

How to Execute a Program in RAM

CC-RL C Compiler for RL78 Family

Microcomputer Tool Product Marketing Department, Tool Business Division

Renesas System Design Co., Ltd.

Jul 24, 2015 Rev. 1.00

R20UT3512EJ0100

© 2015 Renesas System Design Co., Ltd. All rights reserved.

Introduction

- This document describes how to copy a program to RAM and execute it in RAM when using the CC-RL C compiler for the RL78 family.
- This document uses the following tools and versions for descriptions.
 - CC-RL C compiler for the RL78 family V.1.01.00
 - e² studio integrated development environment V.4.0.0.26
 - CS+ integrated development environment V.3.01.00



- How to Execute a Program in RAM
- Memory Map at Linkage
- Memory Maps for Program Execution in Microcontroller
- Adding the far Attribute to Functions in the C Source Code
- Adding Section Settings in the C Source Code
- Linker Settings
 - Section Setting for Mapping from ROM to RAM (-rom Option)
 - Section Allocation Settings (-start Option)
- Adding a Routine for Copying Functions to RAM in the C Source Code

3

Sample Program



How to Execute a Program in RAM

Settings in the C source code

- Add the ____far qualifier to functions to change their attribute to far.
- Use #pragma section to change the name of the section for variables.
- Adding linker settings
 - Specify the section for the functions to be executed in RAM as a section to be mapped from ROM to RAM (-rom option).
 - Specify allocation of the section for the functions to be executed in RAM (-start option).
- Adding a routine for copying functions to RAM
 - Before executing functions, add a routine for copying to RAM the section for the program to be executed in RAM.





Memory Map at Linkage



5



Memory Maps for Program Execution in Microcontroller



6



Adding the far Attribute to Functions in the C Source Code

Using the _far qualifier

 Use the ___far qualifier to explicitly specify the far attribute for functions so as not to affect the memory model settings.





Adding Section Settings in the C Source Code

- Using #pragma section
 - Change the section name to be output by default.
 - Specification format
 - #pragma section [section type] [new section name]
 - Section type
 - text, const, data, and bss
 - Example:



RENESAS



Linker Settings (Section Setting for Mapping from ROM to RAM) (1/2)

- Use the -rom linker option to specify the ROM and RAM sections for the functions to be executed in RAM.
 - Example: e² studio





Linker Settings (Section Setting for Mapping from ROM to RAM) (2/2)

• Example: CS+

1.1										
\land	CC-RL Property	· · · +								
۵	Section									
	Layout sections automatically	No								
	Section start address	.const,.text,.RLIB,.SLIB,.textf,.constf,.data,.sdata/02000,ram_text_f/0								
⊳	Section that outputs external defined symbols to the file	Section that outputs external defined symbols to the file[0]								
⊿	ROM to RAM mapped section	ROM to RAM mapped section[3]								
	[0]	.data=.dataR								
	[1]	sdata= sdataR								
	[2]	ram_text_f=ram_text_fR								
⊳	Verify									
⊳	Message									
⊳	> Others									
ROM to RAM mapped section Specify ROM to RAM mapped section in the format of " <rom name="" section="">=<ram name="" section="">", one per line. This option corresponds to the -ROm option of the rlink command.</ram></rom>										
1	Common Options 🖌 Compile Options 🖌 Assemble Option	Link Options He Output Options / I/O Header File Generation Options / 🖛								



Linker Settings (Section Allocation Settings)

- Use the -start linker option to specify allocation of the ROM and RAM sections for the functions to be executed in RAM.
 - Examples: e² studio

CS+

Address	Section Name		Address	Section		<u>A</u> dd
			0x02000	.const		Modify.
0x00002000	.const			.text		
	.text			.RLIB		New Over
	.RLIB			.SLIB		Remov
	.SLIB			.textf		
	.textf			.constf		
	.constf			.data		
	.data	Add Section				
	edata		0×04000	ram_text_f		
0x00004000	ram_text_f	Remove Section	0xFEF00	 .dataR	J	
0X000FEF00	Juatan			bss		
0.00055.000	1000	Move Up	0xFFC00	ram_text_fR		
0x000FFC00	ram_text_fR		ALFFE20	.oddta R	J	Import.
0x000FFE20	.sdataK	Move Down		.sbss		
	.sbss					Export.
				ОК	Cancel	Help



Adding a Routine for Copying Functions to RAM in the C Source Code

- Add a routine for copying the functions to be executed in RAM.
 - Use <u>sectop</u> and <u>secend</u> in the routine.
 - Execute this copying routine in advance.





Sample Program

The following shows a sample program that uses the codes created through the procedures described before.

```
void copyroutine(void);
                           /* Prototype declaration of a copying routine*/
  <u>_far</u> char f1(char a);
                           /* Prototype declaration of a function to be executed in RAM */
  far int f2(int x);
                            /* Prototype declaration of a function to be executed in RAM */
int a;
void func(void)
           /* Call the copying routine */
           copyroutine();
           /* Call the functions to be executed in RAM */
           a = f2(a);
```







Renesas System Design Co., Ltd.

©2015 Renesas System Design Co., Ltd. All rights reserved.