RL78族 集成开发环境

从CA78K0R转至CCRL的使用指南

（编码篇）

瑞萨电子（中国）有限公司

2016/3/1 Rev.1.00

RENESAS 瑞萨

前言

- 本篇资料中，记述了使用C编译器CA78K0R创建的RL78族MCU的工程或者源码转至使用C编译器CC-RL的源码的差异。
- 本篇资料以用于RL78族的C编译器CA78K0R和CC-RL作为对象进行说明。

  对象版本如下：

  - CA78K0R V1.20或更高
  - CC-RL V1.01.00

目录

RENESAS 瑞萨

编译器语言

编译器语言

- 语言规格上的差异

项目	CA78K0R	CC-RL	备注
语言	C语言	C语言	
语言标准	C89	C90、C99支持的一部分功能	
Endian	little	little	
可使用的多字节字符	EUC、SJIS	EUC、SJIS、UTF-8, big5, gbk	
支持的多字节字符的范围	标注中可以写日文	标注和字符串中可以写日文和中文	
未指定signed / unsigned的char型的处理	有符号整数 当指定-qu选项时，为无符号整数	无符号整数 当指定-signed_char选项时，为有符号整数	
double型	遵照IEEE754-1985 32-bit data	遵照IEEE754-1985 • 使用-dbl_size=4时，为32-bit data • 使用-dbl_size=8时，为64-bit data	-dbl_size=8仅对RL78-S3内核有效

具体请参照编译器用户手册进行修改。

编译器语言

- 语言规格上的差异

项目	CA78K0R	CC-RL	备注
结构体和联合体中的int型位域	按无符号处理	按无符号处理 当指定-signed_bitfield 选项时，为signed int 型。	
结构体和联合体中的位域的分配顺序	由低位到高位进行分配 当指定-rb选项时，由高位到低位进行分配	由低位到高位进行分配	
结构体和联合体中的各成员占用内存的边界	• 限定为1字节 char、signed char、unsigned char • 其它：限定为2字节	• 限定为1字节 char、signed char、unsigned char、_Bool • 其它：限定为2字节	
枚举型说明符	根据枚举常数值的范围，枚举类型为以下中的一种 signed char、unsigned char、signed int	根据枚举常数值的范围，枚举类型为以下中的一种 char、signed char、unsigned char、signed short	

具体请参照编译器用户手册进行修改。

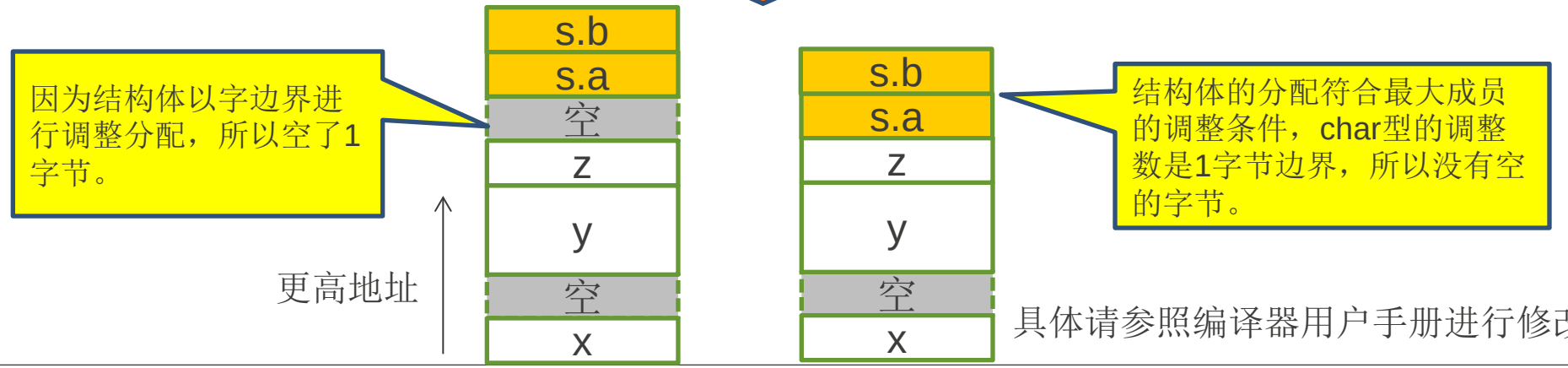
编译器语言

- 结构体和联合体中的各成员占用内存边界的差异

结构体SSS的定义

```
struct    SSS {
            char x;
            short y;
            char z;
            struct {
                        char a;
                        char b;
                    } s;
}
```

结构体SSS的分配　CA78K0R　CC-RL



具体请参照编译器用户手册进行修改。

RENESAS 瑞萨

编译器语言

- 枚举型说明符的差异

内部表现形式因枚举范围的不同而变化。

- CA78K0R的场合（优先顺序）

范围：-128 ~ 127 → signed char

范围： 0 ~ 255 → unsigned char

范围： -32768 ~ 32767 → signed int

- CC-RL的场合（优先顺序）

• 指定-signed_char

范围： -128 ~ 127（包括0 ~ 127）→ char

范围： 0 ~ 255 → unsigned char

范围： 上述以外 → signed short

• 不指定-signed_char

范围： -128 ~ 127 → signed char

范围： 0 ~ 255（包括0 ~ 127） → char

范围： 上述以外 → signed short

具体请参照编译器用户手册进行修改。

编译器语言

- 包含头文件的差异

项目	CA78K0R	CC-RL	备注
include <string> 格式的查找顺序	(1) 由-i 选项指定的文件夹 (2) 由环境变量INC78K0R指定的文件夹 (3) 包含标准include文件的文件夹	(1) 由-I 选项指定的文件夹 (2) 包含标准include文件的文件夹	
#include “string” 格式的查找顺序	(1) 包含源文件的文件夹 (2) 由-i 选项指定的文件夹 (3) 由环境变量INC78K0R指定的文件夹 (4) 包含标准include文件的文件夹	(1) 包含源文件的文件夹 (2) 由-I 选项指定的文件夹 (3) 包含标准include文件的文件夹	

具体请参照编译器用户手册进行修改。

编译器语言

- 翻译限制的差异

项目	CA78K0R	CC-RL
复合语句、循环语句和选择语句的嵌套层数	45	取决于内存大小
条件编译的嵌套层数	255	
在一个声明中，用一个算术型、构造体型、联合体型或是不完全型修饰的指针、数组和函数声明（任意组合）的个数	12	128
一个完整声明中可嵌套的声明层数	—	取决于内存大小
一个完整表达式中可嵌套的表达式的层数	1024	
宏定义名称的有效首字母的个数	256	
内部标识的有效首字母的个数	249	
外部标识的有效首字母的个数	249	
在一个翻译单位内外部标识符的个数	1024	
在一个block中具有块作用域的标识符的个数	255	
在一个翻译单位内可以被同时定义的宏标识符的个数	60000	

※ CA78K0R列，适用于V1.50及更新的版本。

具体请参照编译器用户手册进行修改。

RENESAS 瑞萨

编译器语言

- 翻译限制的差异

项目	CA78K0R	CC-RL
一个函数定义中形式参数的个数	39	取决于内存大小
一个函数调用中实际参数的个数	39	
一个宏定义中形式参数的个数	31	
一个宏调用中实际参数的个数	31	
一个逻辑源行中字符的个数	32767	
（连接后）单字符的字符串文字和宽字符串文字的字符个数	509	
（主机环境下）一个目标的字节数	65535	32767（指定-large_variable 选项时，为65535）
被#include包含的文件的嵌套层数	50	取决于内存大小
一个switch语句（嵌套了switch语句除外）中的case的个数	1024	
一个结构体或者联合体中成员的个数	1024	
一个枚举型中枚举常数的个数	255	
一列成员声明中结构体或者联合体定义的嵌套层数	15	

※ CA78K0R列，适用于V1.50及更新的版本。

具体请参照编译器用户手册进行修改。

RENESAS 瑞萨

编译器语言

- 数值限制的差异

项目	CA78K0R	CC-RL
CHAR_MIN	-128	0 (-128)
CHAR_MAX	+127	+255 (+127)
LLONG_MIN	—	-9223372036854775808
LLONG_MAX	—	+9223372036854775807
ULLONG_MAX	—	+18446744073709551615
DBL_MANT_DIG	+24	+24 (+53)
LDBL_MANT_DIG	+24	+24 (+53)
DBL_DIG	+6	+6 (+15)
LDBL_DIG	+6	+6 (+15)
DBL_MIN_EXP	-125	-125 (-1021)
LDBL_MIN_EXP	-125	-125 (-1021)

※ 对于CHAR_MIN、CHAR_MAX，当指定-signed_char时，()内的值有效。
除此以外，指定-dbl_size=8选项（仅限于RL78-S3内核）时， ()内的值有效。

具体请参照编译器用户手册进行修改。

编译器语言

- 数值限制的差异

项目	CA78K0R	CC-RL
DBL_MIN_10_EXP	-37	-37 (-307)
LDBL_MIN_10_EXP	-37	-37 (-307)
DBL_MAX_EXP	+128	+128 (+1024)
LDBL_MAX_EXP	+128	+128 (+1024)
DBL_MAX_10_EXP	+38	+38 (+308)
LDBL_MAX_10_EXP	+38	+38 (+308)
DBL_MAX	3.40282347E+38F	3.40282347E+38F (1.7976931348623158E+308)
LDBL_MAX	3.40282347E+38F	3.40282347E+38F (1.7976931348623158E+308)
DBL_ EPSILON	1.19209290E-07F	1.19209290E-07F (2.2204460492503131E-016)
LDBL_ EPSILON	1.19209290E-07F	1.19209290E-07F (2.2204460492503131E-016)
DBL_MIN	1.17549435E-38F	1.17549435E-38F (2.2250738585072014E-308)
LDBL_MIN	1.17549435E-38F	1.17549435E-38F (2.2250738585072014E-308)

※指定-dbl_size=8选项（仅限于RL78-S3内核）时， ()内的值有效。

具体请参照编译器用户手册进行修改。

RENESAS 瑞萨

编译器语言

- #pragma指令的差异

项目	CA78K0R	CC-RL	对应方法
数据插入函数__OPC( )的有效化	#pragma opc	—	请删除# pragma指令，然后使用# pragma inline _ asm和汇编指令，对数据进行插入处理。
从boot区向flash区域的函数调用	#pragma ext_func	—	没有相关联的指令。 请删除#pragma指令。 指定绝对地址，然后调用函数。
标准库函数memcpy()和memset()的inline展开指示	#pragma inline	—	请删除#pragma指令。 在CC-RL中，这个意味着用户定义的函数的inline展开。

具体请参照编译器用户手册进行修改。

编译器语言

- 宏定义的差异

CA78K0R的宏定义名称	CC-RL的宏定义名称	值
__K0R_LARGE__	None	—
CPU macro	None	—

具体请参照编译器用户手册进行修改。

编译器语言

- 关键字的差异

功能	关键字	CC-RL中的相关关键字	对应方法
near/far属性	__near/__far	__near/__far	指定位置有差别。
写入saddr区域的bit变量的声明	__boolean boolean bit	—	定义、声明结构体的位域，并请变更位处理。
asm语句	__asm #asm to #endasm	#pragma inline_asm	不可通过使用asm语句在C源代码中直接使用汇编命令。 请用汇编语言函数定义汇编指令部分，并使用#pragma inline_asm。

具体请参照编译器用户手册进行修改。

RENESAS 瑞萨

编译器语言

- 关键字的差异

功能	关键字	CC-RL中的相关关键字	对应方法
兼容78K0	__callf callf	—	不支持兼容78K0，所以请删除相应的关键字。
	noauto	—	
	__leaf norec	—	
	__pascal	—	
	__temp	—	
	__mxcall	—	

具体请参照编译器用户手册进行修改。

编译器语言

- 写入saddr区域的bit变量声明的差异
  - CA78K0R

    格式：bit (或者boolean，或者 __boolean) [变量名]
  - CC-RL
    - 因为没有位变量，所以在结构体中定义位域

```
格式：__saddr struct [标识符名] {
                    [类型名] [域名]: [位宽度];
                    [类型名] [域名]: [位宽度];
                    …
                    [类型名] [域名]: [位宽度];
                    };
```

```
__saddr struct S{
                  char a:1;
                }
```

char型 1bit的数值

    - 位域的成员，可以使用以下类型

      char, signed char, unsigned char, signed short, unsigned short, signed int, unsigned int, signed long, unsigned long, signed long long, unsigned long long

具体请参照编译器用户手册进行修改。

编译器语言

- 插入汇编指令的差异
  - CA78K0R

    格式：: #asm

    ~汇编指令~

    #endasm
  - CC-RL

    #pragma inline_asm [(] 函数名 [,...][)]

```
#pragma inline_asm func

static int func() {
  /* 汇编指令 */
}
```

声明汇编用函数func
在func函数中写入汇编源程序

具体请参照编译器用户手册进行修改。

汇编语言

汇编语言

- 宏运算符的差异

运算类型	CA78K0R	CC-RL	备注
连接符	&	?	

- 运算符的差异

运算类型	CA78K0R	CC-RL	备注
算术运算	+ sign, - sign, +, -, *, /, MOD	+ sign, - sign, +, -, *, /, %	
位逻辑运算	NOT, AND, OR, XOR	~, &, |, ^	
移位运算	SHR (logic), SHL	<<, >>	
段运算	—	STARTOF, SIZEOF	
分离运算	HIGH, LOW, HIGHW, LOWW, MIRHW, MIRLW	HIGH, LOW, HIGHW, LOWW, MIRHW, MIRLW, SMRLW	
比较运算	=(EQ), <>(NE), >(GT), >=(GE), <(LT), <=(LE) 当结果为“真”时，值为“0FFH”。	==, !=, >, >=, <, <= 当结果为“真”时，值为“1”。	
逻辑运算	—	&&, ||	

具体请参照编译器用户手册进行修改。

汇编语言

- 伪指令的差异

指令类型	CA78K0R	CC-RL	备注
段定义伪指令	BSEG —	— .SECTION	
内存初始化或区域配置伪指令	—	.DB8	
	—	.ALIGN	
宏伪指令	MACRO	.MACRO	
	LOCAL	.LOCAL	
	REPT	.REPT	
	IRP	.IRP	
	EXITM	.EXITM	
	—	.EXITMA	
	ENDM	.ENDM	

具体请参照编译器用户手册进行修改。

汇编语言

- 控制指令的差异

指令类型	CA78K0R	CC-RL	备注
include控制指令	—	$BINCLUDE	
条件汇编控制指令	—	$IFNDEF	
	—	$IFN	
	—	$ELSEIFN	

具体请参照编译器用户手册进行修改。

函数调用接口

函数调用接口

- 普通函数调用接口的差异

运算类型	CA78K0R	CC-RL	备注
用于存储返回值的寄存器	CY BC DE	 A AX BC DE	寄存器分配的顺序和组合是不同的，详细信息请参照编译器用户手册。
用于存储参数的寄存器	 AX BC	A, X, C, B, E, D AX BC DE	同上
存储自动变量的位置	Stack saddr area（指定-qr选项时）	Stack	同上

具体请参照编译器用户手册进行修改。

过渡支援功能

过渡支援功能

- CC-RL的过渡支援功能

  CC-RL提供了过渡支持功能。

  通过指定下列选项，使得过渡支持功能有效。

  - 编译器的过渡支援功能：-convert_cc选项
  - 汇编器的过渡支援功能：-convert_asm选项

  （例）-convert_cc=ca78k0r

  -convert_asm

另外，通过指定以下选项，不需要在每一个源文件中使用#include语句包含iodefine.h文件（IDE生成）。 iodefine.h文中定义了SFR和中断请求名。

- 编译预处理控制功能：-preinclude选项

  （例）-preinclude=iodefine.h

过渡支援功能

- 编译器过渡支援功能中的#pragma指令

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

项目	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
C-source级的SFR名称	#pragma sfr	不能转换成CC-RL功能。 请使用下面的方法： #include “iodefine.h” 或 -preinclude=iodefine.h	删除#pragma指令。 请使用iodefine.h（IDE生成）中的定义访问SFR。
中断函数的声明	#pragma vect #pragma interrupt	— #pragma interrupt	格式不同。 请参照手册。

允许中断功能
DI( )
EI( )
	#pragma DI
	#pragma EI
		__DI
		__EI
			删除#pragma 指令，变更为以下函数名：
			__DI( );
			__EI( );

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的#pragma指令

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

项目	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法

允许CPU控制指令
HALT( )
STOP( )
BRK( )
NOP( )
	#pragma HALT
	#pragma STOP
	#pragma BRK
	#pragma NOP
		__halt
		__stop
		__brk
		__nop
			删除#pragma指令，变更为以下函数名：
			__halt( );
			__stop( );
			__brk( );
			__nop( );

变更段名	#pragma section	#pragma section	格式不同。 请参照手册。
变更模块名	#pragma name	—	删除#pragma指令，使用链接器的-rename选项。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的#pragma指令

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

项目	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法

允许旋转功能
rorb( )
rolb( )
rorw( )
rolw( )
	#pragma rot	__rorb
		__rolb
		__rorw
		__rolw
			删除#pragma指令，变更为以下函数名：
			__rorb( );
			__rolb( );
			__rorw( );
			__rolw( );

允许乘法函数
mulu( )
muluw( )
mulsw( )
	#pragma mul	__mulu
		__mului
		__mulsi
			删除#pragma指令，变更为以下函数名：
			__mulu( );
			__mului( );
			__mulsi( );

允许除法函数
divuw( )
moduw( )
	#pragma div	__divui
		__remui
			删除#pragma指令，变更为以下函数名：
			__divui( );
			__remui( );

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的#pragma指令

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

项目	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法

允许乘法累加函数
macuw( )
macsw( )
	#pragma mac	__macui
		__macsi
			删除#pragma指令，变更为以下函数名：
			__macui( );
			__macsi( );

RTOS函数的声明	#pragma rtos_interrupt	#pragma rtos_interrupt	格式不同。 请参照手册。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 中断函数声明的差异
  - CA78K0R

格式：#pragma vect (or interrupt) [中断请求名] [函数名] [堆栈切换设置]

```
#pragma interrupt INTP0 inter rb1

void inter ( void ) {
/* INTP0引脚输入的中断处理 */
}
```

  - CC-RL

格式：#pragma interrupt [(] 中断处理程序请求名 [(中断规格 [,...])][)]

```
#include “iodefine.h”

#pragma interrupt inter (vect=INTP0, bank=RB1)

__near void inter ( void ) {
/* INTP0引脚输入的中断处理 */
}
```

通过指定-preinclude= iodefine.h选项，可以不在源文件中写入。

包含iodefine.h时，可以写中断请求名。

具体请参照编译器用户手册进行修改。

RENESAS 瑞萨

过渡支援功能

- 变更段名的差异
  - CA78K0R

    格式：#pragma section [编译器输出段名] [新段名] [AT起始地址]

    变更编译器输出段名

```
#pragma section @@DATA DD1 AT 2400H
```

变更段名@@DATA为DD1，并指定起始地址为2400H

  - CC-RL

    格式：#pragma section [段类型] [新段名]

    变更与段类型text、const、data或bss相对应的段名
    - near段：新段名 + “_n”
    - far段：新段名 + “_f”
    - saddr段：新段名 + “_s”

```
#pragma section bss DD1
int __far fdata;
```

bss的段名改为 DD1_f

另外，可以通过使用链接器的-start选项指定段的开始地址。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的宏

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

CA78K0R中的宏名称	CC-RL中相应的宏名称	值
__K0R__	__RL78__	十进制常量1
__K0R_SMALL__	__RL78_SMALL__	
__K0R_MEDIUM__	__RL78_MEDIUM__	
__CHAR_UNSIGNED__	__UCHAR	
__RL78_1__	__RL78_S1__	
__RL78_2__	__RL78_S2__	
__RL78_3__	__RL78_S3__	
__CA78K0R__	None	

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的关键字

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

功能	关键字	CC-RL中相应的关键字	不使用过渡支援功能时的对应方法
将变量配置在saddr区域	__sreg sreg	__saddr #pragma saddr	将__sreg变更为__saddr。
指定绝对地址	__directmap	#pragma address	不能由__directmap指定绝对地址。请使用#pragma address。另外，变量的地址不能相互覆盖。
硬件中断函数的声明	__interrupt	#pragma interrupt	使用#pragma interrupt变更声明。
软件中断函数的声明	__interrupt_brk	#pragma interrupt_brk	使用#pragma interrupt_brk变更声明。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的关键字

  当指定-convert_cc=ca78k0r选项时，自动适用于CC-RL规格。

功能	关键字	CC-RL中相应的关键字	不使用过渡支援功能时的对应方法
RTOS函数的声明	__rtos_interrupt	#pragma rtos_interrupt	不需要__rtos_interrupt。从RTOS中断处理函数声明中删除“__rtos_interrupt”。
callt函数的声明	__callt callt	__callt #pragma callt	将callt变更为__callt。
配置到saddr区域的位变量的声明	__boolean boolean bit	—	将__boolean变更为char（指定-ansi选项时）。 将__boolean、boolean、bit变更为_Bool（不指定-ansi 选项时）

- 由于在CC-RL中没有位变量，所以使用过渡支持功能（ -convert_cc=ca78k0r ）时，该位变量作为1个字节的数据进行处理。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的伪指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
段定义伪指令	CSEG	.CSEG	将CSEG变更为.CSEG。 再配置属性的记述形式不同。 再配置属性为UNITP时，将CSEG变更为.CSEG TEXTF、.ALIGN 2。
	DSEG	.DSEG	将DSEG变更为.DSEG。 再配置属性的记述形式不同。
	BSEG	.BSEG	将BSEG变更为.BSEG。
	ORG	.ORG	将ORG变更为.ORG。
符号定义伪指令	EQU	.EQU	将EQU变更为.EQU。
	SET	.SET	将SET变更为.SET。
分支指令、自动选择伪指令	BR	BR !!addr20	将BR变更为BR !!addr20。
	CALL	CALL !!addr20	将CALL 变更为CALL !!addr20。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的伪指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
内存初始化、区域保留伪指令	DB	.DB	将DB变更为.DB。 操作数“(size)”的解释是不同的。
	DW	.DB2	将DW变更为.DB2。 操作数“(size)”的解释是不同的。
	DG	.DB4	将DG变更为.DB4。 操作数“(size)”的解释是不同的。
	DS	.DS	将DS变更为.DS
	DBIT	.DBIT	将DBIT变更为.DBIT。
链接伪指令	PUBLIC	.PUBLIC	将PUBLIC变更为.PUBLIC。
	EXTRN	.EXTERN	将EXTERN变更为.EXTERN。
	EXTBIT	.EXTBIT	降EXTBIT变更为.EXTBIT。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的伪指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
目标模块名的声明伪指令	NAME	按注释处理	删除伪指令NAME。使用链接器的-rename选项。
汇编结束伪指令	END	按注释处理	删除伪指令END

具体请参照编译器用户手册进行修改。

过渡支援功能

- 符号定义伪指令EQU（未使用过渡支援功能时）
  在符号定义伪指令EQU的操作数中，不能记述一个重新定位的标签。
  用重新定位的标签替换掉EQU左侧名字的基准点，并禁用EQU。
  - CA78K0R

```
DMAINP DSEG SADDRP
RABUF1: DS 8
RABUF2: DS 8
OFFSET EQU RABUF2 - RABUF1
FPREAD EQU RABUF1.4
        CSEG
        ADD A,#OFFSET
        CLR1 FPREAD
```

删除

删除

置换

  - CC-RL

置换

```
DMAINP DSEG SADDRP
RABUF1: DS 8
RABUF2: DS 8
        CSEG
        ADD A, #RABUF2 - RABUF1
        CLR1 RABUF1.4
```

具体请参照编译器用户手册进行修改。

过渡支援功能

- 内存初始化、区域保留伪指令（未使用过渡支援功能时）
  内存初始化、区域保留伪指令只能有一个操作数。
  如果要写多个操作数，请分割为多个伪指令。
  - CA78K0R

```
MSGDATA CSEG AT 80H
TMSGOK:
        DB 'OK'
        DB 0DH,0AH
        END
```

修改

  - CC-RL

```
MSGDATA CSEG AT 80H
TMSGOK:
        .DB 'OK'
        .DB 0DH
        .DB 0AH
```

具体请参照编译器用户手册进行修改。

过渡支援功能

- 内存初始化、区域保留伪指令的大小（未使用过渡支援功能时）
  内存初始化和区域保留伪指令中，操作数“(size)”的解释不同。
  - CA78K0R

    格式：DW (size)　指定size的单位为WORD。

```
CSEG
DW (3)
END
```

修改　3 words x 2 = 6 bytes

  - CC-RL

    格式：DW (size)　指定size的单位为BYTE。

```
.CSEG
.DS 6
```

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的控制指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
汇编目标类型指定控制指令	$PROCESSOR ($PC)	按注释处理	指定-dev选项。
include控制指令	$INCLUDE ($IC)	$INCLUDE	变更为$INCLUDE。
RAM区域分配指定控制指令	$RAM_ALLOCATE	按注释处理	使用.CSEG伪指令来配置对象段。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的控制指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
条件汇编控制指令	$IF	$IF	使用-define选项或者.SET。
	$_IF	$IF	变更为$IF。
	$ELSEIF	$ELSEIF	使用-define选项或者.SET。
	$_ELSEIF	$ELSEIF	变更为$ELSEIF。
	$ELSE	$ELSE	
	$ENDIF	$ENDIF	
	$SET, $RESET	按注释处理	删除$SET、$RESET 伪指令。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的控制指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
调试信息输出控制指令	$DEBUG ($DG), $NODEBUG ($NODG)	按注释处理	指定-debug选项。
	$DEBUGA, $NODEBUGA	按注释处理	
交叉引用列表输出指定控制指令	$XREF ($XR), $NOXREF ($NOXR)	按注释处理	删除$XREF ($XR)、$NOXREF ($NOXR)控制指令。
	$SYMLIST, $NOSYMLIST	按注释处理	删除$SYMLIST、$NOSYMLIST控制指令。
汇编列表控制指令	$EJECT ($EJ)	按注释处理	删除$EJECT ($EJ) 控制指令。
	$LIST ($LI), $NOLIST ($NOLI)	按注释处理	删除$LIST ($LI)、$NOLIST ($NOLI)控制指令。
	$GEN, $NOGEN	按注释处理	删除$GEN、$NOGEN控制指令。
	$COND, $NOCOND	按注释处理	删除$COND、$NOCOND控制指令。
	$TITLE ($TI)	按注释处理	删除$TITLE ($TI)控制指令。
	$SUBTITLE ($ST)	按注释处理	删除$SUBTITLE ($ST)控制指令。

具体请参照编译器用户手册进行修改。

过渡支援功能

- 编译器过渡支援功能中的控制指令

  当指定-convert_asm时，自动适用于CC-RL规格。

指令类型	CA78K0R	CC-RL	不使用过渡支援功能时的对应方法
汇编列表控制指令	$FORMFEED, $NOFORMFEED	按注释处理	删除$FORMFEED、$NOFORMFEED控制指令。
	$WIDTH	按注释处理	删除$WIDTH控制指令。
	$LENGTH	按注释处理	删除$LENGTH控制指令。
	$TAB	按注释处理	删除$TAB控制指令。
汉字编码控制指令	$KANJICODE	按注释处理	指定-character_set选项。
其它控制指令	$TOL_INF, $DGS, $DGL	按注释处理	删除$TOL_INF、$DGS、$DGL控制指令。

具体请参照编译器用户手册进行修改。

FAQ

FAQ

- 本部分内容中，记述了关于CA78K0R转至CC-RL时，编译器、链接器相关错误信息的FAQ。
- 关于FAQ，请从瑞萨电子网页上取得。
  - http://cn.renesas.com/products/tools/coding_tools/c_compilers_assemblers/rl78_compiler/index.jsp
    -> FAQ

具体请参照编译器用户手册进行修改。

FAQ

FAQ No.	Q	A

1011661	I get the error message below when I try to access the SFRs. How do I get around this?
	
	E0520020: Identifier "character string" is undefined.
		Include the iodefine.h file that is generated when you use an IDE to create a project. This gives you reserved words to use in access to SFRs. SFRs that are addressable from the compiler in byte or word units and those SFRs having bits which are addressable in bit units (only those bit names corresponding to bit numbers enclosed in squares in the user's manual for the MCU) can be accessed by writing their names.
		（Example）
		#include"iodefine.h"
		ADM2 = 0x12; /* Reserved word for the byte-unit SFR */
		ADTYP = 1; /* Reserved word for the bit-unit SFR */
		You can designate inclusion of the file by a directive as shown above or by designating it with the compiler’s -preinclude option.
		（Example）
		-preinclude=iodefine.h

FAQ

FAQ No.	Q	A

1011662	I get the error message below when I try to access the bits of SFRs. How do I get around this?
	
	E0520020: Identifier "character string" is undefined.
	E0520065: Expected a ";".
		Include the iodefine.h file that is generated when you use an IDE to create a project. This gives you reserved words to use in access to SFRs. SFRs that are addressable from the compiler in byte or word units and those SFRs having bits which are addressable in bit units (only those bit names corresponding to bit numbers enclosed in squares in the user's manual for the MCU) can be accessed by writing their names. In the case of bits for which the numbers are not enclosed in squares, use the reserved word with _bit appended for the name of the byte- or word-unit SFR defined in iodefine.h.
		(Example)
		#include"iodefine.h"
		P0_bit.no2 = 1; /* There is no reserved word for the bit-unit SFR */
		In CC-RL, owing to the porting assistance facilities, you can use the -convert_cc option of the compiler to write it in the style of CA78K0R without using the reserved words for bytes and words with the _bit name attached.
		(Example)
		#include"iodefine.h"
		P0.2 = 1; /* There is no reserved word for the bit-unit SFR */
		You can designate inclusion of the file by a directive as shown above or by designating it with the compiler's -preinclude option.
		（Example）
		-preinclude=iodefine.h

FAQ

FAQ No.	Q	A
1011663	I get the error message below when I use #pragma to define an interrupt function. How do I get around this? E0523005: Invalid pragma declaration	Write the interrupt function in the form of #pragma interrupt [()]interrupt handler name[(interrupt specification [,...])][()]. A file iodefine.h is generated when you create a project in an IDE. Include iodefine.h in the C source file which uses interrupt request names, since it has definitions for the names of interrupt requests. In CC-RL, owing to the porting assistance facilities, you can use the -convert_cc option of the compiler to write it in the style of CA78K0R.
1011664	I get the error message below when I try to define an interrupt function. How do I eliminate this error? E0520065: Expected a ";".	Designate the interrupt function with #pragma interrupt. CC-RL does not have a __interrupt interrupt qualifier. In CC-RL, owing to the porting assistance facilities, you can use the -convert_cc option of the compiler to write it in the style of CA78K0R.
1011665	I get the error message below when I try to define an interrupt function. How do I eliminate this error? E0520014: Extra text after expected end of preprocessing directive.	A file iodefine.h is generated when you create a project in an IDE. Include iodefine.h before issuing the #pragma interrupt, since it has definitions for the names of interrupt requests. You can designate inclusion of the file by a directive as shown above or by designating it with the compiler's -preinclude option. （Example） -preinclude=iodefine.h

FAQ

FAQ No.	Q	A
1011666	get the error message below when I designate a library file. How do I resolve this? E0562201: Illegal library file : "xxxx.lib"	Check that you have not designated a library file for CA78K0R. You cannot use the CC-RL compiler to link a library which was generated with CA78K0R because they are in different object formats. Please recreate the library file for CC-RL.
1011667	I get the error message below when I designate an object file as an input file. How do I resolve this? E0562200: Illegal object file : "xxxx.rel"	Check that you have not designated an object file for CA78K0R. You cannot use the CC-RL compiler to link a object which was generated with CA78K0R because they are in different object formats. Please recreate the object file for CC-RL.
1011668	I get the warning message below when I try to compile files. Why does this happen? W0511179: The evaluation version is valid for the remaining number days.	The message appears because you have not registered your license key for CC-RL. You have a 60-day trial period from first building, and your usage is not restricted over that period as it is a free evaluation copy. The message indicates how much of that period remains. After the trial period, the linkage size is restricted to 64 K or fewer bytes, and the MISRA-C checking function becomes unusable.
1011669	I get the error message below when I attempt access to the PSW. I have included iodefine.h. Is there any way around this? E0520020: Identifier " PSW " is undefined.	There is no PSW definition in iodefine.h file, since you cannot access the PSW directly. Use the following intrinsic functions for PSW operations. - __get_psw This returns the contents of the PSW. - __set_psw This sets a value for the PSW.

修订记录

版本	内容	页
Rev.1.00	初版发行	–

www.cn.renesas.com

RENESAS 瑞萨