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

Introduction

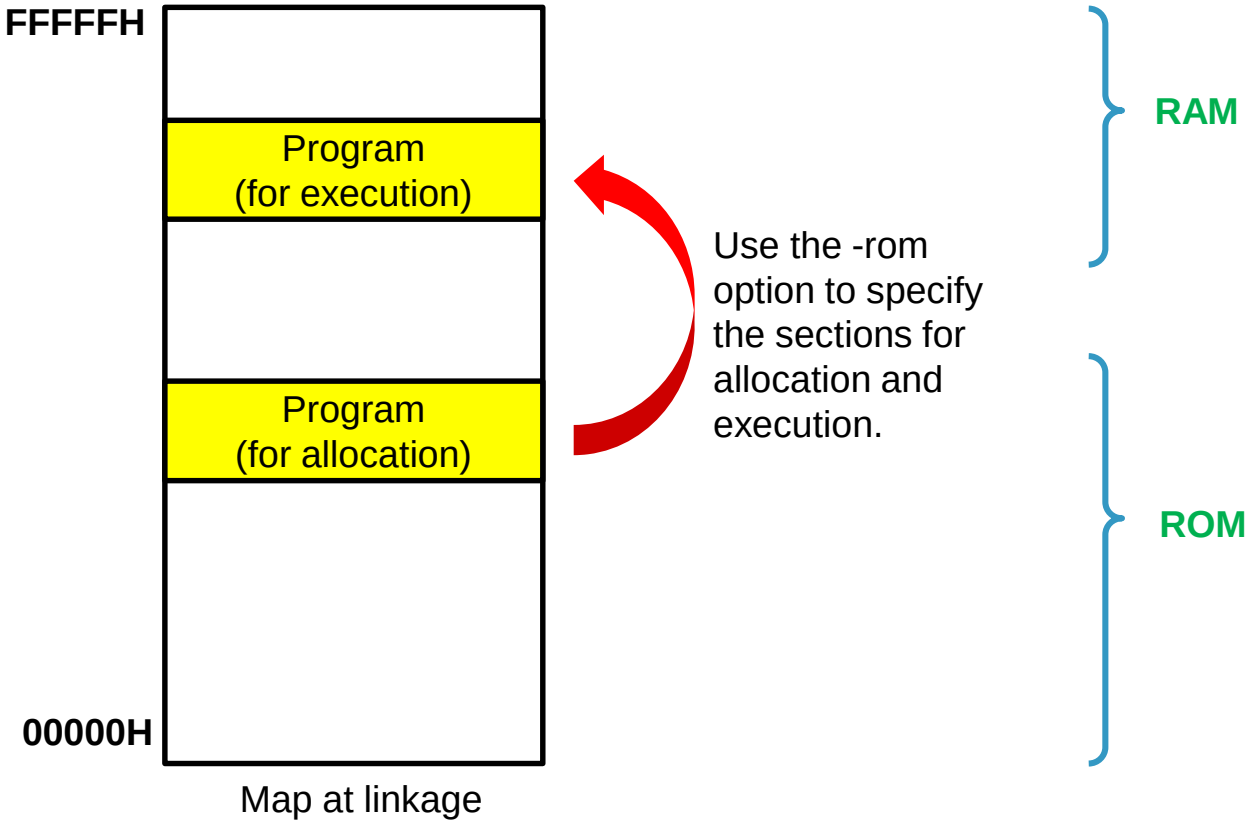
- This document describes how to copy a program to RAM and execute it in RAM when using the CC-RL C compiler for the RL78 family.
- This document uses the following tools and versions for descriptions.
  - CC-RL C compiler for the RL78 family V.1.01.00
  - e^2 studio integrated development environment 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
- Sample Program

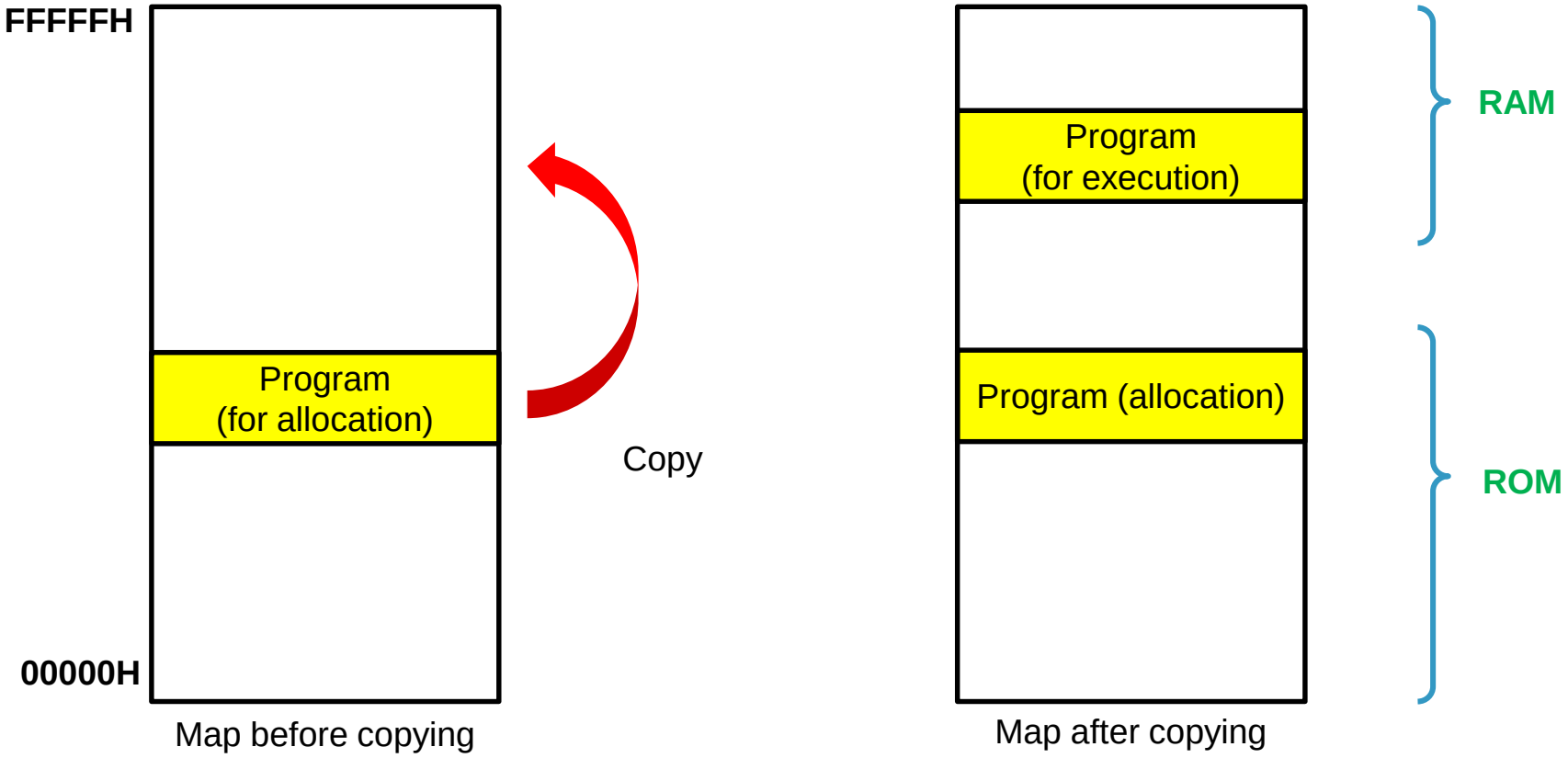
How to Execute a Program in RAM

- 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

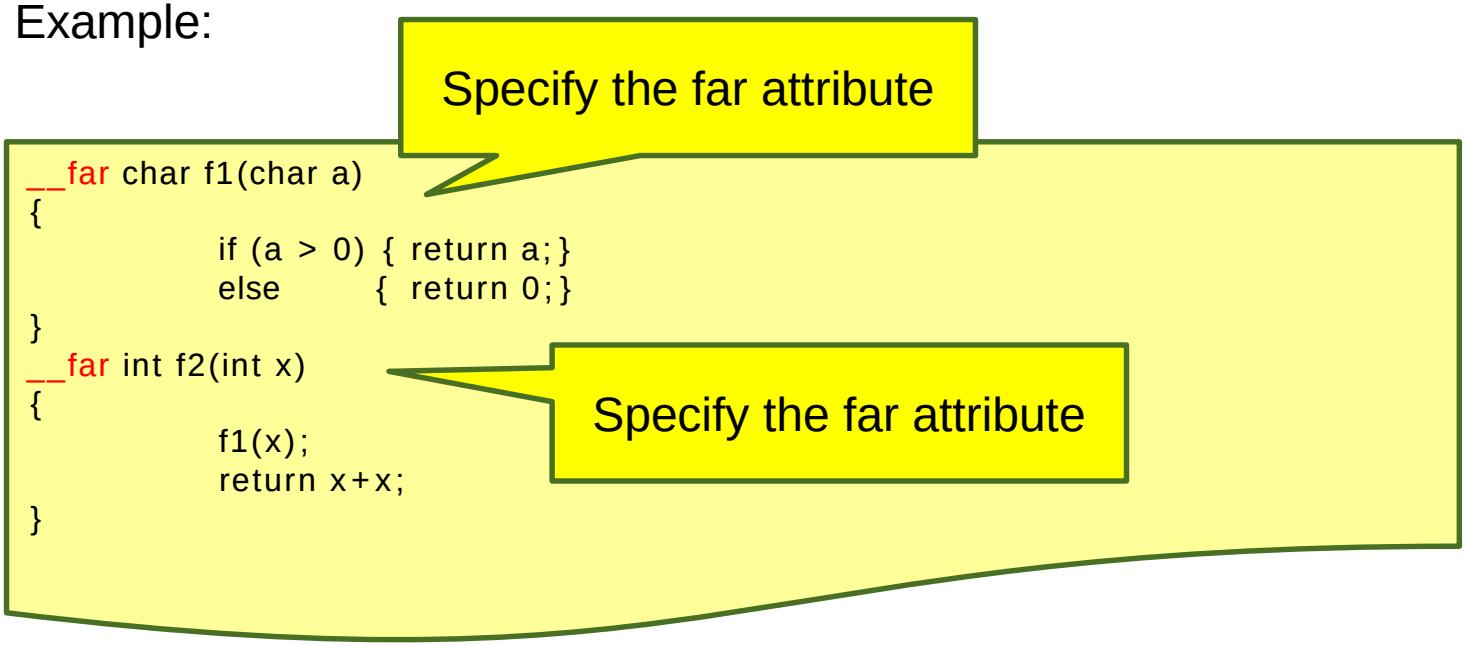


Memory Maps for Program Execution in Microcontroller



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.
  - Example:



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:

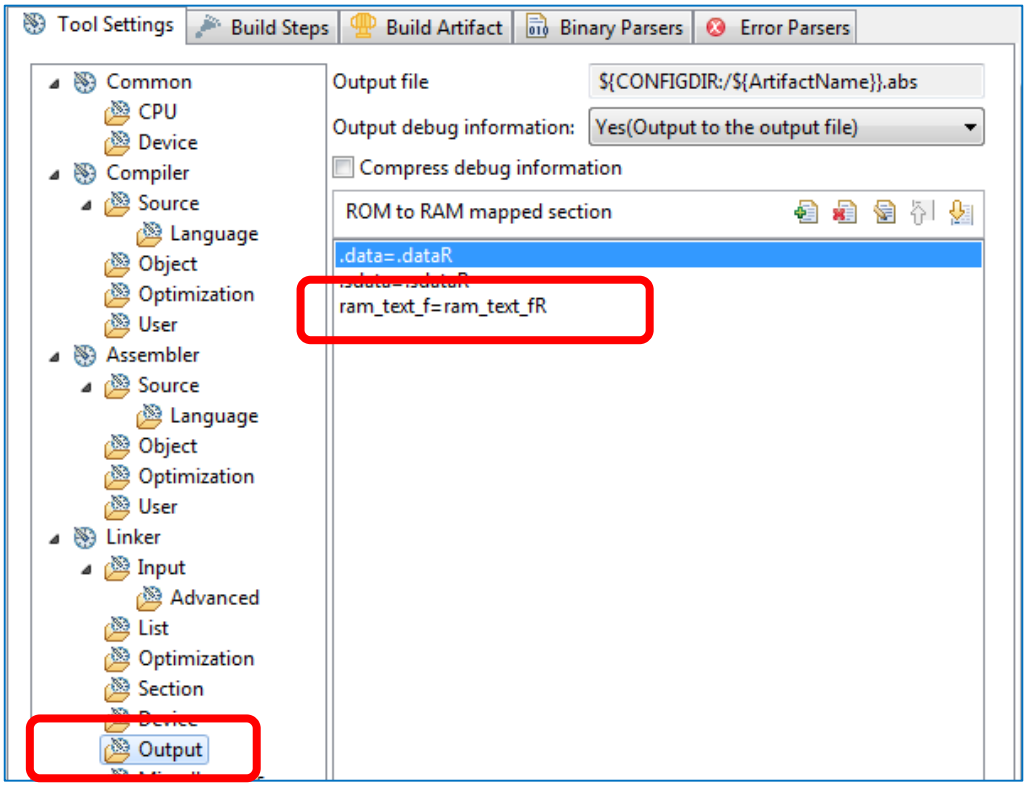
```
#pragma section text ram_text
__far char f1(char a)
{
            if (a > 0) { return a;}
            else       {  return 0;}
}
__far int f2(int x)
{
            f1(x);
            return x+x;
}
#pragma section
```

Change to a user-specified section name.

Restore the default section name.

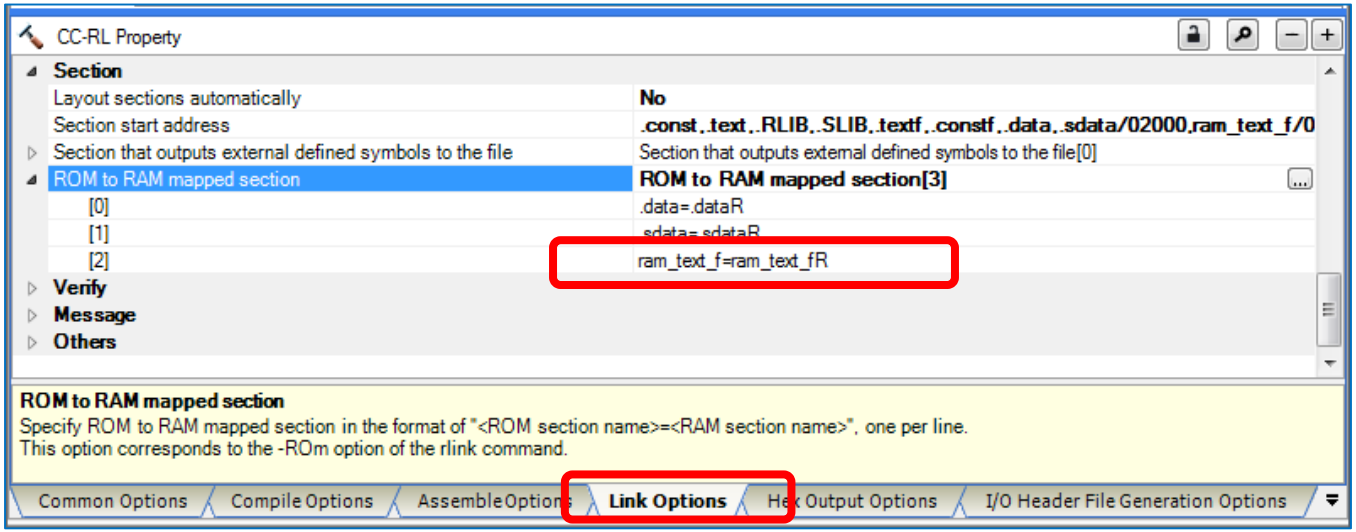
Linker Settings
(Section Setting for Mapping from ROM to RAM) (1/2)

- Use the -rom linker option to specify the ROM and RAM sections for the functions to be executed in RAM.
  - Example: e^2 studio



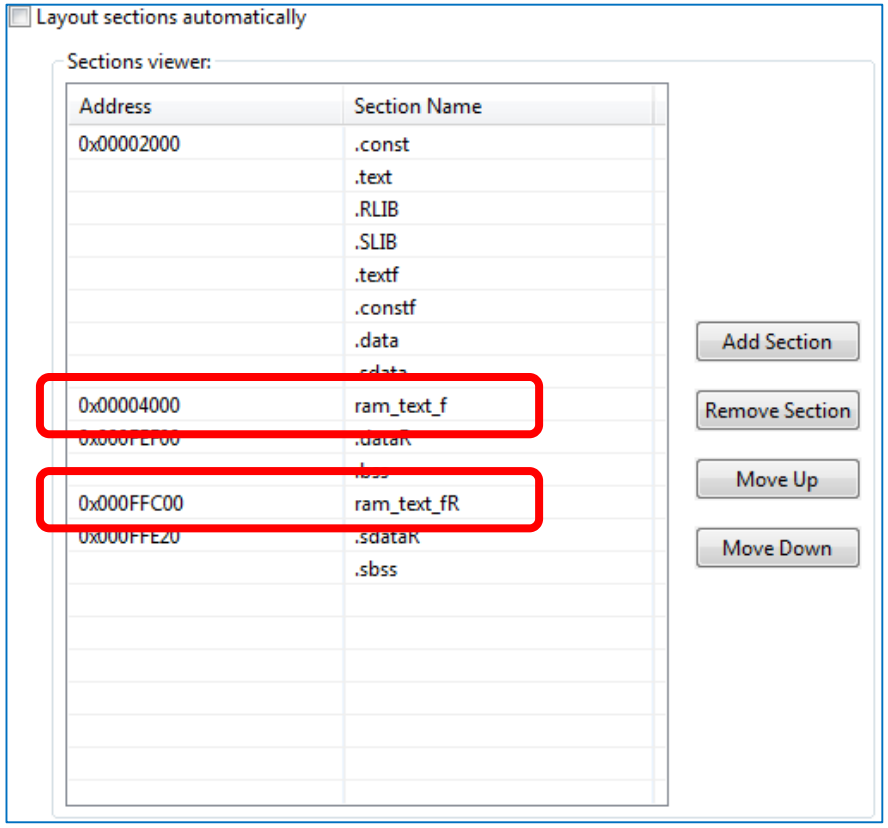
Linker Settings
(Section Setting for Mapping from ROM to RAM) (2/2)

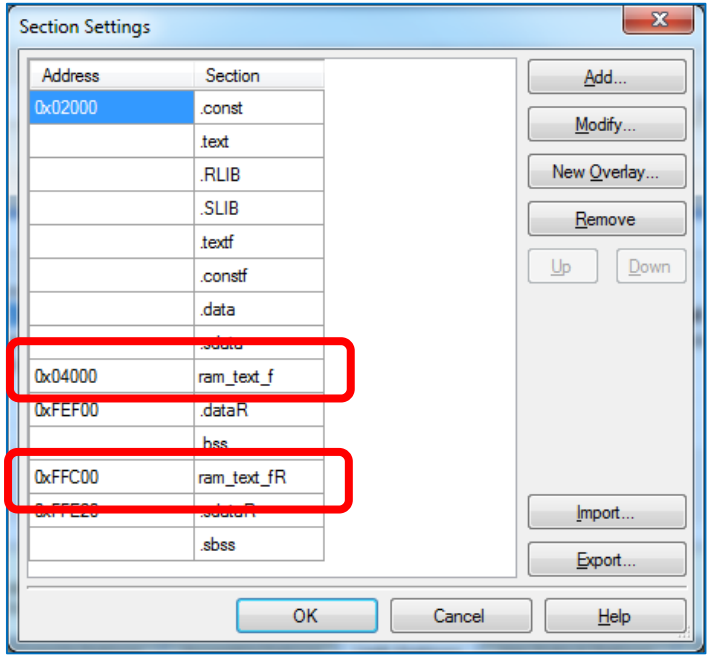
- Example: CS+



Linker Settings (Section Allocation Settings)

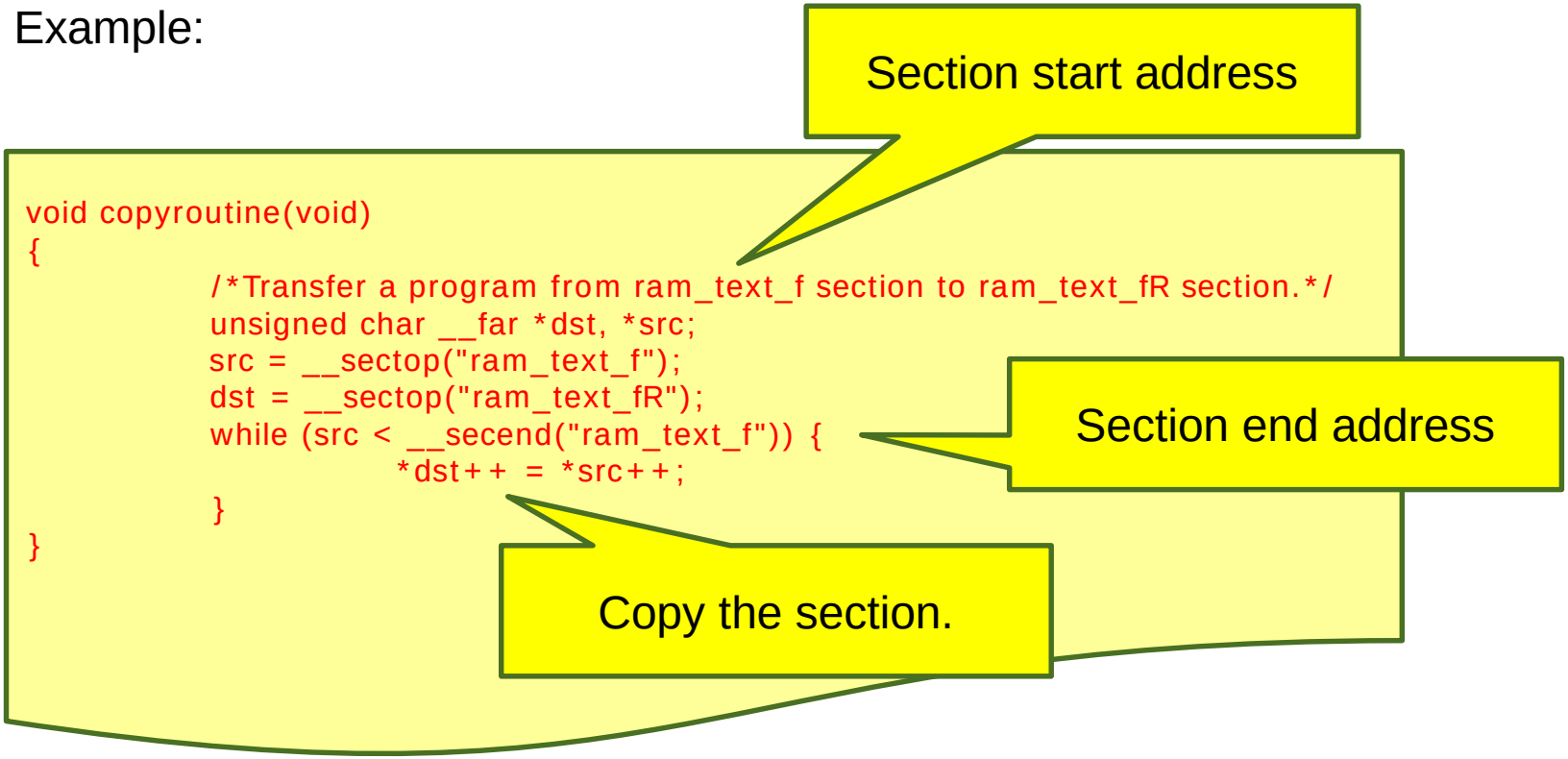
- Use the -start linker option to specify allocation of the ROM and RAM sections for the functions to be executed in RAM.
  - Examples: e^2 studio　　CS+





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 __sectop and __secend in the routine.
  - Execute this copying routine in advance.
  - Example:



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*/
__far 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.