Addition of Variable Sections
CC-RL C Compiler for RL78 Family
Introduction

This document describes how to change the section names to be generated by default and add new sections when using the CC-RL C compiler for the RL78 family.

This document uses the following tools and versions for description.

- 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 Change Variable Sections
- Adding Section Settings to the C Source
- Adding Settings in the Linker
- Adding Initialization Processing
  - Creating the Initialization Routine (C Language)
  - Modifying the Startup Routine (cstrat.asm)
How to Change Variable Sections

- Adding section settings to the C source
  - Change the names of the variable sections with #pragma section.

- Adding settings in the linker
  - Specify the section for initialized variables as the section mapped from ROM to RAM.

- Adding initialization processing
  - As the startup routine has only the processing for the default sections, either of the following processes should be added.
  - Creating the initialization routine (C language)
    - Create initialization tables and an initialization function and call the created function.
  - Modifying the startup routine (cstrat.asm)
    - As this routine has only the processing for the default sections, the following should be added.
      - Add the processing for initializing the uninitialized variable area to 0.
      - Add the processing for copying initial values to the initialized variable area.
Adding Section Settings to the C Source

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

```c
#pragma section data Mydata
__near unsigned char a0 = 0, a1 = 1, a2 = 2;
#pragma section bss Mybss
__near unsigned char b0, b1, b2;
#pragma section
```
Adding Settings in the Linker (1/2)

- Specifying the section for initialized variables as the section mapped from ROM to RAM.
  - Specify the target section with the linker option -rom.
  - Example: e² studio
Adding Settings in the Linker (2/2)

- Example: CS+

![Image of Linker settings configuration]

- ROM to RAM mapped section
  - Specify ROM to RAM mapped section in the format of "<ROM section name>=<RAM section name>", one per line.
  - This option corresponds to the -ROM option of the rlink command.
Creating the Initialization Routine (C Language) (1/4)

- Defining an initialization table (for uninitialized variables)
  - Define the section addresses and size to be used by the initialization function.
  - Remark:
    - The examples in this section (Creating the Initialization Routine (C Language)) use structures that enable multiple sections to be handled.
  - Example: initsct.c

Add the following processing with the name **in blue** changed to the section name output with the #pragma section specification.

```c
#define BSEC_MAX 1 /* Number of BSS sections to be initialized to 0 */

const struct bsec_t {
    char __near *ram_sectop; /* Section start address */
    char __near *ram_secend; /* Section end address + 1 */
} bsec_table[BSEC_MAX] = {
    {(char __near *)__sectop("Mybss_n"),
     (char __near *)__secend("Mybss_n")}};
```
Defining an initialization table (for initialized variables)

- Define the section addresses and size to be used by the initialization function.
- Example: initsct.c

Add the following processing with the `name in blue` changed to the section name output with the `#pragma` section specification and the `name in purple` changed to the section name specified with the `-rom` option.

```c
#define DSEC_MAX 1 /* Number of DATA sections to be copied */

const struct dsec_t {
    char __far *rom_sectop; /* Start address of copy source section */
    char __far *rom_secend; /* End address of copy source section + 1 */
    char __near *ram_sectop; /* Start address of copy destination section */
} dsec_table[DSEC_MAX] = {
    {__sectop("Mydata_n"),
     __secend("Mydata_n"),
     (char __near *)__sectop("Mydata_nR"))};
```
Creating the Initialization Routine (C Language) (3/4)

Creating an initialization function

- Create a function for clearing the uninitialized variables to 0 and copying the initial values to the initialized variables by using the initialization tables.
- Call this function from the main function, etc.
Creating the Initialization Routine (C Language) (4/4)

- Example: initsct.c

```c
#define BSEC_MAX 1 /* Number of BSS sections to be initialized to 0*/
#define DSEC_MAX 1 /* Number of DATA sections to be copied*/

void INITSCT_RL(void)
{
    unsigned int i;
    char __far *rom_p;
    char __near *ram_p;
    for (i = 0; i < BSEC_MAX; i++) {
        ram_p = bsec_table[i].ram_sectop;
        for ( ; ram_p != bsec_table[i].ram_secend; ram_p++) {
            *ram_p = 0;
        }
    }
    for (i = 0; i < DSEC_MAX; i++) {
        rom_p = dsec_table[i].rom_sectop;
        ram_p = dsec_table[i].ram_sectop;
        for ( ; rom_p != dsec_table[i].rom_secend; rom_p++, ram_p++) {
            *ram_p = *rom_p;
        }
    }
}
```

- Initialize to 0.
- Copy the initial values.
Modifying the Startup Routine (cstrat.asm) (1/2)

- Adding the processing for initializing the uninitialized variable area to 0
  - Add the processing for clearing the target section area to 0 by using the section name.
  - Example
    - Add the following processing with the name in blue changed to the section name output with the #pragma section specification.

```assembly
; clear external variables which doesn't have initial value (Mybss_n)
MOVW HL,#LOWW(STARTOF(Mybss_n))
MOVW AX,#LOWW(STARTOF(Mybss_n) + SIZEOF(Mybss_n))
BR $.L2_Mybss_n
.L1_Mybss_n:
  MOV [HL+0],#0
  INCW HL
.L2_Mybss_n:
  CMPW AX,HL
  BNZ $.L1_Mybss_n
```

- Initialize to 0.
Modifying the Startup Routine (cstrat.asm) (2/2)

- Adding the processing for copying the initial values to the initialized variable area.
  - Add the processing for copying the initial values by using the section name.
  - Example
    Add the following processing with the name in blue changed to the section name output with the #pragma section specification and the name in purple changed to the section name specified with the -rom option.

```assembly
; copy external variables having initial value (Mydata_n)
MOV ES,#HIGHW(STARTOF(Mydata_n))
MOVW BC,#LOWW(SIZEOF(Mydata_n))
BR $.L2_Mydata_n
.L1_Mydata_n:
  DECW BC
  MOV A,ES:LOWW(STARTOF(Mydata_n))[BC]
  MOV LOWW(STARTOF(Mydata_nR))[BC],A
.L2_Mydata_n:
  CLRW AX
  CMPW AX,BC
  BNZ $.L1_Mydata_n
```