RL78族 集成开发环境

从CA78K0R转至CCRL的使用指南

（工程操作篇）

瑞萨电子（中国）有限公司

2016/3/1　　Rev.1.00

RENESAS 瑞萨

目录

- 前言 第03页
- 从现有工程向 CS+的CC-RL工程的转移 第05页
- 与CA78K0R工程的差异 第10页

RENESAS 瑞萨

前言

- 本资料描述了将RL78族C编译器CA78K0R的工程转至RL78族C编译器CC-RL时，对CS+工程的操作方法。
- 本资料的说明对象为：集成开发环境CS+、RL78族C编译器CA78K0R、RL78族C编译器CC-RL。

  说明对象的版本如下：

  - CS+ V3.01.00
  - CA78K0R V1.20或更高
  - CC-RL V1.01.00

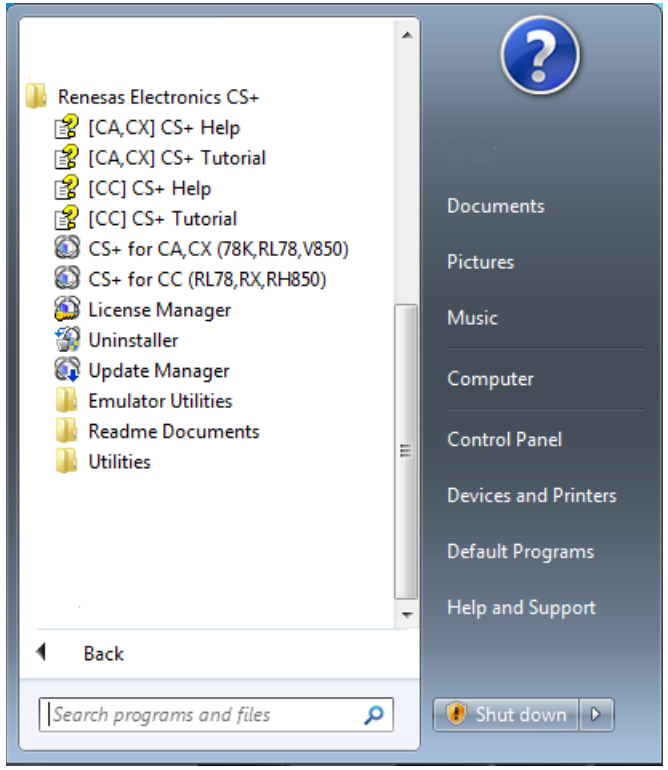
- 从现有工程向CS+的CC-RL工程的转移
  - 启动CS+
  - 新建工程的工程转移
  - 沿用现有工程的工程转移

- 与CA78K0R工程的差异
  - 生成文件
  - 启动文件
  - iodefine.h
  - 段配置
  - 堆栈配置
  - 优化选项

从现有工程向CS+的CC-RL工程的转移

从现有工程向CS+的CC-RL工程的转移
CS+的启动

- 处理CC-RL工程的CS+为：CS+ for CC。
- 从Windows开始菜单中选择并启动“CS+ for CC(RL78,RX,RH850)”。



从现有工程向CS+的CC-RL工程的转移

- 要将已建CS+/Cubesuite的CA78K0R工程转至CS+的CC-RL工程中，可通过以下两种方法实现。
  - 1.通过CS+新建一个工程

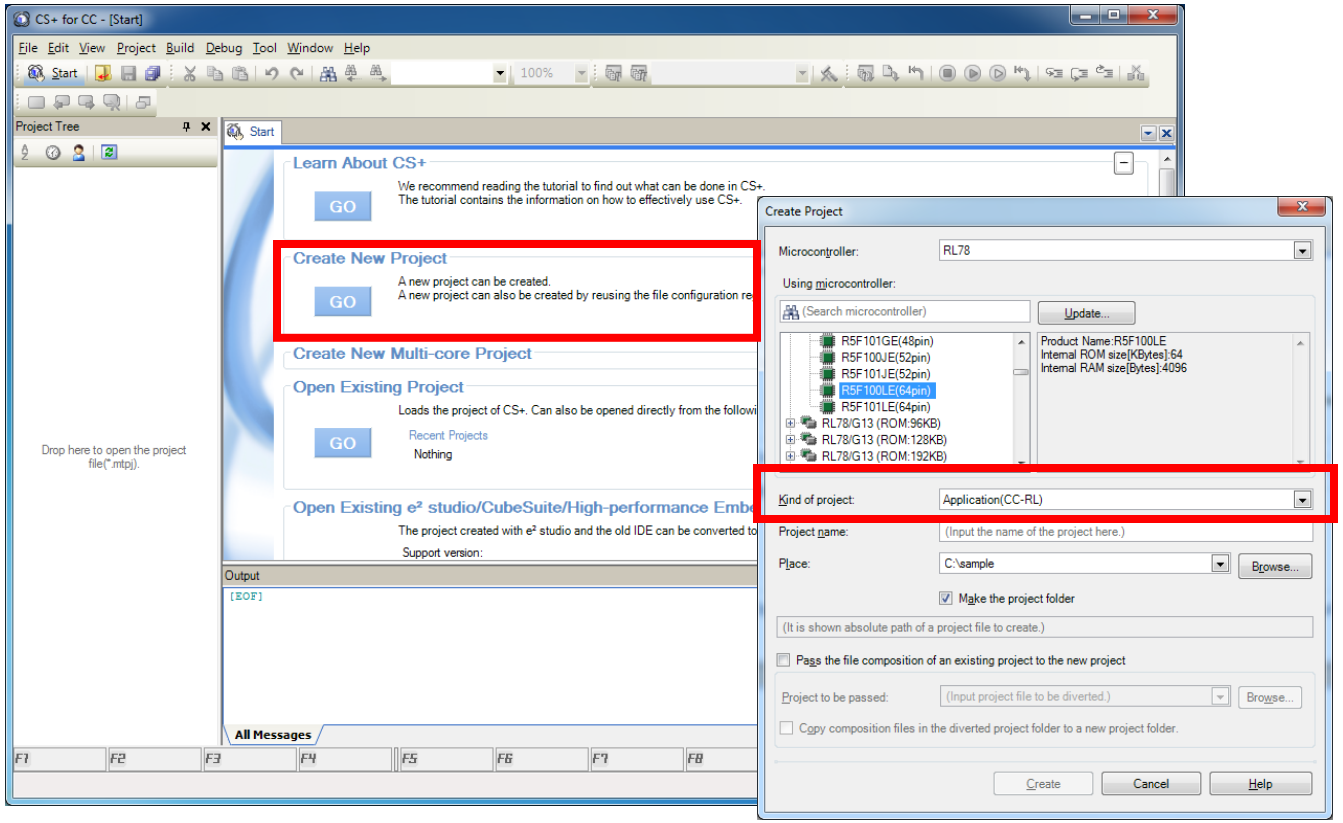
    新建一个RL78的CS+工程，然后将已有的用户源文件进行登录。
  - 2.沿用现有工程

    沿用CA78K0R的CS+/Cubesuite工程，新建一个CS+的CC-RL工程。

内容	方法1	方法2
登录源文件	手动	自动
选项设置	手动	自动（仅部分）
源文件的文件夹位置	无需考虑直接登录	需保持被沿用文件的原结构
源文件与自动生成文件产生冲突	手动登录时需考虑	生成工程后需要更改

通过CS+新建CC-RL工程

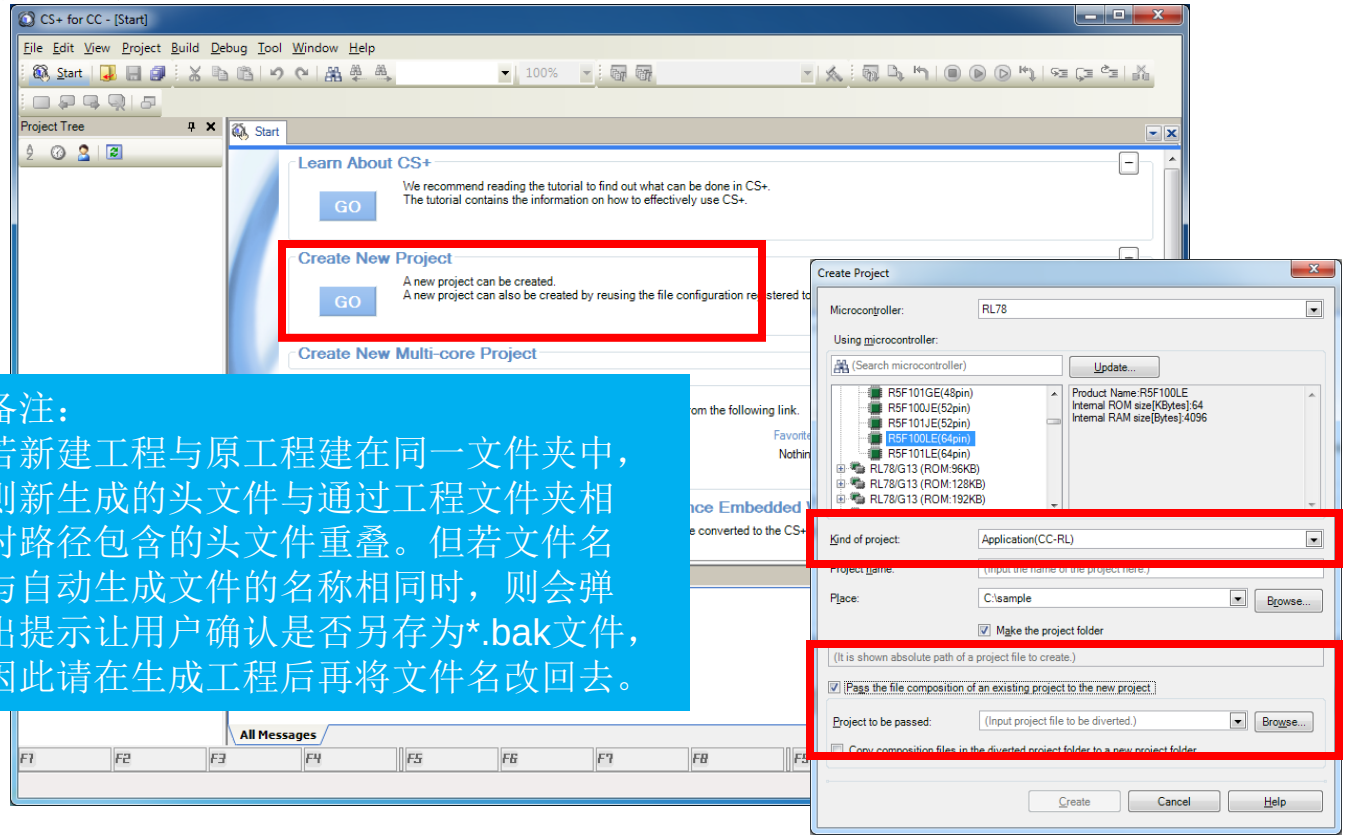
- 通过CS+新建一个CC-RL工程



新建工程后即可登录使用已有的CA78K0R工程。

沿用现有工程的工程转移

- 通过CS+新建一个CC-RL工程



备注：
若新建工程与原工程建在同一文件夹中，则新生成的头文件与通过工程文件夹相对路径包含的头文件重叠。但若文件名与自动生成文件的名称相同时，则会弹出提示让用户确认是否另存为*.bak文件，因此请在生成工程后再将文件名改回去。

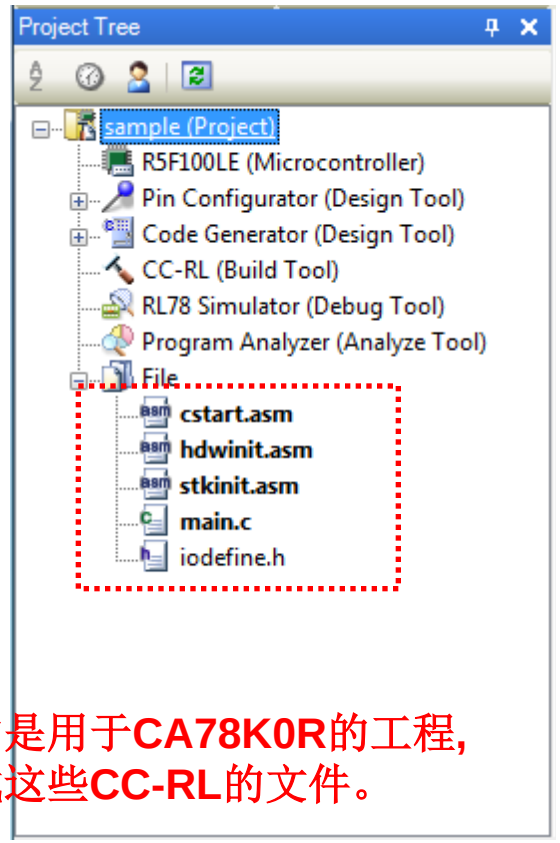
新建工程时选“Pass the file composition of an existing project to the new project”。然后选择用CA78K0R建的工程。

与CA78K0R工程的差异

与CA78K0R工程的差异：生成文件

- 新建CC-RL工程时，会生成如下开发所需文件。
  - 启动文件（ cstart.asm）
  - 初始化hdwinit函数文件（hdwinit.asm）
  - 堆栈初始化stkinit函数文件（stkinit.asm）

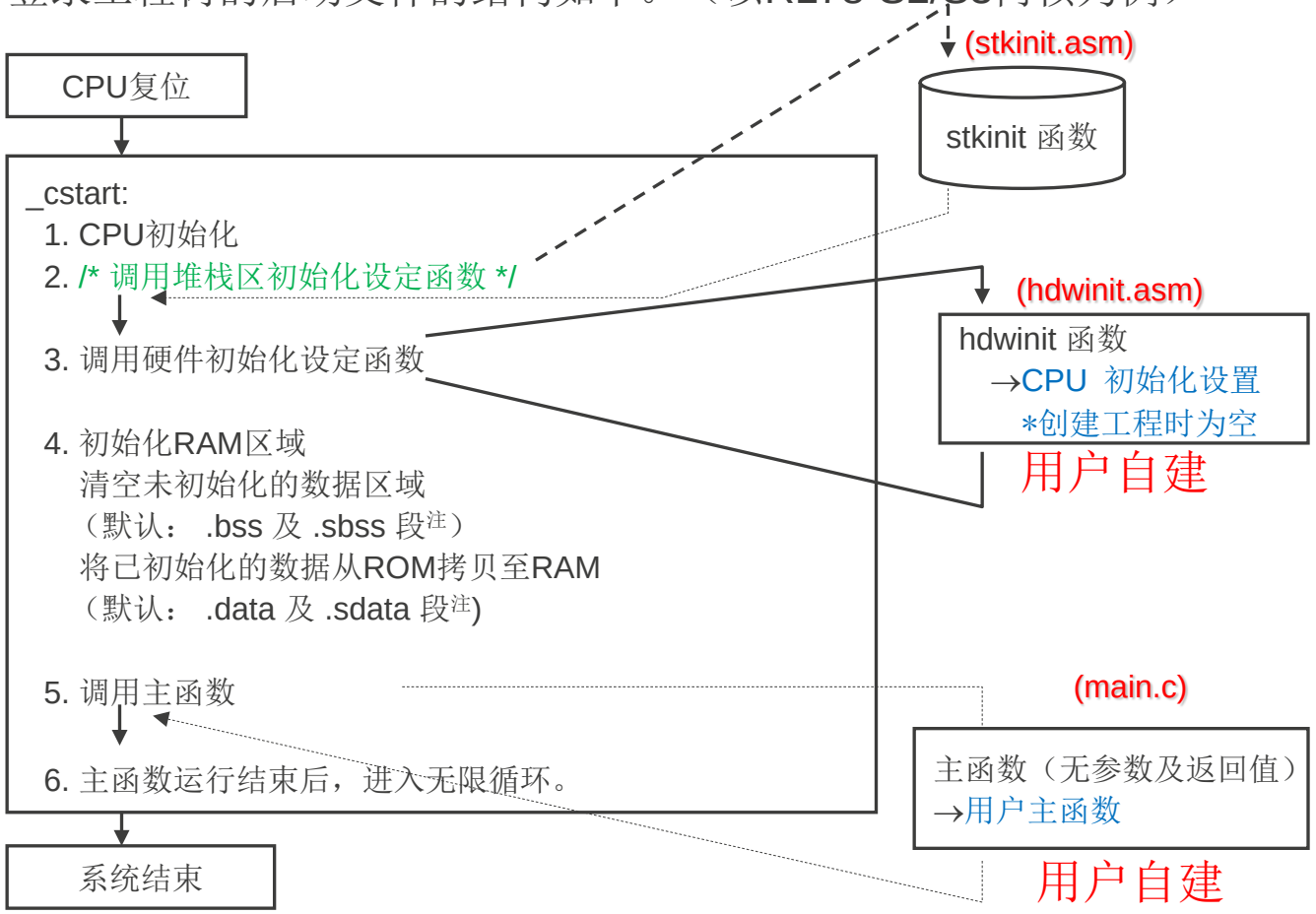
    （RL78-S1内核的MCU不生成此文件。）
  - 主函数文件（main.c）
  - SFR文件（iodefine.h）



注：
如果所建的是用于CA78K0R的工程，
则不会生成这些CC-RL的文件。

与CA78K0R工程的差异：启动文件（1）

- 登录工程树的启动文件的结构如下。（以RL78-S2/S3内核为例）

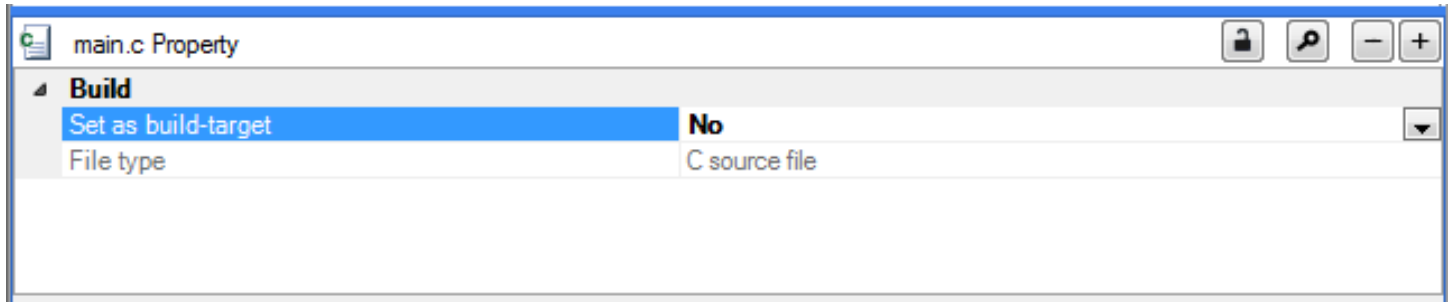


函数的调出已被注释掉，需要时请自行调出。（如使用MCU的RAM奇偶功能时。）

注： far变量的.bssf、.dataf段已被注释掉。
若需定义far变量，请将其设为有效。

与CA78K0R工程的差异：启动文件（2）

- 若转移目标工程中已存在main函数和hdwinit函数，则可通过以下任意一种方法将作成工程时自动生成的文件排除在所建对象之外。
  - 从工程树中删除注册文件。
  - 打开已注册到工程树的main.c、hdwinit.asm文件的属性，在“Set as build-target”处选择“No”。



与CA78K0R工程的差异：iodefine.h（1）

- 本文件中的声明可使一个C源文件获取RL78的SFRs。

```
<iodefine.h>
• • •
typedef struct
{
    unsigned char no0:1;
    unsigned char no1:1;
    unsigned char no2:1;
    unsigned char no3:1;
    unsigned char no4:1;
    unsigned char no5:1;
    unsigned char no6:1;
    unsigned char no7:1;
} __bitf_T;
• • •
#define ADM2      (*(volatile __near unsigned char  *)0x10)
#define ADM2_bit (*(volatile __near __bitf_T *)0x10)
#define P0          (*(volatile __near unsigned char  *)0xFF00)
#define P0_bit      (*(volatile __near __bitf_T *)0xFF00)
#define P1          (*(volatile __near unsigned char  *)0xFF01)
#define P1_bit      (*(volatile __near __bitf_T *)0xFF01)
 • • •
#define INTP0                    0x0008
#define INTP1                    0x000A
#define INTP2                    0x000C
#define INTP3                    0x000E
 • • •
```

```
<获取寄存器的文件>
#include “iodefine.h”
• • •
void main(void)
{
        • • •
        ADM2 = 0x12;
        ADTYP = 1;
        P0_bit.no2 = 1;
        • • •
}
• • •
#pragma interrupt inter (vect=INTP0)
void __near inter ( void ) {
      /*中断处理*/
}
• • •
```

<获取方法>

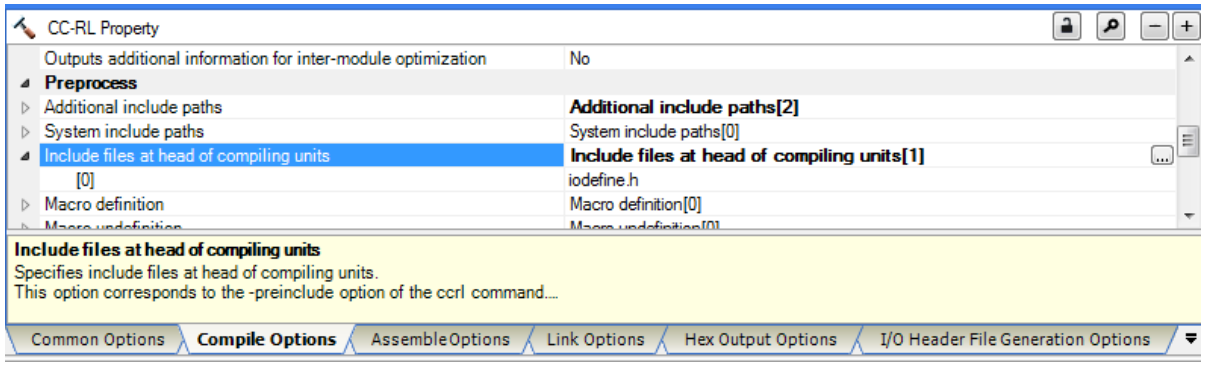
- 可使用 iodefine.h文件中的描述来获取SFR寄存器及其位。
- 对于那些未定义名称的位，可使用带有“_bit”的名称来获取。
- 可通过 #pragma interrupt来声明使用的中断函数。

RENESAS 瑞萨

与CA78K0R工程的差异：iodefine.h（2）

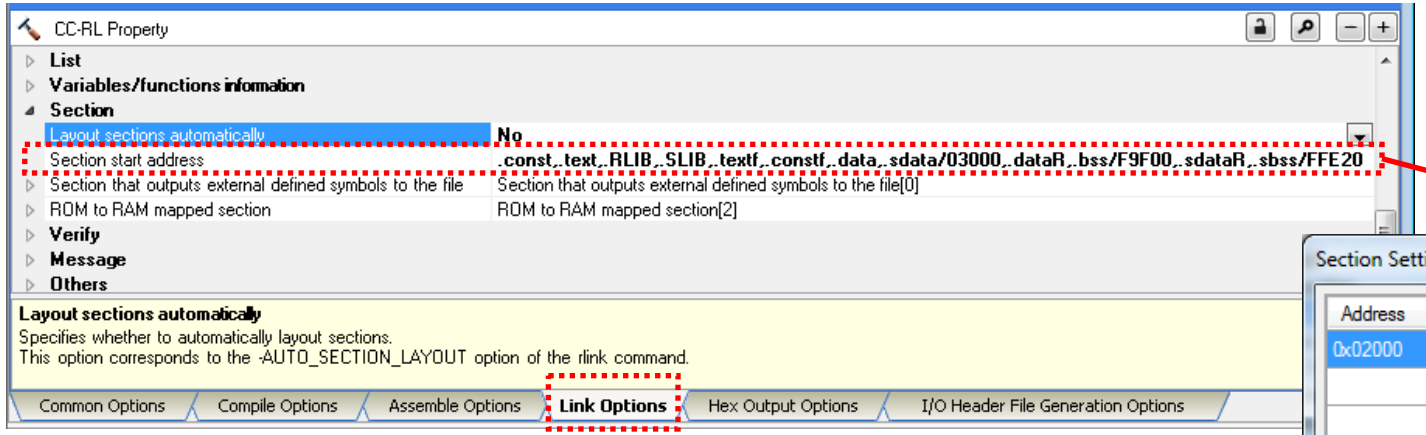
- 可通过以下任意一种方法将 iodefine.h 包含到源文件中。
  - 在各个源文件中写入 #include “iodefine.h”
    - 每个源文件都需要写。
    - 需在向量表（#pragma interrupt）指定的中断请求名和中断所用SFR之前写包含。
  - 在编译选项中指定 -preinclude = iodefine.h
    - 适用于全部的源文件。
    - 若SFR名和中断请求名用作他用，则#define会被iodefine.h中的定义所替换。

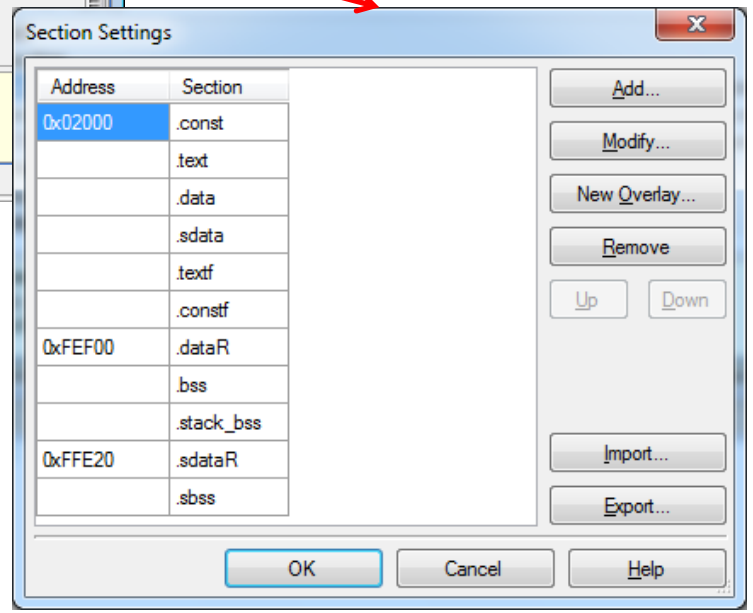
```
<C源文件示例>
#include “iodefine.h”
• • •
void main(void)
{
        ADM2 = 0x12;
        P0_bit.no2 = 1;
}
#pragma interrupt inter (vect=INTP0)
void __near inter ( void ) {
    /*中断处理*/
}
```



与CA78K0R工程的差异：段配置（1）

- 程序及数据的段配置可通过属性中的Link Options来指定。





关于段配置，要指定由编译器生成的段名称。

参照通过CA78K0R作成的链接指令文件，变更配置。

可将任意段指定至任意地址

RENESAS 瑞萨

与CA78K0R工程的差异：段配置（2）

- CC-RL中，通过默认段名来生成段。

默认段名称	重配置属性	内容
.callt0	CALLT0	用于callt函数调用表的段
.text	TEXT	用于代码部分的段（配置于near区域）
.textf	TEXTF	用于代码部分的段（配置于far区域）
.textf_unit64kp	TEXTF_UNIT64KP	用于代码部分的段（将此段的起始地址配置为偶数，且不跨越64 KB – 1边界）
.const	CONST	ROM数据（配置于near区域）（镜像区域内）
.constf	CONSTF	ROM数据（配置于far区域）
.data	DATA	用于初始化数据的段（有初始值，配置于near区域）
.dataf	DATAF	用于初始化数据的段（有初始值，配置于far区域）
.sdata	SDATA	用于初始化数据的段（有初始值，变量配置于saddr）
.sbss_bit	SBSS_BIT	用于位数据区域的段（无初始值，变量配置于saddr）
.bss_bit	BSS_BIT	用于位数据区域的段（无初始值，配置于near区域）

默认段名称	重配置属性	内容
.bss	BSS	用于数据区域的段（无初始值，配置于near区域）
.bssf	BSSF	用于数据区域的段（无初始值，配置于far区域）
.sbss	SBSS	用于数据区域的段（无初始值，变量配置于saddr）
.option_byte	OPT_BYTE	用于用户选项字节以及片上调试设置的指定专用段
.security_id	SECUR_ID	用于安全ID的指定专用段
.vect注	AT	中断向量表
.dataR	DATA	用于初始化数据RAM的段（有初始值，配置于near区域） 在启动文件中定义
.sdataR	DATA	用于初始化数据RAM的段（有初始值，配置于saddr区域） 在启动文件中定义

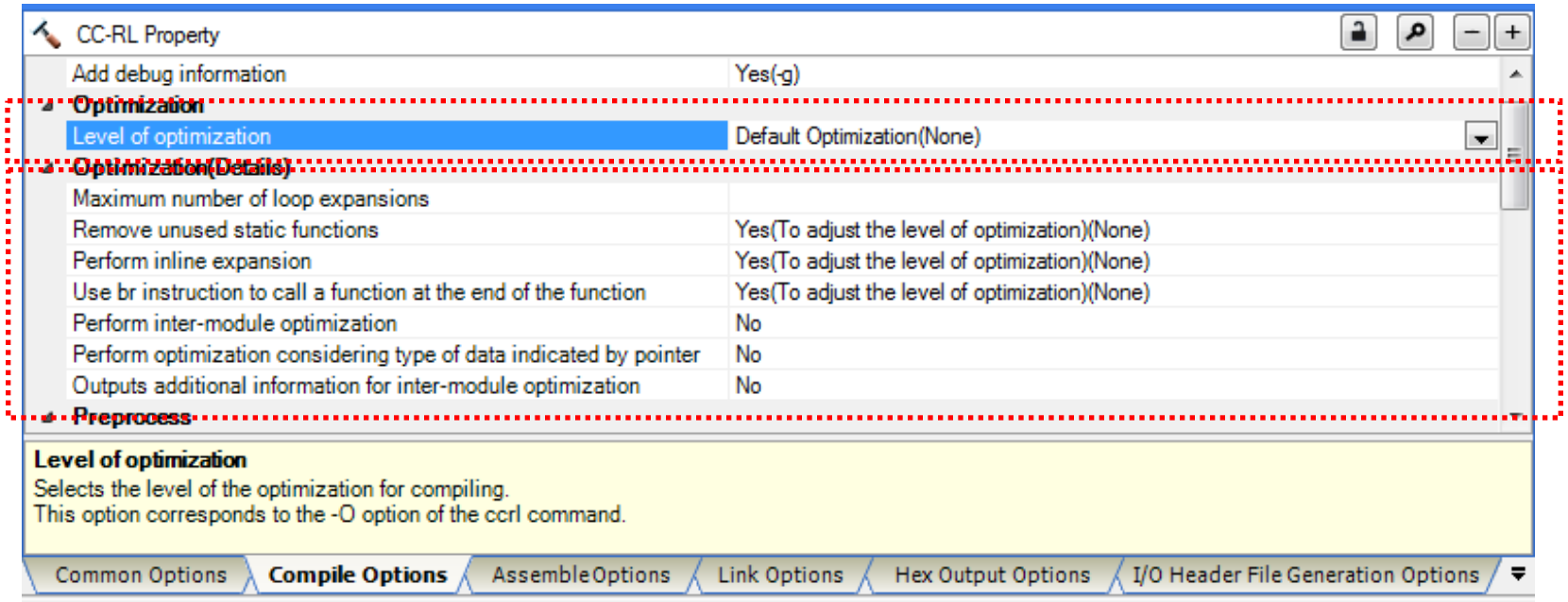
注： 不可通过 #pragma section来更改段名称。

与CA78K0R工程的差异：优化选项（1）

- 适用于瑞萨编译器和链接编辑器的优化技术，在RL78单片机的适用性上进一步加以提升（最优化的寄存器分配、最优化的指令选择和指令调度等），从而生成简洁紧凑的代码。
  - 编译器的优化
    - 优化的简单选择
      - 选择Size优先还是Speed优先
    - 编译时的大范围优化
      - 跨文件的函数inline展开
    - 优化的详细设置
      - 循环展开、函数inline展开、函数末尾用来调用函数的br指令的替换等
  - 优化链接编辑器的优化
    - 模块间的优化
      - 分支指令的优化
    - 优化禁用的详细设置

与CA78K0R工程的差异：优化选项（2）

- 在CC-RL（Build工具）属性的Compile Options中设置



与CA78K0R工程的差异：优化选项（3）

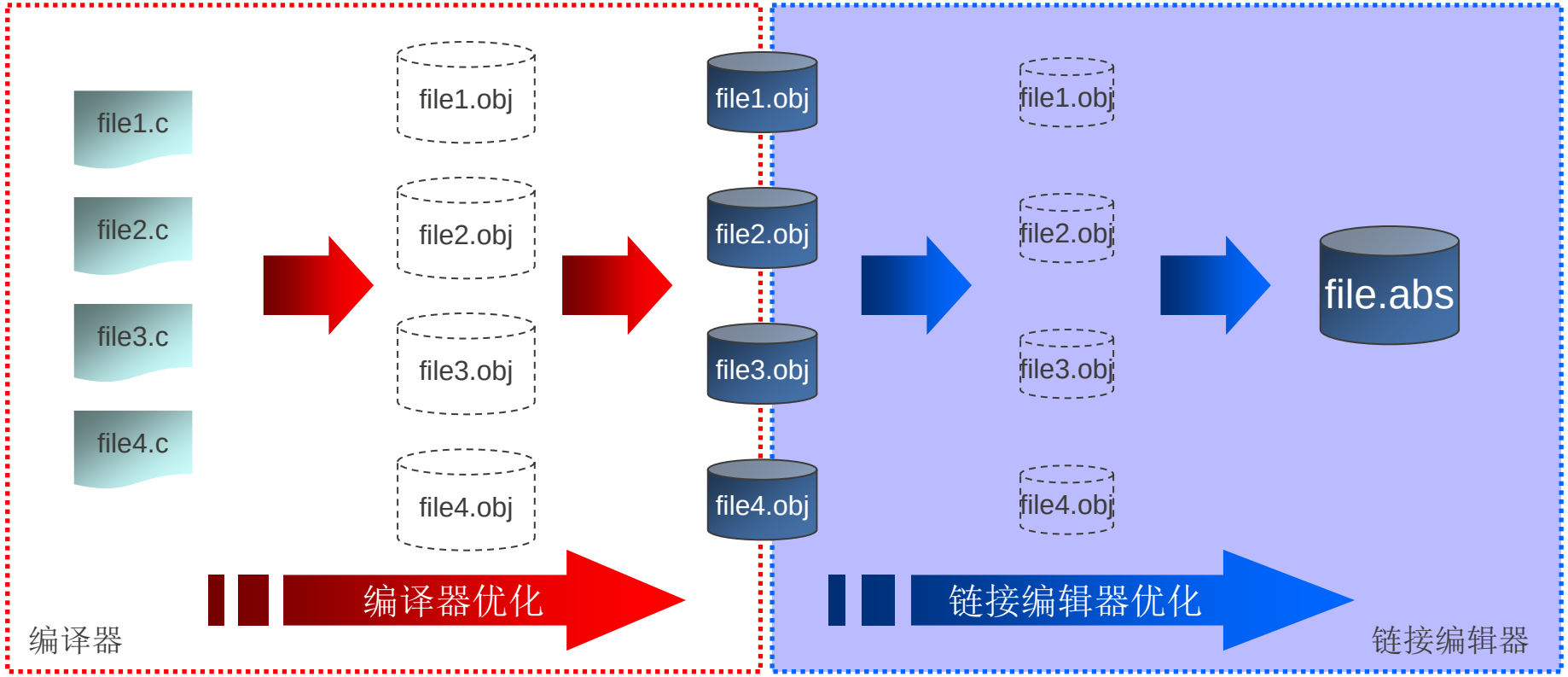
- 可选择ROM大小优先还是执行速度优先。还可以通过更详细的设置实现微调。

优化项目	说明	优化级别			
		-Osize	-Ospeed	-Odefault	-Onothing
unroll	循环展开 （最大展开倍数）	1	2	1	1
delete_static_func	删除未使用static函数	on	on	on	off
inline_level	函数inline展开（展开级别）*1	3	2	3	-
inline_size	inline展开大小 *2	0	100	0	-
tail_call	函数末尾用来调用函数的br指令的替换	On	On	On	Off

*1 展开级别
0: 阻止所有（包括#pragma inline指定函数在内的）inline展开。
1: 仅进行#pragma inline指定函数的展开。
2: 自动判断作为展开对象的函数并对其进行展开。
3: 在尽量不增加代码大小的范围内，自动判断作为展开对象的函数并对其进行展开。
然而，即便指定了1~3，也可能会因函数内容及编译状况等的不同，导致出现由#pragma inline指定的函数不会被展开的情况。
*2 inline展开
此项用来指定inline展开到使代码大小最大增至多少（%）。

与CA78K0R工程的差异：优化选项（4）

- 在RL78的搭建环境下，除编译器优化外，另有链接编辑器优化。通过链接时的信息（配置地址等）实现优化设置，从而生成更为高效高质的代码。



修订记录

版本	内容	页
Rev. 1.00	初版发行	–

www.cn.renesas.com

RENESAS 瑞萨