

# APPLICATION NOTE

## RZ/T1 Group

Guide for Applying the Code Generation Tool to the Sample Program

R01AN3860EJ0110 Rev.1.10 Apr. 11, 2018

## Introduction

This application note describes the process of incorporating a control program for peripheral modules of the microcontroller (device driver program) in the sample program of the RZ/T1 Group Initial Settings. The code for the control program is generated by the automatic generation tool for I/O drivers (hereinafter called the Code Generation Tool).

The resulting sample program enables periodic counter operation of the compare match timer (CMT) that produces compare-match interrupts which are used to switch an LED on and off.

# **Target Devices**

RZ/T1 Group

When applying the program covered in this application note to another microcontroller, modify the program to suit the specifications of the target microcontroller and extensively evaluate the program after modification.



# Table of Contents

1.	Speci	fications	3
2.	Opera	ating Environment	4
3.	Relat	ed Application Notes	5
4.	Perip	heral Functions	6
5.	Hard	vare	7
	5.1	Hardware Configuration Examples	7
	5.2	Pins	8
6.	Softw	are	9
	6.1	Operation Overview	9
	6.1.1	Project Settings	11
	6.1.2	Preparation	13
	6.1.3	Exception Processing Vector Table	13
	6.2	Interrupts	14
	6.3	Procedure for Incorporating Code from the Code Generation Tool	15
	6.3.1	Generating Code with the Code Generation Tool	15
	6.3.2	Incorporation in the Sample Program of RZ/T1 Group Initial Settings	31
	6.4	Fixed-Width Integer Types	43
	6.5	Function	43
	6.6	Flowchart	44
	6.6.1	Loader Program Processing	44
	6.6.2	Processing by the Application Program Created by the Code Generation Tool	45
	6.6.3	Common main Processing	46
	6.6.4	Interrupt Processing by the Timer Selected by the User (CMT0)	46
7.	Samp	le Program	47
8.	Relat	ed Documents	48
9.	Usag	e Note	49

# 1. Specifications

Table 1.1 lists the peripheral functions used and their applications and Figure 1.1 shows the operating environment.

Table 1 1	Porinhoral	Functions and	Their Annlica	tione
	renpilerai	Functions and	ппен Арриса	10115

Peripheral Module	Application
Clock generation circuit (CPG)	Provides the CPU clock and low-speed on-chip oscillator
Interrupt Controller (ICUA)	Processing the compare-match interrupt (CMI0)
Compare match timer (CMT)	Periodic counter operation and tests for matches by the compare match timer
Bus state controller (BSC)	Connection of NOR flash memory to the CS0 and CS1 spaces and SDRAM to the CS2 and CS3 spaces.
SPI multi I/O bus controller (SPIBSC)	Used to connect the serial flash memory to SPI multi I/O space
Error control module (ECM)	Initial setting of the ERROROUT# pin
General I/O port	Control of pin output to turn the LED on and off





# 2. Operating Environment

The sample program covered in this application note runs in the environment below.

Table 2.1	Operating Environment
-----------	-----------------------

Item	Description
Microcomputer	RZ/T1 Group
Operating frequency	CPUCLK = 450 MHz
Operating voltage	3.3 V
Integrated development environment (any of those listed)	From IAR Systems: Embedded Workbench for ARM Version 7.80.2 From ARM: DS-5TM 5.25 From Renesas: e <sup>2</sup> studio 5.2.0
Tool for generating code	From Renesas: AP4 1.07 Note: The e <sup>2</sup> studio includes a plug-in that has equivalent functionality for generating code to that of AP4 1.04.
Operating mode	SPI boot mode 16-bit bus boot mode
Board used	RZ/T1 evaluation board (RTK7910018C00000BE)
Device used (Functions used on the board)	<ul> <li>NOR flash memory (for connection to the CS0 and CS1 spaces) Manufacturer: Macronix International Co., Ltd. Model: MX29GL512FLT2I-10Q</li> <li>SDRAM (for connection to the CS2 and CS3 spaces) Manufacturer: Integrated Silicon Solution Inc. Model: IS42S16320D-7TL</li> <li>Serial flash memory Manufacturer: Macronix International Co., Ltd. Model: MX25L51245G</li> </ul>



# 3. Related Application Notes

Other application notes that are related to this one are listed below. Please refer to them, too.

- RZ/T1 Group Initial Settings (R01AN2554EJ)
- RZ/T1 Group Compare Match Timer (CMT) (R01AN2555EJ)



# 4. Peripheral Functions

See the *RZ/T1 Group User's Manual: Hardware* for basic descriptions of operating modes and the clock generation circuit (CPG), compare match timer (CMT), interrupt controller (ICUA), bus state controller (BSC), SPI multi I/O bus controller (SPIBSC), error control module (ECM), reset system, and general I/O ports.



## 5. Hardware

## 5.1 Hardware Configuration Examples

Figure 5.1 shows an example of the hardware configuration for this application.



Figure 5.1 Example of the Hardware Configuration



#### 5.2 Pins

 Table 5.1 lists pins to be used and their functions.

Table 5.1	Pins and Functions
-----------	--------------------

Pin Name	Input/Output	Function
A1 to A25*1	Output	Address signal output for NOR flash memory and SDRAM
D0 to D15*1	Input/Output	Data signal input and output for NOR flash memory and SDRAM
CS0#*1	Output	Device selection signal output to NOR flash memory in the CS0 space
CS1#*1	Output	Device selection signal output to NOR flash memory in the CS1 space
CS2#*1	Output	Device selection signal output to SDRAM in the CS2 space
CS3#*1	Output	Device selection signal output to SDRAM in the CS3 space
RAS#*1	Output	RAS# control signal output to SDRAM
CAS# *1	Output	CAS# control signal output to SDRAM
RD/WR#*1	Output	Read control signal or write control signal output to SDRAM
CKE*1	Output	CK enabling control signal output to SDRAM
RD#*1	Output	Strobe signal output indicating reading
BS#	Output	Not used in this sample program.
WE0#/DQMLL*1	Output	Write strobe signal output for D15 to D8
WE1#/DQMLU*1	Output	Write strobe signal output for D7 to D0
SPBSSL*1	Output	Slave selection
SPBCLK*1	Output	Clock output
SPBMO/SPBIO0*1	Input/Output	Master output data: data 0
SPBMI/SPBIO1*1	Input/Output	Master input data: data 1
SPBIO2*1	Input/Output	Data 2
SPBIO3*1	Input/Output	Data 3
MD0	Input	Boot mode selection:
MD1	Input	MD0 = "L", MD1 = "L", MD2 = "L" (SPI boot mode) MD0 = "L" MD1 = "H" MD2 = "L" (16-bit bus boot mode)
MD2	Input	
PM7*1	Output	Lighting and darkening LED10

Note: The mark "#" indicates negative logic (i.e. active low).

Note 1. The function for this pin is to be set by the Code Generation Tool for that purpose.



## 6. Software

#### 6.1 Operation Overview

This sample program is based on the sample program of the RZ/T1 Group Initial Settings. The loader program of the sample program of of the Initial Settings is used as it is for the loader program (loader) section. The user application program (user application) incorporates the code generated by the Code Generation Tool with the common main code. The actual process of incorporation will be described later.

See *Application Note: RZ/T1 Group Initial Settings* for more details on the operation of the sample program of the RZ/T1 Group Initial Settings.

Figure 6.1 shows the configuration outline of the sample program produced in this application note.



- The user application section in the sample program of the Initial Settings should be replaced with the user application section generated with the Code Generation Tool.
- Incorporating code from the Code Generation Tool as in this process can, for example, allow more efficient program development in the case of changes to the set-up of peripheral functions.
- A common main function was generated for this sample program to replace that in the sample program of the Initial Settings.

Figure 6.1 Configuration of This Sample Program in Outline



Figure 6.2 shows the operation of the loader and user application initial sections in outline.

After the device is booted up, the loader sets up the stack among other things, copies the user application program, and causes execution to branch to the address where the user application program starts.

As you see in the figure below, both the loader from the sample program of the Initial Settings and the user application code generated through automatic code generation make settings for the clock and bus controllers. Therefore, in this program, conflicts between the settings have been avoided by disabling (commenting out) the settings for the clock and bus controllers in the loader.

The code generated by code generation for the user application sets up the interrupt controller, the clock, the buses and other peripheral modules. Settings in this part of the code are made to suit the operating mode in use (in this case, SPI boot, 16-bit bus boot).

Settings to start cyclic counter operation of the compare match timer (CMT) and for switching the LED on and off in response to compare match A interrupts are made in the common main function, which is independent of the operating mode.

This process of incorporation demonstrates how users are able to use code produced by setting in the Code Generation Tool as desired with the loader section of the sample program of the Initial Settings.



Figure 6.2 Outline of Operations after Booting Up the Device

#### 6.1.1 Project Settings

This sample program includes three projects as listed below.



 Table 6.1 and Table 6.2 show the hierarchies of folders following incorporation of the code generated by the Code

 Generation Tool with that from the sample program of the Initial Settings for two of the IDEs.

The folders and the files incorporated with the sample program are in bold type. Other than those parts, the structure is the same as that of the sample program of the Initial Settings.

See Application Note: RZ/T1 Group Initial Settings for the development environment and project settings.

# Table 6.1Folder Structure of the Sample Program of the Initial Settings Following Incorporation of the Code<br/>from the Code Generation Tool (1/2)





# Table 6.2Folder Structure of the Sample Program of the Initial Settings Following Incorporation of the Code<br/>from the Code Generation Tool (2/2)

		Main Hierarchy of Folders (e <sup>2</sup> st	udio)
Project folder	Subfolder		Remark
RZ_T_nor_sample	.setting —	CodeGenerator	<ul> <li>[Folder generated with the Code Generation Tool]</li> <li>(16-bit bus boot version (NOR))</li> <li>Working file for the Code Generation Tool (.cgp)</li> </ul>
	inc		Folder for storing the include files for the sample program of the Initial Settings
L	src	cg_src	[Folder generated with the Code Generation Tool] (16-bit bus boot version (NOR)) • Output folder for the source code (cg_src)
		common	Sample program folder of the original initial settings program
		drv	Driver folder
		sample user_m	ain.c Folder for the main program of the original initial settings program Common main (user_app_main)
		iodefine.h <sup>*1</sup>	iodefine for the e2studio, transferred from the inc folder
RZ_T_sflash_sample	.setting —	CodeGenerator	[Folder generated with the Code Generation Tool] (SPI boot version (serial)) • Working file of the Code Generation Tool (.cgp)
_	inc		Folder for storing the include files for the sample program of the Initial Settings
L	src	cg_src	[Folder generated with the Code Generation Tool] (SPI boot version (serial)) • Output folder for the source code (cg_src)
		common	Sample program folder of the original initial settings program
		drv	Driver folder
	_	sample user_m	ain.c Folder for the main program of the original initial settings program Common main (user_app_main)
		iodefine.h <sup>*1</sup>	iodefine for the e <sup>2</sup> studio, transferred from the inc folder
RZ_T_ram_sample	.setting —	CodeGenerator	[Folder generated with the Code Generation Tool] (RAM version) • Working file of the Code Generation Tool (.cgp)
_	inc		Include files for the RAM version of the original initial settings program
L	src	cg_src	[Folder generated with the Code Generation Tool] (RAM version) • Output folder for the source code (cg_src)
		common	Sample program folder of the original initial settings program
		drv	Driver folder
	_	sample user_m	ain.c Folder for the main program of the RAM version of the original initial settings program Common main (user_app_main)
		iodefine.h <sup>*1</sup>	iodefine for the e <sup>2</sup> studio, transferred from the inc folder

Note 1. When incorporating code generated with the Code Generation Tool in the e<sup>2</sup>studio environment, place iodefine.h from the inc folders of each of the projects immediately below the project folder.



#### 6.1.2 Preparation

Settings for SW4 on the RZ/T1 evaluation board (RTK7910018C00000BE) depend on the project to be used. Table 6.3 lists the settings. Each setting of SW4 is shown in the *RZ/T1 Evaluation Board RTK7910018C00000BE User's Manual*. For details, see Section 8, Related Documents.

Table 6.3	Settings of	SW4
-----------	-------------	-----

Sample Program	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16-bit bus boot mode version	ON	OFF	ON	ON	ON	OFF
SPI boot mode version	ON	ON	ON	ON	ON	OFF
RAM execution version			Any settir	ng of SW4 above	e	

### 6.1.3 Exception Processing Vector Table

The RZ/T1 has 7 types of exception processing (for resets, undefined instructions, software interrupts, abortion of prefetching, abortion of data, and IRQ and FIQ exceptions). The vectors are allocated to the 34-byte area starting from address 0000 0000H (address range 0000 0000h to 0000 0024h) when the Code Generation Tool is used.

Write the branch instruction for processing of each exception in the exception processing vector table.

Table 6.4 shows the exception processing vector table in this program. Please adjust it as necessary.

•	•	
Exception	Handler Address	Remark
RESET exception	0000 0000h	Branches to itself (to avoid branching to an unknown address)
Undefined instruction exception	0000 0004h	Branches to itself (user definable)
Software interrupt exception	0000 0008h	Branches to itself (user definable)
Prefetch abort exception	0000 000Ch	Branches to itself (user definable)
Data abort exception	0000 0010h	Branches to itself (user definable)
Reserved	0000 0014h	Branches to itself (user definable)
IRQ exceptions	0000 0018h	Branches to itself (to avoid branching to an unknown address)
FIQ exceptions	0000 001Ch	Branches to itself (overwritten by the Code Generation Tool)

#### Table 6.4 Exception Processing Vector Table

Note: In the application program, locations from 0000 001Ch to 0000 0024h including those for FIQ exceptions above should be overwritten by code generated by the Code Generation Tool.



### 6.2 Interrupts

 Table 6.5 shows the interrupt to be used in the sample program.

Interrupt (Source ID)	Priority Level	Outline of Processing
CMT0 interrupt (CMI0)	15	Whenever matches in comparison are detected (at the 100-ms interval specified by the Code Generation Tool), this interrupt is generated and LED10 is turned off if on and on if off as a result.

 Table 6.5
 Interrupt to be Used in the Sample Program



#### 6.3 Procedure for Incorporating Code from the Code Generation Tool

This section gives an example of the procedure when the Code Generation Tool is used while EWARM is the IDE. The assumed operating mode is SPI boot mode.

### 6.3.1 Generating Code with the Code Generation Tool

In this sample program, PORTM7 is set up to switch LED10 on and off in response to compare-match interrupts produced by the cyclic counter operation of CMT0. A description of the example of the procedure for this sample program follows.

- (1) Starting the Tool for Generating Code
- (2) Creating a New Project

The following setting is necessary to create the new project.

• Select the microcontroller (an R7S910018CBG in the figure), and specify the compiler to use (that of IAR EWARM), the name of the project, and where to create the project.



• Note on the Name and Location of the Project

The desired location specified by the user will be the place where the project is to be initially created with the Code Generation Tool, but the location for storage of the output file of the Code Generation Tool is separated per operating mode of the sample program.

Example of the Sample Program:

Folder name for 16-bit bus boot mode: cg\_src\_nor

Folder name for SPI boot mode: cg\_src\_serial

For how to incorporate each created folder into the sample program for making the initial settings of devices, refer to Section 6.3.2 (1), Importing the Environment of the code created by the Code Generation Tool (Copying).



(3) Compare Match Timer (CMT) and Port Settings

[Settings for the Compare Match Timer] Set up the compare match timer for use with the settings below. Channel to use: CMT0 Clock setting: PCLKD/512 Interval: 100 ms Interrupt from compare match (CMI0): Enabled Priority level: 15

#### • Screenshot [1]: Settings of CMT0

	🛐 Generate code [ 🚣 💒 🏙 💑 😂 💷 🔞 🔅 🕲 🕲 🕲 🕲 🖉 🖉 📲 🎜 📲 🕮 🖓 🔬 💉 S	13 14
	CMT0 CMT1 CMT2 CMT3 CMT4 CMT5	
R_CM10_Create	Compare match timer operation setting     Unused     Used	
In Section 2 CMT1 In Section 2 CMT2		
	-Interval value setting	
⊟ <b>U</b> ir_cg_cmt_user.c ⊟⊖⊂CMT0	Interval value (Actual value: 99.997013)	
R_CMT0_Create_UserInit	∠Interrupt setting	
terrer CMT1 terrer CMT2	Priority Level 15 ~	
umita ∭ir_cg_cmth		



[Settings for the Port] Set up the port pin for use as listed below. Used pin: PORTM7 (connected to LED10) Input/output: Output Setting for output: Select "Output 1".

• Screenshot [2]: Setting of PORTM7

Port0	Port1	Port2	Port3	Port4	Port5	Port6	Port7
PortE	PortF	PortG	PortH	PortJ	PortK	PortL	PortM
PM0							
Unused	🔾 In	🔘 Out	Disables inp	out pull-up and p	pull-down resisto	rs 🗸	Output 1
M1			·				
Unused	() In	O Out	Disables inp	out pull-up and j	pull-down resisto	rs 🗸	Output 1
PM2	-	-					
Inused	⊖ In	Out	Disables inc	out pull-up and i	oull-down resisto	rs U	Output 1
M2	0.	0.000					output 1
	∩ h	0.0.4	Disables in	aut pulleup and i	oull-down registo		Contrast 1
C Ondsed	0 11	Out	Disables inp	ar pan ap ana j	pan down resisto	• •	Output
/M4	<u>.</u>	0.0					
Unused	() In	Out	Disables inp	but pull-up and j	pull-down resisto	rs 🗸	Output 1
°M5							
Unused	🔾 In	Out	Disables inp	out pull-up and p	pull-down resisto	rs 🗸	Output 1
'M6							
Unused	$\bigcirc$ In	🔘 Out	Disables inp	out pull-up and p	pull-down resisto	rs 🗸	Output 1
'M7							
<ul> <li>Unused</li> </ul>	() In	Out	Disables inp	out pull-up and p	pull-down resisto	rs 🗸	Output 1

(4) Setting Modules for the Clock Signals and Bus The settings for this example are as shown for the Initial Settings sample program (SPI boot mode).

[Setting the Clock Oscillator]
Make the oscillator settings listed below.
Setting of boot mode: SPI boot
Setting of PLL1 circuit: Check "Operation".
Setting of low-speed on-chip oscillator (LOCO): Check "Operation".
Clock source: PLL1
CPU clock (CPUCLK): 450 (MHz)
External bus clock: 75 (MHz)



• Screenshot [3]: Clock Setting

Clock setting Debug interface setting Block of	diagram	
Doot mode setting	SPI hast	
Main alaak asseillater setting	SP10000	
Main clock oscillation source	肇振子/外部肇振	(Clock source is set based on the status of OSCTH)
Frequency	25	(MH2)
Oscillation atom detection function	Disabled	(00.67)
Oscillation stop detection function	Visabled	
- PLL0 circuit setting	1000	(MILL)
Frequency	1200	(10112)
PLL1 circuit setting		
Frequency	900	(MHz)
Low speed on-chip oscillator (LOCO) setting		
Operation		
Frequency	240	(kHz)
-Internal clock setting (Clock source is PLLO or	PLL 1)	
Clock source	PLL1 V	
CPU clock (CPUCLK)	450 ~	(MH2)
System clock (ICLK)	150	(MHz)
High-speed peripheral module clock (PCLKA	150	(MHz)
High-speed peripheral module clock (PCLKE	2 75	(MHz)
External bus clock (CKIO)	75 ~	(MH2)
Trace interface clock (TCLK)	150	(MH2)
Internal clock patting (Clock prurps in PLL0)		
High-speed peripheral module clock (PCLKC	150	(MH2)
Low-speed peripheral module clock (PCLKD	75	(MH2)
Low-speed peripheral module clock (PCLKE)	75	(MH2)
Low-speed peripheral module clock (PCLKF)		(MH2)
Low-speed peripheral module clock (PCLKG		(MH+)
Law appeal peripheral module clock (POLKU	Neo Vie	(1112)
List and asid she (CERICLE)	60	(MHz)
High-speed serial clock (SERICLK)	150 ~	(MHZ)
-IWDT clock setting		
IWDT clock (IWDTCLK)	120	(kHz)
-ECM clock setting	[]	
ECM clock (ECMCLK)	240	(kHz)
- Ethernet clock setting	105	(101-)
Ethernet Clock D (ETOLKD)	12.5	(19172)
Ethernet clock E (ETCLKE)	25 ~	(MHz)
-Delta-sigma clock setting		
Delta-sigma interface clock 0 clock source	PLL0 ~	
Delta-sigma interface clock 0 supply channe	Clocks input to MCLK0"2 pins	
Delta-sigma interface clock 0 (DSCLK0)	25 ~	(MHz)
DSCLK0 output polarity	Not inverted $\checkmark$	
Delta-sigma interface clock 1 clock source	PLL0 ~	
Delta-sigma interface clock 1 (DSCLK1)	25 ~	(MHz)
DSCLK1 output polarity	Not inverted	



[Setting the Bus State Controller]
Make the settings listed below.
General setting: Use Bus operation setting
Setting for CS0: SRAM
Setting for CS1: SRAM
Setting for CS2: SDRAM
Setting for CS3: Common with those for CS2
Setting for address pin selection (Address pin check setting): Group check
Setting for group selection (Address pin group check setting): A1 to A25

• Screenshot [4]: Settings for the bus state controller (General setting)

neral setting	CS0 CS	1 0	S2 CS3 C	S4 CS5							1 284 422
us operation	setting						-		_		
O Unused								Used			
ternal bus a	rea setting						_				
Use CS	30 (60000000h	to	63FFFFFFh,	40000000h	to	43FFFFFFh	mirro	SRAM			~
🔽 Use CS	31 (64000000h	to	67FFFFFFh,	44000000h	to	47FFFFFFh	mirro	SRAM			~
🔽 Use CS	\$2 (68000000h	to	6BFFFFFFh,	48000000h	to	4BFFFFFFh	mirr	SDRAM			~
Use CS	33 (6C000000h	to	6FFFFFFFh,	4C000000h	to	4FFFFFFFh	mirro	SDRAM			~
Use CS	54 (70000000h	to	73FFFFFFh,	50000000h	to	53FFFFFFh	mirror	SRAM			
	E /24000000	to	775555556	540000006	to	57FFFFFFb	mirro	SRAM			151
U Use Ca	55 (74000000n		in in the	010000000		*****					×.
Idress pin ch	neck setting										×
ldress pin ch	heck setting							) Discr	ete check		
Idress pin ch Group Idress pin er	heck setting check roup check set	ting _					(	) Discr	ete check		
Idress pin ch Group Group pins	heck setting check roup check set start with A1	ting	~	]		Group pins end	( I with	) Discr A25	ete check	~	
Idress pin of Group Group pins Idress outpu	heck setting	ting _	~	]		Group pins end	l with [	O Discr A25	ete check	~	
ddress pin ch Group Group pins Group pins ddress outpu A0	heck setting	ting _	~ ~ 42	) } A4		Group pins end	(   with [ 	<ul> <li>Discr</li> <li>A25</li> <li>A6</li> </ul>	ete check	~	
ddress pin cf Group ddress pin cr Group pins ddress outpu A0 A8	heck setting	ting		A4 1 ☑ A1	2	Group pins end ☑ A5 ☑ A13	(   with [ 	Discr     A25     A6     A14	ete check	~	
ddress pin el Group pins ddress pin er Group pins ddress outpu A0 A8 A16	heck setting	ting	A2 Z A3 A10 Z A1 A18 Z A1	A4 1 🗸 A1 9 🗸 A2	2	Group pins end	(   with [ 	Discr A25	ete check	~	



CS0:

Г

Area setting: Bus width of 16 bits Bus timing setting: Numbers of cycles for each parameter External wait setting: For [External wait mask], select [External wait input is ignored]

• Screenshot [4]: Settings for the bus state controller (Settings for CS0)

🛛 Generate code   🚠 🎬 🟙 🚔 🏹 🗊 🕜 🔇 🖗	0 0 0 0 0 0 0 0	J 🎬 J 📕 🍇	9 👊 📽 👊 🍣
General setting CSU CS1 CS2 CS3 CS4 C	\$85		
Area setting			_
Bus width	16 bits		~
Bus timing setting			
	Number of cycles	Period	
Number of access waits	6 V	80	(ns)
Delay from RD#, WEn# negation to address, CS negati	on 0.5 🗸	6.666667	(ns)
Delay from address, CS assertion to RD#, WEn# asser	tich 2.5 🗸	33.333333	(ns)
Idle insertion between read-read cycles in same CS	0 ~	0	(ns)
Idle insertion between read-write cycles in same CS	0 ~	0	(ns)
Idle insertion between read-read cycles in different C	S 0 ~	0	(ns)
Idle insertion between read-write cycles in different C	s 0 ~	0	(ns)
Idle insertion between write-read and write-write cycl	es 1 ~	13.333333	(ns)
External wait setting		,	
External wait mask	External wait input is ig	nored	$\sim$
External wait timeout detection	Disable		×
External wait cycles	255	3400	(ns)



#### CS1:

Area setting: Bus width of 16 bits Bus timing setting: Numbers of cycles for each parameter External wait setting: For [External wait mask], select [External wait input is ignored]

• Screenshot [4]: Settings for the bus state controller (Settings for CS1)

🕞 Generate code   🚠 🔐 🏙 🖓 🎜 🗱 🛞 🔇 🔇	0 Q. Q Q & &	J 🧌 J 🖩 😂 4	L 📽 ¼ 🍣
General setting CS0 CS1 CS2 CS3 CS4 CS	5		
Area setting			
Bus width	16 bits	~	
Buc timing cetting			
	Number of cycles	Period	
Number of read access waits	6 ~	80	(ns)
Number of write access waits	Same as read 🛛 🤟	80	(ns)
Delay from RD#, WEn# negation to address, CS negation	n 0.5 🗸	6.666667	(ns)
Delay from address, CS assertion to RD#, WEn# asserti	o <mark>n</mark> 2.5 🗸	33.333333	(ns)
Idle insertion between read-read cycles in same CS	0 ~	0	(ns)
Idle insertion between read-write cycles in same CS	0 ~	0	(ns)
Idle insertion between read-read cycles in different CS	0 ~	0	(ns)
Idle insertion between read-write cycles in different CS	0 ~	0	(ns)
Idle insertion between write-read and write-write cycles	s 1 ~	13.333333	(ns)
External wait setting		-	
External wait mask	External wait input is i	nored 🗸 🗸	
External wait timeout detection	Disable	~	
External wait cycles	255	3400	(ns)

#### CS2:

Area setting: Bus width of 16 bits for CS2 (default)

Area setting: Bus width of 16 bits for CS3 (default)

Type: Normal SDRAM (default)

Number of address bits: Numbers of address bits for rows and columns in each area

Mode setting: Burst read/burst write for each area

Clock select: CKIO/16

Constant register value for refresh time (Refresh compare match value): 36

Enable compare match interrupt: Release the checkmark

Bus timing setting: Numbers of cycles for each parameter



• Screenshot [4]: Settings for the bus state controller (Settings for CS2)

General setting CS0 CS1 CS3 CS4 CS5			
Area setting			
CS2 bus width	16 bits	~	
CS3 bus width	16 bits	~	
Туре	Normal SDRAM	~	
Bank active mode	Auto-precharge mode	~	
CS2 number of bits for row address	13 bits	~	I 1
CS2 number of bits for column address	10 bits	~	I 1
CS3 number of bits for row address	13 bits	~	I 1
CS3 number of bits for column address	10 bits	~	
Mode setting			_
CS2 mode register setting	Burst read/burst write	~	I 1
Enable CS2 extension mode register write command (EMRS)			•
CS2 EMRS command value	0×00000000		
CS3 mode register setting	Burst read/burst write	~	
Enable CS3 extension mode register write command (EMRS)			•
CS3 EMRS command value	0×1000000		
Bus refreshing setting			
✓ Enable refresh			
Refresh mode	Auto-refresh mode	~	1
Clock select	CKI0/16	~	J .
Refresh count	1	~	•
Refresh compare match value	36	7680	(ns)
Refresh request during DMA burst transfer	Accept the refresh requ	iest ~	
Enable compare match interrupt	Level 16	~	
Put timing optimg			
- Dus timing setting	Number of cycles	Period	
CS2 CAS latency	2 ~	26.666667	(ns)
CS3 CAS latency	2 ~	26.666667	(ns)
Auto-precharge startup wait cycles	2 ~	26.666667	(ns)
Auto-precharge completion wait cycles	1 ~	13.333333	(ns)
Idle states from REF command/Self-refresh release to ACTV/REF/MRS com	5 ~	66.666667	(ns)
Waits between ACTV command and READ(A)/WRIT(A) command	1 ~	13.333333	(ns)
CS2 idle insertion between read-read cycles in same CS	0 ~	0	(ns)
CS2 idle insertion between read-write cycles in same CS	0 ~	0	(ns)
CS2 Idle insertion between read-read cycles in different CS	0 ~	0	(ns)
CS2 idle insertion between read-write cycles in different CS	0 ~	0	(ns)
CS2 idle insertion between write-read and write-write cycles	0 ~	0	(ns)
CS3 idle insertion between read-read cycles in same CS	0 ~	0	(ns)
CS3 idle insertion between read-write cycles in same CS	0 ~	0	(ns)
CS3 Idle insertion between read-read cycles in different CS		0	(ns)
CS3 idle insertion between read-write cycles in different CS		0	(ns)
Cost and meeting bettern road white cycles in directing of		0	(ne)
CS2 idle incertion between write-read and write-write cucles		IV.	(ris)



Set the output of user functions for the same modules as in the sample program for the Initial Settings.

• Generate your own code to handle processing for the initial set-up (r\_xxx\_user.c).



By default, code is not generated for API functions of the code generation tool that are not displayed with an icon that includes a lock. Here, we describe the setting when the R\_BSC\_Create\_UserInit function is to be used. Select the API function in the code preview of the project tree, right-click on that entry, and select [Generate Code]. This causes the R\_BSC\_Create\_UserInit function to be generated in the r\_cg\_bsc\_user.c file at the time of actual code generation. When the function is selected, the icon of an open lock is displayed. For the details on usage of the Code Generation Tool, refer to *AP4, Applilet3 User's Manual: Common Operations* (R20UT3420EJ).

Created file: r\_cg\_bsc\_user.c

The statements for settings by API functions in the generated code will be explained in point (6), following the description of setting up and generating the code.



[Settings for the SPI Multi I/O Bus Controller]

General setting: Select "External address space read mode" as the module setting.

Transfer speed setting: Set "Base bit rate" as 75 Mbps

Transfer format setting: Settings for clock delay, etc.

• Screenshot [5]: Settings for SPI multi I/O bus controller (General setting)

General setting Setting		ovy 18 vy mer 4∞, 111 var 111 v⊂
-Function setting		***************************************
O Unused	<ul> <li>External address space read mode</li> </ul>	◯ SPI operating mode
- Transfer speed setting -		
Base bit rate	75000 (Kbps)	(Actual value: 75000, Error: 0%)
_Data alignment setting _		
🗹 Enable data swap	in 8-bit units	
-Transfer format setting		
SPBSSL signal polari	ty	Active low ~
SPBCLK output durin	g SPBSSL inactivation (CPOL)	Output level 0 ~
SPBCLK edges for da	ata transmission/reception (CPHAT/CPHAR)	Transmit at even edge/Receive at even edg 🗸
Period from activatio	n of SPBSSL signal to SPBCLK output (Clock delay)	1 SPBCLK cycle
Period from last edge	of SPBCLK signal to SPBSSL inactivation (SPBSSL negati	ion delay) 1.5 SPBCLK cycles
Period from transfer	end to next transfer start (Next access delay)	
r chou nom d'unator		

#### [Note]

To accelerate the serial flash memory, initially set the SPIBSC to the SPI operating mode and the serial flash memory to the quad I/O mode in the sample program for the initial settings. After that, re-set the mode to that of reading from the external address space.

On the other hand, in this procedure, the SPIBSC is set to be used in external address space read mode and the serial flash memory in single I/O mode. The command to be used is the FAST READ4B (0x0C) command.



Г

#### [Setting]:

Other types of settings are made in the red frames of the figure below.

• Screenshot [5]: Settings for the SPI multi I/O bus controller (Setting)

General setting Setting				
Read control setting				-
Read operation	Burst read	~		
Burst length	2	─ (64-t	oit blocks)	
SPBSSL inactivation condition	Inactivate SPBSSL after each	transfer	~	
Bit size setting				
Command bit-size	1-bit size (Command output or	n SPBMO pir	n) ~	]
Optional command bit-size	1-bit size (Optional command	output on Si	PBMOpin) 🗸 🗸	]
Address bit-size	1-bit size (Address output on	SPBMO pin)	~	]
Option data bit-size	1-bit size (Option data output	on SPBMO p	pin) v	]
Dummy cycle bit-size	1-bit size (Dummy Hi-Z on SP	BMI pin)	~	1
Data bit-size	1-bit size (Data input on SPB)	AIpin)	~	]
Data pin status setting				
•	Status during SPBSSL Inactiv	vation	Status for 1-bit/2-bit size	1
SPBIO0 pin	Output value Hi-Z	~	Output value Hi-Z 🗸	]
SPBIO1 pin	Output value Hi-Z	~	Output value Hi-Z	]
SPBIO2 pin	Output last bit of previous tran	nsfi 🗸	Output value 0 $\sim$	
SPBIO3 pin	Output last bit of previous tran	nsfi 🗸	Ouput value 1 $\sim$	
Data format setting				
Command	0x0C			
Address bit count	32 bits	~		
Optional command enable				
Optional command	0×00			
Option data enable				
Option data count	1 byte	~		
Option data	0×00			
🖂 Dummy cycle enable				
Dummy cycle count	8 cycles	~		
. 32-bit extended address setting .				
External address valid range	Bits [24:0]	$\sim$		
Upper address value (EAV)	0×00			



٦

#### (5) Pin Function Settings

Make settings for those pins for which the multiplexed function is actually to be used. Example for this Sample: Each Screen for Setting and Describing Pin Functions from [Device List View] (1/3)

Clock Generation Circuit								
Encoder I/F Interrupt Controller	Locked	Pin Function	Available Assignment Search Available Assignment	Q	Pin Number		Pin Direction	Pin Remark
Bus state controller		A0	Not assigned	•	Not assigned	•	-	
DMA Controller	~	A1	PG0/ A1/ PO2	*	R7	•	Out	
I/O Ports	~	A2	PG1/A2/PO3	-	V6	*	Out	
Port Output Enable 3	~	A3	PG2/ A3/ PO4/ TOC0/ RSPCK1	*	R8	•	Out	
General PWM Timer	~	A4	PG3/ A4/ PO5/ TIC1/ MISO1	•	Т8	*	Out	
16-Bit Timer Pulse Unit	-	A5	PG4/ A5/ PO6/ TOC1/ MOSI1	*	V7	*	Out	
Programmable Pulse Generator	~	A6	PG5/ A6/ TCLKA/ PO7/ SSL10	-	V8	-	Out	
Serial Communications Interface with FIFO	~	A7	PG6/ A7/ TCLKB/ PO8/ SSL11	*	Т9	-	Out	
I2C Bus Interface		A8	PG7/ A8/ PO9	-	R9	-	Out	
Serial Peripheral Interface		A9	PH0/ A9/ PO10		V9	-	Out	
SFI Multi I/O Bus Controller Δ Σ Interface		A10	PH1/A10/MTIOC28/PO11	-	V10	-	Out	
Error Control Module		A11	PH2/ A11/ MTIOC2A/ PO12	+	R10	-	Out	
12-Bit A/D Converter		A12	PH3/ A12/ MTIOC1B/ PO13	-	T10	-	Out	
Gigabit Ethernet MAC		A13	PH4/ IRQ4/ A13/ PQ14		R11		Out	
USB 20 HS Host/Function Module		A14	PH5/ A14/ PO15		T12	-	Out	
CAN Interface		A15	PH6/A15/ MTIOC7D/ RTS0#	-	R12	-	Out	
Serial Sound Interface		A16	PH7/ A16/ MTIC5W	-	V11	-	Out	
Others		A17	P20/447/HTCLKD	-	1/12	-	Out	
		A40		-	VIZ	-	Out	
		A10		-	T14	-	Out	
		A13	P20/A19/ M10C80/ DRC01	•	D44		Out	
		A20	P2// A20/ MITOCOC/ TIOCB0/ RTS0#	•	R14	•	Out	
		A21	PT6/A21/DREQ2	•	J20	•	Out	
		A22	PT// A22/ DACK2	•	J19	•	Out	
		A23	PK2/ A23	*	F15	*	Out	
	~	A24	PK3/ A24	•	G15	•	Out	
	~	A25	P97/ AN107/ IRQ7/ A25/ ADTRG1	•	E18	•	Out	
	~	D0	P00/ D0/ MTIOC6A/ TIOCA1/ ADTRG1/ TRACECTL	٠	U18	٠	In/Out	
	<ul><li>✓</li></ul>	D1	P01/ D1/ MTIC5W/ TIOCA2	*	V19	•	In/Out	
	<ul><li>✓</li></ul>	D2	P02/ D2/ MTIC5V/ TIOCA3	•	V20	•	In/Out	
	✓	D3	P03/ D3/ MTIC5U/ TIOCA4	*	U20	*	In/Out	
	✓	D4	P04/ D4/ MTIOC3C/ TIOCA5	•	U19	•	In/Out	
	✓	D5	P05/ D5/ MTIOC3A	•	V18	•	In/Out	
	-	D6	P06/ D6/ MTIOC2B/ TIOCB0	*	P15	•	In/Out	
	-	D7	P07/ D7/ MTIOC2A/ TIOCB1	•	P16	•	In/Out	
	-	D8	PE0/ D8/ MTIOC1B/ TIOCB2/ TRACEDATA0	•	T19	•	In/Out	
	~	D9	PE1/ D9/ MTCLKD/ TIOCB3/ SSL03/ TRACEDATA1	*	T20	•	In/Out	
	~	D10	PE2/ IRQ2/ D10/ MTCLKC/ TIOCB4/ SSL	•	N15	•	In/Out	
	~	D11	PE3/ IRQ3/ D11/ MTIOC0D/ TIOCB5/ CTS1#/ SSL	-	P18	-	In/Out	
	~	D12	PE4/ D12/ MTIOC0B/ TIOCC0/ RTS1#/ SSL	•	N16	-	In/Out	
	~	D13	PE5/ D13/ MTIOC0C/ TIOCC3/ TXD1/ MOS		N18	•	In/Out	
	~	D14	PE6/ IRQ6/ D14/ MTIOC0A/ TIOCD0/ RXD1/ MIS	-	M16	-	In/Out	
		D15	PE7/ D15/ MTIOC74/ TIOCD3/ POE8#/ SCK4/ PSPC	-	1.16	-	In/Out	



	CW I	riopenty	Device List view						
Clock Generation Circuit Encoder I/F	Locked	Pin Function	Available Assignment	Search Available Assignment 🔎	Pin	Number		Pin Direction	Pin Remark
Bus state controller	<b>V</b>	CS0#	P21/ IRQ1/ CS0#/ MTIC	5V/ TIOCB1/ CTS0#	- V1	3	*	Out	
DMA Controller	1	CS1#	PD1/AN109/CS1#		El	5	*	Out	
I/O Ports		CS2#	P45/ CS2#	,	V1	5	•	Out	
Multi-Function Timer Pulse Unit Port Output Enable 3		CS3#	PT4/ CS3#/ PO29		- M1	9	*	Out	
General PWM Timer		CS4#	Not assigned		No	assigned	*	-	
16-Bit Timer Pulse Unit		CS5#	Not assigned		Not	assigned		-	
Programmable Pulse Generator		RD#	P22/ IRQ2/ RD#/ MTIO	7B/ TIOCD0/ SCK0	- W1	4	*	Out	
Serial Communications Interface with FIFO		RD	P24/ IRQ12/ RD/ WR#/	RXD0	- W1	3	*	Out	
I2C Bus Interface		WR#	P24/ IRQ12/ RD/ WR#/	RXD0	- W1	3	*	Out	
Serial Peripheral Interface		BS#	P41/ BS#/ SCK0		Y1	5		Out	1
SPI Multi I/O Bus Controller Δ Σ Interface		AH#	Not assigned		No	assigned	•		
Error Control Module		WAIT#	Not assigned		No	assigned			
12-Bit A/D Converter		WE0#	P36/ WE0#/ DQMLL/ PO	00	- T7		•	Out	
Gigabit Ethernet MAC EtherCAT Slave Controller		WE1#	P37/ WE1#/ DQMLU/ P	01	- T6		*	Out	
USB 2.0 HS Host/Function Module		WE2#	Not assigned		No	assigned			
CAN Interface		WE3#	Not assigned		No	assigned	+		
Serial Sound Interface		DQMLL	P36/ WE0#/ DQMLL/ PO	00	- T7		•	Out	
Others		DQMLU	P37/ WE1#/ DQMLU/ P	01	- T6			Out	
		DQMUL	Not assigned		No	assigned			
	Π	DQMUU	Not assigned		No	assigned	+		
		RAS#	P90/ AN100/ RAS#/ TIO	CA5/ TXD4	F10	5	•	Out	
		CAS#	PK0/ CAS#/ PO31		H	9		Out	
		CKE	P46/ CKE		V1	5	+	Out	
		CKIO	P10/ IPO0/ CKIO/ TIOC	AN/TRACECIK	- V1		-	Out	

Example for this Sample: Each Screen for Setting and Describing Pin Functions from [Device List View] (2/3)

#### Example for this Sample: Each Screen for Setting and Describing Pin Functions from [Device List View] (3/3)

Clock Generation Circuit Encoder I/E	Locked	Pin Function	Available Assignment	Search Available Assignment	Q	Pin Number		Pin Direction	Pin Remarks
Interrupt Controller			-						
Bus state controller	-	SPBCLK	P62/ SPBCLK		•	W1	-	Out	
DMA Controller	<b>v</b>	SPBSSL	P60/ SPBSSL/ CTXD0/	TEND0	-	U1	-	Out	
I/O Ports		SPBIO0	P63/ SPBMO/ SPBIO0			112		In/Out	
Multi-Function Timer Pulse Unit		EDBIO4			_	1/2	_	la/Out	
Port Output Enable 3		PRIVI	P04/ SPDIVII/ SPDIUT		•	V2	•	in/Out	
General PWM Timer		SPBI02	Not assigned		•	Not assigned	•	-	
Programmable Pulse Generator		SPBI03	Not assigned		•	Not assigned	*	-	
Compare Match Timer W	-	SPBMO	P63/ SPBMO/ SPBIO0		*	U2	-	Out	
Serial Communications Interface with FIFO	<b>v</b>	SPBMI	P64/ SPBMI/ SPBI01		•	V2	-	In	
I2C Bus Interface									
Serial Peripheral Interface									
SPI Multi I/O Bus Controller									
$\Delta \Sigma$ Interface									
Error Control Module									
12-Bit A/D Converter									
Gigabit Ethernet MAC									
EtherCAT Slave Controller									
USB 2.0 HS Host/Function Module									
CAN Interface									



- Select [Device List View] from the icon on the tool bar to select the bus state controller and set BS# pin (PORT41). Setting of high driving ability output from CKIO pin (PORT10) is done in step (6), following the code generation.
- Note: Pins with multiplexed functions will have their default settings immediately after release from the reset state. Make sure that the multiplexed pin functions that are actually to be used are selected. Settings for pins which are not selected as [Locked] can be changed to another setting. To avoid unnecessary contention, please select the pins to be used as [Locked] (recommended) after checking. Make sure that there are no errors that will lead to contention and so on.
- (6) Code Generation and Modification with User-Defined Code

Generate the code after completing the settings from (1) to (5).

When the code is generated, the folder cg\_src is created immediately below the folder cg\_rc\_serial, in which the project of the Code Generation Tool generated. The source code and header code are generated in the former folder.

• Modifying generated code

The Code Generation Tool overwrites code whenever it generates new code. To merge code written by the user with newly generated code, it must be written between specific comment lines to protect it from being overwritten. [Merge file] is the default setting for file generation control.



The comment lines to indicate code to be merged are written in each file of code output by the Code Generation Tool as shown below. The comment lines protect the user code since the code between them is not overwritten in the case of merging with an existing file when code is generated again.

/* Start user code. Do not edit comment generated here */
$\leftarrow$ Write code between these lines.
/* End user code. Do not edit comment generated here */
Note: Please don't modify or move the comments for use in merging. If they are modified or moved, merging will not be successful.



[Setting an API function for the Bus State Controller]

The code shown below should be written in modifying the file created in step (4).

Target file: r\_cg\_bsc\_user.c

Target API function: R\_BSC\_Create\_UserInit

- Setting of high driving ability output for PORT10. Setting the pin function of the clock for the bus (CKIO)
- Setting of WCR for SDRAM to the CS2 space (BSC\_CS2WCR\_1) When the e<sup>2</sup>studio (with equivalent functionality to AP4 1.04) is used, the additional settings should be made here.
- Calling the API function for initial settings of the SDRAM (R\_BSC\_InitializeSDRAM) The API function for the initial settings of the SDRAM is called in the user's own code for initialization processing.

Write the following code between the comment lines to retain user code in the target API function during the process of merging.

/* Start user code. Do not edit comment generated here */				
/* Set PORT1 as high-drive output setting for connecting SDRAM */				
<pre>PORT1.DSCR.WORD = 1;</pre>	←Setting of high driving ability output for PORT10			
/* Set wait control register of CS2 space */				
BSC.CS2WCR.CS2WCR_1.LONG = 0x00000400;	←Setting of WCR for SDRAM to the CS2 space (required only for the e2studio)			
R_BSC_InitializeSDRAM();	←Setting of API function for initial settings of SDRAM			
/* End user code. Do not edit comment generated here $$	*/			

[Setting the API Function for Response to the Compare Match Timer]

The following code is written in the interrupt processing routine by modifying the file created in step (3). Target file:  $r_cg_cmt_user.c$ 

Target API function: r\_cmt\_cmi0\_interrupt

• Write the code for switching LED10 on or off in the interrupt processing routine for the compare match in response to operation of the cyclic counter.

Write the following code between the comment lines to retain user code in the target API function during the process of merging.





[Setting the API Function for the User Application Program]

After the code is generated, the file  $r_cg_main.c$  to contain the main processing is created by the Code Generation Tool under the folder  $cg_src$ . In this step, the file  $r_cg_main.c$  in the folder for the SPI boot mode version  $cg_src$  is needed to be modified.

In the main processing of the user application in the file r\_cg\_main.c, function R\_MAIN\_UserInit is called. For this sample program, the calling function of the main processing as stated in the function should be common to the 16-bit bus boot version and SPI boot (serial) version.

Target file: r\_cg\_main.c

Target API function: R\_MAIN\_UserInit

Added common main function name: user\_app\_main

Write the following code between the comment lines to retain user code in the target API function during the

```
process of merging.
void R_MAIN_UserInit(void)
{
    /* Start user code. Do not edit comment generated here */
    user_app_main();
    /* End user code. Do not edit comment generated here */
}
```

Note: The name of the function to act as the common main function can be as desired by the user, but it must be the same name as that of the common main function created in Section 6.3.2 (4), Creating a Common main File.



## 6.3.2 Incorporation in the Sample Program of RZ/T1 Group Initial Settings

The code generated by the Code Generation Tool should be incorporated in the EWARM version of the sample program environment for making initial settings.

As the example for this sample program, incorporation of the code generated and written in Section 6.3.1, Generating Code with the Code Generation Tool, is described below.

- Remark: From this point forward, the numbers of line and etc. are described on the basis of Rev. 1.30 of the sample program of the Initial Settings. When incorporating the code from the previous steps, use the latest version of the sample program of the Initial Settings at that point in time.
- Importing the Environment of the code created by the Code Generation Tool (Copying) Copy the project folder of the Code Generation Tool itself which was created in 6.3.1 (2), Creating a New Project, to the folder for storage (src) of the source files among the project files of the sample program of the Initial Settings.

Examples for this Sample Program:

Code Generation Tool Project name for 16-bit bus boot mode: cg\_src\_nor Code Generation Tool Project name for SPI boot mode: cg\_src\_serial



Note: Since the name of the folder created to hold the code generated by the Code Generation Tool is fixed to "cg\_src", use different project names for the 16-bit bus boot mode version (cg\_src\_nor) and the SPI boot mode version (cg\_src\_serial) to contain the separate versions of the sample program.

After the code is generated, each of the project folders will contain "cg\_src" folder and the workspace file (.cgp).

#### [Supplementary Note]

When code generated by the code generation tool is incorporated in the e<sup>2</sup>studio environment, move iodefine.h from the inc folder to immediately below the project folder. For details, see Table 6.2, Folder Structure of the Sample Program of the Initial Settings Following Incorporation of the Code from the Code Generation Tool (2/2), in Section 6.1.1, Project Settings.

(2) Editing the Loader Code in the Sample Program of the Initial Setting

Setting of the clock oscillator, the bus state controller, and of the SPI multi I/O bus controller which are used in the sample program of the Initial Setting should be used after having been replaced by those created in Section 6.3.1 (4), Setting Modules for the Clock Signals and Bus, with the settings made in the Code Generation Tool in that step. Therefore, the settings specified by the loader program in the sample program of the Initial Setting should be disabled (commented out).

• Edit the file loader\_init2.c in the common folder for the storage of source files (src) under the project folder (RZ\_T1\_init\_boot) of the sample program of the Initial Setting. The two points to be edited in the loader code are the 111th line and the 119th line, both in function loader init2.

- Note 1. If you plan to have the loader program speed up the clock and buses, it is possible to call the functions for various settings without commenting lines out. However, take care with this, as the settings of the clock and buses will be remade in the initial settings of later processing by the Code Generation Tool in this sample program. For details, see section 9 (1), Usage Note on the bus\_init() Function.
- Note 2. In the bus\_init() function, the setting of the serial flash memory is changed from the single I/O mode to the quad I/O mode. However, when the code is generated in accord with this procedure, the read command (FAST READ4B) is used under the condition of the serial flash memory being in single I/O mode. When the bus\_init() function is used to set the buses in the loader program, be sure to set it to quad I/O mode. For details, see section 9 (1), Usage Note on the bus\_init() Function.

(3) Move the Initial Settings Function for the Error Control Module (ECM) to the Loader Code In the sample program of the Initial Setting the function to set the ECM up is called from the main processing. In this program, the common main created by the user should replace the main of the sample program of the Initial Setting.

Therefore, move the function call for setting up the ECM (ecm\_init), which handles the initial setting up of the ERROROUT pin, to the initial settings part of the loader code.

• Move the ecm\_init function from init\_main.c to loader\_init2.c (or, if you are using the DS-5 IDE, cpu\_init.c). The file init\_main.c should be under the sample folder of the folder for the storage of source files (src) in the project file of the sample program of the Initial Setting.

Setting of extended pseudo-error 35 for ECM is not necessary, so it should be removed.

Processing of the extended pseudo-error for ECM in loader\_init2.c should also be removed.

The following lines of the file init\_main.c should be copied to the file loader\_init2.c.

Source of copying: File init\_main.c

• (1) Function definition, 86th line

void ecm\_init(void);

• (2) Calling function ecm\_init from the 117th (comment) and 118th (call) lines of the main function

```
/* Initialize the ECM function */
ecm_init();
```



• (3) The code for the ecm\_init function is from the 177th line to the 202nd line.

Destination file for copying: File loader\_init2.c

• Add the function declaration that was copied in step (1) to "Private variables and functions" as the 98th line.

• Place the call of ecm\_init() between the calls of function set\_low\_vec and function main in the processing by loader init2.

Add the call of function ecm\_init from step (2) as the 129th line.

```
void loader_init2(void)
{
 *snip*
    /* Set RZ/T1 to Low-vector (SCTLR.V = 0) */
    set_low_vec();
    /* Wait for ensuring the wait setting of ATCM */
    asm("dmb"); /* Ensuring Context-changing */
    /* Initialize the ECM function */
    ecm_init();
    /* Jump to _main() */
    _main();
}
```

Note 1. The processing of function ecm\_init is placed in the ATCM. Therefore, the DMB instruction is added to ensure the wait setting of ATCM, executed by R\_ATCM\_WaitSet() immediately before the processing.



• Add the code for function ecm\_init from step (3) before the end of the file. (Remove the setting of the extended pseudo-error from line 354 to line 359)

```
End of function copy_4byte
* Function Name: ecm init
*snip*
void ecm_init(void)
                           \leftarrowAdd function ecm_init.
{
 volatile uint8_t result;
                           ←Remove the declaration of result, since it is not used
 /* Initialize ECM function */
 R ECM Init();
 \leftarrowRemove the lines from here up to the assignment to result.
 /* Set extended pseudo error 35 */
 /* Enables internal reset configuration */
 result = R_ECM_Write_Reg32(ECM_COMMON, &(ECM.ECMIRCFG1.LONG), 0x00000004);
}
End of function ecm init
/* End of File */
```



• Remove the processing of the extended pseudo-error for ECM from the reset check function in loader init2.

```
* Function Name : reset_check
* Description : Check the reset source and execute the each sequence.
            When error source number 35 is generated, set P77 pin to High.
* Arguments : none
* Return Value : none
void reset check(void)
{
                              \leftarrow Remove the declaration of result, since it is not used
  volatile uint8_t result;
  volatile uint32 t dummy;
                                    ← Remove the declaration of dummy, since it is not used
  /* Check the reset status flag and execute the each sequence */
  if (RST_SOURCE_ECM == SYSTEM.RSTSR0.LONG) // ECM reset is generated
  {
     /* Clear reset status flag */
                               // Enable writing to the RSTSR0 register
     R RST WriteEnable();
     SYSTEM.RSTSR0.LONG = 0x00000000; // Clear reset factor flag
     R_RST_WriteDisable(); // Disable writing to the RSTSR0 register
     /* Check the ECM error source */
                                  ← Remove the if statement and the processing in case the condition
                                     is satisfied below
     if (1 == ECMM.ECMMESSTR1.BIT.ECMMSSE102) // Error source number 35 is generated
      {
      *snip*
      }
   *snip*
End of function reset_check
}
```



(4) Creating a Common main File

In user application programs created with the Code Generation Tool, different files are used in the 16-bit bus boot mode version and SPI boot mode version.

The main processing which is common to both the 16-bit bus boot mode version and the SPI boot mode version should be created in a common main file.

For this sample program, it should be created in the sample folder where the source files are stored (src) in the project file of the same sample program of the Initial Setting as file init\_main.c, since it is the common main function.

The names of files can be selected as desired by the user. In this sample program, the names of the functions to be created should be called from the user application program created with the Code Generation Tool so they should be as below.

Example for this Sample Program:

Common main function to be created: user\_app\_main File to be created: user\_main.c

Create the file as follows with reference to the user\_main.c file for this program.

$\sim$					
/**************************************	* * * * * * * * * * * * * * * * * * * *				
Includes <system includes=""> , "Project Includes"</system>					
***************************************					
<pre>#include "r_cg_cmt.h"</pre>	←Include file of definitions for the CMT0 Function				
#include "iodefine.h"					
#include "r_system.h"					
/**************************************	* * * * * * * * * * * * * * * * * * * *				
*snip*					
/**************************************	* * * * * * * * * * * * * * * * * * * *				
Private variables and functions					
***************************************	****************/				
<pre>void user_app_main(void);</pre>	←Declaration of function user_app_main.				
/**************************************	****				
/*************************************	****				
/*************************************	*******				
<pre>/************************************</pre>	*******				
<pre>/************************************</pre>	***************************************				
<pre>/************************************</pre>	**************************************				
<pre>/************************************</pre>	*************************************				
<pre>/************************************</pre>	*************************************				
<pre>/************************************</pre>	*************************************				
<pre>/************************************</pre>	*************************************				
<pre>/************************************</pre>	**********************/ ←Body of function user_app_main ←Start use of CMT0 by the sample program				
<pre>/************************************</pre>	*************************************				



(5) Setting the target source code from the Code Generation Tool to be compiled In the case of EWARM, set the source code from the Code Generation Tool to be compiled for the sample program in the initial settings by following the procedure below.

[Add Project Connection]

• Select [Add Project Connection...] from the [Project] menu. The [Add Project Connection] dialog box should be displayed.

Select [IAR Project Connection] for [Connect using] and click on [OK].

• Screen [5]-1 Adding a Project Connection



• The [Select IAR Project Connection File] dialog box should be displayed. Select [IAR Project Connection File] (.ipcf) and click on [Open].

The project file for connection includes the registered information of the source files. The files selected here should be created after the code has been generated in the following folder (recommended) by the Code Generation Tool.

Examples of folder names for reference: Folder name of 16-bit bus boot version: cg\_src\_nor Folder name of SPI boot mode version: cg\_src\_serial

The files to be compiled should be added as shown below after they are completed.



• Screen [5]-2 Example of the addition of files to be compiled



Remark: In DS-5 and e2studio, when a file is added to the folder created by the Code Generation Tool, the project automatically recognizes the file.



ſ

- (6) Setting exception from compilation of the main function which is used in the initial set-up
  - Select init\_main file in the work space and right-click on it. Select [Remove] to set the exception from compilation.

init_main.c	Options
DLib_Co	Make Compile
DLib_Th	Rebuild All Clean
r_icu_init	C-STAT Static Analysis >
r_port.h	Stop Build
→ h stdint.h	Add >
k xencodir	Remove Rename
General Strategy Carlos in Stra	Version Control System >
	Open Containing Folder
	File Properties
	Set as Active

- (7) Setting compilation of the common main function
  - Add user\_main.c to be compiled in the init folder.

bus_init_	Make Compile		Select user_main.o	c from specified folder	
	Rebuild All Clean		₩ init_main.c	2016/12/26 21:26	
-⊕ C spibsc_i -⊕ C spibsc_i	C-STAT Static Analysis >	-	₩ user_main.c	2017/02/06 15:10	
-@ C exit.c	Add >	Add Files	<		
-⊕ Cloader_init -⊕ Cr_atcm_init. -⊕ Cr_cpg.c	Remove Rename	Add Group	File Name(N): user_main.c	✓ Sou	rce Files (*.c;*.cpp;*.cc;*.h;*
-@ C r_ecm.c	Version Control System >				Cancer
He C r_mpc.c He C r_ram_init.c	Open Containing Folder File Properties				
L⊞ Byector.asm	Set as Active				
- 🕀 🗀 Output					
	•				
Uutput					



٦

- (8) Set the include path for incorporating of the newly added code output by the Code Generation Tool
  - Select [Options...] from the [Project] menu and select [C/C + + Compiler] → [Preprocessor].
     Add the folder created by the code generation tool to the setting for additional include directories with the path to the working files as in the example below.

Example for this Sample Program:

PROJ\_DIR\$\src\cg\_src\_serial\cg\_src

ect I-jet/JTAGjet Tools Wind	low Help	Options for node "RZ_T1_init_	serial_boot"	
Add Files Add Group		Category:	Castor C.	this as
Add Project Connection Edit Configurations		General Options	Multi-file Compilation	etungs
Remove		C/C++ Compiler	Language 2 Code Optimizations Output List Preprocessor Dia	•
Create New Project		Output Converter	Ignore standard include directories	
Add Existing Project		Custom Build	Additional include directories: (one per line)	_
Options	Alt+F7	Build Actions Linker	\$PROJ_DIR\$¥inc \$PROJ_DIR\$¥src¥cg_src_serial¥cg_src	
Version Control System	>	Debugger Simulator		
Make	F7	Angel	ľ L	
Compile	Ctrl+F7	CADI	Preinclude file:	
Rebuild All		GDB Server		***
Clean		IAR ROM-monitor	Defined symbols: (one per line)	
Batch build	F8	I-jet/JTAGjet	Preprocessor output to file Preserve comments	
C-STAT Static Analysis	>	TI Stellaris	Generate #line directives	
Stop Build	Ctrl+Break	Macraigor PE micro		
Download and Debug	Ctrl+D	Y		
Debug without Downloading			OK Cancel	
Attach to Running Target				
Make & Restart Debugger	Ctrl+R			
Restart Debugger	Ctrl+Shift+R			
Download	>			
SFR Setup				
Open Device Description File	>			
Save List of Registers				



Γ

#### (9) Rebuild All

• Select [Rebuild All] from the [Project] menu as shown below. Rebuilding should start after this is selected. Please check that there are no errors.

Add Files	
Add Group	
Import File List	
Add Project Connection	
Edit Configurations	
Remove	
Create New Project	
Add Existing Project	
Options	Alt+F7
Version Control System	>
Make	F7
Compile	Ctrl+F7
Rebuild All	
Clean	
Batch build	F8
C-STAT Static Analysis	>
Stop Build	Ctrl+Break
Download and Debug	Ctrl+D
Debug without Downloading	
Attach to Running Target	
Make & Restart Debugger	Ctrl+R
Restart Debugger	Ctrl+Shift+R
Download	>
SFR Setup	
Open Device Description File	>



#### (10) Downloading and Debugging

• Select [Download and Debug] from [Project] on the tool bar as shown below. After the emulator is connected, the program will be written to the external serial flash memory by the dedicated flash downloader. You can then start debugging.





#### 6.4 Fixed-Width Integer Types

 Table 6.6 lists fixed-width integer types for the sample code.

Symbol	Description
int8_t	8-bit signed integer (defined in the standard library)
int16_t	16-bit signed integer (defined in the standard library)
int32_t	32-bit signed integer (defined in the standard library)
uint8_t	8-bit unsigned integer (defined in the standard library)
uint16_t	16-bit unsigned integer (defined in the standard library)
uint32_t	32-bit unsigned integer (defined in the standard library)

#### Table 6.6 Fixed-Width Integer Types for the Sample Code

### 6.5 Function

Refer to the application note of the RZ/T1 Group Initial Settings products for the functions of the sample program of the Initial Setting that are also used in this sample program. Table 6.7 below only lists the function to be added to the sample program of the Initial Setting.

#### Table 6.7 Function

Function Name	
user_app_main	



# 6.6 Flowchart

## 6.6.1 Loader Program Processing

Figure 6.3 is a flowchart of processing by the loader program.



Figure 6.3 Loader Program Processing

Refer to the application note on initial settings for RZ/T1 group products for details of the flow of the loader program part.



### 6.6.2 Processing by the Application Program Created by the Code Generation Tool

Figure 6.4 is a flowchart of the application program created by the Code Generation Tool.



Figure 6.4 Application Program Created by the Code Generation Tool

RENESAS

#### 6.6.3 Common main Processing

Figure 6.5 shows the flowchart of the common main processing.



Figure 6.5 Common main Processing

## 6.6.4 Interrupt Processing by the Timer Selected by the User (CMT0)

Figure 6.6 is a flowchart of processing in response to interrupts from the timer selected by the user (CMT0)



Figure 6.6 Interrupt Processing by the Timer Selected by the User (CMT0)



# 7. Sample Program

The sample program can be downloaded from the Renesas Electronics website.



## 8. Related Documents

 User's Manual: Hardware RZ/T1 Group User's Manual: Hardware Download the latest version from the Renesas Electronics website.

RZ/T1 Evaluation Board RTK7910022C00000BR User's Manual Download the latest version from the Renesas Electronics website.

- Technical Update and Technical News Download the latest version from the Renesas Electronics website.
- User's manuals related to the development environment The latest version for the IAR integrated development environment (IAR Embedded Workbench<sup>®</sup> for ARM) is available from the IAR Systems website.

The latest version for the ARM integrated development environment (Development Studio 5<sup>TM</sup>) is available from the ARM website.

The latest version for the Renesas Electronics integrated development environment (e<sup>2</sup>studio) is available from the Renesas Electronics website.



## 9. Usage Note

#### (1) Usage Note on the bus\_init() Function

In this sample program, the bus\_init() function that is the base program of the initial setting sample program is used without lines commented out. Therefore, when the bus\_init() function is used while setting buses in the loader program, take care on the following points.

In the bus\_init() function, the setting of the serial flash memory is changed from the single I/O mode to the quad I/O mode. On the other hand, as the function is expected to operate in single I/O mode in this sample program, reading of the serial flash memory does not proceed normally. When the bus\_init() function is used, settings for the quad I/O mode are required as described in Section 6.3.1 (4), Setting Modules for the Clock Signals and Bus, [Settings for the SPI Multi I/O Bus Controller], such as setting 0xEC (4READ4B) in the command.



# Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/



Revision	History
----------	---------

Guide for Applying the Code Generation Tool to the Sample Program

_	<b>D</b> (		Description
Rev.	Date	Page	Summary
1.00	Jun. 30, 2017	—	First Edition issued
1.10	Apr. 11, 2018	2. Operating	Environment
		4	Table 2.1 Operating Environment, Tool for generating code: The version of the AP4 from Renesas, modified. Note modified.
		3. Related Ap	plication Notes
		5	Application note document numbers, added
		6. Software	
		13	6.1.3 Exception Processing Vector Table: The address of the 34-byte area, modified
		13	Table 6.4 Exception Processing Vector Table, Note: The address, modified
		16	6.3.1 Generating Code with the Code Generation Tool: The description on the code gener- ation tool, modified
		24	Screenshot [5]: Settings for SPI multi I/O bus controller (General setting), modified. Note modified.
		25	Screenshot [5]: Settings for the SPI multi I/O bus controller (Setting), modified
		27	Example for this Sample: Each Screen for Setting and Describing Pin Functions from [Device List View] (3/3), modified
		29	The description of [Setting an API function for the Bus State Controller] in section 6.3.1, modified. The description on the code, modified
		31	6.3.2 Incorporation in the Sample Program of RZ/T1 Group Initial Settings, Remark: The revision of the sample program of the initial settings, modified
		32	6.3.2, (2) Editing the Loader Code in the Sample Program of the Initial Setting: Note 1, modified. Note 2, added.
		33	6.3.2, (3): loader_init2.c processing: The waiting function for the wait setting of ATCM was added beween set_low_vec(); and /* Initialize the ECM function */. Note 1, added.
		9. Usage Not	9
		49	9. Usage Note, added

All trademarks and registered trademarks are the property of their respective owners.

#### General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

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

1. Handling of Unused Pins

Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
- 2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

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

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access
  these addresses; the correct operation of LSI is not guaranteed if they are accessed.
- 4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal.
   Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
- 5. Differences between Products

Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

— The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

#### Notice

- Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
- Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
- 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
- 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product 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; industrial robots; etc.

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

- 6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
- 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for velucating the safety of the final products or systems manufactured by you.
- 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
- 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
- 11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
- Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)



#### 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. Murphy Ranch Road, Milpitas, CA 95035, U.S.A. +1-408-432-8888, Fax: +1-408-434-5351 Renesas Electronics Canada Limited 9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3 Tel: +1-905-237-2004 Renesas Electronics Europe Limited Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +44-1628-651-700, Fax: +44-1628-651-804 **Renesas Electronics Europe GmbH** Arcadiastrasse 10, 40472 Düsseldorf, Germany Tel: +49-211-6503-0, Fax: +49-211-6503-1327 Renesas Electronics (China) Co., Ltd. Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China Tel: +86-10-8235-1155, Fax: +86-10-8235-7679 Renesas Electronics (Shanghai) Co., Ltd. Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China Tel: +86-21-2226-0888, Fax: +86-21-2226-0999 Renesas Electronics Hong Kong Limited Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-2265-6688, Fax: +852 2886-9022 Renesas Electronics Taiwan Co., Ltd. 13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan Tel: +886-2-8175-9600, Fax: +886 2-8175-9670 Renesas Electronics Singapore Pte. Ltd. 80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949 Tel: +65-6213-0200, Fax: +65-6213-0300 Renesas Electronics Malaysia Sdn.Bhd. Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +60-3-7955-9390, Fax: +60-3-7955-9510 Renesas Electronics India Pvt. Ltd. No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India Tel: +91-80-67208700, Fax: +91-80-67208777 Renesas Electronics Korea Co., Ltd. 17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea Tel: +82-2-558-3737, Fax: +82-2-558-5338