

RL78/G24

Flexible Application Accelerator (FAA) Tool Guide: CS+

Introduction

This guide describes the options that must be set for the build process and debugger of the flexible application accelerator (FAA) contained in RL78/G24. It also describes how to operate the debugger.

Target Device

RL78/G24 RL78/G24 Fast Prototyping Board

Chapter Composition

Chapter 1: Overview of Flexible Application Accelerator (FAA)

This chapter describes the overview of the flexible application accelerator (FAA) and program creation.

Chapter 2: Overview of build process and debugger of Flexible Application Accelerator (FAA)

This chapter describes the new project creation procedure and the options that must be set for the build process and debugger of the flexible application accelerator (FAA). It also describes how to operate the debugger.

Chapter 3: Debugger operation using sample project

This chapter describes debugging operations for FAA programs using the sample code and the sample script.

Related Documents

- RL78/G24 User's Manual: Hardware (R01UH0961)
- RL78/G24 Fast Prototyping Board User's Manual (R20UT5091)



Contents

1. Overview	
1.1 Flexible Application Accelerator (FAA)	
1.2 Internal Memory Space of FAA	
1.3 Program for RL78/G24	
1.3.1 Program Structure	
1.3.2 Transfer of Program and Data for FAA	
1.3.3 FAA Program	
1.3.4 Build Process and Debug of FAA Program	6
2. Option Setting and Operation	.7
2.1 Operating Environment	7
2.2 Project Creation	7
2.3 Adding FAA Program	9
2.3.1 Adding FAA Component	9
2.3.2 Overview of FAA library's File Structure	8
2.4 Build Tool Option Setting	19
2.4.1 FAA Assemble Options	20
2.4.2 Link Options	21
2.4.3 Program Building	23
2.5 Debug Tool Option Setting	24
2.5.1 Connect Settings	24
2.5.2 Debug Tool Settings	25
2.5.3 Download File Settings	26
2.5.4 Program Download	27
2.6 FAA Program Debug	28
2.6.1 Debug Target	28
2.6.2 Source File Display	30
2.6.3 Go/Stop	31
2.6.4 Breakpoint	32
2.6.5 Memory	33
2.6.6 Symbol (Label)	33
2.6.7 Register	36
2.6.8 SFR	36
3. Sample Project	38
3.1 Specifications	
3.1.1 Specification Overview	
3.1.2 Operation Overview	
3.2 Operation Confirmation Conditions	
3.3 Hardware Description	



3.3.1	Example of Hardware Configuration	41
3.3.2	List of Used Pins	41
3.4	Software Description	42
3.4.1	Smart Configurator Setting	42
3.4.1.1	Clock	42
3.4.1.2	System	43
3.4.1.3	Component	43
3.4.2	Folder Structure	46
3.4.3	Option Byte Settings	46
3.4.4	List of Constants	47
3.4.5	List of Variables	47
3.4.6	List of Functions	48
3.4.7	Function Specification	48
3.4.8	Flowchart	49
3.4.8.1	Main Process	49
3.4.8.2	r_Config_TKB0_end_count_interrupt Function	. 50
3.4.8.3	FAA Processing	. 51
3.5	Sample Script Specification	. 52
3.5.1	SFR Display Overview	. 52
3.5.2	Operation Overview	. 53
3.5.3	List of Functions	. 55
3.5.4	List of Variables	. 55
3.5.5	Flowchart	. 56
3.5.6	Script Execution	. 61
3.5.7	Basic debug operations	. 62
3.5.8	Cautions When Using the Sample Script	. 64
4. S	ample Code	65
4. 3		.00
5. R	eference Documents	.65
Revisi	on History	.66



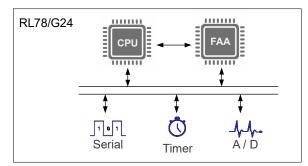
1. Overview

1.1 Flexible Application Accelerator (FAA)

The flexible application accelerator (FAA) contained in RL78/G24 is a Renesas original application accelerator with a Harvard architecture. It can execute 32-bit multiplication, addition, and subtraction in a single cycle.

FAA can access some peripheral functions directly by the address bus select function. Operations by the CPU and FAA can be combined to suit the application, it can improve operation efficiency of the system.

Figure 1-1 Image diagram of RL78/G24 FAA

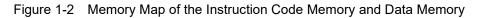


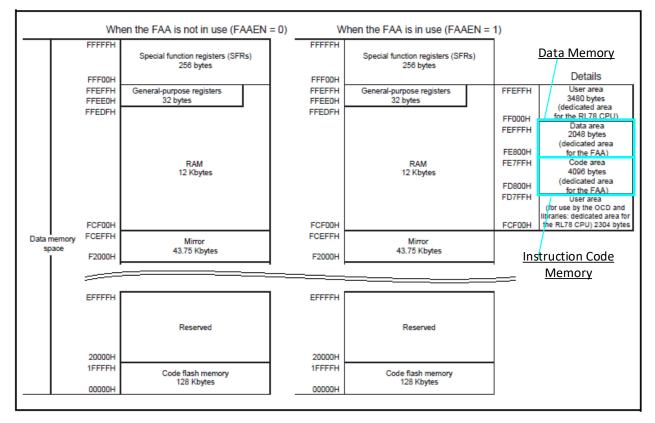
1.2 Internal Memory Space of FAA

When the FAA is in use, some of the RL78/G24's internal RAM is dedicated to the FAA.

Instruction Code Memory: Store the program for FAA

Data Memory: Store the data for FAA





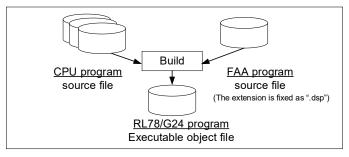


1.3 Program for RL78/G24

1.3.1 Program Structure

Programs for the CPU and programs for the FAA are coded in separate files. FAA programs use the FAAdedicated instruction sets. CPU programs and FAA programs are built together in an object file (load module file) that can be executed in RL78/G24.

Figure 1-3 Program structure when FAA is in use

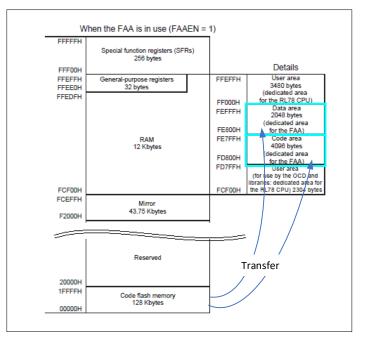


Remark. For instruction sets for FAA, refer to the chapter for FAA in RL78/G24 User's Manual: Hardware (R01UH0961).

1.3.2 Transfer of Program and Data for FAA

An executable object file is written to the RL78/G24 code flash memory. However, FAA programs must be placed in the instruction code memory and FAA data must be placed in the data memory. Therefore, before executing an FAA program, the FAA program and data stored in the code flash memory must be transferred to the instruction code memory and data memory, respectively.

Figure 1-4 Transfer of the program and data for FAA



Remark. FAA component in the RL78 Smart Configurator provides API functions for transfer processing.



1.3.3 FAA Program

You can create an FAA program by either of the following ways:

- Use a provided FAA library according to the purpose. The library is provided in a source file in which code cannot be changed. (FAA library of various function)
- Use a template file to code your own FAA program. (Template (Custom FAA library))

In both cases, add the FAA program to the program project by using the Smart Configurator (SC).

For details about how to use the Smart Configurator (SC) to output an FAA program file (library or template), see 2.3 Adding FAA Program.

1.3.4 Build Process and Debug of FAA Program

To build and debug FAA programs, some options must be set up. This guide describes the options that must be specified for the processing shown in Figure 1-5. It also describes how to use the debugger for debugging FAA programs.

Note that this guide requires the use of FAA programs (libraries or templates) generated by the Smart Configurator (SC).

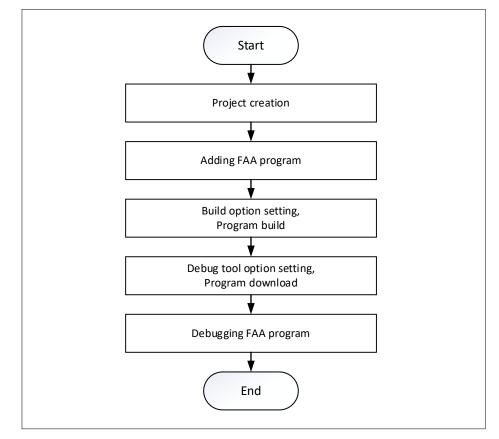


Figure 1-5 Operating instruction in chapter 2 of this guide



2. Option Setting and Operation

This chapter explains the option settings and debugger operation required for building and debugging an FAA program in the CS+ for CC environment.

For options that are not described in this guide, set them if necessary. For details about the options and operations, see the help or documentation of CS+ for CC.

2.1 Operating Environment

This guide uses the following tools:

Integrated development environment	Item	version
CS+	CS+ for CC Manufactured by Renesas Electronics	V8.10.00
	CC-RL Manufactured by Renesas Electronics	V1.12.01
	DSPASM FAA/GREEN_DSP Structured Assembler	V1.04.02
	Manufactured by Renesas Electronics	
	RL78 Smart Configurator Manufactured by Renesas Electronics	V1.8.0

Table 2-2 Hardware tool

Board / Emulator	Item
Board	RL78/G24 Fast Prototyping Board Manufactured by Renesas Electronics
Emulator Note1	E2 emulator Lite Manufactured by Renesas Electronics
	E2 emulator Manufactured by Renesas Electronics

Note1. When the debugger and the RL78/G24 Fast Prototyping Board are connected via COM port, the emulator is not required.

2.2 **Project Creation**

Select the RL78/G24 product as the microcontroller to be used and create a program project.

Procedure:

- 1. Launch the CS+.
- 2. Select [File] menu -> [New] -> [Create New Project] of CS+.

Figure 2-1 [File] menu -> [New] -> [Create New Project]

	CS+ for CC - [Start]	
File	Edit View Project Build Debug Tool	Window Help
	New	Create New Project
	Open Ctrl+O	Create New Project for Multi-core
	Open with Encoding	Create New File Ctrl+N
	Vdd	



3. In the [Create Project] dialog, select the RL78/G24 products, input the project name and click the [Create].

Figure 2-2 [Create Project] dialog

Microcon <u>t</u> roller:	RL78
Using microcontroller:	
₩ G24	Update
R7F101GBG R7F101GG R7F101GG R7F101GG R7F101GG R7F101GG R7F101GG R7F101GG	ixxNP(40pin) Internal ROM size[KBytes]:128 ixxFP(44pin) Internal RAM size[Bytes]:12288 ixxFB(45pin) ixxFA(52pin) ixxFB(64pin) ixxFB(64pin)
Kind of project:	Application(CC-RL)
Project <u>n</u> ame:	sample 1
P <u>l</u> ace:	CA Browse
	Make the project folder
C:\sample1\sample1.mt	pj
Pass the file composi	tion of an existing project to the new project
Project to be passed:	(Input project file to be diverted.) Browse es in the diverted project folder to a new project folder.

After creating the project, change the debug tool you use. In 2.3 Adding FAA Program, the Smart Configurator (SC) sets some options for the debug tool, so you must first select the debug tool you want to use.



Project Tree	₽	×		
ê 🕜 🙎 📓				
🧮 R7F101GLGxFB (Microcontro	ller)			
🖳 Smart Configurator (Design T	ool)			
🔨 CC-RL (Build Tool)				
🖑 Program Analyzer (Analyze T		Jsing Debug Tool 🔹 🕨		RL78 E2
🖮 🎒 File		Property		RL78 E2 Lite
cstart.asm			+	RL78 COM Port
🔤 hdwinit.asm				
stkinit.asm			\checkmark	RL78 Simulator



2.3 Adding FAA Program

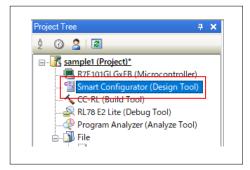
Use the Smart Configurator (SC) to add an FAA program (library or template) to your project.

This guide only describes the procedure for adding an FAA program, [Clock], [System] and [Voltage detection] that need to be set in the CPU program. Please set other peripheral functions as appropriate to suit your system.

2.3.1 Adding FAA Component

Procedure:

- 1. In the CS+ Project Tree, double-click [Smart Configurator (Design Tool)] to launch the RL78 Smart Configurator.
- Figure 2-4 Launch Smart Configurator



2. In the Smart Configurator (SC), click [Clock]. Set various clocks and the operation mode according to your system.

Figure 2-5 Smart Configurator: [Clock] tab

*sample1.scfg ×	: 18
locks configuration	🔂 👜 Generate Code Generate Rep
Operation mode: High-spead main mode 40(1)=4.5(1) *	
E100 Methy: 40 V & EVD00 & E3 V *	12 (MHQ) Higgs
High-speed on-chip oscillator Frequency: 82 * Minzy	PL secher dout
HOCO start satting. Normal +	Frequency 10 + 0/420
There is setting for starting the high-speed en-chip escillator at the times of release from \$100 mode and of transitions to	12000 (HM2)
sit the times of neese than shore most and of the states to SNOOZE most)	Diviser MAAN
Middle-speed on-ohip secliter Fegurer: 4 (0Hz)	12000 (HH)
Leftered. v	. (M8)
X1 oscillator	
Operation model Int autilation	
Prepuency: \$ (Meg)	micot (MNI)
Bable time: 2115/0 + 82428.8(n)	
Low-speed on-chip oscillator	TL 0/H21
Prequency: 12.768 (H1) The Ts, runs while WDT is operating or 10/P select Lew-speed	
on-thip excitator	1277 12768 (Mt)
XT1 oscillator	50X
Operation mode:	
XT1 occiliation mode: Low power concurrentian 1 *	
Supply made: Distilias supply in STOT HALT made	
	Screen size can be adjusted by placing the mouse
rerview Board Clocks System Components Pins Interrupt	cursor on the screen, holding down the "CTRL" key
Console ×	and moving the mouse wheel up or down.



3. Click the [System]. In the [System] tab, set the debug tool and functions to be used, and security ID.

Figure 2-6 Smart Configurator: [System] tab

ystem configuration			🔋 🔐 Generate Code 🛛 Generate Repo
è			
 On-chip debug setting 			
On-chip debug operation set	tting		
○ Unused	Use emulator	O COM Port	
Emulator setting			
○ E2	E2 Lite		
Pseudo-RRM/DMM function	setting		
◯ Unused	Used		
Start/Stop function setting			
Unused	⊖ Used		
Monitoring point function set	tting		
Unused	OUsed		
Trace function setting			
○ Unused	Used		
Security ID setting			
Use security ID			
Security ID	0x000000000000000000000000000000000000		
Security ID authentication fai	ilure setting		
O Do not erase flash memor	ry data		
Erase flash memory data			

4. Click the [Component]. Next, click the [Add component] to open the [New Component] dialog.

Figure 2-7 Smart Configurator: [Component] tab

💰 Smart Configurator	🐻 New Co	omponent				\times
File Window Help <u>R</u> un	Software	Component Selection				
📩 🗁 🔚	Select con	nponent from those available in list			- t	۳.
*sample1.scfg ×						
Software component configuration	Category	All				~
	Function	All				~
Components 🚵 🖾 $l_Z^a = 🗈 configure$	Filter					_
56 1d	The					
type filter text	Compone	ents	Short Name	Туре	Version	^
	🖶 A/D C	Converter		Code Generator	1.4.1	
V 🗁 Startup			r_bsp	RL78 Software In	1.61	
V 🗁 Generic	🖶 Clock	Output /Buzzer Output Controller		Code Generator	1.4.0	
💣 r_bsp				Code Generator	1.3.1	
	Components Short Name Type Image: Application Ackages. Short Name Type Image: Application Ackages. r.bsp RL78 Software In Image: Application Accelerator Code Generator Code Generator Image: Application Accelerator Code Generator Code Generator Image: Application Accelerator Faxion Accelerator FAX Configurator Image: Application Accelerator FAX Configurator FAX Configurator Image: Application Accelerator Code Generator Code Generator Image: Application Accelerator FAX Configurator FAX Configurator Image: Application Accelerator Code Generator Code Generator Image: Application Accelerator FAX Configurator Code Generator Image: Application Accelerator Code Generator Code Generator Image: Application Accelerator Code Generator Code Gene	1.3.0				
		1.1.0				
				Code Generator	1.1.0	
				Code Generator	1.3.1	
	🖶 Delay	Counter		Code Generator	1.4.1	
				Code Generator	1.4.1	
				Code Generator	1.2.0	
	🖶 Extern	nal Event Counter			1.4.1	
Verview Board Clocks System Components Pins Interrupt				FAA Configurator	1.0.0	
verview board clocks system components wins interrupt					1.5.1	
				Code Generator	1.4.1	
					1.2.0	~
		D. J		C-4- C		_
		only latest version				
	Descriptio					
	Download	og to digital (A/D) converter is func I RL78 Software Integration System general settings		alog inputs to digital signa	als.	~
	?	< <u>B</u>	ack <u>N</u> ext >	<u> </u>	Cancel	1



5. In the [New Component] dialog, select [Voltage Detector] and click the [Next].

Figure 2-8 Select [Voltage Detector]

Category	All				\sim
Function	All				\sim
Filter					
Compor	ents	Short Name	Туре	Version	^
# Phase	Counting Mode		Code Generator	1.0.0	
# Ports			Code Generator	1. 4.0	
# Progra	ammable Gain Amplifier		Code Generator	1.0.0	
B PWM	Option Unit A		Code Generator	1. 2.1	_
🖶 SPI (C	SI) Communication		Code Generator	1.4.0	
# Squar	e Wave Output		Code Generator	1. 4.0	
	phase PWM Output		Code Generator	1.2.0	
	Communication		Code Generator		1
L	e Detector		Code Generator	1.3.0	
+ water	idog Timer		Code Generator	1.4.0	~
Show (only latest version				
Descriptio	*				
	age detector is a function tha and generates internal intern			on	< _ >
Download	d RL78 Software Integration S	system modules			
	general settings				

6. Select the [LVD0] at the [Resource]. Check the configuration name and click the [Finish]. (The configuration name can be changed to any name.)

Figure 2-9 Select resource and check configuration name [Voltage Detector]

Voltage Detector Configuration name: Config_LVD0 Resource: LVD0 ~	Add new configura	tion for selecte	ed component		
	Configuration name:				~



7. The Voltage Detector is added to the component tree. In the settings screen, set the Voltage Detector according to your system.

Figure 2-10 Smart Configurator: [Voltage Detector] setting screen

Software component configura	ation		Generate Code	Generate Repor
Components 🚵 🗳 🖓 🕀 🕀	Configure			Œ
Image: Startup ✓ Ima	O Interrupt mode If LVD0 is set to interrupt mode and	et the detection voltage of LVD1 higher than t the LVD0 detection voltage is greater than the VD1 setting following release from the reset s	e LVD1 detection voltag	
Power management and rese	INTLVI priority	Level 3 (low)		
Config_LVD0	Voltage detection setting			
	Reset generation level(VLVD0)	1.65 ~	(V)	
	Interrupt generation level(VLVD0)	1.65 ~	(V)	
< >>				

8. Open the [New Component] dialog again, select the [Flexible Application Accelerator] and click the [Next].

Figure 2-11 Select [Flexible Application Accelerator]

					V
Function	All				~
Filter					
	. ^		-		•
Compone		Short Name	Туре	Version	^
	ansfer Controller		Code Generator	1.2 1	
Delay C			Code Generator	1.4	
Divider			Code Generator	1.4	
	ink Controller		Code Generator	1.2	
	Event Counter		Code Generator		
	Application Accelerator		FAA Configurat	1.0.0	
	nmunication (Master mode)		Code Generator	1.5	
	munication (Slave mode)		Code Generator	1.4	
	apture Function		Code Generator	1.2	
	ulse Interval/Period Measurem		Code Generator	1.4	
	gnal High-/Low-Level Width		Code Generator	1.4	
	ot Controller		Code Generator	1.4	
Interval			Code Generator	1.4	*
	nly latest version				
Description					_
	le application accelerator (FAA) s. It can execute 32-bit multiplic				~ ~
Download	RL78 Software Integration Syste	m modules			
	nero sonmare integration syste	in mounes			



9. Check the configuration name and click the [Finish]. (The configuration name can be changed to any name.)

Figure 2-12 Select resource and check configuration name [Flexible Application Accelerator]

ổ New Componen	t			
Add new configu	ration for select	ted component		
Flexible Applicatio	n Accelerator			
Configuration name	e Config_FAA			
(?)	< Back	Next >	Finish	Cancel
\odot	< Dack	TACVE >	Lunan	Cancer

10. The Flexible Application Accelerator is added to the component tree.

Figure 2-13 Add [Flexible Application Accelerator] component

} *sample1.scfg × Software component confi	juration	Generate Code Generate Repo
Components 🗈 🖬 🗠 🖽	Configure	
Ne type filter text	∲ i Please download FA	A data
 Startup Generic r_bsp Drivers Power management and re Config_LVD0 Middleware 		
 EAA Config_FAA 		
< >		



11. When the FAA component is used for the first time, the download of FAA libraries or template from the configurator's dedicated server is needed. Click the [Update FAA modules] or the [Please download FAA data] to download them. (Please use the [Update FAA modules] to check and obtain the latest version libraries as well.)

Figure 2-14 Update/Download FAA module (Library)

*sample1.scfg ×		
Software component config	uration	Generate Code Generate Repo
Components 🚵 🖄 P _Z 🕞 🕀	Update FAA modules	Q
type filter text	Please download FAA data	
Kerview Board Clocks System Compo	onents Pins Interrupt	

12. Select the library you want to download and click the [Download]. In the disclaimer dialog that follows, click the [Agree].

Figure 2-15 Download FAA module (Library)

Title Calculation Filter Library Custom FAA Library RL78/G24 Common FAA Module	Version 1.0.0 1.0.0 1.0.0 1.0.0		Select All Deselect All
--	---	--	----------------------------

Remark. The content displayed on the actual download screen will differ.

Table 2-3 FAA library

Title	Overview
RL78/G24 Common	The FAA program and data transfer routine described in 1.3.2 Transfer of
FAA Library	Program and Data for FAA. When using FAA libraries/templates, this is always downloaded.
Custom Library	A template for writing FAA programs.
Others	FAA library of various function



13. The downloaded libraries are added. ("RL78/G24 Common FAA Module" is not displayed.)

Figure 2-16 Added FAA library

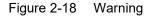
oftware component configuration				
components	Configure			
👫 🕼 🔹 🥫	> +tt Calculation > +tt Custom Library <u>Please download FAA data</u>			
 ✓ Construint ✓ Co				
 Power management and re Config_LVD0 Middleware 				
✓ I Middleware ✓ I FAA				

14. Check the box which libraries/functions you will actually use among the downloaded libraries. If there are any setting items in the properties of the checked function, set them as appropriate.

Figure 2-17 Select/set FAA library

oftware component co	inguration			Generate Code Genera	ate Repo
Components 🛛 🚵 🗳 🖓 🗉	Configure				0
5i 🞼 😜	Calculation		≪µ i		
type filter text	PfcPi Calculation	\mathbf{X}	Property	Value	
✓ ➢ Startup	✓ •t ^a Custom Library		✓		
🗸 🗁 Generic	Template		# FAA operation status checking	Disabled	
💣 r_bsp		-	# Enable Max3 calculation	Enalbed	
V 🗁 Drivers			# Enable Max2 calculation	Disabled	~
Power management and Chock of the contract				/ Disabled	
	the box which libraries/ to use.		/	Enalbed	
✓ ➢ Middleware Teature ✓ ➢ FAA	s to use.		/		
Config_FAA			Set selectable item	is as necessary	
			(reflected in the de		>
			Enable Max2 calc		^
			If set to enabled, generating Max2 calculation	APIS.	
<	> <	>			~

Remark. Two types of libraries and functions are provided: The subprocessor type, which can be used in conjunction with other functions, and the standalone type, which cannot. Do not use the standalone type simultaneously with any other library or function. When a standalone library or function is selected, selecting another library or function causes the following message to appear on the [Console] page.



Console ×	🖳 🚮 🚱 📑 🕻	2 - 📬 - 🖱	
nart Configurator Output			
04050006: Template feature is independent operation type. The independent type cannot be used in combination with	other FAA fe	ature.	1
		>	



15. Click the [Generate Code] to generate source files of FAA library and added peripheral functions.

Figure 2-19 Generate C

Software component confi	guration		Generate Code	Generate Report
Components 🛛 🚵 🗳 🖓 🖽 🖽	Configure			()
54 📸 👈 🖜	✓ • t Galculation ✓ Max Calculation	🍫 i		
type filter text	PfcPi Calculation	Property v @ Configuration	Value	
		# FAA operation status checking	Disabled	
		# Enable Max3 calculation	nable Max3 calculation Enalbed	
 Drivers Power management and Config_LVD0 Middleware FAA 		# Enable Max2 calculation	Disabled	
Config_FAA		<		>
		Calculate the maximum value.		^
< >	< >			>

16. When the [Confirmation linker option change] dialog appears, click the [OK].

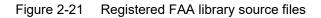
Figure 2-20 [Confirmation linker option change] dialog

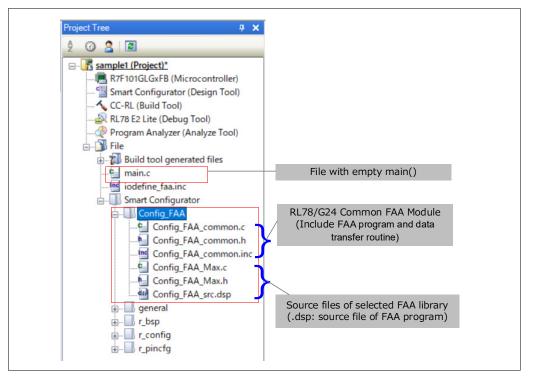
Software compon	ent config	Confirm linker option change	e	- 🗆 X	🔞 👜 Generate Code Generate Rep
Components 🛛 🔛 🗳					
type filter text ✓ ➢ Startup ✓ ➢ Generic ៷ r.bsp	Progress Infor	Setting User option byte value Option byte values for OCD Set debug monitor area Control allocation to self RA	Old value - No No	New value EFFBE8 84 Yes(Specify Yes(Error m	ed
 Drivers Power manag Config_LVI Middleware EAA 			ОК	Cancel	ed Cancel

Remark. Some items set in Smart Configurator's the [Clock], the [System] and the [Voltage Detector] (LVD0) are reflected in option settings of the build tool (CC-RL).



17. Source files of the FAA library and added peripheral functions are generated and registered in the project. The FAA library source files are shown below.





- Remark. For files other than the red frame above, refer to RL78 Smart Configurator User's Guide: CS+ (R20AN0580).
- 18. API functions to control the FAA are defined in the FAA library source file. Call these functions in the CPU program to operate the FAA. Create a CPU program according to your system.



2.3.2 Overview of FAA library's File Structure

The overview of the FAA library file structure is shown below.

	Table 2-4	Overview of FAA library's file structure
--	-----------	--

Library name	Files	Description
RL78/G24 Common FAA Library	<config_faa>_common.c <config_faa>_common.h</config_faa></config_faa>	The transfer processing and common functions to control the FAA are defined. The transfer processing is executed within the peripheral function initialization function (R_Systeminit) generated by SC, so there is no need to call it within the user program.
	<config_faa>_common.inc</config_faa>	SFRs for FAA are defined.
Custom FAA Library	<config_faa>_src.dsp</config_faa>	The template for the FAA source file.
Others	<config_faa>_XXX.c / asm / s <config_faa>_XXX.h /inc <config_faa>_src.dsp</config_faa></config_faa></config_faa>	FAA library of various functions. Refer to documents of each FAA library.

- <Config_FAA> is the configuration name set/checked in the step 9.
- "XXX" depends on each library.
- In the FAA source file (.dsp) provided by the FAA library and the template (Custom FAA Library), the code section name is defined as FAACODE and the data section name is defined as FAADATA.
- When using the Custom FAA Library, add your user code and data to the template. If you build the template as is, an error will occur.



2.4 Build Tool Option Setting

Before starting a build, set the build tool options required to build the FAA program. Some options are set by the Smart Configurator (SC) in 2.3.1 Adding FAA Component. Manually set the options for which "No" is indicated in the "Set by SC" column in Table 2-5.

For build tool options that are not described in this guide, set them if necessary.

How to open the build tool property:

Select the build tool node in the project tree, and then select the [View] menu -> [Property] or select the [Property] from the context menu.

Figure 2-5 shows the build tool options required to build the FAA program.

Tab	Category	Item	Description	Set by SC
FAA Assemble Options	Preprocess	Method for recognizing the text macro	Exact(-macro_identify exact)	Yes
Link Options	Section	Layout section automatically	Yes(-AUTO_SECTION_LAYOUT) or No	No
		Section start address	FAACODE,FAADATA/XXXX XXXX (hexadecimal number without "0x") specifies an even address after address D8H in the code flash memory.	No
		ROM to RAM mapped section	FAACODE=FAACODER FAADATA=FAADATAR	Yes
		Allocate FAA memory area automatically	Yes or Yes(Automatically allocate sections by striding FAA memory area) ^{Note2}	Yes Note1

Table 2-5 Setting options of build tool

Note 1. SC sets "Yes".

2. When the RAM size used by the user program (CPU program) is larger than 2304 bytes (the user RAM area before the FAA code area on RAM), manually set it to "No". Also, when "Yes(Automatically allocate sections by striding FAA memory area)" is specified, the setting "No" of "Layout section automatically" is ignored.



2.4.1 FAA Assemble Options

Figure 2-22 FAA Assemble Options

Debug Information					
Preprocess					
Include paths	Include paths[1]				
First character of text macro					
Text macro definition	Text macro definition[0]				
Allow to redefine text macro	No				
Method for recognizing the text i	nacros Exact(-macro_ident	ify exact)			
Output Code					
Output File					
Others					
thod for recognizing the text m	Let ne				
cifies the method for recognizing					

Table 2-6 FAA Assemble Options, Overview of settings

Category	Item	Description
Preprocess	Metho for recognizing the text macros	Set "Exact(-macro_identify exact)".
		A text macro is replaced in the FAA source file in units of tokens. Unless Exact is specified, replacement is performed even if the identifier to be replaced is included in another identifier.



2.4.2 Link Options

Figure 2-23 Link Options

×	Device Output Code	
	List	
	Variables/functions information	
,	Section	
Г	Layout sections automatically	Yes(-AUTO SECTION LAYOUT)
	Automatically allocate sections per module	No
Γ	Section start address	FAACODE, FAADATA/2000
-	Section that outputs external defined symbols to the	Section that outputs external defined symbols to the file[0]
1	ROM to RAM mapped section	ROM to RAM mapped section[4]
7	[0]	.data=.dataR
	[1]	.sdata=.sdataR
	[2]	FAACODE=FAACODER
	[3]	FAADATA=FAADATAR
Γ	Allocate FAA memory area automatically	Yes
,	Verify	
ŀ	Message	
	Others	

Table 2-7 Link Options, Overview of settings (1/2)

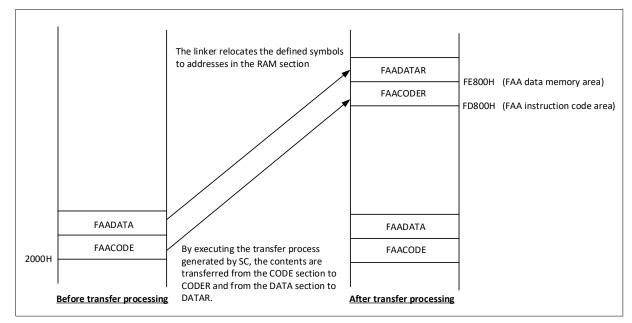
Category	Item	Description
Section	Layout sections automatically	Set "Yes(-AUTO_SECTION_LAYOUT)".
		Sections are automatically allocated based on information in the device file.
		When selecting "No", the address of each section used in the program need to be specified in "Section Start Address".
	Section start address	Set "FAACODE,FAADATA/address".
		Specify the address of code flash memory to store FAA programs and data. In the FAA program file (library or template) generated by the Smart Configurator (SC), the code section name is defined in FAACODE and the data section name is defined in FAADATA. Therefore, specify "FAACODE" and "FAADATA" as the section name. In addition, SC provides the processing (in Config_FAA_Common.c, generated by SC) to transfer the FAA program and data to the instruction code memory and data memory. The processing is performed in units of 2 bytes. Therefore, FAACODE and FAADATA must be aligned to the 2-byte boundary. specify an even number address after D8H. (at address 2000H in the example).



Category	Item	Description
Section	ROM to RAM mapped section	Set "FAACODE=FAACODER,FAADATA=FAADATAR".
		 The definition symbols for the FAA program and data placed in the code flash memory will be relocated to the internal RAM (instruction code memory and data memory). If relocation is not performed, the addresses of the FAA program and data symbols will remain in the code flash memory area, and symbol information cannot be handled correctly during debugging. The left side specifies the FAA program and data sections located in code flash memory. The right side specifies the section of RAM to be transferred. In the processing to transfer the FAA program and data to the instruction code memory and data memory (in Config_FAA_Common.c generated by SC), FAACODER and FAADATAR is handled as the transfer destination RAM section, so the right side specifies FAACODER and FAADATAR.
	Allocate FAA memory area automatically	Set "Yes".
		Reserve a dedicated area for FAA in the internal RAM. Variables for the CPU program will not be placed in the FAA instruction code memory (FD800H-FE7FFH) or data memory (FE800H-FEFFFH) in the internal RAM.

Table 2-8 Link Options, Overview of settings (2/2)

Figure 2-24 Memory image before and after transfer processing





2.4.3 Program Building

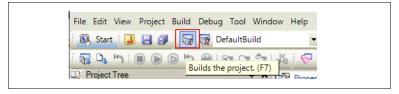
After setting the build tool options necessary to build the FAA program, build it. There are several ways to run a build. Two methods are described here.

- Select the [Build] menu -> [Build Project] (Figure 2-25)
- Click the [Builds the project] on the toolbar (Figure 2-26)

Figure 2-25 [Build] menu

File Edit View Project	Build Debug Tool Window Help	
🔅 🙉 Start 🛛 🚚 😭	Build Project	
	Rebuild Project	
Project Tree	🛃 Clean Project	
Sma 2 3 2 3	🕅 Rapid Build	
	Mundata Dependencies	

Figure 2-26 Build tool bar





2.5 Debug Tool Option Setting

Before downloading an executable object to the RL78/G24 Fast Prototyping Board, set the debug tool options required to debug an FAA program. Some options are set by the Smart Configurator (SC) in 2.3.1 Adding FAA Component. Manually set the options for which "No" is indicated in the "Set by SC" column in Table **2-9**. For debug tool options that are not described in this guide, set them if necessary. After setting the required options, download the object.

How to open the debug tool property:

Select the debug tool node in the project tree, and then select the [View] menu -> [Property] or select the [Property] from the context menu.

Table 2-9 shows the build tool options required to build the FAA program.

Tab	Category	Item	Description	Set by SC
Connect Settings	FAA	Debug FAA	Yes	Yes
Debug Tool Settings	Memory	FAA memory space (n) (n= 1 - 4)	Instruction code space or Data space	No
	Break	Stop FAA when stopping	No or Yes	No
Download File Settings	Download	Specify code section name defined in FAA source file	FAACODER	Yes
		Specify data section name defined in FAA source file	FAADATAR	Yes

Table 2-9 Setting options of debug tool

2.5.1 Connect Settings

Figure 2-27 Connect Settings

Internal ROM	1/RAM			
Clock				
Connection				
	with Target Board			
FAA			-	
Debug FAA		Yes		
Flash				
lash				

Table 2-10 Connect Settings, Overview of settings

Category	Item	Description
FAA	Debug FAA	Set "Yes".
		Enable source debugging of the FAA program.



2.5.2 Debug Tool Settings

Figure 2-28 Debug Tool Settings

¥	Memory		
>	Memory mappings	[9]	
	Verify on writing to memory	Yes	
	FAA memory space (1)	Data space	
	FAA memory space (2)	Instruction code space	
	FAA memory space (3)	Data space	
	FAA memory space (4)	Data space	
>	Access Memory While Running		
~	Break		
	First using type of breakpoint	Software break	
	Stop emulation of timer group when stopping	No	
	Stop emulation of serial group when stopping	No	
	Stop FAA when stopping	No	
>	Trace		1
>	Mask for Input Signal		
>	Step function		

Category	Item	Description
Memory	FAA memory area(n) (n=1 - 4)	Set the FAA space corresponding to FAA memory space (n).
		The debugger can display up to four [Watch] panels and four [Memory] panels each.
		When debugging the FAA program, the space set here is displayed in each panel.
Break	Stop FAA when stopping	Set "No" or "Yes".
		If the debug target is a CPU, select whether to stop the FAA program when the CPU program is stopped by the stop button.



2.5.3 Download File Settings

Figure 2-29 Download File Settings

¥	Download	
>	Download files	[1]
	CPU Reset after download	Yes
	Download Mode	Speed priority
	Erase flash ROM before download	No
	Automatic change method of event setting positio	Suspend event
	Check reserved area overwriting	Yes
	Specify code section name defined in FAA source	FAACODER
	Specify data section name defined in FAA source	FAADATAR
>	Debug Information	

Table 2-12 Download File Settings, Overview of settings

Category	Item	Description
Download	Specify code section name defined in FAA source	Set "FAACODER".
		In the FAA program file (library or template) generated by the Smart Configurator (SC), the code section name is defined in FAACODE.
		However, specify the section name FAACODER to be relocated to the RAM area.
		Reference: The link option "Section mapped from ROM to RAM".
	Specify data section name defined in FAA source	Set "FAADATAR".
		In the FAA program file (library or template) generated by the Smart Configurator (SC), the data section name is defined in FAADATA.
		However, specify the section name FAADATAR to be relocated to the RAM area.
		Reference: The link option "Section mapped from ROM to RAM".



RL78/G24

2.5.4 Program Download

After setting the debug tool options necessary to debug the FAA program, connect PC and RL78/G24 Fast Prototyping Board and then download the object. There are several ways to download. Two methods are described here.

- Select the [Debug] menu -> [Download] (Figure 2-30)
- Click the [Download] on the toolbar (Figure 2-31)

Caution1: Before downloading, check the power supply in the [Connect Settings] tab – [Connection with Target Board] of the debug tool option.

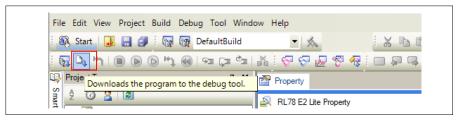
Caution2: The FAA program is not placed in the instruction code memory by simply downloading the object. You need to transfer the FAA program and data from the code flash memory to the instruction code memory and data memory by using the CPU program.

The RL78 Smart Configurator provides transfer processing functions as FAA components. The transfer processing function is executed in the initialization routine before the main function is executed, and the transfer is performed.

Figure 2-30 [Debug] menu

File Edit View Project Build	Debug Tool Window Help	
🕄 🕄 Start 🛛 🚚 🗐	Debug Solutions	• * *
	🗅 Download	
Project Tree	🙀 Build & Download	F6
S A (2) O I (2)	📆 Rebuild & Download	

Figure 2-31 Debug tool bar





2.6 FAA Program Debug

2.6.1 Debug Target

When debugging the RL78/G24 program, select whether to debug the CPU or FAA. Select by using one of the following methods.

- Select the [View] menu -> [Debug Manager] to open [Debug Manager]. Select the debug target on it. (Figure 2-32)
- Select the target debug on the status bar. (Figure 2-33)

Figure 2-32 Open [Debug Manager]

Eile	Edit	Vie	W Project Build Debug Tool Window Hele	🚯 🗅 🐂 I 💷 🔊 🔊 🙌 🚳 I 🖘 Çi 📩 👬
	L	-		Debug target:
8	🐧 Star		Solution List	O CPU
Į,	, D,	6	Project Tree	Debug target status:
	Project		Property	
ŝ	9 0		Smart Browser	Running status: BREAK
Ĩ		g	Code Generator Preview	Target status: Standby&Disable
Smart Manual	.	2	Pin Configurator >	Current PC: 0x000
=)	-	φ,	Smart Manual	
			Output	
		-	Error List	P
	Ė	-	Memory Mapping Profiler	
			Functions and Variables Access Table	
		7	Debug Manager	
			Watch	

Figure 2-33 Status bar

F6 Build & Do. F7 Build Proje. F8 Ignore Bre. F9 Set/Delete. F8 Step Over F77 Step In F12 Jump to Fun.						
FAA	BREAK	Standby&Disable	ᅌ 0x000	🚥 RL78 E2 Lite	👸 Not me	asured
CPU FAA						

Address information is displayed in the address area only for the source file to be debugged, and debugging operations such as step execution are possible at the source level.

It is possible to change the debug target in the following status.

Table 2-13Change debug target

Current debug target	Status	Change from CPU to FAA	Change from FAA to CPU
CPU	CPU program stopping	Available	—
CPU	CPU program running	Not available	—
FAA	FAA program stopping	_	Available
FAA	FAA program running	—	Available



Additionally, CPU or FAA status that is debug target is displayed in the debug manager and status bar. When FAA is the debug target, the status of FAA is as follows. If multiple statuses exist at the same time, the statuses are displayed separated by "&".

Table 2-14 FAA status

Status display FAA status	
Standby	Stops supply of an input clock to FAA. (FAAEN bit = 0)
Disable	Disables FAA operation. (ENB bit = 0)
Sleep	FAA in Low power consumption mode (SLP bit = 1 and EXE bit = 0)



2.6.2 Source File Display

After selecting the FAA as the debug target, display the .dsp file containing the FAA program on the [Editor] panel. The address information appears in the address area, and debug operations such as step execution can be performed at the FAA source level.

The address area indicates the addresses in the FAA instruction code memory space. The address area is not displayed when the debug target is CPU.

Figure 2-34 Source file display

	-			Columns
ine 43 44 45 46 47		Addı		SECTION CODE
48 49 50 51				; [IN] N_Max3_Value1, N_Max3_Value2, N_Max3_Value3 ; [OUT] N_Max3_Result ; [NOTE] -
52 53 54		000	•	.PUBLIC _P_Max3 _P_Max3:
55 56 57		002		MOV #_V_Max3, DP1 MOV #_V_Max3, RP0
58 59 60 61		006 008 00a		MOV (#N_Max3_Value1, DP0), A0 MOV (#N_Max3_Value2, DP1), R0 JSR #P_Max2_Direct ; A0 = Max2(Value1, Value2)
62 63 64		00c 00e	L	MOV (#N_Max3_Value3, DP1), R0 JSR #P_Max2_Direct ; A0 = Max2(A0, Value3)
65 66 67		010 012		MOV A0. (#N_Max3_Result, RP0) STOP
68 69 70				Max2
71 72 73 74				: [IN] A0, R0 : [OUT] A0 : [NOTE] Keep DP0, DP1, RP0
75 76		013	L	P_Max2_Direct: SUB_R
Repla	ice Ne	ext FS	Go	F6 Build & Dow F7 Build Project F8 Ignore Break



2.6.3 Go/Stop

When selecting FAA as the debug target, FAA source debugging is enabled. There are several ways to go/stop FAA program. Two methods are described here.

- Select the [Debug] menu -> [Go] / [Stop]. (Figure 2-35)
- Click the [Go] / [Stop] on the toolbar. (Figure 2-36)
- Click the [Go] / [Stop] on the toolbar of the Debug Manager. (Figure 2-37)

Figure 2-35 [Debug] menu

File	Edit View Project Build	Deb	ug Tool Window Help		0	Debu	ig Tool Window Help		
6	🕅 Start 🛛 🚚 🗐 🛛 🐻		Debug Solutions	•			Debug Solutions		
6	3 D, M I O O M	D,	Download			32	Download		
-		5	Build & Download	F6	1	to l	Build & Download	F6	
~ J		5	Rebuild & Download		1	in.	Rebuild & Download		
Smart Manual			Connect to Debug Tool		6	-	Connect to Debug Tool		
Manu	R7F101GLGxFB (M	D,	Upload		0	2	Upload		
≞_	🖳 Smart Configurate	X	Disconnect from Debug Tool	Shift+F6		X	Disconnect from Debug Too	Shift+F6	
	CC-RL (Build Tool)	-					Using Debug Tool		
	RL78 E2 Lite (Debu		Using Debug Tool	•		~			
	Program Analyzer		Stop	Shift+F5	(Stop	Shift+F5	
	main.c		Go	F5	0		Go	F5	
	iodefine_faa.ir	-	Ignore Break and Go	F8		-	Ignore Break and Go	F8	

Figure 2-36 Debug tool bar

Γ

Г

Sample1 - RL78 E2 Lite - CS+ for CC - [main.c]	Sample1 - RL78 E2 Lite - CS+ for CC - [main.c]			
<u>File Edit View Project Build D</u> ebug <u>T</u> ool <u>W</u> indow <u>H</u>	<u>File Edit View Project Build Debug Tool Window H</u>			
🔯 Start 🚽 📄 🗿 🕅 ன ன DefaultBuild	🚳 Start 🔒 🔚 🗊 🗑 ன DefaultBuild			
💯 🗗 🚽 🛛 🕑 🕪 🚳 🗠 E ČE ÇE	🖉 🗗 🖬 💿 🔊 👘 🎯 🖙 🖓 🖉			

Figure 2-37 Debug Manager

Debug target:	d wj 🛞 25 či çi 🥻	Debug target:	• • • •	1 (1) FI (I CI CI CI (I)
O CPU	FAA	О СРИ		• FAA
Debug target status:		Debug target sta	atus:	
Running status:	BREAK	Running status	:	RUN
Target status:		Target status:		
Current PC:	0x000	Current PC:	¢)	Running



The FAA program control are as follows:

- ✓ If the FAA status is "Standby" or "Disable", program execution cannot start and other debug operations such as step execution are also disabled. When using FAA libraries, FAA programs runs by calling the start function (that executes FAAEN=1, ENB=1) provided by each FAA library.
- ✓ When the debug target is FAA, the operation to execute or stop programs only executes or stops the FAA program. The CPU program is not executed or stopped in synchronization. However, you can use a debug tool option so that stopping a CPU program also stops the FAA program when the debug target is CPU. To do this, on the [Debug Tool Settings] tab, under the [Break] category, select [Yes] for [Stop the FAA when stopping the program].
- ✓ Step execution is applicable only to the FAA.
- Reset operation performs a software reset for the FAA. The whole MCU (CPU and peripheral functions) are not reset. When the debug target is CPU, the whole MCU (CPU and peripheral functions) are reset.
- ✓ Do not proceed with debugging of the FAA during execution of a CPU program that includes operations with the WIND register. Since the debugger temporarily rewrites the WIND register in the debugging operations for the FAA, the use of FAA debugging may make operation of the program being executed by the CPU incorrect.

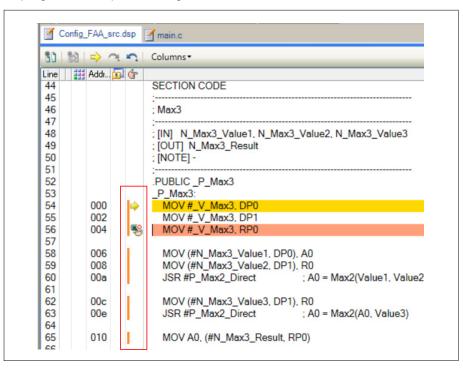
2.6.4 Breakpoint

After selecting the FAA as the debug target, display the FAA source on the [Editor] panel. You can set a breakpoint by clicking the main area on the row on which you want to set the breakpoint. To cancel a breakpoint, click the icon set for the breakpoint.

The breakpoint controls for the FAA program are as follows:

- ✓ 4 points hardware breaks are available. (Break after execution)
- ✓ If the FAA is stopped after detecting a hardware break, the CPU is not synchronously stopped.

Figure 2-38 FAA program, breakpoint setting





2.6.5 Memory

When selecting FAA as the debug target, FAA instruction code memory and data memory are displayed in the [Memory] panel.

The memory display control for the FAA are as follows:

- ✓ The [Memory] panel that specified in 2.5.2 Debug Tool Settings displays FAA instruction code memory and data memory.
- \checkmark Address is the FAA space.
- ✓ When the debug target is CPU, CPU memory is displayed regardless of the settings in 2.5.2 Debug Tool Settings.
- ✓ The display cannot be updated while the FAA program is running.
- ✓ If the FAA status is "Disable" or "Standby", the displayed content will be undefined.

RL78 E2 Lite Property	P -	🖉 🥮 Notation 🗸 Size Notation 🗸 Encoding 🗸 View 🗸	
Memory		Move when Stop	ve
Memory mappings	[16]	+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f ASCII	_
Verify on writing to memory	Yes	000 0A 00 00 01 4 00 00 00 1E 00 00 00 00 00 00 00 00 00	
FAA memory space (1)	Data space		
FAA memory space (2)	Instruction code space	020 00 00 00 00 00 00 00 00 00 00 00 00	
FAA memory space (3)	Data space	040 00 00 00 00 00 00 00 00 00 00 00 00	
FAA memory space (4)	Data space	050 00 00 00 00 00 00 00 00 00 00 00 00	
Access Memory While Runn	ning		
Break		Memory1 Data 🗿 Local Variables 🗱 CPU Register	
Trace			_
Mask for Input Signal		Memory2 Code	џ
 Step function 		🖉 🛞 Notation 🕶 Size Notation 💌 Encoding 👻 View 🕶	
Skip specified section	No		
lemory		Move when Stop Mo	ve
lonioly		+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f ASCII	
		000 80 00 84 00 88 00 41 00 44 01 A0 13 44 02 A0 13 ?. ?. ?. A. D. ?. D. ?.	
		010 47 03 20 4C D0 18 4B 3E 2F 35 3E 00 00 00 00 00 6. L?.K>/5>	
Connect Settings 🔪 Debug Too	S Download File K Hook Transacti		

Figure 2-39 [Memory] panel

2.6.6 Symbol (Label)

When selecting FAA as the debug target, the symbols (labels) defined in the FAA program are displayed in the [Watch] panel.

The watch display control for the FAA are as follows:

- ✓ Data for the FAA has 32-bit width. However, 8-bit width data is displayed when a symbol is registered on a [Watch] panel. Therefore, change the [Size Notation] setting to [4 Bytes].
- \checkmark Address is the FAA space address.
- ✓ If [Immediate address] is specified for the watch-expression, the value corresponding to the address in the space specified in 2.5.2 Debug Tool Settings is displayed.
- ✓ If the debug target is CPU, the value column indicates a question mark (?).
- \checkmark If the FAA status is "Standby" or "Disable", the display contents are undefined.

Remark. To make a symbol accessible to the CPU program, it must be defined with a name starting with "_" and must be declared public in the FAA program.



Figure 2-40 [Watch] panel (Size change of symbol)

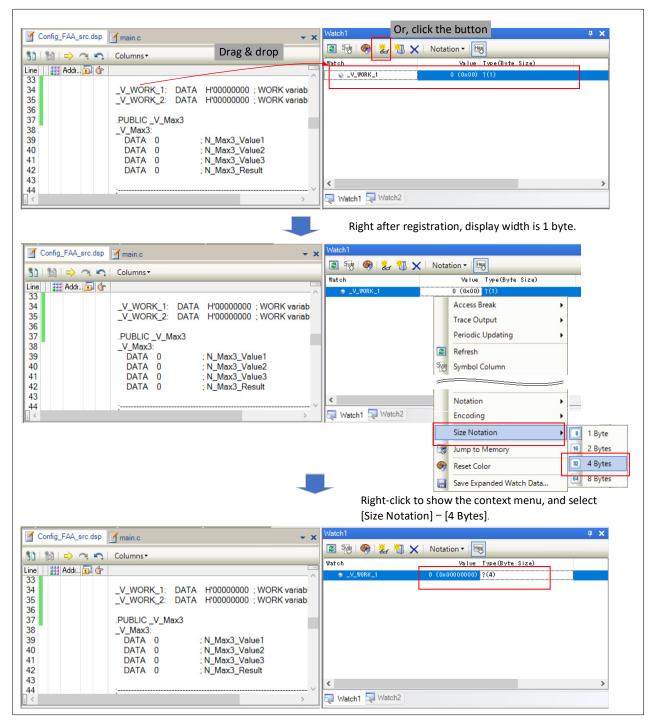




Figure 2-41 [Watch] panel (How to specify with [Immediate address])

Config_FAA_src.dsp	⊻ main.c		→ X	Watch1	Or, click the butt		ų ×	
1 13 ⇒ ~ • •	Columns -	Drag & drop		Vatch	Volue Tw	e(Byte Size)		
🟭 Addı 🗾 👉			~	€ _V_Max3	10 (0x0a) ?(1			
	_V_WORK_1: DATA	H'00000000 ; WORK	K variab					
	_V_WORK_2: DATA	H'00000000 ; WOR	K variab					
	.PUBLIC_V_Max3 _V_Max3:							
	DATA 0	N_Max3_Value1						
	DATA 0 :	N_Max3_Value2 N_Max3_Value3						
	DATA 0 :	N_Max3_Result				_		
	;		~ >	Watch1 🔍 W	atch2		,	
Config_FAA_src.dsp				Watch1			ч х	
	-		→ X		🛃 🗙 🛛 Notation - 🕅			
→ ~	Columns			Watch	Value Ty	pe(Byte Size)	Address Memo	
			^	<pre>_V_Max3 _ [0x0c]</pre>	10 (0x0a) ?(20 (0x14) ?(0x008 0x00c	
	_V_WORK_1: DATA _V_WORK_2: DATA			[0x10]	30 (0x1e) ?(0 (0x00) ?(1)	0×010 0×014	
	.PUBLIC _V_Max3			[0x14]	0 (0,00)	17	08014	
	_V_Max3:	N Max3 Value1						
	DATA 0	N_Max3_Value2						
		: N_Max3_Value3 : N_Max3_Result						
			~	<			>	
			>	🔍 Watch1 🔍 V				
						and the second		
					button to add a new			
				the area	after "_V_Max3" w	ith an imme	diate addr	ess.
				the area Address	after "_V_Max3" w of "V_Max3" is 0>	ith an imme	diate addr	ess.
				the area Address by 4 byte	after "_V_Max3" w of "V_Max3" is 0>	ith an imme	diate addr the addre	ess.
			- x	the area Address by 4 byte ^{Watch1}	after "_V_Max3" w of "V_Max3" is 0› ›s.	ith an imme	diate addr	ess.
b) ⇒ ~ •.			- x	the area Address by 4 byte ^{Watch1}	after "_V_Max3" w of "V_Max3" is 0> es. &	ith an imme	diate addr the addre	ess.
	Columns •		^	the area Address by 4 byte Watch Watch	after "_V_Max3" w of "V_Max3" is 0> es.	ith an imme 2008, specify ^{(Pe(Byte Size)}	diate addre v the addre Address Memo 0x000	ess.
b) ⇒ ~ •.	Columns •	H'00000000 : WOR H'00000000 : WOR	^	the area Address by 4 byte Watch Watch •Max3 • [0x00] • (0x10)	after "_V_Max3" w of "V_Max3" is 0> >>. >>. >>. >>. >>. >>. >>. >>. >>. >	ith an imme (008, specify (008, specify (008, specify) (008, speci	diate addre the addre Address Memo 0x006 0x006 0x010	ess.
b) ⇒ ~ •.	Columns*	H'00000000 : WOR H'00000000 : WOR	^	the area Address by 4 byte Watch Watch (Vatch (0x00)	after "_V_Max3" w of "V_Max3" is 0> es.	re(Byte Size)	diate addre t the addre Address Memo 0x000 0x000 0x000 0x000	ess.
b) ⇒ ~ •.	Columns• _V_WORK_1: DATA _V_WORK_2: DATA .PUBLIC_V_Max3 _V_Max3:		^	the area Address by 4 byte Watch Watch •Max3 • [0x00] • (0x10)	after "_V_Max3" w of "V_Max3" is 0> >>. >>. >>. >>. >>. >>. >>. >>. >>. >	ith an imme (008, specify (008, specify (008, specify) (008, speci	diate addre t the addre Address Memo 0x000 0x000 0x000 0x000	ess.
b) ⇒ ~ •.	Columns• _V_WORK_1: DATA _V_WORK_2: DATA .PUBLIC_V_Max3 _V_Max3: DATA 0 DATA 0	; N_Max3_Value1 ; N_Max3_Value2	^	the area Address by 4 byte Watch Watch •Max3 • [0x00] • (0x10)	after "_V_Max3" w of "V_Max3" is 0> >>. >>. >>. >>. >>. >>. >>. >>. >>. >	re(Byte Size)	diate addre t the addre Address Memo 0x000 0x000 0x000 0x000	ess.
1 1 ⇒ ~ ∽	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	; N_Max3_Value1	^	the area Address by 4 byte Watch Watch •Max3 • [0x00] • (0x10)	after "_V_Max3" w of "V_Max3" is 0> >>. >>. >>. >>. >>. >>. >>. >>. >>. >	ith an imme (008, specify (0) (1) (1) (1) (1) (1) (1) (1) (1	diate addre t the addre Address Memo 0x000 0x000 0x000 0x000	ess.
b) ⇒ ~ •.	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	^	the area Address by 4 byte Vatch •V_Max3 • [0x00] • [0x10] • [0x10]	after "_V_Max3" w of "_V_Max3" is 0> 25.	ith an imme (008, specify (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	diate addre t the addre difference Address Memo 0x008 0x008 0x000 0x010 c t	ess.
b) ⇒ ~ •.	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	^	the area Address by 4 byte Watch Watch •Max3 • [0x00] • (0x10)	after "_V_Max3" w of "_V_Max3" is 0> 25.	ith an imme (008, specify (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	diate addre t the addres Address Memo 0x008 0x0	ess. ess shifte
b) ⇒ ~ •.	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab	the area Address by 4 byte Vatch •V_Max3 • [0x00] • [0x10] • [0x10]	after "_V_Max3" w of "_V_Max3" is 0> 25.	ith an imme (008, specify (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	diate addre t the addres Address Memo 0x008 0x0	ess. ess shifte
1 1 ⇒ ~ ∽	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab	the area Address by 4 byte Vatch •V_Max3 • [0x00] • [0x10] • [0x10]	after "_V_Max3" w of "_V_Max3" is 0> 25.	ith an imme (008, specify (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c	diate addre t the addres Address Memo 0x008 0x0	ess. ess shifte
b) ⇒ ~ •.	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab	the area Address by 4 byte Vatch (0x10) (0x10) (0x10) (0x10) (0x10)	after "_V_Max3" w of "_V_Max3" is 0> 25.	ith an imme (008, specify (008, specify) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1	diate addre the addres Address Memo 0x000 0x00 0x000	ess. ess shifte
b) ⇒ ~ •.	Columns* _V_WORK_1: DATA _V_WORK_2: DATA _V_WORK_2: DATA _V_Max3: _DATA 0 DATA 0 DATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab	the area Address by 4 byte Vatch • [0x10] • [0x10] • [0x10] • [0x10] • [0x10]	after "_V_Max3" w of "V_Max3" is 0> es.	Access Break Trace Output	diate addre the addre diate addre diate addre diate addre diate addre diate addre Address Meno 0x000 0x00	ess. ess shifte
50 🗭 🤇 🧙	Columns• _V_WORK_1: DATA _V_WORK_2: DATA _VUWORK_2: DATA _V_Max3: _V_Max3: _DATA 0 DATA 0 DATA 0 DATA 0 DATA 0 	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab	the area Address by 4 byte Vatch • [0x10] • [0x10] • [0x10] • [0x10] • [0x10]	after "_V_Max3" w of "V_Max3" is 0 ss. Notation • Value T 10 (0x03) ?? 20 (0x13) ?? 20 (0x13) ?? 0 (0x00) ?? vatch2 registered watch exp	Access Break Trace Output	diate addre the addre diate addre diate addre diate addre diate addre diate addre Address Meno 0x000 0x00	ess. ess shifte
bil → 주 ↑	Columns* WORK_1: DATAWORK_2: DATAWORK_2: DATAMAX3:MAX3:MAX3:ATA 0ATA	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab K variab	the area Address by 4 byte Vatch Vatch (0x00) (0x10) (0x14) Select all r context m	after "_V_Max3" w of "_V_Max3" is 0 ss. Notation • Value F 10 (000) ? 20 (0010) ? 20 (0010) ? 0 (0000) ? vatch2 Pegistered watch exp enu, and select [Size	ith an imme (008, specify (0) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1	diate addre the addre diate addre diate addre diate addre diate addre diate addre Address Meno 0x000 0x00	ess. ess shifte
5) 🗭 🤉 🧙	Columns• _V_WORK_1: DATA _V_WORK_2: DATA _VUWORK_2: DATA _V_Max3: _V_Max3: _DATA 0 DATA 0 DATA 0 DATA 0 DATA 0 	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3	K variab K variab	the area Address by 4 byte Vatch •Max3 • [0x00] • [0x10] • [0x10] • (0x10] • (after "_V_Max3" w of "_V_Max3" is 0 ss. Notation · Value T 10 (0x00) ? 20 (0x10) ? 20 (0x10) ? 0 (0x00) ?? watch2 value Ly value Ly value T value T valu	Access Break Trace Output Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation Encoding Size Notation	diate addre the addres Address Memo 0x000 0x00 0	ess. ess shifte
b) → へ へ Hitti Add	Columns* WORK_1: DATAWORK_2: DATAMORK_2: DATAMAx3:ATA 0ATA 0ATA 0ATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3 : N_Max3_Result	K variab K variab	the area Address by 4 byte Vatch (0x00) (0x10) (0x11) Select all r context m Vatch Select all r context m	after "_V_Max3" w of "_V_Max3" is 0 es. Value T 10 (0x03) ? 20 (0x14) ? 20 (0x14) ? 20 (0x14) ? 20 (0x10) ? 0 (0x00) ? value T value T	ith an imme (008, specify (0) (1) (1) (1) (1) (1) (1) (1) (1	diate addre the addre address Memo 0x008 0x000 0x00 0x000	ess. ess shifte
bil → Add □ () iiii Add □ () iiiii Add □ () iiii Add □ () iii Add □ () ii	Columns* WORK_1: DATAWORK_2: DATAWORK_2: DATAMax3:Max3:MAX3:MAX4DATA 0ATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3 : N_Max3_Result	K variab K variab	the area Address by 4 byte Vatch (0x00) (0x1	after "_V_Max3" w of "_V_Max3" is 0 es. Notation · (***) Value T 10 (0x00) ?? 0 (0x10) ?? 0 (0x00) ?? vatch2 vatch2 vatch2 vatch2 vatch2 · (***********************************	ith an imme c008, specify (008, specify (1) (1) (1) (1) (1) (1) (1) (1)	diate addre v the addre a × Address Memo 0x000 0x00 0x	ess. ess shifte
onfig_FAA_src.dsp	Columns* WORK_1: DATAWORK_2: DATAWORK_2: DATAMax3:Max3:Max3:DATA 0DATA 0ATA	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3 : N_Max3_Result 	K variab K variab	the area Address by 4 byte Vatch •Max3 • [0x00] • [0x10] • [0x10] • [0x14] Select all r context m Vatch •Max8 •Max8 •Max8 •Max8	after "_V_Max3" w of "_V_Max3" is 0> es. V_Max3" is 0> es. Value T 10 (0x000) ?? 20 (0x10) ?? 20 (0x10) ?? 0 (0x00) ?? value T value T	ith an imme c008, specify (008, specify (1) (1) (1) (1) (1) (1) (1) (1)	diate addre the addre diate addre diate addre diate addre Address Memo 0x000 0x	ess. ess shifte
b) → へ へ Hitti Add	Columns* WORK_1: DATAWORK_2: DATAWORK_2: DATAMax3:MAX3:MAX3:MAX3MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:MAX3:	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3 : N_Max3_Result 	K variab K variab	the area Address by 4 byte Vatch •Max3 • [0x00] • [0x10] • [0x10] • [0x14] Select all r context m Vatch •Max8 •Max8 •Max8 •Max8	after "_V_Max3" w of "_V_Max3" is 0> es. V_Max3" is 0> es. Value T 10 (0x000) ?? 20 (0x10) ?? 20 (0x10) ?? 0 (0x00) ?? value T value T	ith an imme c008, specify (008, specify (1) (1) (1) (1) (1) (1) (1) (1)	diate addre the addre diate addre diate addre diate addre Address Memo 0x000 0x	ess. ess shifte
b) → へ へ Hitti Add	Columns* V_WORK_1: DATA V_WORK_2: DATA PUBLIC_V_Max3 V_Max3: DATA 0 DATA 0 DATA 0 DATA 0 DATA 0 :	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3 : N_Max3_Result 	K variab K variab	the area Address by 4 byte Vatch •Max3 • [0x00] • [0x10] • [0x10] • [0x14] Select all r context m Vatch •Max8 •Max8 •Max8 •Max8	after "_V_Max3" w of "_V_Max3" is 0> es. V_Max3" is 0> es. Value T 10 (0x000) ?? 20 (0x10) ?? 20 (0x10) ?? 0 (0x00) ?? value T value T	ith an imme c008, specify (008, specify (1) (1) (1) (1) (1) (1) (1) (1)	diate addre the addre diate addre diate addre diate addre Address Memo 0x000 0x	ess. ess shifte
b) → へ へ Hitti Add	Columns* WORK_1: DATAWORK_2: DATAWORK_2: DATAMAX3:MAX3:MAX3:MAX3:MAX3:MATA 0	: N_Max3_Value1 : N_Max3_Value2 : N_Max3_Value3 : N_Max3_Result H'000000000 : WORH H'000000000 : WORH H'000000000 : WORH	K variab K variab	the area Address by 4 byte Vatch •Max3 • [0x00] • [0x10] • [0x10] • [0x14] Select all r context m Vatch •Max8 •Max8 •Max8 •Max8	after "_V_Max3" w of "_V_Max3" is 0> es. V_Max3" is 0> es. Value T 10 (0x000) ?? 20 (0x10) ?? 20 (0x10) ?? 0 (0x00) ?? value T value T	ith an imme c008, specify (008, specify (1) (1) (1) (1) (1) (1) (1) (1)	diate addre the addre diate addre diate addre diate addre Address Memo 0x000 0x	ess. ess shifte

Remark. The above example shows that FAA memory space (1) is specified for the data space in 2.5.2 Debug Tool Settings.



2.6.7 Register

When selecting FAA as the debug target, the operation parameter register set, address pointer set, the processor control register, etc. are displayed in the [CPU Register] panel.

Figure 2-42	[CPU Register] panel
5	L - J JI

Notation - Heg		
Register Name	Value	
⊟ <mark>⊣</mark> Operation parameter register set		
E A0	0×00000000	
E MO	0×00000000	
📰 M1	0x00000000	
📃 L0	0×00000000	
🛅 L 1	0×00000000	
📰 R0	0×00000000	
🖃 🧺 Address pointer set		
E DPO	0×000	
E DP1	0×000	
E RPO	0×000	
📃 P 60	0×000	
E SPO	0×800	
🖃 🧺 Control Registers		
E E AACNT	0×0000	

2.6.8 SFR

When selecting FAA as the debug target, the [SFR] panel displays only SFRs (Special Function Register) that FAA can access. There are two types of SFRs that the FAA can access.

• SFRs of the FAA

Registers that are not affected by the address bus select register (ADBSEL) settings and can be accessed via the FAA bus.

• Registers of the peripheral functions

Registers that can be accessed via the FAA bus when "access from the FAA" is selected in the ADBSEL register.

There are two different types of register access to the peripheral functions as described below.

- \cdot Access to a peripheral function register through the FAA address map
- · Access to a peripheral function register by using the FAA address pointer (FAAAP)

For the address bus select register (ADBSEL) and how to access, refer to RL78/G24 User's Manual: Hardware (R01UH0961).

The SFR display control for the FAA are as follows:

- \checkmark The address area for the FAA SFR displays the FAA addresses.
- ✓ Access to some peripheral function SFRs is enabled by using the address bus function to permit bus access from the FAA. For such SFRs, the display name is suffixed by "_PTR". The address displayed in the address field is the FAA address pointer values that be set in the FAA address pointer (FAAAP) when accessing using the FAAAP register.
- ✓ The debugger reads or writes peripheral function SFR values through bus access from the CPU. Therefore, it cannot access the peripheral function SFRs for which bus access from the FAA is selected by using the address bus selection function, and the displayed values for these SFRs are undefined. To display the values of the peripheral function SFRs for which bus access from the FAA is selected, see 3.5 Sample Script Specification.



Figure 2-43 [SFR] panel

	SFR		Ф	×
	🔳 🧶 🖏 🖉 🗙	Notation - 🖷		
	(Input all or part of the targ	et SFR/Category name for search.)	✓	>
	SFR	Value Type(Byte Size)	Address	^
	PORT			
Registers of the peripheral function	S P1_PTR	0×00 SFR[R/W 8](1)	0x1f01	FAA address pointer
0	PM1_PTR	0xff SFR[R/W 8](1)	0x1f21	
	PU1_PTR	0×00 SFR[R/W 8](1)	0x031	
	PIM1_PTR	0x00 SFR[R/W 8](1)	0x041	
	POM1_PTR	0x00 SFR[R/W 8](1)	0x051	
	PMCA 1_PTR	0xff SFR[R/W 8](1)	0x061	
	🕀 🗂 TMB			
	🕀 🗂 ADC			
	🕀 🗂 DAC			
	I CTHER			
	FAA SFR			
Registers of the FAA	- I V0	0x00000000 SFR[R/W 32](4)	0x012	
Registers of the LAA		0x00000000 SFR[R/W 32](4)	0x013	FAA address
	1 1 1/2	0x00000000 SFR[R/W 32](4)	0×014	
		0.00000000.055 [5/# 001/#)	0.045	~
	🗊 SFR 🗊 CPU Register	Northell Northell		



3. Sample Project

This section describes how to display the SFR values of peripheral functions in the CS+'s [SFR] panel when debugging a FAA program using sample code and sample scripts.

3.1 Specifications

3.1.1 Specification Overview

This sample code uses a 16-bit timer KB30 (TKB30) to perform two PWM outputs.

PWM output is connected to LED1 and LED2. Initialize TKB30 using the CPU program, count the number of TKB30 timer interrupts (INTTKB00), create a fixed cycle (500ms) timing, and start FAA operation at a fixed cycle.

The FAA program controls the LED brightness by changing the duty ratio of the PWM output. After changing the duty ratio, the operation stops.

Table 3-1	Peripheral	Functions	and	Their	Usage
-----------	------------	-----------	-----	-------	-------

Peripheral	Usage
16-bit timer KB30 (TKB30)	Output PWM from TKBO00 pin and TKBO01 pin
Flexible application accelerator (FAA)	Change the duty ratio of PWM output from TKBO00 pin and KBO01 pin

Figure 3-1 Operation overview of PWM output

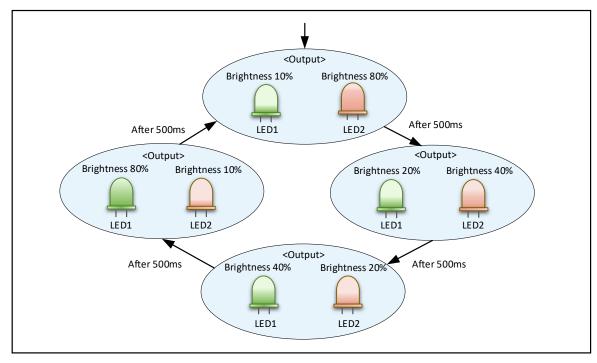


Table 3-2 Relationship between PWM output duty ratio and LED brightness

Duty ratio	Brightness
10%	10%
20%	20%
40%	40%
80%	80%



3.1.2 Operation Overview

In this sample code, 16-bit timer KB30 (TKB30) is used with the standalone mode (period controlled by the TKBCRn0 register), PWM signals are output from P12/TKB000 and P13/TKB001.

The PWM pulse period of TKB30 is 2ms, and the interrupts (INTTKB30) that occur in each period are counted 250 times. Start the FAA from the CPU every 500ms and change the duty ratio of PWM output with FAA.

- 1. [CPU program] Store the initial values of the TKBCR01 register and the TKBCR03 register in variables for checking the duty value.
- 2. [CPU program] Enable the TKB30 operation.
- 3. [CPU program] Set SFR access of the TKB30 to FAA bus.
- 4. [CPU program] Wait until the TKB30 interrupt occurs 250 times (500ms).
- 5. [CPU program] After the TKB30 starts the operation, the TKB30 interrupt occurs every 2ms.
- 6. [CPU program] Count the number of interrupt occurrences in the TKB30 interrupt (INTTKB30).
- 7. [CPU program] When TKB30 interrupt (INTTKB30) occurs 250 times (500ms), clock supply to the FAA is enabled and FAA operation is enabled.
- 8. [CPU program] Set the FAA stack pointer and the start address of the FAA program and start FAA operation. Then wait until the FAA program completes.
- 9. [FAA program] Update the compare register (TKBCR01) and change the duty ratio of TKBO00 output. And update the compare register (TKBCR03) and change the duty ratio of TKBO01 output. Every 500ms, the duty ratio of the TKBO00 output is updated by double in the order of 10% → 20% → 40% → 80%, and after the duty ratio reaches 80%, it is set to 10% again. The duty ratio of the TKBO01 output is updated by 1/2 in the order of 80% → 40% → 20% → 10%, and after the duty ratio is 10%, it is set to 80% again.
- 10. [FAA program] Store the updated duty ratio (values of the TKBCR01 register and the TKBCR03 register) in global variables and the FAA stops operating.
- 11. [CPU program] When FAA program execution is completed, clock supply to the FAA is stopped and FAA operation is disabled.
- 12. [CPU program] Store the updated duty ratio (values of the TKBCR01 register and the TKBCR03 register) in variables for duty value confirmation.
- 13. [CPU program] Return to step 4 and wait for TKB30 interrupts (INTTKB30) to occur 250 times (500ms) again.



3.2 Operation Confirmation Conditions

Table 3-3	Operation Confirmation Conditions
-----------	-----------------------------------

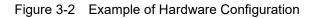
Item	Description
MCU	RL78/G24 (R7F101GLG)
Operating frequency	High-Speed On-Chip Oscillator Clock: 32MHz
	CPU/Peripheral Hardware Clock: 32MHz
Operating voltage	 3.3V (Can operate between 2.7V to 5.5V)
	 LVD0 Operation (VLVD0): Reset Mode
	Rising edge = 2.97V
	Falling edge = 2.91V
Integrated development	CS+ for CC V8.10.00 Manufactured by Renesas Electronics
environment (CS+)	
C compiler (CS+)	CC-RL V1.12.01 Manufactured by Renesas Electronics
Smart Configurator (SC)	Manufactured by Renesas Electronics V1.7.0
Board Support Package (BSP)	Manufactured by Renesas Electronics V1.60
Emulator	E2 Emulator Lite
Board	RL78/G24 Fast Prototyping Board (RTK7RLG240C00000BJ)

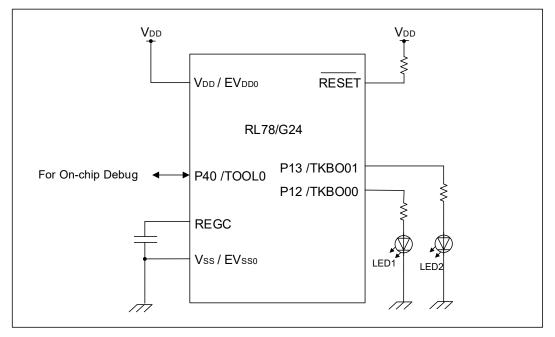


3.3 Hardware Description

3.3.1 Example of Hardware Configuration

The example of the hardware configuration used in this sample code is shown below.





- Note 1. This simplified circuit diagram was created to show an overview of connections only. When actually designing your circuit, make sure the design includes appropriate pin handling and meets electrical characteristic requirements (connect each input-only port to VDD or VSS through a resistor).
- Note 2. Connect any pins whose name begins with EVSS to VSS, and any pins whose name begins with EVDD to VDD, respectively.
- Note 3. VDD must not be lower than the reset release voltage (VLVD0) that is specified for the LVD0.

3.3.2 List of Used Pins

Table 3-1 shows the pins used and their function.

Table 3-4 Pins Used and their Functions

Pin name	I/O	Function
P12 / TKBO00	Output	PWM output (lighting control for LED1)
P13 / TKBO01	Output	PWM output (lighting control for LED2)

Caution. In this application note, only the used pins are processed. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.



3.4 Software Description

3.4.1 Smart Configurator Setting

The Smart Configurator (SC) settings in this sample code are shown below. The items and settings in each SC settings table are explained using the description on the settings screen.

3.4.1.1 Clock

The clock settings used in this sample code are shown below.

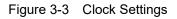
Operation mode: High-speed main mode 2.7(V)~5.5(V)

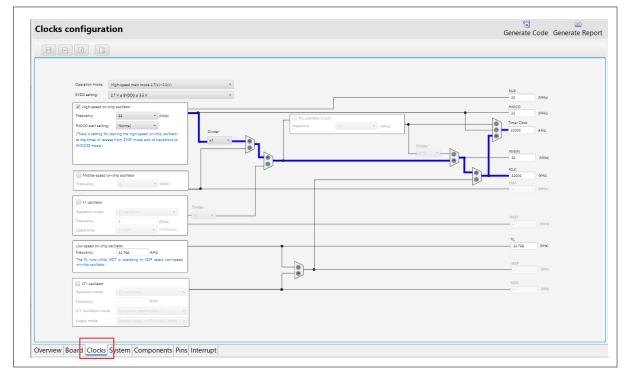
EVDD setting: 2.7 V \leq EVDD0 \leq 5.5V

High-speed on-chip oscillator: 32MHz

fCLK: 32000kHz

Timer Clock: 32000kHz







3.4.1.2 System

The system settings used in this sample code are shown below.

Figure 3-4 System Settings

▼ On-chip debug setting			
On-chip debug operation set	ting		
○ Unused	Use emulator	COM Port	
Emulator setting			
○ E2	E2 Lite		
Pseudo-RRM/DMM function	setting		
⊖ Unused	Used		
Start/Stop function setting			
Unused	◯ Used		
Monitoring point function set	tting		
Unused	Used		
Trace function setting			
◯ Unused	Used		
Security ID setting			
Use security ID			
Security ID	0x000000000000000000000000000000000000		
Security ID authentication fai	lure setting		
O Do not erase flash memor	y data		
Erase flash memory data			

3.4.1.3 Component

The component settings used in this sample code are shown below.

Table 3-5Component settings (LVD0)

Item	Description
Component	Voltage Detector
Configuration name	Config_LVD0
Resource	LVD0

Figure 3-5 LVD0 Settings

Reset mode When setting LVD0 to reset mode, s	set the detection voltage of I)	VD1 bigher than the detection welt	tage of IVD0
When setting LVD0 to reset mode.	set the detection voltage of I	VD1 bigher then the detection volt	tage of IVD0
	set the detection voltage of E	vor higher than the detection vorta	lage of LVD0.
🔿 Interrupt mode			
If LVD0 is set to interrupt mode and	I the LVD0 detection voltage i	is greater than the LVD1 detection	voltage.
LVD0 becomes undefined after the	LVD1 setting following releas	se from the reset state.	
LVD0 becomes undefined after the INTLVI priority	LVD1 setting following releas Level 3 (low)	se from the reset state. \sim	
INTLVI priority		se from the reset state.	
		v (V)	



Table 3-6 Component settings (TKB30)

Item	Description
Component	PWM Output
Operation	Standalone mode (Period controlled by the TKBCRn0 register)
Configuration name	Config_TKB0
Resource	ТКВО

Figure 3-6 TKB30 Settings

Count source setting			
Operation clock	СК20	~	
Clock source	fKBKC	~	(Clock frequency: 32000 kHz, fCLK is selected as fKBK0
PWM output setting			1
PWM period	2 ms	~	(Actual value: 2)
Duty (TKBO00 output)	10 (%)		(Actual value: 10)
Duty (TKBO01 output)	80 (%)		(Actual value: 80)
Delay (TKBO01 output)	0 (%)		(Actual value: 0)
A/D conversion start timing signal output function setti	ng]
TKBTGCR0 value	0		
Output setting]
Enable TKBO00 output			
Default level	Low level ~		
Active level	High level \sim		
Enable TKBO01 output			
Default level	Low level \sim		
Active level	High level \sim		
PWM output smooth start function setting			
Enable TKBO00 smooth start function			
TKBO00 smooth start initial duty	10 (%)		(Actual value: 10)
TKBO00 smooth start step width	1 ~		
Enable TKBO01 smooth start function			
TKBO01 smooth start initial duty	10 (%)		(Actual value: 10)
TKBO01 smooth start step width	1 💛		
Interrupt setting			
Generate interrupt when TKBO00 forced stopping of	f the output is terminated		
Priority	Level 3 (low) \sim		
Generate interrupt when TKBO00 forced stopping of	f the output is activated		
Priority	Level 3 (low) \sim		
Generate interrupt when TKBO01 forced stopping of	f the output is terminated		
Priority	Level 3 (low) \sim		
Generate interrupt when TKBO01 forced stopping of	f the output is activated		
Priority	Level 3 (low) \sim		_
Enable 16-bit timer KB30 end count]



Table 3-7 Component settings (FAA)

Item	Description	
Component	Flexible Application Accelerator	
Configuration name	Config_FAA	

Figure 3-7 FAA Settings

et Crypto Library (AES			
✓ «\$ Custom Library ☑ Template	Property	Value	
ାଙ୍କ Digital Filter	Configuration		
elie FFT			
> •1# LED Control			
SHA Library			
	<		>
	Template file (.dsp) for FAA source is g Add user program in user code area. if	enerated. there is no code and data in the file, FAA	^
	assembler error will occur when buildin		~

Remark. If any FAA library is not displayed after the sample project is opened, refer to step 11 in 2.3.1 Adding FAA Component to download FAA libraries.



3.4.2 Folder Structure

Table 3-8 shows the structure of the source files/header files used in the sample project.

Table 3-8	Folder Structure
-----------	------------------

	Folder, File name	Description	Generated by SC
\sam	ole_project <dir></dir>	Sample project folder	
m	ain.c	Sample source file	
sa	mple_project.py	(File for loading sample script)	
sa	mple_script.py	(Sample script)	
\s	rc <dir></dir>	Program storage folder	
	\smc_gen <dir></dir>	Smart Configurator generated folder	
	\Config_FAA <dir></dir>	FAA program storage folder	
	Config_FAA_common.c	Common FAA module source file	
	Config_FAA_common.h	Common FAA module header file	
	Config_FAA_common.inc	Include file for FAA assembly source file	
	Config_FAA_src.dsp	FAA assembly source file	$\sqrt{Note 1}$
	\Config_TKB0 <dir></dir>	TKB30 program storage folder	
	Config_TKB0.c	TKB30 source file	\checkmark
	Config_TKB0.h	TKB30 header file	
	Config_TKB0_user.c	TKB30interrupt source file	$\sqrt{Note 2}$
	¥general <dir></dir>	Initialization and common program storage	\checkmark
		folder	
	¥r_bsp <dir></dir>	BSP program storage folder	
	¥r_config <dir></dir>	Configuration header storage folder	\checkmark

Note. "<DIR>" indicates a directory.

Note 1. This sample project uses the Custom Library of FAA library. Therefore, file content is only a template and no code right after the file is generated. Sample code has been added.

Note 2. Sample code has been added in the user code area of SC.

3.4.3 Option Byte Settings

Table 3-9 shows the option byte settings.

Table 3-9	Option Byte Settings
-----------	----------------------

Address	Setting value	Description
000C0H/040C0H	1110 1111B (EFH)	Watchdog Timer stopped operation
		(Count stops after reset release)
000C1H/040C1H	1111 1011B (FBH)	LVD0 reset mode.
		Detection voltage: Rising 2.97V / Falling 2.91V
000C2H/040C2H	1110 1000B (E8H)	lash operation mode: High-speed main mode.
		High-speed on-chip oscillator frequency: 32MHz
000C3H/040C3H	1000 0100B (84H)	On-chip debug operation enabled



3.4.4 List of Constants

Table 3-10 and Table 3-11 show constants used in the sample code.

Table 3-10	Constants (CPU program)	
------------	-------------------------	--

Constant name	Value	Description	Function that uses the constant
FAA_BUS_ACCESS	0200H	Enable to access TKB30 register from FAA. (ADBSEL setting value)	main

Table 3-11 Constans (FAA program)

Constant name	Value	Description	
_C_TKBO00_DUTY_INIT	1900H	Initial duty ratio for TKBO00 output (TKBCR01 setting value)	
_C_TKBO01_DUTY_INIT	C800H	Initial duty ratio for TKBO01 output (TKBCR03 setting value)	
_C_TKBTRG_TKBRDT_REQ	1H	H Batch overwrite request of TKB30 compare register	
		(TKBRDT0 setting value)	

3.4.5 List of Variables

Table 3-12 and Table 3-13 show variables used in the sample code.

Table 3-12	Variables	(CPU program)
------------	-----------	---------------

Туре	Variable name	Description	Function that uses the variable
uint32_t	g_work_tkbo00	Variable to check the current duty ratio for TKBO00 output (Value of TKBCR01)	main
uint32_t	g_work_tkbo01	Variable to check the current duty ratio for TKBO01 output (Value of TKBCR03)	main
uint8_t	g_tkb_interrupt_flag	500ms elapsed flag	r_Config_TKB0_end _count_interrupt

Table 3-13 Variables (FAA program)

Size	Variable name	Description
4 bytes	_V_TKBO00_DUTY	Storage the updated duty ratio for TKBO00 output (TKBCR01 setting
		value)
4 bytes	_V_TKBO01_DUTY	Storage the updated duty ratio for TKBO01 output (TKBCR03 setting
		value)



3.4.6 List of Functions

Table 3-14 and Table 3-15show functions and processing used in the sample code. However, functions generated by the Smart Configurator that have not been modified are excluded.

Table 3-14 Functions (CPU program)

Function name	Description	Source file	
main	main process	main.c	
r_Config_TKB0_end_count_interrupt	TKB30 interrupt processing	Config_TKB0_user.c	
	(Count the number of INTTKB00		
	occurrences)		

Table 3-15 Processing (FAA program)

Label name	Description	Source file
_P_TKB_PWM	Change the duty ratio of TKBO00 and TKBO01 output	Config_FAA_src.dsp

3.4.7 Function Specification

The function specifications of the sample code are shown below.

[FUnction name] m	[FUnction name] main()				
Outline	main process				
Header	r_smc_entry.h、Config_TKB0.h				
Declaration	void main(void)				
Description	Start operation of the Timer TKB30, and start operation of the FAA every 500ms.				
Argument	-				
Return value	-				

CPU program

[Function name] r_Config_TKB0_end_count_interrupt()				
Outline	Timer TKB30 interrupt processing			
Header	r_cg_macrodriver.h、r_cg_userdefine.h、Config_TKB0.h			
Declaration	static voidnear r_Config_TKB0_end_count_interrupt(void)			
Description	Count INTTMKB30 occurrences and set the 500ms elapsed flag every 250 interrupts (500ms elapsed).			
Argument	-			
Return value	-			

FAA program

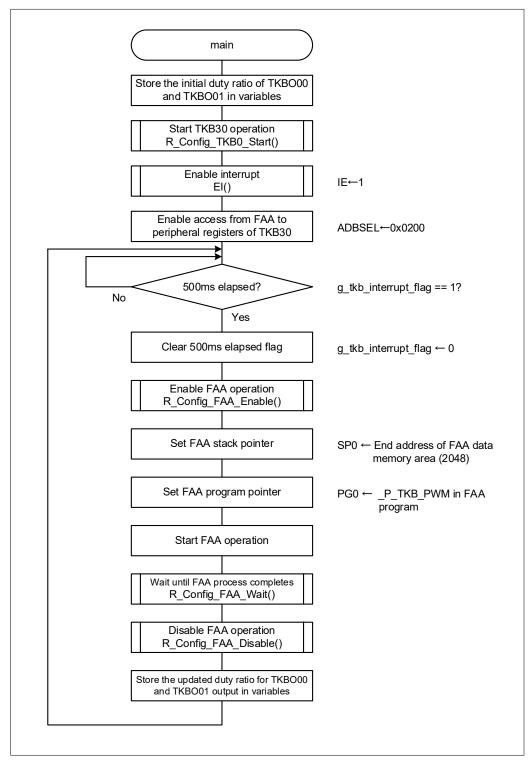
[Label name] _P_TKB_PWM				
Outline	Change processing of the duty ratio for TKBO00 and TKBO01 output			
Header	Config_FAA_common.inc			
Declaration	-			
Description	Change the duty ratio for TKBO00 and TKBO01 output.			
Argument	-			
Return value	-			



3.4.8 Flowchart 3.4.8.1 Main Process

Figure 3-8 shows the flowchart for the main process.

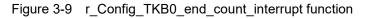
Figure 3-8 Main process

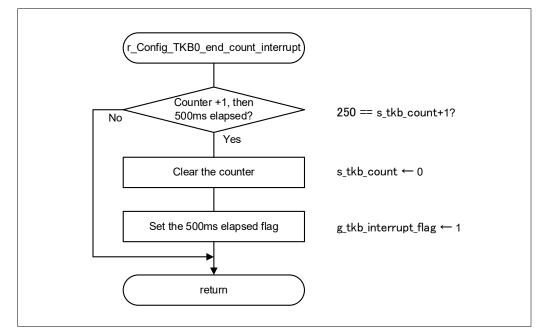




3.4.8.2 r_Config_TKB0_end_count_interrupt Function

Figure 3-9 shows the flowchart of the r_Config_TKB0_end_count_interrupt function.



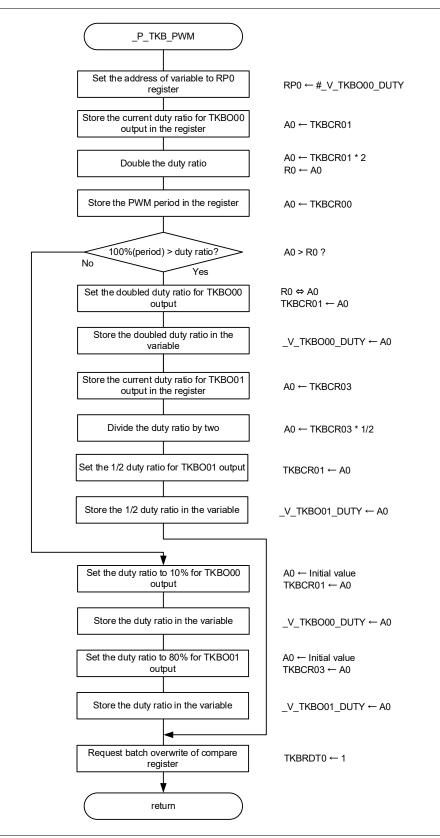




3.4.8.3 FAA Processing

Figure 3-10 shows the flowchart of the r_Config_TKB0_end_count_interrupt function.







3.5 Sample Script Specification

This sample project includes the sample script that manipulates the value of the address bus selection register (ADBSEL) to display peripheral function SFRs on the [SFR] panel in CS+ when debugging an FAA program. (sample_script.py in the sample project)

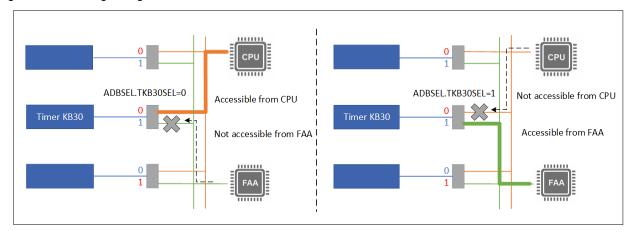
CS+ can be controlled by using a script language IronPython (Python that runs on .NET Framework) and the CS+ Python function. For details about the functions, see the help or documentation of CS+ for CC.

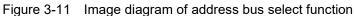
3.5.1 SFR Display Overview

For some peripheral functions of RL78/G24, access from the CPU or from the FAA can be selected with the address bus selection register (ADBSEL). For the address bus select register (ADBSEL), refer to RL78/G24 User's Manual: Hardware (R01UH0961).

The debugger reads or writes peripheral function SFR values through bus access from the CPU. It cannot access the peripheral function SFRs for which bus access from the FAA is selected with the address bus select register (ADBSEL). Therefore, reading from or writing to these peripheral function SFRs cannot be performed on the debugger's [SFR] panel.

To enable read and write on the debugger's [SFR] panel for the peripheral function SFRs for which bus access from the FAA is selected when the debug target is FAA, use the script to manipulate the ADBSEL register value.



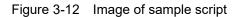


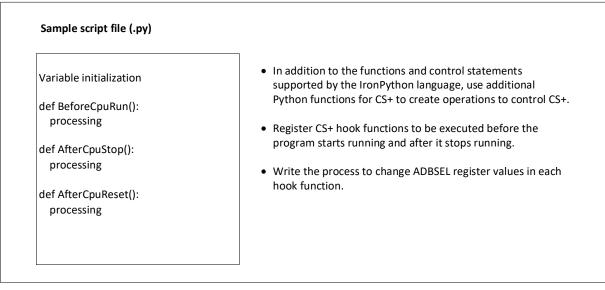


3.5.2 Operation Overview

When the debug target is FAA, after the FAA program is stopped by using the stop button, step execution, or breakpoint, the script assigns the XORed value to the current setting of the ADBSEL register. This temporality permits access from the CPU (the debugger) for the peripheral function SFRs for which access from the FAA is selected. In addition, before the FAA program is executed by using the execution button or step execution, the script assigns the original setting to the ADBSEL register to return the setting to permit access from the FAA.

This allows access from the FAA to the relevant SFRs during execution of the FAA program and, after the FAA program stops, allows the debugger to access the relevant SFRs and read or write values on the [SFR] panel.

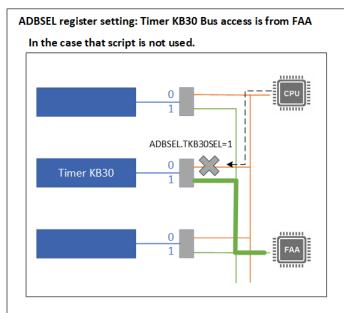




The script file for this sample project is sample_script.py.



Figure 3-13 Image diagram of changing ADBSEL register values by script



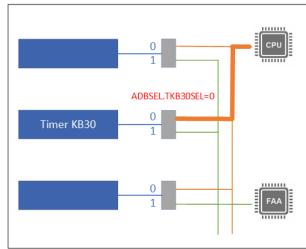
FAA program stopped:

The debugger cannot access Timer KB30' SFRs. (Because the debugger accesses SFRs via CPU bus.)

FAA program running:

The debugger can access Timer KB30' SFRs.





When the FAA program stops:

After the FAA program stops:

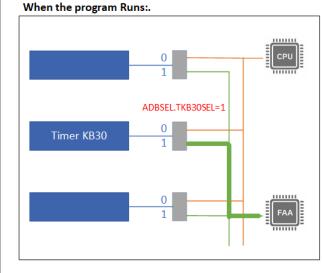
The script assigns the XORed value to the current setting of the ADBSEL Register. This changes the bus access from the FAA to the CPU. The debugger can access Timer KB30' SFRs.

(R/W to Timer KB30' SFRs is possible on the [SFR] panel.)

Before the FAA program runs:

The script assigns the original setting to the ADBSEL register to return the setting to permit access from the FAA.

The FAA program can access Timer KB30' SFRs.





3.5.3 List of Functions

(1) Hook Functions

The sample script uses the CS+ hook functions to change the ADBSEL register value within a hook function that is called when an event occurs. Table 3-16 lists the hook functions used in the script and provides an overview of processing.

Table 3-16	Hook functions used in the	sample script and	processing overview

Hook function name	Event	overview
AfterCpuReset	After CPU reset	Initialize variables used in the sample script.
BeforeCpuRun	Before execute	Write the original value that CPU sets to ADBSEL register to the ADBSEL register.
AfterCpuStop	After break	Write the XORed value of the original value to the ADBSEL register.

(2) CS+ Python Functions

Table 3-17 lists the CS+ Python functions used in the script and provides an overview of processing.

Table 3-17	CS + P	ython functior	e used in t	the sample	script and	nrocessing o	vorviow
	COTIC	yulon luncuor	is used in	uie sample	Script and	processing o	

Function name	Overview	
debugger.DebugTool.GetType	This function displays information about the debug tool.	
debugger.Watch.SetValue	This function sets a variable (SFR) value.	
debugger.Watch.GetValue	This function refers to a variable (SFR) value.	

3.5.4 List of Variables

(1) CS+ Python Property

Table 3-18 lists the CS+ Python property used in the script and provides an overview of processing.

Table 3-18	CS+ Python	propert	v used in the sam	ple script and	processing overview
		propert	y used in the sam	pic script and	processing overview

Property name	Overview
debugger.ProcessorElement	This property sets or refers to the PE of multiple cores with the name. [Value] 1: CPU 2: FAA

(2) Others

Table 3-19 lists the variables other than CS+ Python property used in the script and provides an overview of processing.

T I I A 4A	
1 able 3-19	Other variables used in the sample script and processing overview
	other valiables used in the sample script and processing overview

Variable name	Overview
FaaStatus	The FAA program operation status (Set when Go/Stop button is pressed).
	[Value]
	RUNNING: FAA program is running STOPPING: FAA program is stopping
previousPe	The debug target just before pressing the Go/Stop button.
	[Value] 1: CPU 2: FAA
adbsel_value_cpu	ADBSEL register's value set by the CPU program
number_of_command	The number of times the hook function was executed.

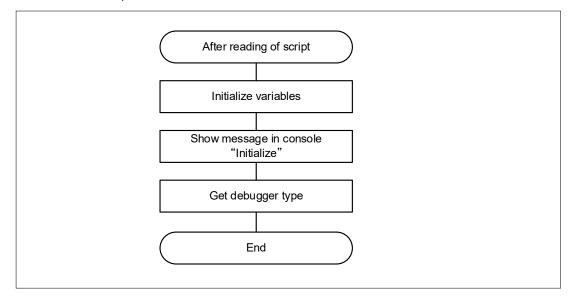


3.5.5 Flowchart

(1) Initialization Process

Figure 3-14 shows the flowchart of the initialization process that is executed after loading the sample script (.py).

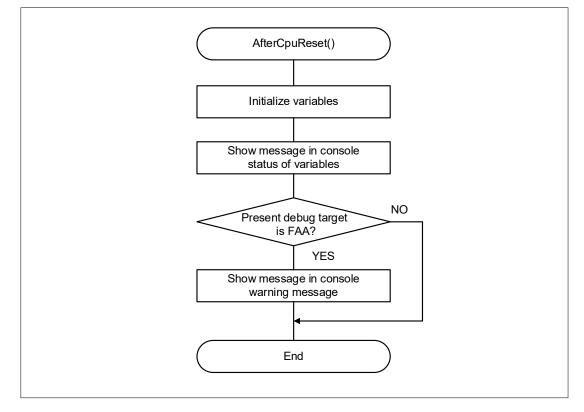
Figure 3-14 Initialization process



(2) AfterCpuReset Process

Figure 3-15 shows the flowchart of the AfterCpuReset process.



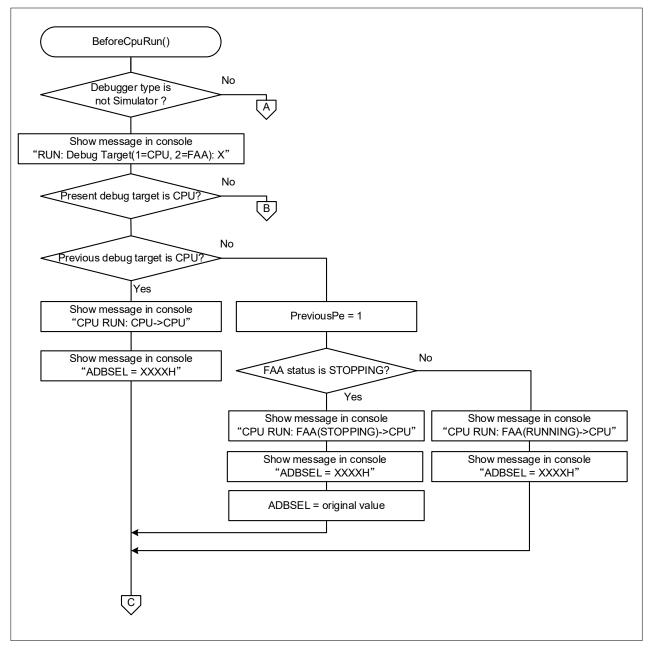




(3) BeforeCpuRun Process

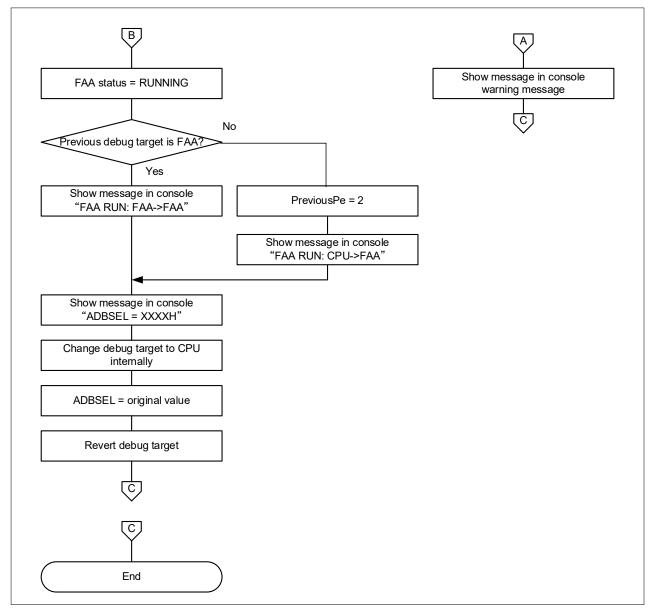
Figure 3-16 and Figure 3-17 show the flowchart of the BeforeCpuRun process.







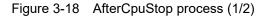






(4) AfterCpuStop Process

Figure 3-18 and Figure 3-19 show the flowchart of the AfterCpuStop process.



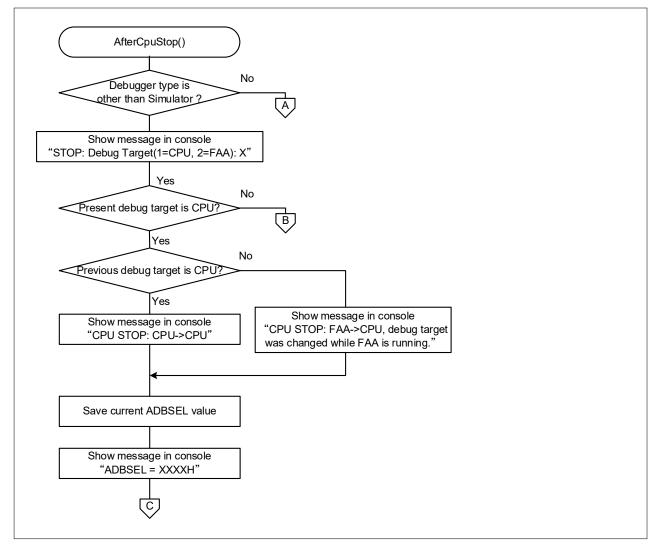
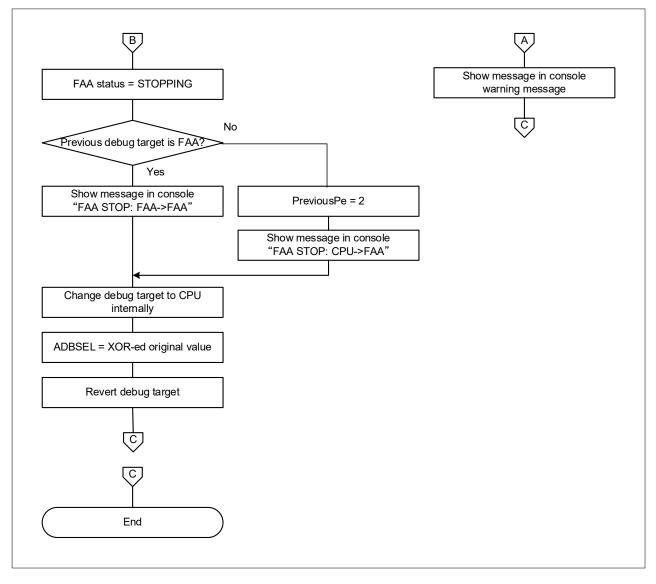




Figure 3-19 AfterCpuStop process (2/2)





3.5.6 Script Execution

There are several ways to execute a script and register hook functions.

• When loading the project file (*project-file-name.py*)

If there is a file in the same folder as the project file, and with the same name as the project file but with the "py" extension, then that file is executed automatically when the project file is loaded. The active project will be processed.

• When downloading the download file (*download-file-name*.py)

If there is a file in the same folder as the download file, and with the same name as the download file but with the "py" extension, then that file is executed automatically after downloading.

• Execute in the CS+ [Python Console] panel

Execute the ".py" file by the CS+ Python function: "Source".

In this sample project, it is executed when the project file is loaded.

The hook functions are declared in the sample_script.py. Also, there is the sample_project.py with the same name as the sample project "sample_project.mtpj", and the sample_project.py hooks the sample_script.py and registers hook functions declared in the sample_script.py. The sample_project.py is executed automatically when the project file is loaded.

Procedure:

- 1. Load the sample_project.mtpj to CS+.
- 2. Select the CS+ [View] menu -> [Python Console].
- 3. In the [Python Console] panel, confirm that the script executes.

Figure 3-20 Python Console

Python Console	д х
>>>(1) (output by script)	
Initialize Script	
E2Lite	
Console Sample Scripts	•



3.5.7 Basic debug operations

This section explains the basic operations of debugging a FAA program using sample code and sample scripts.

Procedure:

- 1. Connect the RL78/G24 Fast Prototyping Board (with the emulator or via COM port) to the PC.
- 2. Select the [Debug] menu -> [Rebuild & Download].
- 3. Select the [View] menu -> [Debug Manager] and select the CPU as debug target.

Figure 3-21 Debug Manager

(1) (1) (1)	🜔 🕞 🖏 🚳 93 ÇI ČI 🡬	
Debug target:		
CPU	○ FAA	
Debug target status	c.	
Running status:	BREAK	
Target status:		
Current PC:	♀ 0x00172	

4. Open the main.c. Click the main area of "FAACNT = 0x0001U;" to set the breakpoint (Software break).

Figure 3-22 main.c (Debug target: CPU)

10 I 161	 	. Co	lumns •
Line	🟭 Address р		
60		1	
61 62			* Function Name: main
63			void main(void)
64			
65			/* TKBCR01, TKBCR03 initial value */
66	00172	-	g_work_tkbo00 = TKBCR01;
67	0017c		g_work_tkbo01 = TKBCR03;
68			# TVD20 + + +/
69 70	00100		/* TKB30 start */
70	00186		R_Config_TKB0_Start();
72	0018a	1.1	EI();
73	00.00		2.0.
74			/* FAA bus select */
75	00190		ADBSEL = FAA_BUS_ACCESS;
76			
77			while (1)
78 79			
79 80	00193		/* Wait INTINTTMKB0 */ while (g_tkb_interrupt_flag == 0U)
81	00135		/ / //////////////////////////////////
82			
83			}
84			
85			/* Clear user flag */
86	00198		g_tkb_interrupt_flag = 0U;
87 88			
89	0019b		/* FAA operation enable */ R_Config_FAA_Enable();
90	00130		
91			/* Set stack pointer for FAA */
92	001a5		SP0 = FAA_ADDR_SP;
93			/* Set program pointer for FAA */
94	001aa		PG0 = FAA_ADDR_CODE(P_TKB_PWM);
95	0011.0	1.0	/* Start FAA program execution */
96 97	001b0	P	FAACNT = 0x0001U;
97			/* Wait until FAA stops */
99	001b1		R Config FAA Wait();
100	00101		/* FAA operation disable */
101	001b5		R_Config_FAA_Disable();
102			/* Store TKBCRO1 value changed by FAA */
103	001b9		g_work_tkbo00 = V_TKBO00_DUTY;
104	001c5		g_work_tkbo01 = V_TKBO01_DUTY;



5. Click the reset button and then click the execution button in the [Debug Manager].

Figure 3-23 Debug Manager

5 D, M I 🔘	D () H)	🕘 93 Ç3 °3 👬	
Debug target:		Go	
CF Rese	t	O FAA	
Debug target status			
Running status:		BREAK	
Target status:			
Current PC:	\Rightarrow	0x00172	

- After the program stopped by the breakpoint, change the debug target to the FAA on the [Debug Manager]. To debug FAA programs, the FAA must be enabled (FAAEN=1, ENB=1). In the sample code, "R_Config_FAA_Enable()" enables the FAA. Therefore, the FAA has been enabled at the breakpoint.
- 7. Register variables (_V_TKBO00_DUTY, _V_TKBO01_DUTY) and SFRs (TKBCR01_PTR, TKBCR03_PTR PPP, RRR) whose values are changed in the FAA program to the [Watch] panel.
 - After registering the variable, change it to 4-byte notation. (Refer to 2.6.6 Symbol (Label)
 - SFRs can also be displayed in the [SFR] panel.
 - The [Watch] panel to display the variables must be set to the FAA data space.(Refer to 2.5.2 Debug Tool Settings)

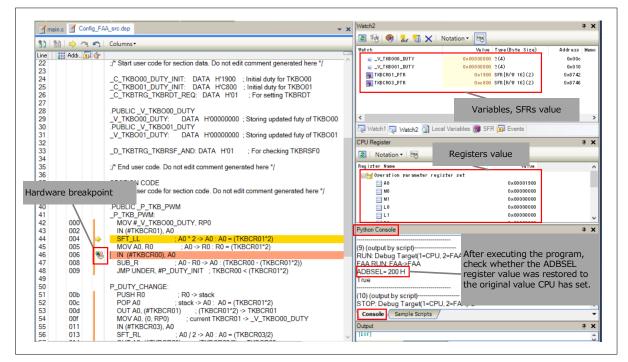
😰 🤫 । 🛠 🖏 🗙	Notation •			
Watch	Value	Type(Byte Size)	Address	Memo
_V_TKB000_DUTY	0x000000x0	?(4)	0x00c	
_V_TKB001_DUTY	0×00000000	?(4)	0×010	
TKBCR01_PTR		SFR[R/W 16](2)	0×8742	
TKBCR03_PTR	0x000x0	SFR[R/W 16](2)	0x8746	
		orn[n/# 10](2)	0.807.40	

Figure 3-24 [Watch] panel



- 8. Step-execute/execute the FAA program and debug while checking the values of variables, SFRs, and registers.
 - Breakpoints can be set by clicking in the main area of the FAA program source. (Refer to 2.6.4 Breakpoint)
 - After running the program, check in the [Python Console] panel whether the ADBSEL register value is the value set in the CPU program.
 (Remark: ADBSEL register is only accessible by the CPU, so the value of the ADBSEL register cannot be displayed in the [SFR] panel while debugging the FAA.)





3.5.8 Cautions When Using the Sample Script

✓ To disable this script (initialize Python), enter the following in the [Console] tab of the [Python console] panel.

common.PythonInitialize()

Alternatively, if you want to re-enable the sample script without reloading the sample project, enter the following in the [Console] tab of the [Python console] panel.

import os

Source (os.path.join(os.path.dirname(project.Path), 'sample_script.py'))

Remark. "os.path.join(os.path.dirname(project.Path)" is a description to get the full path of the file.

- ✓ The operation of sample code is not guaranteed. And the operation of this sample script is not guaranteed with all application programs and debugging operations.
- ✓ This sample script assists in displaying SFRs when debugging FAA programs. After completing debugging, thoroughly evaluate your system without using the sample script.



4. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

5. Reference Documents

RL78/G24 User's Manual: Hardware (R01UH0961) RL78 family User's Manual: Software (R01US0015) DSPASM FAA/GREEN_DSP Structured Assembler User's Manual (R20UT3911) RL78/G24 Fast Prototyping Board User's Manual (R20UT5091) RL78 Smart Configurator User's Gude: CS+ (R20AN0580) CS+ V8.10.00 User's Manual: RL78 Debug Tool (R20UT5301) (The latest version can be downloaded from the Renesas Electronics website.) Technical Update/Technical News (The latest version can be downloaded from the Renesas Electronics website.)

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



Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Nov. 14. 23	-	First edition



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

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

1. Precaution against Electrostatic Discharge (ESD)

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

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

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

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a systemevaluation 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.
- 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
- 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
- 5. 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.
- 6. 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.

- 7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
- 8. 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.
- 9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
- 10. 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.
- 11. 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.
- 12. 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.
- 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.
- (Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
- (Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

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

Contact information

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