Set up the RTC and start its counter and the clock output.</pre>		
	}		
	void func2(void)		
	<pre>{ R_PG_RTC_ClockOut_Disable(); // Stop the clock output. } void func3(void) { R_PG_RTC_ClockOut_Enable(); // Restart the clock output. }</pre>		



5.16.17 R_PG_RTC_ClockOut_Disable

Definition	bool R_PG_RTC_ClockOut_Disable (void)		
Description	Disables the clock output		
Conditions for output	Output of a 1-Hz clock signal from the RTCOUT pin has been enabled.		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function Details Example	R_PG_RTC.c R_RTC_Control • Stops 1-Hz clock output from the RTCOUT pin. The following setting has been made through the GUI. • Enable 1-Hz clock output from the RTCOUT pin. // Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" void func1(void) { R_PG_Clock_Set(); // Set the clock. R_PG_RTC_Start(); // Set up the RTC and start its counter and the clock output. } void func2(void) { R_PG_RTC_ClockOut_Disable(); // Stop the clock output. } void func3(void) { R_PG_RTC_ClockOut_Enable(); // Restart the clock output.</project>		



5.16.18 R_PG_RTC_StartBinary

Definition	bool R_PG_RTC_StartBinary (void)		
Description	Sets up the RTC and starts its counter (binary count mode)		
Parameter	uint32_t count	Alarm count (valid range of values : 0x00000001 to 0xFFFFFFFF)	
	uint32_t mask	Alarm mask (valid range of values : f0x00000001 to 0xFFFFFFFF)	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_CreateBinary		
Details	• Sets up the alarm int	errupt, cyclic interrupt and starts the RTC's counter.	
	Before calling this f	unction, call R_PG_Clock_Set to set the clock.	
	 This function does not set the current time. When the alarm interrupt is to be used, call this function before R_PG_RTC_SetCurrentTimeBinary, which sets the current time. // Include "R_PG_<project name="">.h" to use this function.</project> #include "R_PG_default.h" 		
<u>Example</u>			
	void func(void)		
	{		
	// Set the clock. R_PG_Clock_Set())•	
		,	
	// Set up the RTC and start its counter.		
	R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF); }		



5.16.19 R_PG_RTC_StopBinary

Definition	bool R_PG_RTC_StopBinary (void)		
Description	Suspends counting by the RTC (binary count mode)		
Parameter	None		
Return value	true	Counting was successfully suspended.	
	false	Suspension failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlBinary		
<u>Details</u>	• Suspends counting by	y the RTC.	
	• To restart counting, c	call R_PG_RTC_RestartBInary.	
<u>Example</u>	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func1(void) { // Set the clock. R_PG_Clock_Set(); // Set up the RTC and start its counter. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF); } void func2(void) { R_PG_RTC_StopBinary (); // Suspend counting. } void func3(void) { R_PG_RTC_RestartBinary(); // Restart counting. }</pre>		



5.16.20 R_PG_RTC_RestartBlnary

Definition	bool R_PG_RTC_RestartBinary (void)		
Description	Restarts counting by the	Restarts counting by the RTC (binary count mode)	
Parameter	None		
Return value	true	Counting was successfully restarted.	
	false	Restarting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlBinary		
Details	• Allows the RTC to restart counting that was suspended by R_PG_RTC_StopBinary.		
Example	Refer to the example of R_PG_RTC_StopBinary.		



5.16.21 R_PG_RTC_SetCurrentTimeBinary

Definition	<pre>bool R_PG_RTC_SetCurrentTimeBinary(uint32_t count)</pre>		
Description	Sets the current time (binary count mode)		
Parameter	uint32_t count	Current count (valid range of values: 0x00000000 to 0xFFFFFFFF)	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlBin	ary	
Details	time is set and res	on is called during counting, counting is suspended while the current sumed on completion of the settings.	
Example	• Set up an alarm in	have been made through the GUI.	
		ntFunc as the alarm interrupt notification function.	
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	void func(void)		
	{ // Set the clock. R_PG_Clock_Set();		
	<pre>// Set up the RTC and start its counter. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF);</pre>		
	// Set the current count R_PG_RTC_SetCurrentTimeBinary(0x00001000);		
	<pre>// Set the alarm count and mask R_PG_RTC_SetAlarmTime(0x00002000, 0x0000FFFF); };</pre>		
	<pre>void RtcAlmIntFunc(void) { // Alarm interrupt processing</pre>		



5.16.22 R_PG_RTC_GetStatusBinary

<u>Definition</u>		sBinary t * alarm_count, uint32_t * alarm_mask n, bool * period, bool * reset, bool * running)
Description	•	e current state of the RTC (binary count mode)
Parameter_	uint32_t * count	Destination for storage of the current count
	uint32_t * alarm_count	Destination for storage of the alarm count
	uint32_t * alarm_mask	Destination for storage of the alarm mask
	bool * carry	Destination for storage of the incrementation interrupt flag
	bool * alarm	Destination for storage of the alarm interrupt flag
	bool * period	Destination for storage of the cyclic interrupt flag
	bool * reset	Destination for storage of the reset bit
		(0: normal operation, 1: resetting in progress)
	bool * running	Destination for storage of the start bit
		(0: clock stopped, 1: clock operating)
Return value	t rue	Acquisition succeeded.
	f alse	Acquisition failed.
<u>RPDL function</u> <u>Details</u>	 For the parameter that c address where the value other hand, specify 0. The interrupt flag is cleated. When the value of the intervalue of the inte	a the current state of the RTC. orresponds to each of the items you wish to acquire, specify the is to be stored. For the items you do not wish to acquire, on the ared within this function. accementation interrupt flag is 1, the current time will change being acquired. Read the value again in such cases.
Example		name>.h" to use this function.
	<pre>void func1(void) { // Set the clock. R_PG_Clock_Set(); // Set up the RTC and start its counter. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF); } void func2(void) { do{ // Acquire the values of the current count and incrementation interrupt flag. R_PG_RTC_GetStatusBinary(& & count, // Current count 0, // Alarm count 0, // Alarm mask & &</pre>	






5.16.23 R_PG_RTC_ManualErrorAdjustBinary

Definition	bool R_PG_RTC_ManualErrorAdjustBinary (int8_t cycle)		
Description	Corrects an error of the timer (binary count mode)		
Parameter	int8_t cycle	Value (i.e. a number of subclock cycles) for use in correcting an error of the timer	
		-63 to -1 : Put the timer back 0 to 63 : Put the timer forward	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlBinary		
<u>Details</u>	• Corrects an error (earliness or lateness) of the timer due to the precision of subclock oscillation.		
<u>Example</u>	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	int8_t cycle=-1;		
	void func(void) {		
	R_PG_Clock_Set()	; // Set the clock.	
	<pre>// Set up the RTC and start its counter. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF); }</pre>		
	void RtcPrdIntFunc(vo	id)	
	{ // Correct an error of	of the timer	
	R_PG_RTC_ManualErrorAdjustBinary(cycle); }		



5.16.24 R_PG_RTC_AutoErrorAdjustBinary_Enable

Definition	bool R_PG_RTC_AutoErrorAdjustBinary_Enable (int8_t cycle, int8_t period)	
Description	Enables automatic correction of errors of the timer (binary count mode)	
Conditions for	Automatic correction of	of errors of the timer has been set up.
output		· · · · · · · · · · · · · · · · · · ·
Parameter	int8_t cycle	Value (i.e. a number of subclock cycles) for use in correcting an error of the timer
		-63 to -1: Put the timer back 0 to 63: Put the timer forward
	int8_t period	Adjustment cycle for use in correcting an error of the timer
		8 : Every 8 seconds, 32 : Every 32 seconds
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
<u>RPDL function</u> <u>Details</u>	 R_RTC_ControlBinary Enables automatic correction of errors of the timer. Automatically corrects errors (earliness or lateness) of the timer due to the precision of subclock oscillation over each adjustment cycle specified. 	
<u>Example</u>		oject name>.h" to use this function.
	<pre>int8_t cycle=-60; int8_t period=32;</pre>	
	void func(void) {	
	R_PG_Clock_Set(); // Set the clock.
	<pre>// Set up the RTC and start its counter. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF);</pre>	
	<pre>// Enable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjustBinary_Enable(cycle, period);</pre>	
	<pre>// Set the current count R_PG_RTC_SetCurrentTimeBinary(0x00001000); } void func2(void) { // Disable automatic correction of errors of the timer. R_PG_RTC_AutoErrorAdjust_Disable(); }</pre>	



$5.16.25 \ \ R_PG_RTC_AutoErrorAdjustBinary_Disable$

<u>Definition</u>	bool R_PG_RTC_Autol	ErrorAdjustBinary_Disable (void)	
Description	Disables automatic correction of errors of the timer (binary count mode)		
Conditions for	Automatic correction of	Automatic correction of errors of the timer has been set up.	
<u>output</u>			
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlBinary		
Details	• Disables automatic correction of errors of the timer.		
Example	Refer to the example of R_PG_RTC_AutoErrorAdjustBinary_Enable.		



5.16.26 R_PG_RTC_SetAlarmTimeBinary

Definition	bool R_PG_RTC_SetAlarmTimeBinary(uint32_t count, uint32_t mask)	
Description	Sets the time for an alarm (binary count mode)	
Conditions for	An alarm interrupt	has been set up.
output		
Parameter	uint32 t count	Alarm count
	_	(valid range of values : 0x00000001 to 0xFFFFFFFF)
	uint32_t mask	Alarm mask
		(valid range of values : f0x00000001 to 0xFFFFFFFF)
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_RTC.c	
RPDL function	R_RTC_ControlBinary	
<u>Details</u>	• Sets the count and mask for an alarm.	
<u>Example</u>	Refer to the example of R_PG_RTC_SetCurrentTimeBinary.	



5.16.27 R_PG_RTC_SetPeriodicInterruptBinary

Definition	bool R_PG_RTC_SetPeriodicInterruptBinary (float frequency)			
Description	Specifies the cycle for	Specifies the cycle for generating the cyclic interrupt (binary count mode)		
<u>Conditions for</u> output	The cyclic interrupt h	The cyclic interrupt has been set up.		
Parameter		Frequency for the interrupt (Hz; valid values: 0.5, 1, 2, 4, 16, 64, and 256)		
<u>Return value</u>		Setting was made correctly. Setting failed.		
File for output	R_PG_RTC.c	R_PG_RTC.c		
<u>RPDL function</u> <u>Details</u> <u>Example</u>	 R_RTC_ControlBinary Changes the cycle for generating the cyclic interrupt. The following settings have been made through the GUI. Setting up the cyclic interrupt. Specifying RtcPrdIntFunc as the cyclic interrupt notification function. // Include "R_PG_<project name="">.h" to use this function.</project> #include "R_PG_default.h" void func(void) { // Set the clock. R_PG_Clock_Set(); // Set up the RTC and start its counter. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF); } void RtcAlmIntFunc(void) 			
	<pre>// Cyclic interrupt processing R_PG_RTC_SetPeriodicInterruptBinary(4); // Specify 1/4 second as the cycle for</pre>			



5.16.28 R_PG_RTC_ClockOutBinary_Enable

Definition	bool R_PG_RTC_ClockOutBinary_Enable (void)		
Description	Enables the clock output (binary count mode)		
<u>Conditions for</u> output	Output of clock sig	gnal from the RTCOUT pin has been enabled.	
Parameter	None		
Return value	True	Setting was made correctly.	
	False	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlB	inary	
<u>Details</u>	Starts clock out	put from the RTCOUT pin.	
<u>Example</u>	The following settin	g has been made through the GUI.	
	• Enable specifie	d clock output from the RTCOUT pin.	
	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	void func1(void)		
	{ // Set the clock		
	R_PG_Clock_		
	<pre>// Set up the RTC and start its counter and the clock output. R_PG_RTC_StartBinary(0x0000FFFF, 0x0000FFFF); } void func2(void)</pre>		
	<pre>{ R_PG_RTC_ClockOutBinary_Disable(); // Stop the clock output. }</pre>		
	void func3(void)		
	<pre>{ R_PG_RTC_ClockOutBinary_Enable(); // Restart the clock output. }</pre>		



5.16.29 R_PG_RTC_ClockOutBinary_Disable

Definition	bool R_PG_RTC_ClockOutBinary_Disable (void)		
Description	Disables the clock	Disables the clock output (binary count mode)	
Conditions for	Output of a clock s	ignal from the RTCOUT pin has been enabled.	
<u>output</u>			
Parameter	None		
	r		
<u>Return value</u>	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_RTC.c		
RPDL function	R_RTC_ControlBinary		
Details	Stops clock output from the RTCOUT pin.		
Example	Refer to the example of R_PG_RTC_ClockOutBinary_Enable.		



5.17 Independent Watchdog Timer (IWDTa)

5.17.1 R_PG_Timer_Start_IWDT

Definition	bool R_PG_Timer_Start_IWDT (void)	
Description	Sets up the IWDT and starts its timer	
Conditions for output	Register start mode is selected. (This function is not output if auto-start mode is selected. A macro for setting option function select registers is output to R_PG_MCU_OFS.c.)	
Parameter	None	
<u>Return value</u>	true false	Setting was made correctly Setting failed
File for output	R_PG_Timer_IWDT.c	
RPDL function	R_IWDT_Set	
<u>Details</u>	 This function sets up the IWDT and starts its counter. Before calling this function, call R_PG_Clock_Set to set the clock. 	
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) { //Set up the clocks R_PG_Clock_Set(); //Sets up the IWDT and starts its timer R_PG_Timer_Start_IWDT(); }</project></pre>	



5.17.2 R_PG_Timer_RefreshCounter_IWDT

Definition	bool R_PG_Timer_RefreshCounter_IWDT (void)	
Description	Refresh the counter	
Parameter	None	
Return value	true	Refreshing succeeded
	false	Refreshing failed
File for output	R_PG_Timer_IWDT.c	
RPDL function	R_IWDT_Control	
Details	 Refreshes the IWDT counter After starting the count operation, call this function to clear the counter before the counter underflow. 	
<u>Example</u>		



5.17.3 R_PG_Timer_GetStatus_IWDT

Definition	bool R_PG_Timer_GetStatus_IWDT(uint16_t * counter_val, bool * undf, bool * ref_err)		
Description	Acquires the status flag and count value of IWDT		
Parameter	uint16_t * counter_val	The address of storage area for the IWDT counter value	
	bool * undf	The address of storage area for the underflow flag	
	bool * ref_err	The address of storage area for the refresh error flag	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output RPDL function	R_PG_Timer_IWDT.c R_IWDT_Read		
Details	Acquires the IWDT status flag and counter value.The underflow flag shall be cleared in this function.		
<u>Example</u>	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" uint16_t counter_val; bool undf; bool ref_err;</project></pre>		
		atus flag and counter value s_IWDT(&counter_val, &undf, &ref_err);	



5.18 Serial Communications Interface (SCIe, SCIf)

5.18.1 R_PG_SCI_Set_C<channel number>

Definition	<pre>bool R_PG_SCI_Set_C<channel number=""> (void)</channel></pre>		
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
Description	Set up a SCI channel		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C <channe< td=""><td>l number>.c</td></channe<>	l number>.c	
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
RPDL function	R_SCI_Create, R_SCI_	_Set	
Details	Releases a SCI chann	nel from the module-stop state, makes initial settings.	
	• Function R_PG_Clock_Set must be called before calling this function.		
	• When the name of the notification function has been specified in the GUI, if		
	corresponding event occurs, the function having the specified name will be called. Create		
	the notification function as follows:		
	void <i><name function="" notification="" of="" the=""></name></i> (void)		
	For the notification function, note the contents of this chapter end, Notes on Notification		
	Functions.		
Example	SCI1 has been set in the GUI.		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>void func(void) {</pre>		
	R_PG_Clock_Set R_PG_SCI_Set_C }	<u> </u>	



5.18.2 R_PG_SCI_SendTargetStationID_C<channel number>

Definition	bool R_PG_SCI_SendTargetStationID_C< <i>channel number</i> > (uint8_t id)			
	<channel number="">: 1,</channel>			
Description		Transmits the ID code of the receiving station		
Conditions for		ission is selected for a SCI channel		
<u>output</u>	The multi-processer co communication mode	ommunications function is enabled in the asynchronous serial		
Parameter	uint8_t id	The ID to be transmitted (0 to 255)		
Return value	true	Transmission succeeded		
	false	Transmission failed		
File for output	R_PG_SCI_C <channel< td=""><td>number>.c</td></channel<>	number>.c		
	<channel number="">: 1,</channel>	5, 6, 9, 12		
RPDL function	R_SCI_Send			
<u>Details</u>	• Generates an ID transport station.	nission cycle to transmit the ID code of the destination receiving		
	• This function waits until the ID transmission cycle has been completed.			
<u>Example</u>	A case where the setting	is made as follows.		
	• The function of transm	ission is selected for a SCI1 channel		
	The multi-processer communications function is enabled in the asynchronous ser communication mode			
	• "Wait at the transmission function until all data has been transmitted" is selected as the data transmission method.			
//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>				
uint8_t data[] = "ABCDEFGHIJ";				
	void func(void)	void func(void)		
	{			
	R_PG_Clock_Set() R_PG_SCI_Set_C1	-		
		rgetStationID_C1(5); //Send ID code (ID:5) IData_C1(data, 10); //Send data		



5.18.3 R_PG_SCI_StartSending_C<channel number>

<u>Definition</u>	<pre>bool R_PG_SCI_StartSending_C<channel number=""> (uint8_t * data, uint16_t count) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>		
Description	Start the data transmission		
Conditions for	 The function of transmission is selected for a SCI channel in GUI. 		
	• "Notify the transmiss	ion completion of all data by function call" is selected as the data	
	transmission method		
<u>Parameter</u>	uint8_t * data	The start address of the data to be sent.	
	uint16_t count	The number of the data to be sent. Set this to 0 if the transmit data is a character string (ending with a null character).	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C< <i>channel</i>	number>.c <channel number="">: 1, 5, 6, 9, 12</channel>	
RPDL function	R_SCI_Send		
<u>Details</u>	• This function starts th		
	• This function is generated when "Notify the transmission completion of all data by		
	function call" is selected as the data transmission method in GUI. This function returns		
	immediately and the notification function having the specified name will be called when		
	the last byte has been sent. Create the notification function as follows:		
	void < <i>name of the notification function</i> > (void)		
	For the notification function, note the contents of this chapter end, Notes on Notification		
	Functions.		
	• The number of transmitted data can be acquired by R_PG_SCI_GetSentDataCount_C		
	<i><channel number=""></channel></i> . The transmission can be terminated by calling		
	R_PG_SCI_StopCommunication_C< <i>channel number></i> before all bytes have been sent.		
	• The count of transmitted characters will loop back to 0 if 65536 characters are sent.		
<u>Example</u>	SCI1 has been set as transmitter in the GUI.		
	Sci1TrFunc was specified as the name of the transmit end notification function in the GUI.		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data[255];		
	void func(void)		
	<pre>{ R_PG_Clock_Set(); //The clock-generation circuit has to be set first. R_PG_SCI_Set_C1(); //Set up SCI1. R_PG_SCI_StartSending_C1(data, 255); //Send 255 bytes of binary data. }</pre>		
	//Transmit end notification function that called when all bytes have been sent void Sci1TrFunc(void)		
	R_PG_SCI_StopM	Iodule_C1(); //Shut down the SCI1	

RENESAS

5.18.4 R_PG_SCI_SendAllData_C<channel number>

Definition	<pre>bool R_PG_SCI_SendAllData_C<channel number=""> (uint8_t * data, uint16_t count) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>			
Description	Transmit all data			
Conditions for	• The function of trans	smission is selected for a SCI channel in GUI.		
<u>output</u>	• Other than "Notify the second secon	ne transmission completion of all data by function call" is selected as		
-	the data transmission			
Parameter	uint8_t * data	The start address of the data to be sent.		
	uint16_t count	The number of the data to be sent.		
		Set this to 0 if the transmit data is a character string (ending with a null character).		
Return value	true	Setting was made correctly		
	false	Setting failed		
File for output	R_PG_SCI_C <channe <channel number="">: 1</channel></channe 			
RPDL function	R_SCI_Send	, , , , , , , 12		
Details	 This function transm 	its all data.		
Details		erated when other than "Notify the transmission completion of all data		
		elected as the transmission method in GUI. This function waits until		
	the last byte has been sent.			
	-	tted characters will loop back to 0 if 65536 characters are sent.		
Example	SCI1 has been set as tr	-		
		"Wait at the transmission function until the last byte has been transmitted" is selected as the		
	transmission method in GUI.			
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>			
	uint8_t data[255];			
	void func(void) {			
	R_PG_Clock_Set R_PG_SCI_Set_C	C1(); //Set up SCI1. AllData_C1(data, 255); //Send 255 bytes of binary data.		



5.18.5 R_PG_SCI_I2CMode_Send_C<channel number>

Definition	bool R_PG_SCI_I2CMode_Send_C< <i>channel number</i> >		
	(bool addr_10bit, uint1	6_t slave, uint8_t * data, uint16_t count)	
	<i><channel number=""></channel></i> : 1	, 5, 6, 9, 12	
Description	Transmit data by simpl	e I ² C bus interface	
Conditions for	• Simple I ² C bus interf	face is selected for "Mode".	
<u>output</u>			
Parameter_	bool addr_10bit	Slave address format (1: 10bit 0: 7bit)	
	uint16_t slave	Slave address	
	uint8_t * data	The start address of the data to be sent	
	uint16_t count	The number of the data to be sent	
Return value	true	When [Wait at the transmission function until all data has been	
		transmitted] was selected for data transmission method, the	
		operation completed OK. When except [Wait at the transmission	
		function until all data has been transmitted] is selected for data	
		transmission method, return value is always "true".	
	false	When [Wait at the transmission function until all data has been	
		transmitted] was selected for data transmission method, an error	
		was detected.	
File for output	R PG SCI C <channe< td=""><td>I number > c</td></channe<>	I number > c	
<u>i ne ioi ouput</u>	R_PG_SCI_C <channel number="">.c <channel number="">: 1, 5, 6, 9, 12</channel></channel>		
RPDL function	<crannet number="">: 1, 5, 6, 9, 12 R_SCI_IIC_Write</crannet>		
<u>Details</u>		nle I^2C bus interface	
Example	 Transmit data by simple I²C bus interface. [SCI1] 		
	Mode: Simple I2C mod	le	
	Data transmission method: Notify the transmission completion of all data by function call //Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data_tr[] = "AB uint16_t tr_count;	CDEFGHIJ";	
	void func(void) {		
	R_PG_Clock_Set();//The clock-generation circuit has to be set first.R_PG_SCI_Set_C1();//Set up SCI1.		
	//Transmit data by simple I ² C bus interface R_PG_SCI_I2CMode_Send_C1(0, 0x0006, data_tr, 10);		
	}		
	void Sci1TrFunc(void)		
	{ //Acquire the number of transmitted data		
	R_PG_SCI_GetSentDataCount_C1(&tr_count);		
	}		

RENESAS

5.18.6 R_PG_SCI_I2CMode_SendWithoutStop_C<*channel number*>

<u>Definition</u>		lode_SendWithoutStop_C< <i>channel number</i> > 6_t slave, uint8_t * data, uint16_t count)	
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
Description	Transmit data by simpl	e I ² C bus interface (no stop condition)	
Conditions for	• Simple I ² C bus interf	face is selected for "Mode".	
<u>output</u>			
Parameter	bool addr_10bit	Slave address format (1: 10bit 0: 7bit)	
	uint16_t slave	Slave address	
	uint8_t * data	The start address of the data to be sent	
	uint16_t count	The number of the data to be sent	
Return value	true	When [Wait at the transmission function until all data has been	
		transmitted] was selected for data transmission method, the	
		operation completed OK. When except [Wait at the transmission	
		function until all data has been transmitted] is selected for data	
		transmission method, return value is always "true".	
	false	When [Wait at the transmission function until all data has been	
		transmitted] was selected for data transmission method, an error	
		was detected.	
File for output	R_PG_SCI_C< <i>channe</i>		
	< <i>channel number></i> : 1, 5, 6, 9, 12		
<u>RPDL function</u>	$R_SCI_IIC_Write$		
<u>Details</u> Example	 Transmit data by simple I²C bus interface (no stop condition). [SCI1] 		
<u>.</u>	Mode: Simple I2C mod		
	-	nod: Notify the transmission completion of all data by function call	
	#include "R_PG_defau	ect name>.h" to use this function. lt.h"	
	uint8_t data_tr[10]; uint8_t data_re[10];		
	void func(void)		
	1 R_PG_Clock_Set(); //The clock-generation circuit has to be set first. R_PG_SCI_Set_C1(); //Set up SCI1.		
		simple I ² C bus interface (no stop condition) lode_SendWithoutStop_C1(0, 0x0006, data_tr, 10);	
	void Sci1TrFunc(void)		
	<pre>{ //Receive data by simple I²C bus interface (RE-START condition) R_PG_SCI_I2CMode_RestartReceive_C1(0, 0x0006, data_re, 10); }</pre>		

RENESAS

5.18.7 R_PG_SCI_I2CMode_GenerateStopCondition_C<channel number>

<u>Definition</u>	<pre>bool R_PG_SCI_I2CMode_GenerateStopCondition_C<channel number=""> (void) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>		
Description	Generate a stop condition		
Conditions for	• Simple I ² C bus interface is selected for "Mode".		
output	-		
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C <channel< td=""><td>number>.c</td></channel<>	number>.c	
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
RPDL function	R_SCI_Control		
Details	This function generat	es a stop condition.	
<u>Example</u>	[SCI1]		
	Mode: Simple I2C mode		
	Data transmission method: Transfer the transmitted serial data by DMAC		
	//Include "R_PG_ <proj #include "R_PG_defau</proj 	ect name>.h" to use this function. lt.h"	
	uint8_t data_tr[]="ABC	CDEFGHIJ";	
	void func(void)		
	R_PG_Clock_Set(); //The clock-generation circuit has to be set firs		
	<pre>//Set up a DMAC channel R_PG_DMAC_Set_C0();</pre>		
	<pre>//Set the source address R_PG_DMAC_SetSrcAddress_C0(data_tr);</pre>		
	<pre>//Make the DMAC be ready for the start trigger R_PG_DMAC_Activate_C0();</pre>		
	//Set up a SCI channel R_PG_SCI_Set_C1();		
	<pre>//Transmit data by simple I²C bus interface R_PG_SCI_I2CMode_Send_C1(0, 0x0006, data_tr, 10); }</pre>		
	void Dmac0IntFunc(void)		
	<pre>{ //Generate a stop condition R_PG_SCI_I2CMode_GenerateStopCondition_C1(); }</pre>		



5.18.8 R_PG_SCI_I2CMode_Receive_C<channel number>

Definition	<pre>bool R_PG_SCI_I2CMode_Receive_C<channel number=""></channel></pre>	
	(bool addr_10bit, uint16_t slave, uint8_t * data, uint16_t count)	
	<i><channel number=""></channel></i> : 1, 5, 6, 9, 12	
Description	Receive data by simple	I ² C bus interface
Conditions for	• Simple I ² C bus interf	face is selected for "Mode".
output		
Parameter	bool addr_10bit	Slave address format (1: 10bit 0: 7bit)
	uint16_t slave	Slave address
	uint8_t * data	The start address of the storage area for the expected data.
	uint16_t count	The number of the data to be received.
Return value	true	When [Wait at the reception function until all data has been
		received] was selected for data reception method, the operation
		completed OK. When except [Wait at the reception function until
		all data has been received] is selected for data reception method,
		return value is always "true".
	false	When [Wait at the reception function until all data has been
		received] was selected for data reception method, an error was
		detected.
File for output	R_PG_SCI_C <channe< td=""><td>l number>.c</td></channe<>	l number>.c
	<channel number="">: 1</channel>	, 5, 6, 9, 12
RPDL function	R_SCI_IIC_Read	
Details	• This function receives data by simple I^2C bus interface.	
Example	[SCI1]	
	Mode: Simple I2C mod	le
	Function selection: Transmission and reception //Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" uint8_t data_re[10];</project>	
	void func(void)	
	{	
	R_PG_Clock_Set); //The clock-generation circuit has to be set first.
	R_PG_SCI_Set_C	
	<pre>//Receive data by simple I²C bus interface R_PG_SCI_I2CMode_Receive_C1(0, 0x0006, data_re, 10); }</pre>	



5.18.9 R_PG_SCI_I2CMode_RestartReceive_C<channel number>

bool R_PG_SCI_I2CMode_RestartReceive_C <channel number=""></channel>		
(bool addr_10bit, uint1	6_t slave, uint8_t * data, uint16_t count)	
<i><channel number=""></channel></i> : 1	1, 5, 6, 9, 12	
	e I ² C bus interface (RE-START condition)	
• Simple I ² C bus inter	face is selected for "Mode".	
bool addr_10bit	Slave address format (1: 10bit 0: 7bit)	
uint16_t slave	Slave address	
uint8_t * data	The start address of the storage area for the expected data.	
uint16_t count	The number of the data to be received.	
true	When [Wait at the reception function until all data has been received] was selected for data reception method, the operation completed OK. When except [Wait at the reception function until all data has been received] is selected for data reception method, return value is always "true".	
false	When [Wait at the reception function until all data has been received] was selected for data reception method, an error was detected.	
R_PG_SCI_C <channe< td=""><td>el number>.c <channel number="">: 1, 5, 6, 9, 12</channel></td></channe<>	el number>.c <channel number="">: 1, 5, 6, 9, 12</channel>	
R SCI IIC Read		
 This function receives data by simple I²C bus interface. (RE-START condition) 		
Mode: Simple I2C mode		
//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
uint8_t data_re[10];	uint8_t data_re[10];	
<pre>void func(void) {</pre>		
R_PG_Clock_Set	(); //The clock-generation circuit has to be set first.	
R_PG_SCI_Set_0	C1(); //Set up SCI1.	
R_PG_SCI_I2CM 1, 0x0006, PDL_NO_P PDL_NO_D); //Receive data by R_PG_SCI_I2CM 0, //7 0x00f0, //S		
	(bool addr_10bit, uint1 <channel number="">: 1 Receive data by simple • Simple I²C bus inter bool addr_10bit uint16_t slave uint8_t * data uint16_t count true false R_PG_SCI_C<channel R_SCI_IIC_Read • This function receive [SCI1] Mode: Simple I2C mod //Include "R_PG_defat uint8_t data_re[10]; void func(void) { R_PG_SCI_Set_O //Iransmit data by R_PG_SCI_I2CM 1, 0x0006, PDL_NO_P PDL_NO_P PDL_NO_P); //Receive data by R_PG_SCI_I2CM 0, //T 0x00f0, //S</channel </channel>	

RENESAS

);
}	



5.18.10 R_PG_SCI_I2CMode_ReceiveLast_C<channel number>

Definition	bool R_PG_SCI_I2CMode_ReceiveLast_C <channel number=""> (uint8_t * data)</channel>		
	<i><channel number=""></channel></i> : 1, 5, 6, 9, 12		
Description	Making reception complete in simple I ² C bus interface		
Conditions for	• Simple I ² C bus interf	ace is selected for "Mode".	
output	• "Transfer the received	d serial data by DMAC" or "Transfer the received serial data by	
	DTC" is selected for	data reception method.	
Parameter	uint8_t * data	The start address of the storage area for the expected data.	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C <channel< td=""><td></td></channel<>		
	<i><channel number=""></channel></i> : 1		
RPDL function	R_SCI_IIC_ReadLastB		
Details		transferred by the DMAC or DTC in simple I ² C mode, this function	
	must be called to com		
		e called from a DMA interrupt notification function or receive end	
	notification function.		
Example	[SCI1]		
	Mode: Simple I2C mode		
	Data reception method:	Transfer the received serial data by DMAC	
	[DMAC0]		
	•	: RXI1 (SCI1 receive data full interrupt)	
	Transfer mode: Normal transfer mode		
	Length of a single data: 1 byte		
	Number of times: 4		
	Start address: 8a005h		
	Notify DMA interrupt (DMACIn)	
	[DMAC1]		
	Transfer request source	: TXI1 (SCI1 transmit data empty interrupt)	
	Transfer mode: Normal	transfer mode	
	Length of a single data:	1 byte	
	Number of times: 3		
	Source address update 1	node: Fixed	
	Start address: 8a003h		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data_re[5]; uint8_t dummy_data=0.	xFF;	
	void func(void) {		
	R_PG_Clock_Set(); //The clock-generation circuit has to be set first.	
	R_PG_SCI_Set_C	-	



R_PG_DMAC_Set_C0(); //Set up DMAC. R_PG_DMAC_Set_C1(); //Set up DMAC. R_PG_DMAC_SetDestAddress_C0(data_re); //Set the destination address. R_PG_DMAC_SetSrcAddress_C1(&dummy_data); //Set the source address. R_PG_DMAC_Active_C0(); //Make the DMAC be ready for the start trigger. R_PG_DMAC_Active_C1(); //Make the DMAC be ready for the start trigger. //Receive data by simple I²C bus interface. R_PG_SCI_I2CMode_Receive_C1(0, 0x0006, PDL_NO_PTR, 0); void Dmac0IntFunc(void) { //Making reception complete in simple I²C bus interface. R_PG_SCI_I2CMode_ReceiveLast_C1(&data_re[4]); }



5.18.11 R_PG_SCI_I2CMode_GetEvent_C<channel number>

Definition	<pre>bool R_PG_SCI_I2CMode_GetEvent_C<channel number=""> (bool * nack)</channel></pre>		
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
Description	Get the detected event in the simple I ² C mode		
Conditions for	Simple I ² C bus interfac	e is selected for "Mode".	
output			
Parameter	bool * nack	The address of the storage area for a NACK detection flag.	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_SCI_C <channe< td=""><td>l number>.c</td></channe<>	l number>.c	
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
RPDL function	R_SCI_GetStatus		
Details	This function acquire	es ACK Reception Data Flag in the simple I^2C mode.	
Example	[SCI1]		
	Mode:Simple I ² C mode	2	
	Data transmission method:Notify the transmission completion of all data by function call		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data_tr[]="ABCDEFGHIJ"; bool nack;		
	<pre>void func(void) {</pre>		
	R_PG_Clock_Set();//The clock-generation circuit has to be set first.R_PG_SCI_Set_C1();//Set up SCI1.		
		y simple I ² C bus interface lode_Send_C1(0, 0x0006, data_tr, 10);	
	void Sci1TrFunc(void)		
	<pre>{ //Get the detected event in the simple I²C mode R_PG_SCI_I2CMode_GetEvent_C1(&nack); }</pre>		



5.18.12 R_PG_SCI_SPIMode_Transfer_C<channel number>

Definition	<pre>bool R_PG_SCI_SPIMode_Transfer_C<channel number=""> (uint8_t * tx_start, uint8_t * rx_start, uint16_t count)</channel></pre>	
	<pre><channel number="">: 1, 5, 6, 9, 12</channel></pre>	
Description	Transmit data by simple	
	 Simple SPI mode is s 	
output	5	
Parameter	uint8_t * tx_start	The start address of the data to be transmitted.
	uint8_t * rx_start	The start address of the storage area for the expected data.
	uint16_t count	The number of the data to be transferred.
<u>Return value</u>	true	When [Wait at the transmission/reception function until all data has been transmitted/received] was selected for data transmission/reception method, the operation completed OK. When except [Wait at the transmission/reception function until all data has been transmitted/received] is selected for data transmission/reception method, return value is always "true".
	false	When [Wait at the transmission/reception function until all data has
		been transmitted/received] was selected for data
		transmission/reception method, an error was detected.
File for output	R_PG_SCI_C <channel <channel number="">: 1</channel></channel 	
RPDL function	R_SCI_SPI_Transfer	
Details	This function transmits data by simple SPI mode.	
Example	[SCI1]	
	Mode: Simple SPI mod	e
	Function selection: Transmission and reception	
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	uint8_t data_tr[10]; uint8_t data_re[10];	
	void func1(void)	
	R_PG_Clock_Set(R_PG_SCI_Set_C }	
	void func2(void)	
	{ //Transmit data by R_PG_SCI_SPIM }	simple SPI mode ode_Transfer_C1(data_tr, data_re, 10);



5.18.13 R_PG_SCI_SPIMode_GetErrorFlag_C<channel number>

Definition	<pre>bool R_PG_SCI_SPIMode_GetErrorFlag_C<channel number=""> (bool * overrun) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>		
Description	Get the serial reception error flag in the simple SPI mode		
Conditions for	Simple SPI mode is sele	ected for "Mode".	
<u>output</u>			
Parameter	bool * overrun	The address of the storage area for the overrun error flag.	
Return value	true	Acquisition of the flag succeeded	
	false	Acquisition of the flag failed	
File for output	R_PG_SCI_C <channel <channel number="">: 1</channel></channel 		
RPDL function	R_SCI_GetStatus		
Details		s the serial reception error flag in the simple SPI mode.	
	• Specify 0 for a flag that is not required.		
	• The flags of detected error will be set to 1.		
Example	[SCI1]		
	Mode: Simple SPI mode		
	Function selection: Tran	nsmission and reception	
	Notify receive error detection by function call		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t tx_data[4]; uint8_t rx_data[4]; bool overrun;		
	void func(void) {		
	R_PG_Clock_Set(R_PG_SCI_Set_C		
	<pre>//Transmit data by simple SPI mode R_PG_SCI_SPIMode_Transfer_C1(tx_data, rx_data, 4); }</pre>		
	void Sci1ErFunc(void) {		
	<pre>//Get the serial reception error flag in the simple SPI mode R_PG_SCI_SPIMode_GetErrorFlag_C1(&overrun); }</pre>		



5.18.14 R_PG_SCI_GetSentDataCount_C<channel number>

Definition	<pre>bool R_PG_SCI_GetSentDataCount_C<channel number=""> (uint16_t * count) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>		
Description	Acquire the number of transmitted data		
Conditions for	The function of transmi	ssion is selected for a SCI channel and "Notify the transmission	
output	completion of all data b	y function call" is selected as the data transmission method in GUI.	
Parameter	uint16_t * count	The storage location for the number of bytes that have been transmitted in the current transmission.	
Return value	true	Acquisition of the data count succeeded	
	false	Acquisition of the data count failed	
File for output	R_PG_SCI_C <channel number="">.c <channel number="">: 1, 5, 6, 9, 12</channel></channel>		
RPDL function	R_SCI_GetStatus		
Details	• When "Notify the tra	nsmission completion of all data by function call" is selected as the	
	data transmission me	thod in GUI, the number of transmitted data can be acquired by	
	calling this function.		
Example	SCI1 has been set as tra	insmitter in the GUI.	
	Sci1TrFunc was specified as the name of the transmit end notification function in the GUI.		
	//Include "R_PG_ <proj #include "R_PG_defau</proj 	ect name>.h" to use this function. lt.h"	
	uint8_t data[255];		
	void func(void) {		
	R_PG_Clock_Set();//The clock-generation circuit has to be set first.R_PG_SCI_Set_C1();//Set up SCI1.R_PG_SCI_StartSending_C1(data, 255);//Send 255 bytes of binary data.		
	}		
	//The transmit end notification function that called when all bytes have been sent void Sci1TrFunc(void)		
<pre>{ R_PG_SCI_StopModule_C1(); //Shut down the SCI1 }</pre>		fodule_C1(); //Shut down the SCI1	
	//The function to check the number of transmitted data and terminate the transmission void func_terminate_SCI(void)		
	{ uint16_t count;		
	// Acquire the number of transmitted data R_PG_SCI_GetSentDataCount_C1(&count);		
	<pre>if(count > 32){ R_PG_SCI_StopCommunication_C1(); //Terminate the transmission } }</pre>		



5.18.15 R_PG_SCI_ReceiveStationID_C<channel number>

<u>Definition</u>	bool R_PG_SCI_ReceiveStationID_C <channel number=""> (void)</channel>		
	<i><channel number=""></channel></i> : 1, 5, 6, 9, 12		
Description	Receives the ID code m	atches the ID of the receiving station itself	
Conditions for	• The function of recep	tion is selected for a SCI channel	
<u>output</u>	• The multi-processer communications function is enabled in the asynchronous serial communication mode		
Parameter	None		
Return value	true	Reception succeeded	
	false	Reception failed	
File for output	R_PG_SCI_C <channel< td=""><td>number>.c</td></channel<>	number>.c	
	< <i>channel number></i> : 1, 5, 6, 9, 12		
RPDL function	R_SCI_Receive		
Details	• This function waits until the ID code matches the ID of the receiving station itself h		
	been received.		
Example	A case where the setting is made as follows.		
•	• The function of reception is selected for a SCI1 channel		
•	• The multi-processer communications function is enabled in the asynchronous serial		
	communication mode		
• •		n completion of all data by function call" is selected as the data	
	reception method		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data[10];		
	<pre>void func(void) { R_PG_Clock_Set() R_PG_SCI_Set_C R_PG_SCI_Receiv R_PG_SCI_Receiv }</pre>	1(); //Set up SCI1	



5.18.16 R_PG_SCI_StartReceiving_C<channel number>

< <i>channel number</i> >: 1, 5, 6, 9, 12 Start the data reception		
a		
12		
inction		
liately		
st byte		
ication		
• The number of received data can be acquired by R_PG_SCI_GetReceivedDataCount_C < <i>channel number></i> . The reception can be terminated by calling		
ved.		
he GUI.		
//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
void func(void)		
lata		
lata.		
<pre>} //Receive end notification function that called when all bytes have been received void Sci1ReFunc(void)</pre>		



5.18.17 R_PG_SCI_ReceiveAllData_C<channel number>

Definition	bool R_PG_SCI_ReceiveAllData_C <channel number=""> (uint8_t * data, uint16_t count)</channel>		
	<channel number="">: 1,</channel>	5, 6, 9, 12	
Description	Receive all data		
Conditions for	• The function of recepti	ion is selected for a SCI channel in GUI.	
<u>output</u>	• Other than "Notify the	reception completion of all data by function call" is selected as the	
	data reception method	in GUI	
<u>Parameter</u>	uint8_t * data	The start address of the storage area for the expected data.	
	uint16_t count	The number of the data to be received.	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C <channel< td=""><td>number>.c</td></channel<>	number>.c	
	<channel number="">: 1,</channel>	5, 6, 9, 12	
RPDL function	R_SCI_Receive		
Details	• This function receives	all data.	
•	• This function is generated when other than "Notify the reception completion of all data by		
	function call" is selected as the data reception method in GUI. This function waits until the		
	last byte has been received.		
•	• The maximum number	of characters to be received is 65535.	
Example	SCI1 has been set as receiver in the GUI.		
	"Wait at the reception function until all data has been transmitted" is selected as the reception		
	method in GUI.		
	//Include "R_PG_ <proje #include "R_PG_default</proje 	ect name>.h" to use this function. t.h"	
	uint8_t data[255];		
<pre>void func(void) {</pre>			
	R_PG_Clock_Set() R_PG_SCI_Set_C	1();//Set up SCI1.veAllData_C1(data, 255);//Receive 255 bytes of binary	



5.18.18 R_PG_SCI_ControlClockOutput_C<channel number>

Definition	bool R_PG_SCI_ControlClockOutput_C <channel number=""> (bool output_enable)</channel>		
	< <i>channel number></i> : 1, 5, 6, 9, 12		
Description	Control the output from the SCKn pin (n: 1, 5, 6, 9, or 12)		
Conditions for	• "Smart card interface mode" is selected for mode.		
<u>output</u>	• "Enable (GSM mode)" is selected for GSM mode.	
	• "Output fixed high"	or "Output fixed low" is selected for SCKn pin function.	
Parameter	bool output_enable	Output from the SCKn pin (1: Clock output, 0: Output fixed)	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C <channel number="">.c <channel number="">: 1, 5, 6, 9, 12</channel></channel>		
RPDL function	R_SCI_Control		
<u>Details</u>	• This function control	s the clock output from the SCKn pin.	
Example	[SCI1]		
	Mode: Smart card interface mode		
	GSM mode: Enable		
	SCKn pin function: Output fixed high		
//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>			
	void func(void)		
	{		
	R_PG_Clock_Set(); //The clock-generation circuit has to be set first.	
	R_PG_SCI_Set_C	1(); //Set up SCI1.	
	<pre>//Control the output from the SCKn pin R_PG_SCI_ControlClockOutput_C1(1); }</pre>		



5.18.19 R_PG_SCI_StopCommunication_C<channel number>

Definition	R_PG_SCI_StopCommunication_C< <i>channel number</i> > (void)			
	<i><channel number=""></channel></i> : 1, 5, 6, 9, 12			
Description	Stop transmission and	Stop transmission and reception of serial data		
Parameter	None	None		
Return value	true	Setting was made correctly		
	false	Setting failed		
File for output	R_PG_SCI_C <channe< td=""><td>l number>.c</td></channe<>	l number>.c		
	<channel number="">: 1</channel>	1, 5, 6, 9, 12		
RPDL function	R_SCI_Control	R_SCI_Control		
Details	This function stops data transmission and reception.			
	• When "Notify the transmission completion of all data by function call" is selected as the			
	data transmission method in GUI, the reception can be terminated by calling this function			
	 before the number of bytes specified at R_PG_SCI_StartSending_C<<i>channel number></i> have been received. When "Notify the reception completion of all data by function call" is selected as the data 			
	reception method in GUI, the reception can be terminated by calling this function before the number of bytes specified at R_PG_SCI_StartReceiving_C< <i>channel number></i> have been received.			
<u>Example</u>	Refer to the example of R_PG_SCI_GetSentDataCount_C <channel number=""></channel>			



5.18.20 R_PG_SCI_GetReceivedDataCount_C<channel number>

Definition	<pre>bool R_PG_SCI_GetReceivedDataCount_C<channel number=""> (uint16_t * count) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>		
Description	Acquire the number of received data		
Conditions for	The function of reception	on is selected for a SCI channel and "Notify the reception completion	
<u>output</u>	of all data by function of	call" is selected as the data reception method in GUI.	
Parameter	uint16_t * count	The storage location for the number of bytes that have been received in the current reception process.	
Return value	true	Acquisition of the data count succeeded	
	false	Acquisition of the data count failed	
File for output	R_PG_SCI_C <channel number="">.c <channel number="">: 1, 5, 6, 9, 12</channel></channel>		
RPDL function	R_SCI_GetStatus		
<u>Details</u>	 When "Notify the reception completion of all data by function call " is selected as the receive end notification in GUI, the number of received data can be acquired by calling this function. 		
Example	SCI1 has been set as re-	ceiver in the GUI.	
	Sci1ReFunc was specified as the name of the receive end notification function in the GUI.		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data[255];		
	void func(void)		
	R_PG_Clock_Set(); //The clock-generation circuit has to be set first. R_PG_SCI_Set_C1(); //Set up SCI1. R_PG_SCI_StartReceiving_C1(data, 255); //Receive 255 bytes of binary data.		
	//The receive end notification function that called when all bytes have been received. void Sci1ReFunc(void)		
	{ R_PG_SCI_StopM }	fodule_C1(); //Shut down the SCI1	
	//The function to check the number of received data and terminate the reception void func_terminate_SCI(void)		
	{ uint16_t count;		
	<pre>//Acquire the number of received data R_PG_SCI_GetReceivedDataCount_C1(&count);</pre>		
	<pre>if(count > 32){ R_PG_SCI_StopCommunication_C1(); //Terminate the reception } }</pre>		
	L		



5.18.21 R_PG_SCI_GetReceptionErrorFlag_C<channel number>

Definition	bool R_PG_SCI_GetReceptionErrorFlag_C <channel number=""></channel>		
	(bool * parity, bool * framing, bool * overrun)		
	<i><channel number=""></channel></i> : 1, 5, 6, 9, 12		
Description	Get the serial reception	-	
Conditions for	The function of reception	on is selected for a SCI channel	
<u>output</u>			
Parameter_	bool * parity	The address of storage area for the parity error flag	
	bool * framing	The address of storage area for the framing error flag	
	bool * overrun	The address of storage area for the overrun error flag	
Return value	true	Acquisition of the flags succeeded	
	false	Acquisition of the flags failed	
File for output	R_PG_SCI_C <channel< td=""><td>l number>.c</td></channel<>	l number>.c	
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
RPDL function	R_SCI_GetStatus		
<u>Details</u>	• This function acquire	es the reception error flags.	
	• Specify the address o	f storage area for the flags to be acquired.	
	• Specify 0 for a flag that is not required.		
	• The flags of detected er		
<u>Example</u>	SCI1 has been set as re-		
ScilReFunc was specified as the name of the receive end notification function in the		ied as the name of the receive end notification function in the GUI.	
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" uint8_t data[255]; void func(void) </project></pre>		
	R_PG_Clock_Set(); //The clock-generation circuit has to be set first. R_PG_SCI_Set_C1(); //Set up SCI1. R_PG_SCI_StartReceiving_C1(data, 1); //Receive 1bytes of binary data.		
	//The receive end notification function that called when all bytes have been received. void Sci1ReFunc(void)		
	<pre>// Acquire the reception error flags R_PG_SCI_GetReceptionErrorFlag_C1(&parity, &framing, & overrun); }</pre>		



5.18.22 R_PG_SCI_ClearReceptionErrorFlag_C<channel number>

<u>Definition</u>	<pre>bool R_PG_SCI_ClearReceptionErrorFlag_C<channel number=""> (void)</channel></pre>		
	<i><channel number=""></channel></i> : 1, 5, 6, 9, 12		
Description	Clear the serial reception error flag		
Conditions for	"Asynchronous mode	", "Clock synchronous mode" or "Smart card interface mode" is	
<u>output</u>	selected for mode.		
	• "Reception" or "Tran	smission and reception" is selected for function selection.	
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_SCI_C <channel< td=""><td>number>.c</td></channel<>	number>.c	
	<channel number="">: 1</channel>	, 5, 6, 9, 12	
RPDL function	R_SCI_Control		
Details	 This function clears the serial reception error flag. 		
Example	Mode: Asynchronous mode		
	Function selection: Reception		
	Data reception method: Notify the reception completion of all data by function call		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint8_t data_re[10]; bool parity, framing, overrun;		
	void func(void)		
	R_PG_Clock_Set(); //The clock-generation circuit has to be set first.	
	R_PG_SCI_Set_C	1(); //Set up SCI1.	
	//Start the data reception R_PG_SCI_StartReceiving_C1(data_re, 10);		
	}		
	void Sci1ReFunc(void)		
	{ //Acquire the reception error flags R_PG_SCI_GetReceptionErrorFlag_C1(&parity, &framing, &overrun);		
	<pre>//Clear the serial reception error flag R_PG_SCI_ClearReceptionErrorFlag_C1(); }</pre>		



5.18.23 R_PG_SCI_GetTransmitStatus_C<channel number>

Definition	<pre>bool R_PG_SCI_GetTransmitStatus_C<channel number=""> (bool * complete) <channel number="">: 1, 5, 6, 9, 12</channel></channel></pre>		
Description	Get the state of transmission		
Conditions for	The function of transmi	ission is selected for a SCI channel	
<u>output</u>			
Parameter	bool * complete	The address of storage area for the transmission completion flag (0: Being transmitted 1:Complete)	
Return value	true	Acquisition of the transmission status succeeded	
	false	Acquisition of the transmission status failed	
File for output	R_PG_SCI_C <channel number="">.c <channel number="">: 1, 5, 6, 9, 12</channel></channel>		
RPDL function	R_SCI_GetStatus		
Details	• This function acquires the state of transmission.		
Example	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	bool complete; void func(void) {		
	//Get the state of transmission		
	R_PG_SCI_GetTransmitStatus_C1(&complete); }		


5.18.24 R_PG_SCI_StopModule_C<channel number>

Definition	<pre>bool R_PG_SCI_StopM</pre>	Iodule_C <channel number=""> (void) 5 6 9 12</channel>		
Description	Shut down a SCI chann			
Parameter	None			
Return value	true	Shutting down succeeded		
	false	Shutting down failed		
File for output	R_PG_SCI_C <channel< td=""><td>l number>.c</td></channel<>	l number>.c		
	<channel number="">: 1</channel>	, 5, 6, 9, 12		
RPDL function	R_SCI_Destroy			
Details	• Stops a SCI channel and places it in the module-stop state.			
Example	A case where the setting is made as follows.SCI1 has been set as receptor in the GUI.			
	• "Wait at the reception function until all data has been received" is selected as the data			
	reception method instead of specifying the receive end notification function name in GUI.			
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>			
	uint8_t data[255];			
	void func(void) {			
	R_PG_Clock_Set(R_PG_SCI_Set_C R_PG_SCI_Recei R_PG_SCI_StopM }	'1(); //Set up SCI1. veAllData_C1(data, 255); //Receive 255 bytes of binary data.		



5.19 I²C Bus Interface (RIIC)

5.19.1 R_PG_I2C_Set_C<channel number>

Definition	bool R_PG_I2C_Set_C	< <i>channel number</i> > (void)	
	<channel number="">: 0</channel>		
Description	Set up a I ² C bus interfa	ce channel	
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <channel <channel number="">: 0</channel></channel 		
RPDL function	R_IIC_Set, R_IIC_Create		
Details	• Releases an I^2C bus interface channel from the module-stop state, makes initial settings.		
	Function R_PG_Close	sk_Set must be called before any use of this function.	
Example	RIIC0 has been set in the	ne GUI.	
	//Include "R_PG_ <proj #include "R_PG_defau</proj 	ect name>.h" to use this function. lt.h"	
	void func(void) {		
	R_PG_Clock_Set(R_PG_I2C_Set_C }		

R20UT2489EJ0102 Rev.1.02 May 16, 2014



5.19.2 R_PG_I2C_MasterReceive_C<channel number>

Definition		erReceive_C< <i>channel number</i> > 16_t slave, uint8_t* data, uint16_t count) < <i>channel number</i> >: 0	
Description	Master data reception		
Conditions for	-	is selected for an I ² C bus interface channel in GUI.	
<u>output</u>			
Parameter	bool addr_10bit	Slave address format (1: 10bit 0: 7bit)	
	uint16_t slave	Target slave address	
	uint8_t* data	The start address of the storage area for the expected data.	
	uint16_t count	The number of the data to be received.	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <channel< td=""><td>l number>.c <channel number="">: 0</channel></td></channel<>	l number>.c <channel number="">: 0</channel>	
<u>RPDL function</u> <u>Details</u>	 R_PG_12C_C<channel number="">.c </channel> R_IIC_MasterReceive This function reads data from slave module. The stop condition is generated when the specified number of data has been received and reception completes. If "Wait at the reception function until all data has been transmitted" is selected as the master reception method in GUI, this function waits until the last byte has been received. If "Notify the reception completion of all data by function call" is selected as the master reception method in GUI, this function returns immediately and the notification function having the specified name will be called when the last byte has been receive. Create the notification function as follows: void <<i>name of the notification function></i> (void) For the notification function, note the contents of this chapter end, Notes on Notification functions. A Start condition will be generated automatically. If the previous transfer did not issue a stop condition, a repeated start condition will be generated. In the 7-bit address mode, [7:1] of specified slave address value will be output. In 10-bit address mode, [10:1] of specified slave address will be output. The number of received data can be aquired by R_PG_12C_GetReceivedDataCount_C <<i>channel number></i>. When using 10-bit address mode, select other than [Notify the reception completion of all data by function call] for master reception method in the GUI. 		
<u>Example</u>	A case where the settinThe function of mast	g is made as follows. ter is selected for a RIIC0 on function until all data has been transmitted" is selected as the	



//Include "R_PG_<project name>.h" to use this function. #include "R_PG_default.h" uint8_t iic_data[10]; //The storage area for the received data void func(void) { R_PG_Clock_Set(); //The clock-generation circuit has to be set first R_PG_I2C_Set_C0(); //Set up RIIC0 R_PG_I2C_MasterReceive_C0(//Master reception //Slave address format 0, 6, //Slave address // The start address of the storage area for the received data iic_data, // The number of the data to be received 10); R_PG_I2C_StopModule_C0(); //Stop RIIC0 }



5.19.3 R_PG_I2C_MasterReceiveLast_C<channel number>

<u>Definition</u>	(uint8_t* data)	rReceiveLast_C< <i>channel number</i> >	
Description Conditions for output	 < channel number >: 0 Complete a master reception process The function of master is selected for an I²C bus interface channel in GUI. Select DMAC or DTC transfer as a master reception method 		
Parameter	uint8_t* data	The address of the storage area for the expected data.	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <channel <channel number="">: 0</channel></channel 		
RPDL function	R_IIC_MasterReceiveI	Last	
Details	• This function is gene	etarted when [Transfer the received serial data by DMAC] or	
	[Transfer the receive	ed serial data by DTC] is selected as a master reception method.	
	• In the master recepti	on process that has used the DMAC or DTC transfer, NACK and	
	• stop condition will be issued by calling this function and the reception process will be		
	terminated.		
	• To complete reception process when the DMAC or DTC transfer completes, call this		
	function from DMAC or DTC interrupt notification function.		
	• Extra 1 byte is acquired from the receive data register in this function.		
		been detected during the reception process or the received data count	
	can be acquired by calling R_PG_I2C_GetEvent_Cn or		
El.	R_PG_I2C_GetReceivedDataCount_Cn.		
Example A case where the setting is made as follows.		-	
	• "Transfer the received serial data by DMAC" is selected as the master reception method in RIIC0 setting.		
	 DMAC0 is set as follows 		
	Transfer request source : ICRXI0(receive data full interrupt of TIIC0)		
	Transfer system : Single-operand transfer		
	Unit data size : 1 byte		
	Single operand data count : 1		
	Total transfer data size : Number of dtat to be received by RIIC0		
	Source start address : Address of RIIC0 received data register		
	Destination start address : Destination address of the data transfer		
	DMA interrupt notification fuction name : Dmac0IntFunc		
	//Include "R_PG_ <proj #include "R_PG_defau</proj 	ect name>.h" to use this function. lt.h"	
	void Dmac0IntFunc(){ uint8_t data; //Strag	ge area of extra data	
		TOP condition and complete the reception ReceiveLast(&data);	



} void func(void) { //The clock-generation circuit has to be set first R_PG_Clock_Set(); //Set up RIIC0 R_PG_I2C_Set_C0(); //Set up the DMAC0 R_PG_DMAC_Set_C0(); //Activate the DMAC0 R_PG_DMAC_Activate_C0(); //Master reception R_PG_I2C_MasterReceive_C0(0, //Slave address format 6. //Slave address PDL_NO_PTR, // For DMAC transfer, set PDL_NO_PTR // The number of the data (For DMAC transfer, set 0) 10); }



5.19.4 R_PG_I2C_MasterSend_C<channel number>

Definition		sterSend_C< <i>channel number</i> > int16_t slave, uint8_t* data, uint16_t count) < <i>channel number</i> >: 0	
Description	Master data transmis		
Conditions for		er is selected for an I^2C bus interface channel in GUI.	
<u>output</u>			
Parameter	bool addr_10bit	Slave address format (1: 10bit 0: 7bit)	
	uint16_t slave	Target slave address	
	uint8_t* data	The start address of the data to be sent	
	uint16_t count	The number of the data to be sent	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <chan< td=""><td>nel number>.c <channel number="">: 0</channel></td></chan<>	nel number>.c <channel number="">: 0</channel>	
RPDL function	R_IIC_MasterSend		
Details	• This function send	Is data to the slave module. The stop condition is generated when the	
	specified number of data has been transmitted and transmission completes.		
	• If "Wait at the transmission function until all data has been transmitted" is selected as the		
	data transmission method in GUI, this function waits until the last byte has been		
	transmitted or other events are detected.		
	• If "Notify the transmission completion of all data by function call" is selected as the data		
	transmission method in GUI, this function returns immediately and the notification		
	function having the specified name will be called when the last byte has been transmitted.		
	Create the notifica	tion function as follows:	
	void < <i>name of the notification function</i> > (void)		
	For the notification function, note the contents of this chapter end, Notes on Notification Functions.		
	• A Start condition will be generated automatically. If the previous transfer did not issue a stop condition, a repeated start condition will be generated.		
	 In the 7-bit address mode, [7:1] of specified slave address value will be output. In 10-bit 		
	address mode, [10:1] of specified slave address will be output. In 10-5h		
	 The number of transmitted data can be aquired by R_PG_I2C_GetSentDataCount_C 		
	channel number>.		
	• When using 10-bit address mode, select other than [Notify the transmission completion of		
	all data by function	n call] for master transmission method in the GUI.	
Example	A case where the setting is made as follows.		
	• The function of master is selected for a RIIC0		
	• "Wait at the transn	nission function until all data has been transmitted" is selected as the	
	data transmission	method	
	//Include "R_PG_ <pr< td=""><td>roject name>.h" to use this function.</td></pr<>	roject name>.h" to use this function.	
	#include "R_PG_defa	ault.h"	

RENESAS

```
\ensuremath{\textit{//}} The storage area for the data to be transmitted
uint8_t iic_data[10];
void func(void)
{
     //The clock-generation circuit has to be set first
     R_PG_Clock_Set();
     //Set up RIIC0
     R_PG_I2C_Set_C0();
     //Master transmission
    R_PG_I2C_MasterSend_C0(
        0,
              //Slave address format
        6,
              //Slave address
                     // The start address of the storage area for the data to be transmitted
        iic_data,
        10
              // The number of the data to be transmitted
    );
     //Stop RIIC0
    R\_PG\_I2C\_StopModule\_C0();
}
```



5.19.5 R_PG_I2C_MasterSendWithoutStop_C<channel number>

Definition		terSendWithoutStop_C< <i>channel number</i> > nt16_t slave, uint8_t* data, uint16_t count) < <i>channel number</i> >: 0	
Description		ion (No stop condition)	
Conditions for		er is selected for an I^2C bus interface channel in GUI.	
<u>output</u>			
Parameter	bool addr_10bit	Slave address format (1: 10bit 0: 7bit)	
	uint16_t slave	Target slave address	
	uint8_t* data	The start address of the data to be sent	
	uint16_t count	The number of the data to be sent	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <chann <channel number="">:</channel></chann 		
RPDL function	R_IIC_MasterSend		
Details	• This function sends	s data to the slave module. The stop condition will not be generated.	
	To generate a stop condition, call R_PG_I2C_GenerateStopCondition_C< <i>channel</i>		
	• number>.		
	If "Wait at the transmission function until all data has been transmitted" is selected as the		
	data transmission method in GUI, this function waits until the last byte has been		
	• transmitted or other events are detected.		
	If "Notify the transmission completion of all data by function call" is selected as the data		
	transmission method in GUI, this function returns immediately and the notification		
	function having the specified name will be called when the last byte has been transmitted. Create the notification function as follows:		
	 void <i><name function="" notification="" of="" the=""></name></i> (void) For the notification function, note the contents of this chapter end, Notes on Notification 		
	Functions.		
	• A Start condition will be generated automatically. If the previous transfer did not issue a		
	stop condition, a repeated start condition will be generated.		
	• In the 7-bit address mode, [7:1] of specified slave address value will be output. In 10-bit		
	address mode, [10:1] of specified slave address will be output.		
	The number of transmitted data can be aquired by R_PG_I2C_GetSentDataCount_C		
	<channel number="">.</channel>		
	• When using 10-bit	address mode, select other than [Notify the transmission completion of	
	all data by function	call] for master transmission method in the GUI.	
Example		ng is made as follows.	
	• The function of ma	ster is selected for a RIIC0	
	• "Notify the transmission completion of all data by function call" is selected as the data		
	transmission method		
	• IICOMasterTrFunc was specified as the name of the transmit end notification function		

RENESAS

```
//Include "R_PG_<project name>.h" to use this function.
#include "R_PG_default.h"
// The storage area for the data to be transmitted
uint8_t iic_data[10];
void func(void)
{
    //The clock-generation circuit has to be set first
    R_PG_Clock_Set();
    //Set up RIIC0
    R_PG_I2C_Set_C0();
    //Master transmission
    R_PG_I2C_MasterSendWithoutStop_C0(
                //Slave address format
         0,
         6,
                //Slave address
         iic_data,
                      /\!/ The start address of the storage area for the data to be transmitted
                // The number of the data to be transmitted
         10
    );
}
void IIC0MasterTrFunc(void){
    //Generate stop condition
    R_PG_I2C_GenerateStopCondition_C0();
    //Stop RIIC0
    R_PG_I2C_StopModule_C0();
}
```



5.19.6 R_PG_I2C_GenerateStopCondition_C<channel number>

Definition	bool R_PG_I2C_Gener <channel number="">: 0</channel>	rateStopCondition_C< <i>channel number</i> > (void)	
Description	Generate a stop condition		
Conditions for	-	is selected for an I^2C bus interface channel in GUI.	
output			
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <channel <channel number="">: 0</channel></channel 	l number>.c	
RPDL function	R_IIC_Control		
		tes a stop condition for the transmission started by	
Details	_	endWithoutStop_C< <i>channel number</i> >.	
Example	RIICO has been set in th	-	
<u>Example</u>			
	//Include "R_PG_ <proj #include "R_PG_defau</proj 	ect name>.h" to use this function. lt.h"	
	<pre>// The storage area for the data to be transmitted uint8_t iic_data[10];</pre>		
	<pre>void func(void) {</pre>		
	//The clock-generation circuit has to be set first R_PG_Clock_Set();		
	//Set up RIIC0 R_PG_I2C_Set_C0();		
	//Master transmission R_PG_I2C_MasterSendWithoutStop_C0(
	0, //Slave ad	ldress format	
	6, //Slave address		
	iic_data, // The start address of the storage area for the data to be transmitted 10 // The number of the data to be transmitted		
);		
	void IIC0MasterTrFunc(void)		
	//Generate stop condition R_PG_I2C_GenerateStopCondition_C0();		
	//Stop RIIC0		
	R_PG_I2C_StopModule_C0();		
	}		



5.19.7 R_PG_I2C_GetBusState_C<channel number>

Definition	bool R_PG_I20 <channel nun<="" th=""><th>C_GetBusState_C<<i>channel number</i>> (bool *busy) <i>nber</i>>: 0</th></channel>	C_GetBusState_C< <i>channel number</i> > (bool *busy) <i>nber</i> >: 0	
Description	Get the bus state		
Conditions for	The function of	f master is selected for an I ² C bus interface channel in GUI.	
<u>output</u>			
Parameter_	bool *busy	The address of storage area for the bus busy detection flag	
Return value	true	Acquisition of the flag succeeded	
	false	Acquisition of the flag failed	
File for output		cchannel number>.c	
DDDL function	<channel nun<="" td=""><td></td></channel>		
<u>RPDL function</u>	R_IIC_GetStat		
Details	Bus busy de	n acquires the bus busy detection flag.	
		The I ² C bus is released (bus free state)	
	1	The I^2C bus is occupied (bus busy state or in the bus free state)	
Example	RIIC0 has been	a set in the GUI.	
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	<pre>// The storage area for the data to be transmitted uint8_t iic_data[10]; //Storage for bus busy detection flag uint8_t busy;</pre>		
	void func(void)		
	<pre>{ //The clock-generation circuit has to be set first R_PG_Clock_Set();</pre>		
	//Set up R		
	// Wait for do{	the I ² C bus to be free	
		I2C_GetBusState_C0(& busy); (sy);	
	R_PG_I2C	cansmission _MasterSend_C0(/Slave address format	
		Slave address	
	iic_data 10 /	a, // The start address of the storage area for the data to be transmitted // The number of the data to be transmitted	
);		



5.19.8 R_PG_I2C_SlaveMonitor_C<channel number>

Definition	bool R_PG_I2C_Slav <channel number="">:</channel>	veMonitor_C< <i>channel number</i> > (uint8_t *data, uint16_t count) 0	
Description	Slave bus monitor		
Conditions for	The function of slave	is selected for an I ² C bus interface channel in GUI.	
<u>output</u>			
Parameter	uint8_t* data	The start address of the received data	
	uint16_t count	The number of the data to be received	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <chann <channel number="">:</channel></chann 		
RPDL function	R_IIC_SlaveMonitor		
Details	• This function mon	itors the accesses from master modules.	
	• If "Notify the rece	ption completion of all data, slave read request, or a stop condition	
	detection by function call" is selected as the slave monitor method in GUI, this function		
	returns immediately and the notification function having the specified name will be		
	called when a read access from master module or a stop condition is detected. Create the		
	notification function as follows:		
	void < name of the notification function> (void)		
	For the notification function, note the contents of this chapter end, Notes on Notification Functions.		
	• If "Wait at the monitor function until reception completion, slave read request, or a stop		
	condition detection" is selected as the slave monitor method in GUI, this function waits		
	until a read access from master module or a stop condition is detected.		
	• The received data from a master module is stored in the storage area of specified		
	address. Specify th	e number of data to not exceed the size of storage area. If the number	
	of the data from the master module exceeds the specified number, NACK shall be generated.		
	• The transmit/receive mode can be aquired by calling R_PG_I2C_GetRW_C< <i>channel number</i> >. The data can be transmitted by calling R_PG_I2C_SlaveSend_C< <i>channel</i>		
	<i>number></i> to respon	nd to a transmission (read) request from the master.	
	• Call R_PG_I2C_GetDetectedAddress_C <channel number=""> to acquire a detected slave</channel>		
	address. Call R_PG_I2C_GetEvent_C <channel number=""> to acquire the detected events</channel>		
	(e.g. a stop condition or a start condition).		
	• When using 10-bit address mode, select other than [Notify the transmission completion		
	of all data, slave re monitor method in	ad request, or a stop condition detection by function call] for slave	
Example		ing is made as follows.	
влатре			
	• The function of slave is selected for a RIIC0		
	IICOSlaveFunc wa	s specified as the name of the slave monitor function	

RENESAS

```
//Include "R_PG_<project name>.h" to use this function.
#include "R PG default.h"
// The storage area for the data to be received
uint8 t iic data re[10];
// The storage area for the data to be transmitted (slave address 0)
uint8_t iic_data_tr_0[10];
// The storage area for the data to be transmitted (slave address 1)
uint8_t iic_data_tr_1[10];
//Storage for bus busy detection flag
uint8_t busy;
void func(void)
{
    //The clock-generation circuit has to be set first
    R_PG_Clock_Set();
    //Set up RIIC0
    R_PG_I2C_Set_C0();
    // Slave monitor
    R_PG_I2C_SlaveMonitor_C0(
                          // The start address of the storage area for the received data
          iic_data_re,
         10
                //The number of the data to be received
    );
}
void IIC0SlaveFunc (void)
{
    bool transmit, start, stop;
    bool addr0, addr1;
    //Get the detected events
    R_PG_I2C_GetEvent_C0(0, &stop, &start, 0, 0);
    //Get an access type
    R_PG_I2C_GetTR_C0(&transmit);
    //Get a detected address
     R_PG_I2C_GetDetectedAddress_C0(&addr0, &addr1, 0, 0, 0, 0);
    if (start && transmit && address0) {
         R_PG_I2C_SlaveSend_C(
              iic_data_tr_0,
              10
         );
     }
    else if (start && read && address1) {
         R_PG_I2C_SlaveSend_C(
              iic_data_tr_1,
              10
         );
     }
}
```



5.19.9 R_PG_I2C_SlaveSend_C<channel number>

Definition	bool R_PG_I2C_Slaves <channel number="">: 0</channel>	Send_C< <i>channel number</i> > (uint8_t *data, uint16_t count)	
Description	Slave data transmission	Slave data transmission	
Conditions for	The function of slave is	selected for an I ² C bus interface channel in GUI.	
output			
Parameter	uint8_t* data	The start address of the data to be transmitted	
	uint16_t count	The number of the data to be transmitted	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_I2C_C <channel <channel number="">: 0</channel></channel 		
RPDL function	R_IIC_SlaveSend		
Details	• This function transmits the data to the master module.		
	• If the master requires more data than is supplied, this function shall loop back to the start of the data.		
Example	Refer to the example of	R_PG_I2C_SlaveMonitor_C <channel number=""></channel>	



5.19.10 R_PG_I2C_GetDetectedAddress_C<channel number>

Definition	<pre>bool R_PG_I2C_GetDetectedAddress_C<channel number=""></channel></pre>	
	(bool *addr0, bool *add	lr1, bool *addr2, bool *general, bool *device, bool *host)
	<channel number="">: 0</channel>	
Description	Get the detected addres	s
Conditions for	The function of slave is	selected for an I ² C bus interface channel in GUI.
<u>output</u>		
Parameter	bool *addr0	The address of storage area for slave address 0 detection flag
	bool *addr1	The address of storage area for slave address1 detection flag
	bool *addr2	The address of storage area for slave address 2 detection flag
	bool *general	The address of storage area for general call address detection flag
	bool *device	The address of storage area for device-ID command detection flag
	bool *host	The address of storage area for host address detection flag
Return value	true	Acquisition succeeded
	false	Acquisition failed
File for output	R_PG_I2C_C <channel< td=""><td>number>.c</td></channel<>	number>.c
	<channel number="">: 0</channel>	
RPDL function	R_IIC_GetStatus	
Details	This function acquire	es the detected address.
	• Specify the address of	of storage area for the flags to be acquired.
	• Specify 0 for a flag tha	t is not required.
	• 1 is set to detected add	ress
Example	Refer to the example of	R_PG_I2C_SlaveMonitor_C< <i>channel number</i> >



5.19.11 R_PG_I2C_GetTR_C<channel number>

Definition	<pre>bool R_PG_I2C_GetTR_PG_C<channel number=""> (bool * transmit)</channel></pre>			
Description		Get the transmit/receive mode		
Conditions for	The function of	slave is selected for an I ² C bus interface channel in GUI.		
<u>output</u>				
Parameter	bool * transmit	The address of storage area for the transmit mode flag		
Return value	true	Acquisition succeeded		
	false	Acquisition failed		
File for output	R_PG_I2C_C< <i>channel number</i> >.c			
RPDL function	< <i>channel number</i> >: 0 R IIC GetStatus			
	This function acquires the detected address.			
Detalls	 Specify the address of storage area for the flags to be acquired. 			
	1 0			
	 I is set to detected address. 			
	 This function acquires the the transmit/receive mode. 			
	Transmit mode flag			
	0	Receive mode		
	1	Transmit mode		
Example	Refer to the exa	mple of R_PG_I2C_SlaveMonitor_C <channel number=""></channel>		



5.19.12 R_PG_I2C_GetEvent_C<channel number>

Definition	<pre>bool R_PG_I2C_GetEvent_C<channel number=""></channel></pre>		
	(bool *nack, bool *stop, bool *start, bool *lost, bool *timeout)		
	<channel number="">: (</channel>	<channel number="">: 0</channel>	
Description	Get the detected event	Get the detected event	
Parameter	bool *nack	The address of storage area for a NACK detection flag	
	bool *stop	The address of storage area for a stop condition detection flag	
	bool *start	The address of storage area for a start condition detection flag	
	bool *lost	The address of storage area for an arbitration lost	
	bool *timeout	The address of storage area for a timeout detection	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
<u>File for output</u>	R_PG_I2C_C <channel number="">.c</channel>		
	<channel number="">: 0</channel>		
RPDL function	R_IIC_GetStatus		
Details	 This function acquires the detected event. 		
	• Specify 0 for a flag that is not required.		
	• 1 is set to detected event.		
Example	Refer to the example of R_PG_I2C_SlaveMonitor_C <channel number=""></channel>		



5.19.13 R_PG_I2C_GetReceivedDataCount_C<channel number>

Definition	<pre>bool R_PG_I2C_GetReceivedDataCount_C<channel number=""> (uint16_t *count) <channel number="">: 0</channel></channel></pre>	
Description	Acquires the count of received data	
Parameter	uint16_t *count	The address of storage area for the number of bytes that have been received
Return value	true	Acquisition of the data count succeeded
	false	Acquisition of the data count failed
File for output	R_PG_I2C_C <channel <channel number="">: 0</channel></channel 	number>.c
RPDL function	R_IIC_GetStatus	
<u>Details</u>	• This function acquire reception process.	es the number of bytes that have been received in the current
<u>Example</u>	A case where the setting	g is made as follows.
		er is selected for a RIIC0
•	• "Notify the reception	completion of all data by function call" is selected as the master
	reception method	
	//Include "R_PG_ <proj #include "R_PG_default</proj 	ect name>.h" to use this function. ht.h"
	<pre>// The storage area for the data to be received uint8_t iic_data[256];</pre>	
	<pre>// The storage area for t uint16_t count;</pre>	he number of received data
	void func(void)	
	<pre>{ //The clock-generation circuit has to be set first R_PG_Clock_Set();</pre>	
	//Set up RIIC0 R_PG_I2C_Set_C0();	
	<pre>//Master receive R_PG_I2C_MasterReceive_C0(0, //Slave address format 6, //Slave address iic_data, // The address of storage area for the data to be received 256 //The number of data to be received);</pre>	
	<pre>//Wait until 64 bytes have been received do{ R_PG_I2C_GetReceivedDataCount_C0(&count); } while(count < 64); }</pre>	



5.19.14 R_PG_I2C_GetSentDataCount_C<channel number>

<u>Definition</u>	<pre>bool R_PG_I2C_GetSentDataCount_C<channel number=""> (uint16_t *count) <channel number="">: 0</channel></channel></pre>		
Description	Acquires the count of transmitted data		
Parameter	uint16_t *count	The address of storage area for the number of bytes that have been transmitted	
Return value	true	Acquisition of the data count succeeded	
	false	Acquisition of the data count failed	
File for output	R_PG_I2C_C <channe <channel number="">: (</channel></channe 		
RPDL function	R_IIC_GetStatus		
<u>Details</u>	This function acquir (ICDRT).	res the number of data written in I ² C Bus Transmit Data Register	
	• 0 is acquired when t completed.	he number of transmission specified to the transmitting function is	
Example	A case where the settir	ng is made as follows.	
		ter is selected for a RIIC0	
	-	ssion completion of all data by function call" is selected as the data	
	transmission method		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	<pre>// The storage area for the data to be transmitted uint8_t iic_data[256];</pre>		
	<pre>// The storage area for the number of transmitted data uint16_t count;</pre>		
	<pre>void func(void) { //The clock-generation circuit has to be set first R_PG_Clock_Set(); //Set up RIIC0 R_PG_I2C_Set_C0();</pre>		
<pre>//Master send R_PG_I2C_MasterSend_C0(0, //Slave address format 6, //Slave address iic_data, // The address of storage area for the data to be transmitted 256 //The number of data to be transmitted);</pre>		e address format e address // The address of storage area for the data to be transmitted	
	<pre>//Wait until 64 bytes have been transmitted do{</pre>		
	<pre>} while(count < 64); }</pre>		

5.19.15 R_PG_I2C_Reset_C<channel number>

<pre>bool R_PG_I2C_Reset_C<channel number=""> (void)</channel></pre>		
Reset the bus		
None		
true Setting was made correctly.		
false Setting failed.		
R_PG_I2C_C< <i>channel number</i> >.c		
<channel number="">: 0</channel>		
R_IIC_Control		
• This function resets the module.		
• The settings of the module are preserved.		
A case where the setting is made as follows.		
• The function of master is selected for a RIIC0		
• "Notify the transmission completion of all data by function call" is selected as the data		
transmission method		
IIC0MasterTrFunc was specified as the name of the transmit end notification function		
<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
<pre>// The storage area for the data to be transmitted uint8_t iic_data[256];</pre>		
void func(void)		
{		
<pre>//The clock-generation circuit has to be set first R_PG_Clock_Set();</pre>		
//Set up RIIC0 R_PG_I2C_Set_C0();		
<pre>//Master send R_PG_12C_MasterSend_C0(0, //Slave address format 6, //Slave address iic_data, // The address of storage area for the data to be transmitted</pre>		
<pre>10 //The number of data to be transmitted); }</pre>		
<pre>void IIC0MasterTrFunc(void) { if (error){</pre>		



5.19.16 R_PG_I2C_StopModule_C<channel number>

<u>Definition</u>	<pre>bool R_PG_I2C_StopModule_C<channel number=""> (void) <channel number="">: 0</channel></channel></pre>		
Description	Shut down the I^2C bus interface channel		
Parameter_	None		
Return value	true	Shutting down succeeded.	
	false	Shutting down failed.	
File for output	R_PG_I2C_C <channel <channel number="">: 0</channel></channel 	number>.c	
RPDL function	R_IIC_Destroy		
Details	Stops an I ² C bus inte	rface channel and places it in the module-stop state.	
<u>Example</u>	A case where the setting	g is made as follows.	
•	The function of mast	er is selected for a RIIC0	
•	"Wait at the reception	n function until all data has been transmitted" is selected as the	
	master reception met	hod	
	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" // The storage area for the data to be transmitted uint8_t iic_data[256]; void func(void) { //The clock-generation circuit has to be set first R_PG_Clock_Set(); //Set up RIIC0 R_PG_I2C_Set_CO();</project></pre>		
	<pre>//Master receive R_PG_I2C_MasterReceive _CO(0, //Slave address format 6, //Slave address iic_data, // The address of storage area for the data to be received 10 //The number of data to be received); //Stop the RIIC0</pre>		
	R_PG_I2C_StopModule_C0(); }		



5.20 Serial Peripheral Interface (RSPI)

5.20.1 R_PG_RSPI_Set_C<channel number>

efinition	bool R_PG_RSPI_Set_C< <i>channel number</i> > (void)	
	<channel number="">: 0</channel>	
escription	Set up a RSPI channel	
arameter	None	
eturn value	true	Setting was made correctly
	false	Setting failed
le for output	R_PG_RSPI_C <channel< td=""><td>el number>.c</td></channel<>	el number>.c
	<channel number="">: 0</channel>	
PDL function	R_SPI_Create	
etails •	Releases a serial perip	pheral interface channel from the module-stop state, makes initial
	settings, and sets the	pins to be used.
•	Function R_PG_Cloc	k_Set must be called before calling this function.
•	The commands are no	ot set in this function. To set the commands, call
	R_PG_RSPI_SetCom	nmand_C< <i>channel number</i> >.
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	void func(void) {	
	R_PG_Clock_Set(); //Set up the clocks
	R_PG_RSPI_Set_C0(); //Set up RSPI0	
	R_PG_RSPI_SetCommand_C0(); //Set commands	
eturn value	true false R_PG_RSPI_C< <i>channa</i> < <i>channel number</i> >: 0 R_SPI_Create Releases a serial perip settings, and sets the p Function R_PG_Cloce The commands are no R_PG_RSPI_SetCom //Include "R_PG_defaul void func(void) { R_PG_Clock_Set(R_PG_RSPI_Set_O	Setting failed el number>.c pheral interface channel from the module-stop state, makes initial pins to be used. ck_Set must be called before calling this function. ot set in this function. To set the commands, call nmand_C <channel number="">. ect name>.h" to use this function. lt.h"); //Set up the clocks C0(); //Set up RSPI0</channel>



5.20.2 R_PG_RSPI_SetCommand_C<channel number>

<u>Definition</u>	bool R_PG_RSPI_SetCommand_C< <i>channel number</i> > (void)	
	< <i>channel number></i> : 0	
Description	Set commands	
Parameter	None	
Return value	true	Setting was made correctly
	false	Setting failed
File for output	R_PG_RSPI_C <channel number="">.c <channel number="">: 0</channel></channel>	
RPDL function	R_SPI_Command	
Details	• Set RSPI commands registers.	
	• All commands set in GUI (maximum number of commands: 8) shall be set.	
Example	Refer to the example of R_PG_RSPI_Set_C< <i>channel number</i> >	



5.20.3 R_PG_RSPI_StartTransfer_C<channel number>

<u>Definition</u>	<pre>Transmission and reception operations (Full-duplex synchronous serial communications) bool R_PG_RSPI_StartTransfer_C<channel number=""> (uint32_t * tx_start, uint32_t * rx_start, uint16_t sequence_loop_count) <channel number="">: 0 Serial communications consisting of only transmit operations bool R_PG_RSPI_StartTransfer_C<channel number=""> (uint32_t * tx_start, uint16_t sequence_loop_count) <channel number="">: 0</channel></channel></channel></channel></pre>		
Description Conditions for output	Start the data transfer "Notify the transfer completion and the error detection by function call" has been selected as the transfer method.		
Parameter	uint32_t * tx_start	The start address of the data to be transmitted.	
	uint32_t * rx_start	The start address of the storage area for the expected data.	
	uint16_t sequence_loop_count	The number of times that the command sequence will be executed	
Return value	true	Setting was made correctly	
	false	Setting failed	
File for output	R_PG_RSPI_C <channel number="">.c <channel number="">: 0</channel></channel>		
RPDL function	R_SPI_Transfer		
Details	• Starts the data transfer.		
	•	en "Notify the transfer completion and the error detection	
	•	the data transfer method in GUI.	
	• This function returns immediately and the notification function having the specified		
	name will be called when all commands are executed or error is detected.		
	Create the notification function as follows:		
	void < <i>name of the notification function</i> > (void)		
	For the notification function, note the contents of this chapter end, Notes on Notification		
	Functions.		
Example	A case where the setting is made as follows.		
	RSPI has been set to master mode		
	• "Notify the transfer completion and the error detection by function call" is selected as		
	• the transfer method		
	 rsi0_int_func is specified as a notification function name Number of commands: 1 Number of frames: 4 		
	Data lengh of command 0 is 8 bits //Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h" uint32_t tx_data[4] = { 0x11, 0x22, 0x33, 0x44 }; uint32_t rx_data[4] = { 0x00, 0x00, 0x00 };</project>		



```
bool over_run, mode_fault, parity_error;
void func(void)
{
    R_PG_Clock_Set();
                           //Set up the clocks
    R_PG_RSPI_Set_C0();
                               //Set up RSPI0
    R_PG_RSPI_SetCommand_C0();
                                        //Set commands
    R_PG_RSPI_StartTransfer_C0( tx_data, rx_data, 1 ); //Transfe 4 frames * 8bits
}
void rsi0_int_func (void)
{
    R_PG_RSPI_GetError_C0(&over_run, &mode_fault, &parity_error); //Get error flags
    if( over_run || mode_fault || parity_error ){
         //Processing when an error is detected
     }
    R_PG_RSPI_StopModule_C0();
}
```



5.20.4 R_PG_RSPI_TransferAllData_C<channel number>

<u>Definition</u>	Transmission and reception operations (Full-duplex synchronous serial communications) bool R_PG_RSPI_TransferAllData_C <channel number=""> (uint32_t * tx_start, uint32_t * rx_start, uint16_t sequence_loop_count) <channel number="">: 0 Serial communications consisting of only transmit operations bool R_PG_RSPI_TransferAllData_C<channel number=""> (uint32_t * tx_start, uint16_t sequence_loop_count) <channel number="">: 0 The DTC/DMAC transfer is selected for the transfer method bool R_PG_RSPI_TransferAllData_C<channel number=""> (uint16_t sequence_loop_count) <channel number="">: 0</channel></channel></channel></channel></channel></channel>		
Description Conditions for output	Transfer all data Other than "Notify the transfer comp selected as the transfer method.	letion and the error detection by function call" has been	
Parameter			
<u>r arameter</u>	uint32_t * tx_start uint32_t * rx_start	The start address of the data to be transmitted. The start address of the storage area for the expected data.	
	uint16_t sequence_loop_count	The number of times that the command sequence will be executed	
Return value			
Return value	true	Setting was made correctly	
<u>Return value</u>	true false	Setting was made correctly Setting failed	
<u>Return value</u> <u>File for output</u>			
	false R_PG_RSPI_C< <i>channel number</i> >.c		
File for output	false R_PG_RSPI_C< <i>channel number</i> >.c < <i>channel number</i> >: 0		
File for output RPDL function	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when o</channel></channel>		
File for output RPDL function	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when o</channel></channel>	Setting failed ther than "Notify the transfer completion and the error ted as the transmission method in GUI.	
File for output RPDL function	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when o detection by function call" is selected.</channel></channel>	Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI.	
File for output RPDL function Details	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when o detection by function call" is select This function waits until all comm A case where the setting is made as f RSPI has been set to master mode</channel></channel>	Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI. nands are executed. ollows.	
File for output RPDL function Details	false R_PG_RSPI_C< <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when o detection by function call" is select This function waits until all common A case where the setting is made as for the</channel></channel>	Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI. hands are executed. ollows.	
File for output RPDL function Details	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when or detection by function call" is select This function waits until all commands A case where the setting is made as f RSPI has been set to master mode "Wait until transfer completion" i Number of commands: 1</channel></channel>	Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI. hands are executed. follows. s selected as the transfer method. er of frames: 4	
File for output RPDL function Details	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when o detection by function call" is select This function waits until all comm A case where the setting is made as f RSPI has been set to master mode "Wait until transfer completion" i Number of commands: 1 Number of set to master</channel></channel>	Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI. hands are executed. ollows. s selected as the transfer method. er of frames: 4	
File for output RPDL function Details	false R_PG_RSPI_C <channel number="">.c <channel number="">: 0 R_SPI_Transfer Transfers all data. This function is generated when or detection by function call" is select This function waits until all commands A case where the setting is made as f RSPI has been set to master mode "Wait until transfer completion" i Number of commands: 1</channel></channel>	Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI. hands are executed. ollows. s selected as the transfer method. er of frames: 4	
File for output RPDL function Details	false R_PG_RSPI_C<	Setting failed Setting failed ther than "Notify the transfer completion and the error eted as the transmission method in GUI. hands are executed. collows. s selected as the transfer method. er of frames: 4 to use this function. (x33, 0x44 }; (x00, 0x00 };	



}

R_PG_Clock_Set(); //Set up the clocks
R_PG_RSPI_Set_CO(); //Set up RSPI0
R_PG_RSPI_SetCommand_CO(); //Set commands
R_PG_RSPI_TransferAllData_CO(tx_data, rx_data, 1); //Transfe 4 frames * 8bits
R_PG_RSPI_GetError_CO(&over_run, &mode_fault, &parity_error); //Get error flags
if(over_run || mode_fault || parity_error){
 //Processing when an error is detected
 }
R_PG_RSPI_StopModule_CO();



<u>Definition</u>	<pre>bool R_PG_RSPI_GetStatus_C<channel number=""> (bool * idle) <channel number="">: 0</channel></channel></pre>	
Description	Acquire the transfer status	
Parameter	bool * idle	The address of storage area for the idle flag (0: Idle state 1: Transfer state)
Return value	true	Acquisition succeeded
	false	Acquisition failed
File for output	R_PG_RSPI_C <channel r<br=""><channel number="">: 0</channel></channel>	number>.c
RPDL function	R_SPI_GetStatus	
<u>Details</u>	• Acquires the transfer sta	tus.
	• The error flags (the overrun error flag, the mode fault error flag, and the parity error flag) are cleared in this function. Call R_PG_RSPI_GetError_C< <i>channel number></i> to acquire the error flags before calling this function if needed.	
<u>Example</u>	<pre>the error flags before calling this function if needed. //Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" bool idle; void func(void) { do{ //Get the id R_PG_RSPI_GetStatus_C0(& idle); }while(idle); }</project></pre>	

5.20.5 R_PG_RSPI_GetStatus_C<channel number>



5.20.6 R_PG_RSPI_GetError_C<channel number>

Definition Description	<pre>bool R_PG_RSPI_GetError_C<channel number=""> (bool * over_run, bool * mode_fault, bool * parity_error) <channel number="">: 0 Acquire the error flags</channel></channel></pre>	
Parameter	bool * over_run bool * mode_fault	The address of storage area for the overrun error flag The address of storage area for the mode fault error flag
	bool * parity_error	The address of storage area for the parity error flag
Return value	true	Acquisition succeeded
	false	Acquisition failed
File for output	R_PG_RSPI_C <channel number="">.c <channel number="">: 0</channel></channel>	
RPDL function	R_SPI_GetStatus	
<u>Details</u>	• Acquires the error flags.	
	 Specify the address of storage area for the items to be acquired. Specify 0 for an item that is not required. The error flags shall be cleared in this function. 	
<u>Example</u>	Refer to the example of R_PG_RSPI_StartTransfer_C< <i>channel number</i> >, R_PG_RSPI_TransferAllData_C< <i>channel number</i> >, and R_PG_RSPI_GetCommandStatus_C< <i>channel number</i> >	



5.20.7 R_PG_RSPI_GetCommandStatus_C<channel number>

Definition Description Conditions for output	<pre>bool R_PG_RSPI_GetCommandStatus_C<channel number=""> (uint8_t * current_command, uint8_t * error_command) <channel number="">: 0 Acquire the command status A RSPI channel has been set to the master mode</channel></channel></pre>		
output			
Parameter	uint8_t * current_command	The address of storage area for the current command	
		pointer value (0 to 7)	
	uint8_t * error_command	The address of storage area for the value of command	
		pointer when an error is detected (0 to 7)	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_RSPI_C <channel number<br=""><channel number="">: 0</channel></channel>	r>.c	
RPDL function	R_SPI_GetStatus		
<u>Details</u>	Acquires the current command	Acquires the current command pointer value (0 to 7) and the value of command pointer	
	when an error is detected (0 to	7).	
	• Specify the address of storage area for the items to be acquired. Specify 0 for an item that		
	is not required.		
	• The error flags (the overrun error flag, the mode fault error flag, and the parity error flag)		
	are cleared in this function. Call R_PG_RSPI_GetError_C <channel number=""> to acquire</channel>		
	the error flags before calling this function if needed.		
Example	A case where the setting is made as follows.		
	RSPI has been set to the master mode		
	//Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	bool over_run, mode_fault, parity_error; uint8_t error_command;		
void func(void)			
	<pre>{ R_PG_RSPI_GetError_C0(&over_run, &mode_fault, &parity_error); //Get error flags if(over_run mode_fault parity_error){ R_PG_RSPI_GetCommandStatus_C0(0, &error_command); } }</pre>		
	// Processing when an	error is detected	
	}		



Definition	bool R_PG_RSPI_LoopBack <loopback mode="">_C<channel number=""> (void)</channel></loopback>		
	<loopback mode="">: Direct, Reversed, Disable</loopback>		
	<channel number="">: 0</channel>		
Description	Set loopback mode		
Conditions for	The loopback mode has been set		
<u>output</u>			
Parameter	None		
Return value	true	Setting was made correctly	
	false	Setting failed	

5.20.8 R_PG_RSPI_LoopBack<loopback mode>_C<channel number>

<u>File for output</u> R_PG_RSPI_C<*channel number*>.c <*channel number*>: 0

<u>RPDL function</u> R_SPI_Control

}

- Sets or disables RSPI pins to loopback mode.
- By calling R_PG_RSPI_LoopBackDirect_C<*channel number*>, the input path and output path for the shift register are connected. (transmit data = receive data)
- By calling R_PG_RSPI_LoopBackReversed _C<*channel number*>, the reversed input path and output path for the shift register are connected. (reversed transmit data = receive data)
- By calling R_PG_RSPI_LoopBackDisable_C<*channel number*>, the loopback mode is disabled.

Example

Details

//Include "R_PG_<project name>.h" to use this function.
#include "R_PG_default.h"
void func(void)
{

R_PG_RSPI_LoopBackDirect_C0(); //Set loopback mode



5.20.9 R_PG_RSPI_StopModule_C<channel number>

Definition	<pre>bool R_PG_RSPI_StopModule_C<channel number=""> (void)</channel></pre>		
	<channel number="">: 0</channel>		
Description	Shut down a RSPI channel		
Parameter	None		
Return value	true	Shutting down succeeded	
	false	Shutting down failed	
File for output	R_PG_RSPI_C <channel number="">.c <channel number="">: 0</channel></channel>		
RPDL function	R_SPI_Destroy		
Details	• Stops RSPI channel and places it in the module-stop state.		
Example	Refer to the example of R_PG_RSPI_StartTransfer_C <channel number=""> and</channel>		

R_PG_RSPI_TransferAllData_C<channel number>.



5.21 CRC Calculator (CRC)

5.21.1 R_PG_CRC_Set

Definition	bool R_PG_CRC_Set(void)		
Description	Set up CRC calculator		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_CRC.c		
RPDL function	R_CRC_Create		
Details	• Releases the CRC calculator from the module-stop state, makes initial settings.		
Example	<pre>//Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" uint16_t data;</project></pre>		
	<pre>void func(void) {</pre>		



5.21.2 R_PG_CRC_InputData

Definition	bool R_PG_CRC_InputData (uint8_t data)		
Description	Input a data to CRC calculator		
Parameter	uint8_t data	The data to be used for the calculation	
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_CRC.c		
RPDL function	R_CRC_Write		
Details	This function writes the data into the CRC calculation register		
Example	Refer to the example of R_PG_CRC_Set.		



5.21.3 R_PG_CRC_GetResult

Definition	bool R_PG_CRC_GetResult (uint16_t * data)		
Description	Get the result of calculation		
Parameter_	uint16_t * data	The address of the location where the result shall be stored.	
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_CRC.c		
RPDL function	R_CRC_Read		
<u>Details</u>	This function acquires the the result of calculation		
<u>Example</u>	Refer to the example of R_PG_CRC_Set.		


5.21.4 R_PG_CRC_StopModule

Definition	bool R_CRC_Destroy (uint16_t * data)		
Description	Shut down CRC calcula	Shut down CRC calculator	
Parameter	None		
Return value	true	Acquisition succeeded	
	false	Acquisition failed	
File for output	R_PG_CRC.c		
RPDL function	R_CRC_Destroy		
<u>Details</u>	Stops the CRC calculator and places it in the module-stop state.		
Example	Refer to the example of R_PG_CRC_Set.		



5.22 12-Bit A/D Converter (S12ADb)

5.22.1 R_PG_ADC_12_Set_S12AD0

Definition	bool R_PG_ADC_12_Set_S12AD0 (void)		
Description	Sets up the 12-bit A/D converter		
Parameter	None		
Return value	true	Setting was made correctly.	
	false	Setting failed.	
File for output	R_PG_ADC_12_S12A	D0.c	
RPDL function	R_ADC_12_Set, R_AD	C_12_CreateUnit, R_ADC_12_CreateChannel	
Details	• Releases the 12-bit A	D converter from the module-stop state, makes initial settings, and	
	places the converter	in the conversion-start trigger-input wait state. When the software	
	trigger is selected to	start conversion, conversion is started by calling	
	R_PG_ADC_12_Sta	rtConversionSW_S12AD0.	
	Before calling this function, call R_PG_Clock_Set to set the clock.		
	• The A/D-conversion end interrupt is set in this function. When the name of the interrupt		
	notification function has been specified in the GUI, the function having the specified name will be called when an interrupt request is conveyed to the CPU. Create the interrupt notification function as follows: void <name function="" interrupt="" notification="" of="" the=""> (void)</name>		
	For notes on interrup	t notification functions, refer to "Notes on Notification Functions"	
	provided at the end of this section.		
<u>Example</u>	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func(void) {</project></pre>		



5.22.2 R_PG_ADC_12_StartConversionSW_S12AD0

Definition Description Conditions for output Parameter	 bool R_PG_ADC_12_StartConversionSW_S12AD0(void) Starts A/D conversion (by a software trigger) The A/D converter is in single scan mode (not the double trigger mode) or continuous scan mode. None 	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_ADC_12_S12AI	D0.c
<u>RPDL function</u>	R_ADC_12_Control	
<u>Details</u>	Starts A/D conversio	n by an A/D converter for which the software trigger has been
	selected as the activation source.	
<u>Example</u>	The following setting has been made through the GUI.	
	Select the software trigger as the activation source.	
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
R_PG_ADC_12_Set_S12AD0(); // Set up the 12-bit A/D converter (S12A // A software trigger starts A/D conversion.		t_S12AD0(); // Set up the 12-bit A/D converter (S12AD0). starts A/D conversion.
	R_PG_ADC_12_StartConversionSW_S12AD0(); }	



5.22.3 R_PG_ADC_12_StopConversion_S12AD0

Definition	bool R_PG_ADC_12_StopConversion_S12AD0(void)		
Description	Stops A/D conversion	Stops A/D conversion	
Parameter	None		
Return value	true	Stopping conversion succeeded.	
	false	Stopping conversion failed.	
File for output	R_PG_ADC_12_S12AI	D0.c	
RPDL function	R_ADC_12_Control		
<u>Details</u>	Stops A/D conversion	n in the continuous scan mode. In other modes, this function need	
	not be called after A/	D conversion has ended.	
	After this function ha	as stopped A/D conversion, continuous scanning is resumed on	
	input of the A/D-con	version start trigger. To end continuous scanning, stop the A/D	
	conversion unit by calling R_PG_ADC_12_StopModule_S12AD0.		
Example	The following setting ha	as been made through the GUI.	
	Select the continuous	s scan mode as the operating mode.	
	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	<pre>void func1(void) { R_PG_Clock_Set(); // The clock-generation circuit has to be set first. R_PG_ADC_12_Set_S12AD0(); // Set up the 12-bit A/D converter (S12AD0). }</pre>		
	<pre>void func2(void) { // Stop continuous so R_PG_ADC_12_Sto }</pre>	canning. opConversion_S12AD0();	



5.22.4 R_PG_ADC_12_GetResult_S12AD0

Definition Description		GetResult_S12AD0(uint16_t * result) conversion of an analog input or internal reference voltage	
Parameter	uint16_t * result	Destination for storage of the result of A/D conversion	
Return value	true	Acquisition of the result succeeded.	
	false	Acquisition of the result failed.	
File for output	R_PG_ADC_12_S12A	D0.c	
RPDL function	R_ADC_12_Read		
Details	• At least two 16-byte	spaces are needed for storage of the acquired result.	
	• When A/D conversi	on is in progress at the time of calling this function and a name for	
	-	tion function has not been specified through the GUI, the function f A/D conversion before reading the result.	
Example		have been made through the GUI.	
	Select the group sca	-	
	Trigger for group	A: TRG4AN	
	Trigger for group	B: TRG4BN	
	Select the following	analog input pins.	
	Group A: AN000	and AN015	
	Group B: AN003	and AN006	
	 Specify S12ad0AInt 	Func as the A/D-conversion end interrupt notification function for	
	group A. Specify S12ad0BIntFunc as the A/D-conversion end interrupt notification function for		
group B.			
	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
	<pre>void func(void) {</pre>		
//AN000 and AN015			
		//AN000 and AN015	
	uint16_t result_an000; // Destination for storing the result of A/D conversion on //AN000		
		115; // Destination for storing the result of A/D conversion on //AN015	
	-	ts of A/D conversion for group A. etResult_S12AD0(result);	
	result_an000 = resu result_an015 = resu }		

RENESAS

// A/D-conversion end interrupt notification function for group B
void S12ad0BIntFunc(void)
{
uint16_t result[16]; // Destination for storing the result of A/D conversion on
//AN003 and AN006
uint16_t result_an003; // Destination for storing the result of A/D conversion on
//AN003
uint16_t result_an006; // Destination for storing the result of A/D conversion on
//AN006
// Acquire the results of A/D conversion for group B.
R_PG_ADC_12_GetResult_S12AD0(result);
$result_an003 = result[3];$
$result_an006 = result[6];$
}



5.22.5 R_PG_ADC_12_GetResult_DblTrigger_S12AD0

bool R_PG_ADC_12_GetResult_DblTrigger_S12AD0(uint16_t * result) Gets the result of A/D conversion in response to the second trigger in the double-trigger mode		
in the double-trigger mode.		
Destination for storage of the result of A/D conversion		
Acquisition of the result succeeded.		
Acquisition of the result failed.		
 R_PG_ADC_12_S12AD0.c R_ADC_12_Read Acquires the result of A/D conversion in response to the second trigger in the double-trigger mode. Data on one channel are acquired. 		
 When A/D conversion is in progress at the time of calling this function and a name for the interrupt notification function has not been specified through the GUI, the function waits until the end of A/D conversion before reading the result. The following settings have been made through the GUI. Select the double trigger mode (trigger: TRG4ABN). Select AN003 as an analog input pin. 		
Specify S12ad0AIntFunc as the A/D-conversion end interrupt notification function.		
<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h"</project></pre>		
 (); // The clock-generation circuit has to be set first. (); // The clock-generation circuit has to be set first. (); Set_S12AD0(); // Set up the 12-bit A/D converter (S12AD0). () interrupt notification function (void) () Destination for storing result 1 of A/D conversion on AN003 (003_2; // Destination for storing result 2 of A/D conversion on //AN003 of A/D conversion. GetResult_S12AD0(result); () of A/D conversion. GetResult_DblTrigger_S12AD0(&result_an003_2); 		



5.22.6 R_PG_ADC_12_GetResult_SelfDiag_S12AD0

Definition Description	bool R_PG_ADC_12_GetResult_SelfDiag_S12AD0(uint16_t * result) Gets the result of A/D conversion as part of self diagnosis by the A/D converter		
Parameter	uint16_t * result	Destination for storage of the result of A/D conversion	
Return value	true	Acquisition of the result succeeded.	
	false	Acquisition of the result failed.	
File for output	R_PG_ADC_12_S12AD0	·	
<u>RPDL function</u>	R_ADC_12_Read	<i></i>	
		A/D conversion performed as part of self discussis	
<u>Details</u>	•	A/D conversion performed as part of self diagnosis.	
	• When you use the self-diagnosis facility, self diagnosis takes place once at the beginning		
	within the A/D conver	ting with A/D conversion of one of the three voltages generated	
		A/D conversion includes self-diagnosis status information*, which	
	is in either of the follo	-	
	When the data placeme	ent selected through the GUI is right-alignment	
	b15-b14: Self-diagnosis status information*		
	b11-b0: Result of A/D conversion as part of self diagnosis		
	When the data placement selected through the GUI is left-alignment		
	b15-b4: Result of A/D conversion as part of self diagnosis		
	b1-b0: Self-diagnosis status information*		
	Note: The self-diagnos	sis status information has the following meanings.	
	b'00: Self diagnosis has not been performed.		
	b'01: Self diagnosis on 0[V] voltage has been performed.		
	b'10: Self diagnosis on VREFH0 \times 1/2 voltage has been performed.		
	b'11: Self diagnosis on VREFH0 voltage has been performed.		
Example	The following settings have been made through the GUI.		
<u>.</u>	• Select the single scan mode.		
	 Select AN000 and AN008 as analog input pins. 		
	 Select the software trigger as the activation source. 		
	 Select right-alignment for data placement. 		
	Enable the self-diagnosis facility.		
	 Specify S12ad0AIntFunc as the A/D-conversion end interrupt notification function. 		
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	_	<pre>// Destination for storing the result of A/D conversion as part of // self diagnosis</pre>	
	uint16_t adrd_ad;//uint16_t adrd_diagst;//uint16_t result[16];//	 // Sen diagnosis / Destination for storing the result of 12-bit A/D conversion // Destination for storing the self-diagnosis status information // Destination for storing the result of A/D conversion on AN000 // and AN008 // Destination for storing the result of A/D conversion on AN000 	

RENESAS

```
uint16_t result_an008; // Destination for storing the result of A/D conversion on AN008
void func(void)
{
   R_PG_Clock_Set();
                                 // The clock-generation circuit has to be set first.
   R_PG_ADC_12_Set_S12AD0(); // Set up the 12-bit A/D converter (S12AD0).
   // A software trigger starts A/D conversion.
   R_PG_ADC_12_StartConversionSW_S12AD0();
}
// A/D-conversion end interrupt notification function
void S12ad0AIntFunc(void)
{
   // Acquire the results of A/D conversion as part of self diagnosis.
   R_PG_ADC_12_GetResult_SelfDiag_S12AD0( &result_selfdiag );
   adrd_ad = (result_selfdiag & 0x0fff);
   adrd_diagst = (result_selfdiag >> 14);
   // Acquire the result of A/D conversion on AN000 and AN008.
   R_PG_ADC_12_GetResult_S12AD0( result );
   result_an000 = result[0];
   result_an008 = result[8];
```



5.22.7 R_PG_ADC_12_StopModule_S12AD0

Definition	bool R_PG_ADC_12_StopModule_S12AD0(void)		
Description	Shuts down the 12-bit A/D converter		
Parameter	None		
Return value	true	Shutting down succeeded.	
	false	Shutting down failed.	
File for output	R_PG_ADC_12_S12AI	D0.c	
RPDL function	R_ADC_12_Destroy		
Details	• Stops the 12-bit A/D	converter and places it in the module-stop state.	
<u>Example</u>	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>		
	uint16_t result[16]; // Destination for storage of the result of A/D conversion		
	<pre>void func1(void) { R_PG_Clock_Set(); // The clock-generation circuit has to be set first. R_PG_ADC_12_Set_S12AD0(); // Set up the 12-bit A/D converter (S12AD0). } void func2(void) { // Stop continuous scanning. R_PG_ADC_12_StopConversion_S12AD0();</pre>		
	<pre>// Acquire the result of A/D conversion. R_PG_ADC_12_GetResult_S12AD0(result);</pre>		
	<pre>// Stop the 12-bit A/D converter (S12AD0). R_PG_ADC_12_StopModule_S12AD0(); }</pre>		



5.23 Comparator A (CMPA)

5.23.1 R_PG_CPA_Set_CP<comparator circuit number>

Definition	<pre>bool R_PG_CPA_Set_CP<comparator circuit="" number=""> (void) <comparator circuit="" number="">: A1 or A2</comparator></comparator></pre>	
Description	Sets up comparator n	n: A1 or A2
Parameter	None	
Return value	true	Setting was made correctly.
	false	Setting failed.
File for output	R_PG_CP <comparato< td=""><td><i>r circuit number</i>>.c <<i>comparator circuit number</i>>: A1 or A2</td></comparato<>	<i>r circuit number</i> >.c < <i>comparator circuit number</i> >: A1 or A2
RPDL function	R_CPA_Create	
<u>Details</u>	• This function carries out initial setting of a comparator n. n: A1 or A2	
	• Before calling this fu	nction, call R_PG_Clock_Set to set the clock-generation circuit.
Example	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	void func(void)	
	<pre>{ R_PG_Clock_Set(); // The clock-generation circuit has to be set first.</pre>	
	<pre>// Sets up comparator A1. R_PG_CPA_Set_CPA1(); }</pre>	



5.23.2 R_PG_CPA_Disable_CP<comparator circuit number>

Definition	bool R_PG_CPA_Disable_CP< <i>comparator circuit number</i> > (void)			
	<comparator circuit="" number="">: A1 or A2</comparator>			
Description	Disable comparator n c	Disable comparator n circuit n: A1 or A2		
Parameter	None			
Return value	true	Setting was made correctly.		
	false	Setting failed.		
File for output	R_PG_CP <comparato< td=""><td><i>r circuit number>.c</i> <<i>comparator circuit number></i>: A1 or A2</td></comparato<>	<i>r circuit number>.c</i> < <i>comparator circuit number></i> : A1 or A2		
RPDL function	R_CPA_Control			
Details	• This function disable	s comparator n circuit. n: A1 or A2		
Example	<pre>// Include "R_PG_<project name="">.h" to use this function. #include "R_PG_default.h" void func1(void) { R_PG_Clock_Set(); // The clock-generation circuit has to be set first.</project></pre>			
<pre>// Sets up comparator A1. R_PG_CPA_Set_CPA1(); }</pre>				
	<pre>void func2(void) { // Disable comparator A1 circuit. R_PG_CPA_Disable_CPA1(); }</pre>			



5.23.3 R_PG_CPA_GetStatus

Definition	bool R_PG_CPA_GetStatus	
Description	(bool * cpa1_detect, bool * cpa1_monitor, bool * cpa2_detect, bool * cpa2_monitor) Get comparator A status flag	
Parameter	bool * cpa1_detect	The address of storage area for Comparator A1 Voltage Change Detection Flag
	bool * cpa1_monitor	The address of storage area for Comparator A1 Signal Monitor Flag
	bool * cpa2_detect	The address of storage area for Comparator A2 Voltage Change Detection Flag
	bool * cpa2_monitor	The address of storage area for Comparator A2 Signal Monitor Flag
Return value	true	Acquisition succeeded.
	false	Acquisition failed.
File for output RPDL function Details	 R_PG_CPA.c R_CPA_GetStatus This function acquires the status flag of Comparator A. 	
<u>Example</u>	 Specify 0 for a flag that is not required. The following settings have been made through the GUI. Use comparator A1. [The comparator An interrupt is generated when CMPAn has crossed the CVREFA] is selected as Comparator An mode. [Maskable interrupt] is selected as Comparator An interrupt type. 	
	// Include "R_PG_ <project name="">.h" to use this function. #include "R_PG_default.h"</project>	
	bool cpa1_mon;	
	<pre>void func(void) { R_PG_Clock_Set(); // The clock-generation circuit has to be set first. // Sets up comparator A1.</pre>	
	R_PG_CPA_Set_CPA1(); }	
	<pre>void CMPA1IntFunc(v { // Get comparator A R_PG_CPA_GetSt }</pre>	



5.24 Data Operation Circuit (DOC)

5.24.1 R_PG_DOC_Set

Definition	bool R_PG_DOC_Set (void)		
Description	Set up the Data Operation Circuit			
Parameter	None			
Return value	true	Setting was made correctly.		
	false	Setting failed.		
File for output	R_PG_DOC.c			
RPDL function	R_DOC_Create			
Details	• Releases the DOC fro	om the module-stop and makes initial settings.		
	• In Addition Mode an	interrupt is generated if the result of the addition exceeds FFFFh.		
	• In Subtraction Mode zero.	an interrupt is generated if the result of the subtraction is less than		
	• In Comparison Mode Mismatch) is met.	e an interrupt is generated when the comparison criteria (Match or		
	• After calling the interrupt notification function the DOC flag is automatically cleared.			
Example	A case where the setting is made as follows.			
	• [Data comparison mode] is selected for the operating mode			
		esult of data comparison] is selected for the detection condition		
	Comparison reference	-		
	-	becified as the interrupt notification function name		
		ect name>.h" to use this function.		
	#include "R_PG_defaul			
	#Include K_FO_delaul	1.11		
	uint16_t input_data[10]	$=\{1,0,0,1,0,0,0,0,0,1\};$		
	uint16_t comp_match_c	ent=0;		
	void func(void)			
	{			
	R_PG_DOC_Set()	; //Set up the data operation circuit		
		tData(input_data, 10); //Input data		
	}			
	//Data Operation Circui	t interrupt notification function		
	void DopcfIntFunc(void	1)		
	{			
	comp_match_cnt+	+;		
	}			



5.24.2 R_PG_DOC_GetStatusFlag

Definition	<pre>bool R_PG_DOC_GetStatusFlag (bool * status)</pre>			
Description	Acquire the status of the	e data operation circuit		
Parameter	bool * status	The address of the storage area for the status flag		
Return value	true	Acquisition of the flag succeeded.		
	false	Acquisition of the flag failed.		
File for output	R_PG_DOC.c			
RPDL function	R_DOC_Read			
Details	Acquires the status flag (the result of an operation) of the data operation circuit.			
	The status flag is set to 1 as follows:			
	In Comparison M	ode when the comparison criteria (Match / Mismatch) is met.		
	• In Addition Mode	e if the result of the addition exceeds FFFFh.		
	• In Subtraction Mode if the result of the subtraction is less than zero.			
	• If the DOC flag is se	If the DOC flag is set the flag is cleared after calling this function.		
<u>Example</u>	Refer to the example of	f R_PG_DOC_StopModule		



5.24.3 R_PG_DOC_GetResult

Definition	bool R_PG_DOC_Get	<pre>bool R_PG_DOC_GetResult (uint16_t * result)</pre>		
Description	Acquire the result of da	ata operation		
Parameter	uint16_t * result	The address of the storage area for the operation result		
Return value	true	Acquisition of the result succeeded.		
	false	Acquisition of the result failed.		
File for output	R_PG_DOC.c			
RPDL function	R_DOC_Read			
Details	• Acquires the value of	of DODSR (DOC Data Setting Register).		
	• The content of the a	cquired value of each operating mode is different as follows:		
	Data comparison n	node :Comparison reference		
	Data addition mod	e :The result of data addition		
	Data subtraction m	ode :The result of data subtraction		
Example	Refer to the example o	f R_PG_DOC_StopModule		



Definition	bool R_PG_DOC_Input	bool R_PG_DOC_InputData (uint16_t * data, uint16_t count)		
Description	Input data			
Parameter	uint16_t * data	The address of the storage area for the input data		
	uint16_t count	The number of the input data		
Return value	true	Setting was made correctly.		
	false	Setting failed.		
File for output	R_PG_DOC.c			
RPDL function	R_DOC_Write			
Details	• Data for the operation	on is set to DODIR (DOC Data Input Register).		
	Data comparison n	node :The compared data is set		
	Data addition mod	e :The added data is set		
	Data subtraction m	ode :The subtracted data is set		
Example	Refer to the example o	f R_PG_DOC_Set		

5.24.4 R_PG_DOC_InputData



5.24.5 R_PG_DOC_UpdateData

Definition Description	bool R_PG_DOC_Upda Update data	ateData (uint16_t data)
Parameter	uint16_t data	Data for update
<u>Return value</u>	true false	Setting was made correctly. Setting failed.
File for output RPDL function	R_PG_DOC.c R_DOC_Control	
<u>Details</u> Example	Data comparison m Data addition mode Data subtraction mo	R (DOC Data Setting Register) is updated to the specified data. ode :Comparison reference is updated : Initial value of addition result is updated ode :Initial value of subtraction result is updated s made as follows
Example	 [Detects match as a r Comparison reference DopcfIntFunc was sp //Include "R_PG_<proj.< li=""> #include "R_PG_defaul uint16_t input_data[10] uint16_t comp_match_comp_match_comp_match_comp_match_comp_match_comp_match_comp_match_0 = comp_match_0 = comp_match_1 = comp_mat</proj.<>	<pre>bde] is selected for the operating mode esult of data comparison] is selected for the detection condition e is 0 becified as the interrupt notification function name ect name>.h" to use this function. It.h" ={1,0,0,1,0,0,0,0,0,1}; cnt=0; 0, comp_match_1; ; //Set up the data operation circuit tData(input_data, 10); //Input data comp_match_cnt; ateData(1); //Update data tData(input_data, 10); //Input data comp_match_cnt - comp_match_0; t interrupt notification function d)</pre>
	t comp_match_cnt+	+;



5.24.6 R_PG_DOC_StopModule

<u>Definition</u> <u>Description</u> <u>Parameter</u>	bool R_PG_DOC_StopModule (void) Disable the data operation circuit None					
Return value	true	Setting was made correctly.				
	false	Setting failed.				
File for output	R_PG_DOC.c					
RPDL function	R_DOC_Destroy					
Details	• Enable the DOC mod	dule stop state.				
Example	A case where the setting i	s made as follows.				
	• [Data addition mode] is selected for the operating mode				
	• Initial value of additi	on or subtraction result is 0				
	//Include "R_PG_ <proj< td=""><td>ect name>.h" to use this function.</td></proj<>	ect name>.h" to use this function.				
	#include "R_PG_defaul	t.h"				
	uint16_t result;					
	uint16_t data=0x0000;					
	void func(void)					
	t bool status;					
	//Set up the data op					
	R_PG_DOC_Set()	;				
	while(1){					
	//Input data					
	R_PG_DOC_	InputData(&data, 1);				
	//Acquire the	status of the data operation circuit				
	_	GetStatusFlag(&status);				
	if(status == tr					
	brea	ak;				
	}					
	//Acquire the	result of data operation				
	R_PG_DOC_	GetResult(&result);				
	data++;					
	}					
	//Disable the data of R_PG_DOC_Stop	-				
	J					



5.25 Notes on Notification Functions

5.25.1 Interrupts and processor mode

The RX CPU has two processor modes; supervisor and user. The driver functions will be executed by the CPU in user mode. However any notification functions which are called by the interrupt handlers in Renesas Peripheral Driver Library will be executed by the CPU in supervisor mode. This means that the privileged CPU instructions (RTFI, RTE and WAIT) can be executed by the notification function and any function that is called by the notification function.

The user must:

- Avoid using the RTFI and RTE instructions. These instructions are issued by the API interrupt handlers, so there should be no need for the user's code to use these instructions.
- Use the wait() intrinsic function with caution.
 This instruction is used by some API functions as part of power management, so there should be no need for the user's code to use this instruction.

More information on the processor modes can be found in §1.4 of the RX Family software manual.

5.25.2 Interrupts and DSP instructions

The accumulator (ACC) register is modified by the following instructions:

- DSP (MACHI, MACLO, MULHI, MULLO, MVTACHI, MVTACLO and RACW).
- Multiply and multiply-and-accumulate (EMUL, EMULU, FMUL, MUL, and RMPA)

The accumulator (ACC) register is not pushed onto the stack by the interrupt handlers in Renesas Peripheral Driver Library.

If DSP instructions are being utilised in the users' code, notification functions which are called by the interrupt handlers in Renesas Peripheral Driver Library should either

- 1. Avoid using instructions which modify the ACC register.
- 2. Take a copy of the ACC register and restore it before exiting the callback function.



6. Registering Files with the IDE and Building Them

Note the following points when registering the files generated by the Peripheral Driver Generator with the IDE(High-performance Embedded Workshop, CubeSuite+ or e2 studio) and building them.

- (1) Source files generated by the Peripheral Driver Generator do not include a startup program. For this reason, you need to create a startup program by specifying [Application] as the project type during the process of creating a IDE project.
- (2) Source files registered by the Peripheral Driver Generator with the IDE include an interrupt handler and vector table. Since the interrupt handler and vector table must not overlap with those included in the startup program created by using the IDE, intprg.c and vecttbl.c are excluded from the set of files that are included in the build.
- (3) Source files Interrupt_xxx.c, which includes the interrupt handler that the Peripheral Driver Generator registers with the IDE, is overwritten when the Peripheral Driver GeneratorG generates source files.
- (4) The Renesas Peripheral Driver Library is produced using the default compiler options (except that [Double precision] is selected for [Precision of double]). If you specify the compiler options other than the defaults in your project, you have to utilize Renesas Peripheral Driver Library source under your responsibility.
- (5) The Renesas Peripheral Driver Library has been built specifying double-precision floating point. Therefore, to build the user program with Peripheral Driver Generator-generated files, specify double-precision floating point option in builder settings of IDE as follows. It's unnecessary at the time of e2 studio use.

CubeSuite+

- 1. Open the [CC-RX Property] by double-clicking [CC-RX(Build Tool)] in project tree.
- 2. In the [CPU] category, select [Handles in double precision] for [Precision of the double type and long double type].

High-performance Embedded Workshop

- 1. Select [Build]->[RX Standard Toolchain] from main menu to open the [RX Standard Toolchain] dialog box.
- 2. Select the [CPU] tab.
- 3. Click the [Details] button to open the [CPU details] dialog box.
- 4. Select [Double precision] for [Precision of double].
- (6) The Renesas Peripheral Driver Library use FIXEDVECT section that address is 0xFFFFFD0. Therefore, to build the user program with Peripheral Driver Generator-generated files, specify the linker option in builder setting of IDE as follows. It's necessary at the time of e2 studio use.
 - 1. Select the project on Project Explorer.
 - 2. Select [File]->[Properties] from main menu to open the [Properties] window.
 - 3. Select [C/C++ build] ->[Settings]
 - 4. Select [All configurations] for [Configuration]
 - 5. Select [Linker] -> [Section] to show [Section viewer]
 - 6. Set the address of the FIXEDVECT section as 0xFFFFFFD0.

Appendix 1. Pin Functions for which the Allocation Can be Changed

Table a-1.1 10	0-pin LQFP	(the Upper Row of Each Pair is the Default Selection)
----------------	------------	---

Peripheral module	Pin function	Selection of assignment	Pin No
CAC	CACREF	PA0/MTIOC4A/SSLA1/CACREF	70
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	45
		PH0/CACREF	38
ICUb	IRQ0	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	20
(External	intego	PD0/IRQ0	86
Interrupts)		PH1/TMO0/IRQ0	37
interruptoj	IRQ1	P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	19
	inter	PD1/MTIOC4B/IRQ1	85
		PH2/TMRI0/IRQ1	36
	IRQ2	P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	18
	II QZ	P12/TMCI1/SCL/IRQ2	34
		PD2/MTIOC4D/IRQ2	84
	IRQ3	P33/MTIOC0D/TMRI3/POE3#/RXD6/SMISO6/SSCL6/IRQ3	17
	IKQS	P13/MTIOC0B/TMO3/SDA/IRQ3	33
		PD3/POE8#/IRQ3	
	IRQ4	PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	83 59
	IKQ4		
		P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4 P34/MTIOC0A/TMCI3/POE2#/SCK6/IRQ4	32
			16
	10.05		82
	IRQ5	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	66
		P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	31
		PD5/MTIC5W/POE2#/IRQ5	81
		PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	73
	IRQ6	PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	67
		P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	30
		6/RTCOUT/ADTRG0#	30
		PD6/MTIC5V/POE1#/IRQ6	80
		PE6/IRQ6/AN014	72
	IRQ7	PE2/MTIOC4A/RXD12/RXDX12/SMISO12/SSCL12/IRQ7/AN010	76
		P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
		PD7/MTIC5U/POE0#/IRQ7	79
		PE7/IRQ7/AN015	71
MTU0-5	MTCLKA	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	32
		P24/MTIOC4A/MTCLKA/TMRI1	24
		PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	66
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	46
	MTCLKB	P15/MTIOC0B/MTCLKB/TMCl2/RXD1/SMISO1/SSCL1/IRQ5	31
		P25/MTIOC4C/MTCLKB/ADTRG0#	23
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	64
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	45
	MTCLKC	P22/MTIOC3B/MTCLKC/TMO0	26
		PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	39
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	48
	MTCLKD	P23/MTIOC3D/MTCLKD	25
		PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	67
		PC5/MTIOC0D/MTCLKD/TMRI2/RSPCKA	47



•	-		
MTU0	MTIOC0A	P34/MTIOC0A/TMCI3/POE2#/SCK6/IRQ4	16
		PB3/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	57
	MTIOC0B	P13/MTIOC0B/TMO3/SDA/IRQ3	33
		P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	31
		PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	69
	MTIOC0C	P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	18
		PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	59
	MTIOCOD	P33/MTIOC0D/TMRI3/POE3#/RXD6/SMISO6/SSCL6/IRQ3	17
		PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	67
MTU1	MTIOC1A	P20/MTIOC1A/TMRI0	28
		PE4/MTIOC4D/MTIOC1A/AN012/CMPA2	74
	MTIOC1B	P21/MTIOC1B/TMCI0	27
		PB5/MTIOC2A/MTIOC1B/TMRI1/POE1#/SCK9	55
MTU2	MTIOC2A	P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	22
WH 02	MITIO02/	PB5/MTIOC2A/MTIOC1B/TMRI1/POE1#/SCK9	55
	MTIOC2B	P27/MTIOC2B/TMCI3/SCK1	21
	MITIOCZB	PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	73
MTU3	MTIOC3A	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	32
101103	MITIOCSA		
		P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
			51
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	45
		PJ1/MTIOC3A	6
	MTIOC3B	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
		P22/MTIOC3B/MTCLKC/TMO0	26
		PB7/MTIOC3B/TXD9/SMOSI9/SSDA9	53
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	47
	MTIOC3C	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	30
		6/RTCOUT/ADTRG0#	
		PC0/MTIOC3C/CTS5#/RTS5#/SS5#/SSLA1	52
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	46
		PJ3/MTIOC3C/CTS6#/RTS6#/SS6#	4
	MTIOC3D	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	30
		6/RTCOUT/ADTRG0#	
		P23/MTIOC3D/MTCLKD	25
		PB6/MTIOC3D/RXD9/SMISO9/SSCL9	54
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	48
MTU4	MTIOC4A	P24/MTIOC4A/MTCLKA/TMRI1	24
		PA0/MTIOC4A/SSLA1/CACREF	70
		PB3/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	57
		PE2/MTIOC4A/RXD12/RXDX12/SMISO12/SSCL12/IRQ7/AN010	76
	MTIOC4B	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	20
		P54/MTIOC4B/TMCI1	40
		PC2/MTIOC4B/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3	50
		PD1/MTIOC4B/IRQ1	85
		PE3/MTIOC4B/POE8#/CTS12#/RTS12#/SS12#/AN011/CMPA1	75
	MTIOC4C	P25/MTIOC4C/MTCLKB/ADTRG0#	23
		PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	59
		PE1/MTIOC4C/TXD12/TXDX12/SIOX12/SMOSI12/SSDA12/AN009	77
		PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	73
	MTIOC4D	P31/MTIOC4D/TMCl2/CTS1#/RTS1#/SS1#/IRQ1	19
	W11004D	P55/MTIOC4D/TMCi2/CT3T#/RT3T#/S3T#/IRQT	39
			39
		PC3/MTIOC4D/TXD5/SMOSI5/SSDA5/IRTXD5	49



		PE4/MTIOC4D/MTIOC1A/AN012/CMPA2	74
MTU5	MTIC5U	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR	66
			-
	MTIC5V		
	MTIC5W		
		PA4/MTICSU/MTCLKA/TMRI0/TXD5/SNOSI5/SSDA5//RTXD5/SSLA0//R 66 PD7/MTICSU/POE0#//RQ7 79 PA6/MTICSV/POE1#//RQ6 80 PB0/MTICSV/POE1#//RQ6 80 PD5/MTICSU/POE1#//RQ6 80 PD5/MTICSU/POE2#//RQ5 81 PD5/MTICSU/POE2#//RQ5 81 PC/MTICOC3D/MTCLKC/TMC1//POE1#/SCK5/SSLA0 81 PD7/MTICSU/POE0#//RQ7 79 PB5/MTICOC3D/MTCLKC/TMC1//POE1#/SCK9 85 PD6/MTICSV/POE1#//RQ6 80 P34/MTICOCA/TMC13/POE2#/SCK6/IRQ4 16 PA6/MTICSV//POE2#//RQ5 81 P33/MTICOCA/TMC13/POE2#/SCK6/IRQ4 81 P33/MTICOCA/TMR0/POE3#/RXD6/SMIS06/SSCL6//RQ3 17 PB3/MTICOCA/MTR13/POE3#/RXD6/SMIS06/SSCL6//RQ3 17 PD3/POE3#//RQ4 82 P17/MTIOC3A/MTIOC3B/TMO1/POE3#/SCK6 57 PD4/POE3#//RQ4 83 PD3/MTIOC4/TMR0/POE3#/SCK6 57 PD4/POE3#//RQ3 83 PE3/MTIOC4/MTR1/POE58//SCK1/MISOA/SDA/IRQ7 29 P3/MTIOC0/MTR12/POE58//SCK6//SDA6//RQ4 50 P10//TOC1B/TMR1/POE58//SCK0/SSDA6//RC4	
POE	POE0#		
	POE1#		
	POE2#		
		PD5/MTIC5W/POE2#/IRQ5	81
	POE3#	P33/MTIOC0D/TMRI3/POE3#/RXD6/SMISO6/SSCL6/IRQ3	17
		PB3/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	57
			82
	POE8#	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
		P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	20
		PD3/POE8#/IRQ3	83
		PE3/MTIOC4B/POE8#/CTS12#/RTS12#/SS12#/AN011/CMPA1	75
TMR0	TMCI0	P21/MTIOC1B/TMCI0	27
		PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	59
		PH3/TMCI0	35
	TMRI0	P20/MTIOC1A/TMRI0	28
			66
		PH2/TMRI0/IRQ1	36
	TMO0	P22/MTIOC3B/MTCLKC/TMO0	26
		PH1/TMO0/IRQ0	
TMR1	TMCI1		
	TMRI1		
	TMO1		
	TWICT		
TMR2	TMCI2		
	TWOIZ		
	TMRI2		
	TWITCH		
	TMO2		+1
	TIVIO2		30
			15
TMR3	TMCI3		-
C7IIVI	TIVICI3		
			64
	TMRI3	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	20
		P33/MTIOC0D/TMRI3/POE3#/RXD6/SMISO6/SSCL6/IRQ3	17



-	-		
		P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	18
		P55/MTIOC4D/TMO3	39
RTCc	RTCOUT	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/RTCOUT/ADTRG0#	30
		P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	18
SCI1	RXD1	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	31
	SMISO1	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	20
	SSCL1		
	TXD1	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/RTCOUT/ADTRG0#	30
	SMOSI1	P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	22
	SSDA1		
	SCK1	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
	Contr	P27/MTIOC2B/TMCI3/SCK1	21
	CTS1#	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	32
	RTS1#	P31/MTIOC4D/TMCl2/CTS1#/RTS1#/SS1#/IRQ1	19
	SS1#		13
SCI5	RXD5	PA2/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3	68
000	SMISO5	PA2/RAD3/SMISO3/SSCL3/IRRAD3/SSLA3 PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	67
	SIVII305	PC2/MTIOC6B/MTCLRD/KXD5/SMISO5/SSCL5/IRRXD5/SSLA3	50
	IRRXD5		50
	TXD5	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR	
	TADS	Q5	66
	SMOSI5	PC3/MTIOC4D/TXD5/SMOSI5/SSDA5/IRTXD5	49
	SINOSIS SSDA5		43
	IRTXD5		
	SCK5	PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	69
	3013	PC1/MTIOC0J/MTCLRO/SCR0/SSLA2/CVREFA	51
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	48
	CTS5#	PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	64
	RTS5#	PC0/MTICSV/MTCLRb/TMCI3/POE2#/CT35#/RT35#/S55#/MOSIA PC0/MTICC3C/CTS5#/RTS5#/SS5#/SSLA1	52
	SS5#	F 60/10110636/C133#/R133#/333#/33EA1	52
SCI6	RXD6	P33/MTIOC0D/TMRI3/POE3#/RXD6/SMISO6/SSCL6/IRQ3	17
5010		PB0/MTIC5W/RXD6/SMISO6/SSCL6/RSPCKA	
	SMISO6 SSCL6	PB0/MTIC3W/RAD0/SIMISO0/SSCL0/RSPCKA	61
			10
	TXD6 SMOSI6		18
	SINOSI6 SSDA6	PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	59
	SSDA6	P34/MTIOC0A/TMCI3/POE2#/SCK6/IRQ4	16
	SCRO		
	CTS6#	PB3/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	57
	CTS6#	PB2/CTS6#/RTS6#/SS6#	58
	RTS6#	PJ3/MTIOC3C/CTS6#/RTS6#/SS6#	4
	SS6#		
RIIC0	SCL	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	30
		6/RTCOUT/ADTRG0#	24
	004	P12/TMCI1/SCL/IRQ2	34
	SDA	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
	DODOKA	P13/MTIOC0B/TMO3/SDA/IRQ3	33
RSPI0	RSPCKA		65
		PB0/MTIC5W/RXD6/SMISO6/SSCL6/RSPCKA	61
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	47
	MOSIA	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	30
		6/RTCOUT/ADTRG0#	



i .	I	r	
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	64
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	46
	MISOA	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	29
		PA7/MISOA	63
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	45
	SSLA0	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	66
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	48
	SSLA1	PA0/MTIOC4A/SSLA1/CACREF	70
		PC0/MTIOC3C/CTS5#/RTS5#/SS5#/SSLA1	52
	SSLA2	PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	69
		PC1/MTIOC3A/SCK5/SSLA2	51
	SSLA3	PA2/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3	68
		PC2/MTIOC4B/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3	50
S12AD	ADTRG0#	P07/ADTRG0#	98
		P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/RTCOUT/ADTRG0#	30
		P25/MTIOC4C/MTCLKB/ADTRG0#	23

Table a-1 2	64-pin LQFP	(the Upper Row of Each Pair is the Default Selection)
10010 0 1.2	0+pineori	

Peripheral module	Pin function	Selection of assignment	Pin No.
CAC	CACREF	PA0/MTIOC4A/SSLA1/CACREF	45
CAC	CACKEP	PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	43 27
		PH0/CACREF	24
ICUb	IRQ0	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	14
(External	iitte	PH1/TMO0/IRQ0	23
Interrupts)	IRQ1	P31/MTIOC4D/TMCl2/CTS1#/RTS1#/SS1#/IRQ1	13
interrupts)		PH2/TMRI0/IRQ1	22
	IRQ4	PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	37
	INQ4	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	20
	IRQ5	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	42
		P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	19
		PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	46
	IRQ6	PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	43
		P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/RTCOUT/ADTRG0#	18
	IRQ7	PE2/MTIOC4A/RXD12/RXDX12/SMISO12/SSCL12/IRQ7/AN010	49
		P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	17
MTU0-5	MTCLKA	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	20
		PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	42
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	28
	MTCLKB	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	19
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	41
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	27
	MTCLKC	PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	44
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	30
	MTCLKD	PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	43
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	29
MTU0	MTIOC0A	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	19
		PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	44



	MTIOC0B	P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	12
		PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	37
MTU1	MTIOC1A	P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	16
		PB5/MTIOC2A/MTIOC1B/TMRI1/POE1#/SCK9	35
	MTIOC1B	P27/MTIOC2B/TMCI3/SCK1	15
		PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	46
MTU3	MTIOC3A	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	20
		P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	17
	MTIOC3B	PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	27 17
	WITIOC3B	PB7/PC1/MTIOC3B/TXD9/SMOSI9/SSDA9	33
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	29
	MTIOC3C	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	18
		6/RTCOUT/ADTRG0#	10
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	28
	MTIOC3D	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/RTCOUT/ADTRG0#	18
		PB6/PC0/MTIOC3D/RXD9/SMISO9/SSCL9	34
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	30
MTU4	MTIOC4A	PA0/MTIOC4A/SSLA1/CACREF	45
		PB3/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	36
		PE2/MTIOC4A/RXD12/RXDX12/SMISO12/SSCL12/IRQ7/AN010	49
	MTIOC4B	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	14
		P54/MTIOC4B/TMCI1	26
		PC2/MTIOC4B/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3	32
		PE3/MTIOC4B/POE8#/CTS12#/RTS12#/SS12#/AN011/CMPA1	48
	MTIOC4C	PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	37
		PE1/MTIOC4C/TXD12/TXDX12/SIOX12/SMOSI12/SSDA12/AN009	50
		PE5/MTIOC4C/MTIOC2B/IRQ5/AN013	46
	MTIOC4D	P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	13
		P55/MTIOC4D/TMO3	25
		PC3/MTIOC4D/TXD5/SMOSI5/SSDA5/IRTXD5	31
		PE4/MTIOC4D/MTIOC1A/AN012/CMPA2	47
POE	POE8#	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	17
		P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	14
		PE3/MTIOC4B/POE8#/CTS12#/RTS12#/SS12#/AN011/CMPA1	48
TMR0	TMCI0	PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	37
		PH3/TMCI0	21
	TMRI0	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR	
		Q5	42
		PH2/TMRI0/IRQ1	22
	TMO0	PB3/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	36
		PH1/TMO0/IRQ0	23
TMR1	TMCI1	P54/MTIOC4B/TMCI1	26
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	30
	TMO1	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	17
		P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	16



TMR2	TMCI2	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	19
		P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	13
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	28
	TMRI2	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	20
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	29
	TMO2	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/RTCOUT/ADTRG0#	18
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	27
TMR3	TMCI3	P27/MTIOC2B/TMCI3/SCK1	15
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	41
	TMO3	P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	12
		P55/MTIOC4D/TMO3	25
RTCc	RTCOUT	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/RTCOUT/ADTRG0#	18
		P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	12
SCI1	RXD1	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	19
	SMISO1	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	14
	SSCL1		
	TXD1	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/RTCOUT/ADTRG0#	18
	SMOSI1	P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	16
	SSDA1		
	SCK1	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	17
		P27/MTIOC2B/TMCI3/SCK1	15
	CTS1#	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	20
	RTS1#	P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	13
	SS1#		
SCI5	RXD5	PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	43
	SMISO5	PC2/MTIOC4B/RXD5/SMISO5/SSCL5/IRRXD5/SSLA3	32
	SSCL5		
	TXD5	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR	
		Q5	42
	SMOSI5	PC3/MTIOC4D/TXD5/SMOSI5/SSDA5/IRTXD5	31
	SSDA5		
	SCK5	PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	44
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	30
SCI6	TXD6	P32/MTIOC0C/TMO3/TXD6/SMOSI6/SSDA6/IRQ2/RTCOUT	12
	SMOSI6	PB1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	37
	SSDA6		
RSPI0	RSPCKA	PB0/MTIC5W/RXD6/SMISO6/SSCL6/RSPCKA	39
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	29

MOSIA		
MOSIA	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	18
	6/RTCOUT/ADTRG0#	10
	PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	41
	PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	28
MISOA	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	17
	PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	27
SSLA0	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR	40
	Q5	42
	PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	30

Table a-1.3 48-pin LQFP (the Upper Row of Each Pair is the Def
--

Peripheral	Pin	Selection of assignment	Pin No
module CAC	function CACREF	PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	21
CAC	CACKER	PH0/CACREF	21
ICUb	IRQ0	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	10
(External		PH1/TMO0/IRQ0	19
Interrupts)	IRQ1	P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	9
		PH2/TMRI0/IRQ1	18
	IRQ4	PB1/PC1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	27
		P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	16
	IRQ5	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	32
		P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	15
	IRQ6	PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	33
		P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/ADTRG0#	14
	IRQ7	PE2/MTIOC4A/RXD12/RXDX12/SSCL12/IRQ7/AN010	37
		P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
MTU0-5	MTCLKA	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	16
		PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	32
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	22
	MTCLKB	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	15
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	31
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	21
	MTCLKC	PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	34
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	24
	MTCLKD	PA3/MTIOC0D/MTCLKD/RXD5/SMISO5/SSCL5/IRRXD5/IRQ6	33
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	23
MTU0	MTIOC0B	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	15
		PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	34
MTU2	MTIOC2A	P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	12
		PB5/PC3/MTIOC2A/MTIOC1B/TMRI1/POE1#	25



•	•		Арренал
MTU3	MTIOC3A	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	16
		P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	21
	MTIOC3B	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	23
	MTIOC3C	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/ADTRG0#	14
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	22
	MTIOC3D	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/ADTRG0#	14
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	24
MTU4	MTIOC4A	PB3/PC2/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	26
		PE2/MTIOC4A/RXD12/RXDX12/SSCL12/IRQ7/AN010	37
	MTIOC4B	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	30
		PE3/MTIOC4B/POE8#/CTS12#/RTS12#/AN011/CMPA1	36
	MTIOC4C	PB1/PC1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	27
		PE1/MTIOC4C/TXD12/TXDX12/SIOX12/SSDA12/AN009	38
	MTIOC4D	P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	9
		PE4/MTIOC4D/MTIOC1A/AN012/CMPA2	35
POE	POE8#	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
		P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	30
		PE3/MTIOC4B/POE8#/CTS12#/RTS12#/AN011/CMPA1	36
TMR0	TMCI0	PB1/PC1/MTIOC0C/MTIOC4C/TMCI0/TXD6/SMOSI6/SSDA6/IRQ4	27
		PH3/TMCI0	17
	TMRI0	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR	
		Q5	32
		PH2/TMRI0/IRQ1	18
	TMO0	PB3/PC2/MTIOC0A/MTIOC4A/TMO0/POE3#/SCK6	26
		PH1/TMO0/IRQ0	19
TMR1	TMO1	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
		P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	12
TMR2	TMCI2	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	15
		P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	9
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	22
	TMRI2	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	16
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	23
	TMO2	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ	
		6/ADTRG0#	14
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	21
TMR3	TMCI3	P27/MTIOC2B/TMCI3/SCK1	11
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	31
SCI1	RXD1	P15/MTIOC0B/MTCLKB/TMCI2/RXD1/SMISO1/SSCL1/IRQ5	15
	SMISO1	P30/MTIOC4B/TMRI3/POE8#/RXD1/SMISO1/SSCL1/IRQ0	10



	SSCL1		
	TXD1	P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1	12
	SMOSI1	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/ADTRG0#	14
	SSDA1		
	SCK1	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
		P27/MTIOC2B/TMCI3/SCK1	11
	CTS1#	P14/MTIOC3A/MTCLKA/TMRI2/CTS1#/RTS1#/SS1#/IRQ4	16
	RTS1# SS1#	P31/MTIOC4D/TMCI2/CTS1#/RTS1#/SS1#/IRQ1	9
SCI5	SCK5	PA1/MTIOC0B/MTCLKC/SCK5/SSLA2/CVREFA	34
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	24
RSPI0	RSPCKA	PB0/PC0/MTIC5W/RXD6/SMISO6/SSCL6/RSPCKA	29
		PC5/MTIOC3B/MTCLKD/TMRI2/RSPCKA	23
	MOSIA	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1/MOSIA/SCL/IRQ 6/ADTRG0#	14
		PA6/MTIC5V/MTCLKB/TMCI3/POE2#/CTS5#/RTS5#/SS5#/MOSIA	31
		PC6/MTIOC3C/MTCLKA/TMCI2/MOSIA	22
	MISOA	P17/MTIOC3A/MTIOC3B/TMO1/POE8#/SCK1/MISOA/SDA/IRQ7	13
		PC7/MTIOC3A/TMO2/MTCLKB/MISOA/CACREF	21
	SSLA0	PA4/MTIC5U/MTCLKA/TMRI0/TXD5/SMOSI5/SSDA5/IRTXD5/SSLA0/IR Q5	32
		PC4/MTIOC3D/MTCLKC/TMCI1/POE0#/SCK5/SSLA0	24



RX220 Group Peripheral Driver Generator Reference Manual				
Publication Date:	May 16, 2014	Rev.1.02		
Published by:	Renesas Electronics Corporation			
Edited by:	Microcomputer Tool Development Department 4 Renesas Solutions Corporation			



SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc. 2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A. Tel: +1-408-588-6000, Fax: +1-408-588-6130 Renesas Electronics Canada Limited 1011 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada Tel: +1-905-898-5441, Fax: +1-905-5989-3220 Renesas Electronics Europe Limited Dukes Meadow, Milboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +44-1628-585-100, Fax: +44-1628-585-900 Renesas Electronics Europe GmbH Arcadiastrasse 10, 40472 Disseldoff, Germany 1e1: +49-21-16500-0, Fax: +44-211-6503-127 Renesas Electronics (China) Co., Ltd. Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China Tel: +69-10-8235-1155, Fax: +86-10-8235-7679 Renesas Electronics (Shangha), Co., Ltd. Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China Tel: +66-10-8235-1155, Fax: +86-10-8235-7679 Renesas Electronics (Shangha), Co., Ltd. Rom 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Shanghai, P. R. China 200333 Tel: +86-21-2226-088, Fax: +86-21-2226-0299 Renesas Electronics China Foo, Ltd. Unit 301. Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333 Tel: +86-21-226-6088, Fax: +86-21-2226-0299 Renesas Electronics Taiwan Co., Ltd. 137. No. 308, Fu Shing North Road, Taipei 10543, Taiwan Tel: +886-2-8175-9600, Fax: +862-2175-9670 Renesas Electronics Singapore Pte. Ltd. 00 Bendemeer Road, Unit 400-02 Hyfitx Innovation Centre, Singapore 339949 Tel: +65-213-0200, Fax: +866-213-0300 Renesas Electronics Korae Co., Ltd. 127. 204 Telsaysia Sdn. Bhd. Unit 906, Block B, Meara Amcorp, Amcorp Trade Centre, No. 18, JIn Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +60-3795-9309, Fax: +805-2950-9510

> © 2014 Renesas Electronics Corporation and Renesas Solutions Corporation. All rights reserved. Colophon 3.0

RX220 Group Peripheral Driver Generator Reference Manual

